Стефка Караколева Евелина Велева

ВИСША МАТЕМАТИКА 3

Практикум по „Числени методи“ с *MATLAB*

Русенски Университет „Ангел Кънчев“

Русе, 2004

УДК 519.6 (076)

Караколева,С.Р., Велева,Е.В.
Висша математика 3: Практикум по „Числени методи“ с MATLAB, Русенски Университет „Ангел Кънчев“, гр.Русе, 2004.

В книгата са разгледани основните Числени методи с приложение в инженерната практика: решаване на системи линейни алгебрични уравнения, числено решаване на нелинейни уравнения и системи, апроксимиране на функция, метод на най-малките квадрати, числено решаване на определен интеграл, диференциални уравнения и системи, линейна гранична задача ОДУ от ред 2, статистически методи.

Теоретичният материал се илюстрира с конкретни практически задачи, които се решават в програмната среда MATLAB.

Съдържанието на книгата е съобразено с материала, преподаван в РУ „ Ангел Кънчев“ и сходни български университети. Използвани са най-новите разработки на алгоритми и методи, публикувани в американски, руски и български издания на тази тема.

Книгата е специално подготвена за практически упражнения по дисциплината „Висша математика - 3 част“, изучавана от студентите от всички инженерни специалности в Русенския университет и е от типа «работна книга». Може да бъде използвана от студенти - степен „ бакалавър“ и „ магистър“, дипломанти, докторанти, инженери, научни работници.

Рецензент: доц. д-р Любен Г. Вълков

ISBN 954-712-245-2

Дизайн и предпечатна подготовка с MiKTEX

СЪДЪРЖАНИЕ

СПИСЪК НА ФИГУРИТЕ

СПИСЪК НА ТАБЛИЦИТЕ

УВОД

Проект и експеримент

При изследване на различни процеси в науката и техниката, инженерът се интересува от математиката не като наука сама за себе си, а като инструмент за решаване на конкретни практически задачи. Например:

- ❑ разработване на проект за жилищна сграда;
- ❑ конструиране на нова подемна машина;
- ❑ определяне на производителността на нова поточна линия;
- ❑ създаване на нов препарат за растителна защита.

Един от начините за решаване на всяка от тези задачи е **експериментът**: построява се сграда, конструира се нова машина, изгражда се поточна линия, получава се по химичен път ново вещество. Ако резултатът от експеримента се окаже неудачен, се повтаря отново и т.н. Ясно е, че по този начин може да се получи отговор на конкретен въпрос, но с цената на много време, средства, дори и жертви.

Друг, по-разумен начин е **математическият анализ** на проекта. Той се прилага не към реалните явления, а към техния **математически модел** и е по своята същност **теоретичен експеримент**. Математическият модел е връзка

Той е средство, чрез което практическата задача се решава, анализира, контролира. Изследването с помощта на модел често е единственият възможен способ за изучаване и решаване на важни практически задачи.

Основни етапи на моделирането

☞ **Формулиране на задачата**

Това е един от най-важните етапи, защото не може да се получи точен отговор на неточно поставена задача.

☞ **Построяване на математически модел**

След формулиране на задачата, тя трябва да бъде приведена в удобна за анализ форма. Това става чрез построяване на математически модел, който съдържа в себе си само най-съществено от задачата.

Обект на изучаване може да е целият процес или отделни негови части. Състоянието на обекта на моделиране може да бъде описано с някакво множество размерни величини

$$x_1, x_2, \ldots, x_n, \quad y_1, y_2, \ldots, y_m, \quad a_1, a_2, \ldots, a_p.$$

Тези величини се разделят на две групи:

- променливи на процеса x_i, y_j, $i = \overline{1,n}$, $j = \overline{1,m}$, числовите стойности на които в хода на процеса се изменят в някаква област;
- параметри a_k, $k = \overline{1,p}$, които в дадения процес са постоянни, но могат да имат друга стойност в аналогични процеси.

От друга страна, променливите на модела се разделят на входни (x_i) и изходни (y_j), като вида на съотношението между тях

$$y_j = F(x_i, a_k)$$

зависи от вида на изучавания процес.

Трябва да се има предвид, че математическият модел е идеално приближение на действителността, т.е. при построяване на модела се използват някои предположения, които го опростяват. Това се прави, за да бъде моделът разрешим. Разбира се, опростяването е до такава степен, че моделът да отразява вярно основните закономерности на изследвания процес.

☞ **Решаване на модела**

В зависимост от сложността на модела се използват различни математически методи за неговото решаване.

За най-простите и груби модели вероятно може да се получи **аналитично решение**, но за по-точни и сложни модели решение в аналитична форма се получава рядко. В такава ситуация могат да се използват **приближени математически методи** (напр. разлагане по малък параметър, усредняване, изучаване на асимптотики и др.) Тези прийоми позволяват да се представи някакво приближено решение на задачата в аналитична форма, след което от него да се получат числени резултати като се използват числени методи.

Основни методи за решаване на най-точните и сложни модели са **Числените методи**, за чието прилагане е необходима и компютърна техника.

> Числените методи са *мощно средство на математиката* за решаване на различни практически задачи. Успешното решаване на дадена задача е възможно само при *добро познаване* на основните числени методи.

Използването на компютри в процеса на математическото изследване на модела изисква специфични числени методи, т.е. такава „интерпретация“ на математическия модел, която да може да бъде реализирана с компютър, т.нар. **«дискретен модел»**. Тъй като електронно-изчислителните машини изпълняват само аритметични и логически операции, то за реализация на дискретния модел се разработват и съответните **изчислителни алгоритми**.

Изборът на числения метод и алгоритъма трябва да е съобразен с възможностите на машината (време за изпълнение на операциите, обем оперативна памет). Формата на представяне на алгоритъма зависи от езика на програмиране. За решаване на сложни задачи обикновено се съставя описание на процеса на решаване във вид на структурни схеми, след което следва програмиране, кодиране, логическо тестване на програмата в частни случаи, техническо описание и документация за потребителя. Това е дълъг процес, който изисква задълбочени познания по програмиране и числени методи, които не всеки потребител има.

Но на пазара на компютърните технологии има достатъчно богат избор. Съвременната компютърна математика предлага цял набор «интегрирани» програмни системи и програми за автоматизация на математическите изчисления: Gauss, MathCad, Mathematica, Maple V, Matlab и др.

☞ Проверка на модела и решението

Трябва да се има предвид, че не трябва да се разчита само на интуицията на създателите на модела. Затова трябва да се провери моделът за очевидни грешки и дали той съответства на моделирания процес.

Счита се, че математическият модел е **идентичен** или **адекватен** на моделирания процес, ако някои негови характеристики (динамични, статистически и др.), получени с компютър, съвпадат с експерименталните данни със зададена степен на точност. Като критерий за идентичност се използват различни оценки за разликата между теоретичните и експерименталните характеристики (математическо очакване и моменти на случайна величина).

Полезен подход при проверка на модела е и **ретроспекцията**. Това означава да се използват входни данни, за които решението е вече известно, след което те да се сравнят с експерименталните данни.

☞ **Контрол на решението**

Нека да предположим, че след редица проверки някакъв модел и неговото решение се оказват приемливи. Да предположим също, че решението се използва многократно в практиката. Очевидно е, че решението може да се използва докато моделът е актуален. Но при промяна на условията може да се окаже, че моделът е грешен (например стойностите на входните параметри могат да се променят съществено). Затова е важно тези промени да бъдат уловени навреме и да се предприемат действия за модифицирането на модела.

Разумно е да се подготвят процедури за контрол на решението. За това е необходимо да се определят критични стойности, при които решението се изменя съществено. Това може да се направи чрез анализ на чувствителността на параметрите. По-нататък може да се използва статистически подход (както при контрол на качеството), за да се улови такава промяна в параметрите.

☞ **Прилагане на решението в практиката**

Последен етап при решаването на някаква задача е да се приложи на практика полученият модел. Това е един от най-критичните моменти в цялото изследване на процеса, защото именно тук се вижда ползата от него. Затова е важно, специалистите, участвали при разработването на модела, да вземат донякъде участие и в този етап, за да могат да отстранят недостатъци, които се проявяват в практиката.

В книгата се разглеждат основните числени методи с приложение в инженерната практика, като се набляга на практическото им използване в програмната среда Matlab. Темите са разделени в 10 глави, разработени съответно от Стефка Караколева – Глави 1, 2, 7, 9, 10; Евелина Велева – Глави 3, 4, 5, 6, 8.

Използвани са характерни символи със следното значение: ⌨ Въведете! ✍ Попълнете! ➢ Теорема ➠ Дефиниция ☞ Пример ✎ Задача

Благодарим на д-р инж. Ивелин Иванов за ценните съвети при редактирането на Задачи 9.3 и 9.5.

Стефка Р. Караколева
e-mail: skarakoleva@ru.acad.bg

Евелина И. Велева
e-mail: eveleva@ecs.ru.acad.bg

Русе, 2004

ГЛАВА 1

ВЪВЕЖДАНЕ В ПРОГРАМНАТА СРЕДА MATLAB

Системата Matlab (MATrix LABoratory) е интерактивна система за инженерни и научни изчисления, с използване на масиви от данни. Системата използва математически ко-процесор и има възможност за обръщане към програми, написани на езиците FORTRAN и C++.

Системата Matlab е особено популярна сред студенти, научни работници и инженери, защото е лесна за употреба, дори за начинаещи.

Системата Matlab е създадена от фирмата MathWorks (САЩ, г. Нейтик, щат Масачузетс) в края на 70-те години на XX век. Matlab се разработва повече от 15 години и е възникнал на основата на пакетите LINPACK и EIGPACK и на свой ред е повлиял на създаването на системите MathCad и Mathematica. С времето Matlab се допълва с много приложения (toolboxes), разширяващи неговата приложимост, [1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 17, 18, 21, 22, 25, 27].

Matlab е една от най-съвършените и проверени от времето системи. Това е уникална колекция на съвременни числени методи на компътърната математика, създадени през последните три десетилетия. Тя е събрала в себе си опита, правилата и методите за математически пресмятания, натрупвани хиляди години. Това се съчетава с мощни средства за графична визуализация и дори анимация на графика. С прилаганата към нея обширна документация, системата Matlab може да се използва и като фундаментален електронен справочник по математическо обезпечение на компютъра.

Работата в средата на Matlab се осъществява в 2 режима:

- в режим „калкулатор“;
- чрез извикване на програми, създадени на езика на Matlab и записани на диска (програмен режим).

И в двата режима са достъпни всички изчислителни възможности на системата: операции с вектори, матрици и масиви от данни, ранг и число на обусловеност на матрица, работа с алгебрични полиноми, решаване на нелинейни системи уравнения и задачи за оптимизация, интегриране в квадратури, решаване на диференциални и диференчни уравнения, построяване на графики, тримерни повърхнини и линии на ниво.

Основна отличителна черта на системата е лесното адаптиране на конкретната задача на потребителя. Потребителят може да въведе в системата нова команда, оператор или функция и да ги използва по същия начин, както и вградените функции. При това, за разлика от езиците за програмиране `Pascal` или `C`, не е необходимо предварително описание. Новите програми, функции и процедури в системата Matlab се съхраняват във вид на файлове с разширение `.m`

Последните версии на Matlab 6.x позволяват лесна връзка с текстовия редактор WORD, което дава възможност, при създаване на текстови документи да се използват изчислителните и графични възможности на Matlab.

В книгата се разглежда само **ядрото** на Matlab, т.нар *система* и конкретно Matlab6. Практически всички примери са приложими за всички по-ранни версии, като се започне от 4.0 (1994г.)

1.1 Аритметични операции

Matlab-операторите за четирите аритметични операции са + за събиране, − за изваждане, ∗ за умножение, / и \ съответно за дясно и ляво деление. Когато операндите са числа, резултатът при двете операции е един и същ. В Глава 2 се дава как се използват ляво и дясно деление между вектори и матрици. Следните примери представят тези операции. Въведете и попълнете!

```
>>2+3
......
>>2-2
......
>>2*2
......
```

```
>>2/3
......
>>5*5*5
......
>>5^3
......
```

```
>>5^(2.5)
......
>>5^(-3)
......
>>11*(15/11)-15
......
```

1.2 Някои вградени функции

В Таблица 1.1 са дадени някои вградени Matlab-функции. В следващите примери се използват някои от тях. Въведете и попълнете!

Математически функции	Matlab функции	Стойности на аргумента
$sin\ x$	`sin(x)`	$x_{rad}\ ,\ -\infty < x < \infty$
$cos\ x$	`cos(x)`	$x_{rad}\ ,\ -\infty < x < \infty$
$tg\ x$	`tan(x)`	$x_{rad}\ ,\ x \neq (2k+1)\pi/2,\ k = 0, \pm 1, \pm 2 \ldots$
$arcsin\ x$	`asin(x)`	$-\pi/2 < x < \pi/2$
$arccos\ x$	`acos(x)`	$0 < x < \pi$
$arctg\ x$	`atan(x)`	$-\pi/2 < x < \pi/2$
$ln\ x$	`log(x)`	$x > 0$
$lg\ x$	`log10(x)`	$x > 0$
e^x	`exp(x)`	$-\infty < x < \infty$
$\sqrt{x}$	`sqrt(x)`	$x \geq 0$

Таблица 1.1: Вградени функции

```
>>sin(pi/6)
.........
>>sin(pi)
.........
>>cos(0)
.........
>>tan(pi/4)
.........
>>log(1)
.........
>>asin(1)
.........
>>acos(1)
.........
>>atan(1)
.........
>>exp(2)
.........
>>log10(2)
.........
```

1.3 Имена на променливи

Matlab дава възможност на потребителя да задава свои имена на константи и променливи. В следващия пример се демонстрира използването им.

☞ **Пример 1.1** *Да се пресметнат* $\sin 30^0$, $\sin 52^0$, $\sin 76^0$, *като се използва, че* $x_{rad} = \frac{\pi}{180} x^0$. ⌨ ✍

```
>>alpha=30;
>>beta=52;
>>gamma=76;
>>conf=pi/180;
>>alpha=conf*alpha
.........
>>beta=conf*beta
.........
>>gamma=conf*gamma
.........
>>sin(alpha)
.........
>>sin(beta)
.........
>>sin(gamma)
.........
```

В Matlab могат да се използват имена на променливи, най-много до 19 символа, като по подразбиране Matlab различава малките и големите букви. Команда-

та `casesen off` изключва различаването на малки и големи букви, `casesen on` отново го включва.

```
>>a=2;
>>A=3;
>>2*a
......
```

```
>>2*A
......
>>casesen off
>>a=2;
```

```
>>A=3;
>>2*a
......
>>2*A
```

```
......
```

1.4 Едномерни масиви

Наредено множество $a_1, a_2, \ldots, a_n$ в Matlab се дефинира като масив $A = [a_1, a_2, \ldots, a_n]$.

```
>>prime=[2 3 5 7 11 13]
.........
>>prime(1)
.........
>>length(prime)
.........
>>clear prime
>>prime(1)=2
.........
>>prime(2)=3
.........
>>prime(3)=5
```

```
.........
>>prime(4)=7
.........
>>prime(5)=11
.........
>>prime(6)=13
.........
>>natural=[1,2,3,4,5,6]
.........
>>prime+natural
.........
>>prime-natural
```

```
.........
>>odd=1:2:11
.........
>>even=2:2:12
.........
>>natural=1:6
.........
>>inv_odd=9:-2:1
.........
>>halves=0:.5:10
.........
```

1.5 Операции с едномерни масиви

В Matlab се използват следните операции с едномерни масиви.

`.*` - поелементно умножение;
`./` - поелементно деление;
`.^` - поелементно степенуване;

`'` - транспониране;
`*` - скаларно умножение:
$(m \times n) * (n \times p) = (m \times p)$

Следните примери показват как се използват тези операции.

```
>>natural.*prime
.........
>>natural./prime
.........
>>natural.^2
.........
>>natural*prime'
```

```
.........
>>prime*prime'
.........
>>sum(prime.^2)
.........
>>u=[2;3;5;7;11;13]
.........
```

```
.........
>>u*u'
.........
.........
.........
```

1.6 Матрици, подматрици, операции с матрици

Въвеждането на матрици е подобно на въвеждане на едномерни масиви. Елементите на един ред се отделят с интервал или запетая, а отделните редове - с ; или натикане на клавиш Enter.

```
>>A=[1 2 3;4 5 6;7 8 9]
.........
.........
.........
>>A(2,1)
.........
>>size(A)
.........
>>A=[1,2,3
4,5,6
7,8,9]
>>B=[9,8,7
6,5,4;3,2,1]
>>[A B]
.........
.........
.........
>>size(ans)
.........
>>[A;B]
.........
.........
.........
>>size(ans)
.........
```

```
>>v=A(1,:),w=A(:,1)
.........
.........
>>A(1:2,2:3)
.........
.........
>>C=[10 11;12 13;14 15]
.........
.........
>>S=A+B
.........
.........
.........
>>D=A-B
.........
.........
.........
>>A*B
.........
.........
.........
>>A*C
.........
.........
```

```
>>A^2
.........
.........
.........
>>[m,n]=size(A);
>>ones(m,n)
.........
.........
.........
>>zeros(1,5)
.........
>>eye(4)
.........
.........
.........
.........
>>det(A)
.........
>>inv(A)
.........
.........
.........
.........
```

1.7 Графики

Графика на функция на една променлива $y = f(x)$.

```
>>x=0:0.1:3;
>>y=cos(x)+5.*exp(3*x)-7;
>>plot(x,y)
>>grid
```

Графика на функция, въведена като M-файл.

```
>>fplot('sin',[0,5])
```

```
>>[x,y]=fplot('sin',[0,5]);
>>plot(x,y,'r')
>>help plot
>>title('plot of sin(x)')
>>xlabel('x values')
>>ylabel('y values')
```

Графика на функция, зададена с полярни координати.

```
>>a=1;
>>theta=0:pi/60:2*pi;
>>rho=a*theta;
>>polar(theta,rho)
>>grid
>>title('The spiral of Archimedes, rho=a*theta')
```

Триизмерна графика.

```
>>x=-5:0.35:5;y=x;
>>[a,b]=meshgrid(x,y);
>>z=a.^2+b.^2;
>>w=cos(sqrt(z));
>>v=sin(sqrt(z)+.001)./(sqrt(z)+.001);
>>mesh(w)
>>mesh(v)
>>contour(v,50)
```

Анимация.

```
>>p=0:pi/60:8*pi;
>>d=2;A=0.2;T=5;
>>omega=2*pi/T;
>>x=d*cos(p)/2;y=d*sin(p)/2;
>>z=(1+A*cos(omega*1))*p;
>>plot3(x,y,z)
>>M=moviein(6);
>>for t=1:6
    x=d*cos(p)/2;y=d*sin(p)/2;
    z=(1+A*cos(omega*(t-1)))*p;
    plot3(x,y,z)
    axis([-1 1 -1 1 0 10.0*pi]);
    M(:,t)=getframe;
  end
>>movie(M,10)
```

ГЛАВА 2

РЕШАВАНЕ НА СИСТЕМИ ЛИНЕЙНИ АЛГЕБРИЧНИ УРАВНЕНИЯ

➠**Дефиниция 1** *Система от вида*

$$\left|\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m, \end{array}\right. \tag{2.1}$$

се нарича система от m линейни уравнения с n неизвестни. Числата a_{ij} се наричат коефициенти пред неизвестните x_j, $j = \overline{1,n}$, а b_i, $i = \overline{1,m}$ - свободни коефициенти.

➠**Дефиниция 2** *Решение $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ на системата (2.1) е такъв вектор, че ако неговите координати се заместят вместо $x_1, x_2, \ldots, x_n$ в левите страни на уравненията на системата, се получават тъждествени равенства.*

При решаването на система линейни алгебрични уравнения са възможни различни случаи, които възникват в зависимост от това

- ✓ дали броят на неизвестните n е по-малък, равен или по-голям от броя на уравненията m;
- ✓ дали всички свободни коефициенти b_i са равни на нула (*хомогенна* система) или има поне един различен от нула свободен коефициент (*нехомогенна* система);
- ✓ дали рангът на матрицата от коефициентите пред неизвестните е равен на ранга на разширената матрица или не;
- ✓ дали броят на неизвестните е равен или не на ранга на матрицата от коефициентите пред неизвестните и на ранга на разширената матрица.

2.1 Системи нехомогенни уравнения

➢**Теорема 2.1 (Кронекер-Капелли)** *Разглежда се система нехомогенни уравнения в матричен вид:*

$$A\mathbf{x} = \mathbf{b}, \tag{2.2}$$

където

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

① *Ако* $rank(A) = rank(A \mid \mathbf{b}) = r$, *то (2.2) има решение. При това:*

- ☞ *ако* $r = n$, *системата (2.2) има единствено решение;*
- ☞ *ако* $r < n$, *то (2.2) има безброй много решения, като* r *неизвестни са базисни, а останалите* $(n - r)$ *неизвестни - свободни. В този случай системата се решава спрямо базисните неизвестни, като на свободните неизвестни се дават произволни стойности, т.е. решението е функция на* $(n - r)$ *константи.*

② *Ако* $rank(A) \neq rank(A \mid \mathbf{b})$, *то (2.2) няма решение.*

2.1.1 Графичен метод

В случай, че в системата участват две променливи ($n = 2$), тя може да бъде решена графично.

☞ **Пример 2.1** *Да се решат графично системите линейни алгебрични уравнения:*

$$\text{а).} \left| \begin{array}{rrr} 2x_1 & -x_2 & = 2 \\ x_1 & +x_2 & = 5 \end{array} \right. \quad \text{б).} \left| \begin{array}{rrr} 2x_1 & -x_2 & = 2 \\ 2x_1 & -x_2 & = 0 \end{array} \right. \quad \text{в).} \left| \begin{array}{rrr} 2x_1 & -x_2 & = 2 \\ x_1 & +x_2 & = 5 \\ 6x_1 & -x_2 & = -5 \end{array} \right.$$

Решение: а).  

```
>>subplot(221),axis([0,5,-2,8])
>>plot([0,5],[-2,8]),hold on
>>plot([0,5],[5,0]),grid
```

б).

```
>>subplot(222),axis([0,1,-2,2])
>>plot([0,1],[-2,0]),hold on
>>plot([0,1],[0,2]),grid
```

в).

```
>>subplot(223),axis([-2,5,-10,20])
>>plot([-2,5],[-6,8]),hold on
>>plot([-1,5],[6,0]),grid
>>plot([-2,1],[-7,11])
```

2.1.2 Системи нехомогенни уравнения при r = n

В този случай системата (2.2) има единствено решение, което може да се намери по различни начини, изложени накратко. Най-общо методите се разделят на 2 типа: **точни и итерационни**, [11, 13, 24, 26].

Точни методи за решаване на нехомогенни системи уравнения

Нека $rank(A) = rank(A \mid \mathbf{b}) = r$ е равен на броя на неизвестните n.

☞ **Пример 2.2** *Да се реши системата* $(n = 3)$.

$$\left|\begin{array}{rrrl} x_1 & +2x_2 & +3x_3 & = 4 \\ 2x_1 & +3x_2 & +4x_3 & = 5 \\ 4x_1 & +2x_2 & +5x_3 & = 1 \end{array}\right. \tag{2.3}$$

Решение: Системата (2.3) се записва в матричен вид $A\mathbf{x} = \mathbf{b}$, където:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 2 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

```
>>A=[1,2,3;2,3,4;4,2,5];
>>b=[4;5;1];
>>rank(A),rank([A,b])
```

```
......
>>det(A)
......
```

1. **Формули на Крамер**

   Нека $D1$, $D2$, $D3\ldots$ са матрици, в които съответно $1, 2, 3, \ldots$ стълб на матрицата A е заменен със стълба на свободните коефициенти $\mathbf{b}$. Ако детерминантата $det(A) \neq 0$, решението на система от вида $A\mathbf{x} = \mathbf{b}$ може да се намери по формулите на *Крамер*:

$$\begin{aligned} x_1 &= det(D1)/det(A) \\ x_2 &= det(D2)/det(A) \\ x_3 &= det(D3)/det(A) \ldots\ldots \end{aligned}$$

```
>>D1=A;D1(:,1)=b;
>>D2=A;D2(:,2)=b;
>>D3=A;D3(:,3)=b;
>>d=det(A);
```

```
>>x=[det(D1);det(D2);det(D3)]/d
......
>>A*x
......
```

2. **Решаване на системата чрез обръщане на матрицата A**

   За да се използва този подход, матрицата A трябва да е квадратна и неособена. Матричното уравнение $A\mathbf{x} = \mathbf{b}$ се умножава отляво с A^{-1}:

$$A^{-1}. \mid A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad A^{-1}A\mathbf{x} = A^{-1}\mathbf{b} \Leftrightarrow \quad \mathbf{x} = A^{-1}\mathbf{b}.$$

```
>>det(A)
......
```

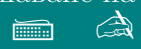
```
>>x=inv(A)*b
......
```

3. **Метод на Гаус**

   Методът на Гаус се състои от два етапа: **прав ход и обратен ход**.

   На първия етап, чрез елементарни операции над редовете, матрицата на системата се преобразува в горна триъгълна матрица, с нули под главния диагонал.

   На втория етап се започва от последното уравнение, от което се намира последното неизвестно; замества се в предпоследното уравнение и се намира предпоследното неизвестно; то от своя страна се замества в следващото (нагоре) уравнение и т.н. в обратен ред се намират останалите неизвестни.

   В Matlab, за решаване на определени системи с метода на Гаус, се използва операторът \. 

```
>>x=A\b
......
```

```
>>rats(x)
......
```

Забележка: Ако се търси вектор x от матричното уравнение $\mathbf{x}A = \mathbf{B}$, то се умножава отдясно с A^{-1} и решението с Matlab е `x=B/A`.

4. **Факторизация на Крут**

Факторизацията на Крут е метод, който използва т.нар. LU разлагане на матрица.

Разглежда се система нехомогенни уравнения в матричен вид:

$$A\mathbf{x} = \mathbf{b}, \tag{2.4}$$

където A е тридиагонална матрица

$$A = \begin{bmatrix} \alpha_1 & \gamma_1 & 0 & \cdots & \cdots & 0 \\ \beta_2 & \alpha_2 & \gamma_2 & \ddots & & 0 \\ 0 & \beta_3 & \alpha_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \alpha_{n-1} & \gamma_{n-1} \\ 0 & \cdots & 0 & \cdots & \beta_n & \alpha_n \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Системата (2.4) се преобразува във вида

$$LU\mathbf{x} = \mathbf{b},$$

където L е долна триъгълна матрица, а U - горна триъгълна матрица. Полага се $\mathbf{y} = U\mathbf{x}$. Първо се решава системата

$$L\mathbf{y} = \mathbf{b},$$

откъдето се намира вектора $\mathbf{y}$, след което се решава системата

$$U\mathbf{x} = \mathbf{y},$$

откъдето се получава решението $\mathbf{x}$ на системата (2.4).

