# Metode de tip Runge Kutta

Gabriel Dimitriu

June 22, 2005

# Contents


**1 Considerente generale** **5**

1.1 Metoda Taylor . . . . . 6
1.1.1 Prezentare implementarii si exemple . . . . . 9
1.2 Consistenta si stabilitate . . . . . 12

**2 Metoda Euler** **17**
2.1 Prezentarea metodei . . . . . 17
2.2 Consistenta . . . . . 18

2.3 Prezentare implementarii si exemple . . . . . 18
2.3.1 Implementare . . . . . 18
2.3.2 Rezultatele rularii . . . . . 19

**3 Metoda Euler-Couchy** **21**
3.1 Prezentare generala . . . . . 21
3.2 Consistenta . . . . . 24
3.3 Prezentarea implementarii si exemple . . . . . 24
3.3.1 Implementare . . . . . 24
3.3.2 Rezultatele rularii . . . . . 25

**4 Metoda Runge-Kutta** **27**
4.1 Prezentarea metodei . . . . . 27
4.2 Consistenta . . . . . 30
4.3 Prezentarea implementarii si exemple . . . . . 31
4.3.1 Implementare . . . . . 31
4.3.2 Rezultatele rularii . . . . . 33

**5 Anexa 1 (parser de functii)** **35**
5.1 Definierea clasei . . . . . 35
5.2 Implementarea clasei . . . . . 36

# Chapter 1

# Considerente generale

In tot ceea ce urmeaza vom considera domeniile de existenta:

$$\begin{aligned} A \times B &\subset R^2 \\ [a,b] &\subset A \end{aligned}$$

Pe aceste domenii de existenta vom considera urmatoare problema Couchy:

$$\begin{aligned} x'(t) &= u(t, x(t)) \\ x(a) &= \alpha_0 \end{aligned} \tag{1.1}$$

cu urmatoare conditii in plus de existenta

$$\begin{gathered} t \in [a,b] \\ \alpha_0 \in R \; fixat \\ u : A \times B \to R \, continua \end{gathered}$$

De la teoria pentru problema Couchy prezentata in [1] avem urmatoarea teorema:

**Teorema 1**

Daca $\exists ! L > 0$ astfel incit $|u(t,z) - u(t,w)| \leq L|z-w| \; \forall t \in A$ si $\forall z, w \in B$ atunci $\exists ! \, x$ solutie pentru ecuatia (1.1).

Vom presupune ca (1.1) are solutie unica conform teoremei anterioare.

Vom presupune ca $u$ este analitica.Atunci se stie ca si solutia $x$ este analitica si poate fi scrisa astfel:

$$x(t) = \sum_{k=0}^{\infty} \frac{x^{(k)}(a)}{k!}(t-a)^k, \; t \in [a, a+r]$$

unde r este raza de convergenta seriei de puteri

$$\sum_{k\geq 0} \frac{x^{(k)}(a)}{k!}(t-a)^k$$

Deci vom putea face aproximarea:

$$x(t) \simeq \sum_{k=0}^{n} \frac{x^{(k)}(a)}{k!}(t-a)^k$$

**Observatie:**
Vom nota

$$T_{n,x,s_0}(t) = \sum_{k=0}^{n} \frac{x^{(k)}(s_0)}{k!}(t-s_0)^k \tag{1.2}$$

Din cursul [3] avem urmatoarele propozitii:

**Propositie 2** *Fie $f, g : A \to R$ continue in $a \in A$ si $\lambda, \mu \in R$ atunci $\lambda f + \mu g$ este continua in a.*

**Propositie 3** *Fie $g : A \to B$ continua in $a \in A$ si fie $g : B \to R$ continua in $f(a)$ atunci $g \circ f : A \to R$ este continua in a.*

## 1.1 Metoda Taylor

Vom considera urmatoarea problema Couchy:

$$\begin{gathered} x'(t) = u(t, x(t)) \\ x(a) = \alpha_0 \\ \forall t \in [a, b] \\ \alpha_0 \in R \ fixat \end{gathered} \tag{1.3}$$

unde

$$\begin{gathered} A \times B \subset R^2 \\ u : A \times B \to R \ si \ u \in C^n \\ [a, b] \in A \end{gathered}$$

Se observa ca aceste conditii sunt o extindere a conditiile necesare prezentate in sectiune 1 ca problema (1.3) sa aiba solutie unica, deci putem considera adevarate toate notatille din sectiunea 1.

In continuare vom prezenta o aproximare a derivatelor de ordin superior pentru solutia x in punctul a.

Fie $u_0(t, y) = y \Leftrightarrow u_0(t, x(t)) = x(t) \Leftrightarrow u_0(a, x(a)) = x(a)$ folosind conditia problemei Couchy avem

$$u_0(a, \alpha_0) = \alpha_0 \tag{1.4}$$

Fie $u_1(t, y) = u(t, y) \Leftrightarrow u_1(t, x(t)) = u(t, x(t))$ folosind ecuatia problemei Couchy avem

$$x'(a) = u_1(a, x(a)) \tag{1.5}$$

si folosind conditia problemei Couchy avem

$$x'(a) = u_1(a, \alpha_0) \tag{1.6}$$

Cu aceste notatii vom prelucra prima ecuatie a problemei Couchy pentru a determina derivatele de ordin superior ale solutiei.

Din ecuatia problemei Couchy cu notatia (1.5) avem

$$x'(t) = u_1(t, x(t)) \tag{1.7}$$

Derivind aceasta relatie in functie de t si y; folosind notatiile anterioare avem:

$$x''(t) = \frac{\partial u_1}{\partial t}(t, x(t)) + \frac{\partial u_1}{\partial y}(t, y) \cdot u(t, x(t)) \tag{1.8}$$

Fie

$$u_2(t, y) = \frac{\partial u_1}{\partial t}(t, y) + \frac{\partial u_1}{\partial y}(t, y) \cdot u(t, y) \tag{1.9}$$

Inlocuind pe $u_2$ in relatia (1.8) avem $x''(t) = u_2(t, x(t))$ iar $x''(a) = u_2(a, x(a))$ folosind ipoteza avem

$$x''(a) = u_2(a, \alpha_0) \tag{1.10}$$

In continuare vom folosi procedeul inductiei matematice pentru a deterimina derivate de orice ordin a solutiei in punctul a.

Presupunem adevarate relatiile $u_k$ deci avem

$$x^{(k)}(t) = u_k(t, x(t)) \tag{1.11}$$

si

$$x^{(k)}(a) = u_k(a, \alpha_0) \tag{1.12}$$

pentru k si vom arata ca ramin adevarate pentru k+1.

Derivam relatia (1.11) in funcite de t si y; folosind notatiile anterioare avem:

$$x^{(k+1)}(t) = \frac{\partial u_k}{\partial t}(t, x(t)) + \frac{\partial u_k}{\partial y}(t, y) \cdot u(t, x(t)) \tag{1.13}$$

Fie

$$u_{k+1}(t, y) = \frac{\partial u_k}{\partial t}(t, y) + \frac{\partial u_k}{\partial y}(t, y) \cdot u(t, y) \tag{1.14}$$

Inlocuind relatia (1.14) in relatia (1.13) avem

$$x^{(k+1)}(t) = u_{k+1}(t, x(t))$$

Iar pentru t=a avem

$$x^{(k+1)}(a) = u_{k+1}(a, \alpha_0) \tag{1.15}$$

Ceea ce incheie procesul de inductie matematica.

Inlocuind relatiile (1.6),(1.10),(1.12) si (1.15) in relatia (1.2) avem

$$T_{n,x,a}(t) = \sum_{k=0}^{n} \frac{u_k(a, \alpha_0)}{k!}(t-a)^k \tag{1.16}$$

unde $u_k$ sunt date de relatiile (1.4),(1.5),(1.9) si (1.14).
Pe intervalul in care variaza t vom lua o diviziune astfel

$$a = t_0 < t_1 < ... < t_m \leq b$$

Fie $y : [a, b] \to R$ construita astfel pentru $\forall t \in [t_0, t_1]$ avem

$$y(t) = \sum_{k=0}^{n} \frac{u_k(t_0, \alpha_0)}{k!}(t-t_0)^k \tag{1.17}$$

Notam $x_0 = \alpha_0$ si atunci relatia (1.17) se scrie astfel

$$y(t) = \sum_{k=0}^{n} \frac{u_k(t_0, x_0)}{k!}(t-t_0)^k \tag{1.18}$$

Vom face iteratii ale $x_i$ in punctele diviziunii pentru a afla termenul general al solutie.
Fie $y : [t_0, t_1] \to R$ si

$$x_1 := y(t_1) = \sum_{k=0}^{n} \frac{u_k(t_0, x_0)}{k!}(t_1-t_0)^k$$

Fie $y : [t_1, t_2] \to R$ si

$$x_2 := y(t_2) = \sum_{k=0}^{n} \frac{u_k(t_1, x_1)}{k!}(t_2-t_1)^k$$

Fie $y : [t_0, t_k] \to R \ \ si \ \ x_k := y(t_k)$
Fie $y : [t_k, t_{k+1}] \to R$ si

$$y(t) = \sum_{j=0}^{n} \frac{u_j(t_k, x_k)}{j!}(t-t_k)^j$$

din definitia lui $x_{k+1}$ avem:

$$x_{k+1} := y(t_{k+1}) = \sum_{j=0}^{n} \frac{u_j(t_k, x_k)}{j!}(t_{k+1}-t_k)^j$$

