

Prelucrarea datelor fizice și metode numerice

Lect. dr. Ioan Dumitru



Obiectivele cursului

- Obiectiv general

- Cunoașterea problematicii calculului științific modern și utilizarea produselor software pentru prelucrarea datelor fizice. Utilizarea algoritmilor numerici pentru rezolvarea unor probleme de fizica.

- Obiectivele specifice

- Sa descrie algoritmii folositi pentru metodele de calcul numeric
 - Sa transfere algoritmii de calcul în limbaj de programare
 - Sa căute, sa prelucre și sa analizeze informații din diverse biblioteci de programe, pentru rezolvarea unor probleme numerice date
 - Sa formuleze critici cu privire la utilitatea unei secvențe de program și sa aprecieze erorile ce pot interveni
 - Sa asambleze metode numerice într-un program de simulare a unui proces sau fenomen fizic.



- Introducere in metode numerice. Elemente de C specifice metodelor numerice
- Reprezentarea numerelor și precizia numerică. Cifre semnificative și cifre exacte ale unui număr. Erori in calculul numeric.
- Rezolvarea numerică a ecuațiilor.
- Elemente de algebră liniară. Operații cu matrici și calculul determinanților. Sisteme de ecuații liniare. Ecuații neliniare și rădăcinile polinoamelor. Metode iterative.
- Aproximarea funcțiilor de o variabilă reală. Interpolarea polinomială, spline.
- Fitarea datelor experimentale. Aproximarea în sensul celor mai mici pătrate. Minimizarea funcțiilor.
- Derivarea si integrarea numerică.
- Rezolvarea ecuațiilor diferențiale. Rezolvarea ecuațiilor diferențiale cu derivate parțiale. Rezolvarea sistemelor de ecuații diferențiale.
- Vectori și valori proprii.
- Elemente de statistică
- Utilizarea librariilor numerice in calculul numeric



Bibliografie

- Ioan Dumitru, Curs Metode numerice - platforma www.phys.uaic.ro
- Catalin AGHEORGHIESEI - Prelucrarea datelor experimentale in fizica si calcul numeric
- Adrin BRADU - Analiza Numerica - exercitii si probleme, Editura UAIC
- C. Berbente, S. Mitran, S. Zancu, Metode Numerice, Editura Tehnica, 1997
- Numerical Recipes in C. The Art of Scientific Computing, 2nd Edition, 1992
- Titus Adrian Beu, Calcul numeric în C, Microinformatica, Cluj, 2000
- Alexandru LUPAS, Metode Numerice, Editura Constant Sibiu, 2001
- Alejandro L. Garcia, Numerical Methods for Physics (Prentice Hall, Englewood Cliffs NJ, 1994)
- J.M. Thijssen. Computational Physics. Springer Verlag, 1999.
- GNU Scientific Library - http://www.gnu.org/software/gsl/manual/html_node/
- Mahinder Kumar - Numerical methods: problems and solutions



- Limbaje de programare

- C, C++,C#
- Fortran
- Java

- Biblioteci numerice

- GNU Scientific Library (GSL)
- LAPACK
- ALGLIB
- NAG Library
- BLAS, LAPACK



Prelucrarea datelor fizice și metode numerice

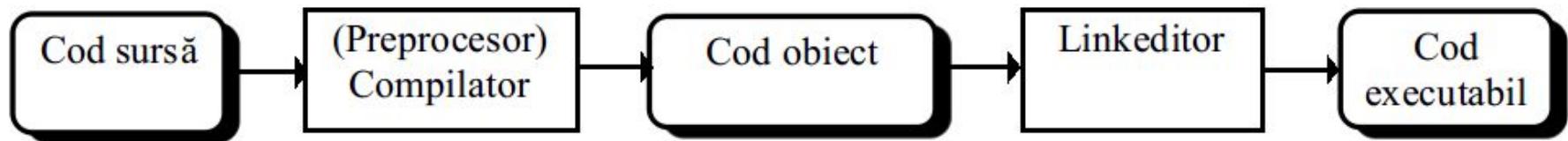
Lect. dr. Ioan Dumitru



Implementarea unui algoritm într-un limbaj de programare

- Sărirea (editarea) programului sursă.
- Compilarea
- Linkeditarea
- Execuția

Etapele necesare obținerii fișierului executabil



● Variabile

- sunt date (obiecte informaționale) ale căror valori se pot modifica în timpul execuției programului
- constau în litere și numere în orice ordine, primul caracter să fie o literă
- ex: x, c14, area, electron_mass, TEMPERATURE

● Declararea variabilelor

- Modul general de declarare a variabilelor este:

```
tip_variabila      listă_nume_variabile;
```

- Într-un program în limbajul C++, declaratiile de variabile pot apărea în orice loc în programul sursă. La declararea variabilelor, se rezervă în memorie un număr de octeți corespunzător tipului variabilei, urmând ca ulterior, în acea zonă de memorie, să fie depusă (memorată, înregistrată) o anumită valoare

- În calculul științific tipic utilizăm variabile integer, float, double

- Ex:

```
int a, b, c;  
double acc, epsilon, t;
```

● Inițializarea variabilelor în declarații

```
int a = 3, b = 5;  
double factor = 1.2e-5;
```

● Declarații multiple

```
int i = j = k = 4
```



- Tipurile de bază sunt:

- **char** un singur octet (1 byte=8 biți), capabil să conțină codul unui caracter din setul local de caractere;
- **int** număr întreg, reflectă în mod tipic mărimea naturală din calculatorul utilizat;
- **float** număr real, în virgulă mobilă, simplă precizie;
- **double** număr real, în virgulă mobilă, dublă precizie.

- În completare există un număr de calificatori, care se pot aplica tipurilor de bază: **short**, **long**, **signed** și **unsigned**.

Tip	Lungimea zonei de memorie ocupate (în biți)		Descriere
	MS-DOS	UNIX-LINUX	
char	8	8	Valoarea unui singur caracter; poate fi întâlnită în expresii cu extensie de semn
unsigned char	8	8	Aceeași ca la char, fără extensie de semn
signed char	8	8	Aceeași ca la char, cu extensie de semn obligatorie
int	16	32	Valoare întreagă
long	32	64	Valoare întreagă cu precizie mare
(long int)			
long long int	32	64	Valoare întreagă cu precizie mare
short int	16	32	Valoare întreagă cu precizie mică
unsigned int	16	32	Valoare întreagă, fără semn
unsigned long int	32	64	Valoare întreagă, fără semn
float	32	32	Valoare numerică cu zecimale, simplă precizie (6)
double	64	64	Valoare numerică cu zecimale, dublă precizie (10)
long double	80	128	Valoare numerică cu zecimale, dublă precizie



Operatori

- Stabilesc legaturile intre variabile si expresii
- Operatori aritmetici unari:

Operator	Semnificație		Exemple
-	Minus unar		-a
++	Operator de incrementare (adună 1 la valoarea operandului)	a++	sau ++a
--	Operator de decrementare (scade 1 din valoarea operandului)	a--	sau --a

- Utilizarea acestor operatori în expresii, în formă prefixată sau postfixată, determină evaluarea acestora în moduri diferite, astfel:

$y=++x$	este echivalent cu:	$x=x+1; y=x;$
$y=x++$	este echivalent cu:	$y=x; x=x+1;$
$y=-x$	este echivalent cu:	$x=x-1; y=x;$
$y=x-$	este echivalent cu:	$y=x; x=x-1;$

- Operatori aritmetici binari:

Operator	Semnificație		Exemple
+	Adunarea celor doi operanzi		$a+b$
-	Scăderea celor doi operanzi		$a-b$
*	Înmulțirea celor doi operanzi		$a*b$
/	Împărțirea celor doi operanzi		a/b
%	Operatorul modulo (operatorul rest) (furnizează restul împărțirii operatorului stâng la operatorul drept).		$a \% b$



- Operatori relaționali binari

Operator	Semnificație	Exemple
<code>==</code>	Egal cu	<code>a==b</code>
<code>!=</code>	Diferit de	<code>a!=b</code>
<code><</code>	Mai mic decât	<code>a<b</code>
<code><=</code>	Mai mic sau egal	<code>a<=b</code>
<code>></code>	Mai mare decât	<code>a>b</code>
<code>>=</code>	Mai mare sau egal	<code>a>=b</code>

- Operatori logici pe cuvânt:

Operator	Semnificație	Exemple
<code>!</code>	Not (negație logică)	<code>!(a==b)</code>
<code>&&</code>	And (conjuncție, și logic)	<code>(a>b) && (b>c)</code>
<code> </code>	Or (disjuncție, sau logic)	<code>(a>b) (b>c)</code>

- Operatorul condițional ?

```
k = (i < 0) ? n : m
```



Biblioteci de functii

- Limbajul C poate include o serie de biblioteci prin apelarea lor în fisierele header
 - `#include <filename>`
- Biblioteca matematică apelată prin fisierul `math.h` conține funcții ca:

<code>acos(d)</code>	<code>double</code>	Return arc cosine of d (in range 0 to pi)
<code>asin(d)</code>	<code>double</code>	Return arc sine of d (in range -pi/2 to pi/2)
<code>atan(d)</code>	<code>double</code>	Return arc tangent of d (in range -pi/2 to pi/2)
<code>atan2(d1, d2)</code>	<code>double</code>	Return arc tangent of d1/d2 (in range -pi to pi)
<code>cbrt(d)</code>	<code>double</code>	Return cube root of d
<code>cos(d)</code>	<code>double</code>	Return cosine of d
<code>cosh(d)</code>	<code>double</code>	Return hyperbolic cosine of d
<code>exp(d)</code>	<code>double</code>	Return exponential of d
<code>fabs(d)</code>	<code>double</code>	Return absolute value of d
<code>hypot(d1, d2)</code>	<code>double</code>	Return <code>sqrt(d1 * d1 + d2 * d2)</code>
<code>log(d)</code>	<code>double</code>	Return natural logarithm of d
<code>log10(d)</code>	<code>double</code>	Return logarithm (base 10) of d
<code>pow(d1, d2)</code>	<code>double</code>	Return d1 raised to the power d2
<code>sin(d)</code>	<code>double</code>	Return sine of d
<code>sinh(d)</code>	<code>double</code>	Return hyperbolic sine of d
<code>sqrt(d)</code>	<code>double</code>	Return square root of d
<code>tan(d)</code>	<code>double</code>	Return tangent of d
<code>tanh(d)</code>	<code>double</code>	Return hyperbolic tangent of d



Biblioteci de functii

- De asemenea biblioteca contine constante matematice predefinite

M_PI	Pi, the ratio of a circle's circumference to its diameter.
M_PI_2	Pi divided by two.
M_PI_4	Pi divided by four.
M_1_PI	The reciprocal of pi (1/pi).
M_SQRT2	The square root of two.
M_SQRT1_2	The reciprocal of the square root of two (also the square root of 1/2).
M_E	The base of natural logarithms.



- În limbajul C, operațiile asupra fișierelor se realizează cu ajutorul unor funcții din biblioteca standard (`stdio.h`). Transferurile cu dispozitivele periferice (tastatură, monitor, disc, imprimantă, etc.) se fac prin intermediul unor dispozitive logice identice numite stream-uri (fluxuri) și prin intermediul sistemului de operare prin funcțiile `scanf`, `printf`, `fscanf` și `fprintf`.

```
#include <stdio.h>
. . .
int k;
double x, y;
. . .
scanf("%d %lf %lf", &k, &x, &y);
. . .

#include <stdio.h>
. . .
int k = 3;
double x = 5.4, y = -9.81;
. . .
printf("k = %3d x + y = %9.4f x*y = %11.3e\n", k, x + y, x*y);
. . .
```



Scriere fisier

```
#include <stdio.h>
. . .
int k = 3;
double x = 5.4, y = -9.81;
FILE *output;
. . .
output = fopen("data.out", "w");
if (output == NULL)
{
    printf("Error opening file data.out\n");
. . .
}
. . .
fprintf(output, "k = %3d  x + y = %9.4f  x*y = %11.3e\n", k, x + y, x*y);
. . .
fclose(output);
. . .
```

Dupa executie programul va scrie in fisierul data.out linia

k = 3 x + y = -4.4100 x*y = -5.297e+01



Instructiuni de control al executiei programului

● Bucla for

- for (expression 1; expression 2; expression 3) statement

```
for(int i=0; i<10; ++i){  
    // any code here is executed 10 times  
}  
  
for(int i=0; i<10; ++i){  
    // nested loop (a loop of a loop)  
    for(int j=0; j<10; ++j){  
        std::cout << i << "\t" << j << std::endl;  
    }  
}  
  
// Single line loop without {}  
for(int i=0; i<100; ++i) std::cout << i << std::endl;
```



Instructiuni de control al executiei programului

● Conditionala if / else

- if (expression) statement 1 else statement 2

```
if(a == b){  
    // this code is only executed if a == b  
}  
  
bool today_is_pancake_day = true;  
bool i_have_flour = false;  
// Do I need to buy flour?  
if(today_is_pancake_day == true && i_have_flour == false){  
    std::cout << "Go to buy flour" << std::endl;  
}  
  
int a = 5;  
int b = 7;  
// check whether a is less than b  
if(a<b){  
    std::cout << "a is less than b" << std::endl;  
}  
// only executed if condition is false (a >= b)  
else{  
    std::cout << "a is not less than b" << std::endl;  
}  
  
// single line version  
// check whether a is less than b  
if(a<b) std::cout << "a is less than b" << std::endl;  
else std::cout << "a is not less than b" << std::endl;
```



Instructiuni de control al executiei programului

```
int a = 5;
int b = 7;
// check whether a is less than b
if(a<b){
    std::cout << "a is less than b" << std::endl;
}
// only executed if condition is false (a >= b)
else if(a==b){
    std::cout << "a is equal to b" << std::endl;
}
else{
    std::cout << "a is greater than b" << std::endl;
}
```



Instructiuni de control al executiei programului

- **Bucla while - o combinatie de for si if**

- while (expression) statement

```
int i = 0;
while(i < 10){
    // this code is only executed if i < 10
    std::cout << i << std::endl;
    // Don't forget to update i each time!
    ++i;
}
```

```
int i = 0;
while(i < 10){
    // this code is only executed if i < 10
    std::cout << i << std::endl;
    // I forgot to update i each time
    // Infinite loop as i=0 every time
    }
```

- **Bucla do while**

- do statement while (expression);

```
int i = 0;
do{
    // this code is only executed if i < 10
    std::cout << i << std::endl;
    // Don't forget to update i each time!
    ++i;
} while(i < 10);
```



Pointeri și referințe

- Pointeri

- Pointerii sunt variabile care contin adresa unei zone de memorie. Ei sunt utilizati pentru a date care sunt cunoscute prin adresa zonei de memorie unde sunt alocate.

Sintaxa utilizata pentru declararea lor este:

```
tip *variabila_pointer;
```

Exemplu:

```
// declaratie variabile
int i = 17, j = 3;
// declaratie pointer
int *p;
```

Continutul memoriei in urma acestor declaratii va fi:

Variabila	p	i	j
Continut	?	17	3
Adresa	2145	2146	2147

Se observă ca pointerul la acest moment nu este initializat. Referirea prin intermediul pointerului neinitializat va genera o eroare la rularea programului.

In lucrul cu pointeri se folosesc doi operatori unari:

- &: extragerea adresei unei variabile
- *: referirea continutului zonei de memorie indicate de pointer (indirectare)



Exemplu:

<pre>// p ia adresa lui i p = &i;</pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Variabila</th><th colspan="4"></th></tr> <tr> <th>Continut</th><th colspan="4"></th></tr> <tr> <th>Adresa</th><th>2145</th><th>2146</th><th>2147</th><th>2148</th></tr> </thead> <tbody> <tr> <td>p</td><td></td><td></td><td>i</td><td>j</td></tr> <tr> <td></td><td>2147</td><td></td><td>17</td><td>3</td></tr> </tbody> </table>	Variabila					Continut					Adresa	2145	2146	2147	2148	p			i	j		2147		17	3
Variabila																										
Continut																										
Adresa	2145	2146	2147	2148																						
p			i	j																						
	2147		17	3																						
<pre>// modificarea // continutul zonei // de memorie // pointate de p (*p) = 6;</pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Variabila</th><th colspan="4"></th></tr> <tr> <th>Continut</th><th colspan="4"></th></tr> <tr> <th>Adresa</th><th>2145</th><th>2146</th><th>2147</th><th>2148</th></tr> </thead> <tbody> <tr> <td>p</td><td></td><td></td><td>i</td><td>j</td></tr> <tr> <td></td><td>2147</td><td></td><td>6</td><td>3</td></tr> </tbody> </table>	Variabila					Continut					Adresa	2145	2146	2147	2148	p			i	j		2147		6	3
Variabila																										
Continut																										
Adresa	2145	2146	2147	2148																						
p			i	j																						
	2147		6	3																						



Tablouri

- Numim tablou o colecție (grup, mulțime ordonată) de date, de același tip, situate într-o zonă de memorie continuă (elementele tabloului se află la adrese succesive). Tablourile sunt variabile compuse (structurate), deoarece grupează mai multe elemente. Variabilele tablou au nume, iar tipul tabloului este dat de tipul elementelor sale. Elementele tabloului pot fi referite prin numele tabloului și indicii (numere întregi) care reprezintă poziția elementului în cadrul tabloului.
- În funcție de numărul indiciilor utilizați pentru a referi elementele tabloului, putem întâlni tablouri unidimensionale (vectorii) sau multidimensionale (matricile sunt tablouri bidimensionale).
- Ca și variabilele simple, variabilele tablou trebuie declarate înainte de utilizare.
- Modul de declarare:

tip nume_tablou[dim_1][dim_2]...[dim_n];

unde: tip reprezintă tipul elementelor tabloului; dim_1, dim_2, ..., dim_n sunt numere întregi sau expresii constante întregi (a căror valoare este evaluată la compilare) care reprezintă limitele superioare ale indiciilor tabloului.

Exemple:

//1

```
int vect[20];           // declararea tabloului vect, de maximum 20 de elemente, de tipul int.  
                        // Se rezervă 20*sizeof(int)=20 * 2 = 40 octeți
```

//2

```
double p,q,tab[10];  
                    // declararea variabilelor simple p, q și a vectorului tab, de maximum 10 elemente, tip double
```

//3

```
#define MAX      10  
char tabc[MAX]; /*declararea tabloului tabc, de maximum MAX (10) elemente de tip char*/
```

//4

```
double matrice[2][3]; // declararea tabloului matrice (bidimensional),  
                      // maximum 2 linii și maximum 3 coloane, tip double
```



Tablourile unidimensionale

- Tablourile unidimensionale sunt tablouri cu un singur indice (vectori). Dacă tabloul conține `dim_1` elemente, indicii elementelor au valori întregi din intervalul $[0, \dim_1 - 1]$.
- La întâlnirea declarației unei variabile tablou, compilatorul alocă o zonă de memorie continuă (dată de produsul dintre dimensiunea maximă și numărul de octeți corespunzător tipului tabloului) pentru păstrarea valorilor elementelor sale. Numele tabloului poate fi utilizat în diferite expresii și valoarea lui este chiar adresa de început a zonei de memorie care i-a fost alocată. Se pot efectua operații asupra fiecărui element al tabloului, nu asupra întregului tablou.

```
int vector[6];

// Inițializarea elementelor tabloului
vector[0]=100;
vector[1]=101;
vector[2]=102;
vector[3]=103;
vector[4]=104;
vector[5]=105;

int vector[6]={100,101,102,103,104,105};

double x=9.8;
double a[5]={1.2, 3.5, x, x-1, 7.5};

for (i=0; i<n i++)
    cout<<a[i]<<' ';
```



Tablourile bidimensionale

- În limbajele C/C++ indicii de linie și de coloană pornesc de la 0

```
double q[3][2];           // declararea matricii q, cu maxim3 linii
    și 2 coloane, tip double

int mat[4][3] = {
    {10, -50, 3},
    {32, 20, 1},
    {-1, 1, -2},
    {7, -8, 19} };
```

- La declararea unei matrici și initializarea elementelor sale, se poate omite numărul maxim de linii, în schimb, datorită modului de memorare, trebuie specificat numărul maxim de coloane:

```
int mat[][][3] = {
    {10, -5, 3},
    {32, 20, 1},
    {-1, 1, -2},
    {7, -8, 9} };
int mat[][][3] = {
    {1, 1},
    {-1},
    {3, 2, 1}};
```

- mat reprezintă o matrice 3 3, ale cărei elemente se initializează astfel:

mat[0][0]=1, mat[0][1]=1, mat[1][0]=-1, mat[2][0]=3, mat[2][1]=2, mat[2][2]=1

Elementele mat[0][2], mat[1][1], mat[1][2] nu sunt inițializate. Ele au valoarea zero dacă tabloul este global și valori inițiale nedeterminate dacă tabloul este automatic.



Tablourile multidimensionale / pointeri

```
int a[2][2][3]={  
    { {10, 20}, {1, -1}, {3, 4} },  
    { {20, 30}, {50, -40}, {11, 12} }  
};
```

- În limbajele C/C++ există o strânsă legătură între pointeri și tablouri, deoarece numele unui tablou este un pointer (constant!) care are ca valoare adresa primului element din tablou.

```
int a[10], ptr;// a este definit ca &a[0]; a este pointer constant  
a = a + 1;      // ilegal  
ptr = a ;        // legal: ptr are aceeași valoare ca și a, respectiv  
                  // adresa elementului a[0]  
                  // ptr este variabilă pointer, a este constantă pointer.  
int x = a[0]; // echivalent cu x = ptr; se atribuie lui x valoarea lui a[0]
```

- Deoarece numele tabloului a este sinonim pentru adresa elementului de indice zero din tablou, asignarea `ptr=&a[0]` poate fi înlocuită, ca în exemplul anterior, cu `ptr=a`.
- Așa cum compilatorul evaluează referința către un tablou unidimensional ca un pointer, un tablou bidimensional este referit într-o manieră similară. Numele tabloului bidimensional, M, reprezintă adresa (pointer) către primul element din tabloul bidimensional, acesta fiind prima linie, `M[0]` (tablou unidimensional). `M[0]` este adresa primului element (`M[0][0]`) din linie (tablou unidimensional), deci `M[0]` este un pointer către int: `M = M[0] = &M[0][0]`. Astfel, M și M[linie] sunt pointeri constanți.



- Funcțiile comunică prin argumente: ele primesc ca parametri (argumente) datele de intrare, efectuează prelucrările descrise în corpul funcției asupra acestora și pot returna o valoare (rezultatul, datele de ieșire). Execuția programului începe cu funcția principală, numită `main`. Funcțiile pot fi descrise în cadrul aceluiași fișier, sau în fișiere diferite, care sunt testate și compilate separat, asamblarea lor realizându-se cu ajutorul linkeditorului de legături.

O funcție este formată din antet și corp:

```
antet_funcție
{
    corpul_funcției
}
```

Sau:

```
tip_val_return nume_func (lista_declarățiilor_param_formali)
{
    declarății_variabile_locale
    instrucțiuni
    return valoare
}
```

- O funcție poate fi apelată printr-o construcție urmată de punct și virgulă, numită instrucție de apel, de forma:

```
nume_funcție (lista_parametrilor_efectivi);
```



- Exemplu:

```
int f1(void)
{ double a,b; int c;
  . . .
  return c; // a, b, c - variabile locale, vizibile doar în
             // corpul funcției
}
void main()
{ . . . . . // variabile a și b nu sunt accesibile în
             // main()
}
```



● Specificarea prototipului

```
#include <iostream.h>
#include <math.h>
double omega(double, int);
// prototipul funcției omega - antet din care lipsesc numele
// parametrilor formali
double psi(double);           // prototipul funcției psi

void main()
{double u, z; int m; cout<<"u="; cin>>u; cout<<"m="; cin>>m;
z=2*omega(2* psi(u) + 1, m) + omega(2*pow(u,2) - 3, m+1);
cout<<"z="<<z<<'\n'; }

double omega(double x, int i);      // definiția funcției omega
{ double s=0; int i;
for (i=1; i<=n; i++)           s += sin (i*x) * cos (i*x);
return s; }

double psi( double x)             // definiția funcției psi
{ return sqrt( 1 + exp (- pow (x, 2))); }
```

- ❖ Prototipurile funcților din biblioteci (predefinite) se găsesc în headere. Utilizarea unei astfel de funcții impune doar includerea în program a headerului asociat, cu ajutorul directivei preprocesor #include.



- Funcțiile comunică între ele prin argumente (parametrii).
- Există următoarele moduri de transfer (transmitere) a parametrilor către funcțiile apelate:
 - Transfer prin valoare;
 - Transfer prin pointeri;
 - Transfer prin referință.
- Transferul prin valoare:

```
double psi(int a, double b)
{
    if (a > 0)          return a*b*2;
    else      return -a+3*b; }

void main()
{ int x=4; double y=12.6, z; z=psi ( 3*x+9, y-5) + 28;
cout<<"z="<<z<<' \n' ; }
```



Transfer prin pointeri

- În unele cazuri, parametrii transmiți unei funcții pot fi pointeri (variabile care conțin adrese). În aceste cazuri, parametrii formali ai funcției apelate vor fi inițializați cu valorile parametrilor efectivi, deci cu valorile unor adrese. Astfel, funcția apelată poate modifica conținutul locațiilor spre care pointează argumentele (pointerii).

```
#include <stdio.h>
#include <stdlib.h>
/* Prototype for function factorial() */
void factorial(int, double *);
int main()
{
    int j;
    double fact;
    /* Print factorials of all integers between 0 and 20 */
    for (j = 0; j <= 20; ++j)
    {
        factorial(j, &fact);
        printf("j = %3d factorial(j) = %12.3e\n", j, fact);
    }
    return 0;
}
//%%%%%%%%%%%%%%%
void factorial(int n, double *fact)
{
    *fact = 1.;
    /* Abort if n is negative integer */
    if (n < 0)
    {
        printf("\nError: factorial of negative integer not defined\n");
        exit(1);
    }
    /* Calculate factorial */
    for (; n > 0; --n) *fact *= (double) n;
    return;
}
```



Pointer la functie

```
#include <iostream>
using namespace std;
int adunare(int a, int b)
{
    return a + b;
}
int scadere(int a, int b)
{
    return a - b;
}
// functie care primeste ca parametru un pointer la functie
int aplica_operatie(int a, int b, int(*pFunctie)(int, int) )
{
    // apel functie prin intermediul pointerului primit ca
    // parametru
    return (*pFunctie) (a,b);
}
```



Pointer la functie

```
void main()
{
    // declarare si citire variabile
    int a, b;
        cout << "a=";
        cin >> a;
        cout << "b=";
        cin >> b;
    // declarare 'pf' ca pointer la functie
    // care primeste doi intregi ca parametri
    // si intoarce un intreg
    int (*pf)( int, int);
    // incarcare pointer la functie
        pf = adunare;
    // apel de functie prin pointer
        cout << "Suma:" << (*pf)(a,b) << endl;
    // trimitera pointerului ca parametru
        cout << "Suma: " << aplica_operatie(a,b, pf) << endl;
    // folosirea directa a numelui functiei
    // pentru trimitera parametrului
    // de tip pointer la functie
        cout << "Diferenta: " << aplica_operatie(a,b, scadere) << endl;
}
```



C3 - Rezolvarea numerica a ecuatiilor algebrice si transcendentale

Lect. dr. Ioan Dumitru



În acest capitol, dacă nu se specifică altfel, ne vom referi la o funcție reală $f : [a, b] \rightarrow \mathbb{R}$ de o variabilă reală $y = f(x)$. Ecuațiile algebrice sau transcendentale sunt relații de tipul:

$$(3.1) \quad f(x) = 0$$

și oricare din valorile $\xi \in [a, b]$ pentru care

$$(3.2) \quad f(\xi) \equiv 0$$

se numește *rădăcină* (sau *soluție*) a ecuației (3.1). Ecuația (3.1) se numește *ecuație algebraică* dacă $f(x)$ este (sau poate fi pusă sub forma) unui polinom de variabilă x :

$$(3.3) \quad f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

Dacă $f(x)$ nu poate fi exprimat sub forma (3.3) atunci ecuația se numește *ecuație transcendentală* sau *neliniară*.



Exemple ecuații algebrice și transcendentă

1. Ecuații algebrice:

$$x^2 - 2x + 1 = 0$$

$$x^3 + 6x^2 - 3x + 4 = 0$$

$$(x + 3)(x^2 + 1) = 0$$

$$\frac{x^2 + 2x + 1}{x + 1} = 0$$

2. Ecuații transcendentă:

$$x^2 - \ln x = 0$$

$$\cos x - x = 0$$

$$\exp(-x^2) + x \sin x = 0$$



Radacini aproximative

În calculul numeric se mai pune și problema găsirii unei rădăcini aproximative încât $f(\xi_a) \approx 0$. Rămâne astfel deschisă problema găsirii unei semnificații pentru semnul \approx ce apare mai sus. Vom considera aici că ξ_a este soluție aproximativă pentru ecuația (3.1) dacă există o valoare reală $\delta > 0$ astfel încât:

$$(3.4) \quad |f(\xi_a)| < \delta$$

Condiția (3.4) este doar o condiție necesară pe care trebuie să o îndeplinească rădăcina aproximativă, nu și suficientă deoarece funcția $f(x)$ poate să ia valori foarte apropiate de 0 fără însă ca acestea să poată fi considerate rădăcini aproximative, de exemplu: $f(x) = e^{-x}$. Din acest motiv la problema găsirii unei rădăcini aproximative se consideră 3 etape distincte ce trebuie parcuse în mod obligatoriu:

1. găsirea unui criteriu de existență a rădăcinilor;
2. dezvoltarea unui algoritm de calcul pentru rădăcinile aproximative;
3. evaluarea erorii de aproximare pentru rădăcinile aproximative, altfel spus trebuie să se găsească $\epsilon > 0$ încât

$$(3.5) \quad |\xi - \xi_a| < \epsilon$$

(soluția aproximativă ξ_a trebuie să fie *suficient de "apropiată"* de soluția exactă ξ)



Interpretare geometrică

Dacă $f : [a, b] \rightarrow \mathbb{R}$ este o funcție de variabilă reală cu valori reale $y = f(x)$ atunci acesteia i se asociază o curbă în spațiul planului xy numită *graficul funcției*. Ecuația (3.1) determină de fapt abscisele punctelor de intersecție cu axa Ox , definită evident de $y = 0$ (v. Figura 3.1). Interpretarea geometrică a rădăcinilor unei ecuații neliniare permite dezvoltarea unei metode rapide de aproximare a rădăcinilor:

1. se reprezintă grafic funcția $f(x)$ pe intervalul de definiție $[a, b]$;
2. se reprezintă axa Ox (condiția $y = 0$);
3. se caută subintervalele în care graficul funcției $f(x)$ intersectează axa Ox ;
4. dacă funcția $f(x)$ este continuă pe aceste subintervale atunci se poate reprezenta funcția pe subintervale separate cu diviziuni din ce în ce mai fine pentru aproximarea rădăcinii;
5. se repetă punctul 4. până când se obține o rădăcină aproximativă pe fiecare interval (în limitele erorilor impuse inițial).



Solutii aproximative si exacte

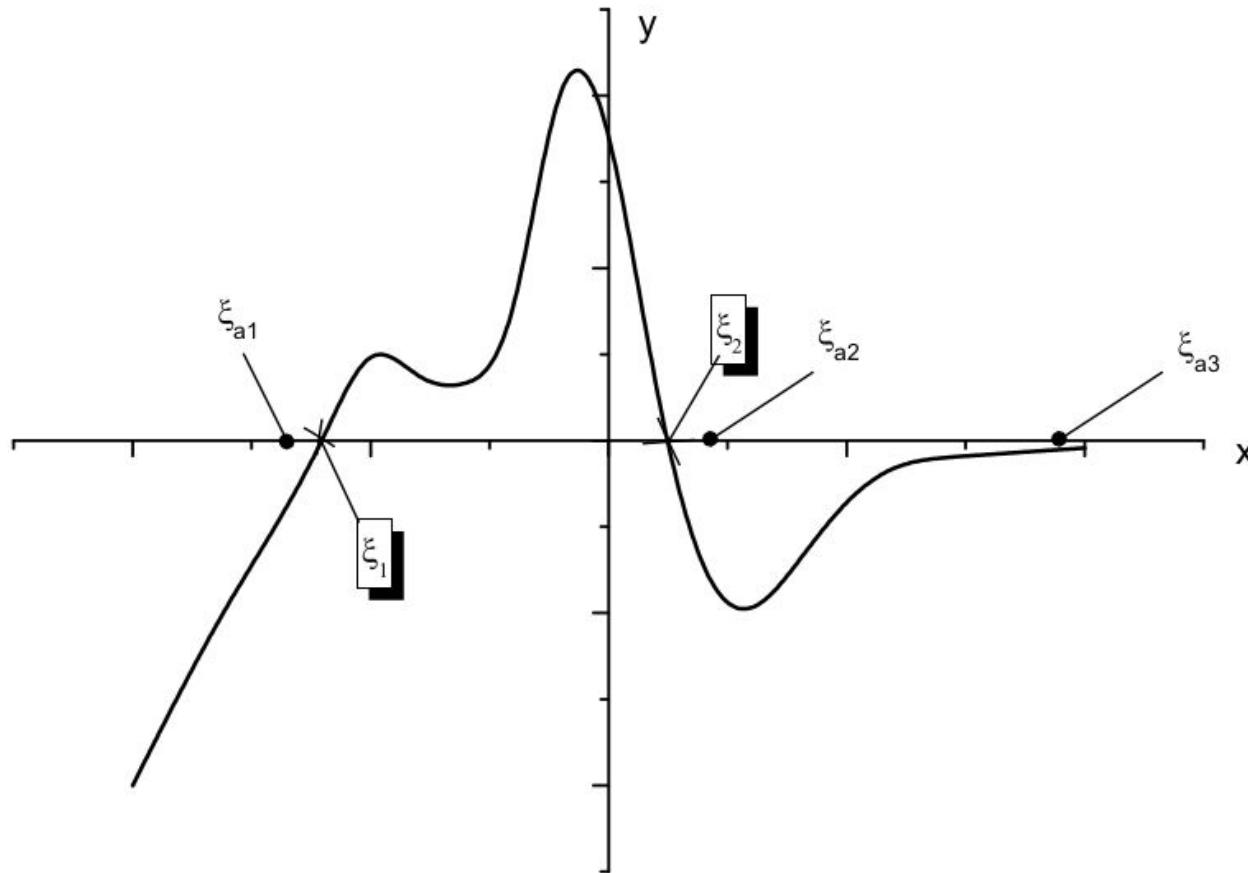


Figura 3.1: Interpretarea geometrică a problemei rezolvării ecuațiilor transcențiente: ξ_{a1} și ξ_{a2} sunt rădăcini aproximative corespunzătoare ξ_1 și ξ_2 soluției exacte iar ξ_{a3} este o rădăcină aproximativă falsă.



Metoda înjumătățirii intervalului

Metoda înjumătățirii intervalului permite aflarea unei soluții (a unei rădăcini) aproximative $\xi_a \in [a, b]$ din intervalul $[a, b]$ dacă sunt îndeplinite următoarele condiții:

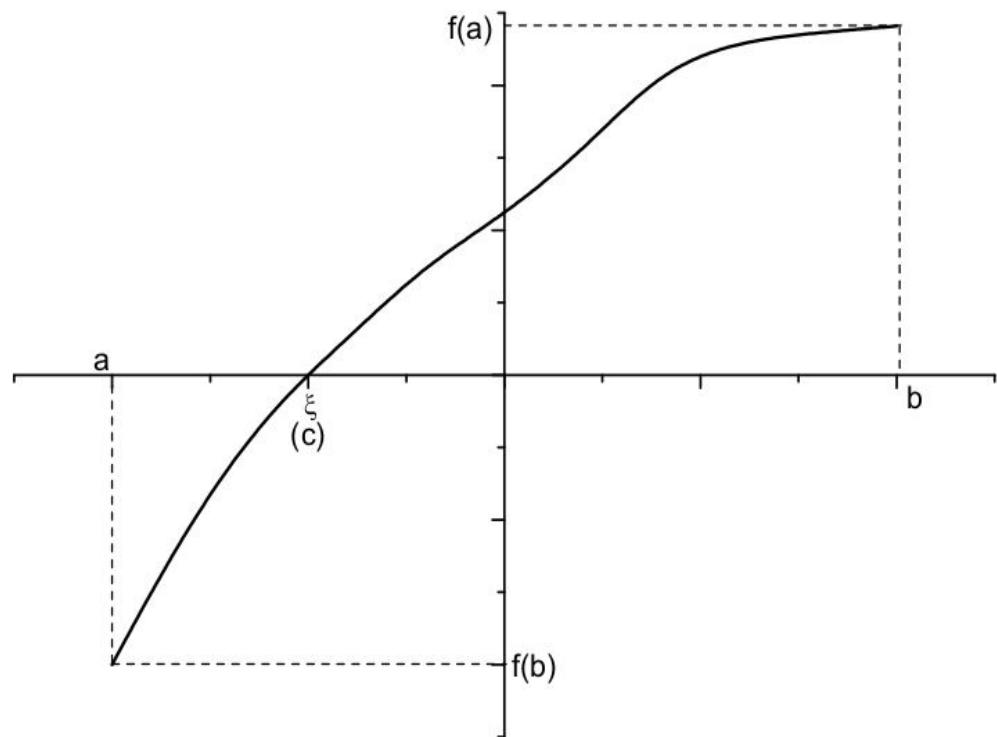
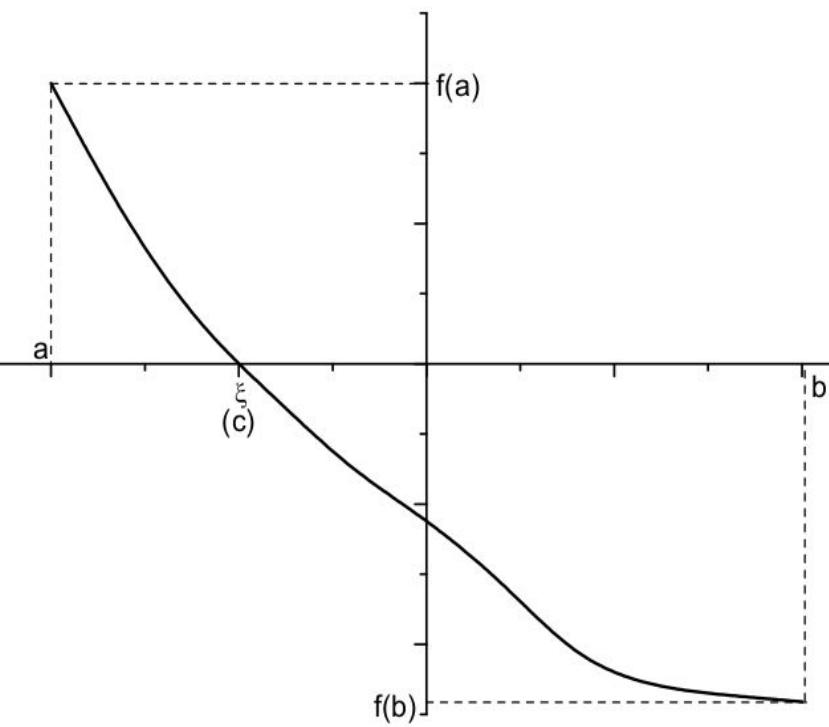
- $f(x)$ este o funcție continuă pe intervalul $[a, b]$;
- $f(a)$ și $f(b)$ au semne opuse (v. Figura 3.2):

$$(3.6) \quad f(a) \cdot f(b) < 0$$

Dacă $f(x)$ este o funcție continuă pe intervalul $[a, b]$ și $f(a)f(b) < 0$ atunci în acest interval există cel puțin o rădăcină $\xi \in [a, b]$ încât $f(\xi) = 0$.



Conditii de aplicare



Algoritmul metodei injumatăririi intervalului

1. se citesc valorile a, b, ϵ (eroarea de calcul);
2. se verifică dacă $f(a)$ sau $f(b)$ sunt rădăcini exacte ($f(a) = 0$ sau $f(b) = 0$);
3. se verifică condiția (3.6) ($f(a)f(b) < 0$);

NU atunci "nu se poate aplica metoda pentru găsirea rădăcinilor în intervalul $[a, b]$ " salt la pasul 1. (sau STOP);

DA atunci continuă;

4. se calculează

$$(3.7) \quad c = \frac{a + b}{2}$$

5. dacă

$$(3.8) \quad |f(c)| < \epsilon$$

atunci " c este soluție aproximativă pentru ecuația $f(x) = 0$ pe intervalul $[a, b]$ "
→ STOP;

6. dacă $f(c) \cdot f(a) < 0$ atunci $b = c$;
7. dacă $f(c) \cdot f(b) < 0$ atunci $a = c$;
8. repetă pașii 4. → 7.



- la fiecare pas i de parcurgere a algoritmului de mai sus se obțin intervale $[a_i, b_i]$ astfel încât

$$(3.9) \quad a \leq a_1 \leq \dots \leq \xi \leq \dots \leq b_n \leq \dots \leq b$$

adică obținem două siruri de numere (a_n) și (b_n) cu limitele:

$$(3.10) \quad \lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n = \xi$$

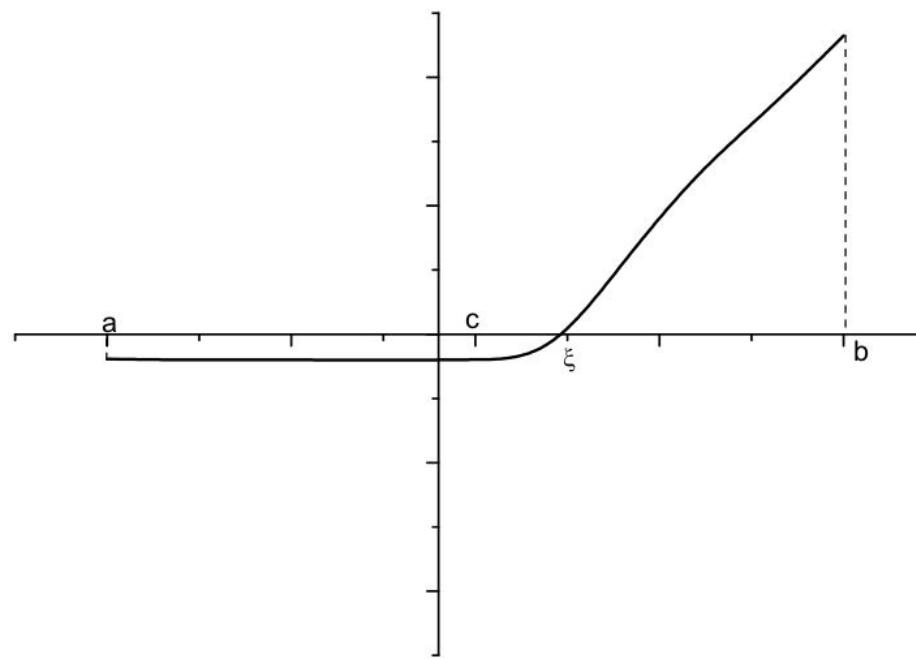
Dacă funcția $f(x)$ variază lent în intervalul $[a, b]$ atunci algoritmul descris mai sus converge greu; mai mult, se poate întâmpla ca și condiția (3.8) să nu determine soluția aproximativă ci o valoare foarte apropiată de zero dar îndepărtată față de soluția exactă ξ (v. Figura 3.4) Din acest motiv trebuie găsit un alt criteriu pentru oprirea iterațiilor în cadrul algoritmului de bisecție. Din fericire un astfel de criteriu există și se poate determina numărul total de iterații pentru obținerea rădăcinii cu toleranță ϵ . Dacă după n iterații intervalul în care se află



rădăcina este de dimensiune ϵ_n atunci după încă o iterație rădăcina se va afla într-un interval de lărgime: $\epsilon_{n+1} = \frac{\epsilon_n}{2}$ de unde numărul de iterații pentru a calcula rădăcina cu toleranță ϵ va fi:

$$(3.11) \quad n = \log_2 \frac{\epsilon_0}{\epsilon}$$

unde ϵ_0 este lărgimea inițială a intervalului de căutare: $\epsilon_0 = b - a$.



- Metoda bisecției are întotdeauna o soluție:
 - dacă în intervalul $[a, b]$ sunt mai multe rădăcini atunci metoda înjumătățirii intervalului va avea ca rezultat o aproximație pentru una din rădăcini;
 - dacă în intervalul $[a, b]$ nu există nici o rădăcină dar $f(a)f(b) < 0$ atunci metoda converge către punctul de singularitate din acel interval.



Exemplu

Funcția $h(x) = x \sin x$ este o funcție ce intervine în studiul oscilațiilor forțate, neamortizate. Se pune problema găsirii valorii lui x din intervalul $[0, 2]$, unde funcția ia valoarea $h(x) = 1$.

Va trebui deci să utilizăm metoda bisecției pentru găsirea zeroului funcției

$f(x) = x \sin x - 1$ cu $a_0 = 0$ și $b_0 = 2$. Se observă că:

$$f(0) = -1 \text{ și } f(2) = 0.818595$$

Ca urmare, ecuația noastră are soluții pe $[0, 2]$. Calcule simple ne conduc la tabelul de date de mai jos. Continuând procesul iterativ vom obține o

TAB 1. Soluții ale ecuației $x \sin x - 1 = 0$ utilizând metoda bisecției.

k (iterație)	a_k	c_k	b_k	$f(c_k)$
0	0	1	2	-0.158529
1	1.0	1.5	2.0	0.496242
2	1.00	1.25	1.50	0.186231
3	1.000	1.125	1.250	0.015051
4	1.0000	1.0625	1.1250	-0.071827
5	1.06250	1.09375	1.12500	-0.028362
:	:	:	:	:

secvență de numere ce converge la $\xi \simeq 1.114157141$.



Metoda coardei (secantei)

Se consideră o funcție $f(x)$ continuă și derivabilă pe intervalul $[a, b]$ astfel încât își modifică semnul, adică este îndeplinită condiția $f(a) \cdot f(b) < 0$. Fără a limita generalitatea metodei presupunem că ecuația $f(x)=0$ are o singură rădăcină $\xi \in (a, b)$ ca în figura 1.2 (cu $f(a) < 0$ și $f(b) > 0$).

