

Оптимальне керування соціально-економічними системами

у середовищі MatLab®

В.М. Кирилич

О.В. Терещук

В.М. Флюд

Навчальний посібник

Львів
ЛНУ імені Івана Франка
2021

УДК 517;519.6;338

Рецензенти:

д-р. фіз.-мат. наук, проф. Капустян О.В. (Київський національний університет імені Т.Шевченка);

д-р. фіз.-мат. наук, проф. Король І.І. (Ужгородський національний університет);

dr. hab., prof. Sushch V. (Koszalin University of Technology).

Рекомендовано Вченою Радою

Львівського національного університету імені Івана Франка.

(Протокол № 7/3 від 31 березня 2021 року)

Кирилич В.М., Терещук О.В., Флюд В.М. Оптимальне керування соціально-економічними системами у середовищі MatLab® :
Навч. посібник. – Львів: ЛНУ імені Івана Франка. 2021. – 412 с.

Подано послідовний виклад комп'ютерної системи MatLab®, основні функції та процедури, графічні можливості, керуючі конструкції мови програмування та їхнє застосування до чисельного знаходження наближеного розв'язку задач оптимального керування, що виникають у багатьох соціально-економічних процесах (економіці, природознавстві, теорії популяцій, екології, природоохоронній діяльності тощо).

Розглянуто систематичний виклад середовища MatLab® і його застосування у задачах оптимального керування, математичні моделі яких описуються за допомогою задач для звичайних і часткових диференціальних рівнянь. У посібнику наведено багато прикладів і задач для самостійного опрацювання. Крім того, подано задачі, які потребують додаткового опрацювання і можуть слугувати основою студентських курсових робіт.

Для студентів математичних, економічних та інших прикладних спеціальностей.

©Кирилич В.М., Терещук О.В., Флюд В.М., 2021

ISBN 978-617-10-0630-0

Зміст

| | |
|--|-----------|
| Вступ | xiii |
| 1 Ознайомлення з MatLab[®] | 1 |
| 1.1. Призначення системи MatLab [®] | 1 |
| 1.2. Початок роботи в системі MatLab [®] | 2 |
| 1.2.1. Ознайомлення з роботою в системі | 2 |
| 1.2.2. Операції редагування стрічки | 3 |
| 1.2.3. Команди управління вікном | 3 |
| 1.3. Оператори та внутрішні функції MatLab [®] | 4 |
| 1.3.1. Синтаксис операторів MatLab [®] | 7 |
| 1.3.2. Оператори керування виконання програми | 10 |
| 2 Спеціальні обчислення | 22 |
| 2.1. Основні операції математичного аналізу | 22 |
| 2.1.1. Табулювання функції | 22 |
| 2.1.2. Обчислення суми елементів масиву чисел | 24 |
| 2.1.3. Обчислення добутку елементів масиву | 25 |
| 2.1.4. Обчислення похідних | 26 |
| 2.1.5. Обчислення границь | 28 |
| 2.1.6. Розвинення функції у степеневий ряд | 30 |
| 2.1.7. Нулі функцій | 32 |
| 2.1.8. Визначення екстремумів функції | 34 |
| 2.1.9. Символьне інтегрування | 41 |
| 2.2. Створення спеціальних додатків для розв'язування типових завдань | 44 |
| 2.2.1. Створення додатку | 44 |
| 3 Графіка у MatLab[®] | 51 |
| 3.1. Двовимірна графіка | 51 |
| 3.1.1. Функція plot(x,y) | 51 |

| | | |
|----------|--|------------|
| 3.1.2. | Функція <code>plot(x,y,s)</code> | 52 |
| 3.1.3. | Функція <code>plot(x1,y1,s1,x2,y2,s2,...,xn,yn,sn)</code> | 53 |
| 3.1.4. | Побудова графіків функцій, заданих параметричним поданням | 55 |
| 3.1.5. | Побудова графіків кусково-неперервних функцій | 56 |
| 3.1.6. | Побудова графіків у логарифмічному масштабі | 58 |
| 3.1.7. | Побудова графіків у полярній системі координат | 60 |
| 3.1.8. | Побудова гістограм | 60 |
| 3.2. | Тривимірні графіки | 61 |
| 3.2.1. | Тривимірні графіки функцій | 61 |
| 3.2.2. | Контурні графіки | 65 |
| 3.2.3. | Оформлення графіка | 67 |
| 3.2.4. | Побудова параметрично заданих поверхонь і просторових кривих | 69 |
| 3.2.5. | Побудова освітленої поверхні | 72 |
| 3.2.6. | Анімовані графіки | 74 |
| 3.3. | Робота з декількома графіками | 74 |
| 3.3.1. | Вивід графіків в окремі вікна | 75 |
| 3.3.2. | Побудова декількох графіків в одній координатній системі | 77 |
| 3.3.3. | Декілька графіків в одному графічному вікні | 79 |
| 3.4. | Завдання для самостійного опрацювання | 81 |
| 4 | m-файли | 85 |
| 4.1. | Побудова файлу-сценарію | 85 |
| 4.2. | Робота в редакторі m-файлів | 86 |
| 4.3. | Скрипти та функції | 88 |
| 4.4. | Увід числових і символьних даних | 89 |
| 4.5. | Виведення результатів обчислень | 90 |
| 4.6. | Типи функцій | 91 |
| 4.7. | Змінні та параметри функцій | 94 |
| 4.7.1. | Глобальні змінні | 94 |
| 4.7.2. | Параметри функцій | 96 |
| 4.7.3. | Функції масиву | 100 |
| 5 | Алгебра векторів і матриць у MatLab® | 103 |
| 5.1. | Утворення векторів і матриць | 103 |
| 5.2. | Перетворення матриць | 105 |
| 5.2.1. | Звертання до елементів матриці та заміна їхніх значень | 105 |
| 5.2.2. | Зміна розмірності вектора та матриці | 106 |

| | | |
|----------|--|------------|
| 5.3. | Математичні операції з векторами та матрицями | 111 |
| 5.3.1. | Визначник матриці | 111 |
| 5.3.2. | Ранг матриці | 112 |
| 5.3.3. | Транспонування матриці | 113 |
| 5.3.4. | Слід матриці | 113 |
| 5.3.5. | Одинична матриця | 114 |
| 5.3.6. | Матриця з одиничними елементами | 115 |
| 5.3.7. | Матриця з нульовими елементами | 116 |
| 5.3.8. | Утворення матриці з наперед заданою діагоналлю . | 116 |
| 5.3.9. | Перестановка елементів матриці | 118 |
| 5.3.10. | Обернена матриця | 119 |
| 5.3.11. | Вектор рівновіддалених точок | 120 |
| 5.3.12. | Виділення трикутних частин матриці | 121 |
| 5.4. | Математичні операції над векторами та матрицями | 122 |
| 5.4.1. | Арифметичні операції над векторами та матрицями | 122 |
| 5.4.2. | Скалярний добуток | 126 |
| 5.4.3. | Обчислення добутків елементів масиву | 127 |
| 5.4.4. | Підсумовування елементів масиву | 129 |
| 5.5. | Завдання для самостійного опрацювання | 130 |
| 6 | Розв'язування задач оптимізації у MatLab[®] | 132 |
| 6.1. | Задачі лінійного програмування | 132 |
| 6.1.1. | Математичні моделі, які приводять до задач лінійного програмування | 135 |
| 6.1.2. | Завдання для самостійного опрацювання | 140 |
| 6.2. | Задачі нелінійного програмування | 147 |
| 6.2.1. | Завдання для самостійного опрацювання | 150 |
| 6.3. | Метод Runge-Kutta розв'язування початкової задачі для звичайного диференціального рівняння першого порядку . | 152 |
| 6.3.1. | Загальне формулювання методів | 152 |
| 6.3.2. | Метод Runge-Kutta. Загальна формула чисельного наближення розв'язку | 154 |
| 6.3.3. | Оцінка похибки чисельного розв'язку | 157 |
| 6.4. | Приклад застосування методу Runge-Kutta | 157 |
| 6.4.1. | Завдання для самостійного опрацювання | 160 |
| 6.5. | Інтерполяція | 162 |
| 6.5.1. | Формулювання задачі інтерполяції | 162 |
| 6.5.2. | Інтерполяція за допомогою многочленів | 163 |
| 6.5.3. | Інтерполяційна формула Lagrange'a | 164 |
| 6.6. | Апроксимація | 166 |
| 6.6.1. | Задача апроксимації | 166 |

| | | |
|----------|--|------------|
| 6.6.2. | Середньоквадратична апроксимація | 167 |
| 6.6.3. | Апроксимація многочленами | 169 |
| 7 | Розв’язування початкових задач для звичайних диференціальних рівнянь у MatLab® | 170 |
| 7.1. | Наближене знаходження розв’язку початкових задач для звичайних диференціальних рівнянь | 170 |
| 7.2. | Завдання для самостійного опрацювання | 177 |
| 8 | Основні моделі та їх застосування | 179 |
| 8.1. | Модель популяції молі spruce budworm | 179 |
| 8.2. | Системи звичайних диференціальних рівнянь. Моделі виживання. | 185 |
| 8.2.1. | Модель хижак-жертва | 185 |
| 8.2.2. | Свинець в організмі | 196 |
| 8.2.3. | Модель автокаталітичної реакції | 201 |
| 8.2.4. | Модель навантаженої балки | 203 |
| 8.2.5. | 3D графіка | 205 |
| 8.2.6. | Завдання для самостійного опрацювання | 210 |
| 8.3. | Оптимальне керування диференціальними системами. Умови оптимальності | 212 |
| 8.3.1. | Формулювання задачі. Принцип Понтрягіна | 212 |
| 8.3.2. | Задача максимізації загального споживання | 219 |
| 8.3.3. | Максимізація генеральної сукупності у системі хижак-жертва | 227 |
| 8.3.4. | Модель інсулінової терапії | 240 |
| 8.3.5. | Приклади | 252 |
| 8.3.6. | Завдання для самостійного опрацювання | 256 |
| 8.4. | Оптимальне керування диференціальними системами. Градієнтний метод | 260 |
| 8.4.1. | Метод проєкції градієнта | 260 |
| 8.4.2. | Метод найшвидшого спуску. Приклад | 267 |
| 8.4.3. | Керування запасами | 287 |
| 8.4.4. | Задача оптимального розмноження | 303 |
| 8.4.5. | Завдання для самостійного опрацювання | 317 |
| 8.5. | Оптимальне розмноження структурованою за віком популяції | 320 |
| 8.5.1. | Динаміка з лінійною віковою структурою | 320 |
| 8.5.2. | Задача оптимального розмноження | 337 |
| 8.5.3. | Логістична модель із періодичним показником природної зміни популяції | 350 |

| | |
|--|------------|
| 8.5.4. Завдання для самостійного опрацювання | 362 |
| 9 Оптимальне керування дифузійними моделями | 365 |
| 9.1. Дифузія у математичних моделях | 366 |
| 9.2. Модель Fisher'a оптимальної народжуваності | 369 |
| 9.3. Приклад: керування системою реакції-дифузії | 381 |
| 9.4. Завдання для самостійного опрацювання | 385 |
| Бібліографія | 389 |
| Предметний покажчик | 396 |

Перелік ілюстрацій

| | | |
|------|---|----|
| 1.1 | Графік функції $f(x) = \sin x - 0.5x$ | 18 |
| 2.1 | Графік до прикладу відшукування нулів функції | 34 |
| 2.2 | Нулі многочлена (до прикладу 2.1.11) | 35 |
| 2.3 | Графік функції $y = xe^{-x}$ | 36 |
| 2.4 | Графік функції $f(x) = \sin^3 x + \cos^3 x$ | 37 |
| 2.5 | Графік функції $f(x) = 2^{-x} - 3^{-x} - 10x$ та точка максимуму | 38 |
| 2.6 | Графік поверхні $f(x, y) = e^{-(x^2+xy+y^2)}(5x + 7y - 25)$ та її лінії рівня | 39 |
| 2.7 | Графік функції до прикладу 2.1.16 | 40 |
| 2.8 | Застосунок у MatLab [®] побудови графіка, точки мінімуму і нуля функції | 50 |
| 3.1 | Графік функції до прикладу 3.1.1 | 52 |
| 3.2 | Графік до прикладу 3.1.2 | 54 |
| 3.3 | Графік до прикладу 3.1.3 | 56 |
| 3.4 | Графік функції $f(x) = 2^{-3\sin x} + 3x^2 - 4$ | 57 |
| 3.5 | Графік циклоїди до прикладу 3.1.5 | 57 |
| 3.6 | Графік до прикладу 3.1.6 | 58 |
| 3.7 | Графік до прикладу 3.1.7 | 59 |
| 3.8 | Графік трипелюсткової троянди | 60 |
| 3.9 | Побудова гістограм | 61 |
| 3.10 | Графіки поверхонь, побудовані за допомогою функцій <code>mesh</code> і <code>surf</code> | 63 |
| 3.11 | Графіки поверхні з використанням різних опцій | 64 |
| 3.12 | Графіки поверхні з використанням <code>colorbar</code> , <code>surfc</code> , <code>contour3</code> . | 65 |
| 3.13 | Графік ліній рівня | 66 |
| 3.14 | Графік ліній рівня: <code>clabel</code> , <code>contourf</code> | 67 |
| 3.15 | Графік поверхні з використанням палітри <code>gray</code> | 68 |
| 3.16 | Використання TeX-символів у описі графіка | 70 |
| 3.17 | Графік параметрично заданої просторової кривої | 71 |

| | | |
|------|--|-----|
| 3.18 | Графік параметричної поверхні | 72 |
| 3.19 | Освітлені графіки | 73 |
| 3.20 | Графіки у різних вікнах (до прикладу 3.3.1) | 77 |
| 3.21 | Два графіки в одній координатній системі: <code>hidden on</code> та <code>hidden off</code> | 78 |
| 3.22 | Кілька графіків в одному вікні | 80 |
| 3.23 | Підграфіки (до прикладу 3.3.2) | 81 |
| 4.1 | Графік функції, побудований за допомогою <code>m</code> -файлу | 86 |
| 4.2 | Графіки функцій $y = e^{-x}$ та $y = \sin x$ | 88 |
| 4.3 | Графік функції, побудований за допомогою <code>m</code> -файлу | 93 |
| 4.4 | До прикладу 4.7.1 | 95 |
| 4.5 | Графіки <code>polyline</code> у випадку 7-ми та 5-ти точок | 98 |
| 4.6 | До прикладу 4.7.7 | 102 |
| 6.1 | Графіки чисельних розв'язків одержаних методом Runge-Kutta та <code>ode45</code> | 159 |
| 6.2 | Інтерполяція | 162 |
| 6.3 | Середньоквадратична апроксимація | 168 |
| 7.1 | Графік наближеного розв'язку задачі (7.1.2) | 172 |
| 7.2 | Графіки наближених розв'язків задачі (7.1.2), обчислені за допомогою <code>ode23</code> і <code>ode45</code> | 175 |
| 7.3 | Графіки точного та наближеного розв'язку задачі (7.1.4), обчисленого за допомогою <code>ode23</code> | 176 |
| 8.1 | Графік $p(N)$ на проміжку $[0, 20]$ для $B = 1.5$, $A = 2$ | 180 |
| 8.2 | Графіки $N(t)$ за різних вхідних даних: а) $r = 0.3$, $K = 5$, $A = 2$, $b = 1.5$, $T = 20$, $N(0) = 30$; б) $r = 1$, $K = 10$, $A = 3$, $b = 2$, $T = 10$, $N(0) = 50$ | 181 |
| 8.3 | Графік приплоду комах для $r = 0.3$, $K = 5$, $u = 0.5$, $T = 20$, $N(0) = 30$ | 183 |
| 8.4 | Графіки приплоду комах для $r = 0.3$, $K = 5$, $T = 20$, $N(0) = 30$ та значень $u = 0.3, 1, 3$ | 185 |
| 8.5 | Графіки чисельних розв'язків моделі "хижак-жертва" за різних початкових даних | 189 |
| 8.6 | Графік розв'язку моделі "хижак-жертва" у фазовій площині (y_1, y_2) | 189 |
| 8.7 | Графік розв'язку моделі "хижак-жертва" за початкових даних $y_1(0) = 0.42$, $y_2(0) = 0.3$ | 190 |
| 8.8 | Графіки станів збудження та відновлення | 193 |

| | | |
|------|---|-----|
| 8.9 | Графіки станів збудження та відновлення у разі дії стимуляції | 194 |
| 8.10 | Графік збудження-відновлення у разі дії стимуляції у фазовій площині | 196 |
| 8.11 | Графіки y_1, y_2, y_3 | 198 |
| 8.12 | Графіки y_1, y_2, y_3 для значення $I(t)$ за формулою (8.2.9) | 200 |
| 8.13 | Графіки розв'язків автокаталітичної моделі: a – графіки $y_1/200, y_2, y_3, y_4/200$; b – графік у фазовій площині y_2Oy_3 | 203 |
| 8.14 | Графіки розв'язків при різних навантаженнях балки: a – випадок $C = 1$; b – випадок $C = 1.75$; c – випадок $C = 2.6$ | 206 |
| 8.15 | Графік гвинтової лінії | 207 |
| 8.16 | Осциляція у бокалі | 208 |
| 8.17 | 3D графіки: a – surf-графік; b – mesh-графік | 210 |
| 8.18 | Залежність функції цілі від точок перемикавання для значення параметрів $r_1 = 0.07, \mu_1 = 1, r_2 = 0.6, \mu_2 = 2, L = 50, h = 0.1, h1 = 1, x(0) = 0.04, y(0) = 0.02, lw = 5$: a – поведінка цільової функції в залежності від керування; b – інша залежність поведінки цільової функції в залежності від точки перемикавання | 239 |
| 8.19 | Результати обчислення за допомогою програми <code>ppr2.m</code> для тих самих значень параметрів моделі як у випадку <code>ppr1.m</code> | 241 |
| 8.20 | Інсулін-глюкозна залежність: a – концентрація інсуліну; b – концентрація глюкози | 243 |
| 8.21 | Залежність вмісту глюкози від ін'єкцій інсуліну | 250 |
| 8.22 | Концентрація глюкози у випадку періодичних ін'єкцій | 250 |
| 8.23 | Оптимальне керування у випадку $x_0 = 0$ | 284 |
| 8.24 | Оптимальне керування у випадку $x_0 = 2$ | 285 |
| 8.25 | Оптимальне керування для даних із Прикладу 8.4.2 | 301 |
| 8.26 | Графіки до: a – Прикладу 8.4.4; b – Прикладу 8.4.4 | 303 |
| 8.27 | Динаміка популяції програми <code>harv1</code> | 308 |
| 8.28 | Оптимальне керування програми <code>harv1</code> при різних початкових даних: a – випадок $x(0) = 1.4$; b – випадок $x(0) = 1.6$ | 309 |
| 8.29 | Керування, одержане за допомогою <code>harv2.m</code> із використанням <code>ProjHarv</code> | 315 |
| 8.30 | Керування, одержане за допомогою <code>harv35b.m</code> із використанням <code>Proj2</code> | 316 |
| 8.31 | Керування, одержане за допомогою <code>harv35a.m</code> із використанням <code>Proj2</code> | 316 |
| 8.32 | Графіки програми <code>pop1.m</code> : a – коефіцієнти фертильності (β) та смертності (μ); b – розподіл популяції | 336 |

-
-
- 8.33 Графіки програми `ohp1.m`: a – графік оптимального керування u^* ; b – графік розв’язку двоїстої задачі p ; 348
- 8.34 Графік оптимального керування при $t = 0.4$ 349

Перелік таблиць

| | | |
|-----|---|-----|
| 1.1 | Список клавіш редагування | 2 |
| 1.2 | Список операторів MatLab [®] | 5 |
| 1.3 | Синтаксис операторів MatLab [®] | 8 |
| 1.4 | Програма-сценарій обчислення значення 5! | 9 |
| 3.1 | Палітра кольорів у середовищі MatLab [®] | 69 |
| 5.1 | Таблиця матричних операторів MatLab [®] | 122 |
| 6.1 | Дані до задачі 6.1.16 | 146 |

Вступ

У цьому курсі вивчається середовище MatLab[®] для операційної системи Windows[®] та його застосування до обчислення задач і моделей математичної економіки, зокрема, оптимального керування соціально-економічними процесами, які описані диференціальними рівняннями. Оскільки будь-яка економічна система має визначену мету функціонування (критерій якості), а також свободу вибору, то математичним апаратом для таких систем є теорія оптимального керування (динамічна оптимізація).

У цьому підручнику проаналізовано деякі задачі оптимального керування динамічними моделями економіки, біології, теорії популяцій, природоохоронної діяльності тощо, тому зроблено детальний аналіз методів теорії оптимального керування, зокрема, огляд принципу максимуму Л.С. Понтрягіна для одержання необхідних умов оптимальності.

Посібник складається з вступу та дев'яти розділів, завдань для самостійної роботи, додатків і списку рекомендованої літератури. Для зручності читачів створено списки графіків і таблиць, які поміщені на початку підручника. Крім того, утворений індексний перелік важливих термінів, які використовуються у підручнику.

Перші п'ять розділів присвячені системі MatLab[®], починаючи від основ програмування у програмному середовищі, спеціальних обчислень, операторів керування, побудови графіків, побудова програм і процедур.

У шостому розділі розглянуто задачі оптимізації у MatLab[®], зокрема, задачі лінійного та нелінійного програмування, ітераційні та апроксимаційні методи, які служать основою для побудови чисельних алгоритмів знаходження наближених розв'язків досліджуваних задач.

Сьомий розділ присвячений наближеному знаходженню у MatLab[®] розв'язків початкових задач для звичайних диференціальних рівнянь.

У восьмому розділі посібника наведені основні програми MatLab[®] та їхнє застосування до задач оптимального керування різноманітними процесами. У цьому розділі також описані математичний алгоритм оптимального керування, принцип максимуму Понтрягіна та його застосу-

вання у багатьох конкретних задачах.

Дев'ятий розділ навчального посібника присвячений оптимальному керуванню дифузійними процесами. Наведено теоретичні обґрунтування існування оптимального керування, алгоритми знаходження чисельного розв'язку досліджуваних моделей.

Посібник укладено так, що у кожному розділі розглянуто конкретні приклади, подано вказівки до розв'язування самостійних завдань, сформульовано дещо складніші завдання для самостійного опрацювання, які можуть слугувати основою для написання курсових чи магістерських робіт.

Наведено велику кількість літературних джерел. Усі англійські матеріали доступні в електронних варіантах і легко використати студентам для самостійної роботи. Для розгляду конкретних тем є літературні посилання. Зазначимо, що наявність великої кількості літературних першоджерел із теорії оптимального керування соціально-економічними процесами свідчить про те, що пропрані методи і підходи з використанням програмного середовища MatLab[®] відкривають можливості для ефективного і простого розв'язування прикладних задач оптимального керування системами звичайних диференціальних рівнянь і з частинними похідними.

У посібнику використано порозділову нумерацію формул, де перший номер (прикладів, завдань тощо) означає номер розділу, другим – номер підрозділу, далі порядковий номер формули. Подібно пронумеровано рисунки та таблиці (першим є число, що означає номер розділу, другим – порядковий номер рисунка чи таблиці).

Числові результати, наведені у посібнику, проводили у системі MatLab[®] : Version 7.9.0.529(R2009b), 64-bit(win64), August 12,2009, License Number: 307463.

Навчальний посібник розрахований для студентів старших курсів бакалаврських та магістерських програм. Курс читали продовж декількох років для студентів механіко-математичного факультету, спеціалізації "Математична економіка та економетрика" Львівського національного університету імені Івана Франка. Основою курсу слугувала монографія [5]. Доповнено завданнями для самостійного опрацювання, застосування MatLab[®] для знаходження розв'язків сформульованих задач.

Автори

Розділ 1

Ознайомлення з MatLab®

У цьому розділі описано основні принципи роботи у системі MatLab®, починаючи від призначення клавіш керування до основ програмування. Очевидно, автори не ставили собі за мету детально описати систему, а сконцентрували увагу на тих можливостях системи, які будуть використані у дослідженні розглядуваних проблем. Детальніше про саму систему MatLab® та її використання для дослідження споріднених проблем читач може знайти за посиланням [75, 76, 46, 47, 74, 21, 68].

1.1. Призначення системи MatLab®

MatLab® реалізує три важливі концепції програмування:

- процедурне модульне програмування, яке спирається на створення модулів – процедур і функцій;
- об'єктно-орієнтоване програмування, особливо важливе в реалізації графічних засобів системи;
- візуально-орієнтоване програмування, основним завданням якого є створення графічного інтерфейсу користувача GUI (Graphics Users Interface).

Мова програмування MatLab® належить до класу інтерпретаторів. Це означає, що будь-яка команда системи розпізнається (інтерпретується) за її іменем (ідентифікатором) і відразу виконується в командній стрічці, що забезпечує легку перевірку частинами довільного програмного коду.

1.2. Початок роботи в системі MatLab®

1.2.1. Ознайомлення з роботою в системі

Для практичної роботи будемо використовувати програмний комплекс MatLab® 8.3.0.532 (R2014a), робота в якому досить легка та комфортна. У стандартному завантаженні (клацнувши мишкою по ярлику програми MatLab®) на екрані відкривається система меню та три вікна: робоча область, вікно історії, командне вікно. Робота з меню виконується за допомогою маніпулятора мишки та набору певних клавіатурних комбінацій ("гарячих клавіш"). У вікні робочої області відображаються імена змінних, уведених користувачем, та їх поточні значення. Вікно історії використовується для зберігання та відображення раніше виконаних команд користувача. Інтерактивний сеанс роботи в командному вікні MatLab® прийнято іменувати сесією (*session*). Сесія, по суті, є поточним документом, який відображає роботу користувача з системою MatLab®. У ній є рядки введення даних, результатів і повідомлень про помилки. За допомогою команд сесії користувач взаємодіє з середовищем пакета. Поіменовані дані поточної сесії, розташовані в робочій області пам'яті (але не саму сесію), можна записати на диск у вигляді файлів формату *.mat, використовуючи команду **save** (зберегти). Командою **load** (завантажити) можна відновити з диска дані робочої області. Фрагменти сесії можна зберегти в певному документі за допомогою команди **diary**. У роботі з MatLab® у командному режимі діє найпростіший рядковий редактор. Команди, які використовуються у редакторі, мають зазвичай відповідні комбінації клавіш для їх виклику, які наведені в табл. 1.1.

Табл. 1.1. Список клавіш редагування

| Комбінація клавіш | Призначення |
|--------------------|--|
| → або Ctrl+b | переміщує курсор вправо на один символ |
| ← або Ctrl+f | переміщує курсор вліво на один символ |
| Ctrl+ → або Ctrl+r | переміщує курсор вправо на одне слово |
| Ctrl+ ← або Ctrl+l | переміщує курсор вліво на одне слово |
| Home або Ctrl+a | переміщує курсор на початок стрічки |

| | |
|---------------------------|--|
| End або Ctrl+e | переміщує курсор на кінець стрічки |
| ↑ i ↓ або Ctrl+p i Ctrl+n | перелистування попередніх команд вгору або вниз для підстановки їх у стрічку вводу |
| Del або Ctrl+d | випирання символів справа від курсора |
| Backspace або Ctrl+h | випирання символів зліва від курсора |
| Ctrl+k | випирання до кінця стрічки |
| Esc | випирання стрічки вводу |
| Ins | вмикання/вимикання режиму вставки |
| PgUp | перегорнання сторінки сесії догори |
| PgDn | перегорнання сторінки сесії донизу |

1.2.2. Операції редагування стрічки

Працюючи з MatLab® у командному режимі діє найпростіший стрічковий редактор. Обмежимося переліком команд стрічкового редагування. Особливу увагу треба звернути на застосування клавіш ↑ та ↓. Вони використовуються для підстановки після маркера стрічки вводу >> раніше введених стрічок, наприклад для їх виправлення, дублювання чи доповнення. Ці клавіші забезпечують перегортання раніше введених стрічок знизу до верху або навпаки. Така можливість існує завдяки організації спеціального стека, який зберігає стрічки з раніше введеними командами.

1.2.3. Команди управління вікном

Корисно на самому початку праці в середовищі засвоїти команди управління вікном командного режиму:

- `clc` – очищує екран і розміщує курсор у лівому верхньому куті порожнього екрана;
- `home` – повертає курсор у лівий верхній кут діалогового вікна;
- `echo <file_name> on` – вмикає режим виводу на екран тексту Script-файлу (файлу-сценарія);
- `echo <file_name> off` – вимикає режим виводу на екран тексту Script-файлу;

- `echo <file_name>` – змінює режим виводу на екран на протилежний;
- `echo on all` – вмикає режим виводу на екран тексту всіх `m`-файлів;
- `verb echo off all` – вимикає режим виводу на екран тексту всіх `m`-файлів;
- `more on` – вмикає режим посторінкового виводу на екран (корисний для перегляду великих `m`-файлів);
- `more off` – вимикає режим посторінкового виводу (в цьому випадку для перегляду великих `m`-файлів користуватися лінійкою прокрутки);

1.3. Оператори та внутрішні функції MatLab®

Оператор – це спеціальне позначення для визначеної операції (дії) над даними – *операндами*. Оператори використовуються сумісно з операндами. Наприклад, у виразі $2+3$ знак “+” є оператором додавання, а числа 2 і 3 – операндами. Оператори також є об’єктами математичних виразів і мов програмування.

Зауважимо що більшість операторів належать до матричних операцій, що може зумовити певні непорозуміння. Наприклад, оператори множення “*” та ділення “/” обчислюють, відповідно, добуток і частку від ділення двох масивів, векторів і матриць. Крім того, є низка спеціальних операторів, наприклад, оператор “\” означає ділення *справа наліво*, а оператори “.*” та “./” означають, відповідно, *поелементне* множення та *поелементне* ділення масивів. Повний список операторів можна отримати, використовуючи команду `help ops`. Система повертає повідомлення, яке зображено у табл. 1.2.

Функції в середовищі MatLab® – об’єкти системи, які мають унікальні імена, виконують визначені перетворення своїх аргументів і повертають результати цих перетворень. *Повернення результатів* – характерна особливість функцій. У цьому випадку результат обчислення функції з одним вихідним параметром підставляється на місце звернення до неї, що дає змогу використовувати функції в математичних виразах, наприклад, функцію `sin` у виразі $2*\sin(\pi/2)$.

У загальному випадку функції мають список аргументів (параметрів), які поміщаємо в круглі дужки. Наприклад, функція Bessel'a записується у вигляді `bessel(NU,X)`. У цьому випадку список параметрів містить два аргумент – `NU` у вигляді скаляра та `X` у вигляді вектора. Деякі функції дають змогу записати в іншому форматі, який відрізняється списком параметрів. Якщо функція повертає декілька значень, то вона записується у вигляді

$$[Y1,Y2,...]=\text{func}(X1,X2,...)$$

де `Y1,Y2...` – список вихідних параметрів; `X1,X2...` – список вхідних аргументів (параметрів).

Перелік елементарних функцій, які вмонтовані в систему MatLab®, можна викликати командою

```
help elfun
```

а зі списком *спеціальних функцій* середовища – командою

```
help specfun
```

Функції можуть бути *вмонтованими* (внутрішніми) та *зовнішніми*, або *m-функціями*

Табл. 1.2. Список операторів MatLab®

```
>>help ops
Operators and special characters.
    Arithmetic operators.
```

| | | |
|----------|-----------------------------------|----|
| plus | - Plus | + |
| uplus | - Unary plus | + |
| minus | - Minus | - |
| uminus | - Unary minus | - |
| mtimes | - Matrix multiply | * |
| times | - Array multiply | .* |
| mpower | - Matrix power | ^ |
| power | - Array power | .^ |
| mldivide | - Backslash or left matrix divide | \ |
| mrdivide | - Slash or right matrix divide | / |
| ldivide | - Left array divide | .\ |
| rdivide | - Right array divide | ./ |

| | | |
|-----------------------|--|-----|
| idivide | - Integer division with rounding option. | |
| kron | - Kronecker tensor product | |
| Relational operators. | | |
| eq | - Equal | == |
| ne | - Not equal | ~= |
| lt | - Less than | < |
| gt | - Greater than | > |
| le | - Less than or equal | <= |
| ge | - Greater than or equal | >= |
| Logical operators. | | |
| relop | - Short-circuit logical AND | && |
| relop | - Short-circuit logical OR | |
| and | - Element-wise logical AND | & |
| or | - Element-wise logical OR | |
| not | - Logical NOT | ~ |
| punct | - Ignore function argument or output | ~ |
| xor | - Logical EXCLUSIVE OR | |
| any | - True if any element of vector is nonzero | |
| all | - True if all elements of vector are nonzero | |
| Special characters. | | |
| colon | - Colon | : |
| paren | - Parentheses and subscripting | () |
| paren | n- Bracketsn | [] |
| paren | - Braces and subscripting | { } |
| punct | - Function handle creation | @ |
| punct | - Decimal point | . |
| punct | - Structure field access | . |
| punct | - Parent directory | .. |
| punct | - Continuation | ... |
| punct | - Separator | , |
| punct | - Semicolon | ; |
| punct | - Comment | % |
| punct | - Invoke operating system command | ! |

| | | |
|--------------------|-----------------------------------|-----------|
| punct | - Assignment | = |
| punct | - Quote | ' |
| transpose | - Transpose | .' |
| ctranspose | - Complex conjugate transpose | ' |
| horzcat | - Horizontal concatenation | [,] |
| vertcat | - Vertical concatenation | [:,] |
| subsasgn | - Subscripted assignment | (),{ },. |
| subsref | - Subscripted reference | (),{ },. |
| subsindex | - Subscript index | |
| metaclass | - Metaclass for MatLab® class | ? |
| Bitwise operators. | | |
| bitand | - Bit-wise AND. | |
| bitcmp | - Complement bits. | |
| bitor | - Bit-wise OR. | |
| bitmax | - Maximum floating point integer. | |
| bitxor | - Bit-wise XOR. | |
| bitset | - Set bit. | |
| bitget | - Get bit. | |
| bitshift | - Bit-wise shift. | |
| Set operators. | | |
| union | - Set union. | |
| unique | - Set unique. | |
| intersect | - Set intersection. | |
| setdiff | - Set difference. | |
| setxor | - Set exclusive-or. | |
| ismember | - True for set member. | |

See also arith, relop, slash, function_handle.

1.3.1. Синтаксис операторів MatLab®

Вхідна мова MatLab® нараховує всього 9 операторів, які використовують 14 службових слів. Синтаксичні конструкції операторів подано у табл. 1.3.

Зауважимо, що мова MatLab® не містить оператора безумовного переходу за міткою go to. Тому у текстах m-файлів немає мітки операторів. Для ідентифікації стрічок, у яких виникають аварійні ситуації, використовуються внутрішні номери, які присвоюються системою автоматично.

Табл. 1.3. Синтаксис операторів MatLab®

| | Форма оператора | Призначення |
|---|--|---|
| 1 | <code>var=expr</code> | Оператор присвоєння. Обчислює значення виразу <code>expr</code> і присвоює його змінній <code>var</code> . |
| 2 | <code>if condition_1 operators_1 [elseif condition_2 operators_2 elseif condition_3 operators_3 else operators] end</code> | Умовний оператор. Якщо перевірка умови <code>condition_1</code> повертає <code>true</code> , тоді виконуються оператори <code>operators_1</code> , якщо справджується <code>condition_2</code> , тоді виконуються <code>operators_2</code> , і т.д. Якщо не справджується жодна з умов, тоді виконуються <code>operators</code> між <code>else</code> та <code>end</code> . |
| 3 | <code>switch expr case val1 operators_1 case val2 operators_2 [otherwise operators] end</code> | Перемикач стосовно значення виразу <code>expr</code> . Якщо це значення збігається з <code>val1</code> , тоді виконується група операторів <code>operators_1</code> ; якщо збігається з <code>val2</code> , тоді виконується група операторів <code>operators_2</code> , і т.д. Якщо значення <code>expr</code> не збігається з жодною із заданих величин, тоді виконуються група операторів між <code>otherwise</code> та <code>end</code> . |
| 4 | <code>for var=e1:[e2:]e3 operators end</code> | Петля типу математичної прогресії, у якій змінна <code>var</code> на кожному кроці повторення тіла петлі змінюється від початкового значення <code>e1</code> із кроком <code>e2</code> до кінцевого значення <code>e3</code> . Якщо кроку <code>e2</code> немає (його наявність є опційною), то за замовчуванням крок дорівнює 1. |
| 5 | <code>while condition operators end</code> | Цикл із передумовою, який повторюється доти, доки умова <code>condition</code> не прийме значення <code>true</code> . |
| 6 | <code>try operators_1 verb"catch" phtwoperators_2 end</code> | Намагання виконати групу операторів <code>operators_1</code> . За умови, що під час виконання <code>operators_1</code> виникає виняткова ситуація, керування передається групі операторів <code>operators_2</code> (використову- |

| | | |
|---|---|--|
| | | ється для опрацювання можливих збійних ситуацій). Якщо при опрацюванні <i>operators_1</i> помилки не виникає, тоді ігнорується виконання групи операторів <i>operators_2</i> . |
| 7 | <code>break</code> | Безумовне переривання керуючих конструкцій типу <code>for</code> , <code>while</code> , <code>switch</code> , <code>try-catch</code> |
| 8 | <code>function f1</code> <code>function</code> <code>f2(x1,x2,...)</code> <code>function</code> <code>y=f3(x1,x2,...)</code> <code>function</code> <code>[y1,y2,...]=f4(x1,x2,...)</code> | Заголовок функції (x_1, x_2, \dots – вхідні параметри; y_1, y_2, \dots – вихідні параметри). |
| 9 | <code>return</code> | Безумовний дотерміновий вихід із тіла функції |

Коментарі у *m*-файлах починаються з символу `%`. Вони можуть бути розташовані на початку рядка або правіше від оператора. Однак у написанні сценарію *m*-файлу коментарі, розташовані на початку, відіграють особливу роль. Перша група рядків послідовно заповнених коментарями до порожнього рядка утворює текст, який система повертає за командою `help` разом із іменем *m*-файлу.

★ **Приклад 1.3.1.** Наведемо приклад. Утворимо сценарій і збережемо його у файлі `fact5.m` обчислення значення $5!$, додавши пояснення до програми (див. табл. 1.4).

Табл. 1.4. Програма-сценарій обчислення значення $5!$

```
% This programm return 5!
a=1;
for i=1:5
    a=a*i;
end
a
```

У біжучому рядку командного вікна впишемо команду

```
>> fact5
```

Тоді система поверне обчислене значення:

```
a=
```

```
120
```

Якщо тепер у біжучому рядку командного вікна звернутися по допомогу до цієї явно несистемної функції, то MatLab® поверне повідомлення:

```
>> help fact5
```

```
This programm return 5!
```

1.3.2. Оператори керування виконання програми

Умовні оператори `if`, `else`, `elseif`

Конструкція оператора `if`

```
if condition
```

```
    block
```

```
end
```

виконує оператори *block*, якщо *condition* повертає вартість `true`. Якщо *condition* повертає `false`, то оператори *block* не виконуються. Якщо ж після умовного оператора `if` йдуть кілька операторів `elseif`

```
if condition
```

```
    block
```

```
elseif condition
```

```
    block
```

```
    .
```

```
    .
```

```
    .
```

```
end
```

то така конструкція працює аналогічно як вище. Якщо не виконується жодна з умов `elseif`, тоді виконуються оператори *block*, що слідує після `else`, який і замикає умовний оператор `if`:

```
    .
```

```
    .
```

```
    .
```

```
else
```

```
    block
```

```
end
```

★ **Приклад 1.3.2.** За допомогою умовного оператора `if` визначимо у середовищі MatLab® функцію `signum`. Для цього створимо такий `m`-файл:

```
function sgn=signum(a)
if a>0;
    sgn=1;
elseif a<0;
    sgn=-1;
else a=0;
    sgn=0;
end
```

та запишемо його під іменем `signum.m`. Після чого у `Command Window` (за умови, що `signum.m` записаний саме у **Current Folder**) обчислимо:

```
>> [signum(-2.456) signum(0) signum(5.23)]
ans =
    -1     0     1
```

Оператори циклу `switch`, `while`, `for`

Оператор `switch` має таку структуру

```
switch expression
    case value1
        block
    case value2
        block
    .
    .
    .
    otherwise
        block
end
```

Обчислюється вираз *expression* і передається на виконання того *block*, для якого *case* збігається з обчисленим значенням. Наприклад, якщо обчислене значення *expression* має значення *value1*, то виконується *block*, який слідує після нього. Якщо значення *expression* не збігається з жодним значенням *case*, то виконується *block* після *otherwise*. Продемонструємо це на прикладі.

★ **Приклад 1.3.3.** Створимо script-файл `valuetrigfunc.m`, за допомогою якого повертається значення тригонометричних функцій `sin`, `cos`, `tan`, та повертає повідомлення про невизначеність функції, якщо не збігається з переліченими:

```
function y=valuetrigfunc(func,x)
switch func
    case 'sin';
        y=sin(x)
    case 'cos';
        y=cos(x)
    case 'tan';
        y=tan(x)
    otherwise
        error('No such function defined')
end
```

Тепер у Command Window обчислимо

```
[valuetrigfunc('sin',pi/6);valuetrigfunc('cos',pi/3);...
valuetrigfunc('tan',pi/4);valuetrigfunc('exp',1)]
```

Отримали результат обчислення у вигляді стовпчика (елементи обчислюваного масиву розділені ”;”), перші три значення обчислені, а четвертий результат у вигляді повідомлення про помилку, позаяк намагалися обчислити функцію, яка не визначена:

```
y =
    0.5000
y =
    0.5000
y =
    0.1000
Error using valuetrigfunc (line 10)
No such function defined
```

Інструкція `while`, яка записується у вигляді

```
while condition
    block
end
```

виконує оператор `block`, поки `condition` повертає `true`. Наступний приклад

ілюструє використання циклу `while`.

★ **Приклад 1.3.4.** Обчислимо, через скільки років первинний капітал у розмірі 1000 \$ за річної відсоткової ставки у 6% зросте до 10000 \$. Цього впровадимо у біжучому рядку вікна `Command Window` системи MatLab® послідовність інструкцій:

```
>> p=1000; years=0;
while p<10000
    years=years+1;
    p=p*(1+0.06);
end
```

Після обчислення візуалізуємо одержаний результат:

```
>> years
```

У діалоговому вікні система поверне обчислене значення:

```
years =
    40
```

Цикл `for` передбачає наявність змінної *target*, яка послідовно приймає значення послідовності *sequence*. Структура циклу така

```
for target=sequence
    block
end
```

У підсумку виконується процедура *block* у кожному послідовному призначеному значенні змінної *target* значень послідовності *sequence*. Розглянемо наступний приклад використання циклу `for`.

★ **Приклад 1.3.5.** Обчислити перші 12 значень чисел Фібоначчі та результат подати у вигляді масиву двох стовпчиків, з яких перший – номер числа Фібоначчі, а другий – значення числа. Послідовність команд у біжучому рядку вікна `Command Window` виглядатиме так:

```
>> y(1)=1; y(2)=1;
for n=1:12;
y(n+2)=y(n)+y(n+1);
c(n)=n;
end
[c;y(1:12)]'
```

```
ans =
```

```

1    1
2    1
3    2
4    3
5    5
6    8
7   13
8   21
9   34
10  55
11  89
12 144

```

Зауважимо, що у цьому циклі *target n* приймає послідовно 12 значень від 1 до 12, для того, щоб обчислити масив *s* номерів, і водночас обчислюються перші 14 чисел Фібоначчі, оскільки перші два числа визначені до початку циклу. Для того, щоб повернути правильну відповідь, необхідно з масиву *y* ”вийняти” перші 12 значень.

Наступний приклад демонструє як у середовищі MatLab® обчислення за допомогою циклу можна організувати безпосередньо, не використовуючи оператор `for`.

★ **Приклад 1.3.6.** Обчислити послідовні значення функції $\cos x$ (протабулювати функцію) від 0 до $\pi/2$ з кроком $\pi/10$. Використовуючи оператор циклу `for` у середовищі MatLab®, процедура виглядатиме так:

```

>> for n=0:5 % loops over the sequence 0 1 2 3 4 5
y(n+1)=cos(n*pi/10);
end

```

```

>> y
y =
1.0000 0.9511 0.8090 0.5878 0.3090 0.0000

```

У цьому завданні потрібно було обчислити масив значень функції на масиві аргументу. А таке обчислення можна виконати безпосередньо:

```

>> n=0:5;
>> y(n+1)=cos(n*pi/10);
>> y
y =

```

```
1.0000    0.9511    0.8090    0.5878    0.3090    0.0000
```

Оператор break

Для переривання циклу використовують інструкцію `break`. Якщо у процесі обчислення програма натрапляє на `break`, то керування переадресується першій інструкції, яка йде після циклу.

★ *Приклад 1.3.7.* Наступний приклад програми, за допомогою якої створюється вектор довільної довжини, компоненти якого впроваджуються за вимогою. Процедура створення вектора припиняється, якщо немає елемента. Утворимо у редакторі середовища MatLab® скриптовий файл такого змісту:

```
function x=buildvec
for i=1:10000
    elem=input('==>'); %Promts for input of elements
    if isempty(elem) %Check for empty element
        break
    end
    x(i)=elem;
end
```

Тепер за допомогою процедури створимо два вектори `x1` та `x2` розмірності 4 та обчислимо їх прямиий добуток і скалярний добуток:

```
>> x1=buildvec
==>2
==>3
==>4
==>5
==>
x1 =
     2     3     4     5
```

```
>> x2=buildvec
==>3
==>4
==>5
==>6
==>
x1 =
```

```

    3  4  5  6
>> x1.*x2
ans =
    6  12  20  30
>> x1*x2'
ans =
    68

```

Отже, прямий добуток векторів $x1.*x2=[6\ 12\ 20\ 30]$, скалярний добуток $x1*x2'=68$. Як бачимо, процедура створення вектора з елементами до запитання створена так, що після кожного впровадження елемента виконується перевірка, чи введено значення цього елемента. Як тільки виявиться, що значення елемента не введено, інструкція `break` перериває цикл і завершує обчислення.

Оператор `continue`

Інструкція `continue` здійснює безумовний перехід до виконання наступної інструкції в циклі, яка йде за нею. Покажемо застосування цього оператора на прикладі

★ *Приклад 1.3.8.* Утворити функцію, яка б у стрічці `s1='This is too bed'` зліквідувала `'too'`. Запишемо `m`-файл такого змісту:

```

function s2=strip(s1)
s2='';    %Create an empty string
for i=1:length(s1)
    if s1(i)=='o'
        elseif s1(i)=='t'
            continue
        else
            s2=strcat(s2,s1(i));    %Concatenation%
        end
    end
end

```

На кожному кроці циклу виконується перевірка значення кожного елемента стрічки `s1`. Якщо елементом стрічки є символ `'o'` або `'t'`, тоді інструкція `continue` передає управління збільшення кроку циклу, тим самим пропустивши команду приєднання `i`-елемента стрічки `s1` до новостворюваної стрічки `s2`. Перевіримо дію утвореного сценарію:

```

>> s1='This is too bad.';

```

```
>> s2=strip(s1)
```

```
s2 =
```

```
Thisisbad.
```

Оператор return

Ця команда повертає керування виконуючій програмі, якщо відбувається звернення поза нею. Однак ця інструкція може використовуватися для закінчення процедури. Розглянемо приклад.

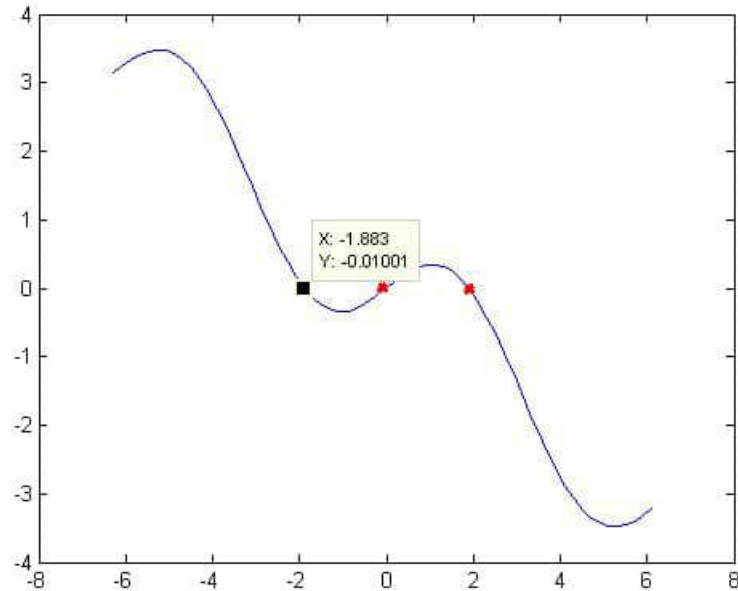
★ **Приклад 1.3.9.** Розглянемо програму знаходження нуля функції $f(x) = \sin x - 0.5x$ методом Newton–Raphson’a. Для заданого значення x (нульове наближення розв’язку) будеться послідовність ітерацій наближеного розв’язку за формулою $x \rightarrow x + \Delta x$, де $\Delta x = -f(x)/f'(x)$, доки Δx не буде менше наперед заданої малої величини. Для того, щоб закінчити процедуру обчислення використовується інструкція `return`. У циклі `for` використаємо 30 ітерацій, яких більше ніж достатньо для обчислення нуля функції з заданою точністю. Утворимо файл-сценарій під назвою `solve.m` такого змісту:

```
function x=solve(x)
numbIt=input('Input numbers iteration: ');
for numIter=1:numbIt
    dx=-(sin(x)-0.5*x)/(cos(x)-0.5); %-f(x)/f'(x)
    x=x+dx;
    if abs(dx)<1.0e-6 %check for convergence
        return
    end
end
error('Too many iterations')
```

Кількість ітерацій у програмі вводиться на вимогу. Обчислимо значення нуля функції для різних значень початкової точки та кількості кроків ітерацій. Для цього обчислимо графік функції $f(x) = \sin x - 0.5x$ на проміжку $[-2\pi, 2\pi]$:

```
>> t=-2*pi:0.2:2*pi;
y=sin(t)-0.5*t;
plot(t,y)
```

У результаті обчислення одержимо графік, з якого наближено можна визначити нулі функції (див. рис. 1.1).

Рис. 1.1. Графік функції $f(x) = \sin x - 0.5x$

Виберемо за початкову точку значення 3 і задамо 4 ітераційні кроки:

```
>> x=solve(3)
Input numbers iteration: 4
Error using solve (line 10)
Too many iterations
```

Програма повертає повідомлення, що при заданій точності кількість ітераційних кроків недостатня для отримання наближеного значення нуля функції.

Змінимо кількість ітераційних кроків при незмінному початковому значенні:

```
>> x=solve(3)
Input numbers iteration: 5
x =
    1.8955
```

Одержали результат, який із заданою точністю визначає один із нулів функції $x = 1.8955$.

Оператор error

У процесі виконання програма може бути зупинена за допомогою інструкції `error` і на моніторі можна одержати повідомлення. Цей приклад демонструє описану ситуацію – задана кількість ітераційних кроків недостатня для обчислень і тоді програма виводить на монітор відповідне повідомлення. Формат інструкції такий:

```
error('message')
```

Наведемо приклад застосування цього оператора.

* **Приклад 1.3.10.** Утворимо процедуру, за допомогою якої визначатимемо розмірність квадратної матриці. У випадку, якщо матриця не є квадратною, процедура повертає повідомлення про помилку. У біжучому рядку вікна `Command Window` послідовно визначимо квадратну матрицю 3×3 :

```
>> mA=randi([-10,10],3)
```

```
mA =
```

```
    7    9   -5  
    9    3    1  
   -8   -8   10
```

та саму програму:

```
>> [m,n]=size(mA);
```

```
if m~=n
```

```
error('Matrix must be square')
```

```
else
```

```
[m,n]
```

```
end
```

```
ans =
```

```
    3    3
```

Перевіримо тепер, який одержимо результат, якщо матриця не є квадратною. Згенеруємо, наприклад, прямокутну матрицю 3×5 та застосуємо до неї процедуру визначення розмірності квадратної матриці:

```
>> mA=randi([-10,10],3,5)
```

```
mA =
```

```
   10   10   -8    6  -10  
   -7    0   -2   10    7  
    7    6    9    3    9
```

```
>> [m,n]=size(mA);
if m~=n
error('Matrix must be square')
else
[m,n]
end
Matrix must be square
```

І накінець, кілька прикладів керуванням виводу одержаної у процесі обчислення інформації.

★ *Приклад 1.3.11.* Вивести на екран позначення величини та значення величини:

```
>> R=input('radius=');
a=pi*R^2;
x=['surface=',num2str(a)];
disp(x)
```

Після натискання Enter (команда на виконання) у командній стрічці вікна Command Window з'явиться запит на очікуване значення радіуса:

```
radius=
```

Уведемо, наприклад, 3 і натиснемо Enter. Система поверне обчислене значення площі кола радіуса $R = 3$:

```
surface=28.2743
```

Тут використали інструкцію num2str(a), яка змінює об'єкт "число" на об'єкт "стрічку". Змінній x присвоєна стрічка – масив символів, довжина якої:

```
>> length(x)
ans =
    15
```

Справді, видрукуємо всі елементи цього масиву:

```
>> for k=1:length(x)
disp(x(k))
end
s
u
r
f
a
```

```
c
e
=
2
8
.
2
7
4
3
```

У процесі обчислення виконано присвоєння числового значення площі змінній **a**, тому змінна **a** залишається числом:

```
>> a
a =
28.274333882308138
```

Якщо необхідно, щоб змінна циклу набувала послідовні значення з наперед заданого списку **i**, наприклад, ці значення були віддруковані на екрані, тоді можна застосувати таку процедуру:

```
>> y=[2 -1 5 -3 7];
for i=y;
x=['value i=',num2str(i)];
disp(x)
end
```

Розділ 2

Спеціальні обчислення

2.1. Основні операції математичного аналізу

У цьому розділі опишемо способи та комп'ютерні технології обчислення суми і добутку послідовності чисел, подання функції у табличному вигляді, обчислення похідних, границь, особливих точок, розвинення функцій у ряди.

2.1.1. Табулювання функції

Математичні функції можуть бути визначені за допомогою формули, таблиці, графіка. Табличне подання функції потрібне у таких випадках:

- визначення похибки інтерполяції;
- обчислення табличних різниць з метою визначення степеня інтерполяції полінома;
- визначення області ізоляції кореня;
- оцінка числового значення функції у широкому діапазоні аргументу.

У системі MatLab[®] табулювання функції виконується за допомогою інструкції `subs` у такому форматі:

$$\text{subs}(f,x,x1)$$

де f – задана аналітично функція; x – аргумент функції f ; $x1$ – вектор значень аргументу, за яких визначаються значення функції f .

Змінна $x1$ може задаватися у вигляді вектора або при постійному

кроці у вигляді: $x_s : \Delta x : x_f$.

Алгоритм табулювання функції $f(x)$ такий:

1. Визначення групи символічних змінних за допомогою функції `syms`.
2. Створення вектора x .
3. Означення табульованої функції $y = f(x)$.
4. Створення функції табулювання `subs`.
5. Отримання результату шляхом компіляції за допомогою натискання `<Enter>`.

За допомогою інструкції `subs` можна табулювати одночасно декілька функцій. Для цього треба функцію f подати у вигляді матриці табульованих функцій, наприклад,

```
f=[exp(x);x.^2;sin(x)]
```

MatLab[®] дає змогу табулювати функції, використовуючи матричні операції, не звертаючись до функції `subs`. Алгоритм обчислення у цьому випадку такий:

1. означення символічних змінних за допомогою функції `syms`;
2. створення вектора аргументу x ;
3. створення матриці, елементами якої є аргумент x і табульовані функції;
4. отримати результат, натиснувши `<Enter>`;
5. у разі потреби отримати результат у вигляді стовпця, використовуючи інструкцію y' .

★ **Приклад 2.1.1.** Продемонструємо дію оператора `syms` на такому прикладі.

```
>> syms x y;  
x=0:.2:1;  
y=[x;exp(x);sin(x);cos(x)];  
y'
```

```
ans =
      0 1.0000      0 1.0000
 0.2000 1.2214 0.1987 0.9801
 0.4000 1.4918 0.3894 0.9211
 0.6000 1.8221 0.5646 0.8253
 0.8000 2.2255 0.7174 0.6967
 1.0000 2.7183 0.8415 0.5403
```

2.1.2. Обчислення суми елементів масиву чисел

Обчислення суми елементів масиву чисел у MatLab[®] виконується за допомогою інструкцій `sum` та `cumsum`, які мають такий синтаксис:

```
sum(x)
cumsum(x)
```

де x вектор або матриця елементів підсумовування. Якщо x – вектор, то `sum(x)` повертає суму елементів вектора, якщо ж x – матриця, то результатом обчислення буде вектор, кожен елемент якого є сумою елементів відповідного стовпця матриці.

Функція `cumsum(x)` повертає суми елементів та всі проміжні результати підсумовування.

* **Приклад 2.1.2.** Розглянемо приклади дії розглянутих інструкцій на елементи різних форматів. Утворимо одновимірний масив $x1$ натуральних чисел від 1 до 10 та застосуємо до нього інструкції `sum` і `cumsum`:

```
>> x1=1:10
```

```
x1=
```

```
 1  2  3  4  5  6  7  8  9 10
```

```
>> s1s=sum(x1)
```

```
s1s=
```

```
 55
```

```
>> s1c=cumsum(x1)
```

```
s1c=
```

```
 1  3  6 10 15 21 28 36 45 55
```

```
>> s1cc=cumsum(x1')
```

```
s1cc=
```

```
1
3
6
10
15
21
28
36
45
55
```

В останньому обчисленні застосовано `cumsum` до масиву-стовпця, в результаті чого одержали масив-стовпець з такими як у випадку рядка акумульованими сумами.

Розглянемо дію інструкцій `sum` і `cumsum` на двомірний масив:

```
>> x2=[1 2 3 4;2 3 4 5;3 4 5 6;4 5 6 7]
```

```
x2=
```

```
1 2 3 4
2 3 4 5
3 4 5 6
4 5 6 7
```

```
>> s2=sum(x2)
```

```
s2=
```

```
10 14 18 22
```

```
>> s2c=cumsum(x2)
```

```
s2c=
```

```
1 2 3 4
3 5 7 9
6 9 12 15
10 14 18 22
```

2.1.3. Обчислення добутку елементів масиву

Обчислення добутку елементів масиву x у MatLab[®] виконуємо за допомогою інструкцій `prod` і `cumprod` у такому форматі

```
prod(x)
cumprod(x)
```

де x – вектор або матриця елементів. Якщо x – вектор, то результатом

обчислення $\text{prod}(x)$ буде добуток елементів вектора x , якщо ж x – двомірний масив (матриця), то результатом обчислення буде вектор, кожен елемент якого є добутком елементів відповідного стовпця матриці.

◆ *Завдання 2.1.1.* Обчислити:

- 1) добуток чисел від 1 до 10 ($10!$);
- 2) добуток елементів вектора $[1,4,9,16,25]$;
- 3) добуток квадратів елементів $[1,4,9,16,25]$;
- 4) часткові добутки стовпців матриці $[1,2,3,4;2,3,4,5;3,4,5,6;4,5,6,7]$.

2.1.4. Обчислення похідних

Обчислення похідної функції у MatLab[®] виконуємо за допомогою інструкції `diff`, формат якої

$$\text{diff}(f,x,n)$$

де f – диференційовна функція; x – змінна інтегрування; n – порядок диференціювання (за замовчуванням $n = 1$).

Алгоритм обчислення похідної функції такий:

- 1) означення символічних змінних за допомогою інструкції `syms`;
- 2) ввід диференційовної функції;
- 3) ввід інструкції `diff(f,x,n)` із заданими x та n ;
- 4) отримання результату диференціювання натисканням `<Enter>`.

★ *Приклад 2.1.3.* Обчислити похідні до третього порядку включно функції $f(x) = x \cos x$. У середовищі MatLab[®] процедура обчислення має вигляд:

```
>> syms x
>> f1=x*cos(x);
>> [diff(f1,x),diff(f1,x,2),diff(f1,x,3)]
ans=
[cos(x)-x*sin(x), -2*sin(x)-x*cos(x), x*sin(x)-3*cos(x)]
```

Оператор `diff` дає змогу обчислювати похідні функції, до якої входять

параметри.

★ **Приклад 2.1.4.** Обчислити похідні функцій $y_1(x) = ax^2$, $y_2(x) = n^x$, $y_3(x) = e^{-ax^5} + \ln(a^n + x^a) - \frac{an}{x^n}$. Процедура обчислення така:

```
>> syms a x n
      y1=a*x^2;y2=n^x;y3=exp(-a*x^5)+log(a^n+x^z)-a*n/(x^3);
>> z1=diff(y1,x)
ans=
      2*a*x
>> z2=diff(y2,x,3)
ans=
      n^x*log(n)^3
>> z3=diff(y3,x)
ans=
      (a*x^(a-1))/(a^n+x^a)+(a*n^2)/x^(n+1)-5*a*x^4*exp(-a*x^5)
```

Оператор диференціювання у середовищі MatLab[®] має деякі особливості. Якщо змінної диференціювання x у виразі `diff` немає, а формат оператора має вигляд `diff(f)`, то програма не поверне повідомлення про помилку. Похідна буде обчислена за параметром, який входить у визначення функції, а якщо їх є декілька, то за тим параметром, який за алфавітом розташований на останньому місці у послідовності параметрів функції. Наприклад, якщо функція містить параметри a , b , c , то похідна буде обчислена за параметром c , тобто, у випадку, коли функція залежить від параметрів і змінної x , тоді `diff(f)` поверне похідну за змінною x , позаяк за алфавітним порядком змінна x розташована на останньому місці. Проілюструємо це на прикладах.

★ **Приклад 2.1.5.** Виконаємо такі обчислення похідної функції, яка залежить від параметрів:

```
>> syms a b c x w
>> diff(a+b^2)
ans=
      2*b
>> diff(a+c*b^2)
ans=
      b^2
```

```
>> diff(a*w+c*b^3)
```

```
ans=
```

```
a
```

```
>> diff(x*a+w+b^3)
```

```
ans=
```

```
a
```

Ще одна особливість програми `diff` у `MatLab`[®] – функція f може бути скаляром, вектором або матрицею. Результатом виконання також буде скаляр, вектор чи матриця.

★ **Приклад 2.1.6.** Обчислити похідну вектор-функції:

```
>> syms a x
```

```
>> y=[x*sin(x) x^5 exp(a*x)];
```

```
>> diff(y,x)
```

```
ans=
```

```
[ sin(x) + x*cos(x), 5*x^4, a*exp(a*x)]
```

2.1.5. Обчислення границь

У випадках, коли серед обчислюваних виразів трапляється невизначеність типу $\frac{0}{0}$, система повертає помилку у вигляді `NaN`, тобто необчислене значення виразу. В цьому випадку належить проаналізувати результат обчислення програми і для такого виду обчислення виразів застосувати правило de L’Hospital’a. Система комп’ютерної алгебри дає змогу знаходити границі функцій і у випадках, коли трапляються невизначеності виду $\frac{0}{0}$, $0 \cdot \infty$, $\frac{\infty}{\infty}$. У системі `MatLab`[®] границі обчислюються за допомогою інструкції `limit` у форматі:

$$\text{limit}(f,x,x_0)$$

де f – функція, якої обчислюється границя; x – аргумент функції f ; x_0 – граничне значення x .

Найчастіше обчислюються границі при $x_0 = 0$ та $x_0 = \infty$. У системі `MatLab`[®] символ ∞ кодується за допомогою ідентифікатора `inf`.

★ **Приклад 2.1.7.** Обчислити границі функцій: $y_1 = \frac{\sin x}{x}$ при $x \rightarrow 0$; $y_2 = x \ln\left(1 + \frac{3}{x}\right)$ при $x \rightarrow \infty$; $y_3 = \frac{1-x}{\ln x}$ при $x \rightarrow 1$.

Обчислення у `MatLab`[®] виконуємо так:

```

>> syms x
>> limit(sin(x)/x,x,0)
ans=
1
>> limit(x*log(1+3/x),x,inf)
ans=
3
>> limit((1-x)/ln(x),x,1)
ans=
-1

```

◆ *Завдання 2.1.2.* Обчислити у середовищі MatLab[®] границі функцій у зазначених граничних точках:

| | Функція | Граничне значення аргументу |
|----|--|-----------------------------|
| 1 | $(1+x)^{\frac{1}{x}}$ | ∞ |
| 2 | $\left(\frac{a^x+b^x}{2}\right)^{\frac{1}{x}}$ | 0 |
| 3 | $\left(1+\frac{1}{x}\right)^{\frac{1}{x}}$ | ∞ |
| 4 | $(1-x)^{\frac{1}{1-x}}$ | 1 |
| 5 | $(1-x)^{\frac{1}{1-x}}$ | ∞ |
| 6 | $\frac{1-e^{-at}}{\ln(1-t)}$ | 0 |
| 7 | $\frac{1-a^n}{an}$ | 0 |
| 8 | $2a\frac{1-e^{-ax}}{2-e^{-ax}}$ | ∞ |
| 9 | $\frac{x-a}{\ln(x-a+1)}$ | a |
| 10 | $\frac{n^2-n^x}{x-2}$ | 2 |

2.1.6. Розвинення функції у степеневий ряд

Розвинення функції у степеневий ряд у системі MatLab[®] виконується за допомогою оператора `taylor`, формат якої

$$\text{taylor}(f(x), x, x_0, n)$$

де $f(x)$ – функція, розвинення у ряд якої обчислюється; x – аргумент функції f ; x_0 – значення аргументу, в околі якого обчислюється розвинення; n – порядок розвинення.

Процедура обчислення розвинення така:

- 1) визначення символічних змінних за допомогою функції `syms`;
- 2) означення функції $f(x)$;
- 3) ввід функції `taylor(y, x, x0, n)`;
- 4) отримання результату (клавіша `<Enter>`).

◆ *Завдання 2.1.3.* Обчислити розвинення в ряд Taylor'а порядку $n = 5$ заданих функцій в околі точки $x_0 = 0$:

| | |
|---|---------------------------------------|
| 1 | $y_1 = \sin x;$ |
| 2 | $y_2 = e^x;$ |
| 3 | $y_3 = \text{sh } x;$ |
| 4 | $y_4 = \frac{3x + 1}{2x^2 + 5x + 1};$ |
| 5 | $y_5 = \ln x;$ |
| 6 | $y_6 = \frac{\sin x}{x}.$ |

♣ *Зауваження 2.1.1.* Функцію $y_5 = \ln x$ програма не розвинула у степеневий ряд, позаяк функція в точці $x = 0$ невизначена.

У разі розвинення функції у степеневий ряд треба брати до уваги таке:

- чи можливо функцію взагалі розвинути у степеневий ряд в околі заданої точки?

- яку кількість членів ряду потрібно взяти, щоб забезпечити потрібну точність?
- як обчислити похибку отриманого розвинення?

Відповідь на перше питання отримуємо з теореми Taylor'a – функція в околі розглядуваної точки має бути $n + 1$ раз неперервно диференційовною. Необхідна кількість членів розвинення залежить від функції та значення аргументу. У випадку, якщо ряд знакопозаперечений, то можна скористатися ознакою збіжності знакопозаперечених рядів. Якщо ж ряд не знакопозаперечений, то правила вибору кількості членів немає.

Похибку ряду можна оцінити, використовуючи такі методи комп'ютерних технологій:

- табулюванням заданої функції та отриманим розвиненням;
- побудовою графіків функцій;
- обчисленням похибок.

Розглянемо перший метод на прикладі.

★ **Приклад 2.1.8.** Розвинемо функцію $y = e^x$ в ряд Taylor'a при $n = 3, 4, 5$:

$$y_3 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6};$$

$$y_4 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24};$$

$$y_5 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}.$$

Протабулюємо функції $y = e^x$, y_3 , y_4 , y_5 при зміні аргументу x в межах від 0 до 2 з кроком $h = 0.2$. Процедура обчислення у системі MatLab® має вигляд:

```
>> syms x y y3 y4 y5;
>> x=0:.2:2;
>> y=exp(x);
>> y3=1+x+(x.^2)/2+(x.^3)/6;
>> y4=1+x+(x.^2)/2+(x.^3)/6+(x.^4)/24;
>> y5=1+x+(x.^2)/2+(x.^3)/6+(x.^4)/24+(x.^5)/120;
```

```
>> z=[x;y;y3;y4;y5] '
z=
    0  1.0000  1.0000  1.0000  1.0000
  0.2000  1.2214  1.2213  1.2214  1.2214
  0.4000  1.4918  1.4907  1.4917  1.4918
  0.6000  1.8221  1.8160  1.8214  1.8220
  0.8000  2.2255  2.2053  2.2224  2.2251
  1.0000  2.7183  2.6667  2.7083  2.7167
  1.2000  3.3201  3.2080  3.2944  3.3151
  1.4000  4.0552  3.8373  3.9974  4.0422
  1.6000  4.9530  4.5627  4.8357  4.9231
  1.8000  6.0496  5.3920  5.8294  5.9869
  2.0000  7.3891  6.3333  7.0000  7.2667
```

Із отриманих результатів обчислень робимо висновок, що похибка зменшується зі збільшенням числа n . Про величину похибки візуально можуть свідчити графіки функцій, які неважко обчислити у MatLab[®]. Найбільш ефективним методом обчислення похибки є визначення середньоквадратичного відхилення.

2.1.7. Нулі функцій

Обчислення наближеного значення точки x нуля функції однієї змінної у MatLab[®] виконуємо за допомогою інструкції

```
x=fzero(func,xs)
```

де `func` – нелінійна функція, для якої обчислюються нулі; `xs` – початкове значення нуля функції.

★ *Приклад 2.1.9.* Знайти наближене значення нуля функції та вивести на екран точки і значення функції у цій точці. Виконаємо це завдання, наприклад, за допомогою процедури:

```
>> clear; f=inline('exp(x^2)-3*sin(2*x)');
xs=input('Start xs=');
x0=fzero(f,xs);
y1=['x0=',num2str(x0)];
y2=['f(x0)=',num2str(f(x0))];
disp(y1);disp(y2);
```

За початкове наближення нуля функції $f(x) = e^{x^2} - 3 \sin(2x)$ виберемо,

наприклад, $x_s=0$ і в результаті виконання процедури одержимо результат:

```
Start xs: 0
xs =
    0
x0= 0.17546
f(x0)= -2.2204e-16
```

Отже, за початкового наближення $x_0=0$ інструкція `fzero` середовища MatLab[®] повернула наближене значення нуля $x_0=0.17546$ і значення функції у цій точці $f(x_0)=-2.2204e-16$.

Виконаємо графічну інтерпретацію попередньої задачі.

★ **Приклад 2.1.10.** Обчислити графіки функцій $f(x) = e^{x^2} - 3\sin(2x)$, $f_1(x) = e^{x^2}$, $f_2(x) = 3\sin(x)$. У першому графіці вивести вісь OX , два останні графіки обчислити в одній системі координат. На обчислених графіках зобразити одержану у попередньому завданні точку $(x_0, f(x_0))$.

```
>> f1=inline('exp(x.^2)'); f2=inline('3*sin(2*x)');
figure(1); ezplot(f, [0,0.6]);
hold on
ezplot('0', [0,.6]); plot(x0,f(x0),'m*');
hold off
figure(2); ezplot(f1, [0,.6]);
hold on
ezplot(f2, [0,.6]); plot(x0,f(x0),'ro');
hold off
```

Результат обчислення зображений на рис. 2.1.

Для обчислення нулів многочлена у MatLab[®] слугує інструкція `roots`, яка має формат

```
r=roots(c)
```

де \mathbf{r} – масив всіх нулів многочлена $p(x) = c_0x_n + c_1x^{n-1} + \dots + c_n$, \mathbf{c} – масив коефіцієнтів многочлена $p(x)$.

★ **Приклад 2.1.11.** Для многочлена $p(x) = x^3 - 10x^2 - 5x + 1$ знайти нулі, побудувати графік і його нулі.

```
>> clear; pol=@(x)x.^3-10*x.^2-5*x+1;
coefPol=[1 -10 -5 1];
rt=roots(coefPol);
lb=min(rt); rb=max(rt);
figure(3); ezplot(pol,[lb-.5,rb+.5]);
hold on
ezplot('0',[lb-.5,rb+.5]);
for k=rt
    plot(k,0,'r*')
end
hold off
```

Результат виконання останньої процедури див. рис. 2.2.

2.1.8. Визначення екстремумів функції

Класично екстремуми функції визначаються за допомогою похідних, який у системі MatLab[®] виконуємо за допомогою процедури

```
diff(f,x)
solve('fun',x)
```

Розглянемо приклад у системі MatLab[®].

★ **Приклад 2.1.12.** Визначити координати максимуму функції $y = xe^{-x}$.

1. Визначимо область $x_1 \leq x \leq x_2$ знаходження максимуму функції.

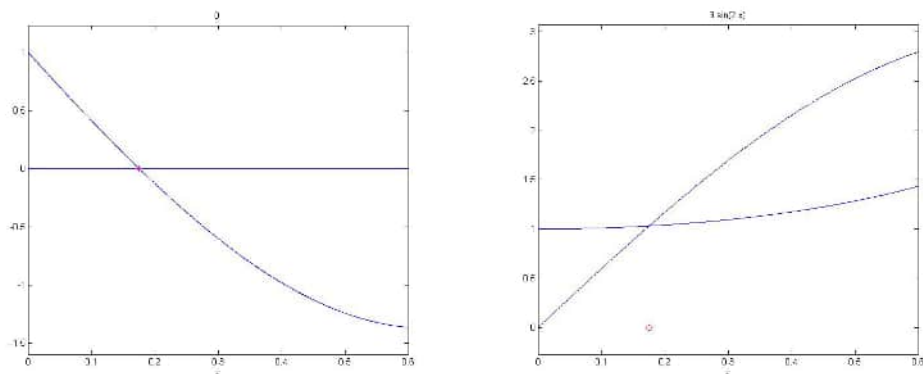


Рис. 2.1. Графік до прикладу відшукування нулів функції

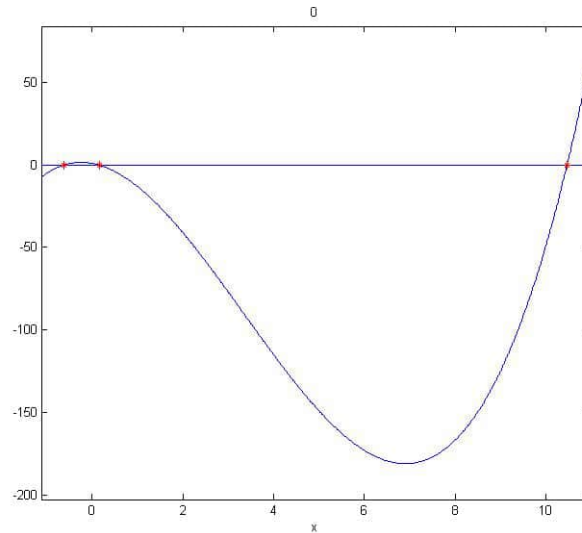


Рис. 2.2. Нулі многочлена (до прикладу 2.1.11)

Побудуємо графік функції на відрізку $[0, 3]$:

```
>> x=0:.1:3;
```

```
>> y=x.*exp(-x);
```

```
>> plot(x,y)
```

З побудованого графіка функції, зображеного на рис. 2.3, визначаємо проміжок $0.5 \leq x \leq 1.5$, на якому міститься максимум функції.

2. Обчислимо похідну функції:

```
>> syms x y z
```

```
>> z=diff(y,x)
```

```
z=
```

```
exp(-x) - x*exp(-x)
```

3. Визначимо корені рівняння:

```
>> solve('exp(-x)-x*exp(-x)=0',x)
```

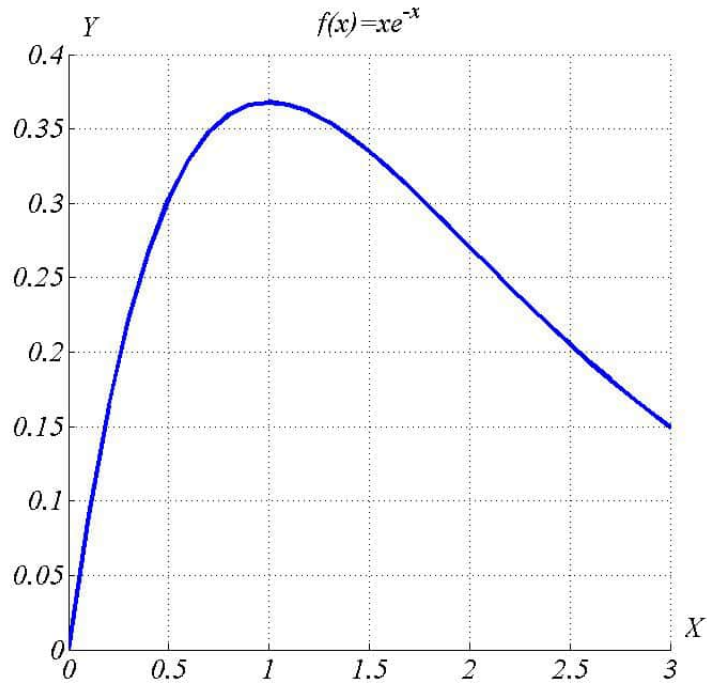
```
ans=
```

```
1
```

4. Визначимо ординату точки екстремуму функції:

```
>> x=1;
```

```
>> y=x*exp(-x)
```

Рис. 2.3. Графік функції $y = xe^{-x}$

```
y=
0.3679
```

Проте MatLab[®] має у своєму розпорядженні функції, які дають змогу обчислити екстремум функції, не визначаючи похідної самої функції. Це є інструкція `fminbnd`, синтаксис якої має вигляд:

```
fminbnd('fun',x1,x2)
```

де `'fun'` – функція, мінімум якої обчислюється, взята в одинарні лапки; `x1,x2` – відповідно лівий і правий кінці проміжку, на якому міститься шуканий мінімум функції.

Якщо маємо за мету обчислити максимум функції $f(x)$, тоді замість $f(x)$ мінімізуємо функцію $-f(x)$. Розглянемо приклади.

★ **Приклад 2.1.13.** Знайти мінімум функції $f(x) = \sin^3 x + \cos^3 x$ на проміжку $[0, 1]$.

Процедура у системі MatLab[®] має вигляд:

```
>> syms x
>> x=fminbnd('sin(x).^3+cos(x).^3',0,1);
x=
    0.7854
>> y=sin(x).^3+cos(x).^3
y=
    0.7071
```

Перший рядок процедури у середовищі MatLab[®] декларує, що змінна величина x є символом. У другому рядку процедури обчислюється власне точка, у якій задана функція досягає мінімуму на проміжку $[0, 1]$, і обчислене значення присвоюється змінній x . У третьому рядку обчислюється мінімальне значення заданої функції на зазначеному проміжку і обчислене значення присвоюється змінній y (рис. 2.4). Побудуємо графік функції на проміжку $[-\pi, \pi]$ та переконаємося у правильності обчислень.

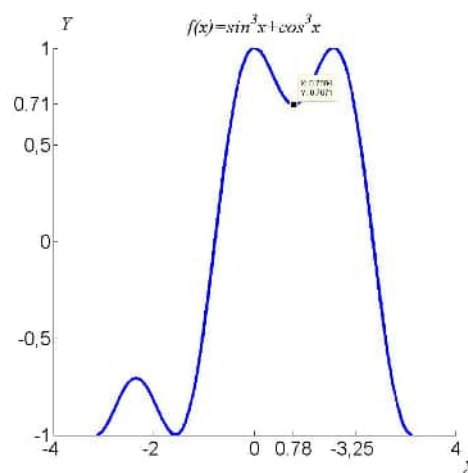


Рис. 2.4. Графік функції $f(x) = \sin^3 x + \cos^3 x$

```
>> x=-pi:.01:pi;
>> plot(x,sin(x).^3+cos(x).^3)
```

* **Приклад 2.1.14.** Визначити максимум функції $f(x) = 2^{-x} - 3^{-x} - 10x$ на проміжку $[-3, 0]$.

У цьому завданні скористаємося дещо іншим форматом визначення функції, екстремум якої треба обчислити. Результат обчислення графіка функції та точки максимуму зображено на рис. 2.5.

```
>> clear x myFun2 y
>> syms x
```

```

>> myFunc=@(x) 2.^(-x)-3.^(-x)-10*x;
>> fMin=@(x) -myFunc(x);
>> xm=fminbnd(fMin,-3,0)
xm=
-2.2734
>> ym=myFunc(xm)
ym=
15.4155
>> t=-4:.01:1;
>> figure(2);plot(t,myFunc(t))
>> hold on;plot(xm,ym,'r*');hold off

```

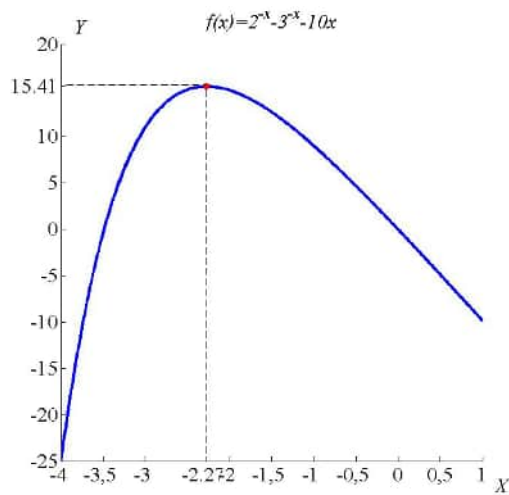


Рис. 2.5. Графік функції $f(x) = 2^{-x} - 3^{-x} - 10x$ та точка максимуму. Виконаємо у системі MatLab®

Для обчислення локального мінімуму функції багатьох змінних у системі MatLab® слугує функція `fminsearch`, синтаксис якої

$$\text{fminsearch}(\text{fun}, x_0)$$

де fun — функція, мінімум якої треба обчислити; x_0 — початкова точка. Розглянемо приклад.

★ **Приклад 2.1.15.**

Знайти екстремум функції $z = e^{-(x^2+xy+y^2)}(5x+7y-25)$.

Побудуємо графік функції та визначимо тип екстре-

послідовність інструкцій:

```

>> [x,y]=meshgrid(-2:.1:2);%генерування сітки
                %зміни аргументів на площині OXY
>> z=exp(-(x.^2+x.*y+y.^2)).*(5*x+7*y-25);%визначення функції
>> surfc(x,y,z)%побудова поверхні з одночасною побудовою
                % ліній рівня

```

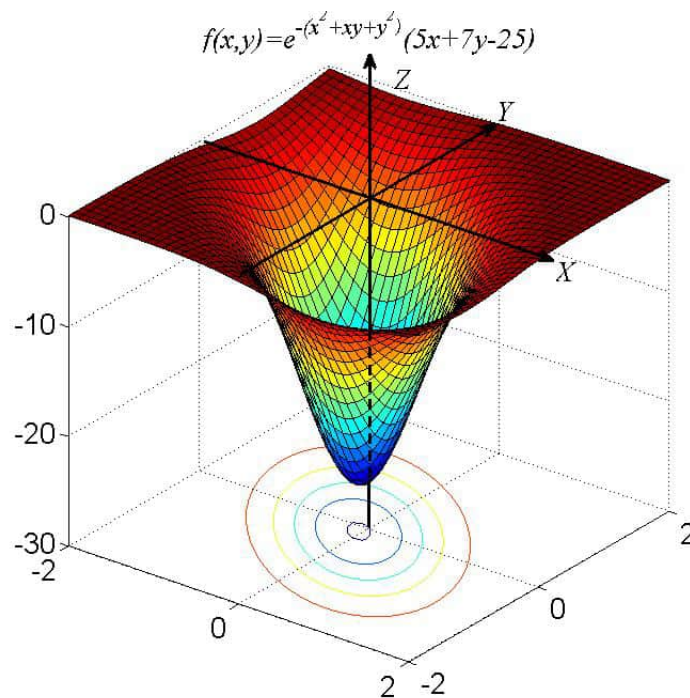


Рис. 2.6. Графік поверхні $f(x, y) = e^{-(x^2+xy+y^2)}(5x + 7y - 25)$ та її лінії рівня

Використавши графік поверхні (див. рис. 2.6), для обчислення мінімального значення функції за початкову виберемо точку $(1, -1)$:

```
>> [x,fVal]=fminsearch('exp(-(x(1)^2+x(1)*x(2)+x(2)^2))*...  
                    (5*x(1)+7*x(2)-25)', [1,-1])
```

```
x=
```

```
-0.0384 -0.1154
```

```
fval=
```

```
-25.5048
```

Отже, у точці $(-0.0384, -0.1154)$ функція набуває мінімального значення -25.5048 .

Після виконання останньої інструкції програма повертає вектор $[x,fVal]$, складовими якого є координати точки мінімуму x і значення функції у цій точці $fVal$. Звернемо увагу на синтаксис останньої команди – при позначенні змінних, наприклад, через x та y , система повертає

помилку. Тому використовуємо ідентифікатор точки x , розрізняючи ці координати як перша $x(1)$ та друга $x(2)$ складові. Розглянемо ще один приклад, використовуючи інший синтаксис функцій середовища.

★ **Приклад 2.1.16.** Знайти локальний екстремум функції

$$f(x, y) = e^{-x^2+y^2} (6xy - 8x^2 - 13y^2 - 0,4x - y)$$

в околі точки $(0, 0)$.

Побудуємо за допомогою MatLab[®] графік функції:

```
>> [x,y]=meshgrid(-.6:.01:.7);%обчислюємо сітку у системі...
                                     %координат оХУ на площині
>> z=exp(-2*x.^2+y.^2).*(-8*x.^2+6*x.*y-13*y.^2-.4*x-y);...
                                     %обчислюємо значення функції
>> figure(5);surfc(x,y,z)%обчислюємо графік із лініями...
                                     % рівня функції
```

Із побудованого графіка переконуємося, що функція в околі початку

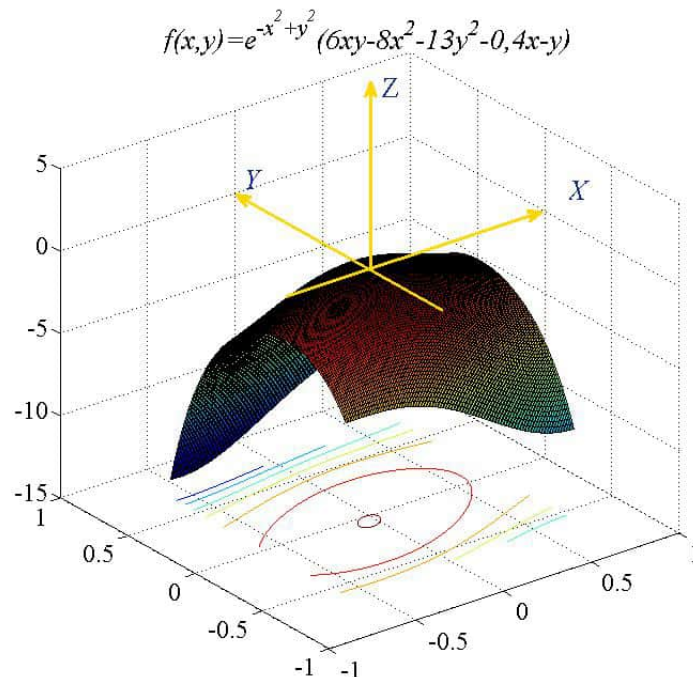


Рис. 2.7. Графік функції до прикладу 2.1.16

координат досягає локального максимуму (див. рис. 2.7). Щоб визначити максимальне значення та точку локального максимуму скористаємося такою процедурою.

```
>> myFun3=@(x) exp(-2*x(1)^2+x(2)^2)*(8*x(1)^2-6*x(1)*x(2)+...
    13*x(2)^2+ 0.4*x(1)+x(2));
>> [xmx,fVal3]=fminsearch(myFun3,[0,0])
xmx=
    -0.0430    -0.0485
fval=
    -0.0329
```

У першій стрічці визначаємо функцію, у другій – обчислюємо точку мінімуму та мінімальне значення функції, яку отримали шляхом домноження на -1 . В результаті виконання останньої інструкції одержали координати точки $xmx = (-0.0430, -0.0485)$ максимуму досліджуваної функції та її максимальне значення, яке дорівнює $-fval = 0.0329$.

2.1.9. Символьне інтегрування

У системі MatLab[®] для символьного обчислення інтеграла використовується функція `int` у таких форматах:

```
int(expr,var)
int(expr,var,a,b)
```

Функція `int(expr,var)` повертає невизначений інтеграл від функції `expr` за змінною `var`. Другий аргумент функції `int` опціональний – якщо він не зазначається, інтегрування проводиться за однією зі змінних виразу `expr`, що визначається за допомогою `syms`.

★ **Приклад 2.1.17.** У програмному середовищі MatLab[®] обчислити невизначені інтеграли:

```
>> syms a b x t% декларуємо символьні змінні
```

1. $\int \frac{x^2}{1+x^3} dx :$

```
>> int(x^2/(1+x^3),x)
ans =
    log(x^3 + 1)/3
```

Застосуємо інший формат інструкції інтегрування:

```
>> int(x^2/(1+x^3))% без декларування змінної
      % інтегрування
ans =
    log(x^3 + 1)/3
```

У двох останніх прикладах результати такі самі.

2. $\int a \cos(ax) dx$ (порівняйте результати, які одержали залежно від застосованого формату інструкції `int`):

```
>> int(a*cos(a*x), x)
ans =
    sin(a*x)
```

```
>> int(a*cos(a*x))% інтегрування за замовчуванням за
      % змінною x
ans =
    sin(a*x)
```

```
>> int(a*cos(a*x), a)% інтегрування за вказаною змінною
ans=
    (cos(a*x) + a*x*sin(a*x))/x^2
```

3. $\int \cos(bt)e^{x \sin(bt)} dt$. Зверніть увагу на одержані результати інтегрування та застосований формат інструкції інтегрування:

```
>> int(cos(b*t)*exp(x*sin(b*t)))
ans=
    exp(x*sin(b*t))*cot(b*t)
```

```
>> int(cos(b*t)*exp(x*sin(b*t)), x)
ans=
    exp(x*sin(b*t))*cot(b*t)
```

```
>> int(cos(b*t)*exp(x*sin(b*t)), t)
ans=
    exp(x*sin(b*t))/(b*x)
```

```
>> int(cos(b*t)*exp(x*sin(b*t)), b)
ans=
    exp(x*sin(b*t))/(t*x)
```

Функція $\text{int}(expr, var, a, b)$ повертає значення визначеного інтеграла від виразу $expr$ за змінною var на проміжку від a до b .

★ **Приклад 2.1.18.** Обчислити визначені інтеграли:

$$1. \int_0^1 \ln(x+a) dx:$$

```
>> int(log(x+a), x, 0, 1);
ans=
(log(a + 1) - 1)*(a + 1) - a*(log(a) - 1)
```

$$2. \int_0^{a/2} \sqrt{\frac{x}{a-x}} dx:$$

```
>> int(sqrt(x/(a-x)), x, 0, a/2);
ans=
(a*(pi - 2))/4
```

$$3. \int_0^1 \frac{dx}{e^x + 1} dx:$$

```
>> int(1/(exp(x)+1), x, 0, 1);
ans=
log(2) - log(exp(1) + 1) + 1
```

$$4. \int_0^4 \frac{dx}{1/7 + \sqrt{4x+7}} dx:$$

```
>> int(1/(1/7+sqrt(4*x+7)), x, 0, 4);
ans=
log(7^(1/2) + 1/7)/14 - log(23^(1/2) + 1/7)/14 -
7^(1/2)/2 + 23^(1/2)/2
```

♣ **Зауваження 2.1.2.** Для символного інтегрування у середовищі MatLab[®] можна скористатися із пакета розширення Symbolic Math Toolbox, викликати який можна натискаючи кнопку **Start** у нижньому лівому куті вікна MatLab: **Start** → **Toolboxes** → **Symbolic Math** → **MuPAD**.

2.2. Створення спеціальних додатків для розв'язування типових завдань

2.2.1. Створення додатку

У процесі розв'язування прикладних задач у системі MatLab[®] досить часто виникають ситуації, коли в алгоритмі розв'язування використовуються стандартні процедури виконання визначених обчислень. Щоб оптимізувати роботу у таких випадках у MatLab[®] є можливість написання і створення додатків пристосованих до потреб користувача. Розглянемо, наприклад, створення додатку у вигляді графічного інтерфейсу користувача додатку, за допомогою якого обчислюється мінімум довільної функції.

Створення додатку поділимо на три етапи:

- перший етап – побудова графіка аналітичного виразу у заданому інтервалі;
- другий етап – розв'язування задачі знаходження кореня та локального мінімуму;
- третій етап – створення додаткових елементів керування.

Перший етап

Створимо дві області впровадження тексту: область впровадження виразу та область границь інтервалу, а також осі для виводу графіка функції та кнопки для побудови графіка. Перед створенням зазначеного елемента треба відкрити список біжучого (зазначеного) елемента (кнопка **Property Inspector**). Створення кожного з елементів будемо проводити у такій послідовності.

1. Зазначити мишкою на панелі елементів управління потрібний елемент.
2. Зазначити у потрібному місці в області заготовки інтерфейсу (з'явиться саме потрібний елемент у рамці).
3. Точніше визначити позицію цього елемента, який створюємо, за допомогою мишки або клавішами управління курсором.
4. Змінити розміри на необхідні, потягнувши мишкою за кут рамки.

5. Встановити значення `Tag` у списку властивостей елемента. Значення `Tag` є ідентифікатором елемента. Система автоматично присвоює елементам ідентифікатори за замовчуванням, ми ж бажаємо, щоб це ім'я було більш пристосоване до дій, які виконуватиме елемент.
6. Витерти присвоєне за замовчуванням присвоєне значення властивості `String` та вписати нове, якщо є така необхідність.

Тепер послідовно утворимо дві області впровадження тексту та області впровадження осі графіка. Для першої області введення тексту властивості `Tag` присвоїмо значення `edEquation`, для другої – `edInterval`, для осей – `axMe`, одночасно очистивши значення властивості `String` кожного елемента.

Plot: `Tag` – `btnPlot`; `String` – `Plot`

На панелі натиснемо кнопку запуску додатку: \triangleright . Після збереження створеного проєкту з'явиться вікно. Впишемо у поле виразів $\sin(x)$, у поле введення меж інтервала, наприклад, $-\pi$ π та натиснемо кнопку `Plot`. Жодної реакції не буде. Тепер треба опрацювати кнопку `Plot`.

Після активування правою кнопкою мишки `Plot` відкриється підменю, в якому виберемо `View Callbacks` \rightarrow `Callback`. У відкритому `m`-файлі запишемо послідовність команд, які треба виконати для побудови графіка функції:

```
% -- Executes on button press in btnPlot.
function btnPlot_Callback(hObject, eventdata, handles) %#ok
% <INUSL,DEFNU>
% hObject handle to btnPlot (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
interval=str2num(get(handles.edInterval,'String'));
f=inline(get(handles.edEquation,'String'));
fplot(f,interval);
```

Другий етап

Створюємо кнопки `Min` та `Zero`:

| | | |
|--------|--------|---------|
| | Min | Zero |
| Tag | btnMin | btnZero |
| String | Min | Zero |

Callback → btnMin:

```
% -- Executes on button press in btnMin.
function btnMin_Callback(hObject, eventdata, handles)
% hObject handle to btnMin (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
interval =str2num(get(handles.edInterval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.edEquation,'String'));
x=fminbnd(f,x1,x2);
y=f(x);
plot(x,y,'r.','MarkerSize',15);
```

Callback → btnZero:

```
% -- Executes on button press in btnZero.
function btnZero_Callback(hObject, eventdata, handles)
% hObject handle to btnMin (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
interval =str2num(get(handles.edInterval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.edEquation,'String'));
x=fzero(f,(x1+x2)/2);
y=f(x);
plot(x,y,'g.','MarkerSize',15);
```

Третій етап

Створення елементів керування, які виводять і ховають координатну сітку, змінюють стиль виводу графіка. Для керування виводу/ховання координатної сітки створюємо елементи керування ChekBox:

| | | |
|--------|-------|-------|
| | GridX | GridY |
| Tag | cbX | cbY |
| String | GridX | GridY |

Callback → cbX:

```
% -- Executes on button press in cbX.
function cbX_Callback(hObject, eventdata, handles)
% hObject handle to cbX (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of cbX
if get(hObject,'Value')
    set(gca,'Xgrid','on')
else
    set(gca,'Xgrid','off')
end
```

Callback → cbY:

```
% -- Executes on button press in cbY.
function cbY_Callback(hObject, eventdata, handles)
% hObject handle to cbY (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of cbX
if get(hObject,'Value')
    set(gca,'Ygrid','on')
else
    set(gca,'Ygrid','off')
end
```

Створення елементів керування стилю графіка:

- Style (Tag - pmStyle, String - значення з підменю)
- Width (Tag - pmWidth, String - значення з підменю)
- Color (Tag - pmColor, String - значення з підменю)

```
% -- Executes on selection change in pmStyle.
function pmStyle_Callback(hObject, eventdata, handles)
% hObject handle to pmStyle (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns
% pmStyle
% contents as cell array contents=get(hObject,'Value') returns
% selected item from pmStyle
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'LineStyle','-');
    case 2
        set(handles.line,'LineStyle','-');
    case 3

% -- Executes on selection change in pmWidth.
function pmWidth_Callback(hObject, eventdata, handles)
% hObject handle to pmWidth (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns
% pmWidth
% contents as cell array contents=get(hObject,'Value') returns
% selected item from pmWidth
```

```
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'LineWidth',1);
    case 2
        set(handles.line,'LineWidth',2);
    case 3
        set(handles.line,'LineWidth',3);
    case 4
        set(handles.line,'LineWidth',4);
end

% -- Executes on selection change in pmColor.
function pmColor_Callback(hObject, eventdata, handles)
% hObject handle to pmColor (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns
% pmColor
% contents as cell array contents=get(hObject,'Value') returns
% selected item from pmColor
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'Color','cyan');
    case 2
        set(handles.line,'Color','red');
    case 3
        set(handles.line,'Color','green');
    case 4
        set(handles.line,'Color','blue');
    case 5
        set(handles.line,'Color','magenta');
```

```

case 6
    set(handles.line,'Color','yellow');
case 7
    set(handles.line,'Color','white');
end

```

Щоб ввести зміну стилів Callback → btnPlot:

```

% -- Executes on button press in btnPlot.
function btnPlot_Callback(hObject, eventdata, handles)
% hObject handle to btnPlot (see GCBO)
% eventdata reserved - to be defined in a future version of
% MATLAB
% handles structure with handles and user data (see GUIDATA)
cla
interval=str2num(get(handles.edInterval,'String'));
f=inline(get(handles.edEquation,'String'));
[x,y]=fplot(f,interval);
handles.line=plot(x,y,'c-');
guidata(gcbo,handles);
hold on

```

Результатом описаних вище дій одержимо застосунок, зображений на рис. 2.8.