Fie $z : \delta = \{t_0, t_1, ..., t_m\} \to R$

$$z(t_k) = x_k = \sum_{j=0}^{n} \frac{u_j(t_{k-1}, x_{k-1})}{j!}(t_k-t_{k-1})^j$$

**Definitie 4** $(x|_\delta \simeq z)$ *se numeste metoda lui Taylor pentru problema Couchy asociata (1.3).*

Metoda Taylor poate fi rescrisa astfel:

$$\begin{aligned} x_0 &= \alpha_0 \\ \frac{x_{k+1}-x_k}{t_{k+1}-t_k} &= \sum_{j=1}^{n} \frac{u_j(t_k,x_k)}{j!}(t_{k+1}-t_k)^{j-1} \quad k=\overline{0,m-1} \end{aligned} \tag{1.19}$$

Acest ultim sistem se numeste sistem de ecuatii cu diferente divizate.

### 1.1.1 Prezentare implementarii si exemple

In continuare vom lua urmatorul exemplul:

$$\begin{aligned} x_0 &= 0 \\ x'(t) &= t + x(t) \;\; t \in [0,1] \end{aligned}$$

Sa se rezolve aproximativ ecuatia cu metoda lui Taylor in cazul in care avem urmatoarele diviziuni $\delta_1 = \{0, 1/2, 1\}$ si $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$.

#### Implementare

Pentru a implementa aceasta metoda avem nevoie de derivata deci va trebui sa implementam o functie care sa faca derivatele partiale ale functiei $u_k(t, y)$.

Pentru implementare vom folosi urmatoarea formula de derivare partiala

$$\frac{\partial f(x,y)}{\partial x}(a,b) = \frac{f(a+h/2,b) - f(a-h/2,b)}{h}$$

Iar cea de ordin doi este

$$\frac{\partial^2 f(x,y)}{\partial x \partial y}(a,b) = \left( \frac{\partial f(x,y)}{\partial x}(a,b+h/2) - \frac{\partial f(x,y)}{\partial x}(a,b-h/2) \right) /h$$

Asadar derivata va fi implementata recursiv conform formulelor prezentate anterior.

```
#include"parser_func.cpp"
#include<math.h>
#include<stdlib.h>
#include<stdio.h>
parser_func *parser;
double *x;
long factorial(long val)
{
    if(val==0L) return 1L;
    else return val*factorial(val-1);
}
double derivata(double t,double y,double h,long ordin)
{
    double val1,val2;
    double valt,valy;
```

```
        double temp;
        if(ordin==0) return x[0];
        else if(ordin==1)
        {
              parser->set_var('t',t);
              parser->set_var('y',y);
              parser->eval_func(&val1);
              return val1;
        }
        else if(ordin==2)
        {
                    parser->set_var('t',t+0.5*h);
                    parser->set_var('y',y);
                    parser->eval_func(&val1);
                    parser->set_var('t',t-0.5*h);
                    parser->set_var('y',y);
                    parser->eval_func(&val2);
                    valt=(val1-val2)/h;
                    parser->set_var('y',y+0.5*h);
                    parser->set_var('t',t);
                    parser->eval_func(&val1);
                    parser->set_var('y',y-0.5*h);
                    parser->set_var('t',t);
                    parser->eval_func(&val2);
                    valy=(val1-val2)/h;
                    parser->set_var('t',t);
                    parser->set_var('y',y);
                    parser->eval_func(&temp);
                    valy=valy*temp;
                    return valt+valy;
        }
        else
        {
                    val1=derivata(t+0.5*h,y,h,ordin-1);
                    val2=derivata(t-0.5*h,y,h,ordin-1);
                    valt=(val1-val2)/h;
                    val1=derivata(t,y+0.5*h,h,ordin-1);
                    val2=derivata(t,y-0.5*h,h,ordin-1);
                    valy=(val1-val2)/h;
                    parser->set_var('t',t);
                    parser->set_var('y',y);
                    parser->eval_func(&temp);
                    valy=valy*temp;
                    return valt+valy;
        }
        return 0.0;
}
int main(int argc, char* argv[])
```

```
{
    double *delta;
    char *dfunc;
    long n,n1;
    double h;
    double sum;
    long i,j;
    parser=new parser_func;
    dfunc=(char *)calloc(100,sizeof(char));
    printf("Introduceti numarul de valori ale diviziunii\n");
    scanf("%ld",&n1);
    delta=(double *)calloc(n1,sizeof(double));
    x=(double *)calloc(n1,sizeof(double));
    printf("Introduceti diviziunea\n");
    for(i=0;i<n1;i++)
        scanf("%lf",&delta[i]);
    printf("Introduceti conditia initiala\n");
    scanf("%lf",&x[0]);
    printf("Introduceti parametrul n\n");
    scanf("%ld",&n);
    printf("Introduceti pasul derivatei\n");
    scanf("%lf",&h);
    printf("Introduceti functia cu t si y\n");
    fflush(stdin);
    dfunc=gets(dfunc);
    parser->set_function(dfunc);
    for(i=1;i<n1;i++)
    {
        sum=0.0;
        for(j=1;j<=n;j++)
            sum=sum+derivata(delta[i-1],x[i-1],h,j)*pow(delta[i]-delta[i-1],j)/factorial(j);
        x[i]=x[i-1]+sum;
    }
    printf("Valorile in diviziune\n");
    for(i=0;i<n1;i++)
        printf("%lf ",x[i]);
    printf("\n");
    return 0;
}
```

Clasa parser_func va fi prezentata in Anexa 1 si contine parserul de functii.

Functia set_var(caracter,valoare) seteaza variabila caracter (care trebuie sa fie un caracter majuscul sau minuscul intre a-z deoarece nu tine cont de litere mari sau mici) cu valoarea data in valoare.

Functia eval_func evalueaza functia setata prin set_function cu variabilele setata anterior.

**Rezultatele rularii**

In cazul acesta vom rula cu pasul pentru h=0.00001

Pentru $\delta_1 = \{0, 1/2, 1\}$ si functia data $t + y$ avem urmatoarele rezultate

| | | | |
|---|---|---|---|
| $t$ | 0 | 0.5 | 1.0 |
| $x(Teoretic)$ | 0 | 0.1487 | 0.71828 |
| $x(n = 1)$ | 0 | 0 | 0.25 |
| $x(n = 2)$ | 0 | 0.125 | 0.64062 |
| $x(n = 3)$ | 0 | 0.1458 | 0.70876 |

Pentru $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$ si functia data $t + y$ avem urmatoarele rezultate

| | | | | | |
|---|---|---|---|---|---|
| $t$ | 0 | 0.25 | 0.5 | 0.75 | 1.0 |
| $x(Teoretic)$ | 0 | 0.034 | 0.1487 | 0.367 | 0.71828 |
| $x(n = 1)$ | 0 | 0 | 0.0625 | 0.20312 | 0.44140 |
| $x(n = 2)$ | 0 | 0.0312 | 0.14160 | 0.35330 | 0.69485 |
| $x(n = 3)$ | 0 | 0.3385 | 0.14828 | 0.36615 | 0.71683 |

Se observa cu cit diviziunea are mai multe puncte cu atit mai bine este aproximata solutia in punctul final.

Se observa ca pentru n=1 este Euler dar pentru n=2 are o precizie mai buna decit Euler-Couchy iar pentru n=3 are o precizie comparabila cu Runge-Kutta. Bineinteles toate acestea cu dezavantajul de a fi mare consumatoare de resurse pentru derivate si factorial.

Pentru n>4 derivata in modul in care a fost implementata nu functioneaza deci trebuie sa ne limitam la n=4.

## 1.2 Consistenta si stabilitate

Pentru problema Couchy (1.1) avem:

Fie $h \in (0, b - a)$ si $n = n_h = [(b - a)/h]$.

Din aceasta creeam diviziunea $\delta_h = \{t_0, t_1, ..., t_n\}$ si $u_h : \delta_h \times B \to R$ si ecuatiile

$$\begin{aligned} x_0 &= \alpha_0 \\ \frac{x_{j+1} - x_j}{h} &= u_h(t_j, x_j) \end{aligned} \tag{1.20}$$

Fie $z$ solutia ecuatiei (1.20).

