Lucrarea de laborator nr. 14

I. Scopul lucrării

Integrarea numerică a ecuațiilor diferențiale şi a sistemelor de ecuații diferențiale ordinare

II. Conținutul lucrării

1. Generalități.
2. Comenzi MAPLE pentru rezolvarea ecuațiilor diferențiale şi a sistemelor de ecuații diferențiale ordinare
3. Metoda Euler. Algoritmul predictor-corector al metodei Euler perfecționate
4. Metoda Runge-Kutta
5. Metoda predictor-corector cu paşi legați a lui Adams
6. Alegerea metodei numerice de rezolvarea a ecuațiilor diferențiale

III. Prezentarea lucrării

III.1. Generalități

O ecuație de forma

$$F(x, y, y') = 0, (x, y, y') \in D \subset \mathbf{R}^3$$

unde $y = y'(x)$ este o funcție necunoscută, se numeşte ecuație diferențială de ordinul întâi. Funcția $y = \varphi(x)$, definită şi derivabilă pe intervalul I, care satisface $F(x, \varphi/x), \varphi'(x)) =$, $x \in I$ se numeşte soluție a ecuației diferențiale. Graficul funcției $y = \varphi(x)$ se numeşte curbă integrală. Soluția generală a ecuației diferențiale poate fi reprezentată sub una din formele:

$y = g(x, C)$ (explicită)

$h(x, y, C) = 0$ (implicită)

$x = h_1(t, C), y = h_2(t, C)$ (parametrică)

unde C este o constantă reală. Problema găsirii soluției $y = y(x)$ a ecuației diferențiale care să verifice condiția inițială:

$$y(x_0) = y_0$$

poartă numele problema Cauchy sau problema cu condiții inițiale. O ecuație diferențială de ordinul întâi rezolvată în raport cu y' are forma

$$y' = f(x,y),\ (x, y) \in D \subset \mathbf{R}^2.$$

Vom considera în continuare ecuații de acest tip.

După cum se ştie găsirea soluției exacte a unei ecuații diferențiale nu este posibilă decât în cazuri cu totul particulare. De aceea suntem nevoiți să apelăm la metode de determinare aproximativă a soluțiilor problemei Cauchy. Metodele aproximative sunt de două tipuri:

1) metode analitice - care dau aproximarea soluției sub forma unor expresii analitice.
2) metode numerice – în cadrul cărora soluția se obține sub forma unui şir de valori, plecând de la o valoare inițială a soluției.

Metodele numerice presupun găsirea unui număr de puncte $y_1, y_2, \ldots, y_n$ care aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_n)$ ale curbei integrale care trece prin punctul inițial (x_0, y_0). Să considerăm pasul de integrare:

$$h = x_{i+1} - x_i,\ i = 0, 1, \ldots, n-1.$$

Putem clasifica metodele numerice de rezolvarea a ecuațiilor diferențiale în:

(1) metode cu paşi separați: necesită pentru determinarea lui y_{i+1} cunoaşterea punctului anterior (x_i, y_i) şi a pasului h Este utilizată dezvoltarea în serie Taylor a funcției $y = y(x)$ în jurul lui x_i

$$y(x_{i+1}) = y(x_i) + h\, y'(x_i) + \frac{h^2}{2!}\, y''(x_i) + \ldots$$

(2) metode cu paşi legați: calculul lui y_{i+1} necesită cunoaşterea pasului h şi a mai multor puncte anterioare $(x_i, y_i), (x_{i-1}, y_{i-1}), \ldots, (x_{i-j}, y_{i-j})$. Valorile $y_j, y_{j-1}, \ldots, y_1$ se determină folosind metode numerice cu paşi separați. În cazul acestor metode se utilizează

$$y(x_{i+1}) - y(x_{i-j}) = \int_{x_{i-j}}^{x_{i+1}} y'(x)dx$$

unde integrantul $y' = f(x, y)$ se aproximează cu un polinom de interpolare.

Ambele tipuri de metode (cu paşi separați sau cu paşi legați) pot folosi pentru generarea valorilor aproximative $y_1, y_2, \ldots, y_n$ algoritmi de tip explicit sau implicit. Pentru o metodă cu paşi separați un algoritm de tip explicit este de forma:

$$y_{i+1} = y_i + h\rho(x_i, y_i, h),\ i = 0, 1, \ldots, n-1$$

iar un algoritm implicit

$$y_{i+1}^c = y_i + h\rho(x_i, y_i, x_{i+1}, y_{i+1}^p, h),\ i = 0, 1, \ldots, n-1$$

Aproximația y_{i+1}^p care apare în partea dreaptă se calculează cu un algoritm explicit şi poartă denumirea de valoare prezisă (de predicție), iar y_{i+1}^c poartă

denumirea de valoare corectată a lui y_{i+1}. În mod asemănător se construiesc algoritmi expliciți sau impliciți pentru metodele cu pași legați.