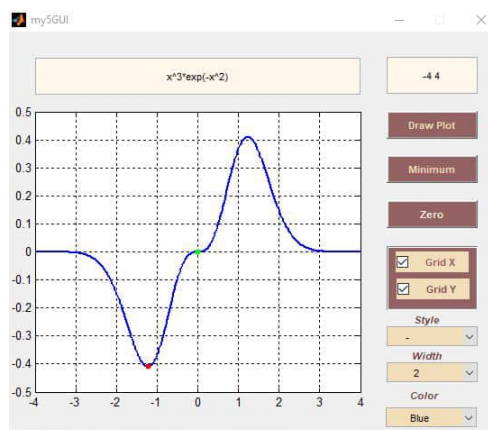


Рис. 2.8. Застосунок у MatLab[®] побудови графіка, точки мінімуму і нуля функції

Розділ 3

Графіка у MatLab[®]

У цьому розділі ознайомимося з основними функціями дво- та тривимірної графіки в середовищі MatLab[®].

3.1. Двовимірна графіка

Основною інструкцією побудови двовимірних графіків у MatLab[®] є `plot`, яка, зокрема, використовується у такому форматі:

```
plot(x,y)
plot(x,y,s)
plot(x1,y1,s1,x2,y2,s2,...,xn,yn,sn)
```

де x – аргумент функції, який задається у вигляді вектора; y – функція, яка задається аналітично або у вигляді вектора, матриці; s – вектор стилів графіка (константа, яка визначає тип ліній графіка, точок, колір лінії); x_1, x_2, \dots, x_n – аргументи функцій, графіки яких обчислюються в одній системі координат; y_1, y_2, \dots, y_n – функції, графіки яких обчислюються в одній системі координат.

3.1.1. Функція `plot(x,y)`

Функція дозволяє будувати графік при заданні функції в аналітичному вигляді, вектора чи матриці. Найчастіше використовується у таких випадках:

- вибір околу точки кореня рівняння $f(x) = 0$;
- визначення координат особливих точок функції;
- перевірка достовірності вибору інтерполяційної функції;

- якісна перевірка подання функції степеневим рядом.

★ **Приклад 3.1.1.** Для функцій $y = 3^x - 9x + 6$ визначити нулі та особливі точки.

```
>> x=0:.1:3.5;  
>> y=3.^x-9*x+6;  
>> figure(1);plot(x,y)
```

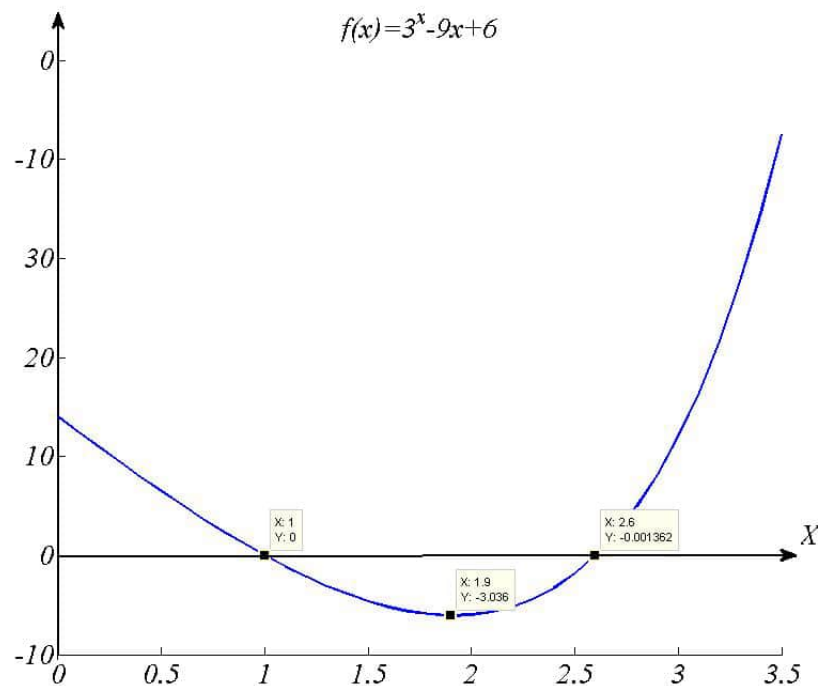


Рис. 3.1. Графік функції до прикладу 3.1.1

Використовуючи наявні інструменти у вікні **Figure**, у наближенні можемо визначити особливі точки або початкові точки для алгоритму наближеного пошуку таких (див. рис. 3.1).

3.1.2. Функція `plot(x,y,s)`

Функція відрізняється від попередньої наявністю необов'язкового параметра s , який, як зазначалось вище, визначає стиль лінії графіка функції $y = f(x)$. Константа s може набувати таких значень, які змінюють

стиль графіка:

| Тип точки | | Колір лінії | | Тип лінії | |
|------------|-------------|-------------|------------|-----------|-------------------|
| • | Точка | Y | Жовтий | - | Суцільна |
| O | Коло | M | Фіолетовий | : | Подвійний пунктир |
| × | Хрест | C | Голубий | -. | Штрих-пунктир |
| + | Плюс | R | Червоний | - | Штрихова |
| * | Зірка | G | Зелений | | |
| S | Квадрат | B | Синій | | |
| D | Ромб | W | Білий | | |
| <, >, V, ^ | Трикутник | K | Чорний | | |
| P | П'ятикутник | | | | |
| H | Шестикутник | | | | |

При заданні стилю s представляється у вигляді вектора, елементами якого є тип точки, колір і тип лінії, які вписуються в апострофи та розділяються комою. Наприклад,

```
plot(x, y, ['R', '*', '-. '])
```

Система побудує графік, лінія якого матиме червоний колір, точки графіка у вигляді зірок і штрих-пунктирною лінією.

3.1.3. Функція `plot(x1,y1,s1,x2,y2,s2,...,xn,yn,sn)`

Ця функція дає змогу обчислювати декілька графіків в одній системі координат. Позначення мають такий зміст:

- x_i – i -й масив аргументів, заданий у вигляді вектора;
- y_i – i -й масив значення функції для заданого вище масиву аргументів;
- s_i – стиль i -ї функції.

Стиль можна не задавати, за замовчуванням система визначає стилі графіки функцій. Функція $y_i(x_i)$ може задаватися аналітично. Якщо це є функція системи, то вона задається за загальними правилами з символічними змінними. Якщо це є функція користувача, тоді спочатку треба створити m -файл. Розглянемо приклад.

★ *Приклад 3.1.2.* Нехай задано табличні значення функції:

| | | | | |
|---|-----|-----|-----|-----|
| x | 1 | 2 | 3 | 4 |
| y | 6.2 | 3.5 | 1.9 | 0.6 |

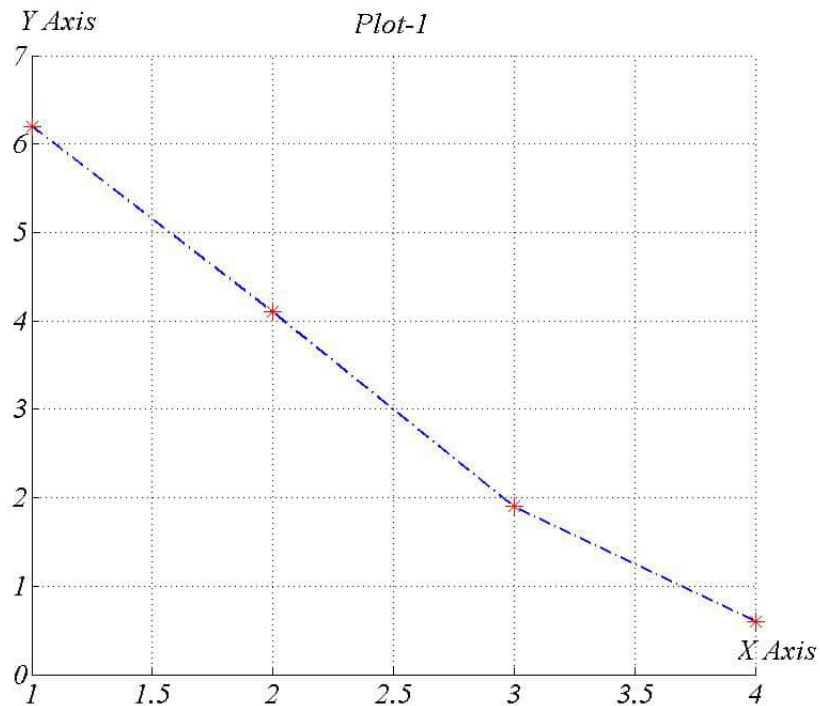


Рис. 3.2. Графік до прикладу 3.1.2

Треба побудувати графік функції та пояснення до нього.

Побудуємо графік функції за такими масивами:

```
>> clear;x=[1 2 3 4];  
y=[6.2 3.5 1.9 .6];  
figure(2); plot(x,y,'-.',x,y,'r*')
```

У разі необхідності у вікні графіка можна помістити заголовок і позначення координатних осей, а також вивести координатну сітку, замість прямокутника (який виводиться за замовчуванням) помістити графік у системі осей. З цією метою впишемо у діалоговому вікні середовища такі команди:

```
>> title('Plot-1');  
xlabel('X Axis');  
ylabel('Y Axis');  
grid on  
box off
```

Після компіляції система поверне обчислений графік, зображений на рис. 3.2. Розташування позначення осей, заголовка, товщину лінії та окремих точок можна змінити у діалоговому вікні Figure системи MatLab®.

★ **Приклад 3.1.3.** В одній системі координат побудувати графіки функцій $\cos 4x$ на проміжку $[-\pi/2, \pi]$ та $\arccos x$ на проміжку $[-1, 1]$.

Машинний код матиме вигляд

```
>> clear; x1=-pi/2:.01:pi;
x2=-1:.01:1;
y1=cos(4*x);
y2=acos(x);
figure(3); plot(x2,y2,x1,y1)
title('f(x)=cos(4x); h(x)=arccos(x)')
box off
xlabel('X');ylabel('Y');
```

Після компіляції отримаємо побудовані графіки, зображені на рис. 3.3.

Якщо функція задана в явному вигляді, то для побудови її графіка використовується функція `ezplot` у такому форматі:

```
ezplot(f,xl,xr)
```

де f – явний вигляд функції; $[xl,xr]$ – проміжок, на якому будується графік функції.

★ **Приклад 3.1.4.** Побудуємо графік функції $f(x) = 2^{-3\sin x} + 3x^2 - 4$ на проміжку $[-0.5, 3]$. У діалоговому вікні Command Window введемо такі команди:

```
>> y='2.^(-3*sin(x))+3*x.^2-4';
figure(4); ezplot(y,-.5,3)
```

Після компіляції система поверне графік, зображений на рис. 3.4.

3.1.4. Побудова графіків функцій, заданих параметричним поданням

Для обчислення графіків параметрично заданих функцій можна використати відому вже функцію середовища `plot`. З цією метою обчислимо спочатку масив значень x та y , попередньо обчисливши вектор значень параметра t , а після цього побудувати графік функції.

★ **Приклад 3.1.5.** Побудувати графік першої арки циклоїди $x = t + \cos t$, $y = 1 - \sin t$. У діалоговому вікні середовища MatLab® впровадимо такі команди:

```
>> clear; t=0:.01:2*pi;  
x=t-sin(t); y=1-cos(t);  
figure(5); plot(x,y)
```

3.1.5. Побудова графіків кусково-неперервних функцій

У побудові графіка кусково-неперервної функції зберігається принцип обчислення графіків у системі MatLab®: обчислення вузлових точок аргументу та значення функції у вузлових точках на кожному підінтервалі, на якому функція не змінює свого вигляду. Іншими словами, це побудова графіків декількох функцій, визначених на різних підінтерва-

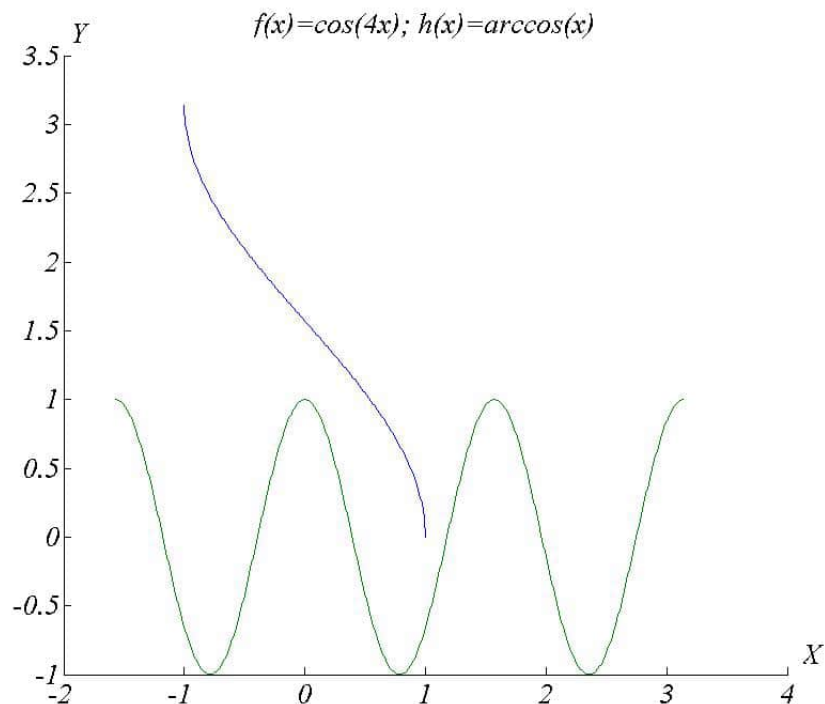


Рис. 3.3. Графік до прикладу 3.1.3

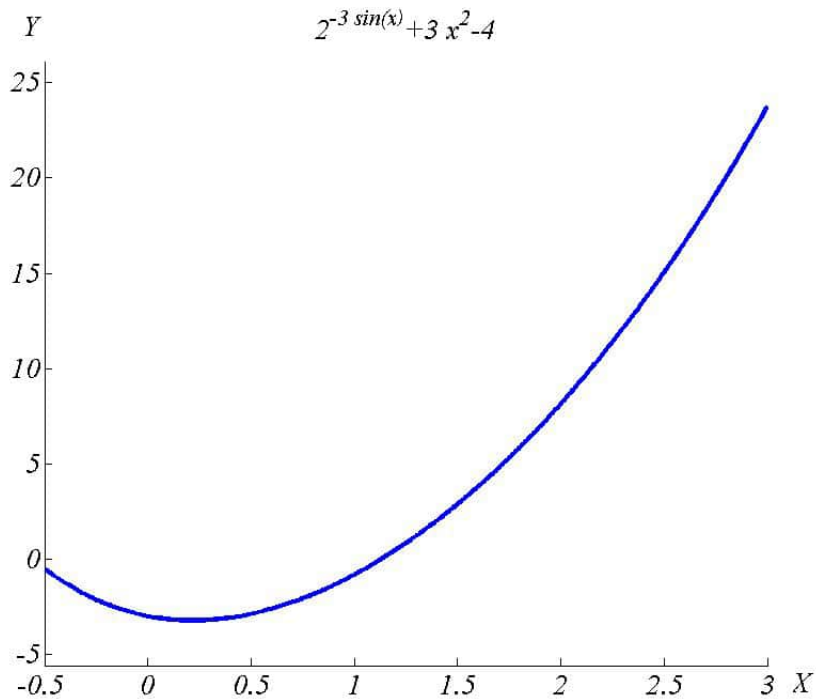
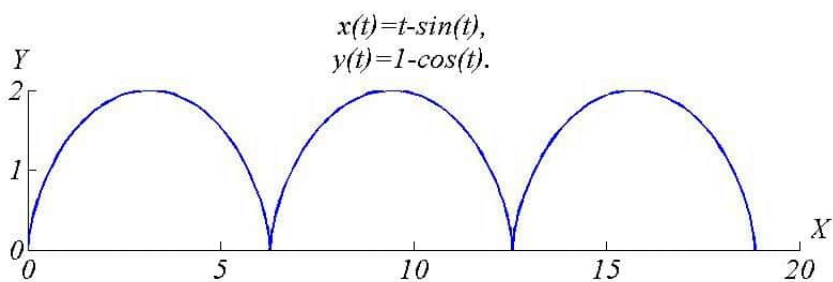
Рис. 3.4. Графік функції $f(x) = 2^{-3 \sin x} + 3x^2 - 4$ 

Рис. 3.5. Графік циклоїди до прикладу 3.1.5

лах, у одній системі координат. Продемонструємо це на такому прикладі.

★ **Приклад 3.1.6.** Побудувати графік функції

$$f(x) = \begin{cases} \pi \sin x, & -2\pi \leq x \leq -\pi; \\ \pi - |x|, & -\pi \leq x \leq \pi; \\ \pi \sin^4 x, & \pi \leq x \leq 2\pi. \end{cases}$$

Реалізація обчислення у MatLab® виглядатиме так:

```
>> clear; x1=-2*pi:pi/30:-pi; y1=pi*sin(x1);
>> x2=-pi:pi/30:pi; y2=pi-abs(x2);
>> x3=pi:pi/30:2*pi; y3=pi*sin(x3).^4;
>> x=[x1 x2 x3]; y=[y1 y2 y3];
>> figure(6) plot(x,y)
```

Якщо останню команду замінити такою,

```
>> figure(7); plot(x1,y1,'rx-',x2,y2,'gs-',x3,y3,'m>-')
```

то в результаті система поверне графік, на якому частини позначені різними кольорами та мітками точок, стосовно яких графік обчислений. Графіки в обидвох випадках мають вигляд зображених на рис. 3.6.

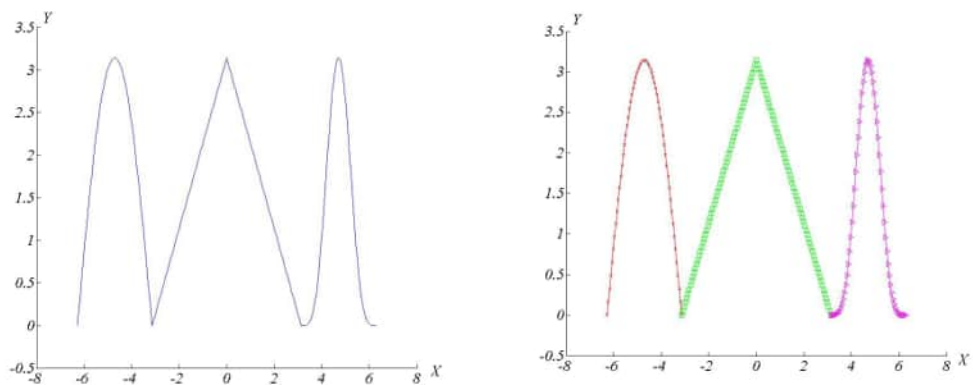


Рис. 3.6. Графік до прикладу 3.1.6

3.1.6. Побудова графіків у логарифмічному масштабі

Для побудови графіків функції у логарифмічному масштабі використовуються функції:

```
loglogx(...)
semilogx(...)
semilogy(...)
```

Функція `loglogx(...)` повертає графік у логарифмічному масштабі за двома осями координатної системи, `semilogx(...)` – стосовно осі oX , `semilogy(...)` – стосовно осі oY . Синтаксис цих функцій такий самий як у функції `plot`.

Побудова графіків у логарифмічному масштабі застосовується у таких випадках:

- дослідження стійкості систем керування частотними методами;
- дослідження якості перехідних процесів на основі логарифмічних амплітудно-частотних характеристик;
- аналіз захисту від перешкод технічних об'єктів;
- наглядність результатів при їх графічному зображенні тощо.

★ **Приклад 3.1.7.** Обчислити графік функції $f(x) = e^x$ на проміжку $[-3, 2]$ з логарифмічним масштабування стосовно осі oY . Після виконання процедури у системі MatLab[®] :

```
>> clear; x=-3:.1:2;  
y=exp(x);  
figure(8); semilogy(x,y); grid on;
```

одержимо графік, зображений на рис. 3.7.

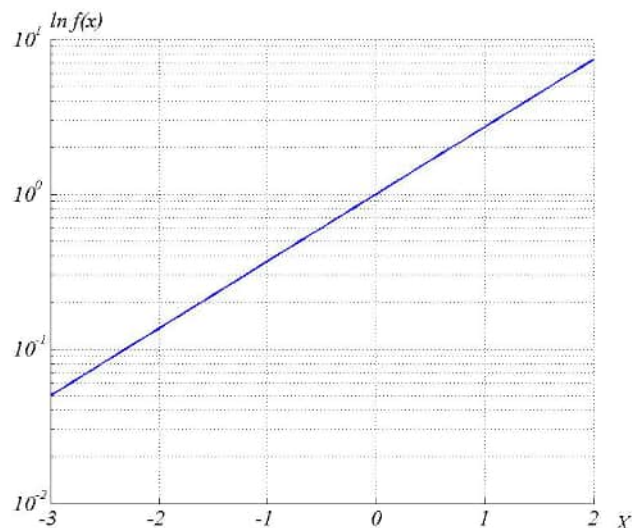


Рис. 3.7. Графік до прикладу 3.1.7

3.1.7. Побудова графіків у полярній системі координат

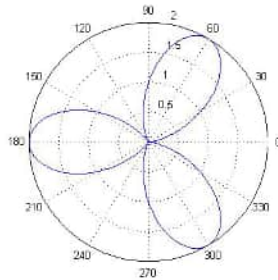


Рис. 3.8. Графік трійчлесткової троянди

Для побудови графіків у полярній системі координат використовуються інструкції у таких форматах:

```
polar(t,r)
polar(t,r,s)
```

де t – аргумент функції, заданої у полярній системі координат; r – функція, яка визначає радіус $r(t)$ залежно від полярного кута t ; s – вектор стилів, аналогічно як у випадку функції `plot`.

★ **Приклад 3.1.8.** У полярній системі координат побудувати графік трійчлесткової троянди $y = 1 - \cos(3t)$:

```
>> clear; t=0:.01:2*pi;
y=1-cos(3*t);
figure(9); polar(t,y)
```

Після обчислення система поверне графік, зображений на рис. 3.8.

3.1.8. Побудова гістограм

Гістограма – це графічний аналіз даних, який дає відповідь на питання, скільки даних потрапляє в певний інтервал розбиття проміжку, з якого взято дані. Тобто, це є частотна характеристика даних. У середовищі MatLab® для побудови гістограм слугує функція `hist`. Наприклад,

```
>> clear; data=randn(100000,1);
figure(10); hist(data)
```

Перша команда генерує 100000 нормально розподілених випадкових чисел у вигляді вектор-стовпця, а за допомогою другої будується гістограма, інтервал зміни даних поділений за замовчуванням на $n = 10$ рівних інтервалів. Якщо збільшити кількість підінтервалів, тоді процедура у MatLab® матиме вигляд:

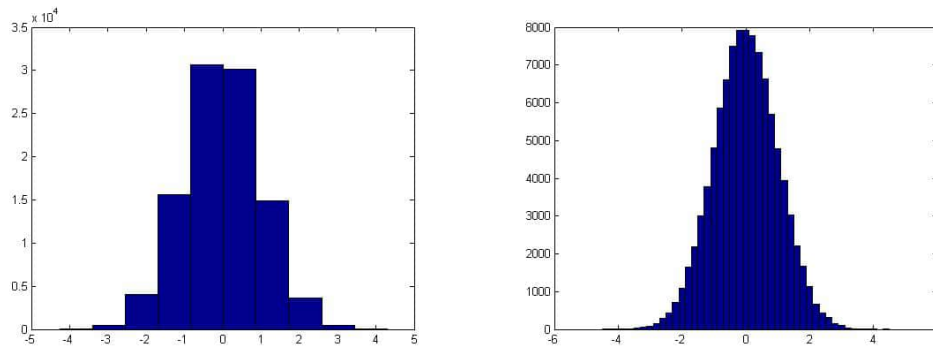


Рис. 3.9. Побудова гістограм

```
>> data=randn(100000,1);  
x=-5:.2:5;  
figure(11); hist(data,x)
```

Результати двох останніх компіляцій зображені на рис. 3.9. У другому випадку зазначені центри поділу проміжка даних, відповідно до яких проміжок зміни даних поділений рівномірно на підінтервали.

3.2. Тривимірна графіка

MatLab[®] дає змогу використовувати різні способи візуалізації функції двох змінних – побудова тривимірних графіків, їхніх ліній рівня, параметрично заданих ліній і поверхонь.

3.2.1. Тривимірні графіки функцій

Для обчислення графіка функції двох змінних необхідно:

- 1) згенерувати матриці з координатами вузлів сітки у прямокутній області визначення функції;
- 2) визначити функцію у вузлах сітки та записати отримані значення у вигляді матриці;
- 3) використати одну з функцій MatLab[®], наприклад `mesh`, для побудови графіка;
- 4) використовуючи передбачені опції нанести на графік, у разі потреби, додаткову інформацію – колір, позначення тощо.

Сітка генерується за допомогою функції `meshgrid`. Аргументами цієї команди є вектори, елементи яких відповідають сітці у прямокутній області побудови графіка. Якщо область побудови графіка функції – квадрат і крок сітки за обома напрямками однакові, то можна застосувати тільки один аргумент команди, наприклад,

```
[X Y]=meshgrid(-1:.01:1)
```

Вихідними параметрами є матриці з абсцисами й ординатами вузлів сітки на площині у заданому квадраті. Їхня структура потребує використання поелементних операцій при обчисленні матриці значень функції у вузлах сітки.

★ **Приклад 3.2.1.** Для прикладу побудуємо поверхню, яка є графіком функції

$$f(x, y) = 4 \sin(2\pi x) \cos(1.5\pi y)(1 - x^2)(y - y^2) \quad (3.2.1)$$

у прямокутнику $-1 \leq x \leq 1$, $0 \leq y \leq 1$. Послідовність у діалоговому вікні MatLab® така. Спочатку обчислимо матрицю вузлів сітки з кроком 0.01 у обидвох напрямках:

```
>> clear; [X Y]=meshgrid(-1:.01:1,0:.01:1);
```

На наступному кроці обчислимо матрицю значень функції у вузлах сітки згідно з формулою задання функції:

```
>> Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*(Y-Y.^2);
```

Тепер можна побудувати графік функції

```
>> figure(1); mesh(X,Y,Z)
```

Використовуючи функцію `mesh`, ми побудували каркасну поверхню. Якщо до обчислених значень X , Y , Z застосуємо команду `surf`, то система побудує поверхню, кожна клітина якої зафарбована відповідним кольором, залежно від відхилення точки графіка по вертикалі:

```
>> figure(2); surf(X,Y,Z)
```

Після проведених обчислень отримаємо відповідні зображення поверхні (див. рис. 3.10).

Зазначимо деякі можливості MatLab® змінювати побудову графіків у просторі. Отже, за замовчуванням команда `surf` будує поверхню, на якій нанесена каркасна сітка і кожна клітина розфарбована у відповідний колір. Нижче наведена процедура побудови поверхні, зазначивши осі та назву графіка (в припущенні, що утворена сітка вузлів $[X, Y]$ та

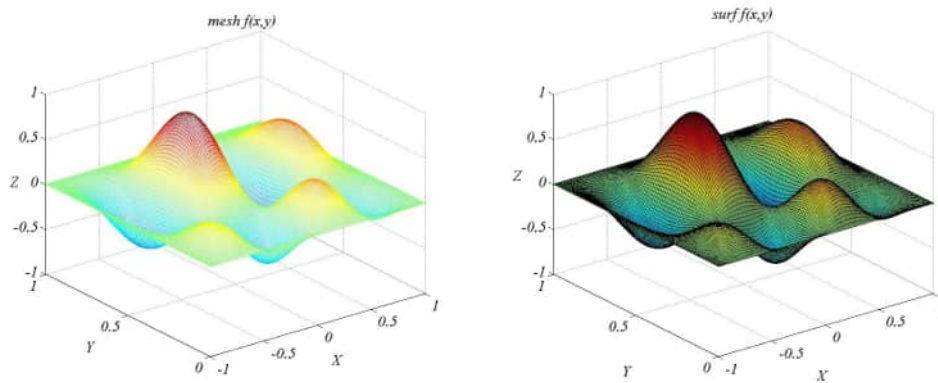


Рис. 3.10. Графіки поверхонь, побудовані за допомогою функцій `mesh` і `surf`

матриця значень функції Z на цій сітці):

```
>> figure(3); surf(X,Y,Z)
axis square
title('Default SURF')
```

Команда `shading flat` дає змогу сховати каркасну сітку:

```
>> figure(4); surf(X,Y,Z)
axis square
shading flat
title('Flat Shading')
```

Для отримання поверхні, рівномірно залитої кольором, використовується команда `shading interp`:

```
>> figure(5); surf(X,Y,Z)
axis square
shading interp
title('INTERP Shading')
```

Щоб повернутися назад до первинного графіка, використаємо послідовність команд:

```
>> figure(6); surf(X,Y,Z)
axis square
shading faceted
title('Faceted Shading')
```

Графіки після компіляції вище зазначених процедур зображені на

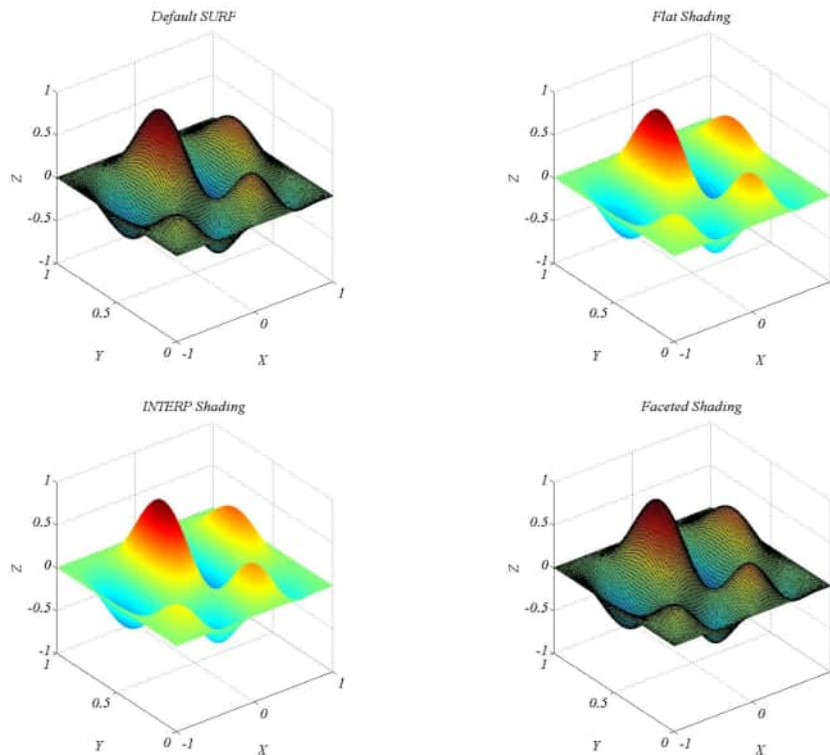


Рис. 3.11. Графіки поверхні з використанням різних опцій

рис. 3.11 (див. назву графіка).

Якщо при побудові графіка функції двох змінних застосувати команду `colorbar`,

```
>> figure(7); surf(X,Y,Z)
colorbar
title('SURF Colorbar')
```

то поряд із побудованою поверхнею система поверне кольорову шкалу, на якій за кольором можна відчитати значення функції (рис. 3.12).

Команда `surfc` обчислює поверхню та її лінії рівня (рис. 3.12):

```
>> figure(8); surfc(X,Y,Z)
colorbar
title('SURFC Colorbar')
```

MatLab® дає змогу будувати поверхню, яка складається з ліній рівня. Для цього слугує команда `contour3`, аргументи якої, за замовчуванням, такі самі як у `surf`. Застосуємо її для побудови такої поверхні у випадку розглядуваної функції. Щоб побудувати густішу сітку ліній рів-

ня, обчислимо спочатку максимальне та мінімальне значення функції (3.2.1) у прямокутнику $-1 \leq x \leq 1$, $0 \leq y \leq 1$. Опісля обчислимо вектор значень рівнів функції та побудуємо графік (рис. 3.12). Для побудови графіка використали команду `grid off`, яка відмінняє вивід на монітор координатної сітки:

```
>> zM=max(max(Z));  
zm=min(min(Z));  
lv=zm:.02:zM;  
figure(9); contour3(X,Y,Z,lv);  
grid off  
title('CONTOUR3')
```

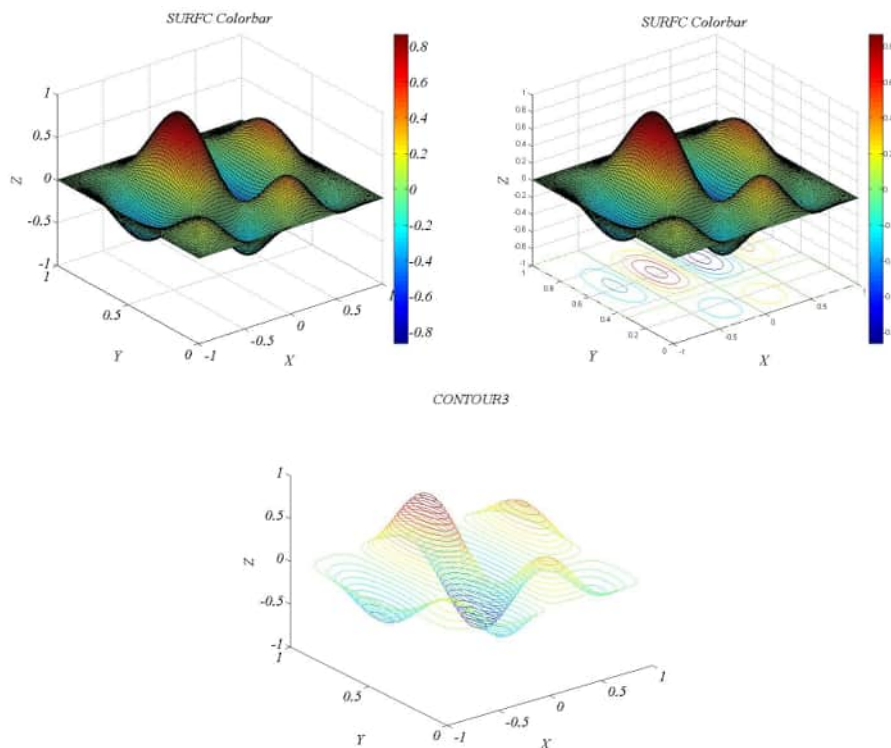


Рис. 3.12. Графіки поверхні з використанням `colorbar`, `surf`, `contour3`

3.2.2. Контурні графіки

За допомогою внутрішніх функцій `contour`, `contourf` у середовищі MatLab[®] можна побудувати контурні графіки. Застосування цих

функцій розглянемо на прикладі функції (3.2.1), тобто вважаємо, що обчислені масиви $[X, Y]$ та Z для функції (3.2.1) (див. обчислення у підрозділі 3.2.1.). Обчислення

```
>> contour(X,Y,Z)
```

поверне графік ліній рівня, показаного на рис. 3.13.

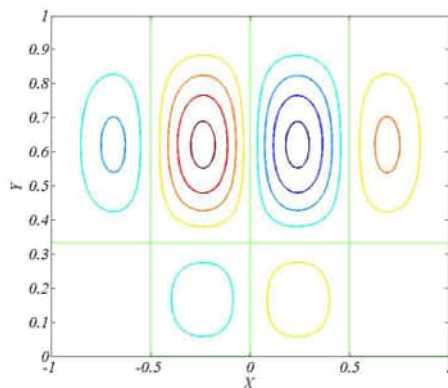


Рис. 3.13. Графік ліній рівня

Такий графік не інформативний, тому що не дає змоги відчитати значення функції на лініях рівня. Використання внутрішньої інструкції системи `colorbar` також не багато допоможе, точні значення функції не вдасться відчитати. У середовищі MatLab® командою `clabel` можна вивести значення рівня кожної лінії контурного графіка функції. Функція `clabel` вживається з двома аргументами: матрицею, що містить інформацію про лінії рівня та вказівником на графік, на якому треба нанести розмітку. Не вдаючись при першому ознайомленні у подробиці, користувачу немає потреби створювати параметри функції `clabel`, тому що функція `contour`, яка скомпільована з двома вихідними параметрами, не тільки побудує лінії рівня, а й згенерує необхідні параметри для `clabel`. Отож, скомпілюємо `contour` з двома вихідними параметрами (`ContMtr` – матриця, в якій міститься інформація про лінії рівня, а вектор `h` – вектор показників):

```
>> figure(2); [ContMtr,h]= contour(X,Y,Z);
clabel(ContMtr,h);
grid on
title('CLABEL')
```

Зауважимо, що перший командний рядок активного вікна **Command Window** системи MatLab® завершили символом `;` для того, щоб на монітор не виводилися значення матриці. Пригадаємо, щоб скомпілювати процедуру (послідовність команд у цілому, на відміну компіляції кожного окремого командного рядка), необхідно перехід до наступного рядка виконувати за допомогою комбінації клавіш **Shift + Enter**. Крім то-

го, помістили на графіку координатну сітку та назву графіка (див. Рис. 3.14).

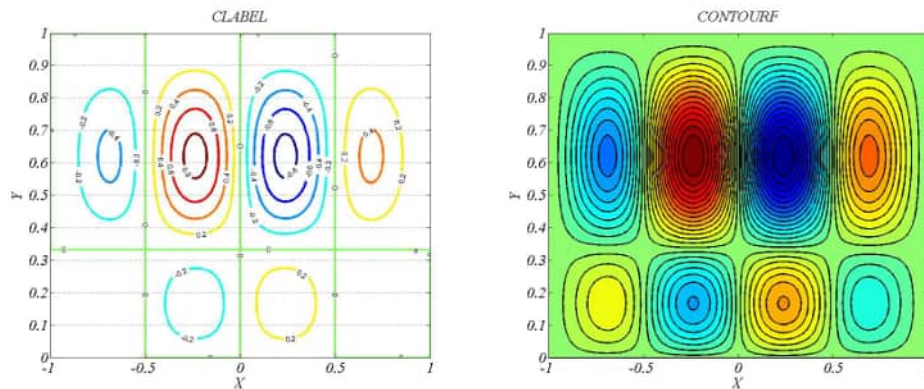


Рис. 3.14. Графік ліній рівня: `clabel`, `contourf`

Додатковим аргументом функції `contour`, так як і функції `contour3`, розглянутій вище, може бути кількість ліній рівня або вектор, який містить значення функції, для яких необхідно побудувати лінії рівня.

Наочну інформацію про зміну значення функції дає кольорова "заливка", тобто неперервне кольорове розфарбування контурного графіка функції залежно від значень функції. Для цього у MatLab[®] слугує функція `contourf`, використання якої не відрізняється від розглянутої функції `contour`. Наприклад, компіляція процедури

```
>> figure(3); contourf(X,Y,Z,30);
title('CONTOURF')
```

обчислить графік із 30 лініями рівня (див. рис. 3.14). Нагадаємо, що у всіх наведених прикладах використані дані для функції (3.2.1).

3.2.3. Оформлення графіка

Простим, проте ефективним способом зміни кольорової гами графіка є встановлення кольорової палітри за допомогою функції `colormap`. Нижче наведений приклад демонструє, як приготувати графік функції для друку на монохромній друкарці, використовуючи палітру `gray` (тут і надалі використовуються обчислені дані $[X, Y]$, Z для функції (3.2.1) з підрозділу 3.2.1.).

```
>> figure(1); surfc(X,Y,Z);
colorbar
colormap(gray)
title('Plot z(x,y)')
xlabel('X'); ylabel('Y'); zlabel('Z');
```

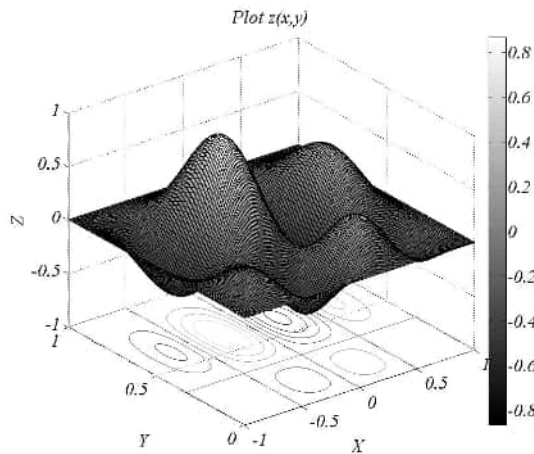


Рис. 3.15. Графік поверхні з використанням палітри gray

Щоб додати до графіка заголовок, а також назви осей, можна скористатися з інструкцій, відповідно:

```
title('Title Name')
xlabel('X Label Name')
ylabel('Y Label Name')
zlabel('Z Label Name')
```

Призначення цих функцій зрозуміле з назви. Наприклад, потрібно помістити у заголовку графіка функції (3.2.1) вигляд самої функції (використовуємо масиви, обчислені у підрозділі 3.2.1.). Процедура обчислення матиме вигляд:

```
>> figure(2); surf(X,Y,Z);
title(' \it {f}( \it {x}, \it {y})= 4\sin(2 \pi \it {x})
      \cos(1.5 \pi \it {y})(1- \it {x}^2) ( \it {y}- \it {y}^2) ');
```

Система поверне графік поверхні, зображений на рис. 3.15. Зауважимо, що зміна палітри графіка приводить до зміни палітри всього вікна. Для того, щоб повернутись до налаштувань за замовчуванням, треба використати інструкцію:

```
colormap('default')
```

Кольорові палітри, які доступні у середовищі MatLab® наведені у табл. 3.1.

Табл. 3.1. Палітра кольорів у середовищі MatLab®

| Палітра | Зміна кольору |
|-----------|--|
| autum | Плавна зміна: червоний-оранжевий-жовтий |
| bone | Подібна на gray, проте з легким відтінком синього |
| colorcube | Кожний колір змінюється від темного до світлого |
| ccol | Відтінок голубого та пурпурового кольорів |
| copper | Відтінки мідного кольору |
| flag | Циклічна зміна: червоний-білий-синій-чорний |
| gray | Відтінки сірого |
| hot | Плавна зміна: чорний-червоний-оранжевий-жовтий-білий |
| hsv | Плавна зміна кольорів спектра |
| jet | Плавна зміна: синій-голубий-зелений-жовтий-червоний |
| pink | Так як gray з легким відтінком коричневого |
| prism | Циклічна зміна: червоний-оранжевий-жовтий-зелений-синій-фіолетовий |
| spring | Відтінки пурпурового та жовтого |
| summer | Відтінки зеленого та жовтого |
| wga | Палітра Windows ps 16 кольорів |
| white | Білий колір |
| winter | Відтінки синього та зеленого |

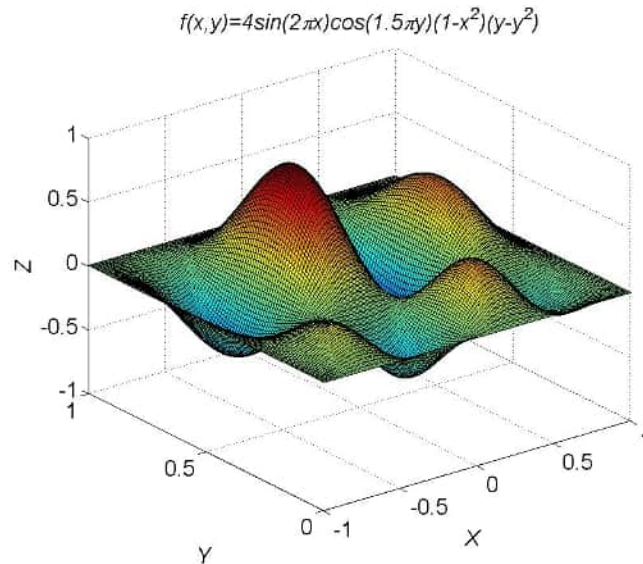
```
xlabel(' \it {X} ');
ylabel(' \it {Y} ');
zlabel(' \it {Z} ');
```

Як видно з останнього наведеного прикладу, у назві графіка, а також у позначеннях осей можна вживати TeX-символи. Результат виконання останньої процедури зображено на рис. 3.16.

3.2.4. Побудова параметрично заданих поверхонь і просторових кривих

MatLab® дає змогу обчислювати графіки тривимірних параметричних кривих, заданих рівнянням

$$x = x(t), y = y(t), z = z(t), \quad t \in [\alpha, \beta] \quad (3.2.2)$$

Рис. 3.16. Використання \TeX -символів у описі графіка

і поверхонь

$$x = x(u, v), \quad y = y(u, v), \quad z = z(u, v), \quad u \in [a, b], \quad v \in [c, d]. \quad (3.2.3)$$

Для обчислення графіків параметрично заданих функцій використовується раніше описана функція

`plot3(X,Y,Z,s)`

де X , Y , Z – масиви, які обчислюють за формулами (3.2.2) або (3.2.3), s – вектор параметрів та опцій графіка.

★ **Приклад 3.2.2.** Побудувати графік параметрично заданої просторової кривої

$$x = e^{-\frac{|t-50|}{50}} \sin t, \quad y = e^{-\frac{|t-50|}{50}} \cos t, \quad z = t, \quad t \in [0, 100] \quad (3.2.4)$$

Отож, перш за все згенеруємо сітку вузлів для параметра t :

```
>> clear; t=0:.02:100;
```

Далі обчислимо масиви значень x , y , z за формулами (3.2.4):

```
>> x=exp(abs(t-50)/50).*sin(t);
```

```
y=exp(abs(t-50)/50).*cos(t);
```

```
z=t;
```

Тепер побудуємо графік просторової кривої, заданої параметричним

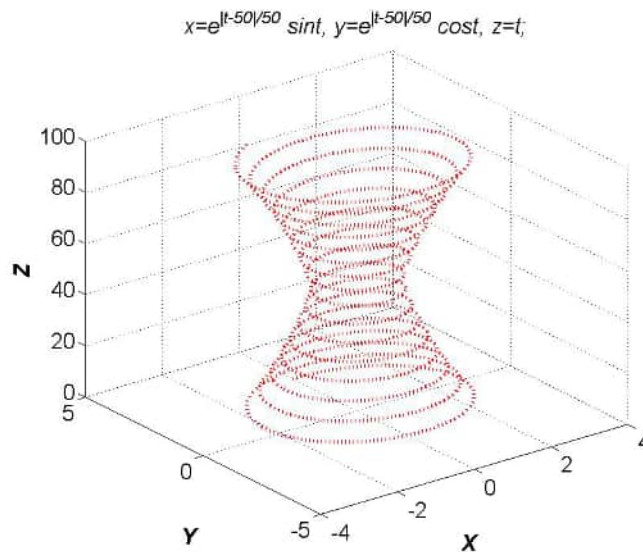


Рис. 3.17. Графік параметрично заданої просторової кривої

рівнянням, зобразивши координатну сітку, назву графіка та мітки координатних осей:

```
>> plot3(x,y,z,'r:');
grid on
title(' \it {x}= \it {e}^{\|- \it {t}-50|/50} \sin \it {t},
      \it {y}= \it {e}^{\|- \it {t}-50|/50} \cos \it {t},
      \it {z}= \it {t}');
xlabel(' \it {\bf {x}}')
ylabel(' \it {\bf {y}}')
zlabel(' \it {\bf {z}}')
```

У підсумку система поверне просторову криву, зображену на рис. 3.17.

★ **Приклад 3.2.3.** Побудувати графік поверхні, заданої своїм параметричним поданням:

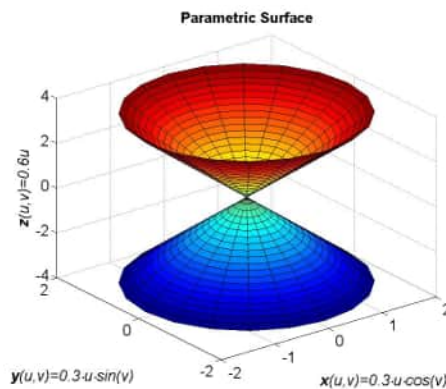
$$x = 0.3u \cos v, \quad y = 0.3u \sin v, \quad z = 0.6u, \quad u, v \in [-2\pi, 2\pi]. \quad (3.2.5)$$

Згенеруємо спочатку сітку вузлів параметрів u та v , причому, важливо те, що для u це має бути масив у вигляді стовпця, а для v – рядка:

```
>> clear; u=[-2*pi:.1*pi:2*pi]';
v=[-2*pi:.1*pi:2*pi];
```

Тепер можемо згенерувати двовимірну сітку значень x та y за форму-

лами (3.2.5) (звернемо увагу на *зовнішній добуток* – зірочка без крапки, який використовуємо для обчислення x і y):



```
>> x=.3*u*cos(v);
```

```
y=.3*u*sin(v);
```

Матриця z має бути тих самих розмірів, що і матриці x , y . Проте у формулі (3.2.5) обчислення z виражена явна залежність тільки від u . Для того, щоб правильно обчислити цю матрицю, вектор u домножимо *зовнішнім добутком* на квадратну матрицю розмірності вектора v , елементами якої є одиниці:

Рис. 3.18. Графік параметричної поверхні

```
>> z=.6*u*ones(size(v));
```

Після чого можемо побудувати поверхню, яка зображена на рис. 3.18:

```
>> figure(2); surf(x,y,z)
```

Крім того, на графік нанесемо позначення осей координат і назву графіка:

```
>> title(' \bf {Parametric Surface.}');
```

```
xlabel(' \it { \bf {x}( \it {u}, \it {v})=0,3 \it {u}cos( \it {v})}')
```

```
ylabel(' \it { \bf {y}( \it {u}, \it {v})=0,3 \it {u}sin( \it {v})}')
```

```
zlabel(' \it { \bf {z}( \it {u}, \it {v})=0,6 \it {u}}')
```

3.2.5. Побудова освітленої поверхні

Припустимо, що поверхня графіка виготовлена з матеріалу із різними фізичними та хімічними властивостями, а отже, має різні властивості відбивання та поглинання світла. Крім того, щоб у такий спосіб передати інформацію про властивості поверхні, джерелом світла можна керувати – змінювати інтенсивність, точку освітлення тощо. У середовищі MatLab® для цих цілей слугує функція

`surf(X,Y,Z)`

де X, Y, Z – масиви вузлів, такі самі як у випадку функції `surf`.

★ **Приклад 3.2.4.** Побудувати освітлену поверхню для функції (3.2.1).

Використовуючи функції `surf`, можна визначати палітри кольорів у системах `copper`, `bone`, `gray`, `pink`. У згаданих системах інтенсивність кольору змінюється лінійно. Для отримання плавної зміни відтінків використовується команда `shading interp`. Отже, процедура побудови зображення виглядатиме так:

```
>> clear; [X Y]=meshgrid(-1:.01:1,0:.01:1);
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*(Y-Y^2);
figure(1); surf(X,Y,Z)
colormap('copper');
shading interp
title(' \it{\bf{f}} (\it{x}, \it{y})= 4sin(2\pi \it{x})
      cos(1.5\pi \it{y}) (1- \it{x}^2)(\it{y}- \it{y}^2)')
xlabel(' \it{\bf{x}}');
ylabel(' \it{\bf{y}}');
zlabel(' \it{\bf{z}}');
```

У підсумку одержимо графік, зображений на рис. 3.19.

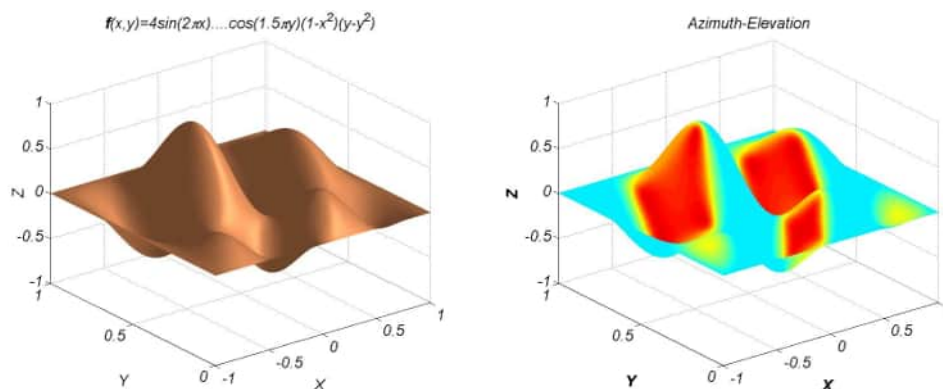


Рис. 3.19. Освітлені графіки

За замовчуванням джерело світла має азимут на 45° більший від спостерігача і такий самий кут піднесення. Додатковим, четвертим, аргументом функції `surf` може слугувати вектор із двох елементів – азимута

та кута піднесення джерела світла. Наприклад, змінимо азимут на -90° в стосунку до спостерігача, а кут спостереження визначимо рівним нулю:

```
>> [Az,E1]=view;
figure(2); surf1(X,Y,Z,[Az-90,0]);
shading interp
```

3.2.6. Анімовані графіки

Вивчаючи рух точки на площині чи у просторі, важливо, часом, мати не тільки траєкторію руху, але й спостереження як відбувається рух. MatLab® дає змогу побудувати анімований графік, на якому кружок, що позначає точку, рухається на площині чи у просторі залишаючи за собою слід у вигляді кривої – траєкторії руху. Для побудови анімованого графіка використовуються інструкції MatLab®, основний формат яких такий:

```
comet(X,Y)
comet3(X,Y,Z)
```

де X , Y , Z – масиви вузлів, які у дво- та тривимірному випадку обчислюються так само, як при побудові дво- чи тривимірної графіки функцій, заданих своїм параметричним поданням.

★ **Приклад 3.2.5.** Побудувати траєкторію руху точки протягом 10 секунд, координати якої змінюються за законом

$$x(t) = \frac{\sin t}{1+t}, \quad y(t) = \frac{\cos t}{1+t}. \quad (3.2.6)$$

Алгоритм побудови процедури обчислення такий самий як при побудові параметрично заданої функції:

```
>> clear; t=[0:.01:10];
x=sin(t)./(1+t); y=cos(t)./(1+t);
figure(3); comet(x,y)
```

3.3. Робота з декількома графіками

У всіх прикладах, розглянутих вище, графіки будувались у спеціальних вікнах із заголовком Figure. Будуючи наступний графік, попередній

зникав у цьому вікні, а на його місці будувався інший графік у цьому ж вікні. MatLab[®] дає змогу працювати одночасно з кількома графіками:

- вивід кожного графіка в окреме вікно;
- вивід кількох графіків в одному вікні(в одній координатній системі);
- вивід в одному вікні декілька графіків, кожний у своїй координатній системі.

3.3.1. Вивід графіків в окремі вікна

Команда `figure` слугує для створення пустого графічного вікна та відображення його на моніторі. Вікно стає активним, тобто всі графічні об'єкти будуть відображатися у цьому вікні. Для того, щоб черговий графік був побудований у новому вікні, треба у новій процедурі вжити команду `figure`. Наприклад, послідовність команд

```
>> clear; [X Y]=meshgrid(-1:.01:1,0:.01:1);  
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*(Y-Y.^2);  
figure(1); mesh(X,Y,Z);  
figure(2); surf1(X,Y,Z)
```

приводить до побудови двох графіків у двох окремих вікнах: у вікні `Figure1` – каркасну поверхню, у `Figure2` – освітлену поверхню. Вікно `Figure2` наразі є активним, позаяк графік був створений останнім, тому послідовність наступних команд стосується освітленої поверхні. Наприклад, послідовність команд

```
>> colormap('copper')  
shading interp
```

приводить до зміни тільки останнього побудованого графіка та висвітлення цих змін у вікні `Figure2`. Якщо маємо намір зробити зміни у побудові графіка, який відображений у вікні `Figure1`, потрібно зробити це вікно активним (вибрати його мишкою та притиснути кнопку мишки), після цього повернутися у `Command Window`, ввести необхідні зміни та скопіювати процедуру. Для того, щоб очистити активне вікно типу `Figure`, вживається функція

```
clf
```

(скорочення від `clear figure`), а для того, щоб прибрати тільки графік, залишивши без змін осі, їх позначення, назву графіка, належить використати функцію

```
cla
```

(від clear axes).

Згаданим вище способом можна одержати скільки завгодно графічних вікон і відобразити у них графіки різних функцій або інші графічні об'єкти. Однак для зміни деякого графіка треба знайти відповідне вікно на моніторі й активувати його натисканням правої кнопки мишки. Такий спосіб не зовсім зручний. У середовищі MatLab® передбачено більш універсальний і зручний спосіб роботи у багатовіконному режимі. Для створення кожного нового графічного вікна за допомогою `figure` потрібно викликати його з вихідним аргументом. Цей аргумент у MatLab® називається вказівником на графічне вікно. Значенням вихідного аргументу є число, яке збігається з номером графічного вікна. Для того, щоб зробити графічне вікно активним, треба викликати `figure`, зазначивши як вхідний аргумент показник необхідного графічного вікна. Розглянемо приклад.

★ **Приклад 3.3.1.** Створити два графічні вікна, побудувати у них графіки функцій $f(x) = \sin x$ та $g(x) = \ln x$, після чого графіки належно оформити: вивести на графіках назву, позначення осей і нанести координатну сітку на другий із них. Послідовність команд така:

```
>> clear; sinGr=figure; lnGr=figure;
x=.1:.05:10;
f=sin(x); g=log(x);
figure(sinGr); plot(x,f);
figure(lnGr); plot(x,g);
figure(sinGr);
title(' \it{\bf {f}} (\it{x}) = sin(\it{x})');
xlabel(' \it{\bf {x}}'); ylabel(' \it{\bf {f(x)}}');
figure(lnGr);
title(' \it{\bf {g}} (\it{x}) = ln(\it{x})');
xlabel(' \it{\bf {x}}'); xlabel(' \it{\bf {g(x)}}');
grid on;
```

Після компіляції система поверне графіки, зображені на рис. 3.20. Для того, щоб очистити графічне вікно з міткою `lnGr`, треба виконати команду

```
>> clf(lnGr)
```

Щоб видалити графік із першого вікна, необхідно виконати команду

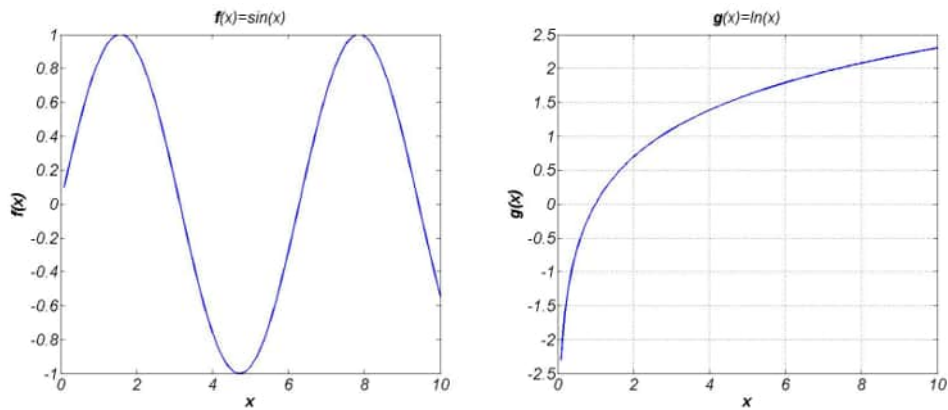


Рис. 3.20. Графіки у різних вікнах (до прикладу 3.3.1)

```
>> cla(sinGr)
```

3.3.2. Побудова декількох графіків в одній координатній системі

Зобразити декілька графіків функцій однієї змінної в одній координатній системі можна за допомогою функцій `plot`, `plotyy`, `semilogx`, `semilogy`, `loglog`. Ці інструкції дають змогу будувати декілька графіків в одній координатній системі, задаючи аргументи цих команд, наприклад, у такому форматі:

```
plot(x,f,x,g)
```

Однак для побудови тривимірних графіків або графіків різних типів, немає можливості об'єднати їх в одній координатній системі. Для такого об'єднання слугує команда

```
hold on
```

яку потрібно виконати перед побудовою наступного графіка, який необхідно обчислити у тій самій координатній системі. У наступному прикладі в одній координатній системі будується перетин площини та конуса, заданого у параметричному вигляді. Результат виконання процедури зображено на рис. 3.21.

```
>> clear; u=[-3*pi:.1*pi:3*pi]'; v=[-3*pi:.1*pi:3*pi];
x=.3*u*cos(v); y=.3*u*sin(v); z=.6*u*ones(size(v));
figure(3); surf(x,y,z);
```

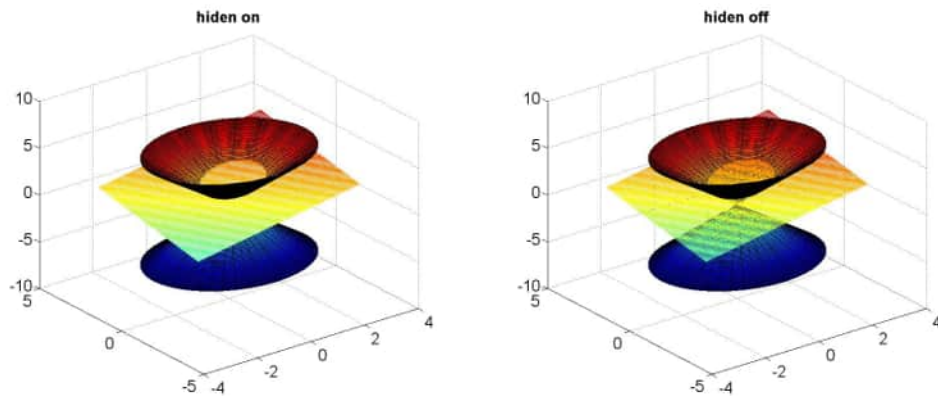


Рис. 3.21. Два графіки в одній координатній системі: `hidden on` та `hidden off`

```
[X,Y]=meshgrid(-3:.1:3); Z=.5*X+.4*Y+2;
hold on
mesh(X,Y,Z);
hidden on
```

Для побудови площини `mesh(X,Y,Z)` та приєднання її графіка до графіка конуса `hold on` використали команду `hidden on` – закрити від спостереження частину конуса, яка є під площиною. Якщо тепер виконати послідовність команд

```
>> surf(x,y,z);
hold on
mesh(X,Y,Z);
hidden off
```

то площина зображається сіткою, крізь яку можна спостерігати конус (рис. 3.21).

Команда `hold on` поширюється на всі графіки, які будуть обчислюватись після її виконання. Для розташування графіків на нових осях потрібно відмінити її такою командою

```
hold off
```

Команду `hold on` можна застосовувати і для графіків функцій однієї змінної. Наприклад, виконання команди

```
plot(x,f,t,g)
```

еквівалентне послідовності

```
plot(x,f);
hold on;
plot(t,g);
hold off;
```

3.3.3. Декілька графіків в одному графічному вікні

MatLab[®] дає змогу розмістити в одному графічному вікні декілька координатних систем і побудувати у кожному з них графіки. Найпростіший спосіб полягає у розбитті на визначену кількість частин по горизонталі та вертикалі у вигляді матриці, використовуючи функцію:

```
subplot(i,j,n)
```

де i та j – кількість підграфіків по вертикалі (кількість рядків) і горизонталі (кількість стовпців); n – номер підграфіка, який потрібно активувати. Нумерація ведеться, починаючи з лівого верхнього кута, по рядках. Наприклад, процедура:

```
>> clear; syms n
for n=1:6
subplot(2,3,n)
end
```

створить у вікні графіки шість координатних систем, розташованих у двох рядках і трьох стовпчиках. А наступна:

```
>> x=[-1:.01:pi]
y=x.^3+sin(x); subplot(2,3,1); plot(x,y);
title(' \it {f}( \it {x}) = \it {x}^3+sin \it {x} ')
y2=sin(x.^2); subplot(2,3,4); plot(x,y2);
title(' \it {g}( \it {x}) = sin( \it {x}^2) ')
y3=exp(-x.^2); subplot(2,3,2); plot(x,y3);
title(' \it { \bf {h(x)=e^{-x^2}}} ')
y4=1./(1+x.^2); subplot(2,3,3); plot(x,y4);
title(' \it { \bf {r(x)=1/(1+x^2)}} ')
y5=sin(x).^2; subplot(2,3,5); plot(x,y5);
title(' \it { \bf {u(x)=sin^2(x)}} ')
y6=cos(x.^3).*sin(x); subplot(2,3,6); plot(x,y6);
title(' \it { \bf {v(x)=cos(x^3)sin(x)}} ')
```

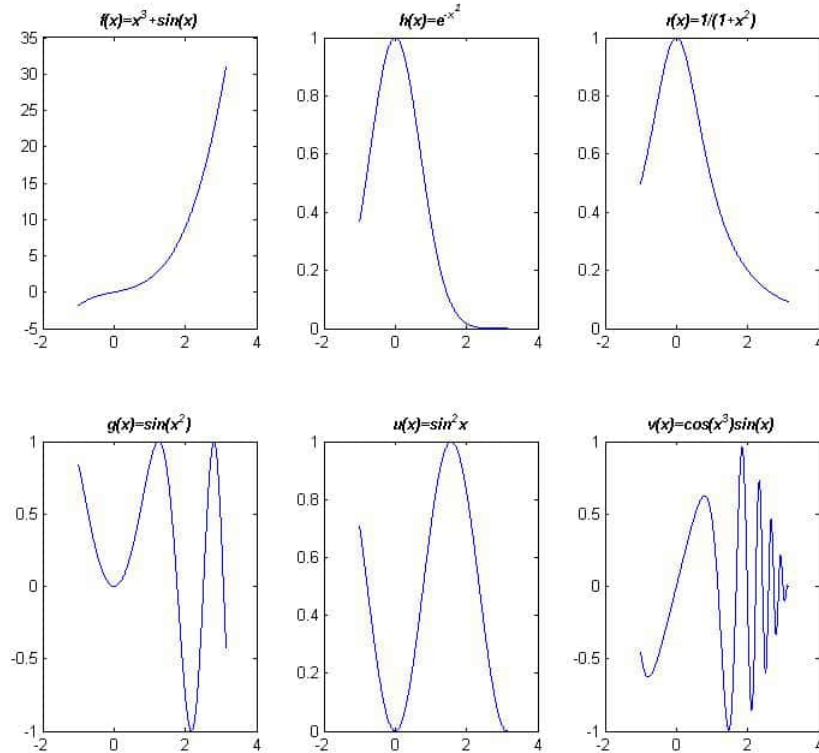


Рис. 3.22. Кілька графіків в одному вікні

побудує графіки у всіх координатних системах (рис. 3.22).

У наступному прикладі побудуємо можливі шість видів графіків функції (3.2.1) та розмістимо їх в одному вікні в різних координатних системах.

★ **Приклад 3.3.2.** Використаємо вікно, створене в попередньому прикладі (вікно, зображене на рис. 3.22).

```
>> clear; [X Y]=meshgrid(-1:.01:1,0:.01:1);
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*(Y-Y^2);
subplot(2,3,1); mesh(X,Y,Z); title(' \bf {MESH} ');
subplot(2,3,2); surf(X,Y,Z); title(' \bf {SURF} ');
subplot(2,3,3); meshc(X,Y,Z); title(' \bf {MESHc} ');
subplot(2,3,4); surfc(X,Y,Z); title(' \bf {SURFc} ');
```

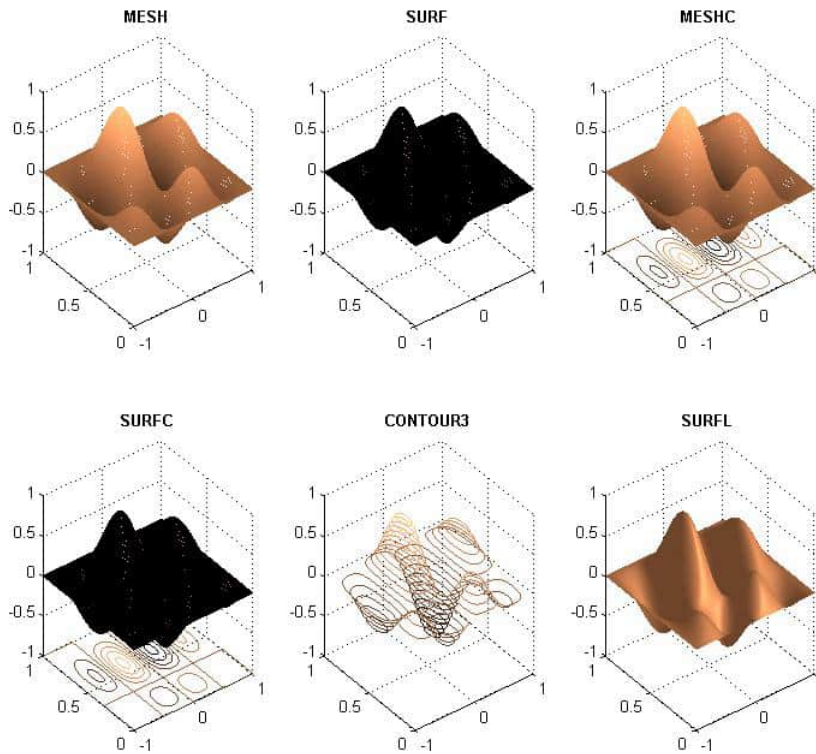


Рис. 3.23. Підграфіки (до прикладу 3.3.2)

```
subplot(2,3,5); contour3(X,Y,Z,20); title(' \bf {CONTOUR3}');
subplot(2,3,6); surf1(X,Y,Z); shading interp;
title(' \bf {SURFL}'); colormap(copper);
```

У результаті обчислення система поверне вікно, зображене на рис. 3.23.

3.4. Завдання для самостійного опрацювання

◆ *Завдання 3.4.1.* Самостійно побудувати графіки функції (3.2.1), використовуючи кольорові палітри з таблиці 3.1, та способи його побудови.

◆ *Завдання 3.4.2.* Побудувати графік функції

$$f(x, y) = -e^{-y^2} \cos(3\pi x)x(1-x)y, \quad x \in [0, 1], y \in [-2, 0], \quad (3.4.1)$$

використовуючи різні способи побудови графіка, кольори, назву, позначення осей.

◆ *Завдання 3.4.3.* Побудувати прозору каркасну поверхню еліпсоїда, заданого рівнянням

$$\begin{cases} x(u, v) = \cos u \cos v, \\ y(u, v) = 0,7 \cos u \sin v, \\ z(u, v) = 0,8 \sin u, \end{cases} \quad (3.4.2)$$

якщо $u, v \in [-2\pi, 2\pi]$.

◆ *Завдання 3.4.4.* Побудувати освітлені графіки для функції (3.2.1), використовуючи різні системи кольорової гами та точок освітлення.

◆ *Завдання 3.4.5.* Побудувати траєкторію руху фіксованої точки на одиничному колі, що рухається по прямій (циклоїду). Циклоїда визначається таким параметричним поданням:

$$x(t) = t - \sin t; \quad y(t) = 1 - \cos t. \quad (3.4.3)$$

◆ *Завдання 3.4.6.* Використовуючи функцію `comet3` для побудови траєкторії руху точки у просторі, що описується рівняннями (3.2.4).

◆ *Завдання 3.4.7.* Побудувати графіки функції однієї змінної на відріжку $[0.01, 2\pi]$

$$f(x) = \frac{\sin x}{x}, \quad g(x) = e^{-x} \cos x.$$

Зобразити ці графіки різними способами

- в окремих графічних вікнах;
- в одному вікні в одній системі координат;
- в одному вікні в різних системах координат.

Надати заголовки графікам, розмістити позначення осей, використати різні стилі, кольори, тип маркерів графіків функцій, нанести координатну сітку.

Побудувати частину графіка для від'ємних значень функції синім

кольором, для додатних – червоним. Прийняти до уваги, що насправді відображається залежність одного вектора від іншого. Отож, для пошуку індексів потрібних елементів вектора зі значеннями функції та індексацію вектором для виділення потрібних компонент можна використати функцію `find`.

◆ *Завдання 3.4.8.* У квадраті $x \in [-1, 1]$, $y \in [-1, 1]$ візуалізувати функцію

$$z(x, y) = (\sin(x^2) + \cos(y^2))^{xy}.$$

Вивести графіки різними способами:

- каркасною поверхнею;
- каркасна поверхня "залита" кольором;
- промаркованими лініями рівня (самостійно вибрати вектор маркування ліній рівня);
- освітленою поверхнею.

Розташувати графіки в окремих графічних вікнах та в одному вікні з відповідною кількістю координатних систем. Подати каркасну та освітлену поверхню з декількох точок спостереження.

На тривимірних графіках відзначити точки локальних екстремумів, при цьому використати інформацію, що значення функції у вузлах сітки зберігаються у матриці. Визначити максимальне значення функції за допомогою `max`. За допомогою функції `find` визначити індекси цих елементів матриці. І накінець, за допомогою `plot3` розташувати маркери в точках тривимірного простору на графіку.

◆ *Завдання 3.4.9.* Зобразити поверхні, які задають функції корисності, вивести криві байдужості:

- 1) $u(x, y) = xy$; 2) $u(x, y) = (x + 1)(y + 1)$;
- 3) $u(x, y) = 2\sqrt{x} + 4\sqrt{y}$; 4) $u(x, y) = \max\{2x + y, 2y + x\}$;
- 5) $u(x, y) = xy + x + y$; 6) $u(x, y) = \min\{2x + y, 2y + x\}$;
- 7) $u(x, y) = \frac{xy}{x + y}$; 8) $u(x, y) = x + \min\{x, y\}$;
- 9) $u(x, y) = x^{1/3}y^{2/3}$; 10) $u(x, y) = \ln(1 + x) + \ln(1 + 2y)$.

◆ *Завдання 3.4.10.* Побудувати векторне поле градієнта функції. Візуалізувати тривимірне векторне поле на поверхнях:

- гіперболоїді;

- параболоїді;
- заданої параметрично рівняннями:

$$x(u, v) = \cos u \cos v, \quad y(u, v) = \sin u \sin v, \quad z(u, v) = u \cdot v, \quad u, v \in [0, 3].$$

◆ *Завдання 3.4.11.* Побудувати векторні поля:

- поле напрямків відбитого світла, яке падає паралельним пучком на поверхню;
- поле напрямків заломаного світла, яке падає паралельним пучком на поверхню;
- поле напрямків відбитого світла, направленою точковим джерелом.

Розглянути зміну поля напрямків в залежності від розташування поверхні та джерела.

При побудові поля напрямків підібрати масштабний множник та кількість точок зачеплення векторів на поверхні для отримання найкращого зображення поля. Поле нормалей до поверхні, необхідне для визначення напрямків променів відбитого та заломленого вектора, обчислити за допомогою команди `surfnorm`.

Розділ 4

m-файли

У попередніх розділах ми розглянули досить прості приклади, для виконання яких необхідно ввести у командній стрічці діалогового вікна *Command Window* декілька команд. Якщо завдання достатньо складне, то кількість команд зростає і праця в командній стрічці стає неефективною. Використання історії команд, збереження змінних робочого середовища чи ведення щоденника за допомогою *diary* незначно підвищують ефективність роботи у середовищі. Ефективне розв'язання полягає у написанні власних алгоритмів у вигляді програм (m-файлів), які можна запускати з робочого середовища або із редактора. Вмонтований у *MatLab*[®] редактор m-файлів дає змогу не тільки набирати текст програми та запускати її цілою або частинами, а й редагувати алгоритм.

4.1. Побудова файлу-сценарію

Розглянемо завдання побудови графіка функції

$$z = \sin(x - y), \quad x \in [0, 4], \quad y \in [0, 6].$$

Для цього створимо у редакторі такий скрипт:

```
%graph2.m-3D plot of function z=sin(x-y)
%x in [0,4], y in [0,6]
clear
h=0.1;
x=0:h:4;
y=0:h:6;
[xx,yy]=meshgrid(x,y);
zz=sin(xx-yy);
```

```

mesh(xx,yy,zz)
xlabel(' \it { \bf {x}}', 'FontSize',20)
ylabel(' \it { \bf {y}}', 'FontSize',20)
zlabel(' \it { \bf {z=sin(x-y)}}', 'FontSize',20)

```

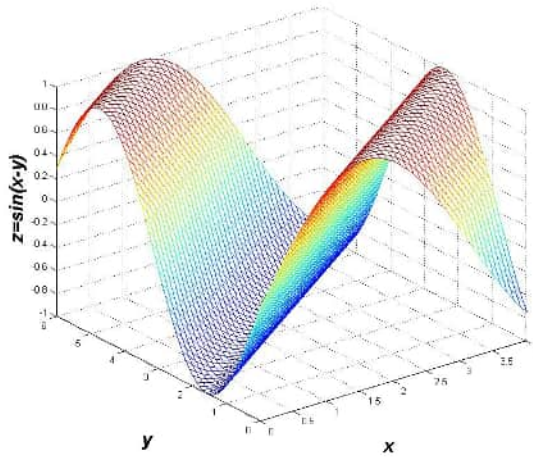


Рис. 4.1. Графік функції, побудований за допомогою m-файлу

редактора. Якщо активне вікно **Editor** і записаний m-файл у цьому вікні, то достатньо натиснути функціональну кнопку на клавіатурі <F5>. Цей спосіб зручний у випадку, коли виконується відладка сценарію. Проте належить пам'ятати, що інформацію про неполадки та помилки система повертає у вікно **Command Window**.

та запишемо його під назвою **graph2.m** із розширенням *.m. Для того, щоб побудувати графік, достатньо у **Command Window** у командній стрічці ввести ім'я файлу **graph2** і натиснути **Enter** (дати команду на виконання), система поверне побудований графік, зображений на рис. 4.1.

Інший спосіб запустити сценарій, який записаний у m-файлі – безпосередньо з вікна

4.2. Робота в редакторі m-файлів

Відкриємо меню **File** середовища **MatLab[®]** і у пункті **New** виберемо підпункт **m-file** або натиснути кнопку **New m-file** на панелі інструментів. У вікні редактора m-файлів відкриється вікно **Editor**. Вид стрічки меню і панелі інструментів залежить від ширини вікна. У разі потреби можна вікно зафіксувати курсором мишки і "розтягнути" до потрібної ширини.

По суті, вікно редактора m-файлів є звичайним текстовим записником для створення скриптів. Якщо користувач добре володіє синтаксисом і досвідом написання програми у середовищі **MatLab[®]**, то m-файл

можна створити у звичайному системному записнику **Windows**, а при записуванні надати розширення ***.m**. Отож, у активному вікні редагування m-файлів середовища **MatLab[®]** введемо такі стрічки:

```
% My mFile_1 - calculus plots
% y=exp(-x) and y=sin(x) in interval [x0,x1].
clear x0 x1 h f g;
x0=input('Input x0: ');
x1=input('Input x1: ');
h=input('Input step h: ');
x=x0:h:x1;
f=exp(-x);
subplot(1,2,1)
plot(x,f);
g=sin(x);
title(' \bf {\it {f(x)=e^{-x}}}', 'FontSize',20)
xlabel(' \bf {\it {x}}', 'FontSize',20)
ylabel(' \bf {\it {f(x)}}', 'FontSize',20)
subplot(1,2,2)
plot(x,g)
title(' \bf {\it {g(x)=sin{x}}}', 'FontSize',20)
xlabel(' \bf {\it {x}}', 'FontSize',20)
ylabel(' \bf {\it {g(x)}}', 'FontSize',20)
```

Збережемо тепер файл під назвою **myMFile_1.m** у робочому каталозі, вибравши у меню **File** редактора пункт **Save as**. Для того, щоб запустити на виконання *всіх команд*, що містяться у файлі, необхідно у меню натиснути **Run** в меню **Debug** або функціональну клавішу **<F5>**. У головному вікні **Command Window** у командній стрічці програма очікуватиме введення значення **x0**: **Input x0: .** Це є значення лівого кінця відрізка, на якому будуть обчислені графіки функцій згідно з командами записаними у m-файлі. Впровадимо, наприклад, $-\pi/4$ та натиснемо **Enter**. Аналогічно, згідно з командами, записаними у m-файлі, у командній стрічці **Command Window** по черзі уведемо 2π для **x1** – значення правого кінця відрізка, та крок **0.01** для **h**. Після підтвердження **Enter** у вікні **Figures** отримаємо поряд розташовані графіки функцій $y = e^{-x}$ та $y = \sin x$ на проміжку $[-\pi/4, 2\pi]$ (див. рис. 4.2).

Звернемо увагу що рядки на те, які набираємо у вікні редактора, автоматично нумеруються. Це дає змогу ідентифікувати повідомлення

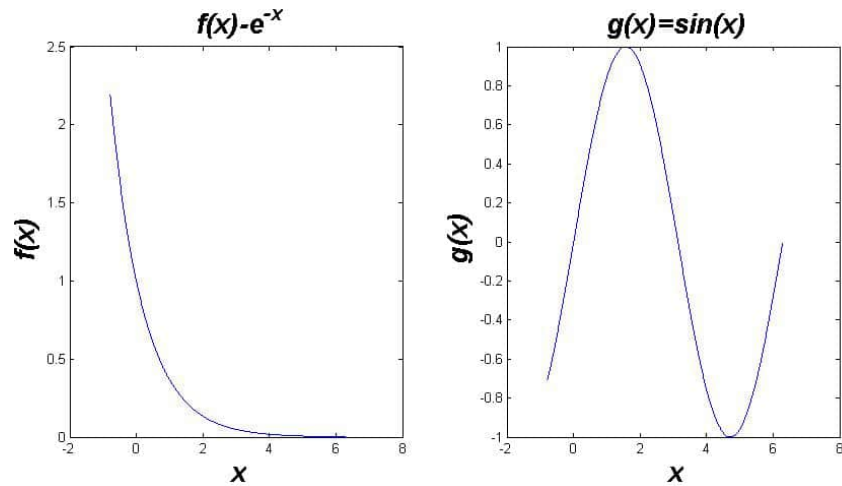


Рис. 4.2. Графіки функцій $y = e^{-x}$ та $y = \sin x$

про помилки набору, які повертаються у командне вікно.

4.3. Скрипти та функції

Зручнішою різновидністю m-файлів є *функції*, першою стрічкою яких є заголовок, який використовує оператор `function`. На відміну від скриптів, функції можуть отримувати вихідні дані у вигляді списку вхідних параметрів і повертати результати обчислень також у вигляді списку вихідних параметрів. Якщо деякі змінні `Workspace` (Робоча область) оголошені глобальними (оператор `global`) і таке оголошення `global` із зазначенням імен спільних змінних є у тілі програми, то функція має доступ до зазначених змінних.

Однією з найважливіших особливостей функцій є апарат локальних змінних. Всі змінні, які появляються у тілі функції, за винятком глобальних змінних, вхідних і вихідних параметрів, вважаються локальними. Вони утворюють локальний робочий простір і доступні тільки у тілі функції, в якій вони визначені, і жодні інші скрипти чи функції не можуть їх використати. Це дуже зручно у написанні програм-функцій, оскільки імена локальних змінних немає необхідності будь із ким узгоджувати.

Зазначимо, що у створенні програм-функцій назва m-файлу, у якому записується програма, обов'язково має збігатися з назвою функції. Розглянемо приклад програми (функції) обчислення факторіала числа. Цій функції дамо назву `factn`.

```
% This program calculus factorial of number n.  
function y=factn(n)  
k=1;  
for(i=1:n)  
    k=k*i;  
end  
y=k;
```

Після цього для обчислення $5!$ достатньо у командній рядку Command Window середовища MatLab[®] увести

```
>> factn(5)
```

і після натискання Enter на екрані можна прочитати:

```
ans=  
    120
```

У *m*-файлі можна визначити кілька функцій. Перша і з них має ті переваги, що до неї можна звернутися ззовні. Решта функцій вважаються внутрішніми і доступні тільки всередині *m*-файлу. Такі функції називають *підфункціями* (subfunctions).

Досить часто для написання програми у середовищі MatLab[®] використовується деякий вираз, що залежить від змінних, і до цього виразу програма звертається кілька разів як до функції цих змінних. Написання *m*-файлу (розглянутий вище спосіб визначення функції) не виправдано, позаяк звертання до цього виразу відбувається невелику кількість разів. З цією метою у MatLab[®] є можливість створити так звані *анонімні функції*. Функція має ім'я (ідентифікатор), визначені змінні та параметри і після її визначення у процесі обчислення програма до неї звертається згідно з її ідентифікатором. Якщо процедура (програма), у якій визначена така анонімна функція, не є активною, то вводять числові та символьні дані.

4.4. Увід числових і символьних даних

Простішим способом уведення числової та символьної інформації за допомогою клавіатури є використання інструкції `input`. Ця функція допускає два формати, перший із яких

```
x=input('prompt')
```

У відповідь на запрошення, яке видає програма, користувач може увести потрібне значення або увести вираз, величина якого буде обчислена

з врахуванням проміжного стану змінних робочого простору та повернення у як значення функції.

Другий формат інструкції `input` має вигляд

```
x=input('promt','s')
```

У цьому випадку текст, який уводиться користувачем, трактується як стрічка символів, яка і повертається як значення функції. В обидвох випадках текст запрошення може містити керуючий символ ”\n”, який керує переведенням курсора на початок наступного рядка. Це забезпечує, у разі необхідності, уведення запрошення, яке складається з декількох рядків.

4.5. Виведення результатів обчислень

Для виведення значення виразу, у частковому випадку якого може бути ім'я довільної змінної, достатньо не набирати символ ”;”, який блокує виведення результату на екран. У інших випадках можна звернутися до функції `disp`, яку відрізняє від автоматичного виведення на екран тільки те, що не відображається на екрані ідентифікатор обчисленої змінної, результат присвоюється системній змінній `ans`. Формат інструкції такий:

```
disp(expr)
```

Зокрема, аргументом функції `disp` може бути стрічка в одинарних лапках:

```
disp('expr')
```

внаслідок чого на екрані буде надрукована стрічка (один із об'єктів MatLab® типу `string`).

Функціями `error` та `warning` можна скористатися для отримання повідомлення про помилку або попередження. Головний їхній аргумент – текст повідомлення, після виведення якого на екран програма буде або завершена (використання функції `error`), або буде продовжена її робота (використання функції `warning`), наприклад,

```
>> warning('Attention!')
Warning: Attention!
>> error('Attention - ERROR!')
Attention - ERROR!
```

Загалом функція `warning` має набагато ширші можливості. Її загальний формат такий:

```
warning('name_w','format',list)
```

Стрічка `name_w` використовується як ідентифікатор групи повідомлень. За необхідності вивід всіх повідомлень однієї групи можна заблокувати, звернувшись до інструкції `warning` так:

```
warning('off','name_w')
```

У цьому випадку не має необхідності видаляти всі стрічки, які містять повідомлення для цієї групи. Якщо необхідно знову включити це повідомлення, то викликається процедура `warning` із параметром `on`:

```
warning('on','name_w')
```

Детальніше формати описані в інструкції `sprintf`. Після компіляції у командній стрічці вікна `Command Window` запис:

```
>> help sprintf
```

система поверне:

```
sprintf - Format data into string
```

```
This MATLAB function formats the data in arrays A1,...,An according to formatSpec in column order, and returns the results to string str.
```

```
str = sprintf(formatSpec,A1,...,An)
[str,errmsg] = sprintf(formatSpec,A1,...,An)
```

[Reference page for sprintf](#)

See also [char](#), [fprintf](#), [fscanf](#), [int2str](#), [num2str](#), [sscanf](#)

Синім кольором виділені посилання на відповідні розширені статті у довіднику `help` середовища `MatLab`[®], із яких можна одержати детальний опис цих інструкцій.

4.6. Типи функцій

Існує кілька різновидів `m`-функцій. У підрозділі 4.3. ми ознайомилися з головними функціями, назви яких збігаються з назвами `m`-файлів. До них можна звертатися із інших функцій чи рядків `Command Window`, передавати їм параметри. Характерною особливістю `m`-функцій є наявність індивідуального робочого простору – механізму локальних змінних. Разом із ними `m`-функції можуть використовувати змінні інших робочих просторів, де ці змінні оголошені як глобальні.

Другий різновид `m`-функцій утворюють *підфункції* (subfunctions), ви-

значення яких містяться у m-файлі одразу за головною функцією. До підфункцій m-файлу не можна звернутися ззовні. Це функції внутрішнього користування. Вони можуть між собою комунікувати в межах одного m-файлу та звертатися до інших головних функцій, що описані у інших m-файлах. Послідовність звертань підфункцій ініціює головна функція m-файлу. Формат підфункцій має вигляд:

```
function y1=f1(p1,p2)
.....
    function y2=f2(x,z) % вкладена у функцію f1
    .....
        function y3=f3(a,b) % вкладена у функцію f2
        .....
            end % кінець функції f3
        .....
    end % кінець функції f2
.....
end % кінець функції f1
```

При звертанні до вкладених функцій зберігається загально прийняте правило: до вкладеної функції може звертатися тільки безпосередньо функція, якій вона безпосередньо підпорядкована. У наведеному вище прикладі функція `f1` може звертатися до `f2` безпосередньо, але не може прямо звертатися до `f3`.

Крім вкладених (nested functions), як ми вже розглядали, у середовищі MatLab[®] є анонімні функції (anonymous functions). У цих функцій немає власного імені, замість нього використовується вказівник функції. Анонімні функції використовуються у двох форматах:

```
h_F=@(parameters) expr
```

та

```
h_F=inline('expr')
```

★ **Приклад 4.6.1.** Наприклад, визначимо функцію двох змінних:

```
>> ff=@(x,y) x.*exp(x.*sqrt(t));
```

та обчислимо її мішану похідну другого порядку:

```
>> syms x y; diff(diff(f,x),y)
```

```
ans =
```

```
(x^2*exp(-x*y^(1/2)))/2 - (x*exp(-x*y^(1/2)))/y^(1/2)
```

★ **Приклад 4.6.2.** Визначимо тепер функцію однієї змінної:

```
>> clear; ff=inline('exp(-x.^2)-3*sin(2*x)');
```

та побудуємо її графік на відрізку $[-1, 4]$ (Рис. 4.3):

```
>> ezplot(ff,-1,4)
```

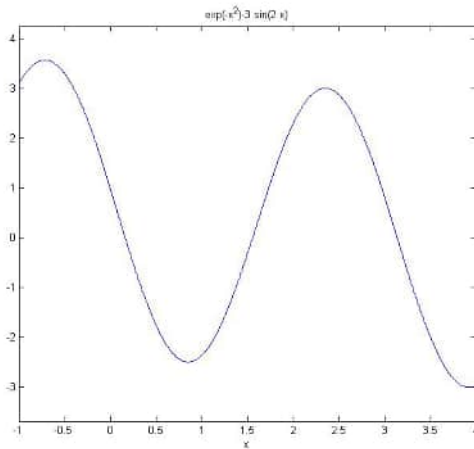


Рис. 4.3. Графік функції, побудований за допомогою *m*-файлу

третього степеня:

```
>> pwr3=@(x) x.^3;
```

Тепер обчислимо $\int_0^1 x^3 dx$ у середовищі MatLab[®], застосувавши внутрішню функцію `quad` обчислення означеного інтеграла квадратурними формулами Simpson'a:

```
>> quad(pwr3,0,1)
```

```
ans=
```

```
0.2500
```

У разі потреби *m*-функції можна помістити у каталог з спеціальним іменем `private`. Ці функції будуть доступними тільки з каталога, який містить `private`. Функції, що містяться у цьому каталозі, вважаються *приватними* (*private functions*). Їхнє призначення – замінити деякі системні функції одноіменними користувача. MatLab[®] починає пошук функцій, до яких звертається, з каталогу `private`, тому вони викликаються найперше.

Очевидно, що такого типу функції зручно використовувати при простому вигляді `expr`. Вище ми розглядали анонімні функції та приклади їх визначення. Зазначимо, що анонімна функція може бути використана іншими функціями як параметр, тобто такі функції можуть бути у вигляді змінних інших функцій. Наприклад, визначимо функцію піднесення до третього степеня:

Накінець варто згадати про так звані *перевантажені функції* (overloaded functions). У об'єктно-орієнтованому програмуванні широко використовується можливість визначати декілька функцій із однаковими ідентифікаторами (іменами), які відрізняються між собою тільки кількістю параметрів, або їхніми типами, або типом повернутого обчислення. Компілятор аналізує чергове звертання до функції і викликає ту з них, яка відповідає шаблону виклику.

4.7. Змінні та параметри функцій

4.7.1. Глобальні змінні

Щоб задекларувати глобальні властивості змінних, використовується інструкція:

```
global a b c
```

Розглянемо приклад.

★ *Приклад 4.7.1.* Нехай задано функцію

$$\psi(n, x) = x^n e^{-nx},$$

де $x \in [0, 10]$, $n = 3, 4, 5, 6$. Змінна x зазвичай є аргументом функції, а n є параметром – глобальною змінною. Утворимо скрипт-файл для функції ψ :

```
function y=psiA(x)
global n
y=x.^n.*exp(-n.*x);
```

та наступний файл-сценарій:

```
%file scenPsiA.m
%plot more curves on the same figure
global n
h=0.001; x=0:h:10;
vn=[1.3 1.5 1.7 1.9 2.5 2.75];
figure(1); hold on
for n=vn
    plot(x,psiA(x),'r-');
end
hold off
```

У цьому прикладі досліджувана функція залежить від змінної x і

параметра n . Оголосивши n глобальною змінною, побудовано однопараметричну сім'ю графіків цієї функції (рис. 4.4).

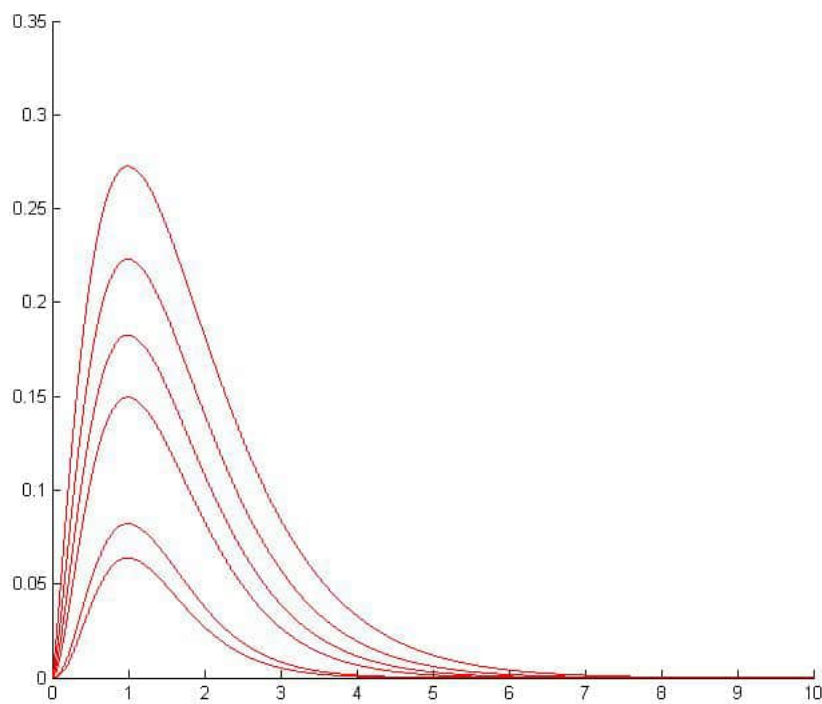


Рис. 4.4. До прикладу 4.7.1

★ **Приклад 4.7.2.** Розглянемо такий приклад, який дає можливість отримати такий самий результат, ввівши функцію двох змінних:

```
function y=psiA2v(x,n)
y=x.^n.*exp(-n.*x);
```

Дещо модифікуємо `scenPsiA.m`, який запишемо під назвою `scenPsiA2v.m`:

```
%file scenPsiA2v.m
%plot more plots on the same figure
clear
h=0.001;
x=0:h:10;
vn=[1.3 1.5 1.7 1.9 2.5 2.75];
figure(2); plot(x,psiA2v(x,vn(1)),'r-');
grid
hold on
for n=2:length(vn)
    plot(x,psiA2v(x,vn(n)),'r-');
end
hold off
```

Результат виконання `scenPsiA2v.m` буде такий самий як у попередньому випадку і зображений на рис. 4.4.

4.7.2. Параметри функцій

Майже усі алгоритмічні мови об'єднують вхідні та вихідні аргументи функцій у загальні дужки і повертають у вигляді свого значення єдиний результат, який зазвичай присвоюється функції. Більшість алгоритмічних мов використовує позиційний принцип задання аргументів. Тільки деякі системи поряд із позиційними аргументами дають змогу використовувати ключовий спосіб задання параметрів у форматі вигляду `name=value`. Ключові параметри завжди одразу слідом за списком позиційних аргументів і можуть бути розташовані у довільному порядку. Функція передає свої параметри через стек, записуючи туди значення або адреси відповідних аргументів. Існує два принципово різних способи записування параметрів у стек – один із них першим поміщає у стек останній аргумент списку, а другий – перший аргумент списку. Зазвичай це потрібно брати до уваги у тих випадках, коли фрагменти програми реалізовані у різних системах програмування.