**Definitie 5** *$x|_{\delta_h} \simeq z$ se numeste metoda ecuatiilor cu diferente.*

**Remarka 6** *Pentru $u_h = u \; \forall h$ atunci se va obitine metoda lui Euler.*

**Remarka 7** *Pentru $u_h = 1/2(u(t, y)+u(t+h, y+hu(t, y))$ atunci se va obtine metoda Euler-Couchy.*

**Remarka 8** *Pentru $u_h = \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4)$ se va obtine metoda lui Runge-Kutta.*

Vom nota $\delta_h^+ = \{t_0, t_1, ..., t_{n_h-1}\}$ *si* $\delta_h^- = \{t_1, t_2, ..., t_{n_h}\}$.

**Definitie 9** *Spunem ca metoda ecuatiilor cu diferente este consistenta daca*

$$\lim_{h\to 0}\max_{t\in\delta_h^+}\left|\frac{x(t+h)-x(t)}{h}-u_h(t,x(t))\right|=0$$

**Teorema 10** *Daca*

$$\lim_{h\to 0}\max_{t\in\delta_h^+}|u(t,x(t))-u_h(t,x(t))|=0 \tag{1.21}$$

*Atunci metoda ecuatiilor cu diferente este consistenta.*

**Demostratie** Presupunem $u$ continua atunci si $x'$ continua.

Atunci relatia (1.21) se scrie:pentru $\forall\varepsilon>0\ \exists h_0$ a.i. $h<h_0$ avem

$$\max_{t\in\delta_h^+}|u(t,x(t))-u_h(t,x(t))|<\varepsilon$$

Explicitind maximul avem ca pentru $\forall t\in\delta_h^+$

$$|u(t,x(t))-u_h(t,x(t))|<\varepsilon \tag{1.22}$$

Dar

$$\left|\frac{x(t+h)-x(t)}{h}-u_h(t,x(t))\right|\le\left|\underbrace{\frac{x(t+h)-x(t)}{h}}_{A}-u(t,x(t))\right|+|u(t,x(t)-u_h(t,x(t))|$$

Explicitind A cu definitia derivate atunci pentru $c\in(t,t+h)$ si utilizind presupunerile din metoda ecuatiilor cu diferente $(u_h(t,x(t))=x'(t))$ avem

$$\left|\frac{x(t+h)-x(t)}{h}-u_h(t,x(t))\right|\le|x'(c)-x'(t)|+|u(t,x(t))-u_h(t,x(t))| \tag{1.23}$$

Deoarece $x'$ este continua pe un interval compact atunci $x'$ este uniform continua si putem scrie

$$\varepsilon>0\ \exists\eta_\varepsilon:|t-s|<\eta_\varepsilon\Rightarrow|x'(t)-x'(s)|<\varepsilon \tag{1.24}$$

Daca $h<\eta_\varepsilon\Rightarrow|x'(t)-x'(c)|<\varepsilon$.

Daca $h<\min(h_0,\eta_\varepsilon)$ atunci folosind (1.22) si (1.24) avem

$$\left|\frac{x(t+h)-x(t)}{h}-u_h(t,x(t))\right|\le\varepsilon+\varepsilon\ \ \forall t\in\delta_h^+$$

Deci

$$\lim_{h\to 0}\max_{t\in\delta_h^+}\left|\frac{x(t+h)-x(t)}{h}-u_h(t,x(t))\right|=0$$

■

**Remarka 11**

$$\lim_{h\to 0}\max_{h\in\delta_h^-}\left|\frac{x(t)-x(t-h)}{h}-u_h(t-h,x(t-h))\right|=0$$

Notam

$$\lambda_h W(t) = \frac{W(t+h) - W(t)}{h} - u_h(t, W(t))$$

unde $W : \delta_h \backslash \{n_h\} \to R$ iar

$$\|\lambda_h W\| = \max_{t \in \delta_h^+} \left| \frac{W(t+h) - W(t)}{h} - u_h(t, W(t)) \right|$$

Cu aceste notatii relatia (1.21) devine

$$\lim_{h \to 0} \|\lambda_h x\| = 0$$

**Definitie 12** *Se spune ca metoda ecuatiilor cu diferente este stabila daca* $\exists \rho > 0, \forall (x_h)_{0<h<b-a}$, $x_h : \delta_h \to R$ *cu proprietatile*

$$\begin{aligned} \exists h_0, \forall h &\leq h_0 \Rightarrow \|x_h - x\|_h \leq \rho \\ x_h(a) &= \alpha_0 \\ \lim_{h \to 0} \|\lambda_h x_h - \lambda_h x\| &= 0 \end{aligned}$$

*sa avem*

$$\lim_{h \to 0} \|x_h - x\| = 0$$

**Teorema 13** *Daca* $\exists L, \rho, h_0$ *cu* $0 < h_0 < b - a$ *a.i.* $\forall h \leq h_0, \forall t \in \delta_h, |y - x(t)| \leq \rho, |W - x(t)| \leq \rho$ *atunci*

$$|u(t, y) - u_h(t, W)| \leq L |y - W| \tag{1.25}$$

*Atunci metoda ecuatiilor cu diferente este stabila.*

**Demostratie** Fie $L, \rho, h_0$ ca in (1.25) iar $\rho, h_0$ satisfac conditiile detfinitiei.

Vom arata ca daca $h \leq h_0$ avem $t \in \delta_h$ si

$$|x_h(t) - x(t)| \leq (t-a) \exp(L(t-a)) \|\lambda_h x_h - \lambda_h x\| \tag{1.26}$$

Daca am demostrat (1.26) atunci avem

$$\max_{t \in \delta_h} |x_h(t) - x(t)| \leq (b-a) \exp(L(b-a)) \|\lambda_h x_h - \lambda_h x\| \overset{definitie}{\to} 0$$

Relatia (1.26) o vom demostra prin inductie.

Pentru $t = a$ avem $|x_h(a) - x(a)| = |\alpha_0 - \alpha_0| = 0$.

Presupunem relatia (1.26) adevarata pentru $t_j = t$ si o demostram pentru $t_{j+1} = t + h$.

Avem

$$\begin{aligned} x_h(t+h) &= x_h(t) + h\lambda_h x_h(t) + h u_h(t, x_h(t)) \\ x(t+h) &= x(t) + h\lambda_h x(t) + h u(t, x(t)) \end{aligned}$$

Scazind in modul aceste doua relatii avem

$$|x_h(t+h) - x(t+h)| \leq |x_h(t) - x(t)| + h |u_h(t, x_h(t)) - u(t, x(t))| + h \|\lambda_h x_h - \lambda_h x\|$$

Aplicind conditiile din definitie avem

$$|x_h(t+h) - x(t+h)| \leq |x_h(t) - x(t)| + hL\,|x_h(t) - x(t)| + h\,\|\lambda_h x_h - \lambda_h x\|$$

Aplicind relatia (1.26) avem

$$|x_h(t+h) - x(t+h)| \leq (1+hL)(t-a)\exp(L(t-a))\,\|\lambda_h x_h - \lambda_h x\| + h\,\|\lambda_h x_h - \lambda_h x\|$$

Desfacind parantezele si facind calcule elementare avem

$$|x_h(t+h) - x(t+h)| \leq (t-a)\exp(L(t+h-a))\,\|\lambda_h x_h - \lambda_h x\| + h\,\|\lambda_h x_h - \lambda_h x\|$$

Facind majorarea $\exp(L(t+h-a)) > \exp(0)$ si dind factor comun avem

$$|x_h(t+h) - x(t+h)| \leq (t+h-a)\exp(L(t+h-a))\,\|\lambda_h x_h - \lambda_h x\|$$

■

# Chapter 2

# Metoda Euler

## 2.1 Prezentarea metodei

Metoda lui Euler este o particularizare a metodei lui Taylor prezentata in sectiunea anterioara.

Vom considera urmatoarea problema Couchy:

$$\begin{gathered} x'(t) = u(t, x(t)) \\ x(a) = \alpha_0 \\ \forall t \in [a, b] \\ \alpha_0 \in R \ fixat \end{gathered} \tag{2.1}$$

unde

$$\begin{gathered} A \times B \subset R^2 \\ u : A \times B \to R \ si \ u \in C^n \\ [a, b] \in A \end{gathered}$$

Aceasta problema Couchy este rezolvata cu metoda lui Taylor pe diviziunea $\delta = \{t_1, ..., t_m\}$ in care n=1 si $x_0 = \alpha_0$ si atunci avem sistemul de ecuatii cu diferente divizate inlocuind aceste particularizari in (1.19):

$$\begin{aligned} x_{k+1} &= x_k + u(t_k, x_k) \cdot (t_{k+1} - t_k) \\ x_0 &= \alpha_0 \end{aligned}$$

sau

$$\begin{aligned} x_0 &= \alpha_0 \\ \frac{x_{k+1} - x_k}{t_{k+1} - t_k} &= u(t_k, x_k) \end{aligned}$$

pentru $\forall k \in \{0, ..., m\}$.