Considerând $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$ considerațiile anterioare rămân valabile pentru sisteme de m ecuații diferențiale.

În metodele numerice pe care le vom întâlni, facem presupunerea că sunt satisfăcute condițiile teoremei de existentă şi unicitate a soluției ecuației sau sistemului de ecuații diferențiale.

III.2. Comenzi MAPLE pentru rezolvarea ecuațiilor diferențiale şi a sistemelor de ecuații diferențiale ordinare

Comanda **dsolve** rezolvă ecuații diferențiale. Comanda are forma:

>dsolve({ecuatii_diferențiale},{variabile}, opțiuni)

>dsolve({ecuatii_diferențiale,conditii_initiale}, {variabile}, opțiuni)

{variabile} reprezintă funcțiile necunoscute din sistemul de ecuații {ecuatii_diferențile}. Dacă se rezolvă o ecuație, atunci prezența acoladelor nu este obligatorie. Opțiunile sunt de forma cuvânt cheie = valoare. Prezența lor nu este obligatorie. Dacă opțiunea type = exact este prezentă se încearcă determinarea unei soluții exacte. Aceasta este opțiunea implicită. Opțiunea type=series determină aplicarea metodei seriilor de puteri. Ordinul formulei utilizate este controlat de variabila globală Order. Opțiunea type = numeric are ca efect aplicarea unei metode numerice de rezolvare a ecuației sau sistemului de ecuații. În cazul în care se utilizează această opțiune trebuie să se specifice condiții inițiale (să avem o problemă Cauchy). Se poate alege metoda numerică folosind opțiunea method. Dacă se utilizează method=rkf45, atunci metoda utilizată este Runge-Kutta de ordinul 4-5 (metoda Fehlberg), iar dacă se utilizează method=dverk78 metoda este Runge-Kutta de ordin 7-8. Opțiunea method =classical determină folosirea unei metode simple, implicit metoda Euler, dar se poate specifica şi o altă metodă după cum urmează:

method =classical[foreuler] – metoda Euler (înainte)
method =classical[heunform] – metoda Euler perfecționată
method =classical[rk2] – metoda Runge-Kutta de ordinul 2
method =classical[rk3] – metoda Runge-Kutta de ordinul 3
method =classical[rk4] – metoda Runge-Kutta de ordinul 4
method =classical[adambash] – metoda Adams-Bashford
method =classical[abmoulton] – metoda Adams-Bashford-Moulton

Implicit în cazul opțiunii numeric se utilizează metoda Runge-Kutta de ordinul 4-5. Opțiunea method = Laplace determină rezolvarea ecuației sau sistemului de ecuații utilizând transformarea Laplace.

Soluția ecuației este returnată sub formă implicită sau parametrică. Dacă se utilizează opțiunea explicit = true, atunci returnată soluția explicită (dacă este posibil). Valoarea implicită este explicit = false.

Pentru scrierea ecuațiilor diferențiale se utilizează diff sau D: $y^{(k)}(x)$ se poate scrie ca diff(y(x), $k) sau (D@@k)(y)(x). Scrierea În MAPLE a unei condiții inițiale de forma $y^{(k)}(x0) = y0k$ este (D@@k)(y)(x0) = y0k.

Pentru reprezentarea grafică a soluțiilor aproximative obținute în urma aplicării comenzii dsolve cu opțiunea numeric se poate utiliza comanda **odeplot**. Această comandă face parte din pachetul plots. Forma comenzii este

>odeplot(s,variabile, domeniu, opțiuni)

unde s reprezintă ieşirea unei comenzi dsolve cu opțiunea numeric, prin variabile se specifică ordinea coordonatele utilizate, domeniu este de forma a..b, iar opțiunile sunt aceleaşi ca în cazul comenzii plot sau plot3D. Prezența opțiunilor nu este obligatorie, iar dacă nu se face specificarea domeniului implicit se consideră –10..10.

Pachetul DEtools este destinat rezolvării ecuațiilor diferențiale şi ecuațiilor cu derivate parțiale. Prezentăm comenzile Dchangevar şi DEplot din acest pachet. Comanda **Dchangevar** are formele

>Dchangevar(transf, ecdif, varindep, nvarindep);

>Dchangevar(transf1, transf2,…, transfN, ecdif, varindep, nvarindep);

unde:

transf = listă sau mulțime de transformări care se substituie în ecuația sau ecuațiile diferențiale date de parametru ecdif

transf1, transf2,…, transfN = transformări de substituit în ecuațiile diferențiale

ecdif = ecuație, listă sau mulțime de ecuații diferențiale

varindep = numele variabilei independente curente

nvarindep = numele noii variabile independente

Parametru nvarindep este opțional. Transformările pot schimba

- doar variabila dependentă
- doar variabila independentă
- şi variabila dependentă şi variabila independentă.