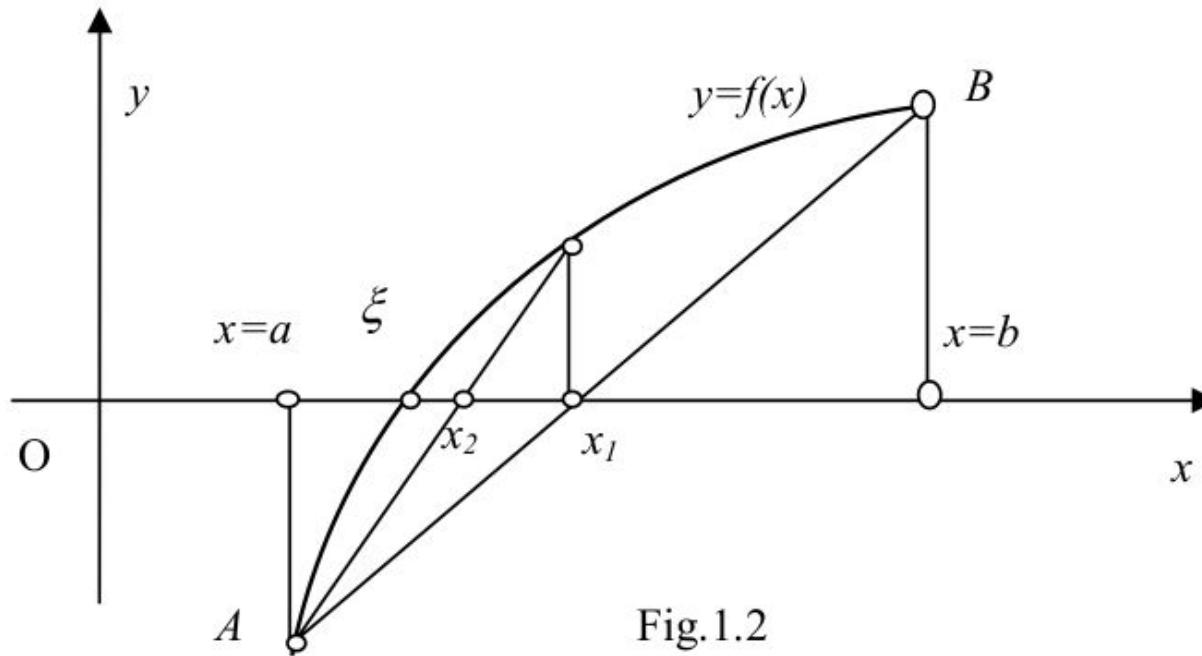


Fig.1.2



Metoda coardei (secantei)

În primă fază, se poate aproxima rădăcina ecuației $f(x)=0$ cu punctul de intersecție cu axa Ox a dreptei care trece prin punctele $A(a, f(a))$ și $B(b, f(b))$ de ecuație:

$$\frac{y-f(a)}{f(b)-f(a)} = \frac{x-a}{b-a} \Leftrightarrow y-f(a) = (x-a) \frac{f(b)-f(a)}{b-a} \quad (1.3)$$

Punctul de intersecție al dreptei cu axa Ox se obține introducând condiția $y=0$ în ecuația (1.3). Se obține:

$$x_1 = a - f(a) \frac{b-a}{f(b)-f(a)} \quad (1.4)$$

Din figura 1.3 rezultă că noul subinterval al rădăcinii ξ este (a, x_1) deoarece $f(a) \cdot f(x_1) < 0$. În continuare algoritmul se repetă.

Presupunem că ultimul subinterval pentru care funcția își modifică semnul este (x_{n-1}, x_n) , adică este îndeplinită condiția: $f(x_{n-1}) \cdot f(x_n) < 0$ (1.5)

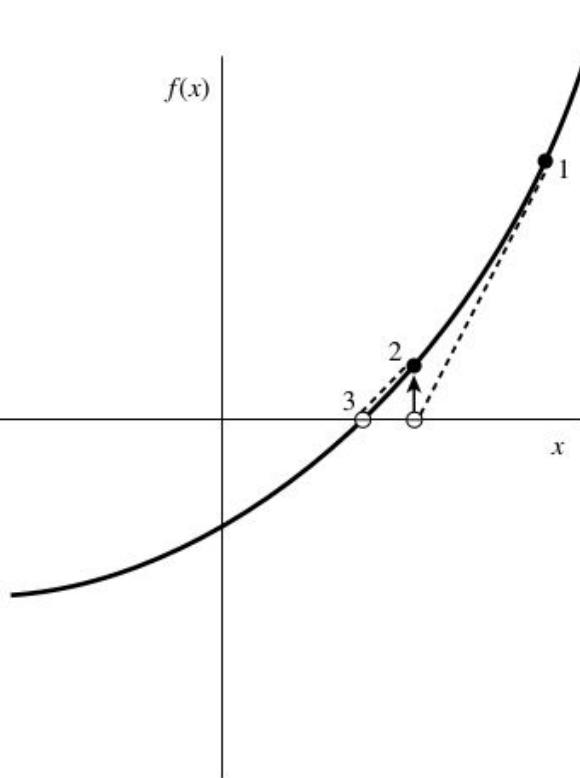
Ținând seama de relația (1.4) se poate scrie următoarea *relație de recurență a metodei coardei sau secantei*:

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \quad (1.6)$$



Metoda tangentei (a lui Newton)

Această metodă mai este denumită și metoda *Newton-Raphson* și necesită evaluarea atât a funcției $f(x)$ cât și a derivatei de ordinul I $f'(x)$ aceleiași funcții pe intervalul $[a, b]$ ce conține cel puțin o rădăcină: condiția suficientă $f(a)f(b) < 0$. Algoritmul metodei Newton-Raphson este similar cu algoritmul metodelor anterioare: se obține un sir de aproximări succesive pentru rădăcina ecuației (v. Figura 3.7):



1. se alege o valoare inițială $x_0 \in [a, b]$ pentru aproximarea rădăcinii;
2. se calculează valoarea $f(x_0)$ și se determină astfel punctul de coordonate $(x_0, f(x_0))$;
3. noua valoare aproximativă pentru rădăcina ecuației este x_1 ce corespunde punctului de intersecție al dreptei tangente la graficul funcției $f(x)$ ce trece prin punctul $(x_0, f(x_0))$ cu axa Ox .
4. se verifică dacă x_1 este o rădăcină aproximativă pentru ecuația transcendentă ce se încadrează în limitele de eroare stabilite de fiecare utilizator;
 - DA atunci se afișează rezultatul și eroarea estimată → STOP.;
 - NU atunci $x_0 \leftarrow x_1$ se repetă pașii 1.-4.



Metoda tangentei (Newton-Raphson)

Ecuația tangentei la curba de ecuație $y = f(x)$ ce trece printr-un punct de coordonate $(x_i, f(x_i))$ este:

$$y = m x + n$$

cu m panta dreptei tangentă

$$m = f'(x_i)$$

și care trebuie să verifice:

$$f(x_i) = f'(x_i) \cdot x_i \Rightarrow$$

$$n = f(x_i) - f'(x_i)x_i$$

de unde vom deduce ecuația tangentei:

$$(3.14) \quad y = f'(x_i)x + f(x_i) - f'(x_i)x_i$$

și condiția de intersecție cu axa Ox : $y = 0$ conduce la relația de recurență:

$$(3.15) \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Relația (3.15) determină noua soluție aproximativă pentru ecuația $f(x) = 0$. Sirul de numere $(x_i)_{i \geq 0}$ este un sir care converge către soluția exactă a ecuației transcendentale: $\lim_{n \rightarrow \infty} x_n = \xi$.



Avantaje/Dezavantaje pentru metoda tangentei

Avantajele metodei:

- evaluarea funcției într-un singur punct x_i ;
- metoda poate fi ușor generalizată pentru cazul unui sistem de ecuații nelineare;
- marele avantaj îl constituie viteza de convergență a sirului de aproximări succesiive către soluția exactă ξ . De exemplu: presupunem că soluția x_i este calculată cu o eroare absolută ϵ_i atunci noua soluție aproximativă este determinată cu eroarea:

$$\epsilon_{i+1} = \epsilon_i^2 \frac{f''(x)}{2f'(x)}$$

adică se eroarea scade cu pătratul ϵ_{i+1}^2 pentru x_{i+1} .

Dezavantaje:

- metoda necesită cunoașterea și evaluarea derivatei funcției f . Dacă această derivată nu se cunoaște atunci se poate approxima prin:

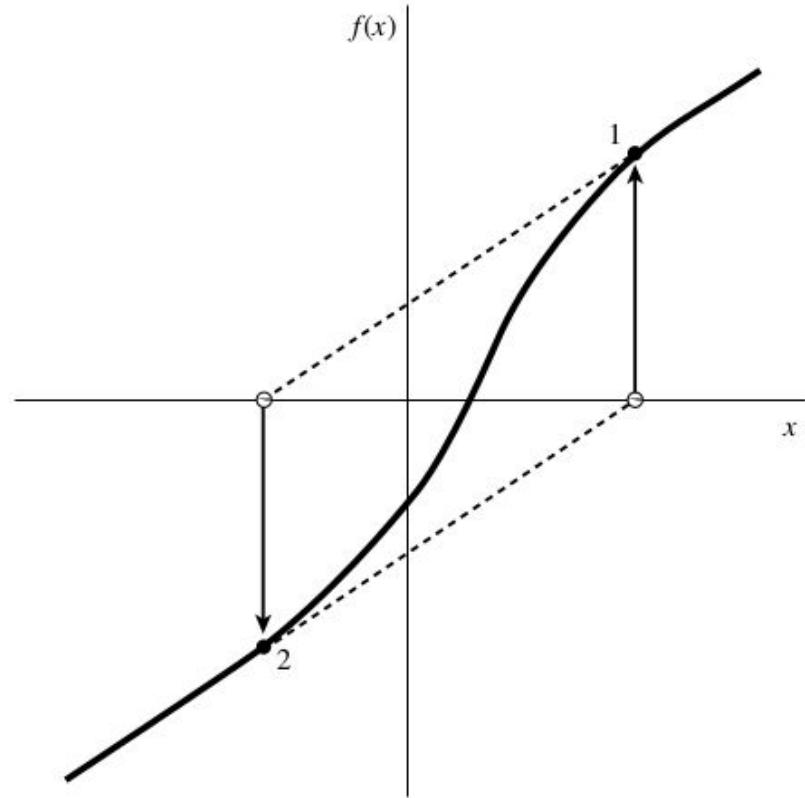
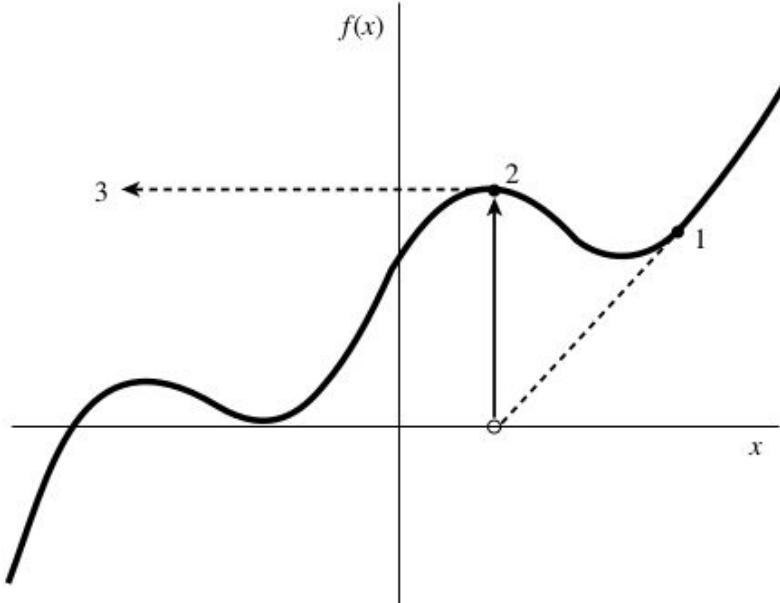
$$f'(x) \approx \frac{f(x + dx) - f(x)}{dx}$$

dar, evident, acest lucru trebuie evitat pentru că introduce noi surse de erori (apare 1 diferență + 1 împărțire) și viteza de convergență scade.

- metoda dă rezultate foarte bune doar pentru o aproximare inițială x_0 apropiată de valoarea exactă ξ , altfel se poate întâmpla ca între ξ și x_0 funcția să prezinte un punct de extremum ($f'(x_i) = 0$) caz în care se întrerupe sirul de calcule



Cazuri in care metoda Newton-Raphson nu converge



Metoda tangentelor de ordinul I a lui Newton

În aceste condiții funcția admite o singură rădăcină în intervalul [a, b] și se poate aplica metoda tangentelor de ordinul I a lui Newton. Prin dezvoltarea în serie Taylor a funcției $f(x)$ în jurul punctului $x=a$ se obține:

$$f(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \frac{(x-a)^3}{3!} f'''(a) + \dots \quad (1.7)$$

Reținând doar primii doi termeni ai acestei dezvoltări, se obține ecuația unei drepte care reprezintă tangenta la graficul funcției în punctul A , aşa cum rezultă și din figura 1.3:

$$y_1 = f(a) + (x-a)f'(a) \quad (1.8)$$

Dacă în ecuația (1.8) se pune condiția $y_1=0$, se obține punctul de intersecție al tangentei cu axa Ox :

$$x_1 = a - \frac{f(a)}{f'(a)} \quad (1.9)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



Metoda tangentelor de ordinul II a lui Newton

Se consideră funcția $f(x)$ care îndeplinește următoarele condiții: este continuă și derivabilă pe intervalul $[a, b]$, își schimbă semnul: $f(a) \cdot f(b) < 0$, este strict monotonă ($f'(x) > 0$ sau $f'(x) < 0$) și graficul ei nu admite nici un punct de inflexiune pe intervalul $[a, b]$: $f''(x) \neq 0$. În aceste condiții funcția admite o singură rădăcină în intervalul $[a, b]$ și se poate aplica metoda tangentelor de ordinul II a lui Newton. Prin dezvoltarea (1.7) în serie Taylor a funcției $f(x)$ în jurul punctului $x=a$ se rețin doar primii doi termeni ai acestei dezvoltări, se obține ecuația unei parabole

$$y = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) \quad (1.14)$$

Se observă din relația (1.14) că funcția $y(x)$ trece prin punctul $A(a, f(a))$ și are aceeași derivate cu $f(x)$ în punctul $x=a$: $y'(a) = f'(a)$ respectiv $y''(a) = f''(a)$:

Punând condiția $y=0$ în ecuația (1.14), se obține ecuația:

$$f(a) + (x-a) \left[f'(a) + \frac{(x-a)}{2!} f''(a) \right] = 0 \quad (1.15)$$

Înlocuind expresia $(x-a)$ din interiorul parantezei drepte cu expresia obținută în cadrul metodei Newton Raphson:



Metoda tangentelor de ordinul II a lui Newton

$$x - a = -\frac{f(a)}{f'(a)} \quad (1.16)$$

se obține ecuația: $f(a) + (x - a) \left[f'(a) - \frac{1}{2} \frac{f(a)}{f'(a)} f''(a) \right] = 0 \quad (1.17)$

Soluția ecuației (1.17) este dată de relația:

$$x = a - \frac{1}{\frac{f'(a)}{f(a)} - \frac{f''(a)}{2 \cdot f'(a)}} \quad (1.18)$$

Dacă această soluție este în afara intervalului () atunci se schimbă punctul de start al metodei în $x=b$, ca la metoda tangentelor de ordinul I:

$$x = b - \frac{1}{\frac{f'(b)}{f(b)} - \frac{f''(b)}{2 \cdot f'(b)}} \quad (1.19)$$

Ținând seama de relațiile (1.18) și (1.19) se deduce *relația de recurență a metodei tangentelor de ordinul II a lui Newton*:

$$x_{n+1} = x_n - \frac{1}{\frac{f'(x_n)}{f(x_n)} - \frac{f''(x_n)}{2 \cdot f'(x_n)}} \quad (1.20)$$



Fie $f(x) = x^2 - x + 2$.

- (a) Să se găsească formula Newton-Raphson $x_k = g(x_{k-1})$.
- (b) Începând cu $x_0 = -1.5$ să se găsească x_1, x_2 , și x_3 .

Rezolvare:

- (a) Conform teoremei Newton-Raphson:

$$x_k = \frac{x_{k-1}^2 - 2}{2x_{k-1}}$$

- (b) $x_1 = 0.125, x_2 = 2.6458, x_3 = 1.1651$.

7. Fie funcția $f(x) = xe^{-x}$.

- (a) Să se găsească formula Newton-Raphson $x_k = g(x_{k-1})$.
- (b) Dacă $x_0 = 0.2$ să se găsească x_1, x_2, x_3 , și x_4 . Cât este $\lim_{n \rightarrow \infty} x_n$?
- (c) Dacă $x_0 = 20$ să se găsească x_1, x_2, x_3 , și x_4 . Cât este $\lim_{n \rightarrow \infty} x_n$?
- (d) Care este valoarea lui $f(x_4)$ de la punctul (c)?

Rezolvare:

- (a) Conform teoremei Newton-Raphson:

$$x_k = \frac{x_{k-1}^3}{x_{k-1}^2 - 1}$$

- (b) $x_1 = -0.0083333333, x_2 = 0.0000005787, x_3 = 0.0$ iar $x_4 = 0.0$.
 $\lim_{n \rightarrow \infty} x_n = 0$
- (c) $x_1 = 20.05012531, x_2 = 20.10012469, x_3 = 20.14999907$ iar $x_4 = 20.19974939$. $\lim_{n \rightarrow \infty} x_n = \infty$
- (d) $f(x_4) = 3.4096 \times 10^{-8}$



Extragerea rădăcinii dintr-un număr pozitiv

Rădăcina de ordinul k dintr-un număr pozitiv N : $x = \sqrt[k]{N}$ este echivalentă cu soluția ecuației: $f(x) = x^k - N = 0$ (1.27)

Folosind relația de recurență (1.11) de la metoda tangentelor de ordinul I a lui Newton în care se înlocuiește derivata: $f'(x) = kx^{k-1}$, se obține următoarea relație de recurență pentru calculul rădăcinii de ordinul k dintr-un număr N :

$$x_{n+1} = \frac{(k-1)x_n^k + N}{kx_n^{k-1}} \quad \text{sau} \quad x_{n+1} = \frac{1}{k} \left[(k-1)x_n + \frac{N}{x_n^{k-1}} \right] \quad (1.28)$$

Folosind relația de recurență (1.28) să se calculeze $\sqrt[7]{5}$ ($k=7$, $N=5$) cu o eroare $\varepsilon < 10^{-5}$.

Rezolvare

Înlocuind $k=7$ și $N=5$ în relația (1.28) se obține relația de recurență:

$$x_{n+1} = \frac{1}{7} \left[6x_n + \frac{5}{x_n^6} \right] \quad (1.29)$$

Dacă se consideră ca punct de start $x_1=1$ se obțin valorile din tabelul 1.9.

Tabelul 1.9

Pas	x_n	x_{n+1}	Eroarea ε
1	1	1,571428	
2	1,571428	1,39437	0,176858
3	1,39437	1,292360	0,102077
4	1,292360	1,261000	0,03136
5	1,261000	1,258514	0,002486
6.	1,258514	1,2584989	0,000015



14. Un proiectil este aruncat din originea unui sistem de coordonate sub un unghi $\alpha_0 = 45^\circ$ cu viteza inițială $v_0 = \sqrt{2} \times 160$ m/s. Rezistența aerului nu este neglijată, presupunând o dependență liniară dintre aceasta și viteza particulei. În acest caz, înălțimea $y = y(t)$ și distanța față de origine $x = x(t)$ va asculta de legile:

$$y = f(t) = (Cv_y + 32C^2)(1 - e^{-t/C}) - 32Ct$$

$$x = r(t) = Cv_x(1 - e^{-t/C})$$

unde $C = m/k$, m este masa particulei, iar k coeficientul de rezistență al aerului. Știind că $C = 10$ să se determine timpul până la impact și distanța străbătută de proiectil.

Rezolvare:

Conform tabelului, înălțimea va fi zero după 8.74217466 s. Pe de altă

Iterație	Timp t_k	Inălțime $f(t_k)$
0	8.00000000	83.22097200
1	8.79773101	-6.68369700
2	8.74242941	-0.03050700
3	8.74217467	-0.00000100
4	8.74217466	0.00000000

parte, utilizând cea de a doua ecuație parametrică vom găsi că distanța străbătută de proiectil este $H = 932.498630$ m.

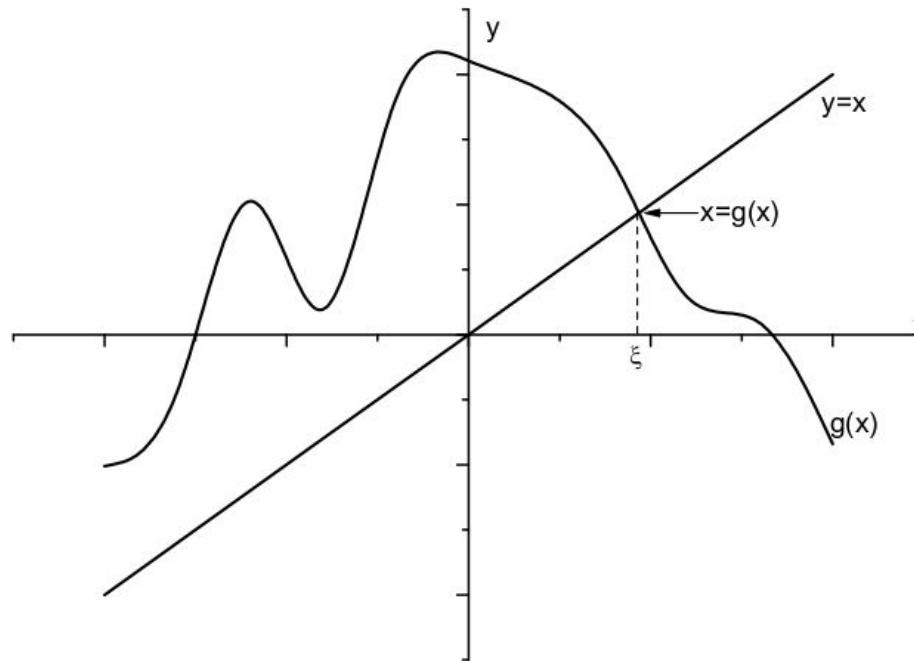


Metoda aproximatiilor succesive

Metoda aproximatiilor succesive are la bază un principiu fundamental în calculul numeric: *iterația*. Astfel orice ecuație neliniară (3.1) se poate rescrie sub forma:

$$(3.16) \quad x = f(x) + x = g(x)$$

Ecuația neliniară scrisă sub forma (3.16) permite dezvoltarea unei metode de tip iterativ. Conform acestei relații rădăcina ξ a funcției $f(x)$ se află la intersecția dintre graficul funcției $g(x)$ și prima bisectoare $y = x$ (Figura 3.9)



Sirul de aproximări successive

În această metodă se construiește un sir de aproximări successive plecând de la o valoare aproximativă inițială x_0 astfel:

$$x_1 = g(x_0)$$

$$x_3 = g(x_2)$$

...

$$x_{n-1} = g(x_{n-2})$$

$$x_2 = g(x_1)$$

$$x_4 = g(x_3)$$

...

$$x_n = g(x_{n-1})$$

Dacă există limita sirului (x_n) atunci:

$$(3.17) \qquad \lim_{n \rightarrow \infty} x_n = \xi$$

Punctul ξ pentru care $g(\xi) = \xi$ se numește *punct fix* al funcției $g(x)$. Cu alte cuvinte rădăcinile ecuației neliniare $f(x)$ sunt puncte fixe pentru funcția $g(x)$.



Condiția de existență a punctelor fixe

Condiția de existență a punctelor fixe determină și posibilitatea de a utiliza metoda aproximățiilor succesive pentru calculul rădăcinilor:

- dacă pentru $y = g(x)$ funcție continuă pe intervalul $[a, b]$ avem $y \in [a, b]$ oricare ar fi $x \in [a, b]$ atunci g are cel puțin un punct fix în $[a, b]$.
- dacă $g(x)$ este derivabilă pe (a, b) și există o constantă pozitivă $K < 1$ astfel încât

$$(3.18) \quad |g'(x)| \leq K < 1 \quad \forall x \in (a, b)$$

atunci g are un punct fix unic $\xi \in [a, b]$.

Ca în orice metodă iterativă și în metoda aproximățiilor succesive trebuie să existe un criteriu de oprire a iterațiilor. Din păcate în practică de cele mai multe ori nu se cunoaște a priori rădăcina exactă și se folosește drept criteriu de oprire a iterațiilor inegalitatea:

$$(3.19) \quad |x_{n+1} - x_n| < \epsilon$$

unde ϵ este o valoare limită pentru eroarea absolută.



Secvența de algoritm de mai jos se poate aplica dacă s-a verificat existența punctului fix pentru $g(x)$:

1. Citire $\epsilon, x_0, \text{maxiter}$ – numărul maxim de iterații
2. Calculul $x_1 = g(x_0)$
3. Se verifică $|x_1 - x_0| < \epsilon$

DA \rightarrow afișează soluția aproximativă x_1 ;

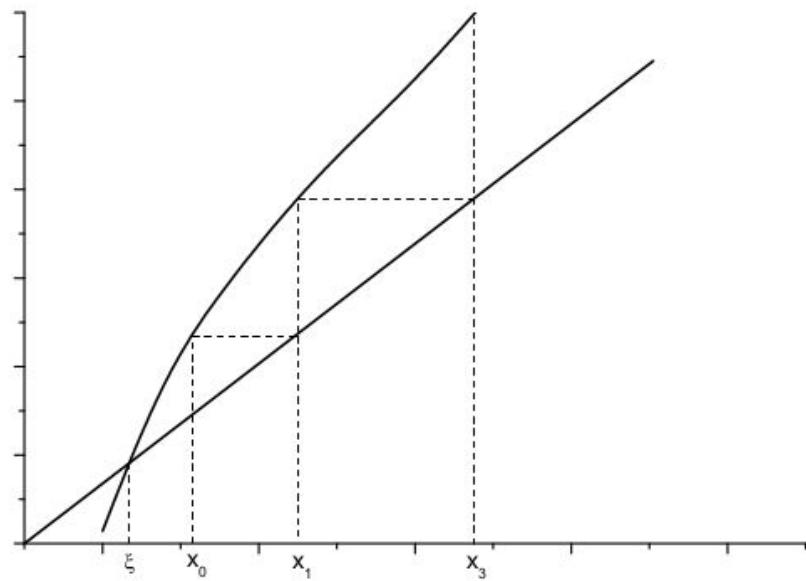
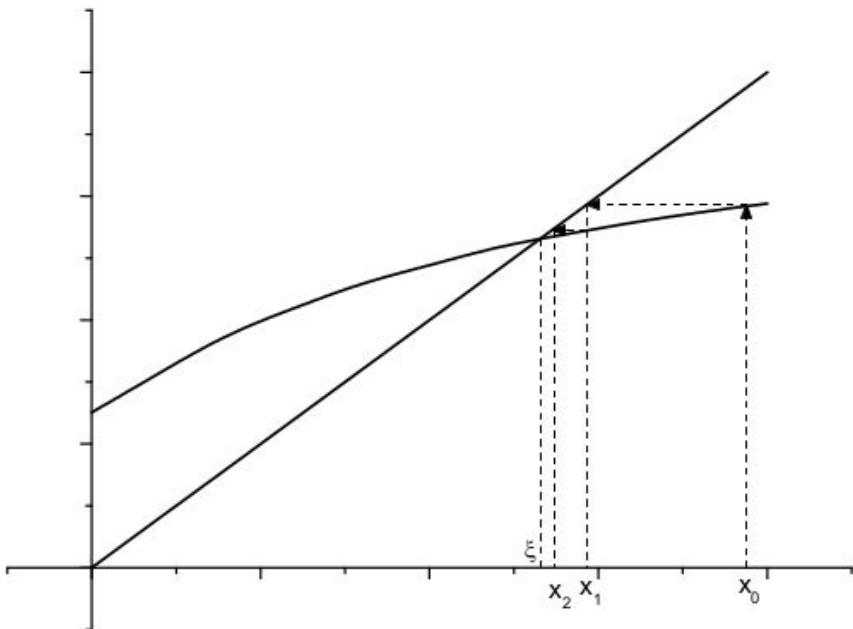
NU \rightarrow se verifică inegalitatea $i > \text{maxiter}$

DA \rightarrow atunci s-a depășit numărul de iterații

NU \rightarrow atunci: $i \rightarrow i + 1, x_0 = x_1$ și se repetă pasul 2.



Cazuri de convergență / divergență



Aplicatie

Folosind metoda iterativă pentru ecuații de forma $x=g(x)$ să se găsească rădăcina ecuației: $3x^4 + 4x - 1 = 0$, cu o eroare $\varepsilon < 10^{-5}$ (cu șase zecimale, ultima fiind rotunjită) știind că această rădăcină se află în intervalul $(0, 1)$.

Rezolvare

Ecuația de mai sus se mai scrie sub forma echivalentă $x=g(x)$ astfel:

$$3x^4 + 4x - 1 = 0 \Leftrightarrow x(3x^3 + 4) = 1 \Leftrightarrow x = \frac{1}{3x^3 + 4} \quad (1.23)$$

Relația de recurență (1.22) pentru acest caz se scrie astfel:

$$x_{n+1} = \frac{1}{3x_n^3 + 4} \quad (1.24)$$

Plecând de la $x_1=0$ și înlocuind în (1.24) se obțin valorile din tabelul 1.6.

Tabelul 1.6

Pas	x_n	x_{n+1}	Eroarea ε
1	0	0,25	0,25
2	0,25	0,2471	0,0029
3	0,2471	0,2472	0,0001
4	0,2472	0,247199	0,000001

Plecând de la $x'_1=1$ și înlocuind în (1.24) se obțin valorile din tabelul 1.7

Tabelul 1.7

Pas	x_n	x_{n+1}	Eroarea ε
1	1	0,142857	0,857143
2	0,142857	0,249454	0,106597
3	0,247123	0,247202	0,002252
4	0,247202	0,247199	0,0000027

Se observă că pentru acest caz metoda este convergentă. O soluție aproximativă a ecuației calculată cu o eroare $\varepsilon < 10^{-5}$ este $\xi = 0,247199$.



C3 - Metode de sortare ale tablourilor unidimensionale

Lect. dr. Ioan Dumitru



Metode de sortare

- Sortarea prin selecție directă
- Sortarea prin inserție
- Metoda bulelor (bubble-sort)
- Sortare cu determinarea poziției finale prin numărare
- Sortarea prin interclasare
- Sortarea rapică (quick-sort)
- Heap-sort
- Sortarea prin numărarea aparițiilor
- Arbore binar de sortare
- Ordonare pe baza cifrelor
- Ordonarea pe grupe



Sortarea prin selecție directă

- În caz de ordonare crescătoare, pornind de la primul element se caută valoarea minimă din tablou. Aceasta se așează pe prima poziție printr-o interschimbare între elementul de pe prima poziție și elementul minim. Apoi, se reia algoritmul, pornind de la a doua poziție și se caută minimul între elementele a_2, \dots, a_n . Acesta se interschimbă cu al doilea dacă este cazul. Procedeul se continuă până la ultimul element.
- Exemplu: fie tabloul $A = (5, 0, 8, 7, 3)$

Pas	Tabloul A	Element minim	Pozitia minimului	Noul tablou A
1	(5, 0, 8, 7, 3)	0	2	(0, 5, 8, 7, 3)
2	(0, 5, 8, 7, 3)	3	5	(0, 3, 8, 7, 5)
3	(0, 3, 8, 7, 5)	5	5	(0, 3, 5, 7, 8)
4	(0, 3, 5, 7, 8)	7	4	

```
Subalgoritm Selecție(n,a)
1: pentru i=1,n-1 execută:
2:   min = a[i]
3:   pentru j=i+1,n execută:
4:     dacă min > a[j] atunci
5:       min = a[j]
6:       k = j
7:     dacă min <> a[i] atunci
8:       aux = a[i]
9:       a[i] = a[k]
10:      a[k] = aux
```

- Deoarece numărul de pași efectuați este $n(n-1)/2$ indiferent de structura datelor de intrare. ordinul de complexitate este $\Theta(n^2)$. Numărul de comparări este tot $n(n-1)/2$.



Sortarea prin inserție

- Este un algoritm eficient pentru sortarea unui număr mic de obiecte.
- Sortarea prin inserție funcționează în același fel în care se sortează un pachet de cărți de joc. Se începe cu pachetul așezat pe masa cu fața în jos și cu mâna stânga goală. Apoi, luam câte o carte de pe masa și o inserăm în poziția corectă în mâna stânga. Pentru a găsi poziția corectă pentru o carte dată, o comparăm cu fiecare dintre cărțile aflate deja în mâna stânga, de la dreapta la stânga (sau de la stânga la dreapta)
- Tabloul este împărțit imaginar în două parti - o parte sortată și o parte nesortată. La început, partea sortată conține primul element al tabloului și partea nesortată conține restul tabloului. La fiecare pas, algoritmul ia primul element din partea nesortată și îl inserează în locul potrivit al partii sortate. Cand partea nesortată nu mai are nici un element, algoritmul se oprește.



Sortarea prin inserție

- Exemplu A =(5, 2, 4, 6, 1, 3)

a1	a2	a3	a4	a5	a5
5	2	4	6	1	3
2	← 5	4	6	1	3
2	4	5	← 6	1	3
2	4	5	6	1	3
1	2	← 4	5	6	3
1	2	3	4	5	← 6

```
void insertion(int a[], int n)
{
    int i, j, aux;
    for (i = 1; i < n; i++)
    {
        j = i;
        while (j > 0 && a[j - 1] > a[j])
        {
            aux = a[j]; a[j] = a[j - 1]; a[j - 1] = aux;
        }
    }
}
```



Metoda bulelor (bubble-sort)

- Algoritmul constă în parcurgerea tabloului A de mai multe ori, până când devine ordonat. La fiecare pas se compară două elemente alăturate.
- Dacă $a_i > a_{i+1}$, ($i = 1, 2, \dots, n - 1$), atunci cele două valori se interschimbă între ele.
- Controlul acțiunii repetitive este dat de variabila booleană ok, care la fiecare reluare a algoritmului primește valoarea inițială adevărat, care se schimbă în fals dacă s-a efectuat o interschimbare de două elemente alăturate. În momentul în care tabloul A s-a parcurs fără să se mai efectueze nici o schimbare, ok rămâne cu valoarea inițială adevărat și algoritmul se termină, deoarece tabloul este ordonat.
- Interschimbarea a două elemente se realizează prin intermediul variabilei auxiliare aux care are același tip ca și elementele tabloului.

$a1 = 0.0$	$a2 = 1.1$	$a3 = 1.0$	$a4 = 1.2$	$a5 = 0.08$	ok
0.0	↔ 1.0	1.1	1.2	0.08	fals
0.0	1.0	1.1	↔ 0.08	1.2	fals

0.0	1.0	↔ 0.08	1.1	1.2	fals
-----	-----	--------	-----	-----	------

0.0	↔ 0.08	1.0	1.1	1.2	fals
-----	--------	-----	-----	-----	------

0.0	0.08	1.0	1.1	1.2	adevărat
-----	------	-----	-----	-----	----------



Metoda bulelor (bubble-sort)

- Singura permutare fără inversiuni este cea ordonată (identică) 1,2,...,n.
- Cazul cel mai favorabil este acela în care datele inițiale sunt ordonate crescător, caz în care se face o singură parcurgere a datelor.
- Cazul cel mai nefavorabil este cel în care datele sunt sortate descrescător, caz în care se vor face $n-1$ parcurgeri. La prima parcurgere se vor face $n-1$ interschimbări, la a doua $n-2$ și aşa mai departe. Așadar numărul de comparații și cel de interschimbări va fi $n(n-1)/2$.

```
void sortareBule(vector<int>& v) {
    bool ordonat = false;
    while (!ordonat) {
        ordonat = true; //presupunem sirul este ordonat
        for (int i = 1; i < v.size(); i++) {
            if (v[i - 1] > v[i]) {
                //interschimbam
                int aux = v[i - 1];
                v[i - 1] = v[i];
                v[i] = aux;
                ordonat = false;
            }
        }
    }
}
```



Sortare cu determinarea poziției finale prin numărare

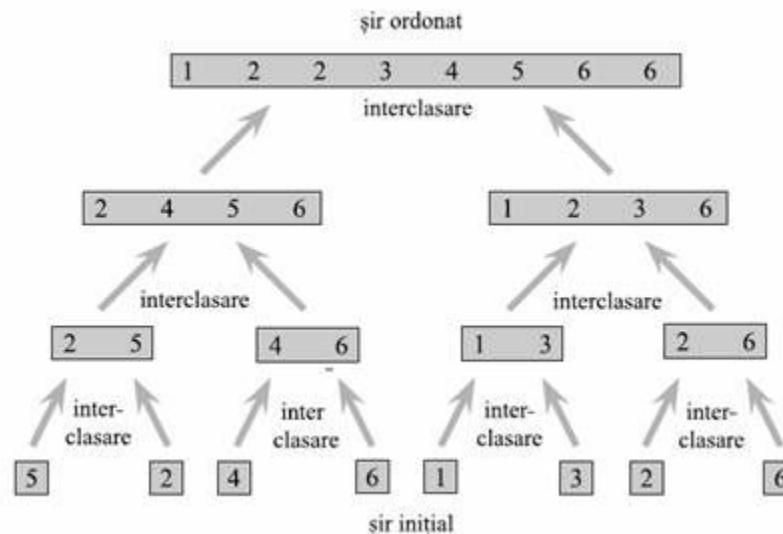
- Această metodă constă în construirea unui nou tablou B care are aceeași dimensiune ca și tabloul A în care depunem elementele din A, ordonate crescător.
- Vom lua fiecare element și îl vom compara cu fiecare alt element din sir pentru a putea reține în variabila k numărul elementelor care sunt mai mici decât elementul considerat. Astfel, vom afla poziția pe care trebuie să-l punem pe acesta în sirul B. Dacă în problemă avem nevoie de sirul ordonat tot în tabloul A, vom copia în A întreg tabloul B.

```
void sortareNumarare(vector& v) {
    //se face o copie a sirului
    vector v2 = v;
    for (int i = 0; i < v.n; i++) {
        //numaram cate elemente sunt mai mici
        int nrMaiMici = 0;
        for (int j = 0; j < v.n; j++) {
            if (v2.e[j] < v2.e[i]) {
                nrMaiMici++;
            }
        }
        v.e[nrMaiMici] = v2.e[i];
    }
}
```



Sortarea prin interclasare

- Mulți algoritmi utili au o structură recursivă: pentru a rezolva o problema dată, aceștia sunt apelați de către ei însăși o data sau de mai multe ori pentru a rezolva subprobleme apropriate.
- Acești algoritmi folosesc de obicei o abordare de tipul divide și stăpânește: ei rup problema de rezolvat în mai multe probleme similare problemei inițiale, dar de dimensiune mai mică, le rezolvă în mod recursiv și apoi le combina pentru a crea o soluție a problemei inițiale.
- Pentru metoda de sortare prin interclasare principiul divide și stăpânește poate fi privit astfel:
 - **Divide:** Împarte sirul de n elemente care urmează a fi sortat în două subșiruri de câte $n/2$ elemente.
 - **Stăpânește:** Sortează recursiv cele două subșiruri utilizând sortarea prin interclasare.
 - **Combina:** Interclasează cele două subșiruri sortate pentru a produce rezultatul final.



Interclasare

- Fiind date două șiruri de numere, ordonate crescător (sau descrescător), se cere să se obțină un șir care să fie de asemenea ordonat crescător (respectiv descrescător) și care să fie formată din elementele șirurilor date.
- Șirul rezultat se poate obține direct (fără o sortare a șirului final) prin parcurgerea secvențială a celor două șiruri, simultan cu generarea șirului cerut. Prin compararea a două elemente din liste de intrare se va decide care element va fi adăugat în lista de ieșire.

```
void adaugaSf(vector& v, int x) {
    v.e[v.n] = x;
    v.n++;
}

void interclasare(vector v1, vector v2, vector& rez) {
    rez.n = 0;
    int i = 0;
    int j = 0;
    while (i < v1.n && j < v2.n) {
        if (v1.e[i] <= v2.e[j]) {
            adaugaSf(rez, v1.e[i]);
            i++;
        } else {
            adaugaSf(rez, v2.e[j]);
            j++;
        }
    }
    for (int k = i; k < v1.n; k++) {
        adaugaSf(rez, v1.e[k]);
    }
    for (int k = j; k < v2.n; k++) {
        adaugaSf(rez, v2.e[k]);
    }
}
```



Bibliografie

- <http://info.mcip.ro/?t=ts>



Rezolvarea sistemelor de ecuatii

Lect. dr. Ioan Dumitru



- Sisteme de ecuații liniare

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \quad \dots \quad \vdots \quad \ddots \quad \dots \quad \vdots \quad \dots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{array} \right| \Leftrightarrow A\mathbf{x} = \mathbf{b}$$

- A matricea sistemului, cu \mathbf{x} vectorul necunoscutele și cu \mathbf{b} vectorul format din termenii liberi ai sistemului

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \dots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ și } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

- Sistemul are un număr de m ecuații cu n necunoscute



- Daca

- $n = m$: numarul de ecuatii este egal cu numarul de necunoscute. Sistemul are solutie unica daca si numai daca matricea A este nesingulara, adica:

$$\det A \neq 0$$

Daca $\det A = 0$ atunci sistemul de ecuatii este degenerat si poate avea o infinitate de solutii sau nici una.

- $m < n$: sistemul este subdeterminat si nu exista solutie unica sau nu are solutie.
 - $m > n$: sistemul este supradeterminat si de regula nu are solutii. Daca insa rang $A = n$ atunci este posibil ca sistemul de mai sus sa aiba o solutie.



- Metodele numerice ce vor fi descrise in continuare sunt propuse pentru un sistem de n ecuatii cu n necunoscute si pot fi clasificate astfel:
 - **metode directe:** solutia aproximativa a sistemului este calculata direct
 - **metode indirecte:** solutia sistemului liniar este calculata prin aproximatii succesive (metode iterative)



Sisteme inferior triunghiulare

- Aceste sisteme sunt de forma următoare:

$$\begin{cases} a_{11}x_1 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n = b_n \end{cases}$$

- Pentru rezolvarea sistemului se aplică metoda substituției. Se calculează soluțiile cu formulele:

$$x_1 = \frac{b_1}{a_{11}} \quad \text{și} \quad x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j}{a_{ii}} \quad i=1,2,3,\dots,n$$



Sisteme superior triunghiulare

- Sistemele de tipul superior triunghiular sunt de forma:

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1 \\ a_{22}x_2 + a_{21}x_1 + a_{23}x_3 + \dots + a_{2n}x_n = b_2 \\ a_{33}x_3 + a_{31}x_1 + a_{32}x_2 + \dots + a_{3n}x_n = b_3 \\ \dots \dots \dots \\ a_{nn}x_n = b_n \end{array} \right.$$

- Calculul soluțiilor sistemului se fac printr-o retrosubstituție cu ajutorul următoarelor formule:

$$x_n = \frac{b_n}{a_{nn}} \quad \text{și} \quad x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}} \quad i=n-1, n-2, \dots, 1$$



Eliminarea Gauss

- Metoda eliminarii Gauss permite rescrierea matricii sistemului sub forma triunghiulara astfel incat, in final se obtine solutia aproximativa a sistemului de ecuatii liniare printr-o substitutie iterativa a necunoscutelor.
- Vom nota cu \tilde{A} matricea extinsa unui sistem de ecuatii liniare formata din matricea sistemului A completata cu inca o coloana a termenilor liberi b. Fara a restrange generalitatea vom aplica algoritmul eliminarii Gauss pentru un sistem de 3 ecuatii cu 3 necunoscute
- In relatiile de mai sus s-a considerat ca toate elementele de pe diagonala principala sunt diferite de zero. Daca acest lucru nu se intampla se realizeaza operatia de pivotare pana se obtin coeficienti nenuli pe diagonala principala.

$$\begin{aligned}\tilde{A} &= \begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix} \left[\begin{array}{c|c} \cdot \frac{1}{a_{11}} & (-a_{21}) \\ \hline + & + \end{array} \right] \left[\begin{array}{c|c} & (-a_{31}) \\ \hline + & + \end{array} \right] \\ &\rightsquigarrow \begin{pmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & b_3^{(1)} \end{pmatrix} \left[\begin{array}{c|c} \cdot \frac{1}{a_{22}^{(1)}} & (-a_{32}^{(1)}) \\ \hline + & + \end{array} \right] \\ &\rightsquigarrow \begin{pmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & b_1^{(1)} \\ 0 & 1 & a_{23}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(2)} & b_3^{(2)} \end{pmatrix} \left[\begin{array}{c|c} \cdot \frac{1}{a_{33}^{(2)}} & \\ \hline & \end{array} \right] \\ &\rightsquigarrow \begin{pmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & b_1^{(1)} \\ 0 & 1 & a_{23}^{(2)} & b_2^{(2)} \\ 0 & 0 & 1 & b_3^{(3)} \end{pmatrix}.\end{aligned}$$



Exemplu

1. Să se găsească parabola $y = A + Bx + Cx^2$ care trece prin punctele $(1, 4)$, $(2, 7)$ și $(3, 14)$.

Rezolvare:

Total revine de fapt la rezolvarea sistemului de ecuații:

$$(25) \quad \begin{cases} A + B + C = 4 \\ A + 2B + 4C = 7 \\ A + 3B + 9C = 14 \end{cases}$$

Pentru comoditate vom lucra cu matricea extinsă a sistemului:

$$\left[\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 1 & 2 & 4 & 7 \\ 1 & 3 & 9 & 14 \end{array} \right] \rightarrow \left[\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 0 & 1 & 3 & 3 \\ 0 & 2 & 8 & 10 \end{array} \right] \rightarrow \left[\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 0 & 1 & 3 & 3 \\ 0 & 0 & 2 & 4 \end{array} \right] \rightarrow \left[\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 0 & 1 & 3 & 3 \\ 0 & 0 & 1 & 2 \end{array} \right]$$

În faza substituției avem de rezolvat sistemul:

$$\begin{cases} A + B + C = 4 \\ B + 3C = 3 \\ C = 2 \end{cases}$$

Rezultă că $A = 5$, $B = -3$ și $C = 2$, parabola căutată fiind:

$$y = 5 - 3x + 2x^2$$



Formule de calcul

- pentru linia pivot

$$\begin{cases} a_{kk}^{(k)} = 1 \\ a_{kj}^{(k)} = a_{kj}^{(k-1)} / a_{kk}^{(k-1)}, \quad j = \overline{k+1, n} \\ b_k^{(k)} = b_k^{(k-1)} / a_{kk}^{(k-1)} \end{cases}$$

- pentru liniile de sub linia pivot

$$\begin{cases} a_{ik}^{(k)} = 0 \\ a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k-1)}, \quad i, j = \overline{k+1, n} \\ b_i^{(k)} = b_i^{(k-1)} - a_{ik}^{(k-1)} b_k^{(k)} \end{cases}$$



Rezolvarea sistemului trunghiular

- Dupa eliminarea Gauss se obtine un sistem de ecuatii liniare cu o matrice sistem triunghiulara (elementele de sub diagonala principala sunt egale cu 0) si calculul necunoscutelor se realizeaza printr-o operatie de substitutie inversa:

$$\begin{cases} x_n &= b_n^{(n)} \\ x_k &= b_k^{(k)} - \sum_{i=k+1}^n a_{ki}^{(k)} x_i, \quad k = \overline{n-1, 1} \end{cases}$$

- Este de remarcat faptul ca determinantul matricii sistemului poate fi calculat ca:

$$\det A = a_{11}a_{22}^{(1)} \cdots a_{nn}^{(n-1)}$$



În cursul efectuării operațiilor din eliminarea Gauss la fiecare pas k se împarte linia pivot la elementul diagonal $a_{kk}^{(k-1)}$. Este posibil ca acest element să fie 0 și trebuie ca să se realizeze o rearanjare a elementelor din matricea sistemului astfel încât să se evite o asemenea situație. Această operație se numește *pivotare*. Mai mult, dacă este necesar să se realizeze pivotarea este bine ca noul element pivot să fie cel mai mare pentru că erorile de rotunjire la împărțire sunt mai mici. De obicei se utilizează metoda *pivotării parțiale pe coloane* astfel încât la pasul k se caută elementul maxim al matricii $A^{(k-1)}$ situat pe coloana k și liniile $l \geq k$ (pe și sub diagonala principală) și se schimbă liniile l și k între ele.