У мові MatLab[®] вхідні та вихідні аргументи функцій строго розділені. Всі вхідні параметри використовують тільки позиційний принцип і задаються у круглих дужках після імені функції. Всі вихідні параметри оголошуються як масив результатів і в операторах присвоєння записуються у квадратних дужках. Функції можуть мати різну кількість параметрів. Для створення такого типу програм використовуються спеціальні засоби – функції (вірніше, макровизначення) `va_start`, `va_end`, `va_arg`, `va_list`.

MatLab[®] поміщає всі вхідні параметри у масив комірок з іменем `varargin` і запам'ятовує кількість переданих аргументів функції у глобальну змінну `nargin`. Функція, що повертає змінну кількість значень, починається з заголовка

```
function[varargout]=function_name(arguments)
```

Це означає, що результати обчислення функції присвоюються компонентам масиву комірок `varargout`. До визначеної так функції можна звертатися з різною кількістю вихідних параметрів:

```
[y1,y2,y3]=function_name(arguments)
[z1,z2]=function_name(arguments)
w1=function_name(arguments)
```

У першому випадку змінній `y1` буде присвоєно значення комірки `varargout(1)`, змінній `y2` – значення комірки `varargout(2)`, змінній `y3` – значення комірки `varargout(3)`. У другому випадку будуть використані значення двох перших комірок, у третьому – єдине значення першої комірки масиву `varargout`. Через глобальну змінну `nargout` функції передається інформація про кількість запрошуваних даних, щоб вона обчислила потрібну кількість результатів.

Для контролю за допустимою кількістю вхідних та вихідних аргументів `m`-функції можна використовувати інструкції `nargchk` і `nargoutchk`:

```
msg=nargchk(min,max,nargin);
msg=nargoutchk(min,max,nargout);
```

Обидві функції генерують повідомлення `msg`, у яких зазначається недопустима кількість вхідних чи вихідних параметрів, яку можна вивести за допомогою функції `error(msg)`. Якщо кількість параметрів міститься в інтервалі `[min, max]`, то повідомлення `msg` зображається пустою стрічкою і повідомлення про помилку не буде.

Продемонструємо використання описаних вище глобальних змінних на прикладах із файла допомоги, які дещо модифіковані з метою спрощення.

★ *Приклад 4.7.3.* Вхідними параметрами функції `polyline.m` є точки з координатами (x_1, y_1) , (x_2, y_2) і т.д.. За заданими точками функція буде ламану. Скрипт має такий вигляд:

```
function polyline(varargin)
for k=1:length(varargin)
    x(k)=varargin(k)(1);
    y(k)=varargin(k)(2);
```

```
end
axis([min(x) max(x) min(y) max(y)])
figure(3)
plot(x,y)
```

Результат звернення до цієї функції у вигляді

```
>> polyline([1 -1],[0 -2],[-3 -1],[-4 1],[-2 2],[1 1],[1 -1])
```

зображено на першому рис. 4.5. Якщо ж компілювати функцію

```
>> polyline([2 2],[1 -1],[-3 4],[-1 5],[2 2])
```

то одержимо графік, зображений на другому рис. 4.5.

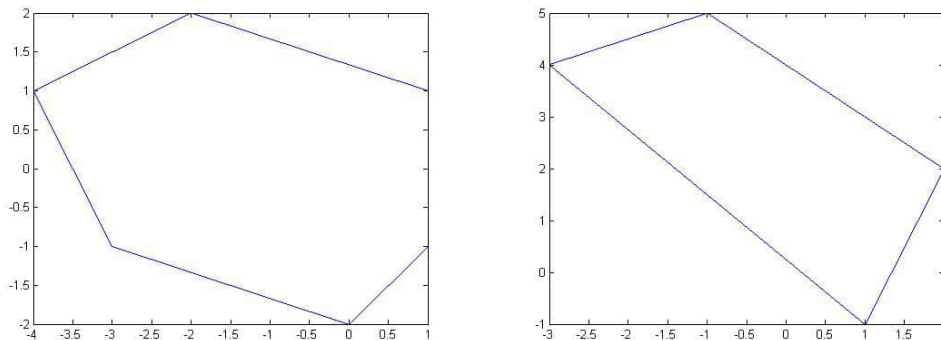


Рис. 4.5. Графіки `polyline` у випадку 7-ми та 5-ти точок

★ *Приклад 4.7.4.* *Передача аргументів функції.* Розглянемо такий приклад. Функція `array2vec` отримує масив розмірності $n \times 2$, координати першого стовпчика якого x_1, x_2, \dots , а другого – y_1, y_2, \dots . Завдання полягає у тому, щоб створити відповідні пари $(x_1, y_1), (x_2, y_2), \dots$. Утворимо m-файл сценарій:

```
function[varargout]=array2vec(a)
for k=1:nargout
    varargoutk=a(k,:);
end
```

Нижче наведено результати двох обчислень за допомогою функції `array2vec`. У першому випадку:

```
>> a1={1 2;3 4;5 6;7 8;9 10;11 12};
```

```
[p1,p2]=array2vec(a1)
p1 =
     1     2
p2 =
     3     4
```

У другому випадку:

```
>> [q1 q2 q3 q4 q5]=array2vec(a1)
q1 =
     1     2
q2 =
     3     4
q3 =
     5     6
q4 =
     7     8
q5 =
     9    10
```

Якщо за вхідний параметр взяти звичайний числовий масив, то результатом обчислення цієї ж функції будуть звичайні числові вектори:

```
>> a1=[1 2;3 4;5 6;7 8;9 10;11 12];
[p1,p2]=array2vec(a1)
p1 =
     1     2
p2 =
     3     4
```

У другому випадку:

```
>> [q1 q2 q3 q4 q5]=array2vec(a1)
q1 =
     1     2
q2 =
     3     4
q3 =
     5     6
q4 =
     7     8
q5 =
     9    10
```

4.7.3. Функції масиву

★ **Приклад 4.7.5.** Розглянемо задачу обчислення графіка функції $f(x) = \frac{xe^{-\sin x}}{1+x^2}$ на проміжку $[-3, 3]$. З цією метою утворимо скриптовий файл funA.m:

```
function y=funA(x)
y=(x*exp(-sin(x)))/(1+x^2);
```

та наступний файл-сценарій pltFunA.m, який дасть відповідь на поставлене завдання:

```
%file pltFunA.m
%Plot a graph by using the function file funA.m
clear
h=0.001;
x=-3:h:3;
n=length(x);
for i=1:n
    y(i)=funA(x(i));
end
figure(4)
plot(x,y,'r. ');
grid on
```

Висновок: для побудови графіка функції $f(x) = \frac{xe^{-\sin x}}{1+x^2}$ ми спочатку побудували масив точок у системі координат та через отримані точки провели лінію, яка є графіком функції. Проте середовище MatLab[®] має змогу побудувати графік цієї функції, не вдаючись безпосередньо до побудови ітерованих значень функції, а безпосередньо обчислювати, тобто будувати масиви за допомогою самих функцій.

★ **Приклад 4.7.6.** Розглянемо таку побудову на прикладі функції $f(x) = \frac{xe^{-\sin x}}{1+x^2}$, але використаємо можливості середовища MatLab[®]. Утворимо такі скрипти-файли:

```
%Def funAsmart.m
function y=funAsmart(x)
y=(x.*exp(-sin(x)))./(1+x.^2);
%file pltFunAsmart.m
%Plotting a graph by using the function file funAsmart.m
clear
```

```
h=0.001;
x=-3:h:3;
y=funAsmart(x);
figure(5); plot(x,y,'r.');
```

$$f(x) = xe^{-\sin x} / (1+x^2)$$

```
title(' \bf { \it {f(x)=xe^{-sinx}/(1+x^2)}}', 'FontSize',20)
grid on
```

Проаналізуємо останні два приклади. Після виконання `pltFunA.m` і `pltFunAsmart.m` система поверне однакові графіки (пропонуємо читачеві самостійно переконатися у цьому). У чому відмінність і про що йдеться? У випадку визначення досліджуваної функції за допомогою `funA.m` використані звичайні операції на скалярах, позаяк ставили собі за мету побудувати поточно графік функції. У другому ж випадку, визначаючи ту саму функцію за допомогою `funAsmart.m`, використані операції на масивах, бо табульовані значення функції обчислені на масиві. Тому використання `funA.m` у процедурі `pltFunAsmart.m` поверне помилку і не обчислить графіка, тоді як у процедурі `pltFunA.m` можна використати `funAsmart.m`.

* **Приклад 4.7.7.** Розв'яжемо задачу знаходження точок нулів функції $f(x) = \frac{xe^{-\sin x}}{1+x^2}$ та визначення її точок мінімуму. Укладемо такий сценарій:

```
%file funA3.m
%test of an array-smart function
clear
h=0.01;
x=-3:h:3;
y=funAsmart(x);
pause on
pause
x1min=fminbnd('funAsmart',-2,-1)
funx1=funAsmart(x1min)
pause
x2min=fminbnd('funAsmart',1,3)
funx2=funAsmart(x2min)
pause
x0=fzero('funAsmart',-0.5)
funx0=funAsmart(x0)
pause off
figure(6)
```

```

plot(x,y,'b.');
```

`grid on`

```

hold on
plot(x2min,funx2,'gs','MarkerSize',10,'MarkerEdgeColor',...
     'b','MarkerFaceColor',[0,1,0])
plot(x0,funx0,'ro','MarkerSize',10,'MarkerEdgeColor',...
     'b','MarkerFaceColor',[0,1,0])
plot(x1min,funx1,'gs','MarkerSize',10,'MarkerEdgeColor',...
     'b','MarkerFaceColor',[0,1,0])
hold off
```

Зазначимо, що в останній процедурі використана інструкція `pause`. У процесі компіляції виконання процедури зупиняється на інструкції `pause`, щоб продовжити обчислення, необхідно в активному вікні `Command Window` дати підтвердження на продовження (натиснути `Enter`). Після кожної паузи в активному рядку `Command Window` система повертає по чергової точці локального мінімуму та значення функції у цій точці, точку перетину графіка з віссю `oX`:

```

x1min =
    -1.3524
funx1 =
    -1.2690
x2min =
     1.8724
funx2 =
     0.1599
x0 =
    2.5164e-18
funx0 =
    2.5164e-18
```

Крім того, згідно з процедурою, обчислено графік функції та вищезгадані точки (див. рис. 4.6).

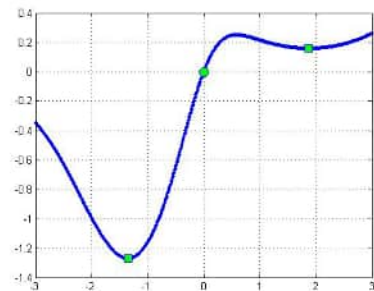


Рис. 4.6. До прикладу 4.7.7

Розділ 5

Алгебра векторів і матриць у MatLab[®]

Назва пакета MatLab походить від скороченої англійської назви *Matrix Laboratory* – дослівно ”матрична лабораторія”. Тому неважко здогадатися, що основним видом даних, які використовуються у пакеті, є матриці. Навіть загальноприйняті скалярні змінні у MatLab[®] трактуються як матриці розмірності 1×1 . З огляду на важливість поняття матриці у пакеті MatLab[®] у цьому розділі детальніше розглянемо основні методи утворення, види та дії на матрицях.

5.1. Утворення векторів і матриць

Вектор або матриця складаються з імені та елементів, взяті у квадратні дужки. Елементи вектора розділяються комою або пробілом. Елементами вектора можуть бути будь-які об’єкти середовища MatLab[®]: дійсні чи комплексні числа, вирази, функції тощо.

★ *Приклад 5.1.1.*

```
>> V=[15,23,-45,7,1/2 ]  
>> V=[-2i 4 10 3+sqrt(2)*i]
```

Для виводу на екран монітора натискаємо клавішу `<Enter>`. Система після компілювання поверне елементи вектора без квадратних дужок, розділених пробілами. У нашому випадку результат буде таким:

```
V =
 15.0000 23.0000 -45.0000 7.0000 0.5000
V =
 0.0000 - 2.0000i 4.0000 + 0.0000i 10.0000 + 0.0000i...
 3.0000 + 1.4142i
```

Зверніть увагу на повернутий після компіляції результат у другому випадку: перший і четвертий елементи вектора – комплексні числа, а другий і третій – дійсні. Результат повернутий у комплексному форматі. Крім того, другим елементом вектора є уявне число, уявна частина якого – обчислене значення. Тому уявну одиницю і "приписано" зліва від числа. Уявна частина четвертого елемента є необчисленим значенням, тобто значення функції `sqrt` у точці 2. Тому між уявною одиницею і та `sqrt(2)` треба вписати знак множення `*`.

Елементами матриці, як і вектора, можуть бути додатні чи від'ємні числа, дійсні чи комплексні. Елементи у рядках, як і векторах, розділяються комою чи пробілом, а рядки – крапкою з комою, `”;`.

★ **Приклад 5.1.2.** Визначимо матрицю 2×3 , у цьому випадку скористаємось інструкцією `randi` пакета MatLab®, яка генерує псевдовипадкові цілі числа, які виберемо з проміжку $[-10, 10]$:

```
M=[randi([-10,10]) randi([-10,10]) randi([-10,10]);...
   randi([-10,10]) randi([-10,10]) randi([-10,10])]
```

Очевидно, при кожній компіляції система поверне різні (випадкові) матриці, а у нашому випадку, наприклад, одержали

```
M =
 10  3 -10
  7  9  4
```

За допомогою інструкції `randi` можна генерувати матриці чи вектори довільної розмірності, використовуючи такий формат

$$\text{randi}(\text{valrange}, sh)$$

де `valrange` – проміжок, із якого генеруються випадкові числа (`value range`), `sh` – розмірність масиву, тобто розмірність вектора, матриці тощо (`shape`).

★ **Приклад 5.1.3.** Визначимо і скомпілюємо вектор `V` і матрицю `M`, відповідно, розмірностей 1×4 і 4×3 :

```
>> V=randi([-11,20],[1 4])
V =
 14 -6 4 3
```

```
>> M=randi([0 20],[4,3])
```

```
M =  
 0 17  0  
 5 14  9  
 0  6  8  
 2 19 16
```

★ *Приклад 5.1.4.* Квадратну матрицю за допомогою інструкції `randi` можна, наприклад 4×4 , згенерувати так:

```
>> M=randi([0 20],4)
```

```
M =  
13 14 10  4  
14 13 20 15  
15  3  7  5  
 5  2 12 10
```

★ *Приклад 5.1.5.* Якщо значення елементів рядка числової матриці утворюють арифметичну прогресію, то матрицю можна згенерувати так:

```
>> M=[1:5 ; -1:0.5:1 ; 3.3:2.3:13.8]
```

```
M =  
 1.0000  2.0000  3.0000  4.0000  5.0000  
-1.0000 -0.5000         0  0.5000  1.0000  
 3.3000  5.6000  7.9000 10.2000 12.5000
```

5.2. Перетворення матриць

Система MatLab[®] дає змогу:

- замінити елементи вектора чи матриці без редагування;
- змінити розмірність вектора чи матриці;
- привести матрицю до іншого вигляду;
- створити матрицю спеціального вигляду.

5.2.1. Звертання до елементів матриці та заміна їхніх значень

Для того, щоб викликати елемент вектора чи матриці, необхідно у круглих дужках справа від імені вектора чи матриці вказати координати

елемента. Продемонструємо це на прикладах.

★ **Приклад 5.2.1.** Утворимо вектор і матрицю, згенерувавши їхні елементи інструкцією `rand`, яка повертає псевдовипадкові числа з проміжку $[0, 1]$:

```
>> V=rand(1,4)
V =
    0.5285    0.1656    0.6020    0.2630
```

```
>> M=rand(3,4)
M =
    0.0540    0.9340    0.4694    0.1622
    0.5308    0.1299    0.0119    0.7943
    0.7792    0.5688    0.3371    0.3112
```

Тепер виконаємо присвоєння змінним `a`, `b`, `c`, `d`, відповідно, значення елементів $V(2)$, $V(3)$ вектора `V` і значення елементів $M(2,1)$, $M(3,3)$ матриці `M`:

```
>> a=V(2); b=V(3); c=M(2,1); d=M(3,3); [a b c d]
ans =
    0.1656    0.6020    0.5308    0.3371
```

★ **Приклад 5.2.2.** Нехай вектор `V` і матриця `M` із попереднього прикладу. Замінімо значення елементів $V(2)$, $V(3)$ вектора `V` і значення елементів $M(2,1)$, $M(3,3)$ матриці `M`, відповідно, значеннями 1, 2, 3, 4:

```
>> V(2)=1; V(3)=2; M(2,1)=3; M(3,3)=4;
```

Перевіримо результати:

```
>> V
V =
    0.5285    1.0000    2.0000    0.2630
```

```
>> M
M =
    0.0540    0.9340    0.4694    0.1622
    3.0000    0.1299    0.0119    0.7943
    0.7792    0.5688    4.0000    0.3112
```

5.2.2. Зміна розмірності вектора та матриці

Зміну розмірності вектора найпростіше виконати шляхом редагування. Змінити розмірності матриці легше виконати за допомогою видалення або приєднання рядків і стовпців. Розмірність матриці (масиву) `M`

повертає інструкція

```
size(M)
```

тоді як інструкція

```
length(M)
```

 повертає максимальну розмірність масиву.

★ **Приклад 5.2.3.** Утворимо тривимірний масив:

```
>> Md=randi([-10 10],[2 4 3])
Md(:,:,1) =
   -10    9    5   -2
     7    4    5    3
Md(:,:,2) =
    -7   -10  -10    7
     4    -5   -8    4
Md(:,:,3) =
    -4   -10   -2    6
     9    -1    6   -7
```

Після чого перевіримо розмірність:

```
>> size(Md)
ans =
     2     3     4
```

та дію інструкції `length`:

```
>> length(Md)
ans =
     4
```

Символ двокрапки ":" у MatLab[®] позначає інструкцію, за допомогою якої повертається діапазон даних. Наприклад, якщо ":" розташувати на першій позиції у круглих дужках при імені матриці M :

$$M(:,n)$$

то MatLab[®] поверне n -й стовпець матриці. Аналогічно, у звертанні

$$M(m,:)$$

одержимо m -й рядок. Тобто, якщо звертання до елемента (m,n) матриці M відбувається так: $M(m,n)$, конкретно зазначивши координати елемента у таблиці M , то, наприклад, звертання $M(m,:)$ означає, що всі елементи стовця (:), які є у m -му рядку. Очевидно, що $M(:,:)$ рівносильне копіюванню M . Проілюструємо це на прикладі.

★ *Приклад 5.2.4.* Утворимо матрицю

```
>> M=rand(4,5)
M =
    0.5060    0.5472    0.8407    0.9293    0.6160
    0.6991    0.1386    0.2543    0.3500    0.4733
    0.8909    0.1493    0.8143    0.1966    0.3517
    0.9593    0.2575    0.2435    0.2511    0.8308
```

Тоді

```
>> M(2,:)
ans =
    0.6991    0.1386    0.2543    0.3500    0.4733
```

```
>> M(:,3)
ans =
    0.8407
    0.2543
    0.8143
    0.2435
```

тобто, виділили другий рядок і третій стовпчик матриці M .

Очевидно, що таким способом можна робити заміну цілого рядка чи стовпця матриці.

★ *Приклад 5.2.5.* Замінімо другий рядок матриці M , згенерованої у попередньому прикладі, на послідовність перших п'яти натуральних чисел. Утворимо рядок (звернемо увагу на спосіб генерування за допомогою ":", зазначивши діапазон зміни), тобто вектор розмірності 5, елементами якого є послідовність перших натуральних чисел:

```
>> Vz=[1:5]
Vz =
     1     2     3     4     5
```

А тепер виконаємо заміну:

```
>> M(2,:)=Vz
M =
    0.5060    0.5472    0.8407    0.9293    0.6160
    1.0000    2.0000    3.0000    4.0000    5.0000
    0.8909    0.1493    0.8143    0.1966    0.3517
    0.9593    0.2575    0.2435    0.2511    0.8308
```

У наступному прикладі з'ясуємо, як за допомогою інструкції ":" ви-

ділити підматриці з цієї матриці.

★ **Приклад 5.2.6.** Нехай матриця M з попереднього прикладу після модифікації. Виділити з неї підматрицю, яка складається з другого по четвертий рядки та перший, другий, третій, п'ятий стовпці:

```
>> M(2:4, [1:3,5])
ans =
    1.0000    2.0000    3.0000    5.0000
    0.8909    0.1493    0.8143    0.3517
    0.9593    0.2575    0.2435    0.8308
```

Зауважимо, що після виділення фрагмента матриці сама матриця не зазнає змін.

Видалення рядка чи стовпця матриці (масиву) M виконується за допомогою інструкції

(:)

який ставиться у дужках після імені матриці:

$M(:,m) = []$ – видалляє рядок m ,

$M(n,:) = []$ – видалляє стовпчик n .

★ **Приклад 5.2.7.** Згенеруємо матрицю M розмірності 5×4 :

```
>> M=randi([0,100],[5,4])
M =
    59    76    53    47
    55    38    78     1
    92    57    94    34
    28     7    13    16
    76     5    57    80
```

Видалити з матриці M другий, третій рядки та третій стовпчик:

```
>> M([2,3],:)=[]
M =
    59    76    47
    28     7    16
    76     5    80
```

```
>> M(:,3)=[]
M =
    59    76
    28     7
    76     5
```

Звернемо увагу, що після видалення рядка/стовпця модифікованої матриці присвоюється ім'я вихідної матриці.

Збільшити розмірність матриці можна шляхом об'єднання матриць малої розмірності в одну більшої. Ця процедура називається *конкатенацією*. Ця процедура виконується шляхом утворення матриці з імен матриць менших розмірностей. У цьому випадку допускаються алгебричні операції над іменами. Розглянемо сказане на прикладах.

* **Приклад 5.2.8.** Утворимо чотири вектори розмірності 4:

```
>> V1=randi([0 50],[1 4]);
    41  46  6  46
>> V2=randi([0 50],[1 4]);
    32  4  14  27
>> V3=randi([0 50],[1 4]);
    48  49  8  49
>> V4=randi([0 50],[1 4]);
    48  24  40  7
```

Утворимо матрицю з векторів V1, V2, V3, V4. Вектори треба розглядати як елементи матриці. Тоді одержимо:

```
>> M=[V1; V2; V3; V4]
M =
    41  46  6  46
    32  4  14  27
    48  49  8  49
    48  24  40  7
```

Виділимо з цієї матриці головні мінори третього порядку:

```
>> M1=M; M2=M;M1(4,:)=[]; M1(:,4)=[]
M1 =
    41  46  6
    32  4  14
    48  49  8
>> M2=M; M2(1,:)=[]; M2(:,1)=[]
M2 =
     4  14  27
    49  8  49
    24  40  7
```

Виконаємо тепер операцію конкатенації. Утворимо тепер із матриць M1 і M2 матрицю розмірності 6×6 за допомогою матриць M1-25, M1-M2;

$2*(M2-25)$, $M1$:

```
>> Mc=[M1-25, M1-M2;2*(M2-25),M1]
Mc =
    16    21   -19 |    37    32   -21
     7   -21   -11 |   -17    -4   -35
    23    24   -17 |    24     9     1
-----|-----
   -42   -22     4 |    41    46     6
    48   -34    48 |    32     4    14
    -2    30   -36 |    48    49     8
```

Остання матриця M_c умовно поділена на підматриці $M1-25$, $M1-M2$; $2*(M2-25)$, $M1$ (для зручності читача), конкатенація яких її утворює.

5.3. Математичні операції з векторами та матрицями

5.3.1. Визначник матриці

Визначник матриці у $\text{MatLab}^{\text{®}}$ обчислюється за допомогою функції $\text{det}(M)$

де M – матриця з дійсними або комплексними елементами.

★ **Приклад 5.3.1.** Утворимо матриці та обчислимо їхні визначники:

```
>> M=randi([-5,10],4)
M =
     8     5    10    10
     9    -4    10     2
    -3    -1    -3     7
     9     3    10    -3

>> det(M)
ans =
   -119.0000

>> MC=[2-3i 4i; -2 1+i];det(MC)
ans =
    5.0000 + 7.0000i
```

Розглянемо функції системи $\text{MatLab}^{\text{®}}$, які дають змогу перетворювати

вектори та матриці, утворювати нові матриці, виконувати математичні операції над елементами векторів і матриць. У практичних обчисленнях такі дії необхідні, якщо обчислення зводяться до матричних операцій.

5.3.2. Ранг матриці

Рангом матриці називається найвищий порядок неособливого мінора цієї матриці. Для визначення рангу матриці у MatLab® слугує інструкція

- rank(A)

★ *Приклад 5.3.2.* Задана матриця

$$\begin{pmatrix} 3 & 10 & 3 & 10 & 3 & 1 & -1 \\ 0 & 2 & 0 & 5 & 0 & 6 & 3 \\ 1 & 9 & 0 & 9 & 10 & 5 & 0 \\ 9 & 4 & 0 & 6 & 5 & 6 & 6 \\ 0 & 2 & 0 & 5 & 0 & 6 & 3 \end{pmatrix}$$

Визначити ранг і ранговий мінор матриці.

```
>> A=[3 10 3 10 3 1 -1;...
0 2 0 5 0 6 3;...
1 9 0 9 10 5 0;...
9 4 0 6 5 6 6;...
0 2 0 5 0 6 3]
```

Визначимо ранг матриці A:

```
>> rank(A)
ans =
4
```

Виділимо з матриці A головний мінор четвертого порядку, тобто мінор, який одержимо з матриці A після вилучення з неї п'ятого рядка та трьох останніх стовпців. Після цього перевіримо ранг одержаного мінора. Може статися так, що ранг цього мінора менший ніж чотири, тоді треба вилучити інший рядок та інші три стовпці. Щоб зберегти первинний вигляд матриці A, зробимо переприсвоєння.

```
>> B=A;B(5,:)=[ ];B(:,5:7)=[ ]
```

```
B =
```

```
 3  10  3  10
 0   2  0   5
 1   9  0   9
 9   4  0   6
```

```
>> rank(B)
```

```
ans =
```

```
 4
```

Отож, B є ранговим мінором матриці A .

5.3.3. Транспонування матриці

Транспонованою до матриці M називається матриця, в якій рядки замінені відповідними стовпцями і навпаки, без зміни порядковості елементів у них. Транспонування у MatLab[®] виконується за допомогою подання вихідної матриці M у вигляді:

M'

★ *Приклад 5.3.3.* Згенеруємо матрицю:

```
>> M=rand(3,5)
```

```
M =
```

```
 0.4218  0.9595  0.8491  0.7577  0.6555
 0.9157  0.6557  0.9340  0.7431  0.1712
 0.7922  0.0357  0.6787  0.3922  0.7060
```

Тоді транспонована матриця до M :

```
>> M'
```

```
ans =
```

```
 0.4218  0.9157  0.7922
 0.9595  0.6557  0.0357
 0.8491  0.9340  0.6787
 0.7577  0.7431  0.3922
 0.6555  0.1712  0.7060
```

5.3.4. Слід матриці

Слідом матриці називається сума її діагональних елементів. У системі MatLab[®] обчислюється за допомогою інструкції

```
trace(M)
```

де M – задана матриця.

★ **Приклад 5.3.4.** Розглянемо матрицю

```
M=randi([-5,10],4)
M =
-5   8  -5   7
-1   6   2  -3
-5   0   1   2
-4  10   7   2
```

Її діагональними елементами є -5, 6, 1, 2, сума яких дорівнює 4:

```
>> trace(M)
ans =
4
```

5.3.5. Одинична матриця

У середовищі MatLab® *одиничну матрицю* генерують за допомогою інструкцій:

- $\text{eye}(n)$ – повертає одиничну квадратну матрицю порядку n ;
- $\text{eye}(m,n)$ – повертає одиничну матрицю розмірності $m \times n$ із одиницями на діагоналі, початок якої у лівому верхньому куті, а решта елементів дорівнюють нулю;
- $\text{eye}(\text{size}(M))$ – повертає одиничну матрицю співрозмірну з матрицею M .

Розглянемо приклади.

★ **Приклад 5.3.5.**

```
>> E3=eye(3)
E3 =
1  0  0
0  1  0
0  0  1
```

★ **Приклад 5.3.6.** Згенеруємо прямокутну матрицю:

```
>> M=rand(3,6)
M =
    0.4733    0.5853    0.2858    0.3804    0.0540    0.9340
    0.3517    0.5497    0.7572    0.5678    0.5308    0.1299
    0.8308    0.9172    0.7537    0.0759    0.7792    0.5688
```

та обчислимо відповідну їй співрозмірну одиничну матрицю:

```
>> eye(size(M))
ans =
    1    0    0    0    0    0
    0    1    0    0    0    0
    0    0    1    0    0    0
```

5.3.6. Матриця з одиничними елементами

У середовищі MatLab[®] матрицю з одиничними елементами генерують за допомогою інструкцій:

- `ones(n)` – повертає квадратну матрицю порядку n , всі елементи якої дорівнюють одиниці;
- `ones(m,n)` – повертає прямокутну матрицю розмірності $m \times n$, всі елементи якої дорівнюють одиниці ;
- `ones(size(M))` – повертає матрицю, співрозмірну із матрицею M , всі елементи якої дорівнюють одиниці.

Розглянемо приклади.

★ *Приклад 5.3.7.*

```
>> O=ones(4)
O =
    1    1    1    1
    1    1    1    1
    1    1    1    1
    1    1    1    1
```

★ *Приклад 5.3.8.* Для матриці з прикладу 5.3.5 обчислимо матрицю з одиничними елементами:

```
>> ones(size(M))
ans =
    1    1    1    1    1    1
    1    1    1    1    1    1
    1    1    1    1    1    1
```

5.3.7. Матриця з нульовими елементами

У середовищі MatLab® матрицю із нульовими елементами генерують за допомогою інструкцій:

- `zeros(n)` – повертає квадратну матрицю порядку n , всі елементи якої дорівнюють нулю;
- `zeros(m,n)` – повертає прямокутну матрицю розмірності $m \times n$, всі елементи якої дорівнюють нулю;
- `zeros(size(M))` – повертає матрицю, співрозмірну із матрицею M , всі елементи якої дорівнюють нулю.

Розглянемо приклади.

★ *Приклад 5.3.9.*

```
>> Z=zeros(4)
Z =
    0    0    0    0
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

★ *Приклад 5.3.10.* Для матриці з прикладу 5.3.5 обчислимо матрицю з нульовими елементами:

```
>> zeros(size(M))
ans =
    0    0    0    0    0    0
    0    0    0    0    0    0
    0    0    0    0    0    0
```

5.3.8. Утворення матриці з наперед заданою діагоналлю

У середовищі MatLab® є можливість утворення матриці з наперед заданою діагоналлю, яка є елементами деякого вектора. Це можна вико-

нати за допомогою таких інструкцій:

- $M=\text{diag}(V,k)$ – повертає квадратну матрицю M , однією з діагоналей є елементи вектора V . Якщо $k=0$, то елементи вектора V слугують головною діагоналлю матриці, якщо $k>0$ – елементи вектора слугують k -ю наддіагоналлю, якщо $k<0$ – елементи вектора – k -ю піддіагоналлю. Решта елементів матриці M дорівнюють нулю;
- $M=\text{diag}(V)$ – повертає квадратну матрицю з діагональними елементами вектора V , а решта елементів дорівнюють нулю;
- $V=\text{diag}(M,k)$ – повертає вектор-стовпчик, елементами якого є k -а діагональ прямокутної матриці M : при $k=0$ – головна діагональ, початок якої у лівому верхньому куті, при $k>0$ – k -а наддіагональ, при $k<0$ – k -а піддіагональ;
- $V=\text{diag}(M)$ – повертає головну діагональ квадратної матриці M у вигляді вектор-стовпчика.

★ *Приклад 5.3.11.*

```
>> V=[4 0 5 -5]
```

```
V =  
 4  0  5 -5
```

```
>> M=diag(V,0)
```

```
M =  
 4  0  0  0  
 0  0  0  0  
 0  0  5  0  
 0  0  0 -5
```

```
>> M=diag(V,2)
```

```
M =  
 0  0  4  0  0  0  
 0  0  0  0  0  0  
 0  0  0  0  5  0  
 0  0  0  0  0 -5  
 0  0  0  0  0  0  
 0  0  0  0  0  0
```

```
>> M=diag(V,-1)
M =
    0    0    0    0    0
    4    0    0    0    0
    0    0    0    0    0
    0    0    5    0    0
    0    0    0   -5    0

>> M=rand(5,4)
M =
    0.0760    0.4173    0.4893    0.7803
    0.2399    0.0497    0.3377    0.3897
    0.1233    0.9027    0.9001    0.2417
    0.1839    0.9448    0.3692    0.4039
    0.2400    0.4909    0.1112    0.0965

>> V=diag(M,-2)
V =
    0.1233
    0.9448
    0.1112
```

5.3.9. Перестановка елементів матриці

Перестановка стовпців і рядків матриці виконується за допомогою таких інструкцій:

- `fliplr(M)` – переставляє стовпці матриці M стосовно вертикальної осі;
- `flipud(M)` – переставляє рядки матриці M стосовно горизонтальної осі.

★ *Приклад 5.3.12.*

```
>> M=randi([-10,10],[3 4])
M =
   -8    2   -3  -10
    9   -9    7   -7
   10   -6  -10    3

>> MC=fliplr(M)
MC =
  -10   -3    2   -8
   -7    7   -9    9
    3  -10   -6   10
```

```
>> MR=flipud(M)
```

```
MR =  
 10  -6  -10   3  
   9  -9   7  -7  
  -8   2   -3 -10
```

Крім того, у системі MatLab[®] є функція перестановки елементів, яка з n вимірного вектора V утворює матрицю розмірності $n \times n!$, тобто, кількість рядків відповідає кількості перестановок елементів матриці V . Ця інструкція використовується у форматі

```
perms(V)
```

де V – n вимірний вектор.

★ *Приклад 5.3.13.*

```
>> V=[1 2 3]; VP=perms(V)
```

```
VP =  
 3 2 1  
 3 1 2  
 2 3 1  
 2 1 3  
 1 2 3  
 1 3 2
```

```
>> length(VP)
```

```
ans =  
 6
```

Тобто, кількість рядків матриці VP , якщо розмірність вектора V є $n = 3$, дорівнює $n! = 3! = 6$.

5.3.10. Обернена матриця

Оберненою до квадратної матриці M називається матриця M^{-1} , одержана у результаті ділення одиничної матриці E співрозмірної з M на саму матрицю M : $M^{-1} = E/M$. Функція MatLab[®], яка повертає обернену до матриці M , застосовується у форматі:

```
inv(M)
```

★ *Приклад 5.3.14.*

```
>> M=randi([-10 10],4)
```

```
M =
    5   -4   -7   -9
    3    5   -3    9
   -1   -7    3    6
    1    4    6    0
```

```
>> MInv=inv(M)
```

```
MInv =
    0.1330    0.0833    0.0746    0.1595
   -0.0373    0.0254   -0.0941    0.0162
    0.0027   -0.0308    0.0503    0.1293
   -0.0227    0.0590    0.0442   -0.0191
```

Зазначимо, що цю саму обернену матрицю одержимо, якщо виконаємо операцію із означення.

★ *Приклад 5.3.15.*

```
>> MI=eye(4)/M
```

```
MI =
    0.1330    0.0833    0.0746    0.1595
   -0.0373    0.0254   -0.0941    0.0162
    0.0027   -0.0308    0.0503    0.1293
   -0.0227    0.0590    0.0442   -0.0191
```

5.3.11. Вектор рівновіддалених точок

Досить часто у написанні програм, які реалізують деякий чисельний алгоритм на проміжку $[a, b]$, виникає необхідність у створенні рівномірної сітки вузлів на цьому проміжку. У середовищі MatLab® для генерації рівномірної сітки вузлів, або скажемо загальніше – утворення вектора з рівновіддаленими елементами, слугує інструкція `linspace`, яку вживаємо у форматах:

- `linspace(a,b)` – повертає масив із 100 точок рівновіддалених на $[a, b]$;
- `linspace(a,b,n)` – повертає масив із n точок рівновіддалених на $[a, b]$.

★ *Приклад 5.3.16.*

```
>> ndAB=linspace(-1,1)
```

```
ndAB =
   -1.0000  -0.9798  -0.9596  ...  0.9596  0.9798  1.0000
```

```
>> ndAB5=linspace(-1,1,5)
ndAB5 =
-1.0000 -0.5000 0 0.5000 1.0000
```

5.3.12. Виділення трикутних частин матриці

Виділення трикутних частин матриці у середовищі MatLab[®] виконується за допомогою таких інструкцій:

- `tril(M)` – повертає нижню трикутну частину матриці M , починаючи від головної діагоналі, решта елементів дорівнює нулю;
- `tril(M,k)` – повертає нижню трикутну матрицю, починаючи від k -ї діагоналі;
- `triu(M)` – повертає верхню трикутну частину матриці M , починаючи від головної діагоналі, решта елементів дорівнює нулю;
- `triu(M)` – повертає верхню трикутну матрицю, починаючи від k -ї діагоналі.

Зазначимо, що матриця M не обов'язково має бути квадратною.

★ *Приклад 5.3.17.* Нехай

```
>> M=randi([-10 10],[3,5])
M =
 3  -4  7  -7  -4
 2  -1 -6  -6  9
-6  -6 -6  -1  -1
```

Тоді

```
>> Tr1=tril(M)
Tr1 =
 3  0  0  0
 2 -1  0  0
-6 -6 -6  0

>> Tr2=tril(M',-2)
Tr2 =
 0  0  0
 0  0  0
 7  0  0
-7 -6  0
-4  9 -1
```

Табл. 5.1. Таблиця матричних операторів MatLab®

| Функція | Назва | Оператор | Синтаксис |
|----------|--|----------|-----------|
| plus | Плюс (додавання матриць) | + | M1+M2 |
| minus | Мінус (віднімання матриць) | - | M1-M2 |
| times | Почленне множення масивів чисел | .* | M1.*M2 |
| mtimes | Матричне множення | * | M1*M2 |
| mpower | Піднесення матриці до степеня | ^ | M1^X |
| power | Почленне піднесення до степеня елементів матриці | .^ | M1.^X |
| mrdivide | Ділення матриці зліва на право | / | M1/M2 |
| mldivide | Обернене ділення матриць | \ | M1\M2 |
| rdivide | Почленне ділення елементів матриці зліва направо | ./ | M1./M2 |
| ldivide | Почленне ділення елементів матриці справа наліво | .\ | M1.\M2 |

```
>> Tr3=triu(M)
Tr3 =
     3    -4     7    -7    -4
     0    -1    -6    -6     9
     0     0    -6    -1    -1
```

5.4. Математичні операції над векторами та матрицями

5.4.1. Арифметичні операції над векторами та матрицями

Над векторами та матрицями можна виконувати практично всі операції, що і над числами: додавання та віднімання, множення і ділення, обчислення елементарних функцій, таких як піднесення до степеня, обчислення квадратного кореня, обчислення логарифма, обчислення тригонометричних функцій. У цьому випадку матричними операторами є майже всі арифметичні оператори (див. табл. 5.1).

★ *Приклад 5.4.1.* Визначимо дві матриці та продемонструємо дію арифметичних операторів на цих матрицях.

```
>> m1=randi([-10 10],[2 3])
```

```
m1 =  
    10     6    -2  
     0    -8     9
```

```
>> m2=randi([-10 10],[2 3])
```

```
m2 =  
     6     3     7  
    10   -10     9
```

Тепер послідовно застосуємо до матриць `m1` та `m2` арифметичні оператори з табл. 5.1.

```
>> m1+m2
```

```
ans =  
    16     9     5  
    10   -18    18
```

```
>> m1-m2
```

```
ans =  
     4     3    -9  
   -10     2     0
```

```
>> m1.*m2
```

```
ans =  
    60    18   -14  
     0    80    81
```

Але для виконання операції матричного добутку треба, щоб матричні множники мали відповідні розмірності. Тому, наприклад,

```
>> m1*m2'
```

```
ans =  
    64    22  
    39   161
```

```
>> m1'*m2
```

```
ans =  
    60    30    70  
   -44    98   -30  
    78   -96    67
```

Наступна матрична операція визначена для квадратної матриці та під-

несення її до скалярного степеня. Тому визначимо квадратну матрицю $m3$ і виконаємо піднесення її до степеня.

```
>> m3=randi([-10 10],3)
```

```
m3 =
     4     -2     4
     5     3    -10
     5     -7     -5
```

```
>> m3^3
```

```
ans =
    -26   -290    444
    485    -43   -950
    435   -713   -775
```

Ця дія рівнозначна такій:

```
>> m3*m3*m3
```

```
ans =
    -26   -290    444
    485    -43   -950
    435   -713   -775
```

Крім того, наприклад,

```
>> m3^(1/3)
```

```
ans =
    1.6828 + 0.1059i   -0.2351 - 0.0962i    0.2718 - 0.2249i
    0.2376 - 0.4959i    1.7520 + 0.4505i   -0.4459 + 1.0532i
    0.1646 - 0.6493i   -0.3822 + 0.5899i    1.4361 + 1.3790i
```

Щоб ліпше зрозуміти наступну операцію, визначимо такі матриці і матричний степінь.

```
>> m4=[1 3 5;2 4 6];m5=[2 1 3;-2 -1 -2];m4.^m5
```

```
ans =
    1.0000    3.0000   125.0000
    0.2500    0.2500    0.0278
```

Наступна операція з таблиці слугує до розв'язування системи лінійних рівнянь $x \cdot A = B$. Тоді розв'язок системи (якщо існує) $x = B/A$ у випадку, коли матриця A є квадратною порядку n , а кількість стовпців матриці B дорівнює n . Розглянемо приклад. Розв'язати систему $X \cdot m3 = m2$, де матриці $m3$ і $m2$ визначені вище.

```
>> X=m2/m3
X =
    1.9082    0.4531   -0.7796
    2.1429   -0.3714    0.6571
```

Перевіримо одержаний результат.

```
>> X*m3
ans =
    6.0000    3.0000    7.0000
   10.0000  -10.0000    9.0000
```

Операція `mldivide` слугує до розв'язування системи лінійних рівнянь $A \cdot X = B$. Якщо розв'язок системи існує, матриця A є квадратною порядку n , а кількість рядків матриці B дорівнює n , то розв'язок системи $X = A \setminus B$. Продемонструємо це на прикладі $m3 \cdot X = m2'$, де матриці $m3$ і $m2$ визначені вище.

```
>> X=m3 \ m2'
X =
    1.1592    0.8122
   -0.3061   -1.4082
    0.1878    0.9837

>> m3*X
ans =
    6.0000   10.0000
    3.0000  -10.0000
    7.0000    9.0000
```

Останні дві операції з табл. 5.1 очевидні – почленне ділення відповідних елементів матриць. Для матриць $m1$ і $m2$, визначених вище, маємо:

```
>> m1 ./ m2
ans =
    1.6667    2.0000   -0.2857
         0    0.8000    1.0000

>> m1 . \ m2
ans =
    0.6000    0.5000   -3.5000
    Inf    1.2500    1.0000
```

Зауважимо, що остання операція рівноважна наступній:

```
>> m2./m1
ans =
    0.6000    0.5000   -3.5000
    Inf    1.2500    1.0000
```

Крім арифметичних операцій на векторах і матрицях, розглянемо матричні операції.

5.4.2. Скалярний добуток

Скалярний добуток двох масивів A і B у середовищі MatLab® обчислюється за допомогою оператора `dot` у таких форматах:

- `dot(A,B)` – повертає скалярний добуток двох масивів; якщо A і B вектори, тоді необхідно, щоб вони були однакової довжини, якщо A і B – матриці чи багатовимірні масиви, тоді їхні розмірності мають бути рівними;
- `dot(A,B,dim)` – повертає скалярний добуток двох масивів A і B в напрямі вимірності dim ; dim є натуральним числом.

★ *Приклад 5.4.2.* Згенеруємо дві матриці A і B :

```
>> A=randi([-10 10],[2 3])
A =
    7   -8    3
    9    9   -8

>> B=randi([-10 10],[2 3])
B =
   -5   10   -7
    1   10   10
```

Тоді скалярний добуток двох векторів $V1=A(1,:)$ і $V2=B(1,:)$ ($V1$ і $V2$ є першими рядками, відповідно, матриць A і B) обчислюється так:

```
>> V1=A(1,:)
V1 =
    7   -8    3

>> V2=B(1,:)
V2 =
   -5   10   -7
```

```
>> dot(V1,V2)
```

```
ans =  
-136
```

Справді

```
>> 7*(-5)+(-8)*10+3*(-7)
```

```
ans =  
-136
```

Скалярним добутком двох матриць є вектор, елементи якого – скалярні добутки відповідних стовпців цих матриць:

```
>> dot(A,B)
```

```
ans =  
-26  10 -101
```

Вимірністю $\text{dim}=2$ матриці є вимірність у напрямі стовпців, тому

```
>> dot(A,B,2)
```

```
ans =  
-136  
19
```

Тобто, кожен елемент повернутого вектора $\text{dot}(A,B,2)$ є скалярним добутком відповідних рядків матриць (порівняти першу компоненту $\text{dot}(A,B,2)$ із $\text{dot}(V1,V2)$ вище).

5.4.3. Обчислення добутків елементів масиву

У середовищі MatLab[®] є вмонтовані інструкції для обчислення добутків елементів масивів:

- $\text{prod}(A)$ – повертає добуток елементів масиву, якщо A вектор і вектор-рядок, якщо A – матриця. У другому випадку кожен елемент вектор-рядка є добутком відповідного стовпця матриці;
- $\text{prod}(A,\text{dim})$ – повертає матрицю з добутками елементів масиву A за стовпцями ($\text{dim}=1$), за рядками ($\text{dim}=2$), за іншими розмірностями, залежно від значення dim ;
- $\text{cumprod}(A)$ – повертає добутки з накопиченням. Якщо A вектор, то $\text{cumprod}(A)$ повертає вектор, елементами якого є добутки A із накопиченням. Якщо A матриця, то $\text{cumprod}(A)$ матрицю тих самих розмірів, перший рядок якої такий самий як A , у другому рядку

добутки відповідних елементів у стовпцях, у третьому – добутки відповідних елементів у стовпці перших трьох рядків і т.д.;

- `cumprod(A,dim)` – повертає добуток із накопиченням елементів за стрічками або стовпцями матриці залежно від значення `dim`.

★ *Приклад 5.4.3.* Згенеруємо матрицю та розглянемо дію зазначених інструкцій.

```
>> A=randi([-1 10],[4 5])
A =
     6     7     6     2     7
    -1     8     1    -1     2
     9     7     7     0    10
    10     3    -1     8    -1
```

Утворимо вектор:

```
>> V=A(2,:)
V =
    -1     8     1    -1     2
```

Застосуємо інструкції обчислення добутків елементів вектора V і матриці A

```
>> prod(V)
ans =
    16
>> prod(A)
ans =
   -540   1176   -42     0   -140
>> prod(A,2)
ans =
   3528
    16
     0
    240
```

Тобто, повернуто добуток елементів матриці A у відповідному рядку.

```
>> cumprod(V)
ans =
    -1    -8    -8     8    16
```

```
>> cumprod(A)
ans =
     6     7     6     2     7
    -6    56     6    -2    14
   -54   392    42     0   140
  -540  1176   -42     0  -140
```

Порівняти перший рядок матриці `cumprod(A)` із першим рядком матриці `A`. Кожен елемент i -го рядка матриці `cumprod(A)` є добутком елементів матриці `A` у відповідному стовпчику від першого до i -го рядка матриці `cumprod(A)`.

5.4.4. Підсумовування елементів масиву

У `MatLab`[®] визначено такі інструкції, за допомогою яких обчислюється сума елементів масиву:

- `sum(A)` – повертає суму елементів масиву, якщо `A` вектор і вектор-рядок, якщо `A` – матриця. У другому випадку кожен елемент вектор-рядка є сумою відповідного стовпця матриці;
- `sum(A,dim)` – повертає матрицю з сумою елементів масиву `A` за стовпцями (`dim=1`), за рядками (`dim=2`), за іншими розмірностями, залежно від значення `dim`;
- `cumsum(A)` – повертає суми з накопиченням. Якщо `A` вектор, то `cumsum(A)` повертає вектор, елементами якого є суми `A` із накопиченням. Якщо `A` матриця, то `cumsum(A)` є матриця тих самих розмірів, перший рядок якої такий самий як `A`, у другому рядку суми відповідних елементів у стовпцях, у третьому – суми відповідних елементів у стовпці перших трьох рядків і т.д.;
- `cumsum(A,dim)` – повертає суми з накопиченням елементів за стрічками або стовпцями матриці залежно від значення `dim`.

Розглянемо дію інструкції підсумовування елементів масиву на прикладах. Для цього визначимо матрицю `A` і вектор `V`.

```
>> A=randi([-10 10],[4 5])
A =
    -1    -7     4     3    10
    -2     0     5    -7    -3
     6    -1    -5    -8     2
     6     3     4     0    -6
```

```
>> V=A(3,:)
V =
    6   -1   -5   -8    2
>> sum(V)
ans =
   -6
>> sum(A)
ans =
    9   -5    8  -12    3
>> sum(A,2)
ans =
    9
   -7
   -6
    7
>> cumsum(A)
ans =
   -1   -7    4    3   10
   -3   -7    9   -4    7
    3   -8    4  -12    9
    9   -5    8  -12    3
>> cumsum(A,2)
ans =
   -1   -8   -4   -1    9
   -2   -2    3   -4   -7
    6    5    0   -8   -6
    6    9   13   13    7
```

5.5. Завдання для самостійного опрацювання

◆ *Завдання 5.5.1.* Для заданих лінійних однорідних алгебричних систем потрібно:

- визначити ранг матриці системи;
- виділити ранговий мінор;
- впровадити необхідну кількість довільних сталих (їхня кількість

дорівнює $n - r$, де n – кількість невідомих системи; r – ранг матриці системи); ці довільні сталі визначити у MatLab[®] як символльні змінні;

г) визначити загальний розв'язок системи за допомогою рангового мінора, використовуючи дії на матрицях;

д) перевірити одержаний результат,

якщо

$$1. \begin{cases} 2x_2 + 8x_3 + 5x_4 = 0, \\ 4x_2 + 8x_3 + 2x_4 + 4x_5 = 0, \\ 8x_1 + 4x_3 + 8x_4 + 4x_5 = 0, \\ 2x_1 + 4x_2 + 9x_3 + 4x_4 + 5x_5 = 0. \end{cases}$$
$$2. \begin{cases} 14x_1 + 10x_2 + 6x_3 - 2x_4 - 2x_5 + 8x_6 = 0, \\ 7x_1 - 1x_2 + 20x_3 + 20x_4 + 15x_5 + 19x_6 = 0, \\ 1x_1 + 32x_2 - 20x_3 - 29x_4 - 22x_5 - 31x_6 = 0, \\ 1x_1 + 20x_2 + 14x_3 + 13x_4 + 10x_5 - 1x_6 = 0. \end{cases}$$

мінімальне значення змінних; x_{\max} – стовпець обмежень на максимальне значення змінних.

У випадку, якщо у задачі лінійного програмування немає обмежень типу рівностей або нерівностей, тоді, відповідно, використовуються такі формати команди:

```
linprog(c,Aneq,bneq,[],[],xmin,xmax)
linprog(c,[],[],Aeq,beq,xmin,xmax)
```

У всіх вищеперелічених випадках система за замовчуванням поверне оптимальний розв'язок, тобто вектор x_{opt} , який є розв'язком задачі лінійного програмування (6.1.1)-(6.1.2). Якщо змінити формат вихідних даних, тобто використати інструкцію `linprog` у такому форматі:

```
[xopt, fval]=linprog(c,Aneq,bneq,Aeq,beq,xmin,xmax)
```

то система поверне оптимальний розв'язок x_{opt} і значення цільової функції f_{val} на ньому.

♣ **Зауваження 6.1.1.** Якщо досліджувана задача лінійного програмування не сформульована у стандартному вигляді (6.1.1)-(6.1.2), тоді її треба звести до такого вигляду шляхом домноження на "-1": *i*) цільову функцію, якщо досліджується максимум цільової функції; *ii*) обмеження типу нерівності, якщо нерівність така " \geq ". Очевидно, що у випадку *i*) максимальне значення цільової функції дорівнюватиме $-f_{\text{val}}$.

Продемонструємо розв'язування задач лінійного програмування у середовищі MatLab®.

★ **Приклад 6.1.1.** Користуючись стандартною інструкцією середовища MatLab®, знайти розв'язок задачі:

$$f(x_1, x_2, x_3, x_4) = x_1 - x_2 - x_3 + 4x_4 \rightarrow \min,$$

$$\begin{cases} x_1 + 2x_2 - x_3 + x_4 = 2, \\ 2x_1 - x_2 + x_3 - 2x_4 = 3, \\ x_1 - 2x_2 + 3x_3 - x_4 = 6, \\ x_i \geq 0, \quad i = 1, 2, 3, 4. \end{cases}$$

У досліджуваній задачі немає умови типу нерівностей, тому замість матриці коефіцієнтів і вільних членів системи нерівностей у інструкцію `linprog` підставимо пусті масиви: `Aneq=[]`, `bneq=[]`. Впровадимо у командній стрічці вхідні дані задачі та інструкцію на розв'язування:

```
>> c=[1 -1 -1 4];
Aneq=[];
bneq=[];
```

```
Aeq=[1 2 -1 1;2 -1 1 -2;1 -2 3 -1];
beq=[2;3;6];
xmin=zeros(1,4);
xmax=[ ];
[xopt, fval]=linprog(c,Aneq,bneq,Aeq,beq,xmin,xmax)
```

Після виконання обчислення система поверне розв'язок:

```
Optimization terminated.
xopt =
    1.0000
    2.0000
    3.0000
    0.0000
fval =
   -4.0000
```

★ **Приклад 6.1.2.** Розв'язати задачу лінійного програмування:

$$f(x_1, x_2, x_3) = 20x_1 + 25x_2 + 17x_3 \rightarrow \min,$$

$$\begin{cases} 4x_1 + 5x_2 + 2x_3 \geq 400; \\ x_1 + x_2 + 4x_3 \geq 250; \\ x_1 + x_2 + x_3 = 150; \\ 2x_1 + x_2 + x_3 \leq 300; \\ x_i \geq 0, i = 1, 2, 3. \end{cases}$$

Сформульована задача записана не у стандартному вигляді (6.1.1)-(6.1.2). Отож, щоб обчислити оптимальний розв'язок цієї задачі зведемо, згідно з зауваженням, до стандартного вигляду, а саме:

$$f(x_1, x_2, x_3) = 20x_1 + 25x_2 + 17x_3 \rightarrow \min,$$

$$\begin{cases} -4x_1 - 5x_2 - 2x_3 \leq -400; \\ -x_1 - x_2 - 4x_3 \leq -250; \\ 2x_1 + x_2 + x_3 \leq 300; \\ x_1 + x_2 + x_3 = 150; \\ x_i \geq 0, i = 1, 2, 3. \end{cases}$$

Тепер впровадимо дані цієї задачі у командний рядок вікна **Command Window** середовища **MatLab®** та виконаємо обчислення:

```
>> clear;
c=[20 25 17];
Aneq=[-4 -5 -2; -1 -1 -4; 2 1 1];
bneq=[-400; -250; 300];
Aeq=[1 1 1];
beq=[150];
xmin=zeros(1,3);
xmax=[ ];
[xopt, fval]=linprog(c, Aneq, bneq, Aeq, beq, xmin, xmax)
```

В результаті виконання цієї процедури одержимо оптимальний розв'язок x_{opt} і мінімальне значення $fval$ цільової функції у вигляді:

```
Optimization terminated.
xopt =
    50.0000
     0.0000
    100.0000
fval =
    2.7000e+03
```

6.1.1. Математичні моделі, які приводять до задач лінійного програмування

Моделі на рівні підприємств промисловості

Модель техніко-економічного планування. Для виробництва l видів продукції використовують n видів технологій і виділено k видів ресурсів. Задача полягає у виборі такої технології, щоб за мінімальних затрат виконати план виробництва та використати виділені ресурси.

Введемо позначення:

- j — вид технології;
- n — кількість видів технологій;
- x_j — інтенсивність використання j -ї технології;
- i — вид продукції;
- l — кількість видів продукції;
- a_{ij} — норма випуску i -го виду продукції, використовуючи j -ту технологію з одиничною інтенсивністю;
- Q_i — план випуску i -го виду продукції;
- s — вид ресурсу;

- b_s — кількість ресурсу s -го виду;
 k — кількість всіх видів ресурсів;
 b_{sj} — норма використання s -го виду ресурсу, застосовуючи j -ту технологію одиничної інтенсивності;
 c_j — норма затрат при використанні j -ї технології одиничної інтенсивності.

Тоді математична модель мінімізації затрат на виробництво продукції

$$\sum_{j=1}^n c_j x_j \rightarrow \min, \quad (6.1.3)$$

при обмеженнях на ресурси

$$\sum_{j=1}^n b_{sj} x_j \leq b_s \quad (s = \overline{1, k}), \quad (6.1.4)$$

при виконанні плану виробництва

$$\sum_{j=1}^n a_{ij} x_j = Q_i \quad (i = \overline{1, l}), \quad (6.1.5)$$

де $x_j \geq 0$ ($j = \overline{1, n}$). Задача (6.1.3)–(6.1.5) – задача лінійного програмування мінімізації лінійної форми (6.1.3) при обмеженнях (6.1.4), (6.1.5).

♣ **Зауваження 6.1.2.** Якщо ми хочемо розв'язати задачу максимізації прибутку, то в моделі (6.1.3)–(6.1.5) замість мінімізації затрат на виробництво продукції розглядають максимізації прибутку

$$\sum_{j=1}^n p_j x_j \rightarrow \max, \quad (6.1.6)$$

де p_j – прибуток від використання j -технології одиничної інтенсивності.

♣ **Зауваження 6.1.3.** Якщо допускається перевиконання плану, то замість умови (6.1.5) використовується нерівність

$$\sum_{j=1}^n a_{ij} x_j \geq Q_i \quad (i \in M), \quad (6.1.7)$$

де M – множина тих видів продукції, для яких допускається перевиконання плану.

Максимізація випуску комплектної продукції. У виробництві продукції, що складається з окремих деталей, треба розподілити комплектуючі вироби так, щоб отримати найбільшу кількість комплектів. В моделі техніко-економічного планування використовується критерій максимуму випуску продукції в заданому асортиментному співвідношенні. Отже, перейдемо до моделювання задачі. Для цього введемо позначення:

- j — вид технології;
- n — кількість видів технології;
- x_j — інтенсивність використання j -го виду технології;
- i — вид деталі (комплектуючого виробу);
- l — кількість видів продукуваних виробів;
- l_i — кількість деталей i -го виду потрібних для комплектування одиниці продукції;
- s — вид ресурсу;
- k — кількість видів ресурсів;
- b_s — кількість ресурсу s -го виду;
- a_{ij} — норма випуску деталей i -го виду, використовуючи j -ту технологію одиничної інтенсивності;
- b_{sj} — норма затрат s -го виду сировини, використовуючи j -ту технологію одиничної інтенсивності;
- z — кількість одиниць комплектної продукції.

Добуток $a_{ij}x_j$ дорівнює кількості виготовлених комплектуючих деталей i -го виду при використанні j -ї технології, а сума всіх таких добутків $\sum_{j=1}^n a_{ij}x_j$ дорівнюватиме кількості виготовлених деталей i -го виду. Поділивши цю суму на l_i , отримаємо кількість одиниць z продукції, що виготовляється. З іншого боку, беручи до уваги норми затрат сировини при використанні певного виду технології, отримаємо обмеження на використання сировини. Враховуючи вищесказане, математична модель сформульованої задачі матиме вигляд:

$$z \rightarrow \max;$$

$$\frac{1}{l_i} \sum_{j=1}^n a_{ij}x_j \geq z \quad (i = \overline{1, l});$$

$$\sum_{j=1}^n b_{sj}x_j \leq b_s \quad (s = \overline{1, k});$$

$$x_j \geq 0 \quad (j = \overline{1, n}).$$

Ця модель, як і попередня, виражена у вигляді задачі лінійного програмування, розв'язуючи яку, ми отримаємо оптимальний план використання технологій x_j ($j = \overline{1, n}$) при виробництві комплектної продукції.

Максимізація випуску комплектної продукції з врахуванням можливості цехів. Модель визначає програму випуску деталей цехами, при якій максимізується випуск комплектної продукції. Введемо позначення для вихідних даних:

- i — вид деталей, з яких складається комплектна продукція;
- l — кількість видів деталей;
- r — вид станка (цеху), на якому виготовляються деталі (кожний станок може випускати деталі одного виду);
- R — кількість видів станків (цехів);
- b_r — кількість станків (цехів) r -го виду;
- l_i — кількість деталей i -го виду, потрібних для комплектації одиниці продукції;
- a_{ir} — кількість деталей i -го виду, яке можна виготовити на станку (в цеху) r -го виду за одиницю часу;
- x_{ir} — кількість станків (цехів) r -го виду, які мають випускати деталі i -го виду.

Якщо через z , як і вище, позначити кількість продукції, яку планується виготовити, то математична модель матиме вигляд:

$$z \rightarrow \max;$$

$$\frac{1}{l_i} \sum_{r=1}^R a_{ir} x_{ir} \geq z \quad (i = \overline{1, l});$$

$$\sum_{j=1}^l x_{ir} = b_r \quad (r = \overline{1, R});$$

$$x_{ir} \geq 0 \quad (i = \overline{1, l}, r = \overline{1, R}).$$

Модель виражена у вигляді задачі лінійного програмування. Розв'язавши цю задачу відомими методами (наприклад, симплекс-методом), отримаємо оптимальний план завантаження станків (цехів) з виготовлення деталей.

Модель оптимальних сумішей

Задача полягає у визначенні набору компонентів продуктів, із яких треба отримати суміш із наперед визначеними властивостями з мінімальними затратами. Введемо позначення для вихідних даних:

- j — вид компонента, який є складником суміші;
 m — кількість видів компонент, необхідних для утворення суміші;
 i — вид властивості, яку має мати суміш;
 n — кількість всіх видів властивостей суміші;
 R_i — кількісна характеристика i -го виду властивості;
 a_{ij} — вміст i -го виду властивості в одиниці j -го виду компоненти;
 c_j — вартість одиниці j -го виду компоненти;
 a_j — найменша можлива кількість j -го виду продукту потрібного для суміші;
 b_j — найбільша можлива кількість j -го виду продукту потрібного для суміші;
 x_j — шукана кількість продукту j -го виду необхідного для утворення суміші.

Математична модель: визначити такі $x_j \geq 0$ ($j = \overline{1, m}$), за яких затрати на виготовлення суміші будуть мінімальними:

$$\sum_{j=1}^m c_j x_j \rightarrow \min \quad (6.1.8)$$

у цьому випадку суміш матиме задані властивості:

$$\sum_{j=1}^m a_{ij} x_j \geq R_i, \quad (i = \overline{1, n}); \quad a_j \leq x_j \leq b_j, \quad (j = \overline{1, m}). \quad (6.1.9)$$

Легко бачити, що задача (6.1.8), (6.1.9) є задачею лінійного програмування.

Приклади та завдання. На металургійному комбінаті виготовляється чавун з різного виду шихтового матеріалу. Потрібно визначити оптимальний набір шихтового матеріалу, щоб отримати чавун, який містить визначений вміст сірки, фосфору, марганцю та інших компонент з мінімальними затратами на виробництво. Скласти математичну модель.

Модель оптимального розкрою матеріалів

На багатьох промислових підприємствах при масовому виробництві продукції необхідно отримати найбільш раціональний розкрій матеріалів (дошки, листи металу, руби, прокат, рулони тканин тощо). План розкрою вважається оптимальним, якщо він забезпечує максимальний вихід заготовок з мінімальними відходами. Розглянемо таку задачу.

Підприємство отримує однотипні рулони матеріалів. Треба знайти такий план розкрою рулонів матеріалу на заготовки по ширині, за якого будуть найменші відходи. Домовимося вважати:

- i — вид заготовки;
- m — кількість всіх видів заготовок;
- j — варіант розкрою рулона по ширині;
- n — кількість всіх варіантів розкрою;
- a_i — задана кількість заготовок i -го виду;
- a_{ij} — кількість заготовок i -го виду, яку можна отримати з одного рулона згідно з j -товим варіантом розкрою;
- c_j — відходи, отримані з рулона матеріалу при j -му варіанті розкрою;
- x_j — шукана кількість рулонів, яка буде розкроєна згідно з j -товим варіантом.

Математична модель сформульованої задачі має вигляд:

$$\sum_{j=1}^n c_j x_j \rightarrow \min;$$

$$\sum_{j=1}^n a_{ij} x_j = a_i \quad (i = \overline{1, m})$$

$$x_j \geq 0 \quad (j = \overline{1, n}).$$

Це задача лінійного програмування, для розв'язування якої можна застосувати симплекс метод.

6.1.2. Завдання для самостійного опрацювання

Знайти розв'язки таких задач лінійного програмування:

◆ Завдання 6.1.1.

$$f(x_1, x_2, x_3) = 10x_1 + 16x_2 + 6x_3 \rightarrow \max,$$

$$\begin{cases} 3x_1 + 6x_2 + 3x_3 \leq 100; \\ 2x_1 + 2x_2 + 4x_3 \leq 30; \\ 3x_1 + 4x_2 + 7x_3 \leq 200; \\ x_i \geq 0, \quad i = 1, 2, 3. \end{cases}$$

◆ Завдання 6.1.2.

$$f(x_1, x_2, x_3, x_4) = 2x_1 - x_2 + 3x_3 - 4x_4 \rightarrow \min,$$

$$\begin{cases} x_1 + 2x_2 - 3x_3 + 4x_4 \leq 5; \\ 4x_1 - x_2 + 2x_3 - 3x_4 \geq 2; \\ 2x_1 + 3x_2 - 5x_3 + x_4 = 8; \\ x_1 \leq 0, x_3 \geq 0, x_4 \geq 0. \end{cases}$$

◆ Завдання 6.1.3.

$$f(x_1, x_2, x_3, x_4) = -x_1 + 2x_2 - 3x_3 + x_4 \rightarrow \min,$$

$$\begin{cases} 3x_1 - x_2 + x_3 - x_4 \geq 4; \\ x_1 + 2x_2 - x_3 + 3x_4 = 2; \\ 2x_1 - x_2 + 2x_3 - 5x_4 \leq 6; \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0. \end{cases}$$

◆ Завдання 6.1.4.

$$f(x_1, x_2, x_3, x_4) = -x_1 + 2x_2 - x_3 + x_4 \rightarrow \min,$$

$$\begin{cases} 2x_1 - x_2 - x_3 + x_4 \leq 6; \\ x_1 + 2x_2 + x_3 - x_4 \geq 8; \\ 3x_1 - x_2 + 2x_3 + 2x_4 \leq 10; \\ -x_1 + 3x_2 + 5x_3 - 3x_4 = 15; \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0. \end{cases}$$

◆ Завдання 6.1.5.

$$f(x_1, x_2, x_3) = 2x_1 - 5x_2 - 3x_3 \rightarrow \min,$$

$$\begin{cases} -x_1 + x_2 + x_3 \geq 4; \\ 2x_1 - x_2 + x_3 \leq 16; \\ 3x_1 + x_2 + 2x_3 \geq 18; \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{cases}$$

◆ Завдання 6.1.6.

$$f(x_1, x_2, x_3) = -3x_1 - 5x_2 - 6x_3 \rightarrow \min,$$

$$\begin{cases} 2x_1 + 5x_2 - 7x_3 \leq 12; \\ -4x_1 + 3x_2 + 8x_3 \geq 15; \\ 3x_1 - 2x_2 + 10x_3 \leq 17; \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{cases}$$

◆ Завдання 6.1.7.

$$f(x_1, x_2, x_3) = -2x_1 + x_2 + 5x_3 \rightarrow \max,$$

$$\begin{cases} 4x_1 + 2x_2 + 5x_3 \leq 12; \\ 6x_1 - 3x_2 + 4x_3 = 18; \\ 3x_1 + 3x_2 - 2x_3 \geq 16; \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{cases}$$

◆ Завдання 6.1.8. Для виготовлення трьох видів виробів A , B і C використовується токарне, фрезерне, зварювальне і шліфувальне обладнання. Затрати часу на обробку одного виробу для кожного з типів обладнання вказані в таблиці нижче. У ній же вказано загальний фонд робочого часу кожного з типів використовуваного обладнання, а також прибуток від реалізації одного виробу кожного виду. Потрібно визначити скільки виробів і якого виду слід виготовити підприємству, щоб прибуток від реалізації був максимальним. Скласти математичну модель задачі та знайти її розв'язок.

| Тип обладнання | Затрати часу | | | Фонд робочого часу обладнання (год.) |
|----------------|--------------|-----|-----|--------------------------------------|
| | A | B | C | |
| Токарне | 2 | 4 | 5 | 120 |
| Фрезерне | 1 | 8 | 6 | 180 |
| Зварювальне | 7 | 4 | 5 | 240 |
| Шліфувальне | 4 | 6 | 7 | 360 |
| Прибуток (грн) | 10 | 14 | 12 | |

◆ Завдання 6.1.9. На підприємстві є $R = 5$ типів станків ($r=1, 2, 3, 4, 5$), на яких виготовляють $l = 2$ види деталей ($i=1, 2$). Комплект продукції складається з двох видів деталей 1-го виду ($l_1 = 2$) та однієї деталі 2-го виду ($l_2 = 1$). Кількість станків 1-го виду $b_1 = 5$, 2-го виду $b_2 = 3$, 3-го виду $b_3 = 40$, 4-го виду $b_4 = 9$, 5-го виду $b_5 = 2$. Продуктивність станків з виготовлення деталей (в кількості штук) протягом одного місяця наведена в таблиці:

| Вид деталі | Вид станка | | | | |
|------------|------------|-----|-----|-----|-----|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | 100 | 400 | 20 | 200 | 600 |
| 2 | 15 | 200 | 2,5 | 50 | 250 |

Знайти максимальну кількість комплектної продукції, яку можна виготовити на цих станках за оптимального завантаження.

◆ *Завдання 6.1.10.* На металургійному комбінаті виплавляють чавун із 4 видів шихтового матеріалу. Визначити оптимальний набір шихтового матеріалу, щоб отримати чавун який містить 2 % сірки, 3 % фосфору та 5 % марганцю за найменшої вартості. Дані про вміст кожної з цих компонент в одиниці кожного з видів шихтового матеріалу та вартість одиниці шихтового матеріалу наведено у табл.

| | Вид шихтового матеріалу | | | |
|-----------|-------------------------|-----|-----|-----|
| | 1 | 2 | 3 | 4 |
| Сірка | 0,3 | 0,4 | 0,6 | 0,1 |
| Фосфор | 0,3 | 0,8 | 0,1 | 0,3 |
| Марганець | 0,4 | 0,7 | 0,2 | 0,9 |
| Вартість | 2 | 4 | 2 | 3 |

◆ *Завдання 6.1.11.* Підприємство отримує однотипні рулони шириною 700 см., які потрібно розрізати на заготовки трьох видів: 1-й шириною 230 см, 2-й – 190 см, 3-й – 80 см. План заготовок такий: заготовок першого виду – 60 шт., 2-го – 90 шт., 3-го – 320 шт. План треба виконати з мінімальними відходами. Для цього складають варіанти j - го розкрою рулонів на заготовки і визначають кількість a_{ij} заготовок i - го виду з одного рулона згідно з j - тим варіантом, і відходи c_j . Ці дані наведені в таблиці:

| Варіант розкрою | Кількість заготовок з одного рулона | | | Відходи (см) |
|-----------------|-------------------------------------|-----------|-----------|--------------|
| | 1-го виду | 2-го виду | 3-го виду | |
| 1 | 3 | – | – | 10 |
| 2 | 2 | 1 | – | 50 |
| 3 | 2 | – | 3 | 0 |
| 4 | 1 | 2 | 1 | 10 |
| 5 | 1 | – | 5 | 70 |
| 6 | 1 | 1 | 3 | 40 |
| 7 | 2 | – | 3 | 0 |
| 8 | – | – | 8 | 60 |
| 9 | – | 2 | 4 | 0 |

Знайти оптимальний план розкрою рулонів на заготовки.

◆ *Завдання 6.1.12.* На підприємство надходять однотипні сувої сукна шириною 700 см, які потрібно розкроїти на заготовки трьох видів: 1-й шириною 230 см, 2-й – 190см, 3-й – 80 см. План заготовок такий: 60 штук 1-го виду, 90 – 2-го, 320 – 3-го. Складено 9 варіантів розкрою. Скласти план оптимального розкрою сукна, за якого відходи будуть мінімальними, якщо задано кількість заготовок кожного з трьох видів із

одного сувою сукна для кожного з 9 варіантів, а також відходи при розкроюванні кожним з 9-ти варіантів:

| Варіант розкрою | Кількість заготовок із одного сувою | | | Відходи, см |
|-----------------|-------------------------------------|-----------|-----------|-------------|
| | 1-го виду | 2-го виду | 3-го виду | |
| 1 | 3 | 0 | 0 | 10 |
| 2 | 2 | 1 | 0 | 50 |
| 3 | 2 | 0 | 3 | 0 |
| 4 | 1 | 2 | 1 | 10 |
| 5 | 1 | 0 | 5 | 70 |
| 6 | 1 | 1 | 3 | 40 |
| 7 | 2 | 0 | 3 | 0 |
| 8 | 0 | 0 | 8 | 60 |
| 9 | 0 | 2 | 4 | 0 |

◆ *Завдання 6.1.13.* Підприємство випускає два види продукції за шістьма технологіями, використовуючи два види сировини. В табл. наведено норми використання кожного виду сировини при застосуванні кожної з технологій, норми випуску кожного виду продукції при використанні кожної з технологій одиничної інтенсивності, запас сировини, план випуску продукції кожного виду та норми затрат при застосуванні кожної з технологій одиничної інтенсивності. Знайти оптимальний план використання таких технологій, щоб за мінімальних затрат виконати план виробництва та використати виділену сировину.

Норми використання сировини та випуску продукції

| Вид сировини | Вид технології | | | | | | Запас сировини |
|---------------|----------------|---|---|---|---|---|----------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| 1 | 4 | 1 | 3 | 6 | 2 | 2 | 340 |
| 2 | 3 | 1 | 6 | 1 | 1 | 3 | 460 |
| Вид продукції | План | | | | | | |
| 1 | 5 | 5 | 2 | 3 | 4 | 1 | 280 |
| 2 | 2 | 4 | 8 | 2 | 6 | 9 | 320 |
| Норми затрат | 6 | 3 | 6 | 2 | 5 | 7 | |

◆ *Завдання 6.1.14.* Для виробництва трьох видів продукції використовуються три види сировини. Норми витрат кожної сировини на виробництво одиниці кожної продукції, запас кожної сировини і прибуток

від реалізації одиниці кожної продукції наведені у таблиці.

| Вид сировини | Вид продукції | | | Запас сировини |
|--------------|---------------|----|---|----------------|
| | 1 | 2 | 3 | |
| 1 | 3 | 6 | 3 | 100 |
| 2 | 2 | 2 | 4 | 30 |
| 3 | 3 | 4 | 7 | 200 |
| Прибуток | 10 | 16 | 6 | |

Скласти такий план випуску продукції, щоб загальний прибуток від виробництва був найбільший.

◆ *Завдання 6.1.15.* Для виготовлення двох виробів можна використати 5 типів станків. Комплект складається з двох деталей 1-го виду й однієї деталі 2-го виду. Кількість станків 1-го типу – 5, 2-го – 3, 3-го – 40, 4-го – 9, 5-го – 2. У табл. наведено норми виробництва деталей на станках протягом місяця. Знайти оптимальний план загрузки станків з виготовлення деталей, при якому максимізується випуск комплектної продукції.

| Вид деталі | Вид станка | | | | |
|------------|------------|-----|-----|-----|-----|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | 100 | 400 | 20 | 200 | 600 |
| 2 | 15 | 200 | 2,5 | 50 | 250 |

◆ *Завдання 6.1.16.* На заводі є $R = 3$ групи обладнання ($r = 1, 2, 3$), на якому виготовляють $l = 3$ види продукції ($i = 1, 2, 3$) по $n = 3$ видах технологій ($j = 1, 2, 3$). У таблиці 6.1 подано норми a_{ij}^r затрати машинного часу r -го обладнання при виготовленні одиниці продукції i -го виду по j -й технології, ресурс b_r обладнання r -го виду, вартість затрат c_{ij} на виготовлення одиниці продукції i -го виду виготовлений по j -й технології та плановий об'єм Q_i випуску i -ї продукції. Знайти оптимальний план завантаження виробничих потужностей за критерієм мінімальних затрат, обмежених можливостях обладнання та виконанні планових завдань.

Табл. 6.1. Дані до задачі 6.1.16

| | i=1 | | | a_{ij}^r | | | i=3 | | | Тис.станко-год. |
|--|----------|----------|----------|------------|----------|----------|----------|----------|----------|-----------------|
| | j=1 | j=2 | j=3 | j=1 | j=2 | j=3 | j=1 | j=2 | j=3 | |
| Групи обладнання (г) | | | | | | | | | | |
| г=1 | 2 | 2 | 1 | 3 | 0 | 4 | 3 | 3 | 0 | 40 |
| г=2 | 3 | 1 | 2 | 1 | 2 | 0 | 5 | 6 | 0 | 34 |
| г=3 | 0 | 1 | 3 | 2 | 3 | 1 | 1 | 0 | 0 | 56 |
| Затрати (c_{ij}) 10^3 г.о. | 2 | 5 | 4 | 1 | 3 | 3 | 8 | 6 | 4 | |
| План (Q_i) 10^4 шт. | 2 | | | | | | | | 25 | |
| Випуск продукції (x_{ij}) 10^3 шт. | x_{11} | x_{12} | x_{13} | x_{21} | x_{22} | x_{23} | x_{31} | x_{32} | x_{33} | |

6.2. Задачі нелінійного програмування

Стандартне формулювання задачі нелінійної оптимізації у середовищі MatLab[®] має такий вигляд:

$$\min_x f(x) \text{ за обмежень: } \begin{cases} c(x) \leq 0, \\ ceq(x) = 0, \\ A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub, \end{cases} \quad (6.2.1)$$

де b , beq – вектори; A , Aeq – матриці коефіцієнтів лінійних обмежень відповідно типу нерівностей і рівностей; $c(x)$, $ceq(x)$ – вектор-функції обмежень, відповідно, типу нерівностей та рівностей; $f(x)$, $c(x)$, $ceq(x)$ можуть бути нелінійними. Для розв’язування задач нелінійного програмування у середовищі MatLab[®] слугує функція

`fmincon('cel_func',x0,A,b,Aeq,beq,lb,ub,'nonlin_constr')`

де

- `cel_func` – цільова функція або ім’я m -файлу, що визначає цю функцію;
- `x0` – масив-стовпчик, який задає початкове значення;
- `A` – матриця коефіцієнтів лінійних обмежень нерівностей \leq ;
- `b` – стовпчик правих частин лінійних обмежень типу \leq ;
- `Aeq` – матриця коефіцієнтів обмежень рівностей;
- `beq` – стовпчик правих частин обмежень типу рівностей;
- `lb` – стовпчик обмежень на мінімальне значення змінних;
- `ub` – стовпчик обмежень на максимальне значення змінних;
- `nonlin_constr` – ім’я m -файлу (функції), що визначає нелінійні обмеження.

Щоб розв’язати задачу нелінійного програмування у середовищі MatLab[®], треба її привести до стандартного вигляду:

- для цільової функції формулюється задача на знаходження мінімуму;

- всі лінійні обмеження звести до стандартного вигляду як у задачі лінійного програмування (див. вище);
- права частина нелінійних обмежень має бути нулем, де всі нелінійні обмеження звести до нерівностей \leq .

★ **Приклад 6.2.1.** Розв'язати задачу нелінійного програмування:

$$f(x_1, x_2, x_3) = \frac{e^{x_1}}{x_2 e^{x_3}} \rightarrow \min,$$

за обмежень

$$\begin{aligned} x_1 + x_2 + x_3 &\geq 50; \\ 3x_1 + 2x_2 &\leq 600; \\ x_2 &\geq 10; \\ x_1 &= 2x_2; \\ x_1x_2 + x_2x_3 &\leq 2000; \\ x_1x_2 &\geq 100; \\ x_2^2 + x_3^2 &= 1000 \\ x_i &\geq 0, \quad i = 1, 2, 3. \end{aligned}$$

Створимо m-файл у редакторі під назвою `cel_fun`:

```
function f=cel_fun(x)
f=exp(x(1))./(x(2).*exp(x(3)));
```

У редакторі створюємо m-файл під назвою `nonlin_rest`, який реалізує нелінійні обмеження. Цей m-файл має зображати функцію, яка має два вихідні параметри: перший – масив нелінійних обмежень типу нерівностей, другий – масив нелінійних обмежень рівностей (власне за зазначеним порядком):

```
function [neq,eq]=nonlin_rest(x)
neq=[x(1).*x(2)+x(2).*x(3)-2000;-x(1).*x(2)+100];
eq=[x(2).^2+x(3).^2-1000];
```

Тут `neq` – масив лівих частин обмежень нерівностей, і відповідно, `eq` – рівностей. Якщо у задачі нелінійного програмування немає однієї з умов, то на їхньому місці у зазначеному записати форматі функції обчислення розв'язку необхідно вказати порожній масив: `[]`.

Накінець, для розв'язування зазначеної задачі створимо файл-сценарій у редакторі `Editor` середовища MatLab®, який запишемо під

іменем nonlinProgr:

```
x0=[1,1,1];
a=[-1,-1,-1;3,2,0;0,-1,0];
b=[-50;600;-10];
ar=[1,-2,0];
br=[0];
xmin=[0;0;0];
[xMin,fVal,k]=fmincon('cel_fun',x0,a,b,ar,br,xmin,[],
'nonlin_rest')
```

Залишається у Command Window ввести назву файлу-сценарію та дати команду на виконання

```
>> nonlinProgr
```

Система поверне відповідь:

[Local minimum found that satisfies the constraints.](#)

Optimization completed because the objective function is non-decreasing in [feasible directions](#), to within the default value of the [function tolerance](#), and constraints are satisfied to within the default value of the [constraint tolerance](#).

[<stopping criteria details>](#)

```
xMin =
    20.0604    10.0302    29.9899

fVal =
    4.5400e-06

k =
    1
```

де $xMin = \{20.0604, 10.0302, 29.9899\}$ – шуканий розв’язок, $fVal = 4.5400e-06$ – мінімальне значення цільової функції, $k = 1$ – код результату: 1 свідчить про те, що розв’язок обчислено успішно.

6.2.1. Завдання для самостійного опрацювання

Задаючи початкове значення, розв'язати задачі нелінійної оптимізації:

◆ Завдання 6.2.1.

$$f(x_1, x_2, x_3) = x_1x_2 + 2x_1x_3 + 2x_2x_3 \rightarrow \min,$$

за обмежень:

$$\begin{cases} x_1x_2x_3 = 100; \\ x_i \geq 0, i = 1, 2, 3. \end{cases}$$

◆ Завдання 6.2.2.

$$f(x_1, x_2) = x_1 - 0, 2x_1^2 + 2x_2 - 0, 2x_2^2 \rightarrow \max,$$

за обмежень:

$$\begin{cases} 13x_1 + 6x_2 \leq 90; \\ 8x_1 + 11x_2 \leq 88; \\ x_i \geq 0, i = 1, 2. \end{cases}$$

◆ Завдання 6.2.3.

$$f(x_1, x_2) = -x_1 + x_2 \rightarrow \min,$$

за обмежень:

$$\begin{cases} x_1^2 + x_2^2 \leq 4, \\ x_1^2 + (x_1 - 2)^2 \geq 1. \end{cases}$$

◆ Завдання 6.2.4.

$$f(x_1, x_2) = \frac{ax_1^2}{2} + \frac{bx_2^2}{2} \rightarrow \text{extr},$$

за обмежень:

$$\begin{cases} x_1^3 + x_2^3 \leq 1, \\ x_1^2 + x_2^2 \geq 1, \\ (0 < a < b). \end{cases}$$

◆ Завдання 6.2.5.

$$f(x_1, x_2) = x_1^2 + (x_2 - 1)^2 \rightarrow \min,$$

за обмежень:

$$\begin{cases} x_1^2 + 4x_2^2 - 4 \leq 0, \\ 2x_1^2 + x_2 - 2 \geq 0, \\ -x_1 + 2x_2 \leq 0. \end{cases}$$

◆ *Задача 6.2.6.*

$$f(x_1, x_2) = x_1 \rightarrow \max,$$

за обмежень:

$$\begin{cases} 2x_1^2 + x_2 - 1 \geq 0 \\ (x_1 - 1)^2 + x_2^2 - 1 \geq 0, \\ x_1 + x_2 - 1 \leq 0. \end{cases}$$

◆ *Задача 6.2.7.*

$$f(x_1, x_2) = -x_1^3 + x_2^3 + 2x_1^2 + x_1 - x_2 \rightarrow \max,$$

за обмежень:

$$\begin{cases} x_1 + 2x_2 \leq 2, \\ x_1 \geq 0, \\ x_2 \geq 0. \end{cases}$$

6.3. Метод Runge-Kutta розв'язування початкової задачі для звичайного диференціального рівняння першого порядку

6.3.1. Загальне формулювання методів

Розглянемо задачу Cauchy для одного рівняння

$$\frac{du}{dt} = f(t, u), \quad t > 0, \quad u(0) = u_0. \quad (6.3.1)$$

Явний m -кроковий метод Runge-Kutta такий. Припустимо, що розв'язок $y_n = y(t_n)$ уже відомий. Задаються числові коефіцієнти a_i , b_{ij} , $i = 1, 2, \dots, m$, $j = 1, 2, \dots, m-1$, σ_i , $i = 1, 2, \dots, m$, і послідовно обчислюються функції

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + a_2\tau, y_n + b_{21}\tau k_1), \\ k_3 &= f(t_n + a_3\tau, y_n + b_{31}\tau k_1 + b_{32}\tau k_2), \\ &\dots\dots\dots \\ k_m &= f(t_n + a_m\tau, y_n + b_{m1}\tau k_1 + b_{m2}\tau k_2 + \dots + b_{mm-1}\tau k_{m-1}). \end{aligned}$$

Тепер із формули

$$\frac{y_{n+1} - y_n}{\tau} = \sum_{i=1}^m \sigma_i k_i \quad (6.3.2)$$

знаходиться нове значення $y_{n+1} = y(t_{n+1})$.

Коефіцієнти a_i , b_{ij} , σ_i вибираються з міркувань точності. Наприклад, для того щоб рівняння (6.3.2) було апроксимацією вихідного рівняння (6.3.1), треба вимагати, щоб $\sum_{i=1}^m \sigma_i = 1$. Зауважимо, що методи Runge-Kutta при $m > 5$ не використовуються.

Розглянемо детальніше окремі методи. При $m = 1$ одержимо *схему Euler'a*. Введемо за змінною t *рівномірну сітку* з кроком $\tau > 0$, тобто розглянемо множину точок

$$\omega_\tau = \{t_n = n\tau, \quad n = 0, 1, 2, \dots\}.$$

Позначимо через $u(t)$ точний розв'язок задачі (6.3.1), а через $y_n = y(t_n)$ – наближений розв'язок. Зазначимо, що наближений розв'язок є

сітковою функцією, тобто визначений тільки у точках сітки ω_τ .

За методом Euler'a рівняння (6.3.1) заміняємо різницеvim рівнянням

$$\frac{y_{n+1} - y_n}{\tau} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \dots, \quad y_0 = u_0. \quad (6.3.3)$$

Розв'язок цього рівняння записується у явному вигляді за рекурентною формулою

$$y_{n+1} = y_n + \tau f(t_n, y_n), \quad n = 0, 1, 2, \dots, \quad y_0 = u_0.$$

При використанні наближених методів важливим є питання збіжності. Поняття збіжності наближеного метода можна сформулювати по-різному. Стосовно різницеvim методів, до яких належить метод Euler'a, найбільше поширення отримало поняття *збіжності при $\tau \rightarrow 0$* , яке означає таке. Фіксуємо точку t і побудуємо послідовність сіток ω_τ таких, що $\tau \rightarrow 0$ і $t_n = \tau n = t$ (тоді необхідно $n \rightarrow \infty$). Тоді кажемо, що метод (6.3.3) є збіжний у точці t , якщо $|y_n - u(t_n)| \rightarrow 0$ при $\tau \rightarrow 0$, $t_n = t$.

Метод *збігається на* відрізьку $(0, T]$, якщо він є збіжний у кожній точці $t \in (0, T]$.

Кажуть, що метод має p -й порядок точності, якщо існує число $p > 0$ таке, що $|y_n - u(t_n)| = O(\tau^p)$ при $\tau \rightarrow 0$. Можна довести, що метод Euler'a має перший порядок точності, тобто $|y_n - u(t_n)| = O(\tau)$ [81].

Симетрична схема методу Euler'a полягає у заміні рівняння (6.3.1) різницеvim рівнянням

$$\frac{y_{n+1} - y_n}{\tau} - \frac{1}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})) = 0, \quad n = 0, 1, 2, \dots, \quad y_0 = u_0. \quad (6.3.4)$$

Порівняно з методом Euler'a цей метод складніший у реалізації, позаяк нове значення y_{n+1} визначається через відоме y_n шляхом розв'язування рівняння

$$y_{n+1} - 0.5\tau f(t_{n+1}, y_{n+1}) = F_n,$$

де $F_n = y_n + 0.5\tau f(t_n, y_n)$. Із цієї причини цей метод називається *неявним*. Проте метод (6.3.4) має ту перевагу порівняно з методом (6.3.3), що дає вищий порядок точності, а саме $O(\tau^2)$.

Наведені приклади ілюструють простіші випадки *різницеvim методів*, або, як їх ще називають, *різницеvim схем*. *Методи Runge-Kutta* відрізняються від різницеvim методів тим, що у них допускається обчислення правої частини рівняння $f(t, u)$, не тільки у точках сітки, але у деяких проміжних точках.

Методи Runge-Kutta другого порядку точності

До методу Runge-Kutta другого порядку точності належать:

– модифікований метод Euler'а:

$$f_1 = f(x_n, y_n), f_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f_1\right), y_{n+1} = y_n + h \cdot f_2;$$

| | | | |
|-----|-----|---|---|
| 0 | 0 | | |
| 1/2 | 1/2 | 0 | |
| | 0 | | 1 |

– метод Euler'а із перерахунком:

$$f_1 = f(x_n, y_n), f_2 = f(x_n + h, y_n + hf_1), y_{n+1} = y_n + \frac{h}{2}(f_1 + f_2);$$

| | | | |
|---|-----|-----|--|
| 0 | 0 | | |
| 1 | 1 | 0 | |
| | 1/2 | 1/2 | |

Методи Runge-Kutta третього порядку точності**1. Метод Heun'а**

$$f_1 = f(x_n, y_n), f_2 = f\left(x_n + \frac{h}{3}, y_n + \frac{h}{3}f_1\right),$$

$$f_3 = f\left(x_n + \frac{2h}{3}, y_n + \frac{2h}{3}f_2\right), y_{n+1} = y_n + \frac{h}{4}(f_1 + 3f_3).$$

Таблиця Butcher'а методу Heun'а

| | | | |
|-----|-----|-----|-----|
| 0 | 0 | | |
| 1/3 | 1/3 | 0 | |
| 2/3 | 0 | 2/3 | 0 |
| | 1/4 | 0 | 1/4 |

Інші методи Runge-Kutta третього порядку, які використовуються при наближеному інтегруванні задачі Cauchy:

2.

$$f_1 = f(x_n, y_n), \quad f_2 = f\left(x_n + \frac{2h}{3}, y_n + \frac{2h}{3}f_1\right),$$

$$f_3 = f\left(x_n + \frac{2h}{3}, y_n - \frac{h}{3}f_1 + hf_2\right), \quad y_{n+1} = y_n + \frac{h}{4}(f_1 + 2f_2 + f_3).$$

Таблиця Butcher'a

| | | | |
|-----|------|-----|-----|
| 0 | 0 | | |
| 2/3 | 2/3 | 0 | |
| 2/3 | -1/3 | 1 | 0 |
| | 1/4 | 2/4 | 1/4 |

3.

$$f_1 = f(x_n, y_n), \quad f_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f_1\right),$$

$$f_3 = f\left(x_n + h, y_n - hf_1 + 2hf_2\right), \quad y_{n+1} = y_n + \frac{h}{6}(f_1 + 4f_2 + f_3).$$

Таблиця Butcher'a

| | | | |
|-----|-----|-----|-----|
| 0 | 0 | | |
| 1/2 | 1/2 | 0 | |
| 1 | -1 | 2 | 0 |
| | 1/6 | 4/6 | 1/6 |

Класичний метод Runge-Kutta четвертого порядку

$$f_1 = f(x_n, y_n), \quad f_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f_1\right),$$

$$f_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f_2\right), \quad f_4 = f(x_n + h, y_n + hf_3),$$

$$y_{n+1} = y_n + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4).$$

Таблиця Butcher'a

| | | | | |
|-----|-----|-----|-----|-----|
| 0 | 0 | | | |
| 1/2 | 1/2 | 0 | | |
| 1/2 | 0 | 1/2 | 0 | |
| 1 | 0 | 0 | 1 | |
| | 1/6 | 2/6 | 2/6 | 1/6 |

6.3.3. Оцінка похибки чисельного розв'язку

Важливим питанням чисельного інтегрування задачі Cauchy для звичайного диференціального рівняння є оцінка похибки одержаного розв'язку. Безпосереднє обчислення у вузлах сітки норми різниці двох розв'язків, аналітичного і чисельного, не завжди можливо, оскільки диференціальна задача у загальному випадку не інтегрується до кінця. Одержати оцінку допущеної похибки можна, якщо відомий порядок точності чисельного методу, який застосовується під час розв'язування задачі. У цьому випадку спочатку треба розв'язати різницеву задачу на сітці з кроком $2h$, а після повторити на сітці з кроком h . Позначимо сіткову функцію, одержану на сітці з кроком $2h$, через $y^{(2h)}$, а на сітці h – $y^{(h)}$. Слід розв'язку диференціальної задачі (тобто точний розв'язок задачі) на сітці $2h$ нехай y^* , а слід $\bar{y}^{(h)}$ на сітці $2h$ – $y^{(h)}$. Тоді правильні рівності

$$\begin{aligned} y^{(2h)} &= y^* + C(2h)^k, \\ y^{(h)} &= y^* + Ch^k, \end{aligned} \quad (6.3.7)$$

де k – порядок наближення вибраного методу. Віднявши від першого рівняння (6.3.7), одержимо

$$y^{(2h)} - y^{(h)} \approx Ch^k(2^k - 1),$$

або

$$Ch^k \approx \frac{y^{(2h)} - y^{(h)}}{2^k - 1}. \quad (6.3.8)$$

Увівши у скінченновимірному просторі, відповідному до сітки $2h$, одну з можливих норм, отримаємо оцінку

$$\|y^{(h)} - y^*\| \approx \frac{\|y^{(2h)} - y^{(h)}\|}{2^k - 1} \leq \varepsilon. \quad (6.3.9)$$

Тут ε – похибка наближення. Якщо умова (6.3.9) не виконується, тоді вдвічі збільшуємо кількість вузлів сітки, одночасно зменшуючи крок, і знову перевіряємо умову (6.3.9) на новій сітці. Так повторюємо доти, доки на буде виконана умова (6.3.9).

6.4. Приклад застосування методу Runge-Kutta

Розглянемо процедуру, яка реалізує метод Runge-Kutta четвертого порядку з фіксованим кроком. Відповідний скрипт-файл RuKu4.m має

ВИГЛЯД

```
function ret=RuKu4(y,x)
% RK4 standard method with fixed step h
% the rhs of the equation is given by zeta.m
global h
xm=x+h/2;
k1=h*zeta(x,y);
k2=h*zeta(xm,y+k1/2);
k3=h*zeta(xm,y+k2/2);
k4=h*zeta(x+h,y+k3);
ret=y+(k1+k4+2.0*(k2+k3))/6.0;
end
```

Розглянемо початкову задачу для звичайного диференціального рівняння першого порядку, яке не інтегрується у квадратурах

$$\begin{cases} y'(x) = x^2 + y^2(x), \\ y(0) = 0. \end{cases} \quad (6.4.1)$$

Порівняємо чисельні розв'язки задачі (6.4.1), одержані за допомогою методу Runge-Kutta та функції `ode45` середовища MatLab®. З цією метою визначимо функцію правої частини рівняння (6.4.1) за допомогою скрипт-файлу `zeta.m`:

```
function z=zeta(x,y)
%rhs function tested method Runge-Kutta
z=x^2+y^2;
end
```

Розглядаємо задачу (6.4.1) на проміжку $[0, L]$. Тоді відповідний скрипт-файл, за допомогою якого тестуватимемо, назвемо `testode1.m`:

```
% file testode1.m
clear
global h
L=input('L: ');
x0=0; y0=0;
[t,w]=ode45('zeta',[x0 L],y0);
N=input('N: ');
h=(L-x0)/N;
for i=1:N+1
x(i)=x0+(i-1)*h;
end
```

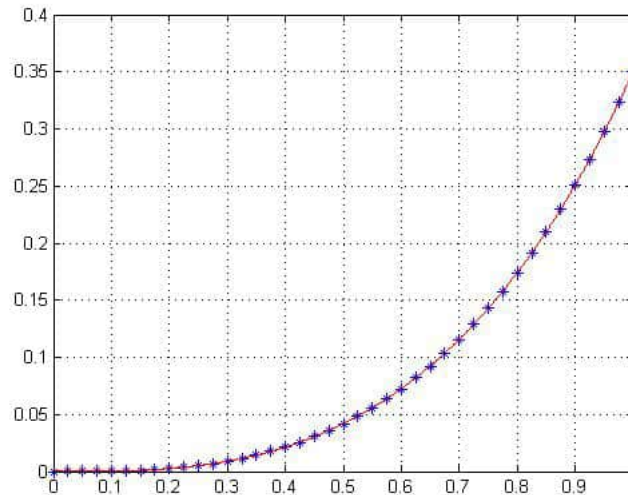


Рис. 6.1. Графіки чисельних розв'язків одержаних методом Runge-Kutta та ode45