☞ **Пример 2.3** *Да се реши чрез факторизация на Крут следната система.*

$$\left|\begin{array}{rrrrl} 2x_1 & -x_2 & & & = 1 \\ -x_1 & +2x_2 & -x_3 & & = 0 \\ & -x_2 & +2x_3 & -x_4 & = 0 \\ & & -x_3 & +2x_4 & = 1 \end{array}\right. \tag{2.5}$$

Решение:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & \frac{3}{2} & 0 & 0 \\ 0 & -1 & \frac{4}{5} & 0 \\ 0 & 0 & -1 & \frac{5}{4} \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 1 & -\frac{2}{3} & 0 \\ 0 & 0 & 1 & -\frac{3}{4} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

В Matlab има команда за LU разлагане на матрица A : `lu(A)`.

```
>>b=[1;0;0;1];
>>A=[2,-1,0,0;-1,2,-1,0
0,-1,2,-1;0,0,-1,2];
>>[L,U]=lu(A)
```

```
>>y=L\b
.........
>>x=U\y
.........
```

Итерационни методи за решаване на нехомогенни системи

1. **Метод на Якоби**

   Методът на Якоби, известен още като **метод на простата итерация**, е **итерационен метод**, при който се намира *приближено решение* на системата

   $$A\mathbf{x} = \mathbf{b}. \tag{2.6}$$

   Етап 1. Подготвителен етап:

   Системата (2.6) се преобразува във вид, удобен за итерации

   $$\mathbf{x} = T\mathbf{x} + C; \tag{2.7}$$

   Етап 2. Начална стъпка:

   Задава се точност $\varepsilon > 0$ и начално (нулево) приближение $\mathbf{x}^{(0)}$.

   Етап 3. k-та итерация, $(k = 1, 2, \ldots)$:

   Замества се $(k-1)$-тото приближение в десните страни на преобразуваната система (2.7) и се пресмята *ново приближение* $\mathbf{x}^{(k)}$ за неизвестния вектор по *итерационната формула*

   $$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + C. \tag{2.8}$$

   Сравняват се (k)-тото и $(k-1)$-тото приближения, като се пресмята

   $$\max_i \mid x_i^{(k)} - x_i^{(k-1)} \mid$$

   и се сравнява полученото число с ε.

   Проверява се дали е изпълнено *условието за край на итерационния процес*

   $$\max_i \mid x_i^{(k)} - x_i^{(k-1)} \mid < \varepsilon \quad , \; k = 1, 2, \ldots \tag{2.9}$$

   - ➥ Ако условието за край на итерационния процес (2.9) е изпълнено, за решение се приема последното получено приближение, т.е. $\mathbf{x}^{(k)}$;
   - ➥ Ако условието за край на итерационния процес (2.9) не е изпълнено, се преминава отново на Етап 3, при което k се увеличава с единица, последното получено приближение се замества в десните страни на системата (2.7) и се намира следващото приближение за неизвестните, след което отново се сравняват последните две приближения и т.н.

☞ **Пример 2.4** *Да се реши чрез* **метод на Якоби** *следната система.*

$$\left|\begin{array}{llll} 5.6x_1 & +2.7x_2 & -1.7x_3 & = 1.9 \\ 3.4x_1 & -3.6x_2 & -6.7x_3 & = -2.4 \\ 0.8x_1 & -1.3x_2 & +3.7x_3 & = 1.2 \end{array}\right.$$

Подготвителен етап: Системата се преобразува във вида (2.7), удобен за итерации:

$$\left|\begin{array}{llll} x_1 = & 0x_1 & -\frac{2.7}{5.6}x_2 & +\frac{1.7}{5.6}x_3 \quad +\frac{1.9}{5.6} \\ x_2 = & \frac{3.4}{3.6}x_1 & +0x_2 & -\frac{6.7}{3.6}x_3 \quad +\frac{2.4}{3.6} \\ x_3 = & -\frac{0.8}{3.7}x_1 & +\frac{1.3}{3.7}x_2 & +0x_3 \quad +\frac{1.2}{3.7} \end{array}\right.$$

Начална стъпка: Задава се точност $\varepsilon > 0$, напр. $\varepsilon = 0.1 \times 10^{-3}$ и начално (нулево) приближение $\mathbf{x}^{(0)}$, например $\mathbf{x}^{(0)} = (0,0,0)^T$.

```
>>A=[5.6,2.7,-1.7;3.4,-3.6,-6.7;.8,-1.3,3.7];
>>b=[1.9;-2.4;1.2];
>>T=[0,-2.7/5.6,1.7/5.6;3.4/3.6,0,-6.7/3.6;-.8/3.7,1.3/3.7,0];
>>C=[1.9/5.6;2.4/3.6;1.2/3.7];
>>x0=[0;0;0];eps=0.1e-03;
```

Първа итерация: Замества се (0)-то приближение в десните страни на преобразуваната система (2.7) и се пресмята $\mathbf{x}^{(1)}$.

Сравняват се (1)-тото и (0)-тото приближения като се пресмята максималната разликата между компонентите им по модул

$$\max_i \mid x_i^{(1)} - x_i^{(0)} \mid$$

и се сравнява полученото число с ε.

```
>>x1=T*x0+C
......
```

```
>>max(abs(x1-x0))<eps
......
```

Условието за край на итерационния процес (2.9) не е изпълнено, затова се връщаме на Етап 3, при което $k = 2$.

Втора итерация: Първото приближение $\mathbf{x}^{(1)}$ се замества в десните страни на системата (2.7) и се пресмята второто приближение $\mathbf{x}^{(2)}$ за неизвестните, след което отново се сравняват последните две приближения, проверява се дали е изпълнено условието за край на итерационния процес (2.9) и т.н.

```
>>x2=T*x1+C
......
```

```
>>max(abs(x2-x1))<eps
......
```

По такъв начин, за векторът $\mathbf{x}$ на неизвестните се получава редица от приближения

$$\mathbf{x}^{(0)}\ ,\ \mathbf{x}^{(1)}\ ,\ \mathbf{x}^{(2)}\ \ldots, \tag{2.10}$$

която, **ако е сходяща**, клони към решението на системата.

➠**Дефиниция 3** *Ако редицата (2.10) е сходяща, итерационният процес на Якоби, построен за системата (2.6) се нарича* **сходящ**.

Възникват логични въпроси: Дали за всяка система линейни алгебрични уравнения е удачно да се приложи методът на Якоби? Какви условия е необходимо да бъдат изпълнени, за да сме сигурни, че итерационният процес, построен за дадена система ще бъде сходящ?

➠**Дефиниция 4 Спектрален радиус** *на матрицата M се нарича числото*

$$\rho(M) = max \mid \lambda \mid,$$

където λ е собствена стойност на M.

➢**Теорема 2.2** *За всяко начално приближение $\mathbf{x}^{(0)}$, редицата*

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + C$$

е сходяща към единственото решение на системата $\mathbf{x} = T\mathbf{x} + C$ тогава и само тогава, когато $\rho(T) < 1$, където $\rho(T)$ е спектрален радиус на матрицата T.

Нека да приложим Теорема 2.2 към Задача 2.4.

```
>>rank(A),rank([A b])
......
```

```
>>max(abs(eig(T)))
......
```

Системата има единствено решение, но спектралният радиус $\rho(T) > 1$, следователно итерационният процес на Якоби, построен за тази система не е сходящ.

Извод: Преди да се прилага итерационен метод към дадена система, трябва да се проверят условията на Теореми 2.1 и 2.2.

☞ **Пример 2.5** *Да се реши чрез метод на Якоби системата:*

$$\left|\begin{array}{rrrrl} 10x_1 & -x_2 & +2x_3 & & = 6 \\ -x_1 & +11x_2 & -x_3 & +3x_4 & = 25 \\ 2x_1 & -x_2 & +10x_3 & -x_4 & = -11 \\ & 3x_2 & -x_3 & +8x_4 & = 15 \end{array}\right. \tag{2.11}$$

Решение:

Подготвителен етап: Системата (2.11) се преобразува във вида (2.7), удобен за итерации:

$$\left|\begin{array}{rrrrl} x_1 = & & +\frac{1}{10}x_2 & -\frac{1}{5}x_3 & +\frac{3}{5} \\ x_2 = & \frac{1}{11}x_1 & & +\frac{1}{11}x_3 & -\frac{3}{11}x_4 & +\frac{25}{11} \\ x_3 = & -\frac{1}{5}x_1 & +\frac{1}{10}x_2 & & +\frac{1}{10}x_4 & -\frac{11}{10} \\ x_4 = & & -\frac{3}{8}x_2 & +\frac{1}{8}x_3 & & +\frac{15}{8} \end{array}\right. \tag{2.12}$$

Въвеждат се матриците и се проверяват условията на Теореми 2.1 и 2.2: ⌨ ✍

```
>>A=[10,-1, 2,0;-1,11,-1,3;2,-1,10,-1;0,3,-1,8];
>>b=[6;25;-11;15];
>>T=[0,1/10,-1/5,0;1/11,0,1/11,-3/11;-1/5,1/10,0,1/10
0,-3/8,1/8,0];
>>C=[3/5;25/11;-11/10;15/8];
>>rank(A),rank([A,b])
......
>>max(abs(eig(T)))
......
```

Начална стъпка: Задава се точност $\varepsilon > 0$, например $\varepsilon = 1 \times 10^{-3}$ и начално (нулево) приближение $\mathbf{x}^{(0)}$, например $\mathbf{x}^{(0)} = (0,0,0,0)^T$.

Итерации: В средата на Matlab се получават последователни приближения, които се записват като стълбове в масива `iter`. На всяка стъпка се проверява условието за край на итерационния процес (2.9). Итерационният процес се прекратява тогава, когато условието (2.9) е изпълнено и за решение на системата се приема последното получено приближение (последният стълб на масива `iter`), което апроксимира точното решение с точност ε. Резултатите се попълват в Таблица 2.1. ⌨ ✍

```
>>x0=[0;0;0;0];eps=1e-03;
>>xs=x0;
>>xn=T*xs+C;
```

```
>>iter=[xs,xn]
......
>>while  max(abs(xn-xs))>=eps
```

k	0	1	2	3	4	5	6	7	8	9	10
$x_1^{(k)}$											
$x_2^{(k)}$											
$x_3^{(k)}$											
$x_4^{(k)}$											

Таблица 2.1: Резултати за Задача 2.5

```
xs=xn;                          | end
xn=T*xs+C;iter=[iter,xn]
```

2. **Метод на Гаус-Зайдел**

Методът на Гаус-Зайдел е модификация на метода на Якоби, при който се намира *приближено решение* на системата (2.6) с точност ε. като се използва *итерационната формула*

$$\mathbf{x}^{(k)} = T_1\mathbf{x}^{(k)} + T_2\mathbf{x}^{(k-1)} + C, \tag{2.13}$$

където матриците T_1 и T_2 са такива, че T_1 е долна триъгълна матрица с нули по и над главния диагонал, T_2 е горна триъгълна матрица с нули по и под главния диагонал и $T_1 + T_2 = T$.

На всяка стъпка (k)-тото приближение за неизвестния вектор на решението се намира като се използват *последните* получени приближения за компонентите му.

Условието за край на итерационния процес (2.9) е същото, както при *метода на Якоби*.

☞ **Пример 2.6** *Да се реши чрез метод на Гаус-Зайдел системата от Задача 2.5.*

Решение:

Подготвителен етап: Системата (2.11) се преобразува във вида (2.12), удобен за итерации.

Начална стъпка: Задава се точност $\varepsilon > 0$, например $\varepsilon = 1 \times 10^{-3}$ и начално (нулево) приближение $\mathbf{x}^{(0)}$, например $\mathbf{x}^{(0)} = (0, 0, 0, 0)^T$.

Първа итерация: Началното приближение се замества в десните страни на първото уравнение на системата (2.12) и се пресмята $x_1^{(1)}$. Това приближение и нулевите приближения за неизвестните $x_2^{(0)}$, $x_3^{(0)}$, $x_4^{(0)}$ се заместват в десните страни на второто уравнение на (2.12) и се пресмята $x_2^{(1)}$.

За да се пресметне $x_3^{(1)}$, в дясната страна на третото уравнение на системата (2.12) се заместват последните получени приближения $x_1^{(1)}$ и $x_2^{(1)}$ и началното приближение $x_4^{(0)}$. Аналогично, в десните страни на четвъртото уравнение се заместват $x_1^{(1)}$, $x_2^{(1)}$, $x_3^{(1)}$ и се пресмята $x_4^{(1)}$:

$$\left|\begin{array}{lllll} x_1^{(1)} = & & \frac{1}{10}x_2^{(0)} & -\frac{1}{5}x_3^{(0)} & +\frac{3}{5} \\ x_2^{(1)} = & \frac{1}{11}x_1^{(1)} & & +\frac{1}{11}x_3^{(0)} & -\frac{3}{11}x_4^{(0)} \quad +\frac{25}{11} \\ x_3^{(1)} = & -\frac{1}{5}x_1^{(1)} & +\frac{1}{10}x_2^{(1)} & & +\frac{1}{10}x_4^{(0)} \quad -\frac{11}{10} \\ x_4^{(1)} = & & -\frac{3}{8}x_2^{(1)} & +\frac{1}{8}x_3^{(1)} & +\frac{15}{8} \end{array}\right.$$

Така се получава първо приближение за неизвестния вектор $\mathbf{x}^{(1)}$.

Проверява се дали е изпълнено *условието за край на итерационния процес* (в случая не е), след което аналогично се пресмята следващото приближение $\mathbf{x}^{(2)}$.

Нека сега да намерим решението на Задача 2.6 по *метода на Гаус-Зайдел* с Matlab.

При многократно използване на набор от команди е удобно да се създаде файл, наречен M-файл, който след това се използва в средата на Matlab по същия начин, както и вградените команди.

За решаването на Задача 2.6 се създава файл `zajdel.m`, в който се дефинира функция с входни параметри матриците t и c, точността tol и началното приближение $\mathbf{x}^0$. При това името на функцията трябва да е същото като името на файла.

`zajdel.m`

```
function iterz=zajdel(t,c,x0,tol)
[m,n]=size(t);
t1=zeros(m,n);t2=zeros(m,n);
for i=1:m
    for j=1:n
       if i>j t1(i,j)=t(i,j);
         else t2(i,j)=t(i,j);
       end
    end
end
xs=x0;
[p,q]=size(xs);xn=zeros(p,q);
for i=1:m
   xn(i,1)=t1(i,:)*xn+t2(i,:)*xs+c(i,1);
end
iterz=[xs,xn]
```

```
17  while max(abs(xn-xs))>=tol
18        xs=xn;
19          for i=1:m
20            xn(i,1)=t1(i,:)*xn+t2(i,:)*xs+c(i,1);
21          end
22  iterz=[iterz,xn]
23  pause
24  end
```

В средата на Matlab се въвеждат матриците T и C както в Задача 2.5 и се проверяват условията на Теорема 2.2.

Задават се начално приближение и точност ε.

```
>>T=[0,1/10,-1/5,0;1/11,0,1/11,-3/11;-1/5,1/10,0,1/10;0,-3/8,1/8,0];
>>C=[3/5;25/11;-11/10;15/8];
>>rank([eye(4)-T])
.................
>>rank([eye(4)-T, C])
.................
>>max(abs(eig(T)))
.................
>>xx0=[0;0;0;0];
>>eps=1e-03;
>>y=zajdel(T,C,xx0,eps)
```

Резултатите се нанасят в Таблица 2.2.

k	0	1	2	3	4	5
$x_1^{(k)}$						
$x_2^{(k)}$						
$x_3^{(k)}$						
$x_4^{(k)}$						

Таблица 2.2: Резултати за Задача 2.6

Извод: Методът на Гаус-Зайдел е по-бързо сходящ от *метода на Якоби*, тъй като решението се достига с по-малък брой итерации.

Забележка: Решаването на задачата по *метода на Якоби* също може да се извърши като се създаде M-файл.

2.1.3 Системи нехомогенни уравнения при $r < n$

В този случай, от Теорема 2.1 следва, че системата (2.6) е неопределена, т.е. има безброй много решения. В Matlab чрез оператора \ се намира (с Хаусхолдерова ортогонализация) едно частно решение на системата, като на свободните неизвестни се дават стойности нули.

☞ **Пример 2.7** *Да се реши с-мата* ($n = 3$)*:*

$$\left|\begin{array}{rrrl} 2x_1 & +3x_2 & +4x_3 & = 4 \\ x_1 & +x_2 & +x_3 & = 5 \end{array}\right.$$

Решение:

```
>>A=[2,3,4;1,1,1];
>>b=[4;5];
>>rank(A),rank([A,b])
```

```
......
>>x=A\b
......
```

Всички решения се получават като на свободните неизвестни се дават произволни стойности и системата се решава спрямо базисните неизвестни. Базисни неизвестни са тези, чиито стълбове съответстват на базисен минор (различен от нула минор от ред, равен на ранга r).

```
>>det(A(1:2,1:2))
......
```

```
>>det(A(1:2,[1,3]))
......
```

Тъй като

$$\left|\begin{array}{cc} 2 & 3 \\ 1 & 1 \end{array}\right| \neq 0,$$

за базисни неизвестни могат да се приемат x_1 и x_2. Решава се системата

$$\left|\begin{array}{rrl} 2x_1 & +3x_2 & = 4 - 4x_3 \\ x_1 & +x_2 & = 5 - x_3 \end{array}\right.$$

и се получава общото решение на системата, в зависимост от параметъра $x_3 = const$:

$$(11 + x_3\ ,\ -6 - 2x_3\ ,\ x_3)^T\ .$$

Ако за базисни неизвестни се приемат x_1 и x_3 и се положи $x_2 = 0$, се получава точно частното решение $(8, 0, -3)^T$.

2.2 Системи линейни хомогенни уравнения

Разглежда се хомогенна система

$$A\mathbf{x} = \mathbf{0}, \tag{2.14}$$

с m линейни уравнения и n неизвестни, в която всички свободни коефициенти са нули. Очевидно, за всяка хомогенна система винаги е вярно условието

$$rank(A) = rank(A \mid \mathbf{b})$$

и от Теорема 2.1 следва, че системата (2.14) винаги е съвместима.

Всяка хомогенна система винаги има поне едно решение $\mathbf{x} = \mathbf{0}$, наречено **тривиално**, с нулеви стойности на неизвестните. Освен нулевото решение, системата (2.14) може да има и други, нетривиални решения, които именно се търсят.

В сила са следните твърдения:

- Системата (2.14), в която $m = n$, има единствено решение (нулево) тогава и само тогава, когато $det(A) \neq 0$;
- Системата (2.14), в която $m = n$, има безброй много решения (нулево и ненулеви) тогава и само тогава, когато $det(A) = 0$;
- Ако $m < n$, то системата (2.14) винаги има ненулево решение.

Търсенето на нетривиалните решения на системата (2.14) се състои от няколко **етапа**:

1. Намира се $rank(A) = r < n$;
2. Определя се базисен минор на матрицата A (различен от нула минор от ред, равен на r);
3. Определят се базисни неизвестни (стълбовете им съответстват на стълбовете на базисния минор);
4. Определят се свободни неизвестни (останалите $n - r$ неизвестни);
5. На свободните неизвестни се задават стойности - координатите на единична система вектори (напр. при $n - r = 3$: $(1, 0, 0)^T$, $(0, 1, 0)^T$, $(0, 0, 1)^T$) и се намира фундаментална система линейно независими вектори $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{n-r}$;
6. Всеки вектор-решение на системата (2.14) е линейна комбинация на векторите от фундаменталната система, т.е.

   $$x = c_1\mathbf{f}_1 + c_2\mathbf{f}_2 + \ldots + c_{n-r}\mathbf{f}_{n-r},$$

   където $c_1, c_2, \ldots, c_{n-r}$ са произволни константи.

☞ **Пример 2.8** *Да се реши хомогенната система:*

$$\left|\begin{array}{rrrrl} x_1 & -x_2 & +x_3 & -x_4 & =0 \\ x_1 & -x_2 & +2x_3 & +3x_4 & =0 \\ x_1 & -x_2 & -x_3 & -9x_4 & =0 \end{array}\right.$$

Решение. Въвежда се матрицата A, определя се ранга ѝ и се търси базисен минор:

```
>>A=[1,-1,1,-1;1,-1,2,3;1,-1,-1,-9]
>>rank(A)
......
>>det(A(1:2,1:2))
```

```
......
>>det(A(1:2,2:3))
......
```

Тъй като $rank(A) = 2$ и минорът

$$\left|\begin{array}{rr} -1 & 1 \\ -1 & 2 \end{array}\right| = -1 \neq 0$$

е базисен, за базисни неизвестни се избират x_2 и x_3, а за свободни - x_1 и x_4. Фундаменталната система вектори се състои от $n - r = 4 - 2 = 2$ вектора. Дадената система се преобразува във вида

$$\left|\begin{array}{rrcrl} -x_2 & +x_3 & = & -x_1 & +x_4 \\ -x_2 & +2x_3 & = & -x_1 & -3x_4 \end{array}\right. \tag{2.15}$$

На свободните неизвестни се задават стойности координатите на единична система вектори $(1,0)^T$ и $(0,1)^T$:

1. Полага се $x_1 = 1$, $x_4 = 0$. Системата (2.15) се решава (с метод на Гаус) спрямо базисните неизвестни x_2 и x_3:

```
>>A1=A(1:2,2:3);
>>B1=-A(1:2,1);
```

```
>>A1\B1
......
```

и се намират $x_2 = 1$ и $x_3 = 0$; следователно $\mathbf{f}_1 = (1,1,0,0)^T$.

2. Полага се $x_1 = 0$, $x_4 = 1$, решава се системата (2.15):

```
>>B2=-A(1:2,4);
>>A1\B2
......
```

и се намират $x_2 = -5$, $x_3 = -4$; следователно $\mathbf{f}_2 = (0,-5,-4,1)^T$.

Общото решение на системата е линейна комбинация на фундаменталната система вектори $\mathbf{f}_1$, $\mathbf{f}_2$:

$$\begin{aligned}\mathbf{x} &= c_1\mathbf{f_1} + c_2\mathbf{f_2} = c_1(1,1,0,0)^T + c_2(0,-5,-4,1)^T = \\ &= (c_1, c_1 - 5c_2, -4c_2, c_2)^T,\end{aligned}$$

където c_1, c_2 са произволни константи.

2.3 Преопределени системи. Метод на най-малките квадрати

Когато броят на уравненията m е по-голям от броя на неизвестните n, системата е **преопределена**. Например, нека променливите x и y са свързани помежду си линейно

$$y = a + bx \tag{2.16}$$

и се прави серия експерименти за определяне на коефициентите a и b. На i-тия експеримент при въведено x_i се измерва изходната стойност y_i. Резултатите от експеримента обикновено се получават с **грешка**, така че ако се начертаят точките (x_i, y_i), те не попадат точно върху правата линия, определена с (2.16), а някъде около нея. Нека ε_i е *грешката* на y_i, съответстваща на стойността, получена от (2.16):

$$y_i = a + bx_i + \varepsilon_i \qquad i = 1, 2, \ldots, m. \tag{2.17}$$

На практика се измерва $y_i - \varepsilon_i$, а не y_i и ако $m > 2$, системата уравнения

$$y_i - \varepsilon_i = a + bx_i \qquad i = 1, 2, \ldots, m. \tag{2.18}$$

обикновено е несъвместима.

Вместо точно решение се търси апроксимация, която да минимизира ефекта на грешките. Обикновено се минимизира сумата на квадратите на грешките

$$\mathcal{E} = \sum_{i=1}^{m} \varepsilon_i^2 = \sum_{i=1}^{m} (y_i - a - bx_i)^2 . \tag{2.19}$$

Търси се кои стойности на a и b минимизират $\mathcal{E}$.

Диференцира се $\mathcal{E}$ спрямо a и b, приравняват се на нула получените частни производни, в резултат на което се получава системата:

$$\left|\begin{array}{rcl} m.a + \sum\limits_{i=1}^{m} x_i.b & = & \sum\limits_{i=1}^{m} y_i \\ \sum\limits_{i=1}^{m} x_i.a + \sum\limits_{i=1}^{m} x_i^2.b & = & \sum\limits_{i=1}^{m} x_i y_i \end{array}\right. , \tag{2.20}$$

или записана в матричен вид $Y = AX + E$, където

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \quad A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}, \quad X = \begin{bmatrix} a \\ b \end{bmatrix}, \quad E = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix}.$$

Тогава системата (2.20) може да се запише в матричен вид така:

$$A^T AX = A^T Y. \tag{2.21}$$

Ако допуснем,че матрицата $A^T A$ е обратима, решението на (2.21) е

$$\begin{bmatrix} a \\ b \end{bmatrix} = \left(A^T A\right)^{-1} A^T Y. \tag{2.22}$$

Екстремумът, определен с (2.20) е минимум. За да се докаже това, е необходимо и достатъчно да се докаже, че

$$\begin{vmatrix} \frac{\partial^2 \mathcal{E}}{\partial a^2} & \frac{\partial^2 \mathcal{E}}{\partial a \partial b} \\ \frac{\partial^2 \mathcal{E}}{\partial b \partial a} & \frac{\partial^2 \mathcal{E}}{\partial b^2} \end{vmatrix} > 0. \tag{2.23}$$

Пресмятат се вторите частни производни от уравненията (2.20) и се заместват в детерминантата (2.23):

$$\begin{aligned} \frac{\partial^2 \mathcal{E}}{\partial a^2} \frac{\partial^2 \mathcal{E}}{\partial b^2} - \left(\frac{\partial^2 \mathcal{E}}{\partial a \partial b}\right)^2 &= m.\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2 = \\ &= (x_1 - x_2)^2 + (x_1 - x_3)^2 + (x_2 - x_3)^2 + \ldots \end{aligned}$$

Вижда се, че получената сума е > 0, което трябваше да се докаже.

☞ **Пример 2.9** *В резултат на експеримент са получени данни за две променливи x и y, които са свързани линейно*

$$y = a + bx, \tag{2.24}$$

дадени в следната таблица:

x	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5
y	2.72	4.35	6.78	8.58	10.63	11.88	13.82	15.93	16.93	18.75

5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
21.03	22.97	24.11	26.28	27.77	29.92	31.99	33.69	35.83	37.55	39.03

Да се намерят по метода на най-малките квадрати стойностите на параметрите a и b.

Решение. В Matlab се въвеждат x и y, намира се матрицата A и се търси вектор $x1$ с елементи a и b.

```
>>x=[0;0.5;1;1.5... ]; %въведете всички данни от таблицата
>>y=[2.72;4.35;6.78;8.58;10.63...];
>>A=[ones(x),x];
>>x1=inv(A'*A)*A'*y
......
```

Същият резултат се получава и с \ оператора:

```
>>x2=A\y
......
```

т.е. в Matlab операцията $A\backslash B$ винаги се изпълнява, като резултатът се получава по различни начини в различните случаи. В този случай решението се получава по метода на най-малките квадрати.