Eliminarea Gauss - Jordon

- Are acelasi principiu ca eliminarea Gauss dar operatiile cu linia pivot se efectueaza si pentru liniile de deasupra liniei pivot rezultand astfel o matrice de tip unitate completata cu termenii liberi.
- Exemplu: sa se rezolve sistemul:

$$\begin{cases} x + y + z = 5 \\ 2x + 3y + 5z = 8 \\ 4x + 5z = 2 \end{cases}$$

- Matricea extinsa este:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array} \right]$$

- si efectuam operatiile cu linia pivot

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array} \right] \xrightarrow{R_2-2R_1} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 4 & 0 & 5 & 2 \end{array} \right]$$
$$\xrightarrow{R_3-4R_1} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array} \right]$$



Eliminarea Gauss - Jordon

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array} \right] \xrightarrow{R_3+4R_2} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{array} \right] \xrightarrow{\frac{1}{13}R_3} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{array} \right]$$

$$\xrightarrow{R_2-3R_3} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array} \right] \xrightarrow{R_1-R_3} \left[\begin{array}{ccc|c} 1 & 1 & 0 & 7 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array} \right]$$

$$\xrightarrow{R_1-R_2} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array} \right]$$

$$x = 3, \quad y = 4, \quad z = -2.$$



Eliminarea Gauss - Jordon - infinitate de solutii

- Exemplu: sa se rezolve sistemul:

$$\begin{cases} 4y + z = 2 \\ 2x + 6y - 2z = 3 \\ 4x + 8y - 5z = 4 \end{cases}$$

- Matricea extinsa este:

$$\left[\begin{array}{ccc|c} 0 & 4 & 1 & 2 \\ 2 & 6 & -2 & 3 \\ 4 & 8 & -5 & 4 \end{array} \right]$$

- si efectuam operatiile cu linia pivot

$$\left[\begin{array}{ccc|c} 0 & 4 & 1 & 2 \\ 2 & 6 & -2 & 3 \\ 4 & 8 & -5 & 4 \end{array} \right] \xrightarrow{R_1 \leftrightarrow R_2} \left[\begin{array}{ccc|c} 2 & 6 & -2 & 3 \\ 0 & 4 & 1 & 2 \\ 4 & 8 & -5 & 4 \end{array} \right]$$

$$\xrightarrow{R_3 - 2R_1} \left[\begin{array}{ccc|c} 2 & 6 & -2 & 3 \\ 0 & 4 & 1 & 2 \\ 0 & -4 & -1 & -2 \end{array} \right]$$

$$\xrightarrow{R_3 + R_2} \left[\begin{array}{ccc|c} 2 & 6 & -2 & 3 \\ 0 & 4 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

$$\xrightarrow{\frac{1}{4}R_2} \left[\begin{array}{ccc|c} 2 & 6 & -2 & 3 \\ 0 & 1 & 1/4 & 1/2 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

$$\xrightarrow{R_1 - 6R_2} \left[\begin{array}{ccc|c} 2 & 0 & -7/2 & 0 \\ 0 & 1 & 1/4 & 1/2 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

$$\xrightarrow{\frac{1}{2}R_1} \left[\begin{array}{ccc|c} 1 & 0 & -7/4 & 0 \\ 0 & 1 & 1/4 & 1/2 \\ 0 & 0 & 0 & 0 \end{array} \right]$$



- Matricea corespunde sistemului:

$$\begin{cases} x - \frac{7}{4}z = 0 \\ y + \frac{1}{4}z = \frac{1}{2} \end{cases}$$

- Se poate exprima solutia sistemului ca:

$$x = \frac{7}{4}z, \quad y = \frac{1}{2} - \frac{1}{4}z.$$

- sau luand o valoare arbitrara pentru z

$$x = \frac{7}{4}t, \quad y = \frac{1}{2} - \frac{1}{4}t, \quad z = t$$

- pentru $t = 4$

$$(x, y, z) = (7, -\frac{1}{2}, 4)$$



- În metoda factorizării LU matricea sistemului A se scrie sub forma unui produs de două matrici:

$$A = L \cdot U$$

- unde L este o matrice triunghiulară inferioară și U este o matrice triunghiulară superioară:

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ \alpha_{21} & 1 & \dots & 0 & 0 \\ \vdots & \dots & \ddots & 0 & 0 \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{n,n-1} & 1 \end{pmatrix} \quad U = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1,n-1} & \beta_{1n} \\ 0 & \beta_{22} & \vdots & \beta_{2,n-1} & \beta_{2n} \\ 0 & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & \beta_{nn} \end{pmatrix}$$

- Cu aceste notări sistemul de ecuații liniare devine echivalent cu rezolvarea a două sisteme de ecuații cu matrici triunghiulare

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

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



Factorizarea LU. Metoda Crout

- În metoda Crout se efectuează operația de înmulțire între matricile L și U și elementele matricii produs sunt identificate cu elementele matricii sistemului A. Se obțin astfel formulele pentru calculul matricilor L și U:

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik} \beta_{kj}, \quad i = \overline{1, j}$$

$$\alpha_{ij} = \frac{1}{\beta_{jj}} \left(a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik} \beta_{kj} \right), \quad i = \overline{j+1, n}$$

- celor două matrici triunghiulare sunt memorate drept noi elemente ale matricii A

$$A_{final} = \begin{pmatrix} \beta_{11} & \dots & \dots & \beta_{1n} \\ \alpha_{21} & \ddots & & \beta_{2n} \\ \vdots & & \ddots & \vdots \\ \alpha_{n1} & \dots & \alpha_{n,n-1} & \beta_{nn} \end{pmatrix}$$

- Determinantul matricii se obține ca:

$$\det A = \prod_{j=1}^n \beta_{jj}$$



Rezolvarea sistemelor trunghiulare se face prin metoda substitutiei

$$\begin{cases} y_1 = b_1 \\ y_i = b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j, \quad i = \overline{2, n} \end{cases}$$

$$\begin{cases} x_n = \frac{y_n}{\beta_{nn}} \\ x_i = \frac{1}{\beta_{ii}} \left(y_i - \sum_{j=i+1}^n \beta_{ij} x_j \right), \quad i = \overline{n-1, 1} \end{cases}$$



Metoda de rezolvare a sistemelor tridiagonale

- Un tip special de sisteme liniare îl reprezintă sistemele tridiagonale care au diagonala principală și diagonalele vecine diferite de zero

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

- O metoda simplă de rezolvare a acestor probleme constă în descompunerea matricii tridiagonale în două matrice L , U astfel incat

$$LU = A$$

- sau devoltat:

$$\begin{pmatrix} l_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ p_2 & l_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & p_3 & l_3 & \dots & 0 & 0 & 0 \\ - & - & - & - & - & - & - \\ 0 & 0 & 0 & \dots & p_{n-1} & l_{n-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & p_n & l_n \end{pmatrix} \begin{pmatrix} 1 & u_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & u_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & u_3 & \dots & 0 & 0 \\ - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & \dots & 1 & u_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_n & b_n \end{pmatrix}$$



Rezolvare sistemelor tridiagonale

- Făcând înmulțirea matricelor L, U și identificând cu matricea A se obțin relațiile de recurență pentru calculul elementelor matricelor L și U:

$$p_i = a_i$$

$$u_i = \frac{c_i}{l_i}$$

$$l_1 = b_1$$

$$l_i = b_i - a_i u_{i-1}$$

- Condiția de existență a relațiilor este $l_i \neq 0 \quad i = 1, 2, \dots, n$
- Sistemul initial poate fi scris sub forma

$$LUX=D$$

sau sub forma a două sisteme

$$\begin{cases} L \cdot Y = D \\ U \cdot X = Y \end{cases}$$

- Cunoaștem pe L și D, deci se calculează mai întâi Y



Rezolvare sistemelor tridiagonale

$$\begin{pmatrix} l_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & l_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & l_3 & \dots & 0 & 0 & 0 \\ - & - & - & - & - & - & - \\ 0 & 0 & 0 & \dots & a_{n-1} & l_{n-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & a_n & l_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

- Prin identificarea în se deduc următoarele relații recursive de calcul :

$$\begin{cases} y_1 = \frac{d_1}{l_1} \\ y_i = \frac{(d_i - a_i y_{i-1})}{t_i}, \quad i=2,3,\dots,n \end{cases}$$

- Din sistemul $UX=Y$ scris detaliat :

$$\begin{pmatrix} 1 & u_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & u_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & u_3 & \dots & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & \dots & 1 & u_{n-1} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & u_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

- rezultă relațiile recursive de calcul ale rădăcinilor. Prin retrosubstituție se obțin:

$$\begin{cases} x_n = y_n \\ x_i = y_i - s_i x_i + 1, \quad i=n-1,\dots,1 \end{cases}$$

care reprezintă rădăcinile sistemului tridiagonal. (Rusu)



- O metoda de tip iterativ pleaca de la o aproximare de ordin zero a solutiei problemei si prin transformari successive se obtine un sir de solutii care sa convearg catre solutia exacta.
- **1. Metoda iterativa Jacobi**

Sistemul de n ecuatii liniare cu n necunoscute se rescrie sub forma:

$$\begin{cases} x_1 &= t_1 + s_{12}x_2 + \dots + s_{1n}x_n \\ x_2 &= s_{21}x_1 + t_2 + \dots + s_{2n}x_n \\ \vdots &\dots\dots\dots \\ x_n &= s_{n1}x_1 + s_{n2}x_2 + \dots + t_n \end{cases}$$

- sau rescris matricial

$$\mathbf{x} = S \cdot \mathbf{x} + \mathbf{t}$$

- cu

$$\begin{cases} s_{ii} &= 0, i = \overline{1, n} \\ s_{ij} &= -a_{ij}/a_{ii}, j = \overline{1, n}, j \neq i \\ t_i &= b_i/a_{ii} \end{cases}$$



Metoda iterativa Jacobi

- Drept aproximatie de ordin 0 pentru solutia sistemului se alege, de obicei, coloana termenilor liberi:

$$\mathbf{x}^{(0)} = \mathbf{t}$$

iar sirul aproximatiilor succesive este

$$\mathbf{x}^{(k)} = S \cdot \mathbf{x}^{(k-1)} + \mathbf{t}$$

- Daca acest sir este convergent atunci limita sa este solutia sistemului de ecuatii.
- In practica, metoda lui Jacobi urmeaza urmatorul algoritm de calcul:

$$\begin{cases} \Delta_i^{(k)} &= \sum_{j=1}^n s_{ij} x_j^{(k-1)} + t_i \\ x_i^{(k)} &= x_i^{(k-1)} + \Delta_i^{(k)}, \quad i = \overline{1, n} \end{cases}$$

- Procedeul iterativ se continua pana cand eroarea relativa maxima a componentelor solutiei devine mai mica decat o eroare maxima admisa:

$$\max_i \left| \Delta_i^{(k)} / x_i^{(k)} \right| \leq \varepsilon$$



Metoda iterativa Jacobi

- Exista cateva cazuri cu in care se pot da conditii suficiente pentru convergenta catre solutia exacta. Astfel daca este valabila una din urmatoarele relatii:

$$\max_i \left| \Delta_i^{(k)} / x_i^{(k)} \right| \leq \varepsilon$$

atunci procesul iterativ converge indiferent de alegerea aproximatiei initiale. Totusi, metoda Jacobi este convergenta chiar daca nu sunt indeplinite conditiile de mai sus

$$\sum_{j=1}^n |s_{ij}| < 1, \quad i = \overline{1, n}$$

$$\sum_{i=1}^n |s_{ij}| < 1, \quad j = \overline{1, n}$$

si, uneori, este suficient ca matricea sistemului sa fie dominant diagonală:

$$|a_{ii}| > \max_{j \neq i} |a_{ij}|, \quad i = \overline{1, n}$$



Metoda Gauss-Seidel

- Metoda Gauss-Seidel difera fata de metoda iterativa Jacobi doar prin faptul ca suma care intra in calculul valorilor pentru $\Delta_i^{(k)}$ este rescrisa prin folosirea valorilor $x_j^{(k)}$ cunoscute deja la momentul calculului din iteratia anterioara

$$\begin{cases} \Delta_i^{(k)} &= \sum_{j=1}^{i-1} s_{ij}x_j^{(k)} + \sum_{j=i}^n s_{ij}x_j^{(k-1)} + t_i \\ x_i^{(k)} &= x_i^{(k-1)} + \Delta_i^{(k)}, \quad i = \overline{1, n} \end{cases}$$

- Unul din avantajele metodei Gauss-Seidel este si acela ca aceasta metoda converge intotdeauna pentru un sistem normal, indiferent de alegerea aproximatiei initiale.

Un sistem liniar $Ax = \mathbf{b}$ se numește normal dacă matricea coeficienților A este simetrică și pozitiv definită:

$$\begin{aligned} A &= A^T \\ \mathbf{x} \cdot A \cdot \mathbf{x} &> 0 \quad (\forall) \mathbf{x} \text{ din spațiul vectorial respectiv.} \end{aligned}$$



- Teoremă: Orice matrice pătratică A poate fi descompusă sub forma $A=QR$ cu Q matrice ortogonală ($Q^TQ = I$) și R matrice superior triunghiulară
- Ideea:
 - Fie un sistem de forma $Ax=b$.
 - A se descompune astfel $A=QR$ (conform teoremei) printr-o metodă dintre cele prezentate în continuare.
 - Rezulta sistemul $QRx=b$
 - Se înmulțește cu Q^T la stânga: $Q^TQRx=Q^Tb$, $Q^TQ = I$ = matricea identitate
 - Obținem $Rx=b_1$ ($b_1=Q^Tb$)
 - Rezolvarea sistemului superior triunghiular este trivială, fiind studiată într-un laborator anterior.
- Labmn05



Matrici ortogonale

- Fie vectorii coloană:

$$x, y \in R^{n \times 1}, x = [x_1; x_2; \dots; x_n], y = [y_1; y_2; \dots; y_n]$$

- Se definesc următoarele operații:

- Produsul scalar:

$$\langle x, y \rangle = \sum_{i=1}^n x_i \cdot y_i = y' \cdot x$$

- Norma euclidiană:

$$\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x' \cdot x} = \sqrt{\langle x, x \rangle}$$

- Vectorii z_1, z_2, \dots, z_n sunt ortogonali dacă:

$$1). \langle z_i, z_j \rangle = 0, \forall i \neq j$$

$$2). \|z_i\|_2 = 1, \forall i = 1:n$$

- O matrice este ortogonală dacă și numai dacă coloanele sale formează o bază ortonormată.



- Proprietati ale matricilor ortogonale (H apartine $R^{n \times m}$)

$$1). H \cdot H^t = H^t \cdot H = I_n$$

$$2). H^t = H^{-1}$$

$$3). \|Hx\|_2 = \|x\|_2$$



Metode de factorizare - Metoda Householder

- Metoda de triunghiularizare ortogonală Householder transformă sistemul $Ax=b$, $A \in \mathbb{R}^{m \times n}$, într-un sistem ortogonal echivalent $HAx = Hb$, cu matricea sistemului HA superior triunghiulară, $H \in \mathbb{R}^{m \times m}$ fiind o matrice ortogonală. Matricea ortogonală H se construiește ca un produs de *reflectori Householder*: $H=H_n \dots H_2 H_1$; un reflector elementar Householder este de forma:

$$H_p = I_m - 2 \frac{\mathbf{v}_p \mathbf{v}_p^T}{\mathbf{v}_p^T \mathbf{v}_p}$$

- În care componentele vectorului Householder $\mathbf{v}_p = [0 \dots 0 \ v_{pp} \ \dots v_{mp}]^T$ se obțin cu relațiile:

$$\sigma_p = \text{sign}(a_{pp}) \sqrt{\sum_{i=p}^m a_{ip}^2}$$

$$v_{pp} = a_{pp} + \sigma_p, \quad \beta_p = \sigma_p v_{pp}, \quad v_{ip} = a_{ip}, \quad i > p$$

- Inmulțirea $HA = H_n \dots H_2 H_1 A$ se scrie $A_{p+1} = H_p A_p$, $p=1:n$, $A_1 = A$. Datorită formei particulare a reflectorilor, înmulțirile matriciale nu se fac efectiv. Astfel înmulțirea $A_{p+1} = H_p A_p$ se face astfel:

- coloanele $j=1:p-1$ rămân neschimbate
- în coloana p : a_{ip} cu $i < p$ rămân neschimbate, $a_{pp} = -\sigma_p$, $a_{ip} = 0$, $i > p$
- în coloanele $j=p+1:n$, a_{ij} cu $i < p$ rămân neschimbate,

$$a_{ij} = a_{ij} - \tau_j v_{ip}, \quad i > p, \quad \tau_j = \frac{\sum_{i=p}^m v_{ip} a_{ij}}{\beta_p}$$



Exemplu

- Determinați factorizarea QR reflector Householder a matricei

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix}$$

- Solutie

$$a_1 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \quad \sigma_1 = \|a_1\|_2 = 3, \quad u_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \|u_1\|_2^2 = 6;$$

$$H_1 = I_3 - \frac{2}{\|u_1\|_2^2} u_1 u_1^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & 2 \\ -2 & 2 & 4 \end{bmatrix} =$$

$$= \frac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix}$$

$$A_2 = H_1 A_1 = \frac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 0 & 3 \\ 0 & 3 & 3 \end{bmatrix}$$



Exemplu p2

• :

$$\bar{a}_2 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \quad \|\bar{a}_2\|_2 = 3, \quad u_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \quad \|u_2\|_2^2 = 2;$$

$$H_2 = I_3 - \frac{2}{\|u_2\|_2^2} u_2 u_2^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$A_3 = R = H_2 A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 3 & 3 & 3 \\ 0 & 0 & 3 \\ 0 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

$$Q = (H_2 H_1)^T = H_1 H_2 = \frac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}.$$



Metode de factorizare QR - Metoda Gram-Schmidt

- Consideram vectorii corespunzatori coloanelor matricii A

$$A = \left[\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{array} \right]$$

- si

$$\mathbf{u}_1 = \mathbf{a}_1, \quad \mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|},$$

$$\mathbf{u}_2 = \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1, \quad \mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}.$$

$$\mathbf{u}_{k+1} = \mathbf{a}_{k+1} - (\mathbf{a}_{k+1} \cdot \mathbf{e}_1)\mathbf{e}_1 - \cdots - (\mathbf{a}_{k+1} \cdot \mathbf{e}_k)\mathbf{e}_k, \quad \mathbf{e}_{k+1} = \frac{\mathbf{u}_{k+1}}{\|\mathbf{u}_{k+1}\|}.$$

- rezulta

$$A = \left[\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{array} \right] = \left[\begin{array}{c|c|c|c} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \end{array} \right] \left[\begin{array}{cccc} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_n \end{array} \right] = QR.$$



Exemplu pentru metode de factorizare Gram-Schmidt

- Fie

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

atunci vectorii coloana sunt: $\mathbf{a}_1 = (1, 1, 0)^T$, $\mathbf{a}_2 = (1, 0, 1)^T$, $\mathbf{a}_3 = (0, 1, 1)^T$

- pentru simplitate renuntam la notatia transpus dar vom retine ca vectorii \mathbf{a} sunt de tip coloana

$$\mathbf{u}_1 = \mathbf{a}_1 = (1, 1, 0),$$

$$\mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} = \frac{1}{\sqrt{2}}(1, 1, 0) = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right),$$

$$\mathbf{u}_2 = \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1 = (1, 0, 1) - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right) = \left(\frac{1}{2}, -\frac{1}{2}, 1\right),$$

$$\mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} = \frac{1}{\sqrt{3/2}}\left(\frac{1}{2}, -\frac{1}{2}, 1\right) = \left(\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}\right),$$



Exemplu la metode de factorizare Gram-Schmidt



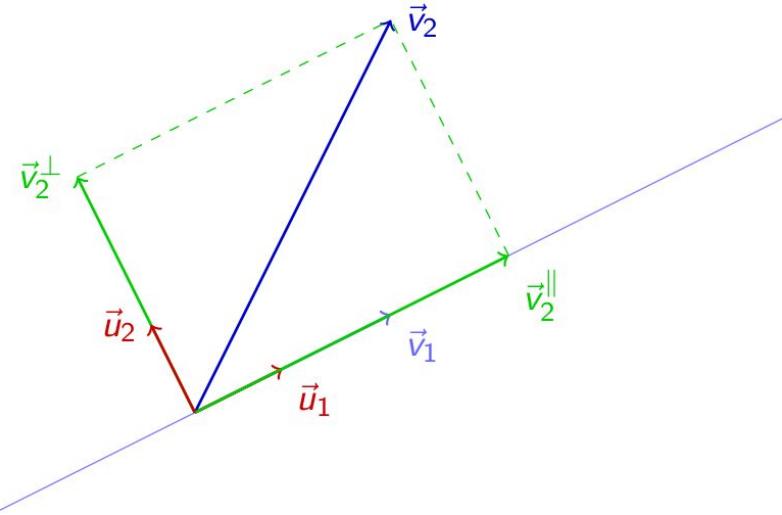
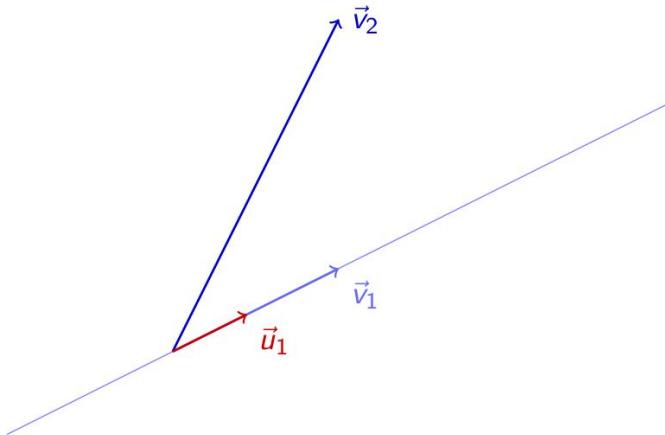
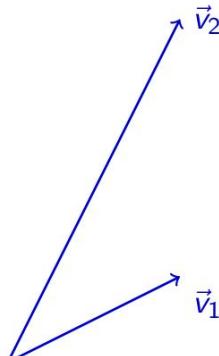
$$\begin{aligned}\mathbf{u}_3 &= \mathbf{a}_3 - (\mathbf{a}_3 \cdot \mathbf{e}_1)\mathbf{e}_1 - (\mathbf{a}_3 \cdot \mathbf{e}_2)\mathbf{e}_2 \\ &= (0, 1, 1) - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right) - \frac{1}{\sqrt{6}}\left(\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}\right) = \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) \\ \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} = \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right).\end{aligned}$$

$$Q = \left[\mathbf{e}_1 \mid \mathbf{e}_2 \mid \cdots \mid \mathbf{e}_n \right] = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{bmatrix},$$

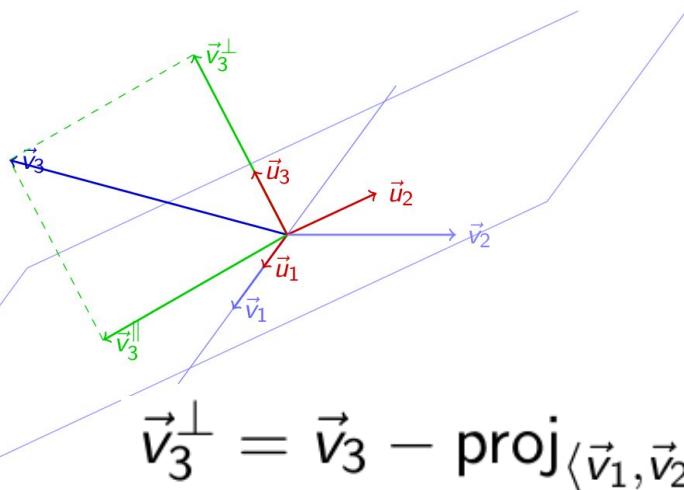
$$R = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \mathbf{a}_3 \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \mathbf{a}_3 \cdot \mathbf{e}_2 \\ 0 & 0 & \mathbf{a}_3 \cdot \mathbf{e}_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{3}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{bmatrix}$$



Interpretare geometrică



$$\vec{v}_2^\perp = \vec{v}_2 - \text{proj}_{\langle \vec{v}_1 \rangle} \vec{v}_2 = \vec{v}_2 - (\vec{u}_1 \cdot \vec{v}_2) \vec{u}_1$$



$$\vec{v}_3^\perp = \vec{v}_3 - \text{proj}_{\langle \vec{v}_1, \vec{v}_2 \rangle} \vec{v}_3 = \vec{v}_3 - (\vec{u}_1 \cdot \vec{v}_3) \vec{u}_1 - (\vec{u}_2 \cdot \vec{v}_3) \vec{u}_2$$



- Cateva lucruri importante pot fi calculate usor daca avem reprezentarea QR, in special pentru matrici patratice

- Deoarece $Q^{-1} = Q^T$ avem:

$$A^{-1} = R^{-1}Q^T$$

si R^{-1} poate fi calculata mult mai usor decat A^{-1}

- Deoarece $\det Q = +/- 1$ atunci

$$\det A = +/- \det R$$

- Valorile proprii ale matricii A pot fi calculate usor daca A se poate factoriza QR



Exercitiu la metoda de factorizare Gram-Schmidt

- Gasiti reprezentarea QR pentru matricea

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

- Solutia este:

$$Q = \begin{bmatrix} 1/2 & 1/2 & -1/2 & -1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \end{bmatrix} \quad R = \begin{bmatrix} 2 & 1 & 1 & 3/2 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & 1 & -1/2 \\ 0 & 0 & 0 & 1/2 \end{bmatrix}$$





În fizică apar foarte des probleme de vectori proprii și valori proprii. Cel mai cunoscut exemplu este rezolvarea ecuației lui Schrödinger:

$$H\Psi = E\Psi \quad (2.1)$$

unde H este hamiltonianul sistemului, E este energia iar Ψ funcția de undă care descrie starea respectivă. Problema se poate generaliza astfel încât se definește noțiunea de *vector propriu* al unui operator liniar A ca fiind un vector \mathbf{x} nenul care prin acțiunea operatorului A se transformă într-un vector proporțional cu \mathbf{x} :

$$A\mathbf{x} = \lambda\mathbf{x} \quad (2.2)$$



Calculul valorilor și vectorilor proprii

În acest caz scalarul λ se numește *valoare proprie* corespunzătoare vectorului propriu \mathbf{x} . Pentru cazul unui spațiu vectorial n-dimensional cu scalari reali (sau din mulțimea numerelor complexe) operatorul A poate fi structurat ca o matrice $n \times n$ și ecuația (2.2) poate fi rescrisă sub forma:

$$(A - \lambda I) \mathbf{x} = 0 \quad (2.3)$$

care este un sistem omogen de n ecuații cu n necunoscute. Un astfel de sistem admite soluții nebanale dacă și numai dacă determinantul sistemului este egal cu zero, adică:

$$\det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \vdots & \dots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{vmatrix} = 0 \quad (2.4)$$

Ecuația (2.4) este o ecuație algebrică în λ de grad n și are n rădăcini complexe. Pentru fiecare valoare proprie λ_j sistemul (2.3) admite soluția $\mathbf{x}^{(j)}$. Toți vectorii proprii pot fi organizați ca o matrice X (numită *matrice modală*):



Calculul valorilor și vectorilor proprii

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ \vdots & \dots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(n)} \end{pmatrix} \quad (2.5)$$

astfel încât vectorii proprii sunt coloane ale acestei matrici și ecuația (2.4) se rescrie:

$$A \cdot X = X \cdot \Lambda \quad (2.6)$$

cu:

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix} \quad (2.7)$$

Dacă matricea X este nesingulară ($\det X \neq 0$) atunci există X^{-1} și relația (2.6) devine:

$$X^{-1} \cdot A \cdot X = \Lambda \quad (2.8)$$

Se observă că matricea valorilor proprii Λ se obține din matricea A printr-o *transformare similară*¹. În plus, două matrici similare admit aceleasi valori proprii.

¹Două matrici A și B se numesc similare dacă există o matrice nesingulară S astfel încât:

$$B = A^{-1} \cdot A \cdot S.$$



Exemplu sistem de vectori și valori proprii

Să se găsească vectorii și valoările proprii pentru matricea

$$A = \begin{bmatrix} 10 & 2 & 1 \\ 2 & 10 & 1 \\ 2 & 1 & 10 \end{bmatrix}$$

Nu... $A \mathbf{x} = \lambda \mathbf{x}$

$$\begin{bmatrix} 10 & 2 & 1 \\ 2 & 10 & 1 \\ 2 & 1 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\det(A - \lambda I) = 0$$

$$\begin{vmatrix} 10-\lambda & 2 & 1 \\ 2 & 10-\lambda & 1 \\ 2 & 1 & 10-\lambda \end{vmatrix} = 0$$



Exemplu sistem de vectori și valori proprii

$$(10-\lambda)^3 + 2 + 4 - 2(10-\lambda) - (10-\lambda) - 4(10-\lambda) = 0$$

$$(10-\lambda)^3 - 7(10-\lambda) + 6 = 0$$

$$10-\lambda = \lambda \Rightarrow \lambda_1 = 9$$

$$10-\lambda = -3 \Rightarrow \lambda_2 = 13$$

$$10-\lambda = 2 \Rightarrow \lambda_3 = 8$$

Substituind $\lambda_1 = 9$:

$$\left\{ \begin{array}{l} x_1 + 2x_2 + x_3 = 0 \\ 2x_1 + x_2 + x_3 = 0 \\ 2x_1 + x_2 + x_3 = 0 \end{array} \right. \quad \left| \begin{array}{l} \\ \\ \end{array} \right. \quad x_1 = x_2$$

$$\text{Dacă } x_3 = t \Rightarrow x_1^{(1)} = -\frac{t}{3}, \quad x_2^{(1)} = -\frac{t}{3}, \quad x_3^{(1)} = t$$

$$\text{Pentru } t = 1: \quad x_1^{(1)} = -\frac{1}{3}, \quad x_2^{(1)} = -\frac{1}{3}, \quad x_3^{(1)} = 1$$



Exemplu sistem de vectori și valori proprii

$$\lambda_2 = 13 \Rightarrow \begin{cases} -3x_1 + 2x_2 + x_3 = 0 \cdot (-1) \\ 2x_1 - 3x_2 + x_3 = 0 \quad \text{+} \\ 2x_1 + x_2 - 3x_3 = 0 \end{cases} \Rightarrow x_1 = x_2 \\ x_1 = x_3$$

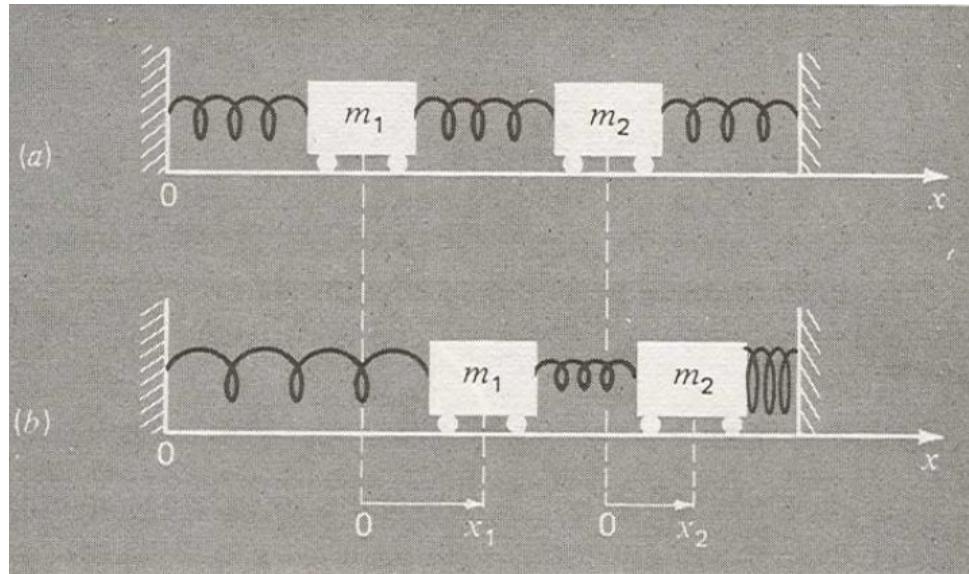
$$x_3 = t \Rightarrow x_1^{(2)} = t, \quad x_2^{(2)} = t, \quad x_3^{(2)} = t$$

$$x_3 = 8 \Rightarrow \begin{array}{l} 2x_1 + 2x_2 + x_3 = 0 \\ 2x_1 + 2x_2 + x_3 = 0 \quad (-1) \\ 2x_1 + x_2 + 2x_3 = 0 \quad \text{+} \end{array} \Rightarrow x_2 = x_3, \quad x_1 = -\frac{3}{2}x_3$$

$$x_3 = t \Rightarrow x_1^{(3)} = -\frac{3}{2}t, \quad x_2^{(3)} = t, \quad x_3^{(3)} = t$$



- Se consideră sistemul



- Ecuatiile de miscare ale corpurilor sunt:

$$m_1 \frac{d^2x_1}{dt^2} = -kx_1 + k(x_2 - x_1)$$

$$m_2 \frac{d^2x_2}{dt^2} = -k(x_2 - x_1) - kx_2$$

$$m_1 \ddot{x}_1 - k(-2x_1 + x_2) = 0$$

$$m_2 \ddot{x}_2 - k(x_1 - 2x_2) = 0$$

- Solutiile sistemului de cauta de forma:

$$x_i = A_i \sin(\omega t)$$

- si substituind:

$$\left(\frac{2k}{m_1} - \omega^2 \right) A_1 - \frac{k}{m_1} A_2 = 0$$
$$- \frac{k}{m_2} A_1 + \left(\frac{2k}{m_2} - \omega^2 \right) A_2 = 0$$

- Particularizare:

$$m_1 = m_2 = 40 \text{ kg} \quad k = 200 \text{ N/m}$$

$$(10 - \omega^2) A_1 - 5 A_2 = 0$$

$$- 5 A_1 + (10 - \omega^2) A_2 = 0$$



- Determinantul sistemului este 0

$$(\omega^2)^2 - 20\omega^2 + 75 = 0$$

- cu solutiile:

$$\omega_1 = 3.873 \text{ / s}$$

$$\omega_2 = 2.236 \text{ / s}$$

- ce corespund la doua perioade/moduri de oscilatie

$$t_1 = 1.62 \text{ s}$$

$$t_2 = 2.81 \text{ s}$$

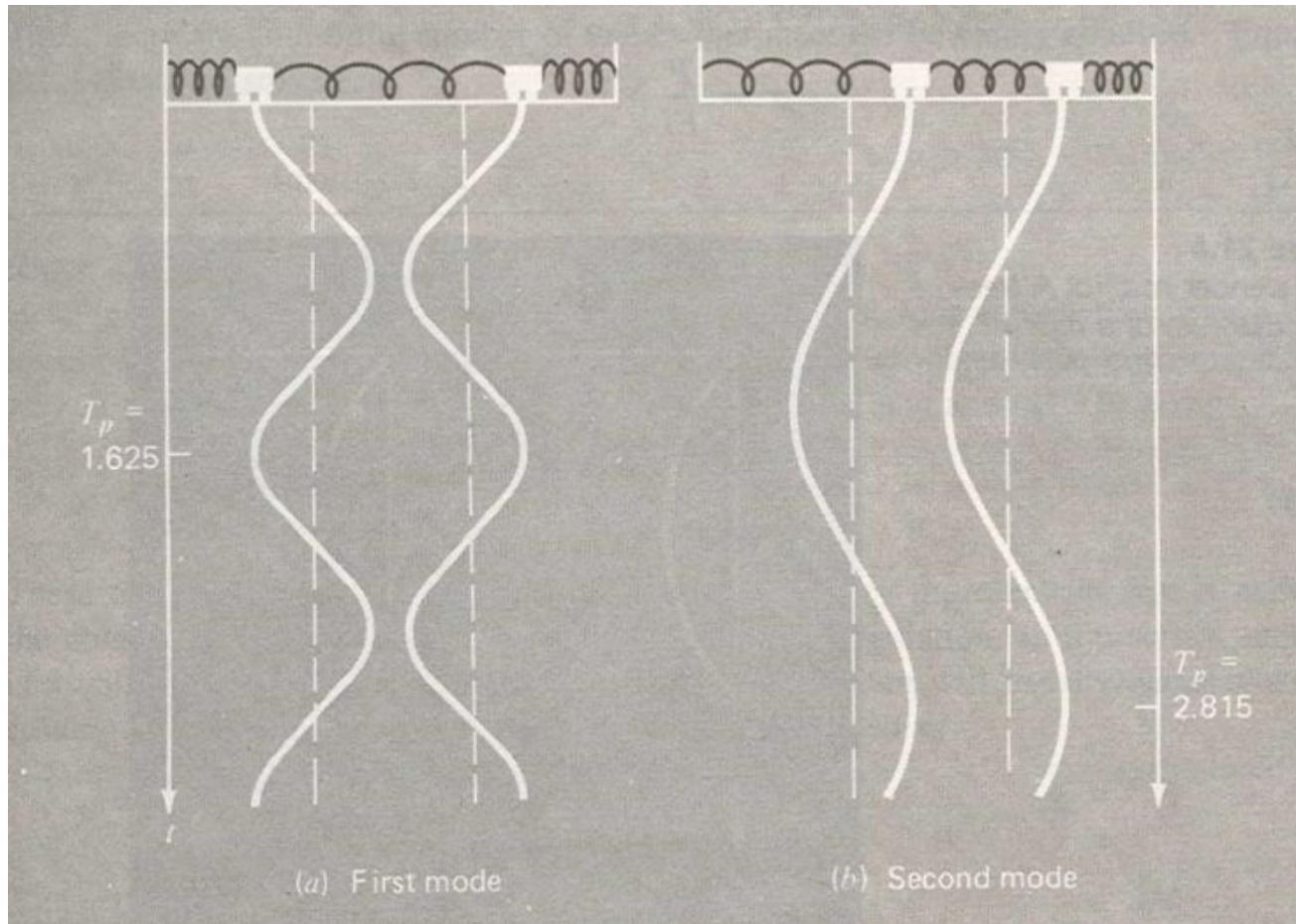
- Pentru aceste moduri avem

$$A_1 = - A_2$$

$$A_1 = A_2$$



- Moduri de oscilatie: in faza si in antifaza



Această metodă permite calculul valorilor proprii ale unei matrice $[A]$ pe baza dezvoltării polinomului caracteristic $D(\lambda)$ cu ajutorul *formulelor lui Newton pentru sumele puterilor rădăcinilor unei ecuații polinomiale*. Determinarea valorilor proprii constă atât în calculul primelor n puteri ale matricei $[A]$ cât și a sumelor termenilor aflați pe diagonala principală a acestor matrice.

Determinantul caracteristic al matricei $[A]$ se scrie sub forma polinomului:

$$D(\lambda) = \det([A] - \lambda[I])_n = (-1)^n (\lambda^n + k_1\lambda^{n-1} + k_2\lambda^{n-2} + k_3\lambda^{n-3} + \dots + k_n) \quad (4.86)$$

Se notează cu s_m suma puterilor de ordinul m ale rădăcinilor polinomului caracteristic (4.86):

$$\begin{aligned} s_m &= \lambda_1^m + \lambda_2^m + \dots + \lambda_n^m \\ m &= 1, 2, 3, \dots, n \end{aligned} \quad (4.87)$$

Formulele lui Newton pentru sumele puterilor de ordinul m ale rădăcinilor în cazul polinomului caracteristic (4.86) se scriu:

$$\begin{aligned} s_m + k_1s_{m-1} + k_2s_{m-2} + \dots + k_{m-1}s_1 &= -k_m \\ m &= 1, 2, 3, \dots, n \end{aligned} \quad (4.88)$$



Metoda Leverrier

Dacă se cunosc sumele puterilor rădăcinilor de ordinul m ale polinomului caracteristic (4.86), atunci sistemul (4.88) permite determinarea coeficienților k_1, k_2, \dots, k_n astfel:

$$\begin{cases} -k_1 = s_1 \\ -2k_2 = s_2 + k_1 s_1 \\ -3k_3 = s_3 + k_1 s_2 + k_2 s_1 \\ \cdots \\ -nk_n = s_n + k_1 s_{n-1} + k_2 s_{n-2} + \dots + k_{n-1} s_1 \end{cases} \quad (4.89)$$

Se poate demonstra că sumele puterilor rădăcinilor de ordinul m ale polinomului caracteristic al unei matrice $[A]$ reprezintă urmele matricilor $[A]^m$:

$$s_m = \lambda_1^m + \lambda_2^m + \dots + \lambda_n^m = \sum_{i=1}^n a_{ii}^{(m)} \quad (4.90)$$

unde $a_{ii}^{(m)}$ sunt termenii de pe diagonala principală a matricei $[A]^m$:

$$[A]^m = [a_{ij}^{(m)}] \quad m = 2, 3, \dots, n \quad (4.91)$$

matricile $[A]^m$ se determină astfel:

$$[A]^m = [A]^{m-1}[A], \quad m = 2, 3, \dots, n \quad (4.92)$$



Aplicatie - Metoda Leverrier

Folosind metoda *Leverrier* să se determine valorile proprii ale matricei $[A]$:

$$[A] = \begin{bmatrix} 7 & -2 & 0 \\ -2 & 6 & -2 \\ 0 & -2 & 5 \end{bmatrix} \quad (4.93)$$

Rezolvare

Se determină matricele $[A]^2$ și $[A]^3$ astfel:

$$\begin{aligned} [A]^2 &= \begin{bmatrix} 53 & -26 & 4 \\ -26 & 44 & -22 \\ 4 & -22 & 29 \end{bmatrix} \\ [A]^3 &= \begin{bmatrix} 423 & -270 & 72 \\ -270 & 360 & -198 \\ 72 & -198 & 189 \end{bmatrix} \end{aligned} \quad (4.94)$$

Sumele s_m ale puterilor rădăcinilor de ordinul m ($m=1,2,3$) ale polinomului caracteristic $D(\lambda)$ se determină folosind relațiile (4.90) :

$$\begin{cases} s_1 = \lambda_1 + \lambda_2 + \lambda_3 = \sum_{i=1}^3 a_{ii}^{(1)} = 18 \\ s_2 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 = \sum_{i=1}^3 a_{ii}^{(2)} = 126 \\ s_3 = \lambda_1^3 + \lambda_2^3 + \lambda_3^3 = \sum_{i=1}^3 a_{ii}^{(3)} = 972 \end{cases} \quad (4.95)$$



Aplicatie - Metoda Leverrier

Coeficienții polinomului caracteristic k_1 , k_2 și k_3 se determină folosind relațiile (4.89)

$$\begin{cases} m_1 = -s_1 = -18 \\ m_2 = -\frac{1}{2}(s_2 + m_1 s_1) = 99 \\ m_3 = -\frac{1}{3}(s_3 + m_1 s_2 + m_2 s_1) = -162 \end{cases} \quad (4.95)$$

Se obține astfel ecuația caracteristică a matricei $[A]$:

$$\lambda^3 - 18\lambda^2 + 99\lambda - 162 = 0 \quad (4.96)$$

Rezolvând ecuația (4.4.11) se obțin valorile proprii ale matricei $[A]$:

$$\lambda_1 = 3; \quad \lambda_2 = 6; \quad \lambda_3 = 9. \quad (4.97)$$

- Mai mult $A^{-1} = [A^2 - 18 A + 99 I] / 162$



Metoda Leverrier - Faddeev

- Fie λ radacina polinomului de ordin n $P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$
$$P(\lambda) = \lambda^n + c_1 \lambda^{n-1} + c_2 \lambda^{n-2} + \dots + c_{n-2} \lambda^2 + c_{n-1} \lambda + c_n$$
- Metoda Faddeev-Levarrier ofera un algoritm eficient de calcul a coeficientilor caracteristici ai polinomului ai permite determinarea facila a inversei matricii A.
- Trasa matricii A este

$$\text{Tr}[\mathbf{A}] = a_{1,1} + a_{2,2} + \dots + a_{n,n}$$

- Algoritmul genereaza o secventa de matrici $\{\mathbf{B}_k\}_{k=1}^n$ si le utilizeaza in calculul coeficientilor polinomului.

$$\mathbf{B}_1 = \mathbf{A} \quad \text{and} \quad p_1 = \text{Tr}[\mathbf{B}_1]$$

$$\mathbf{B}_2 = \mathbf{A} (\mathbf{B}_1 - p_1 \mathbf{I}) \quad \text{and} \quad p_2 = \frac{1}{2} \text{Tr}[\mathbf{B}_2]$$

$$\vdots \qquad \qquad \dots \qquad \vdots$$

$$\mathbf{B}_k = \mathbf{A} (\mathbf{B}_{k-1} - p_{k-1} \mathbf{I}) \quad \text{and} \quad p_k = \frac{1}{k} \text{Tr}[\mathbf{B}_k]$$

$$\vdots \qquad \qquad \dots \qquad \vdots$$

$$\mathbf{B}_n = \mathbf{A} (\mathbf{B}_{n-1} - p_{n-1} \mathbf{I}) \quad \text{and} \quad p_n = \frac{1}{n} \text{Tr}[\mathbf{B}_n]$$



Metoda Leverrier - Faddeev

- Astfel polinomul caracteristic se scrie ca

$$P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - p_2 \lambda^{n-2} - \dots - p_{n-1} \lambda - p_n$$

- iar inversei matricii A

$$A^{-1} = \frac{1}{p_n} (B_{n-1} - p_{n-1} I)$$

- Exemplu: Sa se gaseasca polinomul caracteristic si matricea inversa pentru

$$A = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix}$$



Exemplu - metoda Leverrier - Faddeev

$$B_1 = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix}$$

$$p_1 = \text{Tr}[B_1] = 6$$

$$B_2 = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix} \left(\begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix} - (6) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) = \begin{pmatrix} -6 & 1 & -3 \\ 3 & -8 & -3 \\ -1 & 1 & -8 \end{pmatrix}$$

$$p_2 = \frac{1}{2} \text{Tr}[B_2] = \frac{1}{2} (-22) = -11$$

$$B_3 = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix} \left(\begin{pmatrix} -6 & 1 & -3 \\ 3 & -8 & -3 \\ -1 & 1 & -8 \end{pmatrix} - (-11) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) = \begin{pmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{pmatrix}$$

$$p_3 = \frac{1}{3} \text{Tr}[B_3] = \frac{1}{3} (18) = 6$$

$$P[\lambda] = \lambda^3 - \sum_{i=1}^3 p_i \lambda^{n-i}$$

$$P[\lambda] = -6 + 11\lambda - 6\lambda^2 + \lambda^3$$



Exemplu - metoda Leverrier - Faddeev

- Efectuand calculele direct obtinem acelasi polinom caracteristic

$$A = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix} \quad M = \begin{pmatrix} 2-\lambda & -1 & 1 \\ -1 & 2-\lambda & 1 \\ 1 & -1 & 2-\lambda \end{pmatrix}$$

$$Q[\lambda] = \text{Det}[M] = 6 - 11\lambda + 6\lambda^2 - \lambda^3$$
$$\lambda \rightarrow 1$$
$$\lambda \rightarrow 2$$
$$\lambda \rightarrow 3$$

- Inversa matricii A este se obtine:

$$A^{-1} = \frac{1}{p_3} (B_{3-1} - p_{3-1} I)$$

$$A^{-1} = \frac{1}{6} \left(\begin{pmatrix} -6 & 1 & -3 \\ 3 & -8 & -3 \\ -1 & 1 & -8 \end{pmatrix} - (-11) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right)$$

$$A^{-1} = \frac{1}{6} \left(\begin{pmatrix} -6 & 1 & -3 \\ 3 & -8 & -3 \\ -1 & 1 & -8 \end{pmatrix} - \begin{pmatrix} -11 & 0 & 0 \\ 0 & -11 & 0 \\ 0 & 0 & -11 \end{pmatrix} \right)$$

$$A^{-1} = \frac{1}{6} \begin{pmatrix} 5 & 1 & -3 \\ 3 & 3 & -3 \\ -1 & 1 & 3 \end{pmatrix}$$



Metoda lui Jacobi

- Toate metodele pentru aflarea valorilor proprii și vectorilor proprii folosesc transformările similare pentru a realiza diagonalizarea matricilor iar elementele de pe diagonala principală sunt chiar valorile proprii.