```
y(1)=y0;
for i=1:N
y(i+1)=RuKu4(y(i),x(i));
end
format long
yrk4=y(N+1) % y(L) approximated by RK4
M=length(w)
y45=w(M) % y(L) approximated by ode45
plot(t,w,'*'); grid
hold on
plot(x,y,'r-')
hold off
```

Чисельні результати порівняємо графічно та значення отриманого розв'язку на кінці проміжку $y(L)$ для $L = 1$. Із графіків 6.1 робимо висновок, що розв'язки майже ідентичні.

Значення розв'язку у крайній точці $L = 1$, отриманий за допомогою ode45 дорівнює $y(L) = 0.35023184134841$ для $M = 41$, де $M - 1$ число підінтервалів. Застосування RuKu1 повертає такі значення: $y(L) = 0.35023184453449$ для $N = 100$ та $y(L) = 0.35023184431678$ для $N = 1000$, де N - число підінтервалів. Результат, отриманий за допомогою RuKu1, добре узгоджується з відповідним результатом, отриманим

за допомогою `ode45`, позаяк початкова задача (6.4.1) не є жорсткою.

6.4.1. Завдання для самостійного опрацювання

На заданому інтервалі на рівномірній сітці з одинадцяти вузлів із зазначеною похибкою $\varepsilon = 10^{-4}$ знайти указаним методом розв'язок задачі Cauchy.

◆ *Завдання 6.4.1.* Метод Runge-Kutta 4 порядку точності: $x \in (1, 2)$,

$$\begin{cases} 2x^2yy' + y^2 = 2x^3 + x^2, \\ y(1) = 1. \end{cases}$$

◆ *Завдання 6.4.2.* Метод Runge-Kutta 3 порядку: $x \in (0, 1)$,

$$\begin{cases} (x^2y - 1)y' + xy^2 - 1 = 0, \\ y(0) = 0. \end{cases}$$

◆ *Завдання 6.4.3.* Метод Euler'а з перерахунком: $x \in (1, 1.1)$,

$$\begin{cases} (6xy + x^2 + 3)y' + 3y^2 + 2xy + 2x = 0, \\ y(1) = -1. \end{cases}$$

◆ *Завдання 6.4.4.* Модифікований метод Euler'а: $x \in (1, 2)$,

$$\begin{cases} (xy - x^2)y' + y^2 - 3xy - 2x^2 = 0, \\ y(1) = 1 + \sqrt{2}. \end{cases}$$

◆ *Завдання 6.4.5.* Метод Euler'а з перерахунком: $x \in (1, 1.2)$,

$$\begin{cases} x(2x^2y \cdot \ln(y) + 1)y' = 2y, \\ y(1) = 1. \end{cases}$$

◆ *Завдання 6.4.6.* Модифікований метод Euler'а: $x \in (0, 1)$,

$$\begin{cases} y' - xy^2 - 3xy = 0, \\ y(0) = -3. \end{cases}$$

◆ *Завдання 6.4.7.* Метод Runge-Kutta 3 порядку: $x \in (1, 1.2)$,

$$\begin{cases} 2xyy' - y^2 + 5x = 0, \\ y(1) = 1. \end{cases}$$

◆ *Завдання 6.4.8.* Метод Runge-Kutta 4 порядку точності: $x \in (1, 2)$,

$$\begin{cases} xy' + xy^2 - y = 0, \\ y(1) = 2. \end{cases}$$

◆ *Завдання 6.4.9.* Метод Runge-Kutta 3 порядку: $x \in (1, 2)$,

$$\begin{cases} x(2x - 1)y' + y^2 - (4x + 1)y + 4x = 0, \\ y(1) = 2. \end{cases}$$

◆ *Завдання 6.4.10.* Метод Runge-Kutta 4 порядку точності: $x \in (1, 2)$,

$$\begin{cases} 2x^2y' - 2y^2 - 3xy + 2x = 0, \\ y(1) = \frac{1}{2}. \end{cases}$$

◆ *Завдання 6.4.11.* Модифікований метод Euler'а: $x \in (1, 2)$,

$$\begin{cases} xy' - xy^2 - (2x^2 + 1)y - x^3 = 0, \\ y(1) = -3. \end{cases}$$

◆ *Завдання 6.4.12.* Метод Euler'а з перерахунком: $x \in (1, 2)$,

$$\begin{cases} x^2(y' + y^2) + 4xy + 2 = 0, \\ y(1) = -1. \end{cases}$$

◆ *Завдання 6.4.13.* Метод Runge-Kutta 4 порядку точності: $x \in (1, 2)$,

$$\begin{cases} (y - x^2)y' = x, \\ y(1) = \frac{3}{2}. \end{cases}$$

◆ *Завдання 6.4.14.* Метод Runge-Kutta 3 порядку: $x \in (1, 2)$,

$$\begin{cases} yy' + y^2 + 4x(x + 1) = 0, \\ y(1) = 12. \end{cases}$$

◆ *Завдання 6.4.15.* Модифікований метод Euler'а: $x \in (1, 2)$,

$$\begin{cases} x^3y' - x^4y^2 + x^2y + 20 = 0, \\ y(1) = 4. \end{cases}$$

◆ *Завдання 6.4.16.* Метод Euler'а з перерахунком: $x \in (2, 3)$,

$$\begin{cases} x^2(x - 1)y' - y^2 - x(x - 2)y = 0, \\ y(2) = 4. \end{cases}$$

6.5. Інтерполяція

6.5.1. Формулювання задачі інтерполяції

Сформулюємо задачу інтерполяції: на проміжку $[a, b]$ задані $n + 1$ різних точок x_0, x_1, \dots, x_n , які називаються *вузлами інтерполяції*, а також значення функції $y = f(x)$ у тих точках

$$f(x_0) = y_0, f(x_1) = y_1, \dots, f(x_n) = y_n. \quad (6.5.1)$$

Задача інтерполяції полягає у визначенні наближених значень заданої функції у точках, які не є вузлами, та оцінка похибки наближених значень. Задача полягає у визначенні такої функції $F(x)$, яка називається *інтерполяційною функцією*, яка у вузлах інтерполяції набуває ті самі значення, що і функція $y = f(x)$; див. рис. 6.2.

Можна сказати, що інтерполяція у певному сенсі є задачею оберненою до табулювання функції. При табулюванні, маючи аналітичний явний вигляд функції, будуємо таблицю значень функції у заданих точках, тоді як розв'язуючи задачу інтерполяції на підставі табличних значень функції треба "відновити" аналітичний вигляд функції.

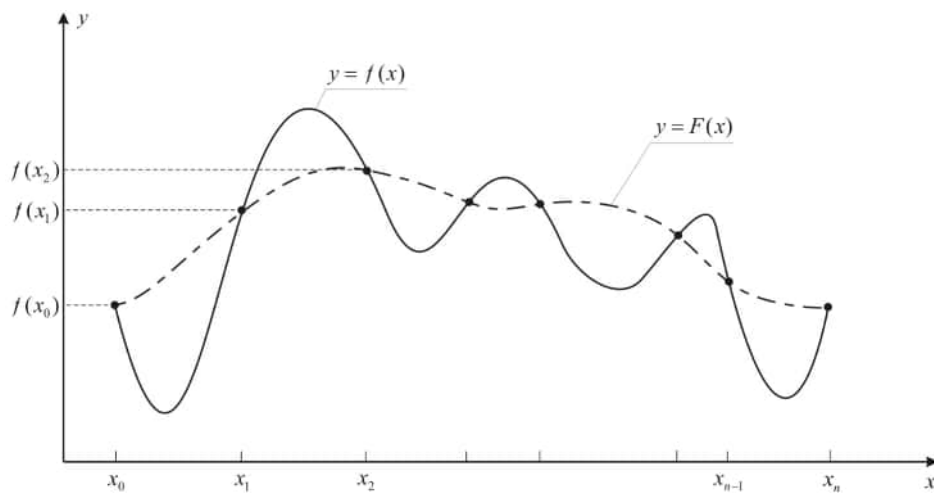


Рис. 6.2. Інтерполяція

Інтерполяційні функції найчастіше будуються у певному визначеному наперед вигляді. Ними можуть бути, наприклад, алгебричні многочлени, тригонометричні многочлени, сплайни тощо. Залежно від вигляду інтерполяційної функції задача інтерполяції може не мати жодного розв'язку

Отже, система (6.5.1) має єдиний розв'язок і значення a_i за формулою Крамер-а обчислюються так:

$$a_i = \frac{1}{D} \sum_{j=0}^n y_j D_{ij}, \quad (6.5.7)$$

де D_{ij} ($j = \overline{0, n}$) – алгебричні доповнення елементів i -го стовпця матриці A .

Із теореми Крамер'а випливає, що існує многочлен вигляду (6.5.2), для якого виконуються умови (6.5.1) і цей многочлен є єдиним. Степінь многочлена є не більший ніж n (якщо $a_0 = 0$, то очевидно одержимо многочлен нижчого степеня).

□

6.5.3. Інтерполяційна формула Lagrange'а

Підставивши (6.5.7) у (6.5.2) та згрупувавши доданки за однакових y_i , одержимо

$$W_n(x) = y_0 \Phi_0(x) + y_1 \Phi_1(x) + \dots + y_n \Phi_n(x), \quad (6.5.8)$$

де функції $\Phi_0(x), \Phi_1(x), \dots, \Phi_n(x)$ є многочленами степеня щонайбільше n .

Як і вище, припустимо, що вузли інтерполяції довільно розміщені на $[a, b]$. Побудуємо многочлен $W_n(x)$ степеня не вище n у вигляді (6.5.8). Для кожного $i = 0, 1, \dots, n$ справджуються рівності

$$W_n(x_i) = y_0 \Phi_0(x_i) + y_1 \Phi_1(x_i) + \dots + y_n \Phi_n(x_i). \quad (6.5.9)$$

Звідки одержуємо таке:

$$\Phi_j(x_i) = \begin{cases} 0 & \text{якщо } j \neq i, \\ 1 & \text{якщо } j = i. \end{cases} \quad (6.5.10)$$

Щоб визначити функції $\Phi_j(x)$, треба побудувати многочлен степеня n , який у точках $x_0, x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n$ тотожно дорівнює нулю, а в точці x_j дорівнює одиниці. Звідси

$$\Phi_j(x) = \lambda(x - x_0)(x - x_1) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n), \quad (6.5.11)$$

а оскільки $\Phi_j(x_j) = 1$, то

$$1 = \lambda(x_j - x_0)(x_j - x_1) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n). \quad (6.5.12)$$

Визначивши λ із (6.5.12) та підставивши у (6.5.11), одержимо

$$\Phi_j(x) = \frac{(x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}, \quad (6.5.13)$$

тобто

$$\begin{aligned} W_n(x) &= y_0 \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{(x_0 - x_1) \dots (x_0 - x_n)} + \\ &+ y_1 \frac{(x - x_0)(x - x_2) \dots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \dots (x_1 - x_n)} + \dots + \\ &+ y_n \frac{(x - x_0)(x - x_1) \dots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1})} = \\ &= \sum_{j=0}^n y_j \frac{(x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}. \end{aligned} \quad (6.5.14)$$

Уведемо позначення

$$\omega_n(x) = (x - x_0)(x - x_1) \dots (x - x_n), \quad (6.5.15)$$

за допомогою якого (6.5.14) запишеться у вигляді

$$W_n(x) = \sum_{j=0}^n y_j \frac{\omega_n(x)}{(x - x_j) \left(\frac{\omega_n(x)}{x - x_j} \right) \Big|_{x=x_j}} = \sum_{j=0}^n y_j \frac{\omega_n(x)}{(x - x_j) \omega'_n(x_j)}, \quad (6.5.16)$$

де $y_j = y(x_j)$, а $\omega'_n(x_j)$ є значенням похідної многочлена $\omega_n(x)$ у вузлі x_j (у якому многочлен $\omega_n(x)$ набуває нульове значення), що неважко довести, взявши до уваги формулу Тейлора в околі x_j .

Одержаний многочлен задовольняє всі умови інтерполяції. Формула (6.5.14) називається *інтерполяційною формулою Lagrange'a*. Врахувавши доведене у попередньому підрозділі Твердження 6.5.1, одержимо, що такий многочлен степеня найвище n є єдиним.

6.6. Апроксимація

6.6.1. Задача апроксимації

У попередньому підрозділі 6.5. була розглянута задача інтерполяції, мета якої полягає у побудові деякої інтерполюючої функції $F(x)$, значення якої на цій дискретній множині аргументів збігаються з значеннями інтерпольованої функції $f(x)$. Така задача має свої недоліки. При інтерполяції многочленами велика кількість вузлів інтерполяції потребує побудови многочлена високого порядку. Крім того, досить часто маємо справу з функцією, значення якої на дискретній множині аргументів визначено емпірично (є результатом спостережуваних величин), а це означає, що ці дані не точні, а тільки наближені. Тому побудова функції, яка б на дискретній множині аргументів приймала наближені значення, немає сенсу.

Розглянемо більш загальну задачу, в якій умова, щоб ця та шукана функції набували рівні значення на дискретній множині аргументів, не обов'язково виконується. Функцію $f(x)$, яка є відомою чи задана таблицею значень, будемо *апроксимувати* іншою функцією $F(x)$, яка називається *апроксимуючою функцією* або *наближенням функції* $f(x)$. Очевидно, таке наближення приводить до появи похибки, яку називають *похибкою апроксимації* і задача оцінки цієї похибки, а також її величина мають суттєвий вплив на вибір методу апроксимації.

Якщо множина, на якій визначаємо похибку апроксимації, є дискретною множиною, то таку апроксимацію називаємо *дискретною*, якщо ж ця множина є проміжком, то таку апроксимація називається *інтегральною*.

Нехай $f(x)$ – функція, яку маємо за завдання апроксимувати, \mathbf{X} – деякий нормований лінійний простір (скінчено або нескінченно вимірний; $f \in \mathbf{X}$), а \mathbf{X}_m – m -вимірний лінійний підпростір простору \mathbf{X} .

Апроксимація функції $f(x)$ полягає на визначенні таких коефіцієнтів a_0, a_1, \dots, a_m функції

$$F(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \dots + a_m\varphi_m(x), \quad (6.6.1)$$

де $\varphi_0, \varphi_1, \dots, \varphi_m$ є базовими функціями $m + 1$ вимірного лінійного підпростору \mathbf{X}_{m+1} так, щоб функція $F(x)$ задовольняла певні умови, наприклад, мінімізувала норму різниці $\|f(x) - F(x)\|$.

Апроксимація, визначена формулою (6.6.1), називається *лінійною апроксимацією* (формула (6.6.1) також називається *узагальненим многочленом*).

Якщо функція $f(x)$ визначена на дискретній множині точок, тоді значення функції $f(x_i)$ у точках множини можна інтерпретувати як координати вектора \mathbf{f} , а отже, розглядається норма

$$\|\mathbf{f}\| = \left(\sum_{i=0}^n (f(x_i))^2 \right)^{\frac{1}{2}}.$$

Задача найкращої апроксимації при вибраних базових функціях $\varphi_k(x)$ зводиться до визначення коефіцієнтів a_k так, щоб мінімізувати вираз

$$\|f(x) - (a_0\varphi_0(x) + a_1\varphi_1(x) + \dots + a_m\varphi_m(x))\| \quad (6.6.2)$$

та отримати єдиний можливий розв'язок задачі визначення коефіцієнтів a_k .

Розглянемо середньоквадратичну апроксимацію.

6.6.2. Середньоквадратична апроксимація

Для функції $f(x)$, визначеній на проміжку $[a, b]$, визначаємо мінімум інтеграла

$$\|F(x) - f(x)\| = \int_a^b w(x) [F(x) - f(x)]^2 dx, \quad (6.6.3)$$

а для функції $f(x)$, визначеній на дискретній множині аргументів, визначаємо мінімум суми (*методом найменших квадратів*)

$$\|F(x) - f(x)\| = \sum_{i=0}^n w(x_i) [F(x_i) - f(x_i)]^2, \quad w(x_i) \geq 0 \quad (i = \overline{0, n}). \quad (6.6.4)$$

Розглянемо середньоквадратичну апроксимацію у випадку, коли апроксимована функція $f(x)$ визначена на дискретній множині аргументів \mathbf{X} . Ідея такої апроксимації показана на рис. 6.3.

Нехай функція $y = f(x)$ у точках x_0, x_1, \dots, x_n множини \mathbf{X} набуває значення y_0, y_1, \dots, y_n . Ці значення наближені, із певними похибками (наприклад, експериментальні значення, які отримані з деяким наближенням – приладам притаманна деяка точність), причому ці похибки можуть бути досить значними, що суттєво впливає на якість апроксимації. Завдання полягає у визначенні такої гладкої функції $F(x)$, яка є наближенням функції $f(x)$, тобто на основі дискретних даних, які містять похибки, функції $f(x)$ отримати гладку функцію, яка з великою ймовірністю на малу величину відхиляється від апроксимованої функції

між вузлами, та у самих вузлах x_0, x_1, \dots, x_n за припущення, що функція $f(x)$ також є гладкою.

Припустимо, що вибраний базис $\varphi_j(x)$ ($j = \overline{0, m}$) підпростору \mathbf{X}_m . Треба визначити такий узагальнений многочлен $F(x)$, який є найкращим середньоквадратичним наближенням функції $f(x)$ на сітці вузлів $\mathbf{X} = (x_j)$, тобто функцію вигляду

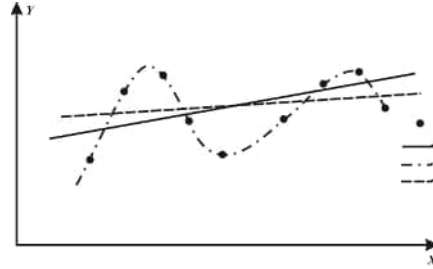


Рис. 6.3. Середньоквадратична апроксимація

$$F(x) = \sum_{i=0}^m a_i \varphi_i(x), \quad (6.6.5)$$

де коефіцієнти a_i визначити так, щоб значення виразу (6.6.4) було мінімальне. Уведемо позначення

$$H(a_0, a_1, \dots, a_m) = \sum_{j=0}^n w(x_j) \left[f(x_j) - \sum_{i=0}^m a_i \varphi_i(x_j) \right]^2, \quad (6.6.6)$$

де $w(x)$ – задана вагова функція така, що $w(x_i) > 0$ для $i = 0, 1, \dots, n$; R_j означає відхилення у точці x_j . Для визначення коефіцієнтів a_i обчислимо частинні похідні функції H за змінними a_i . З умови

$$\frac{\partial H}{\partial a_k} = 0, \quad k = 0, 1, 2, \dots, m, \quad (6.6.7)$$

отримаємо систему $m + 1$ лінійних рівнянь із $m + 1$ невідомими a_i :

$$\frac{\partial H}{\partial a_k} = -2 \sum_{j=0}^n w(x_j) \left[f(x_j) - \sum_{i=0}^m a_i \varphi_i(x_j) \right] \varphi_k(x_j) = 0 \quad (k = \overline{0, m}). \quad (6.6.8)$$

Оскільки $\varphi_j(x)$ утворюють базис простору \mathbf{X}_m , то визначник матриці системи (6.6.8) відмінний від нуля і розв'язок цієї системи мінімізує функцію H .

У матричному вигляді система (6.6.8) записується

$$\mathbf{D}^T \mathbf{D} \mathbf{A} = \mathbf{D}^T \mathbf{f}, \quad (6.6.9)$$

де

$$\mathbf{D} = \begin{bmatrix} \varphi_0(x_0) & \dots & \varphi_m(x_0) \\ \varphi_0(x_1) & \dots & \varphi_m(x_1) \\ \dots & \dots & \dots \\ \varphi_0(x_n) & \dots & \varphi_m(x_n) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

Матриця системи (6.6.8) є симетричною додатно визначеною, отже, існує єдиний розв'язок системи.

6.6.3. Апроксимація многочленами

Якщо за базові функції виберемо послідовність степенів $\{x^i\}_{i=0}^m$, тоді апроксимаційна функція (6.6.10) запишеться так:

$$F(x) = \sum_{i=0}^m a_i x^i, \quad (6.6.10)$$

і формула (6.6.7) набуде вигляду

$$\sum_{j=0}^n \left[f(x_j) - \sum_{i=0}^m a_i x_j^i \right] x_j^k = 0, \quad k = 0, 1, 2, \dots, m. \quad (6.6.11)$$

Змінивши порядок сумування, одержимо

$$\sum_{j=0}^n f(x_j) x_j^k = \sum_{i=0}^m a_i \left(\sum_{j=0}^n x_j^{i+k} \right), \quad k = 0, 1, 2, \dots, m. \quad (6.6.12)$$

Увівши позначення

$$g_{ik} = \sum_{j=0}^n x_j^{i+k}, \quad \rho_k = \sum_{j=0}^n f(x_j) x_j^k, \quad (6.6.13)$$

система (6.6.7) запишеться у вигляді

$$\sum_{i=0}^m a_i g_{ik} = \rho_k, \quad k = 0, 1, 2, \dots, m. \quad (6.6.14)$$

Зазначимо, що у випадку попарно різних вузлів x_0, x_1, \dots, x_n та $m \leq n$, визначник матриці системи (6.6.14) відмінний від нуля, отже, система має єдиний розв'язок. Якщо $m = n$, тоді апроксимуючий поліном $F(x)$ збігається із інтерполяційним поліномом на сітці вузлів x_0, x_1, \dots, x_n , і, крім того, $H = 0$. На практиці степінь полінома вибирають значно меншим від кількості вузлів сітки x_k ($k = \overline{0, n}$).

Розділ 7

Розв'язування початкових задач для звичайних диференціальних рівнянь у MatLab[®]

7.1. Наближене знаходження розв'язку початкових задач для звичайних диференціальних рівнянь

Знаходження розв'язків соціально-економічних моделей, які є об'єктом наших досліджень, часто зводиться до розв'язування початкових задач для звичайних диференціальних рівнянь і систем. Не завжди вдається записати у явному вигляді розв'язок такої задачі. Одним із способів дослідження моделей, які зводяться до початкових задач для диференціальних рівнянь, є побудова наближення розв'язку за допомогою чисельних методів. Насамперед сформулюємо теоретичні основи існування та єдиності розв'язку початкової задачі для диференціального рівняння першого порядку

$$\begin{cases} y'(x) = f(x, y(x)), \\ y(x_0) = y_0, \end{cases} \quad (7.1.1)$$

де $y_0 \in \mathbb{R}$, $f : D \rightarrow \mathbb{R}$, $D = \{(x, y) \in \mathbb{R}^2 : |x - x_0| \leq a, |y - y_0| \leq b\}$, $(a, b > 0)$.

Теорема існування та єдиності розв'язку задачі (7.1.1) формулюється так.

Теорема 7.1.1. *Нехай виконуються умови:*

- (i) *функція f є неперервною в прямокутнику D ;*
- (ii) *f задовольняє умову Lipschitz'a за змінною y в D : існує стала $L > 0$ така, що*

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2| \quad \forall (x, y_1), (x, y_2) \in D.$$

Тоді існує єдиний розв'язок задачі (7.1.1), $y \in C^1([x_0 - \delta, x_0 + \delta])$, де

$$\delta = \min \left\{ a, \frac{b}{M} \right\}$$

та $|f(x, y)| \leq M$ для кожної точки $(x, y) \in D$, $M > 0$.

♣ **Зауваження 7.1.1.** Якщо існує неперервна похідна $\partial f / \partial y$ у прямокутнику D , тоді виконується умова Lipschitz'a (ii).

♣ **Зауваження 7.1.2.** Якщо умову (i) замінити на умову $f \in C^p([x_0 - \delta, x_0 + \delta])$, де $p \in \mathbb{N}$, $p \geq 1$, тоді розв'язок y належить до класу $C^{p+1}([x_0 - \delta, x_0 + \delta])$.

Розглянемо початкову задачу

$$\begin{cases} y'(x) = 1 - 2xy, \\ y(0) = 0. \end{cases} \quad (7.1.2)$$

Розв'язок задачі (7.1.2) має вигляд

$$y(x) = e^{-x^2} \int_0^x e^{t^2} dt. \quad (7.1.3)$$

Для того, щоб знайти розв'язок у точці A , треба обчислити інтеграл $\int_0^A e^{t^2} dt$, який не обчислюється у квадратурах. Створимо файл-сценарій обчислення розв'язку задачі (7.1.2) у довільній точці x . З цією метою спочатку створимо скрипт-файл для підінтегральної функції `fquad`, тобто:

```
% Function fquad to be used by probl.m
% to solve numericaly problem
function y=fquad(t)
q=t.^2; %%% array-smart
y=exp(q);
```

А тепер і сам m-файл обчислення розв'язку:

```
%%% File probl.m Integrate IVP  $y'(x)=1-2xy$ ,  $x$  in  $[0,A]$ ,
% $y(0)=0$ , by using the Method of Variation of arbitrary
%constants and the MATLAB routine quadl %%%
clear
A=input('A:');
temp1=exp(-A*A);
temp2=quadl('fquad',0,A);
format long
yfinal=temp1.*temp2
```

Функція `quad` обчислює визначений інтеграл числовим методом Simpson'a. Ця функція має два параметри – інтегровну функцію та проміжок інтегрування. У наведеному вище прикладі, за допомогою `quad` інтегрується функція `fquad`, яка є функцією масиву і визначена за допомогою скрипт-файлу.

Застосуємо наближені методи знаходження розв'язку за допомогою внутрішніх функцій системи MatLab®

`ode23` та `ode45` (метод Runge-Kutta).

Інструкція `ode23` використовується у форматі:

$$[x,y]=ode23(@rhs2,[x0,x1],y0)$$

та повертає наближений розв'язок, одержаний методом Runge-Kutta порядку 2-3 з адаптивним кроком. Вхідними параметрами цієї функції є назва скрипту правої частини диференціального рівняння в одинарних лапках `'rhs2'`, проміжок інтегрування `[x0,x1]` та початкове наближення шуканого розв'язку `y0`. Зазначимо, що назву скрипт-файлу правої частини рівняння можна записати у вигляді `@rhs2`. Чисельний розв'язок одержимо у вигляді двох векторів `x` та `y`. Вектор $x = [x_i]_{i=1}^n$ визначає вузли,

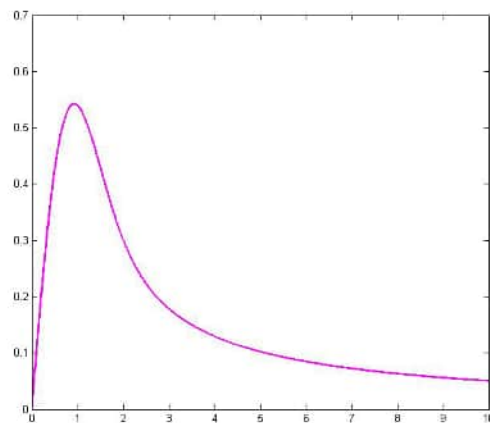


Рис. 7.1. Графік наближеного розв'язку задачі (7.1.2)

обчислені адаптивним методом Runge-Kutta на проміжку $[x_0, x_1]$. Кількість n вузлів a-priori невідома, проте після обчислення розв'язку задачі можна відчитати довжину вектора x за допомогою інструкції `length`. Другий вектор $y = [y_i]_{i=1}^n$ містить відповідні значення шуканого розв'язку y у вузлах x_i : $y_i = y(x_i)$. У середовищі MatLab[®] також є можливість обчислення значення шуканого розв'язку у наперед заданих точках. Наприклад, нехай $[x_0, x_1] = [0, 1]$, тоді, якщо побудуємо вектор

```
>> xspan=[0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
```

то отримаємо

```
[x,y]=ode23(@rhs2,xspan,0)
```

або

```
h=0.01;  
xspan=x0:h:x1;  
[x y]=ode23('rhs2',xspan,y0);
```

Варто зазначити, що отримані масиви слугують для обчислення графіка розв'язку розглядуваної початкової задачі:

```
>> xspan=0:.01:4;  
[x,y]=ode23(@rhs2,xspan,0);  
figure(1); plot(x,y)
```

Застосуємо `ode23` для обчислення наближеного розв'язку задачі (7.1.2). Створимо у редакторі файл, що визначає праву частину рівняння, якому надамо назву `rhs2.m`:

```
function z=rhs2(x,y)  
z=1-2*x.*y;
```

Наступним кроком, або безпосередньо у командній стрічці, або створюємо файл-сценарій, за допомогою якого, наприклад, можна побудувати графік розв'язку:

```
>> x0=0;  
x1=10;  
h=.01;  
y0=0;  
xspan=x0:h:x1;  
[x,y]=ode23('rhs2',xspan,y0);  
figure(1); plot(x,y,'m-', 'LineWidth',2)
```

Після виконання останньої процедури одержимо за допомогою `ode23` наближення розв'язку задачі (7.1.2), графік якого зображено на рис. 7.1.

Іншою функцією в середовищі MatLab® чисельного розв'язування початкових задач для звичайних диференціальних рівнянь є `ode45` (адаптивний метод Runge-Kutta порядку 4-5). Нагадаємо, що у середовищі MatLab® довідку про ці функції можна отримати стандартним способом за допомогою `help ode23` чи `ode45`, або `odeset`.

Поставимо перед собою завдання порівняти два методи знаходження наближеного розв'язку за допомогою функцій `ode23` та `ode45`. У редакторі створимо скрипт під назвою `probl2.m`, використавши визначену функцію `rhs2.m`:

```

%%%%%%%% File probl2.m Integrate the IVP
%%%%%%%% y'(x)=1-2xy, x in [x0,x1], y(0)=0,
%%%%%%%% by using the MATLAB files ode23/ode45
%%%%%%%% and the own function rhs2.m
clear
x0=0;
x1=input('x1: ');
y0=0;
format long
[x,y]=ode23('rhs2',[x0,x1],y0);
N=length(y);
y23=y(N)
figure(2); plot(x,y,'r-', 'LineWidth',2)
grid
hold on
[z,w]=ode45('rhs2',[x0,x1],y0);
M=length(w);
y45=w(M)
plot(z,w,'b:', 'LineWidth',2); legend('ode23','ode45')
hold off

```

Процедурою `probl2.m` обчислено значення розв'язків задачі (7.1.2), одержаних за допомогою інструкцій `ode23` і `ode45`, у кінцевих точках (обчислені розв'язки для $x_0=0$, $x_1=6$) – відповідні рядки у процедурі `N=length(y)`; `y23=y(N)` і `M=length(w)`; `y45=w(M)`. Тому після команди на виконання цього сценарію у вікні `Command Window` система поверне результат:

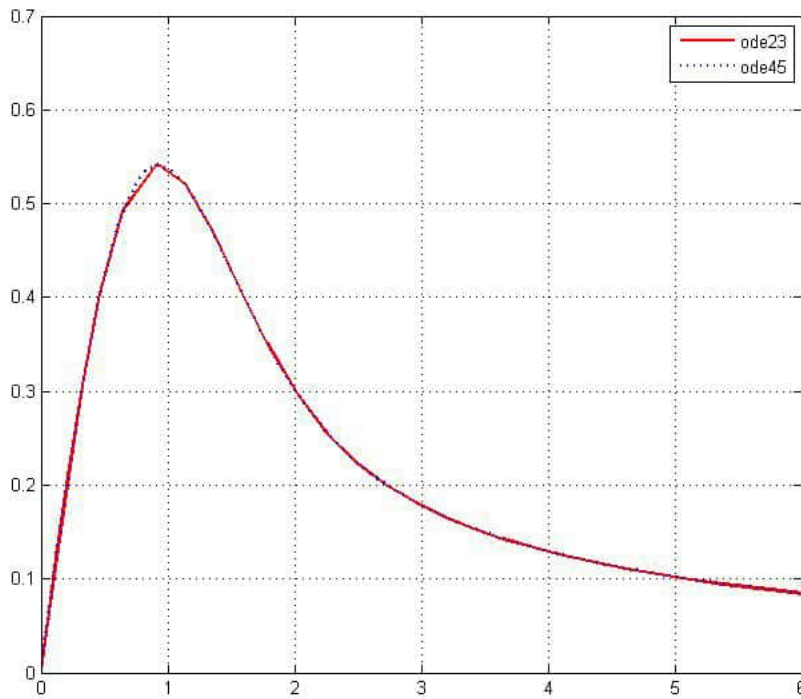


Рис. 7.2. Графіки наближених розв'язків задачі (7.1.2), обчислені за допомогою ode23 і ode45

```

y23 =
    0.084526747274915
y45 =
    0.084550632231825

```

Із одержаних значень можемо зробити висновок, на скільки ці розв'язки різняться між собою. Поведінку розв'язків на всьому проміжку можемо спостерігати на графіку (див. рис. 7.2).

Дослідимо, який із методів – ode12 чи ode45, дає ліпше наближення розв'язку початкової задачі для звичайного диференціального рівняння. Для цього розглянемо задачу:

$$\begin{cases} y'(x) = y - \frac{2x}{y}, \\ y(0) = 1. \end{cases} \quad (7.1.4)$$

Помножимо рівняння (7.1.4) на y

$$yy' = y^2 - 2x, \quad (7.1.5)$$

та впровадимо заміну $z = y^2$. Тоді рівняння (7.1.5) запишеться у вигляді

$$z' = 2z - 4x.$$

Відповідно, початкова умова для z набуде вигляду $z(0) = 1$. Застосувавши метод варіації сталої та провівши обернену заміну, розв'язок задачі (7.1.5) має вигляд

$$y(x) = \sqrt{2x + 1}.$$

Порівняємо точний розв'язок (7.1.4) з наближеним, отриманим за допомогою функції `ode23` середовища MatLab®. Найперше створимо скрипт-файл `rhs3` визначення правої частини рівняння (7.1.4)

```
%rhs3 definite the right-hand side of equation
%y'=y-2x/y
function z=rhs3(x,y);
z=y-2*x./y;
```

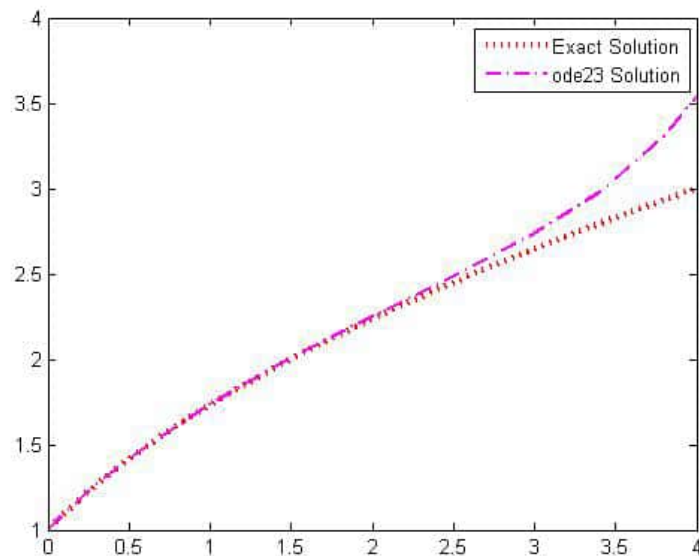


Рис. 7.3. Графіки точного та наближеного розв'язку задачі (7.1.4), обчисленого за допомогою `ode23`

Наступний крок – написати файл-сценарій `probl3_23.m` обчислення наближеного розв’язку за допомогою `ode23` та побудова графіків точного розв’язку і наближеного.

```
%%% file probl3_23.m integrate numerically the IVP
%%% y'(x)=y-2x/y, x in [0,L], y(0)=1,
%%% by using of ode23 and compares on [0,L] with the
%%% mathematical solution y(x)=sqrt(2x+1)
clear; L=input('L: ');
x=0:0.01:L; y=sqrt(2*x+1);
figure(3); plot(x,y,'r:','LineWidth',2)
hold on; [z,w]=ode23('rhs3',[0,L],1);
plot(z,w,'m-.','LineWidth',1)
legend('Exact Solution','ode23 Solution'); hold off
```

Графіки точного та наближеного розв’язків задачі (7.1.4) на проміжку $[0, 4]$ зображені на рис. 7.3. Порівнюючи отримані два графіки, зображені на рис. 7.3, можна зробити висновок, що на відрізку $[0, 2]$ графіки майже збігаються, і сильно відрізняються на відрізку $[2, 4]$. Тобто, починаючи від початкової точки при обчисленні наближеного розв’язку на кожному наступному кроці накопичується похибка обчислення.

Тепер легко написати аналогічний сценарій `probl3_45.m` графічного порівняння точного і наближеного, одержаного за допомогою `ode45`, розв’язків задачі (7.1.4), та провести дослідження. Пропонуємо читачеві виконати це самостійно.

Із останнього прикладу видно, що права частина рівняння слабо задовольняє умови теореми існування та єдиності. Нижче використаємо цей результат для побудови кусково неперервного розв’язку.

7.2. Завдання для самостійного опрацювання

◆ *Завдання 7.2.1.* Розглянути початкову задачу

$$\begin{cases} y'(x) = x + y(x), \\ y(0) = 1, \end{cases}$$

розв’язок якої має вигляд

$$y(x) = 2e^x - (x + 1).$$

Знайти числовий розв'язок на інтервалі $[0, L]$ та побудувати графіки точного та наближеного розв'язку в координатній системі. Порівняти точний і наближені розв'язки, використавши `format long` у точці $x = L$. Порівняти для різних значень L .

◆ *Завдання 7.2.2.* Розглянути початкову задачу

$$\begin{cases} y'(x) = -y(x), \\ y(0) = 1, \end{cases}$$

розв'язком якої є функція

$$y(x) = e^{-x},$$

та провести дослідження, описані у попередньому завданні.

◆ *Завдання 7.2.3.* Для початкової задачі

$$\begin{cases} y'(x) = x^2 + y^2(x), \\ y(0) = -1, \end{cases}$$

на проміжку $[0, L]$ порівняти числові розв'язки, отримані за допомогою функцій `ode23` та `ode45` середовища MatLab®. Побудувати графіки отриманих наближень у координатній системі. Порівняти значення наближень у точці L . Дослідити розв'язки для різних значень L .

◆ *Завдання 7.2.4.* Створити файл-сценарій побудови графіка функції

$$f(x, y) = e^{-((x-3)^2 + (y-2)^2)}, \quad x \in [0, 4], \quad y \in [0, 6].$$

Розділ 8

Основні моделі та їх застосування

У цьому розділі досліджуватимемо моделі, що описуються початковими задачами для звичайних диференціальних рівнянь. Усі моделі мають конкретне практичне застосування і зазвичай їх використовують для розв'язування задач оптимального керування.

8.1. Модель популяції молі spruce budworm

Розглянемо модель поширення популяції молі, яка описується початковою задачею для звичайного диференціального рівняння [38, Розділ 2.3], [2]. Ця моль є комахою, яка нищить ліси північної Америки; пожива для неї – голки хвойних дерев. Нехай $N(t)$ кількість особин у момент часу $t \in [0, T]$. Поширення популяції можна описати за допомогою моделі

$$N'(t) = rN(t) \left(1 - \frac{N(t)}{K}\right),$$

яка означає, що поширення популяції не тільки залежить від природної народжуваності та смертності, а також від ступеня смертності, зумовленого швидким розмноженням. У наведеному вище рівнянні r і K масштабовані параметри, зокрема r є природним коефіцієнтом росту і K виражає властивості середовища. Додатково до коефіцієнта вимірання, який відповідає перенаселенню, $(r/K)N(t)$, додамо доданок, який відповідає хижакам. Наприклад, хижаків репрезентують птахи. Після модифікації рівняння запишеться у вигляді

$$N'(t) = rN(t) \left(1 - \frac{N(t)}{K}\right) - p(N(t)), \quad t \in [0, T],$$

де

$$p(N) = \frac{BN^2}{A^2 + N^2}.$$

У середовищі MatLab[®] процедура побудови графіка $p(N)$ для $B=1.5$, $A=2$ на проміжку $[0, L]$ для $L=20$ має вигляд

```
>> b=1.5;a=2;  
N=0:.01:20;  
p=(b*N.^2)./(a^2+N.^2);  
figure(1);grid on  
plot(N,p,'m-','LineWidth',3)  
title('\it{\bf {p(N): B=1.5, A=2}}','FontSize',16)  
xlabel('\it{\bf {N}}','FontSize',16)  
ylabel('\it{\bf {p(N)}}','FontSize',16)
```

і графік виглядає як на рис. 8.1.

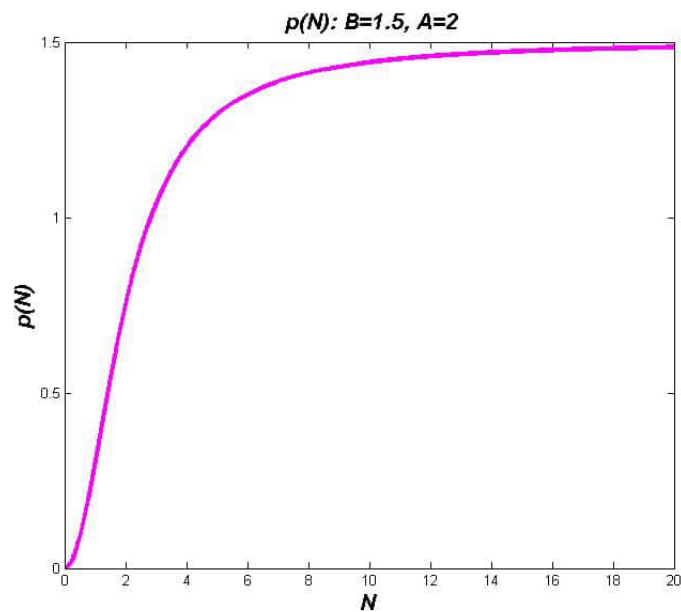


Рис. 8.1. Графік $p(N)$ на проміжку $[0, 20]$ для $B = 1.5$, $A = 2$

Розглянемо тепер початкову задачу для рівняння поширення попу-

ляції

$$\begin{cases} N'(t) = rN(t) \left(1 - \frac{N(t)}{K}\right) - \frac{BN^2(t)}{A^2 + N^2(t)}, & t \in [0, T], \\ N(0) = N_0. \end{cases}$$

Праву частину рівняння визначимо у скрипт-файлі `sbw1.m`:

```
function z=sbw1(y)
global r K A2 B
y2=y*y;
z=r*y*(1-y/K)-B*y2/(A2+y2);
```

Відповідний файл-сценарій

```
% program ist1.m for the spruce budworm model
global r K A2 B
r=input('r='); K=input('K=');
A=input('A='); A2=A*A; B=input('B=');
T=input('T='); N0=input('N(0)=');
h=0.01; tspan=0:h:T;
[x y]=ode45('sbw1',tspan,N0) ;
plot(x,y,'r*');grid on
xlabel('\it{\bf {t}}', 'FontSize',16)
ylabel('\it{\bf {N(t)}}', 'FontSize',16)
```

Динаміку поширення популяції молі spruce budworm для різних значень параметрів зображено на рис. 8.2.

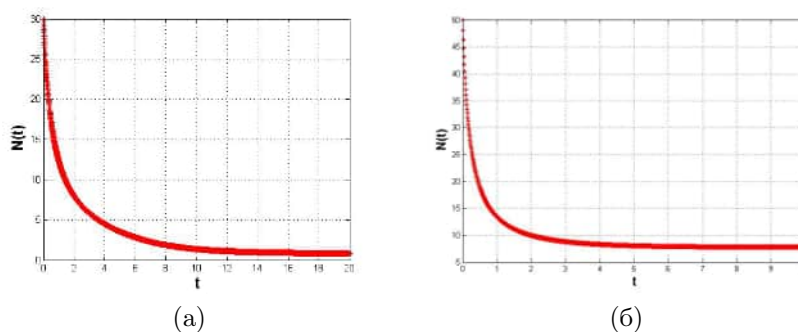


Рис. 8.2. Графіки $N(t)$ за різних вхідних даних: *a)* $r = 0.3$, $K = 5$, $A = 2$, $b = 1.5$, $T = 20$, $N(0) = 30$; *б)* $r = 1$, $K = 10$, $A = 3$, $b = 2$, $T = 10$, $N(0) = 50$.

★ **Приклад 8.1.1.** Дослідити початкову задачу, яка є моделлю поширення популяції комах (подібно як для моделі поширення популяції моли spruce budworm)

$$\begin{cases} N'(t) = rN(t) \left(1 - \frac{N(t)}{K}\right) - u(t)N(t), & t \in [0, T], \\ N(0) = N_0. \end{cases}$$

Значення параметрів моделі взяти такі самі, як у моделі розмноження моли spruce budworm, а саме: $r = 0.3$, $K = 5$, $T = 0$, $N_0 = 30$. Обчислити графіки поширення популяції $N(t)$ за таких значень плодючості $u(t)$:

- (i) $u(t) = 0.5$ на проміжку $[0, T]$;
- (ii) $u(t) = 1$ на проміжку $[0, T]$;
- (iii) $u(t) = 3$ на проміжку $[0, T]$.

У всіх випадках обчислити популяцію за формулою $\int_0^T u(t)N(t)dt$.

✓ *Вказівка.* Необхідно змінити *глобальні змінні* у програмі `ist1.m` та ввести нову величину

```
u=input('u(t)=');
```

Треба також модифікувати програму, яка визначає праву частину рівняння та у подальшому використовується оператором `ode45`. Тобто, функцію правої частини рівняння запишемо у вигляді скрипт-файлу `sbw2.m`:

```
function z=sbw2(x,y)
global r K u
z=r*y*(1-y/K)-u*y;
end
```

Тоді файл-сценарій обчислення популяції комах, який назвемо `ist2.m`, запишеться так:

```
% program ist2.m compute the model sbw2
global r K u
r=input('r=');
K=input('K=');
u=input('u=');
T=input('T=');
N0=input('N(0)=');
```

```
h=0.01;
tspan=0:h:T;
[x y]=ode45('sbw2',tspan,N0);
harv=u*trapz(x,y);
uh=['u=' num2str(u) '; harvest=' num2str(harv) ';'];
disp(uh);
figure(2)
plot(x,y,'r-');grid
xlabel('\it{\bf {t}}','FontSize',16)
ylabel('\it{\bf {N(t)}}','FontSize',16)
```

Після введення у Command Window назви файлу-сценарію та значення параметрів моделі $r = 0.3$, $K = 5$, $u = 0.5$, $T = 20$, $N_0 = 30$, отримаємо обчислені значення та графік (рис. 8.3)

```
>> ist2
r=.3
K=5
u=.5
T=20
N(0)=30
u=0.5; harvest=19.0655;
```

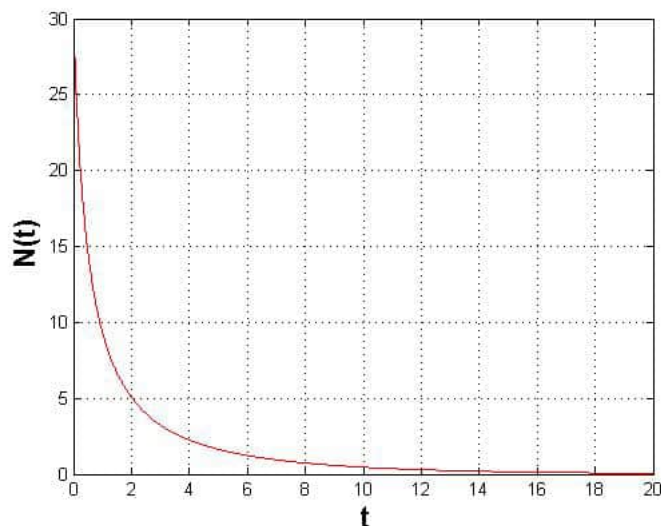


Рис. 8.3. Графік приплоду комах для $r = 0.3$, $K = 5$, $u = 0.5$, $T = 20$, $N(0) = 30$

★ *Приклад 8.1.2.* Написати файл-сценарій виводу на екран обчислених значень приплоду комах для параметрів $r = 0.3$, $K = 5$, $T = 20$, $N_0 = 30$ і послідовності значень $u = 0.5, 1, 3$ та зобразити на одному рисунку графіки популяції.

Відповідний файл-сценарій під назвою `ist2U123.m`, у якому використовуємо функцію `sbw2`, має вигляд:

```
%ist2U123.m compute harvest for u=0.5, 1, 3, sbw2 call
global r K u
vu=[.5 1 3];
figure(3)
r=.3; K=5; T=20; N0=30; h=0.01;
tspan=0:h:T;
l=1;
hold on
for u=vu;
[x y]=ode45('sbw2',tspan,N0);
harv=u*trapz(x,y);
uh=['u=' num2str(u) '; harvest=' num2str(harv) ''];
disp(uh);
plot(x,y,'LineWidth',2,'Color',[l*0.3, 1-l*.2, l*.1]);grid
xlabel('\it{\bf {t}}','FontSize',16)
ylabel('\it{\bf {N(t)}}','FontSize',16)
l=l+1
end
legend('u=0.5','u=1','u=3')
hold off
```

Внаслідок виконання програми отримаємо такі чисельні результати:

```
>> ist2U123
u=0.5; harvest=19.0655;
u=1; harvest=21.2371;
u=3; harvest=25.5577;
```

та графік, зображений на рис. 8.4.

У процесі обчислення приплоду комах ми використали функцію `trapz` середовища `MatLab`[®]. Ця інструкція обчислює визначені інтеграли від функцій, використовуючи метод трапецій.

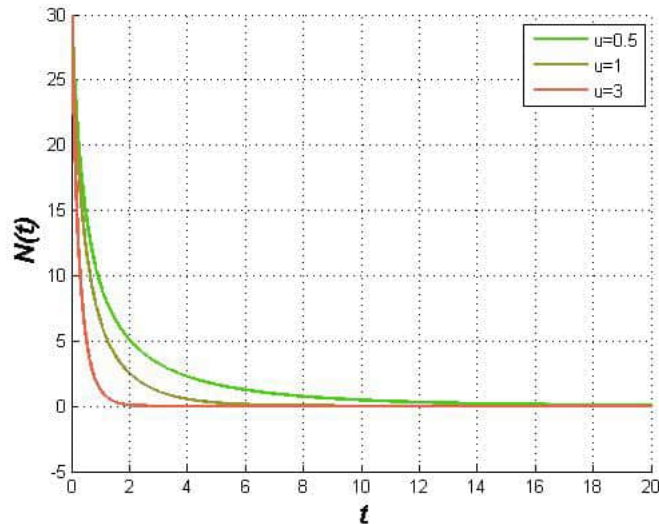


Рис. 8.4. Графіки приплоду комах для $r = 0.3$, $K = 5$, $T = 20$, $N(0) = 30$ та значень $u = 0.3, 1, 3$

8.2. Системи звичайних диференціальних рівнянь. Моделі виживання.

У цьому підрозділі розглянемо біологічну модель, яка описується системою n звичайних диференціальних рівнянь із n невідомими функціями.

8.2.1. Модель хижак-жертва

Розглянемо модель, в якій спостерігається перенаселення популяція жертви [22, Розділ 5.4], [66, Розділ 14]. Тобто, апетит хижаків задоволений і популяція жертви зростає. Позначимо популяцію хижаків у момент часу t через $y_1(t)$, а кількість особин жертви – $y_2(t)$. Тоді зміна у системі *хижак-жертва* описується такою системою звичайних диференціальних рівнянь:

$$\begin{cases} y_1' = -ay_1 + \frac{by_2}{c + ky_2}y_1, \\ y_2' = (d - ey_2)y_2 - \frac{fy_2}{c + ky_2}y_1, \end{cases} \quad (8.2.1)$$

для $t \in [0, L]$. Тут a, b, c, d, e, f суть невід’ємні сталі величини. Додатний параметр k означає граничне перенасичення хижаків. Малі значення k означають, що хижаки жеребкують жертви перед кожним своїм насиченням. Велике значення k означає, що перенасичення швидко з’являється, як тільки збільшується кількість жертв. Для розв’язку системи (8.2.1) сформулюємо такі початкові умови:

$$y_1(0) = 0.5, \quad y_2(0) = 1. \quad (8.2.2)$$

Для нашого чисельного експерименту розглянемо такі значення параметрів: $a = 0.5$, $b = d = e = f = 1$, $c = 0.3$, $k = 0.7$, кінцевий момент часу $L = 200$. Ці значення запишемо у текстовий файл `file1.txt`, вміст якого такий:

```
% file1.txt - file of data value of Predator-Prey model
%using in load sp1.m
0.5
1
0.3
1
1
1
1
0.7
```

Перш ніж перейдемо до написання програми обчислення розв’язку задачі (8.2.1)-(8.2.2), запишемо визначення функції правої частини системи (8.2.1) у скрипт-файл `sprhs1.m`:

```
function out1=sprhs1(t,y)
global a b c d e f k
out1=[ -a*y(1)+b*y(1)*y(2)/(c+k*y(2)); ...
(d-e*y(2))*y(2)-f*y(1)*y(2)/(c + k*y(2))];
end
```

Тепер програма знаходження чисельного розв’язку задачі (8.2.1), (8.2.2) записується у вигляді

```
%Suitable Predation model - sp1.m
clear
global a b c d e f k
load file1.txt
disp('get model parameters');
a=file1(1);
b=file1(2);
c=file1(3);
```

```

d=file1(4);
e=file1(5);
f=file1(6);
k=file1(7);
disp('get data');
L=input('final time :');
y01=input('y1(0) :');
y02=input('y2(0) :');
lw=input('LineWidth : ');
tspan=0:0.01:L;
[t y]=ode45('sprhs1',tspan,[y01 ; y02]);
% graphs
figure(4); plot(t,y(:,1),'LineWidth',lw); grid on
hold on
plot(t,y(:,2),'r','LineWidth',lw)
legend('predator','prey',0)
hold off

```

Прокоментуємо процедуру, записану вище. Значення параметрів задачі завантажуються за допомогою файлу `file1.txt`. У цьому файлі кожна числова величина записана в окремому рядку. Цей текстовий файл може бути створений редактором `MatLab`[®] або кожним іншим текстовим процесором OS (Operating System). Інструкція

```
load file1.txt
```

зчитує вектор даних, записаних у цьому файлі. Ці величини присвоюються відповідним змінним згідно з інструкцією

```
a=file1(1);
```

```
...
```

Зауважимо, що є відмінність у розв'язуванні одного диференціального рівняння та системи звичайних диференціальних рівнянь. Розглянемо спочатку параметри функції `ode45`. Першим параметром є права частина системи рівнянь. Не змінюється і проміжок інтегрування, який у нашому випадку $[0, L]$. Зміни стосуються початкових умов, позаяк у випадку нашої системи цих умов є дві, на протигагу одній у випадку одного диференціального рівняння. Синтаксис початкових умов показаний у самій процедурі вище, тобто має такий формат `[y01;y02]`, отож, початкові умови задаються як вектор-стовпець. Відрізняється результат дії оператора `ode45`. Вектор $t = [t_i]_{i=1}^n$ містить вузли з проміжку $[0, L]$, які отримані під час використання адаптивного методу. Тут u є матрицею, яка містить n рядків і два стовпці, де $n = \text{length}(t)$. Перший стовпець містить чисельні

значення y_1 , а другий – y_2 . Отож, на одному рисунку обчислені графіки зображають y_1 (популяцію хижаків), та y_2 – популяцію жертви.

Інструкція `legend` повертає умовні позначення на рисунку. Текст написаний, відповідно, у вигляді графіка, побудованого на тому ж рисунку. Значення параметра `0` відповідає розташуванню на тому самому рисунку. Можливі значення параметра:

- `0` = розташування тексту за замовчуванням;
- `1` = розташування у правому верхньому куті;
- `2` = розташування у лівому верхньому куті;
- `3` = розташування у лівому нижньому куті;
- `4` = розташування у правому нижньому куті;
- `-1` = розташування справа від графіка.

Детальнішу інформацію можна одержати, скориставшись інструкцією `help legend`.

Пояснимо зараз наявність вектора `tspan`, який містить вузлові точки. Ці вузли використовуються у програмі `ode45` як точки поділу відрізка інтегрування. Тому зі зростанням кількості точок (компонент вектора) отримуємо кращий рисунок графіка. Крім того, інструкція `plot` містить опцію `'Line Width'`, яка дає змогу встановлювати товщину лінії графіка розв'язку. Відповідні значення можуть надаватися безпосередньо числові або за допомогою змінної (`lw` як у нашому випадку). Наприклад, на побудованих графіках вибрано `lw=3`, але можна вибрати інше значення.

Пояснимо тепер як правильно написати значення вихідних параметрів, які ми присвоїли змінній величині `out1`. Нам треба ввести функції правих частин обидвох рівнянь системи. Для цього записуємо праві частини у вигляді вектор-стовпця: у квадратних дужках розділені `;`. Причому, для y_1 використовуємо позначення $y1$, а для y_2 – $y2$. Отримані чисельні розв'язки моделі "хижак-жертва" зображені на рис. 8.5а.

Щоб графік був більш зрозумілий, додамо на кінець до сценарію `sp1` такі два рядки

```
text(100,0.65,' \bf predator', 'FontSize',14)
text(100,0.05,' \bf prey', 'FontSize',14)
```

Результат зображений на рис. 8.5б. Викликавши допомогу середовища `MatLab®`, отримаємо довідку про те, що `text(X,Y,'string')` поміщає

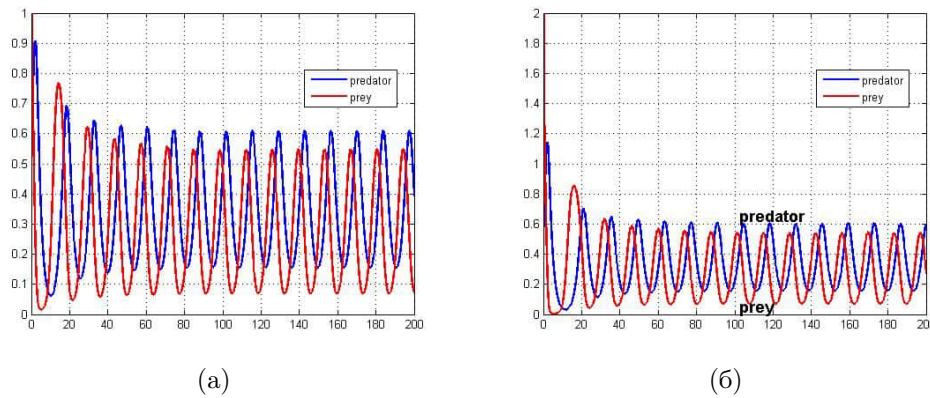


Рис. 8.5. Графіки чисельних розв'язків моделі "хижак-жертва" за різних початкових даних

текстове поле з координатами X, Y у цій системі координат, використовуючи масштаб самого графіка.

Зауважимо, що обидві популяції є стійкими періодичними функціями.

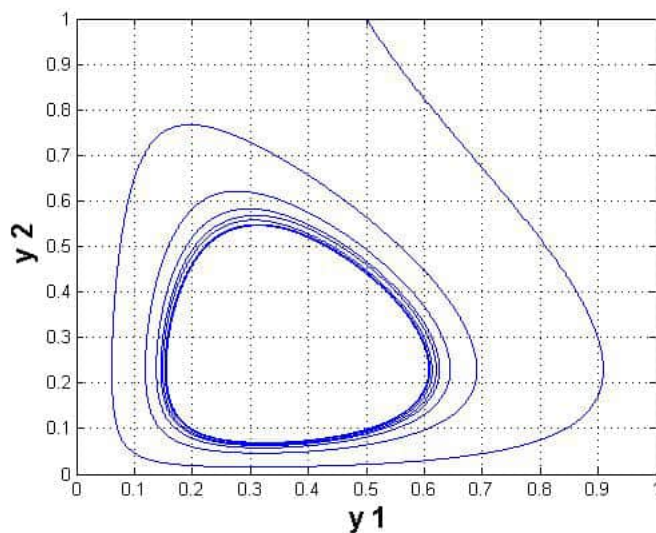


Рис. 8.6. Графік розв'язку моделі "хижак-жертва" у фазовій площині (y_1, y_2)

І нарешті, щоб графічно зобразити чисельний розв'язок моделі

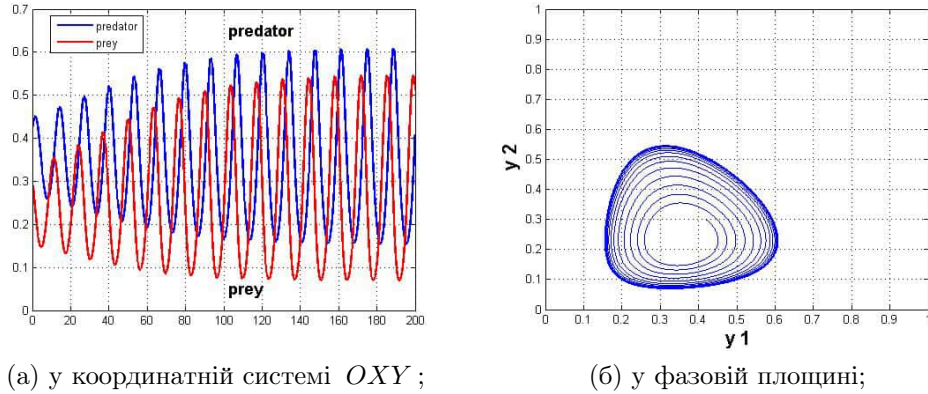


Рис. 8.7. Графік розв'язку моделі "хижак-жертва" за початкових даних $y_1(0) = 0.42$, $y_2(0) = 0.3$.

(8.2.1), (8.2.2) у фазовій площині (y_1, y_2) , у файлі-сценарії `sp1.m` замінимо машинний код частини `graphs`

```
...
% graphs-B
plot(y(:,1),y(:,2),'-');grid
axis([0 1 0 1])
xlabel('\bf y 1','FontSize',16);
ylabel('\bf y 2','FontSize',16);
```

Відповідний графік зображено на рис. 8.6.

Якщо за початкові значення візьмемо

$$y_1(0) = 0.42, \quad y_2(0) = 0.3, \quad (8.2.3)$$

то графіки компонент розв'язку початкової задачі (8.2.1),(8.2.3) зображені на рис. 8.7а. Графік розв'язку у фазовій площині (y_1, y_2) на рис. 8.7б.

Модель нейронної активності. Рівняння Fitzhugh–Nagumo

Рівняння *Fitzhugh–Nagumo* презентує електричну активність нейрона (див. [57, Розділ 6.5], [66, Розділ 14]). Нейрон є збуджуючою системою, яка стимулюється зовнішніми чинниками, наприклад, електричним розрядом. Стан збудження описується функцією y_1 , яка означає напругу у нейроні як функція часу. Коли нейрон збуджений, фізіологічні процеси змушують відновлюватися після збудження. Відновлення описується

функцією y_2 . Запишемо систему Fitzhugh–Nagumo

$$\begin{cases} y_1' = c \left(y_1 + y_2 - \frac{y_1^3}{3} \right), \\ y_2' = \frac{1}{c} (a - y_1 - by_2), \end{cases}$$

при $t \in [0, L]$. Можна зауважити два явища у нейронах:

- реакція y_1 нейрона приводить до стійкого стану після сильного збудження; нейрон збуджується; це одинарний потенціал дії;
- реакція (відгук) y_2 – періодична функція; нейрон отримує періодичні імпульси.

Параметри a , b , c задовольняють такі обмеження, які виражають поведінку

$$1 - \frac{2}{3}b < a < 1, \quad 0 < b < 1, \quad b < c^2.$$

У чисельних експериментах візьмемо такі значення параметрів $a = 0.75$, $b = 0.5$, $c = 1$. Запишемо їх у текстовому файлі `fileFN.txt`. Початкові дані виберемо такі:

$$y_1(0) = 3, \quad y_2(0) = 0. \tag{8.2.4}$$

Утворимо скриптові файли `sprhs1.m` та `fitzhughNagumoEq.m`. Перший із них визначає праву частину системи Fitzhugh–Nagumo, а другий – файл-сценарій чисельного розв’язування системи та побудови графіків функції:

```
function out1=fitzrhs1(t,y)
global a b c
out1=[c*(y(2)+y(1)-(y(1)^3)/3);(a-b*y(2)-y(1))/c];
end
%Suitable Fitzhugh–Nagumo Equations - fitzhughNagumoEq.m
clear
global a b c
load fileFN.txt
disp('get model parameters');
a=fileFN(1);
b=fileFN(2);
c=fileFN(3);
```

```

disp('get data');
L=input('final time :');
y01=input('y1(0) :');
y02=input('y2(0) :');
lw=input('LineWidth : ');% for graphical use (plot)
tspan=0:0.01:L;
[t y]=ode23('fitzrhs1',tspan,[y01 y02]);
figure(5)
% graphs-A
plot(t,y(:,1),'LineWidth',lw);grid
hold on
plot(t,y(:,2),'r','LineWidth',lw)
legend('excitation','recovery',0)
hold off
text(6,1.3,' \it{ \bf {excitation}}', 'FontSize',14)
text(6,-0.5,' \it{ \bf {recovery}}', 'FontSize',14)

```

Після виконання сценарію `fitzhughNagumoEq` у Command Window середовища MatLab[®] введемо дані діалогу, передбачені у сценарії:

```

>> fitzhughNagumoEq
get model parameters
get data
final time :20
y1(0) :3
y2(0) :0
LineWidth : 2

```

одержимо графіки, зображені на рис. 8.8. Зауважимо, що функцію та сценарій, використовуючи глобальні змінні, ми утворили аналогічно як у випадку `sprhs1` та `sp1`, розглянуті вище.

♣ **Зауваження 8.2.1.** Зазначимо, що без написаного сценарію чисельного розв'язування системи диференціальних рівнянь MatLab[®] безпосередньо не повертає розв'язки. Тобто, використовуючи функцію `fitzrhs1` із задекларованими глобальними змінними a , b , c , безпосередньою командою

```
>> ode23('fitzrhs1',[0 20],[3 0])
```

у діалоговому вікні Command Window, система повертає повідомлення про помилку такого змісту:

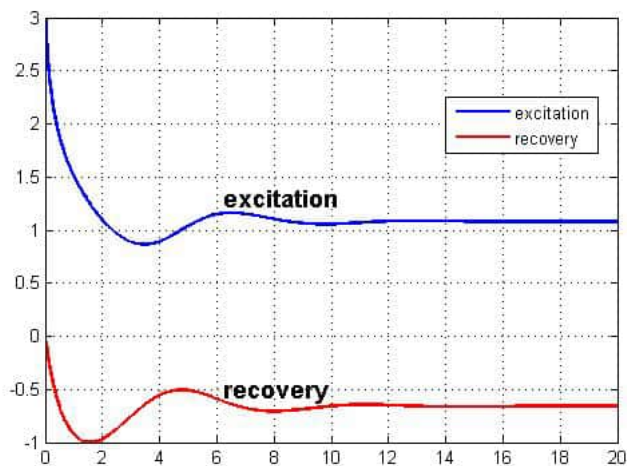


Рис. 8.8. Графіки станів збудження та відновлення

Error using `odearguments` (line 92)

FITZRHS1 returns a vector of length 0, but the length of initial conditions vector is 2. The vector returned by FITZRHS1 and the initial conditions vector must have the same number of elements.

Error in `ode23` (line 112)

[`neq`, `tspan`, `ntspan`, `next`, `t0`, `tfinal`, `tdir`, `y0`, `f0`, `odeArgs`, `odeFcn`, ...]

Помилка полягає у тому, що значення глобальних змінних мають присвоюватися у написаному сценарії.

Уведемо у модель подразник, визначений залежною від часу функцією z

$$\begin{cases} y_1' = c \left(y_1 + y_2 - \frac{y_1^3}{3} \right) - z, \\ y_2' = \frac{a - y_1 - by_2}{c}, \end{cases} \quad (8.2.5)$$

для $t \in [0, L]$. Для числових розрахунків моделі прийmemo

$$z(t) = \begin{cases} 0, & \text{для } t \in [0, t^*], \\ v & \text{для } t \in [t^*, L], \end{cases} \quad (8.2.6)$$

де $0 < t^* < L$ є точкою перемикування і $0 < v < 1$ – величина стимулювання. Відповідна процедура, якій присвоїмо назву `fitz2`, має вигляд:

```

% the Fitzhugh-Nagumo equation - fitz2.m
% z = the stimulus to the neuron
clear
global a b c
global v tsw
load file2.txt
disp('get model parameters');
a=file2(1);
b=file2(2);
c=file2(3);
disp('get data');
L=input('final time :');
y01=input('y1(0) :');
y02=input('y2(0) :');
v=input('stimulus :'); % stimulus value
tsw=input('switch time :'); % switch time for the stimulus
tspan=0:0.01:L;
[t y]=ode45('fitzrhs2',tspan,[y01 y02]);
figure(6)
plot(t,y(:,1),'*',t,y(:,2),'ro');grid
legend('excitation','recovery',0)
text(40,1.5,'\it{\bf{excitation}}','FontSize',16)
text(40,-0.5,'\it{\bf{recovery}}','FontSize',16)

```

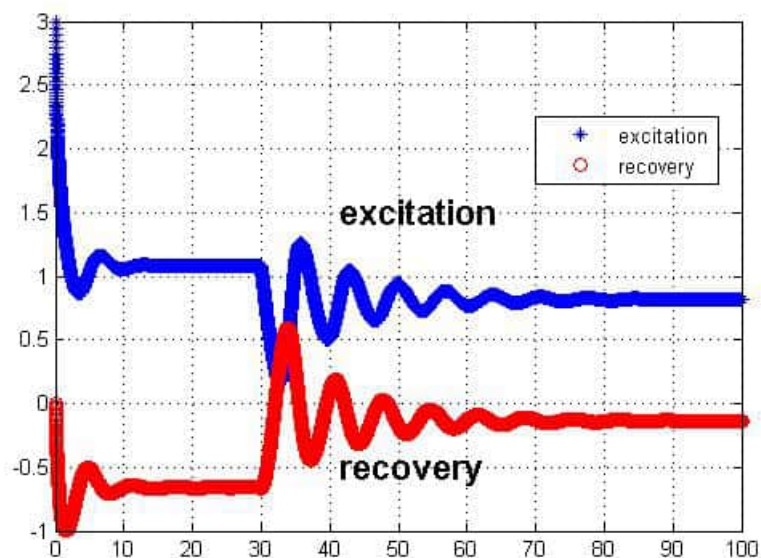


Рис. 8.9. Графіки станів збудження та відновлення у разі дії стимуляції

```

text(40,-0.5,' \it { \bf {recovery}}', 'FontSize',16)
n=length(tspan);
for i=1:n
if tspan(i)>tsw
z(i)=v;
else
z(i)=0;
end
end
figure(7)
plot(tspan,z,'*');
grid
axis([0 L 0 v])
xlabel(' \it { \bf {t}}', 'FontSize',16)
ylabel(' \it { \bf {stimulus z(t)}}', 'FontSize',16)
figure(8)
plot(y(:,1),y(:,2));grid
xlabel(' \it { \bf {y_1}}', 'FontSize',16)
ylabel(' \it { \bf {y_2}}', 'FontSize',16)

```

Скрипт, який визначає праву частину системи (8.2.5), запишемо під назвою `fitzrhs2` так:

```

function out1=fitzrhs2(t,y)
global a b c
global v tsw
if t>tsw
z=v;
else
z=0;
end
out1=[c*(y(2)+y(1)-y(1)^3/3)-z;(a-b*y(2)-y(1))/c];
end

```

В означенні функції `fitzrhs2` уведена змінна t , яка означає часову змінну і слугує для обчислення стимуляцію z . Чисельний розв'язок моделі одержано для значень параметрів a , b , c як у попередньому випадку: $L = 100$, $v = 0.5$, $t^* = 30$ (у програмі ця змінна позначена `tsw`) з формули (8.2.6) та початкові значення (8.2.4). Стан збудження та відновлення, які відповідають стимуляції z , зображено на рис. 8.9. Аналогічно як у випадку моделі хижак-жертва, у програмі чисельного розв'язування побудований графік залежності збудження-відновлення у

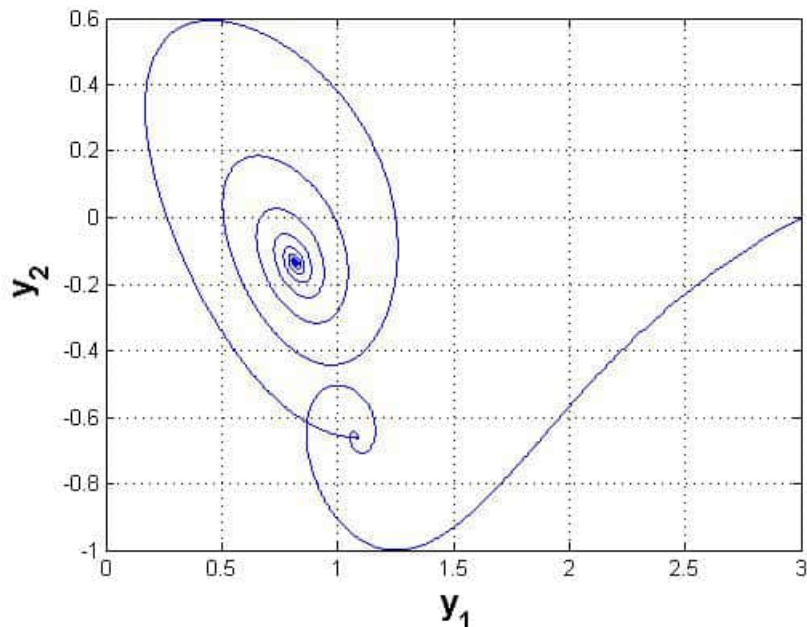


Рис. 8.10. Графік збудження-відновлення у разі дії стимуляції у фазовій площині

фазовій площині (y_1, y_2) (рис. 8.10). Зауважимо, що кожен із графіків побудований у окремому вікні за допомогою команд `figure(6)`, `figure(7)`, `figure(8)`. Крім згаданих графіків, програма обчислює графік стимулятора $z(t)$.

Наступна модель, яку досліджуватимемо, містить три рівняння з трьома шуканими функціями.

8.2.2. Свинець в організмі

Свинець є токсичною речовиною, яка зазвичай може бути у різних продуктах. Модель, яка є об'єктом дослідження у цьому розділі, описує поширення свинцю в організмі на рівні комірок (див., наприклад, [22, Розділ 7.1]). Ми розглянемо три комірки: **кров** (комірка 1), **тканина** (комірка 2), **кістка** (комірка 3). Позначимо через I_1 вміст свинцю в крові та через $y_i(t)$, $i = 1, 2, 3$, кількість свинцю у i -й комірці у момент часу t . Відбувається обмін свинцем у крові і тканині та крові і кістках. Відповідні коефіцієнти обміну між клітинами крові та тканини позначимо k_{21} та k_{12} і через k_{31} , k_{13} відповідні коефіцієнти обміну між

клітинам крові та клітці. Свинець може переноситися через кров в урину (коефіцієнт k_{01}), через клітини тканини у клітини волосся, нігтів і піт (коефіцієнт k_{02}). Отже, одержимо систему диференціальних рівнянь

$$\begin{cases} y_1' = -(k_{01} + k_{21} + k_{31})y_1 + k_{12}y_2 + k_{13}y_3 + I_1, \\ y_2' = k_{21}y_1 - (k_{02} + k_{12})y_2, \\ y_3' = k_{31}y_1 - k_{13}y_3, \end{cases} \quad (8.2.7)$$

для $t \in [0, L]$.

Систему розглядаємо за таких початкових умов

$$y_1(0) = y_2(0) = y_3(0) = 0, \quad (8.2.8)$$

які означають, що у початковий момент часу свинцю в організмі немає. Програма чисельного інтегрування початкової задачі та побудова послідовності графіків для кожної складової розв'язку має вигляд:

```
% Lead in the Body - lead1.m
% y(1) = lead in Blood
% y(2) = lead in Tissues
% y(3) = lead in Bones
clear
global k01 k21 k31
global k02 k12 k13
global I1
load file3.txt
disp('get model parameters');
k01=file3(1);
k21=file3(2);
k31=file3(3);
k02=file3(4);
k12=file3(5);
k13=file3(6);
I1=file3(7);
disp('get data');
L=input('final time :');
y01=input('y1(0)=');
y02=input('y2(0)=');
y03=input('y3(0)=');
tspan=0:0.01:L;
[t y]=ode45('lrhs1',tspan,[y01;y02;y03]);
figure(9)
```

```

plot(t,y(:,1),'ro',t,y(:,2),'*',t,y(:,3),'gs');grid
legend('blood','tissues','bone',0)
text(400,400,' \it{ \bf {tissues}}', 'FontSize',12)
text(400,1400,' \it{ \bf {blood}}', 'FontSize',12)
text(400,2000,' \it{ \bf {bones}}', 'FontSize',12)

```

Система трьох диференціальних рівнянь із трьома невідомими функціями відрізняється від раніше досліджуваних систем двох рівнянь. Вектор початкових даних містить три компоненти. Програма повертає побудовані три графіки $y_i(t)$ ($i = 1, 2, 3$) у координатній системі. Для більшої наочності побудованих графіків використали текстову інформацію.

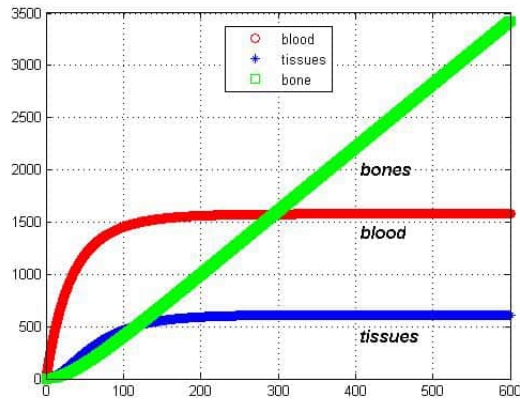


Рис. 8.11. Графіки y_1 , y_2 , y_3

Праву частину, якій присвоїмо назву `lrhs1`, системи запишемо у вигляді скрипту:

```

function out1=lrhs1(t,y)
global k01 k21 k31
global k02 k12 k13
global I1
out1=[-(k01+k21+k31)*y(1)+k12*y(2)+k13*y(3)+I1;...
k21*y(1)-(k02+k12)*y(2);k31*y(1)-k13*y(3)];
end

```

Звернемо увагу, що вираз, визначений у `lrhs1`, містить три складові, які відповідають функціям правої частини кожного з рівнянь системи диференціальних рівнянь (8.2.7), які відокремлені між собою ”;”.

Чисельний експеримент проведемо за таких значень числових параметрів системи: початкових умовах (8.2.8), $L = 600$ (днів) і числових коефіцієнтах $k_{01} = 0.0211$, $k_{21} = 0.0111$, $k_{31} = 0.0039$, $k_{02} = 0.0162$, $k_{12} = 0.0124$, $k_{13} = 0.000035$, значення яких та $I_1 = 49.3(\mu\text{g}/\text{день})$ помі-

стимо у текстовий файл у вигляді стовпчика, зберігаючи послідовність як вище. Графіки чисельних розв'язків зображено на рис. 8.11.

Розглянемо задачу, в якій сталу величину I_1 замінимо на кусково-сталу функцію

$$I(t) = \begin{cases} I_1 & \text{для } t \in [0, t^*], \\ I_2 & \text{для } t \in [t^*, L], \end{cases} \quad (8.2.9)$$

де I_1 , I_2 сталі величини і $0 < t^* < L$. Введемо дві глобальні змінні у відповідний скрипт, а також момент часу **tsw** (the switch time t^*) переключення. Тоді **lrhs**-функція запишеться у вигляді:

```
function out1=lrhs2(t,y)
global k01 k21 k31
global k02 k12 k13
global I1 I2 tsw
if t>tsw
a=I2;
else
a=I1;
end
out1=[-(k01+k21+k31)*y(1)+k12*y(2)+k13*y(3)+a; ...
k21*y(1)-(k02+k12)*y(2);k31*y(1)-k13*y(3)];
end
```

Утворимо **file32** числових параметрів задачі (8.2.7), (8.2.8), взявши за основу **file3**, у якому на восьму позицію впишемо значення $I_2 = 2$. Зробивши відповідні зміни у скрипті **lead1**, утворимо програму **lead2**, яка має вигляд:

```
% Lead in the Body - lead2.m with step function I(t)
% y(1) = lead in Blood
% y(2) = lead in Tissues
% y(3) = lead in Bones
clear
global k01 k21 k31
global k02 k12 k13
global I1 I2 tsw
load file32.txt
disp('get model parameters');
k01=file4(1);
k21=file4(2);
```

```

k31=file4(3);
k02=file4(4);
k12=file4(5);
k13=file4(6);
I1=file4(7);
I2=file4(8);
disp('get data');
L=input('final time :');
tsw=input('t*=');
y01=input('y1(0)=');
y02=input('y2(0)=');
y03=input('y3(0)=');
tspan=0:0.01:L;
[t y]=ode45('lrhs2',tspan,[y01;y02;y03]);
figure(10); plot(t,y(:,1),'ro',t,y(:,2),'* ',t,y(:,3),'gs');
grid on
legend('blood','tissues','bone',0)
text(300,400,' \it{\bf {tissues}}','FontSize',12)
text(410,1600,' \it{\bf {blood}}','FontSize',12)
text(500,2700,' \it{\bf {bones}}','FontSize',12)

```

Обчислимо наближений розв'язок задачі для значень параметрів $L = 800$ та $t^* = 400$. Програма побудує графіки y_1 , y_2 , y_3 , зобра-

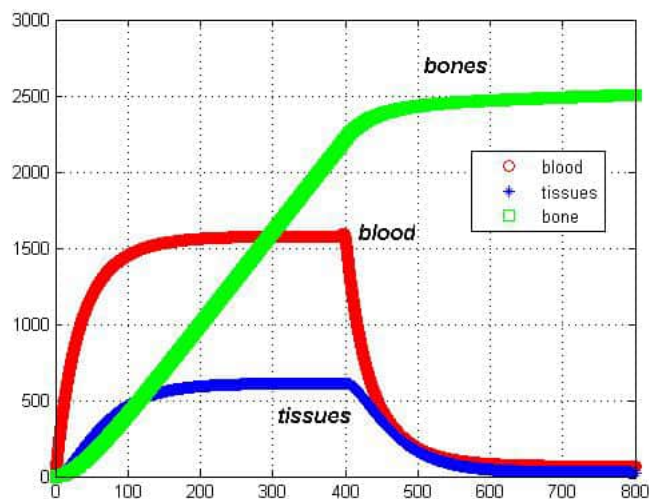


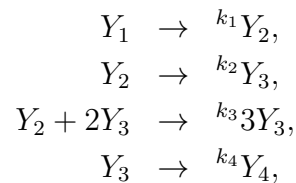
Рис. 8.12. Графіки y_1 , y_2 , y_3 для значення $I(t)$ за формулою (8.2.9)

жені на рис. 8.12.

У наступному підрозділі дослідимо модель, що описується системою чотирьох диференціальних рівнянь із чотирма невідомими функціями.

8.2.3. Модель автокаталітичної реакції

Про модель автокаталітичної реакції можна знайти, наприклад, у [22, Розділ 5.1]. Розглянемо елементи Y_i , $i = \overline{1,4}$ та автокаталітичну реакцію



зі сталими швидкості реакції $k_i > 0$, $i = 1, 2, 3, 4$. Нехай $y_i(t)$ означає концентрацію Y_i в момент часу $t \in [0, L]$. Тоді модель має вигляд початкової задачі для системи диференціальних рівнянь

$$\begin{cases} y_1' = -k_1 y_1, \\ y_2' = k_1 y_1 - k_2 y_2 - k_3 y_2 y_3^2, \\ y_3' = k_2 y_2 - k_4 y_3 + k_3 y_2 y_3^2, \\ y_4' = k_4 y_3, \\ y_1(0) = \alpha, \quad y_2(0) = y_3(0) = y_4(0) = 0, \end{cases} \quad (8.2.10)$$

для $t \in [0, L]$. Час і концентрації записані у безрозмірних величинах. Програма обчислення наближеного розв'язку записується так:

```
% Autocatalytic reaction - auto1.m
clear
global k1 k2 k3 k4
load file4.txt
disp('get model parameters');
k1=file4(1);
k2=file4(2);
k3=file4(3);
k4=file4(4);
disp('get data');
L=input('final time : ');
alpha=input('y1(0)=');
```

```

lw=input('LineWidth : '); % for graphical use (plot)
tspan=0:0.01:L;
[t y]=ode45('arerhs1',tspan,[alpha; 0; 0; 0]);
z1=y(:,1)/200;
z4=y(:,4)/200;
plot(t,z1,'*',t,y(:,2),'r',t,y(:,3),'g',t,z4,'co') % plot 1
grid on
xlabel(' \it{\bf {t}}', 'FontSize',16)
text(50,2,' \it{\bf {y1/200}}')
text(100,0.25,' \it{\bf {y4/200}}')
text(250,2.5,' \it{\bf {y2}}')
text(200,0.2,' \it{\bf {y3}}')
figure(11) % plot 2
plot(t,y(:,2),'r','LineWidth',lw); grid on
xlabel(' \it{\bf {t}}', 'FontSize',16)
hold on
plot(t,y(:,3),'g','LineWidth',lw)
text(250,2.5,' \it{\bf {y2}}', 'FontSize',16)
text(200,0.2,' \it{\bf {y3}}', 'FontSize',16)
figure(12)
plot(y(:,2),y(:,3),' : '); grid on % plot 3
xlabel(' \it{\bf {y2}}', 'FontSize',16)
ylabel(' \it{\bf {y3}}', 'FontSize',16)
hold off

```

Щоб графіки були співрозмірні, поділимо y_1 та y_4 на 200. Одержано графічне зображення $y_2(t)$ та $y_3(t)$ у спільній системі координат та у фазовій площині y_2Oy_3 .

Праву частину системи запишемо у вигляді скрипт-функції середовища MatLab[®]:

```

function out1=arerhs1(t,y)
global k1 k2 k3 k4
out1=[-k1*y(1);k1*y(1)-y(2)*(k2+k3*y(3)^2);...
y(2)*(k2+k3*y(3)^2)-k4*y(3);k4*y(3)];
end

```

Створимо текстовий файл file4.txt, в якому запишемо у стовпець значення $k_1 = 0.002$, $k_2 = 0.08$, $k_3 = k_4 = 1$. Обчислимо наближений розв'язок для значення $L = 400$ та $\alpha = 500$. Графічний результат обчислення зображений на рис. 8.13.

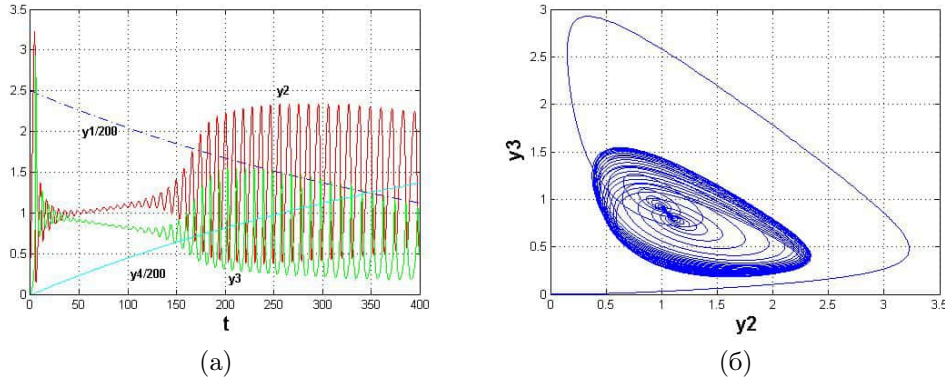


Рис. 8.13. Графіки розв'язків автокаталітичної моделі: a – графіки $y_1/200$, y_2 , y_3 , $y_4/200$; b – графік у фазовій площині y_2Oy_3

8.2.4. Модель навантаженої балки

Розглянемо тепер модель із механіки. Ця модель нас цікавить радше з боку чисельної реалізації, точніше, наскільки питання адекватності моделі залежать від порядку параметрів.

Нехай маємо балку, зосереджену на проміжку $[0, L]$, кінець якої $x = 0$ зафіксований. Другий кінець $x = L$ вільний і перебуває під навантаженням P . Позначимо через $y(x)$ зміщення навантаженої балки в точці $x \in [0, L]$. Математична модель описаної задачі має вигляд (див. [35, Розділ 8.14])

$$\begin{cases} \frac{y''}{(1 + (y')^2)^{3/2}} = \frac{P}{EI}(L - x), & x \in [0, L], \\ y(0) = y'(0) = 0. \end{cases} \quad (8.2.11)$$

Зазначимо, що $y(x)$ означає абсолютне зміщення, тому розглядатимемо $-y(x)$ як від'ємне відхилення стосовно осі OX . У задачі (8.2.11) E означає модуль Young'а, а I – момент інерції поперечного перерізу балки. Припускаємо, що фізичні величини є сталими (незалежні від x), тому надалі уведемо позначення $C = P/(EI)$.

Розглянемо два випадки.

1. Навантаження P є "малим", а отже, швидкість зміщення також "мала". Тоді $1 + (y')^2 \approx 1$ і задача (8.2.11) запишеться

$$\begin{cases} y'' = C(x - L), & x \in [0, L], \\ y(0) = y'(0) = 0. \end{cases} \quad (8.2.12)$$

Інтегруючи останню задачу отримаємо її розв'язок

$$y(x) = \frac{C}{6}x^2(3L - x), \quad x \in [0, L]. \quad (8.2.13)$$

2. Навантаження P не є "малим", а отже, $(y')^2$ нехтувати не можна. Розв'язок задачі (8.2.11) обчислимо наближеними методами. Уведемо дві функції $y_1 = y$, $y_2 = y'$ та запишемо задачу (8.2.11) у вигляді

$$\begin{cases} y_1' = y_2, & x \in [0, L], \\ y_2' = C(1 + y_2^2)^{3/2}(L - x), & x \in [0, L], \\ y_1(0) = y_2(0) = 0. \end{cases} \quad (8.2.14)$$

Нижче наведено скрипт графічного порівняння розв'язку (8.2.13) задачі (8.2.12) та наближеного розв'язку задачі (8.2.14), отриманого за допомогою `ode45`. Зауважимо, що многочлен у (8.2.13) можна завжди обчислити для кожного $x \in [0, L]$.