☞ **Пример 2.10** *Да се реши системата по метода на най-малките квадрати:*

$$\left|\begin{array}{rcl} 2x_1 & -x_2 & = 2 \\ x_1 & +x_2 & = 5 \\ 6x_1 & -x_2 & = -5 \end{array}\right.$$

Решение:

```
>>A=[2,-1;1,1;6,-1]
>>B=[2;5;-5];
```

```
>>xx=A\B
......
```

Нека вместо x_2 в първото уравнение запишем y_1, във второто - y_2 и в третото - y_3. Тогава можем да начертаем три прави линии с уравнения $y_1 = 2x_1 - 2$, $y_2 = 5 - x_1$, $y_3 = 6x_1 + 5$ и точката с координати - решението xx, получено по метода на най-малките квадрати.

```
>>x=-2:6;
>>y1=2*x-2;
>>y2=5-x;
>>y3=6*x+5;
>>plot(x,y1,x,y2,x,y3,xx(1),xx(2),'*')
```

2.4 Задачи за упражнение

✎ **Задача 2.1** *Да се решат системите линейни алгебрични уравнения с Графичен метод.*

а). $\left|\begin{array}{rrl} x_1 & +2x_2 & = 3 \\ x_1 & -x_2 & = 0 \end{array}\right.$ *б).* $\left|\begin{array}{rrl} x_1 & +2x_2 & = 0 \\ x_1 & -x_2 & = 0 \end{array}\right.$ *в).* $\left|\begin{array}{rrl} x_1 & +2x_2 & = 3 \\ 2x_1 & +4x_2 & = 6 \end{array}\right.$

г). $\left|\begin{array}{rrl} -x_1 & +2x_2 & = 3 \\ -2x_1 & -4x_2 & = 6 \end{array}\right.$ *д).* $\left|\begin{array}{rrl} x_1 & +2x_2 & = 0 \\ 2x_1 & +4x_2 & = 0 \end{array}\right.$ *е).* $\left|\begin{array}{rrl} 2x_1 & +x_2 & = -1 \\ x_1 & +x_2 & = 2 \\ x_1 & -3x_2 & = 5 \end{array}\right.$

✎ **Задача 2.2** *Да се решат системите линейни алгебрични уравнения с Метод на Гаус.*

а). $\left|\begin{array}{rrrrl} x_1 & +x_2 & & +3x_4 & = 4 \\ 2x_1 & +x_2 & -x_3 & +x_4 & = 1 \\ 3x_1 & -x_2 & -x_3 & +2x_4 & = -3 \\ -x_1 & +2x_2 & +3x_3 & -x_4 & = 4 \end{array}\right.$ *б).* $\left|\begin{array}{rrrl} 4x_1 & -x_2 & +x_3 & = 8 \\ 2x_1 & +5x_2 & +2x_3 & = 3 \\ x_1 & +2x_2 & +4x_3 & = 11 \end{array}\right.$

в). $\left|\begin{array}{rrrrl} x_1 & -x_2 & +2x_3 & -x_4 & = -8 \\ 2x_1 & -2x_2 & +3x_3 & -3x_4 & = -20 \\ x_1 & +x_2 & +x_3 & & = -2 \\ x_1 & -x_2 & +4x_3 & +3x_4 & = 4 \end{array}\right.$ *г).* $\left|\begin{array}{rrrl} x_1 & -x_2 & +3x_3 & = 2 \\ 3x_1 & -3x_2 & +x_3 & = -1 \\ x_1 & +x_2 & & = 3 \end{array}\right.$

д). $\left|\begin{array}{rrrl} 2x_1 & -1.5x_2 & +3x_3 & = 1 \\ -x_1 & & +2x_3 & = 3 \\ 4x_1 & -4.5x_2 & +5x_3 & = 1 \end{array}\right.$ *е).* $\left|\begin{array}{l} 2x_1 = 3 \\ x_1 + 1.5x_2 = 4.5 \\ -3x_2 + 0.5x_3 = -6.6 \\ 2x_1 - 2x_2 + x_3 + x_4 = 0.8 \end{array}\right.$

ж). $\left|\begin{array}{rrrrl} x_1 & -\frac{1}{2}x_2 & +x_3 & & = 4 \\ 2x_1 & -x_2 & -x_3 & +x_4 & = 5 \\ x_1 & +x_2 & & & = 2 \\ x_1 & -\frac{1}{2}x_2 & +x_3 & +x_4 & = 5 \end{array}\right.$ *з).* $\left|\begin{array}{rrrrl} x_1 & +x_2 & & +x_4 & = 2 \\ 2x_1 & +x_2 & -x_3 & +x_4 & = 1 \\ 4x_1 & -x_2 & -2x_3 & +2x_4 & = 0 \\ 3x_1 & -x_2 & -x_3 & +2x_4 & = -3 \end{array}\right.$

✎ **Задача 2.3** *Да се решат системите линейни алгебрични уравнения с Гаусова елиминация.*

а). $\left|\begin{array}{rrrl} \frac{1}{4}x_1 & +\frac{1}{5}x_2 & +\frac{1}{6}x_3 & = 9 \\ \frac{1}{3}x_1 & +\frac{1}{4}x_2 & +\frac{1}{5}x_3 & = 8 \\ \frac{1}{2}x_1 & +x_2 & +2x_3 & = 8 \end{array}\right.$ *б).* $\left|\begin{array}{rrrl} 3.333x_1 & +15920x_2 & -10.333x_3 & = 15913 \\ 2.222x_1 & +16.71x_2 & +9.612x_3 & = 28.544 \\ 1.5611x_1 & +5.1791x_2 & +1.6852x_3 & = 8.4254 \end{array}\right.$

в). $\left|\begin{array}{rrrrl} x_1 & +\frac{1}{2}x_2 & +\frac{1}{3}x_3 & +\frac{1}{4}x_4 & = \frac{1}{6} \\ \frac{1}{2}x_1 & +\frac{1}{3}x_2 & +\frac{1}{4}x_3 & +\frac{1}{5}x_4 & = \frac{1}{7} \\ \frac{1}{3}x_1 & +\frac{1}{4}x_2 & +\frac{1}{5}x_3 & +\frac{1}{6}x_4 & = \frac{1}{8} \\ \frac{1}{4}x_1 & +\frac{1}{5}x_2 & +\frac{1}{6}x_3 & +\frac{1}{7}x_4 & = \frac{1}{9} \end{array}\right.$ *г).* $\left|\begin{array}{l} 2x_1 + x_2 - x_3 + x_4 - 3x_5 = 7 \\ x_1 + 2x_3 - x_4 + x_5 = 2 \\ -2x_2 - x_3 + x_4 - x_5 = -5 \\ 3x_1 + x_2 - 4x_3 + 5x_5 = 6 \\ x_1 - x_2 - x_3 - x_4 + x_5 = 3 \end{array}\right.$

✎ **Задача 2.4** *(а). Да се реши системата линейни алгебрични уравнения с Формули на Крамер; (б). Покажете, че няма решение; в). Покажете, че има безброй много решения.*

а).
$$\left|\begin{array}{rrrl} 2x_1 & +3x_2 & -x_3 & = 4 \\ x_1 & -2x_2 & +x_3 & = 6 \\ x_1 & -12x_2 & +5x_3 & = 10 \end{array}\right.$$

б).
$$\left|\begin{array}{rrrl} 2x_1 & +3x_2 & -x_3 & = 4 \\ x_1 & -2x_2 & +x_3 & = 6 \\ -x_1 & -12x_2 & +5x_3 & = 9 \end{array}\right.$$

в).
$$\left|\begin{array}{rrrl} 2x_1 & +3x_2 & -x_3 & = 4 \\ x_1 & -2x_2 & +x_3 & = 6 \\ -x_1 & -12x_2 & +5x_3 & = 10 \end{array}\right.$$

✎ **Задача 2.5** *Да се решат системите линейни алгебрични уравнения с Факторизация на Крут.*

а).
$$\left|\begin{array}{rrrl} x_1 & -x_2 & & = 0 \\ -2x_1 & +4x_2 & -2x_3 & = -1 \\ & -x_2 & +2x_3 & = 1.5 \end{array}\right.$$

б).
$$\left|\begin{array}{rrrl} 3x_1 & +x_2 & & = -1 \\ 2x_1 & +4x_2 & +x_3 & = 7 \\ & 2x_2 & +5x_3 & = 9 \end{array}\right.$$

в).
$$\left|\begin{array}{rrrl} 2x_1 & -x_2 & & = 3 \\ -x_1 & +2x_2 & -x_3 & = -3 \\ & -x_2 & +2x_3 & = 1 \end{array}\right.$$

г).
$$\left|\begin{array}{l} 0.5x_1 + 0.25x_2 = 0.35 \\ 0.35x_1 + 0.8x_2 + 0.4x_3 = 0.77 \\ 0.25x_2 + x_3 + 0.5x_4 = -0.5 \\ x_3 - 2x_4 = -2.25 \end{array}\right.$$

✎ **Задача 2.6** *Да се решат системите линейни алгебрични уравнения с Метод на Якоби и Метод на Гаус-Зайдел*

а).
$$\left|\begin{array}{rrrl} 3x_1 & -x_2 & +x_3 & = 1 \\ 3x_1 & +6x_2 & +2x_3 & = 0 \\ 3x_1 & +3x_2 & +7x_3 & = 4 \end{array}\right.$$

б).
$$\left|\begin{array}{rrrl} 10x_1 & -x_2 & & = 9 \\ -x_1 & +10x_2 & -2x_3 & = 7 \\ & -2x_2 & +10x_3 & = 6 \end{array}\right.$$

в).
$$\left|\begin{array}{l} 10x_1 + 5x_2 = 6 \\ 5x_1 + 10x_2 - 4x_3 = 25 \\ -4x_2 + 8x_3 - x_4 = -11 \\ -x_3 + 5x_4 = -11 \end{array}\right.$$

г).
$$\left|\begin{array}{l} 4x_1 + x_2 - x_3 + x_4 = -2 \\ x_1 + 4x_2 - x_3 - x_4 = -1 \\ -x_1 - x_2 + 5x_3 + x_4 = 0 \\ x_1 - x_2 + x_3 + 3x_4 = 1 \end{array}\right.$$

д).
$$\left|\begin{array}{l} 4x_1 + x_2 + x_3 + x_5 = 6 \\ -x_1 - 3x_2 + x_3 + x_4 = 6 \\ 2x_1 + x_2 + 5x_3 - x_4 - x_5 = 6 \\ -x_1 - x_2 - x_3 + 4x_4 = 6 \\ 2x_2 - x_3 + x_4 + 4x_5 = 6 \end{array}\right.$$

е).
$$\left|\begin{array}{l} 4x_1 - x_2 - x_4 = 0 \\ -x_1 + 4x_2 - x_3 - x_5 = 5 \\ -x_2 + 4x_3 - x_6 = 0 \\ -x_1 + 4x_4 - x_5 = 6 \\ -x_2 - x_4 + 4x_5 - x_6 = -2 \\ -x_3 - x_5 + 4x_6 = 6 \end{array}\right.$$

☆ ☆

ГЛАВА 3

РЕШАВАНЕ НА НЕЛИНЕЙНИ УРАВНЕНИЯ

В практиката често се налага решаване на нелинейни уравнения, което може да представлява самостоятелна задача, или да е част от по-сложен проблем. Както е известно, много уравнения и системи уравнения нямат аналитично решение. Това се отнася за по-голямата част от трансцендентните уравнения . Доказано е също така, че не могат да се построят формули, с които да се реши произволно алгебрично уравнение от степен по-висока от четири. Освен това, в някои случаи коефициентите на уравнението са известни само приблизително и следователно, самата задача за точното определяне на корените губи смисъл. В тези случаи се използват итерационни методи със зададена степен на точност, [11, 13, 24, 26].

Нека е дадено уравнението

$$f(x) = 0 \tag{3.1}$$

➠**Дефиниция 5** *Корен на уравнението (3.1) се нарича числото* ξ*, за което* $f(\xi) = 0$.

Да се реши уравнението с итерационен метод означава да се установи, има ли то корени, колко са те и да се намерят стойностите им с нужната точност ε, т.е. за всеки корен да се намери число x, такова че

$$| x - \xi | < \varepsilon.$$

Задачата за намиране на корените на нелинейно уравнение се състои от **два етапа**:

1. *Отделяне на корените* – намиране на приближени стойности на корените или интервали, които ги съдържат;

2. *Уточняване на първоначалните приближения на корените* - довеждането им до зададената степен на точност.

3.1 Отделяне на корените

Процесът на отделяне на корените започва с установяване на знаците на функцията $f(x)$ в граничните точки от областта ѝ на съществуване.

➠**Дефиниция 6** *Интервал на локализация (изолация) на корен ξ на уравнението $f(x) = 0$, се нарича интервал $[a, b]$, за който са изпълнени условията:*

1. *Функцията $f(x)$ е непрекъсната в интервала $[a, b]$;*

2. *Функцията $f(x)$ е строго монотонна в (a, b), т.е. $f'(x) > 0$ или $f'(x) < 0$ за $\forall x \in (a, b)$;*

3. *$f(a)f(b) < 0$, т.е. функцията $f(x)$ има различни знаци в краищата на интервала.*

☞ **Пример 3.1** *Отделете корените на уравнението $f(x) = x^3 - 6x + 2 = 0$ в интервала $[-8, 8]$.*

Да съставим приблизителната схема:

x	-8	-3	-1	0	1	3	8
f(x)	−	−	+	+	−	+	+

Таблица 3.1: Отделяне на корените за Пример 3.1

Следователно, уравнението има три реални корена, лежащи в интервалите $[-3, -1]$, $[0, 1]$ и $[1, 3]$.

Приблизителни стойности на корените (началните приближения) могат да бъдат известни и от физическия смисъл на решаваната задача, от решението на аналогична задача при други изходни данни, или могат да бъдат намерени по графичен начин.

Да изчертаем с Matlab графиката на функцията от Пример 3.1 в интервала [-3;3]. Това може да стане със следните команди.

```
>>x=-3:0.1:3;
>>f=x.^3-6.*x+2;
>>plot(x,f),grid
```

Въз основа на чертежа се определят: $\xi_1 \approx -2.6$, $\xi_2 \approx 0.3$, $\xi_3 \approx 2.3$.

3.2 Уточняване на корените

Итерационният процес се състои в последователно уточняване на началното приближение. На всяка стъпка (итерация) се построява ново приближение на търсения корен. В резултат на това се получава редица от приближени стойности на корена $x_1, x_2, \ldots, x_n \ldots$.

➠**Дефиниция 7** *Ако редицата от последователни приближения клони, при $n \to \infty$ към корена ξ на уравнението $f(x) = 0$, то итерационният процес се нарича* ***сходящ****.*

3.2.1 Метод на разполовяването

Нека $[a, b]$ е интервал на локализация на корен на уравнението. *Алгоритъмът* на метода на разполовяването се състои в построяването на последователност от вложени един в друг интервали, в краищата на които функцията приема стойности с различни знаци. Всеки следващ интервал се получава от делението на предходния на две равни части.

Да опишем една стъпка (итерация) от метода. Нека на k-тата стъпка е намерен интервал $[a^{(k)}, b^{(k)}]$ такъв, че $f(a^{(k)})f(b^{(k)}) < 0$. Средата на този интервал е $c^{(k)} = \frac{a^{(k)}+b^{(k)}}{2}$. Ако $f(c^{(k)}) = 0$, то $c^{(k)}$ е корена и задачата е решена. Ако не, то от двете половинки на интервала избираме тази, в краищата на която функцията приема стойности с различни знаци:

$$\begin{aligned} a^{(k+1)} &= a^{(k)}, \quad b^{(k+1)} = c^{(k)}, \quad \text{ако} \quad f(a^{(k)})f(c^{(k)}) < 0 \\ a^{(k+1)} &= c^{(k)}, \quad b^{(k+1)} = b^{(k)}, \quad \text{ако} \quad f(a^{(k)})f(c^{(k)}) > 0 \end{aligned}$$

Критерий за завършване на итерационния процес: ако дължината на поредния интервал стане по-малка от 2ε, итерациите се прекратяват и за стойност на корена със зададената точност ε се приема средата на този последен интервал.

По този начин броят на итерациите до достигане на зададена точност може да се изчисли предварително, още преди започване на процеса, с формулата:

$$k_{max} = \left[\log_2\left(\frac{b-a}{\varepsilon}\right)\right].$$

Методът на разполовяването е лесен за прилагане, полученият итерационен процес е винаги сходящ. Той може да се осъществи дори и само с помощта на калкулатор. На практика той се прилага само за грубо уточняване на корените, тъй като в сравнение с други числени методи за решаване на нелинейни уравнения е бавен, т.е. необходими са повече итерации до достигане на зададената точност ε.

☞ **Пример 3.2** *По метода на разполовяването уточнете корена на уравнението* $f(x) = x^4 + 2x^3 - x - 1 = 0$, *заключен в интервала* $[0; 1]$.

Решение: Последователно пресмятаме:

$$\begin{aligned} f(0) < 0, f(1) > 0 \quad &\rightarrow \quad \xi \in (0,1) \ ; \\ f\left(\frac{1}{2}\right) < 0 \quad &\rightarrow \quad \xi \in \left(\frac{1}{2}, 1\right) \ ; \\ f\left(\frac{3}{4}\right) < 0 \quad &\rightarrow \quad \xi \in \left(\frac{3}{4}, 1\right) \ . \end{aligned}$$

Може да се приеме, че

$$\xi \approx \frac{3/4 + 1}{2} = 0.875 \quad \text{с точност} \quad \varepsilon = \frac{1 - 3/4}{2} = 0.125 \ .$$

За да се намери корена с точност $\varepsilon = 10^{-4}$, ще са необходими $k_{max} = [\log_2 10^4] = 13$ итерации.

3.2.2 Метод на Нютон

Един от най-разпространените методи за намиране на корени на уравнение е *методът на Нютон* и неговите модификации.

Да разложим функцията $f(x)$ в ред на Тейлър в околност на началното приближение x_0 на корена ξ и да се ограничим само с първите два члена от разложението:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \ldots$$

Понеже ξ е корен на уравнението, то $f(\xi) = 0$. Следователно,

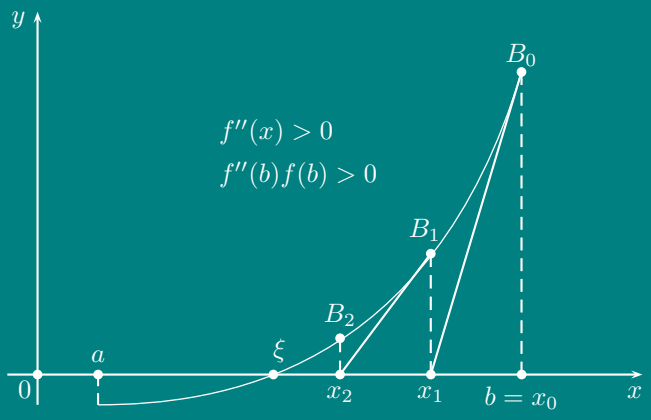
$$\xi \approx x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Ако ни е известна приближена стойност на корена на уравнението, получената формула ни дава възможност да го уточним. Тази процедура може да се повтори многократно. Поредното k -то приближение се намира по формулата:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \ .$$

Спираме, щом $\mid x_{k+1} - x_k \mid < \varepsilon$.

Ограничавайки се в разложението само до първите два члена, ние фактически заменяме функцията $f(x)$ с права линия, допирателна в точката x_0, затова метода на Нютон е известен още като метод на допирателните (Фигура 3.1).



Фигура 3.1: Метод на допирателните (Нютон)

3.2.3 Метод на секущите

Далеч не винаги е удобно да се намери аналитичен израз за производната на функцията. Това не е и напълно необходимо, тъй като на всяка стъпка ние получаваме само приближена стойност на корена, така че за неговото изчисляване може да се използва и приближена стойност за производната, въз основа на стойностите на функцията за последните две приближения:

$$f'(x_k) = \lim_{\delta \to 0} \frac{f(x_k + \delta) - f(x_k)}{\delta} \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \ ,$$

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} = \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})} \ .$$

В този вид методът се нарича **метод на секущите (secant method)**. Обикновено за първо приближение се взема $x_1 = x_0 + \varepsilon$, т.е. първата стъпка се извършва без използването на формулата, а всички следващи – след заместване в нея.

3.2.4 Други методи

Идеята на метода на секущите е развита в **метода на Мюлер**. При този метод за намиране на поредното приближение се използват три поредни точки. По този начин метода използва не линейна, а квадратична интерполация на функцията (с

интерполиране на функции ще се занимаем в отделно упражнение). Разчетните формули на метода са следните:

$$\begin{aligned} q &= \frac{x_k - x_{k-1}}{x_{k-1} - x_{k-2}} \\ A &= qf(x_k) - q(1+q)f(x_{k-1}) + q^2 f(x_{k-2}) \\ B &= (2q+1)f(x_k) - (1+q)^2 f(x_{k-1}) + q^2 f(x_{k-2}) \\ C &= (1+q)f(x_k) \\ x_{k+1} &= x_k - (x_k - x_{k-1})\left(\frac{2C}{B \pm \sqrt{B^2 - 4AC}}\right) \end{aligned}$$

Знакът пред корена се избира така, че абсолютната стойност на знаменателя да е максимална.

За намиране на корените на полином се използват специални методи. След като е намерен един от корените, степента на полинома може да бъде понижена, след което търсенето на корен се повтаря. На този принцип е основан например метода на **Лагер (Laguerre's method).**

Друг метод, който се използва за намиране на корени на полином е **метода на съпровождащата матрица (companion matrix)**. Може да се докаже, че матрицата

$$A = \begin{pmatrix} -\frac{a_{n-1}}{a_n} & -\frac{a_{n-2}}{a_n} & \dots & -\frac{a_1}{a_n} & -\frac{a_0}{a_n} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix},$$

наричана съпровождаща матрица за полинома $P(x) = \sum_{i=0}^{n} a_i x^i$, има собствени стойности, равни на корените на полинома. По този начин, задачата за намиране на корени на полином се свежда до задачата за намиране на собствените стойности на съпровождащата матрица.

3.3 Решаване на нелинейни уравнения с Matlab

В Matlab, за решаване на нелинейни уравнения се използват командите `fzero` и `roots`.

Командата `fzero` извиква изпълнението на програмен файл със същото име и разширение `.m`. Алгоритъмът е на Т. Декер, той използва комбинация от метод на разполовяването, метод на секущите и обратна квадратична интерполация. Командата `fzero` дефинира понятието корен на функция като точка, в която функцията пресича оста Ox. Точките, в които функцията се допира, но не

пресича абсцисната ос, не се възприемат като корени на уравнение. Например, $y = x^2$ е парабола, която се допира до оста Ox в точката $(0, 0)$. Тъй като обаче, тази функция никога не пресича абсцисната ос, `fzero` не дава това решение.

Командата `roots` е предназначена за решаване на алгебрични уравнения (корени на полиноми). Алгоритъмът се състои в намирането на собствените стойности на съпровождащата матрица.

☞ **Пример 3.3** *(Използване на команда* `roots`*.) Да решим уравнението от Пример 3.2.*

Решение: Задаването в Matlab на полином става чрез задаване на коефициентите на полинома във вид на вектор.

```
>>p=[1,2,0,-1,-1];
>>roots(p)
```

Matlab отпечатва на екрана резултата от действието на командата `roots`. Получават се всички корени – реални и комплексни. Точността по подразбиране е 10^{-4}.

☞ **Пример 3.4** *(Използване на команда* `fzero`*.) Да се реши с точност* 10^{-4} *уравнението:*

$$\sin x + e^{-x} + x^3 = 5 \ .$$

Решение: Най-напред уравнението се привежда във вида $f(x) = 0$. Функцията $f(x)$ от лявата част на уравнението трябва предварително да се зададе в отделен файл с разширение `.m`, напр. `ime.m`. Съдържанието на този файл трябва да е:

ime.m

```
1 function y=ime(x)
2 y=sin(x)+exp(-x)+x.^3-5;
```

След това, от командния ред на Matlab се изпълнява командата:

```
>>fplot('ime',[-5 5]),grid
```

Резултатът от тази команда е изчертаване на графиката на функцията $f(x)$ в интервала $[-5; 5]$. От чертежа се вижда, че уравнението има два реални корена, като $\xi_1 \approx -4.6$, а $\xi_2 \approx 1.6$. За да намерим корените с точност 10^{-4}, изпълняваме командите:

```
>>fzero('ime',1.6)
.................
```

```
>>fzero('ime',-4.6)
.................
```

Командата

```
fzero('ime',x0,tol)
```

дава възможност да се изчисли корена с различна точност. Предварително трябва да е въведена стойността на променливата `tol`, задаваща исканата точност, например:

```
>>tol=1e-6;format long
>>fzero('ime',1.6,tol)
>>format short
```

3.4 Задачи за упражнение

✎ **Задача 3.1** *Намерете ненулевия корен на трансцендентното уравнение*

1. $x^2 - 5\sin x = 0$;
2. $\sin x - 1/x = 0$;
3. $\lg x = \cos x$

с четири знака след десетичната точка. Кореният отделете графично.

✎ **Задача 3.2** *Намерете реалните корени на уравнението, с точност* 10^{-4}

1. $x^3 - 2x^2 + x = 3$;
2. $x^3 - 2x^2 + 3x = 5$;
3. $x^4 - 5x^3 + 2x^2 - 10x + 1 = 0$

✎ **Задача 3.3** *Определете броя на кореният на уравнението и за всеки корен намерете интервал, който го съдържа:* $x^4 - 1 - \cos x = 0$.

✎ **Задача 3.4** *По метода на разполовяването намерете кореният на уравнението* $x^3 + 2x - 6 = 0$ *с точност* $\varepsilon = 0.3$. *Колко итерации трябва да се извършат за да се достигне точност* $\varepsilon = 0.01$*?*

★ ★

ГЛАВА 4

РЕШАВАНЕ НА СИСТЕМИ НЕЛИНЕЙНИ УРАВНЕНИЯ

Дадена е произволна система от n нелинейни уравнения с n неизвестни

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ &\\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

или във векторна форма

$$f(x) = 0,$$

където

$$f = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad , \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

4.1 Метод на Нютон

Основната идея на метода на Нютон се състои в отделяне от уравненията на системата на линейните им части, които са главни при малки нараствания на аргументите. Това позволява изходната задача да се сведе до решаване на последователност от линейни системи, [11, 13, 24, 26].

Нека е известно приближение $x^{(k)}$ на корена ξ. Тогава поправката $\Delta x = \xi - x^{(k)}$ може да се намери, като се реши системата

$$f\left(x^{(k)} + \Delta x\right) = 0.$$

За определянето на Δx да развием векторната функция f в ред по Δx. Запазвайки само линейните членове, ще получим

$$0 = f\left(x^{(k)}\right) + J\Delta x + \ldots,$$

където J е матрицата на Якоби с елементи

$$J_{ij} = \left.\frac{\partial f_i}{\partial x_j}\right|_{x^{(k)}} .$$

Следователно, за изчисляването на поправката Δx се достига до системата линейни уравнения

$$J\Delta x = -f\left(x^{(k)}\right).$$

Ако $\det J \neq 0$, то

$$\Delta x = -J^{-1} f\left(x^{(k)}\right).$$

По този начин се получава схема за уточняване на решението на изходната система нелинейни уравнения, аналогична на метода на Нютон за едно уравнение:

$$x^{(k+1)} = x^{(k)} - J^{-1} f\left(x^{(k)}\right).$$

Ако $\det J \neq 0$, то в достатъчно малка околност на корена ξ, итерационният процес е сходящ, при това с квадратична скорост, т.е.

$$\| x^{(k+1)} - x^{(k)} \|^2 \geq \| x^{(k+1)} - \xi \| .$$

Ако началното приближение $x^{(0)}$ е удачно избрано, то метода на Нютон е много бързо сходящ. В качество на критерий за завършване на итерационния процес може да се използва условието

$$\| x^{(k+1)} - x^{(k)} \|^2 < \varepsilon$$

Методът на Нютон може да бъде направен по-устойчив относно избора на началното приближение, със следната модификация:

Последователните приближения се изчисляват по формулата:

$$x^{(k+1)} = x^{(k)} - \lambda J^{-1} f\left(x^{(k)}\right),$$

като λ се избира така, че $\| f\left(x^{(k+1)}\right) \|$ да е колкото се може по-близо до нула.

Методът може да бъде направен по-ефективен, с използването на една и съща матрица J в няколко последователни стъпки. (reduced Newton method).