Daca notam $z(t_k) = x_k$ atunci

**Definitie 14** *$x|_\delta \simeq z$ se numeste metoda lui Euler.*

## 2.2 Consistenta

Fie

$$u_h(t,y) = u(t,y)$$

Pentru a aplica Teorema (1.21) trebuie sa facem diferenta

$$u_h(t,x(t)) - u(t,x(t)) = 0$$

pentru $\forall t \in \delta_h \backslash \{t_{n_h}\}$.
Deci

$$|u_h(t,x(t)) - u(t,x(t))| = 0$$

Asadar metoda Euler este consistenta.

## 2.3 Prezentare implementarii si exemple

In continuare vom lua urmatorul exemplul:

$$\begin{aligned} x_0 &= 0 \\ x'(t) &= t + x(t) \;\; t \in [0,1] \end{aligned}$$

Sa se rezolve aproximativ ecuatia cu metoda lui Euler in cazul in care avem urmatoarele diviziuni $\delta_1 = \{0, 1/2, 1\}$ si $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$.

### 2.3.1 Implementare

```
#include"parser_func.cpp"
    #include<math.h>
    #include<stdlib.h>
    #include<stdio.h>
    int main(int argc, char* argv[])
    {
        parser_func *parser;
        double *delta;
        char *dfunc;
        long n,n1;
        double *x;
        double val1,val2,temp;
        long i;
        parser=new parser_func;
        dfunc=(char *)calloc(100,sizeof(char));
        printf("Introduceti numarul de valori ale diviziunii\n");
```

```
    scanf("%ld",&n1);
    delta=(double *)calloc(n1,sizeof(double));
    x=(double *)calloc(n1,sizeof(double));
    printf("Introduceti diviziunea\n");
    for(i=0;i<n1;i++)
        scanf("%lf",&delta[i]);
    printf("Introduceti conditia initiala\n");
    scanf("%lf",&x[0]);
    printf("Introduceti functia cu t si y\n");
    fflush(stdin);
    dfunc=gets(dfunc);
    parser->set_function(dfunc);
    for(i=1;i<n1;i++)
    {
        parser->set_var('t',delta[i-1]);
        parser->set_var('y',x[i-1]);
        parser->eval_func(&temp);
        x[i]=x[i-1]+temp*(delta[i]-delta[i-1]);
    }
    printf("Valorile in diviziune\n");
    for(i=0;i<n1;i++)
        printf("%lf ",x[i]);
    printf("\n");
    return 0;
}
```

Clasa parser_func va fi prezentata in Anexa 1 si contine parserul de functii.

Functia set_var(caracter,valoare) seteaza variabila caracter (care trebuie sa fie un caracter majuscul sau minuscul intre a-z deoarece nu tine cont de litere mari sau mici) cu valoarea data in valoare.

Functia eval_func evalueaza functia setata prin set_function cu variabilele setata anterior.

### 2.3.2 Rezultatele rularii

Pentru $\delta_1 = \{0, 1/2, 1\}$ si functia data $t + y$ avem urmatoarele rezultate

| | | | |
|---|---|---|---|
| $t$ | 0 | 0.5 | 1.0 |
| $x(Euler)$ | 0 | 0 | 0.25 |
| $x(Teoretic)$ | 0 | 0.1487 | 0.71828 |
| $\|x(Teoretic) - x(Euler)\|$ | 0 | 0.1487 | 0.46828 |

Pentru $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$ si functia data $t + y$ avem urmatoarele rezultate

| | | | | | |
|---|---|---|---|---|---|
| $t$ | 0 | 0.25 | 0.5 | 0.75 | 1.0 |
| $x(Euler)$ | 0 | 0.0 | 0.0625 | 0.2031 | 0.441 |
| $x(Teoretic)$ | 0 | 0.034 | 0.1487 | 0.367 | 0.71828 |
| $\|x(Teoretic) - x(Euler)\|$ | 0 | 0.034 | 0.0862 | 0.1639 | 0.27728 |

Se observa cu cit diviziunea are mai multe puncte cu atit mai bine este aproximata solutia in punctul final.

# Chapter 3

# Metoda Euler-Couchy

## 3.1 Prezentare generala

Fie $h \in (0, b-a)$ o constanta aleasa atunci vom defini

$$n_h = \left[\frac{b-a}{h}\right]$$

cu aceasta valoare vom defini diviziunea $\delta$ astfel

$$\begin{aligned} t_0 &= a \\ t_i &= t_0 + i \cdot h \quad i = \overline{0, n_h} \end{aligned}$$

deci deviziunea $\delta = \{t_0, t_1, ..., t_{n_h}\}$
Cu aceste precizari sistemul de ecuatii cu diferente divizate (1.19) se rescrie astfel:

$$\begin{aligned} x_0 &= \alpha_0 \\ x_{k+1} &= x_k + h\sum_{j=1}^{n} \frac{u_j(t_k, x_k)}{j!} h^{j-1} \end{aligned}$$

Vom nota

$$f_n(t, y) = \sum_{j=1}^{n} \frac{u_j(t, y)}{j!} h^{j-1}$$

si cu acesta notatie relatia (1.16) devine

$$T_{n,x,x_0}(s+h) = x(s) + h \cdot f_n(s, x(s)) \tag{3.1}$$

deci sistemul de ecuatii cu diferente divizate se rescrie astfel

$$\begin{aligned} x_0 &= \alpha_0 \\ x_{k+1} &= x_k + h \cdot f_n(t_k, x_k) \end{aligned} \tag{3.2}$$

**Teorema 15** *Cresterilor finite*

Daca $\exists L > 0$ astfel incit $|u(t,y) - u(t,v)| \leq L\,|y - v|\ \forall t \in [a,b]$ *si* $\forall y, v \in B$ si $|f_n(t,y) - f_n(t,v)| \leq |y - v|\ \ \forall t \in [a,b]$ *si* $\forall y, v \in B$ atunci urmatoarele formule au loc:

$$|x(t_k) - x_k| \leq \frac{h^n}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty \frac{\exp(Lkh) - 1}{L} \quad k = \overline{0,n} \tag{3.3}$$

unde $\left\|x^{(n+1)}\right\|_\infty = \sup\limits_{t\in[a,b]} \left|x^{(n+1)}(t)\right|$.

**Demostratie** Vom demostra prin inductie dupa k.

Pentru $k = 0$ avem $|x(t_0) - x_0| = |\alpha_0 - \alpha_0| = 0 \leq 0$.

Presupunem adevarata relatia (3.3) pentru $k$ si o demostram pentru $k+1$.

Conform formulei lu Taylor cu reste Lagrange avem

$$x(t_{k+1}) = T_{n,x,t_k}(t_{k+1}) + \underbrace{\frac{1}{(n+1)!} x^{(n+1)}(\xi) \cdot (t_{k+1} - t_k)^{n+1}}_{r}$$

sau

$$x(t_{k+1}) = T_{n,x,t_k}(t_k + h) + r$$

unde

$$|r| \leq \frac{1}{(n+1)!} x^{(n+1)}(\xi) \cdot h^{n+1}$$

Atunci

$$x(t_{k+1}) \overset{(3.1)}{=} x(t_k) + h \cdot f_n(t_k, x(t_k)) + r$$

Vom face diferenta $x(t_{k+1}) - x_{k+1}$ in care vom aplica relatia anterioara si avem

$$x(t_{k+1}) - x_{k+1} = x(t_k) - x_{k+1} + h \cdot f_n(t_k, x(t_k)) + r$$

In aceasta relatie aplicind (3.2) avem

$$x(t_{k+1}) - x_{k+1} = x(t_k) - x_k + h(f_n(t_k, x(t_k)) - f_n(t_k, x_k)) + r$$

Aplicind modul avem ambilor termeni si efectuind operatii elementare cu module avem

$$|x(t_{k+1}) - x_{k+1}| \leq |x(t_k) - x_k| + h\,|f_n(t_k, x(t_k)) - f_n(t_k, x_k)| + |r|$$

Aplicind ipoteza referitoare la functia $f_n$ (lischitziana in al doilea argument) avem

$$|x(t_{k+1}) - x_{k+1}| \leq |x(t_k) - x_k|\,(1 + hL) + |r|$$

Aplicind ipoteza de inductie matematica avem

$$|x(t_{k+1}) - x_{k+1}| \leq (1 + hL)\frac{h^n}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty \frac{\exp(LKh) - 1}{L} + \frac{h^{n+1}}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty$$

Efectuind citeva prelucrari elementare avem

$$|x(t_{k+1}) - x_{k+1}| \leq \frac{h^n}{L(n+1)!} \left\|x^{(n+1)}\right\|_\infty ((1+hL)(\exp(Lkh) - 1) + Lh)$$