Metoda lui Jacobi se folosește pentru diagonalizarea matricilor simetrice ($A^T = A$) și folosește transformări similare prin utilizarea unor *matrici ortogonale*²

$$R^T \cdot A \cdot R = \Lambda \quad (2.9)$$

Transformările similare cu matrici ortogonale se numesc *transformări de similitudine*. Matricile ortogonale care se folosesc în metoda Jacobi generalizează matricile de rotație și au forma:

$$R(i,j) = \begin{pmatrix} & \text{col. } i & \text{col. } j \\ 1 & \vdots & \dots & \vdots & 0 \\ \dots & \cos \varphi & \dots & -\sin \varphi & \dots \\ & \vdots & \ddots & \vdots & \\ \dots & \sin \varphi & \dots & \cos \varphi & \dots \\ 0 & \vdots & \dots & \vdots & 1 \end{pmatrix} \quad \begin{array}{l} | \text{linia } i \\ | \text{linia } j \end{array} \quad (2.10)$$

²o matrice ortogonală R este o matrice inversabilă definită de relația:

$$R \cdot R^T = R^T \cdot R = I \Leftrightarrow R^{-1} = R^T$$



Metoda lui Jacobi

Se realizează transformarea de similitudine:

$$A' = R^T(i, j) A R(i, j) \quad (2.11)$$

și se impune ca elementele extradiagonale ale matricii A' să se anuleze. În acest fel se obține:

$$\begin{cases} a'_{ik} = a'_{ki} = a_{ik} \cos \varphi + a_{jk} \sin \varphi, & k = \overline{1, n} \\ a'_{jk} = a'_{kj} = -a_{ik} \sin \varphi + a_{jk} \cos \varphi, & k \neq i, j \\ a'_{ii} = a_{ii} \cos^2 \varphi + 2a_{ji} \sin \varphi \cos \varphi + a_{jj} \sin^2 \varphi \\ a'_{jj} = a_{ii} \sin^2 \varphi - 2a_{ji} \sin \varphi \cos \varphi + a_{jj} \cos^2 \varphi \\ a'_{ij} = a'_{ji} = a_{ji} (\cos^2 \varphi - \sin^2 \varphi) + (a_{jj} - a_{ii}) \sin \varphi \cos \varphi \end{cases} \quad (2.12)$$

și impunând ca elementele $a'_{ij} = a'_{ji}$ să se anuleze rezultă valoarea unghiului de rotație φ :

$$\tan \varphi = \left[\frac{a_{ii} - a_{jj}}{2a_{ji}} \pm \sqrt{\left(\frac{a_{ii} - a_{jj}}{2a_{ji}} \right)^2 + 1} \right]^{-1} \quad (2.13)$$

de unde se calculează:

$$\cos \varphi = \frac{1}{\sqrt{1 + \tan^2 \varphi}}, \quad \sin \varphi = \cos \varphi \tan \varphi \quad (2.14)$$

Metoda lui Jacobi

La fiecare pas de calcul în algoritmul Jacobi se efectuează transformările de similitudine prin înmulțirea cu matricea $R_l = R(i_l, j_l)$ astfel încât să anuleze elementele extradiagonale de la intersecția liniilor și coloanelor i_l, j_l . Definim astfel matricile ortogonale:

$$X_l = R_0 R_1 \cdots R_l, \quad l = \overline{0, 1, 2, \dots} \quad (2.15)$$

Şirul transformărilor ortogonale va fi:

$$A_0 = A, \quad A_1 = X_1^T \cdot A \cdot X_1, \dots, \quad A_l = X_l^T \cdot A \cdot X_l, \dots \quad (2.16)$$

Conform relației de recurență:

$$X_l = X_{l-1} \cdot R_l(i, j) \quad (2.17)$$

se obține:

$$\begin{cases} x'_{ki} &= x_{ki} \cos \varphi + x_{kj} \sin \varphi, \quad k = \overline{1, n} \\ x'_{kj} &= -x_{ki} \sin \varphi + x_{kj} \cos \varphi \end{cases} \quad (2.18)$$

Şirul iterațiilor este întrerupt când valoarea absolută a elementelor extradiagonale ale matricii A_l sunt mai mici decât $\varepsilon > 0$ ales:

$$\max_{i \neq j} |a'_{ij}| \leq \varepsilon \quad (2.19)$$





Sistemele de ecuații neliniare sunt relații de forma:

$$(5.1) \quad \begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \dots \dots \dots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

unde f_1, \dots, f_n sunt funcții de neliniare de variabile reale x_1, \dots, x_n . Sistemul (5.1) se poate scrie sub formă condensată (matriceală):

$$(5.2) \quad f(X) = 0 \quad \text{cu } f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \text{ și } X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$



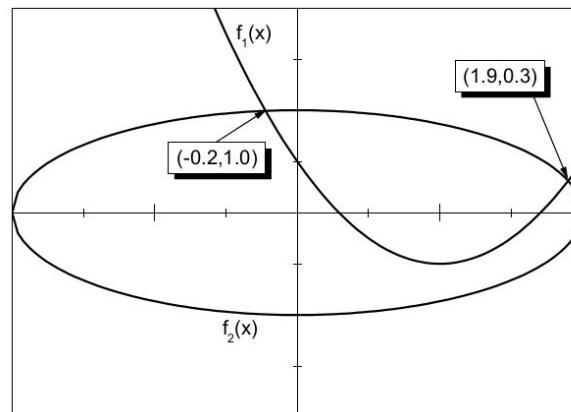
Sisteme de ecuații neliniare

Exemplu: Un sistem neliniar de 2 ecuații cu două necunoscute pentru care $f_1 = x^2 - 2x - y + 0.5$ și $f_2 = x^2 + 4y^2 - 4$ unde $x_1 = x$, $x_2 = y$. Rezolvarea unui astfel de sistem are și o interpretare geometrică: soluția sistemului corespunde punctelor din planul $x - y$ corespunzătoare intersecției celor două curbe definite de f_1 (parabola) și f_2 (elipsa) reprezentate în figura 5.1 În general un sistem de relații de tipul

$$\begin{cases} u &= f_1(x, y) \\ v &= f_2(x, y) \end{cases}$$

reprezintă o transformare a punctelor din planul (x, y) în planul (u, v) . În fizică astfel de transformări sunt utilizate de exemplu pentru coordonatele polare plane.

Sistemele de ecuații neliniare se pot rezolva numeric utilizând metode iterative (metoda iterăției de punct fix sau metoda Newton).



Metoda Newton pentru rezolvarea sistemelor de ecuații neliniare

Metoda Newton pentru rezolvarea sistemelor de ecuații neliniare este o generalizare a metodei iterative Newton pentru rezolvarea ecuației lor neliniare. Pentru a folosi această metodă sistemul de ecuații (5.1) sau (5.2) trebuie rescris sub formă iterativă echivalentă:

$$x^{(p)} = G(x^{(p-1)}, x^{(p-2)}, \dots, x^{(0)})$$

adică aproximarea de ordin (p) este exprimată printr-o relație între aproximările de ordin $(0), \dots, (p-1)$. Aici aproximarea de ordin (p) : $x^{(p)}$ corespunde la n valori approximative:

$$x^{(p)} = \begin{pmatrix} x_1^{(p)} \\ \vdots \\ x_n^{(p)} \end{pmatrix}$$



Metoda Newton pentru rezolvarea sistemelor de ecuații neliniare



În acest fel soluția exactă a sistemului de ecuații neliniare x_s se scrie sub forma:

$$(5.3) \quad x_s = x^{(p)} + \epsilon^{(p)}$$

unde $\epsilon^{(p)}$ este corecția (termenul de eroare) corespunzătoare soluției aproximative $x^{(p)}$ încât:

$$f(x^{(p)} + \epsilon^{(p)}) = 0$$

Relația de mai sus se dezvoltă în jurul valorii $x^{(p)}$:

(5.4)

$$\left\{ \begin{array}{l} 0 = f_1(x^{(p)} + \epsilon^{(p)}) = \frac{\partial f_1}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_1}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_1}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} + f_1(x^{(p)}) \\ 0 = f_2(x^{(p)} + \epsilon^{(p)}) = \frac{\partial f_2}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_2}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_2}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} + f_2(x^{(p)}) \\ \dots \\ 0 = f_n(x^{(p)} + \epsilon^{(p)}) = \frac{\partial f_n}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_n}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_n}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} + f_n(x^{(p)}) \end{array} \right.$$



Metoda Newton pentru rezolvarea sistemelor de ecuații neliniare

obținându-se un sistem liniar de ecuații cu necunoscutele $\epsilon_1^{(p)}, \dots, \epsilon_n^{(p)}$:

$$(5.5) \quad \left\{ \begin{array}{l} \frac{\partial f_1}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_1}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_1}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} = -f_1(x^{(p)}) \\ \frac{\partial f_2}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_2}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_2}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} = -f_2(x^{(p)}) \\ \dots \\ \frac{\partial f_n}{\partial x_1} \Big|_{x^{(p)}} \epsilon_1^{(p)} + \frac{\partial f_n}{\partial x_2} \Big|_{x^{(p)}} \epsilon_2^{(p)} + \dots + \frac{\partial f_n}{\partial x_n} \Big|_{x^{(p)}} \epsilon_n^{(p)} = -f_n(x^{(p)}) \end{array} \right.$$

sau sub formă matriceală:

$$(5.6) \quad J(f)|_{x^{(p)}} \epsilon^{(p)} = -f(x^{(p)})$$

unde:

$$(5.7) \quad J(f) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

este *matricea Jacobiană* pentru f . Dacă matricea $J(f)|_{x^{(p)}}$ este inversabilă (este nesin-



Metoda Newton pentru rezolvarea sistemelor de ecuații neliniare



gulară) atunci există $J^{-1}(f)|_x^{(p)}$ și din (5.6) rezultă:

$$(5.8) \quad \epsilon^{(p)} = - J^{-1}(f)|_x^{(p)} f(x^{(p)}) \quad \text{și}$$

$$(5.9) \quad x^{(p+1)} = x^{(p)} - J^{-1}(f)|_x^{(p)} f(x^{(p)})$$

Relația (5.9) este relația de iterare pentru metoda Newton de rezolvare a sistemelor de ecuații neliniare.

Observație: Relația (5.9) este echivalentă cu relația care se aplică în cazul metodei Newton pentru calculul soluției aproximative a unei ecuații neliniare (3.15) cu $J^{-1}(f)|_{x_i} = \frac{1}{f'(x_i)}$





Pentru calculul soluțiilor aproximative a unui sistem de ecuații neliniare prin metoda Newton se urmăresc următorii pași (inclusi într-un algoritm iterativ):

1. se evaluatează:

$$f(x^{(k)}) = \begin{pmatrix} f_1(x_1^{(k)}, \dots, x_n^{(k)}) \\ \vdots \\ f_n(x_1^{(k)}, \dots, x_n^{(k)}) \end{pmatrix}$$

2. se calculează matricea Jacobiană în $x^{(k)}$ conform relației (5.7)
3. se rezolvă sistemul de n ecuații liniare cu n necunoscute $\epsilon_1^{(k)}, \dots, \epsilon_n^{(k)}$ dat de relația (5.6);
4. se calculează aproximarea de ordin $(k + 1)$:

$$x^{(k+1)} = x^{(k)} + \epsilon^{(k)}$$



Interpolarea polinomială a funcțiilor reale

Lect. dr. Ioan Dumitru



În aproape toate lucrările experimentale de fizică se pune problema măsurării sau determinării prin calcul a unor mărimi fizice într-un număr finit de puncte. Pentru simplitate vom considera că se cunosc două șiruri de numere reale x_i și y_i cu $i = \overline{1, n}$ între care există o dependență funcțională, adică există o funcție de variabilă o variabilă reală pentru care are loc:

$$y_i \approx f(x_i) \quad (4.1)$$

Condițiile experimentale, de obicei, nu permit cunoașterea funcției $f : [a, b] \rightarrow \mathbb{R}$ în mod analitic încât să se cunoască valoarea în orice punct x_c din intervalul de definiție. Din acest motiv, pe baza măsurătorilor experimentale, și prin alegerea convenabilă a tipului de funcție f (clasă de continuitate, derivabilitate, model teoretic, etc.) se poate determina valoarea $y_c \approx f(x_c)$ în orice punct din $[a, b]$.



Principiul metodei

Se consideră funcția reală de variabilă reală $f : [a, b] \rightarrow \mathbb{R}$ ale cărei valori sunt cunoscute într-o rețea de noduri $a = x_0 < x_1 < \dots < x_n = b$:

$$\begin{array}{cccccc} x : & x_0 & x_1 & \dots & x_n \\ y : & y_0 & y_1 & \dots & y_n \end{array}$$

în care $y_k = f(x_k)$.

Problema fundamentală a interpolării constă în determinarea unei funcții $g : [a, b] \rightarrow \mathbb{R}$, care aproximează funcția f satisfăcând condițiile: $g(x_0) = y_0, g(x_1) = y_1, \dots, g(x_n) = y_n$. De obicei funcția g este căutată de forma unei combinații liniare

$$g(x) = \sum_{k=0}^n c_k b_k(x) \tag{11.1}$$

de funcții $b_k : [a, b] \rightarrow \mathbb{R}$, numite funcții de bază. Dacă în general problema interpolării nu are soluție unică, prin alegerea funcțiilor de bază problema este bine formulată și are soluție unică, cu condiția ca aceste funcții să fie liniar independente. În acest caz, problema interpolării se reduce la determinarea coeficienților c_0, c_1, \dots, c_n , care alcătuiesc vectorul $\mathbf{c} = [c_0, c_1, \dots, c_n]^T \in \mathbb{R}^{(n+1)}$.



Metoda clasică

Alegând funcțiile de bază de forma $1, x, x^2, \dots, x^n$, respectiv $b_k = x^k$, *funcția de interpolare este un polinom:*

$$g(x) = \sum_{k=0}^n c_k x^k,$$

de gradul n care satisface *condițiile de interpolare:*

$$g(x_k) = y_k, \quad k = 0, 1, 2, \dots, n.$$

Există deci o legătură strânsă între gradul polinomului de interpolare și numărul de puncte ale tabelului de valori, și anume gradul polinomului este cu 1 mai mic decât numărul de puncte din tabel (prin două puncte trece o dreaptă, prin trei puncte trece o parabolă, etc.).

În consecință, coeficienții polinomului de interpolare satisfac sistemul de ecuații algebrice liniare:

$$\begin{aligned} c_0 + c_1 x_0 + c_2 x_0^2 + \cdots + c_n x_0^n &= y_0 \\ c_0 + c_1 x_1 + c_2 x_1^2 + \cdots + c_n x_1^n &= y_1 \\ &\vdots \\ c_0 + c_1 x_n + c_2 x_n^2 + \cdots + c_n x_n^n &= y_n \end{aligned} \tag{11.2}$$



Metoda clasică

sau sub forma matriceală:

$$\mathbf{A}\mathbf{c} = \mathbf{y},$$

în care $\mathbf{y} = [y_0, y_1, \dots, y_n]^T \in \mathbb{R}^{(n+1)}$ iar

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \in \mathbb{R}^{(n+1) \times (n+1)} \quad (11.3)$$

este o matrice nesingulară, dacă $x_i \neq x_j$ pentru $i \neq j$.

În acest fel, problema interpolării presupune parcurgerea etapelor:

- determinarea coeficienților polinomului de interpolare prin rezolvarea unui sistem liniar de ecuații algebrice;
- evaluarea polinomului interpolant.

Această tehnică de interpolare poate fi aplicată doar pentru valori mici ale gradului ($n < 5$), deoarece are două mari dezavantaje:

1. efortul de calcul pentru determinarea coeficienților este relativ mare, ordinul de complexitate al celui mai eficient algoritm de rezolvare a unui sistem liniar general fiind $O(2n^3/3)$;
2. erorile soluției sunt mari, deoarece sistemul este slab condiționat pentru valori mari ale gradului n .



Metoda Lagrange

O metodă care evită aceste dezavantaje este metoda Lagrange, în care funcțiile de bază se aleg de forma:

$$\begin{aligned} b_0(x) &= (x - x_1)(x - x_2) \cdots (x - x_n) \\ b_1(x) &= (x - x_0)(x - x_2) \cdots (x - x_n) \\ &\vdots \\ b_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned} \tag{11.4}$$

sau în general:

$$b_k(x) = \prod_{i=0, i \neq k}^n (x - x_i) \tag{11.5}$$

Impunând polinomului

$$g(x) = \sum_{k=0}^n c_k \prod_{i=0, i \neq k}^n (x - x_i) \tag{11.6}$$

condițiile de interpolare, rezultă sistemul de ecuații algebrice liniare:



Metoda Lagrange

$$\begin{aligned}g(x_0) &= c_0(x_0 - x_1)(x_0 - x_2) \cdots (x_0 - x_n) = y_0 \\g(x_1) &= c_1(x_1 - x_0)(x_1 - x_2) \cdots (x_1 - x_n) = y_1 \\\vdots & \\g(x_n) &= c_n(x_n - x_0) \cdot (x_n - x_1) \cdots (x_n - x_{n-1}) = y_n\end{aligned}\tag{11.7}$$

cu structura diagonală. Soluția acestui sistem este:

$$c_k = \frac{y_k}{\prod_{i=0, i \neq k}^n (x_k - x_i)}, \tag{11.8}$$

iar polinomul de interpolare are expresia

$$g(x) = \sum_{k=0}^n y_k \prod_{i=0, i \neq k}^n \frac{(x - x_i)}{(x_k - x_i)} = \sum_{k=0}^n y_k l_k(x), \tag{11.9}$$

în care s-a notat cu $l_k(x)$ polinomul lui Lagrange.

Metoda Lagrange elimină dezavantajele metodei "clasice", în schimb timpul necesar evaluării polinomului de interpolare crește de la ordinul liniar $O(n)$ la cel pătratic $O(n^2)$.



O altă metodă pentru determinarea polinomului de interpolare este metoda Newton, în care funcțiile de bază se aleg de forma:

$$\begin{aligned} b_0(x) &= 1 \\ b_1(x) &= (x - x_0) \\ b_2(x) &= (x - x_0)(x - x_1) \\ &\vdots \\ b_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned} \tag{11.10}$$

sau în general:

$$b_k(x) = \prod_{i=0}^{k-1} (x - x_i).$$

Impunând polinomului

$$g(k) = \sum_{k=0}^n c_k \prod_{i=0}^{k-1} (x - x_i) \tag{11.11}$$



Metoda Newton

condițiile de interpolare:

$$\begin{aligned}g(x_0) &= c_0 = y_0 \\g(x_1) &= c_0 + c_1(x_1 - x_0) = y_1 \\g(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2 \\\vdots \\g(x_n) &= c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \cdots = y_n\end{aligned}\tag{11.12}$$

rezultă un sistem algebric liniar cu structura triunghiular inferioară, a cărui rezolvare poate fi realizată prin metoda substituției progresive. Coeficienții polinomului de interpolare Newton sunt:

$$\begin{aligned}c_0 &= y_0 \\c_1 &= (y_1 - c_0)/(x_1 - x_0) = (y_1 - y_0)(x_1 - x_0) \\c_2 &= (y_2 - c_0 - c_1(x_2 - x_0))/(x_2 - x_0)/(x_2 - x_1) \\\vdots \\c_n &= (y_n - c_0 - c_1(x_n - x_0) - c_2(x_n - x_0)(x_n - x_1) \cdots)/(x_n - x_0)/ \cdots /(x_n - x_{n-1})\end{aligned}\tag{11.13}$$



Polinomul lui Newton cu diferențe divizate

Un concept aflat într-o strânsă relație cu polinomul de interpolare Newton este cel al *diferențelor divizate de ordinul k ale unei funcții f*, valori notate cu $f[\cdots]$ și care se definesc recursiv prin:

$$f[x_0, x_1, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0}, \quad (11.14)$$

iar în particular, în cazul diferenței divizate de ordinul 1:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}. \quad (11.15)$$

Se constată că valorile coeficienților c_k din polinomul de interpolare Newton sunt date de diferențele divizate

$$c_k = f[x_0, x_1, \dots, x_n]$$

și:

$$g(x) = \sum_{k=0}^n c_k b_k(x) = \sum_{k=0}^n f[x_0, x_1, \dots, x_k] \prod_{i=0}^{k-1} (x - x_i). \quad (11.16)$$



Polinomul lui Newton cu diferențe divizate

Determinarea coeficienților polinomului Newton este facilitată de utilizarea tabelei diferențelor divizate, în care elementele se calculează recursiv:

x	y	ord.1	ord.2	ord.3	ord.4
x_0	y_0	$f[x_0, x_1]$			
x_1	y_1	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	$f[x_0, \dots, x_3]$	
x_2	y_2	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_1, \dots, x_4]$	$f[x_0, \dots, x_4]$
x_3	y_3	$f[x_3, x_4]$	$f[x_2, x_3, x_4]$		
x_4	y_4
...

Prima ”linie” a acestui tabel conține chiar coeficienții polinomului de interpolare Newton. Diferențele divizate de diferite ordine reprezintă aproximări ale derivatelor de ordin corespunzător ale funcției în sensul că dacă $f(x)$ are m derive pe intervalul $[a, b]$ există un punct $a \leq z \leq b$ astfel încât:

$$f[x_0, x_1, \dots, x_m] = \frac{f^{(m)}(z)}{m!} \quad (11.17)$$

în care x_0, x_1, \dots, x_m sunt puncte distincte din intervalul $[a, b]$.



Exemplu - Polinomul lui Newton cu diferențe divizate

Să se găsească polinomul de interpolare pentru următorul set de date:

x	1	2	3	4	5
$f(x)$	2	5	10	17	26

A. Polinomul lui Newton cu diferențe divizate

1. Obținerea diferențelor divizate:

x	$f(x)$	$\mathcal{D}^1 f(x)$	$\mathcal{D}^2 f(x)$	$\mathcal{D}^3 f(x)$	$\mathcal{D}^4 f(x)$
1	2	$\frac{5 - 2}{2 - 1} = 3$			
2	5	$\frac{10 - 5}{3 - 2} = 5$	$\frac{5 - 3}{3 - 1} = 1$	$\frac{1 - 1}{4 - 1} = 0$	
3	10	$\frac{17 - 10}{4 - 3} = 7$	$\frac{7 - 5}{4 - 2} = 1$	$\frac{1 - 1}{5 - 2} = 0$	$\frac{0 - 0}{5 - 1} = 0$
4	17	$\frac{26 - 17}{5 - 4} = 9$			
5	26				



Exemplu - Polinomul lui Newton cu diferențe divizate

- Simina - MN

Şirul diferențelor divizate este sirul primelor valori de pe fiecare coloană, începând cu valorile funcției, adică:

$$[2, 3, 1, 0, 0].$$

2. Obținerea polinomului de interpolare

$$\begin{aligned}P(x) &= 2 + 3 \cdot (x - 1) + 1 \cdot (x - 1)(x - 2) + 0 \cdot (x - 1)(x - 2)(x - 3) + \\&\quad + 0 \cdot (x - 1)(x - 2)(x - 3)(x - 4) = \\&= 2 + 3x - 3 + x^2 - 3x + 2 = \\&= x^2 + 1.\end{aligned}$$



Interpolarea cu funcții spline

Fie o funcție tabelată, dată prin lista ordonată a variabilelor, $[x_1 = \min, x_2, \dots, x_n = \max]$, și lista valorilor sale, $[f_1, f_2, \dots, f_n]$.

Deoarece interpolarea polinomială globală pe tot intervalul $[x_1, x_n]$ (de exemplu, polinoamele de interpolare Newton și Lagrange) nu converge întotdeauna, apare ideea de interpolare polinomială pe porțiuni (interpolare spline), la care pe fiecare subdiviziune a intervalului $[x_0, x_n]$ definim un alt polinom de interpolare.

Funcția spline polinomială de ordinul întâi are expresia

$$S(x) = S_i(x) = s_{i,0} + s_{i,1}(x - x_i), \quad x \in [x_{i-1}, x_i] \quad (4.16)$$

unde coeficienții s_{ik} se determină din condițiile:

$$\begin{aligned} S(x_i) &= y_i, \quad i = \overline{0, n}; \\ S_i(x_i) &= S_{i+1}(x_i), \quad i = \overline{1, n-2}. \end{aligned} \quad (4.17)$$



Exemplu - Funcția spline polinomială de ordinul întâi

- Să se găsească funcțiile spline de interpolare de ordinul întâi pentru următorul set de date (funcția sinus):

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$f(x)$	0	1	0	-1	0

- Avem 5 puncte, deci $n = 4$ intervale.

Funcția căutată va avea expresia:

$$S(x) = \begin{cases} s_{10} + s_{11}(x - 0) & , x \in [0, \frac{\pi}{2}] \\ s_{20} + s_{21}(x - \frac{\pi}{2}) & , x \in [\frac{\pi}{2}, \pi] \\ s_{30} + s_{31}(x - \pi) & , x \in [\pi, \frac{3\pi}{2}] \\ s_{40} + s_{41}(x - \frac{3\pi}{2}) & , x \in [\frac{3\pi}{2}, 2\pi]. \end{cases}$$

Notăm:

$$\begin{cases} S_1 = s_{10} + s_{11}(x - 0) \\ S_2 = s_{20} + s_{21}(x - \frac{\pi}{2}) \\ S_3 = s_{30} + s_{31}(x - \pi) \\ S_4 = s_{40} + s_{41}(x - \frac{3\pi}{2}) \end{cases}$$

- Simina*



Exemplu - Funcția spline polinomială de ordinul întâi

Condițiile care se pun sunt următoarele:

- funcția trece prin puncte

$$S_1(0) = 0 \Rightarrow s_{10} = 0$$

$$S_2\left(\frac{\pi}{2}\right) = 1 \Rightarrow s_{20} = 1$$

$$S_3(\pi) = 0 \Rightarrow s_{30} = 0$$

$$S_4\left(\frac{3\pi}{2}\right) = -1 \Rightarrow s_{40} = -1$$

$$S_4(2\pi) = 0 \Rightarrow s_{41} = \frac{2}{\pi}$$

- funcția este continuă

$$S_1\left(\frac{\pi}{2}\right) = S_2\left(\frac{\pi}{2}\right) \Rightarrow s_{11} = \frac{2}{\pi}$$

$$S_2(\pi) = S_3(\pi) \Rightarrow s_{21} = -\frac{2}{\pi}$$

$$S_3\left(\frac{3\pi}{2}\right) = S_4\left(\frac{3\pi}{2}\right) \Rightarrow s_{31} = -\frac{2}{\pi}$$

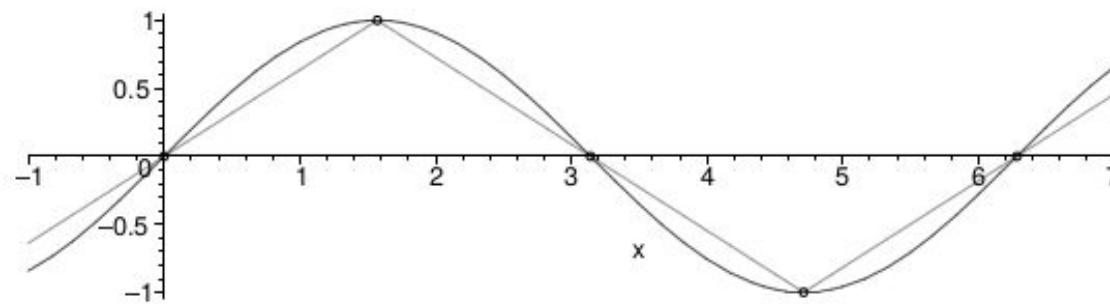


Exemplu - Funcția spline polinomială de ordinul întâi

- după efectuarea calculelor, funcția căutată devine:

$$S(x) = \begin{cases} \frac{2}{\pi}x & , x \in [0, \frac{\pi}{2}] \\ -\frac{2}{\pi}x + 2 & , x \in [\frac{\pi}{2}, \pi] \\ -\frac{2}{\pi}x + 2 & , x \in [\pi, \frac{3\pi}{2}] \\ \frac{2}{\pi}x - 4 & , x \in [\frac{3\pi}{2}, 2\pi]. \end{cases}$$

- Graficul funcției spline S în raport cu funcția inițială ($\sin x$):



Funcția spline polinomială de ordinul al doilea

Funcția spline polinomială de ordinul al doilea are expresia

$$S(x) = S_i(x) = s_{i,0} + s_{i,1}(x - x_i) + s_{i,2}(x - x_i)^2, \quad x \in [x_{i-1}, x_i] \quad (4.18)$$

unde coeficienții s_{ik} se determină din condițiile:

$$\begin{aligned} S(x_i) &= y_i, \quad i = \overline{0, n}; \\ S_i(x_i) &= S_{i+1}(x_i), \quad i = \overline{1, n-1}; \\ S'_i(x_i) &= S'_{i+1}(x_i), \quad i = \overline{1, n-1}. \end{aligned} \quad (4.19)$$

Deoarece avem $3n$ coeficienți și $3n - 1$ condiții, pentru a determina în mod unic funcția spline polinomială de ordinul al doilea mai avem nevoie de o condiție suplimentară. Această condiție suplimentară se referă la prima derivată, și are următoarea seminficație: unul din capetele funcției spline de ordinul al doilea trebuie să fie punct de extrem local. De regulă, se alege

$$S'_0(x_0) = 0. \quad (4.20)$$

Pentru diferite condiții suplimentare, se obțin diferite funcții spline.



Funcția spline polinomială cubică

Funcția spline polinomială cubică (de ordinul al treilea) are expresia

$$S(x) = S_i(x) = s_{i,0} + s_{i,1}(x - x_i) + s_{i,2}(x - x_i)^2 + s_{i,3}(x - x_i)^3, \quad x \in [x_{i-1}, x_i] \quad (4.21)$$

unde coeficienții s_{ik} se determină din condițiile:

$$\begin{aligned} S(x_i) &= y_i, \quad i = \overline{0, n}; \\ S_i(x_i) &= S_{i+1}(x_i), \quad i = \overline{1, n-1}; \\ S'_i(x_i) &= S'_{i+1}(x_i), \quad i = \overline{1, n-1}; \\ S''_i(x_i) &= S''_{i+1}(x_i), \quad i = \overline{1, n-1}. \end{aligned} \quad (4.22)$$

Deoarece avem $4n$ coeficienți și $4n - 2$ condiții, pentru determinarea în mod unic a funcției spline polinomiale cubice este nevoie de două condiții suplimentare. Aceste condiții suplimentare pot fi

- libere (sau naturale):

$$S''_1(x_0) = S''_{n-1}(x_{n-1}) = 0 \quad (4.23)$$

caz în care vorbim de o funcție spline cubică naturală;

- "clamped":

$$S'_1(x_0) = y'_0, \quad S'_{n-1}(x_{n-1}) = y'_{n-1}. \quad (4.24)$$



Exemplu - Funcția spline polinomială de ordinul trei - condiții

- funcția trece prin puncte

$$S_1(0) = 0$$

$$S_2\left(\frac{\pi}{2}\right) = 1$$

$$S_3(\pi) = 0$$

$$S_4\left(\frac{3\pi}{2}\right) = -1$$

$$S_4(2\pi) = 0$$

- funcția este continuă

$$S_1\left(\frac{\pi}{2}\right) = S_2\left(\frac{\pi}{2}\right)$$

$$S_2(\pi) = S_3(\pi)$$

$$S_3\left(\frac{3\pi}{2}\right) = S_4\left(\frac{3\pi}{2}\right)$$

- derivata este continuă

$$S_1'\left(\frac{\pi}{2}\right) = S_2'\left(\frac{\pi}{2}\right)$$

$$S_2'(\pi) = S_3'(\pi)$$

$$S_3'\left(\frac{3\pi}{2}\right) = S_4'\left(\frac{3\pi}{2}\right)$$

- derivata a doua este continuă

$$S_1''\left(\frac{\pi}{2}\right) = S_2''\left(\frac{\pi}{2}\right)$$

$$S_2''(\pi) = S_3''(\pi)$$

$$S_3''\left(\frac{3\pi}{2}\right) = S_4''\left(\frac{3\pi}{2}\right)$$

- condiții suplimentare

$$S_1''(0) = 0$$

$$S_4''(2\pi) = 0$$



Exemplu - Funcția spline polinomială de ordinul trei

- Soluția acestui sistem va fi:

$$s_{10} = 0$$

$$s_{11} = \frac{3}{\pi}$$

$$s_{12} = 0$$

$$s_{13} = \frac{4}{\pi^3}$$

$$s_{20} = 1$$

$$s_{21} = 0$$

$$s_{22} = -\frac{6}{\pi^2}$$

$$s_{23} = \frac{4}{\pi^3}$$

$$s_{30} = 0$$

$$s_{31} = -\frac{3}{\pi}$$

$$s_{32} = 0$$

$$s_{33} = \frac{4}{\pi^3}$$

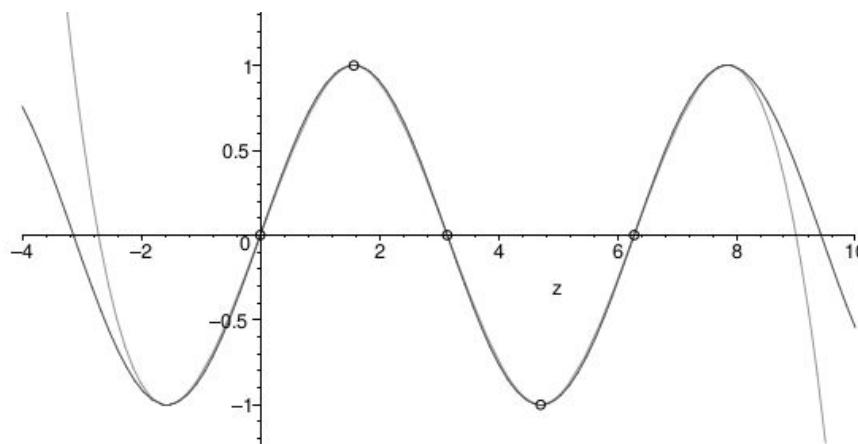
$$s_{40} = -1$$

$$s_{41} = 0$$

$$s_{42} = \frac{6}{\pi^2}$$

$$s_{43} = -\frac{4}{\pi^3}$$

$$S(x) = \begin{cases} \frac{-x(-3\pi^2 + 4x^2)}{\pi^3}, & x \in [0, \frac{\pi}{2}] \\ \frac{(-\pi^3 - 12\pi x^2 + 9x\pi^2 + 4x^3)}{\pi^3}, & x \in [\frac{\pi}{2}, \pi] \\ \frac{(x - \pi)(\pi^2 + 4x^2 - 8x\pi)}{\pi^3}, & x \in [\pi, \frac{3\pi}{2}] \\ \frac{(-26\pi^3 - 24\pi x^2 + 45x\pi^2 + 4x^3)}{\pi^3}, & x \in [\frac{3\pi}{2}, 2\pi] \end{cases}$$



Să se găsească funcția spline de ordinul întâi și funcția spline cubică naturală pentru următoarele seturi de date:

a)
$$\begin{array}{|c|cccc|} \hline x & -1 & 0 & 1 & 2 \\ \hline f(x) & 2 & 1 & 0 & -1 \\ \hline \end{array}$$

b)
$$\begin{array}{|c|cccc|} \hline x & \frac{\pi}{2} & \pi & \frac{3\pi}{2} & 2\pi \\ \hline f(x) & 1 & 0 & -1 & 0 & 1 \\ \hline \end{array}$$



Exemplu - Funcția spline polinomială de ordinul trei

- Să se găsească funcțiile spline de interpolare cubice pentru următorul set de date (funcția sinus):

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$f(x)$	0	1	0	-1	0

$$S(x) = \begin{cases} s_{10} + s_{11}(x - 0) + s_{12}(x - 0)^2 + s_{13}(x - 0)^3 & , x \in [0, \frac{\pi}{2}] \\ s_{20} + s_{21}(x - \frac{\pi}{2}) + s_{22}(x - \frac{\pi}{2})^2 + s_{23}(x - \frac{\pi}{2})^3 & , x \in [\frac{\pi}{2}, \pi] \\ s_{30} + s_{31}(x - \pi) + s_{32}(x - \pi)^2 + s_{33}(x - \pi)^3 & , x \in [\pi, \frac{3\pi}{2}] \\ s_{40} + s_{41}(x - \frac{3\pi}{2}) + s_{42}(x - \frac{3\pi}{2})^2 + s_{43}(x - \frac{3\pi}{2})^3 & , x \in [\frac{3\pi}{2}, 2\pi]. \end{cases}$$

Notăm:

$$S(x) = \begin{cases} S_1 = s_{10} + s_{11}(x - 0) + s_{12}(x - 0)^2 + s_{13}(x - 0)^3 \\ S_2 = s_{20} + s_{21}(x - \frac{\pi}{2}) + s_{22}(x - \frac{\pi}{2})^2 + s_{23}(x - \frac{\pi}{2})^3 \\ S_3 = s_{30} + s_{31}(x - \pi) + s_{32}(x - \pi)^2 + s_{33}(x - \pi)^3 \\ S_4 = s_{40} + s_{41}(x - \frac{3\pi}{2}) + s_{42}(x - \frac{3\pi}{2})^2 + s_{43}(x - \frac{3\pi}{2})^3 \end{cases}$$



Algoritm - funcții spline de interpolare cubice

- Se definesc două valori pentru derivata a doua:

$$s_i''(x_i) = f_i \quad s_i''(x_{i+1}) = f_{i+1}$$

- Interpoland liniar intr-un punct x intre x_i si x_{i+1} cu valorile f_i, f_{i+1} se obtine

$$s_i''(x) = \frac{f_i}{x_{i+1} - x_i}(x_{i+1} - x) + \frac{f_{i+1}}{x_{i+1} - x_i}(x - x_i)$$

- Integrând de două ori relația obținem $s_i(x)$ ca functie de f_i si f_{i+1} :

$$s_i(x) = \frac{f_i}{6(x_{i+1} - x_i)}(x_{i+1} - x)^3 + \frac{f_{i+1}}{6(x_{i+1} - x_i)}(x - x_i)^3 + c(x - x_i) + d(x_{i+1} - x)$$



Algoritm - funcții spline de interpolare cubice

- Utilizând condițiile $s_i(x_i) = y_i$ and $s_i(x_{i+1}) = y_{i+1}$ putem determina constantele c și d din care rezultă:

$$s_i(x) = \frac{f_i}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_{i+1}}{6(x_{i+1}-x_i)}(x-x_i)^3 \\ + \left(\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_{i+1}(x_{i+1}-x_i)}{6} \right)(x-x_i) + \left(\frac{y_i}{x_{i+1}-x_i} - \frac{f_i(x_{i+1}-x_i)}{6} \right)(x_{i+1}-x)$$

- Pentru a determina valorile derivatei a doua f_i and f_{i+1} utilizăm condiția de continuitate pentru prima derivată:

$$s'_{i-1}(x_i) = s'_i(x_i)$$

- pentru $x=x_i$. Definim

$$h_i = x_{i+1} - x_i$$

- Obținem în final:

$$h_{i-1}f_{i-1} + 2(h_i + h_{i-1})f_i + h_if_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$



Algoritm - funcții spline de interpolare cubice

- Utilizând notațiile:

$$u_i = 2(h_i + h_{i-1}), \quad v_i = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$

- Putem re scrie problema ca un sistem liniar de ecuații tridiagonală

$$\begin{bmatrix} u_1 & h_1 & 0 & \dots & & \\ h_1 & u_2 & h_2 & 0 & \dots & \\ 0 & h_2 & u_3 & h_3 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & \dots & & 0 & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & 0 & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \\ f_{n-2} \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$



Functii de interpolare B-spline

- B-spline sunt functii polinomiale cubice definite pe subintervale si care sunt construite pentru a avea derivatele de ordinul unu si doi continue la granitele dintre intervale. Acestea formeaza un set de functii de baza, functiile spline intr-un punct fiind o combinatie liniara de functii pe subintervale (echidistante).
- *Exemplu:*

$$B_0(x) = \begin{cases} 0 & x \leq x_0 - 2h \\ \frac{1}{6}(2h + (x - x_0))^3 & x_0 - 2h \leq x \leq x_0 - h \\ \frac{2h^3}{3} - \frac{1}{2}(x - x_0)^2(2h + (x - x_0)) & x_0 - h \leq x \leq x_0 \\ \frac{2h^3}{3} - \frac{1}{2}(x - x_0)^2(2h - (x - x_0)) & x_0 \leq x \leq x_0 + h \\ \frac{1}{6}(2h - (x - x_0))^3 & x_0 + h \leq x \leq x_0 + 2h \\ 0 & x \geq x_0 + 2h \end{cases}$$

unde

$$h = x_{k+1} - x_k = \frac{x_N - x_0}{N}$$

Se poate observa ca B_0 satisface conditiile de continuitate derivatele de ordinul unu si doi in punctele $-2h, -h, 0, h, 2h$.

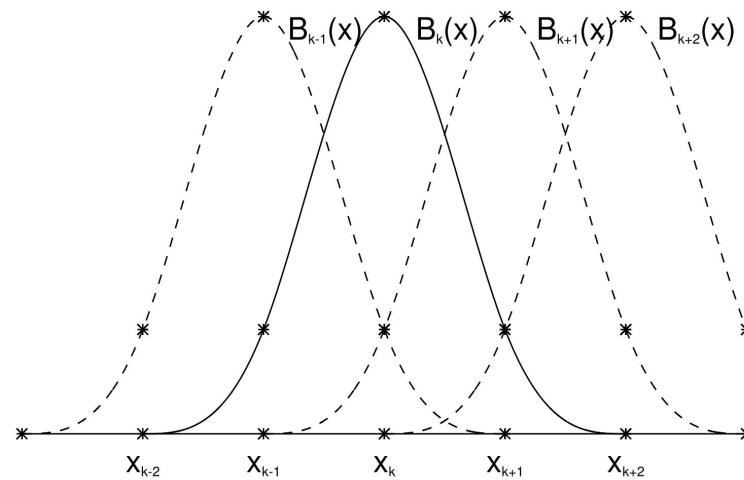
Se defineste

$$B_k(x) = B_0(x - kh + x_0)$$

ca B_0 mutata la dreapta cu k noduri.



$B_{k-1}, B_k, B_{k+1}, B_{k+2}$ nenule in intervalul (x_k, x_{k+1})



Functia cubica $S_3(x)$ se poate scrie ca o combinatie liniara de baza B_k

$$S_3(x) = \sum_{k=-1}^{N+1} a_k B_k(x)$$

iar in x_k :

$$S_3(x_k) = a_{k-1}B_{k-1}(x_k) + a_kB_k(x_k) + a_{k+1}B_{k+1}(x_k) + a_{k+2}B_{k+2}(x_k) = f(x_k)$$

deoarece celelalte functii de baza sunt nule



- Din definitii avem:

$$B_k(x_k) = B_0(x_0) = \frac{2h^3}{3}$$

$$B_{k-1}(x_k) = B_0(x_0 + h) = \frac{h^3}{6}$$

$$B_{k+1}(x_k) = B_0(x_0 - h) = \frac{h^3}{6}$$

$$B_{k+2}(x_k) = B_0(x_0 - 2h) = 0$$

- si substituind in S3 avem (A)

$$a_{k-1} + 4a_k + a_{k+1} = \frac{6}{h^3} f(x_k) \quad k = 0, \dots, N.$$

- Consideram suplimentar

$$S_3''(x_0) = S_3''(x_N) = 0$$

- Diferentiind:

$$S_3''(x) = \sum_{k=-1}^{N+1} a_k B_k''(x)$$

$$B_0''(x) = \begin{cases} 0 & x \leq x_0 - 2h \\ 2h + (x - x_0) & x_0 - 2h \leq x \leq x_0 - h \\ -2h - 3(x - x_0) & x_0 - h \leq x \leq x_0 \\ -2h + 3(x - x_0) & x_0 \leq x \leq x_0 + h \\ 2h - (x - x_0) & x_0 + h \leq x \leq x_0 + 2h \\ 0 & \text{else} \end{cases}$$



B-spline

$$\begin{aligned} 0 = S_3''(x_0) &= a_{-1}B_{-1}''(x_0) + a_0B_0''(x_0) + a_1B_1''(x_0) + a_2B_2''(x_0) \\ &= a_{-1}B_0''(x_0 + h) + a_0B_0''(x_0) + a_1B_0''(x_0 - h) \\ &= a_{-1}h - 2ha_0 + a_1h \\ &= a_{-1} - 2a_0 + a_1 \end{aligned}$$

- iar din (A)

$$a_{-1} + 4a_0 + a_1 = \frac{6}{h^3}f(x_0)$$

- Scazand cele doua ecuatii obtinem

$$a_0 = \frac{1}{h^3}f(x_0)$$

- Similar

$$a_N = \frac{1}{h^3}f(x_N) \quad a_{-1} = 2a_0 - a_1$$

- Iar pentru ceilalti coeficienti

$$\left(\begin{array}{cccccc} 1 & 0 & 0 & & & \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & & & & & \\ 0 & 0 & & 1 & 4 & 1 & \\ 0 & 0 & & 0 & 1 & & \end{array} \right) \left(\begin{array}{c} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \\ a_N \end{array} \right) = \frac{1}{h^3} \left(\begin{array}{c} f(x_0) \\ 6f(x_1) \\ 6f(x_2) \\ \vdots \\ 6f(x_{N-1}) \\ f(x_N) \end{array} \right)$$



Fitarea funcțiilor reale

Lect. dr. Ioan Dumitru



Presupunem că avem o funcție definită printr-un tabel de valori. Dacă tabelul este obținut în urma unor măsurători fizice, atunci elementele lui pot avea erori. Elementele care diferă mult de celelalte pot fi eliminate. Problema care se pune este să determinăm funcția analitică care aproximează cel mai bine datele din tabel sau curba care trece prin punctele tabelului.