При т.нар. квази - Нютонови методи, вместо да се пресмятат частните производни, с помощта на които се определят коефициентите на матрицата на Якоби, те се оценяват, като се използват стойностите на функцията f, изчислени при предищните итерации, т.е. $f\left(x^{(k)}\right)$, $f\left(x^{(k-1)}\right)$, $f\left(x^{(k-2)}\right)$, ...

4.2 Решаване на системи нелинейни уравнения с Matlab

В Matlab, системите нелинейни уравнения се решават с командата `fsolve`, която включва квази – Нютонови методи, заедно с различни техники за повишаване на надеждността на решението. Така например, ако уравненията в системата не могат едновременно да приемат стойност нула, поради неточности в коефициентите, или просто системата няма решение, алгоритъмът въпреки това връща стойностите за неизвестните, при които отклоненията от нула са минимални едновременно за всички уравнения. По точно, използва се идеята на метода на най-малките квадрати, като се минимизира сумата от квадратите на левите части на уравненията. Ще припомним, че с помощта на този подход Matlab дава и решение на преопределени системи от линейни уравнения, което става с командата \. Метода на най-малките квадрати е подробно разгледан в Раздел 6.

В някои случаи (например, при $\det J = 0$) е възможно командата `fsolve` да даде резултат, който не е решение на системата от уравнения. В този случай трябва да се опита отново с друго начално приближение.

В най-опростения си вид командата има следния синтаксис:

```
fsolve('ime',x0)
```

където:

- **ime** е името на файл с разширение `.m`, в който предварително са зададени уравненията на системата, във вид на вектор-стълб;
- **x0** е вектор от началните приближения за неизвестните.

☞ **Пример 4.1** *Да се реши системата от уравнения*

$$\begin{aligned} x^2 + y^2 &= 4 \\ e^x - y &= 1 \end{aligned}$$

при начално приближение $x = 0.5$, $y = 0.5$. 

Решение: Създава се файл `ime.m`. Съдържанието на файла е:

ime.m

```
1 function f=ime(X)
2 x=X(1);y=X(2);
3 f=[x.^2+y.^2-4;exp(x)-y-1];
```

От командния ред на Matlab се въвежда:

```
>>fsolve('ime',[0.5,0.5])
.......................
```

В по-общ вид командата `fsolve` изглежда така:

```
fsolve('ime',x0,options)
```

като `options` е вектор с 18 елемента, съдържащ контролни параметри. Подробна информация за параметрите може да се получи с команда

```
>>help fsolve
```

или

```
>>help foptions
```

Например, ако се зададе стойност 1 за първия елемент на `options`, на екрана се отпечатват междинни резултати и евентуални предупреждения:

```
>>options(1)=1;
>>fsolve('ime',[0.5,0.5],options)
```

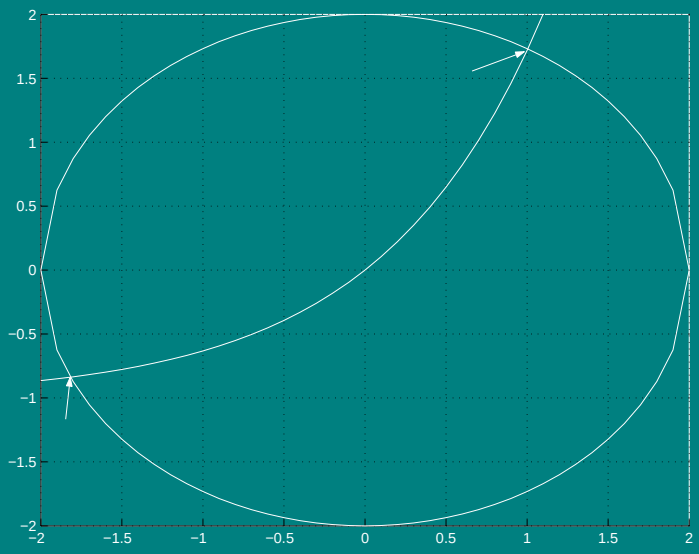
Колкото началното приближение е по-близко до точното решение, толкова е по-вероятно да получим бързо и коректно истинско решение на системата. Както и в случая на едно нелинейно уравнение, тук също можем да използваме графичните възможности на Матлаб при определянето на начални стойности за неизвестните.

☞ **Пример 4.2** *Като се използват графичните възможности на Matlab, да се провери има ли и друго решение на системата от Пример 4.1.*

Решение: За изчертаването (Фигура 4.1) на графиките на функциите в левите страни на уравненията има два начина.

I начин: Предварително се изразява от всяко уравнение едно от неизвестните чрез другото. Графиките се изчертават в равнината и се търсят пресечните точки на двете криви:

```
>>x=-2:0.1:2;
>>y1=sqrt(4-x.^2);
>>y2=-sqrt(4-x.^2);
>>y3=exp(x)-1;
>>axis([-2 2 -2 2])
>>plot(x,y1,x,y2,x,y3),grid
```



Фигура 4.1: Решение на Пример 4.2

II начин: С помощта на команда `contour`, за левите части на уравненията, като функции на две променливи, се изчертават на един и същи чертеж линиите на ниво за стойност нула и се определят пресечните им точки:

```
>>[x,y]=meshgrid(-2:0.2:2,-2:0.2:2);
>>z1=x.^2+y.^2-4;
>>z2=exp(x)-y-1;
>>contour(x,y,z1,[0,0]),hold on
>>contour(x,y,z2,[0,0]),grid
```

И по двата начина се вижда, че системата има още един корен, като за начални приближения могат да се вземат $x^{(0)} = -1.8$ и $y^{(0)} = -0.8$.

4.3 Задачи за упражнение

✎ **Задача 4.1** *Да се реши системата нелинейни уравнения*

$$\begin{aligned} tg4x - \cos 3y &= 0 \\ 2.3y^3 - x^2 - 4x &= 3 \end{aligned}$$

✎ **Задача 4.2** *Да се реши системата нелинейни уравнения*

$$\begin{aligned} \sin(x + 2.1) - 3y &= -0.4 \\ \cos(y + 1.8) + 1.2x &= 0 \end{aligned}$$

✎ **Задача 4.3** *Да се реши системата нелинейни уравнения:*

$$\begin{aligned} 3x_1 - \cos(x_2 x_3) &= \frac{1}{2} \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 &= -1.06 \\ e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} &= 0 \end{aligned}$$

За начално приближение да се избере $(x_1, x_2, x_3) = (0.1, 0.1, -0.1)$.

✎ **Задача 4.4** *Да се реши системата нелинейни уравнения:*

$$\begin{aligned} 4x_1 - x_2 + x_3 &= x_1 x_4 \\ -x_1 + 3x_2 - 2x_3 &= x_2 x_4 \\ x_1 - 2x_2 + 3x_3 + &= x_3 x_4 \\ x_1^2 + x_2^2 + x_3^2 &= 1 \end{aligned}$$

За начални приближения да се изберат $x^{(0)} = (1, 1, 1, 1)$, $x^{(0)} = (1, 0, 0, 0)$, $x^{(0)} = (-1, 0, 0, 0)$

✎ **Задача 4.5** *Да се реши системата нелинейни уравнения:*

$$\begin{aligned} 2x_1 - 3x_2 + x_3 &= 4 \\ 2x_1 + x_2 - x_3 &= -4 \\ x_1^2 + x_2^2 + x_3^2 - 4 &= 0 \end{aligned}$$

За начално приближение да се избере $(x_1, x_2, x_3) = (-0.5, -1.5, 1.5)$.

✎ **Задача 4.6** *Да се реши системата нелинейни уравнения:*

$$\begin{aligned} \sin(4\pi x_1 x_2) - 2x_2 - x_1 &= 0 \\ \left(\frac{4\pi - 1}{4\pi}\right)\left(e^{2x_1} - e\right) + 4ex_2^2 - 2ex_1 &= 0 \end{aligned}$$

ГЛАВА 5

АПРОКСИМИРАНЕ НА ФУНКЦИИ

Апроксимацията на функция се състои в приближената замяна на дадена функция $f(x)$ с функция $\varphi(x)$ така, че отклонението на $\varphi(x)$ от $f(x)$ в дадена област да е възможно най-малко. При това $\varphi(x)$ се нарича *апроксимираща функция*.

Според вида на множеството от точки, върху което се извършва апроксимацията, се различават **дискретна** и **непрекъсната** апроксимации:

- Когато приближението се извършва върху крайно или изброимо множество от точки, апроксимацията се нарича **точкова** или **дискретна**;
- Когато замяната се извършва върху непрекъснато множество от точки (интервал), апроксимацията се нарича **непрекъсната** или **интегрална**. Пример за такава апроксимация е развитието на функция в ред на Тейлър, ограничавайки се само до първите n члена, т.е. замяната на функцията с полином.

5.1 Интерполация на функция

Най-често използваната в практиката точкова апроксимация е **интерполацията**, [11, 13, 24, 26]. Необходимостта от прилагането ѝ е свързана основно със следните две причини:

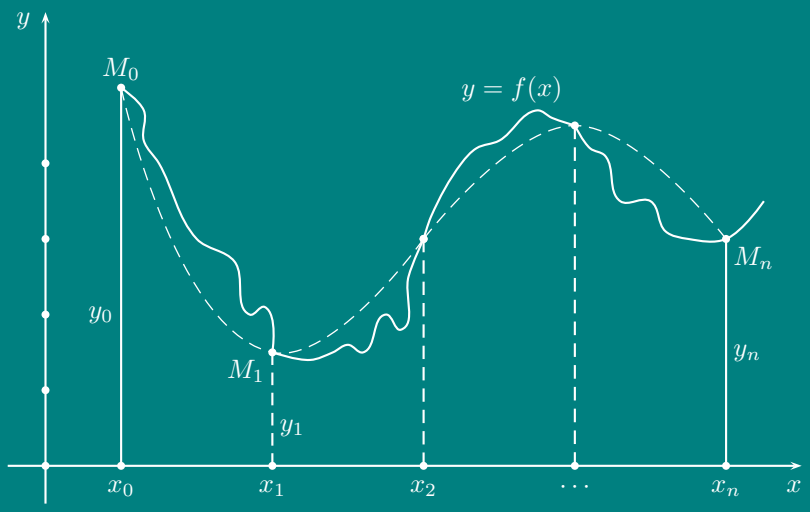
- Функцията $f(x)$ има сложен аналитичен израз, който създава допълнителни трудности при нейното използване (например, $f(x)$ е специална функция – Гама-функция, елиптична функция и др.);

➨ Аналитичният израз на функцията $f(x)$ не е известен, т.е. $f(x)$ е зададена таблично, например вследствие резултати от измервания. При това е необходимо да се разполага с аналитичния вид на експерименталната зависимост (например, за изчисляване на стойностите на $f(x)$ в произволни точки, пресмятане на интеграли и производни от $f(x)$ и т.н.).

Най-простата интерполационна задача се състои в следното: Върху интервала $[a, b]$ са дадени $n + 1$ точки $x_0, x_1, \ldots, x_n$, които се наричат **интерполационни възли**; известни са стойностите на функцията $f(x)$ в тези точки: $f(x_0) = y_0$, $f(x_1) = y_1 , \ldots , f(x_n) = y_n$; трябва да се построи функция $\varphi(x)$ (**интерполираща функция**), принадлежаща на известен клас от функции (напр. алгебричен или тригонометричен полином) и приемаща в интерполационните възли същите стойности като $f(x)$ (Фигура 5.1).

При такава обща постановка, задачата може да има безброй много решения или изобщо да няма решение.

Задачата става еднозначна, ако вместо произволна функция $\varphi(x)$ се търси полином $p(x)$ (**интерполационен полином**) от степен не по-висока от n.



Фигура 5.1: Интерполация на функция

В случая, когато полиномът е един за цялата интерполационна област, се казва, че интерполацията е **глобална**.

В случаите, когато между различните възли полиномите са различни, интерполацията е **частична** или **локална интерполация**.

След като е намерен интерполационният полином, могат да се изчисляват стойностите на функцията $f(x)$ в точки между възлите (т.е. да се извършва **интерполация в тесен смисъл**), а също така и да се пресмятат стойностите на функцията $f(x)$ в точки извън границите на дадения интервал (т.е. да се извършва **екстраполация**).

5.2 Интерполационен полином на Лагранж

Една от формите на запис на *интерполационния полином* е **полиномът на Лагранж**:

$$L_n(x) = \sum_{j=0}^{n} y_j \, l_j(x),$$

където:

$$l_j(x) = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} = \frac{(x - x_0)\,(x - x_1)\ \ldots\ (x - x_{j-1})(x - x_{j+1})\ \ldots\ (x - x_n)}{(x_j - x_0)(x_j - x_1)\ldots(x_j - x_{j-1})(x_j - x_{j+1})\ldots(x_j - x_n)}\,.$$

Лесно се вижда, че $l_j(x)$ са полиноми от степен n, които в интерполационните възли приемат стойности

$$l_j(x_i) = \begin{cases} 1\,, & i = j \\ 0\,, & i \neq j \end{cases}\,.$$

Поради това степента на полинома на Лагранж $L_n(x)$ също е n и при $x = x_i$ всички събираеми в сумата стават равни на нула, с изключение на събираемото с номер $j = i$, равно на y_i. Следователно, полиномът на Лагранж удовлетворява условието

$$L_n(x_i) = y_i,$$

т.е графиката му минава през всички точки (x_i, y_i), откъдето следва, че той е интерполационен полином.

☞ **Пример 5.1** *Да се построи полиномът на Лагранж, интерполиращ таблично зададената функция $F(x)$. Да се пресметнат $F(0.75)$ и $F(1.3)$.*

x	-1	0	1	2
F(x)	4	2	0	1

Таблица 5.1: Данни за Пример 5.1

Решение:

$$L_3(x) = 4\frac{x(x-1)(x-2)}{(-1)(-2)(-3)} + 2\frac{(x+1)(x-1)(x-2)}{1\cdot(-1)(-2)} + 1\frac{(x+1)\,x\,(x-1)}{3\cdot 2\cdot 1} = \frac{1}{2}x^3 - \frac{5}{2}x + 2.$$

Построяването на полинома в програмната среда Matlab става с изпълнението на командата

```
p=polyfit(x,y,k)
```

където:

- `x` е вектор, съдържащ интерполационните възли;
- `y` е вектор от съответните стойности на интерполираната функция;
- `k` е степента на полинома, която се определя по следната формула:

Степента на полинома на Лагранж	=	Броят на интерполационните възли минус единица

Командата `polyfit` създава вектор `p`, съдържащ коефициентите пред степените на x в намаляващ ред (т.е. първият елемент на p е коефициента пред най-високата степен на x, ... , последният - свободният коефициент в полинома на Лагранж).

Пресмятането на стойностите на полинома за различни стойности на аргумента x, става с командата

```
polyval(p,x0)
```

която дава стойността на интерполиращия полином в точка x_0.

За данните от Пример 5.1, трябва да се въведат командите:

```
>>x=[-1,0,1,2];
>>y=[4,2,0,1];
>>p=polyfit(x,y,3)
..................
```

```
>>polyval(p,0.75)
.........
>>polyval(p,1.3)
.........
```

Един от начините да се провери качеството на интерполацията, е графичният.

В някои случаи е възможно да бъде получена **оценка на грешката** от замяната на $f(x)$ с полинома $L_n(x)$.

Да предположим, че в интервала $[a, b]$ функцията $f(x)$ има непрекъсната производна от ред $(n+1)$. Може да се докаже, че за абсолютната стойност на грешката $\varepsilon(x) = \mid f(x) - L_n(x) \mid$ е в сила оценката:

$$\varepsilon(x) \leq \frac{\mid (x-x_0)(x-x_1)\dots(x-x_n) \mid}{(n+1)!} M_{n+1}, \tag{5.1}$$

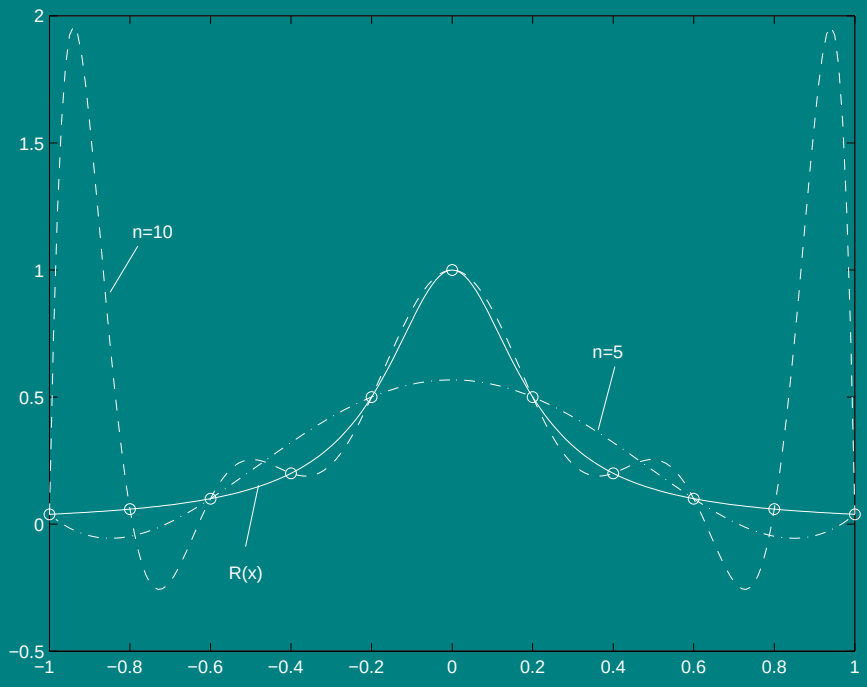
където

$$M_{n+1} = \max_{x\in[a,b]} \mid f^{(n+1)}(x) \mid .$$

Да видим какво ще се получи, ако интерполираме функцията $f(x)$ във все повече и повече точки от фиксирания интервал.

Изразът, оценяващ грешката се състои от три части. Факториелът и произведението от разликите с увеличаването на n ще намаляват грешката, но редът на производната при това ще расте. За много функции, величината M_{n+1} нараства по-бързо в сравнение с $(n+1)!$. В резултат на това, за интерполиращия полином от висока степен са възможни големи отклонения от интерполираната функция в точките, различни от интерполационните възли. Затова на практика най-често се използват полиноми от степен не по-висока от 5-6.

За пример може да послужи т.нар. функция на Рунге: $R(x) = 1/(1+25x^2)$, графиката на която е представена на Фигура 5.2. С увеличаването на степента на интерполиращия полином, при равномерно разпределение на възлите върху интервала $[-1, 1]$, се наблюдава влошаване на качеството на приближение в краищата на интервала. Това може да се обясни с факта, че производните на $R(x)$, които фигурират в оценката за грешката (5.1), бързо нарастват с увеличаването на n.



Фигура 5.2: Функция на Рунге

Точността на приближението зависи не само от броя на интерполационните възли (т.е. от степента на интерполиращия полином), но и от тяхното разположение върху интервала $[a, b]$. В най-простия случай се избира равномерно разположение на точките $x_i, i = 0, \ldots, n$ върху интервала $[a, b]$ със стъпка $h = (b-a)/(n-1)$, т.е. $x_{i+1} = x_i + h$.

По-добри приближения обаче се получават, ако разположението на възлите се определя по **формулата на Чебишев**:

$$x_{i+1} = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{(2i+1)\pi}{2n+2}, \quad i = 0, 1, \ldots, n. \tag{5.2}$$

☞ **Пример 5.2** *Апроксимирайте функцията* $\ln(x)$ *върху интервала* $[0.25, 1.75]$

1. *с интерполационен полином* p *от степен 4. Използвайте* p*, за да пресметнете* $\ln(1.75)$*. Какви са абсолютната и относителната грешки? С Matlab постройте чертеж, показващ функциите* $\ln(x)$ *и* $p(x)$*. Постройте втори чертеж, показващ разликата между* $\ln(x)$ *и апроксимиращия полином. Каква е максималната грешка на тази апроксимация?*

2. *Повторете същата процедура за полином от трета степен и проверете какво е качеството на приближение.*

Решение: Най-напред се определят 5 интерполационни възела по формулата на Чебишев.

```
>>a=.25;
>>b=1.75;
>>n=4;
>>i=0:n;
>>xc=(a+b)/2+(b-a)/2.*cos((2.*i+1).*pi./(2*n+2));
```

Построяването на полинома става със следните команди:

```
>>y=log(xc);
>>p=polyfit(xc,y,4)
.................
```

Апроксимирането в точката $x = 1.75$ и пресмятането на абсолютната и относителна грешки се извършва с командите:

```
>>px=polyval(p,1.75)
.........
>>fx=log(1.75)
.........
```

```
>>abs_error=abs(px-fx)
.........
>>rel_error=abs_error/fx*100
.........
```

Изчертаването на двата чертежа става с командите:

```
>>x=.25:.1:1.75;
>>px=polyval(p,x);
>>fx=log(x);
>>plot(x,fx,x,px,'b')
>>plot(x,px-fx)
```

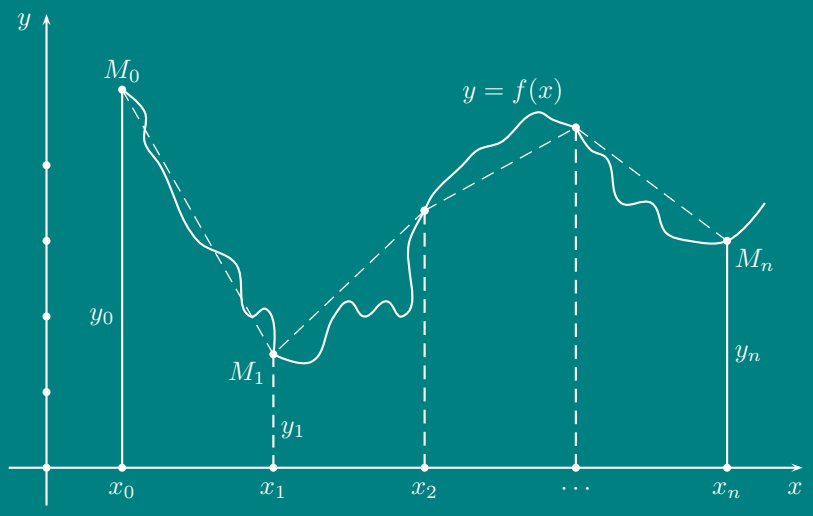
Първият чертеж показва, че интерполационния полином постига графична точност (неговата графика не се отличава от графиката на $\ln(x)$). Вторият, даващ графиката на грешките, показва, че максималното отклонение е в левия край на интервала, т.е. $x = .25$, и е около .02.

Следователно, за разглеждания интервал $[0.25, 1.75]$, вместо функцията $\ln(x)$ може да се използва получения интерполационен полином (напр. при изчисления, пресмятане на определен интеграл и др.).

5.3 Интерполация на функция «на части»

В практиката, най-често се използва «на части» *полиномиална интерполация*: зададеният интервал се разбива на части и във всеки подинтервал изходната функция се заменя с полином от невисока степен.

Най-простият и често използван вид локална интерполация е *линейната интерполация*. В този случай зададените точки $M(x_i, y_i)$, $i = 0, 1, \ldots, n$ се съединяват с отсечки и функцията $f(x)$ се приближава с начупена линия с върхове в дадените точки (Фигура 5.3).



Фигура 5.3: Локална линейна интерполация

При *квадратичната интерполация*, за интерполираща функция в интервала (x_{i-1}, x_{i+1}), се избира полином от втора степен, а при *кубичната интерполация*

– във всеки подинтервал (x_{i-2}, x_{i+1}) изходната функция се заменя с полином от трета степен.

5.4 Апроксимация със сплайни

Ако изходната функция е гладка (т.е. е достатъчен брой пъти диференцируема) и е необходимо апроксимиращата функция също да е такава, то «на части» полиномиалната интерполация е неприемлива. В този случай се използват **сплайни** – построени по специален начин гладки «на части» полиномиални функции.

Нека интервалът $[a, b]$ е разбит с помощта на междинни точки на n подинтервала (x_i, x_{i+1}).

➠**Дефиниция 8** ***Сплайн*** *от степен m се нарича функцията $S_m(x)$, притежаваща свойствата:*

1. *функцията $S_m(x)$ е непрекъсната в интервала $[a, b]$ заедно с производните си до някакъв ред;*
2. *във всеки подинтервал (x_i, x_{i+1}), функцията $S_m(x)$ съвпада с алгебричен полином $P_{m,i}(x)$ от степен m.*

Най-широко приложение са получили сплайните от степен 3, т.нар. **кубични сплайни**.

В Matlab, интерполирането с «на части» полиномиални криви се осъществява посредством командата `interp1`, чийто синтаксис е:

```
interp1(x,y,xi,method)
```

където:

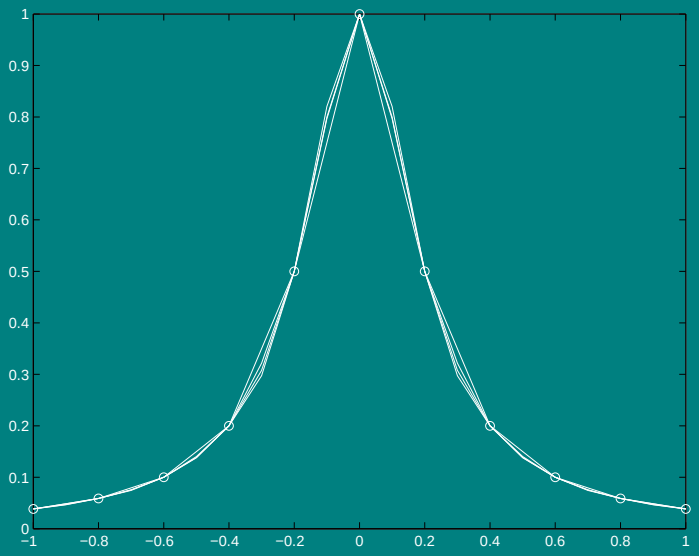
- `x` е вектор, съдържащ интерполационните възли;
- `y` е вектор от съответните стойности на интерполираната функция;
- `xi` е стойност или вектор от стойности, в които желаем да се извърши интерполацията;
- `method` е една от следните ключови думи:
  - `'linear'` – извършва линейна интерполация;
  - `'spline'` – извършва кубична сплайн интерполация;

- `'cubic'` – извършва кубична интерполация.

Забележка: И трите интерполационни метода изискват векторът x, съдържащ интерполационните възли да е монотонен.

☞ **Пример 5.3** *Да се интерполира функцията на Рунге в интервала* $[-1, 1]$ *(виж Фигура 5.2), като се използва «на части» полиномиална интерполация.*

Решение: На Фигура 5.2 е изобразен интерполационният полином от степен 10,



Фигура 5.4: Функция на Рунге - «на части» полиномиална интерполация

получен при глобална интерполация върху целия интервал $[-1, 1]$ по 11 равноотдалечени интерполационни възела. Като се използват същите възли и командата `interp1` в Matlab, се виждат (виж Фигура 5.4) предимствата на локалната, «на части» полиномиална интерполация.

```
>>x=-1:.2:1;
>>y=1./(1+25*x.^2);
>>plot(x,y,'o'),hold on
>>xi=-1:0.1:1;
>>y1=interp1(x,y,xi,'linear');
>>y2=interp1(x,y,xi,'spline');
>>y3=interp1(x,y,xi,'cubic');
>>f=1./(1+25*xi.^2);
>>plot(xi,y1,'b',xi,y2,'g',xi,y3,'y',xi,f)
```

5.5 Задачи за упражнение

✎ **Задача 5.1** *Функцията f е зададена с Таблица 5.2:*

x_k	3.92	3.94	3.96	3.98	4.0	4.02
$f(x_k)$	.4004451	1.4186012	2.4573263	3.5170344	4.5881505	5.701106

Таблица 5.2: Данни за Задача 5.1

Необходимо е да се извърши интерполация за $x = 3.947$.