```
% file beam1.m
% the loaded beam fixed at one end
% the numerical integration is made by ode45
% a comparison is made to the simplified model
clear
global L C
L=input('L : ')
C=input('C : ')
L3=3.0 * L;
C6=C/6.0;
z=0:0.001:L;
[x y]=ode45('b',z,[0; 0]);
plot(x,-y(:,1),'s');grid
xlabel('\it{\bf{x}}','FontSize',16)
ylabel('\it{\bf{-y(x)}}','FontSize',16)
hold on
w=C6*(z.^2).*(L3-z);
plot(z,-w,'r*')
hleg=legend('\it{\bf{Numerical}}','\it{\bf{Polynomial}}')
set(hleg,'FontSize',16)
hold off
```

Тут $[x \ y]$ пара векторів, яку повертає після обчислення `ode45`, а $[z \ w]$

пара векторів, що відповідає обчисленому розв'язку (8.2.13) спрощеної моделі. Зауважимо, що w обчислено, використовуючи масив z , тому для визначення w використали операції на масивах. Функція `rhs`, яка використовується у `ode45`, визначена файлом `b.m` – права частина системи (8.2.14).

Для порівняння розв'язків для значення $x = L$ до скрипту `beam1.m` додамо послідовність команд:

```
format long
N=length(x) ;
ynum=y(N,1) % numerical value at x = L
M=length(w) ;
ypol=w(M) % polynomial value at x = L
```

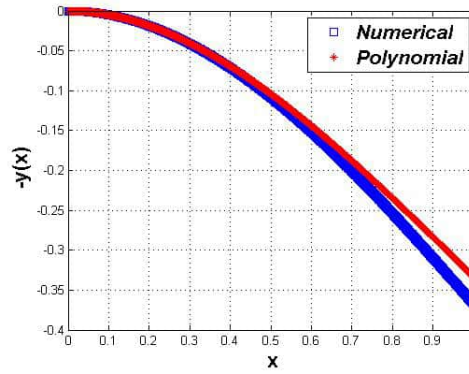
Обчислимо розв'язки задачі для $L = 1$ і різних значень величини C . Позаяк $C = P/(EI)$, тому у наших обчисленнях значення таке ж як навантаження P . Для випадку $C = 1$ графіки зображені на рис. 8.14а. На графіках спостерігаємо відмінність між розв'язками у наближенні до правого кінця балки. В програмі для позначення розв'язків використали функцію `text`. Так `Numerical` означає "чисельний розв'язок", а `Polynomial` – "розв'язок многочленом".

Для значення $C = 1.75$ розв'язки зображені на рис. 8.14б. Можна зауважити ще більшу відмінність між ними у наближенні до правого навантаженого кінця балки. Причому графік розв'язку, отриманий за допомогою `ode45`, розташований нижче від графіка розв'язку многочленом. Як і очікувалося спрощена модель не є правильною, коли значення C великі.

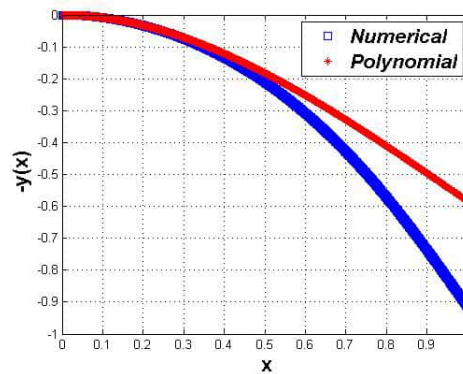
Накінець, розв'язки, одержані для значення $C = 2.6$, графічно зображені на рис. 8.14в. Наближений розв'язок у деяких випадках не має значення, що з фізичного погляду зору означає руйнування балки під дією великого навантаження. Розв'язок за допомогою полінома існує, оскільки поліном (8.2.13) визначений для кожного x , але немає жодного відношення до точного розв'язку. Звідси напрощується очевидний висновок, що спрощена модель, записана за допомогою полінома, неадекватна, коли навантаження великі (великих значеннях параметра C).

8.2.5. 3D графіка

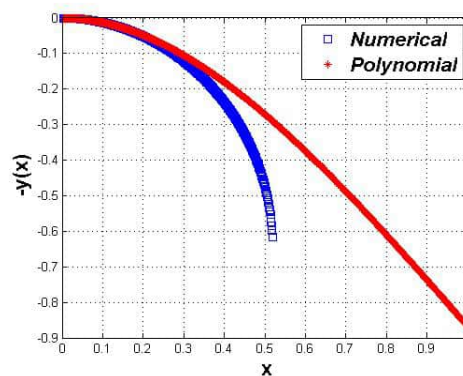
Еквівалентною інструкцією до `plot` у випадку 3D, є `plot3`. Ця команда використовує функцію, графік якої треба побудувати, задану в параметричній формі. Найпростіший формат цієї інструкції



(a)



(б)



(в)

Рис. 8.14. Графіки розв'язків при різних навантаженнях балки: *a* – випадок $C = 1$; *б* – випадок $C = 1.75$; *в* – випадок $C = 2.6$

`plot3(x,y,z)`

де x, y, z – вектори однакової довжини, припустимо N . Процедура `plot3` обчислює $3D$ криву, яка проходить через точки (x_i, y_i, z_i) , $i = \overline{1, N}$.

Розглянемо приклади, які можна знайти у документації `help plot3` середовища `MatLab`[®]. А саме, гвинтову просторову лінію, яка задається параметричним рівнянням

$$\begin{cases} x = \cos t, \\ y = \sin t, \\ z = t, \end{cases} \quad (8.2.15)$$

для $t \in [0, 6\pi]$. Запишемо скрипт `helix.m` обчислення цієї кривої:

```
% file helix.m
% parametric representation with plot3
h=pi/1000;
t=0:h:6*pi;
figure(13)
plot3(sin(t),cos(t),t,'*'); grid
xlabel(' \it{\bf{sin(t)}}', 'FontSize', 16)
ylabel(' \it{\bf{cos(t)}}', 'FontSize', 16)
zlabel(' \it{\bf{t}}', 'FontSize', 16)
```

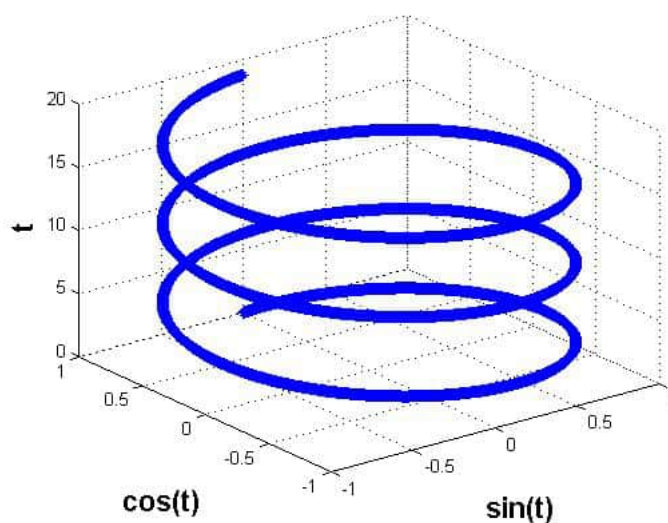


Рис. 8.15. Графік гвинтової лінії

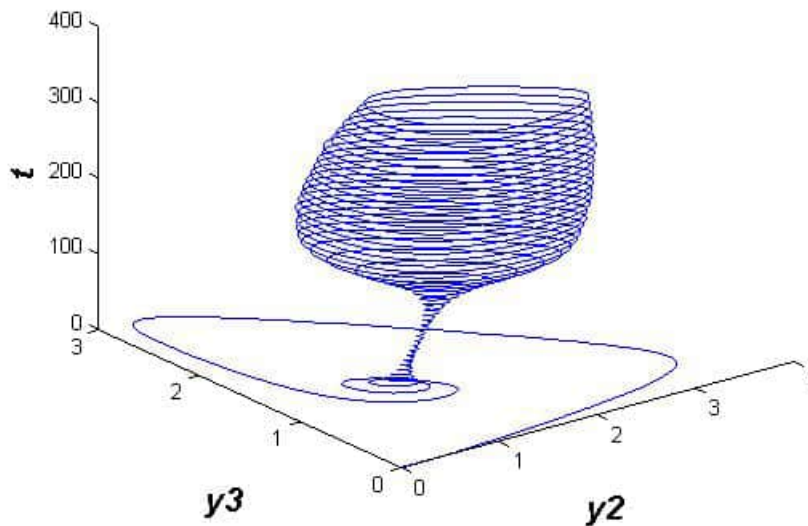


Рис. 8.16. Осциляція у бокалі

Результат дії програми зображено на рис. 8.15.

Повернемось до моделі автокаталітичної реакції, розглянутої у підрозділі 8.2.3. (програма `auto1.m`). Додамо до програми такі рядки:

```
figure(14)
plot3(y(:,2),y(:,3),t)
xlabel(' \it{ \bf {y2}}', 'FontSize', 16)
ylabel(' \it{ \bf {y3}}', 'FontSize', 16)
zlabel(' \it{ \bf {t}}', 'FontSize', 16)
```

У результаті виконання програми зі значеннями параметрів як у підрозділі 8.2.3., одержимо графік, зображений на рис. 8.16 (розглянута модель називається *осциляцією у бокалі*).

Для побудови графіка функції $z = f(x, y)$ використовуються команди `mesh` та `surf`. Припустимо, що функція $z = f(x, y)$ визначена у прямокутнику $[a, b] \times [c, d]$. Спочатку обчислимо вектори x та y , відповідні компоненти яких є координатами вузлів, відповідно, на осі OX та OY

```
x=a:hx:b;
y=c:hy:d;
```

Опісля за допомогою команди `meshgrid` утворюємо матрицю X на Y

```
[X,Y]=meshgrid(x,y);
```

Припустимо, що $\text{length}(x)$ дорівнює n , а $\text{length}(y)$ – m . Тоді розмірності кожної матриці дорівнює $m \times n$. Кожний m -й рядок X дорівнює вектору x , всі стовпці Y дорівнюють вектору y . Після цього створюємо матрицю Z

```
Z=f(X,Y);
```

де $f.m$ – відповідна функція масиву. Тоді 3D "сітчасту поверхню" обчислюємо за допомогою команди

```
mesh(X,Y,Z)
```

а 3D "гранену поверхню" генеруємо за допомогою команди

```
surf(X,Y,Z)
```

Побудувати 3D поверхні можна безпосередньо

```
mesh(X,Y,f(X,Y))
```

Аналогічно у випадку команди `surf`.

Розглянемо приклад обчислення графіків, зображених на рис. 8.17. Створимо програму `Graph3D.m` обчислення графіків функції $z = \sin(3x^2 + 5y^2)$ за допомогою команд `surf` і `mesh`. Треба звернути увагу на спосіб побудови вузлів сітки, за якою обчислюються значення функції, і у такий спосіб обчислюється масив точок поверхні у просторі. Як зазначалося вище, для обчислення вузлів слугує команда `meshgrid`. Причому `meshgrid` об'єднує у масив вузлів два задані наперед масиви (як буде продемонстровано у програмі далі), а можна створити ці вузли безпосередньо

```
[X Y]=meshgrid(a:hx:b c:hy:d)
```

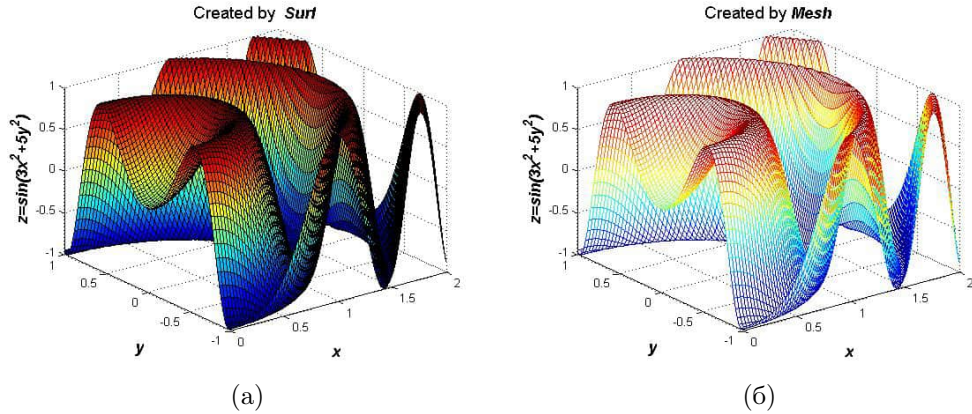
або

```
[X Y]=meshgrid(a:h:b)
```

якщо розглядається сітка вузлів у квадраті.

Отож, програма `Graph3D.m`:

```
%file Graph3D.m created the surface
%using statement surf and mesh
hx=0.03;hy=0.02;
xg=0:hx:2; yg=-1:hy:1;
[Xs Ys]=meshgrid(xg,yg);
zs=sin(3*Xs.^2+5*Ys.^2);
figure(15); surf(Xs,Ys,zs)
```

Рис. 8.17. 3D графіки: *a* – surf-графік; *б* – mesh-графік

```

xlabel(' \it{ \bf {x}}', 'FontSize', 14)
ylabel(' \it{ \bf {y}}', 'FontSize', 14)
zlabel(' \it{ \bf {z=sin(3x^2+5y^2)}}', 'FontSize', 14)
title('Created by \it{ \bf {Surf}}', 'FontSize', 14)
figure(16)
mesh(Xs, Ys, zs)
xlabel(' \it{ \bf {x}}', 'FontSize', 14)
ylabel(' \it{ \bf {y}}', 'FontSize', 14)
zlabel(' \it{ \bf {z=sin(3x^2+5y^2)}}', 'FontSize', 14)
title('Created by \it{ \bf {Mesh}}', 'FontSize', 14)

```

8.2.6. Завдання для самостійного опрацювання

◆ *Завдання 8.2.1.* Розглянути початкову задачу на проміжку $[0, L]$

$$\begin{cases} y'(x) = x + y(x), \\ y(0) = 1. \end{cases}$$

У середовищі MatLab[®] побудувати графік точного розв'язку (отриманого методом варіації сталих)

$$y(x) = 2e^x - (x + 1),$$

та наближеного. Чисельний розв'язок побудувати, використовуючи команди `ode23` та `ode45`. Порівняти графічно отримані розв'язки. Порівняти числові значення у форматі `long` трьох розв'язків у кінцевій точці проміжку $x = L$. Зробити порівняння для різних значень L .

◆ *Завдання 8.2.2.* Завдання, сформульоване вище, виконати для початкової задачі

$$\begin{cases} y'(x) = y, \\ y(0) = 1, \end{cases}$$

точний розв'язок якої

$$y(x) = e^{-x}.$$

◆ *Завдання 8.2.3.* Розглянути початкову задачу

$$\begin{cases} y'(x) = x^2 + y^2, \\ y(0) = -1. \end{cases}$$

На проміжку $[0, L]$ побудувати наближені розв'язки, отримані за допомогою `ode23` та `ode45`. Зробити графічне порівняння отриманих розв'язків і значення їх у кінцевій точці проміжку. Зробити це для різних значень L .

◆ *Завдання 8.2.4.* Дослідити початкову задачу для звичайного диференціального рівняння, яка є моделлю поширення популяції комах (аналогічна до моделі spruce budworm)

$$\begin{cases} N'(t) = rN(t) \left(1 - \frac{N(t)}{K}\right) - u(t)N(t), & t \in [0, T], \\ N(0) = N_0. \end{cases}$$

Знайти розв'язок моделі для значення параметрів $r = 0.3$, $K = 5$, $T = 20$, $N_0 = 30$. Побудувати графік популяції комах $N(t)$ для $u(t) = t$ $t \in [0, T]$. Обчислити приплив комах за формулою $\int_0^T u(t)N(t)dt$.

◆ *Завдання 8.2.5.* Побудувати графік функції $z = \sin(x - y)$, $x \in [0, 4]$, $y \in [0, 6]$. На графіку вказати назву осей, назву графіка.

◆ *Завдання 8.2.6.* Таке саме завдання як і вище для функції $z = e^{-((x-2)^2 + (y-2)^2)}$, $x \in [0, 4]$, $y \in [0, 6]$.

8.3. Оптимальне керування диференціальними системами. Умови оптимальності

Численна наукова література присвячена теорії оптимального керування. Ця тематика почала бурхливо розвиватися після перших піонерських робіт Понтрягіна та його учнів. Одне з важливих завдань у дослідженні задач оптимального керування – отримати необхідні умов оптимальності першого порядку (принцип Понтрягіна). Подамо список найбільш важливих монографій на цю тематику: [78, 79, 80, 55, 51, 18, 19, 67]. Застосуванням задач оптимального керування присвячені такі праці: [51, 31, 19, 13, 71]. Застосування середовища MatLab[®] для розв’язування задач оптимального керування у біології див. [56].

У цьому розділі, як і у наступному, розглядаються базові ідеї та техніка дослідження у теорії оптимального керування системами звичайних диференціальних рівнянь. Ми не будемо розглядати задачу оптимального керування чи принцип Понтрягіна у загальному вигляді, а опрацюємо безпосередньо задачі оптимального керування, які виникають у природознавстві та економіці й описуються системами звичайних диференціальних рівнянь. Намагатимемося звернути увагу на основні, принципові етапи дослідження задач оптимального керування через конкретні приклади, для яких ці етапи подібні. Однак у деяких прикладах є відмінні технічні труднощі.

Основна мета цього розділу – довести існування та єдиність оптимального керування, отримати необхідні умови оптимальності для деяких важливих задач оптимального керування. Із необхідних умов оптимальності отримаємо важливу інформацію про структуру оптимального керування. Наведені чисельні алгоритми апроксимації оптимального керування та відповідні програми у MatLab[®].

Формулювання принципу Понтрягіна для задач оптимального керування звичайних диференціальних рівнянь у загальному вигляді можна знайти, наприклад, у працях [18, 19, 77].

8.3.1. Формулювання задачі. Принцип Понтрягіна

У загальному формулюванні задача оптимального керування для звичайного диференціального рівняння формулюється так: максимізувати функціонал

$$\mathcal{L}(u, x^u) = \int_0^T G(t, u(t), x^u(t)) dt + \varphi(x^u(T)) \rightarrow \max \quad (P_1)$$

стосовно $u \in K \subset L^2(0, T; \mathbb{R}^m)$ ($T > 0$), де x^u – розв’язок Carathéodory початкової задачі

$$\begin{cases} x'(t) = f(t, u(t), x(t)), & t \in (0, T), \\ x(0) = x_0. \end{cases} \quad (8.3.1)$$

Тут

$$\begin{aligned} G &: [0, T] \times \mathbb{R}^m \times \mathbb{R}^N \rightarrow \mathbb{R}, \\ \varphi &: \mathbb{R}^N \rightarrow \mathbb{R}, \\ f &: [0, T] \times \mathbb{R}^m \times \mathbb{R}^N \rightarrow \mathbb{R}, \end{aligned}$$

$x_0 \in \mathbb{R}^N$, $m, N \in \mathbb{N}^*$, і $K \in L^2(0, T; \mathbb{R}^m)$ – опукла замкнута підмножина. Тут і нижче всі елементи \mathbb{R}^n , $n \in \mathbb{N}^*$ розглядаємо як вектор стовпці (\mathbb{N}^* – множина цілих додатних чисел).

Нагадаємо, що *розв’язком Carathéodory* (або просто розв’язком) називаємо функцію x^u , що належить до $AC([0, T]; \mathbb{R}^N)$ ¹ та задовольняє початкову задачу

$$\begin{cases} (x^u)'(t) = f(t, u(t), x^u(t)), & t \in (0, T), \\ x^u(0) = x_0. \end{cases}$$

$L^2(0, T; \mathbb{R}^m)$ є простором керувань.

Задача оптимального керування може бути сформульована у вигляді задачі мінімізації

$$-\mathcal{L}(u, x^u) \rightarrow \min,$$

¹ • *О з н а ч е н н я.* Нехай X – дійсний простір Banach’a з нормою $\|\cdot\|$, $a, b \in \mathbb{R}$, $a < b$. Функцію $x : [a, b] \rightarrow X$ називаємо неперервною у $t_0 \in [a, b]$, якщо дійсна функція $g : [a, b] \rightarrow \mathbb{R}$,

$$g(t) = \|x(t) - x(t_0)\|, \quad t \in [a, b],$$

є неперервною у t_0 .

• *О з н а ч е н н я.* Функцію $x : [a, b] \rightarrow X$ називаємо абсолютно неперервною на $[a, b]$, якщо $\forall \varepsilon > 0, \exists \delta(\varepsilon) > 0$, що виконується

$$\sum_{k=1}^N \|x(t_k) - x(s_k)\| < \varepsilon,$$

за умови $\sum_{k=1}^N |t_k - s_k| < \delta(\varepsilon)$, $(t_k, s_k) \cap (t_j, s_j) = \emptyset$ для $k \neq j$ ($t_k, s_k \in [a, b]$ $\forall k \in \{1, 2, \dots, N\}$).

Позначимо через $AC([a, b]; X)$ простір абсолютно неперервних функцій $x : [a, b] \rightarrow X$.

стосовно керування $u \in K \subset L^2(0, T; \mathbb{R}^m)$.

Припускаємо, $\forall u \in L^2(0, T; \mathbb{R}^m)$ фіксованою задачею (8.3.1) має єдиний розв'язок, який позначатимемо x^u . Рівняння (8.3.1) називається *станом задачі (рівняння)*, а:

- $u \in K$ називається *керуванням (або регулятором)*. Це керування є умовним, позаяк $u \in K$, а K – підмножина $L^2(0, T; \mathbb{R}^m)$;
- x^u – *стан*, що відповідає керуванню u , і є відображенням;
- $u \rightarrow \mathcal{L}(u, x^u) = \Phi(u)$ – *функціонал корисності*.

Кажуть, що $u^* \in K$ є *оптимальним розв'язком* задачі (P₁), якщо

$$\mathcal{L}(u, x^u) \leq \mathcal{L}(u^*, x^{u^*})$$

для кожного $u \in K$. Пара (u^*, x^{u^*}) називається *оптимальною парою*, а $\mathcal{L}(u^*, x^{u^*})$ називається *оптимальним значенням функціонала корисності*. Також говоримо, що (u^*, x^*) – оптимальна пара, якщо u^* є оптимальним керуванням, а $x^* = x^{u^*}$.

Нехай $u^* \in K$ – оптимальне керування задачі (P₁), тоді

$$\int_0^T G(t, u^*(t), x^{u^*}(t)) dt + \varphi(x^{u^*}(T)) \geq \int_0^T G(t, u(t), x^u(t)) dt + \varphi(x^u(T)),$$

для кожного $u \in K$.

Припускаємо, що наступна послідовність операцій та аргументів є допустимими (якщо справджуються деякі гіпотези – враховуючи диференційовність за Gâteaux – щодо G, φ, f). Прийmemo такі позначення:

$$\begin{cases} f_u = \frac{\partial f}{\partial u}, & f_x = \frac{\partial f}{\partial x}, \\ G_u = \frac{\partial G}{\partial u}, & G_x = \frac{\partial G}{\partial x}, \\ \varphi_x = \frac{\partial \varphi}{\partial x}. \end{cases}$$

Тут G_u, G_x, φ_x розглядаються як вектор-стовпчики.

Припускаємо, що кожна функція, визначена на $L^2(0, T; \mathbb{R}^m)$, $u \rightarrow x^u$ є всюди диференційовна за Gâteaux. Цей диференціал ми позначатимемо dx^u .

Розглянемо

$$V = \{v \in L^2(0, T; \mathbb{R}^m); u^* + \varepsilon v \in K \text{ для достатньо малого } \forall \varepsilon > 0\}.$$

Для кожної $v \in V$ позначимо $z = dx^{u^*}(v)$; z – розв’язок початкової задачі

$$\begin{cases} z'(t) = f_u(t, u^*(t), x^{u^*}(t))v(t) + \\ \quad f_x(t, u^*(t), x^{u^*}(t))z(t), \quad t \in (0, T), \\ z(0) = 0. \end{cases} \quad (8.3.2)$$

Для кожної довільної фіксованої $v \in V$ матимемо

$$\int_0^T G(t, u^*(t), x^{u^*}(t))dt + \varphi(x^{u^*}(T)) \geq \int_0^T G(t, u^*(t) + \varepsilon v(t), x^{u^*+\varepsilon v}(t))dt + \varphi(x^{u^*+\varepsilon v}(T)),$$

звідки одержуємо

$$\int_0^T \frac{1}{\varepsilon} \left[G(t, u^*(t) + \varepsilon v(t), x^{u^*+\varepsilon v}(t)) - G(t, u^*(t), x^{u^*}(t)) \right] dt + \frac{1}{\varepsilon} [\varphi(x^{u^*+\varepsilon v}(T)) - \varphi(x^{u^*}(T))] \leq 0,$$

для кожного $v \in V$ і для кожного достатньо малого $\varepsilon > 0$.

Перейшовши до границі в останній нерівності при $\varepsilon \rightarrow 0^+$, одержимо

$$\int_0^T v(t) \cdot G_u(t, u^*(t), x^{u^*}(t)) + z(t) \cdot G_x(t, u^*(t), x^{u^*}(t))dt + z(T) \cdot \varphi_x(x^{u^*}(T)) \leq 0 \quad (8.3.3)$$

для кожного $v \in V$ (тут “ \cdot ” означає скалярний добуток у відповідних просторах \mathbb{R}^m та \mathbb{R}^N).

Нехай p – розв’язок Carathéodory (припускаємо, що цей розв’язок існує і є єдиним), який називатимемо просто розв’язком, *спряженої задачі (рівняння)*

$$\begin{cases} p'(t) = -f_x^*(t, u^*(t), x^{u^*}(t))p(t) - G_x(t, u^*(t), x^{u^*}(t)), \quad t \in (0, T), \\ p(T) = \varphi_x(x^{u^*}(T)) \end{cases} \quad (8.3.4)$$

(p називається *спряженим станом*; рівняння (8.3.4) є лінійним).

Пригадаємо, якщо $A : \mathbb{R}^k \rightarrow \mathbb{R}^s$ є лінійним (і обмеженим) оператором (A може задаватися матрицею, і також позначаємо через A), тоді спряжений оператор $A^* : \mathbb{R}^s \rightarrow \mathbb{R}^k$ (також лінійний і обмежений), може визначатися транспонованою до A матрицею і також позначають A^* (або A^T).

Помножимо (8.3.2) на p і, проінтегрувавши частинами, одержимо

$$z(T) \cdot p(T) - \int_0^T z(t) \cdot p'(t) dt = \int_0^T [f_u(t, u^*(t), x^{u^*}(t))v(t) + f_x(t, u^*(t), x^{u^*}(t))z(t)] \cdot p(t) dt,$$

$\forall v \in V$. Із (8.3.4) маємо

$$z(T) \cdot \varphi_x(x^{u^*}(T)) + \int_0^T z(t) \cdot [f_x^*(t, u^*(t), x^{u^*}(t))p(t) + G_x(t, u^*(t), x^{u^*}(t))] dt = \int_0^T [v(t) \cdot f_u^*(t, u^*(t), x^{u^*}(t))p(t) + z(t) \cdot f_x^*(t, u^*(t), x^{u^*}(t))p(t)] dt,$$

звідки випливає

$$\int_0^T z(t) \cdot G_x(t, u^*(t), x^{u^*}(t)) dt + z(T) \cdot \varphi_x(x^{u^*}(T)) = \int_0^T v(t) \cdot f_u^*(t, u^*(t), x^{u^*}(t))p(t) dt,$$

$\forall v \in V$. І накінець із (8.3.3) одержимо нерівність

$$\int_0^T v(t) \cdot [G_u(t, u^*(t), x^{u^*}(t)) + f_u^*(t, u^*(t), x^{u^*}(t))p(t)] dt \leq 0, \quad \forall v \in V.$$

А це означає, що

$$G_u(\cdot, u^*, x^{u^*}) + f_u^*(\cdot, u^*, x^{u^*})p \in N_K(u^*), \quad (8.3.5)$$

де $N_K(u^*)$ – нормальний конус до K в u^* . Такий самий результат одержимо, якщо (8.3.4) домножимо на z (після відповідних перетворень).

Рівняння (8.3.1), (8.3.4) і (8.3.5) репрезентують принцип Понтрягіна (або максимуму), а (8.3.4) та (8.3.5) є *необхідними умовами оптимальності першого порядку* для цієї задачі оптимального керування.

Наше головне завдання полягає в тому, щоб використати принцип максимуму до обчислення оптимального керування u^* або апроксимувати його, використовуючи апріорні чисельні методи. Для того, щоб використати (8.3.5), ми повинні визначити множину $N_K(u^*)$.

Якщо взяти, наприклад, $K \in L^2(0, T; \mathbb{R}^m)$, тоді $\forall u \in K = L^2(0, T; \mathbb{R}^m)$, $N_K(u) = \{0\} \subset L^2(0, T; \mathbb{R}^m)$.

Якщо візьмемо $m = 1$ і

$$K = \{w \in L^2(0, T); L_1 \leq w(t) \leq L_2, t \in (0, T)\},$$

де $L_1, L_2 \in \mathbb{R}$, $L_1 < L_2$, тоді для кожного $u \in K$ отримаємо

$$N_K(u) = \{w \in L^2(0, T) : w(t) \geq 0 \text{ якщо } u(t) = L_2; w(t) \leq 0, \\ \text{якщо } u(t) = L_1; w(t) = 0, \text{ якщо } L_1 < u(t) < L_2, \forall t \in (0, T)\}.$$

Загальна схема доведення існування оптимального керування така.

Нехай

$$d = \sup_{u \in K} \mathcal{L}(u, x^u) \in \mathbb{R}.$$

Тоді для кожного $n \in \mathbb{N}^*$ існує $u_n \in K$ таке, що

$$d - \frac{1}{n} < \mathcal{L}(u_n, x^{u_n}) \leq d.$$

Крок 1. Доведення існування підпослідовності $\{u_{n_k}\}$ такої, що

$$u_{n_k} \rightarrow u^* \text{ слабко у } L^2(0, T; \mathbb{R}^m).$$

Якщо, наприклад, K є обмеженою множиною, тоді такий висновок виконується відразу. Оскільки K – випукла замкнута підмножина в $L^2(0, T; \mathbb{R}^m)$, то K є слабко замкнутою, звідки впливає $u^* \in K$.

Крок 2. Доведення існування підпослідовності $\{x^{u_{n_k}}\}$, яку позначимо $\{x^{u_{nr}}\}$, збіжної до x^{u^*} в просторі $C(0, T; \mathbb{R}^N)$ (в деяких випадках

достатньо збіжності у $L^2(0, T; \mathbb{R}^N)$.

Крок 3. Із оцінки

$$d - \frac{1}{n_r} < \mathcal{L}(u_{n_r}, x^{u_{n_r}}) < d,$$

перейшовши до границі, одержимо

$$\mathcal{L}(u^*, x^{u^*}) = d,$$

звідки випливає, що u^* є розв'язком задачі (P_1) .

Зауважимо, що (8.3.4) і (8.3.5) можемо записати через Hamiltonian H

$$H(t, u, x, p) = G(t, u, x) + f(t, u, x) \cdot p.$$

Тоді у термінах Hamiltonian'а матимемо рівняння стану

$$x' = H_p,$$

а спряженим буде рівняння

$$p' = -H_x$$

і (8.3.5) можемо записати у вигляді

$$H_u \in N_K(u^*).$$

Зазначимо, що деякі автори таку задачу розглядають як спряжену

$$\begin{cases} p'(t) = -f_x^*(t, u^*(t), x^{u^*}(t))p(t) + G_x(t, u^*(t), x^{u^*}(t)), & t \in (0, T), \\ p(T) = -\varphi_x(x^{u^*}(T)). \end{cases}$$

Розв'язком цієї задачі є $p = -\tilde{p}$, де \tilde{p} – розв'язок (8.3.4). Умова (8.3.5) набуває вигляду

$$G_u(\cdot, u^*, x^{u^*}) - f_u^*(\cdot, u^*, x^{u^*})p \in N_K(u^*),$$

а Hamiltonian, відповідно, запишеться

$$H(t, u, x, p) = -G(t, u, x) + f(t, u, x) \cdot p.$$

У наступному розділі використовуються обидва означення для спряженої задачі.

У більшості випадках (8.3.1) є напівлінійною задачею. Тоді f набуває наступного вигляду

$$f(t, u, x) = Ax + \tilde{f}(t, u, x),$$

де $A : \mathbb{R}^N \rightarrow \mathbb{R}^n$ є, зокрема, лінійним оператором. Тоді

$$f_u = \tilde{f}_u, \quad f_x = A + \tilde{f}_x.$$

Деякі задачі оптимального керування, пов'язані з віково-структурованими моделями, напівлінійними параболічними рівняннями, інтегродиференціальними рівняннями параболічного типу, в абстрактній формі можуть бути записані у вигляді (P_1) , (8.3.1), де

$$\begin{aligned} G : [0, T] \times U \times X &\rightarrow \mathbb{R}, \\ \varphi : X &\rightarrow \mathbb{R}, \end{aligned}$$

(U, X апіорі є дійсними просторами Hilbert'a), $x_0 \in X$, і $K \subset L^2(0, T; U)$ є замкненою опуклою підмножиною. Тут f має наведений вище вигляд, а A є лінійним (загалом необмеженим) оператором, $A : D(A) \subset X \rightarrow X$.

Опишемо тільки загальну схему знаходження розв'язку задачі оптимального керування без строгого доведення принципу максимуму, яке, наприклад, можна знайти в працях [78, 79, 80].

У наступних розділах з'ясуємо як за допомогою цієї схеми отримати розв'язки на прикладі задач оптимального керування, які виникають у науці про життя, економічному регулюванні тощо й описуються звичайними диференціальними рівняннями, та рівняннями з частинними похідними. У всіх наведених прикладах ми строго будемо застосовувати принцип максимуму.

8.3.2. Задача максимізації загального споживання

Розглянемо математичну модель спрощеної економіки. Нехай $x(t)$ означає продуктивність виробництва у момент часу t (випуск продукції). Тоді

$$x(t) = I(t) + C(t), \quad t \geq 0,$$

де $I(t)$ – капіталовкладення в момент часу t ; $C(t)$ – норма споживання в момент часу t . Позначимо через $u(t) \in [0, 1]$ частку продукції $x(t)$, призначену на інвестування в момент часу t , тому

$$I(t) = u(t)x(t).$$

Тоді для $C(t)$ отримаємо

$$C(t) = (1 - u(t))x(t), \quad t \geq 0.$$

Розглянемо простий випадок, коли ріст продуктивності пропорціональний до капіталовкладення. Це означає, що

$$x'(t) = \gamma u(t)x(t), \quad t \geq 0,$$

де $\gamma \in (0, +\infty)$.

Введемо функцію "корисності" $F(C)$ і шукатимемо керування, яке максимізує сукупний добробут

$$\int_0^T e^{\delta t} F(C(t)) dt.$$

Тут $T > 0$, $\delta \geq 0$ – дисконтна ставка (міра переваги споживання в минулому над майбутнім).

Спростимо нашу модель, прийнявши $F(C) = C$ та $\delta = 0$. Сумарне споживання на часовому проміжку $[0, T]$ становить

$$\int_0^T C(t) dt = \int_0^T (1 - u(t))x(t) dt.$$

Отож, отримаємо таку задачу оптимального керування [19]

$$\int_0^T (1 - u(t))x^u(t) dt \rightarrow \max \quad (P_2)$$

стосовно $u \in L^2(0, T)$, $0 \leq u(t) \leq 1$ для $t \in (0, T)$, де x^u – розв'язок початкової задачі

$$\begin{cases} x'(t) = \gamma u(t)x(t), & t \in (0, T), \\ x(0) = x_0 > 0. \end{cases} \quad (8.3.6)$$

Задача полягає у знаходженні такого керування u , яке максимізує сукупне споживання на проміжку $[0, T]$.

Розв'язок x^u задачі (8.3.6) записується явно

$$x^u(t) = x_0 \exp \left(\int_0^t \gamma u(s) ds \right), \quad t \in [0, T].$$

Задача (P_2) є частинним випадком задачі (P_1) для $m = 1$, $N = 1$,

$$\begin{aligned} G(t, u, x) &= (1 - u)x, \\ \varphi(x) &= 0, \\ f(t, u, x) &= \gamma ux, \end{aligned}$$

$$K = \{w \in L^2(0, T), 0 \leq w(t) \leq 1, t \in (0, T)\}.$$

Існування оптимальної пари задачі (P_2)

Позначимо

$$\Phi(u) = \int_0^T (1 - u(t))x^u(t)dt, \quad u \in K$$

і приймаємо

$$d = \sup_{u \in K} \Phi(u).$$

Оскільки для кожного $u \in K$ правильні оцінки

$$0 < x^u(t) \leq x_0 e^{\gamma t}, \quad t \in [0, T],$$

то одержимо

$$0 \leq \Phi(u) = \int_0^T (1 - u(t))x^u(t)dt \leq x_0 T e^{\gamma T}.$$

У підсумку $d \in (0, +\infty)$.

Для кожного $n \in \mathbb{N}^*$ існує $u_n \in K$, такі, що

$$d - \frac{1}{n} < \Phi(u_n) \leq d. \quad (8.3.7)$$

Підмножина K є обмеженою у $L^2(0, T)$, звідки випливає існування під-послідовності $\{u_{n_k}\}_{k \in \mathbb{N}^*}$ слабо збіжної у $L^2(0, T)$

$$u_{n_k} \rightarrow u^*. \quad (8.3.8)$$

Границя u^* належить до K , позаяк K – замкнута опукла підмножина у $L^2(0, T)$, а отже, слабо замкнута. З останньої збіжності та явного зображення для x^u отримуємо, що

$$x^{u_{n_k}} \rightarrow x^{u^*} \quad \text{у} \quad L^2(0, T). \quad (8.3.9)$$

Із (8.3.7) отримуємо таке

$$d - \frac{1}{n_k} < \int_0^T (1 - u_{n_k}(t))x^{u_{n_k}}(t)dt \leq d \quad \text{для кожного } k \in \mathbb{N}^*. \quad (8.3.10)$$

Використовуючи (8.3.8) та (8.3.9), перейдемо до границі у (8.3.10), внаслідок чого отримаємо

$$d = \int_0^T (1 - u^*(t))x^{u^*}(t)dt,$$

а це означає, що (u^*, x^{u^*}) є *оптимальною парою* (і відповідно u^* оптимальним керуванням) задачі (P_2) . Для простоти запису позначимо $x^* := x^{u^*}$.

Принцип максимуму

Для кожної довільної, але фіксованої функції $v \in V = \{w \in L^2(0, T); u^* + \varepsilon w \in K \text{ для кожного достатньо малого } \varepsilon > 0\}$ позначимо через z розв'язок початкової задачі

$$\begin{cases} z'(t) = \gamma u^*(t)z(t) + \gamma v(t)x^*(t), & t \in (0, T), \\ z(0) = 0. \end{cases} \quad (8.3.11)$$

Розв'язок початкової задачі (8.3.11) для лінійного рівняння першого порядку записується у явному вигляді

$$z(t) = \int_0^t \exp\left(\int_s^t \gamma u^*(\tau)d\tau\right) \gamma v(s)x^*(s)ds, \quad t \in [0, T]. \quad (8.3.12)$$

Оскільки правильно

$$\int_0^T (1 - u^*(t))x^*(t)dt \geq \int_0^T (1 - u^*(t) - \varepsilon v(t))x^{u^* + \varepsilon v}(t)dt,$$

для кожного достатньо малого додатного ε , то отримаємо

$$\int_0^T \left[(1 - u^*(t)) \frac{x^{u^* + \varepsilon v}(t) - x^*(t)}{\varepsilon} - v(t)x^{u^* + \varepsilon v}(t) \right] dt \leq 0. \quad (8.3.13)$$

Доведемо правильність таких граничних переходів

$$x^{u^*+\varepsilon v} \rightarrow x^* \quad \text{у } C([0, T]),$$

і

$$\frac{x^{u^*+\varepsilon v} - x^*}{\varepsilon} \rightarrow z \quad \text{у } C([0, T]),$$

якщо $\varepsilon \rightarrow 0^+$.

Справді, для кожного достатньо малого $\varepsilon > 0$ маємо

$$\begin{aligned} x^{u^*+\varepsilon v}(t) &= x_0 \exp \left(\gamma \int_0^t (u^*(s) + \varepsilon v(s)) ds \right) = \\ &= x^{u^*}(t) \exp \left(\varepsilon \gamma \int_0^t v(s) ds \right), \quad t \in [0, T], \end{aligned}$$

звідки випливає

$$|x^{u^*+\varepsilon v}(t) - x^{u^*}(t)| = |x^{u^*}(t)| \cdot \left| \exp \left(\varepsilon \gamma \int_0^t v(s) ds \right) - 1 \right|, \quad t \in [0, T].$$

Оскільки

$$\left| \exp \left(\varepsilon \gamma \int_0^t v(s) ds \right) - 1 \right| \xrightarrow{\varepsilon \rightarrow 0^+} 0,$$

рівномірно на $[0, T]$, то можемо зробити висновок, що

$$x^{u^*+\varepsilon v} \xrightarrow{\varepsilon \rightarrow 0^+} x^* \quad \text{у } C([0, T]).$$

Для кожного достатньо малого $\varepsilon > 0$ розглянемо

$$w_\varepsilon(t) = \frac{x^{u^*+\varepsilon v} - x^*}{\varepsilon} - z(t), \quad t \in [0, T].$$

Функція w_ε є розв'язком початкової задачі для лінійного рівняння першого порядку

$$\begin{cases} w'(t) = \gamma u^*(t)w(t) + \gamma v(t) (x^{u^*+\varepsilon v}(t) - x^{u^*}(t)), & t \in (0, T), \\ w(0) = 0, \end{cases}$$

і записується у явному вигляді

$$w(t) = \gamma \int_0^t v(s) e^{\gamma \int_s^t u^*(\tau) d\tau} [x^{u^*+\varepsilon v}(s) - x^{u^*}(s)] ds, \quad t \in [0, T].$$

Взявши до уваги збіжність $x^{u^*+\varepsilon v}$ до x^* при $\varepsilon \rightarrow 0^+$, із останньої формули отримуємо, що у $C([0, T])$ є

$$w_\varepsilon \rightarrow 0,$$

звідки випливає

$$\frac{x^{u^*+\varepsilon v} - x^*}{\varepsilon} \rightarrow z \quad \text{у просторі } C([0, T]).$$

Тепер із (8.3.13) отримуємо

$$\int_0^T [(1 - u^*(t))z(t) - v(t)x^*(t)] dt \leq 0. \quad (8.3.14)$$

Нехай p – розв'язок початкової задачі

$$\begin{cases} p'(t) = -\gamma u^*(t)p(t) + u^*(t) - 1, & t \in (0, T), \\ p(T) = 0, \end{cases} \quad (8.3.15)$$

і записується у вигляді

$$p(t) = - \int_0^t e^{\int_t^s \gamma u^*(\tau) d\tau} (u^*(s) - 1) ds, \quad t \in [0, T].$$

Домножимо диференціальне рівняння (8.3.15) на z та проінтегруємо на проміжку $[0, T]$

$$\int_0^T p'(t)z(t) dt = - \int_0^T \gamma u^*(t)p(t)z(t) dt + \int_0^T (u^*(t) - 1)z(t) dt.$$

Інтегруючи зліва частинами (використаємо (8.3.11) та (8.3.15)), останню рівність запишемо у вигляді

$$- \int_0^T p(t)z'(t) dt = - \int_0^T \gamma u^*(t)p(t)z(t) dt + \int_0^T (u^*(t) - 1)z(t) dt.$$

Зліва в останній рівності за z' підставимо праву частину (8.3.11)

$$\begin{aligned} - \int_0^T \gamma u^*(t) z(t) p(t) dt - \int_0^T \gamma v(t) x^*(t) p(t) dt = \\ - \int_0^T \gamma u^*(t) p(t) z(t) dt + \int_0^T (u^*(t) - 1) z(t) dt, \end{aligned}$$

звідки випливає

$$\int_0^T (1 - u^*(t)) z(t) dt = \int_0^T \gamma v(t) x^*(t) p(t) dt.$$

З останнього співвідношення та нерівності (8.3.14) отримаємо

$$\int_0^T x^*(t) (\gamma p(t) - 1) v(t) dt \leq 0, \quad (8.3.16)$$

для кожної функції $v \in V$. Остання нерівність еквівалентна твердженню

$$(\gamma p - 1)x^* \in N_K(u^*).$$

Взявши до уваги структуру $N_K(u^*)$, можемо зробити висновок, що

$$u^*(t) = \begin{cases} 0, & \text{якщо } \gamma p(t) - 1 < 0, \\ 1 & \text{якщо } \gamma p(t) - 1 > 0, \end{cases} \quad (8.3.17)$$

для майже всіх $t \in (0, T)$. Доведемо зараз безпосередньо (8.3.17), опираючись на (8.3.16). Уведемо позначення

$$A = \{t \in (0, T); \gamma p(t) - 1 < 0\}.$$

Доведемо, що $u^*(t) = 0$ майже всюди на A .

Від супротивного, існує підмножина $\tilde{A} \subset A$, міра Lebesgue'а, якої є додатною ($meas(\tilde{A}) > 0$), така, що $u^*(t) > 0$ майже всюди на \tilde{A} . Тоді можна вибрати таку функцію $v \in L^2(0, T)$, $v(t) < 0$ майже всюди в \tilde{A} , $v(t) = 0$ майже всюди на множині $(0, T) \setminus \tilde{A}$ і $0 < u^*(t) - \varepsilon v(t) \leq 1$ майже всюди на $(0, T)$. За зазначених умов отримуємо

$$\int_0^T x^*(t) (\gamma p(t) - 1) v(t) dt = \int_{\tilde{A}} x^*(t) (\gamma p(t) - 1) v(t) dt > 0,$$

позаяк $v(t) < 0$, $\gamma p(t) - 1 < 0$, $x^*(t) > 0$ на \tilde{A} та $meas\tilde{A} > 0$. Остання нерівність суперечить (8.3.16), що і завершує доведення. Аналогічно доводиться, що

$$u^*(t) = 1 \quad \text{майже всюди для } t \in \{s \in (0, T); \gamma p(s) - 1 > 0\}.$$

Зробимо підсумок отриманих результатів як відповідне зауваження.

♣ **Зауваження 8.3.1.** За допомогою рівнянь (8.3.6), (8.3.15) і (8.3.17) сформульовано принцип максимуму, а (8.3.15), (8.3.17) – необхідні умови оптимальності першого порядку задачі (P₂).

Обчислення оптимального керування u^*

На наступному етапі застосуємо принцип Понтрягіна для того, щоб отримати більше інформації про оптимальне керування u^* . З'ясуємо як у частинних випадках обчислити оптимальний розв'язок.

Нехай $(T - \eta, T]$ ($\eta > 0$) – максимальний інтервал, на якому неперервна функція p задовольняє нерівність $\gamma p(t) < 1$. Тоді з (8.3.17) і (8.3.15) легко бачити, що

$$p'(t) = -1, \quad t \in [T - \eta, T],$$

звідки одержуємо

$$p(t) = T - t, \quad t \in [T - \eta, T].$$

Крім того, якщо $\gamma T > 1$, тоді отримаємо

$$p(t) = T - t, \quad t \in [T - 1/\gamma, T]$$

і

$$u^*(t) = 0 \quad \text{майже всюди для } t \in (T - 1/\gamma, T).$$

Позаяк $p(T - 1/\gamma) = 1/\gamma$, правильна нерівність $p'(t) \leq 0$ на інтервалі максимальної довжини $(T - 1/\gamma - \delta, T - 1/\gamma]$ ($\delta > 0$), і крім того, $\gamma p(t) > 1$ на цьому інтервалі. Звідси випливає також, що на інтервалі $(T - 1/\gamma - \delta, T - 1/\gamma)$

$$\begin{cases} p'(t) = \gamma p(t), \\ u^*(t) = 1. \end{cases}$$

Звідки випливає, що

$$p(t) = \frac{1}{\gamma} \exp \left\{ \gamma \left(T - \frac{1}{\gamma} - t \right) \right\} \quad t \in [T - 1/\gamma - \delta, T - 1/\gamma].$$

Це означає, що $\delta = T - 1/\gamma$ і $u^*(t) = 1$ майже всюди для $t \in [0, T - 1/\gamma)$

Отож, у підсумку отримали:

- якщо $\gamma T > 1$, тоді

$$u^*(t) = \begin{cases} 1, & \text{якщо } t \in [0, T - 1/\gamma), \\ 0, & \text{якщо } t \in [T - 1/\gamma, T]; \end{cases} \quad (8.3.18)$$

- якщо $\gamma T \leq 1$, тоді

$$u^*(t) = 0, \quad t \in [0, T]. \quad (8.3.19)$$

Висновки з отриманих результатів такі: якщо часовий інтервал достатньо великий, тоді на деякому проміжку часу капіталовкладення максимальні. Опісля ми не інвестуємо нічого (тільки споживаємо).

Керування u^* , множина значень якого є скінченною $\{\alpha_1, \alpha_2, \dots, \alpha_k\}$, і крім того, $(u^*)^{-1}(\alpha_i)$ – вимірна множина для кожного $i \in \{1, \dots, k\}$, називається двопозиційним керуванням ("bang-bang control").

Якщо існують $t_0 < t_1 < \dots < t_k$ такі, що u^* є сталим на кожному інтервалі (t_{i-1}, t_i) ($i = \overline{1, k}$), тоді u^* є двопозиційним керуванням на (t_0, t_k) і t_1, t_2, \dots, t_{k-1} називаються точками перемикання.

♣ **Зауваження 8.3.2.** (i) У розглянутому прикладі оптимальне керування двопозиційне і має тільки одну точку перемикання, а саме $T - 1/\gamma$.

(ii) У наведеному прикладі нам вдалося знайти оптимальне керування, яке визначене формулами (8.3.18) та (8.3.19). Це, звичайно, винятковий випадок.

(iii) Формально принцип максимуму Понтрягіна ми в змозі написати після визначення \mathcal{L} , G , φ , f і K , що було доведено і застосовано до визначення оптимального керування.

8.3.3. Максимізація генеральної сукупності у системі хижак-жертва

Система Lotka-Volterra

$$\begin{cases} x'(t) = r_1 x(t) - \mu_1 x(t)y(t), & t \in (0, T), \\ y'(t) = -r_2 y(t) + \mu_2 x(t)y(t), & t \in (0, T), \end{cases}$$

($T > 0$) описує динаміку системи хижак-жертва на часовому інтервалі $(0, T)$. Тут $x(t)$ описує густоту популяції хижаків у момент часу t , а $y(t)$ – густоту популяції жертви у момент часу t :

- $r_1 > 0$ відповідає природному темпу росту жертви та відсутність хижаків;

- $r_2 > 0$ означає швидкість зменшення популяції хижаків і відсутність жертви;
- μ_1 та μ_2 – додатні сталі; $\mu_1 y(t)$ означає додатковий коефіцієнт смертності жертви в момент часу t , обумовлений хижаками (і є пропорціональним до густоти популяції хижаків); $\mu_2 x(t)$ є додатковим темпом росту жертви у момент часу t , зумовлений наявністю хижаків (і є пропорціональним до густоти популяції жертви).

Загальнішу модель хижак-жертва розглядали у підрозділі 8.2.1.

Якщо жертви частково відокремлені від хижаків, то функціональна залежність щодо хижаків набуває зміни і система записується у вигляді

$$\begin{cases} x'(t) = r_1 x(t) - \mu_1 u(t)x(t)y(t), & t \in (0, T), \\ y'(t) = -r_2 y(t) + \mu_2 u(t)x(t)y(t), & t \in (0, T), \end{cases} \quad (8.3.20)$$

де $1 - u(t)$ означає рівень відокремлення в момент часу t ($0 \leq u(t) \leq 1$). Розглянемо початкові умови

$$\begin{cases} x(0) = x_0 > 0, \\ y(0) = y_0 > 0. \end{cases} \quad (8.3.21)$$

Нас цікавить максимізація сукупної кількості особин обидвох популяцій у момент часу $T > 0$. Задача може бути сформульована так (див. [19, 73]): максимізувати

$$\{x^u(T) + y^u(T)\} \rightarrow \max \quad (P_3)$$

стосовно $u \in L^2(0, T)$, $0 \leq u(t) \leq 1$ майже всюди для $t \in (0, T)$, де (x^u, y^u) є розв'язком задачі (8.3.20), (8.3.21).

Задача (P₃) є частковим випадком задачі (P₁) при $m = 1$, $N = 2$,

$$\begin{aligned} G(t, u, (x, y)) &= 0, \\ \varphi(x, y) &= x + y, \end{aligned}$$

$$f(t, u, (x, y)) = \begin{pmatrix} r_1 x - \mu_1 u x y \\ -r_2 y + \mu_2 u x y \end{pmatrix}$$

i

$$K = \{w \in L^2(0, T), 0 \leq w \leq 1 \text{ майже всюди } t \in (0, T)\}.$$

Існування оптимальної пари для задачі (P₃)

Позначимо

$$\Phi(u) = x^u(T) + y^u(T), \quad u \in K,$$

і нехай

$$d = \sup_{u \in K} \Phi(u).$$

Очевидно, що $d \in [0, +\infty)$. Для кожного $n \in \mathbb{N}^*$ існує таке $u_n \in K$, що

$$d - \frac{1}{n} < \Phi(u_n) \leq d.$$

Оскільки

$$x^{u_n}(t) = x_0 \exp \left\{ \int_0^t (r_1 - \mu_1 u(s) y^{u_n}(s)) ds \right\} > 0,$$

$$y^{u_n}(t) = y_0 \exp \left\{ \int_0^t (-r_2 + \mu_2 u(s) x^{u_n}(s)) ds \right\} > 0,$$

для $t \in [0, T]$, то отримуємо, що $x^{u_n}(t), y^{u_n}(t) > 0$ для кожного $t \in [0, T]$, і, крім того,

$$0 \leq (x^{u_n})'(t) \leq r_1 x^{u_n}(t) \quad \text{майже всюди } t \in (0, T).$$

Звідки випливає

$$0 \leq x^{u_n}(t) \leq x_0 e^{r_1 T}, \quad t \in [0, T],$$

і, крім того, $\{(x^{u_n})'\}_n$ обмежена у $L^\infty(0, T)$. З іншого боку маємо

$$0 \leq y^{u_n}(t) \leq y_0 \exp\{(-r_2 + \mu_2 x_0 e^{r_1 T})T\}, \quad t \in [0, T],$$

і, крім того, послідовність $\{(y^{u_n})'\}_n$ є обмеженою у $L^\infty(0, T)$. Звідси випливає, що послідовності $\{x^{u_n}\}_n$ і $\{y^{u_n}\}_n$ є обмеженими у $C([0, T])$ та рівномірно рівностепенно неперервні. За теоремою Arzelà та взявши до уваги, що послідовність $\{u_n\}_n$ обмежена у $L^2(0, T)$, для підпослідовності отримаємо, що

$$\begin{aligned} u_{n_k} &\rightarrow u^* \quad \text{слабко } L^2(0, T), \\ x^{u_{n_k}} &\rightarrow x^* \quad \text{на } C([0, T]), \\ y^{u_{n_k}} &\rightarrow y^* \quad \text{на } C([0, T]), \end{aligned} \tag{8.3.22}$$

($u^* \in K$, оскільки K є замкненою опуклою підмножиною $L^2(0, T)$, звідки випливає, що вона слабо замкнена).

Оскільки

$$x^{u_{n_k}}(t) = x_0 + \int_0^t (r_1 x^{u_{n_k}}(s) - \mu_1 u_{n_k} x^{u_{n_k}}(s) y^{u_{n_k}}(s)) ds,$$

$$y^{u_{n_k}}(t) = y_0 + \int_0^t (-r_2 y^{u_{n_k}}(s) + \mu_2 u_{n_k} x^{u_{n_k}}(s) y^{u_{n_k}}(s)) ds,$$

для кожного $t \in [0, T]$, звідки, взявши до уваги (8.3.22), отримаємо

$$x^*(t) = x_0 + \int_0^t (r_1 x^*(s) - \mu_1 u^* x^*(s) y^*(s)) ds,$$

$$y^*(t) = y_0 + \int_0^t (-r_2 y^*(s) + \mu_2 u^* x^*(s) y^*(s)) ds,$$

для кожного $t \in [0, T]$. А це означає, що (x^*, y^*) є розв'язком задачі (8.3.20), (8.3.21), що відповідає u^* (тобто, $x^* = x^{u^*}$, $y^* = y^{u^*}$). З іншого боку, з нерівностей

$$d - \frac{1}{u_{n_k}} < x^{u_{n_k}}(T) + y^{u_{n_k}}(T) \leq d \quad \forall k \in \mathbb{N}^*,$$

використовуючи (8.3.22), граничним переходом отримаємо

$$d = x^*(T) + y^*(T),$$

а це означає, що u^* – розв'язок задачі (P₃); $(u^*, (x^*, y^*))$ є оптимальною парою для задачі (P₃), тобто, u^* є оптимальним керуванням і $x^* = x^{u^*}$, $y^* = y^{u^*}$.

Принцип максимуму для задачі (P₃)

Для кожної довільної $v \in V = \{w \in L^2(0, T) : u^* + \varepsilon w \in K, \forall \varepsilon > 0 \text{ достатньо малого}\}$ розглянемо початкову задачу, де (z_1, z_2) – її розв'язок

$$\begin{cases} z_1' = r_1 z_1 - \mu_1 u^* z_1 y^* - \mu_1 u^* x^* z_2 - \mu_1 v x^* y^*, & t \in (0, T), \\ z_2' = -r_2 z_2 + \mu_2 u^* z_1 y^* + \mu_2 u^* x^* z_2 + \mu_2 v x^* y^*, & t \in (0, T), \\ z_1(0) = z_2(0) = 0. \end{cases} \quad (8.3.23)$$

Оскільки правильна нерівність

$$x^*(T) + y^*(T) \geq x^{u^*+\varepsilon v}(T) + y^{u^*+\varepsilon v}(T),$$

то отримуємо

$$\frac{x^{u^*+\varepsilon v}(T) - x^*(T)}{\varepsilon} + \frac{y^{u^*+\varepsilon v}(T) - y^*(T)}{\varepsilon} \leq 0, \quad (8.3.24)$$

для достатньо малого значення $\varepsilon > 0$.

Для довільного достатньо малого додатного значення ε функція $x^{u^*+\varepsilon v}$ задовольняє оцінку

$$(x^{u^*+\varepsilon v})'(t) \leq r_1 x^{u^*+\varepsilon v}(t) \text{ для майже всіх } t \in (0, T),$$

то звідси, відповідно, випливає, що існує така стала величина $M \in (0, +\infty)$, що справджується нерівність

$$0 \leq x^{u^*+\varepsilon v}(t) \leq M \quad \forall t \in [0, T],$$

для кожного достатньо малого значення $\varepsilon > 0$. З іншого боку, маємо

$$(y^{u^*+\varepsilon v})'(t) \leq (-r_2 + M\mu_2)y^{u^*+\varepsilon v}(t) \text{ для майже всіх } t \in (0, T),$$

звідки робимо висновок, що послідовність $\{y^{u^*+\varepsilon v}\}$ – обмежена на $C([0, T])$ (для кожного достатньо малого $\varepsilon > 0$). Отже, обидві послідовності $\{x^{u^*+\varepsilon v}\}$ та $\{y^{u^*+\varepsilon v}\}$ рівномірно обмежені та рівномірно неперервні на $[0, T]$. За теоремою Arzelà отримаємо, що для кожної послідовності $\varepsilon_n \searrow 0$ правильні граничні переходи

$$\begin{aligned} x^{u^*+\varepsilon_n v} &\rightarrow \tilde{x} \text{ у } C([0, T]), \\ y^{u^*+\varepsilon_n v} &\rightarrow \tilde{y} \text{ у } C([0, T]). \end{aligned} \quad (8.3.25)$$

Позаяк для $x^{u^*+\varepsilon_n v}$ та $y^{u^*+\varepsilon_n v}$ правильні подання

$$x^{u^*+\varepsilon_n v}(t) = x_0 +$$

$$\int_0^t [r_1 x^{u^*+\varepsilon_n v}(s) - \mu_1(u^*(s) + \varepsilon_n v(s)) x^{u^*+\varepsilon_n v}(s) y^{u^*+\varepsilon_n v}(s)] ds,$$

$$y^{u^*+\varepsilon_n v}(t) = y_0 +$$

$$\int_0^t [-r_2 y^{u^*+\varepsilon_n v}(s) + \mu_2(u^*(s) + \varepsilon_n v(s)) x^{u^*+\varepsilon_n v}(s) y^{u^*+\varepsilon_n v}(s)] ds,$$

для кожного $t \in [0, T]$, то виконавши граничний перехід (скористаємося (8.3.25)), отримаємо

$$\begin{aligned}\tilde{x}(t) &= x_0 + \int_0^t (r_1 \tilde{x}(s) - \mu_1 u^*(s) \tilde{x}(s) \tilde{y}(s)) ds, \\ \tilde{y}(t) &= y_0 + \int_0^t (-r_2 \tilde{y}(s) + \mu_2 u^*(s) \tilde{x}(s) \tilde{y}(s)) ds,\end{aligned}$$

для кожного $t \in [0, T]$. А це означає, що (\tilde{x}, \tilde{y}) є розв'язком задачі (8.3.20), (8.3.21), яке відповідає керуванню u^* ; де $\tilde{x} = x^{u^*}$, $\tilde{y} = y^{u^*}$.

Уведемо позначення

$$\begin{aligned}\alpha_n(t) &:= \frac{1}{\varepsilon_n} [x^{u^* + \varepsilon_n v}(t) - x^*(t)] - z_1(t), \quad t \in [0, T], \\ \beta_n(t) &:= \frac{1}{\varepsilon_n} [y^{u^* + \varepsilon_n v}(t) - y^*(t)] - z_2(t), \quad t \in [0, T].\end{aligned}$$

$(\alpha_n(t), \beta_n(t))$ є розв'язком початкової задачі

$$\begin{cases} \alpha'_n = r_1 \alpha_n - \mu_1 u^* \alpha_n y^* - \mu_1 u^* x^* \beta_n + f_{1n}(t), & t \in (0, T), \\ \beta'_n = -r_2 \beta_n + \mu_1 u^* \alpha_n y^* + \mu_2 u^* x^* \beta_n + f_{2n}(t), & t \in (0, T), \\ \alpha_n(0) = \beta_n(0) = 0 \end{cases}$$

і правильні граничні переходи $f_{1n} \rightarrow 0$, $f_{2n} \rightarrow 0$ у просторі $L^\infty(0, T)$.

Звідси отримуємо

$$\begin{aligned}\alpha_n(t)^2 + \beta_n(t)^2 &\leq c \int_0^t [\alpha_n(s)^2 + \beta_n(s)^2] ds + \\ &2 \int_0^t [f_{1n}(s) \alpha_n(s) + f_{2n}(s) \beta_n(s)] ds \leq \\ &(c+1) \int_0^t [\alpha_n(s)^2 + \beta_n(s)^2] ds + \int_0^T [f_{1n}(s)^2 + f_{2n}(s)^2] ds,\end{aligned}$$

$t \in [0, T]$, $c > 0$ – незалежна від n стала. Застосовуючи до цієї нерівності лему Bellman'а (візьмемо до уваги умови $\alpha_n(0) = \beta_n(0) = 0$),

отримаємо таку нерівність

$$0 \leq \alpha_n(t)^2 + \beta_n(t)^2 \leq e^{(c+1)t} \int_0^T [f_{1n}(s)^2 + f_{2n}(s)^2] ds,$$

для кожного значення $t \in [0, T]$. Перейшовши до границі, отримаємо, що

$$\alpha_n \rightarrow 0, \quad \beta_n \rightarrow 0 \text{ у } C([0, T]).$$

Звідси впливає правильність граничних переходів

$$\begin{aligned} \frac{x^{u^*+\varepsilon_n v} - x^*}{\varepsilon_n} &\rightarrow z_1 \text{ у } C([0, T]), \\ \frac{y^{u^*+\varepsilon_n v} - y^*}{\varepsilon_n} &\rightarrow z_2 \text{ у } C([0, T]). \end{aligned}$$

Використавши тепер (8.3.24) одержимо остаточно

$$z_1(T) + z_2(T) \leq 0. \quad (8.3.26)$$

Нехай тепер (p_1, p_2) є розв'язком початкової задачі

$$\begin{cases} p_1' = -r_1 p_1 + \mu_1 u^* p_1 y^* - \mu_2 u^* y^* p_2, & t \in (0, T), \\ p_2' = r_2 p_2 + \mu_1 u^* p_1 x^* - \mu_2 u^* x^* p_2, & t \in (0, T), \\ p_1(T) = p_2(T) = 1. \end{cases} \quad (8.3.27)$$

Домножимо перше рівняння (8.3.27) на z_1 , а друге – на z_2 , додамо сторонами і після інтегрування на проміжку $[0, T]$ отримаємо інтегральну рівність

$$\begin{aligned} &\int_0^T [p_1'(t)z_1(t) + p_2'(t)z_2(t)] dt = \\ &\int_0^T [-r_1 p_1(t)z_1(t) + \mu_1 u^*(t)y^*(t)p_1(t)z_1(t) - \mu_2 u^*(t)y^*(t)p_2(t)z_1(t) + \\ &\quad \mu_1 u^*(t)x^*(t)p_1(t)z_2(t) - \mu_2 u^*(t)x^*(t)p_2(t)z_2(t) + r_2 p_2(t)z_2(t)] dt. \end{aligned}$$

Проінтегруємо в останній рівності частинами, використавши (8.3.23), після очевидних перетворень отримаємо таку рівність:

$$p_1(T)z_1(T) + p_2(T)z_2(T) - p_1(0)z_1(0) + p_2(0)z_2(0) = \int_0^T x^*(t)y^*(t)v(t)[\mu_2 p_2(t) - \mu_1 p_1(t)]dt,$$

звідки, і з (8.3.23) та (8.3.26), одержимо

$$z_1(T) + z_2(T) = \int_0^T x^*(t)y^*(t)v(t)[\mu_2 p_2(t) - \mu_1 p_1(t)]dt \leq 0,$$

для кожної $v \in V$. Звідси випливає (аналогічно як у попередньому підрозділі), що

$$u^*(t) = \begin{cases} 0, & \text{якщо } x^*(t)y^*(t)[\mu_2 p_2(t) - \mu_1 p_1(t)] < 0, \\ 1 & \text{якщо } x^*(t)y^*(t)[\mu_2 p_2(t) - \mu_1 p_1(t)] > 0, \end{cases}$$

майже всюди на $(0, T)$. Оскільки $x_0, y_0 > 0$ і x^* та y^* є додатними функціями, то остаточно отримуємо

$$u^*(t) = \begin{cases} 0, & \text{якщо } \mu_2 p_2(t) - \mu_1 p_1(t) < 0, \\ 1 & \text{якщо } \mu_2 p_2(t) - \mu_1 p_1(t) > 0, \end{cases} \quad (8.3.28)$$

майже всюди на $(0, T)$.

Рівняння (8.3.27) і (8.3.28) є необхідними умовами оптимальності першого порядку, а за допомогою задач (8.3.20), (8.3.21) та (8.3.27), (8.3.28) сформульований принцип максимуму задачі (P₃).

Структура оптимального керування для задачі (P₃)

Дослідимо зараз структуру оптимального керування u^* .

- Якщо $\mu_2 < \mu_1$, тоді $\mu_2 p_2(T) - \mu_1 p_1(T) = \mu_2 - \mu_1 < 0$, тоді можна вибрати інтервал максимальної довжини $(T - \eta, T]$ ($\eta > 0$), у якому $\mu_2 p_2(t) - \mu_1 p_1(t) < 0$. Із (8.3.28) випливає, що $u^*(t) = 0$ на $(T - \eta, T]$, а отже,

$$p_1'(t) = -r_1 p_1(t), \quad p_2'(t) = r_2 p_2(t), \quad \text{для майже всіх } t \in (T - \eta, T).$$

Отож, із останніх рівнянь отримуємо

$$p_1(t) = \exp(-r_1(t - T)), \quad p_2(t) = \exp(r_2(t - T)), \quad t \in [T - \eta, T].$$

Функція $t \rightarrow \mu_2 \exp(r_2(t - T)) - \mu_1 \exp(-r_1(t - T))$ є зростаючою на $[T - \eta, T]$, звідки випливає, що $T - \eta = 0$ і

$$\mu_2 p_2(t) - \mu_1 p_1(t) < 0, \quad t \in (0, T),$$

тому $u^*(t) = 0$ майже всюди на $(0, T)$.

- Якщо $\mu_2 = \mu_1$, тоді (p_1, p_2) є розв'язком задачі

$$\begin{cases} p_1' = -r_1 p_1 - \mu_1 u^* y^*(p_2 - p_1), & t \in (0, T), \\ p_2' = r_2 p_2 - \mu_1 u^* x^*(p_2 - p_1), & t \in (0, T), \\ p_1(T) = p_2(T) = 1. \end{cases}$$

Підсумовуючи

$$p_2(t) - p_1(t) = - \int_0^T [r_2 p_2(s) + r_1 p_1(s)] \exp \left\{ \mu_1 \int_s^t u^*(\tau) [y^*(\tau) - x^*(\tau)] d\tau \right\} ds,$$

$t \in [0, T]$. Отже, $p_2(t) - p_1(t) < 0$ на інтервалі максимальної довжини $(T - \eta, T]$ ($\eta > 0$), аналогічно як у попередньому випадку, звідси випливає, що $u^*(t) = 0$ майже всюди на $(0, T)$.

- Якщо $\mu_2 > \mu_1$, тоді існує максимальний інтервал $(T - \eta, T)$ ($\eta > 0$) такий, що

$$\mu_2 p_2(t) - \mu_1 p_1(t) > 0, \quad t \in (T - \eta, T].$$

Із (8.3.28) матимемо, що $u^*(t) = 1$ на $(T - \eta, T]$. Доведемо, що $T - \eta$ є точкою перемикання для оптимального керування u^* . Справді, з (8.3.27) отримаємо

$$\begin{aligned} \mu_2 p_2(t) - \mu_1 p_1(t) = & \\ & - \int_t^{T-\eta} [r_2 \mu_2 p_2(s) + r_1 \mu_1 p_1(s)] \times \\ & \exp \left\{ \int_s^t u^*(\tau) [\mu_2 x^*(\tau) - \mu_1 y^*(\tau)] d\tau \right\} ds, \end{aligned} \quad (8.3.29)$$

$t \in [0, T - \eta]$. З іншого боку, (p_1, p_2) – розв’язок початкової задачі

$$\begin{cases} p_1' = -p_1(r_1 - \mu_1 y^*) - \mu_2 y^* p_2, & t \in (T - \eta, T), \\ p_2' = -p_2(\mu_2 x^* - r_2) + \mu_1 x^* p_1, & t \in (T - \eta, T), \\ p_1(T) = p_2(T) = 1. \end{cases} \quad (8.3.30)$$

Позаяк $\mu_2 p_2(t) - \mu_1 p_1(t) > 0$ для кожного $t \in (T - \eta, T]$, то отримаємо, що

$$p_1(t) \geq \exp\{r_1(T - t)\} \geq 1, \quad t \in [T - \eta, T].$$

Використовуючи те, що $\mu_2 p_2(T - \eta) - \mu_1 p_1(T - \eta) = 0$ та (8.3.29), отримаємо, що $p_2(T - \eta) > 0$ і тому $\mu_2 p_2(t) - \mu_1 p_1(t) < 0$ на максимальному інтервалі $(T - \eta - \varepsilon, T - \eta]$ ($\varepsilon > 0$). Звідки випливає, що $u^*(t) = 0$ на $(T - \eta - \varepsilon, T - \eta]$. Але на цьому інтервалі справджуються рівності

$$\begin{aligned} p_1(t) &= p_1(T - \eta) \exp\{r_1(T - \eta - t)\}, \\ p_2(t) &= p_2(T - \eta) \exp\{r_2(t - T + \eta)\}, \end{aligned}$$

у підсумку $\mu_2 p_2 - \mu_1 p_1$ – зростаюча функція на $(T - \eta - \varepsilon, T - \eta)$. Оскільки

$$\mu_2 p_2(t) - \mu_1 p_1(t) < 0, \quad t \in (T - \eta - \varepsilon, T - \eta),$$

то звідки $T - \eta - \varepsilon = 0$. Накінець матимемо

$$u^*(t) = \begin{cases} 0, & t \in [0, T - \eta], \\ 1, & t \in (T - \eta, T], \end{cases} \quad (8.3.31)$$

майже всюди на $(0, T)$.

Отже, ми отримали двопозиційне оптимальне керування з щонайменше однією точкою перемикання. Можемо визначити *точку перемикання* $T - \eta$ або опираючись на (8.3.30) і на рівність $\mu_2 p_2(T - \eta) - \mu_1 p_1(T - \eta) = 0$, або обчисливши таку точку $T - \eta \in [0, T]$, яка максимізує $\Phi(u^*)$, де u^* має вигляд (8.3.31).

Апроксимація оптимального керування задачі (P₃)

Для того, щоб апроксимувати оптимальне керування u^* спочатку необхідно обчислити η у формулі (8.3.31). З цією метою спробуємо підібрати τ (яке означає $T - \eta$ із (8.3.31)) як точку перемикання керування

серед точок сітки вузлів, визначеної на $[0, L]$ (L прийmemo, що дорівнює T), і отримаємо її як точку, що максимізує функцію $\Phi(u)$. Тобто, прийmemo

$$u(t) = \begin{cases} 0, & t \in [0, \tau], \\ 1, & t \in (\tau, L]. \end{cases}$$

Алгоритм розв'язування такий.

Алгоритм 2.1

Будуємо сітку вузлів.

```
tspan=0:h1:L;
```

Проводимо обчислення у вузлах сітки.

```
m=length(tspan);
for i=1 to m
    tau=tspan(i);
```

Крок 1. Будуємо відповідне керування u_τ

$$u_\tau(t) = \begin{cases} 0, & t \in [0, \tau], \\ 1 & t \in (\tau, L]. \end{cases}$$

Крок 2. Обчислюємо стан $[x, y]$ як розв'язок системи (8.3.20), що відповідає керуванню $u := u_\tau$, з початковими умовами

$$x(0) = x_0, \quad y(0) = y_0.$$

Крок 3. Обчислюємо значення цільової функції Φ у вузлах побудованої сітки.

```
fiu(i)=x(L)+y(L);
end-for
```

Крок 4. Визначення максимального значення вектора fiu .

Сценарій `ppp1.m` обчислення наближеного розв'язку задачі (P_3) такий:

```
% file ppp1.m
% predator-prey model with bang-bang optimal control
clear
global r1 r2 mu1 mu2
global tsw
disp('get model parameters');
r1=input('r1 : ');
mu1=input('mu1 : ');
```

```

r2=input('r2 : ');
mu2=input('mu2 : ');
disp('get data');
L=input('final time : ');
h=input('grid step : ');
h1=input('switch step : ');
x0=input('x(0) : ');
y0=input('y(0) : ');
lw=input('LineWidth : '); % for graphs ( plot )
tt=0:h:L; % ODE integration grid
n=length(tt);
tspan=0:h1:L; % switching points grid
m=length(tspan);
for i=1:m
    i
    tsw=tspan(i); % tsw stands for switching point
    [t q]=ode45('bp2',tt,[x0; y0]);
    k=length(t);
    fiu(i)=q(k,1) + q(k,2); % store cost functional value
    clear t q % clear memory to avoid garbage for the next
iteration
end
w=fiu';
save cont.txt w -ascii
disp('FILE MADE');
[vmax,j]=max(fiu); % maximal value and corresponding index
j
a1=['max=', num2str(vmax)];
disp(a1);
a2=['switch=', num2str(tspan(j))];
disp(a2);
figure(1); plot(tspan,fiu,'LineWidth',lw); grid
xlabel('\bf{u switch}','FontSize',16)
ylabel('\bf{\Phi(u \tau)}','FontSize',16)
figure(2)
bar(fiu)
title('\bf{\Phi(u \tau)}','FontSize',16)

```

У процедурі `ode45` використано вектор `tt`, вектор `tspan` сітки вузлів для обчислення точки перемикання, щоб прискорити роботу програми. У програмі вище використовується функція `bp2.m`, яка визначена так:

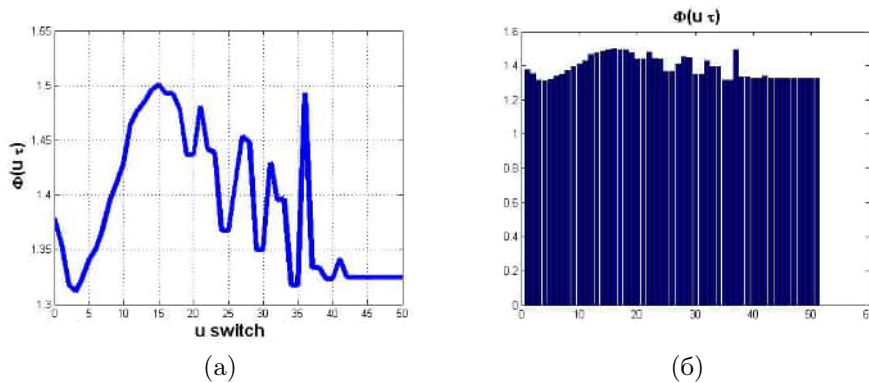


Рис. 8.18. Залежність функції цілі від точок перемикання для значення параметрів $r_1 = 0.07$, $\mu_1 = 1$, $r_2 = 0.6$, $\mu_2 = 2$, $L = 50$, $h = 0.1$, $h_1 = 1$, $x(0) = 0.04$, $y(0) = 0.02$, $lw = 5$: *a* – поведінка цільової функції в залежності від керування; *б* – інша залежність поведінки цільової функції в залежності від точки перемикання

```
function out1=bp2(t,q)
global r1 r2 mu1 mu2
global tsw
if t > tsw
u=1;
else
u=0;
end
out1 =[r1*q(1)-mu1*u*q(1)*q(2);mu2*u*q(1)*q(2)-r2*q(2)];
end
```

Для проведення числових експериментів візьмемо такі значення вхідних параметрів: $r_1 = 0.07$, $\mu_1 = 1$, $r_2 = 0.6$, $\mu_2 = 2$, $L = 50$, $h = 0.1$, $h_1 = 1$, $x(0) = 0.04$, $y(0) = 0.02$, $lw = 5$. Графік функції $\tau \rightarrow \Phi(u - \tau)$, де τ є точкою перемикання керування u_τ , зображено на рис. 8.18а та рис. 8.18б.

Отримано глобальний максимум на $[0, L]$ при $\tau^* = 15$ і максимальне значення цільової функції, яке становить 1.5006. У програмі `ppr2.m` використовується точка перемикання оптимального керування при побудові графіків для відповідного стану компонент.

```

% file ppp2.m
% predator-prey model with bang-bang optimal control
% makes graphs by using the switching point obtained by
ppp1.m
clear all
global r1 r2 mu1 mu2 tsw
%...read parameters and data as in ppp1.m (except h) ...
tspan=0:h1:L;
% graph of the control
m=length(tspan);
for i=1:m
    if tspan(i)>tsw
        z(i)=1;
    else
        z(i)=0;
    end
end
figure(1); plot(tspan,z,'rs'); grid
axis([0 L -0.2 1.2])
xlabel('\bf {t}', 'FontSize', 16)
ylabel('\bf {u(t)}', 'FontSize', 16)
[t q]=ode45('bp2',[0 L],[x0; y0]);
% predator-prey populations graph
figure(2)
plot(t,q(:,1),'*',t,q(:,2),'ro'); grid
xlabel('\bf {t}', 'FontSize', 16)
legend('prey','predator',0)
% x0y graph
figure(3)
plot(q(:,1),q(:,2),'LineWidth',lw); grid
xlabel('\bf {x}', 'FontSize', 16)
ylabel('\bf {y}', 'FontSize', 16)

```

Результати обчислення за допомогою програми ppp2.m зображено на рис. 8.19а та рис. 8.19б.

8.3.4. Модель інсулінової терапії

Розглянемо модель інсулінової терапії пацієнта хворого на діабет. Важливою проблемою для таких пацієнтів є підтримання глюкози у крові на належному рівні та запобігати різкому відхиленню від нього. На прак-

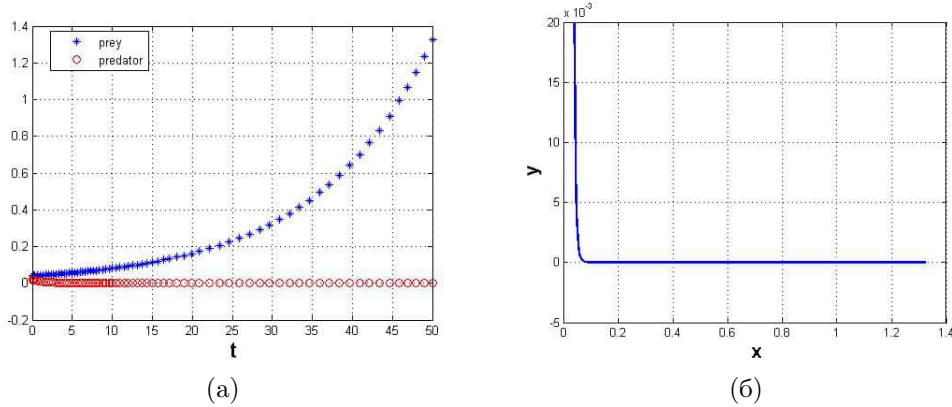


Рис. 8.19. Результати обчислення за допомогою програми `ppr2.m` для тих самих значень параметрів моделі як у випадку `ppr1.m`

тиці хворі отримують інсулін у вигляді ін'єкцій. Досліджується задача оптимального керування з імпульсним контролем підтримання стійкого рівня глюкози у крові. Ця задача не є задачею (P_1) , розглянутою у підрозділі 8.3.1., бо керування у досліджуваній задачі імпульсного типу. Однак поставимо собі за завдання отримати необхідні умови оптимальності першого порядку і використати їх під час написання програми.

У випадку діабету підшлункової залози (бета клітини) неможливо забезпечити належного рівня засвоєння інсуліну. Концентрація глюкози у крові збільшується, коли вміст глюкози керований у тварин, тоді як збільшення інсуліну приводить до виведення глюкози з плазми. Внаслідок чого вміст цукру у крові спадає до нормального рівня 0.8-1.2g/l. Позначимо через $I(t)$ концентрацію інсуліну і через $G(t)$ – глюкози у момент часу $t \in [0, L]$ ($L > 0$).

Розглянемо пацієнта, хворого на діабет, організм якого не в стані продукувати достатньо інсуліну. Інсулін постачається за допомогою ін'єкцій. Концентрація глюкози визначається (вимірюється) легко. Відповідна спрощена *модель динаміки інсулін-глюкозової системи* така [31, Розділ 3.]:

$$\begin{cases} I'(t) = dI(t), & t \in (0, L), \\ G'(t) = bI(t) + aG(t), & t \in (0, L), \\ I(0) = I_0, \quad G(0) = G_0, \end{cases} \quad (8.3.32)$$

де $d < 0$ ($|d|$ – швидкість зменшення інсуліну); a – швидкість зростання глюкози ($a \neq d$); b – від'ємна константа, яку можна поміряти; I_0 – початкова концентрація інсуліну (ін'єкція); G_0 – початкова концентрація

глюкози. Результати чисельних обчислень свідчать про те, що модель добре працює у випадку, коли $I(t)$ та $G(t)$ перебувають у властивих межах. У випадку, коли I_0 та G_0 розташовані поза допустимими межами, то модель повертає від'ємні значення $I(t)$ і $G(t)$, а це означає, що модель неправильна. Точніша модель розглядається у кінці підрозділу. Співвідношення між $I(t)$ та $G(t)$ у (8.3.32) локально лінеаризовані, загальнішу модель розглянемо пізніше.

Приводимо нижче програму `dbt1` побудови графіків концентрації інсуліну та глюкози.

```
% file dbt1.m
% blood insulin-glucose system
% y(1)=insulin concentration
% y(2)=glucose concentration
clear
global a b d
L=input('final time : ');
h=input('h : ');
I0=input('I(0) : ');
G0=input('G(0) : ');
a=0.0343;
b=-0.05;
d=-0.5;
tspan=0:h:L;
[t y]=ode45('hum1',tspan,[I0; G0]);
figure(1);plot(t,y(:,1),'*'); grid
xlabel('\bf {t}', 'FontSize',16)
ylabel('\bf {I(t)}', 'FontSize',16)
figure(2)
plot(t,y(:,2),'r*'); grid
xlabel('\bf {t}', 'FontSize',16)
ylabel('\bf {G(t)}', 'FontSize',16)
```

Функція `hum1.m`, визначена у середовищі MatLab[®], яка використовується у програмі вище й описує праву частину системи (8.3.32).

```
function out1=hum1(t,y)
global a b d
out1=[d*y(1);b*y(1)+a*y(2)];
end
```

Чисельне тестування моделі проведемо за таких значень параметрів

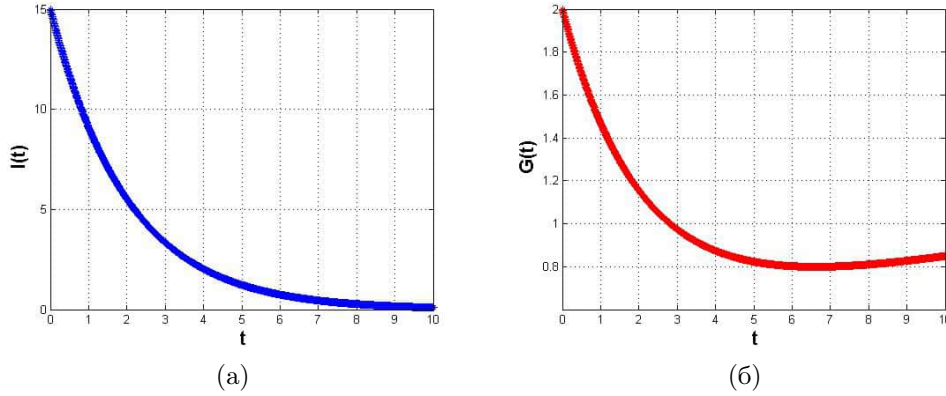


Рис. 8.20. Інсулін-глюкозна залежність: *a* – концентрація інсуліну; *b* – концентрація глюкози

моделі: $L = 10$, $h = 0.01$, $I_0 = 15$, $G_0 = 2$. Отримані графічні результати обчислення динаміки вмісту інсуліну та глюкози зображені на рис. 8.20. Зауважимо, що концентрація інсуліну (рис. 8.20а) спадає до нуля (ефект швидкості зменшення інсуліну), тоді як концентрація глюкози (рис. 8.20б) досягає властивого рівня. Інсулін викликає позитивні наслідки, позаяк рівень глюкози на початку становив $G_0 = 2$ і досягнув, наближено, значення 0.8 в момент часу $t = 6$. Після моменту $t = 7$ дія інсуліну спадає і вміст глюкози повільно росте.

Систему (8.3.32) можна проінтегрувати в явному вигляді. З цією метою розглянемо спочатку задачу динаміки вмісту інсуліну

$$\begin{cases} I'(t) = dI(t), & t \in (0, L), \\ I(0) = I_0, \end{cases} \quad (8.3.33)$$

розв'язок якої запишемо у явному вигляді

$$I(t) = I_0 e^{dt}, \quad t \in [0, L]. \quad (8.3.34)$$

Підставимо (8.3.34) у задачу (8.3.32), отримаємо задачу для динаміки вмісту глюкози

$$\begin{cases} G'(t) = bI_0 e^{dt} + aG(t), & t \in (0, L), \\ G(0) = G_0, \end{cases}$$

розв'язок якої має вигляд

$$G(t) = G_0 e^{at} + \frac{bI_0}{d-a} (e^{dt} - e^{at}), \quad t \in [0, L]. \quad (8.3.35)$$

Отже, розв'язок задачі (8.3.32) визначається формулами (8.3.34), (8.3.35). Програма обчислення наближеного розв'язку описаної вище задачі записується так (m-файл dbt2.m):

```
% file dbt2.m
% blood insulin-glucose system
% y1(t)=insulin concentration
% y2(t)=glucose concentration
% mathematical integration
clear
L=input('final time : ');
h=input('h : ');
I0=input('insulin(0) : ');
G0=input('glucose(0) : ');
a=0.0343;
b=-0.05;
d=-0.5;
temp=b*I0/(d-a);
t=0:h:L;
v=exp(d*t);
w=exp(a*t);
y1=I0*v;
y2=G0*w+temp*(v-w);
figure(1)
plot(t,y1,'*'); grid
xlabel('\bf {t}', 'FontSize',16)
ylabel('\bf {I(t)}', 'FontSize',16)
figure(2)
plot(t,y2,'r*'); grid
xlabel('\bf {t}', 'FontSize',16)
ylabel('\bf {G(t)}', 'FontSize',16)
```

Обчислені графіки аналогічні до побудованих раніше.

Розглянемо задачу оптимального керування з імпульсним керуванням, щоб отримати алгоритм інсулінової терапії для забезпечення належного контролю за вмістом глюкози у крові на заданому відрізку часу. Позначимо через A бажаний рівень глюкози у крові. Припустимо, що пацієнт отримує m ін'єкцій у моменти часу

$$0 = t_1 < t_2 < \dots < t_m = L,$$

відповідними дозами $c_j(t)$, $j = 1, 2, \dots, m$ і початкова концентрація інсуліну $I_0 = 0$. Зазвичай ін'єкції інсуліну робимо через фіксовані про-

міжки часу, тобто $t_{j+1} - t_j = h$ ($j = \overline{1, m-1}$). Тоді динаміка інсулін-глюкозної залежності описується такою початковою задачею:

$$\begin{cases} I'(t) = dI(t) + \sum_{j=1}^m c_j \delta_{t_j}, \\ G'(t) = bI(t) + aG(t), \\ I(0) = 0, \quad G(0) = G_0, \end{cases} \quad (8.3.36)$$

де δ_{t_j} – функція Дірас'а у момент часу t_j . Задача (8.3.36) еквівалентна такій:

$$\begin{cases} I'(t) = di(t), & t \in (t_j, t_{j+1}), \quad j \in \{1, 2, \dots, m-1\}, \\ I(0) = 0, \\ I(t_j+) = I(t_j-) + c_j, & j \in \{1, 2, \dots, m-1\}, \\ G'(t) = bI(t) + aG(t), & t \in (0, L), \\ G(0) = G_0. \end{cases} \quad (8.3.37)$$

Розв'язок задачі (8.3.36) у сенсі теорії розподілів (узагальнений розв'язок, який є також розв'язком задачі (8.3.37)) має вигляд

$$\begin{cases} I(t) = \sum_{j=1}^m c_j H(t - t_j) e^{d(t-t_j)}, \\ G(t) = G_0 e^{at} + \frac{b}{d-a} S(t), \quad t \in [0, L], \end{cases} \quad (8.3.38)$$

де

$$S(t) = \sum_{j=1}^m c_j H(t - t_j) [e^{d(t-t_j)} - e^{a(t-t_j)}], \quad (8.3.39)$$

а H – функція Heaviside'а (функція стрибка)

$$H(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

Крім того, функція $t \mapsto H(t - t_j)$ у формулах (8.3.38) і (8.3.39), визначена для $t \in [0, L]$ та набуває значення

$$H(t - t_j) = \begin{cases} 1, & t \in [t_j, L], \\ 0, & t \in [0, t_j). \end{cases}$$

Із явного подання для розв'язку G випливає, що дія інсулінової ін'єкції, отриманої у момент часу $t = t_j$, триває тільки після t_j , тобто для $t > t_j$. Вплив ін'єкції затухає протягом деякого часового проміжку за експоненціальним законом із від'ємним показником.

Отож, задача оптимального керування (інсулінової терапії), що залежить від (8.3.36), має вигляд

$$\Psi(c) = \frac{1}{2} \int_0^L [G(t) - A]^2 dt \rightarrow \min, \quad (\text{I})$$

стосовно $c = (c_1, c_2, \dots, c_m) \in \mathbb{R}^m$, де (I, G) – розв'язок задачі (8.3.36). Тут вектор c є керуванням (тобто імпульсним керування – керуванням, яке діє у визначені моменти часу).

Функціонал Ψ є квадратичним функціоналом стосовно c , а це означає, що існує мінімум щодо оптимального керування $c = (c_1, c_2, \dots, c_m) \in \mathbb{R}^m$. Оптимальне керування задовольняє лінійну систему алгебричних рівнянь із невідомими c_j ($j = \overline{1, m}$) вигляду

$$\frac{\partial \Psi}{\partial c_j}(c) = 0, \quad j = \overline{1, m}. \quad (8.3.40)$$

Обчислимо часткові похідні, використавши формулу (8.3.40), отримаємо таку алгебричну систему:

$$\sum_{i=1}^m q_{ij} c_i = B_j, \quad j \in \{1, 2, \dots, m\},$$

де

$$\begin{aligned} q_{ij} &= \alpha \int_0^L H(t - t_i) H(t - t_j) e_i(t) e_j(t) dt, \\ B_j &= \int_0^L H(t - t_j) e_j(t) (A - G_0 e^{at}) dt = \int_{t_j}^L e_j(t) (A - G_0 e^{at}) dt, \end{aligned} \quad (8.3.41)$$

$i, j \in \{1, 2, \dots, m\}$. Вище позначено

$$\alpha = \frac{b}{d - a},$$

$$e_j(t) = e^{d(t-t_j) - e^a(t-t_j)}, \quad t \in [0, L], \quad j \in \{1, 2, \dots, m\}.$$

Якщо $i > j$, тоді $t_i > t_j$ і формула (8.3.41) набуде вигляду

$$q_{ij} = \alpha \int_{t_i}^L e_i(t)e_j(t)dt.$$

Наша мета – знайти розв’язок системи (8.3.40). Якщо якась компонента c є від’ємною, то з погляду медицини така ситуація не має сенсу. Якщо ввести обмеження $c_j \geq 0$, $j \in \{1, 2, \dots, m\}$, то отримаємо складну задачу лінійного програмування. Інша можливість полягає у тому, що обмеження беремо у вигляді $0 \leq c_j \leq \bar{c}$, $j \in \{1, 2, \dots, m\}$, та використати метод градієнтів знаходження розв’язку задачі. Але ця задача також є складною. Для визначення лікарняної тактики приймемо $c_j = 0$, якщо $c_j < 0$. Тоді одержуватимемо додаткову глюкозу зазвичай під час споживання їжі, або замінимо від’ємні дози ін’єкцій $c_j = 0$, і звідси отримаємо квазіоптимальне керування. У чисельних експериментах моделі, проведених при апріорних значеннях $G(0)$, розв’язок додатний.

Повернемося до лінійної системи. Алгоритм обчислення транспонованої матриці Q такої, що $Q^T = \{q_{ij}\}$, такий:

```
for j=1 to m
  for i=1 to j
    compute  $q_{ij} = \alpha \int_{t_j}^L e_i(t)e_j(t)dt$ 
  end-for
  for i=j+1 to m
    compute  $q_{ij} = \alpha \int_{t_i}^L e_i(t)e_j(t)dt$ 
  end-for
end-for
```

Транспонуючи матрицю, елементи якої обчислюються за вище зазначеним алгоритмом, отримуємо матрицю Q .

Значення параметрів системи приймемо такими: $a = 0.1$, $b = -0.05$, $d = -0.5$. Нижче наведена програма `dbt3.m` обчислення рівня глюкози залежно від ін’єкцій інсуліну:

```
% file dbt3.m
% blood insulin-glucose system
% Insuline Shots L=48; L=60
% mathematical integration
clear
```

```

global a d b a1 ti tj G0
L=input('final time: ');
h=input('h: ');
I0=input('insulin(0): ');
G0=input('glucose(0): ');
a1=input('Desired Level Of Glucose A: ');
m=L/h+1;
a=0.1;
b=-0.05;
d=-0.5;
t=0:h:L;
alf=b/(d-a);
Q=zeros(m-1);
for j=1:m-1
    tj=t(j);
    for i=1:j
        ti=t(i);
        Q(i,j)=alf*quadl('fi1',tj,L);
    end
    for i=j+1:m-1
        ti=t(i);
        Q(i,j)=alf*quadl('fi2',ti,L);
    end
end
Q=Q';
for j=1:m-1
    tj=t(j);
    B(j)=quadl('psiIG',tj,L);
end
B=B';
% solve system Qc=B
c=Q\B;
ttl=[' \bf {Insulin Shots: A=}',num2str(a1),'; G(0)=',...
      num2str(G0),'; I(0)=',num2str(I0),'; L=',num2str(L),...
      '; h=',num2str(h),'; m=',num2str(m),'.'];
figure(1)
bar(c); grid
xlabel(' \bf {t}', 'FontSize',10)
title( ttl, 'FontSize',10)

```

У цьому випадку використовуються функції `fi1.m` та `fi2.m`, за допомогою

яких обчислюємо, відповідно, компоненти q_{ij} матриці Q для $i \leq j$ та $i > j$:

```
function y=fi1(t)
global ti tj
global a d
y=0;
if t>=tj
    temp1=exp(d*(t-tj))-exp(a*(t-tj));
    temp2=exp(d*(t-ti))-exp(a*(t-ti));
y=temp1.*temp2;
end
```

```
function y=fi2(t)
global ti tj
global a d
y=0;
if t>=ti
    temp1=exp(d*(t-tj))-exp(a*(t-tj));
    temp2=exp(d*(t-ti))-exp(a*(t-ti));
y=temp1.*temp2;
end
```

Функція *psiIG.m* обчислює компоненти B_j правої частини системи.