Aplicind aproximatia $1 + hL < \exp(hL)$ relatia devine

$$|x(t_{k+1}) - x_{k+1}| \leq \frac{1}{L}\frac{h^n}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty ((\exp(hkL) - 1)\exp(hL) + Lh)$$

sau

$$|x(t_{k+1}) - x_{k+1}| \leq \frac{1}{L}\frac{h^n}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty (\exp(Lh(k+1)) - \exp(Lh) + Lh)$$

Aplicind inca o data aproximatia $1 > -\exp(hL) + Lh$ relatia devine

$$|x(t_{k+1}) - x_{k+1}| \leq \frac{1}{L}\frac{h^n}{(n+1)!} \left\|x^{(n+1)}\right\|_\infty (\exp(Lh(k+1)) - 1)$$

■

Fie $x : [a, b] \to R$ solutia problemei (2.1).
Fie $h < b - a$ si $n_h = \left[\frac{b-a}{h}\right]$ iar $t_i = t_0 + ih \ \forall i = \overline{0, n_h}$ iar $t \in [a, b)$ cu $t + h \in [a, b)$.
Integrind x'(s) intre t si t+h avem

$$x(t+h) - x(t) = \int_t^{t+h} x'(s)ds$$

Aceata integrala o vom aproxima prin formula trapezului si avem

$$x(t+h) - x(t) \simeq \frac{h}{2}(x'(t+h) + x'(t))$$

Dar din furmulare aproblemei Couchy avem $x'(t) = u(t, x(t))$, cu aceasta inlocuind in relatia anterioara avem

$$x(t+h) - x(t) \simeq \frac{h}{2}(u(t+h, x(t+h)) + u(t, x(t)))$$

Aproximam $u(t + h, x(t + h))$ cu teorema cresterilor finite si avem

$$x(t+h) - x(t) \simeq \frac{h}{2}(u(t+h, x(t) + h \cdot u(t, x(t))) + u(t, x(t))$$

de aici deducem ecuatiile

$$\begin{aligned} x_0 &= \alpha_0 \\ \frac{x_{k+1} - x_k}{t_{k+1} - t_k} &= \frac{1}{2}(u(t_k + h, x_k + h \cdot u(t_k, x_k)) + u(t_k, x_k)) \end{aligned}$$

Daca notam $z(t_k) = x_k \ \ k = \overline{0, n}$ avem urmatoarea definitie

**Definitie 16** *$x|_{\delta_h} \simeq z$ se numeste metoda Euler-Couchy.*

## 3.2 Consistenta

Fie

$$u_h(t,y) = 1/2(u(t,y) + u(t+h, y+hu(t,y))$$

Pentru a aplica Teorema (1.21) trebuie sa facem diferenta

$$u_h(t,x(t)) - u(t,x(t)) = \frac{1}{2}(u(t+h, x(t)+hu(t,x(t))) - u(t,x(t))$$

Deoarece functia $x'(t) = u(t,x(t)) := u(t,y)$ este de fapt functia din conditia couchy si aplicind (2) si (3) ea este continua pe $\{s, x(t)+hx'(t) | t,s \in [a,b]\} = K$.

Se observa ca $K$ este marginita si inchisa deci este compacta pe $[a,b]$.

Cum $u$ este continua pe $K$ atunci ea este uniform continua asadar avem din definitia uniform continuitatii: $\varepsilon > 0, \exists \eta_\varepsilon > 0$ *a.i.* $|t-s| < \eta_\varepsilon$ *si* $|t-w| < \eta_\varepsilon$ *cu* $(t,y),(s,w) \in K$ atunci $|u(t,y) - u(s,w)| < \varepsilon$.

atunci $h < \eta_\varepsilon$ si $h * \max_{t\in[a,b]}(x'(t)) < \eta_\varepsilon$

Deci

$$|u(t+h, x(t)+hx'(t)) - u(t,x(t))| < \varepsilon$$

Asadar metoda Euler-Couchy este consistenta.

## 3.3 Prezentarea implementarii si exemple

In continuare vom lua urmatorul exemplul:

$$\begin{aligned} x_0 &= 0 \\ x'(t) &= t + x(t) \;\; t \in [0,1] \end{aligned}$$

Sa se rezolve aproximativ ecuatia cu metoda lui Euler-Couchy in cazul in care avem urmatoarele diviziuni $\delta_1$pentru care $h = 1/2$ si $\delta_2$ pentru care $h = 1/4$.

### 3.3.1 Implementare

```
#include"parser_func.cpp"
    #include<math.h>
    #include<stdlib.h>
    #include<stdio.h>
    int main(int argc, char* argv[])
    {
        parser_func *parser;
        double *delta,a,b,h;
        char *dfunc;
        long n;
        double *x;
        double sum1,sum2;
        long i;
```

```
parser=new parser_func;
dfunc=(char *)calloc(100,sizeof(char));
printf("Introduceti pasul h=");
scanf("%lf",&h);
printf("Introduceti limita inferioara a intervalului\n");
scanf("%lf",&a);
printf("Introduceti limita superioara a intervalului \n");
scanf("%lf",&b);
n=(long)((b-a)/h)+1;
x=(double *)calloc(n,sizeof(double));
delta=(double *)calloc(n,sizeof(double));
delta[0]=a;
for(i=1;i<n;i++)
      delta[i]+=i*h;
printf("Introduceti conditia initiala\n");
scanf("%lf",&x[0]);
printf("Introduceti functia cu t si y\n");
fflush(stdin);
dfunc=gets(dfunc);
parser->set_function(dfunc);
parser->set_var('t',delta[0]);
parser->set_var('y',x[0]);
parser->eval_func(&sum1);
for(i=1;i<n;i++)
{
      parser->set_var('t',delta[i-1]+h);
      parser->set_var('y',x[i-1]+h*sum1);
      parser->eval_func(&sum2);
      x[i]=x[i-1]+(delta[i]-delta[i-1])*(sum2+sum1)/2;
      sum1=sum2;
}
printf("Valorile in diviziune\n");
for(i=0;i<n;i++)
      printf("%lf ",x[i]);
printf("\n");
return 0;
}
```

Clasa parser_func va fi prezentata in Anexa 1 si contine parserul de functii.

Functia set_var(caracter,valoare) seteaza variabila caracter (care trebuie sa fie un caracter majuscul sau minuscul intre a-z deoarece nu tine cont de litere mari sau mici) cu valoarea data in valoare.

Functia eval_func evalueaza functia setata prin set_function cu variabilele setata anterior.

### 3.3.2 Rezultatele rularii

Pentru $\delta_1 = \{0, 1/2, 1\}$ adica pentru pasul $h = 0.5$ si intervalul $[0, 1.0]$ precum si functia data $t + y$ avem urmatoarele rezultate

| | | | |
|---|---|---|---|
| $t$ | 0 | 0.5 | 1.0 |
| $x(Euler - Couchy)$ | 0 | 0.125 | 0.5937 |
| $x(Teoretic)$ | 0 | 0.1487 | 0.71828 |
| $\|x(Teoretic) - x(Euler - Couchy)\|$ | 0 | 0.0237 | 0.124 |

Pentru $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$ adica pentru pasul $h = 0.25$ si intervalul $[0, 1.0]$ precum si functia data $t + y$ avem urmatoarele rezultate

| | | | | | |
|---|---|---|---|---|---|
| $t$ | 0 | 0.25 | 0.5 | 0.75 | 1.0 |
| $x(Euler - Couchy)$ | 0 | 0.03125 | 0.136719 | 0.3403 | 0.66961 |
| $x(Teoretic)$ | 0 | 0.034 | 0.1487 | 0.367 | 0.71828 |
| $\|x(Teoretic) - x(Euler - Couchy)\|$ | 0 | 0.00275 | 0.012 | 0.0267 | 0.04867 |

Se observa cu cit diviziunea are mai multe puncte cu atit mai bine este aproximata solutia in punctul final.

# Chapter 4

# Metoda Runge-Kutta

## 4.1 Prezentarea metodei

Fie urmatoarea problema Couchy

$$\begin{aligned} x(a) &= \alpha_0 \\ x'(t) &= u(t, x(t)) \end{aligned} \tag{4.1}$$

cu urmatoarele constringeri

$$\begin{aligned} t &\in [a, b] \\ \alpha_0 &\in B \\ u &: A \times B \to R \ \textit{continua} \end{aligned}$$

Se observa ca aceste conditii sunt identice cu conditiile necesare prezentate in sectiune 1 ca problema (1.3) sa aiba solutie unica, deci putem considera adevarate toate notatille din sectiunea 1. Fie acesta solutie unica $x$.

Fie $0 < h < (b-a)$ fixat si $n_h = [(b-a)/h]$ precum si diviziunea $\delta_h = \{t_0, t_1, ..., t_{n_h}\}$ definita astfel $t_0 = a, t_j = t_0 + jh \ \forall j = \overline{0, n_h}$ iar $t \in [a, b)$ cu $t + h \in [a, b)$.