Considerăm că funcția tabelată poate fi aproximată cu funcția: $\bar{y} = f(x)$. Deoarece funcția tabelată este cunoscută numai în anumite puncte, în număr de m , în aceste puncte avem următoarele erori:

$$\begin{cases} e_1 = y_1 - \bar{y}_1 = y_1 - f(x_1) \\ e_2 = y_2 - \bar{y}_2 = y_2 - f(x_2) \\ \cdots \\ e_m = y_m - \bar{y}_m = y_m - f(x_m) \end{cases} \quad (7.1)$$

care ar putea fi calculate dacă am cunoaște funcția $\bar{y} = f(x)$.

Pentru determinarea funcției, vom anticipa întâi o funcție care să se asemene foarte mult cu curba realizată cu punctele tabelului. Această funcție poate fi: liniară, hiperbolică, logaritmică, exponențial geometrică, trigonometrică etc.



Metoda celor mai mici patrate

Al doilea pas este determinarea funcției din condițiile ca suma abaterilor:

$$e = \sum_{i=1}^m (y_i - \bar{y}_i)$$

să fie minimă.

În cazul unei drepte dată prin două puncte, abaterea (7.2) este nulă pentru toate dreptele ce trec prin mijlocul segmentului format de cele două puncte. Ca urmare problema nu este unic determinată. Această problemă se elimină dacă considerăm valoarea absolută a erorilor:

$$|e| = \sum_{i=1}^m |y_i - \bar{y}_i| \quad (7.2a)$$

Inconvenientul în cazul formulei (7.2a) îl prezintă funcția valoare absolută care nu are derivată în punctele în care se anulează funcția. Pentru eliminarea acestor inconveniente s-a trecut la suma pătratelor erorilor, pe care o notăm cu E .

$$E = \sum_{i=1}^m (y_i - \bar{y}_i)^2 \quad (7.3)$$

Metoda de optimizare a primit denumirea de *metoda celor mai mici pătrate*, conform formulei (7.3) pentru că determină funcția $\bar{y} = f(x)$ pentru eroarea maximă și mai poartă denumirea și de *regresie* deoarece problema este de deducere a funcției cunoscând valorile ei într-un anumit număr de puncte.

- Ioan Rusu - Metode numerice



Regresia liniara

Se consideră funcția tabelată:

Tabelul 7.1

x	x_1	x_2	x_3	x_4	x_m
y	y_1	y_2	y_3	y_4	y_m

unde m reprezintă numărul de măsurători sau de valori ale funcției. Se cere să se determine funcția liniară de forma generală

$$y = ax + b \quad (7.4)$$

care să aproximeze cel mai bine funcția tabelată (tabelul 7.1) astfel ca eroarea:

$$E = \sum_{i=1}^m (ax_i + b - y_i)^2 \quad (7.5)$$

să fie minimă. Necunoscutele a și b se determină din condiția de minim a lui E care se realizează pentru anularea derivatelor parțiale în raport cu a și b ale lui E :

$$\begin{cases} \frac{\partial E}{\partial a} = \sum_{i=1}^m 2(ax_i + b - y_i) \cdot (-1) = 0 \\ \frac{\partial E}{\partial b} = \sum_{i=1}^m 2(ax_i + b - y_i) \cdot (-x_i) = 0 \end{cases} \quad (7.6)$$

Rezultă sistemul în necunoscutele a și b :



Regresia liniara

$$\begin{cases} m \cdot b + \left(\sum_{i=1}^m x_i \right) a = \sum_{i=1}^m y_i \\ \left(\sum_{i=1}^m x_i \right) b + \left(\sum_{i=1}^m x_i^2 \right) a = \sum_{i=1}^m x_i y_i \end{cases} \quad (7.7)$$

Soluțiile sistemului liniar (7.7) în a și b sunt:

$$b = \frac{\sum_{i=1}^m x_i^2 \sum_{i=1}^m y_i - \sum_{i=1}^m x_i \sum_{i=1}^m x_i y_i}{m \sum_{i=1}^m x_i^2 - \left(\sum_{i=1}^m x_i \right)^2} \quad (7.8)$$

$$a = \frac{m \sum_{i=1}^m x_i y_i - \sum_{i=1}^m x_i \sum_{i=1}^m y_i}{m \sum_{i=1}^m x_i^2 - \left(\sum_{i=1}^m x_i \right)^2} \quad (7.9)$$

Numitorii expresiilor lui a și b se anulează numai dacă x_i sunt identice, caz exclus. Înlocuind valorile lui a și b în ecuația (7.4) avem funcția liniară care aproximează cel mai bine funcția tabelată dată (7.1).



Exemplu - Regresia liniara

2. Legea lui Hook arată că $F = kx$, unde F este forța exprimată în newtoni utilizată pentru a întinde un resort, iar x este alungirea resortului exprimată în metri. Să se găsească constanta de elasticitate k . Datele experimentale sunt prezentate în tabelele următoare.

(a)	x_i	F_i
	0.2	3.6
	0.4	7.3
	0.6	10.9
	0.8	14.5
	1.0	18.2

(b)	x_i	F_i
	0.2	5.3
	0.4	10.6
	0.6	15.9
	0.8	21.2
	1.0	26.4

Rezolvare:

- (a) $k = A = 18.2$, $B = -0.02$.
- (b) $k = A = 26.4$, $B = 0.04$.



Coeficientul de corelare R²

Calitatea unei drepte de regresie poate fi analizată după coeficientul de corelare R², care se calculează cu relația:

$$R^2 = 1 - \frac{\sum_i (y_i^{calc} - y_i^{exp})^2}{\sum_i (\bar{y}^{exp} - y_i^{exp})^2}$$

Valoarea 1 pentru acest coeficient are semnificația că funcția model explică întreaga variabilitate a lui y, iar valoarea 0 că nu există nici o relație liniară între variabila răspuns și variabila x (între y și x). O valoare de 0.5 a lui R² poate fi interpretată în felul următor: aproximativ 50% din variația variabilei răspuns poate fi explicată de către variabila independentă.



Regresia polinomială

Dacă regresia liniară nu este satisfăcătoare se caută o funcție polinom de un anumit grad (2, 3, 4...n).

Fie funcția dată în tabelul (7.1). Considerăm că reprezentarea grafică a acestei funcții se asemănă foarte mult cu curba unui polinom de gradul n de forma:

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (7.10)$$

Notăm cu y_i expresia:

$$y_i = a_n x_i^n + a_{n-1} x_i^{n-1} + \dots + a_1 x_i + a_0 \quad (7.11)$$

Se minimizează funcția ţintă:

$$E = \sum_{i=1}^m \left(y_i - a_n x_i^n - a_{n-1} x_i^{n-1} - \dots - a_1 x_i - a_0 \right)^2 \quad (7.12)$$

unde m reprezintă numărul de valori ale funcției tabelate.

Necunoscutele sunt: a_n, a_{n-1}, \dots, a_0 . (7.13)

Se derivează în raport cu necunoscutele (7.13) și se obține sistemul:

$$\left\{ \begin{array}{l} ma_0 + \left(\sum_{i=1}^m x_i \right) a_1 + \left(\sum_{i=1}^m x_i^2 \right) a_2 + \dots + \left(\sum_{i=1}^m x_i^k \right) a_k + \left(\sum_{i=1}^m x_i^n \right) a_n = \sum y_i \\ \left(\sum_{i=1}^m x_i \right) a_0 + \left(\sum_{i=1}^m x_i^2 \right) a_1 + \left(\sum_{i=1}^m x_i^3 \right) a_2 + \dots + \left(\sum_{i=1}^m x_i^{k+1} \right) a_k + \dots + \left(\sum_{i=1}^m x_i^{n+1} \right) a_n = \sum x_i y_i \\ \hline \left(\sum_{i=1}^m x_i^p \right) a_0 + \left(\sum_{i=1}^m x_i^{p+1} \right) a_1 + \left(\sum_{i=1}^m x_i^{p+2} \right) a_2 + \dots + \left(\sum_{i=1}^m x_i^{p+k} \right) a_k + \dots + \left(\sum_{i=1}^m x_i^{p+n} \right) a_n = \sum x_i^p y_i \\ \left(\sum_{i=1}^m x_i^n \right) a_0 + \overline{\left(\sum_{i=1}^m x_i^{n+1} \right) a_1 + \left(\sum_{i=1}^m x_i^{n+2} \right) a_2 + \dots + \left(\sum_{i=1}^m x_i^{n+k} \right) a_k + \dots + \left(\sum_{i=1}^m x_i^{2n} \right) a_n} = \sum x_i^n y_i \end{array} \right. \quad (7.14)$$

Sistemul obținut este un sistem liniar în necunoscutele $a_0, a_1, a_2, \dots, a_n$ și se rezolvă



Particularizare pentru regresia polinomială

2. Fitarea curbelor de tip $y = Ax^M$

În această situație (M este o constantă cunoscută), impunând de asemenea condiția de minimizare a erorii pătratice medii, se găsește:

$$(40) \quad A = \sum_{k=1}^N x_k^M y_k / \sum_{k=1}^N x_k^{2M}$$

5. Un student a colectat datele din tabelele de mai jos, în care d este distanța exprimată în metri, iar t este timpul exprimat în secunde, legătura dintre d și t fiind: $d = \frac{1}{2}gt^2$.

Să se găsească constanta gravitațională g .

(a)	$t(\text{sec})$	$d(\text{m})$
	0.2	0.1960
	0.4	0.7850
	0.6	1.7675
	0.8	3.1405
	1.0	4.9075

(b)	$t(\text{sec})$	$d(\text{m})$
	0.2	0.1960
	0.4	0.7835
	0.6	1.7630
	0.8	3.1345
	1.0	4.8975

Rezolvare:

- (a) Utilizând formula 40, rezultă că $A = 4.9073$. Ca urmare, $g = 2 \times A = 9.8146$.
- (b) $A = 4.895$, $g = 9.7950$.



Exemplu - Regresia polinomială

2. Să se gasească parabola celor mai mici pătrate pentru patru puncte, $(-3, 3), (0, 1), (2, 1), (4, 3)$.

$$f(x) = Ax^2 + Bx + C$$

și ca urmare trebuie rezolvat sistemul de ecuații liniare:

$$\sum_{i=1}^N x_i^4 A + \sum_{i=1}^N x_i^3 B + \sum_{i=1}^N x_i^2 C = \sum_{i=1}^N x_i^2 y_i$$

$$\sum_{i=1}^N x_i^3 A + \sum_{i=1}^N x_i^2 B + \sum_{i=1}^N x_i C = \sum_{i=1}^N x_i y_i$$

$$\sum_{i=1}^N x_i^2 A + \sum_{i=1}^N x_i B + NC = \sum_{i=1}^N y_i$$

Sumele ce apar în sistemul de ecuații de mai sus pot fi calculate foarte ușor cu ajutorul tabelului prezentat în continuare. Astfel, vom avea de rezolvat

TAB 2. Găsirea coeficienților pentru cazul exercițiului 2

x_i	y_i	x_i^2	x_i^3	x_i^4	$x_i y_i$	$x_i^2 y_i$
-3	3	9	-27	81	-9	27
0	1	0	0	0	0	0
2	1	4	8	16	2	4
4	3	16	64	256	12	48
3 8 29 45 353 5 79						

sistemul de ecuații:

$$\begin{cases} 353A + 45B + 29C = 79 \\ 45A + 29B + 3C = 5 \\ 29A + 3B + 4C = 8 \end{cases}$$

Soluțiile acestui sistem sunt: $A = 0.178462$, $B = -0.192495$, $C = 0.850518$, polinomul căutat fiind:

$$y = 0.178462x^2 - 0.192495x + 0.850518$$



Regresia exponentială

Se constată că funcția numerică din tabelul 7.1 reprezentată grafic, se aseamănă foarte mult cu o exponențială. Considerăm funcția exponențială de forma generală:

$$y = a \cdot b^x \quad (7.20)$$

în care a și b sunt constante pozitive.

Pentru un calcul comod al constantelor a și b se logaritmează funcția (7.20)

$$y = \ln a + x \ln b \quad (7.21)$$

și se determină constantele astfel ca eroarea:

$$E = \sum_{i=1}^m (\ln a + x_i \ln b - \ln y_i)^2 \quad (7.22)$$

să fie minimă.

Prin derivarea parțială a funcției E în raport cu a și b se obține următorul sistem:

$$\begin{cases} m \ln a + \left(\sum_{i=1}^m x_i \right) \ln b = \sum_{i=1}^m \ln y_i \\ \left(\sum_{i=1}^m x_i \right) \ln a + \left(\sum_{i=1}^m x_i^2 \right) \ln b = \sum_{i=1}^m x_i \ln y_i \end{cases} \quad (7.23)$$

Soluțiile sistemului obținut (7.23) sunt:

$$a = \exp \left\{ \frac{\left[\sum_{i=1}^m \ln y_i \sum_{i=1}^m x_i^2 - \sum_{i=1}^m x_i \sum_{i=1}^m \ln y_i x_i \right]}{\left[m \sum_{i=1}^m x_i^2 - \left(\sum_{i=1}^m x_i \right)^2 \right]} \right\} \quad (7.24)$$

$$b = \exp \left\{ \frac{\left[m \sum_{i=1}^m x_i \ln y_i - \sum_{i=1}^m x_i \sum_{i=1}^m \ln y_i \right]}{\left[m \sum_{i=1}^m x_i^2 - \left(\sum_{i=1}^m x_i \right)^2 \right]} \right\} \quad (7.25)$$



Dacă funcția numerică din tabelul 7.1 se aseamănă cu o funcție de tip geometric, vom căuta să determinăm funcția de forma:

$$y = ax^b \quad (7.26)$$

care să aproximeze cel mai bine funcția numerică.

Se logaritmează cel mai bine funcția (7.26) și se construiește funcția:

$$E = \sum_{i=1}^m (\ln a + b \ln x_i - \ln y_i)^2 \quad (7.27)$$

Se calculează valorile lui a și b astfel ca funcția E să fie minimă. Prin derivarea parțială în raport cu a și b a expresiei E și egalarea derivatelor cu zero se obține următorul sistem în necunoscutele a și b :

$$\begin{cases} m \ln a + b \left(\sum_{i=1}^m \ln x_i \right) = \sum_{i=1}^m \ln y_i \\ \left(\sum_{i=1}^m \ln x_i \right) \ln a + b \sum_{i=1}^m (\ln x_i)^2 = \sum_{i=1}^m \ln x_i \ln y_i \end{cases} \quad (7.28)$$

Rezolvând sistemul se obțin următoarele valori pentru a și b :

$$a = \exp \frac{\left(\sum_{i=1}^m \ln(y_i) \right) \left(\sum_{i=1}^m \ln^2(x_i) \right) - \left(\sum_{i=1}^m \ln(x_i) \right) \left(\sum_{i=1}^m \ln(x_i) \ln(y_i) \right)}{m \left(\sum_{i=1}^m \ln^2(x_i) \right) - \left(\sum_{i=1}^m \ln(x_i) \right)^2} \quad (7.29)$$

$$b = \frac{m \left(\sum_{i=1}^m \ln(y_i) \ln(x_i) \right) - \left(\sum_{i=1}^m \ln(x_i) \right) \left(\sum_{i=1}^m \ln(y_i) \right)}{m \left(\sum_{i=1}^m \ln^2(x_i) \right) - \left(\sum_{i=1}^m \ln(x_i) \right)^2} \quad (7.30)$$



Regresia trigonometrică

Se consideră funcția numerică dată în tabelul 7.1 și funcția trigonometrică de forma:

$$y = a + b \cos \omega x \quad (7.31)$$

care aproximează cel mai bine funcția numerică.

Se determină constantele a și b astfel ca funcția de a și b

$$E = \sum_{i=1}^m (y_i - a - b \cos \omega x_i)^2 \quad (7.32)$$

Prin egalarea cu zero a derivatelor parțiale în raport cu a și b ale funcției E se obține următorul sistem în necunoscutele a și b :

$$\begin{cases} ma + \left(\sum_{i=1}^m \cos \omega x_i \right) b = \sum_{i=1}^m y_i \\ \left(\sum_{i=1}^m \cos \omega x_i \right) a + \left(\sum_{i=1}^m \cos^2 \omega x_i \right) b = \sum_{i=1}^m y_i \cos \omega x_i \end{cases} \quad (7.33)$$

Soluțiile sistemului (7.32) sunt:

$$a = \frac{\left(\sum_{i=1}^m y_i \right) \left(\sum_{i=1}^m \cos^2 \omega x_i \right) - \left(\sum_{i=1}^m \cos \omega x_i \right) \left(\sum_{i=1}^m y_i \cos \omega x_i \right)}{m \left(\sum_{i=1}^m \cos^2 \omega x_i \right) - \left(\sum_{i=1}^m \cos \omega x_i \right)^2} \quad (7.34)$$

$$b = \frac{m \left(\sum_{i=1}^m y_i \cos \omega x_i \right) - \left(\sum_{i=1}^m \cos \omega x_i \right) \left(\sum_{i=1}^m y_i \right)}{m \left(\sum_{i=1}^m \cos^2 \omega x_i \right) - \left(\sum_{i=1}^m \cos \omega x_i \right)^2} \quad (7.35)$$

$\omega = 2 \cdot \pi \cdot f = \frac{2 \cdot \pi}{T}$ unde f - frecvență, ω - pulsăție, iar T - perioada. ω se stabilește funcție de periodicitatea funcției numerice date.



Regresia multipla

Această regresie se referă la funcțiile de mai multe variabile. Considerând o funcție numerică de n variabile $y = f(x_1, x_2, \dots, x_n)$ și un tip de funcție analitică de n variabile care se asemănă cu cea numerică, se pune problema determinării funcției analitice astfel ca ea să aproximeze cel mai bine funcția numerică. Această funcție se determină prin minimizarea sumei pătratelor erorilor în punctele funcției numerice. Pentru exemplificare considerăm funcția de două variabile numerică (x, y, z) unde $z = g(x, y)$. Suprafața punctelor z se poate asemăna cu un plan, hiperboloid de rotație, elipsoid de rotație, cilindru, con etc.

Vom considera funcția numerică dată în figura 7.1

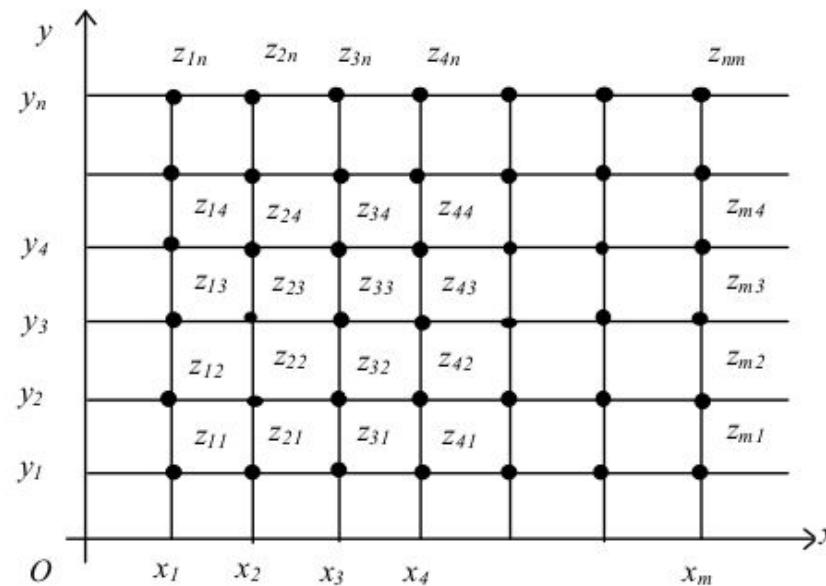


Fig. 7.1. Reprezentarea funcției multiple.



Regresia multipla

Considerăm că punctele z_1, z_2, \dots, z_n se află aproximativ pe un plan:

$$z = Ax + By + C \quad (7.36)$$

Se pune problema determinării acestui plan, adică a valorilor constantelor A, B, C astfel ca planul să aproximeze cel mai bine funcția numerică dată în tabelul 7.2. Se construiește funcția sumelor pătratelor erorilor în punctele rețelei:

$$E = \sum_{i=1}^m \sum_{j=1}^n (z_{ij} - Ax_i - By_j - C)^2 \quad (7.37)$$

Prin egalarea derivatelor parțiale ale funcției E în raport cu necunoscutele A, B, C , rezultă un sistem liniar funcție de aceste necunoscute:

$$\begin{cases} mnC + \left(\sum_{i=1}^m x_i \right) nA + \left(\sum_{j=1}^n y_j \right) mB = \sum_{i=1}^m \sum_{j=1}^n z_{ij} \\ \left(\sum_{i=1}^m x_i \right) nC + \left(\sum_{i=1}^m x_i^2 \right) nA + \left(\sum_{i=1}^m \sum_{j=1}^n x_i y_j \right) B = \sum_{i=1}^m \sum_{j=1}^n x_i z_{ij} \\ \left(\sum_{j=1}^n y_j \right) mC + \left(\sum_{i=1}^m \sum_{j=1}^n x_i y_j \right) A + \left(\sum_{j=1}^n y_j^2 \right) mB = \sum_{i=1}^m \sum_{j=1}^n y_j z_{ij} \end{cases} \quad (7.38)$$

Utilizând una dintre metodele numerice de rezolvare a sistemelor liniare din capitolul 3 se determină A, B, C , deci planul căutat.



Aproximarea prin serii Fourier

Vom considera acum o altă alegere a funcțiilor liniar independente din (1.6), anume funcțiile trigonometrice (fig. 1.5)

$$\cos(2\pi kx), k \in \overline{0, N}; \sin(2\pi mx), m \in \overline{1, N-1}. \quad (1.54)$$

Această bază este deosebit de eficace în aproximarea funcțiilor periodice $f(x) = f(x + 1)$. Funcțiile periodice cu o altă perioadă $f(z) = f(z + T)$ pot fi aduse la forma anterioară prin transformarea $x = z/T$. În cazul interpolării cu funcții trigonometrice, funcțiile sinus și cosinus împreună formează o bază⁵. Avem un număr de $2N$ funcții în această bază. Ca atare, vom considera un număr par $2N$ de puncte de diviziune echidistante pe intervalul $[0, 1]$

$$x_j = j/2N, j \in \overline{0, 2N-1}. \quad (1.55)$$

Se verifică direct că setul (1.54) prezintă următoarele proprietăți de ortogonalitate pe mulțimea discretă de puncte $\{x_i\} = \{0, 1/2N, 2/2N, \dots, (2N-1)/2N\}$

$$\sum_{j=0}^{2N-1} \cos 2\pi kx_j \cos 2\pi mx_j = \begin{cases} 0, & k \neq m \\ N, & k = m \neq 0, N \\ 2N, & k = m = 0, N \end{cases} \quad (1.56)$$

$$\sum_{j=0}^{2N-1} \sin 2\pi kx_j \sin 2\pi mx_j = \begin{cases} 0, & k \neq m \\ N, & k = m \end{cases} ; \quad \sum_{j=0}^{2N-1} \cos 2\pi kx_j \sin 2\pi mx_j = 0,$$

$\forall k \in \overline{0, N}, m \in \overline{1, N-1}$. Demonstrația se construiește prin transformarea produselor de funcții trigonometrice în sume de sinus și cosinus. Acestea se pot înlocui cu funcții exponențiale, $\sin x = (e^{ix} - e^{-ix})/2i$, $\cos x = (e^{ix} + e^{-ix})/2$,



Aproximarea prin serii Fourier

Utilizând setul de funcții de bază (1.54), aproximanta prin interpolare se scrie sub forma *polinomului Fourier*

$$g(x) = \frac{1}{2}a_0 + \sum_{k=1}^{N-1} [a_k \cos(2\pi kx) + b_k \sin(2\pi kx)] + \frac{1}{2}a_N \cos 2\pi Nx , \quad (1.57)$$

ce satisface $2N$ condiții de interpolare

$$g(x_j) = f(x_j) \equiv y_j, j \in \overline{0, 2N-1} .$$

Coeficienții a_k, b_k se determină prin utilizarea proprietăților de ortogonalitate (1.56). Prezentăm calculul doar pentru coeficienții b_k . Polinomul (1.57) se evaluează în punctele x_j , relația obținută se amplifică cu $\sin 2\pi mx_j$ iar apoi se calculează suma de la $j = 0$ la $j = 2N - 1$

$$\begin{aligned} \sum_{j=0}^{2N-1} y_j \sin 2\pi mx_j &= \frac{a_0}{2} \sum_{j=0}^{2N-1} \sin 2\pi mx_j + \sum_{k=1}^{N-1} \left[a_k \sum_{j=0}^{2N-1} (\cos 2\pi kx_j \sin 2\pi mx_j) + \right. \\ &\quad \left. b_k \sum_{j=0}^{2N-1} (\sin 2\pi kx_j \sin 2\pi mx_j) \right] + \frac{a_N}{2} \sum_{j=0}^{2N-1} (\cos 2\pi Nx_j \sin 2\pi mx_j) . \end{aligned}$$

Se poate observa schimbarea ordinii de însumare din relația de mai sus ce permite aplicarea (1.56) obținându-se

$$\sum_{i=0}^{2N-1} g(x_j) \sin(2\pi mx_j) = \sum_{i=0}^{2N-1} y_j \sin(2\pi mx_j) = b_m N .$$



Aproximarea prin serii Fourier

Un calcul analog pentru ceilalți coeficienți conduce la relațiile

$$a_k = \frac{1}{N} \sum_{j=0}^{2N-1} y_j \cos 2\pi kx_j, \quad b_m = \frac{1}{N} \sum_{j=0}^{2N-1} y_j \sin 2\pi mx_j \quad (1.58)$$

cu $k \in \overline{0, N}$, $m \in \overline{1, N-1}$.

În aplicații, coeficienții a_k , b_k se evaluează *mult* mai economic decât prin calculul direct al sumelor de mai sus prin folosirea transformării Fourier rapide prezentate în 1.1.4. Se poate lesne observa din (1.58) că vom avea toți $a_k = 0$ pentru funcții impare $f(-x) = -f(x)$ și toți $b_m = 0$ pentru funcții pare $f(-x) = f(x)$.

Apariția unei oarecare asimetrii – termenii în cos sunt mai numeroși decât cei în sin – este legată de alegerea unui număr par de $2N$ intervale în care se divide perioada funcției. Dacă se aleg $2N + 1$ intervale, forma funcției de interpolare este

$$g(x) = \frac{1}{2}a_0 + \sum_{k=1}^N [a_k \cos(2\pi kx) + b_k \sin(2\pi kx)] ,$$



Aproximarea prin serii Fourier

Pentru aproximarea funcțiilor periodice care satisfac condițiile *Dirichlet* se folosesc dezvoltările în serii Fourier sau descompunerea lor în armonice.

Fie o funcție periodică $f(t)$ de perioadă T , definită pe intervalul $[0, T]$ care satisfac condițiile *Dirichlet*, adică este o funcție uniform mărginită, are cel mult un număr finit de puncte de discontinuitate de speță întâi și un număr finit de puncte de maxim și minim. O astfel de funcție se poate dezvolta în serie *Fourier* conform relației:

$$f(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt) \quad (6.94)$$

în care coeficienții seriei *Fourier* a_0 , a_k și b_k se calculează cu ajutorul formulelor:

$$\begin{aligned} a_0 &= \frac{1}{T} \int_0^T f(t) dt; \\ a_k &= \frac{2}{T} \int_0^T f(t) \cos(k\omega t) dt; \\ b_k &= \frac{2}{T} \int_0^T f(t) \sin(k\omega t) dt. \end{aligned} \quad (6.95)$$

- Anton Hadar - Metode numerice în inginerie



Se întâlnesc următoarele două cazuri pentru valoarea perioadei T a funcției periodice $f(t)$:

$$a. \quad T = 2\pi, \text{ în acest caz: } \omega = \frac{2\pi}{T} = 1; \quad (6.96)$$

$$b. \quad T = \pi, \text{ în acest caz: } \omega = \frac{2\pi}{T} = 2. \quad (6.97)$$

Dacă funcția periodică $f(x)$ este definită domeniul $[a, b]$, atunci făcând schimbarea de variabilă:

$$\begin{aligned} t &= T \frac{x-a}{b-a} & \Rightarrow dt = \frac{Tdx}{b-a} \\ x = a &\Rightarrow t = 0; & x = b &\Rightarrow t = T \end{aligned} \quad (6.98)$$

se obține funcția periodică $f(t)$ având domeniul de definiție $[0, T]$.

Observații

- Dacă funcția periodică $f(t)$ definită pe intervalul $[-\pi, \pi]$ este *impară* atunci conform relațiilor (6.95) coeficienții a_k sunt nuli;
- Dacă funcția periodică $f(t)$ definită pe intervalul $[-\pi, \pi]$ este *pară* atunci conform acelorași relații, coeficienții b_k sunt nuli.



Exemplul 1 - Aproximare prin serii Fourier

Să se aproximeze prin serii Fourier funcția periodică impară de perioadă $T=2\pi$, definită astfel (fig. 6.3):

$$f(t) = \begin{cases} 1 & \text{pentru } t \in (0, \pi) \\ -1 & \text{pentru } t \in (\pi, 2\pi) \end{cases} \quad (6.99)$$

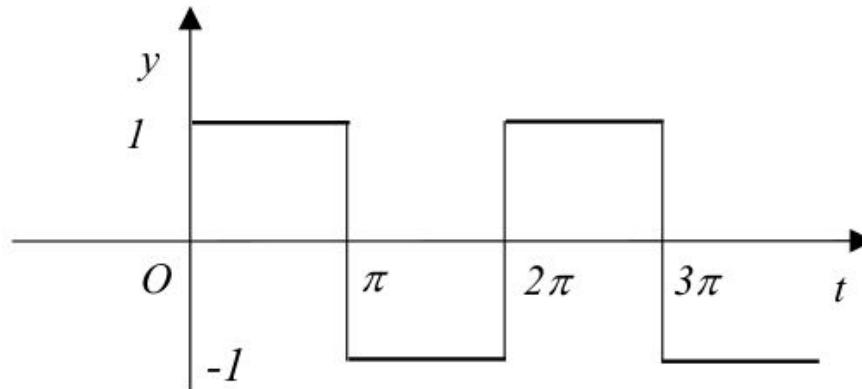


Fig.6.3

Coeficienții Fourier se calculează conform relațiilor (6.95):

$$\begin{aligned} a_0 &= \frac{1}{2\pi} \int_0^{2\pi} f(t) dt = \frac{1}{2\pi} \left[\int_0^{\pi} f(t) dt + \int_{\pi}^{2\pi} f(t) dt \right] = 0 \\ a_k &= \frac{1}{\pi} \int_0^{2\pi} f(t) \cos kt dt = \frac{1}{\pi} \left[\int_0^{\pi} \cos kt dt - \int_{\pi}^{2\pi} \cos kt dt \right] = 0 \quad (6.100) \\ b_k &= \frac{1}{\pi} \int_0^{2\pi} f(t) \sin kt dt = \frac{1}{\pi} \left[\int_0^{\pi} \sin kt dt - \int_{\pi}^{2\pi} \sin kt dt \right] = \frac{2}{\pi} \cdot \frac{1 - (-1)^k}{k} \end{aligned}$$



Exemplul 1 - Aproximare prin serii Fourier

Deoarece k este un număr natural, coeficienții b_k se mai scriu:

$$b_k = \begin{cases} 0 & \text{pentru } k = 2n \\ \frac{4}{\pi} \frac{1}{(2n-1)} & \text{pentru } k = 2n-1 \end{cases} \quad (6.101)$$

Desvoltarea în serie Fourier a funcției definite prin relația (6.99) se scrie:

$$f(t) = \frac{4}{\pi} \left(\frac{\sin t}{1} + \frac{\sin 3t}{3} + \frac{\sin 5t}{5} + \dots \right) \quad (6.102)$$

Pentru $t=\pi/2$ valoarea funcției este $f(\pi/2)=1$ iar din relația (6.102) rezultă:

$$1 = \frac{4}{\pi} \left(\frac{\sin(\pi/2)}{1} + \frac{\sin 3(\pi/2)}{3} + \frac{\sin 5(\pi/2)}{5} + \dots \right) \quad (6.103)$$

adică se obține suma seriei următoare:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4} \quad (6.104)$$



Exemplul 2 - Aproximare prin serii Fourier

Să se aproximeze cu ajutorul seriilor Fourier funcția periodică pară d perioadă $T=2\pi$ definită astfel (fig.6.4):

$$f(t) = \begin{cases} t & \text{pentru } t \in (0, \pi) \\ 2\pi - t & \text{pentru } t \in (\pi, 2\pi) \end{cases} \quad (6.105)$$

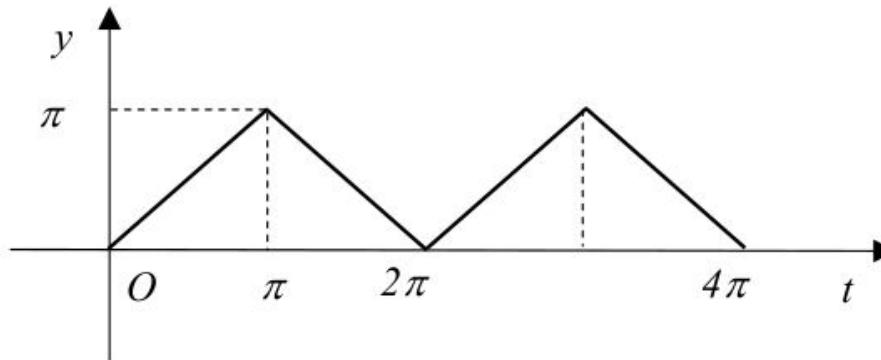


Fig.6.4

Coeficienții *Fourier* se calculează conform relațiilor (6.95):

$$\begin{aligned} a_0 &= \frac{1}{2\pi} \int_0^{2\pi} f(t) dt = \frac{1}{2\pi} \left[\int_0^{\pi} t dt + \int_{\pi}^{2\pi} (2\pi - t) dt \right] = \frac{\pi}{2} \\ a_k &= \frac{1}{\pi} \int_0^{2\pi} f(t) \cos kt dt = \frac{1}{\pi} \left[\int_0^{\pi} t \cos kt dt + \int_{\pi}^{2\pi} (2\pi - t) \cos kt dt \right] = \frac{2}{\pi} \cdot \frac{1 - (-1)^k}{k^2} \quad (6.106) \\ b_k &= \frac{1}{\pi} \int_0^{2\pi} f(t) \sin kt dt = \frac{1}{\pi} \left[\int_0^{\pi} t \sin kt dt + \int_{\pi}^{2\pi} (2\pi - t) \sin kt dt \right] = 0 \end{aligned}$$



Exemplul 2 - Aproximare prin serii Fourier

Deoarece k este un număr natural, coeficienții a_k se mai scriu:

$$a_k = \begin{cases} 0 & \text{pentru } k = 2n \\ \frac{4}{\pi} \frac{1}{(2n-1)^2} & \text{pentru } k = 2n-1 \end{cases} \quad (6.107)$$

Dezvoltarea în serie Fourier a funcției periodice (6.105) se scrie:

$$f(t) = \frac{\pi}{2} - \frac{4}{\pi} \left(\frac{\cos t}{1} + \frac{\cos 3t}{3^2} + \frac{\cos 5t}{5^2} + \dots \right) \quad (6.108)$$

Deoarece $f(0) = 0$, relația (6.108) pentru $t=0$ devine :

$$0 = \frac{\pi}{2} - \frac{4}{\pi} \left(1 + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \dots \right) \quad (6.109)$$

Rezultă suma seriei:

$$\frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \dots = \frac{\pi^2}{8} \quad (6.110)$$



Exemplul 3 - Aproximare prin serii Fourier

Să se aproximeze cu ajutorul seriilor *Fourier* funcția periodică impară de perioadă $T=2\pi$ definită astfel (fig.6.7):

$$f(t) = \begin{cases} t & \text{pentru } t \in (0, \pi/2) \\ -t + \pi & \text{pentru } t \in (\pi/2, 3\pi/2) \\ t - 2\pi & \text{pentru } t \in (3\pi/2, 2\pi) \end{cases} \quad (6.122)$$

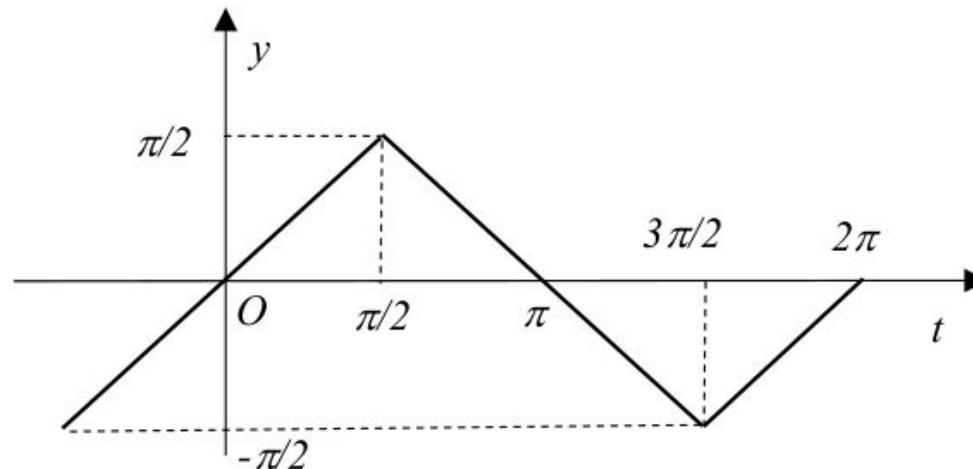


Fig. 6.7

Coeficienții *Fourier* se calculează conform relațiilor (6.95):

$$\begin{aligned} a_0 &= \frac{1}{2\pi} \int_0^{2\pi} f(t) dt = 0; & a_k &= \frac{1}{\pi} \int_0^{2\pi} f(t) \cos kt dt = 0 \\ b_k &= \frac{1}{\pi} \left[\int_0^{\pi/2} t \sin kt dt + \int_{\pi/2}^{3\pi/2} (-t + \pi) \sin kt dt + \int_{3\pi/2}^{2\pi} (t - 2\pi) \sin kt dt \right] = \frac{4}{\pi} \frac{\sin \frac{k\pi}{2}}{k^2} \end{aligned} \quad (6.123)$$



Exemplul 3 - Aproximare prin serii Fourier

Deoarece k este un număr natural, coeficienții b_k se mai scriu:

$$b_k = \begin{cases} 0 & \text{pentru } k = 2n \\ \frac{4}{\pi} \frac{-1}{(4n-1)^2} & \text{pentru } k = 4n-1 \\ \frac{4}{\pi} \frac{1}{(4n+1)^2} & \text{pentru } k = 4n+1 \end{cases} \quad (6.124)$$

Desvoltarea în serie Fourier a funcției (6.124) se scrie:

$$f(t) = \frac{4}{\pi} \left(\frac{\sin t}{1^2} - \frac{\sin 3t}{3^2} + \frac{\sin 5t}{5^2} - \frac{\sin 7t}{7^2} + \dots \right) \quad (6.125)$$

Deoarece $f(\pi/2) = \pi/2$ din relația (6.125) se obține suma seriei:

$$\frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \dots = \frac{\pi^2}{8} \quad (6.126)$$



Derivarea numerică

Lect. dr. Ioan Dumitru



Definitia matematică a derivatei unei functii

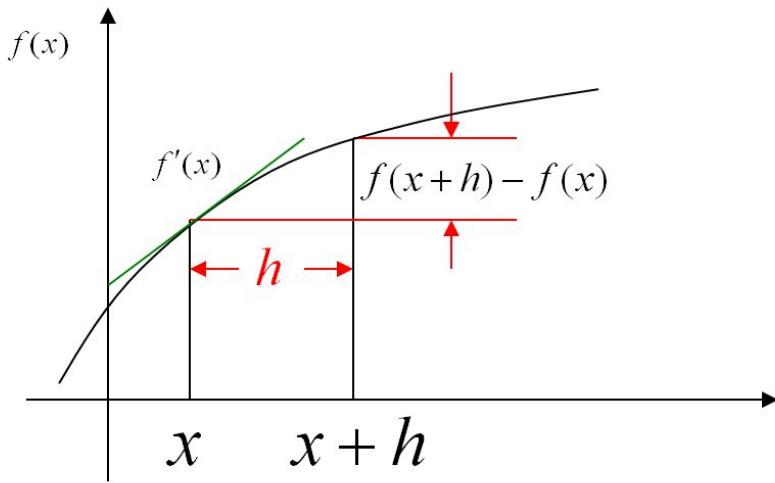
- Integrare și diferențierea numerică sunt unele dintre metodele cele mai des utilizate în fizica computațională.
- Definiția matematică a derivatei unei funcții $f(x)$ este

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

unde h este pasul de derivare.

- Derivata functiei reprezinta tangenta la graficul functiei
- Nu putem calcula direct limita cand $h > 0$ și va trebui să aproximăm limita pentru un pas h finit

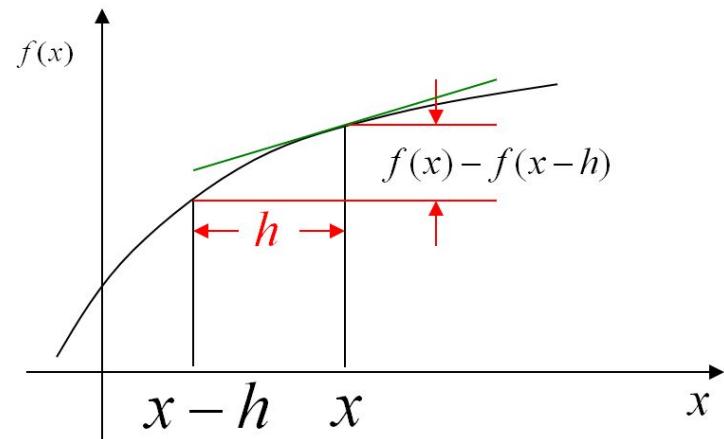
$$f'_c(x) \approx \frac{f(x + h) - f(x)}{h}$$



Derivata progresiva / regresiva

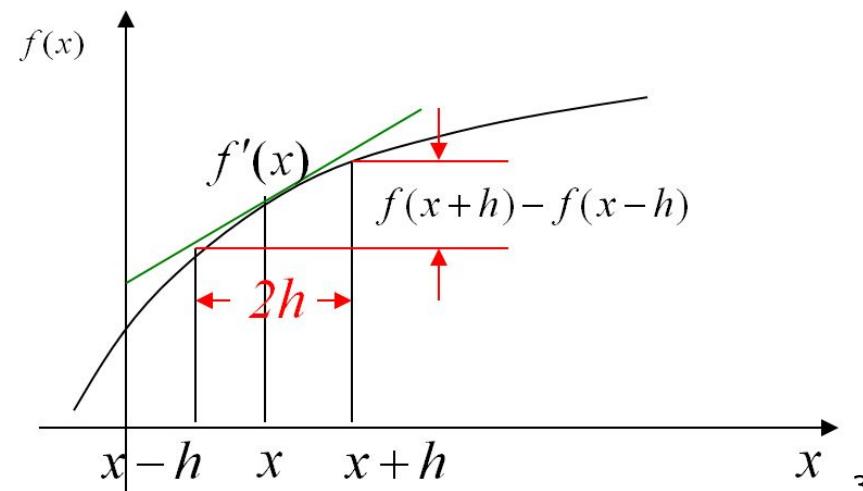
- Derivata la dreapta (progresiva) / stanga (regresiva)

$$\frac{f(x) - f(x-h)}{h}$$



- Putem media cele două deriveate
- Derivata centrata:

$$f'(x) \approx \frac{1}{2} \left(\frac{f(x+h) - f(x)}{h} + \frac{f(x) - f(x-h)}{h} \right) = \frac{f(x+h) - f(x-h)}{2h}$$



- Putem estima erorile de calcul?