```
function y=psiIG(t)
global tj
global a d
global a1 G0
y=0;
if t >= tj
    temp1=exp(d*(t-tj))-exp(a*(t-tj));
    temp2=a1-G0*exp(a*t);
y=temp1.*temp2;
end
```

Перейдемо до чисельних експериментів.

* **Приклад 8.3.1.** Прийmemo $L = 48$ (годин), $G_0 = 2$, $A = 1$, $m = 9$ (кількість ін'єкцій). Звідси випливає, що інтервал між введенням двох послідовних ін'єкцій становить $h = 6$ (годин). На рис. 8.21а зображено ін'єкції інсуліну на часовому інтервалі 48 годин.

* **Приклад 8.3.2.** Другий чисельний експеримент проведемо для таких значень параметрів системи: $L = 60$, $G_0 = 2$, $A = 0.8$, $m = 11$

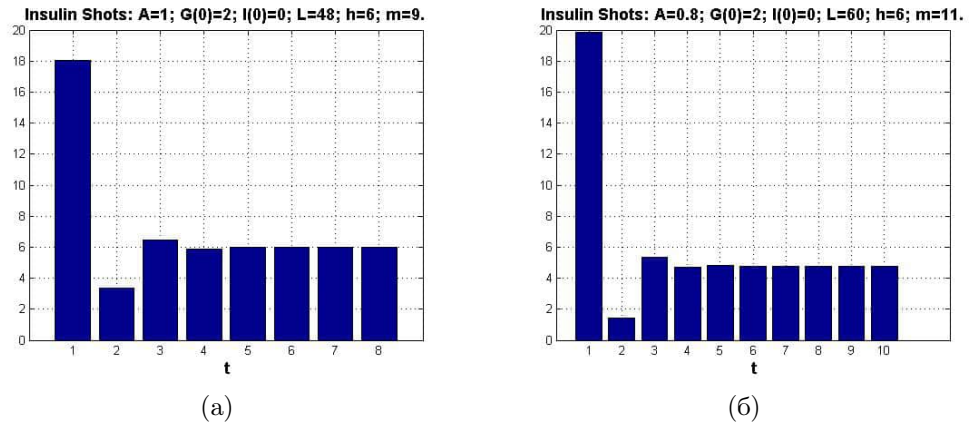


Рис. 8.21. Залежність вмісту глюкози від ін'єкцій інсуліну

($h = 6$). Результат обчислення зображений на Рис. 8.21б.

На завершення наших досліджень обчислимо рівень глюкози за формулами (8.3.38), (8.3.39). Графік концентрації глюкози у крові у другому числовому експерименті зображений на рис. 8.22.

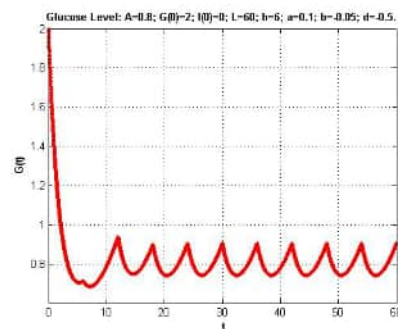


Рис. 8.22. Концентрація глюкози у випадку періодичних ін'єкцій

Зауважимо, що рівень глюкози спадає з початкового $G_0 = 2$ до бажаного рівня $A = 0.8$ та залишається на цьому рівні. Для першого числового експерименту поведінка $G(t)$ аналогічна. Програма обчислення рівня глюкози у циклічному введенні ін'єкцій така:

```
% file dbt4.m
% blood insulin-glucose system
% y2(t)=glucose concentration
% mathematical integration
```

```
clear
global a d b a1 ti tj G0 cj
L=input('final time : ');
h=input('h : ');
I0=input('insulin(0) : ');
G0=input('glucose(0) : ');
m=L/h+1;
a1=.8;
a=0.1;
b=-0.05;
d=-0.5;
alf=b/(d-a);
t=0:h:L;
Q=zeros(m-1);
for j=1:m-1
    tj=t(j);
    for i=1:j
        ti=t(i);
        Q(i,j)=alf*quadl('fi1',tj,L);
    end
    for i=j+1:m-1
        ti=t(i);
        Q(i,j)=alf*quadl('fi2',ti,L);
    end
end
Q=Q';
for j=1:m-1
    tj=t(j);
    B(j)=quadl('psiIG',tj,L);
end
B=B';
% solve system Qc=B
c=Q\B;
s=0:h1:L;
slngth=length(s);
v=G0*exp(a*s);
temp1=zeros(1,slngth);
for j=1:m-1
    tj=t(j);
    cj=c(j);
    temp2=zeros(1,slngth);
    for i=1:slngth
```

```

        temp2(i)=funcS(s(i));
    end
    temp1=temp1+temp2;
end
temp=alf*temp1;
y2=v+temp;
figure(1)
plot(s,y2,'r. '); grid
ttl=[[' \bf {Glucose Level: A=}',num2str(a1),'; G(0)=',...
      num2str(G0),'; I(0)=',num2str(I0),'; L=',...
      num2str(L),'; h=',num2str(h),'; a=',...
      num2str(a),'; b=',num2str(b),'; d=',...
      num2str(d),'. ']];
title(ttl,'FontSize',10)
xlabel(' \bf {t}', 'FontSize',10)
ylabel(' \bf {G(t)}', 'FontSize',10)

```

У програмі `dbt4.m` використовується процедура `funcS`, яка слугує для обчислення функції (8.3.39) і у машинному коді записується:

```

function y=funcS(t)
global cj a d tj
y=0;
if t>=tj
    temp1=exp(d*(t-tj))-exp(a*(t-tj));
    y=cj*temp1;
end
end

```

♣ **Зауваження 8.3.3.** Система (8.3.40) описує умови оптимальності першого порядку для задачі (I).

8.3.5. Приклади

★ **Приклад 8.3.3. (ВІЛ терапія)** Розглянемо математичну модель, яка описує взаємодію імунної системи з вірусом ВІЛ (HIV – human immunodeficiency virus), запропоновану у [50]. Зокрема, дослідимо дві задачі оптимального керування, які базуються на хіміотерапії з ефектами вірусної інфекції або вірусної продуктивності.

Імунна система описується у термінах поширення популяції $CD4^+$ T клітин (див. [60], [44], [62]). Нехай:

$T(t)$ означає концентрацію $CD4^+$ T клітин;

$T_i(t)$ означає концентрацію інфікованих $CD4^+$ T клітин;
 $V(t)$ означає концентрацію вільних від вірусної інфекції частин у момент часу t .

Динаміка системи моделюється такою початковою задачею:

$$\begin{cases} T'(t) = \frac{s}{1+V(t)} - \mu_1 T(t) + rT(t) \left(1 - \frac{T(t) + T_i(t)}{T_{\max}}\right) - \\ \qquad \qquad \qquad k_1 V(t) T(t), \\ T_i'(t) = k_1 V(t) T(t) - \mu_2 T_i(t), \\ V'(t) = -k_1 V(t) T(t) - \mu_3 V(t) + N\mu_2 T_i(t), \\ T(0) = T_0, \quad T_i(0) = T_{i0}, \quad V(0) = V_0, \end{cases} \quad (8.3.42)$$

$t \in (0, L)$ ($L > 0$), де $s, k_1, r, N, \mu_1, \mu_2, \mu_3, T_{\max}$ – додатні сталі, $T_0, T_{i0}, V_0 \geq 0$ – початкові концентрації $CD4^+$ T клітин, інфікованих $CD4^+$ T клітин і вільних від вірусних інфекцій частин, відповідно.

Доданок $s/(1+V(t))$ – основний член; залежність від вірусної концентрації V моделює той факт, що може відбутися попереднє зараження T клітин, а отже, знижує продукування неінфікованих T клітин.

Доданок $-k_1 VT$ у першому рівнянні (8.3.42) разом із доданком $+k_1 VT$ другого рівняння (8.3.42) описує як інфікування T клітин обумовлене концентрацією вірусу V ; доданок $-k_1 VT$ у третьому рівнянні задачі (8.3.42) описує опір вірусів у неінфікованих T клітинах, отже, є головним.

μ_1, μ_2, μ_3 означають природню швидкість згасання.

Доданок $N\mu_2 T_i$ у третьому рівнянні (8.3.42) описує продукування вірусів під час зменшення інфікованих T клітин.

Доданок

$$r \left(1 - \frac{T(t) + T_i(t)}{T_{\max}}\right)$$

описує швидкість продукування T клітин.

Хіміотерапія лікарськими препаратами може бути такою.

- Вплив вірусної інфективності, тоді друге рівняння (8.3.42) запишеться у вигляді (див. [25])

$$T_i'(t) = u(t)k_1 V(t)T(t) - \mu_2 T_i(t), \quad t \in (0, L),$$

де $u(t)$ – змінна керування, що означає ефективність хіміотерапії. Перше і третє рівняння можуть бути, відповідно, модифіковані.

- Або зменшення продукування вірусу найчастіше застосовують у терапії лікарськими препаратами такими як інгібітор протеаза (див. [50]). Тоді третє рівняння (8.3.42) змодифікується до вигляду

$$V'(t) = -k_1 V(t)T(t) - \mu_3 V(t) + u(t)N\mu_2 T_i(t), \quad t \in (0, L).$$

Друге рівняння може бути відповідно змодифіковане.

У кожному з випадків максимізується функціонал корисності

$$\int_0^L \left[aT(t) - \frac{(1-u(t))^2}{2} \right] dt \rightarrow \max$$

($a > 0$) стосовно $u \in L^2(0, L)$, $0 \leq u(t) \leq 1$ для майже кожного $t \in (0, L)$, що означає максимізацію кількості неінфікованих T клітин, водночас мінімізують затрати хіміотерапії.

Більша чи менша величина a відповідає більшому чи меншому впливу мінімізації ціни хіміотерапії.

Пропонується самостійно отримати необхідні умови оптимальності першого порядку задачі оптимального керування.

✓ *Вказівка.* Перша задача оптимального керування, запропонована вище, є частинним випадком задачі (P_1) (пункт 8.3.1.), в якій

$$G(t, u, T, T_i, V) = aT - \frac{(1-u)^2}{2}, \quad \varphi(T, T_i, V) = 0,$$

$$f(t, u, T, T_i, V) = \begin{pmatrix} \frac{s}{1+V} - \mu_1 T + rT \left(1 - \frac{T+T_i}{T_{\max}} \right) - k_1 uVT \\ k_1 uVT - \mu_2 T_i \\ -k_1 uVT - \mu_3 V + N\mu_2 T_i \end{pmatrix},$$

і

$$K = \{w \in L^2(0, L) : 0 \leq w(t) \leq 1 \text{ для майже всіх } t \in (0, L)\}.$$

Друга задача оптимального керування також є частинним випадком задачі (P_1) для таких самих G , φ , K (як у попередній задачі), а f має вигляд

$$f(t, u, T, T_i, V) = \begin{pmatrix} \frac{s}{1+V} - \mu_1 T + rT \left(1 - \frac{T+T_i}{T_{\max}} \right) - k_1 VT \\ k_1 VT - \mu_2 uT_i \\ -k_1 VT - \mu_3 V + uN\mu_2 T_i \end{pmatrix}.$$

★ **Приклад 8.3.4. (Керування у SIR моделі)** У цьому підрозділі опишемо динаміку поширення захворювання (яка поширюється тільки через контакт між інфікованими та схильними до захворювання особами) у біологічній популяції, опираючись на таку стандартну SIR модель динаміки демографії (див. [28])

$$\begin{cases} S'(t) = mN - mS(t) - cS(t)I(t) - u(t)S(t), \\ I'(t) = -mI(t) - cS(t)I(t) - dI(t), \\ R'(t) = mR(t) + u(t)S(t) + dI(t), \end{cases} \quad (8.3.43)$$

для $t \in (0, L)$ ($L > 0$), з такими початковими умовами

$$S(0) = S_0 > 0, \quad I(0) = I_0 > 0, \quad R(0) = R_0 \geq 0. \quad (8.3.44)$$

Вище введені позначення, які означають:

- $S(t)$ – густину схильних до захворювання осіб,
- $I(t)$ – густину інфікованих осіб,
- $R(t)$ – густину вилікуваних (або імуннозахищених) осіб

у момент часу t . $N = S(t) + I(t) + R(t) = S_0 + I_0 + R_0 > 0$ є сталою величиною і означає густину загальної популяції, кількість якої припускаємо незмінною. Тут m , c , d деякі додатні сталі. Відсоток хворих описується доданком $cS(t)I(t)$. Стала d означає швидкість одужання інфікованих осіб. Керування u означає частину осіб популяції, схильних до захворювання, які піддаються вакцинації. Припускається, що вакциновані особи виліковуються.

Пропонується дослідити таку задачу оптимального керування: мінімізувати функціонал

$$\int_0^L [I(t) + au^2(t)] dt \rightarrow \min,$$

($a > 0$) стосовно $u \in L^2(0, L)$, $0 \leq u(t) \leq M$ ($M > 0$) для майже всіх $t \in (0, L)$, де (S, I, R) є розв'язком початкової задачі (8.3.43), (8.3.44).

Із формулювання задачі випливає, що досліджуватимемо мінімізацію інфекційних захворювань популяції, що приводить до мінімізації коштів вакцинації. Більше чи менше значення a означає більші чи менші мінімальні кошти вакцинації.

Записати принцип максимуму.

✓ *Вказівка.* Сформульована вище задача є частинним випадком задачі (P₁) (підрозділ 8.3.1.) у випадку $m = 1$, $N = 3$, $T := L$,

$$G(t, u, S, I, R) = I + au^2, \quad \varphi(S, I, R) = 0,$$

$$f(t, u, S, I, R) = \begin{pmatrix} mN - mS - cSI - uS \\ -mI + cSI - dI \\ -mR + uS + dI \end{pmatrix}$$

$$K = \{w \in L^2(0, L); 0 \leq w(t) \leq M \text{ для майже всіх } t \in (0, L)\}.$$

Наступну важливу задачу оптимального керування, пов'язану із SIR моделлю, пропонується розглянути самостійно: мінімізувати функціонал кошту

$$\int_0^L [I(t) - \tilde{I}(t)]^2 dt$$

стосовно $c \in [0, M]$ ($M > 0$), де $\tilde{I} \in C([0, L])$, $\tilde{I}(t) \geq 0$ для кожного $t \in [0, L]$ відома функція і (S, I, R) є розв'язком початкової задачі

$$\begin{cases} S'(t) = mN - mS(t) - cS(t)I(t), & t \in (0, L), \\ I'(t) = -mI(t) + cS(t)I(t) - dI(t), & t \in (0, L), \\ R'(t) = -mR(t) + dI(t), & t \in (0, L), \\ S(0) = S_0, \quad I(0) = I_0, \quad R(0) = R_0. \end{cases}$$

Тут m , d , S_0 , I_0 , R_0 задані сталі. Це така задача. Знаючи кількість інфікованих осіб у кожний момент часу, прагнемо визначити рівень інфікованих c (коефіцієнт).

8.3.6. Завдання для самостійного опрацювання

◆ *Завдання 8.3.1.* Сформулювати принцип максимуму задачі

$$\{x^u(T) + \gamma y^u(T)\} \rightarrow \max$$

де $u \in L^2(0, T)$, $0 \leq u(t) \leq 1$ для майже всіх $t \in (0, T)$, а (x^u, y^u) є

розв'язком системи хижак-жертва

$$\begin{cases} x'(t) = r_1x(t) - \mu_1u(t)x(t)y(t), & t \in (0, T), \\ y'(t) = -r_2y(t) + \mu_2u(t)x(t)y(t), & t \in (0, T), \\ x(0) = x_0, \quad y(0) = y_0. \end{cases}$$

✓ *Вказівка.* Ця задача є частинним випадком задачі (P₁), розглянутої у підрозділі 8.3.3. для $m = 1$, $N = 2$,

$$G(t, u, x, y) = 0, \quad \varphi(x, y) = x + \gamma y,$$

$$f(t, u, x, y) = \begin{pmatrix} r_1x - \mu_1uxy \\ -r_2y + \mu_2uxy \end{pmatrix},$$

$$K = \{w \in L^2(0, T); 0 \leq w(t) \leq 1 \text{ для майже всіх } t \in (0, T)\}.$$

◆ *Завдання 8.3.2.* Сформулювати принцип максимуму для задачі

$$\{x^u(T) + y^u(T)\} \rightarrow \max,$$

де $u \in L^2(0, T)$, $0 \leq u(t) \leq 1$ для майже всіх $t \in (0, T)$, а (x^u, y^u) є розв'язком системи хижак-жертва

$$\begin{cases} x'(t) = r_1x(t) - kx^2(t) - \mu_1u(t)x(t)y(t), & t \in (0, T), \\ y'(t) = -r_2y(t) + \mu_2u(t)x(t)y(t), & t \in (0, T), \\ x(0) = x_0, \quad y(0) = y_0. \end{cases}$$

Тут r_1 , r_2 , k , μ_1 , μ_2 – додатні константи; kx – додатковий коефіцієнт смертності зумовлений перенаселенням; kx^2 – логістичний доданок для популяції жертви.

✓ *Вказівка.* Розглянути аналогічно як у підрозділі 8.3.1.. Ця задача є частинним випадком задачі (P₁) з підрозділу 8.3.1..

◆ *Завдання 8.3.3.* Сформулювати принцип Понтрягіна для задачі оптимального розмноження

$$\int_0^T u(t)x^u(t)dt \rightarrow \max,$$

де $u \in L^2(0, T)$, $0 \leq u(t) \leq M$ ($M > 0$) для майже всіх $t \in (0, T)$, а x^u є розв'язком моделі Malthus'a динаміки популяції

$$\begin{cases} x'(t) = r(t)x(t) - u(t)x(t), & t \in (0, T), \\ x(0) = x_0. \end{cases}$$

Тут $x^u(t)$ означає щільність особин популяції виду у момент часу t ; $r \in C([0, T])$ – коефіцієнт розмноження; $u(t)$ – керування і відіграє роль додаткового коефіцієнта смертності. $\int_0^T u(t)x^u(t)dt$ означає загальний приплид популяції на часовому проміжку $[0, T]$.

✓ *Вказівка.* Нехай u^* означає оптимальне керування. Необхідні умови оптимальності першого порядку запишуться у вигляді

$$\begin{cases} p'(t) = -r(t)p(t) + u^*(t)(1 + p(t)), & t \in (0, T), \\ p(T) = 0, \end{cases}$$

$$u^*(t) = \begin{cases} 0, & \text{якщо } 1 + p(t) < 0, \\ M, & \text{якщо } 1 + p(t) > 0. \end{cases}$$

◆ *Завдання 8.3.4.* Записати принцип максимуму для задачі розмноження

$$\int_0^T u(t)x^u(t)dt,$$

де $u \in L^2(0, T)$, $0 \leq u(t) \leq M$ ($M > 0$) для майже всіх $t \in (0, T)$, а x^u є розв'язком наступної логістичної моделі динаміки популяції

$$\begin{cases} x'(t) = rx(t) - kx^2(t) - u(t)x(t), & t \in (0, T), \\ x(0) = x_0 > 0. \end{cases}$$

Тут r , k , x_0 задані додатні сталі.

◆ *Завдання 8.3.5.* Записати умови оптимальності задачі

$$\int_0^T u(t)x^u(t)dt - c \int_0^T u^2(t)dt \rightarrow \max,$$

де $u \in L^2(0, T)$, $0 \leq u(t) \leq M$ ($M > 0$) для майже всіх $t \in (0, T)$, а x^u є розв'язком наступної логістичної моделі динаміки популяції

$$\begin{cases} x'(t) = rx(t) - kx^2(t) - u(t)x(t), & t \in (0, T), \\ x(0) = x_0 > 0. \end{cases}$$

Тут c , r , k , x_0 – задані додатні сталі. Задача полягає у максимізації врожайності за мінімізацією видатків.

◆ *Завдання 8.3.6.* Створити процедуру обчислення матриці Q (див. алгоритм на стор. 247).

◆ *Завдання 8.3.7.* Дослідити задачу оптимального керування

$$\Psi(c) = \frac{1}{2} \int_0^L [G(t) - A]^2 dt \rightarrow \max, \quad (II)$$

стосовно $c = (c_1, c_2, \dots, c_m) \in \mathbb{R}^m$, де (I, G) – розв’язок уточненої задачі

$$\begin{cases} I'(t) = dI(t) + \sum_{j=1}^m c_j \delta_{t_j}, \\ G'(t) = (bI(t) + a)G(t), \\ I(0) = 0, \quad G(0) = G_0, \end{cases} \quad (8.3.45)$$

Для дослідження обидвох задач оптимального керування важлива умова обмеження на керування

$$c_j \geq 0, \quad j \in \{1, 2, \dots, m\}.$$

Звичайно, найліпшим контролем за рівнем глюкози є вплив на вміст інсуліну (виконуючи ін’єкції), так само як і зберігання концентрації глюкози (через дотримання дієти).

8.4. Оптимальне керування диференціальними системами. Градієнтний метод

У цьому розділі розглянемо апроксимаційні методи зазвичай градієнтного типу, розв'язування задач оптимального керування звичайними диференціальними рівняннями. Головне завдання полягає у написанні програм розв'язування такого типу задач у середовищі MatLab®. Обчислення градієнта функції корисності допоможе нам побудувати алгоритм градієнтного типу. Досліджуватимемо задачі мінімізації і максимізації функції корисності. Як буде показано нижче, загальний підхід до розв'язування задач обох типів градієнтним методом є такий самий.

Вперше чисельна апроксимація оптимального керування була розглянута у праці [61] і [29]. У пізніших виданнях отримано нові результати, наприклад, у [30], [42], [56].

Як факт, у цьому розділі градієнтний метод як ітераційна процедура використаний у вигляді формули $u^{(k+1)} = u^{(k)} + \rho_k w^{(k)}$, де $u^{(k)}$ – відоме керування, $w^{(k)}$ – шуканий напрямок і ρ_k – крок ітерації. В описаних вище процедурах використано тільки напрям градієнта для $w^{(k)}$ і метод Армію для обчислення ρ_k . Програма, в якій використано згадані значення, працює добре. Взагалі у літературі використовуються інші методи обчислення та вибору $w^{(k)}$ та ρ_k (див., наприклад, [15, Розділ 2]).

Наприклад, у розділі 8.4.2. ми цитували [51], а у розділі 8.4.3. [33]. Приклад у розділі 8.4.4. використаний із [24]. Темі оптимального розмноження присвячено досить багато літератури. У розділі 8.5. будуть розглянуті узагальнення моделей із розділу 8.4.4.

8.4.1. Метод проєкції градієнта

Опишемо й обґрунтуємо чисельну апроксимацію побудови розв'язку задач оптимального керування методом градієнтів. Застосування побудованих алгоритмів знаходження розв'язків задач оптимального керування розглянемо на прикладах із підрозділу 8.3.1. Метод градієнтів використовується до задач максимізації та мінімізації і підходить до побудови розв'язків у обидвох випадках принципово не відрізняються. Такий метод інтерактивний а отже, на кожному кроці ітерації локальний пошук поліпшує значення функції корисності (значення спадають у випадку мінімізації та зростають у випадку максимізації). Якщо значення керування на k -му кроці ітерації $u_k = u^{(k)}$, тоді, відповідно, обчислюємо x^{u_k} (розв'язок рівняння стану, в якому підставили u_k), p_k (розв'язок спряженого рівняння, в якому підставили u_k, x^{u_k}) та градієнт $\Phi_u(u_k)$,

який позначимо $\nabla_u \Phi(u_k)$. Формулу, яка містить p^{u_k} (але не містить x^{u_k}), отримано з градієнта. Цей метод, який називається виділенням стану, ввів Ж. С'еа [30]. Для прикладу розглянемо задачу керування із підрозділу 8.3.1. у випадку $\varphi = 0$

$$\Phi(u) = \mathcal{L}(u, x^u) = \int_0^T G(t, u(t), x^u(t)) dt \rightarrow \max \quad (P1')$$

стосовно $u \in K \subset L^2(0, T; \mathbb{R}^m)$ ($T > 0$), де x^u є розв'язком початкової задачі

$$\begin{cases} x'(t) = f(t, u(t), x(t)), & t \in (0, T), \\ x(0) = x_0. \end{cases} \quad (8.4.1)$$

Визначимо функції G і f

$$\begin{aligned} G &: [0, T] \times \mathbb{R}^m \times \mathbb{R}^N \rightarrow \mathbb{R}, \\ f &: [0, T] \times \mathbb{R}^m \times \mathbb{R}^N \rightarrow \mathbb{R}^N, \end{aligned}$$

$x_0 \in \mathbb{R}^N$, $K \subset U$ – замкнута опукла підмножина.

Оскільки функція корисності у моделях, які досліджуються у цьому розділі, не містить значення розв'язку у кінцевий момент часу $x(T)$, виключимо доданок $\varphi(x^u(T))$, який є у задачах підрозділу 8.3.1. Однак дослідження аналогічне як і у випадку, коли $\varphi(x^u(T))$ входить до функції корисності. Задачу (P1') можна переформулювати також як задачу мінімізації

$$\Psi(u) = -\mathcal{L}(u, x^u) \rightarrow \min,$$

стосовно $u \in K \subset U$, де $\Psi(u) = -\Phi(u)$.

Нехай (u^*, x^{u^*}) – оптимальна пара задачі (P1'). У кожен момент часу припускаємо, що G і f достатньо гладкі і всі оператори, які визначаються через ці функції, визначені.

Припустимо, що функція $u \mapsto x^u$ диференційовна за Gâteaux. Позначимо через dx^u цей диференціал. Розглянемо довільні фіксовані функції $u, v \in U$ і нехай

$$V = \{v \in U : u + \varepsilon v \in K \text{ для кожного достатньо малого } \varepsilon > 0\}.$$

Для кожної функції $v \in V$ визначимо $z = dx^u(v)$; z є розв'язком початкової задачі

$$\begin{cases} z'(t) = f_u(t, u(t), x^u(t))v(t) + f_x(t, u(t), x^u(t))z(t), & t \in (0, T), \\ z(0) = 0. \end{cases} \quad (8.4.2)$$

Для довільної фіксованої функції $v \in V$ маємо

$$(v, \Phi_u(u)) = \lim_{\varepsilon \rightarrow 0^+} \frac{\Phi(u + \varepsilon v) - \Phi(u)}{\varepsilon},$$

де (\cdot, \cdot) – скалярний добуток у U . Звідси випливає

$$(v, \Phi_u(u)) = \int_0^T [v(t) \cdot G_u(t, u(t), x^u(t)) + z(t) \cdot G_x(t, u(t), x^u(t))] dt. \quad (8.4.3)$$

Нехай p^u є розв'язком спряженої задачі

$$\begin{cases} p'(t) = -f_x^*(t, u(t), x^u(t))p(t) - G_x(t, u(t), x^u(t)), & t \in (0, T), \\ p(T) = 0. \end{cases} \quad (8.4.4)$$

Помножимо рівняння (8.4.2) на p^u та проінтегруємо частинами на проміжку $[0, T]$. Взявши до уваги (8.4.3), аналогічно як у підрозділі 8.3.1., отримаємо

$$\int_0^T z(t) \cdot G_x(t, u(t), x^u(t)) dt = \int_0^T v(t) \cdot f_u^*(t, u(t), x^u(t)) p^u(t) dt. \quad (8.4.5)$$

Отож, із (8.4.5) і (8.4.3) отримаємо

$$\Phi_u(u) = G_u(\cdot, u, x^u) + f_u^*(\cdot, u, x^u) p^u. \quad (8.4.6)$$

Використовуючи (8.4.1), (8.4.4) та (8.4.6) побудуємо ітерації, за допомогою яких обчислимо значення функції корисності на кожному ітераційному кроці (або, що означає те ж саме, апроксимуємо оптимальне керування u^*). Використаємо градієнт функціонала корисності Φ для того, щоб отримати спадну послідовність функціонала корисності $\Phi(u)$ на кожному кроці ітерації. Застосуємо метод проєкції градієнта, щоб керувати обмеженнями $u \in K$. Нижче наводимо алгоритм розв'язування задачі (P'_1) , знаний як алгоритм Uzawa (див. наприклад, [15, розділ 2.5])

S0: Беремо $u^{(0)} \in K$.

Прийmemo $k := 0$.

S1: Знайдемо розв'язок $x^{(k)}$ задачі (8.4.1), який відповідає $u := u^{(k)}$

$$\begin{cases} x'(t) = f(t, u^{(k)}(t), x(t)), & t \in (0, T), \\ x(0) = x_0. \end{cases}$$

S2: Визначимо $p^{(k)}$ розв'язок (8.4.4) для $u := u^{(k)}$, $x := x^{(k)}$:

$$\begin{cases} p'(t) = -f_x^*(t, u^{(k)}(t), x^{(k)}(t))p(t) - G_x(t, u^{(k)}(t), x^{(k)}(t)), & t \in (0, T), \\ p(T) = 0. \end{cases}$$

S3: Обчислимо градієнт $w^{(k)}$ за формулою (8.4.6):

$$w^{(k)} := \Phi_u(u^{(k)}) = G_u(\cdot, u^{(k)}, x^{(k)}) + f_u^*(\cdot, u^{(k)}, x^{(k)})p^{(k)}.$$

S4: Виберемо крок ітерації $\rho_k \geq 0$ так, щоб

$$\Phi(P_k(u^{(k)} + \rho_k w^{(k)})) = \max_{\rho \geq 0} \Phi(P_k(u^{(k)} + \rho w^{(k)})).$$

S5: $u^{(k+1)} := P_k(u^{(k)} + \rho_k w^{(k)})$.

S6: (Критерій завершення обчислень)

Якщо $\|u^{(k+1)} - u^{(k)}\| < \varepsilon$
тоді STOP ($u^{(k+1)}$ є наближенням керування)
якщо ні $k := k + 1$; на виконання **S1**.

У випадку задачі на мінімум алгоритм наближеного знаходження розв'язку методом проєкції градієнта у загальному буде таким самим, за винятком двох відмінностей:

- на етапі **S3** приймаємо $w^{(k)} := -\Phi_u(u^{(k)})$ для того, щоб отримати спадну послідовність;
- на етапі **S4** замість задачі максимізації $-\max_{\rho \geq 0}$ знаходимо $\min_{\rho \geq 0}$.

У сучасній науковій літературі цей метод називають *методом найшвидшого спуску*. Крім того, "напрямок" означає вектор, значення якого визначено $\rho > 0$. Взагалі кажучи, на практиці можна використати інші критерії зупинки програми обчислення наближеного розв'язку, які застосовують для розв'язування конкретних прикладів.

Припустимо, що P_K визначений вище, є проєкцією оператора на опуклу множину K ; це означає, що $P_K : U \rightarrow K$ визначений так:

$$\|P_K(u) - u\| \leq \|w - u\| \quad \text{для кожної функції } w \in K,$$

де через $\|\cdot\|$ позначили норму у просторі U . $\varepsilon > 0$ означає точність обчислення на кроці **S6**. Якщо керування необмежене (тобто $K = U$),

тоді $P_K = P_U = I_U$ і на етапі **S4** поступаємо чинимо так.

Обчислимо крок ітерації $\rho_k \geq 0$ так, що

$$\Phi(u^{(k)} + \rho_k w^{(k)}) = \max_{\rho \geq 0} \{\Phi(u^{(k)} + \rho w^{(k)})\}.$$

Обчислення кроку ітерації ρ_k за алгоритмом на кроці **S4** є трудомною задачею. Для того, щоб обчислити величину ρ_k , спробуємо визначити надійний ефективний метод. Якщо відома величина $\rho > 0$, тоді обчислюємо $\Phi(P_K(u^{(k)} + \rho w^{(k)}))$, а для розв'язування задачі багатократно повторюємо крок **S4**. Такі перевірки потрібні для знаходження правильного кроку ітерації. Для цього ми повинні розглянути такі підетапи:

- обчислити $u := u^{(k)} + \rho w^{(k)}$;
- обчислити $\bar{u} := P_K(u)$;
- обчислити \bar{x} , розв'язок початкової задачі (8.4.1), що відповідає $u := \bar{u}$;
- обчислити відповідне значення функції корисності $\Phi(\bar{u})$.

Очевидно, що ця задача потребує великих обчислень. Про метод спуску й ефективні алгоритми обчислення ітераційного кроку детальніше можна знайти у [15, Розділ 2.3] та [42, Глава 2].

Можна застосувати такий простий алгоритм у частковому випадку, коли $K = U$.

Рекурсивний метод прогонки

NS'0 Вибираємо $u^{(0)} \in U$.

Приймаємо $k := 0$.

NS'1 Обчислюємо розв'язок $x^{(k)}$ задачі (8.4.1), що відповідає $u := u^{(k)}$

$$\begin{cases} x'(t) = f(t, u^{(k)}(t), x(t)), & t \in (0, T), \\ x(0) = x_0. \end{cases}$$

NS'2 Обчислюємо розв'язок $p^{(k)}$ задачі (8.4.4), який відповідає $u := u^{(k)}$

$$\begin{cases} p'(t) = -f_x^*(t, u^{(k)}(t), x^{(k)}(t))p(t) - G_x(t, u^{(k)}(t), x^{(k)}(t)), & t \in (0, T), \\ p(T) = 0. \end{cases}$$

NS'3 Обчислюємо $u^{(k+1)}$ як розв'язок рівняння

$$G_u(t, u(t), x^{(k)}(t)) + f_u^*(t, u(t), x^{(k)}(t))p^{(k)} = 0.$$

NS'4 (Критерій зупинки) (Критерій завершення обчислень)

Якщо $\|u^{(k+1)} - u^{(k)}\| < \varepsilon$
 тоді STOP ($u^{(k+1)}$ – наближення керування)
 якщо ні $k := k + 1$; на виконання **NS'1**.

Рекурсивний метод прогонки використовується тоді, коли є обмеження на керування.

Подамо теоретичні основи методу найшвидшого спуску задачі мінімізації та пов'язану з нею задачі максимізації. Розглянемо задачу мінімізації функції $\Psi : U \mapsto \mathbb{R}$ у дійсному просторі Hilbert'a U . Ми припускаємо, що U ототожнюється зі спряженим до себе. Використовуватимемо стандартні позначення $\nabla\Psi$ для Ψ , оскільки функція Ψ залежить тільки від u . Мінімізуючі ітерації будемо за формулою

$$u^{(k+1)} = u^{(k)} + \rho_k w^{(k)}, \quad (8.4.7)$$

де $w^{(k)} \in U$ – *шуканий напрям*, а $\rho_k \in \mathbb{R}$ є *кроком* ітерації. Кажемо, що $w \in U$ визначає напрям спуску функції Ψ у точці $u \in U$, якщо існує $\rho_0 > 0$ таке, що

$$\Psi(u + \rho w) < \Psi(u) \quad \forall \rho \in (0, \rho_0]. \quad (8.4.8)$$

Припустимо, що градієнт Ψ є неперервним на U . Тоді w буде напрямком спуску в точці $u \in U$, якщо виконується нерівність

$$(w, \nabla\Psi(u)) < 0, \quad (8.4.9)$$

де (\cdot, \cdot) означає скалярний добуток в U , а через $\|\cdot\|$ будемо позначати відповідну йому норму.

Розглянемо питання визначення кроку напрямку спуску. Якщо $\nabla\Psi(u) \neq 0$, то, застосовуючи формулу Taylor'a, одержимо

$$\Psi(u + \rho w) = \Psi(u) + \rho(w, \nabla\Psi(u)) + o(\rho).$$

Якщо w є напрямком спуску в u , тоді виконується (8.4.9) і крок спуску визначимо, мінімізуючи $(w, \nabla\Psi(u))$. Отож, ми одержимо

$$\min\{(w, \nabla\Psi(u)); \|w\| = 1\}. \quad (8.4.10)$$

Використовуючи нерівність Schwarz'a

$$|(w, \nabla\Psi(u))| \leq \|\nabla\Psi(u)\|,$$

одержимо

$$-\|\nabla\Psi(u)\| \leq |(w, \nabla\Psi(u))| \quad \text{якщо} \quad \|w\| = 1.$$

Очевидно, що $w = -\nabla\Psi(u)/\|\nabla\Psi(u)\|$ визначає мінімум (8.4.10) і напрям найшвидшого спуску в u дорівнює $-\nabla\Psi(u)$.

Розглянемо задачу знаходження максимуму $\Psi(u)$ стосовно $u \in U$. Напрямок w , який забезпечує локальне зростання Ψ по u , визначимо аналогічно як у випадку спадання функції, так:

$$(w, \nabla\Psi(u)) > 0. \quad (8.4.11)$$

Аналогічно, застосувавши формулу Taylor'а, одержимо

$$\max\{(w, \nabla\Psi(u)); \|w\| = 1\} \quad (8.4.12)$$

максимум зростання Ψ в u . Із нерівності Schwarz'а легко отримаємо

$$|(w, \nabla\Psi(u))| \leq \|\nabla\Psi(u)\|, \quad \text{якщо} \quad \|w\| = 1.$$

Очевидно, що $w = \nabla\Psi(u)/\|\nabla\Psi(u)\|$ максимізує (8.4.12). Більше того, напрям локального максимального зростання Ψ в $u \in U$ є $\nabla\Psi(u)$. Отже, відмінність між задачею мінімізації і максимізації полягає у знаку градієнта. Однак обґрунтування збіжності у методі найшвидшого спуску складніше, і важливу роль відіграє послідовність $\{\rho_k\}$ кроку ітерації ([15, розділ 2], [42, розділи 7,8]). Варто зауважити, що можна назвати інші джерела вивчення задач оптимального керування, де описані зв'язані з ними методи проєкції градієнта. Деякі з таких задач розглянемо у наступних підрозділах.

Повернемося до задачі керування. Розглянемо задачу мінімізації

$$\Psi(u) \rightarrow \min,$$

стосовно $u \in K \subset U$, де K – замкнута опукла підмножина. Доведемо збіжність методу проєкції градієнта до єдиного оптимального керування, якщо функціонал Ψ строго опуклий і достатньо гладкий (градієнт є неперервним) і якщо K є обмеженою множиною або Ψ є коерцетивний на K ($\lim_{\|u\| \rightarrow \infty} \Psi(u) = +\infty$). Якщо керування не є обмеженим, тоді метод проєкції градієнта є збіжний (див. [15, розділ 2.1]). У інших випадках обґрунтування складніше [61, 30, 42, 15]. Загалом маємо алгоритм знаходження зростаючих / спадних ітерацій для функції корисності.

Метод прогонки є наслідком принципу Понтрягіна. Якщо позначити $u^{(k+1)} = \Gamma u^{(k)}$, де Γ – оператор алгоритму, тоді метод ітерацій з теореми Banach'а про нерухому точку можна використати для обґрунтування збіжності (у випадку обмеженості Γ).

8.4.2. Метод найшвидшого спуску. Приклад

Розглянемо приклад застосування методу найшвидшого спуску, обчислення градієнта та подамо відповідні програми. На цьому простому прикладі продемонструємо ефективні методи апроксимації оптимального керування та порівняємо апроксимоване керування з точним.

★ *Приклад 8.4.1.* У біохімічній реакції складова додається зі сталою швидкістю протягом часового інтервалу $[0, L]$ ($L > 0$) ([51, розділ III, параграф 1]). Нехай $x(t)$ означає відхилення рН рівня у момент часу t . Ми повинні контролювати рН баланс, бо від цього залежить якість продуктів. Керування виконується за допомогою концентрації $u(t)$ керуючого інгредієнта. Динаміка змінної x описується такою початковою задачею

$$\begin{cases} x'(t) = \alpha x(t) + \beta u(t), & t \in (0, L), \\ x(0) = x_0, \end{cases} \quad (8.4.13)$$

де α і β – відомі додатні сталі, а x_0 є початковим відхиленням рН балансу. Припустимо, що відбувається зростання відхилення рН балансу і відповідна зміна рН становить $\int_0^L x^2(t) dt$. Припустимо також, що порядок ціни відповідного зусилля u пропорціональний до u^2 . Отож, ця біохімічна модель зводиться до такої задачі оптимального керування:

$$\frac{1}{2} \int_0^L [a(x^u(t))^2 + u^2(t)] dt \rightarrow \min, \quad (\text{TP})$$

стосовно $u \in U = L^2(0, L)$, де x^u є розв'язком (8.4.13) і a додатною сталою. Варто зазначити, що у цій задачі немає обмеження на керування.

Задача (TP) є частинним випадком задачі (P₁) (підрозділ 8.3.1.), де $m = 1$, $N = 1$, $T = L$,

$$\begin{aligned} G(t, u, x) &= -\frac{1}{2}(ax^2 + u^2), \\ \varphi(x) &= 0, \\ f(t, u, x) &= \alpha x + \beta u. \end{aligned}$$

У нашому випадку

$$\mathcal{L}(u, x) = -\frac{1}{2} \int_0^L [ax^2(t) + u^2(t)] dt,$$

$$\Phi(u) = -\frac{1}{2} \int_0^L [a(x^u(t))^2 + u^2(t)] dt,$$

а тому задача (TP) може бути переформульована як задача мінімізації

$$-\Phi(u) \rightarrow \min \quad \text{стосовно} \quad u \in U.$$

Для того, щоб викладки методу були зрозумілими, позначимо через $\Psi(u)$ функціонал корисності вихідної задачі, тобто

$$\Psi(u) = \frac{1}{2} \int_0^L [a(x^u(t))^2 + u^2(t)] dt,$$

і тоді задача зводиться до мінімізації функціоналу Ψ

$$\Psi(u) \rightarrow \min,$$

стосовно $u \in U$. Найперше обчислимо градієнт функціонала корисності. Як вже було відзначено, звідси ми отримаємо необхідні умови оптимальності. А це означає, що градієнт функціонала корисності обчислимо за допомогою спряженої системи (шляхом виключення стану).

Розглянемо дві довільні фіксовані функції $u, v \in L^2(0, L)$. Нехай z задовольняє таку початкову задачу:

$$\begin{cases} z'(t) = \alpha z(t) + \beta v(t), & t \in (0, L), \\ z(0) = 0. \end{cases} \quad (8.4.14)$$

Запишемо спряжену задачу

$$\begin{cases} p'(t) = -\alpha p(t) + ax^u(t), & t \in (0, L), \\ p(L) = 0. \end{cases} \quad (8.4.15)$$

Для кожного $\varepsilon \in \mathbb{R}^*$ матимемо

$$\begin{aligned} \Psi(u + \varepsilon v) - \Psi(u) = \\ \frac{1}{2} \int_0^L [a(x^{u+\varepsilon v}(t))^2 + (u(t) + \varepsilon v(t))^2 - a(x^u(t))^2 - u^2(t)] dt. \end{aligned}$$

Оскільки задача (8.4.13) є лінійна, то неважко записати її розв'язок

$$x^{u+\varepsilon v} = x^u + \varepsilon z,$$

звідки одержуємо

$$\frac{1}{\varepsilon}[\psi(u + \varepsilon v) - \Psi(u)] = \frac{1}{2} \int_0^L [2ax^u(t)z(t) + \varepsilon az^2(t) + 2u(t)v(t) + \varepsilon v^2(t)] dt.$$

Спрямувавши ε до нуля, одержимо

$$(v, \Psi_u(u)) = \int_0^L [ax^u(t)z(t) + u(t)v(t)] dt, \quad (8.4.16)$$

де (\cdot, \cdot) позначено скалярний добуток у $L^2(0, L)$, який обчислюється за формулою

$$(g_1, g_2) = \int_0^L g_1(t)g_2(t) dt, \quad g_1, g_2 \in L^2(0, L).$$

Наступним кроком виключимо x^u із (8.4.16). Для цього домножимо диференціальне рівняння (8.4.2) на p^u та проінтегруємо на проміжку $[0, L]$

$$\int_0^L z'(t)p^u(t) dt = \int_0^L [\alpha z(t) + \beta v(t)]p^u(t) dt,$$

звідки, беручи до уваги $z(0) = p(L) = 0$, одержуємо

$$-\int_0^L z(t)(p^u)'(t) dt = \int_0^L [\alpha z(t) + \beta v(t)]p^u(t) dt.$$

Із (8.4.14) і (8.4.15) випливає

$$-\int_0^L z(t)[- \alpha p^u(t) + ax^u(t)] dt = \int_0^L [\alpha z(t) + \beta v(t)]p^u(t) dt,$$

а отже,

$$\int_0^L ax^u(t)z(t) dt = - \int_0^L \beta v(t)p^u(t) dt. \quad (8.4.17)$$

Із (8.4.17) та (8.4.16) одержимо

$$(v, \Psi_u(u)) = \int_0^L v(t)[u(t) - \beta p^u(t)]dt.$$

Остання рівність справджується для кожної функції $v \in L^2(0, L)$. Звідси робимо висновок, що

$$\Psi_u(u) = u - \beta p^u. \quad (8.4.18)$$

Якщо u^* є оптимальним керуванням задачі (TP), тоді

$$\Psi_u(u^*) = u^* - \beta p^{u^*},$$

де p є розв'язок задачі (8.4.15), який відповідає $u := u^*$.

Отже, підводячи підсумок, подамо алгоритм градієнтного методу, в якому використовуємо $-\Psi_u(u)$ як напрям найшвидшого спуску при керуванні u .

Алгоритм методу найшвидшого спуску розв'язування задачі (TP)

S0: Виберемо $u^{(0)} \in U$.

Приймаємо $k := 0$.

S1: Обчислимо розв'язок $x^{(k)}$ задачі (8.4.13), в якій замість u підставимо $u^{(k)}$

$$\begin{cases} x'(t) = \alpha x(t) + \beta u^{(k)}(t), & t \in (0, L), \\ x(0) = x_0. \end{cases}$$

S2: Обчислимо розв'язок $p^{(k)}$ задачі (8.4.15), в якій замінімо x^u на $x^{(k)}$

$$\begin{cases} p'(t) = -\alpha p(t) + \beta x^{(k)}(t), & t \in (0, L), \\ p(L) = 0. \end{cases}$$

S3: Обчислимо градієнт $w^{(k)}$ за формулою (8.4.18)

$$w^{(k)} := \Psi_u(u^{(k)}) = u^{(k)} - \beta p^{(k)}.$$

S4: Обчислимо крок ітерації ρ_k із співвідношення

$$\Psi(u^{(k)} - \rho_k w^{(k)}) = \min_{\rho \geq 0} \{\Psi(u^{(k)} - \rho w^{(k)})\}.$$

S5: Обчислимо значення керування на наступному ітераційному кроці

$$u^{(k+1)} := u^{(k)} - \rho_k w^{(k)}.$$

S6: (Критерій закінчення ітераційного процесу)

Якщо $\|u^{(k+1)} - u^{(k)}\| < \varepsilon$,

тоді закінчити ітераційний процес ($u^{(k+1)}$ – наближене значення керування);

в противному випадку $k := k + 1$ і повторюємо ітераційний цикл із **S1**.

На кроці **S6** використовується дискретна норма, а ε є точністю наближення. Нижче доведемо, що критерій зупинки ітераційного обчислення можна використати на практиці.

З метою практичного застосування дамо декілька коментарів. У багатьох задачах на керування накладаються певні обмеження; наприклад, $u \in K \subset U$, де K – замкнена опукла підмножина. У цьому випадку може бути використаний метод проєкції градієнта (див. напр., [15, розділ 2.5]). Наприклад, використовуючи алгоритм Uzawa, формула на кроці **S4** має вигляд

$$\Psi(P_K(u^{(k)} - \rho_k w^{(k)})) = \min_{\rho \geq 0} \{\Psi(P_K(u^{(k)} - \rho w^{(k)}))\},$$

де $P_K : U \rightarrow K$ означає проєкцію оператора (див. напр., [15, розділ 2.4]). Тоді формула на кроці **S5** ітераційного процесу, відповідно, набуває вигляду

$$u^{(k+1)} := P_K(u^{(k)} - \rho_k w^{(k)}).$$

Щоб можна було застосувати вищезазначені поправки в алгоритмі, ми повинні подати формулу для обчислення P_K з метою їх застосування у разі написання програми. Наприклад, зазвичай обмеження на керування задається у вигляді

$$|u(t)| \leq M \quad \text{для майже всіх } t \in (0, L)$$

($K = \{w \in L^2(0, L); |w(t)| \leq M \text{ для майже всіх } t \in (0, L)\}$). У цьому випадку маємо $P_K : L^2(0, L) \rightarrow K$,

$$P_K(u)(t) = Proj(u(t)) \quad \text{для майже всіх } t \in (0, L),$$

де

$$Proj(w) = \begin{cases} w, & \text{якщо } -M \leq w \leq M, \\ -M, & \text{якщо } w < -M, \\ M, & \text{якщо } w > M. \end{cases}$$

Звідси останню формулу використаємо для визначення функції у середовищі MatLab[®]

```
function y=Proj(u)
    global M
    y=u;
    if u<=-M
        y=-M;
    end
    if u>M
        y=M ;
    end
end
```

Перейдемо до написання програми чисельного розв'язання задачі оптимального керування методом проєкції градієнта, описаного вище. Щоб програма була зрозуміліша, ми програму поділимо на частини, кожна з яких обчислює певні кроки алгоритму. Загалом програма є конкатенацією (об'єднанням) частин.

Найперше зауважимо, що часовий інтервал розіб'ємо сіткою вузлів так:

$$t_i = (i - 1)h, \quad i = 1, 2, \dots, n,$$

де

$$h = \frac{L}{n - 1}.$$

Використаємо такі ідентифікатори для позначення векторів

$$\mathbf{u}^{\text{old}} \text{ для } u^{(k)}, \quad \mathbf{u}^{\text{new}} \text{ для } u^{(k+1)}.$$

Розпочнемо з частини першої PART 1, у якій зазвичай вводяться початкові дані, вхідні параметри, а також крок **S0** алгоритму. Виберемо за початкове значення керування $u^{(0)}(t)$ деяку сталу, а це означає, що деякі вхідні параметри будуть визначені пізніше.

```
% file CONTO.m
% Optimal control for the pH in a (bio)chemical reaction
% gradient algorithm with Armijo method for the
% steplength rho
% =====
% PART 1
% =====
```

```

clear
global alf bet ind uold
global u
global x
L=input('final time: ');
alf=input('alpha: '); % alpha
bet=input('beta: '); % beta
a=input('a: ');
x0=input('x(0): ');
h=input('h: '); % time grid step
t=0:h:L; % time grid
t=t'; % change to column vector
n=length(t);
x=zeros(n,1); % state vector
x1=zeros(n,1); % state vector to be used for rho loop
                % (Armijo method)
p=zeros(n,1); % adjoint state vector
uold=zeros(n,1); % control vector corresponding to  $u^{\wedge}(k)$ 
unew=zeros(n,1); % control vector corresponding to  $u^{\wedge}(k+1)$ 
u=zeros(n,1); % control vector to be used for rho loop
                % (Armijo method)
% S0: control initialization
u0=input('u0: ');
for i=1:n
uold(i)=u0;
end
disp('enter control data');
eps=input('precision: '); % precision epsilon for the
                        % gradient algorithm
maxit=input('maxiter: '); % max. no. iterations -
                        % gradient algorithm
disp('RO data');
roin=input('RO: '); % initial value for gradient
                % steplength rho
bro=input('b for RO: '); % b parameter for rho loop
eps1=input('precision for RO: '); % precision epsilon_1 for
                % steplength rho
maxro=input('max for RO: '); % max. no. of iterations -
                % rho loop
ro=roin; % initialization of steplength rho
flag1=0; % convergence indicator for gradient algorithm
ii=0; % index for gradient values

```

Значення `flag1` означає:

- 0 – якщо не досягнуто заданої точності;
- 1 – точність обчислення досягнута.

Змінна `ii` зберігає індекси вектора `grad`, якому присвоюється обчислений градієнт $\Psi_u(u^{(k)})$ при обчисленні наступних ітерацій. Цей вектор містить компоненти після їх обчислення на кожному кроці ітерації. Перейдемо до написання другої частини програми.

```
% =====
% PART 2
% =====
% gradient loop starts
for iter=1:maxit
    iter
    % S1: solve the state equation for x(t) by
    %Runge-Kutta method
    x(1)=x0; % the initial condition
    for i=1:n-1
        ind=i; % ind is a global index variable used by the
                % rhs of system

        x(i+1)=RK41(x(i),t(i));
    end
    disp('SE solved');
    if iter == 1
        Q=a*x.^2 + uold.^2;
        temp=trapz(t,Q);
        cvold=temp/2
        cost(1)=cvold; % store the value of the cost functional
        jj=1; end
```

Задача (8.4.13) розв'язується стандартним методом Runge-Kutta порядку 4 (див. підрозділ 6.3.). Для знаходження наближеного розв'язку не використовуємо функцію `ode`, бо ускладнюється обчислення керування u . Застосовуючи внутрішню функцію `ode`, MatLab[®] повертає відразу розв'язок на розглядуваному часовому інтервалі, а нам потрібне значення вектора у момент часу t . Тому, як зазначено вище, ми безпосередньо будемо процедуру обчислення розв'язку методом Runge-Kutta і ввели глобальну змінну `ind` і відповідне значення `uold(ind)`. Визначимо функцію `RK41.m`, за допомогою якої обчислюємо методом Runge-Kutta

```
function yout=RK41(x,t)
    global h
    tm=t+h/2;
    k1=h*F1(t,x);
    k2=h*F1(tm,x+k1/2);
    k3=h*F1(tm,x+k2/2);
    k4=h*F1(t+h,x+k3);
    yout=x+(k1+k4+2.0*(k2+k3))/6.0;
end
```

Функція F1.m, яка використовується у RK41.m, визначається так:

```
function yout=F1(t,x)
    global alf bet ind
    global uold
    yout=alf*x+bet*uold(ind);
end
```

Отже, при $\text{iter}=1$ обчислюємо значення функціонала корисності, використовуючи функцію `trapz`, і результат присвоюємо `cvold` та `cost(1)`. Змінна `jj` дорівнює індексу вектора `cost`, який зберігає значення функціонала корисності $\Psi(u^{(k)})$ для наступних ітерацій. Після кожного кроку ітерації цьому вектору присвоюються обчислені компоненти.

Перейдемо до етапу **S2** алгоритму. Для знаходження розв'язку двоїстої задачі (8.4.15) застосуємо стандартний метод Runge-Kutta порядку 4. Він відрізняється від попереднього випадку тільки тим, що під час обчислення змінюється напрям зміни t від L до 0. Обчислимо градієнт за допомогою процедури:

```
% =====
% PART 3
% =====
% S2 : solve adjoint equation for p(t):
p(n)=0;
for j=1:n-1
    i=n-j;
    ip1=i+1;
    ind=ip1;
    p(i)=RK43SM(p(ip1),t(ip1));
end
disp('AE solved');
% S3: compute the gradient
```

```
w=uold-bet*p;
disp('GRADIENT COMPUTED');
```

Скрипт функції RK43, яка проводить обчислення за методом Runge-Kutta у зворотному за часом напрямі, такий:

```
function yout=RK43(q,t)
    global h
    h1=-h;
    tm=t+h1/2;
    k1=h1*F3(t,q);
    k2=h1*F3(tm,q+k1/2);
    k3=h1*F3(tm,q+k2/2);
    k4=h1*F3(t+h1,q+k3);
    yout=q+(k1+k4+2.0*(k2+k3))/6.0;
end
```

Функція *F3.m* визначається так:

```
function yout = F3(t,p)
    global alf bet ind
    global a
    global x
    yout=a*x(ind)-alf*p;
end
```

Перш ніж перейти до наступного кроку **S4**, обговоримо критерій зупинки алгоритму. Один із них описаний на етапі **S6**, але замінимо його на один із таких:

- **SC1** : $\|\Psi_u(u^{(k)})\| < \varepsilon$;
- **SC2** : $\|\Psi(u^{(k+1)}) - \Psi_u(u^{(k)})\| < \varepsilon$;
- **SC3** : $\|u^{(k+1)} - u^{(k)}\| < \varepsilon$;
- **SC4** : $\rho_k < \varepsilon_1$.

Зауважимо, що **SC1** і **SC3** еквівалентні, оскільки

$$\|u^{(k+1)} - u^{(k)}\| = \rho_k \|\Psi_u(u^{(k)})\|.$$

Із тестування умови **SC4** випливає, що ітераційний крок ρ дуже малий і у підсумку чисельно не вдається отримати спуску у напрямі $-w^{(k)}$, де $w^{(k)} = \Psi_u(u^{(k)})$. Однак можна довести, що критерій **SC4** чисельно еквівалентний **SC2**. Ідея процедури полягає у тому, щоб зупинити її

виконання, коли буде виконуватися один із критеріїв. Іншими словами, якщо матимемо обчислений градієнт $\Psi_u(u^{(k)})$, тоді можемо застосувати критерій **SC1** і продовжити виконання програми за допомогою такого сценарію:

```
% =====
% PART 4
% =====
normg=sqrt(sum(w.^2)) % compute the gradient norm
ii=ii+1;
grad(ii)=normg;
% verify SC1
if normg < eps
    disp('CONVERGENCE by GRADIENT')
    flag1=1;
    break
end
```

Опишемо виконання частини **PART 4** програми, визначеної вище. Якщо виконується критерій зупинки (тобто $\text{normg} < \text{eps}$), тоді індикатору збіжності **flag1** присвоюється значення 1 і виконується безумовна інструкція **break**, яка зупиняє ітераційний цикл, що починається у частині **PART 2** програми командою `for iter=1:maxit`. Перейдемо зараз до кроку **S5** алгоритму як одного з найскладніших, внаслідок виконання якого обчислюється крок ρ_k . Формулою, яка записана в **S5**, користуватися незручно, позаяк зазвичай не можна обчислити мінімуму. Тому ми не будемо обчислювати точного мінімуму, розглянемо апроксимаційний алгоритм, який називають *методом Арміжо* (детальніше див. [14], [15, розділ 2.3], [61, додаток С.2]). Отже, маємо обчислене значення $\Psi(u^{(k)})$. Останнє обчислене значення кроку ρ дорівнює ρ_{k-1} . Також відома величина параметра $b \in (0.5, 0.8)$, яка є наближенням ρ (вводиться у програмі командою `input` і присвоюється змінній `bro`). Також треба наголосити у позначенні $\Psi(u) = \psi(u, x^u)$ важливість залежності від траєкторії x . Тут

$$\psi(u, x^u) = \frac{1}{2} \int_0^L [ax^2(t) + u^2(t)] dt.$$

Метод Арміжо визначення кроку ітерації

- **A0:** Приймемо $\bar{\rho} := \rho_{k-1}$.
- **A1:** $u := u^{(k)} - \bar{\rho} w^{(k)}$.

- **A2:** Обчислимо розв'язок x_1 початкової задачі, підставивши обчислене значення u

$$\begin{cases} x_1'(t) = \alpha x(t) + \beta u(t), & t \in (0, L), \\ x_1(0) = x_0, \end{cases}$$

- **A3:** Обчислимо $\psi(u, x_1)$.
- **A4: (Критерій зупинки)**
Якщо правильно $\psi(u, x_1) \geq \psi(u^{(k)}, x^{(k)})$
тоді $\bar{\rho} := b\bar{\rho}$; перейти на **A1**
у противному випадку $\rho_k := \bar{\rho}$; ЗУПИНКА алгоритму.

Значення ρ_k використовується на етапі **S5** алгоритму спуску і на наступному кроці ітерації використовується значення $\bar{\rho}$. У програмі відповідні змінні позначені так:

- $\rho_{k-1} = \text{ro}$;
- $\bar{\rho} = \text{robar}$;
- $u = \text{u}$;
- $x_1 = \text{x1}$;
- $\psi(u^{(k)}, x^{(k)}) = \text{cvold}$;
- $\psi(u, x_1) = \text{cv}$;
- $b = \text{bro}$;
- $\rho_k = \text{ro}$ (актуалізоване значення ρ_k).

Виконання циклу методу Армїїо контролюється оператором `for` стосовно змінної `count` та індикатором зупинки `flag2`. Змінна `flag2` може набувати двох значень, які означають таке:

0 – точність при обчисленні за алгоритмом Армїїо не досягнута;

1 – точність при обчисленні за алгоритмом Армїїо досягнута.

Тоді ця частина програми набуває вигляду:

```

% =====
% PART 5
% =====
% S4 : FIT RO
robar=ro;
flag2=0;
flag3=0;
% start loop to fit ro
for count=1:maxro
    count
    u=uold-robar*w;
    % solve state equation for input u and get state x1
    x1(1)=x0;
    for j=1:n-1
        ind=j;
        x1(j+1)=RK42(x1(j),t(j));
    end
    % test for SC4
    if robar < eps1
        flag3=1;
        break % leave loop for count ...
    end
    % compute current cost value and test barrho
    Q1=a*x1.^2 + u.^2;
    temp=trapz(t,Q1);
    cv=temp/2;
    if cv >=cvold % no decrease of cost for minimization
        robar=bro*robar
    else
        flag2=1;
        break % leave loop for count ...
    end
end % for count

```

Розв'язок задачі (8.4.13), який залежить від керування u , обчислюється аналогічно, як вище, тільки з незначними відмінностями, які створюють труднощі у написанні програми. Керування u ми присвоюємо змінній $uold$. Тому під час написання програми замість RK41 використовуємо RK42. Функція RK42 відрізняється від RK41 тим, що замість F1 використовуємо F2, яка визначена нижче:

```
function yout=F2(t,x)
    global alf bet ind
    global u
    yout=alf*x+bet*u(ind);
end
```

Отже, стосовно циклу зі змінною `count`, підведемо підсумки і з'ясуємо як працює алгоритм. Якщо `flag2=1`, тоді алгоритм Армїїо завершується і ми отримуємо нове значення функціонала корисності (яке присвоюється змінній `cvnew`). Згідно з критерієм **SC4** також завершується цикл, оскільки $\bar{\rho} < \varepsilon_1$ і `flag3=1`. У цьому випадку значення $u^{(k)}$ наближене до оптимального і у зовнішньому циклі `for iter ...` значення `flag3` викликає заупинку виконання програми. На наступному кроці також перевіряємо критерій закінчення програми **SC2** та **SC3**. У протилежному випадку градієнтний алгоритм буде неправильний, оскільки крок ітерації ρ_k не буде обчислений алгоритмом Армїїо. Відповідна частина програми має такий вигляд:

```
% =====
% PART 6
% =====
if flag3 == 1
    disp('CONVERGENCE BY RO')
    break % leave loop for iter ...(SC4 satisfied)
end
if flag2==1
    cvnew=cv
    jj=jj+1;
    cost(jj)=cvnew;
    unew=u;
    ro=robar;
    % verify SC2
    if abs(cvnew-cvold) < eps
        disp('CONVERGENCE by COST')
        flag1=1;
        break % leave the loop for iter ...(SC2 satisfied)
    end
    % verify SC3
    d=uold-unew;
    dif=sqrt(sum(d.^2));
    if dif<eps
```

```

disp('CONVERGENCE by CONTROL')
flag1=1;
break % leave the loop for iter ...(SC3 satisfied)
end
else
error('NO CONVERGENCE FOR RO') % STOP PROGRAM
end

```

І накінець остання частина програми, в якій завершується виконання алгоритму та повернення результатів обчислення у вигляді графіків та обчислених величин:

```

% =====
% PART 7
% =====
% prepare a new iteration
uold=unew;
cvold=cvnew;
% x=x1; to be introduced in the final version below
end % for iter start in PART 2
if (flag1==1)|(flag3==1) % | means logical or
plot(t,unew,'*');grid
xlabel(' \it{\bf {t}}', 'FontSize',16)
ylabel(' \it{\bf {u(t)}}', 'FontSize',16)
figure(2)
plot(t,x1,'r*');grid
xlabel(' \it{\bf {t}}', 'FontSize',16)
ylabel(' \it{\bf {x(t)}}', 'FontSize',16)
else
error('NO CONVERGENCE FOR DESCENT METHOD')
end
grad=grad';
cost=cost';
save grad.txt grad -ascii % save vector grad into
                        % file grad.txt
save cost.txt cost -ascii % save vector cost into
                        % file cost.txt
disp('END OF JOB')

```

Отже, програма завершена. Займемося тепер поліпшенням програми, зробивши її більш швидкодією. Зауважимо, що у циклі методу Арміджо (for count ...) обчислюється значення пари $[u, x1]$ та відповідне значе-

ння функціонала корисності. У випадку, коли досягається точність, тоді змінна `flag2` набуває значення 1 і у підсумку одержуємо послідовність:

```
if flag2==1
    cvnew=cv;
    ...
    unew=u;
    ...
end
```

Якщо зовнішній цикл градієнтного методу (`for iter ...`) не закінчується, оскільки не досягнута точність обчислень, тоді запишемо таку ітерацію:

```
uold=unew;
cvold=cvnew;
```

Нова ітерація починається з обчислення `uold` як розв'язку (8.4.13) (крок **S1**). Очевидно, що ми знаходимо розв'язок `x1` для значення `u`. Звідси випливає, що немає більше необхідності розв'язувати задачу (8.4.13). Приймаючи це до уваги, відповідний проміжок програми змодифікуємо так:

```
for iter=1:maxit
    iter
    if iter==1
        % S1 : solve problem for x by Runge-Kutta method
        ...
        jj=1;
    end
    ...
    uold=unew;
    cvold=cvnew;
    x=x1;
end % for iter
```

У файлі `CONTO0.m` виконаємо дві модифікації, а саме:

- у програму, починаючи від `if iter==1`, впишемо послідовність команд розв'язування задачі (8.4.13);
- впишемо також присвоєння `x=x1`; на останній позиції послідовності команд програми `for iter`

Звернемо увагу, що після такої модифікації програма матиме чотири критерії її завершення (від **SC1** до **SC4**). Під час тестування програми

спочатку виконаємо її з усіма критеріями. Якщо у результаті виконання програми буде досягнута точність, тоді по чергово виключатимемо критерії (чи заново вводити), щоб з'ясувати, який із алгоритмів найліпший.

Перший чисельний тест виконаємо для таких значень: $L = 1$, $\alpha = 2$, $\beta = 0.7$, $a = 3$. Виконання програми розглянемо за таких значень числових параметрів моделі: $h = 0.001$; початкове значення ρ (roinit) дорівнює 1; максимальна кількість ітерацій у методі Армїо (maxro) дорівнює 20; коефіцієнт b (bro) дорівнює 0.55. Точність для зупинки програми $\varepsilon = \varepsilon_1 = 0.001$. Якщо у задачі (8.4.13) приймемо за $x_0 = 0$, то отримаємо для оптимальної пари (u^*, x^*) , що $u^* = 0$, $x^* = 0$ для $t \in [0, L]$. Очевидно, що оптимальне значення також дорівнює нулю. Як вже зазначалося у підрозділі 8.4.1., алгоритм є збіжним до оптимального керування (до оптимальної пари), бо функціонал корисності строго увігнутий. У кожному з цих випадків поведінка програми може бути досить цікавою. Вибравши початкове наближення оптимального керування $u_0 = 10$ на першому кроці ітерації $[u^{(0)}, x^{(0)}]$, буде сильно відрізнятись від оптимальної пари. Програма чисельно збіжна до оптимальної пари. Збіжність отримуємо так:

- **SC1** (критерій корисності), після 15 ітерацій;
- **SC2** (критерій керування), після 31 ітерацій;
- **SC3** (градієнтний критерій), після 34 ітерацій.

Значення функціонала корисності після 34 ітерацій становить 2.984×10^{-10} .

Проведемо такий чисельний тест програми з такими самими значеннями вхідних параметрів, тільки за початкове наближення керування приймемо $u_0 = 25$. Збіжність буде досягнута за різних критеріїв так:

- **SC1** (критерій корисності), після 17 ітерацій;
- **SC2** (критерій керування), після 34 ітерацій;
- **SC3** (градієнтний критерій), після 37 ітерацій.

Наближене значення функціонала корисності після 37 кроків дорівнюватиме 1.92×10^{-10} . Звідси можна зробити такі висновки.

- Для обчислення оптимальної пари потрібна велика кількість ітерацій.
- При різних критеріях збіжності для досягнення відповідної точності обчислення необхідна різна кількість кроків ітерацій.

- Для $u_0 = 25$ треба більше ітерації.
- На рис. 8.23 зображено графік оптимального керування для випадку $u_0 = 25$.

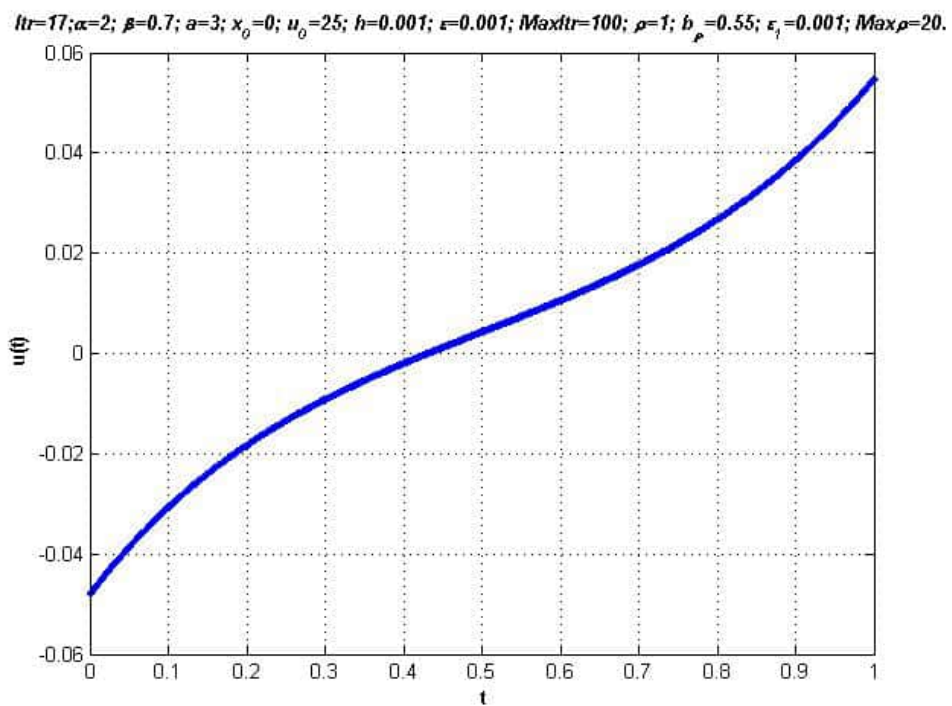


Рис. 8.23. Оптимальне керування у випадку $x_0 = 0$

Щоб дослідити швидкість збіжності до нуля алгоритму, проведемо чисельний експеримент при значеннях $u_0 = 10$ та $x_0 = 0$. Результати є такими:

| iteration | cost |
|-----------|----------|
| 1 | 197.1929 |
| 2 | 82.1698 |
| 3 | 34.2942 |
| 4 | 14.3325 |
| 5 | 6.0014 |
| 6 | 2.5172 |

| | |
|----|--------|
| 7 | 1.0583 |
| 8 | 0.4458 |
| 9 | 0.1883 |
| 10 | 0.0798 |
| 11 | 0.0339 |
| 12 | 0.0144 |
| 13 | 0.0062 |
| 14 | 0.0027 |
| 15 | 0.0011 |
| 16 | 0.0005 |

У наступному чисельному експерименті прийmemo $x(0) = 2$ та $u_0 = 5$. Збіжність алгоритму досягнemo після 15 ітерацій. Графік керування у цьому випадку зображено на рис. 8.24.

itr=15; $\alpha=2$; $\beta=0.7$; $a=3$; $x_0=2$; $u_0=5$; $h=0.001$; $\epsilon=0.001$; $Maxitr=100$; $\rho=1$; $b_p=0.55$; $\epsilon_1=0.001$; $Max\rho=20$.

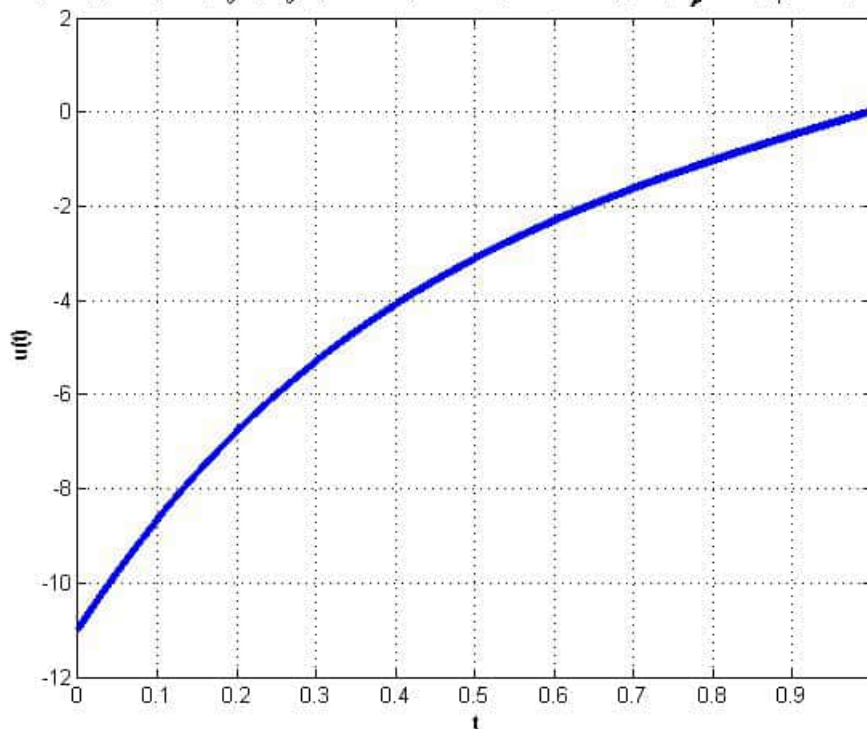


Рис. 8.24. Оптимальне керування у випадку $x_0 = 2$

Зауважимо, що оптимальне керування можна обчислити за допомогою принципу максимуму. Нехай (u^*, x^*) – оптимальна пара і, відповід-

но, спряжений стан p^* . Тому x^* і p^* є розв'язками таких задач:

$$\begin{cases} (x^*)'(t) = \alpha x^*(t) + \beta u^*(t), & t \in (0, L), \\ x^*(0) = x_0, \end{cases} \quad (8.4.19)$$

$$\begin{cases} (p^*)'(t) = -\alpha p^*(t) + a x^*(t), & t \in (0, L), \\ p^*(L) = 0. \end{cases}$$

Крім того, справедлива рівність

$$\beta p^*(t) = u^*(t) \quad \text{для майже всіх } t \in (0, L).$$

Це означає, що u^* можна ототожнити (у $L^2(0, L)$) з абсолютно неперервною функцією βp^* . Отож, підставимо u^* замість βp^*

$$\begin{cases} (u^*)'(t) = -\alpha u^*(t) + a\beta x^*(t), & t \in (0, L), \\ u^*(L) = 0, \end{cases} \quad (8.4.20)$$

звідки одержимо

$$x^* = \frac{1}{a\beta} ((u^*)' + \alpha u^*).$$

Виключивши із (8.4.19) і (8.4.20) x^* , одержимо для u^* задачу

$$\begin{cases} (u^*)'(t) - (\alpha^2 + a\beta^2)u^*(t) = 0, & t \in (0, L), \\ (u^*)'(0) + \alpha u^*(0) = a\beta x_0, \\ u^*(L) = 0. \end{cases}$$

Розв'язок цієї задачі записується у явному вигляді

$$u^*(t) = c_1 e^{rt} + c_2 e^{-rt} \quad \text{для майже всіх } t \in (0, L), \quad (8.4.21)$$

де $r = \sqrt{\alpha^2 + a\beta^2}$ і

$$\begin{cases} c_1 = \frac{a\beta x_0}{(r+a) + (r-\alpha)e^{2rL}}, \\ c_2 = -\frac{a\beta x_0 e^{2rL}}{(r+a) + (r-\alpha)e^{2rL}}. \end{cases}$$

Візьмемо значення параметрів такі ж як у останньому чисельному експерименті. Якщо обчислити графік оптимального керування за формулою (8.4.21) та порівняти його з наближеним розв'язком, то ці обидва графіки накладуться.

8.4.3. Керування запасами

У цьому пункті дослідимо *задачу керування запасами* та знайдемо оптимальну стратегію.

Розглянемо фірму, яка має контракт на постачання продукції у заданому інтервалі часу $[0, T]$ ($T > 0$). Відповідно до контракту, план постачання описується відомою кусково неперервною функцією $g : [0, T] \rightarrow \mathbb{R}^+$. Функція $g(t)$ визначає об'єм продукції, що постачається у момент часу t . Через $u(t)$ позначимо об'єм продукції, що виготовляє фірма у момент часу t і через $x(t)$ її запаси. Зміну запасів можна описати такою задачею:

$$\begin{cases} x'(t) = u(t) - g(t), & t \in (0, T), \\ x(0) = x_0 \in \mathbb{R}. \end{cases} \quad (8.4.22)$$

Об'єм запасів у момент часу t визначається як розв'язок задачі (8.4.22) і записується за допомоги формули

$$x(t) = x_0 + \int_0^t [u(s) - g(s)] ds.$$

- Якщо $x(t) \geq 0$, то в момент часу t постачання не відтермінується.
- Якщо $x(t) > 0$, то фірма збільшує свої запаси.
- Якщо $x(t) < 0$, то відбувається відтермінування постачання у момент часу t . Фірма постачає продукцію обсягом $|x(t)|$.

Уведемо функцію $\psi(x)$, яка описує кошти утримування запасів або штраф за невчасне постачання. Точніше, $\psi(x) =$ ціні утримування запасу у випадку $x \geq 0$ і $\psi(x) =$ величині штрафу за несвоєчасне постачання у випадку $x < 0$. Функція ψ записується у вигляді

$$\psi(x) = \begin{cases} c_1 x^2, & x \geq 0, \\ c_2 x^2, & x < 0, \end{cases} \quad (8.4.23)$$

де $c_1 > 0$ і $c_2 > 0$. Наприклад, якщо прийємо $c_1 = 0.001$ і $c_2 = 0.003$, то графік функції $\psi(x)$ на проміжку $[-10, 10]$ зображено на рис. ???. Значення параметрів означають, що відтермінування у постачанні

є коштовніше за складування. Якщо за $a > 0$ прийняти ціну одиниці продукції, то загальний кошт фірми визначається так:

$$\Phi(u) = \int_0^T [au(t) + \psi(x^u(t))]dt,$$

де x^u є розв'язком задачі (8.4.22). Отже, задача оптимального керування полягає у мінімізації корисності

$$\int_0^T [au(t) + \psi(x^u(t))]dt \rightarrow \min, \quad (\text{PS})$$

стосовно $u \in K = \{w \in L^2(0, T) : u_1 \leq u(t) \leq u_2 \text{ для майже всіх } t \in (0, T)\}$, де $u_1, u_2 \in \mathbb{R}$, $0 \leq u_1 < u_2$. Обмеження на керування означає, що обсяг виробництва продукції обмежений. Крім того, очевидно, що

$$u_1 \leq g(t) \leq u_2, \quad t \in [0, T].$$

Задача (PS) еквівалентна такій:

$$-\Phi(u) \rightarrow \max, \quad (\text{PS}')$$

стосовно $u \in K$.

Для кожної довільної функції $u \in U = L^2(0, T)$ розглянемо p^u (спряжений стан) як розв'язок задачі

$$\begin{cases} p'(t) = -\psi'(x^u(t)), & t \in (0, T), \\ p(T) = 0. \end{cases} \quad (8.4.24)$$

Обчислимо градієнт функціонала корисності. Для довільних $u, v \in U$ та $\varepsilon \in \mathbb{R}^*$ справджується рівність

$$\frac{1}{\varepsilon}[\Phi(u + \varepsilon v) - \Phi(u)] = \int_0^T \frac{1}{\varepsilon}[\psi(x^{u+\varepsilon v}(t)) - \psi(x^u(t))]dt + a \int_0^T v(t)dt. \quad (8.4.25)$$

Нехай z є розв'язком задачі

$$\begin{cases} z'(t) = v(t), & t \in (0, T), \\ z(0) = 0. \end{cases} \quad (8.4.26)$$

У підрозділах 8.3.2. і 8.3.3. доведено, що

$$x^{u+\varepsilon v} \rightarrow x^u \quad \text{у } C([0, T]),$$

а також

$$\frac{x^{u+\varepsilon v} - x^u}{\varepsilon} \rightarrow z \quad \text{у } C([0, T]),$$

при $\varepsilon \rightarrow 0$. Виконавши граничний перехід у (8.4.25), одержимо

$$(v, \Phi_u(u))_{L^2(0, T)} = a \int_0^T v(t) dt + \int_0^T \psi'(x^u(t)) z(t) dt. \quad (8.4.27)$$

Домножимо рівняння (8.4.24) на z та проінтегруємо на проміжку $[0, T]$. Після чого одержимо

$$\int_0^T (p^u)'(t) z(t) dt = - \int_0^T \psi'(x^u(t)) z(t) dt.$$

Проінтегруємо зліва частинами та використавши (8.4.24) і (8.4.26), одержимо

$$\int_0^T \psi'(x^u(t)) z(t) dt = \int_0^T p^u v(t) dt.$$

Тоді (8.4.27) запишеться у вигляді

$$(v, \Phi_u(u))_{L^2(0, T)} = \int_0^T v(t) [p^u(t) + a] dt.$$

На підставі того, що $v \in L^2(0, T)$ довільна функція, з останньої рівності одержимо

$$\Phi_u(u) = p^u + a. \quad (8.4.28)$$

Отож, тепер можемо написати алгоритм методу спуску знаходження розв'язку задачі (PS), використавши $-\Phi_u(u)$ як напрям градієнта за такого керування u . Оскільки на керування накладені умови, то скористаємося методом проєкції градієнта (*алгоритм Uzawa*).

Алгоритм методу ґрадієнта для задачі (PS)

S0: Виберемо нульове наближення $u^{(0)} \in K$.
Приймемо $j := 0$.

S1: Обчислимо розв'язок $x^{(j)}$ задачі (8.4.22), підставивши $u^{(j)}$ замість u :

$$\begin{cases} x'(t) = u^{(j)}(t) - g(t), & t \in (0, T), \\ x(0) = x_0. \end{cases}$$

S2: Обчислимо розв'язок $p^{(j)}$ задачі (8.4.24), до якої замість x підставимо обчислене наближення $x^{(j)}$

$$\begin{cases} p'(t) = -\psi'(x^{(j)}(t)), & t \in (0, T), \\ p(T) = 0. \end{cases}$$

S3: Обчислимо наближення напряму ґрадієнта $w^{(j)}$ за допомогою формули (8.4.28)

$$w^{(j)} := \Phi_u(u^{(j)}) = p^{(j)} + a.$$

S4: (Критерій закінчення алгоритму)

Якщо виконується умова: $\|\Phi_u(u^{(j)})\| < \varepsilon$,
тоді STOP ($u^{(j)}$ є наближенням керування);
якщо умова не виконується, тоді перехід на **S5**.

S5: Обчислити крок ρ_j із умови

$$\Phi(P_K(u^{(j)} - \rho_j w^{(j)})) = \min_{\rho \geq 0} \Phi(P_K(u^{(j)} - \rho w^{(j)})).$$

S6: Обчислити наближення керування $u^{(j+1)}$:

$$u^{(j+1)} := P_K(u^{(j)} - \rho_j w^{(j)});$$

присвоїти $j := j + 1$; перехід на **S1**.

Оператор проєктування $P_K : L^2(0, T) \rightarrow K$ поточково стосовно t , а саме

$$P_K(u)(t) = \text{Proj}(u(t)) \quad \text{для майже всіх } t \in (0, T),$$

де

$$\text{Proj}(w) = \begin{cases} w, & \text{якщо } u_1 \leq w \leq u_2, \\ u_1, & \text{якщо } w < u_1, \\ u_2, & \text{якщо } u_2 < w. \end{cases}$$

Як зазначалось у підрозділі 8.4.2., для зупинки алгоритму можна використати різні критерії (від **SC1** до **SC4**). У згаданих підрозділах концептуально описано алгоритм із використанням **SC2**. У цьому випадку скористаємося критерієм **SC1**, позаяк його можна вписати у різних місцях алгоритму. У програмі будуть використані критерії **SC2** та **SC4**. Сама структура програми повністю аналогічна до відповідної програми, написаної у підрозділі 8.4.2. Розглянемо тільки ті місця програми, якими вони відрізняються. Задекларуємо глобальні змінні: `ind`, `L2`, `c1`, `c2`, `u1`, `u2`, `g1`, `g2`, `g3`, `g4`, `h`, `uold`, `u`, `x`. Тут `L` – кінцевий момент часу `T` і

```
L2=L/2;% the midinterval for function g
```

На наступному кроці впровадимо відповідні дані. Для чисельних експериментів приймемо

$$g(t) = \begin{cases} g_1 t + g_2, & t \in [0, T/2], \\ g_3 - g_4 t & t \in (T/2, T]. \end{cases}$$

Прийнявши

$$g_1 = g_4 = \frac{2}{T}(u_2 - u_1), \quad g_2 = u_1, \quad g_3 = 2u_2 - u_1,$$

функція $g(t)$ задовольняє обмеження $u_1 \leq g(t) \leq u_2$. Крім того, g неперервна на $[0, T]$ та $g(0) = g(T) = u_1$, $g(T/2) = u_2$. Це відповідний план постачання для фірми. У програмі змінна L означає T та, крім того, введемо позначення $L2 = L/2$.

Продовжимо модифікацію програми, використовуючи поділ на частини як у підрозділі 8.4.2.. Використовуючи вище згадану програму та відповідні допоміжні функції, які належить модифікувати, будемо надавати їм змінені змістовні назви, а саме, доповнимо назви літерами **SM**, що означає "Stock Management". Частина **PART 2** програми **CONT0.m** запишемо так:

```
=====
PART 2
=====
%gradient loop starts
for iter=1:maxit
    iter
```

```

if iter==1
    %S1 : solve IVP for x by RK4 method
    x(1)=x0;
    for i=1:n-1
        ind=i;
        x(i+1)=RK41SM(x(i),t(i));
    end
    disp('State Problem solved');
    for i=1:n
        z(i)=f(x(i)); % function f stands for psi
    end
    Q=a*uold+z;
    temp=trapz(t,Q);
    cvold=temp
    cost(1)=cvold;
    jj=1;
end % end if

```

m-файл `RK41SM.m` залишається незмінним, тоді як функцію `F1.m` необхідно змінити, бо рівняння траєкторії x відрізняється від відповідного рівняння з підрозділу 8.4.2. Функція-файл `F1SM.m` записується у вигляді

```

function yout=F1SM(t,x)
    global ind
    global uold
    yout=uold(ind)-g(t);
end

```

Визначимо тепер функцію g у середовищі `MatLab®` за допомогою `g.m` так:

```

function y=g(t)
    global L2
    global g1 g2 g3 g4
    if t<=L2
        y=g1*t+g2;
    else
        y=g3-g4*t;
    end
end

```

Функцію-файл `f.m` для обчислення функції ψ у машинному кодї `MatLab®` визначимо так

```
function y=f(x)
global c1 c2
temp=x*x;
if x>=0
    y=c1*temp;
else
    y=c2*temp;
end
```

PART 3 програми залишається без змін за винятком формули для визначення градієнта. Відповідне місце у програмі запишемо так:

```
% S3: compute the gradient
w=p+a;
disp('GRADIENT COMPUTED');
```

Спряжене рівняння також відрізняється від відповідного рівняння, використаного у підрозділі 8.4.2., тому функція F3SM.m із RK43SM.m, набуде вигляду:

```
function yout=F3SM(t,p)
global ind
global x
yout=-fder(x(ind));
end
```

Тепер треба визначити функцію fder.m, яка є похідною ψ згідно з формулою (8.4.23)

```
function y=derf(x)
global c1 c2
if x>=0
    y=2*c1*x;
else
    y=2*c2*x;
end
end
```

PART 4 програми залишається без змін (застосовується тест **SC1**). Позаяк для мінімізації функціонала використовується метод проекції градієнта, тому треба внести зміни у PART 5. Обчислення проміжного значення керування виконуємо за допомогою процедури

```

for count=1:maxro
    count
    for i=1:n
        temp=uold(i)-robar*w(i);
        u(i)=Proj(temp);
    end
end

```

Функцію Proj.m, за допомогою якої обчислюється оператор проектування, визначимо так:

```

function y=Proj(u)
    global u1 u2
    y=u;
    if u<u1
        y=u1;
    end
    if u>u2
        y=u2;
    end
end
end

```

Функція RK42SM.m така ж як і RK42.m, за винятком F2, яку необхідно замінити на F2SM (формула обчислення Runge-Kutta та сама), де F2SM.m визначається так:

```

function yout=F2SM(t,x)
    global ind
    global u
    yout=u(ind)-g(t);
end

```

Ще одна зміна у PART 5 відповідає обчисленню вектора траєкторії x_1 , який використовується при визначенні вектора керування u та відповідного значення функції корисності. Ці зміни мають вигляд:

```

for j=1:n
    z1(j)=f(x1(j));
end
Q1=a*u+z1;
temp=trapz(t,Q1);
cv=temp;

```

Отож, після зазначених модифікацій та очевидних змін у програмі CONT0, програма StockManagement набуде вигляду

```
% file StockManagement.m
% gradient algorithm with Armijo method for the steplength
rho
% =====
% PART 1
% =====
clear
global ind uold
global L2 L
global c1 c2 u1 u2
global g1 g2 g3 g4
global a h u x
load DataSM.txt;
L=DataSM(1);
L2=L/2; % the midinterval for function g
a=DataSM(2);
x0=DataSM(3);
u1=DataSM(4);
u2=DataSM(5);
u0=DataSM(6);
c1=DataSM(7);
c2=DataSM(8);
g1=DataSM(9);
g2=DataSM(10);
g3=DataSM(11);
g4=DataSM(12);
h=DataSM(13);
t=0:h:L; % time grid
n=length(t);
x=zeros(n,1); % state vector
x1=zeros(n,1); % state vector to be used for rho loop
                % (Armijo method)
p=zeros(n,1); % adjoint state vector
uold=zeros(n,1); % control vector corresponding to  $\hat{u}(k)$ 
unew=zeros(n,1); % control vector corresponding to  $\hat{u}(k+1)$ 
u=zeros(n,1); % control vector to be used for rho loop
                % (Armijo method)
% S0: Control initialization
for i=1:n
    uold(i)=u0;
end
```

```

eps=DataSM(14);% precision epsilon for the gradient algorithm
maxit=DataSM(15);% max. no. iterations - gradient algorithm
disp('RO data');
roin=DataSM(16);% initial value for gradient steplength rho
bro=DataSM(17);% b parameter for rho loop
eps1=DataSM(18);% precision epsilon_1 for steplength rho
maxro=DataSM(19);% max. no. of iterations - rho loop
ro=roin; % initialization of steplength rho
flag1=0; % convergence indicator for gradient algorithm
ii=0; % index for gradient values
hold on
% =====
% PART 2
% =====
% gradient loop starts
for iter=1:maxit
    iter;
    if iter==1
% S1: solve the state equation for x(t) by Runge-Kutta method
x(1)=x0;
for i=1:n-1
    ind=i; % ind is a global index variable used
            % by the rhsHarv of system for x(t)
    x(i+1)=RK41SM(x(i),t(i));
end
disp('State Problem for X solved');
z=zeros(n,1);
for i=1:n
    z(i)=f(x(i));%function f stands for psi
end
Q=a*uold+z;
temp=trapz(t,Q);
cvold=temp;
cost(1)=cvold; % store the value of the cost functional
jj=1;
disp('End Step One!');
end
% =====
% PART 3
% =====
% S2 : solve adjoint equation for p(t):

```

```
p(n)=0;
for j=1:n-1
    i=n-j;
    ip1=i+1;
    ind=ip1;
    p(i)=RK43SM(p(ip1),t(ip1));
end
disp('AE solved');
% S3: compute the gradient
w=p+a;
disp('GRADIENT COMPUTED');
% =====
% PART 4
% =====
normg=sqrt(sum(w.^2)); % compute the gradient norm
ii=ii+1;
grad(ii)=normg;
% verify SC1
if normg < eps
    dspCGr=['CONVERGENCE by GRADIENT', ' u0=', num2str(u0), ...
           ' ; u1=', num2str(u1), ' ; u2=', num2str(u2), ...
           ' ; x0=', num2str(x0), ' ; a=', num2str(a), ...
           ' ; c1=', num2str(c1), ' ; c2=', num2str(c2), ...
           ' ; ro=', num2str(roin), ' ; b=', num2str(bro), ...
           ' ; h=', num2str(h), ' ; eps=', num2str(eps), ...
           ' ; eps1=', num2str(eps1)];
    disp(dspCGr)
    flag1=1;
    break
end
% =====
% PART 5
% =====
% S4 : FIT RO
robar=ro;
flag2=0;
flag3=0;
% start loop to fit ro
for count=1:maxro
    count;
```

```

for i=1:n
    temp=uold(i)-robar*w(i);
    u(i)=Proj(temp);
end
% solve state equation for input u and get state x1
x1(1)=x0;
for j=1:n-1
    ind=j;
    x1(j+1)=RK42SM(x1(j),t(j));
end
plot(t,u,'g-');grid
title('\bf {u}', 'FontSize', 16)
dispRo=['Robar' (num2str(count)) '=' num2str(robar)];
disp(dispRo)
% test for SC4
if robar<eps1
    flag3=1;
    break % leave loop for count ...
end
% compute current cost value and test barrho
z1=zeros(n,1);
for j=1:n
    z1(j)=f(x1(j));
end
Q1=a*u+z1;
temp=trapz(t,Q1);
cv=temp;
if cv >=cvold % no decrease of cost for minimization
    robar=bro*robar;
else
    flag2=1;
    break % leave loop for count ...
end
end % for count
% =====
% PART 6
% =====
if flag3==1
    dspCRo=['CONVERGENCE BY RO:', ' u0=', num2str(u0), ...
           '; u1=', num2str(u1), '; u2=', num2str(u2), ...

```

```

        '; x0=',num2str(x0),' ; a=',num2str(a),...
        '; c1=',num2str(c1),' ; c2=',num2str(c2),...
        '; ro=',num2str(roin),' ; b=',num2str(bro),...
        '; h=',num2str(h),' ; eps=',num2str(eps),...
        '; eps1=',num2str(eps1)];
disp(dspCRo)
break % leave loop for iter ...(SC4 satisfied)
end
if flag2==1
cvnew=cv;
jj=jj+1;
cost(jj)=cvnew;
unew=u;
ro=robar;
% verify SC2
if abs(cvnew-cvold)<eps
dspCC=['CONVERGENCE by COST:', ' u0=',num2str(u0),...
        '; u1=',num2str(u1),' ; u2=',num2str(u2),...
        '; x0=',num2str(x0),' ; a=',num2str(a),...
        '; c1=',num2str(c1),' ; c2=',num2str(c2),...
        '; ro=',num2str(roin),' ; b=',num2str(bro),...
        '; h=',num2str(h),' ; eps=',num2str(eps),
        '; eps1=',num2str(eps1)];
disp(dspCC)
flag1=1;
break % leave the loop for iter ...(SC2 satisfied)
end
% verify SC3
d=uold-unew;
dif=sqrt(sum(d.^2));
if dif<eps
dspCContr=['CONVERGENCE by CONTROL:',...
        ' u0=',num2str(u0),' ; u1=',num2str(u1),...
        '; u2=',num2str(u2),' ; x0=',num2str(x0),...
        '; a=',num2str(a),' ; c1=',num2str(c1),...
        '; c2=',num2str(c2),' ; ro=',num2str(roin),...
        '; b=',num2str(bro),' ; h=',num2str(h),...
        '; eps=',num2str(eps),' ; eps1=',num2str(eps1)];
disp(dspCContr)
flag1=1;

```

```

        break % leave the loop for iter ...(SC3 satisfied)
    end
else
    error('NO CONVERGENCE FOR RO') % STOP PROGRAM
end
% =====
% PART 7
% =====
    % prepare a new iteration
    uold=unew;
    cvold=cvnew;
% x=x1; to be introduced in the final version below
end % for iter start in PART 2
if (flag1==1)|| (flag3==1) % | means logical or
    plot(t,unew,'r. '); grid
    ttltxt=['u_0=',num2str(u0),','; u_1=',num2str(u1),...
           ','; u_2=',num2str(u2),','; x_0=',num2str(x0),...
           ','; a=',num2str(a),','; c_1=',num2str(c1),...
           ','; c_2=',num2str(c2),','; \rho=',num2str(roin),...
           ','; b=' num2str(bro,),' ; h=',num2str(h),...
           ','; \epsilon=',num2str(eps),...
           ','; \epsilon_1=' num2str(eps1)];
    title(ttltxt)
    xlabel('\bf {t}','FontSize',16)
    ylabel('\bf {u(t)}','FontSize',16)
% figure
% plot(t,x1,'r* ');grid
% xlabel('\bf {t}','FontSize',16)
% ylabel('\bf {x(t)}','FontSize',16)
else
    error('NO CONVERGENCE FOR DESCENT METHOD')
end
hold off
grad=grad';
cost=cost';
save gradSM.txt grad -ascii % save vector grad into file
grad.txt
save costSM.txt cost -ascii % save vector cost into file
cost.txt
disp('END OF JOB')

```

Перейдемо до прикладів обчислення.

★ **Приклад 8.4.2.** Прийmemo $L=12$ (місяців), $a=0.1$, $x_0=5$, $u_1=10$, $u_2=16$, $u_0=10.2$. Коефіцієнти $c_1=0.001$, $c_2=0.003$ функції $\psi(x)$, які означають, що затримки у постачанні більш вартісні, ніж додатний баланс запасів. Коефіцієнти функції $g(t)$ мають такі значення: $g_1=g_4=0.3333$, $g_2=10$, $g_3=14$. Значення числових параметрів градієнтного методу такі: $h=0.001$, $\text{eps}=0.001$. У методі Армїю визначення кроку ітерації параметрам присвоїмо значення $\text{roinit}=1$, $b=0.6$, $\text{eps1}=0.001$, $\text{maxro}=20$. Наближення оптимального керування зображено на рис. 8.25. Відповідне значення функціонала корисності дорівнює $1.2445243e+01$.

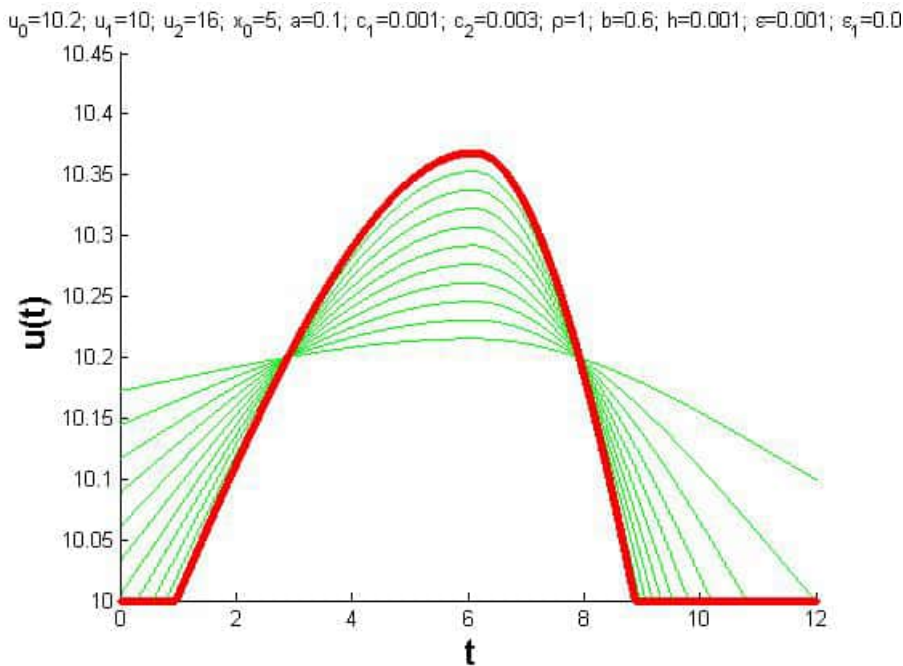


Рис. 8.25. Оптимальне керування для даних із Прикладу 8.4.2

Кілька зауважень щодо програми та отриманих результатів. Вхідні дані задачі, як легко бачити з програми, записані у текстовий файл DataSM.txt у вигляді

```
12    % L
.1    % a
5     % x0
10    % u1
16    % u2
```

```

10.2   % u0
.001   % c1
.003   % c2
.33333 % g1
10     % g2
14     % g3
.33333 % g4
.001   % h
.001   % eps
30     % MaxItr
1      % ro
.6     % b
.001   % eps1
20     % MaxRo

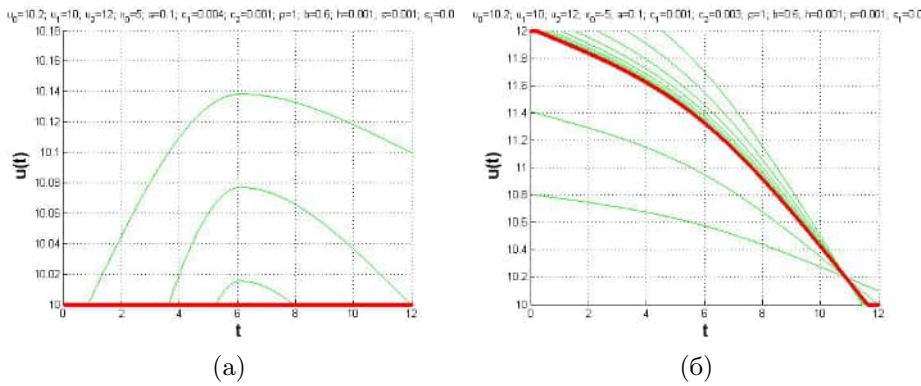
```

У наступних обчисленнях із іншими даними треба внести відповідну зміну значення параметра у файлі `DataSM.txt`, записати і запустити на виконання програму `StockManagement`. По закінченні обчислень отримуємо повідомлення про збіжність ітераційного процесу на підставі критерію точності обчислень функціонала корисності та значення основних параметрів задачі. Отриманий графік містить графічне зображення послідовних наближень і наближення з заданою точністю оптимального керування. Для зручності перегляду над графіком замість заголовка поміщені значення параметрів обчисленої задачі. Послідовність наближень значення функціонала корисності та його градієнта записано у файли `costSM.txt` та `gradSM.txt`. Якщо є необхідність побудови графіка траєкторії, то необхідно у кінці програми закоментувати команди обчислення графіка керування та розкоментувати послідовність команд побудови графіка відповідної траєкторії.

★ **Приклад 8.4.3.** Виберемо значення числових параметрів такі самі як у Прикладі 8.4.2, за винятком коефіцієнтів функції $\psi(x)$, яким присвоїмо значення $c1=0.004$ та $c2=0.001$. Це означає, що додатний запас вигідніший від затримки у постачанні. У цьому випадку графік наближення оптимального керування зображено на рис. 8.26а та відповідне значення функції корисності дорівнює $1.2676467e+01$.

★ **Приклад 8.4.4.** Значення параметрів такі самі, як у Прикладі 8.4.2, змінимо тільки початкове положення траєкторії, взявши $x(0) = -5$. Результат обчислення зображено на рис. 8.26б та відповідне значення функції корисності дорівнює $1.6190569e+01$.

Накінець відзначимо, що від'ємні початкові запаси ($x(0) = -5$ для $t = 0$) спричиняють підвищення рівня виробництва (порівняти значення

Рис. 8.26. Графіки до: *a* – Прикладу 8.4.3; *б* – Прикладу 8.4.4

стосовно осі oY) і зменшення його у першій частині інтервалу порівняно з Прикладом 8.4.2, у якому $x(0) = 5$.

8.4.4. Задача оптимального розмноження

Розглянемо просту задачу розмноження ([24, розділи 8.2.3, 12.4]). А саме, нехай $x(t)$ для $t \in [0, L]$ означає відновлення популяції (наприклад, популяції риби, пасовища, лісу тощо). Наше завдання полягає в *оптимальному керуванні розмноженням*. Розглянемо рівняння

$$x'(t) = F(x(t)) - h(t), \quad t \in (0, L), \quad (8.4.29)$$

де F – закон росту; $h(t)$ – швидкість розмноження. Розглянемо вигляд функції зростання F , запропоновану Verhulst

$$F(x) = rx \left(1 - \frac{x}{k}\right). \quad (8.4.30)$$

Параметр $r > 0$ означає швидкість зростання, $k > 0$ – продуктивність середовища (див. у розділі 8.1. аналогічну модель). Розглянемо

$$h(t) = u(t)x(t),$$

де u , інтенсивність риболовлі, пропорційна до рівня популяції. Наша мета – максимізувати економічну ренту, яка визначається як різниця між доходами та витратами. Візьмемо $revenue = ah$, де $a > 0$ ціна одиниці продукції, $cost = cu$, де $c > 0$ – стала величина: витрати пропорційні інтенсивності риболовлі. Тоді задача керування полягає у

максимізації функціонала

$$\Phi(u) = \int_0^L e^{-\delta t} (au x^u - cu) dt \quad (\text{PH})$$

стосовно

$$u \in K = \{w \in L^2(0, L); 0 \leq w(t) \leq \bar{u} \text{ для майже всіх } t \in (0, L)\}, \quad \bar{u} > 0,$$

де x^u є розв'язок задачі стану

$$\begin{cases} x'(t) = F(x(t)) - u(t)x(t), & t \in (0, L), \\ x(0) = x_0 > 0. \end{cases} \quad (8.4.31)$$

Обмеження на керування можна інтерпретувати як обмеження вилову риби. Крім того, $\delta > 0$ означає дисконтну ставку. Очевидно, що розв'язок задачі (8.4.31) додатний. Нехай (u^*, x^*) – оптимальна пара (існування та єдиність доводиться аналогічно як у розділі 8.3.2. і 8.3.3.). Розглянемо довільні фіксовані $u \in K$ та $v \in V = \{w \in L^2(0, T) : u + \varepsilon w \in K \forall \varepsilon > 0 \text{ достатньо малого}\}$. Обчислимо градієнт $\Phi_u(u)$. Функція $z = z^u$ є розв'язок задачі

$$\begin{cases} z'(t) = F'(x^u(t))z(t) - u(t)z(t) - v(t)x^u(t), & t \in (0, L), \\ z(0) = 0. \end{cases} \quad (8.4.32)$$

Запишемо $\varepsilon^{-1}[\Phi(u + \varepsilon v) - \Phi(u)]$, та зробивши у цьому виразі граничний перехід при $\varepsilon \rightarrow 0$, одержимо

$$(v, \Phi_u(u)) = \int_0^L e^{-\delta t} (au(t)z(t) + ax^u(t)v(t) - cv(t)) dt. \quad (8.4.33)$$

Уведемо спряжений стан системи $p = p^u$ як розв'язок спряженої задачі

$$\begin{cases} p'(t) = -e^{-\delta t} au(t) - [F'(x(t)) - u(t)]p(t), & t \in (0, L), \\ p(L) = 0. \end{cases} \quad (8.4.34)$$

Помножимо диференціальне рівняння (8.4.32) на p^u та проінтегруємо частинами на проміжку $[0, L]$. Після цього помножимо диференціальне

рівняння (8.4.34) на z , проінтегруємо на $[0, L]$ та порівнявши ці формули, одержимо

$$\int_0^L e^{-\delta t} auz dt = - \int_0^L x^u p^u v dt.$$

Підставивши у (8.4.33), отримаємо

$$\Phi_u(u) = e^{-\delta t}(ax^u - c) - x^u p^u. \quad (8.4.35)$$

Крім того, оптимальне керування u^* задовольняє

$$\Phi_u(u^*) = e^{-\delta t}(ax^{u^*} - c) - x^{u^*} p^{u^*} \in N_K(u^*). \quad (8.4.36)$$

Отож (уведемо позначення $x^* = x^{u^*}$, $p^* = p^{u^*}$):

- якщо $e^{-\delta t}(ax^* - c) - x^* p^* > 0$, тоді $u^*(t) = \bar{u}$.
- якщо $e^{-\delta t}(ax^* - c) - x^* p^* < 0$, тоді $u^*(t) = 0$.

Проаналізуємо, що буде із $u^*(t)$, якщо t належить множині $B = \{t : e^{-\delta t}(ax^* - c) - x^* p^* = 0 \text{ м.в.}\}$. Із означення B визначимо p^* , продиференціюємо та використаємо (8.4.34), після чого одержимо рівняння, розв'язком якого є x^*

$$F'(x) + \frac{cF(x)}{x(ax - c)} = \delta. \quad (8.4.37)$$

Підставимо (8.4.30) у (8.4.37), одержимо рівняння другого порядку стосовно x^* , а саме

$$\alpha x^2 + \beta x + \gamma = 0,$$

де

$$\alpha = \frac{2ar}{k} > 0, \quad \beta = a(\delta - r) - \frac{cr}{k}, \quad \gamma = -c\delta < 0.$$

Легко бачити, що квадратне рівняння, записане вище, має два корені, один з яких додатний, інший – від'ємний. Позначимо через \tilde{x} додатний із них. Звідки випливає $x^*(t) = \tilde{x}$ на B та з (8.4.31) одержимо відповідне керування $\tilde{u} = F(\tilde{x})/\tilde{x}$. Отже, оптимальне керування може набувати тільки значення $\{0, \tilde{u}, \bar{u}\}$. Крім того, з умов на коефіцієнти випливає, що $0 < \tilde{u} < \bar{u}$. Неважко переконатися, що $0 < \tilde{x} < k$. Позаяк $F(\tilde{x}) > 0$, тому $\tilde{u} > 0$. З іншого боку, $\tilde{u} < \bar{u}$ означає, що $F(\tilde{x})/\tilde{x} < \bar{u}$, звідки легко одержимо необхідну умову $\bar{u} > r$.

У праці [24] запропоновано як "оптимальне" керування у вигляді

$$u(t) = \begin{cases} \bar{u}, & x^*(t) > \tilde{x}, \\ \tilde{u}, & x^*(t) = \tilde{x}, \\ 0, & x^*(t) < \tilde{x}. \end{cases} \quad (8.4.38)$$

Керування, визначене за формулою (8.4.38), загалом не є оптимальним. У цьому можна переконатися, провівши чисельні тести. Це керування має властивість збільшувати чисельність популяції $x(t)$ до рівня \tilde{x} . Запишемо програму у машинному коді середовища MatLab[®] для того, щоб показати цю властивість керування і після того обчислимо відповідне значення функціонала корисності.