Pentru reprezentarea grafică a soluției unei ecuații diferențiale (de grad m ≥1) sau a unui sistem de două ecuații diferențiale ordinare:

$$\begin{cases} x = f_1(t, x, y) \\ y = f_2(t, x, y) \end{cases}$$

se poate utiliza comanda DEplot. Formele acestei comenzi sunt

>DEplot(ecdif, var, tdom, opțiuni);

>DEplot(ecdif, var, tdom, init, opțiuni);

>DEplot(ecdif, var, tdom, ydom, xdom, opțiuni);

>DEplot(ecdif, var, tdom, init, xdom, ydom, opțiuni);

Prezența opțiunilor nu este obligatorie. Semnificația parametrilor este următoarea:

ecdif = listă sau mulțime de ecuații de ordinul întâi sau o ecuație de orice ordin

var = variabilă dependentă, listă sau mulțime de variabile independente

tdom = domeniul variabilei independente; se specifică sub forma t = a..b, sau a..b

ydom = domeniul primei variabile dependente; se specifică sub forma y(t) = y1..y2, sau y1..y2.

xdom = domeniul celei de-a doua variabile dependente; se specifică sub forma x(t) = x1..x2, sau x1..x2.

init = condițiile inițiale pentru soluțiile ce urmează a fi reprezentate; se specifică sub forma [[x(t0) = x0, y(t0) = y0] , [x(t1) = x1, y(t1) = y1], …]

Metoda de integrare este o metodă numerică, implicit utilizându-se metoda Runge-Kutta de ordinul 4 (method = classical[rk4]). Se poate utiliza opțiunea method pentru specificarea altei metode.

Dacă sistemul este autonom (f_1 şi f_2 nu depind de t), atunci se poate desena câmpul de vectori – rețea de vectori tangenți la curbele ce reprezintă soluțiile. Vectorii sunt reprezentați grafic prin săgeți controlate de opțiunea arrows. Pentru fiecare punct al (x, y) al reprezentării, săgeata este centrată în punct şi are panta $\frac{dy}{dx}$. Panta se calculează cu formula $\frac{dy}{dx} = \frac{dy}{dt} \Big/ \frac{dx}{dt}$. Dacă nu sunt indicate condițiile inițiale atunci se desenează doar săgețile, dacă este posibil, iar dacă nu, nu se desenează nimic.

Opțiunile sunt de forma cuvânt cheie = valoare. Exemple de opțiuni:

arrows = tip, unde tip poate lua una din valorile SMALL, MEDIUM, LARGE, LINE, NONE.

dirgrid = [întreg, întreg]; specifică numărul de puncte pe orizontal şi vertical pentru generarea săgeților. Minim se poate lua dirgrid = [2, 2], iar implicit dirgrid = [20, 20].

iterations = întreg; reprezintă numărul de paşi care se execută (numărul de puncte ale discretizării de pas indicat de stepsize)

stepsize = real; reprezintă distanța dintre două puncte ale discretizării. Implicit valoarea este (b-a)/20. Dacă valoarea indicată este prea mare, atunci ea este înlocuită cu valoarea implicită.

obsrange = TRUE sau FALSE; indică oprirea generării de puncte dacă s-a depăşit domeniul variabilei dependente sau nu. Implicit obsrange = TRUE.

scene = [nume, nume]; determină ce se reprezintă grafic, astfel dacă scene =[x, y] se reprezintă $t \to (x(t), y(t))$ cu x(t) pe orizontală, iar dacă scene =[t, y] se reprezintă $t \to$ y(t) cu t pe orizontală.

Pentru reprezentări grafice tridimensionale se poate folosi comanda **DEplot3d**. Comenzile Dchangevar, DEplot şi DEplot3d fac parte din pachetul DEtools, deci înainte de a fi folosite trebuie încărcat pachetul printr-o comandă with(DEtools), sau apelul lor trebuie făcut sub forma with(DEtools, comanda).

Exemple.