Последователно постройте интерполационни полиноми от степен 1, 2, 3, 4 и 5, като за интерполационни възли избирате необходимия брой точки от таблицата, които са най-близко до $x = 3.947$.

Нанесете върху чертеж, с помощта на Matlab, графиките на получените интерполационни полиноми и определете този, който най-точно описва зависимостта. В него заместете x с 3.947, за да интерполирате таблично зададената функция в зададената точка.

Всъщност, $f(x) = e^x - 50$. Сравнете вашата апроксимация с действителната стойност. Определете абсолютната и относителната грешки.

✎ **Задача 5.2** *С помощта на Matlab интерполирайте функцията $f(x) =\mid x - 1 \mid$ в интервала $[0, 2]$, използвайки различен брой равноотдалечени интерполационни възли; възли получени по формулата на Чебишев; глобална и локална интерполации. Оценете графично качеството на приближение, като използвате графиките на $p(x)$, $f(x)$, и $p(x) - f(x)$ в интервала $[0, 2]$. От чертежа оценете грешката*

$$\| f - p \|_\infty = \max_{x \in [0,2]} \{ \mid f(x) - p(x) \mid \}.$$

❋ ❋

ГЛАВА 6

МЕТОД НА НАЙ-МАЛКИТЕ КВАДРАТИ

6.1 Същност на метода на най-малките квадрати

Нека на входа на някакво устройство се подава сигнал x, а на изхода се измерва сигнал y. Известно е, че величините x и y са свързани с функционална зависимост. Необходимо е, въз основа на опитни данни, да бъде определена приближено тази функционална зависимост $y = \varphi(x)$.

Нека в резултат на n измервания са получени експериментални точки (x_i, y_i). Известно е, че през n точки може винаги да се построи крива линия, чийто аналитичен израз е полином от степен $(n-1)$. Този полином се нарича *интерполационен*. Изобщо, замяната на функцията $\varphi(x)$ с функция $\psi(x)$ така, че техните стойности да съвпадат в зададените точки

$$\varphi(x_i) = \psi(x_i) \ , \quad i = 1, 2, \ldots, n \, ,$$

се нарича *интерполация*.

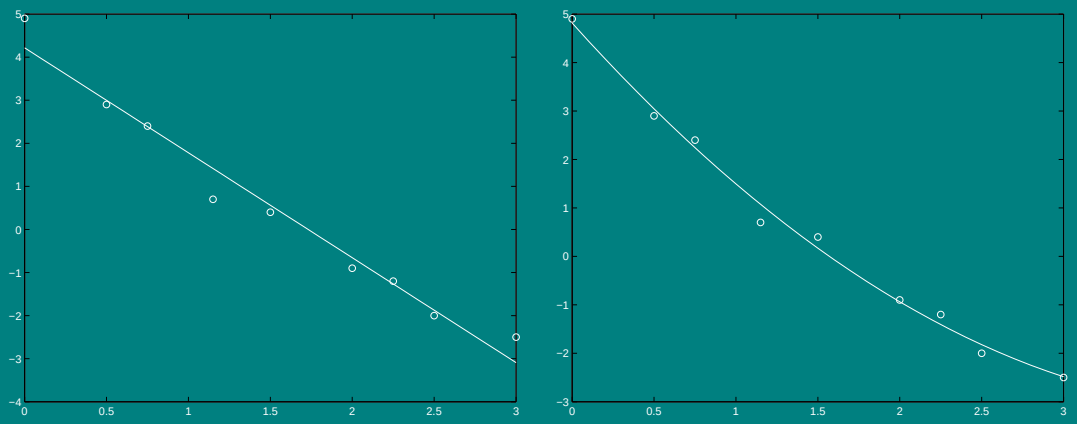
В случая, такова решение на проблема не е удовлетворително, поради неотчитането на други фактори (т.нар. случайни фактори), влияещи върху работата на устройството, като например влажност и температура на въздуха, които варират в някакви граници, субективни грешки при измерването и др., и в действителност, за получените стойности y_i е в сила представянето:

$$y_i = \varphi(x_i) + \varepsilon_i \, ,$$

където ε_i е случайната грешка.

Затова не е необходимо да се търси в случая крива линия, която да минава през всички експериментални точки, а такава, която да «изглади» наблюдаваните стойности, така че да изглежда наистина, че евентуалните отклонения от тази линия са малки и имат случаен характер.

Така например, за експерименталната зависимост, представена на Фигура 6.1 се вижда, че полиномът от втора степен на чертежа отдясно е по-подходящ за описването ѝ, в сравнение с правата линия на чертежа отляво.



(а) (б)

Фигура 6.1: Метод на най-малките квадрати

Методът на най-малките квадрати дава следния подход за определянето на апроксимиращата функция $\varphi(x)$: В декартова координатна система се нанасят получените експериментални точки $(x_i\,, y_i)$. По тяхното разположение се изказва предположение за принадлежността на търсената функция $\varphi(x)$ към даден клас функции, напр. линейна $\varphi(x) = a_0 + a_1 x$, квадратна $\varphi(x) = a_0 + a_1 x + a_2 x^2$ и т.н. В общия случай $\varphi(x) = \varphi(x, a_0, a_1, \ldots, a_r)$. Неизвестните параметри $a_0, a_1, \ldots, a_r$ се определят от изискването за минимум на сумата от квадратите на случайните грешки ε_i, т.е. минимум на величината

$$\mathcal{E} = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \varphi(x, a_0, a_1, \ldots, a_r))^2 \ . \tag{6.1}$$

Необходимо условие за минимум на функция на няколко променливи е обръщането в нула на първите ѝ частни производни:

$$\frac{\partial \mathcal{E}}{\partial a_j} = (-2) \sum_{i=1}^{n} (y_i - \varphi(x, a_0, a_1, \ldots, a_r)) \frac{\partial \varphi}{\partial a_j} = 0 \ , \quad j = 0, 1, \ldots, r \ . \tag{6.2}$$

Ако функцията $\varphi(x)$ е линейна по отношение на параметрите $a_0, a_1, \ldots, a_r$, горната система от $(r+1)$ уравнения с $(r+1)$ неизвестни също ще е линейна. След като системата се реши, се намират неизвестните параметри в търсената функционална връзка $y = \varphi(x)$, [11, 13, 24, 26].

6.2 Приближаване с линейна функция

Нека апроксимиращата функция е права линия

$$\varphi(x) = a_0 + a_1 x.$$

В този случай, системата (6.2) приема вида:

$$\left|\begin{array}{l} \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i) = 0 \\ \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)x_i = 0. \end{array}\right.$$

След преобразувания, се получава следната линейна система от две уравнения с неизвестни – параметрите a_0, a_1 на правата линия:

$$\left|\begin{array}{l} na_0 + a_1\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ a_0\sum_{i=1}^{n} x_i + a_1\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i. \end{array}\right. \tag{6.3}$$

След определянето на коефициентите, на база на получените експериментални стойности (x_i, y_i), системата лесно може да бъде решена.

6.3 Приближаване с квадратна функция

За коефициентите на полином от втора степен (парабола), т.е. при

$$\varphi(x) = a_0 + a_1 x + a_2 x^2,$$

се достига до система, аналогична на система (6.3), но вече с три уравнения и три неизвестни – коефициентите на параболата:

$$\left|\begin{array}{l} na_0 + a_1\sum_{i=1}^{n} x_i + a_2\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0\sum_{i=1}^{n} x_i + a_1\sum_{i=1}^{n} x_i^2 + a_2\sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i. \\ a_0\sum_{i=1}^{n} x_i^2 + a_1\sum_{i=1}^{n} x_i^3 + a_2\sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i. \end{array}\right. \tag{6.4}$$

За определянето на коефициентите на полином от произволна степен е необходимо да бъде решена система от линейни уравнения, аналогична на системи (6.3) и

(6.4). Може да се докаже, че детерминантите на тези системи са винаги различни от нула, т.е. решението винаги съществува и е единствено.

В програмната среда Matlab, апроксимацията с полином по метода на най-малките квадрати се осъществява посредством командата `polyfit`, която има следния синтаксис:

```
p=polyfit(x,y,k)
```

където:

- `x` е вектор от стойности на независимата променлива;
- `y` е векторът от съответните стойности на зависимата променлива;
- `k` е степента на полинома, с който желаем да интерполираме.

Командата `polyfit` връща вектор p, съдържащ коефициентите на търсения полином.

Пресмятането на стойностите на полинома за стойност на аргумента x_0, става с командата

```
polyval(p,x0)
```

В случай, че избраната стойност за k е с единица по-малка от броя на експерименталните точки (x_i, y_i), зададени с векторите x и y, полученият полином минава точно през всички точки (x_i, y_i), т.е. това е интерполационният полином за тези точки.

☞ **Пример 6.1** *Нека за дадена функция са известни следните експериментални стойности:*

x	0	0.5	0.75	1.15	1.5	2	2.25	2.5	3
y	4.9	2.9	2.4	0.7	0.4	-0.9	-1.2	-2	-2.5

Таблица 6.1: Данни за Пример 6.1

Да се апроксимира по метода на най-малките квадрати с полином и се начертае графиката му.

Решение: Първо трябва да се въведат дадените стойности на x и y от таблицата:

```
>>x=[0,0.5,0.75,1.15,1.5,2,2.25,2.5,3];
>>y=[4.9,2.9,2.4,0.7,0.4,-0.9,-1.2,-2,-2.5];
```

Най-напред да приближим с полином от първа степен (права линия):

```
>>1;m=1;
```

```
>>2;p=polyfit(x,y,m)
```

Да пресметнем стойностите, които приема полинома за стойностите на x от таблицата:

```
>>3;px=polyval(p,x)
```

Да пресметнем величината $\mathcal{E}$, зададена с (6.1) и да построим чертеж:

```
>>4;eps(m)=sum((y-px).^2)
>>5;xi=min(x):0.1:max(x);
```

```
>>6;pxi=polyval(p,xi);
>>7;plot(x,y,'o',xi,pxi)
```

От чертежа (Фигура 6.1а) и от изчислената стойност на $\mathcal{E} = \ldots\ldots$ се вижда, че получената права линия с уравнение $y = \ldots\ldots\ldots\ldots$ не е съвсем подходяща за описание на изследваната експериментална зависимост и вероятно биха се получили по-добри резултати при апроксимиране с полином от втора степен. За да се направи това, е необходимо да бъдат извикани команди от 1 до 7 в същата последователност, при това единствената промяна е в ред 1, на който трябва да се зададе новата стойност на m : $m = 2$.

В резултат получаваме, че най-подходящият полином от втора степен (в смисъла на метода на най-малките квадрати) е този с уравнение: $y = \ldots\ldots\ldots\ldots\ldots$ За него величината $\mathcal{E} = \ldots\ldots$ е по-малка, и от чертежа (Фигура 6.1б) се вижда, че той може да играе ролята на «изглаждаща» функция и следователно е подходящ за описание на изследваната зависимост.

6.4 Приближаване с нелинейна функция

Много често срещани в практиката видове зависимости, могат чрез подходяща смяна на променливите x и y, да бъдат сведени до линейни или квадратични зависимости. Например, ако зависимостта $y = ae^{bx}$ чрез логаритмуване се преобразува във вида: $\ln y = \ln a + bx$ и се извърши смяна на променливите $u = \ln y, y = x$, тя може да се запише: $u = a' + b'v$, където $a' = \ln a, b' = b$. Тоест може да се търси линейна връзка между променливите u и v. В случая, в качеството на експериментални стойности за u трябва да се вземат логаритмите от измерените стойности на y, а за стойности на променливата v – стойностите за x без промяна. След определяне на коефициентите a', b' на праволинейната зависимост по метода на най-малките квадрати, трябва да се извърши преход

Вид зависимост	Смяна на променливите		Ограничения	Обратна смяна на коефициентите	
Хиперболична $y = a + \frac{b}{x}$	$u = y$	$v = \frac{1}{x}$	$x \neq 0$	$a = a'$	$b = b'$
Дробно-рационална $y = \frac{x}{ax + b}$	$u = \frac{1}{y}$	$v = \frac{1}{x}$	$x \neq 0$, $y \neq 0$	$a = a'$	$b = b'$
Логаритмична $y = a + b \ln x$	$u = y$	$v = \ln x$	$x > 0$	$a = a'$	$b = b'$
Експоненциална $y = ae^{bx}$	$u = \ln y$	$v = x$	$x > 0$, $y > 0$	$a = e^{a'}$	$b = b'$
Степенна $y = ax^b$	$u = \ln y$	$v = \ln x$	$x > 0$, $y > 0$	$a = e^{a'}$	$b = b'$
Комбинирана $y = \frac{1}{a + be^{-x}}$	$u = \frac{1}{y}$	$v = e^{-x}$	$y \neq 0$	$a = a'$	$b = b'$

Таблица 6.2: Зависимости и свеждането им до линейна зависимост

към оригиналните коефициенти a и b, които се заместват накрая в изходното уравнение по отношение на x и y.

В Таблица 6.2 са дадени примери за зависимости, които могат да бъдат сведени до линейни чрез подходяща смяна на променливите, както и обратната трансформация за преход към оригиналните коефициенти a и b.

☞ **Пример 6.2** *Нов изолационен материал е подложен на проверка. За различни стойности на налягането x (10 паунда на кв.инч), приложено върху образец с дебелина 2 инча, се измерва свиването y на материала (0.1 инча). Получените стойности са дадени в Таблица 6.3.*

x	1	2	3	4	5	6
y	.9	1.2	1.6	2.5	3.6	4.8

Таблица 6.3: Данни за Пример 6.2

Определете параметрите в експоненциалния модел $y = a_0 e^{a_1 x}$ за описание на свойствата на материала.

Решение: Въвеждат се експерименталните стойности:

```
>>x=[1,2,3,4,5,6];
>>y=[0.9,1.2,1.6,2.5,3.6,4.8];
```

За да се провери дали зададеният модел е подходящ за описание на характеристиките на материала, трябва да се види дали зависимостта между x и $\ln y$ е приблизително линейна (виж Таблица 6.2).

Това лесно може да стане графично, с помощта на командата `semilogy`, която има същото действие като команда `plot`, но използва логаритмична скала по оста y. Аналогични са командите `semilogx` (логаритмична скала по оста x) и `loglog` (логаритмична скала и по двете оси), които са полезни при търсене съответно на логаритмична и степенна зависимости.

```
>>semilogy(x,y,'o')
```

Определянето на търсените коефициенти, пресмятането на грешката $\mathcal{E}$ от (6.1) на модела, както и изчертаването му стават с въвеждането на следната последователност от команди:

```
>>1;u=log(y);
>>2;v=x;
>>3;p=polyfit(v,u,1)
.........
>>4;a0=exp(p(2))
.........
>>5;a1=p(1)
.........
```

```
>>6;fx=a0.*exp(a1.*x)
.........
>>7;eps=sum((y-fx).^2)
.........
>>xi=min(x):0.1:max(x);
>>8;fxi=a0.*exp(a1.*xi);
>>9;plot(x,y,'o',xi,fxi)
```

Така, за коефициентите на търсения модел се получават стойности $a_0 = \ldots\ldots$, $a_1 = \ldots\ldots$. От получената стойност за $\mathcal{E} = \ldots\ldots$ се вижда, че полученият модел може да се използва за описание на свойствата на изследвания материал.

Ако видът на модела не е предварително известен, бихме могли след изчертаване на експерименталните точки, включително и със смяна на променливите по едната или по двете оси, да се ориентираме по разположението на точките при избора на определен тип зависимост.

Самата реализация на метода става аналогично на горния пример, като за всеки следващ модел се извикват само командите с номера от 1 до 9, при това редовете 3, 7 и 9 са без промяна, а останалите коригираме съобразно конкретния модел и Таблица 6.2.

6.5 Приближаване с функция, линейна по отношение на неизвестните параметри

Методът на най-малките квадрати е приложим и за доближаване с произволна функция, която е линейна по отношение на неизвестните параметри, т.е. функция от вида: $y = a_0+a_1f(x)+a_2g(x)+\ldots+a_kt(x)$, например $y = a_0+a_1e^{-x}+a_2xe^{-x}$ или $y = a_0 + a_1\cos x + a_2x^3$. Могат да се търсят и коефициенти в модели, които зависят от две и повече независими променливи $x_1, x_2, \ldots$ (за които разбира се разполагаме с експериментални данни), но и в този случай условието е зависимостта да бъде линейна спрямо неизвестните параметри в модела. Причината за това е, както отбелязахме още при изложението на метода в началото, че в този случай системата (6.2) за определяне на неизвестните коефициенти е линейна. В програмната среда Matlab тя може да бъде решена с използването на командата \. Следващият пример илюстрира това.

☞ **Пример 6.3** *Получени са следните експериментални стойности, (Таблица 6.4)*

x	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
y	3.0	3.3	3.4	3.5	3.5	3.4	3.3	3.2	3.1	3.0	2.9

Таблица 6.4: Данни за Пример 6.3

Определете коефициентите в модела $y = a_0 + a_1e^{-x} + a_2xe^{-x}$.

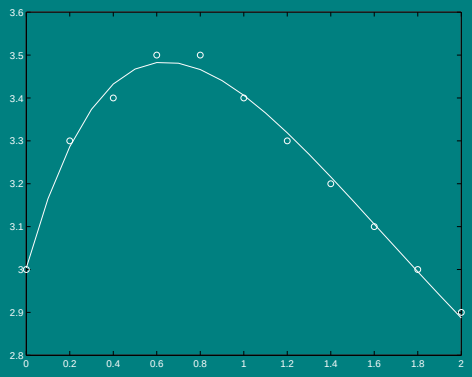
Решение: Най-напред трябва да се въведат данните като вектор-стълбове:

```
>>x=0:0.2:2;x=x';
>>y=[3;3.3;3.4;3.5;3.5;3.4;3.3;3.2;3.1;3;2.9];
```

За определянето на неизвестните параметри се прави следното. Конструира се матрица, която да има толкова редове, колкото е броят на експерименталните стойности за y. Всеки стълб на матрицата съответства на неизвестен коефициент в модела; неговите елементи са равни на стойността, която приема функцията, пред която стои този коефициент:

```
>>c=[ones(size(x)),exp(-x),x.*exp(-x)]
```

Определянето на неизвестните коефициенти в модела става със следната команда:



Фигура 6.2: Решение на Пример 6.3

```
>>a=c\y
```

```
.........
```

При това, Matlab автоматично съставя и решава системата (6.2), като привежда само крайния резултат – стойностите на неизвестните коефициенти във вид на вектора a, като в случая първият елемент на a е търсената стойност за a_0, вторият - за a_1 и третият – за a_2.

Пресмятането на сумата от квадратите на отделните грешки, $\mathcal{E}$, става със следните команди:

```
>>fx=c*a
```

```
>>eps=sum((y-fx).^2)
```

За изчертаването на получения модел (виж Фигура 6.2) в случая (понеже стойностите на x са близки една до друга) е достатъчна командата:

```
>>plot(x,y,'o',x,fx)
```

В общия случай изчертаването може да стане с командите:

```
>>xi=min(x):0.1:max(x);xi=xi';
>>fxi=[ones(size(xi)),exp(-xi),xi.*exp(-xi)]*a;
>>plot(x,y,'o',xi,fxi)
```

Начинът, по който е решен този пример може да се използва в частност и за доближаване с полином, тъй като всеки полином е също линейна функция от коефициентите си. Все пак използването на командата `polyfit` е по-удобно.

x	0.1	0.2	0.5	0.7	3.1	3.5
y	0.3948	1.7811	3.6137	4.2867	7.2628	7.5055

Таблица 6.5: Данни за Задача 6.1

Определянето на коефициентите в зависимости, които не са линейни по отношение на търсените параметри, също е възможно, но се извършва с помощта на методи за намиране на минимум на функция на няколко променливи. В Matlab командата, която има такова предназначение е `fmins`.

6.6 Задачи за упражнение

✎ **Задача 6.1** *Да се приближи таблично зададената функция (Таблица 6.5) по метода на най-малките квадрати с функция от вида* $y = a + b \ln x$.

✎ **Задача 6.2** *Да се приближи таблично зададената функция (Таблица 6.6) по метода на най-малките квадрати с функция от вида* $y = a \exp(b/x)$.

x	0.1	1	2	2.2	3	3.5
y	73.89	12.21	11.05	10.95	10.69	10.59

Таблица 6.6: Данни за Задача 6.2

✎ **Задача 6.3** *Да се приближи таблично зададената функция (Таблица 6.7) по метода на най-малките квадрати с функция от вида* $y = ax^b$ *и да се пресметне стойността ѝ за* $x = 6.1$.

x	0.1	1	2	3	4	5.5
y	2e-03	0.2	0.8	1.8	3.2	6.05

Таблица 6.7: Данни за Задача 6.3

✎ **Задача 6.4** *При изследването на даден показател* y *са получени следните експериментални стойности за зависимостта му от факторната променлива* x, *(Таблица 6.8. Определете коефициентите в модела:* $y = a_0 + a_1 x + a_2 x^2 +$

x	1	1.5	2	2.5	3	3.5	4	4.5	5
y	6.9	6.8	7.6	8.9	9.4	8.2	5.5	2.7	0.7

Таблица 6.8: Данни за Задача 6.4

$a_3 \cos 2x$.

Упътване: `>>c=[ones(size(x)),x,x.^2,cos(2.*x)]`

* *

ГЛАВА 7

ЧИСЛЕНО ИНТЕГРИРАНЕ

7.1 Уводни бележки

Нека в интервала $[a, b]$ е зададена функцията $y = f(x)$. С помощта на точките $x_0, x_1, \ldots, x_n$ се разбива интервала $[a, b]$ на подинтервали $[x_{j-1}, x_j]$, $j = 1, 2, \ldots, n$; при това $x_0 = a$, $x_n = b$. Във всеки от тези подинтервали се избира произволна точка ξ_j $(x_{j-1} \leq \xi_j \leq x_j)$ и се намира произведението s_j от стойностите на функцията в тази точка $f(\xi_j)$ и дължината на всеки подинтервал $\Delta x_j = x_{j+1} - x_j$:

$$s_j = f(\xi_j)\Delta x_j. \tag{7.1}$$

Съставя се сумата на всички такива произведения:

$$S_n = s_1 + s_2 + \ldots + s_n = \sum_{j=1}^{n} f(\xi_j)\Delta x_j. \tag{7.2}$$

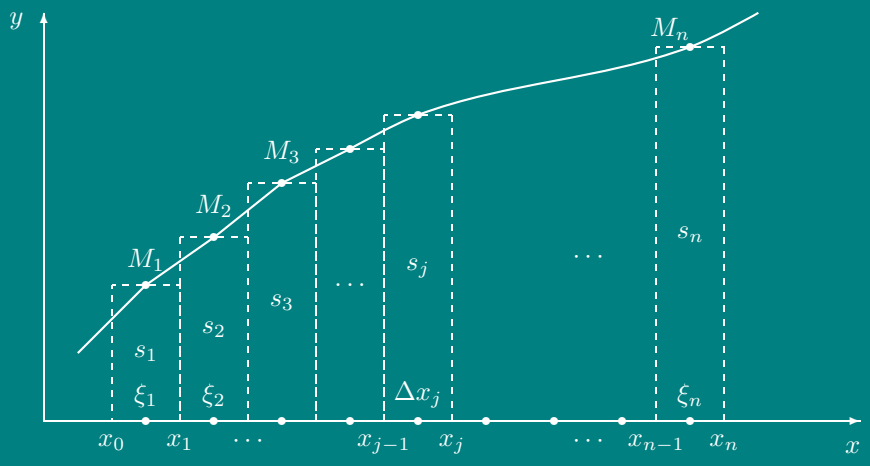
Тази сума се нарича **интегрална сума**.

➠**Дефиниция 9 Определен интеграл** *в интервала* $[a, b]$ *се нарича границата на интегралната сума* S_n *при неограничено увеличаване на броя на точките на разбиване; при това дължината на най-големия от под-интервалите клони към нула, т.е.*

$$\int_a^b f(x)\,dx = \lim_{\max \Delta x_j \to 0} \sum_{j=1}^{n} f(\xi_j)\Delta x_j. \tag{7.3}$$

Геометричен смисъл: (фиг.1) при $f(x) > 0$.

Изразите (7.1) при $j = 1, 2, \ldots, n$ са лицата на елементарните правоъгълници (с пунктирани линии), интегралната сума (7.2) - лицето на стъпаловидната



Фигура 7.1: Геометричен смисъл на численото интегриране

фигура. При $n \to \infty$ и $\Delta x_j \to 0$ горната граница на фигурата (начупената линия) клони към линията $y = f(x)$. Лицето на получената фигура (криволинеен трапец) е равно на определения интеграл (7.3).

Трябва да се отбележи, че към пресмятане на определен интеграл се свеждат много практически задачи: пресмятане на лице на фигура, дължина на крива, обем, център на тежестта и др. Решаването на кратни интеграли в крайна сметка също може да бъде сведено до пресмятане на определен интеграл.

В много случаи, когато подинтегралната функция е зададена в аналитичен вид, определеният интеграл може да се изчисли непосредствено с помощта на неопределен интеграл по *Формулата на Нютон-Лайбниц*:

$$\int_a^b f(x)\,dx = F(x)\Big|_a^b = F(b) - F(a),$$

където $F(x)$ е примитивна функция на $f(x)$.

Но на практика често *Формулата на Нютон-Лайбниц* е неприложима по две основни причини:

❶ Функцията $f(x)$ е от вид, който не допуска непосредствено интегриране, т.е. примитивната $F(x)$ не може да се изрази чрез елементарни функции;

❷ Стойностите на функцията $f(x)$ са зададени само във фиксиран краен брой точки, т.е. подинтегралната функция е зададена таблично.

В тези случаи се използват **методи на числено интегриране**. Те се основават на **апроксимация** на подинтегралната функция с по-прости изрази, напр. полиноми.

Един от тези способи (за случай 1) е представяне на подинтегралната функция във вид на степенен ред (ред на Тейлър или Маклорен). Така интегрирането на

сложна функция се свежда до интегриране на полином, състоящ се от първите няколко члена на реда, при което естествено се допуска грешка $\epsilon > 0$. По този начин интегрират и електронните калкулатори.

☞ **Пример 7.1** *Да се изчисли* $\int_0^1 e^{-x^2}\,dx$ *с грешка* 10^{-2}.