Integrind x'(s) intre t si t+h avem

$$x(t+h) - x(t) = \int_t^{t+h} x'(s)ds$$

Acum vom aproxima integrala prin formula lui Simpson si impartind cu h avem

$$\frac{x(t+h) - x(t)}{h} \simeq \frac{1}{h}(\frac{h}{6}(x'(t) + 4x'(t + h/2) + x'(t+h)))$$

sau

$$\frac{x(t+h) - x(t)}{h} \simeq \frac{1}{6}\left(x'(t) + 2 \cdot \underbrace{x'(t + h/2)}_{A} + 2 \cdot \underbrace{x'(t + h/2)}_{B} + x'(t+h)\right) \tag{4.2}$$

Fie $x'(t) = u(t, x(t))$ si fie

$$v_1(t, y) = u(t, y) \tag{4.3}$$

deci

$$x'(t) = v_1(t, x(t)) \tag{4.4}$$

Luam separat termenii A si B.
Pentru A avem:

$$x'(t + h/2) = u(t + h/2, x(t + h/2)) \tag{4.5}$$

Din definitia derivatei avem

$$\frac{x(t + h/2) - x(t)}{h/2} = x'(c) \;\; c \in [t, t + h/2]$$

Vom aproxima $x'(t) \simeq x'(t)$ si cu acesta relatia (4.5) devine:

$$x'(t + \frac{h}{2}) \simeq u(t + \frac{h}{2}, x(t) + \frac{h}{2}x'(t))$$

aplicind (4.4) avem

$$x'(t + \frac{h}{2}) \simeq u(t + \frac{h}{2}, x(t) + \frac{h}{2}v_1(t, x(t)))$$

Fie

$$v_2(t, y) = u(t + \frac{h}{2}, y + \frac{h}{2}v_1(t, y)) \tag{4.6}$$

Cu aceasta notatie avem

$$x'(t + \frac{h}{2}) \simeq v_2(t, x(t)) \tag{4.7}$$

Pentru B avem:

$$x'(t + h/2) = u(t + h/2, x(t + h/2)) \tag{4.8}$$

Din definitia derivatei avem

$$\frac{x(t + h/2) - x(t)}{h/2} = x'(c) \;\; c \in [t, t + h/2]$$

Vom aproxima $x'(t) \simeq x'(t + h/2)$ si cu aceasta relatia (4.8) devine:

$$x'(t + \frac{h}{2}) \simeq u(t + \frac{h}{2}, x(t) + \frac{h}{2}x'(x + \frac{h}{2}))$$

inlocuind relatia (4.7) in relatia anterioara avem

$$x'(t + \frac{h}{2}) \simeq u(t + \frac{h}{2}, x(t) + \frac{h}{2}v_2(t, x(t)))$$

Fie

$$v_3(t, y) = u(t + \frac{h}{2}, y + \frac{h}{2}v_2(t, y)) \tag{4.9}$$

Cu aceasta notatie avem

$$x'(t+\frac{h}{2}) \simeq v_3(t,x(t)) \tag{4.10}$$

In continuare vom aproxima cel de-al treilea termen

$$x'(t+h) = u(t+h, x(t+h))$$

Din definitia derivatei avem

$$\frac{x(t+h/2)-x(t)}{h/2} = x'(c) \;\; c \in [t, t+h/2]$$

Vom aproxima $x'(c) \simeq x'(t+h/2)$ si vom avea

$$x'(t+h) \simeq u(t+h, x(t)+h \cdot x'(t+h/2))$$

in care daca introducem relatia (4.10) devine

$$x'(t+h) \simeq u(t+h, x(t)+h \cdot v_3(t,x(t)))$$

Fie

$$v_4(t,y) = u(t+h, y+h \cdot v_3(t,y)) \tag{4.11}$$

cu aceasta notatie avem

$$x'(t+h) \simeq v_4(t,x(t)) \tag{4.12}$$

Cu aceste notatii relatia (4.2) devine

$$\frac{x(t+h)-x(t)}{h} \simeq \frac{1}{6}(v_1(t,x(t)) + 2 \cdot v_2(t,x(t)) + 2 \cdot v_3(t,x(t)) + v_4(t,x(t)))$$

Fie ecuatia cu diferente finite care aproximeaza solutia problemei Couchy (4.1)

$$\begin{aligned} x_0 &= \alpha_0 \\ \frac{x_{j+1}-x_j}{h} &= \frac{1}{6}(v_1(t_j,x_j) + 2v_2(t_j,x_j) + 2v_3(t_j,x_j) + v_4(t_j,x_j)) \;\; j = \overline{0, n_h - 1} \end{aligned}$$

unde:

$$\begin{aligned} v_1(t,y) &= u(t,y) \\ v_2(t,y) &= u(t+\frac{h}{2}, y+\frac{h}{2}v_1(t,y)) \\ v_3(t,y) &= u(t+\frac{h}{2}, y+\frac{h}{2}v_2(t,y)) \\ v_4(t,y) &= u(t+h, y+h \cdot v_3(t,y)) \end{aligned}$$

Fie $z : \delta_h \to R$ definita astfel $z(t_j) = x_j$ cu $x_j$ solutii ale ecuatiei (4.1).

**Definitie 17** *Aproximarea $x|_{\delta_h} \simeq z$ se numeste metoda Runge-Kutta.*

## 4.2 Consistenta

Fie

$$u_h(t,y) = \frac{1}{6}(v_1(t,y) + 2v_2(t,y) + 2v_3(t,y) + v_4(t,y))$$

unde:

$$\begin{aligned} v_1(t,y) &= u(t,y) \\ v_2(t,y) &= u(t+\frac{h}{2}, y+\frac{h}{2}v_1(t,y)) \\ v_3(t,y) &= u(t+\frac{h}{2}, y+\frac{h}{2}v_2(t,y)) \\ v_4(t,y) &= u(t+h, y+h\cdot v_3(t,y)) \end{aligned}$$

Explicitind in parte avem

$$\begin{aligned} u_h(t,x(t)) &= \frac{1}{6}(u(t,x(t)) + 2u(t+\frac{h}{2}, x(t)+\frac{h}{2}u(t,x(t))) + \\ &\quad 2u(t+\frac{h}{2}, x(t)+\frac{h}{2}v_2(t,x(t))) + u(t+h, x(t)+hv_3(t,x(t))) \end{aligned}$$

Efectuind diferenta

$$\begin{aligned} u_h(t,x(t)) - u(t,x(t)) &= \frac{1}{6}(-5u(t,x(t)) + 2u(t+\frac{h}{2}, x(t)+\frac{h}{2}u(t,x(t))) + \\ &\quad 2u(t+\frac{h}{2}, x(t)+\frac{h}{2}v_2(t,x(t))) + u(t+h, x(t)+hv_3(t,x(t))) \end{aligned} \tag{4.13}$$

Deoarece functia $x'(t) = u(t,x(t)) := u(t,y)$ este de fapt functia din conditia couchy si aplicind (2) si (3) ea este continua pe $\{(s, x(t)+\frac{h}{2}x'(t)) | t,s \in [a,b], x(t)+\frac{h}{2}x'(t) \in B\} = K_1$.

Se observa ca $K_1$ este marginita si inchisa deci este compacta.

Cum $u$ este continua pe $K_1$ atunci ea este uniform continua asadar avem din definitia uniform continuitatii: $\varepsilon_1 > 0, \exists \eta_\varepsilon > 0$ $a.i. |t-s| < \eta_{\varepsilon_1}$ $si$ $|t-w| < \eta_{\varepsilon_1}$ $cu$ $(t,y),(s,w) \in K_1$ atunci $|u(t,y) - u(s,w)| < \varepsilon_1$.

Deoarece functia $x'(t) = u(t,x(t)) := u(t,y)$ este de fapt functia din conditia couchy si aplicind (2) si (3) ea este continua pe $\{(s, x(t)+\frac{h}{2}v_2(t,x(t))) | t,s \in [a,b], x(t)+\frac{h}{2}v_2(t,x(t)) \in B\} = K_2$.

Cum $u$ este continua pe $K_2$ atunci ea este uniform continua asadar avem din definitia uniform continuitatii: $\varepsilon_2 > 0, \exists \eta_{\varepsilon_2} > 0$ $a.i. |t-s| < \eta_{\varepsilon_2}$ $si$ $|t-w| < \eta_{\varepsilon_2}$ $cu$ $(t,y),(s,w) \in K_2$ atunci $|u(t,y) - u(s,w)| < \varepsilon_2$.

Deoarece functia $x'(t) = u(t,x(t)) := u(t,y)$ este de fapt functia din conditia couchy si aplicind (2) si (3) ea este continua pe $\{(s, x(t)+\frac{h}{2}v_3(t,x(t))) | t,s \in [a,b], x(t)+\frac{h}{2}v_3(t,x(t)) \in B\} = K_3$.