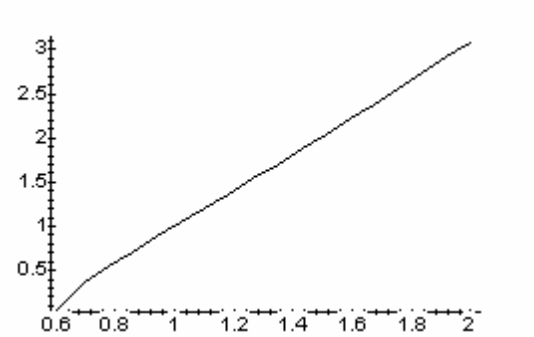
```
> dsolve(diff(y(x), x)-2*x*(1+y(x)^2)=0, y(x));
                arctan(y(x)) - x^2  = _C1
>   dsolve(diff(y(x),   x)-2*x*(1+y(x)^2)=0,   y(x),
explicit);
                   y(x) = tan(x^2  + _C1)
>dsolve(diff(y(x),x)-2*x*(1+y(x)^2)=0, y(x),series);
 y(x) = y(0) + (1+y(0)^2) x^2 + y(0)(1+y(0)^2)x^4  + O(x^6)
> Order:=8;
                        Order := 8
>dsolve(diff(y(x),x)-2*x*(1+y(x)^2)=0, y(x),series);
    y(x) = y(0) + (1+y(0)^2) x^2 + y(0)(1+y(0)^2)x^4  +
          (4/3 y(0)^2  + y(0)^4  + 1/3 1/3)x^6  + O(x^8)

>dsolve({diff(y(x),x)-2*x*(1+y(x)^2)=0,
y(0)=0},y(x));
                      y(x) = tan(x^2)
>dsolve({diff(y(x),x)-2*x*(1+y(x)^2)=0,
y(0)=0},y(x),series);
               y(x) = x^2  + 1/3 x^4  + O(x^6)
> Order:=6;
                        Order := 6
>dsolve({diff(y(x),x)-2*x*(1+y(x)^2)=0,
y(0)=0},y(x),series);
                       y(x) = x^2
>dsolve({diff(y(x),x)-2*x*(1+y(x)^2)=0,y(0)=0},
y(x), methods=laplace);
                      y(x) = tan(x^2)
> dsolve(diff(y(x),x)-(y(x)^2+x^2)/(x*y),y(x));
```

$$y(x)^2 = 2\,x^2\,\ln(x) + x^2\,\_C1$$

```
>dsolve(diff(y(x),x)-(y(x)^2+x^2)/(x*y),y(x),
explicit);
```

$$y(x) = -(2\ln(x) + \_C1)^{1/2}x,\quad y(x) = (2\ln(x) + \_C1)^{1/2}x$$

```
>with(plots);
>odeplot(dsolve({diff(y(x),x)-(y(x)^2+x^2)/(x*y),
y(1)=1},y(x),numeric),[x,y(x)],-2...2,color=black);
```




```
>sol:=dsolve({diff(y(x),x)-(y(x)^2+x^2)/(x*y),
y(1)=t},y(x));
```

$$sol := y(x) = -(2\ln(x)+t^2)^{1/2}x,\quad y(x) = -(2\ln(x)+t^2)x$$

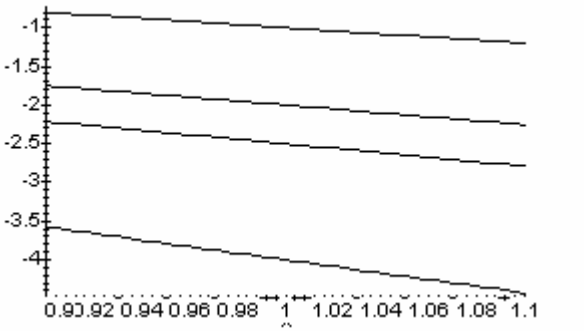
```
> sol[2];
```

$$y(x) = -(2\ln(x)+t^2)^{1/2}x\ x$$

```
>grafice:=seq(subs(t=i,rhs(sol[2])),i=[1,2,2.5,4]);
```

$$grafice := -(2\ln(x)+1)^{1/2}x,\ -(2\ln(x)+4)^{1/2}x,$$
$$-(2\ln(x)+6.25)x,\ -(2\ln(x)+16)^{1/2}x$$

```
> plot({grafice},x=0.9..1.1, color=black);
```




```
> eq:=diff(y(x),x)=cos(x)*y+sin(x);
```

$$eq := \frac{\partial}{\partial x}\, y(x) = \cos(x)\, y + \sin(x)$$

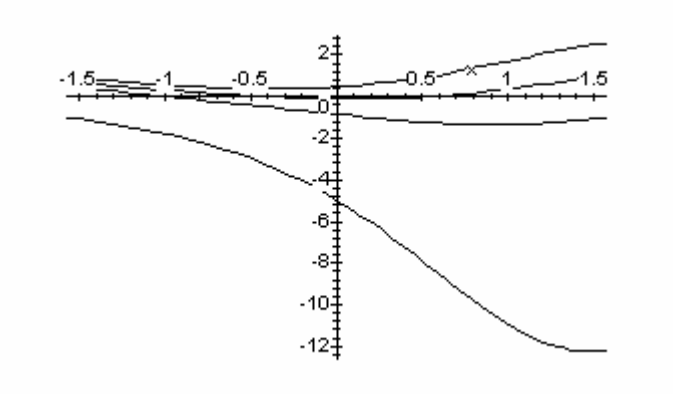
```
> sol1:=dsolve({eq,y(u)=v},y(x));
```

$$sol1 := y(x) = e^{\sin(x)} \int_u^x \frac{\sin(t)}{e^{\sin(t)}}\, dt + \frac{e^{\sin(x)}}{e^{\sin(u)}}\, v$$