Решение:

От известното разлагане на експоненциалната функция в ред на Тейлър (Маклорен)

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \frac{x^{n+1}}{(n+1)!}e^{\theta x},$$

където последното събираемо е грешката R_n, $0 < \theta < 1$. При замяна на x с $(-x^2)$ се получава реда

$$e^{-x^2} = 1 - x^2 + \frac{x^4}{2!} + \ldots + \frac{(-x^2)^n}{n!} + \frac{(-x^2)^{n+1}}{(n+1)!}e^{\theta x}.$$

Тъй като $0 < x < 1$ и $0 < \theta < 1$, грешката $\mid R_n \mid < 1/(n+1)!$ и за да бъде тя от порядък 10^{-2} трябва да се подбере такова n, че $\mid R_n \mid < 10^{-2}$, т.е. $(n+1)! > 100$. Вижда се, че последното неравенство е изпълнено за $n = 4$, т.е. за да се изчисли интеграла със зададената точност трябва да се вземат първите 5 члена на реда:

$$e^{-x^2} \approx 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!}.$$

Интегрира се в граници от 0 до 1:

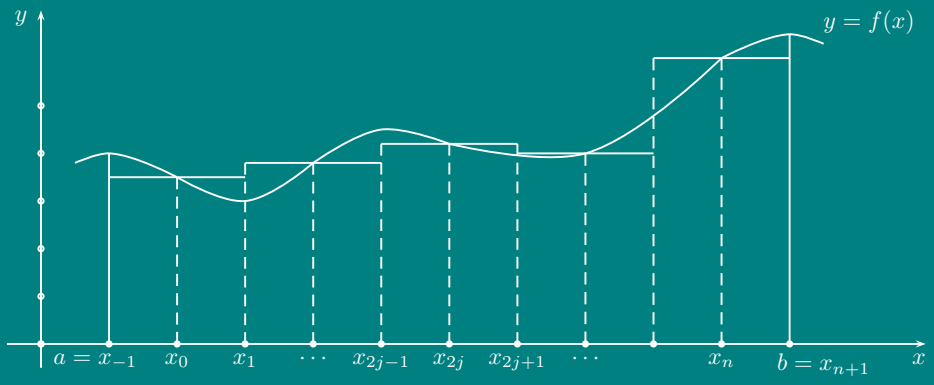
$$\int_0^1 e^{-x^2} \approx \left(x - \frac{x^3}{3} + \frac{1}{2!}\frac{x^5}{5} - \frac{1}{3!}\frac{x^7}{7} + \frac{1}{4!}\frac{x^9}{9} \right)\Bigg|_0^1 .$$

```
>>1-1/3+1/(5*prod(1:2))-1/(7*prod(1:3))+1/(9*prod(1:4))
......
```

По-универсални методи, които могат да се използват и в двата случая са методите на числено интегриране, основани на **апроксимация** на подинтегралната функция с **интерполационни** полиноми. По-нататък ще използваме частична (локална) интерполация. В зависимост от апроксимиращата функция се получават различни методи (методи на правоъгълници, трапеци, параболи, сплайни и др.), [11, 13, 24, 26].

7.2 Метод на правоъгълниците

Най-простият метод на числено интегриране е *Методът на правоъгълниците*. При него определеният интеграл се заменя непосредствено с интегралната сума



Фигура 7.2: Метод на средната точка

(7.2). За точки ξ_j могат да се избират левите ($\xi_j = x_{j-1}$) или десните ($\xi_j = x_j$) граници на под-интервалите, (Фиг. 7.1). За двата случая формулите на *метода на правоъгълниците* са съответно:

$$\int_a^b f(x)\,dx \approx \sum_{j=1}^{n} f(x_{j-1})\Delta x_j, \tag{7.4}$$

$$\int_a^b f(x)\,dx \approx \sum_{j=1}^{n} f(x_j)\Delta x_j, \tag{7.5}$$

Широко разпространени и по-точни са формулите на правоъгълниците, използващи стойностите на функцията в средните точки на елементарните отсечки. Тази модификация на метода на правоъгълниците се нарича **метод на средната точка**. Интервалът $[a, b]$ се разделя на $n+2$ подинтервала с постоянна стъпка $h = (b-a)/(n+2)$, (Фиг. 7.2).

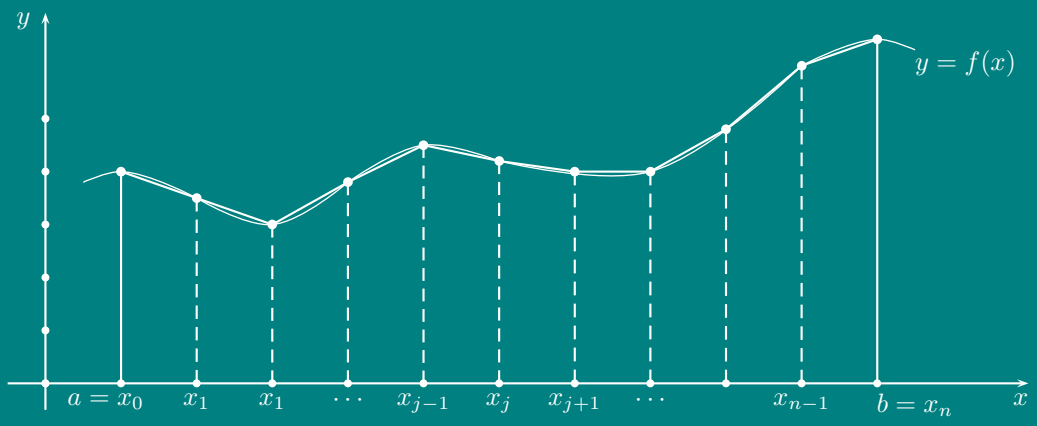
Определеният интеграл се приближава по формулата на средната точка:

$$\int_a^b f(x)\,dx = 2h\sum_{j=0}^{n/2} f(x_{2j}) + \frac{(b-a)h^2}{6} f''(\xi), \tag{7.6}$$

за някое $\xi \in (a, b)$, където $x_j = a+(j+1)h$ за всяко $j = -1, 0, \ldots, n+1$. Последното събираемо в (7.6) е грешката на *Метода на средната точка.*

7.3 Метод на трапеците

Методът на трапеците използва локална линейна интерполация, т.е. графиката на функцията $y = f(x)$ се заменя с начупена линия, съединяваща точките (x_j, y_j), (Фиг. 7.3). Интервалът $[a, b]$ се разделя на n подинтервали с постоянна стъпка



Фигура 7.3: Метод на трапеците

$h = (b-a)/n$ и възли $x_j = a + jh$ за всяко $j = 0, 1, \ldots, n$. Определеният интеграл се пресмята приближено по *Формулата на трапеците*:

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{j=1}^{n-1} f(x_j)\right] - \frac{(b-a)h^2}{12} f''(\xi), \qquad (7.7)$$

за някое $\xi \in (a, b)$. Последното събираемо в (7.7) е грешката на *Метода на трапеците*.

☞ **Пример 7.2** *Да се пресметне по Метода на трапеците*

$$\int_{0^0}^{90^0} \sin x\,dx,$$

ако подинтегралната функция е зададена таблично в 7 точки (Табл.7.1).

x^0	0	15	30	45	60	75	90
x_{rad}							
$\sin x$							

Таблица 7.1:

```
>>angle=0:15:90;x=(pi*angle/180)'
>>n=length(x);br=n-1;a=x(1);b=x(n);h=(b-a)/br;
>>y=sin(x)      % попълнете таблицата
```

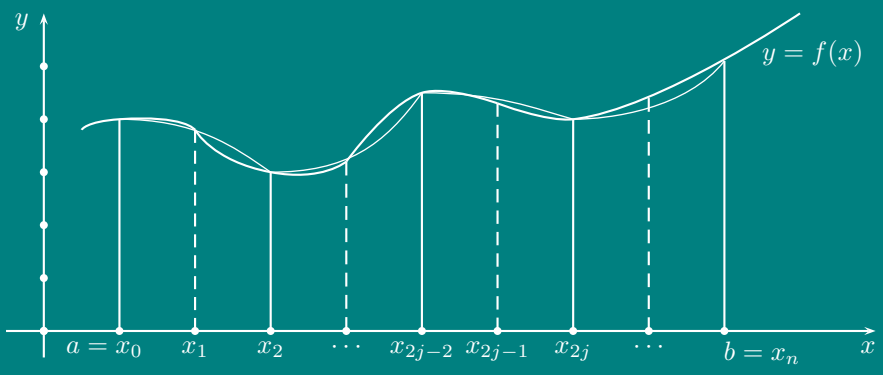
```
>>IT=(y(1)+y(n)+2*sum(y(2:n-1)))*h/2
......
>>err=((b-a)*(h^2))/12
.......
>>fplot('sin',[x(1),x(n)]),hold on
>>plot(x,y,'g')
```

Тъй като $f''(x) = -\sin x$ е ограничена функция, то за грешката на *метода на трапеците* е вярно неравенството

$$\left|\frac{(b-a)h^2}{12}\sin\xi\right| \leq \frac{(b-a)h^2}{12}.$$

7.4 Метод на параболите (Симпсън)

Методът на Симпсон използва локална квадратична интерполация. Интервалът $[a,b]$ се разделя на **четен** брой подинтервали $n = 2k$ с постоянна стъпка $h = (b-a)/n$ и възли $x_j = a + jh$ за всяко $j = 0, 1, \ldots, n$. Във всяка двойка съседни интервали $[x_{2j-2}, x_{2j-1}]$ и $[x_{2j-1}, x_{2j}]$ графиката на функцията $y = f(x)$ се заменя с парабола, (Фиг. 7.4).



Фигура 7.4: Метод на параболите

Определеният интеграл се пресмята приближено по *Формулата на Симпсон*:

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)\right] - \frac{(b-a)h^4}{180}f^{(4)}(\xi), \tag{7.8}$$

за някое $\xi \in (a,b)$. Последното събираемо в (7.8) е грешката на *Метода на Симпсон*.

☞ **Пример 7.3** *Да се пресметне по Метода на Симпсон интегралът от Пример 7.2. Да се сравнят получените стойности за интеграла и грешките на двата метода.*

Решение: (продължение)

```
>>c=[1,4,1];
>>IS=0;
>>for i=1:2:(n-2)
      IS=IS+c*[y(i);y(i+1);y(i+2)];
  end
>>IS=IS*h/3
......
>>err_s=((b-a)*(h^4))/180
.......
>>format short e
>>err,err_s
>>IT,IS
```

Тъй като $f^{(4)}(x) = \sin x$ е ограничена функция, то за грешката на *Метода на Симпсън* е вярно неравенството

$$\left| -\frac{(b-a)h^4}{180} \sin \xi \right| \leq \frac{(b-a)h^4}{180}.$$

7.5 Други методи на числено интегриране

Изложените по-горе методи (в случая на постоянна стъпка h) се използват и с променлива стъпка h_i.

Много популярна е техниката, наречена «Интегриране на Ромберг», при която стойностите на интеграла за различни набори от подинтервали се комбинират с цел - повишаване на точността.

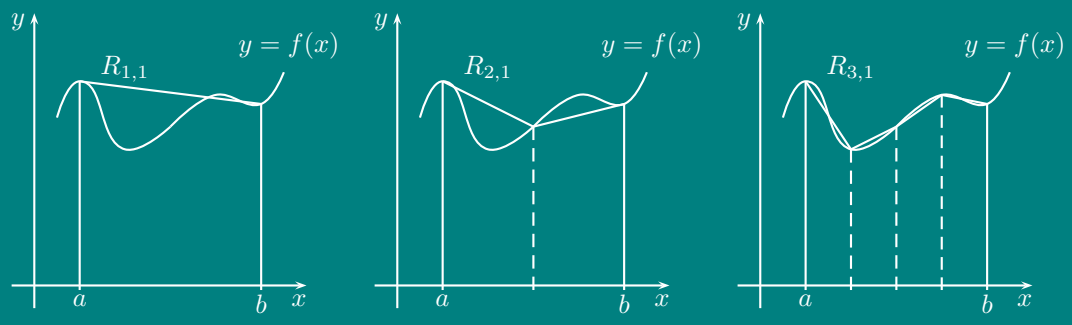
Например, нека да разгледаме процеса на интегриране при реализиране на *метода на трапеците по схемата на Ромберг.*

Първата стъпка в процеса на Ромберг включва апроксимация по правилото на трапеците, като $[a, b]$ се дели на $1, 2, 4, \ldots, 2^{n-1}$ части, $n \in \mathcal{N}$ (виж Фиг.7.5 при $n = 3$), като стойностите, получени при $n = 1$ се използват при $n = 2$ и т.н.

$$\begin{aligned} R_{1,1} &= \frac{h_1}{2}\left[f(a) + f(b)\right] = \frac{(b-a)}{2}\left[f(a) + f(b)\right]; \\ R_{2,1} &= \frac{1}{2}\left[R_{1,1} + h_1 f(a + h_2)\right]; \\ R_{3,1} &= \frac{1}{2}\left\{R_{2,1} + h_2\left[f(a + h_3) + f(a + 3h_3)\right]\right\}. \end{aligned}$$

Горните три рекурентни формули могат да се запишат така:

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right] \quad \forall k = 2, 3, \ldots, n.$$



Фигура 7.5: Метод на Ромберг

☞ **Пример 7.4** *Като се използва правилото на трапеците, да се направи първата стъпка от схемата на Интегриране на Ромберг при* $n = 6$*, за апроксимиране на интеграла*

$$\int_0^{\pi} \sin x \, dx.$$

Решение:

$$\begin{aligned}
R_{1,1} &= \frac{\pi}{2}[\sin 0 + \sin \pi] = 0, \qquad R_{2,1} = \frac{1}{2}\left[R_{1,1} + \pi \sin \frac{\pi}{2}\right] = 1.57079633,\\
R_{3,1} &= \frac{1}{2}\left[R_{2,1} + \frac{\pi}{2}\left(\sin \frac{\pi}{4} + \sin \frac{3\pi}{4}\right)\right] = 1.89611890,\\
R_{4,1} &= \frac{1}{2}\left[R_{3,1} + \frac{\pi}{4}\left(\sin \frac{\pi}{8} + \sin \frac{3\pi}{8} + \sin \frac{5\pi}{8} + \sin \frac{7\pi}{8}\right)\right] = 1.97423160,\\
R_{5,1} &= 1.99357034, \qquad R_{6,1} = 1.99839336.
\end{aligned}$$

На **Втората стъпка** в процеса на Ромберг се прилага екстраполация на Ричардсън, като се пресмятат за $\forall k = 2, 3, 4, \ldots, n$ и $j = 2, \ldots, k$ апроксимационните формули с $\mathcal{O}(h_k^{2j})$:

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}\,.$$

Резултатите, които се получават от тази формула, са дадени в Табл. 7.2, като попълването е по редове.

7.6 Числено интегриране с Matlab

Компютърните програми използват **адаптивни процедури за квадратури**. Това са числени алгоритми, при които се използват една или две основни квадратурни формули и автоматично се определят големините на подинтервалите

$R_{1,1}$					
$R_{2,1}$	$R_{2,2}$				
$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
$\vdots$	$\vdots$	$\vdots$		$\ddots$	
$R_{n,1}$	$R_{n,2}$	$R_{n,3}$	$R_{n,4}$	$\cdots$	$R_{n,n}$

Таблица 7.2: Резултати при Метода на Ромберг

така, че получената стойност за интеграла да удовлетворява зададена точност. В различните части на интервала на интегриране се използват мрежи с различни размери, като мрежите с по-големи размери се използват там, където подинтегралната функция е гладка и се мени бавно, докато по-финитe мрежи се използват в области, където интегрирането става трудно.

В Matlab за решаване на определени интеграли се използват функциите `quad` и `quad8`, със следния синтаксис:

```
quad('fname',a,b,tol,trace)
```

```
quad8('fname',a,b,tol,trace)
```

където:

- `fname` - име на подинтегралната функция, зададена като `.m` - файл;
- `a,b` - граници на интегриране;
- `tol` - точност (грешка) на изчисление;
- `trace` - Ако `trace`$\neq 0$, се чертае графика, показваща процеса на интегриране.

Програмата `quad` използва адаптивен рекурсивен метод на Симпсън, а `quad8` - адаптивен рекурсивен метод с квадратурни формули на Нютон-Коутс от 8-ми ред. Командата

```
quaddemo
```

демонстрира използването на командата `quad` за числено интегриране.

☞ **Пример 7.5** *Да се пресметне интегралът:* $\int_0^{\pi/2} t \sin t \, dt$.

Решение: Подинтегралната функция трябва предварително да се зададе в отделен файл с разширение `.m`, (напр. `ime.m`). Съдържанието на този файл е:

ime.m

```
1 function y=ime(t)
2 y=t.*sin(t);
```

След като се запише файла `ime.m`, от командния ред на Matlab се изпълнява командата

```
>>quad('ime',0,pi/2)
```

```
>>quad('ime',0,pi/2,1e-04,1)
```

7.7 Задачи за упражнение

Да се решат интегралите:

Задача 7.1 $\int_{10}^{20} \arcsin(\sin x^3)\,dx$

Задача 7.2 $\int_{1}^{9} 3\sqrt{x}(1+\sqrt{x})\,dx$

Задача 7.3 $\int_{0}^{0.9} \sqrt{1-t^4}\,dt$

Задача 7.4 $\int_{1}^{2} (\sqrt{z}-1)^2\,dz$

Задача 7.5 $\int_{0}^{\pi/3} \cos^2(3t)\,dt$

Задача 7.6 $\int_{0}^{1} e^t\sqrt{1-e^t}\,dt$

Задача 7.7 $\int_{0}^{\pi/2} \sin 2t \cos 3t\,dt$

Задача 7.8 $\int_{0}^{\sqrt{3}} \frac{s\,ds}{\sqrt{4-s^2}}$

Задача 7.9 $\int_{0.1}^{2} \sin\frac{1}{x}\,dx$

Задача 7.10 $\int_{0.5}^{1} e^{ln(arctan\frac{1}{x})}\,dx$

Задача 7.11 $\int_{0}^{1} \frac{x^2}{1+x^2}\arcsin\,x\,dx$

Задача 7.12 $\int_{1}^{1.5} x^2 ln(x)\,dx$

Задача 7.13 $\int_{0}^{1} x^2 e^{-x}\,dx$

Задача 7.14 $\int_{0}^{0.35} \frac{2}{x^2-4}\,dx$

Задача 7.15 $\int_{0}^{\pi/4} x^2 \sin x\,dx$

Задача 7.16 $\int_{0}^{\pi/4} e^{3x}\sin 2x\,dx$

Задача 7.17 $\int_{1}^{1.6} \frac{2x}{x^2-4}\,dx$

Задача 7.18 $\int_{3}^{3.5} \frac{x}{\sqrt{1+x^2}}\,dx$

Задача 7.19 $\int_{0}^{\pi/4} (\cos x)^2\,dx$

Задача 7.20 $\int_{0}^{1} x^{1/3}\,dx$

ГЛАВА 8

ЧИСЛЕНО РЕШАВАНЕ НА ОБИКНОВЕНИ ДИФЕРЕНЦИАЛНИ УРАВНЕНИЯ И СИСТЕМИ

Обикновените диференциални уравнения (ОДУ) са най-често използваното средство в науката и инженерната практика за моделиране на динамични системи. Те се използват в различни области, например за описание на движението на космическите тела, за моделиране на химични процеси; развитието на животински популации; борсовите цени на акции, в механиката, електротехниката и др., [11, 13, 24, 26]

В общия случай, трябва да се намери функция $Y(t)$, удовлетворяваща уравнението

$$Y'(t) = F(t, Y(t)), \quad \text{за} \quad t \geq t_0 \tag{8.1}$$

при начално условие

$$Y(t_0) = Y_0. \tag{8.2}$$

Най-често променливата t е времето и задачата е да се намери решението $Y(t)$ за $t_0 \leq t \leq t_f$, където t_f е зададено финално време.

Задачата (8.1)–(8.2) е известна като *задача на Коши*. Функцията $Y(t)$ може да бъде и векторна

$$Y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{pmatrix}, \quad \text{тогава} \quad Y'(t) = \begin{pmatrix} y_1'(t) \\ y_2'(t) \\ \vdots \\ y_n'(t) \end{pmatrix}.$$

Уравнение (8.1) в този случай задава система от диференциални уравнения от първи ред. Ако дясната част на (8.1) е достатъчно гладка функция, задачата

на Коши има единствено решение.

Числените алгоритми за решаване на (8.1) пресмятат приближения Y_0, Y_1, Y_2, ... ,Y_N за стойностите на търсената функция $Y(t)$ във фиксирани моменти от време $t_0 < t_1 < \cdots < t_N = t_f$. Нека $h_k = t_{k+1} - t_k$.

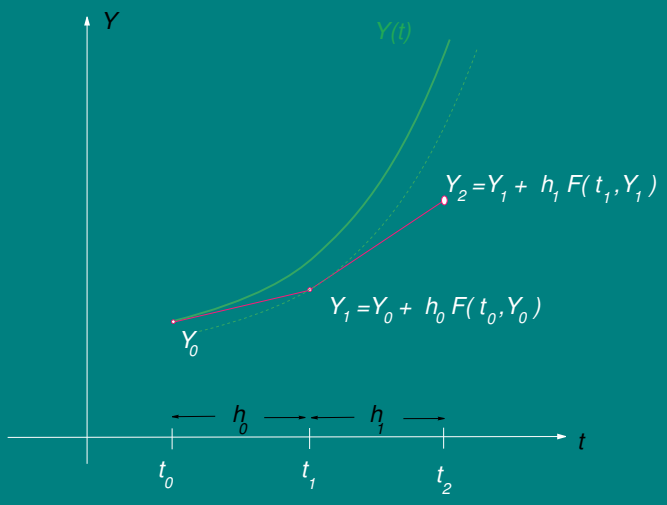
8.1 Метод на Ойлер

В началото, от началното условие, е известна само стойността Y_0 на Y в момента време t_0. От уравнение (8.1) се вижда, че ъгловият коефициент на допирателната към решението $Y(t)$ в точка t_0 е $F(t_0, Y_0)$. Ако предположим, че $Y(t)$ се изменя бавно (или все едно, че стъпката h_0 е достатъчно малка), то можем да заменим Y в околност на точката t_0 с допирателната ѝ в тази точка. Стойността на Y в точката t_1 тогава може да бъде приближено пресметната като се замести в уравнението на допирателната с $t = t_1$:

$$Y_1 = Y_0 + h_0 F(t_0, Y_0).$$

Като се повторят горните разсъждения, последователно се определят приближени стойности на Y за следващите моменти от време: най-напред за Y_2, след това за Y_3 и т.н., накрая за Y_N по формулата

$$Y_{k+1} = Y_k + h_k F(t_k, Y_k). \tag{8.3}$$



Фигура 8.1: Метод на Ойлер

Грешката, която се допуска при замяна на действителните стойности на Y с така изчислените по формула (8.3), е най-малка на първата стъпка, при пресмятането на Y_1, защото след това се наслагват и грешките от предишните стъпки.

По формулата на Тейлър

$$Y(t_1) = Y(t_0 + h_0) = Y(t_0) + h_0 Y'(t_0) + \mathcal{O}(h_0^2) = Y_0 + h_0 F(t_0, Y_0) + \mathcal{O}(h_0^2),$$

затова

$$\varepsilon_1 = Y(t_1) - Y_1 = \mathcal{O}(h_0^2).$$

Ако се въведат означенията

$$\varepsilon = \max_{1 \le k \le N} \mid \varepsilon_k \mid = \max_{1 \le k \le N} \mid Y(t_k) - Y_k \mid, \quad h = \max_{1 \le k \le N} h_k,$$

може да се докаже, че $\varepsilon = \mathcal{O}(h)$. Това означава, че при $h \to 0$ тоталната грешка ε ще намалява пропорционално на h, т.е. ако h се намали два пъти, то и ε ще намалее два пъти. Затова се казва, че методът на Ойлер има точност от първи ред.

8.2 Модифициран метод на Ойлер

От Фигура 8.1 се вижда, че биха се получили по-точни резултати, ако на всяка стъпка функцията $Y(t)$ се замени не с допирателната ѝ в точката t_k, а с права линия, чиито ъглов коефициент е някъде между ъгловите коефициенти на допирателните в точките t_k и t_{k+1}, например $\frac{1}{2}\left[Y'(t_k) + Y'(t_{k+1})\right]$. Тъй като стойността на $Y(t)$ за $t = t_{k+1}$ не е известна (всъщност точно нея се опитваме да намерим на поредната стъпка), за да я заместим в дясната част на уравнението и изчислим $Y'(t_{k+1})$, се налага да се извърши едно междинно изчисление по формула (8.3) от метода на Ойлер за определяне на приближение за $Y(t_{k+1})$:

$$Y_{k+1} = Y_k + h_k F(t_k, Y_k),$$

и след това то да бъде доуточнено по формулата:

$$Y_{k+1} = Y_k + \frac{h_k}{2}\left[F(t_k, Y_k) + F(t_{k+1}, Y_{k+1})\right].$$

Може да се докаже, че точността на така получените формули вече е от втори ред. Това означава, че ако разстоянията между моментите от време t_k бъдат намалени два пъти (например с въвеждането на допълнителни точки), грешката при определянето на $Y(t_k)$ ще се намали четири пъти.

Горните формули могат да бъдат записани по следният начин:

$$\begin{aligned} K_0 &= h_k F(t_k, Y_k) \\ K_1 &= h_k F(t_k + h_k, Y_k + K_0) \\ Y_{k+1} &= Y_k + (K_0 + K_1)/2 \end{aligned}$$

8.3 Метод на Рунге – Кута

Модифицираният метод на Ойлер може да се разглежда като частен случай на по-общия **метод на Рунге – Кута**. При него, на всяка стъпка се извършват следните изчисления:

$$\begin{aligned} K_0 &= h_k F(t_k + \alpha_0 h_k, Y_k) \\ K_1 &= h_k F(t_k + \alpha_1 h_k, Y_k + \beta_{10} K_0) \\ K_2 &= h_k F(t_k + \alpha_2 h_k, Y_k + \beta_{20} K_0 + \beta_{21} K_1) \\ &\dots \\ K_r &= h_k F(t_k + \alpha_r h_k, Y_k + \beta_{r0} K_0 + \beta_{r1} K_1 + \ldots + \beta_{r,r-1} K_{r-1}) \end{aligned}$$

$$Y_{k+1} = Y_k + \omega_0 K_0 + \omega_1 K_1 + \ldots + \omega_r K_r. \tag{8.4}$$

Точност от четвърти ред се получава, ако в горните формули

$$r = 3\ , \quad \alpha_0 = 0, \quad \alpha_1 = \alpha_2 = 1/2, \quad \alpha_3 = 1, \quad \beta_{10} = 1/2, \quad \beta_{20} = 0, \quad \beta_{21} = 1/2$$
$$\beta_{30} = \beta_{31} = 0, \quad \beta_{32} = 1, \quad \omega_0 = \omega_3 = 1/6, \quad \omega_1 = \omega_2 = 1/3.$$

8.4 Адаптивни алгоритми

Често в практиката, решението $Y(t)$ на диференциалното уравнение (8.1) се изменя твърде бързо около някои критични стойности на времето t. Това влошава точността на приближенията Y_k и затова около такива стойности на t е необходимо изчисленията да се извършват с по-малка стъпка h_k. Другаде пък, с цел намаляване на пресмятанията, е възможно стъпката да бъде увеличена, като при това точността остава в избраните граници. Съвременните изчислителни алгоритми автоматично определят големините на стъпките h_k, така че да се подсигури зададената от потребителя точност.

При адаптивния алгоритъм на Дорманд - Принс, заложен в командата `ode45` на Matlab за решаване на ОДУ, на всяка стъпка се изчислява Y_{k+1} като се използват два различни набора от стойности ω_i в (8.4): единият подсигурява точност от ред 4, а другият – от ред 5. Изчислява се разликата между така пресметнатите две оценки за $Y(t_{k+1})$:

$$e = Y_{k+1}^{[4]} - Y_{k+1}^{[5]},$$

като оценка за локалната грешка. Ако $\mid e \mid$ е по-малка от зададената от потребителя точност tol, за оценка на $Y(t_{k+1})$ се приема $Y_{k+1}^{[5]}$ и се преминава към следващата стъпка. В противен случай, изчисленията се повтарят с по-малка стъпка. И в двата случая, размера на новата стъпка се определя по формулата:

$$h_{new} = 0.9h \left(\frac{tol}{\mid e \mid} \right)^{1/5},$$

където h е размера на последната стъпка (независимо дали е успешна или не), за която е изчислена $\mid e \mid$.