Exemplu

- Calculam $f'(1)$ utilizand expresiile pentru derivatele la stanga (DS), dreapta (DD) si centrata (DC) pt doi pasi

$$f(x) = x^3 \quad (\therefore f'(x) = 3x^2, \quad f'(1) = 3)$$

- DS

$$h=0.1 \quad f'(1) = \frac{f(1.1) - f(1)}{0.1} = 3.31 \quad error = 0.31$$

$$h=0.05 \quad f'(1) = \frac{f(1.05) - f(1)}{0.05} = 3.1525 \quad error = 0.1525$$

- DD

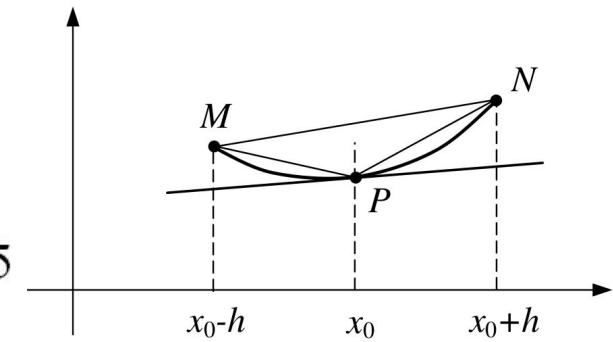
$$h=0.1 \quad f'(1) = \frac{f(1) - f(0.9)}{0.1} = 2.71 \quad error = 0.29$$

$$h=0.05 \quad f'(1) = \frac{f(1) - f(0.95)}{0.05} = 2.8453 \quad error = 0.1547$$

- DC

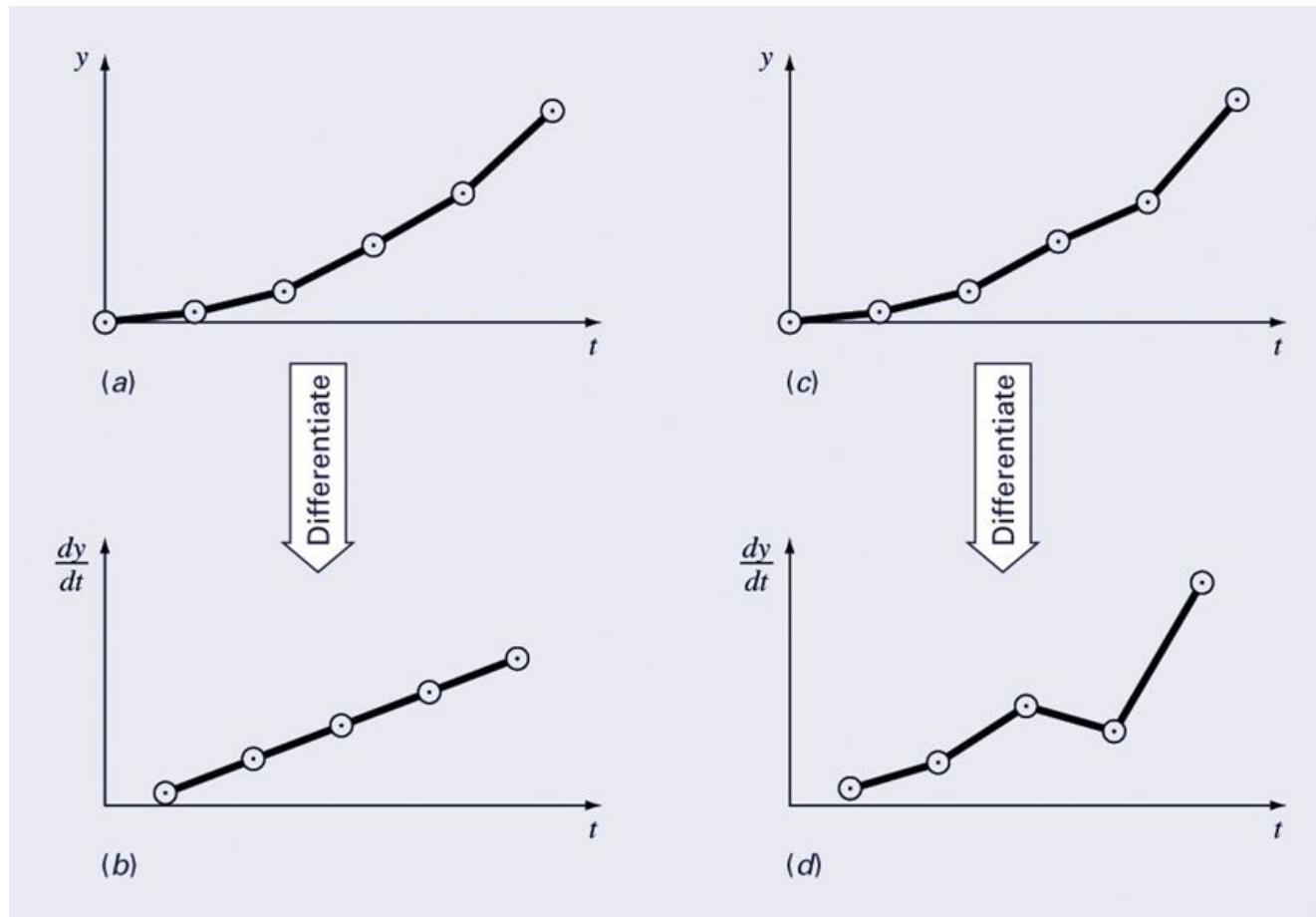
$$h=0.1 \quad f'(1) = \frac{f(1.1) - f(0.9)}{0.2} = 3.01 \quad error = 0.01$$

$$h=0.05 \quad f'(1) = \frac{f(1.05) - f(0.95)}{0.1} = 3.00250 \quad error = 0.0025$$



Amplificarea erorilor la derivare

- Trebuie avut în vedere faptul că derivarea numerică conduce la o amplificare a erorilor.



Derivata progresivă - regresivă

- Aceasta inseamna ca putem reprezenta derivata ca

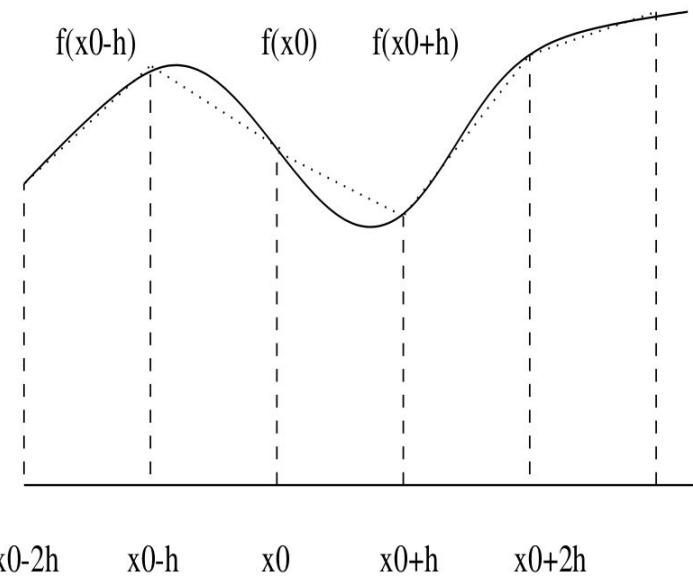
$$f'_2(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

unde indicele 2 se refera la faptul ca utilizam doua puncte pentru a defini derivata iar $O(h)$ reprezinta ordinul polinomului de eroare de ordinul 1 in h .

- Observam ca aceasta reprezinta derivata la dreapta a functiei (progresivă). Alternativ putem utiliza formula derivatei la stanga a functiei regresivă:

$$f'_2(x) = \frac{f(x) - f(x-h)}{h} + O(h)$$

- Dacă cea de a doua derivată este aproape zero această formulă poate fi utilizată pentru a aproxima derivata.



Exemplu

- Daca avem functia $f(x) = a + bx^2$ observam ca putem aproxima derivata ca:

$$f'_2(x) = 2bx + bh$$

in timp de valoarea exacta a derivatei este $2bx$

- Daca b nu este foarte mare putem aproxima derivata din ce in ce mai corect considerand valori din ce in ce mai mici pentru h . Totusi erorile de rotunjire $f(x + h) - f(x)$ ce apar la numarator devin din ce in ce mai mari.
- Un aspect important al aproximării numerice este stabilirea ordinului erorii $O(h^k)$. Simbolul O este folosit pentru a elimina constantele ce inmultesc expresia de interes de sub paranteze, in cazul de față h^k . Ordinul erorii este exponentul k .
- Se observă ca la micșorarea pasului h , eroarea e va scădea mai repede cu cât ordinul erorii k este mai mare. Spre exemplu, pentru $k = 1$ o injumătățire a pasului conduce tipic la o eroare de două ori mai mică, pe când pentru $k = 2$ eroarea scade de patru ori.



- Dacă utilizăm dezvoltarea Taylor pentru $f(x+h)$ putem scrie:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2 f''(x)}{2} + \dots$$

se observă că valoarea derivatei calculată conform definiției este

$$f'_c(x) \approx \frac{f(x+h) - f(x)}{h} \approx f'(x) + \frac{hf''(x)}{2} + \dots$$

- Relația dovedește că ordinul polinomului de eroare este 1 în cazul în care se utilizează derivata în două puncte.
- Observăm de asemenea că eroare data prin renunțarea la termenii superiori din dezvoltarea Taylor pentru funcție este mai mică decât pentru derivata.



Derivarea in trei puncte

- Vom incerca o aproximatie mai buna pentru derivata functiei utilizand trei puncte: x_0 , x_0-h , x_0+h .
- Dezvoltand functia in serii Taylor in punctele $x_0 +/- h$ avem:

$$f(x = x_0 \pm h) = f(x_0) \pm hf' + \frac{h^2 f''}{2} \pm \frac{h^3 f'''}{6} + O(h^4)$$

- ce se rescrie ca:

$$f_{\pm h} = f_0 \pm hf' + \frac{h^2 f''}{2} \pm \frac{h^3 f'''}{6} + O(h^4)$$

$$f_{-h} = f(x_0 - h)$$

$$f_0 = f(x_0)$$

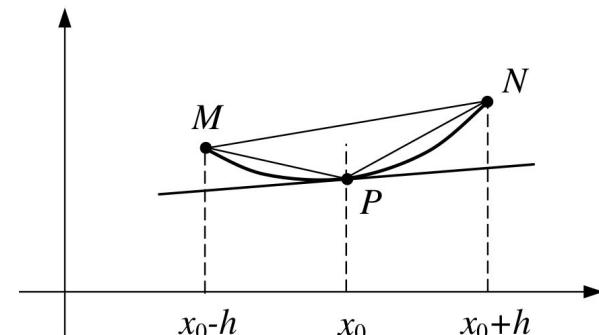
$$f_h = f(x_0 + h)$$

- Prin scaderea expresiilor obtinem

$$f'_3 = \frac{f_h - f_{-h}}{2h} - \frac{h^2 f'''}{6} + O(h^3)$$

- Se observa ca eroarea de trunchere este un polinom in h^2

- Pentru functia patratrica $f(x) = a + bx^2$ formula derivatei in trei puncte ofera valoarea exacta $2bx$



Derivata in cinci puncte

- Utilizand relatiile

$$f_h - 2f_0 + f_{-h} = h^2 f'' + O(h^4)$$

- obtinem expresiile derivatelor de ordin doi

$$f'' = \frac{f_h - 2f_0 + f_{-h}}{h^2} + O(h^2)$$

- In mod asemanator putem calcula derivata utilizand cinci puncte x_0-2h , x_0-h , x_0 , x_0+h , x_0+2h

$$f_{\pm 2h} = f_0 \pm 2hf' + 2h^2 f'' \pm \frac{4h^3 f'''}{3} + O(h^4)$$

- obtinandu-se

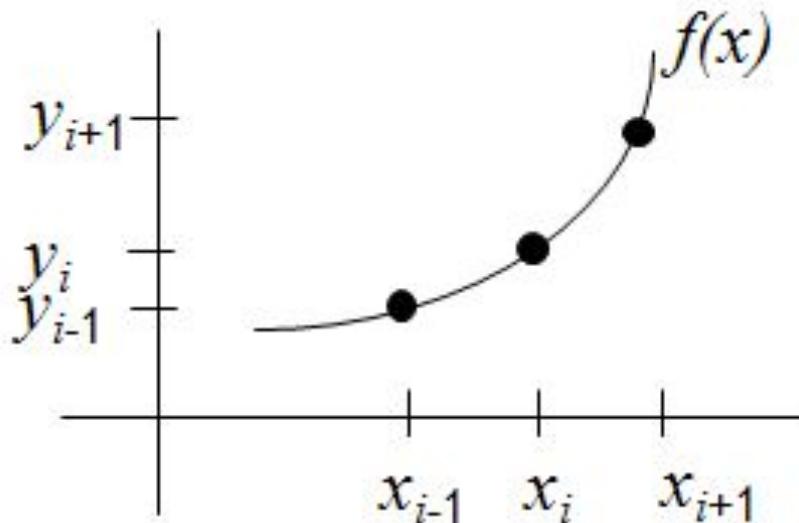
$$f'_{5c} = \frac{f_{-2h} - 8f_{-h} + 8f_h - f_{2h}}{12h} + O(h^4)$$

- in care eroarea dominantă este de ordinul h^4 .



Formule de derivare pentru diferențe finite

- Dacă valorile funcției sunt disponibile numai ca perechi de puncte $x_i, y_i = f(x_i)$, atunci valorile diferențelor finite sunt calculate prin formulele (presupunând că x_i sunt sortate)



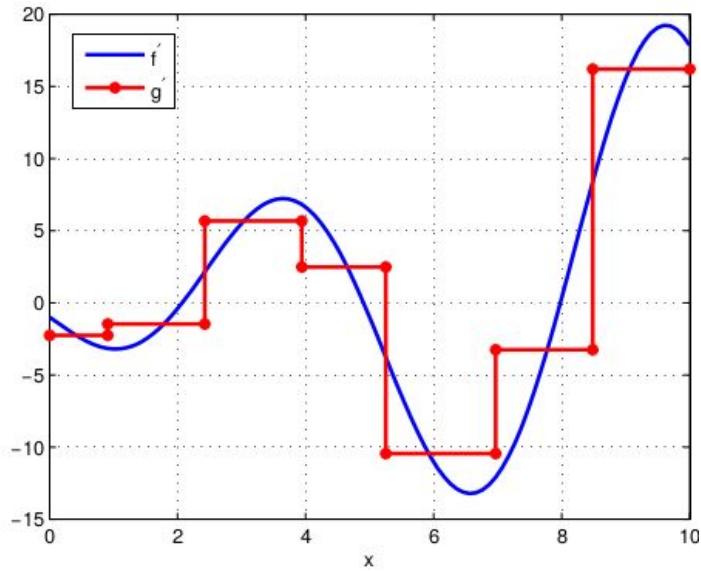
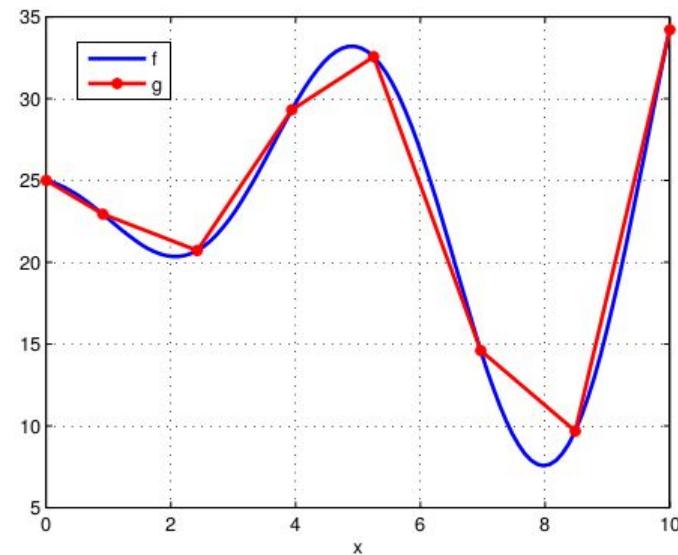
$$f'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}$$

x	x_0	x_1	x_2	\dots	x_n
$f(x)$	y_0	y_1	y_2	\dots	y_n

Derivata funcției nu poate fi calculată în toate punctele date

Interpolarea functiei si derivata

- Presupunem ca utilizam doua puncte si reprezentam functia f ca o linie intre valorile functiei in punctele respective.
- Functia reală f și interpolarea ei pe porțiuni g .
- Aproximarea derivatei f' cu valoarea g' .



Exemplu numeric

- În tabelul alăturat sunt prezentate rezultatele evaluării numerice, utilizând diferiți pasi de derivare, pentru derivata a doua a funcției $\exp(x)$ utilizând aproximarea

$$f'' = \frac{f_h - 2f_0 + f_{-h}}{h^2} + O(h^2)$$

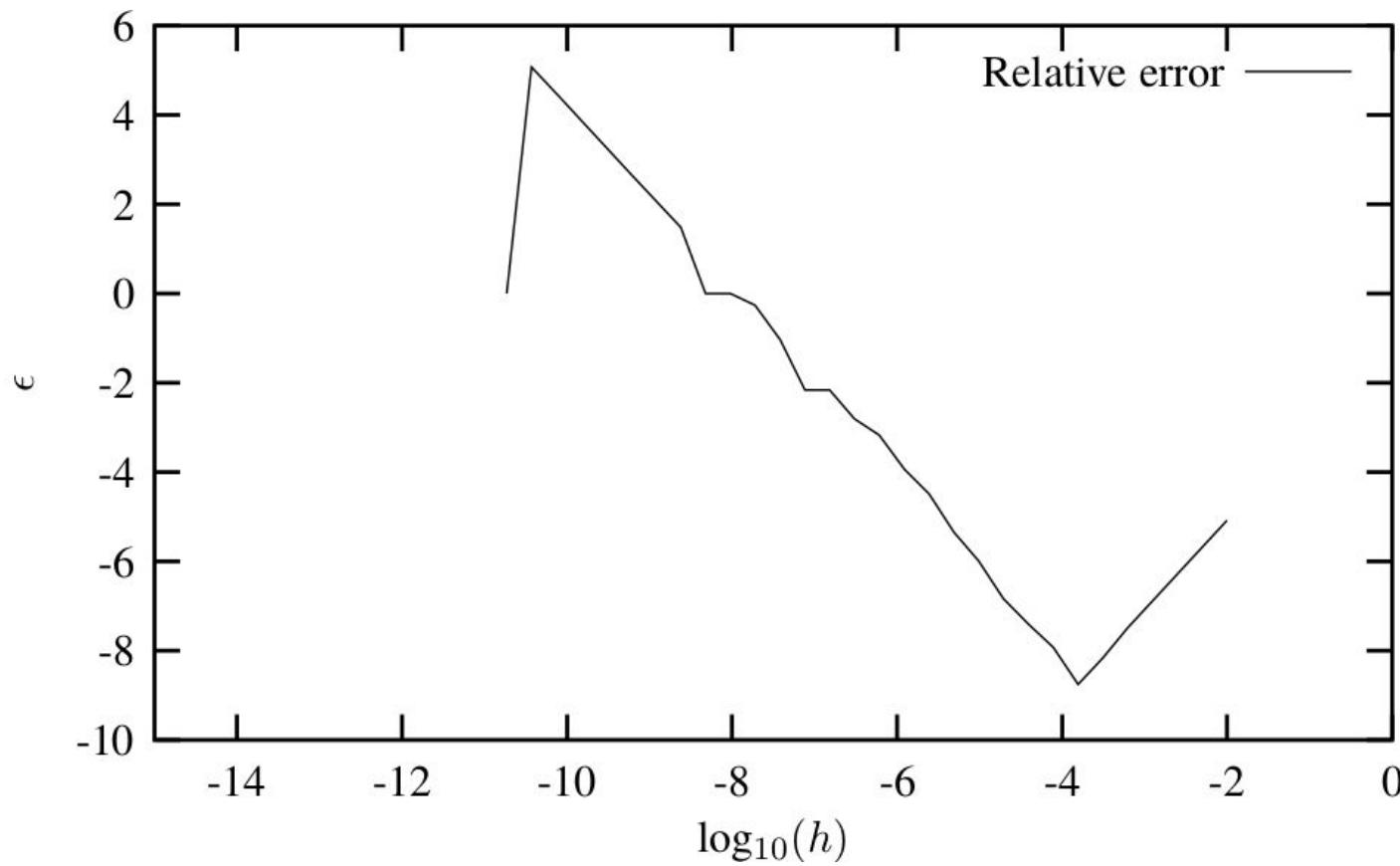
x	$h = 0.1$	$h = 0.01$	$h = 0.001$	$h = 0.0001$	$h = 0.000001$	Exact
0.0	1.000834	1.000008	1.000000	1.000000	1.010303	1.000000
1.0	2.720548	2.718304	2.718282	2.718282	2.753353	2.718282
2.0	7.395216	7.389118	7.389057	7.389056	7.283063	7.389056
3.0	20.102280	20.085704	20.085539	20.085537	20.250467	20.085537
4.0	54.643664	54.598605	54.598155	54.598151	54.711789	54.598150
5.0	148.536878	148.414396	148.413172	148.413161	150.635056	148.413159



Analiza erorii de calcul

- Dependenta Log-log a erorii relative a derivatei a doua pentru functia e^x de valoarea pasului h . Derivata a doua a fost calculata in punctul $x = 10$

$$\epsilon = \log_{10} \left(\left| \frac{f''_{\text{computed}} - f''_{\text{exact}}}{f''_{\text{exact}}} \right| \right)$$



Estimarea h optim în cazul formulei de derivare progresivă de ordinul

Deoarece $f \in C^2([a, b], \mathbb{R})$ și $x_0 \in (a, b)$ considerăm dezvoltarea în serie Taylor a lui f în punctul x_0 cu restul sub forma Lagrange de ordinul doi:

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!} \cdot h + \frac{f''(\xi)}{2!} \cdot h^2,$$

unde $\xi \in (x_0, x_0 + h) \subset [a, b]$. Deci

$$\left| \frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) \right| = \left| \frac{f''(\xi)}{2} \cdot h \right| \leq \frac{M_2}{2} \cdot h,$$

unde M_2 este o constantă care depinde de f și nu de h . Menționăm că deoarece $f \in C^2([a, b], \mathbb{R})$, deci f'' este continuă, conform teoremei lui Weierstrass f'' este mărginită pe $[a, b]$ și își atinge marginile, deci există constantă finită

$$M_2 = \sup\{|f''(x)| / x \in [a, b]\} = \max\{|f''(x)| / x \in [a, b]\}. \text{ q.e.d.}$$

Din evaluarea prezentată în propoziția anterioară deducem că dacă ne interesează valoarea lui $f'(x_0)$ cu o precizie $\varepsilon > 0$ atunci alegem h astfel ca să aibă loc inegalitatea $\frac{M_2}{2} \cdot h < \varepsilon$, adică $h < \frac{2\varepsilon}{M_2}$, din care rezultă că

$$\left| \frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) \right| < \varepsilon.$$



Exemplu

- **Exemplu.** Fie funcția: $f(x) = \ln(x)$, $x_0 = 1.8$

Formula diferențelor progresive pentru aproximarea derivatei în punctul x_0 este:

$$f'(1.8) = \frac{f(1.8 + h) - f(1.8)}{h}$$

cu o eroare: $Er = \frac{|h \cdot f''(\xi)|}{2} = \frac{|h|}{2\xi^2} \leq \frac{|h|}{2(1.8)^2}, \quad 1.8 < \xi < 1.8 + h$

h	$f(1.8+h)$	$\frac{f(1.8+h) - f(1.8)}{h}$	$\frac{ h }{2(1.8)^2}$
0.1	0.641853	0.54067	0.01543
0.01	0.593326	0.55401	0.00154
0.001	0.588342	0.55540	0.00015

Valoarea exactă a derivatei $f'(x) = \frac{1}{x}$ în punctul x_0 este

$$f'(1.8) = 0.555$$

iar limitele erorii sunt destul de apropiate de eroarea de aproximare.



Efectul erorilor de rotunjire

Pe lângă erorile de trunchiere, un rol important în cazul derivării numerice îl joacă și erorile de rotunjire. De exemplu, putem scrie:

$$f(x_0 - h) = y_{-1} + e_{-1} \quad \text{și} \quad f(x_0 + h) = y_1 + e_1$$

unde cele două funcții sunt approximate prin y_{-1} și y_1 , fiindu-le asociate erorile de rotunjire e_{-1} respectiv e_1 . Acum se poate exprima derivata funcției f prin:

$$f'(x_0) = \frac{y_1 - y_{-1}}{2h} + E(f, h)$$

unde termenul de eroare va conține atât termenul de trunchiere cât și cel de rotunjire:

$$E(f, h) = \frac{e_1 - e_{-1}}{2h} - \frac{h^2 f^{(3)}(c)}{6}$$

Dacă $|e_{-1}| \leq \epsilon$, $|e_1| \leq \epsilon$ și $M = \max(|f^{(3)}(x)|)$, atunci putem exprima eroarea prin:

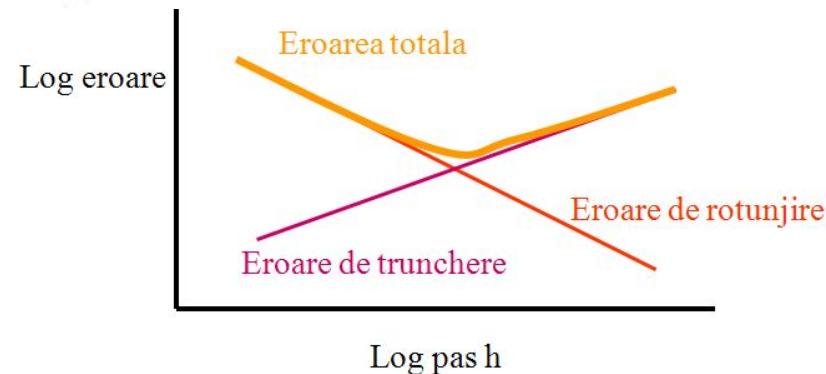
$$|E(f, h)| \leq \frac{\epsilon}{h} + \frac{Mh^2}{6}$$

eroare ce se va mininimiza pentru:

$$h = \left(\frac{3\epsilon}{M}\right)^{1/3}$$

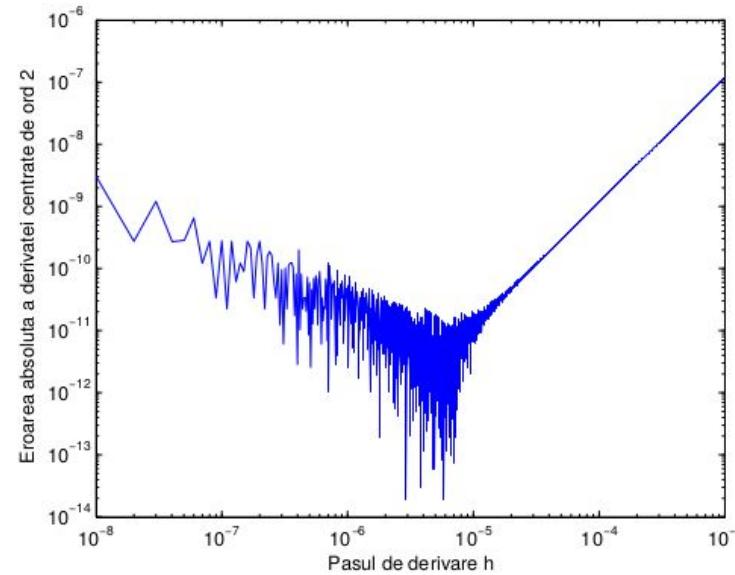
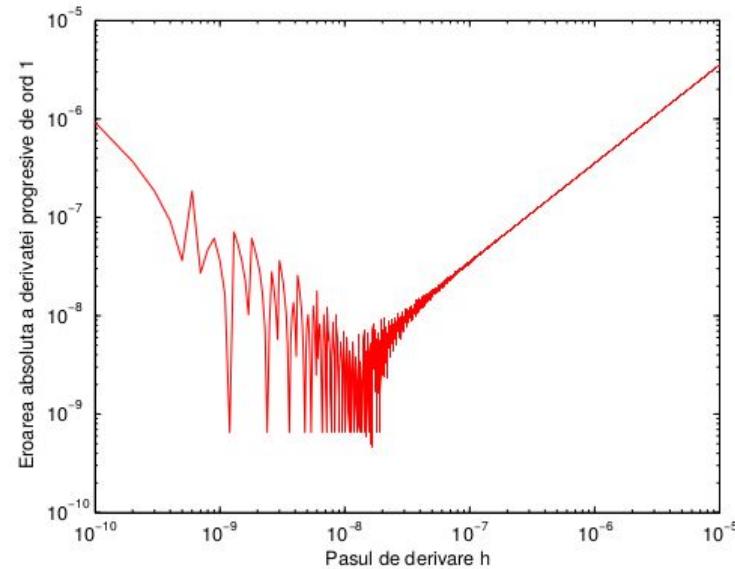
Analog, pentru cazul obținerii derivatei unei funcții de ordin 4, se găsește:

$$|E(f, h)| \leq \frac{3\epsilon}{2h} + \frac{Mh^4}{30} \quad \text{și} \quad h = \left(\frac{45\epsilon}{4M}\right)^{1/5}$$



Exemplu numeric - eroarea totală ca funcție de pas

- Pasul de discretizare trebuie ales de utilizator.
- Eroarea de trunchiere este cu atât mai mică cu cât pasul este mai mic. Dar, există și o eroare de rotunjire, care este cu atât mai mare cu cât pasul este mai mic.
- Teste numerice pentru $\sin'(\pi/4)$: Eroarea în funcție de pasul de derivare pentru formula de derivare progresivă de ordinul 1 (sus) și formula de derivare centrată de ordinul 2 (jos).
- Pas optim de derivare numerică = pasul de la care încep să predomine erorile de trunchiere.
- Atenție: Nu este pasul pentru care eroarea este minimă!



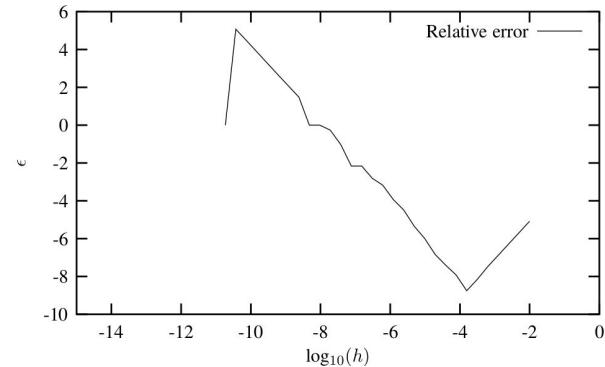
Evaluarea pasului pentru minimizarea erorii

Pentru cazul derivatei a doua calculate anterior cu formula

$$f_0'' = \frac{f_h - 2f_0 + f_{-h}}{h^2}$$

eroarea de trunchere este

$$\epsilon_{\text{approx}} \approx \frac{f_0^{(4)}}{12} h^2$$



iar cea de rotunjire se calculeaza plecand de la formula derivatei

$$f_0'' = \frac{f_h - 2f_0 + f_{-h}}{h^2} = \frac{(f_h - f_0) + (f_{-h} - f_0)}{h^2}$$

cu $(f_{\pm h} - f_0) \approx \epsilon_M$

pentru simpla precizie $|\epsilon_M| \leq 10^{-7}$ iar pentru dubla precizie $|\epsilon_M| \leq 10^{-15}$

Eroarea totala este: $|\epsilon_{\text{tot}}| \leq \frac{2\epsilon_M}{h^2} + \frac{f_0^{(4)}}{12} h^2$

care se minimizeaza (egaland cu zero derivata erorii totale in raport cu h) pentru

$$h = \left(\frac{24\epsilon_M}{f_0^{(4)}} \right)^{1/4}$$

Considerand reprezentarea in duble precizie, pentru $x = 10$, $h \approx 10^{-4}$



Extrapolarea Richardson

- Presupunem ca derivatele de ordin superior există și că avem o valoare fixă pentru x , atunci putem scrie:

$$f'(x) = \frac{1}{2h} [f(x+h) - f(x-h)] + a_2 h^2 + a_4 h^4 + a_6 h^6 + \dots$$

- Extrapolarea Richardson examinează variația operatorului următor ca funcție de h :

$$\varphi(h) = \frac{1}{2h} [f(x+h) - f(x-h)]$$

- Acesta funcție aproximează $f'(x)$ la $O(h^2)$.
- Da miscoram pasul h prin injumatatire

$$\varphi(h) = f'(x) - a_2 h^2 - a_4 h^4 - a_6 h^6 - \dots$$

$$\varphi\left(\frac{h}{2}\right) = f'(x) - a_2 \left(\frac{h}{2}\right)^2 - a_4 \left(\frac{h}{2}\right)^4 - a_6 \left(\frac{h}{2}\right)^6 - \dots$$

- Multiplicând ultima ecuație cu 4 și scăzând-o din prima ecuație, termenul în h^2 se anulează și avem



Extrapolarea Richardson

$$\varphi(h) - 4\varphi\left(\frac{h}{2}\right) = -3f'(x) - \frac{3}{4}a_4h^4 - \frac{15}{16}a_6h^6 - \dots$$

$$f'(x) = \varphi\left(\frac{h}{2}\right) + \frac{1}{3} \left[\varphi\left(\frac{h}{2}\right) - \varphi(h) \right] + O(h^4)$$

Estimare nouă

Estimare initială

- ordinului erorii a fost crescut la $O(h^4)$ -> precizia de calcul creste -> Ok!!!!.
- Putem crește ordinul erorii la h^6 ? Da, prin miscorarea pasului la $h/4$.
- Pentru a efectua calculele considerăm urmatoarea proprietate:

$$\begin{aligned}\varphi(h) &= f'(x) - \sum_{k=1}^{\infty} a_{2k} h^{2k} \\ &= L - \sum_{k=1}^{\infty} a_{2k} h^{2k}\end{aligned}$$

- unde L este necunoscută $L = \lim_{h \rightarrow 0} \varphi(h) = f'(x)$ atât timp cat nu cunoaștem a_{2k} .



Extrapolarea Richardson

- Nu uitam

$$\varphi(h) = \frac{1}{2h} [f(x+h) - f(x-h)]$$

- și introducem urmatoarele notatii

$$\begin{aligned} D(n,0) &\equiv \varphi\left(\frac{h}{2^n}\right) \\ &= L + \sum_{k=1}^{\infty} A(k,0)\left(\frac{h}{2^n}\right)^{2k} \end{aligned}$$

- $D(n,0)$ reprezinta derivata centrala pentru diferite valori ale lui h . Incercam sa calculam $D(n,0)$ pentru cateva valori ale lui n .
- Rescriem formula de calcul a derivatei calculata anterior pentru ordinului erorii $O(h^4)$ folosind noile notatii:

$$\begin{aligned} f'(x) &= \varphi\left(\frac{h}{2}\right) + \frac{1}{3} \left[\varphi\left(\frac{h}{2}\right) - \varphi(h) \right] + O(h^4) \\ &= D(1,0) + \frac{1}{4-1} [D(1,0) - D(0,0)] + O(h^4) \end{aligned}$$



Extrapolarea Richardson

- Daca reducem pasul h la h/2 atunci putem scrie la modul general:

$$f'(x) = D(n, 0) + \frac{1}{4-1} [D(n, 0) - D(n-1, 0)] + O\left(\left(\frac{h}{2^n}\right)^4\right)$$

- si notand:

$$D(n, 1) = D(n, 0) + \frac{1}{4^1 - 1} [D(n, 0) - D(n-1, 0)]$$

- sau la modul general

$$D(n, m) = D(n, m-1) + \frac{1}{4^m - 1} [D(n, m-1) - D(n-1, m-1)]$$

$$(1 \leq m \leq n)$$

↑
Estimare nouă

↑
Estimare initială

- Folosind acesti termeni rezulta:

$$D(n, m) = L + \sum_{k=m+1}^{\infty} A(k, m) \left(\frac{h}{2^n}\right)^{2k}$$



Extrapolarea Richardson

- Deoarece $m \leq n$ rezulta o matrice bidimensională triunghiulară pentru valorile D:
- Trebuie să dam o valoare initială pentru h și numărul maxim de iteratii N

$$\begin{matrix} D(0,0) \\ D(1,0) & D(1,1) \\ D(2,0) & D(2,1) & D(2,2) \\ \vdots & \vdots & \vdots & \ddots \\ D(N,0) & D(N,1) & D(N,2) & \cdots & D(N,N) \end{matrix}$$

$$f(x) = \frac{(\cos(100x^2))^5}{x^3}$$

$$x = 1.3, h = \frac{1}{128}$$

$$\text{extrapolate } \frac{1}{3}$$

$$h = \frac{1}{4096} = 0.00244$$

$$\text{extrapolate } \frac{1}{15}$$

$$D(0,0) = 16.696386$$

$$D(1,0) = 40.583393 \quad D(1,1) = 48.583393$$

$$D(2,0) = 109.322528 \quad D(2,1) = 132.235574 \quad D(2,2) = 137.814897$$

$$D(3,0) = 135.031747 \quad D(3,1) = 143.601487 \quad D(3,2) = 144.359214$$

$$D(4,0) = 142.068615 \quad D(4,1) = 144.414238 \quad D(4,2) = 144.468421$$

$$D(5,0) = 143.866937 \quad D(5,1) = 144.466377 \quad D(5,2) = 144.469853$$



Extrapolarea Richardson

16.696386					
40.583393	48.583393				
109.322528	132.235574	137.814897			
135.031747	143.601487	144.359214	144.463092		
142.068615	144.414238	144.468421	144.470154	144.470182	
143.866937	144.466377	144.469853	144.469876	144.469875	$D(5,5) = 144.469875$

extrapolate $\frac{1}{3}$

extrapolate $\frac{1}{15}$

extrapolate $\frac{1}{63}$

extrapolate $\frac{1}{255}$

extrapolate $\frac{1}{1023}$

- Eroarea de calcul a derivatei este:

$$\begin{aligned} D(5,5) &= L + \sum_{k=5+1}^{\infty} A(k,5) \left(\frac{h}{2^5} \right)^{2k} \\ &= f'(1.3) + A(6,5) \left(\frac{1}{4096} \right)^{12} + \sum_{k=7}^{\infty} A(k,5) \left(\frac{h}{2^5} \right)^{2k} \end{aligned}$$

- Valoarea exactă a derivatei este:

$$f'(1.3) = 144.469874253\dots$$



Derivarea utilizand interpolarea polinomiala Lagrange

- Determinam functia care trece prin punctele experimentale utilizand o interpollare Lagrange polinomiala

$$f(x) = f(x_{i-1}) \frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + f(x_i) \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} + f(x_{i+1}) \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$

$$f'(x) = f(x_{i-1}) \frac{2x - x_i - x_{i+1}}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + f(x_i) \frac{2x - x_{i-1} - x_{i+1}}{(x_i - x_{i-1})(x_i - x_{i+1})} + f(x_{i+1}) \frac{2x - x_{i-1} - x_i}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$

- nu este necesar ca punctele sa fie echidistante



Derivate calculate folosind funcțiile spline

- Funcția spline polinomială de ordinul 3 poate fi utilizată pentru calculul aproximativ al derivatelor de ordinul 1 și 2. În acest scop, se determină coeficienții m_i, a_i, b_i , restricția funcției spline pe intervalul (x_i, x_{i+1}) fiind

$$p_{3,i}(x) = y_i + m_i(x - x_i) + b_i(x - x_i)^2 + a_i(x - x_i)^3$$

- derivatele de ordin 1 și doi se obțin atunci ca

$$y' = m_i + 2b_i(x - x_i) + 3a_i(x - x_i)^2, \quad y'' = 2b_i + 6a_i(x - x_i)$$

- Eroarea la aproximarea derivatei de ordinul k poate fi evaluată cu ajutorul relației

$$\max_{x \in [a,b]} |f^{(k)}(x) - s_m^{(k)}(x)| \leq \frac{(b-a)^{m-k}}{(m-k)!} \max_{x \in [a,b]} |f^{(m)}(x) - s_m^{(m)}(x)|$$

- unde m este ordinul funcției spline ($m = 3$ pentru funcția spline cubică). Deoarece funcția spline este derivabilă de $m - 1$ ori pe (a, b) , am notat $s_m^{(m)}$ funcție treaptă, obținută prin derivarea restricțiilor funcției s pe subintervale.



Derivate folosind interpolarea polinomiala Fourier

- Functia se poate aproxima prin

$$g(x) = \frac{1}{2}a_0 + \sum_{k=1}^{N-1} [a_k \cos(2\pi kx) + b_k \sin(2\pi kx)] + \frac{1}{2}a_N \cos 2\pi Nx$$

- Atunci derivata functiei se poate aproxima prin

$$\frac{df}{dx} = y' \cong \frac{dg}{dx} =$$

$$2\pi \sum_{k=1}^{N-1} [-ka_k \sin(2\pi kx) + kb_k \cos(2\pi kx)] - \pi N a_N \sin 2\pi Nx$$

- Rezultatul este o nouă funcție ce are coeficienții Fourier $-2\pi ka_k, -2\pi kb_k$
- Prin analogie cu analiza de ordin de eroare de la formulările anterioare, spunem că aproximarea trigonometrică a derivatei este de ordin infinit.
- Aplicabilitatea generală a procedeului este însă limitată de cerința de netezime și considerarea unui numar cat mai mare de puncte.



Derivate partiale

- Estimarea derivatelor partiale ale unei funcții se face folosind aceleași abordări prezentate pentru derivatele totale. De exemplu, să considerăm o funcție reală ce depinde de două variabile, definită pe un domeniu dreptunghiular

$$f(x, y) : [a, b] \times [c, d] \rightarrow \mathbb{R}$$

- Domeniul de definiție se discretizează cu ajutorul a două rețele de puncte, una pe axa Ox, iar cealaltă pe axa Oy:

$$a = x_0, x_1, \dots, x_i, \dots, x_n = b$$

$$c = y_0, y_1, \dots, y_j, \dots, y_m = d$$

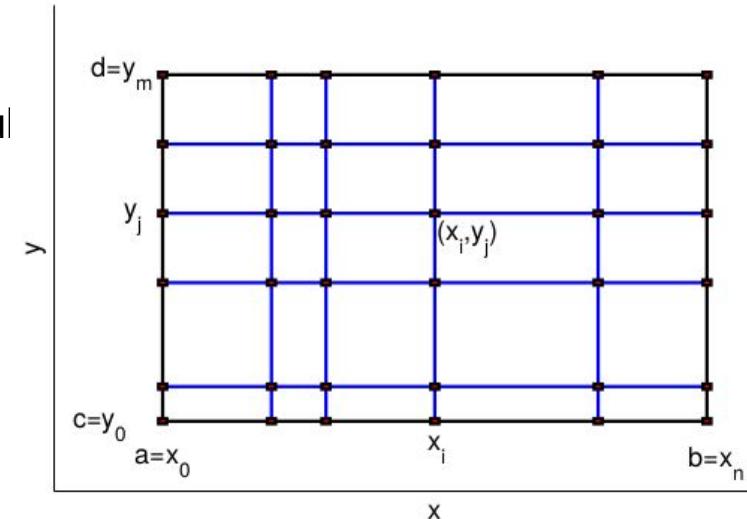
- Un nod al acestei rețele bidimensionale este caracterizat de coordonatele (x_i, y_j) , unde $i = 0, n; j = 0, m$ iar

$$f(x_i, y_j) = z_{i,j}, \quad i = 0, n \quad j = 0, m$$

- Evaluarea numerică a derivatelor partiale se reduce la evaluarea numerică a unor derivate obișnuite, deoarece

$$\frac{\partial f}{\partial x} \Big|_{x=x_0, y=y_0} = \lim_{x \rightarrow x_0} \frac{f(x, y_0) - f(x_0, y_0)}{x - x_0}$$

$$\frac{\partial f}{\partial y} \Big|_{x=x_0, y=y_0} = \lim_{y \rightarrow y_0} \frac{f(x_0, y) - f(x_0, y_0)}{y - y_0}$$



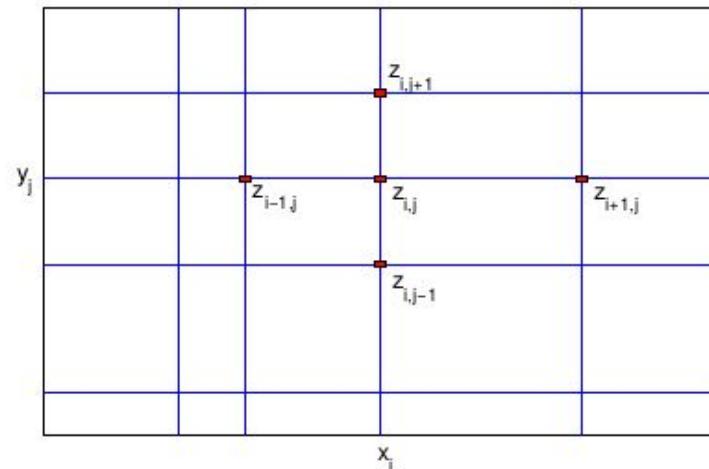
Derivate partiale

- Formulele de derivare progresivă de ordinul 1 pentru calculul derivatelor sunt:

$$\frac{\partial g}{\partial x}(x_i, y_j) = \frac{z_{i+1,j} - z_{i,j}}{x_{i+1} - x_i}, \quad \frac{\partial g}{\partial y}(x_i, y_j) = \frac{z_{i,j+1} - z_{i,j}}{y_{j+1} - y_j}$$

- Discretizarea operatorului Laplace

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$



- În cazul în care pașii de discretizare pe cele două rețele sunt egali: $x_{i+1} - x_i = y_{j+1} - y_j = h$, pentru orice $i = 0, n - 1; j = 0, m - 1$ și folosind formula de derivare în trei puncte de ordinul 2 rezultă că o aproximare numerică a Laplaceanului funcției f într-un punct (x_i, y_j) poate fi calculată ca:

$$\Delta g = \frac{z_{i-1,j} - 2z_{i,j} + z_{i+1,j}}{h^2} + \frac{z_{i,j-1} - 2z_{i,j} + z_{i,j+1}}{h^2} = \frac{z_{i-1,j} + z_{i+1,j} + z_{i,j-1} + z_{i,j+1} - 4z_{i,j}}{h^2}$$



GSL - exemplu derivare functie

- Urmatorul cod estimeaza derivata functiei $f(x) = x^{3/2}$ in punctele $x=2$ si $x=0$. Functia $f(x)$ nu este definita pentru $x<0$ si derivata in $x=0$ este calculata utilizand `gsl_deriv_forward`.

```
#include <stdio.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_deriv.h>

double f (double x, void * params)
{
    (void)(params); /* avoid unused parameter warning */
    return pow (x, 1.5);
}

int
main (void)
{
    gsl_function F;
    double result, abserr;

    F.function = &f;
    F.params = 0;

    printf ("f(x) = x^(3/2)\n");

    gsl_deriv_central (&F, 2.0, 1e-8, &result, &abserr);
    printf ("x = 2.0\n");
    printf ("f'(x) = %.10f +/- %.10f\n", result, abserr);
    printf ("exact = %.10f\n\n", 1.5 * sqrt(2.0));

    gsl_deriv_forward (&F, 0.0, 1e-8, &result, &abserr);
    printf ("x = 0.0\n");
    printf ("f'(x) = %.10f +/- %.10f\n", result, abserr);
    printf ("exact = %.10f\n", 0.0);

    return 0;
}
```

Function: int `gsl_deriv_central` (const `gsl_function` * *f*, double *x*, double *h*, double * *result*, double * *abserr*)

This function computes the numerical derivative of the function *f* at the point *x* using an adaptive central difference algorithm with a step-size of *h*. The derivative is returned in *result* and an estimate of its absolute error is returned in *abserr*.

The initial value of *h* is used to estimate an optimal step-size, based on the scaling of the truncation error and round-off error in the derivative calculation. The derivative is computed using a 5-point rule for equally spaced abscissae at $x-h$, $x-h/2$, x , $x+h/2$, $x+h$, with an error estimate taken from the difference between the 5-point rule and the corresponding 3-point rule $x-h$, x , $x+h$. Note that the value of the function at *x* does not contribute to the derivative calculation, so only 4-points are actually used.

```
$ ./a.out
f(x) = x^(3/2)
x = 2.0
f'(x) = 2.1213203120 +/- 0.0000005006
exact = 2.1213203436

x = 0.0
f'(x) = 0.0000000160 +/- 0.0000000339
exact = 0.0000000000
```



Bibliografie

- Gabriela Ciuprina - Algoritmi numerici pentru calcule științifice în ingineria electrică, Editura MatrixROM, 2013, <http://www.lmn.pub.ro/~gabriela/>
- Corneliu Berbente - Metode Numerice, Ed. Tehnică, 1998
- T.A. Beu, Calcul numeric în C, Editia a 2-a, Ed. Alabastra, Cluj-Napoca, 2000
- W.H. Press, S.A. Teukolsky, W.T. Vetterling și B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Second Edition (Cambridge University Press, Cambridge, 1992).