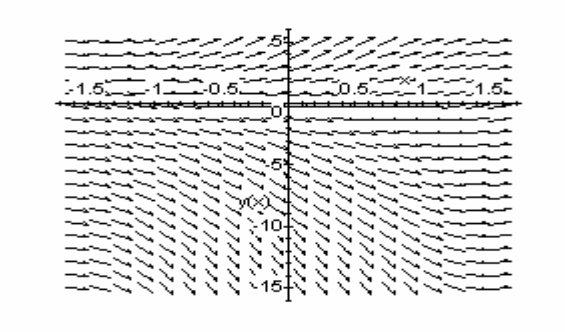
```
>   grafice2:=seq(seq(subs({v=j,u=i},rhs(sol1)),i=[-
Pi/2,Pi/2]),j=[-1,1]);
```

$$grafice2 := e^{\sin(x)} \int_{-\frac{1}{2}\pi}^x \frac{\sin(t)}{e^{\sin(t)}}\, dt - \frac{e^{\sin(x)}}{e^{\sin\left(-\frac{1}{2}\pi\right)}},\ e^{\sin(x)} \int_{\frac{1}{2}\pi}^x \frac{\sin(t)}{e^{\sin(t)}}\, dt - \frac{e^{\sin(x)}}{e^{\sin\left(\frac{1}{2}\pi\right)}},\ e^{\sin(x)} \int_{-\frac{1}{2}\pi}^x \frac{\sin(t)}{e^{\sin(t)}}\, dt + \frac{e^{\sin(x)}}{e^{\sin\left(-\frac{1}{2}\pi\right)}},\ e^{\sin(x)} \int_{\frac{1}{2}\pi}^x \frac{\sin(t)}{e^{\sin(t)}}\, dt - \frac{e^{\sin(x)}}{e^{\sin\left(\frac{1}{2}\pi\right)}}$$

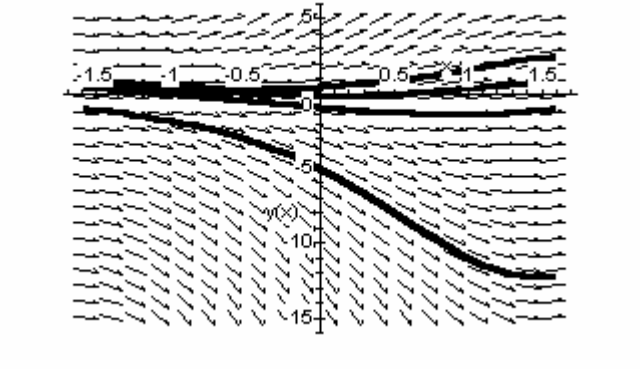
```
> plot({grafice2}, x=-Pi/2..Pi/2, color=black);
```




```
>with(DEtools);
>DEplot(eq,{y(x)},x=-Pi/2..Pi/2,y=-15..5,
color=black);
```




```
> DEplot(eq,{y(x)},x=-Pi/2..Pi/2,{[-Pi/2,-1],[-Pi/2,
1],[Pi/2,-1],[Pi/2,1]},y=-15..5,color=black,
linecolor=black);
```




```
> eqr:=diff(y(x),x)+y(x)^2*sin(x) =
2*sin(x)/cos(x)^2;
```

$$\mathtt{eqr} := \left(\frac{\partial}{\partial x} y(x)\right) + y^2(x)\sin(x) = 2\frac{\sin(x)}{\cos(x)^2}$$

```
>dsolve(eqr,y(x));
```

```
> eql:=Dchangevar(y(x)=1/z(x)+1/cos(x),eqr,x);
```

$$\mathtt{eql} := -\frac{\frac{\partial}{\partial x} z(x)}{z(x)^2} + \frac{\sin(x)}{\cos(x)^2} + \left(\frac{1}{z(x)} + \frac{1}{\cos(x)}\right)^2 \sin(x) = 2\frac{\sin(x)}{\cos(x)^2}$$

```
> sol3:=dsolve(eql,z(x));
```

$$\text{sol3} := z(x) = -\frac{1}{3}\frac{\cos(x)^3 - 3\_C1}{\cos(x)^2}$$

```
> c:=fsolve(subs(x=0,1/rhs(sol3)+1/cos(x))=-2, _C1);
```

$$c := 0$$

```
> yc:=unapply(1/rhs(sol3)+1/cos(x),x,_C1);
```

$$yc := (x, \_C1) \to -3\frac{\cos(x)^2}{\cos(x)^3 - 3\_C1} + \frac{1}{\cos(x)}$$

```
> y0:=unapply(subs(_C1=c,yc(x,_C1)),x);
```

$$y0 := x \to -\frac{2}{\cos(x)}$$

```
> eq3:=diff(y(x),x$2)-y(x)=1;
```

$$eq3 := \left(\frac{\partial^2}{\partial x^2} y(x)\right) - y(x) = 1$$

```
> dsolve(eq3,y(x));
```

$$y(x) = -1 + \_C1\, e^{x} + \_C2\, e^{-x}$$

```
> dsolve(eq3,y(x),output=basis);
```