```
% file harv1.m
% a harvesting problem
clear
global r k
global util ubar xtil eps1
% Introduce below input statements for
% L, y0 (for x(0)), r, k, a, c, d (for delta), ubar (for
baru),...
% eps1 (for epsilon_1 ), h.
% compute xtil=tildex and util=tildeu
alf=2*a*r/k; % this is alpha
b=a*(d-r)-c*r/k; % this is beta
cd=-c*d; % this is gamma
w=[alf,b,cd] % this is the polynomial
x=roots(w)
if x(1) > 0
    xtil=x(1);
else
    xtil=x(2);
end
xtil
util=FHarv(xtil) / xtil
% end of sequence for util and xtil
tspan=0:h:L;
[t y]=ode45('rhsHarv',tspan,y0);
plot(t,y,'r*'); grid on
xlabel(' \bf {t}', 'FontSize',16)
ylabel(' \bf {x(t)}', 'FontSize',16)
n=length(t);
u=zeros(n,1);
w=zeros(n,1);
% compute the control tildeu
for i=1:n
    if y(i)>=xtil+eps1
```

```

    u(i)=ubar;
end
if abs(y(i)-xtil)<eps1
    u(i)=util;
end
end
figure(2)
plot(t,u,'*'); grid
xlabel(' \bf {t}', 'FontSize',16)
ylabel(' \bf {u(t)}', 'FontSize',16)
for i=1:n
    w(i)=exp(-d*t(i))*(a*y(i)-c)*u(i);
end
cost=trapz(t,w)

```

Визначимо функцію rhsHarv.m правої частини рівняння стану

```

function out1=rhsHarv(t,y)
global util ubar xtil eps1
temp=0 ;
if abs(y-xtil)<eps1
    temp=util;
end
if y>=xtil+eps1
    temp=ubar;
end
out1=FHarv(y)-temp*y;
end

```

Тут і у скрипт-файлі ми змушені керувати обчисленнями цієї задачі. У задачі використовується формула (8.4.38). Перекласти математичний тест $x(t) = \tilde{x}$ на $y == xtil$ у програмі немає сенсу, оскільки арифметичні обчислення мають помилку заокруглення. Нерівність $|x(t) - \tilde{x}| < \varepsilon_1$ є правильним числовим тестом, де $\varepsilon_1 > 0$ означає точність обчислення (змінна eps1 у програмі). Це означає, що формулу (8.4.38) потрібно замінити такою:

$$u(t) = \begin{cases} \bar{u}, & x(t) \geq \tilde{x} + \varepsilon_1, \\ \tilde{u}, & x(t) \in (\tilde{x} - \varepsilon_1, \tilde{x} + \varepsilon_1), \\ 0, & x(t) \leq \tilde{x} - \varepsilon_1. \end{cases} \quad (8.4.39)$$

Файл-функція FHarv.m визначає функцію, що задається формулою (8.4.30)

```
function y=FHarv(x)
global r k
y=r*x*(1-x/k);
end
```

Виконаємо чисельне тестування моделі. Прийmemo $L=1$, $r=0.3$, $k=5$, $a=1$, $c=1$, $\text{delta}=0.5$, $\text{eps}=0.001$. Одержимо значення $\tilde{x} = 1.5396$, $\tilde{u} = F(\tilde{x})/\tilde{x} = 0.2076$. У наслідок чого отримуємо $\bar{u} = 0.5$. Стан популяції, що відповідає початковим даним $x(0) = 1.4$ та $x(0) = 1.6$ зображено на рис. 8.27. Зауважимо, що у обидвох випадках траєкторії "збігаються" до значення \bar{x} . Відповідні керування зображені на рис. 8.28.

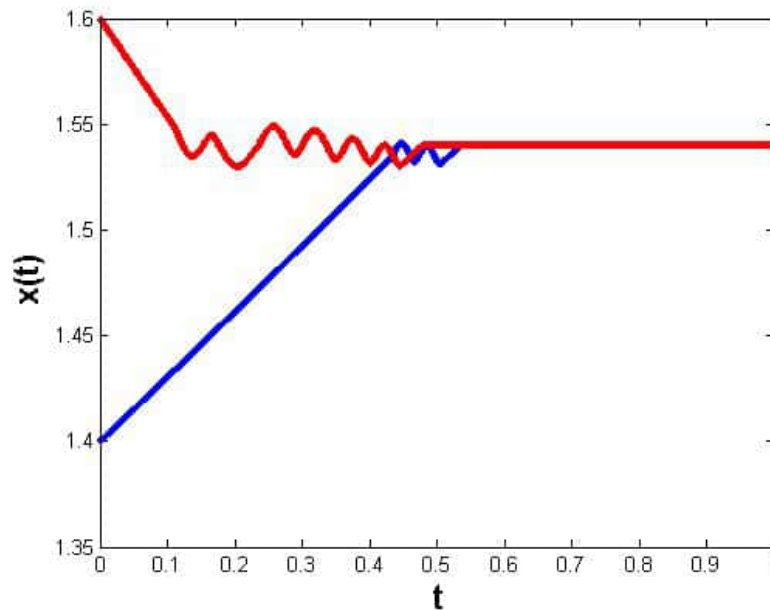


Рис. 8.27. Динаміка популяції програми *harv1*

Звернемо увагу на "несталий" характер параметра ε_1 (*eps1*). Це означає, що значення параметра треба задавати з великою точністю. Наприклад, для малого значення обчислення довший період будуть нестійкі. Зрозуміло, що не можна вибирати досить великими значення ε_1 . У цьому випадку значення $\varepsilon_1 = 0.001$ є властивим. Нижче цю проблему розглянемо детальніше на чисельному прикладі.

Повернемося до чисельної апроксимації оптимального значення функціонала корисності. Оскільки досліджується максимум задачі ке-

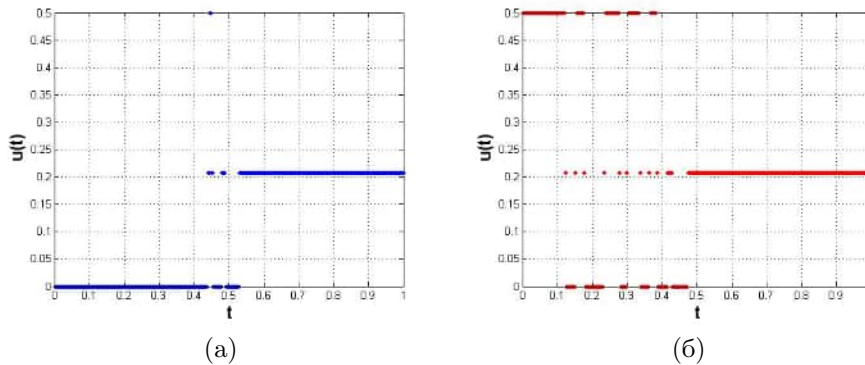


Рис. 8.28. Оптимальне керування програми *harv1* при різних початкових даних: *a* – випадок $x(0) = 1.4$; *б* – випадок $x(0) = 1.6$

рування, то з цією метою використаємо напрям градієнта $\Phi_u(u^{(j)})$ для керування $u^{(j)}$. Алгоритм такий. Використаємо критерій зупинки **SC1**, аналогічно як у попередньому розділі. Пізніше доведемо, що у цьому випадку градієнтний метод дещо ускладнюється.

Алгоритм методу проєкції градієнта задачі (РН)

S0: Вибрати $u^{(0)} \in K$.

Прийняти $j := 0$.

S1: Обчислити розв'язок $x^{(j)}$ задачі (8.4.31), підставивши $u^{(j)}$

$$\begin{cases} x'(t) = F(x(t)) - u^{(j)}(t)x(t), & t \in (0, L), \\ x(0) = x_0. \end{cases}$$

S2: Обчислити розв'язок $p^{(j)}$ спряженої задачі (8.4.34), підставивши $x^{(j)}$

$$\begin{cases} p'(t) = -e^{-\delta t} a u^{(j)}(t) - [F'(x^{(j)}(t)) - u^{(j)}(t)]p(t), & t \in (0, L), \\ p(L) = 0. \end{cases}$$

S3: Обчислити напрям градієнта $w^{(j)}$ за формулою (8.4.35)

$$w^{(j)} := \Phi_u(u^{(j)}) = e^{-\delta t} (a x^{(j)} - c) - x^{(j)} p^{(j)}.$$

S4: (Критерій зупинки)

Якщо $\|w^{(j)}\| < \varepsilon$

тоді STOP ($u^{(j)}$ – наближення керування)
у протилежному випадку перехід на **S5**.

S5: Обчислити крок ρ_j такий, що

$$\Phi(P_K(u^{(j)} + \rho_j w^{(j)})) = \max_{\rho \geq 0} \{\Phi(P_K(u^{(j)} + \rho w^{(j)}))\}.$$

S6: Обчислити таке наближення керування:

$$u^{(j+1)} := P_K(u^{(j)} + \rho_j w^{(j)});$$

прийmemo $j := j + 1$; перехід на **S1**.

Оператор проєкції Proj визначаємо поточково за t як у попередньому розділі

$$(\text{Proj}(u))(t) = \text{Proj}(u(t)) \quad \text{д.м.в. } t \in [0, L].$$

Крім того, поточковий оператор проєкції $\text{Proj} : \mathbb{R} \mapsto \{0, \tilde{u}, \bar{u}\}$ визначений, як було доведено вище, бо оптимальне керування набуває тільки значення 0 , \bar{u} , \tilde{u} . Тому ми використовуємо мінімум відстані до однієї з трьох точок. Запишемо у простішій формі відповідну функцію-файл Proj2.m (зауважимо, що ми використовуємо позначення util для \tilde{u} та ubar для \bar{u}):

```
function y=Proj2(u)
global ubar util
if u<=0
    y=0;
    return
end
if u>=ubar
    y=ubar;
    return
end
z(1)=u;
z(2)=abs(u-util);
z(3)=abs(u-ubar);
[zm,j]=min(z);
if j==1
    y=0;
    return
end
if j==2
    y=util;
```

```

    return
end
y=ubar;
end

```

Для поліпшення алгоритму, що буде обґрунтовано нижче, ми також використаємо класичний оператор проєктування з \mathbb{R} на $[0, \bar{u}]$. Тому запишемо відповідну функцію ProjHarv.m:

```

function y=ProjHarv(u)
global ubar
y=u;
if u<0
    y=0;
end
if u>ubar
    y=ubar;
end
end

```

Чисельні тестування цієї задачі досить трудомісткі. У самій програмі ми використовуємо такі підпрограми (процедури):

- `harv1.m` для обчислення значення корисності, що відповідає керуванню, обчисленого за формулою (8.4.39);
- `harv2.m` для обчислення оптимальної пари (використовуючи `ProjHarv.m` та критерії закінчення алгоритму **SC1**, **SC2**, **SC3**);
- `harv35b.m` для обчислення оптимальної пари (використовуючи `Proj2.m` та критерії закінчення алгоритму **SC1**, **SC4**);
- `harv35a.m` для обчислення оптимальної пари з використанням початкового наближення оптимального керування як у `harv2.m` (використовуючи `Proj2.m` та критерії закінчення алгоритму **SC1**, **SC4**).

За винятком `harv1.m`, решта три процедури подібні, відмінність між якими описана вище. Опишемо відмінності між `harv2.m` та навчальної програми `CONT0.m` з розділу 8.4.2.. Позаяк у чисельних тестах прийнято $\delta = 0$, програма містить формули алгоритму проєкції градієнта у яких $\delta = 0$.

Почнемо з `PART1`, які містить визначення глобальних змінних (`r`, `k`, `a`, `ubar`, `util`, `ind`, `h`, `uold`, `u`, `x`), визначення величин вхідних параметрів, а також послідовність присвоєнь

```
t=0:h:L;
t=t';
n=length(t);
```

У процедурах `harv35b.m` та `harv35a.m`, які використовують `Proj2.m`, впровадимо послідовність команд із `harv1.m` для обчислення `xtil` та `util`:

```
% compute xtil= $\tilde{x}$  and util= $\tilde{u}$ 
.....
% end of sequence for util and xtil
```

У другій частині `PART2` програми `CONT0.m` програми зміниться тільки обчислення функціонала корисності і ці зміни запишемо у вигляді

```
disp('SE solved');
for i=1:n
    q(i)=uold(i)*(a*x(i)-c);
end
temp=trapz(t,q);
cvold=temp
```

Функція `F1.m`, яку для програми `harv2` перейменуємо на `F1Harv2.m`, і яка викликається процедурою `RK41.m` (перейменуємо на `RK41Harv2.m`), відповідно, запишеться у вигляді

```
function yout=F1Harv2(t,x)
global ind
global uold
yout=FHarv(x)-uold(ind)*x;
end
```

Функція-файл `FHarv.m`, яка визначена формулою (8.4.30), визначена вище. Розглянемо таку частину програми `PART3`. Для того, щоб обчислити розв'язок спряженої системи, визначимо функцію `F3Harv2.m`, яка викликається у процедурі `RK43Harv2.m`, так:

```
function yout=F3Harv2(t,p)
global a
global ind
global uold
global x
yout=(p-a)*uold(ind)-FHarvDer(x(ind))*p;
end
```

Функція-файл `FHarvDer.m` повертає похідну функції $F(x)$, визначену формулою (8.4.30) і яка записується у середовищі `MatLab®` так:

```
function y=FHarvDer(x)
global r k
y=r*(1-2*x/k);
end
```

Інші зміни у PART3 стосуються обчислення градієнта:

```
% S3: compute the gradient
for i=1:n
    w(i)=(a-p(i))*x(i)-c;
end
disp('GRADIENT COMPUTED');
```

Запишемо зміни у PART5. Почнемо з обчислення кроку `steplength`

```
for count=1:maxro
    count
    for i=1:n
        temp=uold(i)+robar*w(i);
        u(i)=ProjHarv(temp); % use Proj2 for harv35a,b.m
    end
end
```

Функція-файл `F2Harv2.m`, яка викликається процедурою `RK42Harv2.m`, у цій програмі набуває вигляду

```
function yout=F2Harv2(t,x)
global ind
global u
yout=FHarv(x)-u(ind)*x;
end
```

Функціонал корисності обчислюється так:

```
for i=1:n
    q1(i)=u(i)*(a*x1(i)-c);
end
temp=trapz(t,q1);
cv=temp;
```

На кінець, обчислення значення корисності зміниться, бо ми маємо справу з задачею максимізації. Для кращого розуміння подамо послідовність команд програми до кінця циклу `count`. Порівняємо її з відповідною послідовністю у програмі `CONT0.m` із розділу 8.4.1. для знаходження мінімуму.

```

if cv<=cvold % no increase of cost for maximization
    robar=bro*robar
else
    flag2=1;
    break
end
end % for count

```

Отже, програма записана повністю у кожній із версій. Перейдемо до чисельних експериментів.

★ **Приклад 8.4.5.** Прийmemo $L=10$, $r=0.3$, $k=5$, $a=1$, $c=1$, $\delta=0$. На підставі обчислення одержимо $x_{til}=3$, $u_{til}=0.12$. Звідси, вибравши $x(0)=x_0=2$, $\bar{u}=1.5$, $u(0)=u_0=1.4$ (запишемо ці дані у DataHarv2.txt), та перейдемо до виконання програми. Програма поверне такі значення.

- `harv1.m` яка використовує формулу (8.4.39). Протестуємо також параметр $\varepsilon_1 = \text{eps1}$. Програма за різних значень параметра поверне такі відповідні значення корисності при $h=0.001$:

| epsilon_1 | cost |
|-----------|--------|
| 10^{-1} | 1.7646 |
| 10^{-2} | 1.7537 |
| 10^{-3} | 1.7665 |
| 10^{-4} | 0.7232 |

Звернемо увагу на результат при $\varepsilon_1 = 10^{-4}$. Це означає, що ε_1 достатньо мале.

- `harv35b.m` (із параметрами $b = 0.6$, $h = 0.001$, $\varepsilon_1 = 10^{-3}$, $\text{maxro}=20$) після 4 ітерацій (враховуючи модифікацію ρ) отримано значення корисності 1.9724 та керування, графік якого зображено на рис. 8.30. Зі зменшенням точності обчислення $\varepsilon = 10^{-4}$ значення корисності не змінюється.
- `harv2.m` із класичним методом проєкції після 72 ітерації отримано значення корисності 2.5653458, де використано критерій точності **SC2**. Графік відповідного керування зображений на рис. 8.29.

Зауважимо, що керування $u(t)$ набуває у багатьох точках значення 0, $\tilde{u} = 0.12$, та \bar{u} . На кінець, запишемо обчислені значення оптимального керування у файл UHarv2.txt за допомогою команд

```
save UHarv2.txt unew -ascii
```

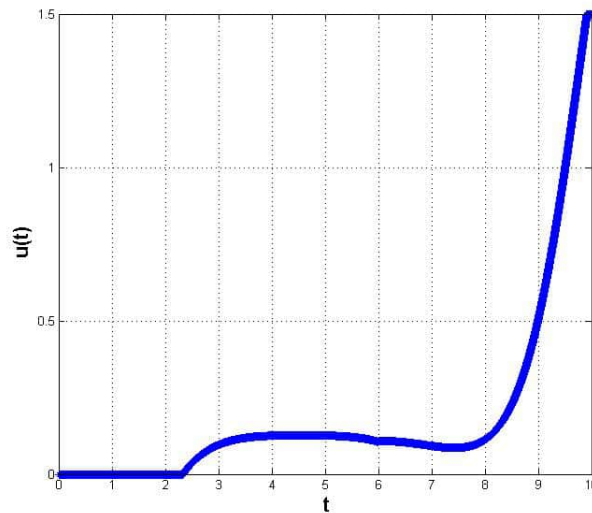


Рис. 8.29. Керування, одержане за допомогою `harv2.m` із використанням ProjHarv

- `harv35a.m` починає обчислення, використовуючи керування, отримане за допомогою процедури `harv2.m`. Відповідна послідовність команд у програмі має вигляд

```
load UHarv2.txt
for i=1:n
    uold(i)=UHarv2(i);
end
```

Після 23 ітерацій (включно з циклом для обчислення `ro`) отримаємо поліпшене значення функціонала корисності, а саме 2.5781. Графік відповідного керування зображений на рис. 8.31.

На кінець, запишемо чисельні результати значень функціонала корисності у звідну таблицю. Цікавим є те, що ці значення утворюють зростаючу послідовність. Отримаємо

| program | cost |
|-----------|--------|
| harv1.m | 1.7665 |
| harv35b.m | 1.9724 |
| harv2.m | 2.5653 |
| harv35a.m | 2.5781 |

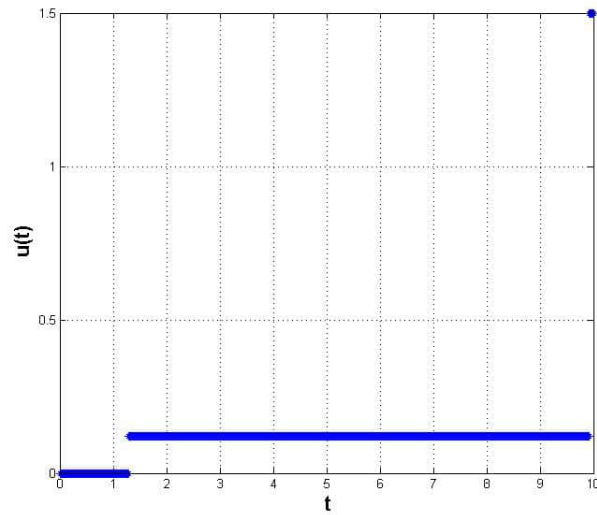


Рис. 8.30. Керування, одержане за допомогою `harv35b.m` із використанням Proj2

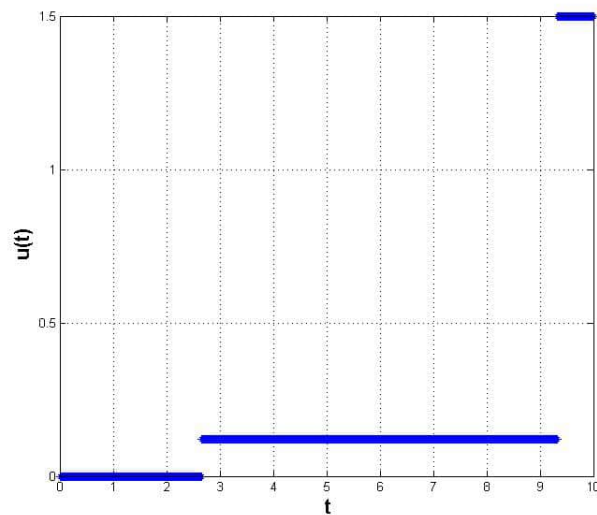


Рис. 8.31. Керування, одержане за допомогою `harv35a.m` із використанням Proj2

★ *Приклад 8.4.6.* Вхідні дані виберемо як у прикладі 8.4.5, за винятком початкових даних $u(0) = 0.12$, $x(0) = 3$. Виконавши чисельні обчислення за допомогою програм як у прикладі 8.4.5, одержимо такі результати:

| program | cost |
|-----------|--------|
| harv1.m | 0.7232 |
| harv35b.m | 2.8989 |
| harv2.m | 3.2142 |
| harv35a.m | 3.2266 |

★ *Приклад 8.4.7.* Вхідні дані виберемо як у прикладі 8.4.5, за винятком початкових даних $u(0) = 0.12$, $x(0) = 2$, проміжком спостереження $L=5$ та точністю обчислень $\varepsilon = 10^{-3}$. За допомогою тих самих програм одержимо такі значення:

| program | cost |
|-----------|--------|
| harv1.m | 0.5489 |
| harv35b.m | 1.3534 |
| harv2.m | 1.3726 |
| harv35a.m | 1.3789 |

8.4.5. Завдання для самостійного опрацювання

◆ *Завдання 8.4.1.* Розглянемо задачу оптимального керування:

$$\frac{1}{2} \int_0^1 (u^2(t) + x^2(t)) dt \rightarrow \min,$$

стосовно $u \in L^2(0, 1)$, де $x = x^u(t)$ є розв'язком початкової задачі

$$\begin{cases} x'(t) = u(t), & t \in (0, 1), \\ x(0) = 1. \end{cases}$$

Записати умови оптимальності (а також градієнт $\Phi_u(u)$) та обчислити оптимальне керування.

✓ *Вказівка.* Задача є частковим випадком задачі, розглянутої у розділі 8.4.2.. Нехай p^u є розв'язком спряженої задачі

$$\begin{cases} p'(t) = -x^u(t), & t \in (0, 1), \\ p(1) = 0. \end{cases}$$

Градiєнт записується у вигляді

$$\Phi_u(u) = u + p^u.$$

Оптимальне керування набуває вигляду

$$u^*(t) = -\frac{\text{sh}(1-t)}{\text{ch}(1)}.$$

Відповідна оптимальна траєкторія $x^* = x^{u^*}$:

$$x^*(t) = \frac{\text{ch}(1-t)}{\text{ch}(1)}.$$

Написати програму знаходження наближеного значення оптимальної пари задачi та порiвняти отриманi числовi результати з точним розв'язком.

◆ *Завдання 8.4.2.* Модифікувати функцію корисності задачi (P1') з розділу 8.4.1., додавши до неї вираз $\varphi(x^u(T))$. Вивести необхідні умови та написати градієнтний алгоритм обчислення наближеного розв'язку задачi аналогічний як у розділі 8.4.1..

◆ *Завдання 8.4.3.* Дослідити задачу оптимального керування

$$\int_0^T u(t)x^u(t)dt - c \int_0^T u^2(t)dt \longrightarrow \max_{u \in L^2(0,T)},$$

де $0 \leq u(t) \leq M$ ($M > 0$) м.в. $\forall t \in (0, L)$, x^u – розв'язок логістичної моделі динаміки популяції

$$\begin{cases} x'(t) = rx(t) - kx^2(t) - u(t)x(t), & t \in (0, T), \\ x(0) = x_0 > 0. \end{cases}$$

Тут c, r, k, x_0 деякі додатні сталі. Обчислити градієнт функції корисності і написати програму, використовуючи градієнтний метод.

✓ *Вказівка.* Див. Завдання 8.3.5 та взяти до уваги обмеження на керування (які необхідні для методу проєкції градієнта).

◆ *Завдання 8.4.4.* Задача (PS) – задача мінімізації, тоді як (PS') є задачею на знаходження максимуму. Трансформувати програму `StockManagement.m` із розділу 8.4.3., написану для задачi (PS), до програми знаходження числового розв'язку задачi (PS'). Провести такі самі числові тести і висновки, якщо будуть одержані протилежні величини для функціонала корисності.

✓ *Вказівка.* Для актуалізації величини корисності (метод Арміїо) взяти до уваги програму `harv2` із розділу 8.4.4., яка написана для задачі максимізації.

◆ *Завдання 8.4.5.* Дослідити задачі оптимального керування з Прикладу 8.3.3 (ВІЛ терапія). Для кожної з них обчислити градієнт функціонала корисності та написати програму, використавши градієнтний метод.

◆ *Завдання 8.4.6.* Дослідити задачі оптимального керування з Прикладу 8.3.4 (Керування у SIR моделі). Для кожної з них обчислити градієнт функціонала корисності та написати програму, використавши градієнтний метод.

8.5. Оптимальне розмноження структурованою за віком популяції

Цей розділ слугує своєрідним містком між науковими дослідженнями та теорією оптимального керування. Задачі, які досліджуються у цьому розділі, складніші від тих, які розглядалися у попередніх розділах. Сконцентруємо свою увагу на дослідженні задач оптимального керування поширення віковоструктурованої популяції, які дуже важливі з біологічного й економічного погляду. Варто зазначити, що рівень складності досліджуваних задач на порядок вищий від задач, розглянутих у розділі 8.3..

Вік є важливим параметром у динаміці популяції. Цей розділ присвячено дослідженню важливої моделі вікової залежності, що описується динамікою популяції окремого виду. Досліджено основні властивості розв'язків моделі. Отримано чисельний розв'язок моделі у середовищі MatLab[®]. Розглянуто також задачу оптимального керування приросту популяції, що описує лінійну динаміку віковозалежної популяції. Оптимальне керування приростом апроксимовано у середовищі MatLab[®]. Досліджується також оптимальне керування динамікою віковозалежної популяції з періодичним коефіцієнтом смертності та логістичним параметром.

Задачі оптимальної народжуваності для віковозалежної динаміки популяції з заданою початковою щільністю присвячена значна кількість наукових робіт. Найважливіші результати досліджені й отримані у [40, 23, 19, 12, 13, 45, 37]. Аналіз та керування віково залежною динамікою популяції у випадку багатовимірного середовища досліджені у працях [32, 49].

Однак небагато праць присвячено важливому випадковій періодичній віково структурованій динаміці популяції. Відмітимо праці [12, 13, 54] у випадку лінійної моделі та [6] у випадку нелінійної.

Чисельні розв'язки задачі оптимального розмноження можна знайти у працях [3, 4].

8.5.1. Динаміка з лінійною віковою структурою

Дослідимо деякі основні властивості розв'язків лінійної моделі динаміки популяції з віковою структурою. Отримані результати важливі для дослідження задач оптимального приросту популяції.

Приймемо позначення: $A, T \in (0, +\infty)$, $Q_T = (0, A) \times (0, T)$, $Dy -$

похідна за напрямом $(1, 1)$ від y , тобто

$$Dy(a, t) = \lim_{h \rightarrow 0} \frac{y(a+h, t+h) - y(a, t)}{h}.$$

Розглянемо біологічну популяцію одного виду і позначимо через $y(a, t)$ кількість (щільність, густину) особин віком $a \in [0, A]$ у момент часу $t \in [0, T]$. Припустимо, що y – достатньо гладка.

Нехай $\mu(a, t)$ означає коефіцієнт смертності особин віком $a \in [0, A]$ у момент часу $t \in [0, T]$. Він визначає частку смертності особин і залежить від віку a і часу t . Закон рівноваги стверджує, що кількість особин віку a у момент часу t , які померли на часовому інтервалі $[t, t+dt]$, обчислюється за формулою

$$y(a, t) - y(a+dt, t+dt) = \mu(a, t)y(a, t)dt.$$

Поділивши рівняння на dt , одержимо

$$Dy(a, t) + \mu(a, t)y(a, t) = 0.$$

Якщо є притік популяції, тоді динаміка популяції описується рівнянням

$$Dy(a, t) + \mu(a, t)y(a, t) = f(a, t), \quad (a, t) \in Q_T, \quad (8.5.1)$$

де A – максимальний вік популяції виду і $f(a, t)$ є густиною притоку популяції віку a у момент часу t .

Рівняння (8.5.1) було запропоновано F.R. Sharpe, A. Lotka та A.G. McKendrick (див., наприклад, [13], [48], [70], [72]). Якщо y достатньо гладка функція, тоді

$$Dy(a, t) = \frac{\partial y}{\partial a} + \frac{\partial y}{\partial t}.$$

Отже, одержимо рівняння, яке у наукових працях називають рівнянням Sharpe–Lotka–McKendrick's, і записують у вигляді

$$\frac{\partial y}{\partial a}(a, t) + \frac{\partial y}{\partial t}(a, t) + \mu(a, t)y(a, t) = f(a, t),$$

яке є рівнянням з частинними похідними першого порядку.

Процес народження описується законом відновлення

$$y(0, t) = \int_0^A \beta(a, t)y(a, t)da, \quad t \in (0, T), \quad (8.5.2)$$

де $y(0, t)$ – кількість новонароджених у момент часу t ; $\beta(a, t)$ – коефіцієнт смертності, який дорівнює відношенню новонароджених до зрілої популяції віком a у момент часу t . Варто зазначити, що умова (8.5.2) нелокальна.

Початковий розподіл популяції за віком визначається умовою

$$y(a, 0) = y_0(a), \quad a \in (0, A), \quad (8.5.3)$$

де $y_0(a)$ задана функція.

У реальних умовах можна спостерігати, що для кожного фіксованого значення $t \geq 0$ графіки $\beta(\cdot, t)$ і $\mu(\cdot, t)$ мають вигляд, зображений на рис. 8.32а. На рисунках зображені графіки функцій $\beta(a) = 10a^2(A - a)(1 + \sin(\pi/Aa))$ та $\mu(a) = \exp(-a)/(A - a)$, де $A = 1$. Зауважимо, що A перемасштабовано до 1.

Із практичних спостережень за біологічними популяціями випливають такі припущення щодо функцій β , μ та y_0 :

$$(H1) \quad \beta \in L^\infty(Q_T), \quad \beta(a, t) \geq 0 \text{ д.м.в. } (a, t) \in Q_T;$$

$$(H2) \quad \mu \in L^1_{loc}([0, A] \times [0, T]), \quad \mu(a, t) \geq 0 \text{ д.м.в. } (a, t) \in Q_T;$$

$$(H3) \quad y_0 \in L^1(0, A), \quad y_0(a) \geq 0 \text{ д.м.в. } a \in (0, A);$$

$$f \in L^1(Q_T), \quad f(a, t) \geq 0 \text{ д.м.в. } (a, t) \in Q_T.$$

Розв'язком задачі (8.5.1)-(8.5.3) називаємо функцію $y \in L^\infty(0, T; L^1(0, A))$, абсолютно неперервну вздовж кожної характеристики (рівняння характеристики $a - t = k$, $(a, t) \in \bar{Q}_T$, $k \in \mathbb{R}$), та задовольняє задачу

$$\begin{cases} Dy(a, t) + \mu(a, t)y(a, t) = f(a, t), & \text{м.в. у } Q_T, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t)da, & \text{д.м.в. } t \in (0, T), \\ y(a, 0) = y_0(a), & \text{д.м.в. } a \in (0, A). \end{cases} \quad (8.5.4)$$

Позаяк розв'язок y задачі (8.5.1)-(8.5.3) абсолютно неперервний вздовж кожної характеристики, то умови (8.5.4)_{2,3} є суттєвими, зокрема, $y(0, t)$ та $y(a, 0)$ означають:

$$y(0, t) = \lim_{\varepsilon \rightarrow 0^+} y(\varepsilon, t + \varepsilon), \quad \text{д.м.в. } t \in (0, T),$$

$$y(a, 0) = \lim_{\varepsilon \rightarrow 0^+} y(a + \varepsilon, \varepsilon), \quad \text{д.м.в. } a \in (0, A).$$

Теорема 8.5.1. *Розв'язок задачі (8.5.1)-(8.5.3) єдиний і невід'ємний.*

Доведення. Нехай y – розв'язок задачі (8.5.1)-(8.5.3). Із означення розв'язку одержуємо (інтегруючи вздовж характеристики)

$$y(a, t) = \exp \left\{ - \int_0^a \mu(s, t - a + s) ds \right\} b(t - a) + \int_0^a \exp \left\{ - \int_s^a \mu(\tau, t - a + \tau) d\tau \right\} f(s, t - a + s) ds, \quad (8.5.5)$$

м.в. на $\{(a, t) \in Q_T; a < t\}$, і

$$y(a, t) = \exp \left\{ - \int_0^t \mu(a - t + s, s) ds \right\} y_0(a - t) + \int_0^t \exp \left\{ - \int_s^t \mu(a - t + \tau, \tau) d\tau \right\} f(a - t + s, s) ds, \quad (8.5.6)$$

м.в. на $\{(a, t) \in Q_T; a > t\}$.

Уведемо позначення

$$b(t) = \int_0^A \beta(a, t) y(a, t) da \quad \text{д.м.в.} \quad t \in (0, T). \quad (8.5.7)$$

На підставі властивостей β і y , беручи до уваги (8.5.7), робимо висновок, що $b \in L^\infty(0, T)$. Отож, припустивши, що правильне включення $b \in L^\infty(0, T)$, одержимо, що y , визначена формулами (8.5.5), (8.5.6), є єдиним розв'язком задачі

$$\begin{cases} Dz(a, t) + \mu(a, t)z(a, t) = f(a, t), & (a, t) \in Q_T, \\ z(0, t) = b(t), & t \in (0, T), \\ z(a, 0) = y_0(a), & a \in (0, A). \end{cases} \quad (8.5.8)$$

Тут під розв'язком задачі (8.5.8) розуміємо функцію з класу $z \in L^\infty(0, T; L^1(0, A))$, абсолютно неперервну вздовж кожної характеристики, яка задовольняє

$$\begin{cases} Dz(a, t) + \mu(a, t)z(a, t) = f(a, t), & \text{м.в. на } Q_T, \\ \lim_{\varepsilon \rightarrow 0+} z(\varepsilon, t + \varepsilon) = b(t), & \text{д.м.в.} \quad t \in (0, T), \\ \lim_{\varepsilon \rightarrow 0+} z(a + \varepsilon, \varepsilon) = y_0(a), & \text{д.м.в.} \quad a \in (0, A). \end{cases}$$

Розв'язок y задачі (8.5.1)-(8.5.3) задовольняє (8.5.5), (8.5.6), де b визначається (8.5.7). Отож, можемо зробити висновок, що

$$\begin{aligned}
 b(t) = & \int_0^A \beta(a, t) y(a, t) da = \\
 & \int_0^t \beta(a, t) \exp \left\{ - \int_0^a \mu(s, t - a + s) ds \right\} b(t - a) da + \\
 & \int_0^t \beta(a, t) \int_0^a \exp \left\{ - \int_s^a \mu(\tau, t - a + \tau) d\tau \right\} f(s, t - a + s) ds da + \\
 & \int_t^A \beta(a, t) \exp \left\{ - \int_0^t \mu(a - t + s, s) ds \right\} y_0(a - t) da + \\
 & \int_t^A \beta(a, t) \int_0^t \exp \left\{ - \int_s^t \mu(a - t + \tau, \tau) d\tau \right\} f(a - t + s, s) ds da
 \end{aligned}$$

для майже всіх $0 < t < \min\{T, A\}$, і

$$\begin{aligned}
 b(t) = & \int_0^A \beta(a, t) y(a, t) da = \\
 & \int_0^A \beta(a, t) \exp \left\{ - \int_0^a \mu(s, t - a + s) ds \right\} b(t - a) da + \\
 & \int_0^A \beta(a, t) \int_0^a \exp \left\{ - \int_s^a \mu(\tau, t - a + \tau) d\tau \right\} f(s, t - a + s) ds da
 \end{aligned}$$

для майже всіх $\min\{T, A\} = A < t < T$ (якщо $A < T$). І на кінець, b задовольняє таке інтегральне рівняння Volterra

$$b(t) = F(t) + \int_0^t K(t, t - s) b(s) ds \quad \text{д.м.в.} \quad t \in (0, T) \quad (8.5.9)$$

(яке відоме як *рівняння відновлення* або *рівняння Lotka*). Тут

$$F(t) := \int_0^t \beta(a, t) \int_0^a \exp \left\{ - \int_s^a \mu(\tau, t - a + \tau) d\tau \right\} f(s, t - a + s) ds da +$$

$$\int_t^A \beta(a, t) \int_0^t \exp \left\{ - \int_s^t \mu(a - t + \tau, \tau) d\tau \right\} f(a - t + s, s) ds da +$$

$$\int_t^A \beta(a, t) \exp \left\{ - \int_0^t \mu(a - t + s, s) ds \right\} y_0(a - t) da$$

для майже всіх $0 < t < \min\{T, A\}$, і

$$F(t) := \int_0^A \beta(a, t) \int_0^a \exp \left\{ - \int_s^a \mu(\tau, t - a + \tau) d\tau \right\} f(s, t - a + s) ds da$$

для майже всіх $\min\{T, A\} = A < t < T$ (якщо $A < T$), а також

$$K(t, a) := \begin{cases} \beta(a, t) e^{-\int_0^a \mu(s, t-a+s) ds}, & \text{д.м.в. } (a, t) \in Q_T, a < t, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Із припущень щодо гладкості функцій випливає, що

$$\begin{cases} K \in L^\infty(Q_T), & K(t, a) \geq 0 \text{ д.м.в. } (a, t) \in Q_T, \\ F \in L^\infty(0, T), & F(t) \geq 0 \text{ д.м.в. } t \in (0, T). \end{cases} \quad (8.5.10)$$

Використовуючи теорему Banach'а про нерухому точку, доведемо, що рівняння відновлення має єдиний розв'язок $b \in L^\infty(0, T)$. Для цього уведемо норму у просторі $L^\infty(0, T)$ так:

$$\|w\| = \text{Ess sup}_{t \in (0, T)} (e^{-\gamma t} |w(t)|), \quad w \in L^\infty(0, T),$$

де $\gamma > 0$ стала, яку визначимо нижче; ця норма еквівалентна звичайній нормі у $L^\infty(0, T)$.

Розглянемо оператор

$$\mathcal{F} : L^\infty(0, T) \rightarrow L^\infty(0, T)$$

такий, що

$$(\mathcal{F}w)(t) = F(t) + \int_0^t K(t, t-s)w(s)ds \quad \text{д.м.в.} \quad t \in (0, T).$$

Для кожних $b_1, b_2 \in L^\infty(0, T)$:

$$\begin{aligned} & \|(\mathcal{F}b_1)(t) - (\mathcal{F}b_2)(t)\| = \\ & = \text{Ess sup}_{t \in (0, T)} \left(e^{-\gamma t} \left| \int_0^t K(t, t-s)(b_1 - b_2)(s)ds \right| \right) \leq \\ & \leq \text{Ess sup}_{t \in (0, T)} \left(e^{-\gamma t} \|K\|_{L^\infty(Q_T)} \int_0^t e^{\gamma s} e^{-\gamma s} |(b_1 - b_2)(s)| ds \right) \leq \\ & \leq \text{Ess sup}_{t \in (0, T)} \left(e^{-\gamma t} \|\beta\|_{L^\infty(Q_T)} \cdot \|b_1 - b_2\| \cdot \frac{e^{\gamma t}}{\gamma} \right). \end{aligned}$$

Звідси, для кожної сталої $\gamma > \|\beta\|_{L^\infty(Q_T)}$, \mathcal{F} є стискующим відображенням на $(L^\infty(0, T), \|\cdot\|)$. Тому з теореми Ванаш'а про нерухому точку випливає, що існує єдиний розв'язок $b \in L^\infty(0, T)$ інтегрального рівняння (8.5.9). Крім того, b є границею у $(L^\infty(0, T), \|\cdot\|)$ послідовності наближень розв'язку $\{b_n\}$, визначену так:

$$\begin{aligned} b_0(t) &= F(t), & \text{д.м.в.} \quad t \in (0, T), \\ b_{n+1}(t) &= F(t) + \int_0^t K(t, t-s)b_n(s)ds, & \text{д.м.в.} \quad t \in (0, T), \quad n \in \mathbb{N}. \end{aligned} \tag{8.5.11}$$

Із (8.5.10) випливає, що $b_n(t) \geq 0$ д.м.в. $t \in (0, T)$, $\forall n \in \mathbb{N}$, з чого робимо висновок, що b є єдиним невід'ємним на $(0, T)$ розв'язком інтегрального рівняння (8.5.9).

Розглянемо зараз y , що визначається за допомогою (8.5.5) і (8.5.6), де b – єдиний розв'язок інтегрального рівняння (8.5.9). Припущення **(Н1)**-**(Н3)**, (8.5.5) та (8.5.6) дають змогу зробити висновок, що $y(a, t) \geq 0$ м.в. на Q_T та $\mu y \in L^1(Q_T)$.

Використовуючи повторно (8.5.5) та (8.5.6), одержимо, що y – розв'язок задачі (8.5.1)-(8.5.3). Цей розв'язок єдиний, оскільки b як розв'язок інтегрального рівняння (8.5.9) єдиний і невід'ємний, позаяк b, y_0, f є невід'ємними. Теорема доведена. \square

Теорема 8.5.2. Якщо коефіцієнт смертності задовольняє додаткову умову

$$\int_0^A \mu(a, t - A + a) da = +\infty \quad \text{д.м.в.} \quad t \in (0, T), \quad (\text{H4})$$

де μ доповнено нулем на $(0, A) \times (-\infty, 0)$, тоді розв'язок задачі (8.5.1)-(8.5.3) задовольняє рівність

$$\lim_{\varepsilon \rightarrow 0+} y(A - \varepsilon, t - \varepsilon) = 0 \quad \text{д.м.в.} \quad t \in (0, T),$$

тобто $y(A, t) = 0$ д.м.в. $t \in (0, T)$.

Доведення. Із (8.5.6) д.м.в. $t \in (0, \min\{T, A\})$ і для достатньо малого значення $\varepsilon > 0$ одержуємо

$$\begin{aligned} y(A - \varepsilon, t - \varepsilon) &= \exp \left\{ - \int_0^{t-\varepsilon} \mu(A - t + s, s) ds \right\} y_0(A - t) + \\ &\quad \int_0^{t-\varepsilon} \exp \left\{ - \int_s^{t-\varepsilon} \mu(A - t + \tau, \tau) d\tau \right\} f(A - t + s, s) ds \xrightarrow{\varepsilon \rightarrow 0+} \\ &\quad \exp \left\{ - \int_0^t \mu(A - t + s, s) ds \right\} y_0(A - t) + \\ &\quad \int_0^t \exp \left\{ - \int_s^t \mu(A - t + \tau, \tau) d\tau \right\} f(A - t + s, s) ds = 0. \end{aligned}$$

Що і треба було довести. □

♣ Зауваження 8.5.1. Якщо y_0 , β , μ і f задовольняють сильніші умови (які є у реальних моделях), а саме: $y_0 \in C([0, A])$, $\beta \in C(\bar{Q}_T)$, $\mu \in C([0, A] \times [0, T])$, $f \in C(\bar{Q}_T)$, тоді одержимо доведення аналогічне як у теоремі 8.5.1, що розв'язок y задачі (8.5.1)-(8.5.3) є неперервний на $D_1 \cup D_2$, де $D_1 = \{(a, t) \in \bar{Q}_T; a > t\}$, $D_2 = \{(a, t) \in \bar{Q}_T; a < t\}$

Доведемо таку важливу теорему.

Теорема 8.5.3. Нехай y є розв'язком задачі (8.5.1)-(8.5.3).

(i) Якщо $f(a, t) > 0$ м.в. на Q_T , тоді $y(a, t) > 0$ м.в. на Q_T .

(ii) Якщо $\beta_i, \mu_i, y_{0i}, f_i$ задовольняють умови **(H1)**-**(H3)** ($i \in \{1, 2\}$)
і, крім того,

$$\begin{aligned} \beta_1(a, t) &\geq \beta_2(a, t), \quad \mu_1(a, t) \leq \mu_2(a, t), \quad \text{м.в. у } Q_T, \\ y_{01}(a) &\geq y_{02}(a), \quad \text{м.в. на } (0, T), \\ f_1(a, t) &\geq f_2(a, t), \quad \text{м.в. у } Q_T, \end{aligned}$$

тоді $y^1(a, t) \geq y^2(a, t)$ м.в. у Q_T , де y^i є розв'язком задачі (8.5.1)-
(8.5.3), що відповідає $\beta := \beta_i, \mu := \mu_i, y_0 := y_{0i}, f := f_i, i \in \{1, 2\}$.

Крім того, якщо $f_n \rightarrow f$ у $L^1(Q_T)$ і f_n задовольняє **(H3)**, тоді

$$y_n \rightarrow y$$

у $L^\infty(0, T; L^1(0, A))$, де y_n є розв'язком (8.5.1)-(8.5.3) при $f := f_n$,
відповідно.

Доведення. Із теореми 8.5.1 випливає, що розв'язок задачі (8.5.1)-
(8.5.3) визначається формулами (8.5.5) і (8.5.6), де b , розв'язок інте-
грального рівняння (8.5.9), невід'ємний. Якщо $f(a, t) > 0$ м.в. на Q_T ,
тоді з (8.5.5) і (8.5.6) випливає, що $y(a, t) > 0$ м.в. на Q_T .

Із (8.5.11) випливає, що розв'язок b рівняння (8.5.9) можна отримати
граничним переходом у просторі $L^\infty(0, T)$ у послідовності

$$\begin{cases} b_0^{\beta, \mu, y_0, f}(t) = F^{\beta, \mu, y_0, f}(t), \\ b_{n+1}^{\beta, \mu, y_0, f}(t) = F^{\beta, \mu, y_0, f}(t) + \int_0^t K^{\beta, \mu, y_0, f}(t, t-s) b_n^{\beta, \mu, y_0, f}(s) ds, \end{cases} \quad (8.5.12)$$

д.м.в. $t \in (0, T)$, $n \in \mathbb{N}^*$, де $F^{\beta, \mu, y_0, f}$ і $K^{\beta, \mu, y_0, f} \in f$ і K визначені раніше
(як наголошувалось, ці функції залежать від β, μ, y_0, f).

Якщо

$$\begin{aligned} \beta_1(a, t) &\geq \beta_2(a, t), \quad \mu_1(a, t) \leq \mu_2(a, t), \quad \text{м.в. у } Q_T, \\ y_{01}(a) &\geq y_{02}(a), \quad \text{м.в. на } (0, T), \\ f_1(a, t) &\geq f_2(a, t), \quad \text{м.в. у } Q_T, \end{aligned}$$

тоді

$$\begin{aligned} K^{\beta_1, \mu_1, y_{01}, f_1}(t, a) &\geq K^{\beta_2, \mu_2, y_{02}, f_2}(t, a) \quad \text{м.в. на } Q_T, \\ F^{\beta_1, \mu_1, y_{01}, f_1}(t) &\geq F^{\beta_2, \mu_2, y_{02}, f_2}(t, a) \quad \text{м.в. на } (0, T), \end{aligned}$$

та з (8.5.12) робимо висновок, що $b_0^{\beta_2, \mu_2, y_{02}, f_2}(t) \leq b_0^{\beta_1, \mu_1, y_{01}, f_1}(t)$ м.в.на $(0, T)$. А з (8.5.5) і (8.5.6) випливає, що $y^1(a, t) \geq y^2(a, t)$ м.в.на Q_T .

І накінець, якщо $f_n \rightarrow f$ у просторі $L^1(Q_T)$, то робимо висновок, що $F_n \rightarrow F$ у просторі $L^\infty(0, T)$ (де F_n і F відповідають $f := f_n$). Справді,

$$F_n(t) - F(t) = \int_0^t \beta(a, t) \int_0^a e^{-\int_s^a \mu(\tau, t-a+\tau) d\tau} (f_n - f)(s, t-a+s) ds da + \\ \int_t^A \beta(a, t) \int_0^t e^{-\int_s^t \mu(a-t+\tau, \tau) d\tau} (f_n - f)(a-t+s, s) ds da$$

д.м.в. $t \in (0, \min\{T, A\})$, а тому

$$F_n(t) - F(t) = \int_0^A \beta(a, t) \int_0^a e^{-\int_s^a \mu(\tau, t-a+\tau) d\tau} (f_n - f)(s, t-a+s) ds da$$

д.м.в. $A = \min\{T, A\} < t < T$ (у випадку $A < T$). Отож, відповідно, одержимо

$$\|F_n - F\|_{L^\infty(0, T)} \leq \|\beta\|_{L^\infty(Q_T)} \cdot \|f_n - f\|_{L^1(Q_T)},$$

звідки випливає $F_n \rightarrow F$ у просторі $L^\infty(0, T)$ при $n \rightarrow +\infty$.

Використавши (8.5.9) та лему Bellman'а (див. [13]) одержимо, що $b_{f_n} \rightarrow b$ у просторі $L^\infty(0, T)$, де b_{f_n} є розв'язок (8.5.9), що відповідає $F := F_n$ (і відповідно $f := f_n$).

Із (8.5.5) і (8.5.6) випливає, що $y_n \rightarrow y$ у просторі $L^\infty(0, T; L^1(0, A))$, що і завершує доведення теореми. \square

♣ Зауваження 8.5.2. Для кожної функції $f \in L^1(Q_T)$ такої, що $f(a, t) > 0$ м.в. у Q_T , розв'язок y задачі (8.5.1)-(8.5.3) задовольняє оцінку

$$y(a, t) > 0 \text{ д.м.в. } (a, t) \in Q_T.$$

Ця нерівність підтверджує біологічну інтерпретацію A як максимальний від популяції виду.

Наведемо декілька зауважень щодо поведінки на великому проміжку часу розв'язку лінійної віковозалежної динаміки популяції у випадку незалежного від часу коефіцієнта смертності та нульового зовнішнього

приросту популяції. Розглянемо таку модель:

$$\begin{cases} Dy(a, t) + \mu(a)y(a, t) = 0, & (a, t) \in Q, \\ y(0, t) = \int_0^A \beta(a)y(a, t)da, & t \in (0, +\infty), \\ y(a, 0) = y_0(a), & a \in (0, A), \end{cases} \quad (8.5.13)$$

де $Q = (0, T) \times (0, +\infty)$.

Нехай виконуються умови

(A1) $\beta \in L^\infty(0, A)$, $\beta(a) \geq 0$ д.м.в. $a \in (0, A)$, $\beta \neq 0_{L^\infty(0, A)}$;

(A2) $\mu \in L^1_{loc}([0, A])$, $\mu(a) \geq 0$ д.м.в. $a \in (0, A)$,

$$\int_0^A \mu(a)da = +\infty,$$

(A3) $y_0 \in L^1(0, A)$, $y_0(a) > 0$ д.м.в. $a \in (0, A)$.

Позначимо через R величину

$$R = \int_0^A \beta(a) e^{-\int_0^a \mu(s)ds} da,$$

і назвемо її коефіцієнтом репродукції.

Теорема 8.5.4. *Якщо виконуються умови (A1)-(A3), тоді*

$$\begin{aligned} \lim_{t \rightarrow \infty} \|y(t)\|_{L^\infty(0, A)} &= 0, \quad \text{якщо } R < 1, \\ \lim_{t \rightarrow \infty} \|y(t)\|_{L^1(0, A)} &= +\infty, \quad \text{якщо } R > 1, \\ \lim_{t \rightarrow \infty} \|y(t) - \tilde{y}\|_{L^\infty(0, A)} &= 0, \quad \text{якщо } R = 1, \end{aligned}$$

де y є розв'язок (8.5.13) і \tilde{y} нетривіальний стійкий розв'язок задачі (8.5.13)_{1,2}.

Доведення цієї теореми можна знайти у [13] і [48]. Твердження теореми будуть перевірені числовими тестами.

♣ **Зауваження 8.5.3.** Нехай виконуються умови **(A1)**-**(A3)** і $y_0 \in C([0, A])$. Якщо виконується умова

$$y_0(0) = \int_0^A \beta(a)y(a)da, \quad (\mathbf{Ac})$$

тоді розв'язок b інтегрального рівняння (8.5.9) є неперервним, $b(0) = y_0(0)$, а з (8.5.5) і (8.5.6) одержуємо, що $y \in C([0, A] \times [0, +\infty))$ (див. [13]).

Якщо умова погодження **(Ac)** не виконується, тоді $\{(a, a); a \in [0, A]\}$ є лінією розриву y розв'язку задачі (8.5.13). Геометричну інтерпретацію цього твердження отримаємо у чисельній симуляції у MatLab[®].

Чисельний розв'язок

Побудуємо апроксимацію типу Euler'а для задачі (8.5.1)-(8.5.3), у якій візьмемо $f \equiv 0$. При написанні апроксимаційного алгоритму визначення наближеного розв'язку задачі візьмемо до уваги метод доведення Теорема 8.5.1, а саме, інтегрування вздовж характеристики. Насамперед визначимо сітки вузлів

$$a_i = (i - 1)h, \quad i = 1, 2, \dots, M,$$

і

$$t_j = (j - 1)h, \quad j = 1, 2, \dots, N,$$

де $h > 0$ є крок сітки, а кількість вузлів відповідає крайнім точкам A та T , відповідно, тобто

$$M = 1 + A/h, \quad N = 1 + T/h.$$

Для позначення апроксимації функції введемо позначення

$$\varphi(a_i, t_j) \approx \varphi_i^{(j)}.$$

Після дискредитації у точці (a_i, t_j) рівняння (8.5.1) запишеться

$$\frac{y_i^{(j)} - y_{i-1}^{(j-1)}}{h} + \mu_i^{(j)} y_i^{(j)} = 0. \quad (8.5.14)$$

Використаємо для похідної першого порядку обернену скінченорізницевою апроксимацію. Нехай φ є функцією класу C^2 . Тоді за формулою Taylor'а

$$\varphi(x - h) = \varphi(x) - h\varphi'(x) + \frac{h^2}{2}\varphi''(\xi),$$

для деякого $\xi \in (x - h, x)$. Звідки справджується рівність

$$\varphi'(x) = \frac{\varphi(x) - \varphi(x - h)}{h} + \mathcal{O}(h). \quad (8.5.15)$$

Зауважимо, щоб одержати формулу (8.5.14), треба застосувати (8.5.15) вздовж діагоналі добутку сіток на $[0, A] \times [0, T]$. Із (8.5.14) легко одержуємо

$$y_i^{(j)} = (1 + h\mu_i^{(j)})^{-1} y_{i-1}^{(j-1)}.$$

Позаяк μ є незалежна від t у досліджуваному випадку, тому вище записана формула набуде простішого вигляду, а саме

$$y_i^{(j)} = (1 + h\mu_i)^{-1} y_{i-1}^{(j-1)}. \quad (8.5.16)$$

Для апроксимації інтеграла в умові (8.5.2) застосуємо формулу трапецій, згідно з якою одержимо рівність

$$\int_c^d \varphi(x) dx = \frac{h}{2} [\varphi(c) + \varphi(d)] + h \sum_{i=2}^{n-1} \varphi(x_i) + \mathcal{O}(h^2),$$

де $h > 0$ є кроком сітки

$$c = x_1 < x_2 < \dots < x_n = d,$$

за припущення, що $\varphi \in C^2([c, d])$.

Опишемо апроксимаційний метод побудови алгоритму обчислення наближеного розв'язку задачі (8.5.1)-(8.5.3). Через $z(i, j)$ позначимо $z(a_i, t_j)$.

Алгоритм 8.1

/* Обчислимо розв'язок на першому за часом кроці ($j = 1$) із початкової умови (8.5.3) */

```
for i = 1 to M
  y(i, 1) = y0(a(i))
end-for
```

/* Обчислимо розв'язок на вищих рівнях за часом, використовуючи формулу (8.5.16) та формулу трапецій

```

for j = 2 to N
  for i = 2 to M
    y(i,j) = y(i - 1,j - 1)/(1 + h * mu(i))
  end-for
  for i = 1 to M
    w(i) =beta(i) * y(i,j)
  end-for
  y(1,j) = trapz(a,w)
end-for

```

Нагадаємо, що тут `trapz` є функцією MatLab[®] чисельного інтегрування. Відзначимо, що $y(1, j)$ невідоме (ще не обчислене значення) у циклі

```

for i = 1 to M
  w(i) = beta(i) * y(i,j)
end-for

```

Тому $w(1)$ також невідоме на момент обчислення `trapz(a,w)`. Це логічний висновок, проте, взявши у наших чисельних тестах $\beta(1) = 0$, і тоді $w(1) = 0$. Отож, обчислення значення $y(1, j)=\text{trapz}(a,w)$ буде проводитись правильно.

Запишемо програму обчислення розв'язку задачі (8.5.1)-(8.5.3) у випадку, коли $\beta(a, t) = B \cdot a^2(A - a)(1 + \sin(\pi/Aa))$, де $B > 0$, $\mu(a, t) = \exp(-a)/(A - a)$, $y_0(a) = \exp(-a^2/2)$:

```

% file pop1.m
% Program to compute the density of population
% using the backward Euler method ( step is h )
clear
A=input('The maximal age is A: ');
T=input('The final time is T: ');
h=input('The discretization parameter is h: ');
ct=input('constant for beta: '); % constant B>0 for beta
lw=input('LineWidth: ');
M=1+A/h;
N=1+T/h;
for i=1:M
  a(i)=(i-1)*h;
end
for j=1:N
  t(j)=(j-1)*h;
end
beta=ct*a.^2.*(A-a).*(1+sin(pi/A*a));

```

```

miu=exp(-a)/(A-a);
R=trapz(a,beta.*exp(-miu));
y=zeros(M,N);
for i=1:M
    y(i,1)=exp(-0.5*a(i)^2);
end
for j=2:N
    j
    for i=2:M
        y(i,j)=y(i-1,j-1)/(1+h*miu(i));
    end
    for i=1:M
        w(i)=y(i,j);
    end
    y(1,j)=trapz(a,beta.*w);
end
for i=1:M
    for j=1:N
        age(i,j)=a(i);
        time(i,j)=t(j);
    end
end
end
% make figures
K=max(max(y));
meshz(age,time,y); % or mesh
axis([0 A 0 T 0 K])
xlabel('\bf{Age a}','FontSize',16)
ylabel('\bf{Time t}','FontSize',16)
zlabel('\bf{y(a,t)}','FontSize',16)
figure(2)
plot(a, beta,'LineWidth',lw); grid
title('\bf{\beta and \mu}','FontSize',16)
xlabel('\bf{a}','FontSize',16)
hold on
plot(a, miu,'r','LineWidth',lw);
text(0.6,4,'\bf{\beta(a)}','FontSize',16)
text(0.8,6,'\bf{\mu(a)}','FontSize',16)
hold off

```

Координати об'єкта `text` у координатній системі залежать від розташування графіка.

Для одержання точнішої апроксимації рівняння (8.5.1), використа-

ємо центровану скінчено різницеву схему. Нехай φ – тричі неперервно диференційовна, тобто, функція класу C^3 . Тоді за формулою Taylor'а запишемо

$$\begin{aligned}\varphi(x+h) &= \varphi(x) + h\varphi'(x) + \frac{h^2}{2}\varphi''(x) + \frac{h^3}{6}\varphi'''(\xi_1), \\ \varphi(x-h) &= \varphi(x) - h\varphi'(x) + \frac{h^2}{2}\varphi''(x) + \frac{h^3}{6}\varphi'''(\xi_2).\end{aligned}$$

Звідки, віднявши обидві формули вище, одержимо

$$\varphi'(x) = \frac{\varphi(x+h) - \varphi(x-h)}{2h} + \mathcal{O}(h^2).$$

Отож, використавши зображення похідної, апроксимація рівняння (8.5.1) виглядатиме так:

$$\frac{y_{i+1}^{(j+1)} - y_{i-1}^{(j-1)}}{2h} + \mu_i^{(j)} y_i^{(j)} = 0.$$

Взявши до уваги, що $\mu_i^{(j)} = \mu_i$, остаточно одержимо ітераційну формулу апроксимації рівняння (8.5.1)

$$y_{i+1}^{(j+1)} = y_{i-1}^{(j-1)} - 2h\mu_i y_i^{(j)}. \quad (8.5.17)$$

Замінивши формулу (8.5.16) на (8.5.17), одержимо такий алгоритм обчислення наближеного розв'язку задачі.

Алгоритм 8.2

/* Використовуючи умову (8.5.3), обчислимо розв'язок на першому за часом кроці */

```
for i=1 to M
  y(i,1)=y0(a(i))
end-for
```

/* Використовуючи формулу (8.5.16) та формулу трапецій, обчислимо розв'язок на другому за часом кроці */

```
j=2
i=2 to M
  y(i,j)=y(i-1,j-1)/(1 + h*mu(i))
end-for
for i=1 to M
  w(i)=beta(i)*y(i,j)
end-for
y(1,j)=trapz(a,w)
```

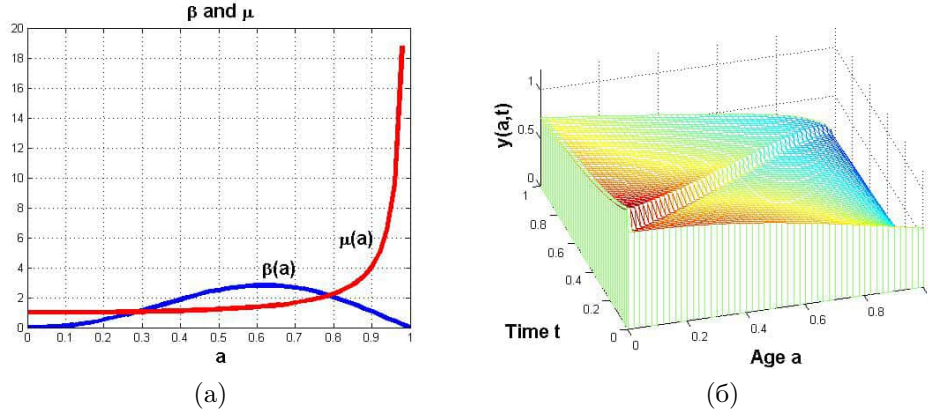


Рис. 8.32. Графіки програми pop1.m: *a* – коефіцієнти фертильності (β) та смертності (μ); *b* – розподіл популяції

/* Використовуючи (8.5.17) та формулу трапецій, обчислимо розв'язок на кроках $j = 3, \dots, N$, дискретизованих за часом */

```

for j=2 to N-1
  for i=2 to M-1
    y(i+1,j+1)=y(i-1,j-1)-2*h*mu(i)*y(i,j)
  end-for
  i=2
  y(i,j+1)=y(i-1,j)/(1 + h*mu(i-1))
  for i=1 to M
    w(i)=beta(i)*y(i,j+1)
  end-for
  y(1,j+1)=trapz(a,w)
end-for

```

Звернемо увагу на те, що для $j = 2$ і $i = 2$, відповідно, використано однокрокові формули. Наприклад, $y(:, 1)$ обчислюється з початкової умови (8.5.3), тоді як $y(:, 2)$ обчислюється за допомогою формули (8.5.16), і накінець $y(:, j)$ для $j = 3, \dots, N$ обчислюється за допомогою формули (8.5.17).

♣ **Зауваження 8.5.4.** Вздовж лінії $a = t$ відбувається розрив, позаяк початкова умова не задовольняє умову погодження

$$y_0(0) = \int_0^A \beta(a)y_0(a)da.$$

Детальніше див. [13].

Вектори β , μ , і $y(:,1)$ з програми вище, які відповідають значенням у вузлах сітки функцій β , μ , y_0 , обчислюються так:

$$\beta(i) = \beta(a_i), \quad \mu(i) = \mu(a_i), \quad y(i,1) = y_0(a_i), \quad i = 1, 2, \dots, M,$$

можна ввести з файлів. Припустимо, що утворені файли `beta.txt`, `mu.txt`, `y0.txt` у форматі стовпчика значень (одне значення у рядку). Тоді за допомогою послідовності команд введемо дані у програму:

```
load mu.txt
mu=mu'; % transform into a row vector
load beta.txt
beta=beta'; % transform into a row vector
load y0.txt
y0=y0'; % transform into a row vector
```

8.5.2. Задача оптимального розмноження

У цьому розділі дослідимо задачу оптимального розмноження, яка описує динаміку лінійної віково структурованої популяції. Відповідна модель у вигляді рівняння з частинними похідними була сформульована у попередньому розділі. У цьому розділі доведемо існування оптимального керування, введемо необхідні умови оптимального першого порядку і використаємо їх під час побудови чисельного алгоритму знаходження наближеного розв'язку задачі. Цей алгоритм є основою програми апроксимації оптимального керування (та оптимального значення функції кошту).

Досліджувана задача має важливе значення з економічної погляду (оскільки визначає оптимальні стратегії) та з біологічного.

Отож, сформулюємо задачу оптимального розмноження: максимізувати функціонал

$$\int_0^T \int_0^A u(a,t) y^u(a,t) da dt \rightarrow \max \quad (\text{ОНР})$$

стосовно $u \in K = \{w \in L^\infty(Q_T) : 0 \leq w(a, t) \leq L \text{ м.в. на } Q_T\}$, де y^u є розв'язком задачі

$$\begin{cases} Dy(a, t) + \mu(a, t)y(a, t) = f(a, t) - u(a, t)y(a, t), & (a, t) \in Q_T, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t)da, & t \in (0, T), \\ y(a, 0) = y_0(a), & a \in (0, A). \end{cases} \quad (8.5.18)$$

Тут $L > 0$, K є множиною обмежень керування. $u \in K$ – керування або зусилля спрямовані на розмноження. Також u відіграє роль додаткового коефіцієнта смертності. Інтеграл у (ОНР) дорівнює значенню сукупного розмноження.

Наше завдання полягає у знаходженні такого керування (зусилля спрямовані на розмноження), за якого розмноження є максимальне – таке керування називаємо *оптимальним* (оптимальне зусилля спрямоване на розмноження).

Дослідження, запрезентовані у цьому розділі, стимульовані працею [3].

З метою спрощення припустимо, що виконуються умови (Н1)-(Н4). Оскільки $\mu := \mu + u$ задовольняє (Н2) ($\forall u \in K$), то можна зробити висновок, що задача (8.5.18) має єдиний розв'язок y^u . Цей розв'язок невід'ємний.

Існування оптимального керування

Теорема 8.5.5. *Задача (ОНР) допускає щонайменше одне оптимальне керування.*

Доведення. Позначимо

$$\Psi(u) = \int_0^T \int_0^A u(a, t)y^u(a, t)da dt, \quad u \in K,$$

і нехай

$$d = \sup_{u \in K} \Psi(u).$$

Із теореми 8.5.3 одержимо оцінку

$$0 \leq \Psi(u) \leq L \int_0^T \int_0^A y^0(a, t)da dt < +\infty,$$

(y^0 є розв'язок (8.5.18), який відповідає $u \equiv 0$) для довільного керування $u \in K$, звідки випливає, що $d \in [0, +\infty)$.

Нехай $\{u_n\}_{n \in \mathbb{N}^*} \subset K$ є послідовністю керувань, які задовольняють таку оцінку

$$d - \frac{1}{n} < \Psi(u_n) \leq d.$$

Із цієї ж теореми 8.5.3 випливає, що

$$0 \leq y^{u_n}(a, t) \leq y^0(a, t) \text{ м.в. у } Q_T,$$

і, крім того, для підпослідовності, яку позначимо $\{y^{u_n}\}$, матимемо

$$y^{u_n} \rightarrow y^* \text{ слабко у } L^2(Q_T).$$

Сформулюємо(без наведення доведення) таке допоміжне твердження, відоме як теорема Mazur'а (див. [17])

Твердження 8.5.1. (Mazur) Припустимо, що $\{x_n\}_{n \in \mathbb{N}}$ слабко збіжна послідовність у дійсному просторі Banach'а X до $x \in X$. Тоді існує послідовність $\{y_n\}_{n \in \mathbb{N}} \subset X$, $y_n \in \text{conv}\{x_k : k \geq n+1\}$, $n \in \mathbb{N}$ така, що $\{y_n\}_{n \in \mathbb{N}}$ збіжна (сильно) до x .

Використовуючи твердження 8.5.1, одержимо, що існує послідовність $\{\tilde{y}_n\}$ така, що

$$\tilde{y}_n = \sum_{i=n+1}^{k_n} \lambda_i^n y^{u_i}, \quad \lambda_i^n \geq 0, \quad \sum_{i=n+1}^{k_n} \lambda_i^n = 1$$

($k_n \geq n+1$), і

$$\tilde{y}_n \rightarrow y^* \text{ у } L^2(Q_T).$$

Нехай керування \tilde{u}_n визначається так:

$$\tilde{u}_n(a, t) = \begin{cases} \frac{\sum_{i=n+1}^{k_n} \lambda_i^n(a, t) u_i(a, t)}{\sum_{i=n+1}^{k_n} \lambda_i^n y^{u_i}(a, t)}, & \text{якщо } \sum_{i=n+1}^{k_n} \lambda_i^n y^{u_i}(a, t) \neq 0, \\ 0, & \text{якщо } \sum_{i=n+1}^{k_n} \lambda_i^n y^{u_i}(a, t) = 0. \end{cases}$$

Це керування задовольняє $\tilde{u}_n \in K$, а також

$$\tilde{y}_n(a, t) = y^{\tilde{u}_n}(a, t) \text{ м.в. у } Q_T.$$

Можна вибрати підпослідовність (яку також позначимо $\{\tilde{u}_n\}$) таку, що

$$\begin{cases} \tilde{u}_n \rightarrow u^*, & \text{слабко у } Q_T, \\ \Psi(\tilde{u}_n) = \sum_{i=n+1}^{k_n} \lambda_i^n \Psi(u_i) \rightarrow d, & \text{при } n \rightarrow +\infty, \end{cases}$$

звідки робимо висновок, що

$$d = \lim_{n \rightarrow +\infty} \Psi(\tilde{u}_n) = \int_0^T \int_0^A u^*(a, t) y^*(a, t) da dt. \quad (8.5.19)$$

З іншого боку, з слабкої збіжності $\tilde{u}_n \rightarrow u^*$ у $L^2(Q_T)$ одержимо, що $y^{\tilde{u}_n} \rightarrow y^{u^*}$ слабко у $L^2(Q_T)$, що означає

$$y^*(a, t) = y^{u^*}(a, t) \text{ м.в. у } Q_T$$

(позаяк слабка границя єдина). Із (8.5.19) одержимо, що $d = \Psi(u^*)$, а отже, u^* є оптимальним керуванням (ОНР).

□

Принцип максимуму

Теорема 8.5.6. *Припустимо додатково, що $f(a, t) > 0$ м.в. у Q_T . Якщо u^* є оптимальним керуванням задачі (ОНР) і p є розв'язком задачі*

$$\begin{cases} Dp - \mu p = u^*(1 + p) - \beta(a, t)p(0, t), & (a, t) \in Q_T, \\ p(A, t) = 0, & t \in (0, T), \\ p(a, T) = 0, & a \in (0, A), \end{cases} \quad (8.5.20)$$

тоді

$$u^*(a, t) = \begin{cases} 0, & \text{якщо } 1 + p(a, t) < 0, \\ L, & \text{якщо } 1 + p(a, t) > 0. \end{cases} \quad (8.5.21)$$

Доведення. Для кожної функції $v \in L^\infty(Q_T)$ такої, що $u^* + \varepsilon v \in K$, та для довільного достатньо малого значення $\varepsilon > 0$, маємо

$$\int_0^T \int_0^A u^* y^{u^*} da dt \geq \int_0^T \int_0^A (u^* + \varepsilon v) y^{u^* + \varepsilon v} da dt,$$

звідки

$$\int_0^T \int_0^A u^* \frac{y^{u^* + \varepsilon v} - y^{u^*}}{\varepsilon} da dt + \int_0^T \int_0^A v y^{u^* + \varepsilon v} da dt \leq 0. \quad (8.5.22)$$

Збіжність $y^{u^*+\varepsilon v} \rightarrow y^{u^*}$ у просторі $L^\infty(0, T; L^1(0, A))$ при $\varepsilon \rightarrow 0+$ випливає з теореми 8.5.3.

Для завершення доведення теореми нам необхідне твердження, яке сформулюємо та доведемо у вигляді леми.

Лема 8.5.1. *Якщо z – розв’язок задачі*

$$\begin{cases} Dz(a, t) + \mu(a, t)z(a, t) = -vy^{u^*} - u^*z, & (a, t) \in Q_T, \\ z(0, t) = \int_0^A \beta(a, t)z(a, t)da, & t \in (0, T), \\ z(a, 0) = 0, & a \in (0, A), \end{cases} \quad (8.5.23)$$

тоді справедлива збіжність:

$$\frac{1}{\varepsilon}[y^{u^*+\varepsilon v} - y^{u^*}] \rightarrow z \text{ у } L^\infty(0, T; (L^1(0, A))), \text{ при } \varepsilon \rightarrow 0+.$$

Доведення леми. Існування та єдиність розв’язку задачі (8.5.23) випливає з теореми 8.5.1. Уведемо позначення

$$w_\varepsilon(a, t) = \frac{1}{\varepsilon}[y^{u^*+\varepsilon v}(a, t) - y^{u^*}(a, t)] - z(a, t), \quad (a, t) \in Q_T.$$

Неважко переконатися, що w_ε є розв’язком задачі

$$\begin{cases} Dw(a, t) + \mu(a, t)w(a, t) = -u^*w - v[y^{u^*+\varepsilon v} - y^{u^*}], & (a, t) \in Q_T, \\ w(0, t) = \int_0^A \beta(a, t)w(a, t)da, & t \in (0, T), \\ w(a, 0) = 0, & a \in (0, A). \end{cases}$$

Позаяк $y^{u^*+\varepsilon v} - y^{u^*} \rightarrow 0$ у $L^\infty(0, T; L^1(0, A))$ при $\varepsilon \rightarrow 0+$ та використавши результат теореми 8.5.3, одержимо, що $w_\varepsilon \rightarrow 0$ у просторі $L^\infty(0, T; L^1(0, A))$ при $\varepsilon \rightarrow 0+$, що і завершує доведення леми. \square

Доведення теореми 8.5.6, (продовження). Виконавши граничний перехід у (8.5.22), одержимо

$$\int_0^T \int_0^A (u^*z + vy^{u^*})da dt \leq 0. \quad (8.5.24)$$

Помножимо (8.5.20)₁ на z та проінтегрувавши у прямокутнику $[0, T] \times [0, A]$, одержимо

$$\int_0^T \int_0^A (Dp - \mu p)z da dt = \int_0^T \int_0^A [u^*(1+p)z - \beta(a, t)p(0, t)z(a, t)] da dt.$$

Використавши (8.5.20) і (8.5.23)₂ та після очевидних перетворень, одержимо

$$- \int_0^T \int_0^A p(Dz + \mu z) da dt = \int_0^T \int_0^A u^*(1+p)z da dt. \quad (8.5.25)$$

Використовуючи (8.5.25) можна записати

$$\int_0^T \int_0^A pvy^{u^*} da dt = \int_0^T \int_0^A u^*z da dt,$$

а отже, з (8.5.24) випливає, що

$$\int_0^T \int_0^A v(a, t)(1 + p(a, t))y^{u^*}(a, t) da dt \leq 0,$$

для довільної функції $v \in L^\infty(Q_T)$ такої, що $u^* + \varepsilon v \in K$ для достатньо малого значення $\varepsilon > 0$. Міркуваннями, аналогічними як у розділі 8.3., одержимо

$$u^*(a, t) = \begin{cases} 0, & \text{якщо } (1 + p(a, t))y^{u^*}(a, t) < 0, \\ L, & \text{якщо } (1 + p(a, t))y^{u^*}(a, t) > 0. \end{cases}$$

Оскільки за припущенням $f(a, t) > 0$ м.в. у Q_T , то $y^{u^*}(a, t) > 0$ м.в. у Q_T , а отже, правильне подання (8.5.21). Теорема доведена. \square

♣ Зауваження 8.5.5. Як наслідок із теореми 8.5.6 одержуємо, що p (розв'язок задачі (8.5.20)) є розв'язком задачі

$$\begin{cases} Dp - \mu p = L(1+p)^+ - \beta(a, t)p(0, t), & (a, t) \in Q_T, \\ p(A, t) = 0, & t \in (0, T), \\ p(a, T) = 0, & a \in (0, A). \end{cases} \quad (8.5.26)$$

Детальніше, враховуючи єдиність, див. [13] і [3]. Щодо єдиності оптимального керування – у наступному зауваженні.

♣ **Зауваження 8.5.6.** Припустимо, що u^* є оптимальним керуванням у задачі (ОНР). Нехай, крім того, $f(a, t) > 0$ м.в. на Q_T та

$$\left\{ \begin{array}{l} \mu(a, t) > 0 \text{ м.в. на } Q_T, \\ \text{і, д.м.в. } t \in (0, T), \frac{\beta}{\mu}(\cdot, t) \text{ не є строго додатною сталою} \\ \text{на кожній підмножині з додатною вимірністю.} \end{array} \right. \quad (\text{H5})$$

За цих додаткових умов із рівняння (8.5.20)₁ випливає, що

$$\mathcal{D} = \{(a, t) \in Q_T : p(a, t) = -1\}$$

є множиною міри нуль за Lebesgue (p є розв'язком задачі (8.5.20)) та u^* є двопозиційним керуванням (u^* набуває всюди скінчену кількість значень). Більше того, оптимальне керування єдине.

♣ **Зауваження 8.5.7.** За виконання припущення зауваження 8.5.2., розв'язок p двоїстої задачі (8.5.26) не залежить або від f , або від y_0 .

Позначимо через u^* оптимальне керування. Останнє зауваження дає змогу сформулювати такий результат для загального випадку $f(a, t) \geq 0$ м.в. на Q_T .

Теорема 8.5.7. При виконанні умов (H1)-(H4) керування u^* також є оптимальним для задачі (ОНР) у випадку невід'ємності функції f .

Доведення. Розглянемо послідовність $\{f_n\}$ функцій $f_n \in L^1(Q_T)$ таких, що $f_n(a, t) > 0$ м.в. у Q_T , причому $f_n \rightarrow f$ у просторі $L^1(Q_T)$ при $n \rightarrow +\infty$. Через $\Psi_n(u)$ позначимо функцію кошту у задачі (ОНР), що відповідає $f := f_n$ та через y_n^u розв'язок задачі (8.5.4), що відповідає керуванню u і $f := f_n$.

Теорема 8.5.3 дає підстави стверджувати, що для довільного керування $u \in K$

$$y_n^u \rightarrow y^u$$

у просторі $L^\infty(0, T; L^1(0, A))$ при $n \rightarrow +\infty$. Звідси випливає

$$\Psi_n(u) \rightarrow \Psi(u) \quad \text{якщо } n \rightarrow +\infty.$$

Позаяк для довільного керування $u \in K$ правильна нерівність $\Psi_n(u^*) \geq \Psi_n(u)$ / тому беручи до уваги останню збіжність, одержуємо, що u^* є

оптимальним керуванням задачі (ОНР).

□

♣ **Зауваження 8.5.8.** Якщо виконуються умови (Н1)-(Н4), то існує єдине оптимальне керування u^* задачі (ОНР). Для визначення u^* необхідно знайти розв'язок p задачі (8.5.26). Тоді оптимальне керування визначається за формулою (8.5.21).

Чисельний розв'язок

За теоремою 8.5.7 та задачі (8.5.26) із її наслідку одержуємо простий спосіб обчислення наближеного розв'язку задачі оптимального керування. Опишемо алгоритм розв'язування задачі.

ОН1: Обчислимо розв'язок p задачі (8.5.26), тобто задачі

$$\begin{cases} Dp - \mu p = L(1 + p)^+ - \beta(a, t)p(0, t), & (a, t) \in Q_T, \\ p(A, t) = 0, & t \in (0, T), \\ p(a, T) = 0, & a \in (0, A). \end{cases}$$

ОН2: Обчислимо оптимальне керування u^* за формулою (8.5.21)

Задача (ОН1) розв'язується у напрямі спадання за часовою змінною. Багатократно застосовуючи обернену скінчено-різницеву апроксимацію для диференціального оператора, ми дискредитуємо перше рівняння (для p) неявно так:

$$\frac{p_i^{(j)} - p_{i-1}^{(j-1)}}{h} - \mu_i p_i^{(j)} = L\Pi(p_i^{(j)}) - \beta_i p_1^{(j)}. \quad (8.5.27)$$

де

$$\Pi(p) = (1 + p)^+.$$

Як наслідок одержуємо

$$p_{i-1}^{(j-1)} = (1 - h\mu_i)p_i^{(j)} - h \left[L\Pi(p_i^{(j)}) - \beta_i p_1^{(j)} \right].$$

Цикл для обчислення p , відповідно, має вигляд

```

for j=1 to N
  p(M,j)=0;
end-for
for i=1 to M
  p(i,N)=0;
end-for
for j=N down to 2
  for i=2 to M
    temp=h*(L*pp(p(i,j))-beta(i)*p(1,j));
    p(i-1,j-1)=(1-h*mu(i))*p(i,j)-temp;
  end-for
end-for

```

Тут `pp` – додатна частина функції Π і у машинному кодї визначається за допомогою файл-функції середовища `MatLab`[®]:

```

function out=pp(p)
p1=p + 1;
if (p1>= 0)
  out=p1;
else
  out=0;
end

```

На жаль, що стосується чисельних тестів, то точний метод має недолік. У наших чисельних експериментах розглянемо функцію

$$\mu(a) = \frac{e^{-a}}{A - a},$$

яку (див. попередню програму) у середовищі `MatLab`[®] визначимо

```
mu=exp(-a)/(A-a);
```

Очевидно, що значення `mu(M)` невизначене і `MatLab`[®] поверне значення `Inf`, тобто $+\infty$. Однак, як видно з записаного вище алгоритму, у циклі `for i to M` обчислюється значення `mu(i)`. Тому ми змушені присвоїти значення `NaN` (Not a Number) для багатьох значень p .

Тому, замінивши μ_i на μ_{i-1} , одержимо напівявну формулу

$$\frac{p_i^{(j)} - p_{i-1}^{(j-1)}}{h} - \mu_{i-1} p_{i-1}^{(j-1)} = L\Pi(p_i^{(j)}) - \beta_i p_1^{(j)},$$

звідки

$$p_{i-1}^{(j-1)} = (1 + h\mu_{i-1})^{-1} \left[p_i^{(j)} - h \left(L\Pi(p_i^{(j)}) - \beta_i p_1^{(j)} \right) \right].$$

Запишемо відповідний алгоритм.

Алгоритм 8.3.