Cum $u$ este continua pe $K_2$ atunci ea este uniform continua asadar avem din definitia uniform continuitatii: $\varepsilon_3 > 0, \exists \eta_{\varepsilon_3} > 0$ $a.i. |t-s| < \eta_{\varepsilon_3}$ $si$ $|t-w| < \eta_{\varepsilon_3}$ $cu$ $(t,y),(s,w) \in K_3$ atunci $|u(t,y) - u(s,w)| < \varepsilon_3$.

Deoarece $u_h(t,x(t))$ trebuie sa fie definita atunci ea este definita pe $K_1 \cap K_2 \cap K_3 \cap B = K$ deoarece este intersectie de multimi compacte multimea rezultata este tot o multime compacta asadar $K$ este o multime compacta.

Deoarece $K$ este o multime compacta iar functia $u(t,y)$ este continua pe $K$ atunci $u(t,y)$ este uniform continua pe $K$.

Relatia (4.13) se poate scrie astfel:

$$\begin{aligned} u_h(t,x(t)) - u(t,x(t)) &= \frac{1}{3}\left(u(t+\frac{h}{2}, x(t)+\frac{h}{2}u(t,x(t))) - u(t,x(t))\right) \\ &+\frac{1}{3}\left(u(t+\frac{h}{2}, x(t)+\frac{h}{2}v_2(t,x(t))) - u(t,x(t))\right) \\ &+\frac{1}{6}\left(u(t+h, x(t)+hv_3(t,x(t)) - u(t,x(t))\right) \end{aligned}$$

Aplicind uniform continuitatea pe multimile $K_1, K_2, K_3$ si daca $h < \min\{2 * \min\{\eta_1, \eta_2\}, \eta_3\}$ si

$$\begin{aligned} \frac{h}{2}\max_{t\in[a,b]}|x'(t)| &< \eta_1 \\ \frac{h}{2}\max_{t\in[a,b]}|v_2(t,x(t))| &< \eta_2 \\ h\max_{t\in[a,b]}|v_3(t,x(t))| &< \eta_3 \end{aligned}$$

avem

$$|u_h(t,x(t)) - u(t,x(t))| < \frac{1}{3}\varepsilon_1 + \frac{1}{3}\varepsilon_2 + \frac{1}{6}\varepsilon_3 = \varepsilon$$

Deci metoda Runge-Kutta este consistenta.

## 4.3 Prezentarea implementarii si exemple

In continuare vom lua urmatorul exemplul:

$$\begin{aligned} x_0 &= 0 \\ x'(t) &= t + x(t) \;\; t \in [0,1] \end{aligned}$$

Sa se rezolve aproximativ ecuatia cu metoda lui Runge-Kutta in cazul in care avem urmatoarele diviziuni $\delta_1$pentru care $h = 1/2$ si $\delta_2$ pentru care $h = 1/4$.

### 4.3.1 Implementare

```
#include"parser_func.cpp"
    #include<math.h>
    #include<stdlib.h>
    #include<stdio.h>
    int main(int argc, char* argv[])
    {
        parser_func *parser;
        double *delta,a,b,h;
```

```
    char *dfunc;
    long n;
    double *x;
    double v1,v2,v3,v4;
    long i;
    parser=new parser_func;
    dfunc=(char *)calloc(100,sizeof(char));
    printf("Introduceti pasul h=");
    scanf("%lf",&h);
    printf("Introduceti limita inferioara a intervalului\n");
    scanf("%lf",&a);
    printf("Introduceti limita superioara a intervalului \n");
    scanf("%lf",&b);
    n=(long)((b-a)/h)+1;
    x=(double *)calloc(n,sizeof(double));
    delta=(double *)calloc(n,sizeof(double));
    delta[0]=a;
    for(i=1;i<n;i++)
        delta[i]+=i*h;
    printf("Introduceti conditia initiala\n");
    scanf("%lf",&x[0]);
    printf("Introduceti functia cu t si y\n");
    fflush(stdin);
    dfunc=gets(dfunc);
    parser->set_function(dfunc);
    for(i=0;i<n-1;i++)
    {
        parser->set_var('t',delta[i]);
        parser->set_var('y',x[i]);
        parser->eval_func(&v1);
        parser->set_var('t',delta[i]+h/2);
        parser->set_var('y',x[i]+h*v1/2);
        parser->eval_func(&v2);
        parser->set_var('y',x[i]+h*v2/2);
        parser->eval_func(&v3);
        parser->set_var('t',delta[i+1]);
        parser->set_var('y',x[i]+h*v3);
        parser->eval_func(&v4);
        x[i+1]=x[i]+h*(v1+2*v2+2*v3+v4)/6;
    }
    printf("Valorile in diviziune\n");
    for(i=0;i<n;i++)
        printf("%lf ",x[i]);
    printf("\n");
    return 0;
}
```

Clasa parser_func va fi prezentata in Anexa 1 si contine parserul de functii.

Functia set_var(caracter,valoare) seteaza variabila caracter (care trebuie sa fie un caracter majuscul sau minuscul intre a-z deoarece nu tine cont de litere mari sau mici) cu valoarea data in valoare.

Functia eval_func evalueaza functia setata prin set_function cu variabilele setata anterior.

### 4.3.2 Rezultatele rularii

Pentru $\delta_1 = \{0, 1/2, 1\}$ adica pentru pasul $h = 0.5$ si intervalul $[0, 1.0]$ precum si functia data $t + y$ avem urmatoarele rezultate

| | | | |
|---|---|---|---|
| $t$ | 0 | 0.5 | 1.0 |
| $x(Runge - Kutta)$ | 0 | 0.1484 | 0.71734 |
| $x(Teoretic)$ | 0 | 0.1487 | 0.71828 |
| $\|x(Teoretic) - x(Runge - Kutta)\|$ | 0 | 0.0003 | 0.00094 |

Pentru $\delta_2 = \{0, 1/4, 1/2, 3/4, 1\}$ adica pentru pasul $h = 0.25$ si intervalul $[0, 1.0]$ precum si functia data $t + y$ avem urmatoarele rezultate

| | | | | | |
|---|---|---|---|---|---|
| $t$ | 0 | 0.25 | 0.5 | 0.75 | 1.0 |
| $x(Runge - Kutta)$ | 0 | 0.034 | 0.14869 | 0.3669 | 0.71821 |
| $x(Teoretic)$ | 0 | 0.034 | 0.1487 | 0.367 | 0.71828 |
| $\|x(Teoretic) - x(Runge - Kutta)\|$ | 0 | 0 | 0.00001 | 0.0001 | 0.00007 |

Se observa cu cit diviziunea are mai multe puncte cu atit mai bine este aproximata solutia in punctul final.

# Chapter 5

# Anexa 1 (parser de functii)

In aceasta anexa vom prezenta parserul de functii. Atesta este o versiune modifica a parserului prezentat in [2].

## 5.1 Definierea clasei

```
// parser_func.h: interface for the parser_func class.
    //
    //////////////////////////////////////////////////////////////////////
    #if _MSC_VER > 1000
    #pragma once
    #endif // _MSC_VER > 1000
    #include <stdlib.h>
    #include<string.h>
    #include<ctype.h>
    #include<math.h>
    #define DELIMITATOR 1
    #define VARIABILA 2
    #define NUMAR 3
    #define FUNCTION 4
    class parser_func
    {
    public:
        double get_var(char var);
        void unset_var(char var);
        void set_var(char var,double val);
        void empty_vars();
        parser_func();
        virtual ~parser_func();
        void set_function(char *func);
        void eval_func(double *rez);
    protected:
```

```
        char * get_argument(char *work);
        bool is_function;
        double eval_math(char *s);
        double find_var(char *s);
        long isdelim(char c);
        void serror(long error);
        void putback();
        void get_token(void);
        void atom(double *rez);
        void eval_func6(double *rez);
        void eval_func5(double *rez);
        void eval_func4(double *rez);
        void eval_func3(double *rez);
        void eval_func2(double *rez);
        void eval_func1(double *rez);
        char simb[80];
        char tip_simb;
        double *vars;
        char *prog;
        char *function;
        double (*math_f[13])(double arg);
    };
```