$$[[e^{x}, e^{-x})], -1]$$

```
> dsolve({eq3,y(0)=-1,D(y)(0)=1},y(x));
```

$$y(x) = \frac{-e^{x} + \frac{1}{2}\left(e^{x}\right)^2 - \frac{1}{2}}{e^{x}}$$

```
> eq4:=diff(y(x),x$4)-y(x)=1;
```

$$eq4 := \left(\frac{\partial^4}{\partial x^4} y(x)\right) - y(x) = 1$$

```
> dsolve(eq4,y(x));
```

$$y(x) = -1 + \_C1\, e^{x} + \_C2\cos(x) + \_C3\sin(x) + \_C4\, e^{-x}$$

```
> dsolve({eq4,y(0)=-1,D(y)(0)=1,(D@@2)(y)(0)=-
2,(D@@3)(y)(0)=3},y(x));
```

$$y(x) = \frac{-e^{x} + \frac{1}{2}\left(e^{x}\right)^2 + \cos(x)e^{x} - \sin(x)e^{x} - \frac{3}{2}}{e^{x}}$$

III.3. Metoda Euler. Algoritmul predictor-corector al metodei Euler perfecționate

Fie problema Cauchy

$$\begin{cases} y' = f(x,y),\ x \in [a, b] \\ y(x_0) = y_0 \end{cases}$$

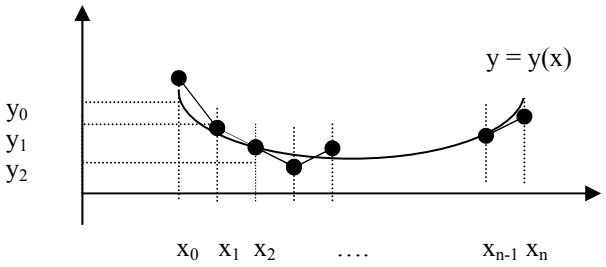
unde $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$. Metoda Euler constă în găsirea unui număr de puncte $y_1, y_2, \ldots, y_n$, care aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_n)$ ale soluției problemei Cauchy, după cum urmează :

$y_0 = y(x_0)$

$y_{i+1} = y_i + hf(x_i, y_i)$, $i = 0, 1, \ldots, n-1$

unde $h = \dfrac{b-a}{n}$ și $x_i = a + hi$, $i = 0,1, \ldots, n$.

Din punct de vedere geometric, în cazul m = 1, metoda constă în înlocuirea curbei integrale y = y(x) (soluția problemei Cauchy) cu linia poligonală construită cu ajutorul segmentelor: M_0M_1, M_1M_2, …, $M_{n-1}M_n$, unde $M_i (x_i, y_i)$, $i = 0,1, \ldots, n$.



Eroarea comisă pentru calculul unui y_{i+1} este compusă din eroarea de trunchiere (deoarece se trunchiază după primii doi termeni dezvoltarea în serie Taylor a funcției y = y(x) în jurul punctului y_i) și eroarea de rotunjire. Aceste erori se propagă de la o etapă de calcul a alta, mărimea erorii totale depinzând de numărul de puncte n. Metoda Euler are precizie destul de mică .

Pentru obținerea unei precizii satisfăcătoare h trebuie să fie foarte mic. Metoda Euler perfecționată este mai eficientă din acest punct de vedere. Metoda Euler perfecționată presupune un algoritm implicit. Formula

$$y_{i+1}^{p} = y_i + hf(x_i, y_i),\ i = 0, 1, \ldots, n-1$$

se numeşte formulă predictoare, iar formula

$$y_{i+1}^{c} = y_i + \frac{h}{2}\left(f(x_i, y_i) + f(x_i, y_{i+1}^{p})\right)$$

se numeşte formulă corectoare. Determinarea lui y_{i+1} se face iterativ:

- Se calculează y_{i+1}^{p} cu formula predictor.
- Pentru prima iterație se calculează y_{i+1}^{c} cu formula corector cu ajutorul lui y_{i+1}^{p} obținut din formula predictor, iar pentru următoarele iterații în locul lui y_{i+1}^{p} se ia y_{i+1}^{c} obținut în iterația precedentă. Astfel aplicarea formulei corector se repetă până se ajunge la precizia impusă $\varepsilon > 0$, adică până când modul (sau norma în cazul m – dimensional) a două valori y_{i+1}^{c} consecutive devin mai mică decât ε.