8.5 Решаване на ОДУ с Matlab

Няколко са **командите в Matlab** за решаване на ОДУ – `ode45`, `ode23`, `ode113`, а също и серия от команди за решаване на нееластични ОДУ – `ode15s`, `ode23s`, `ode23t`, `ode23tb`. Всички те имат сходен синтаксис, което улеснява тяхното използване. Версии 3 и 4 поддържат само `ode45` и `ode23` със следния синтаксис:

```
[T,Y]=команда('име',t0,tf,y0,tol,trace)
```

където:

- **име** е името на файл, описващ системата ОДУ;
- **t0** задава начална стойност на t ;
- **tf** задава крайна стойност за t ;
- **y0** е вектор-стълб от начални стойности за $Y(t)$ при $t = t_0$;
- **tol** е параметър, задаващ точността на полученото решение; по подразбиране $tol = 10^{-3}$;
- **trace** дава възможност при задаване на стойност, различна от нула, например 1, да се извеждат на екрана всички междинни стъпки; по подразбиране `trace=0`.

Изходните аргументи са:

- **T** – вектор-стълб от моментите време, за които е пресметнато решението;
- **Y** е матрица, i-тия стълб на която съдържа получените стойности на i-тата компонента на решението Y, в моментите от време, зададени в T; Ако (8.1) задава не система от ОДУ, а само едно диференциално уравнение, то Y е вектор-стълб.

Командата `ode45` реализира изложения по-горе адаптивен алгоритъм на Дорманд – Принс. Командата `ode23` се базира на аналогичен адаптивен алгоритъм от по-нисък ред на точност и е по-подходяща за по-груби изчисления на $Y(t)$, както и при ниска степен на нееластичност на задачата. Командата `ode113` има променлив ред на точност. Тя е особено подходяща, когато изчисляването на дясната страна на (8.1) е скъпо струващо. Алгоритъмът използва на всяка стъпка стойностите на $Y(t)$, получени на няколко предходни стъпки.

След изпълнението на някоя от горните команди, с помощта на команда `plot`, получените стойности на решението, съдържащи се в изходната променлива Y, за моментите от време, зададени в изходната променлива T, могат да бъдат графично изобразени с чертеж.

Често полученото решение е с необходимата точност, но броят на стъпките е твърде малък, за да се получи гладка линия на чертежа. Затова, по подразбиране, командата `ode45` извежда на всяка стъпка решението в четири точки: $t_k + 0.25h_k$, $t_k + 0.5h_k$, $t_k + 0.75h_k$, t_{k+1}. Това става с помощта на сплайн-интерполация, въз основа на стойностите $Y(t_k), Y'(t_k) = F(t_k, Y_k), Y(t_{k+1}), Y'(t_{k+1}) = F(t_{k+1}, Y_{k+1})$.

☞ **Пример 8.1** *Цилиндричен контейнер с радиус R е пълен с вода. Водата изтича през широк отвор с радиус r. Нека h е нивото на водата над центъра на отвора. Във всеки момент, h удовлетворява следното диференциално уравнение:*

$$h' = -C\sqrt{2gh}\left(\frac{r}{R}\right)^2,$$

където g е земното ускорение, а C е дебитен коефициент. За целите на упражнението, да предположим, че C е константа. Начертайте височината на течността като функция на времето, за t от 0 до 100 секунди. Параметрите са: $R = 1.0\,m$, $r = 0.1\,m$, $g = 9.81\,m/s^2$, $C = 0.4$. Началното условие е $h_0 = 1\,m$.

Решение: Създава се файл с разширение `m`, в който се задава дясната част на уравнението. Съдържанието на файла е:

ime1.m

```
1 function dh=ime1(t,h)
2 R=1;r=0.1;g=9.81;C=0.4;
3 dh=-C.*sqrt(2.*g.*h).*(r/R).^2;
```

След това, в Matlab се изпълняват командите:

```
>>[T,Y]=ode45('ime1',0,100,1);
>>plot(T,Y)
```

Ако е необходимо да бъде пресметнато решението на диференциалното уравнение за точно определена стойност на времето t, която липсва във вектора , например за $t = 31$, това може да стане чрез интерполация на база на получените стойности T и Y, с командата:

```
>>spline(T,Y,31)
```

В случая, лесно може да се намери точното решение на диференциалното уравнение. То е

$$h = \left(1 - C\sqrt{\frac{g}{2}}\left(\frac{r}{R}\right)^2 t\right)^2.$$

Сравняването на числено решение с аналитично получено може да стане с командите:

```
>>R=1;r=0.1;g=9.81;C=0.4;
>>h=(1-C*sqrt(g/2)*(r/R)^2.*T).^2;
>>plot(T,Y,'r',T,h,'b')
>>plot(T,Y-h)
>>max(abs(Y-h))
```

☞ **Пример 8.2** *Диференциалното уравнение*

$$Y'(t) = 2\left(1 - \frac{Y(t)}{3}\right)Y(t), \quad t \geq 0$$

$$Y(0) = Y_0$$

е основа на много модели на биологични популации. $Y(t)$ е броят на организмите в момента време t. За малки $Y(t)$, $1 - Y(t)/3 \approx 1$ и уравнението е $Y'(t) \approx 2Y(t)$. Тогава популацията ще нараства експоненциално, но приближи ли се до 3, $1 - Y(t)/3 \approx 0$ и растежът ще спре. Следователно популацията ще се доближи до горна граница, дължаща се на недостига на храна, например. Това уравнение се нарича логистично уравнение. Решете го с помощта на Matlab, при $Y_0 = 0.2$ за период от време $[0, 10]$.

Решение: Конструира се `m` - файл със следното съдържание:

ime2.m

```
1 function yp=ime2(t,y)
2 yp=2.*(1-y./3).*y;
```

В програмната среда Matlab се изпълняват командите:

```
>>[T,Y]=ode45('ime2',0,10,0.2);
>>plot(T,Y)
```

☞ **Пример 8.3** *При моделиране на екосистеми, за развитието на два конкуриращи се вида, се използват уравненията:*

$$\begin{aligned} u'(t) &= (2 - v(t))u(t), \quad u(0) = 3, \\ v'(t) &= (u(t) - 1)v(t), \quad v(0) = 2. \end{aligned}$$

Например, $u(t)$ е броят на зайците (в стотици), а $v(t)$ - броят на лисиците в дадена местност. Първото уравнение отразява факта, че без лисици броят на зайците би нараствал експоненциално. Второто уравнение сочи, че лисиците биха умрели, освен ако няма зайци, които да изяждат. Решете системата от уравнения за период от време $[0, 12]$.

Решение: Въвежда се означението

$$Y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} u(t) \\ v(t) \end{pmatrix}.$$

Тогава

$$y'(t) = \begin{pmatrix} y_1'(t) \\ y_2'(t) \end{pmatrix} = \begin{pmatrix} u'(t) \\ v'(t) \end{pmatrix} = \begin{pmatrix} (2 - v(t))u(t) \\ (u(t) - 1)v(t) \end{pmatrix} = \begin{pmatrix} (2 - y_2)y_1 \\ (y_1 - 1)y_2 \end{pmatrix} = F(t, Y(t)).$$

Създава се m-файл със следното съдържание:

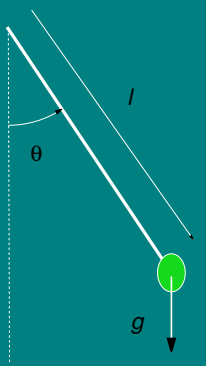
ime3.m

```
1 function yp=ime3(t,y)
2 yp=[(2-y(2)).*y(1);(y(1)-1).*y(2)];
```

От командния ред на Matlab се изпълняват командите:

```
>>[T,Y]=ode45('ime3',0,12,[3;2]);
>>plot(T,Y(:,1),'x',T,Y(:,2),'o')
>>title('Rabbits (x) and Foxes (o)')
>>plot(Y(:,1),Y(:,2))
>>xlabel('Rabbits'); ylabel('Foxes')
```

☞ **Пример 8.4** *Да разгледаме махалото, показано на Фигура 8.2. Нека ϑ е*



Фигура 8.2: Движение на махало

въглът, сключен с вертикалата в момента време t. В началния момент $\vartheta(0) = \pi/4$, $\vartheta'(0) = 0$. Движението на махалото се описва от диференциалното уравнение

$$\vartheta''(t) + gl \sin(\vartheta(t)) = 0,$$

където $g = 9.81\, m/s^2$, а $l = 0.5\, m$ е дължината на махалото. Решете уравнението в интервала $0 \le t \le 5$.

Решение: Диференциални уравнения, в които участват производни от втори и по-висок ред лесно могат да се сведат до системи уравнения от първи ред, с полагане. В случая, с полагането $\vartheta = y_1$, $\vartheta' = y_2$, уравнението може да се запише като система:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -gl \sin y_1 \end{aligned}$$

Това означава, че трябва да се създаде `m`– файл със следното съдържание:

ime4.m

```
1 function yp=ime4(t,y)
2 yp=[y(2);-9.81*0.5.*sin(y(1))];
```

От командния ред на Matlab се изпълняват командите:

```
>>[T,Y]=ode45('ime4',0,5,[pi/4;0]);
>>plot(T,Y(:,1))
```

8.6 Задачи за упражнение

Задача 8.1 *Решете следните диференциални уравнения с начални условия в интервала $[t_0, t_f]$ и изобразете решението.*

(а) $w' + (1.2 + \sin 10t)w = 0, \quad t_0 = 0, \quad t_f = 5, \quad w(t_o) = 1$

(б) $3w' + \dfrac{1}{1+t^2} w = \cos t, \quad t_0 = 0, \quad t_f = 5, \quad w(t_0) = 1$

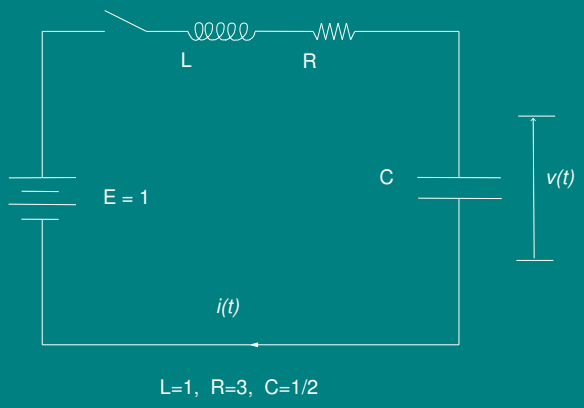
Задача 8.2 *Решете следните диференциални уравнения с начални условия в интервала $[t_0, t_f]$, като преди това ги сведете към система диференциални уравнения от първи ред. Изобразете решението графично.*

(а) $(1+t^2)w'' + 2tw' + 3w = 2, \quad t_0 = 0, \quad t_f = 5, \quad w(t_0) = 0, \quad w'(t_0) = 1$

(б) $w''' - 5\dfrac{2t}{(t+1)^2} w'' + w' + \dfrac{1}{3+\sin t} w = \cos t, \quad t_0 = 0, \quad t_f = 5,$
$w(t_0) = 1, \quad w'(t_0) = 0, \quad w''(t_0) = 2$

Задача 8.3 *Диференциалното уравнение от втори ред*

$$v'' + e(v^2 - 1)v' + v = 0,$$



Фигура 8.3: Електрическа верига

$$v(2) = 1,\ v'(2) = 0$$

е известно като уравнение на Ван дер Пол и описва напрежението в дадена електрическа верига, виж Фигура 8.3. За примера, да приемем, че $e = m/\sqrt{LC} - R\sqrt{C/L} \approx 1$. *Намерете решението в интервала от време* $[2, 10]$.

ГЛАВА 9

ЧИСЛЕНО РЕШАВАНЕ НА ЛИНЕЙНА ГРАНИЧНА ЗАДАЧА ОБИКНОВЕНИ ДИФЕРЕНЦИАЛНИ УРАВНЕНИЯ ОТ РЕД 2

Дадена е Линейна гранична задача от ред 2:

$$y'' = p(x)y' + q(x)y + r(x), \quad \text{за} \quad a \leq x \leq b \tag{9.1}$$

при гранични условия

$$y(a) = \alpha \quad , \quad y(b) = \beta. \tag{9.2}$$

9.1 Метод на стрелбата

При *Метода на стрелбата*, за апроксимиране на единственото решение на задачата (9.1) - (9.2), се разглеждат две начални задачи:

$$y'' = p(x)y' + q(x)y + r(x), \quad \text{за} \quad a \leq x \leq b \quad \text{при} \quad y(a) = \alpha \quad , \quad y'(a) = 0, \tag{9.3}$$

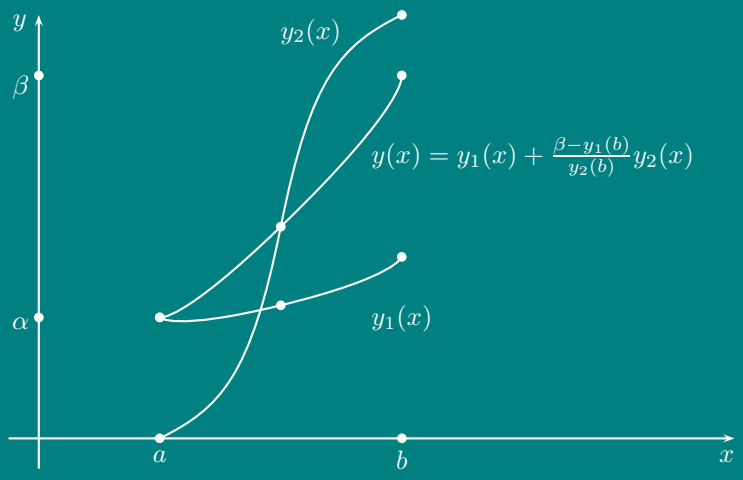
$$y'' = p(x)y' + q(x)y, \quad \text{за} \quad a \leq x \leq b \quad \text{при} \quad y(a) = 0 \quad , \quad y'(a) = 1. \tag{9.4}$$

Нека $y_1(x)$ е решение на (9.3), $y_2(x)$ - решение на (9.4). Тогава

$$y(x) = y_1(x) + \frac{\beta - y_1(b)}{y_2(b)} y_2(x) \tag{9.5}$$

е единственото решение на (9.1) - (9.2).

За апроксимиране на (9.3) и (9.4) се използват числени методи за решаване на задачата на Коши, [11, 13, 24, 26]. Графично методът е изобразен на Фигура 9.1.



Фигура 9.1: Метод на стрелбата

☞ **Пример 9.1** *Да се реши граничната задача*

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2} \ , \quad за \ \ 1 \le x \le 2 \ \ при \ \ y(1) = 1 \ \ , \ \ y(2) = 2. \qquad (9.6)$$

Да се сравнят стойностите на точното и приближеното решение, ако точното решение е:

$$y = c_1 x + \frac{c_2}{x^2} - \frac{3}{10}\sin(\ln x) - \frac{1}{10}\cos(\ln x),$$

където

$$c_2 = \frac{1}{70}\left[8 - 12\sin(\ln 2) - 4\cos(\ln 2)\right] \ , \ c_1 = \frac{11}{10} - c_2 \ .$$

Решение:

За прилагане на метода на стрелбата се решават приближено следните две начални задачи:

$$y_1'' = -\frac{2}{x}y_1' + \frac{2}{x^2}y_1 + \frac{\sin(\ln x)}{x^2} \ \ , \ 1 \le x \le 2 \ \ , \ \ y_1(1) = 1 \ \ , \ \ y_1'(1) = 0, \qquad (9.7)$$

и

$$y_2'' = -\frac{2}{x}y_2' + \frac{2}{x^2}y_2 \ \ , \ 1 \le x \le 2 \ \ , \ \ y_2(1) = 0 \ \ , \ \ y_2'(1) = 1 \ . \qquad (9.8)$$

За да се решат началните задачи (9.7) и (9.8), те се свеждат до системи диференциални уравнения от първи ред, които се решават в Matlab:

В началната задача (9.7) се полага $y_1' = u$ и $y_1 = v$, откъдето след диференциране се получава $y_1'' = u'$ и $y_1' = v'$. Получава се системата

$$\begin{aligned} u' &= -\frac{2}{x}u + \frac{2}{x^2}v + \frac{\sin(\ln x)}{x^2} \quad , \\ v' &= u \ , \end{aligned} \tag{9.9}$$

при начално условие

$$u(1) = 0 \quad , \quad v(1) = 1.$$

Във файл `fun1.m` се въвеждат десните страни на системата (9.9) като векторна функция `z`:

`fun1.m`

```
1 function yp=fun1(x,z)
2 u=z(1);v=z(2);
3 yp=[-2.*u./x+2.*v./(x.^2)+sin(log(x))./(x.^2);u];
```

В средата на Matlab се решава системата (9.9), като се въвеждат следните команди:

```
>>x0=1;xend=2;
>>z0=[0;1]   % начално условие
```

```
>>[x,z]=ode45('fun1',x0,xend,z0);
>>y1=z(:,2);
```

Вторият стълб на `z` е търсената функция y_1 в 11 точки от интервала $[1, 2]$; (първият стълб е стойността на y_1').

Аналогично, в началната задача (9.8) се полага $y_2' = u$ и $y_2 = v$, откъдето следва $y_2'' = u'$ и $y_2' = v'$. Получава се системата

$$\begin{aligned} u' &= -\frac{2}{x}u + \frac{2}{x^2}v \quad , \\ v' &= u \ , \end{aligned}$$

при начално условие

$$u(1) = 1 \quad , \quad v(1) = 0.$$

Във файл `fun2.m` се въвеждат десните страни на системата като векторна функция `t`:

`fun2.m`

```
1 function yp=fun2(x,t)
2 u=t(1);v=t(2);
3 yp=[-2.*u./x+2.*v./(x.^2);u];
```

В Matlab се въвеждат командите:

```
>>x0=1;xend=2;
>>t0=[1;0]  % начално условие
```

```
>>[x,t]=ode45('fun2',x0,xend,t0);
>>y2=t(:,2);
```

Вторият стълб на `t` е търсената функция y_2 в 11 точки от $[1,2]$.

От (9.5) за решението на граничната задача (9.1) имаме

$$y(x) = y_1(x) + \frac{2 - y_1(2)}{y_2(2)} y_2(x).$$

Тъй като знаем точното решение, можем да сравним съответните стойности на точното и приближеното решения:

```
>>n=length(x);c=(2-y1(n))/y2(n);
>>y=y1+c.*y2     % решение на ГЗ
>>plot(x,y,x,y1,'g',x,y2,'b'),grid
>>c2=(8-12*sin(log(2))-4*cos(log(2)))/70;c1=1.1-c2;
>>yt=c1.*x+c2./(x.^2)-0.3.*sin(log(x))-cos(log(x))./10;
>>plot(x,yt,x,y,'o')
>>plot(x,y-yt)
```

9.2 Мрежов метод за Линейна гранична задача от втори ред

Методът на стрелбата често води до големи грешки от закръгляване. *Мрежовите методи* са по-точни, но изискват повече пресмятания за достигане на необходимата точност.

Мрежовият метод за решаване на линейна гранична задача от втори ред

$$y'' = p(x)y' + q(x)y + r(x), \quad \text{за} \quad a \le x \le b \tag{9.10}$$

при гранични условия

$$y(a) = \alpha \quad , \quad y(b) = \beta. \tag{9.11}$$

изисква апроксимация на производните y' и y''.

Първо се избира естествено число N и се разделя интервала $[a,b]$ на $(N+1)$ равни подинтервали с краища точките $x_i = x_0 + ih$ за $i = 0, 1, \ldots, N, N+1$ и $h = (b-a)/(N+1)$.

Във всички вътрешни възли на мрежата x_i за $i = 1, 2, \ldots, N$, диференциалното уравнение се апроксимира с

$$y''(x_i) = p(x_i)y'(x_i) + q(x_i)y(x_i) + r(x_i). \tag{9.12}$$

Развива се функцията $y(x)$ в ред на Тейлър от трета степен около точката x_i, след което се пресмята за $x = x_{i+1}$, $x = x_{i-1}$, като се предполага, че $y \in \mathcal{C}^{(4)}[x_{i-1}, x_{i+1}]$,

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^+)$$

за някое $\xi_i^+ \in (x_i, x_{i+1})$ и

$$y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^-)$$

за някое $\xi_i^- \in (x_{i-1}, x_i)$.

След събиране на двете уравнения, имаме

$$y(x_{i+1}) + y(x_{i-1}) = 2y(x_i) + h^2y''(x_i) + \frac{h^4}{24}\left[y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)\right],$$

откъдето с прости алгебрични пресмятания се получава

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] - \frac{h^2}{24}\left[y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)\right].$$

Като се използва *Теоремата за средните стойности*, може да се опрости остатъчния член. Получава се формула за пресмятане на $y''(x_i)$, наречена **централна диференчна производна**:

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] - \frac{h^2}{12}y^{(4)}(\xi_i)$$

за някакво $\xi_i \in (x_{i-1}, x_{i+1})$.

По подобен начин се получава и *централна диференчна производна* $y'(x_i)$:

$$y'(x_i) = \frac{1}{2h}\left[y(x_{i+1}) - y(x_{i-1})\right] - \frac{h^2}{6}y^{(3)}(\eta_i) \quad , \quad \eta_i \in (x_{i-1}, x_{i+1}).$$

Като се използват диференчните производни за уравнението (9.12), се получава уравнението

$$\begin{aligned}\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} &= p(x_i)\left[\frac{y(x_{i+1}) - y(x_{i-1})}{2h}\right] + q(x_i)y(x_i) \\ &+ r(x_i) - \frac{h^2}{12}\left[2p(x_i)y^{(3)}(\eta_i) - y^{(4)}(\xi_i)\right].\end{aligned}$$

Мрежовият метод от порядък $\mathcal{O}(h^2)$, получен с използването на тези уравнения и граничните условия $y(a) = \alpha$, $y(b) = \beta$ води до системата:

$$w_0 = \alpha \quad , \quad w_{N+1} = \beta$$

$$\frac{2w_i - w_{i+1} - w_{i-1}}{h^2} + p(x_i)\frac{w_{i+1} - w_{i-1}}{2h} + q(x_i)w_i = -r(x_i), \text{за } i = 1, 2, \ldots, N.$$

Ние ще я разгледаме във вида

$$-\left(1 + \frac{h}{2}p(x_i)\right) w_{i-1} + \left(2 + h^2 q(x_i)\right) w_i - \left(1 - \frac{h}{2}p(x_i)\right) w_{i+1} = -h^2 r(x_i),$$

за $i = 1, 2, \ldots, N$, $w_0 = \alpha$, $w_{N+1} = \beta$.

Получената система е система линейни алгебрични уравнения $A\,w = b$ с три-диагонална матрица, където:

$$A = \begin{bmatrix} 2 + h^2 q(x_1) & -1 + \frac{h}{2}p(x_1) & 0 & \cdots & 0 \\ -1 - \frac{h}{2}p(x_2) & 2 + h^2 q(x_2) & -1 + \frac{h}{2}p(x_2) & & \\ & \ddots & \ddots & \ddots & \\ 0 & & & & -1 + \frac{h}{2}p(x_{N-1}) \\ & & 0 & -1 - \frac{h}{2}p(x_N) & 2 + h^2 q(x_N) \end{bmatrix},$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N-1} \\ w_N \end{bmatrix}, \quad b = \begin{bmatrix} -h^2 r(x_1) + \left(1 + \frac{h}{2}p(x_1)\right) w_0 \\ -h^2 r(x_2) \\ \vdots \\ -h^2 r(x_{N-1}) \\ -h^2 r(x_N) + \left(1 - \frac{h}{2}p(x_N)\right) w_{N+1} \end{bmatrix}, \quad w_0 = \alpha\ ,\ w_{N+1} = \beta.$$

Тази система има единствено решение при предположение, че p, q, r са непрекъснати върху $[a, b]$ и $q(x) > 0$ за $x \in [a, b]$, и че $h < 2/K$, където

$$K = max_{a \le x \le b} \mid p(x) \mid .$$

За решаването на системата

$$A\,w = b \tag{9.13}$$

с три-диагонална матрица A се използва **LU**-факторизация на Крут, (виж стр. 15). Системата (9.13) се преобразува във вида

$$LUw = b,$$

където L е долна триъгълна матрица (с нули под главния диагонал), а U - горна триъгълна матрица.

Полага се $v = Uw$. Първо се решава системата $Lv = b$, откъдето се намира вектора v, след което се решава системата $Uw = v$, откъдето се получава решението на системата (9.13).

В Matlab командата `lu(A)` представя квадратната матрица A като произведение на две триъгълни матрици, едната от които е пермутация на долна триъгълна матрица, а другата е горна триъгълна матрица. Използваният алгоритъм е Гаусова елиминация.

`[L,U]=lu(A)` дава горна триъгълна матрица в U и произведение от долна триъгълна и пермутационна матрица в L, такива, че `A=L*U`.

`[L,U,P]=lu(A)` дава горна триъгълна матрица U, долна триъгълна матрица L и пермутационна матрица P, така, че `L*U=P*A`.

- В случай, че матрицата P не е единична, за да се реши матричното уравнение `Ax=b`, двете му страни се умножават с P, т.е. `(P*A)x=P*b` $\Leftrightarrow$ `LUx=b1`, където `P*b=b1`.

  Полага се `y=Ux`, решава се уравнението `Ly=b1` с командата `y=L\b1`, след което се решава и уравнението `Ux=y` с командата `x=U\y`.

- Ако матрицата `P=eye(A)`, за да се реши уравнението `Ax=b`, т.е. `LUx=b`, се полага `y=Ux`; решава се уравнението `Ly=b` с командата `y=L\b`, след което от уравнението `Ux=y` се намира `x=U\y`.

☞ **Пример 9.2** *С използване на Линеен мрежов метод да се апроксимира решението на граничната задача от втори ред*

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2} \quad , \quad за \;\; 1 \le x \le 2 \;\; при \;\; y(1) = 1 \quad , \quad y(2) = 2. \qquad (9.14)$$

Решение:  

В три файла се дефинират функциите $p(x)$, $q(x)$ и $r(x)$:

p.m

```
1 function y=p(x)
2 y=-2.*(x.^(-1));
```

q.m

```
function y=q(x)
y=2.*(x.^(-2));
```

r.m

```
function y=r(x)
y=sin(log(x)).*(x.^(-2));
```

Формира се матрицата A, като интервалът $[1,2]$ се дели на 10 подинтервали. От командния ред се въвежда следното:

```
>>clear x
>>x0=1;xend=2;alpha=1;beta=2;N=9;
>>h=(xend-x0)/(N+1);h2=h/2;hkw=h*h;
>>for i=1:N
      x(i)=x0+i*h;
  end
>> % формиране на три-диагонална матрица
>>A=zeros(N);b=zeros(1,N);
>>for i=1:N
      Q(i)=2+hkw*q(x(i));
      PL(i)=-1-h2*p(x(i));
      PU(i)=-1+h2*p(x(i));
      R(i)=-hkw*r(x(i));
  end
>>A=diag(PL(2:N),-1)+diag(Q)+diag(PU(1:N-1),1);
>>b(1,1)=-PL(1)*alpha;b(1,N)=-PU(N)*beta;
>>b=b+R;b=b';
>> % Решаване на системата Aw=b
>>[L,U,P]=lu(A)
>>if P-eye(N) == zeros(N)
     v=L\b;w=U\v
  else
     b1=P*b;v=L\b1;w=U\v
  end
>>xx=[1;x';2];ww=[alpha;w;beta];
>>plot(xx,ww)
>> % Сравняване с точното решение
>>c2=(8-12*sin(log(2))-4*cos(log(2)))/70;c1=1.1-c2;
>>y=c1.*xx+c2./(xx.*xx)-3.*sin(log(xx))./10;
>>plot(xx,ww,'r',xx,y,'b')
>>plot(xx,ww-y)
```

9.3 Задачи за упражнение

✎ **Задача 9.1** *Граничната задача*

$$y'' = y' + 2y + \cos x \quad за \quad 0 \le x \le \frac{\pi}{2} \quad при \quad y(0) = -0.3 \quad , \quad y\left(\frac{\pi}{2}\right) = 0.1$$

има точно решение

$$y(x) = -\frac{1}{10}(\sin x + 3\cos x) \ .$$

Използвайки Метода на стрелбата, апроксимирайте решението и сравнете резултата с точното решение със стъпка $h = \pi/6$.