```

for j=1 to N
  p(M,j)=0
end-for
for i=1 to M
  p(i,N)=0
end-for
for i=N downto 2
  for i=2 to M
    temp=h*(L*pp(p(i,j))-beta(i)*p(1,j))
    p(i-1,j-1)=(p(i,j)-temp)/(1+h*mu(i))
  end-for
end-for
for i=1 to M
  for j=1 to N
    if (1+p(i,j)>0)
      u(i,j)=L
    else
      u(i,j)=0
    end-if
  end-for
end-for

```

Відповідна програма запишеться, наприклад, так:

```

% file ohp1.m
% Program for optimal harvesting - age dependent population
% (semi-implicit method)
clear all
A=input('The maximal age is A=');
T=input('The final time is T=');
h=input('The discretization parameter is h=');
L=input('L=');
ct=input('constant for beta: '); % constant B>0 for beta
M=1+A/h;
N=1+T/h;
N1=N+1;
for i=1:M
  a(i)=(i-1)*h;
end
for j=1:N
  t(j)=(j-1)*h;

```

```
end
beta=ct*a.^2.*(A-a).*(1+sin(pi/A*a));
mu=exp(-a)/(A-a);
p=zeros(M,N);
u=zeros(M,N);
for j=1:N
    p(M,j)=0;
end
for i=1:M
    p(i,N)=0;
end
for k=1:N-1
    j=N1-k;
    for i=2:M
        temp=h*(L*pp(p(i,j))-beta(i)*p(1,j));
        p(i-1,j-1)=(p(i,j)-temp)/(1+h*mu(i-1));
    end
end
for i=1:M
    for j=1:N
        temp=1+p(i,j);
        if (temp>0)
            u(i,j)=L;
        end
    end
end
for i=1:M
    for j=1:N
        age(i,j)=a(i);
        time(i,j)=t(j);
    end
end
% make figures
K=max(max(u));
meshz(age,time,u);
axis([0 A 0 T 0 K]);
xlabel('\bf {Age a}','FontSize',16)
ylabel('\bf {Time t}','FontSize',16)
zlabel('\bf {control u}','FontSize',16)
figure(2)
```

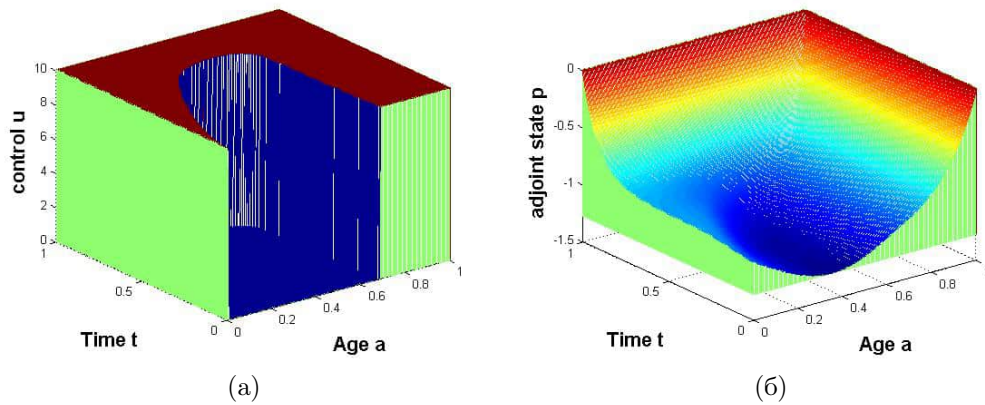


Рис. 8.33. Графіки програми `ohp1.m`: *a* – графік оптимального керування u^* ; *б* – графік розв’язку двоїстої задачі p ;

```

meshz(age,time,p)
xlabel('\bf{Age a}','FontSize',16);
ylabel('\bf{Time t}','FontSize',16);
zlabel('\bf{adjoint state p}','FontSize',16);
tl=input('time level: ');
tval=t(tl)
for i=1:M
w(i)=u(i,tl);
end
figure(3)
plot(a,w,'ks'); grid
axis([0 1 -2 12])
xlabel('\bf{a}','FontSize',16)
w1=w';
save w1.txt w1 -ascii

```

Тут `pp` функція, яка визначена вище. Масив `w` слугує для побудови графіка відображення $a \mapsto u(a, t)$ на кроці `tl` за часом, що відповідає значенню `t` визначеному за допомогою `tval` у програмі.

Тестування програми `ohp1.m` проведені за таких значень параметрів моделі: $A=1$, $T=1$, $L=10$, $h=0.005$, $B=10$ (константа для функції β). Графіки оптимального керування u та розв’язку p двоїстої задачі зображені на рис. 8.33а, 8.33б, відповідно.

Вибравши крок часу `tl=81`, одержимо вектор `w`, що відповідає зна-

ченню часу $t=0.4$. Зріз для керування u зображений на рис. 8.34.

Розглянемо тепер інший непрямий метод дискредитації рівняння (8.5.26). Для цього формулу (8.5.27) замінимо такою:

$$\frac{p_i^{(j)} - p_{i-1}^{(j-1)}}{h} - \mu_{i-1} p_{i-1}^{(j-1)} = L\Pi \left(p_{i-1}^{(j-1)} \right) - \beta_{i-1} p_1^{(j-1)},$$

й одержимо для $p_{i-1}^{(j-1)}$ рівняння

$$a_1 p_{i-1}^{(j-1)} + a_2 \Pi \left(p_{i-1}^{(j-1)} \right) - a_3 = 0,$$

де

$$\begin{aligned} a_1 &= 1 + h\mu_{i-1}, \\ a_2 &= hL, \\ a_3 &= p_i^{(j)} + h\beta_{i-1} p_1^{(j-1)}. \end{aligned}$$

Звідси одержуємо, що $p_{i-1}^{(j-1)}$ є розв'язком такого рівняння

$$a_1 x + a_2 (x + 1)^+ - a_3 = 0.$$

Елементарні обчислення дають два можливі розв'язки цього рівняння, а саме

$$x_1 = \frac{a_3 - a_2}{a_1 + a_2} \geq -1,$$

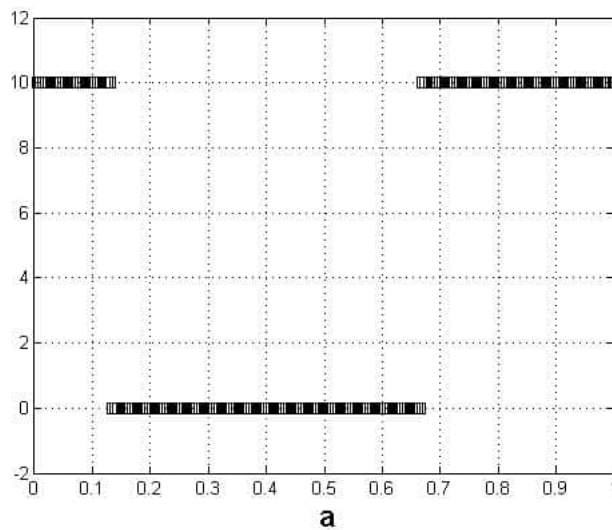


Рис. 8.34. Графік оптимального керування при $t = 0.4$

та

$$x_2 = \frac{a_3}{a_1} < -1.$$

Виберемо один із них і запишемо відповідний алгоритм.

Алгоритм 8.4

```

for j=1 to N
  p(M,j)=0
end-for
for i=1 to M
  p(i,N)=0
end-for
a2=h*L;
for j=N downto 2
  for i=2 to M
    a1=1+h*mu(i-1);
    a3=p(i,j) + h*beta(i-1)*p(1,j-1);
    x1=(a3-a2)/(a1+a2);
    x2=a3/a1;
    if (x1<-1) and (x2>=-1)
      error('NO ROOT');
    end-if
    if (x2<-1)
      p(i-1,j-1)=x2;
    else
      p(i-1,j-1)=x1;
    end-if
  end-for
end-for
end-for

```

/* u is computed as in Algorithm 8.3 */

Зауважимо, що порівняно з алгоритмом 8.3 змінено тільки for i=2 to M всередині циклу.

У наступному розділі дослідимо складнішу модель.

8.5.3. Логістична модель із періодичним показником природної зміни популяції

У цьому розділі дослідимо оптимальну задачу приросту з віково структурованою динамікою популяції із логістичним членом та періодич-

ним показником природної зміни популяції. Відправною точкою є така нелінійна модель із віково структурованою динамікою популяції

$$\begin{cases} Dy(a, t) + \mu(a, t)y + \mathcal{M}(t, Y(t))y = -u(t)y, & (a, t) \in Q, \\ Y(t) = \int_0^A y(a, t) da, & t > 0, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t) da, & t > 0, \\ y(a, 0) = y_0(a), & a \in (0, A), \end{cases} \quad (8.5.28)$$

де $Q = (0, A) \times (0, +\infty)$. Демографічний показник β (показник народжуваності) і μ (коефіцієнт смертності), залежні від віку та часу, припускаються T -періодичними за змінною t . $Y(t)$ означає щільність загальної популяції у момент часу t і $\mathcal{M}(t, Y(t))$ означає додатковий коефіцієнт смертності пов'язаний із перенаселенням. T -періодична функція u означає зусилля спрямовані на розмноження (керування). Зауважимо, що (8.5.28) є крайовою задачею. У [6] і [4] за певних умов доведено, що розв'язок y^u задачі (8.5.28) має властивість

$$\lim_{t \rightarrow +\infty} \|y^u(t) - \tilde{y}^u(t)\|_{L^\infty(0, A)} = 0,$$

де \tilde{y}^u є максимальним невід'ємним періодичним розв'язком такої задачі

$$\begin{cases} Dy(a, t) + \mu(a, t)y + \mathcal{M}(t, Y(t))y = -u(t)y, & (a, t) \in Q, \\ Y(t) = \int_0^A y(a, t) da, & t > 0, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t) da, & t > 0, \\ y(a, t) = y(a, t + T), & a \in (0, A). \end{cases} \quad (8.5.29)$$

Відомо, що задача (8.5.29) має щонайменше два невід'ємні розв'язки ($y \equiv 0$ звичайно є невід'ємним розв'язком задачі).

У цьому контексті дослідимо задачу знаходження T -періодичного зусилля направлено на розмноження u , яка приведе нас до максимальної народжуваності на часовому інтервалі $[t, t + T]$ при $t \rightarrow +\infty$.

Беручи до уваги, що для кожного додатного початкового значення

y_0 ,

$$\int_t^{t+T} \int_0^A u(s) y^u(a, s) da ds \rightarrow \int_0^T \int_0^A u(s) \tilde{y}^u(a, s) da ds,$$

при $t \rightarrow +\infty$ (загальний приріст популяції на часовому проміжку $[t, t+T]$, що описується (8.5.28), збіжний до загального приросту популяції на часовому проміжку довжини T за періодичної популяції, що описується (8.5.29)), задачу можна сформулювати так: максимізувати

$$\int_0^T \int_0^A u(s) \tilde{y}^u(a, s) da ds \rightarrow \max, \quad (\text{ОН})$$

стосовно $u \in K$, де множина керувань K визначена так:

$$K = \{v \in L^\infty(\mathbb{R}^+) : 0 \leq v(t) \leq L, v(t) = v(t+T) \text{ м.в. в } \mathbb{R}^+\},$$

і $L \in (0, +\infty)$.

Нехай виконуються такі умови (див. [6] і [4]):

$$\begin{aligned} & \beta \in C(\mathbb{R}^+; L^\infty(0, A)), \\ (\text{Нур1}) \quad & \beta(a, t) \geq 0, \beta(a, t) = \beta(a, t+T) \text{ д.м.в. } (a, t) \in Q, \\ & \exists \delta, \tau > 0, \exists a_0 \in (0, A) : a_0 + T \leq A, \beta(a, \tau) \geq \delta \\ & \text{д.м.в. } a \in (a_0, a_0 + T). \end{aligned}$$

Останнє припущення щодо β означає, що віковий період репродукції популяції виду є довшим або дорівнює періоду T .

$$\begin{aligned} (\text{Нур2}) \quad & \mu \in C(\mathbb{R}^+; L^\infty(0, \tilde{a})) \quad \forall \tilde{a} \in (0, A), \\ & \mu(a, t) \geq 0, \mu(a, t) = \mu(a, t+T) \text{ д.м.в. } (a, t) \in Q. \end{aligned}$$

$$(\text{Нур3}) \quad y_0 \in L^\infty(0, A), y_0(a) > 0 \text{ м.в. на } (0, A).$$

(Нур4) $\mathcal{M} : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ є неперервною функцією, неперервно диференційовною за другою змінною і похідна \mathcal{M}_Y додатна на $\mathbb{R}^+ \times \mathbb{R}^+$. Крім того,

$$\mathcal{M}(t, Y) = \mathcal{M}(t+T, Y) \quad \forall t, Y \in \mathbb{R}^+,$$

$$\mathcal{M}(t, 0) = 0 \quad t \in \mathbb{R}^+,$$

$$\lim_{Y \rightarrow +\infty} \mathcal{M}(t, Y) = +\infty \text{ рівномірно за } t.$$

Під розв'язком задачі (8.5.28) розуміємо абсолютно неперервну вздовж характеристики (рівняння якої $a - t = k$, $(a, t) \in Q$, $k \in \mathbb{R}$)

функцію $y^u \in L^\infty(0, \tilde{T}; (0, A))$ ($\forall \tilde{T} > 0$), яка задовольняє

$$\left\{ \begin{array}{l} Dy^u(a, t) = -\mu(a, t)y^u(a, t) - \mathcal{M}(t, Y(t))y^u(a, t) - u(t)y^u(a, t), \\ \qquad \qquad \qquad \text{д.м.в. } (a, t) \in Q, \\ Y^u(t) = \int_0^A y^u(a, t) da, \\ \qquad \qquad \qquad \text{д.м.в. } t > 0, \\ y^u(0, t) = \int_0^A \beta(a, t)y^u(a, t) da, \\ \qquad \qquad \qquad \text{д.м.в. } t > 0, \\ y^u(a, 0) = y_0(a), \\ \qquad \qquad \qquad \text{д.м.в. } a \in (0, A). \end{array} \right.$$

Під розв'язком задачі (8.5.29) розуміємо абсолютно неперервну вздовж характеристики $a - t = k$ ($(a, t) \in Q$, $k \in \mathbb{R}$) функцію \tilde{y}^u , яка задовольняє систему

$$\left\{ \begin{array}{l} D\tilde{y}^u(a, t) = \mu(a, t)\tilde{y}^u(a, t) - \mathcal{M}(t, \tilde{Y}^u(t))\tilde{y}^u(a, t) - u(t)\tilde{y}^u(a, t), \\ \qquad \qquad \qquad \text{д.м.в. } (a, t) \in Q, \\ \tilde{Y}^u(t) = \int_0^A \tilde{y}^u(a, t) da, \\ \qquad \qquad \qquad \text{д.м.в. } t > 0, \\ \tilde{y}^u(0, T) = \int_0^A \beta(a, t)\tilde{y}^u(a, t) da, \\ \qquad \qquad \qquad \text{д.м.в. } t > 0, \\ \tilde{y}^u(a, 0) = \tilde{y}^u(a, t + T), \\ \qquad \qquad \qquad \text{д.м.в. } (a, t) \in Q. \end{array} \right.$$

Із ергодичного результату, отриманого у [69], випливає існування єдиної пари $(\alpha, y^*) \in \mathbb{R} \times C(\mathbb{R}^+; L^\infty(0, A))$ такої, що y^* є невід'ємним розв'язком задачі

$$\left\{ \begin{array}{l} Dy^*(a, t) + \mu(a, t)y^* + \alpha y^* = 0, \quad (a, t) \in Q, \\ y^*(0, t) = \int_0^A \beta(a, t)y^*(a, t) da, \quad t > 0, \\ y^*(a, t) = y^*(a, t + T), \quad (a, t) \in Q, \end{array} \right. \quad (8.5.30)$$

який задовольняє

$$\text{Ess sup}\{y^*(a, t) : (a, t) \in Q\} = 1.$$

Крім того, y^* є додатна на $(0, A) \times (0, +\infty)$. Множина розв'язків (8.5.30) утворює одномірний лінійний простір.

Сформулюємо теорему про асимптотичний характер поведінки розв'язку y^u задачі (8.5.28) (див. [13, 48]).

Теорема 8.5.8. *Для довільного керування $u \in K$ задача (8.5.28) має єдиний розв'язок y^u такий, що*

$$\lim_{t \rightarrow +\infty} \|y^u(t) - \tilde{y}^u\|_{L^\infty(0,A)} = 0,$$

де \tilde{y}^u – максимальний невід'ємний розв'язок задачі (8.5.29).

Крім того,

(i) якщо $T\alpha > \int_0^T u(t)dt$, тоді \tilde{y}^u є єдиним нетривіальним невід'ємним розв'язком (8.5.29);

(ii) якщо $T\alpha \leq \int_0^T u(t)dt$, тоді $\tilde{y}^u \equiv 0$ є єдиним невід'ємним розв'язком (8.5.29).

♣ **Зауваження 8.5.9.** Якщо $T\alpha > \int_0^T u(t)dt$, тоді $\tilde{y}^u(a, t) = c_0 y^*(a, t) h^u(t)$, $(a, t) \in Q$, де $c_0 \in (0, +\infty)$ – довільна стала і h^u – єдиний нетривіальний розв'язок задачі

$$\begin{cases} h'(t) + \mathcal{M}(t, Y_0^*(t)h(t))h(t) - \alpha h(t) = -u(t)h(t), & t \in \mathbb{R}^+, \\ h(t) = h(t+T), & t \in \mathbb{R}^+, \end{cases} \quad (8.5.31)$$

де $Y_0^*(t) = c_0 \int_0^A y^*(a, t) da$, $t > 0$.

Очевидно, що h^u через Y_0^* залежить від c_0 . Отже, $c_0 h^u$ не залежить від c_0 .

♣ **Зауваження 8.5.10.** Результат Теорема 8.5.8 дає змогу апроксимувати максимальний невід'ємний розв'язок (8.5.29), взявши за нульове наближення розв'язок задачі (8.5.28) (див. алгоритм (NSM) у кінці цього розділу).

Задача оптимального розмноження

Із Теорема 8.5.8 одержуємо, що для довільного $u \in K$

$$\int_0^T \int_0^A u(t) \tilde{y}^u(t) da dt = \int_0^T u(t) Y_0^*(t) h^u(t) dt,$$

Найперше доведемо, що розв'язок q задачі (8.5.32) єдиний. Використовуючи $\gamma(t)$, задачу (8.5.32) можна записати у вигляді

$$\begin{cases} q'(t) = -\gamma(t)q(t) + \mathcal{M}_Y(t, Y_0^*(t)h^*(t))Y_0^*(t)h^*(t)q(t) + \\ \quad u^*(t)Y_0^*(t), & t \in \mathbb{R}^+, \\ q(t) = q(t+T), & t \in \mathbb{R}^+. \end{cases}$$

Звідси $\forall t \in \mathbb{R}^+$ розв'язок q задовольняє рівність

$$q(t) = q(0) \exp \left\{ \int_0^t [-\gamma(s) + \mathcal{M}_Y(s, Y_0^*(s)h^*(s))Y_0^*(s)h^*(s)] ds \right\} + \\ \int_0^t u^*(s)Y_0^*(s) \exp \left\{ \int_s^t [-\gamma(\theta) + \mathcal{M}_Y(\theta, Y_0^*(\theta)h^*(\theta))Y_0^*(\theta)h^*(\theta)] d\theta \right\} ds.$$

Використовуючи умову $q(0) = q(T)$, з останньої рівності $\forall t \in \mathbb{R}^+$ одержимо

$$q(0) = q(0) \exp \left\{ \int_0^T [-\gamma(s) + \mathcal{M}_Y(s, Y_0^*(s)h^*(s))Y_0^*(s)h^*(s)] ds \right\} + \\ \int_0^T u^*(s)Y_0^*(s) \exp \left\{ \int_s^T [-\gamma(\theta) + \mathcal{M}_Y(\theta, Y_0^*(\theta)h^*(\theta))Y_0^*(\theta)h^*(\theta)] d\theta \right\} ds,$$

і, відповідно,

$$q(0) = \left(1 - \exp \left\{ \int_0^T [-\gamma(s) + \mathcal{M}_Y(s, Y_0^*(s)h^*(s))Y_0^*(s)h^*(s)] ds \right\} \right)^{-1} \times \\ \int_0^T u^*(s)Y_0^*(s) \exp \left\{ \int_s^T [-\gamma(\theta) + \mathcal{M}_Y(\theta, Y_0^*(\theta)h^*(\theta))Y_0^*(\theta)h^*(\theta)] d\theta \right\} ds < 0$$

(що випливає з припущення щодо \mathcal{M} та додатності h^* і Y_0^*). Одержана T -періодична функція q є єдиним розв'язком задачі (8.5.32).

Розглянемо тепер довільну T -періодичну функцію $v \in L^\infty(\mathbb{R}^+)$ так, що для довільного достатньо малого значення $\varepsilon > 0$ $u^* + \varepsilon v \in K$. Очевидно, що за достатньо малого $\varepsilon > 0$ правильна нерівність $T\alpha >$

$\int_0^T u^*(t)dt + \varepsilon \int_0^T v(t)dt$. Позаяк u^* є оптимальним керуванням задачі (ОН), то правильна нерівність

$$\int_0^T u^*(t)Y_0^*(t)h^*(t)dt \geq \int_0^T (u^*(t) + \varepsilon v(t))Y_0^*(t)h^{u^*+\varepsilon v}(t)dt,$$

із якої одержуємо

$$\int_0^T u^*(t)Y_0^*(t) \frac{h^{u^*+\varepsilon v}(t) - h^*(t)}{\varepsilon} dt + \int_0^T v(t)Y_0^*(t)h^{u^*+\varepsilon v}(t)dt \leq 0,$$

за довільного достатньо малого значення $\varepsilon > 0$.

Сформулюємо допоміжне твердження у вигляді леми, яке доводиться аналогічно як у попередньому розділі.

Лема 8.5.2. *Виконуються такі граничні переходи*

$$h^{u^*+\varepsilon v} \rightarrow h^* y C([0, T]),$$

$$\frac{h^{u^*+\varepsilon v} - h^*}{\varepsilon} \rightarrow zy C([0, T]),$$

при $\varepsilon \rightarrow 0+$, де z є розв'язком задачі

$$\begin{cases} z'(t) = -\gamma(t)z(t) - \mathcal{M}_Y(t, Y_0^*(t)h^*(t))Y_0^*(t)h^*(t)z(t) - v(t)h^*(t), & t \in \mathbb{R}^+, \\ z(t) = z(t+T), & t \in \mathbb{R}^+. \end{cases}$$

Виконавши граничний перехід ($\varepsilon \rightarrow 0+$) в останній нерівності та використавши лему 8.5.2, одержимо нерівність

$$\int_0^T Y_0^*(t)[u^*(t)z(t) + v(t)h^*(t)]dt \leq 0.$$

Домножимо рівняння (8.5.32)₁ на z та після інтегрування на проміжку $[0, T]$ одержимо таку рівність

$$\begin{aligned} \int_0^T q'(t)z(t)dt &= \int_0^T z(t) [-\gamma(t)q(t) + \\ &\mathcal{M}_Y(t, Y_0^*(t)h^*(t))Y_0^*(t)h^*(t)q(t) + u^*(t)Y_0^*(t)] dt. \end{aligned}$$

Позаяк

$$\int_0^T q'(t)z(t)dt = - \int_0^T q(t)z'(t)dt =$$

$$\int_0^T q(t) [-\gamma(t)z(t) + \mathcal{M}_Y(t, Y_0^*(t)h^*(t))Y_0^*(t)h^*(t)z(t) + v(t)h^*(t)] dt,$$

то звідси одержимо

$$\int_0^T u^*(t)Y_0^*(t)z(t)dt = \int_0^T v(t)h^*(t)q(t)dt.$$

Це дає нам підставу стверджувати, що

$$\int_0^T v(t)h^*(t)(Y_0^*(t) + q(t))dt \leq 0,$$

для кожної T -періодичної функції $v \in L^\infty(\mathbb{R}^+)$ такої, що $u^* + \varepsilon v \in K$ і $T\alpha > \int_0^T u^*(t)dt + \varepsilon \int_0^T v(t)dt$ за довільного достатньо малого $\varepsilon > 0$. Із останнього співвідношення випливає (8.5.33). Теорема доведена. \square

Чисельні алгоритми

Розглянемо тепер концептуальні алгоритми апроксимації розв'язку оптимальної задачі народжуваності (**ОН**). Основу алгоритмів становлять умови оптимальності, сформульовані у теоремі 8.5.9.

Опишемо спочатку *метод чисельної стабілізації (Numerical Stabilization Method (NSM))* для періодичної задачі. Розглянемо таку загальну задачу. Знайти єдиний розв'язок $x \in X$ задачі

$$\begin{cases} (Fx)(a, t) = z(a, t), & (a, t) \in Q, \\ x(a, t) = x(a, t + T), & (a, t) \in Q, \end{cases} \quad (\text{S})$$

де $z \in Z$ є T -періодичний розв'язок за змінною t і $F : X \rightarrow Z$ є заданий оператор, що відображає заданий простір X у заданий Z . Задача (S) є T -періодичною стосовно змінної t . Як зазначалося у попередньому

розділі, розв'язок такої задачі можна обчислити за допомогою початкової задачі

$$\begin{cases} (Fx)(a, t) = z(a, t), & (a, t) \in Q, \\ x(a, 0) = x_0(a), & a \in [0, A]. \end{cases} \quad (S_0)$$

Опишемо метод (NSM).

Алгоритм (NSM)

Step 0: Розв'язуємо задачу (S_0) для $t \in [0, T]$ та визначимо чисельний розв'язок $x^{(0)}$; прийmemo $k := 1$.

Step 1: Розв'язуємо задачу

$$\begin{cases} (Fx)(a, t) = z(a, t), & a \in [0, A), \quad t \in [kT, (k+1)T], \\ x(a, kT) = x^{(k-1)}(a), & a \in [0, A), \end{cases} \quad (S_k)$$

та визначаємо чисельний розв'язок $x^{(k)}$.

Step 2: Критерій зупинки:

якщо $\|x^{(k)} - x^{(k-1)}\| < \varepsilon$

тоді **stop** ($x^{(k)}$ є розв'язок)

у противному випадку $k := k + 1$; перехід на **Step 1**.

На кроці **Step 2** норма визначається апіорі, а ε є заданим параметром збіжності. На кожному часовому проміжку $[kT, (k+1)T]$ довжини T визначимо рівномірну сітку вузлів

$$kT = t_1 < t_2 < \dots < t_N = (k+1)T$$

й апроксимуємо відповідні значення $x^{(k)} = (x_i^{(k)})_i$, $i = 1, 2, \dots, N$, які означають $x_i^{(j)} \approx x^{(k)}(t_i)$ для $i = 1, 2, \dots, N$. Очевидно, кожне значення $x_i^{(k)}$ також необхідно апроксимувати за змінною a . Тому вище згадану норму можна визначити дискретно стосовно $x^{(k)} = (x_i^{(k)})_i$. Після завершення обчислення за алгоритмом (NSM) $x^{(k)}$ трактуються як T -періодичні розв'язки задачі (S).

Біологічні системи, які тут досліджуються, мають властивість стійкості, яка передбачена в алгоритмі (NSM). Незалежно від початкової функції x_0 розв'язок x є стійким після декількох часових інтервалів довжиною T .

Перейдемо до наступного алгоритму знаходження розв'язку оптимальної задачі народжуваності – *метод проєкції ґрадієнта* (*Projected Gradient Method* (PGM)). Взявши до уваги обмеження на керування, використаємо алгоритм Rosen'a (див. [15, р. 44]). У процесі обчислень також скористаємося алгоритмом (NSM) як підпрограмою. Для спрощення формул у задачі приймемо $M(t, Y) = Y$ та $c_0 = 1$. Зауважимо, що метод (PGM), який опишемо нижче, правильний для $\alpha > 0$.

Алгоритм методу проєкції ґрадієнта (PGM)

- **S0:** Обчислимо параметр α .

S0.0: Виберемо $y_0(a) > 0$ та, використавши алгоритм (NSM), обчислимо розв'язок такої задачі:

$$\left\{ \begin{array}{l} Dy(a, t) + \mu(a, t)y + Y(t)y(a, t) = 0, \quad (a, t) \in Q, \\ Y(t) = \int_0^A y(a, t) da, \quad t > 0, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t) da, \quad t > 0, \\ y(a, t) = y(a, t + T), \quad (a, t) \in Q. \end{array} \right.$$

$$\mathbf{S0.1:} \quad \alpha = \frac{1}{T} \int_{kT}^{(k+1)T} \int_0^A y(a, t) da dt, \text{ де } k \text{ одержуємо на другому}$$

кроці **Step 2** із алгоритму (NSM).

S0.2: Приймемо $j := 0$, $u^{(j)}(t) := u_0(t)$, де u_0 задана функція.

- **S1:** Використовуючи алгоритм (NSM), обчислимо розв'язок $y^{(j)}$ задачі

$$\left\{ \begin{array}{l} Dy(a, t) + \mu(a, t)y + Y(t)y(a, t) = -u^{(j)}(t)y(a, t), \quad (a, t) \in Q, \\ Y(t) = \int_0^A y(a, t) da, \quad t > 0, \\ y(0, t) = \int_0^A \beta(a, t)y(a, t) da, \quad t > 0, \\ y(a, t) = y(a, t + T), \quad (a, t) \in Q. \end{array} \right.$$

Тут $y^{(j)}$ відповідний \tilde{y}^u розв'язок задачі (8.5.29).

- **S2:** Використовуючи алгоритм (NSM), обчислимо розв'язок $h^{(j)}$ задачі

$$\begin{cases} h'(t) + \left(\int_0^A y^{(j)}(a, t) da \right) h(t) - \alpha h(t) = -u^{(j)}(t)h(t), & t \in \mathbb{R}^+, \\ h(t) = h(t + T), & t \in \mathbb{R}^+. \end{cases}$$

Тут $h^{(j)}$ відповідний розв'язок задачі (8.5.31).

- **S3:** Обчислимо $q^{(j)}$.

$$\mathbf{S3.1:} \quad Y_0^{(j)}(t) = \frac{1}{h^{(j)}(t)} \int_0^A y^{(j)}(a, t) da.$$

S3.2: Обчисливши насамперед $q(0)$, обчислимо розв'язок $q^{(j)}$ задачі

$$\begin{cases} q'(t) - 2 \left(\int_0^A y^{(j)}(a, t) da \right) q(t) + \alpha q(t) = \\ \qquad \qquad \qquad - u^{(j)}(t) \left(Y_0^{(j)}(t) + q(t) \right), \\ q(t) = q(t + T), \quad t \in \mathbb{R}^+. \end{cases}$$

Зауважимо, що $q^{(j)}$ є розв'язком задачі (8.5.32).

- **S4:** Обчислимо $w^{(j)}$ за формулою (порівняти з (8.5.33))

$$w^{(j)} = \begin{cases} 0, & \text{якщо } Y_0^{(j)}(t) + q^{(j)}(t) < 0, \\ L, & \text{якщо } Y_0^{(j)}(t) + q^{(j)}(t) > 0. \end{cases}$$

- **S5:** Обчислимо наближення керування $u^{(j+1)}$.

S5.1: Обчислимо розв'язок $\lambda_j \in [0, 1]$ задачі знаходження максимуму

$$\Phi(\lambda u^{(j)} + (1 - \lambda)w^{(j)}) \xrightarrow{\lambda \in [0, 1]} \max,$$

де Φ – функціонал кошту з (ОН).

S5.2: Обчислити наближене значення керування $u^{(j+1)}$ за формулою

$$u^{(j+1)} = \lambda_j u^{(j)} + (1 - \lambda_j)w^{(j)}.$$

- **S6:** Критерій зупинки.

Якщо $\|u^{(j+1)} - u^{(j)}\| < \varepsilon$,
тоді **stop** ($u^{(j+1)}$ є шуканий розв'язок),
у противному випадку: $j := j + 1$; перехід на **S1**.

Чисельні експерименти

Найперше звернемо увагу, що опукла комбінація двох двопозиційних керувань, які приймають тільки значення 0 або L , не обов'язково повертає ці два значення. Це слугує причиною, щоб замінити опуклу комбінацію $\lambda u^{(j)} + (1 - \lambda)w^{(j)}$ із **Step 5.1** алгоритму метода проєкції градієнта (PGM), скориставшись ідеєю К. Glashoff та Е. Sachs [41] (детальніше див. [15, pp.137–143]). Ідея полягає у виборі такої опуклої комбінації точок переключення $u^{(j)}$ та $w^{(j)}$, для якої одержуємо систему точок переключення для нового двопозиційного керування.

8.5.4. Завдання для самостійного опрацювання

◆ *Завдання 8.5.1.* Написати програму для алгоритму 8.5.1, взявши за зразок `pop1.m`.

✓ *Вказівка.* Для чисельних тестів взяти такі значення числових параметрів $A = 1$, $T = 1$, $h = 0.02$, $B = 10$, $lw = 4$. Для обидвох алгоритмів одержимо графіки для β , μ та y (рис. 8.32).

◆ *Завдання 8.5.2.* Написати програму для алгоритму 8.4, взявши за основу `pop1.m`.

✓ *Вказівка.* Для чисельних тестів взяти такі значення числових параметрів $A=1$, $T=1$, $L=100$, $h=0.01$, $B=10$ (мультиплікативна константа для функції `beta`), і `tl=71`, що відповідає `t=0.7` для кроку за часом.

◆ *Завдання 8.5.3.* Довести існування оптимального керування задачі **ОН** (див. [4]).

◆ *Завдання 8.5.4.* Використовуючи алгоритми (NSM) та (PGM), написати програму апроксимації оптимального значення загальної народжуваності. Використати такі дані:

$$\beta(a, t) = B \cdot a^2(1 - a)(1 + \sin(\pi a)) \left| \sin \frac{2\pi t}{T} \right|,$$

$$\mu(a, t) = \frac{e^{-4a}(2 + \cos \frac{2\pi t}{T})}{(1 - a)^{1.4}},$$

$$A = 1, \quad T = 0.5, \quad y_0(a) = 3, \quad L = 1.5, \quad B = \{50, 75, 100\}.$$

◆ *Завдання 8.5.5.* Обчислити та побудувати графік розв'язку задачі

$$\begin{cases} Dy(a, t) + \frac{a}{\pi - a}y(a, t) = t, & (a, t) \in (0, \pi) \times (0, 1), \\ y(0, t) = \int_0^\pi \sin(a) y(a, t) da, & t \in (0, 1), \\ y(a, 0) = 1, & a \in (0, \pi). \end{cases}$$

◆ *Завдання 8.5.6.* Вивести необхідні умови оптимальності першого порядку для такої загальної задачі:

$$\int_0^T \int_0^A u(a, t)g(a)y^u(a, t)da dt \rightarrow \max \quad (\text{ОНР-1})$$

стосовно $u \in K\{w \in L^\infty(Q_T) : 0 \leq w(a, t) \leq L \text{ м.в. у } Q_T\}$, де y^u є розв'язком задачі (8.5.18), L – додатна стала.

Припускаємо, що g задовольняє умови

$$g \in C^1([0, A]), \quad g(a) > 0 \quad \forall a \in [0, A].$$

Функція $g(a)$ означає вагу (кошт) особини віку a . Тому

$$\int_0^T \int_0^A u(a, t)g(a)y^u(a, t)da dt$$

визначає загальну вагу (кошт) народжуваності популяції.

✓ *Вказівка.* Позначити $z^u(a, t) = g(a)y^u(a, t)$ та сформулювати задачу (ОНР-1) стосовно u та z^u . Тоді можна скористатися з результатів отриманих у розділі 8.5.2.

◆ *Завдання 8.5.7.* Обчислити оптимальне зусилля й оптимальний кошт народжуваності популяції задачі (ОНР-1), взявши за β , μ та y_0 значення, як у розділі 8.5.2., і $g(a) = 1 + a$.

◆ *Завдання 8.5.8.* Вивести необхідні умови оптимальності першого порядку для такої задачі:

$$\int_0^T \int_0^A u(a, t)y^u(a, t)da dt \rightarrow \max \quad (\text{ОНР-2})$$

стосовно $u \in \mathcal{V} = \{w \in L^\infty(Q_T) : w_1(a, t) \leq w(a, t) \leq w_2(a, t) \text{ м.в. у } Q_T\}$, де y^u є розв'язком задачі (8.5.18). Тут $w_1, w_2 \in$

$L^\infty(Q_T)$, $0 \leq w_1(a, t) \leq w_2(a, t)$ м.в. у Q_T .

✓ *Вказівка.* Припустимо, що u^* – оптимальне керування задачі (ОНР-2), і спряжений стан є розв'язком задачі

$$\begin{cases} Dp + \mu p = u^*(1 + p) - \beta(a, t)p(0, t), & (a, t) \in Q_T, \\ p(A, t) = 0, & t \in (0, T), \\ p(a, T) = 0, & a \in (0, A). \end{cases}$$

Тоді

$$u^*(a, t) = \begin{cases} w_1(a, t), & \text{якщо } 1 + p(a, t) < 0, \\ w_2(a, t), & \text{якщо } 1 + p(a, t) > 0. \end{cases}$$

◆ *Завдання 8.5.9.* Дослідити задачу максимізації народжуваності $\int_0^A u(a)y^u(a, t)da$ при $t \rightarrow +\infty$ у випадку залежних від часу β і μ , керування належить класу

$$\mathcal{V} = \{v \in L^\infty(0, A) : 0 \leq v(a) \leq L \text{ д.м.в. } a \in (0, A)\}.$$

◆ *Завдання 8.5.10.* Дослідити задачу максимізації народжуваності

$$\int_t^{t+T} \int_0^A u(a, s)y^u(a, s)da ds$$

при $t \rightarrow +\infty$, якщо керування належить класу

$$\mathcal{V} = \{v \in L^\infty(Q) : 0 \leq v(a) \leq L, v(a, t) = v(a, t + T) \text{ д.м.в. } (a, t) \in Q\}.$$

Розділ 9

Оптимальне керування дифузійними моделями

Математична біологія має свої коріння в екології популяції, яка трактує математичне моделювання взаємодії видів визначених математиками А. Lotka (1924) та V. Volterra (1926) у вигляді нелінійних звичайних диференціальних рівнянь.

Головним у моделях типу Lotka-Volterra є припущення кількісного підходу до еволюції взаємодії популяцій у часі. Важливими є аспекти, якими не можна нехтувати на просторовій структурі, що є значимим для популяції.

У цьому розділі досліджуються дві задачі оптимального керування пов'язані з дифузійними моделями. Доведений принцип максимуму для такого типу задач та визначені чисельні алгоритми апроксимації оптимального значення функціонала кошту. Крім того, у кінці розділу у вигляді вправ для самостійного опрацювання сформульовано оптимальні задачі динаміки вікової та віково структурованої популяції.

Задачам оптимального керування дифузійними моделями присвячена численна література (див., наприклад, [53]). Зокрема, динамічні моделі популяції досліджені у [13]. Моделі епідемій досліджували у [28] та бібліографія роботи. Різноманітні моделі економіки, як і фізики та техніці описані у [19] і цитованій там літературі. Різноманітні задачі керування просторовими структурованими епідемологічними системами можна знайти у [7, 8, 9, 10].

Що стосується чисельних методів розв'язування задач оптимального керування для рівнянь у частинних похідних, то вони описані у праці [15].

9.1. Дифузія у математичних моделях

Після праць R. A. Fisher'a [36], A. N. Kolmogorov'a, I. G. Petrovskogo, N. S. Piskunov'a [52], J. Skellam'a [64] (див. також [65, 16]) математичне моделювання просторово структурованих популяцій були ретельно проаналізовані, поклали початок бурхливому потоку наукової літератури, присвячених так званих систем реакції-дифузії, в яких дифузія відповідної популяції додається до нелінійної динаміки їхньої взаємодії (див. [59, 57]).

Найперше пригадаємо два найважливіші та універсальні способи визначення дифузії: перший це тривіальний наслідок з закону збереження у поєднанні із відомим законом Fick'a, та другий завдяки відповідному масштабуванню у часі та просторі простого випадкового блукання.

Розглянемо поширення популяції чи дифузію деякої речовини, щільність якої у точці x та у момент часу t дорівнює $y(x, t)$ (тут $x \in \bar{\Omega} \subset \mathbb{R}^N$ – середовище поширення, $N \in \mathbb{N}^*$, $t \geq 0$); функція y така, що популяція у момент часу $t \geq 0$ у кожній області $V \subset \Omega$ (V є відкритою обмеженою підмножиною з гладкою межею, тобто ∂V належить до класу C^1) визначається за формулою

$$Y_V(t) = \int_V y(x, t) dx. \quad (9.1.1)$$

Нехай V – довільна підобласть із вище зазначеними властивостями. Припустимо, на деякий час, що y є гладкою функцією. Згідно з законом Fick'a "міграція популяції через точку $x \in \partial V$ дорівнює

$$J(x, t) = (\gamma(x, t) \nabla_x y(x, t)) \cdot \nu(x),$$

де $\nu(x)$ – зовнішня нормаль до ∂V у точці x ; $\gamma(x, t)$ – параметр дифузії; $\nabla_x y = y_x$ – градієнт y стосовно x ".

Якщо через $f(x, t)$ позначимо довільний приріст динаміки популяції у точці $x \in V$, $t \geq 0$, тоді закон збереження при зміні популяції в області V запишеться

$$Y'_V(t) = \int_{\partial V} J(x, t) d\sigma + \int_V f(x, t) dx.$$

Застосувавши теорему Stock'a та (9.1.1), одержимо

$$\int_V \left[\frac{\partial y}{\partial t}(x, t) - \operatorname{div}_x(\gamma(x, t) \nabla_x y(x, t)) - f(x, t) \right] dx = 0.$$

Для довільної підобласті V остання рівність справджується тоді і тільки тоді, якщо

$$\frac{\partial y}{\partial t}(x, t) - \operatorname{div}_x(\gamma(x, t)\nabla_x y(x, t)) - f(x, t) = 0. \quad (9.1.2)$$

У випадку сталої величини γ рівняння (9.1.2) запишеться у вигляді

$$\frac{\partial y}{\partial t}(x, t) - \gamma\Delta y(x, t) = f(x, t), \quad (9.1.3)$$

де $\Delta = \Delta_x$ – оператор Laplace'а стосовно x .

Рівняння Fisher'а є частковим випадком рівняння (9.1.3) для випадку однієї популяції та $f(x, t)$ описує процес народжуваності-смертності

$$\frac{\partial y}{\partial t}(x, t) - \gamma\Delta y(x, t) = -\mu y(x, t) + \beta y(x, t) \left(1 - \frac{y(x, t)}{k}\right). \quad (9.1.4)$$

Тут β та μ означають коефіцієнти природної народжуваності та смертності, відповідно, k – деяка додатна стала. Очевидно, це – логістичне рівняння (оскільки $(\beta/k)y(x, t)$ – додатковий коефіцієнт смертності, спричинений перенаселенням, пропорціональний до густини популяції).

Виведемо рівняння дифузії іншим способом, опираючись на прості випадкові події. Проста випадкова подія – це ланцюг Markov'а $(X_n) - n \in \mathbb{N}$ на зліченому просторі станів $E = \mathbb{Z}$, множині всіх цілих чисел, із матрицею переходу

$$\begin{aligned} \operatorname{prob}(X_{n+1} = k + 1 \mid X_n = k) &= p_{k, k+1} = p, \\ \operatorname{prob}(X_{n+1} = k - 1 \mid X_n = k) &= p_{k, k-1} = 1 - p, \\ \operatorname{prob}(X_{n+1} = j \mid X_n = k) &= p_{k, j} = 0, \end{aligned}$$

для кожного $k \in E$ та $j \in E$, $j \neq k - 1$, $j \neq k + 1$, де $p \in (0, 1)$. У симетричному випадку $p = 1/2$.

Якщо прийняти $p_k(n) = \operatorname{prob}(X_n = k)$, тоді з теореми про повну ймовірність і властивості ланцюгів Маркова

$$\begin{aligned} \operatorname{prob}(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) &= \\ \operatorname{prob}(X_{n+1} = j \mid X_n = i) &= p_{i, j}, \end{aligned}$$

для довільних $i, j \in E$ одержимо

$$p_k(n) = \frac{p_{k+1}(n) + p_{k-1}(n)}{2}.$$

Звідки

$$p_k(n+1) - p_k(n) = \frac{p_{k+1}(n) - 2p_k(n) + p_{k-1}(n)}{2}. \quad (9.1.5)$$

Якщо взяти $\Delta t \in \mathbb{R}_+^*$ як крок за часом, а $\Delta x \in \mathbb{R}_+^*$ як кроком за просторовою змінною, то рівняння (9.1.5) запишеться у вигляді

$$\begin{aligned} p_{k\Delta x}((n+1)\Delta t) - p_{k\Delta x}(n\Delta t) = \\ \frac{p_{(k+1)\Delta x}(n\Delta t) - 2p_{k\Delta x}(n\Delta t) + p_{(k-1)\Delta x}(n\Delta t)}{2}. \end{aligned}$$

Прийmemo $k\Delta x = x$, $n\Delta t = t$ та позначивши

$$p_{n\Delta t} := y(x, t),$$

одержимо еквівалентний запис

$$y(x, t + \Delta t) - y(x, t) = \frac{y(x + \Delta x, t) - 2y(x, t) + y(x - \Delta x, t)}{2}.$$

Застосувавши формулу Taylor'а, останню рівність перепишемо у вигляді

$$y(x, t + \Delta t) - y(x, t) = \frac{1}{2}(\Delta x)^2 \frac{\partial^2 y}{\partial x^2}(x, t) + o((\Delta x)^2),$$

з якої одержимо

$$\frac{y(x, t + \Delta t) - y(x, t)}{\Delta t} = \frac{1}{2} \frac{(\Delta x)^2}{\Delta t} \frac{\partial^2 y}{\partial x^2}(x, t) + \frac{o((\Delta x)^2)}{\Delta t}. \quad (9.1.6)$$

Виконавши граничний перехід $\Delta t \rightarrow 0$, $\Delta x \rightarrow 0$ так, що

$$\frac{(\Delta x)^2}{\Delta t} = 2\gamma = \text{constant},$$

із рівності (9.1.6) одержимо рівняння дифузії зі сталим коефіцієнтом дифузії

$$\frac{\partial y}{\partial t}(x, t) = \gamma \frac{\partial^2 y}{\partial x^2}(x, t), \quad x \in \mathbb{R}, \quad t > 0.$$

Додамо початкову умову

$$y(\cdot, 0) = \delta_0,$$

(функція Dirac'а у точці 0), тоді з теорії рівнянь із частинними похідними одержимо фундаментальний розв'язок

$$y(x, t) = \frac{1}{\sqrt{4\pi\gamma t}} \exp\left(-\frac{x^2}{4\gamma t}\right), \quad x \in \mathbb{R}, \quad t > 0,$$

який є функцією густини ймовірності Gauss'a $N(0, 2\gamma t)$, і означає граничний розподіл броунівського руху у момент часу t . Детальніше ці викладки можна знайти у працях [65, 59, 58].

До рівнянь (9.1.3), (9.1.4) та інших нелінійних параболічних рівнянь можна дописати стандартні граничні умови трьох типів.

- Умова Dirichlet:

$$y(x, t) = \varphi(x, t), \quad x \in \partial\Omega, \quad t \geq 0,$$

де φ – задана функція. Випадок $\varphi \equiv 0$ означає, що для біологічної популяції межа $\partial\Omega$ абсолютно непридатна для життя.

- Умова Neumann'a:

$$\frac{dy}{d\nu}(x, t) = \varphi(x, t), \quad x \in \partial\Omega, \quad t \geq 0,$$

де φ – задана функція. З біологічного погляду ця умова означає, що рух популяції через межу описується заданою функцією φ . Якщо $\varphi \equiv 0$, тоді немає руху популяції через границю, а отже, популяція ізольована в ареалі проживання Γ .

- Умова Robin'a:

$$\frac{\partial y}{\partial \nu}(x, t) + \alpha(x, t) = \varphi(x, t), \quad x \in \partial\Omega, \quad t \geq 0,$$

де α та φ відомі функції. Ця умова (для популяції) означає, що потік популяції через границю пропорціональний до різниці між густиною популяції $y(x, t)$ всередині ареалу проживання Ω та назовні Ω :

$$\frac{\partial y}{\partial \nu}(x, t) = -\alpha(x, t)(y(x, t) - \tilde{y}(x, t)), \quad x \in \partial\Omega, \quad t \geq 0,$$

звідки

$$\frac{\partial y}{\partial \nu}(x, t) + \alpha(x, t)y(x, t) = \alpha(x, t)\tilde{y}(x, t), \quad x \in \partial\Omega, \quad t \geq 0,$$

9.2. Модель Fisher'a оптимальної народжуваності

Розглянемо модель Fisher'a, яка описує динаміку біологічної популяції, яка вільно може переміщатися в ізольованому ареалі життєдіяльності

Ω :

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = ry \left(1 - \frac{y}{k}\right) - m(x)u(x,t)y(x,t), & (x,t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x,t) = 0, & (x,t) \in \Sigma_T, \\ y(x,0) = y_0(x), & x \in \Omega. \end{cases} \quad (9.2.1)$$

Тут Ω – обмежена відкрита підмножина у \mathbb{R}^N ($N \in \mathbb{N}^*$) з межею класу C^1 . T , γ , r , k – додатні сталі, $Q_T = \Omega \times (0, T)$, $\Sigma_T = \partial\Omega \times (0, T)$, $y_0 \in L^\infty(\Omega)$, $y_0(x) > 0$ д.м.в. $x \in \Omega$ (y_0 – початкова густина популяції). Ця задача випливає з (9.1.4) у випадку, коли коефіцієнт природної народжуваності більший від коефіцієнта природної смертності.

Нехай u означає сприяння народжуваності і має зміст на не пустій відкритій підмножині $\omega \subset \Omega$, а m – характеристична функція ω . Тоді згідно з позначенням

$$m(x)u(x,t) = \begin{cases} u(x,t), & x \in \omega, t \in (0, T), \\ 0, & x \in \Omega \setminus \bar{\omega}, t \in (0, T). \end{cases}$$

Загально народжуваність на часовому проміжку $[0, T]$ визначається згідно з формулою

$$\int_0^T \int_\omega u(x,t)y^u(x,t) dx dt,$$

де y^u – розв’язок задачі (9.2.1) і задача оптимального керування стосовно (9.2.1) формулюється так:

$$\int_0^T \int_\omega u(x,t)y^u(x,t) dx dt \rightarrow \max, \quad (\text{DP1})$$

стосовно $u \in K = \{w \in L^2(\omega \times (0, T)) : 0 \leq w(x,t) \leq L \text{ м.в. } \} \quad (L > 0)$, де y^u – розв’язок задачі (9.2.1).

Формулювання задач та основні властивості розв’язків параболічних рівнянь можна знайти детальніше у [20].

Якщо для $u \in K$, задача (9.2.1) допускає розв’язок y^u , тоді з принципу максимуму для параболічного рівняння (див. [63]) одержуємо оцінку

$$0 < y^u(x,t) \leq M = \|y_0\|_{L^\infty(Q_T)} e^{rT} \text{ м.в. } (x,t) \in Q_T.$$

Із теореми Ванаш’а про фіксовану точку звідси одержуємо, що для кожного $u \in K$ нелінійна задача (9.2.1) допускає єдиний розв’язок y^u , який

задовольняє вищезазначену оцінку.

Існування оптимального керування

Означимо функціонал

$$\Phi(u) = \int_0^T \int_{\omega} u(x, t) y^u(x, t) dx dt, \quad u \in K,$$

і нехай

$$d = \sup_{u \in K} \Phi(u).$$

Застосувавши принцип максимуму для параболічного рівняння, одержимо

$$0 < y^u(x, t) \leq y^0(x, t) \text{ д.м.в. } (x, t) \in \Omega \times (0, T),$$

звідки випливає

$$0 \leq \int_0^T \int_{\omega} u(x, t) y^u(x, t) dx dt \leq L \int_0^T \int_{\omega} y^0(x, t) dx dt.$$

У підсумку одержуємо, що

$$d \in \mathbb{R}^+.$$

Нехай $\{u_n\}_{n \in \mathbb{N}^*} \subset K$ – послідовність керувань, які задовольняють систему нерівностей

$$d - \frac{1}{n} < \Phi(u_n) \leq d. \quad (9.2.2)$$

Позаяк $\{u_n\}_{n \in \mathbb{N}^*}$ є обмеженою послідовністю у $L^2(\omega \times (0, T))$, тоді існує така підпослідовність, за якою збережемо те саме позначення $\{u_n\}_{n \in \mathbb{N}^*}$, що

$$u_n \rightarrow u^* \text{ слабо у } L^2(\omega \times (0, T)),$$

і

$$m u_n \rightarrow m u^* \text{ слабо у } L^2(\omega \times (0, T)).$$

Оскільки K – замкнута опукла підмножина у $L^2(\omega \times (0, T))$, а отже, слабо замкнена, тому $u^* \in K$ (див. [26]).

Уведемо позначення

$$a_n(x, t) = r y^{u_n}(x, t) \left(1 - \frac{y^{u_n}(x, t)}{k} \right) - m(x) u_n(x, t) y^{u_n}(x, t), \quad (x, t) \in Q_T.$$

Очевидно, що $\{a_n\}_{n \in \mathbb{N}^*}$ є обмеженою послідовністю у $L^\infty(Q_T)$ (а також у $L^2(Q_T)$), і y^{u_n} задовольняє задачу

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = a_n(x, t), & (x, t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ y(x, 0) = y_0(x), & x \in \Omega. \end{cases}$$

Із обмеженості послідовності $\{a_n\}_{n \in \mathbb{N}^*}$ одержуємо (використавши результат компактності для параболічних рівнянь, див. [20]), що існують такі підпослідовності $\{a_{n_k}\}_{k \in \mathbb{N}^*}$ та $\{y^{u_{n_k}}\}_{k \in \mathbb{N}^*}$, що

$$\begin{cases} a_{n_k} \rightarrow a^* & \text{слабко у } L^2(Q_T), \\ y^{u_{n_k}} \rightarrow y^* & \text{м.в. у } Q_T, \end{cases} \quad (9.2.3)$$

де y^* є розв'язком задачі

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = a^*(x, t), & (x, t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ y(x, 0) = y_0(x), & x \in \Omega. \end{cases}$$

З іншого боку, із (9.2.3), використовуючи слабку збіжність $\{u_n\}_{n \in \mathbb{N}^*}$ та обмеженість $\{y^{u_{n_k}}\}_{n \in \mathbb{N}^*}$ у просторі $L^\infty(Q_T)$, можемо зробити висновок, що

$$a_{n_k} \rightarrow ry^* \left(1 - \frac{y^*}{k}\right) - mu^*y^* \text{ у } L^2(Q_T),$$

а отже,

$$a^* = ry^* \left(1 - \frac{y^*}{k}\right) - mu^*y^* \text{ у } L^2(Q_T).$$

І на завершення, y^* є розв'язком задачі (9.2.1), який відповідає $u := u^*$, тобто $y^* = y^{u^*}$.

Якщо тепер перейдемо до границі у (9.2.2), то одержимо таке:

$$d = \Phi(u^*),$$

а це означає, що u^* – оптимальне керування задачі (DP1).

Принцип максимуму

Задачу (9.2.1) можна переписати як початкову у просторі $L^2(\Omega)$

$$\begin{cases} y'(t) = f(t, u(t), y(t)), & t \in (0, T), \\ y(0) = y_0, \end{cases}$$

де

$$f(t, u, y) = Ay + ry \left(1 - \frac{y}{k}\right) - muy.$$

Тут A – лінійний необмежений оператор. Справді, A можна визначити так:

$$D(A) = \{w \in H^2\Omega; \frac{\partial w}{\partial \nu} = 0 \text{ на } \partial\Omega\},$$

$$Ay = \gamma\Delta y, \quad y \in D(A).$$

Означення та основні властивості просторів Sobolev'a детальніше можна знайти у [1, 26]. Рівняння, розв'язком якого є спряжений стан, було виведене у розділі 8.3.2. і записується

$$p'(t) = -A^*p - rp + \frac{2r}{k}y^u p + tu(1+p), \quad t \in (0, T).$$

Зауважимо, що A є самоспряженим оператором, а тому A^* можна замінити A . Доведемо таку теорему.

Теорема 9.2.1. *Якщо (u^*, y^{u^*}) є оптимальною парою задачі (DP1), а p – розв'язком задачі*

$$\begin{cases} \frac{\partial p}{\partial t} + \gamma\Delta p = -rp + \frac{2r}{k}y^{u^*} p + tu^*(1+p), & (x, t) \in Q_T, \\ \frac{\partial p}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ p(x, T) = 0, & x \in \Omega, \end{cases} \quad (9.2.4)$$

тоді оптимальне керування допускає зображення

$$u^*(x, t) = \begin{cases} 0, & \text{якщо } 1 + p(x, t) < 0, \\ L, & \text{якщо } 1 + p(x, t) > 0, \end{cases} \quad (9.2.5)$$

д.м.в. $(x, t) \in \omega \times (0, T)$.

Доведення. Розглянемо множину

$$V = \{w \in L^2(\omega \times (0, T)) : u^* + \varepsilon w \in K\},$$

для довільних достатньо малих $\varepsilon > 0$. Для кожної довільної фіксованої функції $v \in V$ правильна нерівність (на підставі означення оптимального розв'язку)

$$\int_0^T \int_{\omega} u^*(x, t) y^{u^*}(x, t) dx dt / gqs \int_0^T \int_{\omega} (u^*(x, t) + \varepsilon v(x, t)) y^{u^* + \varepsilon v}(x, t) dx dt,$$

звідки

$$\int_0^T \int_{\omega} u^* \frac{y^{u^* + \varepsilon v} - y^{u^*}}{\varepsilon} dx dt + \int_0^T \int_{\omega} v y^{u^* + \varepsilon v} dx dt \leq 0, \quad (9.2.6)$$

для довільного достатньо малого $\varepsilon > 0$.

Щоб завершити доведення теореми, сформулюємо допоміжне твердження.

Лема 9.2.1. *Нехай z є розв'язком задачі*

$$\begin{cases} \frac{\partial z}{\partial t} - \gamma \Delta z = rz - \frac{2r}{k} y^{u^*} z - mu^* z - mv y^{u^*}, & (x, t) \in Q_T, \\ \frac{\partial z}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ z(x, 0) = 0, & x \in \Omega, \end{cases} \quad (9.2.7)$$

тоді справедливі граничні переходи

$$\begin{aligned} y^{u^* + \varepsilon v} &\longrightarrow y^{u^*} & y & L^\infty(Q_T), \\ \frac{1}{\varepsilon} [y^{u^* + \varepsilon v} - y^{u^*}] &\longrightarrow z & y & L^\infty(Q_T). \end{aligned}$$

Доведення теореми (продовження). Перейшовши до границі у нерівності (9.2.6) (використаємо теорему Lebesgue та лему 9.2.1), одержимо

$$\int_0^T \int_{\omega} [u^*(x, t) z(x, t) + v(x, t) y^{u^*}(x, t)] dx dt \leq 0. \quad (9.2.8)$$

Помножимо рівняння (9.2.7)₁ на p та проінтегрувавши по області Q_T , одержимо

$$\int_0^T \int_{\Omega} p \left[\frac{\partial z}{\partial t} - \gamma \Delta z \right] dx dt = \int_0^T \int_{\Omega} p \left[rz - \frac{2r}{k} y^{u^*} z - mu^* z - mv y^{u^*} \right] dx dt.$$

Після інтегрування частинами (стосовно змінної t) та застосування формули Green'a (стосовно x), остання інтегральна рівність набуде вигляду

$$-\int_0^T \int_{\Omega} z \left[\frac{\partial p}{\partial t} + \gamma \Delta p \right] dx dt = \int_0^T \int_{\Omega} p \left[rz - \frac{2r}{k} y^{u^*} z - mu^* z - mvy^{u^*} \right] dx dt.$$

Позаяк p є розв'язком задачі (9.2.4), то

$$\begin{aligned} & -\int_0^T \int_{\Omega} z \left[-rp + \frac{2r}{k} y^{u^*} p + mu^*(1+p) \right] dx dt = \\ & = \int_0^T \int_{\Omega} p \left[rz - \frac{2r}{k} y^{u^*} z - mu^* z - mvy^{u^*} \right] dx dt, \end{aligned}$$

отже, можна зробити висновок, що

$$\int_0^T \int_{\Omega} m(x) u^*(x, t) z(x, t) dx dt = \int_0^T \int_{\Omega} m(x) v(x, t) y^{u^*}(x, t) p(x, t) dx dt.$$

А це еквівалентно інтегральній рівності

$$\int_0^T \int_{\omega} u^*(x, t) z(x, t) dx dt = \int_0^T \int_{\omega} v(x, t) y^{u^*}(x, t) p(x, t) dx dt. \quad (9.2.9)$$

Остаточно, із нерівності (9.2.8) та інтегральної рівності (9.2.9), одержимо

$$\int_0^T \int_{\omega} v(x, t) y^{u^*}(x, t) (1 + p(x, t)) dx dt \leq 0 \quad \forall v \in V.$$

Аналогічно, як у розділі 8.3., міркуючи (використавши додатність y^{u^*}), одержимо, що u^* задовольняє (9.2.5) і, крім того, $\forall u \in U = L^2(\omega \times (0, T))$ можна довести, що

$$\Phi_u(u) = y^u(1 + p).$$

Теорема доведена. □

Доведення лєми. Доведемо першу частину лєми. Друга частина доводиться аналогічно.

Для довільної $v \in V$ та довільного достатньо малого $\varepsilon > 0$ матимемо, що

$$0 \leq y^{u^*}(x, t), \quad y^{u^*+\varepsilon v} \leq M \quad \text{д.м.в. } (x, t) \in Q_T.$$

Якщо позначити $w_\varepsilon = y^{u^*+\varepsilon v} - y^{u^*}$, то w_ε є розв'язком задачі

$$\begin{cases} \frac{\partial w}{\partial t} - \gamma \Delta w = rw - \frac{r}{k} w(y^{u^*+\varepsilon v} + y^{u^*}) - mww - \varepsilon mv y^{u^*+\varepsilon v}, & (x, t) \in Q_T, \\ \frac{\partial w}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ w(x, 0) = 0, & x \in \Omega. \end{cases}$$

Якщо в черговий раз застосуємо принцип максимуму для параболічного рівняння, то одержимо такі оцінки:

$$w_{1\varepsilon}(x, t) \leq w(x, t) \leq w_{2\varepsilon}(x, t) \quad \text{д.м.в. } (x, t) \in Q_T, \quad (9.2.10)$$

де $w_{1\varepsilon}$ є розв'язком задачі

$$\begin{cases} \frac{\partial w_1}{\partial t} - \gamma \Delta w_1 = \left(r + \frac{2rM}{k} + L \right) w_1 - \varepsilon M \|v\|_{L^\infty(Q_T)}, & (x, t) \in Q_T, \\ \frac{\partial w_1}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ w_1(x, 0) = 0, & x \in \Omega, \end{cases}$$

і w_2 – розв'язком задачі

$$\begin{cases} \frac{\partial w_2}{\partial t} - \gamma \Delta w_2 = \left(r + \frac{2rM}{k} + L \right) w_2 + \varepsilon M \|v\|_{L^\infty(Q_T)}, & (x, t) \in Q_T, \\ \frac{\partial w_2}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ w_2(x, 0) = 0, & x \in \Omega, \end{cases}$$

Безпосередньою перевіркою можна переконатися, що

$$w_{1\varepsilon} = -\varepsilon M \|v\|_{L^\infty(Q_T)} \int_0^t e^{M_0(t-s)} ds \quad \text{д.м.в. } (x, t) \in Q_T,$$

і також

$$w_{2\varepsilon} = \varepsilon M \|v\|_{L^\infty(Q_T)} \int_0^t e^{M_0(t-s)} ds \quad \text{д.м.в. } (x, t) \in Q_T,$$

де $M_0 = r + 2rM/k + L$.

Із явних зображень розв'язків $w_{1\varepsilon}$ та $w_{2\varepsilon}$ одержуємо

$$w_{1\varepsilon}, w_{2\varepsilon} \xrightarrow{\varepsilon \rightarrow 0^+} 0 \quad \text{у } L^\infty(Q_T).$$

Звідси та з (9.2.10) випливає таке:

$$w_\varepsilon \xrightarrow{\varepsilon \rightarrow 0^+} 0 \quad \text{у } L^\infty(Q_T),$$

що і завершує доведення першої частини леми.

Для того, щоб довести другу частину леми, приймемо

$$l_\varepsilon = \frac{1}{\varepsilon} [y^{u^* + \varepsilon v} - y^{u^*}] - z,$$

і тим самим способом, що і вище (використовуючи принцип максимуму для параболічного рівняння) доведемо, що

$$l_\varepsilon \xrightarrow{\varepsilon \rightarrow 0^+} 0 \quad \text{у } L^\infty(Q_T),$$

чим і завершимо доведення другої частини леми, а отже, доведення леми у цілому. □

Важливо зауважити, що при доведенні другої частини леми суттєво використовується результат першої частини доведення.

Дослідимо тепер задачу без логістичного члена, яка формулюється так:

$$\int_0^T \int_\omega u(x, t) y^u(x, t) dx dt \longrightarrow \max \quad (\text{DP1}')$$

стосовно $u \in K$, де y^u – розв'язок задачі

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = ry - m(x)u(x, t)y(x, t), & (x, t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ y(x, 0) = y_0(x), & x \in \Omega. \end{cases}$$

Існування оптимального керування доводиться аналогічно як у попередньому випадку. Сформулюємо необхідні умови оптимальності першого порядку.

Теорема 9.2.2. Якщо (u^*, y^{u^*}) є оптимальною парою задачі (DP1') і p розв'язком задачі

$$\begin{cases} \frac{\partial p}{\partial t} + \gamma \Delta p = -rp + u^*(1+p), & (x, t) \in Q_T, \\ \frac{\partial p}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ p(x, T) = 0, & x \in \Omega, \end{cases} \quad (9.2.11)$$

тоді оптимальне керування u^* має вигляд

$$u^*(x, t) = \begin{cases} 0, & \text{якщо } 1 + p(a, t) < 0, \\ L, & \text{якщо } 1 + p(a, t) > 0, \end{cases} \quad (9.2.12)$$

д.м.в. $(x, t) \in \omega \times (0, T)$.

Із (9.2.11) та (9.2.12) одержимо у цьому випадку, що p є єдиним розв'язком задачі

$$\begin{cases} \frac{\partial p}{\partial t} + \gamma \Delta p = -rp + mL(1+p)^+, & (x, t) \in Q_T, \\ \frac{\partial p}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ p(x, T) = 0, & x \in \Omega. \end{cases} \quad (9.2.13)$$

Якщо побудувати апроксимацію розв'язку p задачі (9.2.13) (яка не залежить від u^* і y^{u^*}), тоді, використавши (9.2.12), одержимо оптимальне керування u^* . Отже, алгоритм знаходження розв'язку задачі (DP1') такий:

ОР1: обчислити розв'язок p задачі (9.2.13);

ОР2: обчислити оптимальне керування u^* за формулою (9.2.12).

Розв'язування (ОР1) відбувається у зворотному за часом напрямі. Очевидно, що обчислення керування є стійкою апроксимацією в стосунку до того, що завдання (ОР1) проводиться чисельними методами. Однак алгоритм досить простий.

Із (9.2.11) і (9.2.12) можна зробити висновок, що u^* – двопозиційне керування. Крім того, оптимальне керування u^* явно не залежить від y_0 .

Чисельний алгоритм розв'язування завдання (DP1)

Остання теорема та вигляд функціонала Φ_u дають змогу укласти алгоритм побудови наближення оптимального значення цільового функціонала. Нижче опишемо алгоритм типу проєкції градієнта (метод Uzawa).

Метод Uzawa для (DP1)**S0:** Вибрати $u^{(0)} \in K$.Прийняти $j := 0$.**S1:** Обчислити $y^{(j)}$, розв'язок (9.2.1), підставивши $u^{(j)}$

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = ry \left(1 - \frac{y}{k}\right) - m(x)u^{(j)}y, & (x, t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ y(x, 0) = y_0(x), & x \in \Omega. \end{cases}$$

S2: Обчислити $p^{(j)}$, розв'язок (9.2.4) із $y^{(j)}$, $u^{(j)}$

$$\begin{cases} \frac{\partial p}{\partial t} + \gamma \Delta p = -rp + \frac{2r}{k}y^{(j)}p + mu^{(j)}(1+p), & (x, t) \in Q_T, \\ \frac{\partial p}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ p(x, T) = 0, & x \in \Omega. \end{cases}$$

S3: Обчислити напрям градієнта $w^{(j)}$ за формулою

$$w^{(j)} := \Phi_u(u^{(j)}) = y^{(j)}(1 + p^{(j)}).$$

S4: (Критерій завершення алгоритма)Якщо $\|w^{(j)}\| < \varepsilon$,тоді STOP ($u^{(j)}$ є апроксимація керування),у противному випадку перехід на **S5**.**S5:** Обчислити крок ρ_j із умови

$$\Phi(P_K(u^{(j)} + \rho_j w^{(j)})) = \max_{\rho \geq 0} \{\Phi(P_K(u^{(j)} + \rho w^{(j)}))\}.$$

S6: Обчислити подальше наближення керування за формулою

$$u^{(j+1)} := P_K(u^{(j)} + \rho_j w^{(j)}).$$

 $j := j + 1$; перехід на **S1**.

Якщо використати формулу (9.2.12) для побудови оптимального керування, тоді одержимо метод проєкції градієнта типу Rosen'a.

Метод Rosen'а для (DP1)

S0: Вибрати $u^{(0)} \in K$.
Прийняти $j := 0$.

S1: Обчислити $y^{(j)}$, розв'язок (9.2.1), підставивши $u^{(j)}$

$$\begin{cases} \frac{\partial y}{\partial t} - \gamma \Delta y = ry \left(1 - \frac{y}{k}\right) - mu^{(j)}y, & (x, t) \in Q_T, \\ \frac{\partial y}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ y(x, 0) = y_0(x), & x \in \Omega. \end{cases}$$

S2: Обчислити $p^{(j)}$, розв'язок (9.2.4) із $y^{(j)}$, $u^{(j)}$

$$\begin{cases} \frac{\partial p}{\partial t} + \gamma \Delta p = -rp + \frac{2r}{k}y^{(j)}p + mu^{(j)}(1+p), & (x, t) \in Q_T, \\ \frac{\partial p}{\partial \nu}(x, t) = 0, & (x, t) \in \Sigma_T, \\ p(x, T) = 0, & x \in \Omega. \end{cases}$$

S3: Обчислити $v^{(j)}$ за формулою

$$v^{(j)}(x, t) = \begin{cases} 0, & \text{якщо } 1 + p^{(j)}(x, t) < 0, \\ L, & \text{якщо } 1 + p^{(j)}(x, t) \geq 0, \end{cases}$$

для $(x, t) \in \omega \times (0, T)$. Наближення $v^{(j)}$ оптимального керування u^* обчислили за формулою (9.2.5).

S4: Обчислити $\lambda_j \in [0, 1]$ із умови

$$\max_{\lambda \in [0, 1]} \Phi(\lambda u^{(j)} + (1 - \lambda)v^{(j)}).$$

S5: Обчислити чергове наближення керування $u^{(j+1)}$ за формулою

$$u^{(j+1)} = \lambda_j u^{(j)} + (1 - \lambda_j)v^{(j)}.$$

S6: (Критерій завершення алгоритму)

Якщо $\|u^{(j+1)} - u^{(j)}\| < \varepsilon$,
тоді STOP ($u^{(j)}$ є апроксимація керування),
у противному випадку $j := j + 1$; перехід на **S1**.

Зауважимо, що опукла комбінація двох двопозиційних керувань набуває значення між 0 та L , але не обов'язково ці два значення. Це дає пояснення чому ліпше замінити опуклу комбінацію $\lambda u^{(j)} + (1 - \lambda)v^{(j)}$ у **S4** та **S5** опуклою комбінацією точок переключення $u^{(j)}$ і $v^{(j)}$ як було обґрунтовано у розділі 8.5.3.

Щодо використання MatLab[®], то можна сказати, що для $\Omega \subset \mathbb{R}$ можна застосувати скінчені різницеві схеми для апроксимації розв'язку рівнянь із частинними похідними на етапі **S1** та **S2**, написати відповідну програму. У випадку $\Omega \subset \mathbb{R}^2$ може бути використаний метод скінчених елементів (Finite Element Method (FEM)) знаходження розв'язку рівняння з частинними похідними.

9.3. Приклад: керування системою реакції-дифузії

У цьому розділі продовжимо дослідження моделі хижак-жертва з розділу 8.3.3.. Припустимо, що дві популяції не змішуються в тому ж просторі життєдіяльності Ω .

Наступне відхилення від систем реакції-дифузії виникають у динаміці популяції або в епідемології – та інших областях застосування – є у члені реакції, який нелокальний, тобто зображається інтегральним членом. Повертаючись до нашої мотиваційної задачі, біомаса схоплених і спожитих жертв у точці x середовища існування Ω у подальшому розподіляється на всю територію розташування хижаків, Ω . Це приводить до локальних/нелокальних міжвидових взаємодій між двома видами хижаків, тобто, функціональна реакція на хижаків є локальною тоді, як чисельна реакція на хижаків нелокальна і розподілена на Ω (див. також [34, 39, 43, 11]).

Це знову відхилення від більшості стандартних моделей хижак-жертва, див. [57]. Із феноменологічного погляду маємо змогу впровадити досить загальний член інтегрального ядра для моделювання просторового розподілу біомаси. У випадку нелокальних дифузійних моделей епідемії див. [27].

Сформулюємо математичну задачу досліджуваної моделі. Нехай $y_1(x, t)$ означає щільність видів жертви у точці x у момент часу t розподіленої в області $\Omega \subset \mathbb{R}^N$, $N = 1, 2$, або, 3 , і припустимо, що просторово-часова динаміка описується рівнянням

$$\frac{\partial y_1}{\partial t} - d_1 \Delta y_1 = r_1 y_1, \quad x \in \Omega, \quad t \in (0, T),$$

де r_1 означає коефіцієнт природного зростання, $d_1 > 0$ є коефіцієнт дифузії і $T > 0$. Нехай $y_2(x, t)$ означає щільність у точці x у момент часу t розподілу видів хижаків у цьому самому просторі життєдіяльності; за відсутності вищезгаданих жертв – припускається існування їхнього своєрідного запасу – популяція хижаків зменшується за експоненціальним законом із коефіцієнтом $r_2 > 0$ і просторово-часова динаміка описується рівнянням

$$\frac{\partial y_2}{\partial t} - d_2 \Delta y_2 = -r_2 y_2, \quad x \in \Omega, \quad t \in (0, T),$$

де $d_2 > 0$ – коефіцієнт дифузії.

У випадку наявності двох популяцій відбувається полювання хижаків у Ω ; припустимо, що функціонал є типу Lotka–Volterra (див. [57]). Динаміка популяції жертви модифікується у зв'язку з наявністю хижаків, у зв'язку з чим рівняння набуде вигляду

$$\frac{\partial y_1}{\partial t} - d_1 \Delta y_1 = r_1 y_1 - \mu_1 u(x, t) y_1 y_2, \quad x \in \Omega, \quad t \in (0, T), \quad (9.3.1)$$

де $\mu_1 > 0$ і $1 - u(x, t)$ означає розділення (сегрегацію) двох популяцій. u фактично є керуванням.

Схоплена і спожита жертва у момент часу t у точці $x' \in \Omega$ перетворюється у біомасу і через фактор перетворення впливає на кількість хижаків $\mu_2 u(x', t) y_1(x', t) y_2(x', t)$ ($\mu_2 = \text{constant} > 0$). Припускаємо, що ця величина розподілена на Ω за допомогою невід'ємного ядра $\ell(x, x')$ ($\ell \in L^\infty(\Omega \times \Omega)$, $\ell(x, x') \geq 0$ д.м.в. $(x, x') \in \Omega \times \Omega$) так, що $\ell(x, x') \mu_2 u(x, x') y_1(x, x') y_2(x, x')$ біомаса у точці $x \in \Omega$ є результатом хижацтва у точці $x' \in \Omega$. Із наведених міркувань щодо біомаси впливає, що має виконуватися умова $\int_{\Omega} \ell(x, x') dx = 1$ д.м.в. $x' \in \Omega$.

Беручи до уваги сказане вище, динаміка хижака описується рівнянням

$$\frac{\partial y_2}{\partial t} - d_2 \Delta y_2 = -r_2 y_2 + \mu_2 \int_{\Omega} \ell(x, x') u(x', t) y_1(x', t) y_2(x', t) dx', \quad (9.3.2)$$

$$x \in \Omega, \quad t \in (0, T).$$

Щоб завершити побудову моделі, треба записати крайові умови для обидвох видів. Візьмемо крайові умови без міграцій, що відповідають ізоляції популяцій

$$\frac{\partial y_1}{\partial \nu}(x, t) = \frac{\partial y_2}{\partial \nu}(x, t) = 0, \quad x \in \partial\Omega, \quad t \in (0, T). \quad (9.3.3)$$

І на завершення, невід'ємні й обмежені початкові умови задаються у початковий момент часу $t = 0$

$$\begin{cases} y_1(x, 0) = y_{01}(x), & x \in \Omega, \\ y_2(x, 0) = y_{02}(x), & x \in \Omega. \end{cases} \quad (9.3.4)$$

Отже, задача (9.3.1)-(9.3.4) є основною моделлю керування системою хижак-жертва.

Припустимо, що

$$y_{01}, y_{02} \in L^\infty(\Omega),$$

та

$$y_{01}(x), y_{02}(x) > 0, \quad \text{д.м.в. } x \in \Omega.$$

Наша мета – максимізувати загальну кількість особин обидвох популяцій у момент часу $T > 0$. Ця задача споріднена з задачею (P₃), яку досліджено у розділі 8.3.3.

Формулювання задачі звучить так:

$$\Psi(u) = \int_{\Omega} [y_1^u(x, T) + y_2^u(x, T)] dx \longrightarrow \max, \quad (\mathbf{DP2})$$

стосовно $u \in L^2(\Omega \times (0, T))$, $0 \leq u(x, t) \leq 1$ д.м.в. $t \in (0, T)$, де (y_1^u, y_2^u) – розв'язок задачі (9.3.1)-(9.3.4).

Побудуємо нижче алгоритм методу проєкції градієнта типу Rosen'а апроксимацій оптимального керування задачі (DP2) у припущенні, що $K = \{w \in L^2(\Omega \times (0, T)) : 0 \leq w(x, t) \leq 1 \text{ м.в. } \}$.

Алгоритм методу проєкції градієнта для задачі (DP2) (метод Rosen'а)

S0: Вибрати $u^{(0)} \in K$.

Прийняти $j := 0$.

S1: Обчислити $y^{(j)} = (y_1^{(j)}, y_2^{(j)})$ розв'язок задачі (9.3.1)-(9.3.4), у якій

підставити $u^{(j)}$

$$\begin{cases} \frac{\partial y_1}{\partial t} - d_1 \Delta y_1 = r_1 y_1 - \mu_1 u^{(j)}(x, t) y_1 y_2, & x \in \Omega, t \in (0, T), \\ \frac{\partial y_2}{\partial t} - d_2 \Delta y_2 = -r_2 y_2 + \mu_2 \int_{\Omega} \ell(x, x') u^{(j)}(x', t) y_1(x', t) y_2(x', t) dx', & \\ & x \in \Omega, t \in (0, T), \\ \frac{\partial y_1}{\partial \nu}(x, t) = \frac{\partial y_2}{\partial \nu}(x, t) = 0, & x \in \partial\Omega, t \in (0, T), \\ y_1(x, 0) = y_{01}(x), \quad y_2(x, 0) = y_{02}(x), & x \in \Omega. \end{cases}$$

S2: Обчислити $p^{(j)} = (p_1^{(j)}, p_2^{(j)})$, розв'язок задачі

$$\begin{cases} \frac{\partial p_1}{\partial t} + d_1 \Delta p_1 = -r_1 p_1 + \mu_1 u^{(j)} y_2^{(j)} p_1 - \\ \quad - \mu_2 u^{(j)}(x, t) y_2^{(j)}(x, t) \int_{\Omega} \ell(x', x) p_2(x', t) dx', & (x, t) \in \Omega \times (0, T), \\ \frac{\partial p_2}{\partial t} + d_2 \Delta p_2 = r_2 p_2 + \mu_1 u^{(j)} y_1^{(j)} p_1 - \\ \quad - \mu_2 u^{(j)}(x, t) y_1^{(j)}(x, t) \int_{\Omega} \ell(x', x) p_2(x', t) dx', & (x, t) \in \Omega \times (0, T), \\ \frac{\partial p_1}{\partial \nu}(x, t) = \frac{\partial p_2}{\partial \nu}(x, t) = 0, & (x, t) \in \partial\Omega \times (0, T), \\ p_1(x, T) = p_2(x, T) = 1, & x \in \Omega. \end{cases}$$

S3: Обчислити $v^{(j)}$ за формулою з теореми 9.4.1

$$v^{(j)}(x, t) = \begin{cases} 0, & \text{якщо } \mu_2 \int_{\Omega} \ell(x', x) p_2^{(j)}(x', t) dx' - \mu_1 p_1^{(j)}(x, t) < 0, \\ 1, & \text{якщо } \mu_2 \int_{\Omega} \ell(x', x) p_2^{(j)}(x', t) dx' - \mu_1 p_1^{(j)}(x, t) > 0, \end{cases}$$

м.в. на $\Omega \times (0, T)$.

S4: Обчислити значення $\lambda_j \in [0, 1]$, яке є розв'язком задачі максимізації

$$\max_{\lambda \in [0,1]} \Psi(\lambda u^{(j)} + (1 - \lambda)v^{(j)}).$$

S5: Обчислити чергове наближення керування $u^{(j+1)}$ за формулою

$$u^{(j+1)} = \lambda_j u^{(j)} + (1 - \lambda_j)v^{(j)}.$$

S6: (Критерій завершення алгоритму)

Якщо $\|u^{(j+1)} - u^{(j)}\| < \varepsilon$,

тоді **STOP** ($u^{(j+1)}$ є наближенням керування)

у протилежному випадку $j := j + 1$; перехід на **S1**.

Варто зауважити, що опукла комбінація двох двопозиційних керувань, які набувають значення 0 або 1, не обов'язково набуватиме тільки тих значень. Із цієї причини замінили опуклу комбінацію $\lambda u^{(j)} + (1 - \lambda)v^{(j)}$ у **S4** та **S5** опуклою комбінацією точок переключення $u^{(j)}$ та $v^{(j)}$, аналогічно як у розділі 8.5.3.

9.4. Завдання для самостійного опрацювання

◆ *Завдання 9.4.1.* Довести, що для довільного u задача (9.3.1)-(9.3.4) допускає єдиний розв'язок.

◆ *Завдання 9.4.2.* Довести таку теорему.

Теорема 9.4.1. Нехай $(u^*, (y_1^*, y_2^*))$ – оптимальна пара задачі **(DP2)**, і $p = (p_1, p_2)$ – розв'язок задачі

$$\left\{ \begin{array}{l} \frac{\partial p_1}{\partial t} + d_1 \Delta p_1 = -r_1 p_1 + \mu_1 u^* y_2^* p_1 - \mu_2 u^*(x, t) y_2^*(x, t) \int_{\Omega} \ell(x', x) p_2(x', t) dx', \\ \frac{\partial p_2}{\partial t} + d_2 \Delta p_2 = r_2 p_2 + \mu_1 u^* y_1^* p_1 - \mu_2 u^*(x, t) y_1^*(x, t) \int_{\Omega} \ell(x', x) p_2(x', t) dx', \\ (x, t) \in \Omega \times (0, T), \\ \frac{\partial p_1}{\partial \nu}(x, t) = \frac{\partial p_2}{\partial \nu}(x, t) = 0, \quad (x, t) \in \partial \Omega \times (0, T), \\ p_1(x, T) = p_2(x, T) = 1, \quad x \in \Omega, \end{array} \right.$$

тоді

$$u^*(x, t) = \begin{cases} 0, & \text{якщо } \mu_2 \int_{\Omega} \ell(x', x) p_2(x', t) dx' - \mu_1 p_1(x, t) < 0, \\ 1, & \text{якщо } \mu_2 \int_{\Omega} \ell(x', x) p_2(x', t) dx' - \mu_1 p_1(x, t) > 0, \end{cases}$$

м.в. на $\Omega \times (0, T)$.

♣ **Зауваження 9.4.1.** Досить цікаво вивчити задачі оптимального керування досліджених моделей із просторовою структурою (додаючи за необхідності дифузійний доданок).

◆ **Завдання 9.4.3.** Записати принцип максимуму для такої задачі:

$$\int_0^T \int_0^A \int_{\omega} u(x, a, t) y^u(x, a, t) dx da dt \longrightarrow \max,$$

де $u \in L^2(\Omega \times (0, A) \times (0, T))$, $0 \leq u(x, a, t) \leq L$ м.в., y^u є розв'язком задачі

$$\begin{cases} \frac{\partial y}{\partial t} + \frac{\partial y}{\partial a} + \mu(a)y = -m(x)u(x, t)y(x, a, t), & (x, a, t) \in Q, \\ \frac{\partial y}{\partial \nu}(x, a, t) = 0, & (x, t) \in \Sigma, \\ y(x, 0, t) = \int_0^A \beta(a)y(x, a, t) da, & (x, t) \in \Omega \times (0, T), \\ y(x, a, 0) = y_0(x, a), & (x, t) \in \Omega \times (0, A), \end{cases}$$

де $Q = \Omega \times (0, A) \times (0, T)$, $\Sigma = \partial\Omega \times (0, A) \times (0, T)$. Тут $\gamma > 0$, Ω , і ω задовольняють припущення з розділу 9.2., а β і μ – припущення з розділу 8.5.1.

✓ **Вказівка.** Детальніше див. [13].

◆ **Завдання 9.4.4.** За припущень із розділу 9.3., записати принцип максимуму такої задачі:

$$\int_{\Omega} [y_1^u(x, T) + y_2^u(x, T)] dx \longrightarrow \max, \quad (\mathbf{DP2}')$$

стосовно $u \in L^2(\Omega \times (0, T))$, $0 \leq u(x, t) \leq 1$ д.м.в. $t \in (0, T)$, де (y_1^u, y_2^u) є розв'язком задачі

$$\begin{cases} \frac{\partial y_1}{\partial t} - d_1 \Delta y_1 = r_1 y_1 - \mu_1 u(x, t) y_1 y_2, & (x, t) \in \Omega \times (0, T), \\ \frac{\partial y_2}{\partial t} - d_2 \Delta y_2 = -r_2 y_2 + \mu_2 u(x, t) y_1 y_2, & (x, t) \in \Omega \times (0, T), \\ \frac{\partial y_1}{\partial \nu}(x, t) = \frac{\partial y_2}{\partial \nu}(x, t) = 0, & (x, t) \in \partial\Omega \times (0, T), \\ y_1(x, 0) = y_{01}(x) \quad y_2(x, 0) = y_{02}(x), & x \in \Omega. \end{cases}$$

✓ *Вказівка.* (y_1^u, y_2^u) – розв'язок такої початкової задачі (у $L^2(\Omega) \times L^2(\Omega)$):

$$\begin{cases} y'(t) = f(t, u(t), y(t)), & t \in (0, T), \\ y(0) = y_0, \end{cases}$$

$$y_0 = \begin{pmatrix} y_{01} \\ y_{02} \end{pmatrix},$$

$$f(t, u, y) = Ay + \begin{pmatrix} r_1 y_1 - \mu_1 u y_1 y_2 \\ -r_2 y_2 + \mu_2 u y_1 y_2 \end{pmatrix}.$$

Тут

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},$$

і A – лінійні необмежені оператори. Справді, A визначається так:

$$D(A) = \{w = (w_1, w_2) \in H^2(\Omega) \times H^2(\Omega) : \frac{\partial w_1}{\partial \nu} = \frac{\partial w_2}{\partial \nu} = 0 \text{ на } \partial\Omega\},$$

$$Ay = \begin{pmatrix} d_1 \Delta y_1 \\ d_2 \Delta y_2 \end{pmatrix}, \quad y \in D(A).$$

◆ *Завдання 9.4.5.* За припущень із розділу 9.3., сформулювати принцип максимуму для задачі

$$\int_{\Omega} [y_1^u(x, T) + y_2^u(x, T)] dx \longrightarrow \max, \quad (\text{DP2}'')$$

стосовно $u \in L^2(\Omega \times (0, T))$, $0 \leq u(x, t) \leq 1$ д.м.в. $t \in (0, T)$, де (y_1^u, y_2^u) є розв'язком задачі

$$\begin{cases} \frac{\partial y_1}{\partial t} - d_1 \Delta y_1 = r_1 y_1 \left(1 - \frac{y_1}{k}\right) - \mu_1 u(x, t) y_1 y_2, & (x, t) \in \Omega \times (0, T), \\ \frac{\partial y_2}{\partial t} - d_2 \Delta y_2 = -r_2 y_2 + \mu_2 u(x, t) y_1 y_2, & (x, t) \in \Omega \times (0, T), \\ \frac{\partial y_1}{\partial \nu}(x, t) = \frac{\partial y_2}{\partial \nu} = 0, & (x, t) \in \partial\Omega \times (0, T), \\ y_1(x, 0) = y_{01}(x), \quad y_2(x, 0) = y_{02}(x), & x \in \Omega, \end{cases}$$

де k додатна стала.

Бібліографія

- [1] Adams, R.A. *Sobolev Spaces*. Academic Press, New York (1975)
- [2] Allen, L.J.S. *An Introduction to Mathematical Biology*. Pearson Prentice–Hall, Upper Saddle River, NJ (2007)
- [3] Anița, S., Arnăutu, V. Some aspects concerning the optimal harvesting problem. *Sci. Annals Univ. "Ion Ionescu de la Brad" Iași* **48**, 65–73 (2005)
- [4] Anița, S., Arnăutu, V., Ștefănescu, R. Numerical optimal harvesting for a periodic age-structured population dynamics with logistic term. *Numer. Funct. Anal. Optim.* **30**, 183–198 (2009)
- [5] Anița, S., Arnăutu, V., Capasso, V. *An Introduction to Optimal Control Problems in Life Sciences and Economics. From Mathematical Models to Numerical Simulation with MatLab[®]*. Springer New York, Dordrecht, Heidelberg, London (2011).
- [6] Anița, L.-I., Anița, S., Arnăutu, V. Global behavior for an age-dependent population model with logistic term and periodic vital rates. *Appl. Math. Comput.* **206**, 368–379 (2008)
- [7] Anița, S., Capasso, V. A stabilizability problem for a reaction-diffusion system modelling a class of spatially structured epidemic model. *Nonlinear Anal. Real World Appl.* **3**, 453–464 (2002)
- [8] Anița, S., Capasso, V. A stabilization strategy for a reaction-diffusion system modelling a class of spatially structured epidemic systems (think globally, act locally). *Nonlinear Anal. Real World Appl.* **10**, 2026–2035 (2009)
- [9] Anița, S., Capasso, V. Stabilization for a reaction-diffusion system modelling a class of spatially structured epidemic systems. The periodic case. In Sivasundaram, S., et al. (Eds.) *Advances in Dynamics*

- and Control: Theory, Methods, and Applications*, Cambridge Scientific, Cambridge, MA (2009)
- [10] Anița, S., Capasso, V. On the stabilization of reaction-diffusion system modelling a class of man-environment epidemics: A review. *Math. Methods Appl. Sci.* 2009; Special Issue (J. Banasiak et al. Eds.) 1–6 (2009)
- [11] Anița, S., Fitzgibbon, W., Langlais, M. Global existence and internal stabilization for a class of predator-prey systems posed on non coincident spatial domains. *Discrete Cont. Dyn. Syst. B* **11**, 805–822 (2009)
- [12] Anița, S., Iannelli, M., Kim, M.-Y., Park, E.-J. Optimal harvesting for periodic age-dependent population dynamics. *SIAM J. Appl. Math.* **58**, 1648–1666 (1998)
- [13] Anița, S. *Analysis and Control of Age-Dependent Population Dynamics*. Kluwer Academic, Dordrecht (2000)
- [14] Armijo L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pac. J. Math.* 16, 1–3 (1966)
- [15] Arnăutu, V., Neittaanmäki, P. *Optimal Control from Theory to Computer Programs*. Kluwer Academic, Dordrecht (2003)
- [16] Aronson, D.G. The role of diffusion in mathematical population biology: Skellam revisited. In: Levin, S. (Ed.) *Mathematics in Biology and Medicine*, Springer-Verlag, Berlin (1985)
- [17] Barbu, V., Precupanu, T. *Convexity and Optimization in Banach Spaces*. D. Reidel, Dordrecht (1986)
- [18] Barbu, V. *Analysis and Control of Nonlinear Infinite Dimensional Systems*. Academic Press, Boston (1993)
- [19] Barbu, V. *Mathematical Methods in Optimization of Differential Systems*. Kluwer Academic, Dordrecht (1994)
- [20] Barbu, V. *Partial Differential Equations and Boundary Value Problems*. Kluwer Academic, Dordrecht (1998)
- [21] John T. Bett. *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.) - Society for Industrial and Applied Mathematics. Philadelphia, 2010 – 449.

-
- [22] Borrelli, R.L., Coleman, C.S. *Differential Equations. A Modeling Perspective*. John Wiley & Sons, New York (1998)
- [23] Brokate, M. Pontryagin's principle for control problems in age-dependent population dynamics. *J. Math. Biol.* **23**, 75–101 (1985)
- [24] Burghes, D., Graham, A. *Introduction to Control Theory, Including Optimal Control*. Ellis Horwood Division of John Wiley & Sons, Chichester, UK (1980)
- [25] Butler, S., Kirschner, D., Lenhart, S. Optimal control of chemotherapy affecting the infectivity of HIV. In: Arino, O., Axelrod, D., Kimmel, M. (Eds.) *Advances in Mathematical Population Dynamics—Molecules, Cells and Man*. World Scientific, Singapore (1997)
- [26] Brezis, H. *Analyse Fonctionnelle. Théorie et Applications*. Masson, Paris (1983)
- [27] Capasso, V. Asymptotic stability for an integro-differential reaction-diffusion system. *J. Math. Anal. Appl.* **103**, 575–588 (1984)
- [28] Capasso, V. *Mathematical Structures of Epidemic Systems*. Lecture Notes in Biomathematics, Vol. 97, 2nd corrected printing, Springer, Heidelberg (2008)
- [29] Céa, J. *Optimisation. Théorie et Algorithmes*. Dunod, Paris (1971)
- [30] Céa, J. *Lectures on Optimization. Theory and Algorithms*. Tata Institute of Fundamental Research, Springer-Verlag, Bombay (1978)
- [31] Cherruault, Y. *Mathematical Modelling in Biomedicine. Optimal Control of Biomedical Systems*. D. Reidel, Dordrecht (1986)
- [32] Cusulin, C., Iannelli, M., Marinoschi, G. Age-structured diffusion in a multi-layer environment. *Nonlinear Anal. R. World. Appl.* **6**, 207–223 (2005)
- [33] Cruceanu, Ș., Vârsan, C. *Elements of Optimal Control and Applications in Economics* (in Romanian). Editura Tehnică, Bucharest (1978)
- [34] Capasso, V., Wilson, R.E. Analysis of a reaction-diffusion system modelling man–environment–man epidemics. *SIAM J. Appl. Math.* **57**, 327–346 (1997)

-
- [35] Dorn, W.S., McCracken, D.D. *Numerical Methods with FORTRAN IV. Case Studies*. John Wiley & Sons, New York (1972)
- [36] Fisher, R.A. The wave of advance of advantageous genes. *Ann. Eugen.* **7**, 355–369 (1937)
- [37] Fister, K.R., Lenhart, S. Optimal harvesting in an age-structured predator-prey model. *Appl. Math. Optim.* **54**, 1–15 (2006)
- [38] Istas, J. *Mathematical Modeling for the Life Sciences*. Springer-Verlag, New York, (2005)
- [39] Gourley, S.A. Travelling front solutions of a nonlocal Fisher equation. *J. Math. Biol.* **41**, 272–284 (2000)
- [40] Gurtin, M.E., Murphy, L.F. On the optimal harvesting of age-structured populations: some simple models. *Math. Biosci.* **55**, 115–136 (1981)
- [41] Glashoff, K., Sachs, E. On theoretical and numerical aspects of the bang-bang principle. *Numer. Math.* **29**, 93–113 (1977)
- [42] Gurtin, M.E., Murphy, L.F. On the optimal harvesting of age-structured populations: some simple models. *Math. Biosci.* **55**, 115–136 (1981)
- [43] Genieys, S., Volpert, V., Auger, P. Pattern and waves for a model in population dynamics with nonlocal consumption of resources. *Math. Model. Nat. Phenom.* **1**, 65–82 (2006)
- [44] Ho, D.D., Neumann, A.U., Perelson, A.S., Chen, W., Leonard, J.M., Markowitz, M. Rapid turnover of plasma virions and CD4 lymphocytes in HIV-1 infection. *Nature* **373**, 123–126 (1995)
- [45] Hritonenko, N., Yatsenko, Y. Optimization of harvesting age in integral age-dependent model of population dynamics. *Math. Biosci.* **195**, 154–167 (2005)
- [46] Brian R. Hunt, Ronald L. Lipsman, John E. Osborn, Jonathan M. Rosenberg *Differential Equations with MatLab[®]. Updated for MatLab[®] 2011b (7.13), Simulink[®] 7.8, and Symbolic Math ToolboxTM 5.7 (Third Edition)*. - John Wiley & Sons, Inc. New York (2012) – 296

-
- [47] B.Hunt, R.Lipsman, J.Rosenberg, K.Coombes, J. Osborn, G.Stuck A Guide to MatLab[®] for Beginners and Experienced Users. Updated for MatLab[®] 7 and Simulink[®] 6 (Second Edition). - Cambridge University Press. The Edinburgh Building, Cambridge cb2 2ru, UK (2006) – 329.
- [48] Iannelli, M. *Mathematical Theory of Age-Structured Population Dynamics*. Giardini Editori e Stampatori, Pisa (1995)
- [49] Iannelli, M., Marinocchi, G. Harvesting control for an age-structured population in a multi-layered habitat. *J. Optim. Theory Appl.* **142**, 107–124 (2009)
- [50] Kirschner, D., Lenhart, S., Serbin, S. Optimizing chemotherapy of HIV infection: scheduling, amounts and initiation of treatment. *J. Math. Biol.* **35**, 775–792 (1997)
- [51] Knowles, G. *An Introduction to Applied Optimal Control*. Academic Press, London (1981)
- [52] Kolmogorov, A.N., Petrovskii, I.G., Piskunov, N.S. Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique. *Bull. Univ. État Moscou, Sér. Int., Sect. A: Math et Mécan.* **1**, 1–25 (1937)
- [53] Lions, J.L. *Optimal Control of Systems Governed by Partial Differential Equations*. Springer-Verlag, Berlin (1972)
- [54] Luo, Z., Li, W.T., Wang, M. Optimal harvesting control problem for linear periodic age-dependent population dynamics. *Appl. Math. Comput.* **151**, 789–800 (2004)
- [55] Lee, E.B., Markus, L. *Foundation of Optimal Control Theory*. John Wiley, New York (1967)
- [56] Lenhart, S., Workman, J.T. *Optimal Control Applied to Biological Models*. Chapman & Hall/CRC, Boca Raton, FL (2007)
- [57] Murray, J.D. *Mathematical Biology*. Springer-Verlag, Berlin (1989)
- [58] Okubo, A. Dynamical aspects of animal grouping: swarms, schools, flocks and herds. *Adv. Biophys.* **22**, 1–94 (1986)
- [59] Okubo, A. *Diffusion and Ecological Problems: Mathematical Models*. Springer-Verlag, Berlin (1980)

-
- [60] Perelson, A.S., Kirschner, D.E., De Boer, R. Dynamics of HIV infection of CD4 + T cells. *Math. Biosci.* 114, 81–125 (1993)
- [61] Polak, E. *Computational Methods in Optimization. A Unified Approach.* Academic Press, New York (1971)
- [62] Perelson, A.S., Nelson, P.W. Modeling viral infections. In: Sneyd, J. (Ed.) *An Introduction to Mathematical Modeling in Physiology, Cell Biology, and Immunology.* AMS, Providence, RT (2002)
- [63] Protter, M.H., Wienberger, H.F. *Maximum Principles in Differential Equations.* Springer-Verlag, New York (1984)
- [64] Skellam, A.J. Random dispersal in theoretical populations. *Biometrika* **38**, 196–218 (1951)
- [65] Skellam, J.G. The formulation and interpretation of mathematical models of diffusional processes in population biology. In: Bartlett, M.S., Hiorns, R.W. (Eds.) *The Mathematical Theory of the Dynamics of Biological Populations.* Academic Press, New York (1973)
- [66] Smoller, J. *Shock Waves and Reaction–Diffusion Equations.* Springer-Verlag, New York (1983)
- [67] Sontag, E.D. *Mathematical Control Theory. Deterministic Finite Dimensional Systems*, 2nd edition. Springer-Verlag, New York (1998)
- [68] Stormy Attaway MatLab[®]. *A Practical Introduction to Programming and Problem Solving (Second Edition).* Elsevier, 2012. - 524.
- [69] Thieme, H. Renewal theorems for linear periodic Volterra integral equations. *J. Integral Eqs.* **7**, 253–274 (1984)
- [70] Thieme, H. *Mathematics in Population Biology.* Princeton University Press, Princeton NJ and Oxford (2003)
- [71] Trélat, E. *Contrôle Optimal. Théorie et Applications.* Vuibert, Paris (2005)
- [72] Webb, G. *Theory of Nonlinear Age-Dependent Population Dynamics.* Marcel Dekker, New York (1985)
- [73] Yosida, S. An optimal control problem of the prey–predator system. *Funkt. Ekvacioj* **23**, 283–293 (1982)

-
- [74] Ануфриев И.Е., Смирнов А.Б., Смирнова Е.Н. MATLAB 7. – Санкт-Петербург: БХВ-Петербург, 2005. 1104 с.: ил.
- [75] Дьяконов В. П. MATLAB 7.*/R2006/R2007: Самоучитель. – Москва: ДМК Пресс, 2008. 768 с.: ил.
- [76] Кетков Ю.Л., Кетков А.Ю., Шульц М.М. MATLAB 7: программирование, численные методы. – Санкт-Петербург: БХВ-Петербург, 2005. 752 с: ил.
- [77] Матвеев А.С., Якубович В.А. Оптимальные системы управления: Обыкновенные дифференциальные уравнения. Специальные задачи: учебн. пособие – Санкт-Петербург: Из-во С.-Петербургского ун-та. 2003. 540.
- [78] Розоноэр Л.И. Принцип максимума Л. С. Понтрягина в теории оптимальных систем. I, Автомат. и телемех., 1959, том 20, выпуск 10, 1320–1334
- [79] Розоноэр Л.И. Принцип максимума Л. С. Понтрягина в теории оптимальных систем. II, Автомат. и телемех., 1959, том 20, выпуск 11, 1441–1458
- [80] Розоноэр Л.И. Принцип максимума Л. С. Понтрягина в теории оптимальных систем. III, Автомат. и телемех., 1959, том 20, выпуск 12, 1561–1578
- [81] Хайпер Э. Нёрсетт С. Ваннер Г. *Решение обыкновенных дифференциальных уравнений. Нежесткие задачи*: Пер. с англ. – Москва: Мир, 1990. 512 с.
- [82] <http://www.math.hmc.edu/resources/odes/odearchitect/examples/>

Предметний покажчик

- алгоритм
 - методу найшвидшого спуску, 270
 - методу проєкції градієнта (метод Rosen'a), 383
 - методу проєкції градієнта, 309, 360
 - Uzawa, 289
- анонімна функція, 89, 92
- апроксимація, 166
 - дискретна, 166
 - лінійна, 166
 - неперервна (інтегральна), 166
 - похибка, 166
 - середньоквадратична, 167
 - функції, 166
- базові функції, 166
- вузли інтерполяції, 162
- задача
 - аксимізації загального споживання, 219
 - керування запасами, 287
 - лінійного програмування, 132
 - оптимального розмноження, 354
 - оптимального керування розмноженням, 303
 - оптимального розмноження, 303, 337
 - спряжена (рівняння), 215
- збіжність
 - на відрізьку, 153
 - різницевих методів, 153
- інтерполяційна
 - формула Lagrange'a, 165
 - функція, 162
- керування
 - (регулятор), 214
 - двопозиційне (bang-bang control), 227
 - двопозиційне оптимальне, 227
 - оптимальне, 227
 - системою реакції-дифузії, 381
 - у SIR моделі, 255
- коефіцієнт репродукції, 330
- матриця
 - визначник, 111
 - із одиничними елементами, 115
 - конкатенація, 110
 - обернена, 119
 - одинична, 114
 - ранг, 112
 - слід, 113
 - транспонована, 113
 - з нульовими елементами, 116
- метод
 - найменших квадратів, 167
 - найшвидшого спуску, 263
 - неявний, 153
 - проєкції градієнта (Projected Gradient Method (PGM)), 360
 - чисельної стабілізації (NSM),

- 358
 Armijo, 277
 Euler'a, 153
 Euler'a із перерахунком, 155
 Euler'a модифікований, 155
 Heun'a, 155
 Rosen'a, 380
 Runge-Kutta четвертого поряд-
 ку класичний, 156
 Uzawa, 379
- методи
 Runge-Kutta другого порядку,
 155
 Runge-Kutta третього порядку,
 155
- модель
 динаміки популяції із віковою
 структурою, 320
 інсулін-глюкозової системи, 241
 інсулінової терапії, 240
 із періодичним показником при-
 родної зміни популяції, 350
 максимізації випуску комплек-
 тної продукції, 138
 оптимальних сумішей, 138
 оптимального розкрою матеріа-
 лів, 139
 техніко-економічного планува-
 ння, 135
 хижак-жертва, 190
- необхідні умови оптимальності пер-
 шого порядку, 217, 234
 оптимальна пара, 214, 222, 230
 оптимальне значення функціоналу
 корисності, 214
 осциляція у бокалі, 208
 порядок p точності методу, 153
 принцип Понтрягіна, 212
- рівняння
 різницеве, 153
 стану, 214
- Fitzhugh–Nagumo, 190
 Lotka (рівняння відновлення),
 325
- розв'язок
 оптимальний, 214
 Carathéodory, 213
- схема
 методу Euler'a симетрична, 153
 різницева, 153
 Euler'a, 152
- система
 хижак-жертва, 185, 227
 Lotka-Volterra, 227
- скалярний добуток, 126
 спряжений стан, 216
 стан, що відповідає керуванню, 214
- таблиця Butcher'a, 154
 терапія ВІЛ, 252
- точка перемикання, 227, 236
 узагальнений многочлен, 166
- формула
 рекурентна, 153
- функціонал корисності, 214
- функція
 апроксимуюча, 166
 наближення, 166
 сіткова, 153

Навчальне видання

Кирилич Володимир Михайлович
Терещук Оксана Володимирівна
Флюд Володимир Михайлович

**Оптимальне керування соціально-економічними
системами у середовищі MatLab®**

Навчальний посібник

Редактор *Н. Й. Плиса*

Комп'ютерний набір і верстання *В. М. Флюд*

Підп. до друку __. __. 2021. Формат __ × __/__. Папір друк. Друк офсет.
Гарнітура ТрХ. Умовн. друк. арк. __. __. Обл.-вид. арк. __. __. Тираж ___
прим.

Видавець та виготовлювач:

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, м. Львів, 79000

СВІДОЦТВО

*про внесення суб'єкта видавничої справи до Державного реєстру видавців,
виготівників і розповсюджувачів видавничої продукції: Серія ДК №3059 від
13.12.2007 р.*