Algoritm

Date de intrare
funcția f
(x0, y0) – condiția inițială y(x0) = y0
h, n – pasul şi numărul de puncte
eps – precizia
Nmax – numărul maxim de iterații
Date de ieşie: y – tablou n+1 -dimensional

y_0	y_1	…	y_n

(y_0 = y0, y1, y_2, …, y_n, aproximează valorile adevărate $y(x_1)$, $y(x_2)$, …., $y(x_n)$ ale soluției problemei Cauchy, iar x_i = x0 + hi , i =0,1, …, n)

```
x00 : = x0;
y00 : = y0;
y[0] : = y00;
pentru i = 1,n,1 execută
        yp : = y0 + h*f(x00,y00);
        yc : = y0 + h*( f(x00,y00) + f(x00+h,yp))/2;
        k:=0;
        cât timp (||yc – yp|| ≥ eps) şi ( k ≤ Nmax) execută
                yp : = yc;
                yc : = y00 + h*( f(x00,y00) + f(x00+h,yp))/2;
                k : = k +1;
        dacă k > Nmax atunci
                scrie "nu converge în Nmax iteraţii"
                stop
        y[i] : = yc;
        y00 : =yc;
        x00 : = x00 + h
```

Proceduri MAPLE

Procedura `mEuler` întoarce un tablou n+1 dimensional ce conţine aproximaţiile valorilor soluţiei problemei Cauchy y' = f(x,y), y(x0) = y0, obţinute prin metoda Euler. Procedura `repgraf` reprezintă grafic în acelaşi sistem de coordonate soluţia adevărată a problemei Cauchy şi linia poligonală obţinută ca urmare a aplicării metodei Euler. Procedura `eroaremax` afişează valorile aproximate şi valorile adevărate, precum şi eroarea maximă în cazul aplicării metodei Euler (pentru o problemă Cauchy ce poate fi rezolvată prin metode exacte). Procedura `compara` afişează un tabel ce conţine valorile aproximative obţinute prin procedura creată de noi şi valorile aproximative obţinute prin aplicarea comenzii MAPLE dsolve cu opţiunea method=classical. Toate aceste proceduri au drept parametri de intrare funcţia f (se rezolva aproximativ ecuaţia y' = f(x,y)), x0, y0 ce dau condiţia iniţială (y(x0) = y0), pasul h, şi numărul de puncte n aproximate.

Procedura `mEulerP` este analogul procedurii `mEuler`, singura diferenţă fiind că se aplică metoda Euler perfecţionată. La fel procedura `eroaremaxp` (comparativ cu procedura `eroaremax`). Procedura `graficEP` reprezintă grafic soluţia unei probleme Cauchy folosind odeplot şi ieşirea unei comenzi dsolve cu opţiunea method=classical[heunform]. Pe

același grafic se reprezintă punctele obținute prin aplicarea procedurii mEulerP. Toate aceste proceduri au drept parametri de intrare funcția f (se rezolva aproximativ ecuația y' = f(x,y)), x0, y0 ce dau condiția inițială (y(x0) = y0), pasul h, numărul de puncte aproximate n, eroarea eps ce stabileşte precizia pentru valorile corectate, şi numărul maxim de corecții Nmax.

```
>mEuler := proc(f, x0, y0, h, n)
local yp, i;
    yp := array(0 .. n);
    yp[0] := y0;
    for i from 0 to n - 1 do
        yp[i+1] := yp[i]+evalf(f(x0+h*i, yp[i]))*h
    od;
    RETURN(evalm(yp))
end;
```

```
>repgraf := proc(f, x0, y0, h, n)
local i, yE, p1, p2, p3, sol;
    yE := mEuler(f, x0, y0, h, n);
    p1 := seq(
        line([x0 + h*i, yE[i]], [x0 + h*(i + 1),
yE[i + 1]]),
        i = 0 .. n - 1);
    sol := rhs(dsolve({y(x0) = y0, diff(y(x), x) =
f(x, y(x))},
        y(x), explicit));
    p2 := plot(sol, x = x0 .. x0 + n*h);
    p3 := p1, p2;
    display(p3)
end;
```

```
>eroaremax := proc(f, x0, y0, h, n)
local i, yE, y12, er, sol;
    yE := mEuler(f, x0, y0, h, n);
    y12 := matrix(2, n);
    for i to n do y12[1, i] := yE[i] od;
    sol := rhs(dsolve({y(x0) = y0, diff(y(x), x) =
f(x, y(x))},
        y(x), explicit));
```