- Formula lui Taylor cu restul lui Lagrange

Teorema 1.37 (Formula lui Taylor cu restul lui Lagrange) Fie $I \subset \mathbb{R}$ un interval deschis, $a \in I$ și $n \in \mathbb{N}$. Dacă $f : I \rightarrow \mathbb{R}$ este o funcție de $(n + 1)$ ori derivabilă pe I , atunci pentru orice $x \in I$, $x \neq a$ există $c \in (x, a)$ sau $c \in (a, x)$ astfel încât

$$\begin{aligned}f(x) &= f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \dots \\&\quad + \frac{f^{(n)}(a)}{n!}(x - a)^n + \frac{f^{(n+1)}(c)}{(n + 1)!} \cdot (x - a)^{n+1}.\end{aligned}$$



Exemplu de calcul a derivatei

Example:

The **heat flux** at the soil-air interface can be computed with *Fourier's Law*:

$$q(z=0) = -k\rho C \frac{dT}{dz} \Big|_{z=0}$$

q = heat flux

k = coefficient of thermal diffusivity in soil ($\approx 3.5 \times 10^{-7} \text{ m}^2/\text{s}$)

ρ = soil density ($\approx 1800 \text{ kg/m}^3$)

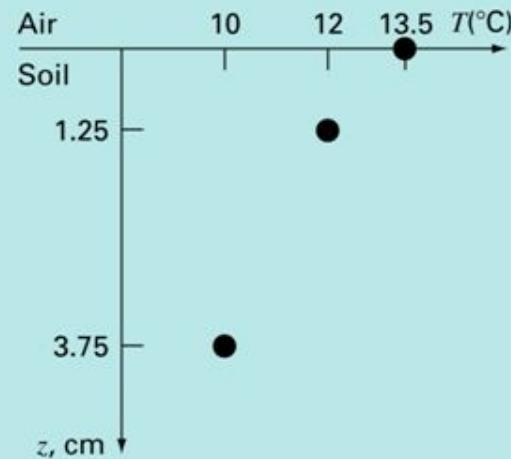
C = soil specific heat ($\approx 840 \text{ J/kg} \cdot \text{C}^\circ$)

*Positive flux value means heat is transferred from the air to the soil

Calculate dT/dz ($z=0$) first and then determine the heat flux.

A temperature gradient can be measured down into the soil as shown below.

MEASUREMENTS



$$\begin{aligned} f'(z=0) &= 13.5 \frac{2(0) - 1.25 - 3.75}{(0 - 1.25)(0 - 3.75)} \\ &\quad + 12 \frac{2(0) - 0 - 3.75}{(1.25 - 0)(1.25 - 3.75)} \\ &\quad + 10 \frac{2(0) - 0 - 1.25}{(3.75 - 0)(3.75 - 1.25)} \\ &= -14.4 + 14.4 - 1.333 = -1.333 \text{ } ^\circ\text{C/cm} \end{aligned}$$

which can be used to compute the **heat flux** at $z=0$:

$$q(z=0) = -3.5 \times 10^{-7} (1800)(840)(-133.3 \text{ } ^\circ\text{C/m}) = 70.56 \text{ W/m}^2$$



Caracterizarea metodelor de integrare numerică

- Integrarea numerică se aplică funcțiilor care nu pot fi integrate analitic (sau ar necesita calcule complicate) sau în cazul funcțiilor date tabelar, de exemplu cele rezultate experimental.
- Se dorește determinarea aproximativă a integralei în punctele unde se cunoaște valoarea funcției, cât și în alte puncte.
- În general, metodele de integrare numerică se bazează pe aproximarea funcției de integrat prin functii mai simple, a căror integrală se poate evalua ușor. O metodă de aproximare frecvent utilizată în cadrul integralelor este metoda interpolării polinomiale pe porțiuni.
- Spre deosebire de derivarea numerică, integrarea este o operație relativ stabilă numeric, dar mai lentă, din cauza numărului sporit de calcule care trebuie efectuate.



Exemplu

- Stabilirea cantităților de energie consumate, pe baza înregistrărilor de putere.
- Se consideră un receptor de energie electrică pentru care se cunoaște puterea activă consumată în decursul unei zile. Se cere să se determine energia activă zilnică consumată de receptor, pe baza prelucrării curbei de sarcină prin integrare numerică.

$$W_{zi} = \int_0^{24} P(t) \cdot dt$$

t [h]	P [MW]
0	1,42
2	1,06
4	2,17
6	2,83
8	2,75
10	2,67
12	2,59
14	2,55
16	2,57
18	2,84
20	3,43
22	3,02
24	1,53

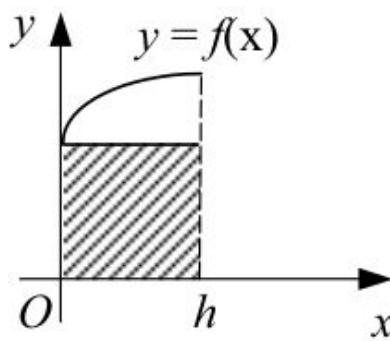


Prima teoremă a valorii medii

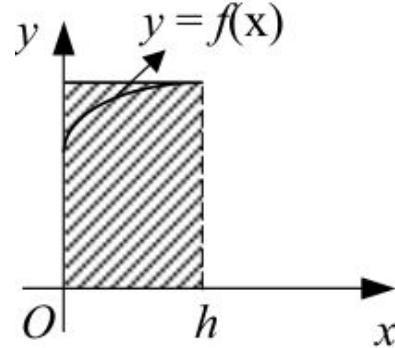
Dacă $f \in C([a, b], \mathbb{R})$ este o funcție continuă atunci există un punct $\xi \in [a, b]$ astfel încât

$$\int_a^b f(x)dx = f(\xi)(b - a)$$

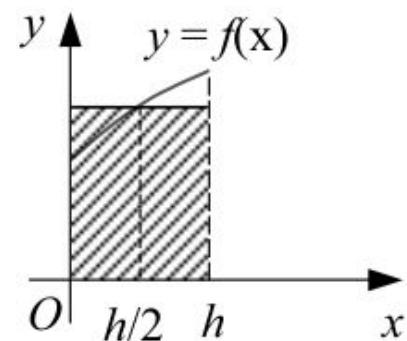
Prima dată vom deduce formula elementară a dreptunghiului: fie $f : [0, h] \rightarrow \mathbb{R}$ o funcție de clasă $C^1([0, h], \mathbb{R})$, adică o dată derivabilă și cu prima derivată continuă. Atunci avem următoarele formule elementare pentru calculul ariei folosind formule de aproximare cu ajutorul ariilor unor dreptunghiuri:



$$\int_0^h f(t)dt \approx h \cdot f(0)$$



$$\int_0^h f(t)dt \approx h \cdot f(h)$$



$$\int_0^h f(t)dt \approx h \cdot f(h/2)$$



Marginirea integralei

- O metoda de integrare comună este bazată pe integrarea pe subintervale

$$\int_{x_0}^{x_n} f(x) dx$$

- Divizând $[x_0, x_n]$ în n subintervale ($n > 1$)

$$\int_{x_0}^{x_n} f(x) dx = \int_{x_0}^{x_1} f(x) + \int_{x_1}^{x_2} f(x) + \cdots + \int_{x_{n-1}}^{x_n} f(x)$$

- Presupunem $f(x) > 0$ în toate punctele
- În fiecare subinterval gasim o valoare maximă pentru funcție și avem

$$\int_{x_0}^{x_n} f(x) \leq \sum_{i=0}^{n-1} M_i (x_{i+1} - x_i) \quad M_i = \sup \{f(x) : x_i \leq x \leq x_{i+1}\}$$

- dar și o valoare minimă

$$\int_{x_0}^{x_n} f(x) \geq \sum_{i=0}^{n-1} m_i (x_{i+1} - x_i) \quad m_i = \inf \{f(x) : x_i \leq x \leq x_{i+1}\}$$

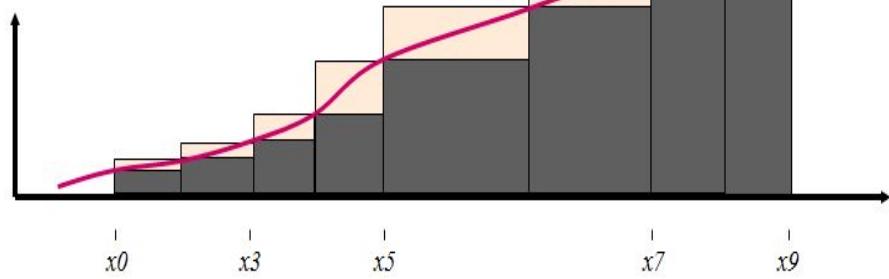
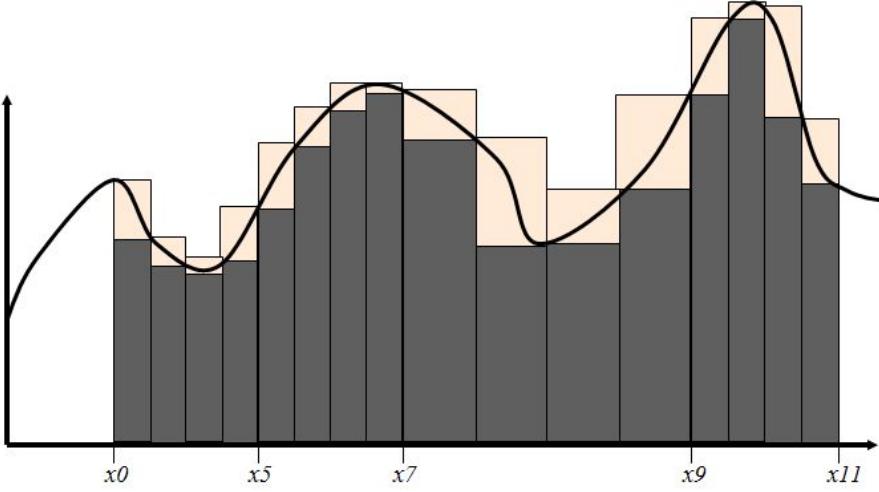
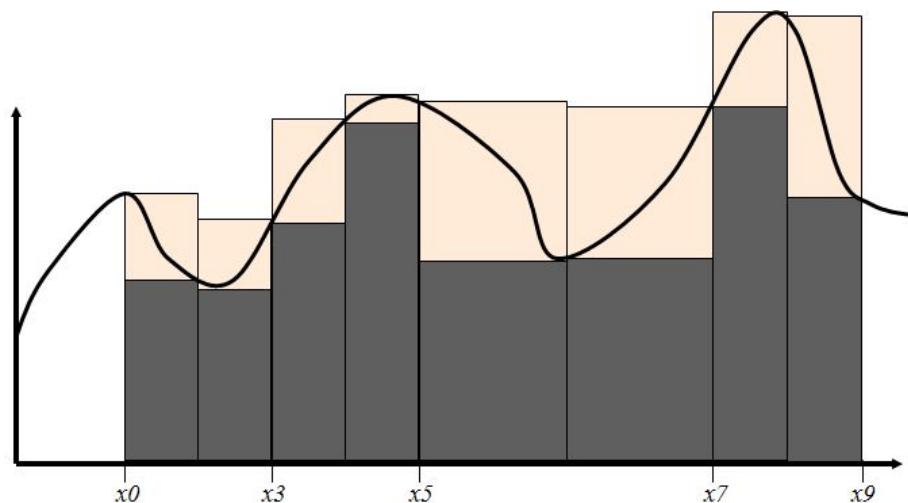
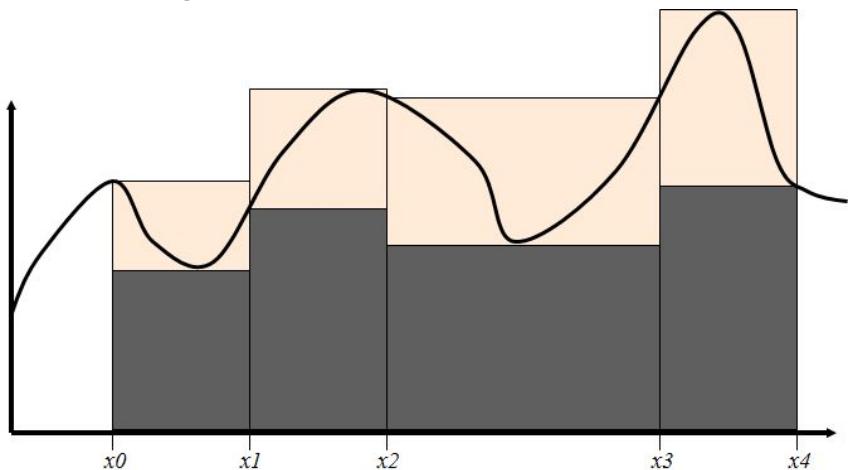
- și integrala este marginita

$$\sum_{i=0}^{n-1} m_i (x_{i+1} - x_i) \leq \int_{x_0}^{x_n} f(x) \leq \sum_{i=0}^{n-1} M_i (x_{i+1} - x_i)$$



Efectul cresterii numarului de subintervale

- Daca $n \rightarrow \infty$ limita superioara si inferioara se apropiie si spunem ca functiile sunt functii integrabile Riemann



- Da functia este monoton crescatoare limita inferioara coincide cu cea superioara din pasul precedent



Principiul metodei. Metoda trapezelor.

Se consideră funcția reală $f : [a, b] \rightarrow \mathbb{R}$, definită prin cod sau tabelar, într-o rețea de $n + 1$ noduri: x_0, x_1, \dots, x_n . Se formulează problema calculului integralei:

$$I = \int_a^b f(x) dx$$

- Pe o rețea de $n + 1$ noduri echidistante, pasul rețelei are valoarea

$$h = \frac{x_n - x_0}{n}$$

- Metoda se bazează pe aproximarea variației funcției de integrat

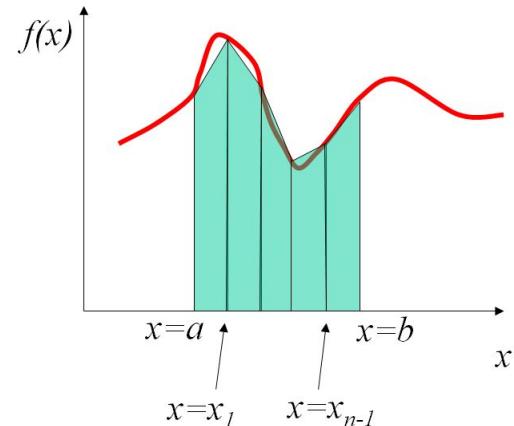
între două noduri succesive ale rețelei printr-un polinom de gradul întâi (o dreaptă). Aceasta echivalează cu aproximarea ariei subîntinse de graficul funcției între două noduri succesive (i și $i + 1$), cu aria trapezului "sprijinit" de axa Ox, determinat de abscisele x_i , x_{i+1} și ordonatele $f(x_i)$ și $f(x_{i+1})$. Prin însumarea ariilor tuturor trapezelor de acest fel care se formează între punctele a și b , se obține valoarea aproximativă a integralei definite a lui f , între aceste puncte.

- Aria unui trapez este

$$I_i = \frac{h(f(x_i) + f(x_{i+1}))}{2}$$

- Rezultă, prin însumare pentru $i = 0, \dots, n - 1$, formula de integrare prin metoda trapezelor:

$$I = \int_a^b f(x) dx \approx \frac{h}{2}(f(x_0) + 2f(x_1) + \dots + 2f(x_{n-1}) + f(x_n))$$



Metoda Simpson 1/3

Se aproximează variația funcției de integrat între trei noduri succesive $(i - 1, i, i + 1)$, printr-un polinom de interpolare de gradul doi: $P(x) = a_0 + a_1x + a_2x^2$ (o parabolă).

Presupunând aceeași rețea de $n + 1$ noduri echidistante, dar cu n par, și considerând originea în punctul x_i , se poate scrie:

$$f(x_{i-1}) = P(-h) = a_0 - a_1h + a_2h^2,$$

$$f(x_i) = P(0) = a_0,$$

$$f(x_{i+1}) = P(h) = a_0 + a_1h + a_2h^2.$$

Așadar, se aproximează aria subîntinsă de funcția de integrat între punctele x_{i-1} și x_{i+1} cu aria subîntinsă de parabola P dintre punctele $-h$ și h :

$$I_i = \int_{-h}^h P(x) dx = \int_{-h}^h (a_0 + a_1x + a_2x^2) dx = 2ha_0 + 2\frac{h^3}{3}a_2$$

Din condițiile de interpolare se calculează:

$$a_0 = f(x_i);$$

$$a_2 = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{2h^2}$$



Metoda Simpson 1/3

necesare în evaluarea integralei

$$I_i = \frac{h(f(x_{i-1}) + 4f(x_i) + f(x_{i+1}))}{3}.$$

Însumând toate ariile I_i , pentru $i = 1, 3, 5, \dots, n - 1$, rezultă

$$I = \int_a^b f(x) dx \approx \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \dots + f(x_n)),$$

cunoscută sub numele de *Formula lui Simpson 1/3*.



Exemplu

- Sa se calculeze integrala functiei $f(x) = x^3$ intre $x=1$ si $x=2$
- Prin calcul analitic avem:

$$\int_1^2 x^3 dx = \frac{1}{4} x^4 \Big|_1^2 = \frac{1}{4} (2^4 - 1^4) = 3.75$$

- Pentru calcul discretizam intervalul $[1,2]$ in 4 subintervale

$$\Delta x = \frac{2-1}{4} = 0.25$$

- Conform regulii dreptunghiului avem:

i	x_i^*	$f(x_i^*)$
1	1.125	1.42
2	1.375	2.60
3	1.625	4.29
4	1.875	6.59

$$\begin{aligned}\int_1^2 x^3 dx \\ &= \Delta x [f(1.125) + f(1.375) + f(1.625) + f(1.875)] \\ &= 0.25(14.9) = 3.725\end{aligned}$$



Estimarea erorii la integrarea numerică

- Strategia este să se găsească o dezvoltare în serie Taylor pentru $f(x)$ în subintervale.

- Evaluam

$$\int_{-h}^{+h} f(x) dx$$

- considerând dezvoltarea în serii Taylor a funcției în jurul punctului x_0

$$f(x = x_0 \pm h) = f(x_0) \pm hf' + \frac{h^2 f''}{2} \pm \frac{h^3 f'''}{6} + O(h^4)$$

- Impartim integrala în două integrale una de la $-h$ la x_0 și una de la x_0 la h și putem utiliza expresia derivatei numerice în două puncte:

$$f'(x_0 \pm h) = \frac{\mp f(x_0 \pm h) \pm f(x_0)}{h}$$

- Pe fiecare subinterval avem:

$$f(x) = f_0 + \frac{f_h - f_0}{h}x + O(x^2) \quad x = x_0 \quad x = x_0 + h$$

$$f(x) = f_0 + \frac{f_0 - f_{-h}}{h}x + O(x^2) \quad x = x_0 - h \quad x = x_0$$



Estimarea erorii la integrarea numerica - regula trapezului

- Dupa integrare se obtine:

$$\int_{-h}^{+h} f(x)dx = \frac{h}{2} (f_h + 2f_0 + f_{-h}) + O(h^3)$$

expresia fiind similara cu cea oferita de regula trapezului.

- Subliniem ca eroarea calculata

$$O(h^3) = O((b-a)^3/N^3)$$

este una locala. Pentru ca am impartit intervalul $[a,b]$ in N subintervale trebuie sa efectuam N adunari de arii si astfel eroarea totala devine

$$O(h^2)$$



Estimarea erorii la integrarea numerica - regula Simpson

- Similar, calculam integrala folosind trei puncte si expresiile specifice ale derivatelor de ordin 1 si 2:

$$f(x) = f_0 + \frac{f_h - f_{-h}}{2h}x + \frac{f_h - 2f_0 + f_{-h}}{2h^2}x^2 + O(x^3)$$

$$\int_{-h}^{+h} f(x)dx = \frac{h}{3} (f_h + 4f_0 + f_{-h}) + O(h^5)$$

care reprezinta regula Simpson.

- Observam ca utilizand o aproximatie mai buna in evaluarea derivatelor obtinem o mai buna aproximare a integralei, ordinul erorii locale fiind in acest caz $O(h^5)$, eroarea totala fiind

$$O(h^4)$$



Integrarea Romberg

Valoarea numerică a integralei depinde de pasul de integrare h , eroarea de trunchiere scăzând cu micșorarea pasului h . Se notează cu I_0 valoarea exactă a integralei și cu $I = I(h)$ valoarea ei numerică. Deoarece funcția $I(h)$ este pară (trapezele obținute pornind din a cu pasul h sunt aceleași cu cele obținute pornind din b cu pasul $-h$), prin dezvoltarea ei în serie Taylor în jurul lui 0 se obține, în cazul metodei trapezelor:

$$I = I_0 + a_0 h^2 + a_1 h^4 + a_2 h^6 + \dots$$

Reținând din această serie numai doi termeni, se obțin pentru două valori diferite h_1 și h_2 ale pasului de integrare:

$$I_1 = I_0 + a_0 h_1^2;$$

$$I_2 = I_0 + a_0 h_2^2.$$

Din cele două ecuații rezultă următoarea aproximare pentru valoarea exactă a integralei:

$$I_0 = \frac{h_2^2 I_1 - h_1^2 I_2}{h_2^2 - h_1^2},$$

cunoscută sub numele de *formula de integrare Romberg*.



Integrarea Romberg

Reducând succesiv valoarea lui h : $h_1 > h_2 > h_3 > \dots > h_k \dots$, se obțin valorile integralei numerice:

$$I_{0k} = \frac{h_{k+1}^2 I_k - h_k^2 I_{k+1}}{h_{k+1}^2 - h_k^2}, \quad k = 1, 2, \dots,$$

care au precizie din ce în ce mai bună.

Formula lui Romberg se simplifică, dacă la fiecare iterație se înjumătățește pasul de integrare ($h_{k+1} = h_k/2$):

$$I_{0k} = \frac{4I_{k+1} - I_k}{3}$$

Valorile numerice ale integralelor $I_{01}, I_{02}, I_{03}, \dots$, calculate pe rețele tot mai fine, pot fi folosite pentru calculul unei valori numerice de precizie ridicată, prin aplicarea recursivă a formulei Romberg:

- Nr. intervale: $n \quad 2n \quad 4n \dots$
- Met. trapezelor: $I_{01} \quad I_{02} \quad I_{03} \dots$
- Romberg ord. 1: $I_{11} = \frac{4I_{02} - I_{01}}{3} \quad I_{12} = \frac{4I_{03} - I_{02}}{3} \dots$
- Romberg ord. 2: $I_{21} = \frac{16I_{12} - I_{11}}{15} \dots$



În general, formula lui Romberg de ordinul k pentru înjumătățirea pasului este

$$I_{mk} = \frac{4^m I_{m-1,k+1} - I_{m-1,k}}{4^m - 1}$$

cu $k = 1, 2, \dots$ și $m = 1, 2, \dots$. Iterațiile se pot opri pentru acel ordin m , la care eroarea relativă

$$\left| \frac{I_{mk} - I_{m-1,k}}{I_{mk}} \right|$$

este mai mică decât o valoare impusă.



Exemplu

- Regula trapezului

i	x_i	$f(x_i)$
	1	1
1	1.25	1.95
2	1.5	3.38
3	1.75	5.36
	2	8

$$\begin{aligned} & \int_1^2 x^3 dx \\ &= \Delta x \left[\frac{1}{2} f(1) + f(1.25) + f(1.5) + f(1.75) + \frac{1}{2} f(2) \right] \\ &= 0.25(15.19) = 3.80 \end{aligned}$$

- Regula Simpson

$$\begin{aligned} \int_1^2 x^3 dx &= \frac{\Delta x}{3} [f(1) + 4f(1.25) + 2f(1.5) + 4f(1.75) + f(2)] \\ &= \frac{0.25}{3} (45) = 3.75 \end{aligned}$$



Calculul numeric al integralelor duble

- Pentru simplitate vom considera domeniul de integrare al funcției de două variabile un dreptunghi

$$\iint_D f(x,y)dx = \int_a^b \int_c^d f(x,y)dxdy$$

- Se împart în subintervalele $[a, b]$ și $[c, d]$

$$h = \frac{b-a}{n}, \quad k = \frac{d-c}{m}$$

și se consideră dreptunghiul cu vârfurile

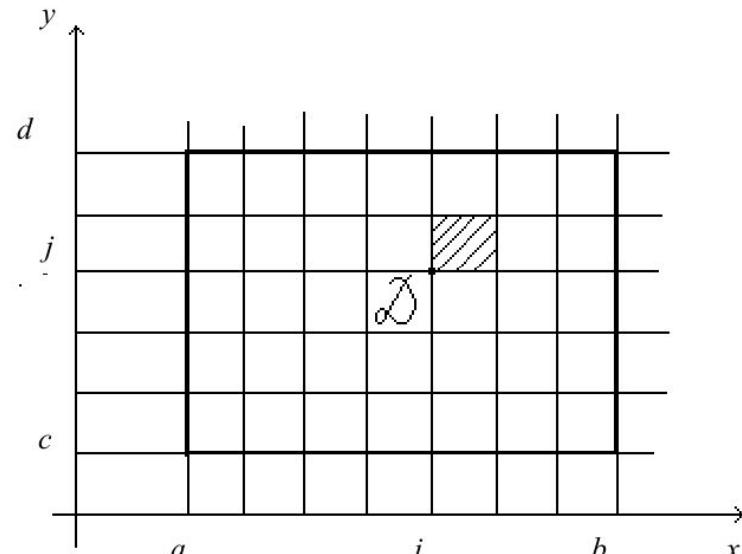
$$[x_i, y_i], [x_{i+1}, y_i], [x_{i+1}, y_{i+1}], [x_i, y_{i+1}]$$

unde

$$x_i = a + i \cdot h, \quad y_j = c + j \cdot k$$

- Pentru dreptunghiul dat care conține vârful $[x_i, y_j]$ se calculează integrala I_{ij} aplicând **formula trapezului**.

$$\begin{aligned} I_{ij} &= \int_{x_i}^{x_{i+1}} dx \int_{y_j}^{y_{j+1}} f(x,y) dxdy \approx \int_{x_i}^{x_{i+1}} \left\{ \frac{k}{2} \left[f(x, y_j) + f(x, y_{j+1}) \right] \right\} dx = \\ &= \frac{k}{2} \left[\int_{x_i}^{x_{i+1}} f(x, y_j) dx + \int_{x_i}^{x_{i+1}} f(x, y_{j+1}) dx \right] = \end{aligned}$$



$$I_{ij} = \frac{k \cdot h}{4} [f(x_i, y_j) + f(x_i, y_{j+1}) + f(x_{i+1}, y_j) + f(x_{i+1}, y_{j+1})]$$

- Integrala pe întreg dreptunghiul $[a, b, c, d]$ este:

$$I \cong \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} I_{ij} = \frac{kh}{4} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} [f(x_i, y_j) + f(x_i, y_{j+1}) + f(x_{i+1}, y_j) + f(x_{i+1}, y_{j+1})]$$

expresie cunoscută sub numele de formula de cubatură a trapezului.



Calcul numeric al integralelor duble - formula Simpson de cubatura

- Pentru același dreptunghi $[a, b, c, d]$ vom aplica formula lui Simpson de integrare. Vom considera dreptunghiul de integrare cu vârfurile

$$[x_i, y_i], [x_{i+1}, y_i], [x_{i+1}, y_{i+1}], [x_i, y_{i+1}]$$

- și cu punctul central $[x_i, y_i]$

$$\begin{aligned} I_{ij} &\cong \int_{x_{i-1}}^{x_{i+1}} \int_{y_{j-1}}^{y_{j+1}} f(x,y) dx dy = \int_{x_{i-1}}^{x_{i+1}} \frac{k}{3} \left[f(x, y_{j-1}) + 4f(x, y_j) + f(x, y_{j+1}) \right] dx = \\ &= \frac{kh}{9} \left\{ f(x_{i-1}, y_{j-1}) + f(x_{i+1}, y_{j-1}) + f(x_{i-1}, y_{j+1}) + f(x_{i+1}, y_{j+1}) + \right. \\ &\quad \left. + 4 \left[f(x_i, y_{j+1}) + f(x_i, y_{j-1}) + f(x_{i-1}, y_j) + f(x_{i+1}, y_j) \right] \right\} + 16f(x_i, y_i) \end{aligned}$$

- Integrala pe întreg dreptunghiul $[a, b, c, d]$ este dată de formula de cubatură a lui Simpson:

$$\begin{aligned} I &= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} I_{ij} = \frac{kh}{9} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \left\{ f(x_{i-1}, y_{j-1}) + f(x_{i+1}, y_{j-1}) + f(x_{i-1}, y_{j+1}) + f(x_{i+1}, y_{j+1}) + \right. \\ &\quad \left. + 4 \left[f(x_i, y_{j+1}) + f(x_i, y_{j-1}) + f(x_{i-1}, y_j) + f(x_{i+1}, y_j) \right] + 16f(x_i, y_j) \right\} \end{aligned}$$



Metoda de integrare Monte Carlo

- Analiza numerică clasică - *metode deterministe* - algoritmii ofera același rezultat pe orice calculator prin aceeași succesiune finită de operații.
- În fizică - sisteme cu număr mare de grade de libertate.
- Ex: Funcția de partiție clasică a unui gaz din N atomi care interacționează prin $v(r_{ij})$:

$$Z = \int \cdots \int d^3r_1 \cdots d^3r_N \exp \left[-\frac{1}{k_B T} \sum_{i < j}^N v(r_{ij}) \right]$$

Pentru $N = 20$ și $n = 10$ puncte de integrare pe fiecare dimensiune $\sim 10^{60}$ operații.
Supercalculator — 10^{10} operații / secundă $\sim 3 \cdot 10^{42}$ ani!

- Alternativa -> metode probabilistice.
- Metoda Monte Carlo - nume generic după faimosul cazinou din Monaco.
- *Metodă nedeterministă* - utilizează rezultatele unor experiente aleatoare.
- Mărimea căutată = valoarea medie a unei mărimi aleatoare.
- Mod mai avantajos de scalare a efortului de calcul cu creșterea dimensiunii problemei.
- Evaluarea integralelor simple prin metoda Monte Carlo este ineficientă. Calculul integralelor multiple prin metoda Monte Carlo mult mai avantajoasă odată cu creșterea dimensiunii.



Metoda de integrare Monte Carlo

Teorema fundamentală a metodei Monte Carlo — integrala unei funcții f pe un domeniu \mathcal{D} — produsul dintre valoarea medie a funcției și volumul V al domeniului,

$$\int_{\mathcal{D}} f dV \approx V \langle f \rangle \pm \sigma$$

$\langle f \rangle$ — pe un eșantion de n puncte, $\mathbf{x}_1, \dots, \mathbf{x}_n$, distribuite **aleator** și **uniform** în \mathcal{D} .

σ — **deviația standard** — incertitudinea — măsură a abaterii funcției f de la medie.

σ^2 — **varianță**.

$$\sigma = V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{n}} = \frac{V}{\sqrt{n}} \sigma_f$$

$$\langle f \rangle \equiv \frac{1}{n} \sum_{i=1}^n f(\mathbf{x}_i), \quad \langle f^2 \rangle \equiv \frac{1}{n} \sum_{i=1}^n f^2(\mathbf{x}_i)$$

Comparație cu eroarea formulelor deterministe de integrare

σ descrește lent numărul punctelor de eșantionare $\sim n^{-1/2}$ — nu depinde de dimensiune!

Formula trapezelor unidimensională — eroarea $\sim n^{-2}$

Formula trapezelor d -dimensională — eroarea $\sim n^{-2/d}$ (n evaluări / dimensiune).

Pentru $d > 4$ metoda Monte Carlo devine mai eficientă.



Particularizare - metoda de integrare Monte Carlo

- Integrala unidimensională

$$I = \int_a^b f(X) dX$$

Schimbare de variabilă $X = (b - a)x + a$:

$$I = (b - a) \int_0^1 f((b - a)x + a) dx$$

Formula Monte Carlo:

$$I \approx \frac{b - a}{n} \sum_{i=1}^n f((b - a)x_i + a)$$

x_i — n abscise aleatoare uniform distribuite în $[0, 1]$

$X_i = (b - a)x_i + a$ — punctele de eşantionare din $[a, b]$.

- Exemplu:

$$\int_0^1 xe^{-x} dx = 1 - 2e^{-1} \approx 0.26424$$

n	I	σ
100	0.26906	0.01080
1000	0.26324	0.00337
10000	0.26416	0.00105



Integrale multidimensionale

Integrarea funcției f pe domeniul complex \mathcal{D} , pentru care eșantionarea este dificilă.
Domeniu "extins" $\tilde{\mathcal{D}}$ (include \mathcal{D}), care poate fi eșantionat cu ușurință (hiper-paralelipiped).
Nou integrand

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{dacă } \mathbf{x} \in \mathcal{D} \\ 0 & \text{dacă } \mathbf{x} \notin \mathcal{D}, \end{cases}$$

Formula de integrare:

$$\int_{\mathcal{D}} f dV \approx \tilde{V}\langle \tilde{f} \rangle \pm \tilde{\sigma},$$

$\langle \tilde{f} \rangle$ și $\tilde{\sigma}$ sunt calculate relativ la domeniul extins $\tilde{\mathcal{D}}$.

● Exemplu

$$I = \iint_{x^2+y^2 \leq 1} dx dy = 4 \int_0^1 dx \int_0^{\sqrt{1-x^2}} dy = \pi,$$

Domeniu de integrare \mathcal{D} — sectorul de cerc din primul cadran.

Domeniu extins pătratul $\tilde{\mathcal{D}} = [0, 1] \times [0, 1]$

Rescriem integrala:

$$I = 4 \int_0^1 dx \int_0^1 dy \theta[1 - (x^2 + y^2)],$$

Discriminarea punctelor interioare sectorului \mathcal{D} de cele exterioare — funcția treaptă

$$\theta(x) = \begin{cases} 0 & \text{dacă } x < 0 \\ 1 & \text{dacă } x \geq 0. \end{cases}$$



Integrale multidimensionale

Formula de integrare Monte Carlo:

$$I \approx \frac{4}{n} \sum_{i=1}^n \theta[1 - (x_i^2 + y_i^2)] = 4 \frac{n_i}{n}$$

n — numărul punctelor (x_i, y_i) distribuite aleator și uniform în pătratul $\tilde{\mathcal{D}}$.

n_i — numărul punctelor interioare sectorului de cerc \mathcal{D} .

Aria sectorului de cerc — proporțională cu **probabilitatea** generării punctelor interioare.
Valoarea medie și valoarea pătratică medie a integrandului sunt egale, $\langle \theta \rangle = \langle \theta^2 \rangle = n_i/n$.

Tehnica poate fi generalizată pentru calculul volumelor multidimensionale:

$$V = \lim_{n \rightarrow \infty} \frac{n_i}{n} \tilde{V}.$$

Contabilizarea punctelor interioare domeniului inițial în raport cu numărul total de puncte.

n	I	σ	n	I	σ
10	2.80000	0.57966	1000	3.12800	0.05223
20	3.40000	0.31937	2000	3.10600	0.03726
50	3.20000	0.22627	5000	3.14400	0.02320
100	3.20000	0.16000	10000	3.15520	0.01633
200	3.06000	0.11992	20000	3.13760	0.01163
500	3.22400	0.07074	50000	3.14128	0.00734



Integrarea numerică - generalizare

- Fie o funcție $f(x)$, $f : [a,b] \rightarrow R$ și $F(x)$ o primitivă a sa. Se consideră că funcțiile $f(x)$ și $F(x)$ sunt continue. Integrala funcției $f(x)$ pe intervalul $[a, b]$, se calculează cu ajutorul primitivei $F(x)$ conform formulei *Newton-Leibnitz*:

$$\int_a^b f(x)dx = F(x) \Big|_a^b = F(b) - F(a), \quad (8.1)$$

În unele cazuri este foarte dificil sau chiar imposibil de determinat forma primitivei $F(x)$ pentru a putea calcula integrala funcției $f(x)$ conform formulei (8.1). În astfel de cazuri, se folosesc diferite metode numerice, care în principiu aproximează funcția dată $f(x)$ cu o funcție polinomială $g(x)$, astfel încât integrala se calculează cu aproximare cu ajutorul primitivei $G(x)$ a funcției $g(x)$:

$$\int_a^b f(x)dx \cong \int_a^b g(x)dx = G(x) \Big|_a^b \quad (8.2)$$

În cadrul metodelor numerice de integrare se utilizează în general următorul algoritm:

1. se divizează intervalului $[a, b]$ în n subintervale cu ajutorul a $n+1$ puncte de diviziune x_i , $i=0, 1, 2, 3, \dots, n$;
2. se scrie funcția $f(x)$ ca suma dintre o *funcție de aproximare* $g(x)$ și o *funcție rest* $r(x)$:

$$f(x) = g(x) + r(x) \quad (8.3)$$

3. se integrează funcția $f(x)$ scrisă astfel obținându-se:

$$\int_a^b f(x)dx = \int_a^b g(x)dx + \int_a^b r(x)dx \quad (8.4)$$



Integrarea numerică - aproximare cu funcție polinomială

Dacă $g(x)$ este o *funcție polinomială* de forma:

$$g(x) = \sum_{k=1}^n a_k q_k(x) \quad (8.4')$$

unde $q_k(x)$ reprezintă un set de funcții polinomiale independente, atunci calculul integralei lui $g(x)$ devine:

$$\int_a^b g(x) dx = \int_a^b \sum_{k=1}^n a_k q_k(x) dx = \sum_{k=1}^n \int_a^b a_k q_k(x) dx = \sum_{k=1}^n a_k \int_a^b q_k(x) dx \quad (8.5)$$

sau:

$$\int_a^b g(x) dx = \sum_{k=1}^n a_k I_k, \quad I_k = \int_a^b q_k(x) dx \quad (8.5')$$

4. se aproximează integrala $\int_a^b f(x) dx$ cu integrala $\int_a^b g(x) dx$ prin minimizarea integralei funcției $r(x)$: $\delta = \int_a^b r(x) dx$ (8.6)

In continuare sunt prezentate metode de integrare numerică utilizând diferite tipuri de polinoame de interpolare și puncte de diviziune (numite și puncte de bază):

- *formule de integrare cu interval închis* (capetele intervalului $[a, b]$ sunt printre punctele de bază);
- *formule de integrare cu interval deschis* (capetele intervalului $[a, b]$ nu sunt printre punctele de bază).

Formulele de integrare numerică se mai numesc și *cuadraturi*.



Cuadratura Newton-Cotes

Formula de integrare *Newton-Cotes* utilizează pentru aproximarea funcției $f(x)$ polinoamele de interpolare *Lagrange* $L(x)$. Cele $n+1$ puncte de bază x_i sunt echidistante (situate între ele la distanța h) și includ și capetele intervalului $[a, b]$.

Polinoamele de interpolare *Lagrange* $L(x)$ au expresia:

$$L(x_0 + qh) = \sum_{k=0}^n \frac{(-1)^{n-k} q^{[n+1]}}{k!(n-k)!(q-k)!} y_k; \quad q = \frac{x - x_0}{h} \quad (8.7)$$

în care s-a notat: $q^{[n+1]} = q(q-1)(q-2)\dots(q-n)$

Integrala (8.2) devine:

$$\int_a^b f(x) dx \cong \int_0^n \left(\sum_{k=0}^n \frac{(-1)^{n-k} q^{[n+1]}}{k!(n-k)!(q-k)!} y_k \right) hdq \quad (8.8)$$

în care s-a ținut seama de schimbarea de variabilă:

$$\begin{aligned} q &= \frac{x - x_0}{h} & \Rightarrow dq &= \frac{dx}{h}; \quad dx = hdq \\ x = x_0 &= a & \Rightarrow q &= 0; \\ x = x_n &= b & \Rightarrow q &= n; \end{aligned} \quad (8.9)$$

Ținând seama că cele $n+1$ puncte de bază x_i sunt echidistante (situate la distanța $h = (b - a)/n$), relația (8.9) devine:



Formula de integrare Newton-Cotes. Cazuri particulare ale cuadraturii

$$\int_a^b f(x)dx \cong (b-a) \sum_{k=0}^n \frac{1}{n} \frac{(-1)^{n-k}}{k!(n-k)!} y_k \int_0^{q^{[n+1]}} \frac{q^{[n+1]}}{(q-k)} dq \quad (8.10)$$

S-a obținut formula de integrare *Newton-Cotes*:

$$I_{n+1} = (b-a) \sum_{k=0}^n H_k \cdot y_k \quad (8.11)$$

unde cu H_k s-au notat *coeficienții Newton-Cotes*:

$$H_k = \frac{(-1)^{n-k}}{n \cdot k! (n-k)!} \int_0^{q^{[n+1]}} \frac{q^{[n+1]}}{(q-k)} dq \quad (8.12)$$

În funcție de numărul n de puncte de bază (puncte de diviziune) ale quadraturii *Newton – Cotes* s-au prezentat următoarele cazuri particulare:

- pentru $n=2$ puncte de diviziune (capetele intervalului: $x_0=a$ și $x_1=b$), relația (8.11) se scrie:

$$I_2 = (b-a)(H_0 y_0 + H_1 y_1) \quad (8.14)$$

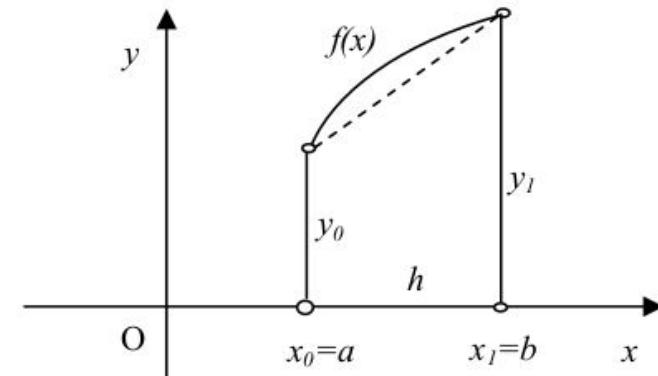
unde coeficienții Cotes H_0 și H_1 se determină conform relației (8.12) :

$$H_0 = -\frac{1}{1 \cdot 0! \cdot 1!} \int_0^1 \frac{q(q-1)}{q} dq = q - \frac{q^2}{2} \Big|_0^1 = \frac{1}{2} \quad (8.15)$$

$$H_1 = \frac{1}{1 \cdot 1! \cdot 0!} \int_0^1 \frac{q(q-1)}{q-1} dq = \frac{q^2}{2} \Big|_0^1 = \frac{1}{2}$$

Înlocuind în relația (8.14) se obține **formula trapezelor** (fig. 8.1):

$$I_2 = \frac{h}{2}(y_0 + y_1) \quad (8.16)$$



Cazuri particulare ale cuadraturii Newton-Cotes.

- pentru $n=3$ puncte de diviziune $x_0=a$, $x_1=a+h$ și $x_2=b$, $h=(b-a)/2$ relația (8.11) se scrie:

$$I_3 = (b-a)(H_0 y_0 + H_1 y_1 + H_2 y_2) \quad (8.17)$$

unde coeficienții Cotes H_0 , H_1 și H_2 se determină conform relației (8.12):

$$\begin{aligned} H_0 &= \frac{1}{2 \cdot 0! \cdot 2!} \int_0^2 (q-1)(q-2) dq = \frac{1}{4} \left(2q - 3\frac{q^2}{2} + \frac{q^3}{3} \right) \Big|_0^2 = \frac{1}{6} \\ H_1 &= -\frac{1}{2 \cdot 1! \cdot 1!} \int_0^2 q(q-2) dq = -\frac{1}{2} \left(-q^2 + \frac{q^3}{3} \right) \Big|_0^2 = \frac{2}{3} \\ H_2 &= \frac{1}{2 \cdot 2! \cdot 0!} \int_0^2 q(q-1) dq = \frac{1}{4} \left(-\frac{q^2}{2} + \frac{q^3}{3} \right) \Big|_0^2 = \frac{1}{6} \end{aligned} \quad (8.18)$$

Înlocuind în relația (8.17) se obține *formula 1/3 Simpson*:

$$I_3 = \frac{h}{3} (y_0 + 4y_1 + y_2) \quad (8.19)$$



Cuadratura Gauss-Legendre

Spre deosebire de formulele de integrare *Newton-Cotes* cu interval închis, în care puncte de bază sunt echidistante și conțin capetele intervalului, în cazul formulele de integrare cu interval deschis, *punctele de bază nu sunt echidistante și nu conțin capetele intervalului* fiind rădăcinile unor polinoame ortogonale cum ar fi: *Legendre, Cebîșev, Hermite, Bessel*, etc.

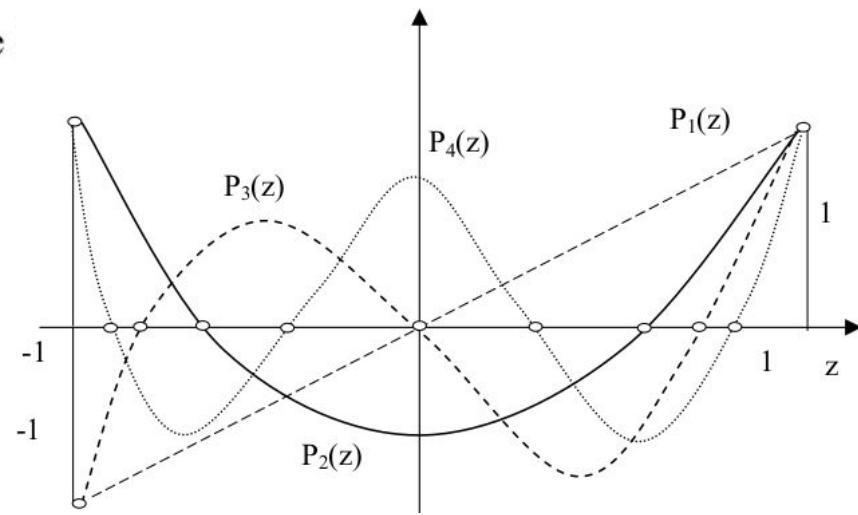
Cuadratura *Gauss-Legendre* are ca puncte de bază rădăcinile z_i ale polinoamelor ortogonale *Legendre* care sunt puncte de interpolare al funcției $f(x)$ pentru polinoamele de interpolare *Lagrange*.