## 5.2 Implementarea clasei

```
// parser_func.cpp: implementation of the parser_func class.
    //
    //////////////////////////////////////////////////////////////////////
    #include "stdafx.h"
    #include "parser_func.h"
    #include <iostream.h>
    #include<stdio.h>
    //////////////////////////////////////////////////////////////////////
    // Construction/Destruction
    //////////////////////////////////////////////////////////////////////
    parser_func::parser_func()
    {
        vars=(double *)calloc(26,sizeof(double));
        prog=NULL;
        function=NULL;
        math_f[0]=cos;
        math_f[1]=acos;
        math_f[2]=sin;
        math_f[3]=asin;
        math_f[4]=tan;
        math_f[5]=atan;
        math_f[6]=cosh;
```

```
        math_f[7]=sinh;
        math_f[8]=tanh;
        math_f[9]=exp;
        math_f[10]=log;
        math_f[11]=log10;
        math_f[12]=fabs;
        is_function=false;
    }
    parser_func::~parser_func()
    {
        if(vars!=NULL) free(vars);
        if(prog!=NULL) free(prog);
    }
    void parser_func::eval_func(double *rez)
    //input point
    {
        prog=function;
        get_token();
        if(!*simb)
        {
            serror(2);
            return;
        }
        eval_func1(rez);
        if(*simb) serror(0); //last simbol is null
    }
    void parser_func::eval_func1(double *rez)
    //processing a atribution
    {
        long fanta;
        char temp_simb[80],sseg_tip;

        if(tip_simb==VARIABILA)
        {
            //save the old simbol;
            strcpy(temp_simb,simb);
            sseg_tip=tip_simb;
            //computin the index of the variable
            fanta=toupper(*simb)-'A';
            get_token();
            if(*simb!='=')
            {
                putback();//return the curent segment
                //restore the old simbol without atributs
                strcpy(simb,temp_simb);
                tip_simb=sseg_tip;
            }
            else
```

```
            {
                get_token(); //get the next part of the expresion
                eval_func2(rez);
                vars[fanta]=*rez;
                return;
            }
        }
        eval_func2(rez);
    }
    void parser_func::eval_func2(double *rez)
    //add or difference of two elements
    {
        char op;
        double temp;

        eval_func3(rez);
        op=*simb;
        while(op=='+' || op=='-')
        {
            get_token();
            eval_func3(&temp);
            switch(op)
            {
            case '-':
                *rez=*rez-temp;
                break;
            case '+':
                *rez=*rez+temp;
                break;
            }
            op=*simb;
        }
    }
    void parser_func::eval_func3(double *rez)
    //product or division of two elements
    {
        char op;
        double temp;

        eval_func4(rez);
        while((op=*simb)=='*' || op=='/' || op=='%')
        {
            get_token();
            eval_func4(&temp);
            switch(op)
            {
            case '*':
                *rez=(*rez)*temp;
```

```
                break;
            case '/':
                *rez=(*rez)/temp;
                break;
            case '%':
                *rez=(long)*rez%(long)temp;
                break;
            }
        }
}
void parser_func::eval_func4(double *rez)
//process an exponent
{
        double temp,ex;
        eval_func5(rez);
        if(*simb=='^')
        {
            get_token();
            eval_func4(&temp);
            *rez=pow(*rez,temp);
        }
}
void parser_func::eval_func5(double *rez)
//eval + or - unary
{
        char op;
        op=0;
        if((tip_simb==DELIMITATOR) && *simb=='+' || *simb=='-')
        {
            op=*simb;
            get_token();
        }
        eval_func6(rez);
        if(op=='-') *rez=-(*rez);
}

void parser_func::eval_func6(double *rez)
//process parantheses expresion
{
        if(*simb=='(')
        {
            get_token();
            eval_func2(rez);
            if(*simb!=')')
                serror(1);
            get_token();
        }
        else atom(rez);
```

```
}
double parser_func::eval_math(char *s)
//evaluate a mathematical predefinite function
{
    double rez;
    int findex;
    if(strcmp(s,"cos")==0) //we have cos
        findex=0;
    else if(strcmp(s,"acos")==0) //we have arccos
        findex=1;
    else if(strcmp(s,"sin")==0) //we have sin
        findex=2;
    else if(strcmp(s,"asin")==0) //we have arcsin
        findex=3;
    else if(strcmp(s,"tan")==0) //we have tangent
        findex=4;
    else if(strcmp(s,"atan")==0) //we have arctangent
        findex=5;
    else if(strcmp(s,"cosh")==0) //we have hiperbolic cosinus
        findex=6;
    else if(strcmp(s,"sinh")==0) //we have hyperbolic sinus
        findex=7;
    else if(strcmp(s,"tanh")==0) //we have hyperbolic tangent
        findex=8;
    else if(strcmp(s,"exp")==0) //we have exponential
        findex=9;
    else if(strcmp(s,"log")==0) //we have natural logarithm
        findex=10;
    else if(strcmp(s,"log10")==0) //we have 10 base logarithm
        findex=11;
    else if(strcmp(s,"fabs")==0) //we have absolute value
        findex=12;
    else findex=-1;
    char *argument;
    char *work;
    char old_tip_simb;
    old_tip_simb=tip_simb;
    work=(char *)calloc(80,sizeof(char));
    argument=get_argument(work);
    eval_func(&rez);
    free(work);
    prog=argument;
    if(findex==-1)
        return 0.0;
    else return math_f[findex](rez);
}
void parser_func::atom(double *rez)
//read the value of the number or a variable
```

```
{
    switch(tip_simb)
    {
    case VARIABILA:
        *rez=find_var(simb);
        get_token();
        return;
    case NUMAR:
        *rez=atof(simb);
        get_token();
        return;
    case FUNCTION:
        *rez=eval_math(simb);
        get_token();
        return;
    default:
        serror(0);
    }
}
void parser_func::get_token()
//return the next simbol
{
    char *temp;
    tip_simb=0;
    temp=simb;
    *temp='\0';
    if(!prog) return; //end of expresion
    while(isspace(*prog)) ++prog;
    if(strchr("+-*/%^=()",*prog))
    {
        tip_simb=DELIMITATOR;
        *temp++=*prog++;
    }
    else if(isalpha(*prog) && !isalpha(*(prog+1)))
    {
        while(!isdelim(*prog)) *temp++=*prog++;
        tip_simb=VARIABILA;
    }
    else if(isalpha(*prog) && isalpha(*(prog+1)))
    {
        while(!isdelim(*prog)) *temp++=*prog++;
        tip_simb=FUNCTION;
    }
    else if(isdigit(*prog))
    {
        while(!isdelim(*prog)) *temp++=*prog++;
        tip_simb=NUMAR;
    }
```

```
    *temp='\0';
}
void parser_func::putback()
//return one simbol from the input flux
{
    char *s;
    s=simb;
    for(;*s;s++) prog--;
}
void parser_func::serror(long error)
//print a sintax error
{
    static char *e[]={
        "Sintax error",
        "Unbalance bracket",
        "Not an expresion"
    };
    cout<<e[error]<<endl;
}
double parser_func::find_var(char *s)
//return the value of a variable
{
    if(!isalpha(*s))
    {
        serror(1);
        return 0.0;
    }
    return vars[toupper(*simb)-'A'];
}
long parser_func::isdelim(char c)
//return 0 is c is a delimitator
{
    if(strchr(" +-/*%^=()",c) || c==9 || c=='\r' || c==0)
        return 1;
    return 0;
}
void parser_func::empty_vars()
//empty the variable memory
{
    if(vars!=NULL) free(vars);
    vars=(double *)calloc(26,sizeof(double));
}
void parser_func::set_var(char var, double val)
//set a variable to a value
{
    if(!isalpha(var))
    {
        serror(1);
```

```
            return;
        }
        vars[toupper(var)-'A']=val;
    }
    void parser_func::unset_var(char var)
    //unset a variable
    {
        if(!isalpha(var))
        {
            serror(1);
            return;
        }
        vars[toupper(var)-'A']=0.0;
    }
    double parser_func::get_var(char var)
    //get a variable
    {
        if(!isalpha(var))
        {
            serror(1);
            return 0.0;
        }
        return vars[toupper(var)-'A'];
    }
    void parser_func::set_function(char *func)
    //set the function to the system
    {
        if(function!=NULL) free(function);
        function=(char *)calloc(strlen(func)+1,sizeof(char));
        strcpy(function,func);
        prog=function;
    }
    char * parser_func::get_argument(char *work)
    //get the argument of a predefined function
    {
        char *temp;
        prog++;
        long i=0;
        long parantheses=1;
        while(parantheses!=0)
        {
            if(*prog=='(') parantheses++;
            if(*prog==')') parantheses--;
            if(parantheses==0) break;
            work[i]=*prog;
            i++;
            prog++;
        }
```

```
        prog++;
        temp=prog;
        prog=work;
        return temp;
    }
```

# Bibliography

[1] Rosca,I., "Lectii de Ecuatii fiderentiale si cu derivate partiale",Editura Fundatiei "Romania de Maine",2000,ISBN 973-582-190-7

[2] Schildt, H.,"C manual complet",Editura "Teora",1998,ISBN973-601-471-1

[3] Grigore,G., Note de curs Analiza Matematica I,Universitatea Spiru Haret, Facultatea de Matematica-Informatica.

[4] Grigore,G., Note de curs Analiza Numerica II,Universitatea Spiru Haret, Facultatea de Matematica-Informatica.