✎ **Задача 9.2** *Да се решат граничните задачи с Метод на стрелбата и да се сравни решението с точното решение.*

(а) $y'' = -\frac{4}{x}y' - \frac{2}{x^2}y - \frac{2\ln x}{x^2}, \quad за\ 0 \le x \le 2$

$при\ y(1) = \frac{1}{2}, \quad y(2) = \ln 2, \quad h = 0.05,$

$Точно\ решение: \quad y(x) = \frac{4}{x} - \frac{2}{x^2} + \ln x - \frac{3}{2};$

(б) $y'' = -4y + \cos x, \quad за\ 0 \le x \le \pi/4$

$при\ y(0) = 0, \quad y(\pi/4) = 0, \quad h = \frac{\pi}{20},$

$Точно\ решение: \quad y(x) = -\frac{1}{3}\cos 2x - \frac{\sqrt{2}}{6}\sin 2x + \frac{1}{3}\cos x;$

(в) $y'' = 2y' - y + xe^x - x, \quad за\ 0 \le x \le 2$

$при\ y(0) = 0, \quad y(2) = -4, \quad h = 0.2,$

$Точно\ решение: \quad y(x) = \frac{1}{6}x^3e^x - \frac{5}{3}xe^x + 2e^x - x - 2.$

✎ **Задача 9.3** *Нека* u *е електростатичният потенциал между две концентрични метални сфери с радиуси* r_1 *и* r_2 $(r_1 < r_2)$*, такъв, че потенциалът на вътрешната сфера е постоянен –* V_1 *волта, а потенциалът на външната сфера е* 0 *волта. Потенциалът в областта между двете сфери се подчинява на уравнението на Лаплас, което за този пример е зададено във вида:*

$$\frac{d^2u}{dr^2} + \frac{2}{r}\frac{du}{dr} = 0, \quad за \quad r_1 \le r \le r_2 \quad при \quad u(r_1) = V_1 \ , \ u(r_2) = 0.$$

Нека $r_1 = 4\,см$, $r_2 = 8\,см$, $V_1 = 110\,V$.

а). *Апроксимирайте* $u(6)$ *като използвате Линеен метод на стрелбата;*

б). *Сравнете резултата от (а) с точната стойност за потенциала* $u(6)$, *където:*

$$u(r) = \frac{V_1 r_1}{r}\left(\frac{r_2 - r}{r_2 - r_1}\right) .$$

✎ **Задача 9.4** *Граничната задача*

$$y'' + y = 0 \quad за \quad 0 \le x \le \frac{\pi}{4} \quad при \quad y(0) = 1 \quad , \quad y\left(\frac{\pi}{4}\right) = 1$$

има точно решение

$$y(x) = \cos x + (\sqrt{2} - 1)\sin x \ .$$

Използвайки Линеен мрежов метод, апроксимирайте решението и сравнете резултата с точното решение със стъпка $h = \pi/20$.

✎ **Задача 9.5** *Правоъгълна плоча е натоварена с равномерно разпределен напречен товар под действието на мембранна сила. Провисването на плочата се описва с диференциално уравнение от втори ред. Нека* F *е мембранната сила, а* q *– интензитетът на напречното натоварване. Провисването* w *по дължината на плочата се дава с диференциалното уравнение:*

$$w''(x) - \frac{F}{D}w(x) = -\frac{ql}{2D}x + \frac{q}{2D}x^2 \quad за \quad 0 \le x \le l \quad при \quad w(0) = w(l) = 0,$$

където l *е дължината на плочата и* D *– коравина на огъване на плочата.*

Нека $q = 200\,N/m$, $F = 100\,N/m$, $D = 60.6 \times 10^7\,N/m$, $l = 1\,m$. *Да се апроксимира провисването през интервал от* $0.2\,m$ *с използване на Линеен мрежов метод за гранична задача от втори ред.*

* *

ГЛАВА 10

МАТЕМАТИЧЕСКА СТАТИСТИКА

10.1 Статистическа обработка на извадки с малък обем

☞ **Пример 10.1** *За оценка на качеството на изделията, произвеждани от автомат, се прави извадка с обем* $n = 10$*, като се измерват дължините на изделията. Случайната величина* X *е дължината на изделие, произвеждано от автомата.*

а). *Да се намери вариационният ред на извадката;*

б). *Да се построи Таблица на статистическото разпределение на извадката;*

в). *Да се начертае графиката на статистическата функция на разпределение на извадката.*

Решение: Резултатите се нанасят в Таблица 10.1.

```
>>x=[9.3,10.8,9.3,10.8,11.3,8.1,10.8,10.8,11.3,10.8]; %извадка
>>broj=length(x) %обем на извадката
>>y=sort(x)      %вариационен ред на извадката
.........................................
>>%Построяване на статистическия ред
>>xx=y(1);
>>for i=2:broj
      if y(i)~=y(i-1)
         xx=[xx,y(i)];
      end
  end
```

```
>>xx %елементи без повторения
.............................
>>k=length(xx)   %брой елементи без повторения
..............
>>%абсолютни честоти
>>n=zeros(1,k);
>>for i=1:k,for j=1:broj
      if y(j)==xx(i)
         n(i)=n(i)+1;
      end
  end,end
>>n % абсолютни честоти за xx
>>w=n/broj %относителни честоти
>>wstar=cumsum(w) %сумарни относителни честоти
>>%Графика на статистическата функция на разпределение
>>stairs(xx,wstar)
```

Елементи без повторения				
Абсолютни честоти				
Относителни честоти				
Сумарни относителни честоти				

Таблица 10.1: Таблица на статистическото разпределение на извадката

10.2 Статистическа обработка на извадки с голям обем

При голям обем n на извадката, елементите ѝ се обединяват в групи, като резултатите се представят във вид на групиран статистически ред. Интервалът, съдържащ всички елементи на извадката $[x_{min}, x_{max}]$, се разделя на m равни части, като m се пресмята по формулата

$$m \approx 1 + 3.2 \, lg \, n \ .$$

Пресмята се дължината на интервала $d = \frac{\omega}{m}$, където $\omega = x_{max} - x_{min}$ е *размаха* на извадката, а m - брой на под-интервалите. Получава се статистически ред, за който се определят *абсолютните честоти* n_i^* за всеки подинтервал, *относителните честоти* $w_i^* = \frac{n_i^*}{n}$ и *сумарните относителни честоти*

$$\sum_{j=1}^{n} n_j^*/n.$$

Получените данни се записват в *Таблица на групираните данни*, след което се изобразяват графично чрез *полигон* или *хистограм*.

☞ **Пример 10.2** *В Таблица 10.2 са дадени данни за случайната величина X - дължина на детайл, произвеждан от автомат, с нормален закон на разпределение. Да се групират данните при $n = 50$.*

а). *Да се намери средната стойност $\overline{X}$ и статистическата дисперсия s_n^2 на извадката;*

б). *Да се начертае хистограма на емпиричното разпределение;*

в). *Да се намерят доверителните интервали, гарантирани с вероятност $\gamma = 0.9$ за математическото очакване $E[X]$ и за дисперсията $\sigma^2[X]$ на генералната съвкупност;*

г). *Ако на всеки час, в продължение на 5 часа, от продукцията на автомата се прави по една извадка с обем $n = 5$, да се направят контролни карти на математическото очакване и дисперсията, ако е известно, че съответните им доверителни интервали, са получените от в).*

№	Дължина на изделие									
1	10.00	11.45	10.50	11.51	11.06	11.14	11.73	11.06	10.46	11.19
2	11.70	11.95	12.27	11.43	10.41	12.75	11.70	11.86	10.03	11.99
3	10.72	11.35	12.02	10.62	11.11	10.62	12.39	11.22	11.21	12.20
4	12.28	11.45	12.70	11.96	12.60	11.14	9.88	10.67	9.94	11.28
5	11.60	7.99	11.97	10.75	11.00	11.78	11.10	12.46	11.30	11.82
6	11.77	10.27	10.68	10.65	12.36	10.88	9.75	11.50	9.30	12.10
7	10.90	11.82	12.83	11.68	10.04	10.20	13.20	10.80	10.75	9.69
8	11.05	9.33	10.52	11.85	11.06	12.70	10.25	12.87	12.28	10.08
9	10.30	10.28	9.21	12.06	10.41	11.53	8.94	10.40	10.56	10.75
10	8.83	11.56	12.72	11.30	10.43	10.63	10.15	10.11	10.83	10.03
11	9.36	10.77	12.40	12.03	12.23	12.73	10.28	12.00	11.06	11.40
12	11.49	10.25	10.32	11.87	10.76	12.61	12.44	11.13	9.90	10.26
13	10.23	10.73	12.39	12.61	10.26	12.19	13.12	12.00	11.04	10.41
14	8.88	12.95	11.28	12.29	12.75	10.46	11.39	10.65	11.55	11.60
15	10.93	10.01	12.70	11.78	11.80	9.62	9.53	12.00	11.43	12.17
16	12.00	11.28	11.51	10.80	11.00	9.82	11.09	10.63	10.50	11.05
17	13.65	11.95	11.70	10.55	10.48	12.75	13.39	11.85	10.68	10.05
18	10.65	11.17	12.09	11.10	10.02	10.30	9.36	10.93	11.54	11.47
19	12.20	11.89	11.57	12.30	10.51	10.96	11.00	12.25	11.38	9.03

Продължава ...

№	Дължина на изделие									
20	11.45	11.83	11.65	13.69	12.10	9.30	8.60	11.34	12.25	11.38
21	13.32	12.77	12.15	10.80	10.81	10.80	11.00	11.26	10.45	13.03
22	11.80	11.40	9.78	12.19	10.11	10.95	12.80	10.96	10.52	11.59
23	9.98	12.11	10.97	11.50	10.75	10.00	12.34	12.10	10.85	12.10
24	12.30	12.71	10.85	10.39	11.78	10.65	10.71	10.01	9.25	10.41
25	11.31	11.17	11.62	12.03	11.59	10.81	11.87	11.10	11.27	10.01
26	13.28	11.32	11.12	11.76	10.08	11.30	10.65	9.24	10.85	11.11
27	10.30	12.05	11.32	10.88	10.57	11.33	12.36	11.40	11.65	11.70
28	12.41	11.66	12.14	10.07	12.61	10.65	11.10	12.85	13.09	11.20
29	12.23	10.97	11.21	10.75	9.82	10.50	10.50	9.96	10.79	10.73
30	12.78	12.45	10.95	11.15	9.16	11.04	10.61	8.10	11.00	9.66
31	11.53	11.63	11.49	12.75	10.44	11.54	10.08	10.15	10.29	10.60
32	11.58	10.83	10.52	10.17	10.92	11.77	12.70	10.24	10.25	10.14
33	10.85	12.39	12.10	10.63	10.97	10.67	10.10	10.30	12.60	11.36
34	11.27	10.55	10.77	11.10	10.66	10.95	9.85	11.26	10.98	11.06
35	10.40	12.40	10.97	11.40	9.71	9.47	9.84	10.10	9.62	10.53
36	10.68	8.80	10.50	10.50	12.75	11.05	10.38	11.88	10.30	10.65
37	9.63	12.06	9.50	10.75	13.03	9.78	12.15	9.36	11.70	10.08
38	10.68	11.30	10.97	11.95	12.46	10.71	10.42	12.00	10.55	11.00
39	10.32	11.00	10.32	9.60	10.93	9.32	8.33	9.91	12.88	10.28
40	9.55	10.74	11.69	10.50	10.79	9.62	10.34	11.83	10.65	12.00
41	11.63	9.32	11.15	12.30	10.51	10.54	11.77	10.39	12.16	11.38
42	10.50	11.38	10.96	11.13	13.69	12.40	10.39	10.00	10.64	11.25
43	8.26	10.80	10.46	11.52	9.91	12.02	12.10	12.83	11.65	11.05
44	13.62	10.84	11.45	10.39	12.02	10.04	9.98	9.60	8.39	11.10
45	11.31	11.70	9.73	11.38	12.12	10.80	10.80	11.74	9.97	11.00
46	11.44	10.60	11.50	9.75	11.06	10.58	10.66	10.51	12.07	13.43
47	12.70	12.80	11.73	12.45	10.55	11.25	11.71	11.11	10.00	9.94
48	10.05	11.19	11.94	12.98	10.90	11.10	10.78	13.25	11.63	9.29
49	12.32	11.56	12.11	9.11	11.51	9.64	8.67	11.19	12.08	10.68
50	12.41	11.45	11.70	11.08	10.28	10.60	10.36	11.18	10.95	10.90

Таблица 10.2: Данни за дължината на изделия, произвеждани от автомат

Решение:  

а), б). Във файл `stat.m` се въвеждат 50 данни (една колона на Таблица 10.2:

`stat.m`

```
%50 данни по статистика
stat=[...
...
...];
```

От командния ред на Matlab се въвеждат командите:

```
>>help stat
>>stat;
>>n=length(stat);
```

```
>>mean(stat) %средна стойност
......
>>std(stat) %коригирано средно-квадратично отклонение
......
>>std(stat)^2 %коригирана дисперсия
......
>>min(stat),max(stat)
.....................
>>y=sort(stat); %вариационен ред
>>m=fix(1+3.2*log(n)/log(10)) %брой групи
......
>>omega=y(n)-y(1)  %размах на извадката
......
>>d=omega/m %дължина на интервала
......
>>i=0:m;xr=y(1)+i*d %граници на интервалите
>>%попълнете Таблица 3
>>[nstar,xstar]=hist(y,m) %nstar-абсолютни честоти; xstar-среди
>>wstar=nstar/n %относителни честоти
>>G=cumsum(wstar) %сумарни относителни честоти
>>F=wstar/d;
>>bar(xstar,F) %чертае хистограм
>>stairs(xr(:,1:m),G) %групирана стат. функция на разпределение
```

№	Граници на интервала $[x_i^*, x_{i+1}^*]$	Среди на интервала x_i^*	Относителна честота w_i^*	Сумарна относителна честота
1				
2				
3				
4				
5				
6				

Таблица 10.3: Таблица на групираните данни

в). Доверителният интервал за средната стойност на случайната величина X, с нормален закон на разпределение, гарантиран с вероятност γ, се задава с формулата

$$I_{\gamma;n}^{E[X]} = \left(\overline{X}_n - \delta_{\gamma;n}\ ;\ \overline{X}_n + \delta_{\gamma;n}\right) \quad , \quad \delta_{\gamma;n} = t_{\gamma;n-1} \cdot \frac{s_n}{\sqrt{n}} \ ,$$

където:

- $\overline{X}_n$ – статистическа средна стойност на случайната величина, получена за извадка с обем n;
- s_n – коригирано средно-квадратично отклонение, получено за извадка с обем n;
- $t_{\gamma;n-1}$ – квантил от ред γ на t-разпределение на Стюдънт, с $(n-1)$ степени на свобода при двустранно ограничение.

Максималната относителна грешка на интервалната оценка (доверителния интервал) за $E[X]$, гарантиран с вероятност γ е

$$\Delta_{\gamma;n} = \frac{\delta_{\gamma;n}}{\overline{X}_n} \cdot 100\% \ .$$

Пресмята се $\Delta_{\gamma;n}$:

$$\Delta_{\gamma;n} = \Delta_{0.9;50} = t_{0.9;49} \cdot \frac{s_n}{\sqrt{n}} = 1.6775 \cdot \frac{s_n}{\sqrt{n}}$$

```
>>delta=1.6775*std(stat)/sqrt(50)
>>left_IE=mean(stat)-delta
>>right_IE=mean(stat)+delta
>>Delta=delta*100/mean(stat) %попълнете
```

$$I^{E[X]}_{0.9;50} = (\qquad ; \qquad) \ , \quad \Delta_{0.9;50} = \qquad \%.$$

Доверителният интервал за дисперсията на случайната величина X, с нормален закон на разпределение, гарантиран с вероятност γ, се задава с формулата

$$I^{\sigma^2[X]}_{\gamma;n} = \left(\frac{(n-1)s_n^2}{\chi^2_{\frac{1-\gamma}{2};n-1}} \ ; \ \frac{(n-1)s_n^2}{\chi^2_{\frac{1+\gamma}{2};n-1}} \right) \ ,$$

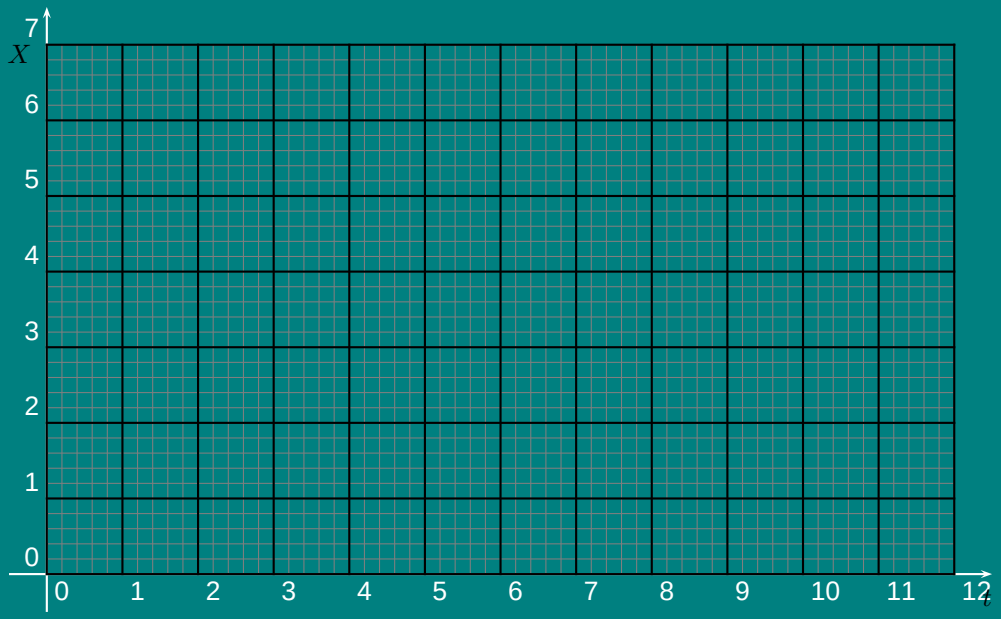
където $\chi^2_{\frac{1-\gamma}{2};n-1}$ и $\chi^2_{\frac{1+\gamma}{2};n-1}$ са критични точки на χ^2-разпределението с $n-1$ степени на свобода.

От Таблицата на χ^2-разпределението се определят

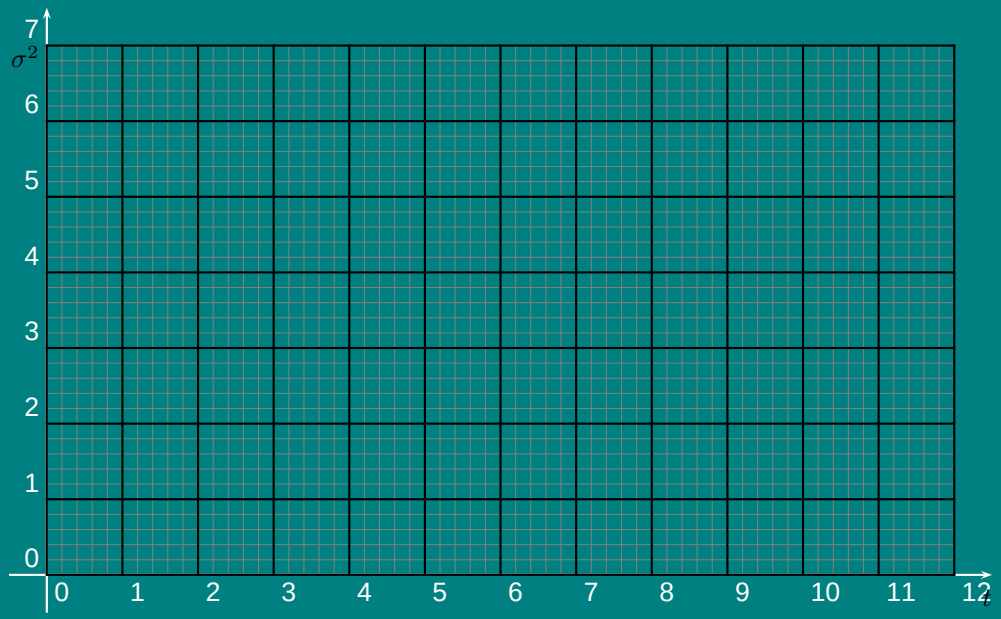
$$\chi^2_{\frac{1-\gamma}{2};n-1} = \chi^2_{\frac{1-0.9}{2};49} = \chi^2_{0.05;49} \approx 67.5 \ ,$$

$$\chi^2_{\frac{1+\gamma}{2};n-1} = \chi^2_{\frac{1+0.9}{2};49} = \chi^2_{0.95;49} \approx 34.8 \ .$$

```
>>left_ID=(n-1)*(std(stat)^2)/67.5
>>right_ID=(n-1)*(std(stat)^2)/34.8 %попълнете
```



Фигура 10.1: Контролна карта за математическото очакване



Фигура 10.2: Контролна карта за дисперсията

$$I_{0.9;50}^{\sigma^2[X]} = (\qquad ; \qquad),$$

г). На всеки час по случаен начин се избират 5 детайла от продукцията на автомата (извадка с обем $n = 5$) и за нея се пресмятат средната стойност и дисперсията. За първата и втората извадки този избор и пресмятане в Matlab се извършва с командите

```
>>t1=round(49*rand(1,5)+1) %5 случайни цели числа от 1 до 50
>>i=1:5;x1(i)=stat(t1(i))  %5 случайни елементи от stat
>>mean(x1)                 %средна стойност при n=5
>>std(x1)^2                %дисперсия при n=5
>>t2=round(49*rand(1,5)+1) %нови 5 случайни числа от 1 до 50
>>i=1:5;x2(i)=stat(t2(i))  %нови 5 случайни елементи от stat
>>mean(x2)
>>std(x2)^2                %аналогично за останалите 3 извадки
>>%попълнете Таблица 4
```

№	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
	$X1$	$X2$	$X3$	$X4$	$X5$
1					
2					
3					
4					
5					
$\overline{X}_n$					
s_n^2					

Таблица 10.4: Данни за контролни карти

✶ ✶

БИБЛИОГРАФИЯ

[1] Дьяконов, В. *MATLAB*6 *Учебны курс*, „Питер“, Санкт Петербург, 2001.

[2] Йорданов, Й. *Приложение на MATLAB в инженерните изследвания*, Университетско издателство „А.Кънчев“, Русе, 2004.

[3] *Консультационный центр MATLAB компании SoftLine*, Москва. http://www.matlab.ru

[4] Лазарев, Ю. *MATLAB* 5.*x*, „Ирина“, BHV, Киев, 2000.

[5] *Образовательный математический сайт* Exponenta.ru, Москва. http://www.exponenta.ru/

[6] Потемкин, В.Г. *Система инженерных и научных расчетов MATLAB* 5.x, *Том 1,2*, „Диалог-Мифи“, Москва, 1999.

[7] Приклонский, В.И., МГУ, Москва, 2003. http://fiziki.uniyar.ac.ru/educate/lectures/digital.html

[8] Auzinger, W., Fabianek G., Holy, P. , Pawlik, S. *An Introduction to Matlab*, Austria, 2002. http://fsmat.at/~cfabiane/docs/matlab.pdf

[9] Beardah, C. *Matlab 6 and Student Version of Matlab*, UK, 2001. http://ltsn.mathstore.at.uk/newsletter/nov2001/pdf/matlab6_applet4teach.pdf

[10] Biran, A., M. Breiner. *MATLAB for engineers.* Addison-Wesley Publishing Company, USA, 1995.

[11] Chapra, S., Canale,R. *Numerical methods for Engineers* 2nd ed. , McGraw-Hill Book Company, USA, 1988.

[12] Driscoll,T. *Crash course in MATLAB*, USA, 2002.
http://www.math.udel.edu/~driscoll/teaching/matlab_adv.pdf

[13] Faires,J.Douglas, Burden,R. *Numerical Methods*, 2nd ed., Brooks/Cole Publishing Company, USA, 1998.

[14] *Getting started with MATLAB, Version 6*, The MathWorks Company, USA, 2002. http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

[15] *Matlab Links*, Tallinn, 2003.
http://staff.ttu.ee/~alahe/aMatlab.html

[16] He, J. *Matlab Toolbox Quick Reference*, USA.
http://tiger.la.asu.edu/Quick_Ref/matlab_toolbox_quickref.pdf

[17] *ITG: Research Computing: Matlab*, USA.
http://www.som.yale.edu/itg/research_unix/matlab.asp

[18] *Matlab Assignments at Ohio University*,
http://www.math.ohiou.edu/courses/matlab/

[19] *Matlab Reference guide.* The Math Works Ink., 1992.

[20] *Matlab Tips and Tricks and...*, USA.
http://www.ee.columbia.edu/~marios/matlab/matlab_tricks.html

[21] *Matlab Tutorial*, USA, 2003.
http://www.cs.rpi.edu/~magdon/courses/csci6964_fall2003/notes/matlab.pdf

[22] *Online Manuals (in PDF)*, USA.
http://www-ccs.ucsd.edu/matlab/fuldocset.html

[23] *Online Matlab Tutorials*, USA.
http://spicerack.sr.unh.edu/~mathadm/tutorial/software/matlab

[24] Recktenwald, G. *Numerical Methods with Matlab*, USA, 2003.
http://www.me.pdx.edu/~gerry/nmm/course/

[25] *Using MATLAB, Version 6*, The MathWorks Company, USA, 2002.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

[26] *Web-Based Resources for a Course in Numerical Methods*, USA, 2002.
http://numericalmethods.eng.usf.edu/resource_text.html

[27] Xenophontos, C. *A Beginer's Guide to MATLAB*, USA, 1999.
http://www.clarkson.edu/classs/ma339/Xeno_MATLAB_guide.pdf

Стефка Караколева

Евелина Велева

ВИСША МАТЕМАТИКА 3

Практикум по „Числени методи“

с *MATLAB*

ISBN 954-712-245-2
Първо издание
Формат: 70/100/16
Печатни коли: 14
Издателски коли: 7
Тираж: 500

Печатна база при Русенски Университет „Ангел Кънчев“

Русе, 2004