Polinoamele Legendre sunt definite pe intervalul $[-1, 1]$ prin următoarea formulă de recurență:

$$P_n(z) = \frac{2n-1}{n} z P_{n-1}(z) - \frac{n-1}{n} P_{n-2}(z); \quad (8.34)$$
$$P_0(z) = 1; \quad P_1(z) = z$$

Pentru $n=2, 3, 4$ și 5 se obțin polinoamele Legendre

- $P_2(z) = \frac{1}{2}(3z^2 - 1)$
- $P_3(z) = \frac{1}{2}(5z^3 - 3z)$
- $P_4(z) = \frac{1}{8}(35z^4 - 30z^2 + 3)$
- $P_5(z) = \frac{1}{8}(63z^5 - 70z^3 + 15z)$



Polinoame Legendre

Proprietățile polinoamelor *Legendre* sunt:

1. iau valorile 1, ± 1 la capetele intervalului:

$$P_n(1) = 1, \quad P_n(-1) = (-1)^n \quad (8.36)$$

2. sunt ortogonale între ele oricare ar fi m și n :

$$\int_{-1}^1 P_n(z) P_m(z) dz = \begin{cases} 0, & n \neq m \\ \frac{2}{2n+1}, & n = m \end{cases} \quad (8.37)$$

3. sunt ortogonale cu orice polinom $Q(z)$ având gradul mai mic decât acestea:

$$\int_{-1}^1 P_n(z) Q_k(z) dz = 0, \quad k < n \quad (8.38)$$

4. au toate rădăcinile reale și distințe situate în intervalul $[-1, 1]$.

$$P_2(z) = 0 \Rightarrow \begin{cases} z_1 = -\frac{1}{\sqrt{3}} = -0,57735027 \\ z_2 = \frac{1}{\sqrt{3}} = 0,57735027 \end{cases}$$

$$P_3(z) = 0 \Rightarrow \begin{cases} z_1 = -\sqrt{\frac{3}{5}} = -0,77459667 \\ z_2 = 0 \\ z_3 = \sqrt{\frac{3}{5}} = 0,77459667 \end{cases}$$

$$P_4(z) = 0 \Rightarrow \begin{cases} z_1 = -\sqrt{\frac{15+\sqrt{120}}{35}} = -0,86113631 \\ z_2 = -\sqrt{\frac{15-\sqrt{120}}{35}} = -0,33998104 \\ z_3 = \sqrt{\frac{15-\sqrt{120}}{35}} = 0,33998104 \\ z_4 = \sqrt{\frac{15+\sqrt{120}}{35}} = 0,8611361 \end{cases}$$



Calculul integralei definite - cuadratura Gauss-Legendre

Pentru calculul integralei definite $\int_a^b f(x)dx$ prin cuadratura Gauss

Legendre se face schimbarea de variabilă :

$$x = \frac{a+b}{2} + \frac{b-a}{2}z; \quad dx = \frac{b-a}{2}dz. \quad (8.43)$$

$$x = a \Rightarrow z = -1; \quad x = b \Rightarrow z = 1;$$

Integrala devine:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 g(z)dz \quad (8.44)$$

Integrala $\int_{-1}^1 g(z)dz$ se calculează aproximând funcția $g(z)$ cu ajutorul polinoamelor de interpolare Lagrange $L(z)$ având ca puncte de interpolare rădăcinile z_i ale polinoamelor Legendre:

$$g(z) = \sum_{k=1}^n g(z_k) L_k(z) = \sum_{k=1}^n \prod_{i=1, i \neq k}^n \frac{z - z_i}{z_k - z_i} g(z_k) \quad (8.45)$$

Tinând seama de relațiile (8.44) și (8.45) se obține *formula de calcul a integralei prin cuadratura Gauss Legendre*:

$$\int_a^b f(x)dx = \frac{b-a}{2} \sum_{k=1}^n A_k f(x_k) \quad (8.46)$$

z_k – punctele de bază ale cuadraturii Gauss-Legendre.

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}z_k \quad A_k = \int_{-1}^1 [L_k(z)]dz \quad \text{ponderile cuadraturii Gauss-Legendre.}$$



Cuadratura Gauss-Legendre

- Calculul integralei unei funcții $f(x)$ prin *cuadratura Gauss-Legendre* necesită determinarea a $2n$ parametri:
 - n rădăcini ale polinoamelor *Legendre* z_k ;
 - n ponderi ale cuadraturii *Gauss-Legendre* A_k .

Ponderile cuadraturii A_k se calculează conform relației (8.46') folosind polinoamele de interpolare Lagrange având ca puncte de interpolare punctele de bază z_i .

Pentru determinarea polinoamelor Legendre se aproximează funcția de interpolare Lagrange $g(z)$ cu un polinom de gradul $2n-1$ de forma:

$$g(z) = P_n(z)Q(z) + R(z) \quad (8.47)$$

în care: $P_n(z)$ este polinomul Legendre de gradul n

$Q(z)$ este un polinom oarecare având gradul maxim $n-1$

$R(z)$ un polinom de gradul $2n-1$ având proprietatea: $R(z_k) = g(z_k)$

Integrând pe intervalul $[-1, 1]$ relația (8.47), se obține:

$$\int_{-1}^1 g(z) dz = \int_{-1}^1 P_n(z)Q(z) dz + \int_{-1}^1 R(z) dz \quad (8.48)$$

Dacă se își pună condiția ca prima integrală să fie egală cu

ultima: $\int_{-1}^1 g(z) dz = \int_{-1}^1 R(z) dz \quad (8.49)$

unde: $\int_{-1}^1 g(z) dz = \sum_{k=1}^n g(z_k); \quad \int_{-1}^1 R(z) dz = \sum_{k=1}^n R(z_k) A_k \quad (8.49')$



Cuadratura Gauss-Legendre

- rezultă că polinoamele $P_n(z)$ și $Q(z)$ sunt ortogonale:

$$\int_{-1}^1 P_n(z) Q(z) dz = 0 \quad (8.50)$$

Luând pentru $Q(z)$ cele mai simple polinoame de forma: $Q(z)=z^k$, $k \leq n-1$ și înlocuind în relația (8.50) se obține:

$$\int_{-1}^1 P_n(z) z^k dz = 0, \quad k = 0, 1, 2, 3, \dots, n-1 \quad (8.51)$$

Relația (8.51) permite determinarea polinoamelor Legendre care conform proprietății (8.38) sunt ortogonale cu orice polinom $Q(z)$.

Aceste polinoame se scriu sub forma generală:

$$P_n(z) = a_0 + a_1 z + a_2 z^2 + \dots + a_n z^n \quad (8.52)$$



Cuadratura Gauss-Legendre

Scriind cele n ecuații corespunzătoare integralelor (8.51) și ținând seama de proprietatea (8.36) a polinoamelor Legendre $P_n(1)=1$ se obține următorul sistem de $n+1$ ecuații cu $n+1$ necunoscute:

$$\begin{cases} a_0 + \frac{a_2}{3} + \frac{a_4}{5} + \dots + \frac{a_n}{n+1} = 0 & (n = par) \\ \frac{a_1}{3} + \frac{a_3}{5} + \frac{a_5}{7} + \dots + \frac{a_{n-1}}{n+1} = 0 & (n-1 = impar) \\ \frac{a_0}{3} + \frac{a_2}{5} + \frac{a_4}{7} + \dots + \frac{a_n}{n+3} = 0 \\ \frac{a_1}{5} + \frac{a_3}{7} + \frac{a_5}{9} + \dots + \frac{a_{n-1}}{n+3} = 0 \\ \frac{a_0}{5} + \frac{a_2}{7} + \frac{a_4}{9} + \dots + \frac{a_n}{n+5} = 0 \\ \frac{a_1}{7} + \frac{a_3}{9} + \frac{a_5}{11} + \dots + \frac{a_{n-1}}{n+5} = 0 \\ \dots \\ a_0 + a_1 + a_2 + \dots + a_n = 1 \end{cases}$$

➤ pentru $n=2$:

$$\begin{cases} a_0 + \frac{a_2}{3} = 0 \\ \frac{a_1}{3} = 0 \\ a_0 + a_1 + a_2 = 1 \end{cases} \Rightarrow \begin{cases} a_0 = -\frac{1}{2} \\ a_1 = 0 \\ a_2 = \frac{3}{2} \end{cases} \Rightarrow P_2(z) = -\frac{1}{2} + \frac{3}{2}z^2$$

➤ pentru $n=3$:

$$\begin{cases} a_0 + \frac{a_2}{3} = 0 \\ \frac{a_1}{3} + \frac{a_3}{5} = 0 \\ \frac{a_0}{3} + \frac{a_2}{5} = 0 \\ a_0 + a_1 + a_2 + a_3 = 1 \end{cases} \Rightarrow \begin{cases} a_0 = 0 \\ a_1 = -\frac{3}{2} \\ a_2 = 0 \\ a_3 = \frac{5}{2} \end{cases} \Rightarrow P_3(z) = -\frac{3}{2}z + \frac{5}{2}z^3$$



Cuadratura Gauss-Legendre

- Ponderile cuadraturii Gauss Legendre se determină conform relației (8.46') astfel:

$$A_k = \int_{-1}^1 \prod_{i=1, i \neq k}^n \frac{z - z_i}{z_k - z_i} dz, \quad k = 1, 2, 3, \dots, n \quad (8.58)$$

Astfel ținând seama de relațiile (8.39)... (8.42) pentru se obțin punctele de bază z_i respectiv ponderile A_i din tabelul 8.3:

Tabelul 8.3

n	z_i	A_i
2	$x_1 = -0,57735027$ $x_2 = 0,57735027$	$A_1 = 1$ $A_2 = 1$
3	$x_1 = -0,77459667$ $x_2 = 0$ $x_3 = 0,77459667$	$A_1 = 0,555555$ $A_2 = 0,888888$ $A_3 = 0,555555$
4	$x_1 = -0,86113631$ $x_2 = -0,33998104$ $x_3 = 0,33998104$ $x_4 = 0,86113631$	$A_1 = 0,347854$ $A_2 = 0,652145$ $A_3 = 0,652145$ $A_4 = 0,347854$



Exemplu - cuadratura Gauss-Legendre

Să se calculeze integrala $I = \int_{-1}^5 \frac{x^2}{1+x} dx$ folosind formulele cuadraturii

Gauss-Legendre corespunzătoare pentru $n=2, 3, \dots, 7$ puncte de bază.

Rezolvare

Introducând valorile punctelor de bază z_i și ponderilor A_i corespunzătoare din tabelul 8.3 în formula cuadraturii Gauss Legendre (8.46) s-au obținut valorile din tabelul 8.4.

Tabelul 8.4

Nr. puncte de bază	z_i	A_i	x_i	$f(x_i)$	Valoarea integralei
$n=2$	-0.57735027	1.00000000	1.84529946	1.19675632	9.09090909
	0.57735027	1.00000000	4.15470054	3.34869823	
$n=3$	-0.77459667	0.55555556	1.45080666	0.85883558	9.09803922
	0.00000000	0.88888889	3.00000000	2.25000000	
	0.77459667	0.55555556	4.54919334	3.72939971	
$n=4$	-0.86113631	0.34785484	1.27772738	0.71676148	9.09857035
	-0.33998104	0.65214516	2.32003792	1.62123930	
	0.33998104	0.65214516	3.67996208	2.89363903	
	0.86113631	0.34785484	4.72227262	3.89702836	



Utilizarea GSL în calculul numeric al integralelor

```
// Each algorithm computes an approximation to a definite integral of
// the form,
//
// I = \int_a^b f(x) w(x) dx
//
// where w(x) is a weight function (for general integrands w(x)=1). The
// user provides absolute and relative error bounds (epsabs, epsrel)
// which specify the following accuracy requirement,
//
// |RESULT - I| <= max(epsabs, epsrel |I|)
//
// where RESULT is the numerical approximation obtained by the
// algorithm. The algorithms attempt to estimate the absolute error
// ABSERR = |RESULT - I| in such a way that the following inequality
// holds,
//
// |RESULT - I| <= ABSERR <= max(epsabs, epsrel |I|)
//
// The routines will fail to converge if the error bounds are too
// stringent, but always return the best approximation obtained up to
// that stage.
//
//
// QAGS adaptive integration with singularities
//
// Function: int gsl_integration_qags (const gsl_function * f,
// double a, double b,
// double epsabs, double epsrel,
// size_t limit,
// gsl_integration_workspace * workspace,
// double *result,
// double *abserr)
//
// This function applies the Gauss-Kronrod 21-point integration rule
// adaptively until an estimate of the integral of f over (a,b) is
// achieved within the desired absolute and relative error limits,
// epsabs and epsrel. The results are extrapolated using the
// epsilon-algorithm, which accelerates the convergence of the integral
// in the presence of discontinuities and integrable singularities. The
// function returns the final approximation from the extrapolation,
// result, and an estimate of the absolute error, abserr. The
// subintervals and their results are stored in the memory provided by
// workspace. The maximum number of subintervals is given by limit,
// which may not exceed the allocated size of the workspace.
//
//
// The integrator QAGS will handle a large class of definite integrals.
// For example, consider the following integral, which has a
// algebraic-logarithmic singularity at the origin,
//
// \int_0^1 x^{-1/2} log(x) dx = -4
//
// The program below computes this integral to a relative accuracy bound
// of 1e-8.
```

```
// include files
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cmath>
using namespace std;

#include <gsl/gsl_integration.h>

// function prototypes
double my_integrand (double x, void *params);

//*****
int
main (void)
{
    gsl_integration_workspace *work_ptr
        = gsl_integration_workspace_alloc (1000);

    double lower_limit = 0;      /* lower limit a */
    double upper_limit = 1;      /* upper limit b */
    double abs_error = 1.0e-8;   /* to avoid round-off problems */
    double rel_error = 1.0e-8;   /* the result will usually be much better */
    double result;               /* the result from the integration */
    double error;                /* the estimated error from the integration */

    double alpha = 1.0;          /* parameter in integrand */
    double expected = -4.0;      /* exact answer */

    gsl_function My_function;
    void *params_ptr = &alpha;

    My_function.function = &my_integrand;
    My_function.params = params_ptr;

    gsl_integration_qags (&My_function, lower_limit, upper_limit,
                         abs_error, rel_error, 1000, work_ptr, &result,
                         &error);

    cout.setf (ios::fixed, ios::floatfield);      // output in fixed format
    cout.precision (18);                          // 18 digits in doubles

    int width = 20; // setw width for output
    cout << "result      = " << setw(width) << result << endl;
    cout << "exact result = " << setw(width) << expected << endl;
    cout << "estimated error = " << setw(width) << error << endl;
    cout << "actual error   = " << setw(width) << result - expected << endl;
    cout << "intervals = " << work_ptr->size << endl;

    return 0;
}

//*****
double
my_integrand (double x, void *params)
{
    // Mathematica form: Log[alpha*x]/Sqrt[x]

    // The next line recovers alpha from the passed params pointer
    double alpha = *(double *) params;

    return (log (alpha * x) / sqrt (x));
}
```



Note bibliografice

- Anton Hadar - Metode numerice in inginerie, Politehnica Press, Bucuresti, 2004
- Ioan RUSU - Metode numerice în electronică, 2006
- T.A. Beu, Calcul numeric in C, Editia a 2-a, Ed. Alabastrea, Cluj-Napoca, 2000
- W.H. Press, S.A. Teukolsky, W.T. Vetterling și B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Second Edition (Cambridge University Press, Cambridge, 1992).



Rezolvarea numerică a ecuațiilor diferențiale

Lect. dr. Ioan Dumitru



- În acest capitol ne restrângem atenia la ecuații diferențiale ordinare. Ne concentrăm pe probleme cu valoari inițiale și pe condiții pe frontieră și prezentăm câteva cele mai frecvent utilizate metode de rezolvare a acestor probleme numerice.



Ecuatii diferențiale ordinare

- Ordinul ecuației diferențiale ordinare reprezinta ordinul derivatiei din membrul stang a ecuației

$$\frac{dy}{dt} = f(t, y)$$

Aceasta este o ecuație diferențială de ordin unu, unde f este o funcție oarecare.

- O ecuație de ordin doi se scrie ca:

$$\frac{d^2y}{dt^2} = f(t, \frac{dy}{dt}, y)$$

- Una dintre cele mai cunoscute ecuații diferențiale de ordin doi este ecuația lui Newton:

$$m \frac{d^2x}{dt^2} = -kx$$

în care k este constanta electrică a unui resot de care este agatat un corp de masa m .
Ecuația diferențială depinde de o singură variabilă.



Ecuatii diferențiale ordinare

- O ecuație de tipul ecuației cu derivate partiale Schrödinger dependente de timp

$$i\hbar \frac{\partial \psi(\mathbf{x}, t)}{\partial t} = \frac{\hbar^2}{2m} \left(\frac{\partial^2 \psi(\mathbf{r}, t)}{\partial x^2} + \frac{\partial^2 \psi(\mathbf{r}, t)}{\partial y^2} + \frac{\partial^2 \psi(\mathbf{r}, t)}{\partial z^2} \right) + V(\mathbf{x})\psi(\mathbf{x}, t)$$

- depinde de mai multe variabile. În anumite condiții funcția de undă poate fi factorizată în funcții de variabile separate și ecuația se poate scrie printr-un set de ecuații diferențiale.



Metoda diferențelor finite

- Aceste metode se încadrează în clasa generală a metodelor intr-un singur pas. Algoritmul este destul de simplu. Presupune avem o valoare inițială pentru y funcția (t) dată de

$$y_0 = y(t = t_0)$$

- Suntem interesati in rezolvarea unei ecuații diferențiale într-o regiune în spațiu [a, b]. Definim un pas h de divizarea intervalului în intr-un numar N de subintervale, astfel încât să avem

$$h = \frac{b - a}{N}$$

- Cu acest pas si derivata lui y putem construi următoarea valoare a funcției y

$$y_1 = y(t_1 = t_0 + h)$$

- si asa mai departe. În cazul în care funcția este destul de liniă în domeniul [a, b], se poate folosi un pas fix pentru divizare cu pas fix.
- Incercăm să generalizăm procedura de mai sus prin scrierea y_{i+1} în functie de y_i din pasul anterior.

$$y_{i+1} = y(t = t_i + h) = y(t_i) + h\Delta(t_i, y_i(t_i)) + O(h^{p+1})$$

- unde $O(h^{p+1})$ reprezinta eroarea de trunchere



Metoda diferențelor finite - metoda Euler

- Pentru a determina D dezvoltam y în serii Taylor

$$y_{i+1} = y(t = t_i + h) = y(t_i) + h(y'(t_i) + \dots + y^{(p)}(t_i) \frac{h^{p-1}}{p!}) + O(h^{p+1})$$

- de unde

$$\Delta(t_i, y_i(t_i)) = (y'(t_i) + \dots + y^{(p)}(t_i) \frac{h^{p-1}}{p!})$$

- Cum

$$y'(t_i) = f(t_i, y_i)$$

- și dacă truncăm D oprindu-ne la derivata de ordin unu avem:

$$y_{i+1} = y(t_i) + h f(t_i, y_i) + O(h^2)$$

- care împreună cu

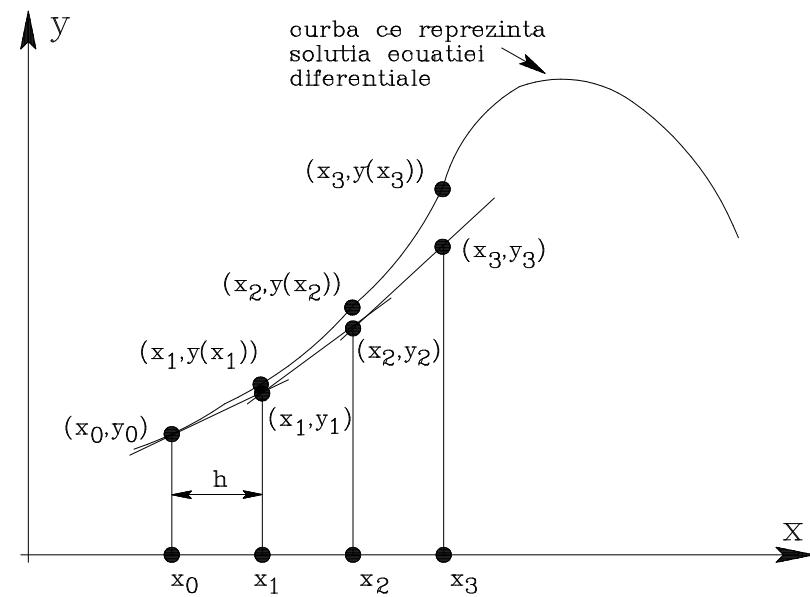
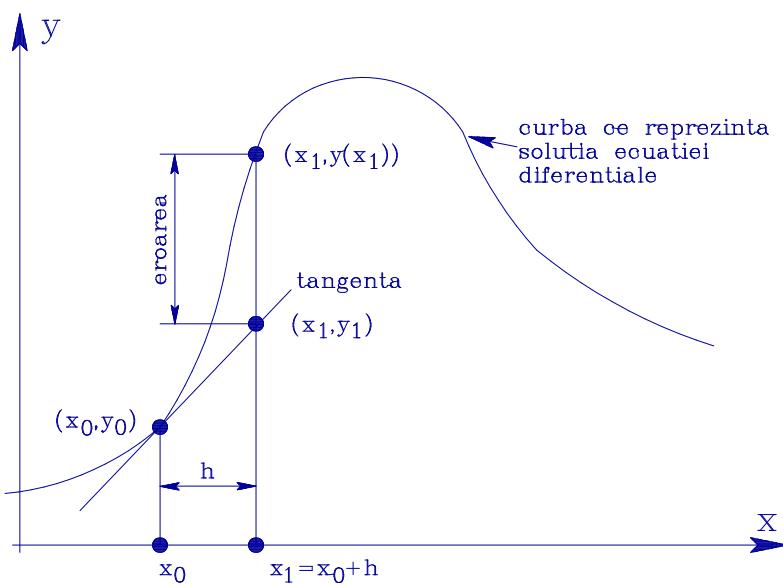
$$t_{i+1} = t_i + h$$

- formează algoritmul pentru metoda Euler.



Metoda Euler - interpretare geometrica

- Metoda Euler - cunoscuta si sub denumirea de metoda tangentelor
- Se poate gasi un punct $(x_1, y_1) = (x_0+h, y_1)$ pe tangenta la curba ce reprezinta solutia ecuatiei diferențiale in punctul (x_0, y_0) , asa cum se prezinta in figura alaturata.



$$\frac{y_1 - y_0}{(x_0 + h) - x_0} = y'_0 \Leftrightarrow y_1 = y_0 + h y'_0$$

$$y'_0 = f(x_0, y_0)$$



Exemplu de utilizare a metodei Euler

- Considerand ecuatia diferențiala:

$$y' = 2xy$$

si conditia initiala $y(1)=1$, sa se obtina o aproximatie pentru a gasi valoarea lui y in punctul de abscisa $x = 1.5$, utilizand un pas h cu valorile 0.1, 0.05 si 0.01, apoi sa se calculeze si erorile relative raportate la valoarea exacta, cu patru zecimale.

- Solutia analitica a ecuatiei diferențiale considerate este:

$$y(x) = e^{x^2 - 1}$$

Tab. 1 Valorile obtinute pentru $h = 0.10$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2000	1.2337	0.0337	2.73
1.20	1.4640	1.5527	0.0887	5.71
1.30	1.8154	1.9937	0.1784	8.95
1.40	2.2874	2.6117	0.3244	12.42
1.50	2.9278	3.4904	0.5625	16.12 ←



Exemplu de utilizare a metodei Euler

Tab. 2 Valorile obtinute pentru $h = 0.05$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2155	1.2337	0.0182	1.47
1.20	1.5044	1.5527	0.0483	3.11
1.30	1.8955	1.9937	0.0982	4.93
1.40	2.4311	2.6117	0.1806	7.98
1.50	3.1733	3.4904	0.3171	9.08 ←

Tab. 3 Valorile obtinute pentru $h = 0.01$

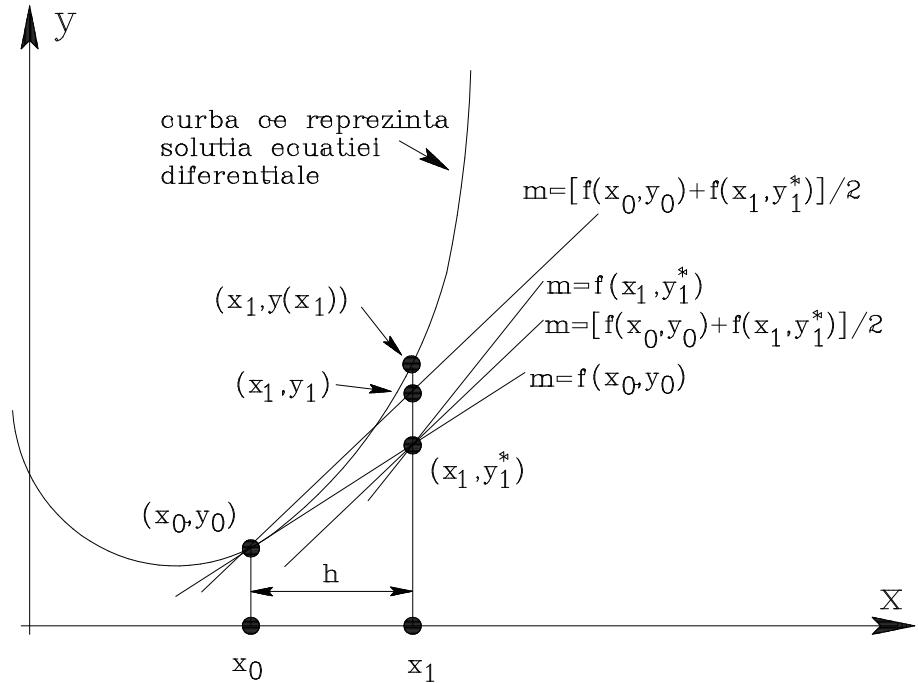
x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2298	1.2337	0.0039	0.31
1.20	1.5423	1.5527	0.0104	0.67
1.30	1.9723	1.9937	0.0214	1.07
1.40	2.5719	2.6117	0.0398	1.52
1.50	3.4197	3.4904	0.0707	2.03 ←



Metoda Euler - Heun

- metoda este cunoscuta si sub denumirea de metoda Euler imbunatatita, interpretarea geometrica fiind prezentata in figura de mai jos pentru primul pas de integrare.
- procesul iterativ pentru solutionarea numerica este descris de relatiile:

$$y_{n+1} = y_n + h \frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)}{2},$$
$$y_{n+1}^* = y_n + h f(x_n, y_n).$$



- se poate constata ca valoarea lui y_1 este mai apropiata de solutie decat valoarea obtinuta pentru y_1^* , ceea ce poate incadra aceasta metoda in clasa metodelor predictor-corector.



- In continuare se prezinta rezultatele obtinute prin aplicarea acestei metode la rezolvarea ecuatiei diferențiale anterioare.

Tab. 4 Valorile obtinute pentru $h = 0.1$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2320	1.2337	0.0017	0.14
1.20	1.5479	1.5527	0.0048	0.31
1.30	1.9832	1.9937	0.0106	0.53
1.40	2.5908	2.6117	0.0209	0.80
1.50	3.4509	3.4904	0.0394	1.13 ←

Tab.5 Valorile obtinute pentru $h=0.05$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2332	1.2337	0.0004	0.04
1.20	1.5514	1.5527	0.0013	0.08
1.30	1.9909	1.9937	0.0029	0.14
1.40	2.6060	2.6117	0.0057	0.22
1.50	3.4795	3.4904	0.0108	0.31 ←

Tab.6 Valorile obtinute pentru $h = 0.01$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2337	1.2337	0.0000	0.00
1.20	1.5527	1.5527	0.0000	0.00
1.30	1.9936	1.9937	0.0001	0.005
1.40	2.6115	2.6117	0.0002	0.010
1.50	3.4899	3.4904	0.0005	0.025 ←



Metoda diferențelor finite de ordin superior

- De notat că la fiecare pas facem o eroare de ordinul lui $O(h^2)$. Eroarea totală este suma erorilor făcute în toți cei N pași. Cum $N = (b-a)/h$, rezulta o eroare globală $NO(h^2) \approx O(h)$. Pentru a face mai i precisă metoda Euler se poate descrește h (crescând pe N).
- Pentru o micsorare a erorii de calcul se pot considera termeni de ordin superior în dezvoltarea Taylor

$$y_{i+1} = y(t = t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(t_i) + O(h^3)$$

- dar

$$y'' = f' = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f$$

- și obținem

$$y_{i+1} = y(t = t_i + h) = y(t_i) + hf(t_i) + \frac{h^2}{2} \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f \right) + O(h^3)$$

- având eroarea de aproximare locală $O(h^3)$ și eroarea globală $O(h^2)$.



Metoda diferențelor finite de ordin superior

$$y_{i+1} = y(t = t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(t_i) + \frac{h^3}{6} y'''(t_i) + O(h^4)$$

$$y''' = \frac{dy''}{dt} = \frac{\partial^2 f}{\partial t^2} + 2f \frac{\partial^2 f}{\partial t \partial y} + \frac{\partial f}{\partial y} \frac{\partial f}{\partial t} + f \left(\frac{\partial f}{\partial y} \right)^2 + f^2 \frac{\partial^2 f}{\partial y^2}$$

- Metode poate generaliza utilizand o dezvoltare în serie care se oprește la derivata de ordin p:

$$y_{i+1} = y(t = t_i + h) = y(t_i) + h(f(t_i, y_i) + \dots + f^{(p-1)}(t_i, y_i) \frac{h^{p-1}}{p!}) + O(h^{p+1})$$

- Aceste metode bazate pe derivate de ordin superior nu sunt în general utilizate în metode numerice deoarece necesita calculul numeric al derivatelor de ordin superior ale funcției și metoda este consumatoare de timp.
- Pe de altă parte în calculul derivatelor pot interveni erori de aproximare importante dacă valorile funcției de la numărătorul derivatei sunt foarte apropiate.



Metoda diferențelor finite - Metoda Runge - Kutta

- Este bazată pe dezvoltarea Taylor și conduce la algoritmi eficienți de determinare a soluției unei ecuații diferențiale ordinare. Filozofia metodei constă în prevederea unui pas intermediar în calculul valorii lui y_{i+1} .

$$\frac{dy}{dt} = f(t, y),$$

$$y(t) = \int f(t, y) dt,$$

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t, y) dt.$$

- Se utilizează dezvoltarea Taylor a funcției $f(t, y)$ în jurul valorii centrale a intervalului de integrare $[t_i, t_{i+1}]$, $t = t_i + h/2$, h fiind pasul.
- Utilizând formula pentru punctul intermediar

$y(t_i + h/2) = y_{i+1/2}$ și $t_i + h/2 = t_{i+1/2}$ se obține:



Metoda Runge - Kutta de ordin 2, RK2

$$\int_{t_i}^{t_{i+1}} f(t, y) dt \approx h f(t_{i+1/2}, y_{i+1/2}) + O(h^3)$$

- Aceasta înseamnă că avem:

$$y_{i+1} = y_i + h f(t_{i+1/2}, y_{i+1/2}) + O(h^3)$$

- Nu este cunoscută valoarea lui $y_{i+1/2}$ dar se poate aproxima folosind din nou o dezvoltare în serii Taylor

$$y_{(i+1/2)} = y_i + \frac{h}{2} \frac{dy}{dt} = y(t_i) + \frac{h}{2} f(t_i, y_i)$$

- Putem defini următorul algoritm pentru RK2:

$$k_1 = h f(t_i, y_i)$$

$$k_2 = h f(t_{i+1/2}, y_i + k_1/2)$$

$$y_{i+1} \approx y_i + k_2 + O(h^3)$$

- Diferența, față de precedenta metodă într-un singur pas, este că avem nevoie de un pas intermediar în evaluare la $t_{i+h/2} = t_{i+1/2}$. Aceasta implică mai multe calcule dar și o stabilitate mai ridicată a soluției.



Metoda Runge - Kutta de ordin 4, RK4

- Similar se poate defini următorul algoritm pentru RK4:

$$k_1 = h f(t_i, y_i),$$

$$k_2 = h f(t_i + h/2, y_i + k_1/2),$$

$$k_3 = h f(t_i + h/2, y_i + k_2/2)$$

$$k_4 = h f(t_i + h, y_i + k_3)$$

$$y_{i+1} = y_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

- Algoritmul constă în primul pas în calcularea lui k_1 cu t_i , y_i și f ca date de intrare. După aceasta se crește pasul cu $h/2$ și se calculează k_2 , apoi k_3 și în final k_4 . Se poate obține astfel valoarea variabilei y .



Regresia polinomială

- În continuare se prezintă rezultatele numerice obținute în cazul aplicării acestei metode pentru rezolvarea ecuației diferențiale considerate anterior

Tab. 7 Valorile obținute pentru $h = 0.1$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2337	1.2337	0.0000	0.00
1.20	1.5527	1.5527	0.0000	0.00
1.30	1.9937	1.9937	0.0000	0.00
1.40	2.6116	2.6117	0.0001	0.0038
1.50	3.4902	3.4904	0.0002	0.0076

Tab. 8 $h = 0.05$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2337	1.2337	0.0000	0.00
1.20	1.5527	1.5527	0.0000	0.00
1.30	1.9937	1.9937	0.0000	0.00
1.40	2.6117	2.6117	0.0000	0.00
1.50	3.4903	3.4904	0.0001	0.0028

Tab. 9 Valorile obținute pentru $h = 0.01$

x_n	y_n	Exact	Eroare	Eroare %
1.00	1.0000	1.0000	0.0000	0.00
1.10	1.2337	1.2337	0.0000	0.00
1.20	1.5527	1.5527	0.0000	0.00
1.30	1.9937	1.9937	0.0000	0.00
1.40	2.6117	2.6117	0.0000	0.00
1.50	3.4903429	3.4904	0.0000571	0.0016



Metoda Runge-Kutta

Metodele Runge-Kutta presupun căutarea lui y_{m+1} sub forma:

$$(80) \quad y_{m+1} = y_m + \sum_{i=1}^k w_i k_i$$

unde

$$\begin{cases} k_i = h f(\xi_i, \eta_i) \\ \xi_i = x_m + \alpha_i h \\ \eta_i = y_m + \sum_{j=1}^{i-1} \beta_{ij} k_j \end{cases}$$

unde α_i , β_{ij} și w_i sunt constante ce trebuie determinate. Pentru cazul metodei Runge-Kutta de ordin $k = 4$, se găsește:

$$(81) \quad y_{i+1} = y_i + \frac{h(f_1 + 2f_2 + 2f_3 + f_4)}{6}$$

unde

$$(82) \quad \begin{cases} f_1 = f(x_i, y_i) \\ f_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}f_1\right) \\ f_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}f_2\right) \\ f_4 = f(x_i + h, y_i + hf_3) \end{cases}$$



Metoda Runge-Kutta

Cu ajutorul metodelor de tip Runge-Kutta pot fi rezolvate de asemenea și sisteme de ecuații. Astfel pentru o problemă de tipul:

$$(83) \quad \begin{cases} \frac{dx}{dt} = f(t, x, y) \\ \frac{dy}{dt} = g(t, x, y) \end{cases} \quad \text{cu} \quad \begin{cases} x(t_0) = x_0 \\ y(t_0) = y_0 \end{cases}$$

conform metodei Runge-Kutta de ordin 4 formulele de calcul ale soluției sunt:

$$(84) \quad \begin{cases} x_{i+1} = x_i + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4) \\ y_{i+1} = y_i + \frac{h}{6}(g_1 + 2g_2 + 2g_3 + g_4) \end{cases}$$

$$\begin{cases} f_1 = f(t_i, x_i, y_i) \\ f_2 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}f_1, y_i + \frac{h}{2}g_1\right), \\ f_3 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}f_2, y_i + \frac{h}{2}g_2\right), \\ f_4 = f(t_i + h, x_i + hf_3, y_i + hg_3), \end{cases} \quad \begin{cases} g_1 = g(t_i, x_i, y_i) \\ g_2 = g\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}f_1, y_i + \frac{h}{2}g_1\right), \\ g_3 = g\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}f_2, y_i + \frac{h}{2}g_2\right), \\ g_4 = g(t_i + h, x_i + hf_3, y_i + hg_3). \end{cases}$$



Regresia polinomială

1. Utilizați metoda Runge-Kutta de ordin 4 pentru rezolvarea ecuațiilor diferențiale de ordin 1 de mai jos. Efectuați primii doi pași utilizând $h = 0.2$ și primii patru pași utilizând $h = 0.1$

- (a) $y' = x^2 - y$ unde $y(0) = 1$.
- (b) $y' = 3y + 3x$ unde $y(0) = 1$.

Rezolvare:

- (a) Pasul 1. Conform condiției inițiale, vom porni de la $x_0 = 0$, $y_0 = 1$ și $h = 0.2$. Conform formulelor 81 și 82 avem:

$$\begin{cases} f_1 = f(0, 1) = -1 \\ f_2 = f(0.1, 1 + 0.1 \times (-1)) = f(0.1, 0.9) = -0.89 \\ f_3 = f(0.1, 1 + 0.1 \times (-0.89)) = f(0.1, 0.911) = -0.901 \\ f_4 = f(0.2, 1 + 0.2 \times (-0.901)) = f(0.2, 0.8198) = -0.7798 \end{cases}$$

Ca urmare, rezultă că:

$$y_1 = 1 + \frac{0.2(-1 + 2 \times (-0.89) + 2 \times (-0.901) + (-0.7798))}{6} = 0.821273$$

Pasul 2. Pornind acum cu $x_1 = 0.2$ și $y_1 = 0.821273$, printr-un calcul analog primului pas se ajunge la concluzia că $y_2 = 0.689688$. Rezultatele complete pentru acest subpunkt al exercițiului sunt prezentate în tabelul următor:

x_i	$y_i(h = 0.1)$	$y_i(h = 0.2)$
0.0	1.000000	1.000000
0.1	0.905163	
0.2	0.821269	0.821273
0.3	0.749182	
0.4	0.689680	0.689688

b. x_i	$y_i(h = 0.1)$	$y_i(h = 0.2)$
0.0	1.000000	1.000000
0.1	1.366450	
0.2	1.896082	1.895200
0.3	2.645982	
0.4	3.693210	3.689997



Regresia polinomială

7. Fie sistemul de ecuații:

$$\begin{cases} x'(t) = x + 2y \\ y'(t) = 3x + 2y \end{cases} \quad \text{cu} \quad \begin{cases} x(0) = 6 \\ y(0) = 4 \end{cases}$$

Să se utilizeze metoda Runge-Kutta de ordin 4 pentru rezolvarea acestui sistem pe intervalul $[0, 0.2]$ utilizând 10 intervale de pas $h = 0.02$.

Rezolvare:

Vom face detaliat calculele pentru primul punct, $t_1 = 0.02$, urmând ca pentru celelalte puncte calculele să se efectueze analog. Pentru calculul lui x_1 și y_1 va trebui să calculăm valorile lui $f_1, f_2, f_3, f_4, g_1, g_2, g_3$ și g_4 :

$$f_1 = f(0, 6, 4) = 14 \quad g_1 = g(0, 6, 4) = 26$$

$$f_2 = f(0.01, 6.14, 4.26) = 14.66 \quad g_2 = g(0.01, 6.14, 4.26) = 26.94$$

$$f_3 = f(0.01, 6.1466, 4.2694) = 14.6854 \quad g_3 = g(0.01, 6.1466, 4.2694) = 26.9786$$

$$f_4 = f(0.02, 6.2937, 4.5395) = 15.372852 \quad g_4 = g(0.02, 6.2937, 4.5395) = 27.960268$$

de unde:

$$x_1 = 6 + \frac{0.02}{6}(14 + 2 \times 14.66 + 2 \times 14.6854 + 15.372852) = 6.293545$$

$$y_1 = 4 + \frac{0.02}{6}(26 + 2 \times 26.94 + 2 \times 26.9786 + 27.960268) = 4.539324$$

Procedând în aceeași manieră, se pot calcula celealte valori, $\{x_i, y_i\}$. În tabelul următor sunt prezentate rezultările complete pentru acest exercițiu. Având în vedere faptul că în general, rezolvarea unui sistem de ecuații diferențiale cu ajutorul metodei Runge-Kutta este foarte greoi este de preferat utilizarea unui mic program ce poate rezolva problema cerută foarte rapid.

t_i	x_i	y_i
0.00	6.000000	4.000000
0.02	6.293546	4.539325
0.04	6.615622	5.119486
0.06	6.968525	5.743965
0.08	7.354743	6.416533
0.10	7.776973	7.141272
0.12	8.238137	7.922604
0.14	8.741405	8.765317
0.16	9.290210	9.674595
0.18	9.888271	10.656056
0.20	10.539623	11.715781



Regresia polinomială

11. Un sistem rezonant oarecare execută mișcări oscilatorii forțate modelate de ecuația:

$$x''(t) + 25x(t) = 8 \sin(5t)$$

cu:

$$x(0) = 0 \text{ și } x'(0) = 0$$

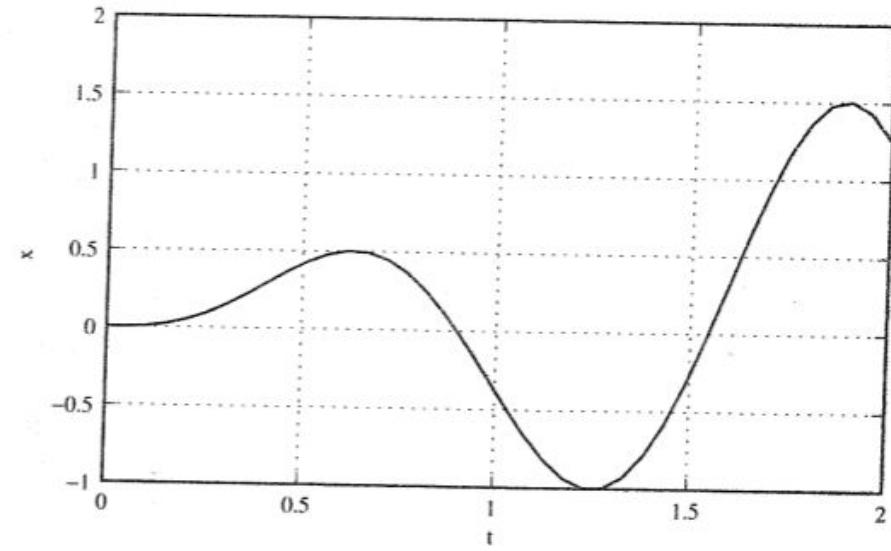
Utilizând metoda Runge-Kutta de ordin 4 să se rezolve ecuația dată pe intervalul $[0, 2]$ folosind 40 de pași și o lărgime a subintervalelor $h = 0.05$.

În urma schimbării de variabilă $y = x'$, rezultă că ecuația oscilatorului forțat de mai sus este echivalentă cu sistemul de ecuații diferențiale:

$$\begin{cases} x' = y \\ y' = -25x + 8 \sin(5t) \end{cases}$$

cu condițiile inițiale:

$$\begin{cases} x(0) = 0 \\ y(0) = 0 \end{cases}$$



- Să presupunem ca am putut construi o soluție aproximativă până la un punct x_n și că am pastrat, de asemenea, toate informațiile aferente punctului x_{n-1} . Știind, aşadar, punctele (x_n, f_n) și (x_{n-1}, f_{n-1}) putem construi fără probleme un polinom de interpolare pentru f care ia forma

$$F(x) = \frac{x - x_n}{x_{n-1} - x_n} f_{n-1} + \frac{x - x_{n-1}}{x_n - x_{n-1}} f_n. \quad (2.1)$$

Cu el putem determina imediat soluția aproximativă a ecuației diferențiale în punctul x_{n+1} , calculând o integrală extremă de simplă, i.e.,

$$\begin{aligned} y_{n+1} &\approx y_n + \int_{x_n}^{x_{n+1}} dx F(x), \\ &= y_n + \int_{x_n}^{x_{n+1}} dx \left(\frac{x - x_n}{x_{n-1} - x_n} f_{n-1} + \frac{x - x_{n-1}}{x_n - x_{n-1}} f_n \right), \\ &= y_n + h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right). \end{aligned} \quad (2.2)$$

Pentru a spori acuratețea rezultatului anterior, putem crește ordinul polinomului de interpolare (în calculul precedent, doar o dreaptă), prin adăugarea punctului (x_{n-2}, f_{n-2}) . Noul polinom de interpolare (acum de ordinul doi) este dat de



$$\begin{aligned} F(x) &= \frac{x - x_n}{x_{n-2} - x_n} \frac{x - x_{n-1}}{x_{n-2} - x_{n-1}} f_{n-2} + \frac{x - x_n}{x_{n-1} - x_n} \frac{x - x_{n-2}}{x_{n-1} - x_{n-2}} f_{n-1} \\ &\quad + \frac{x - x_{n-2}}{x_n - x_{n-2}} \frac{x - x_{n-1}}{x_n - x_{n-1}} f_n, \end{aligned}$$

iar noua soluție aproximativă este dată de

$$\begin{aligned} y_{n+1} &\approx y_n + \int_{x_n}^{x_{n+1}} dx F(x), \\ &= y_n + \frac{1}{h^2} \int_{x_n}^{x_{n+1}} dx \left[\frac{1}{2} (x - x_n) (x - x_{n-1}) - (x - x_n) (x - x_{n-2}) \right. \\ &\quad \left. + \frac{1}{2} (x - x_{n-2}) (x - x_{n-1}) \right], \\ &= y_n + h \left(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2} \right) \end{aligned}$$



Metode multipas - Metode Adams-Bashforth

- Metodele derivate anterior sunt cunoscute în literatura de specialitate drept metode Adams-Bashforth
- Prezentam mai jos formule Adams-Bashforth de la ordinul I, formula care se identifica cu metoda Euler, pana la ordinul 8

$$y_{n+1} = y_n + h f_n \quad (2.24)$$

$$y_{n+1} = y_n + h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) \quad (2.25)$$

$$y_{n+1} = y_n + h \left(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2} \right) \quad (2.26)$$

$$y_{n+1} = y_n + h \left(\frac{55}{24} f_n - \frac{59}{24} f_{n-1} + \frac{37}{24} f_{n-2} - \frac{9}{24} f_{n-3} \right) \quad (2.27)$$

$$y_{n+1} = y_n + h \left(\frac{1901}{720} f_n - \frac{1387}{360} f_{n-1} + \frac{109}{30} f_{n-2} - \frac{637}{360} f_{n-3} + \frac{251}{720} f_{n-4} \right) \quad (2.28)$$

$$y_{n+1} = y_n + h \left(\frac{4277}{1440} f_n - \frac{2641}{480} f_{n-1} + \frac{4991}{720} f_{n-2} - \frac{3649}{720} f_{n-3} + \frac{959}{480} f_{n-4} - \frac{95}{288} f_{n-5} \right) \quad (2.29)$$

$$\begin{aligned} y_{n+1} = y_n + h & \left(\frac{198721}{60480} f_n - \frac{18637}{2520} f_{n-1} + \frac{235183}{20160} f_{n-2} - \frac{10754}{945} f_{n-3} + \frac{135713}{20160} f_{n-4} \right. \\ & \left. - \frac{5603}{2520} f_{n-5} + \frac{19087}{60480} f_{n-6} \right) \end{aligned} \quad (2.30)$$

$$\begin{aligned} y_{n+1} = y_n + h & \left(\frac{16083}{4480} f_n - \frac{1152169}{120960} f_{n-1} + \frac{242653}{13440} f_{n-2} - \frac{296053}{13440} f_{n-3} + \frac{2102243}{120960} f_{n-4} \right. \\ & \left. - \frac{115747}{13440} f_{n-5} + \frac{32863}{13440} f_{n-6} - \frac{5257}{17280} f_{n-7} \right) \end{aligned} \quad (2.31)$$



Ecuatii diferențiale cu derivate partiale