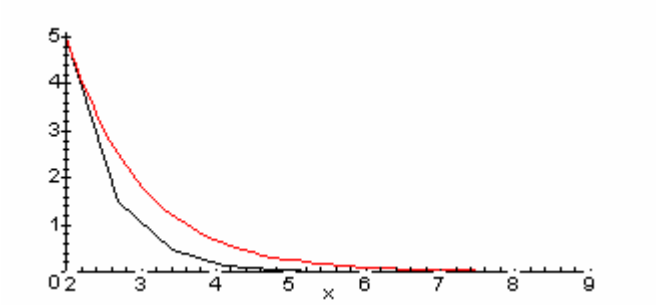
```
    for i to n do y12[2, i] := evalf(subs(x = x0 +
h*i, sol)) od;
    er := y12[2, 1] - y12[1, 1];
    for i from 2 to n do
        if abs(er) < abs(y12[2, i] - y12[1, i]) then
            er := y12[2, i] - y12[1, i]
        fi
    od;
    print(y12);
    print(`eroare maxima`, er)
end;


>compara := proc(f, x0, y0, h, n)
local i, yE, y12, er, sol;
    yE := mEuler(f, x0, y0, h, n);
    y12 := matrix(2, n);
    for i to n do y12[1, i] := yE[i] od;
    sol := dsolve({y(x0) = y0, diff(y(x), x) = f(x,
y(x))}, y(x),
        numeric, method = classical, start = x0,
stepsize = h);
    for i to n do y12[2, i] := rhs(sol(x0 + h*i)[2])
od;
    RETURN(evalm(y12))
end;


>mEulerP := proc(f, x0, y0, h, n, eps, Nmax)
local yp, yc, ypct, x00, y00, i, k;
    ypct := array(0 .. n);
    ypct[0] := y0;
    x00 := x0;
    y00 := y0;
    for i to n do
        yp := y00 + evalf(f(x00, y00))*h;
        yc := y00
             + 1/2*(evalf(f(x00, y00)) + evalf(f(x00
+ h,yp)))*h;
        k := 1;
        while eps <= abs(yc - yp) and k <= Nmax do
            yp := yc;
            yc := y00
```

```
                + 1/2*
               (evalf(f(x00, y00)) + evalf(f(x00 +
h, yp)))*h;
            k := k + 1
        od;
        if Nmax < k then print(
            `Metoda Euler perfectionata nu converge
la pasul`, i)
        fi;
        ypct[i] := yc;
        y00 := yc;
        x00 := x00 + h
    od;
    RETURN(ypct)
end;


>eroaremaxp := proc(f, x0, y0, h, n, eps, Nmax)
local i, yEp, y12, er, sol;
    yEp := mEulerP(f, x0, y0, h, n, eps, Nmax);
    y12 := matrix(2, n);
    for i to n do y12[1, i] := yEp[i] od;
    sol := rhs(dsolve({diff(y(x), x) = f(x, y(x)),
y(x0) = y0},
        y(x), explicit));
    for i to n do y12[2, i] := evalf(subs(x = x0 +
h*i, sol)) od;
    er := y12[2, 1] - y12[1, 1];
    for i from 2 to n do
        if abs(er) < abs(y12[2, i] - y12[1, i]) then
            er := y12[2, i] - y12[1, i]
        fi
    od;
    print(y12);
    print(`eroare maxima`, er)
end;

>
graficEP := proc(f, x0, y0, h, n, eps, Nmax)
local yEp, p1, p2, p3, sol, y1, y2, ra, rb, i;
    yEp := mEulerP(f, x0, y0, h, n, eps, Nmax);
    y1 := min(seq(yEp[i], i = 0 .. n));
    y2 := max(seq(yEp[i], i = 0 .. n));
```

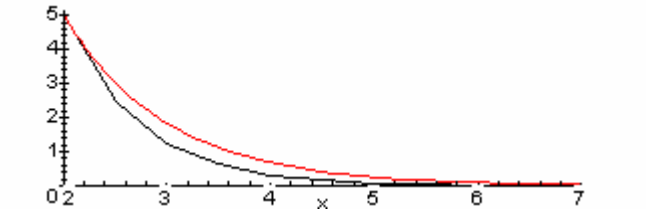
```
    ra := 1/100*h*n;
    rb := 1/100*y2 - 1/100*y1;
    p1 := seq(ellipse([x0 + h*i, yEp[i]], ra, rb,
filled = true,
        color = black), i = 0 .. n);
    sol := dsolve({diff(y(x), x) = f(x, y(x)), y(x0)
= y0}, y(x),
        numeric, method = classical[heunform], start
= x0,
        stepsize = h);
    p2 := odeplot(sol, [x, y(x)], x0 .. x0 + h*n,
color = black);
    p3 := p1, p2;
    display(p3)
end;
```

Exemple

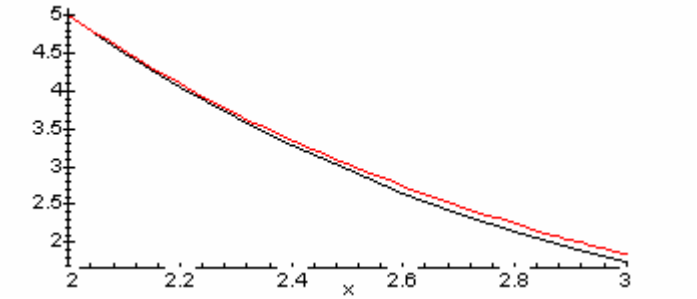
```
>with(linalg);
>with(plots);
>with(plottools);
>with(DEtools);
> f:=(x,y)->-y;
> yE:=mEuler(f,2,5,0.01,10);
                        yE := yp
> print(yE);
array(0 .. 10, [
    (0) = 5
    (1) = 4.995
    (2) = 4.990005
    (3) = 4.985014995
    (4) = 4.980029980
    (5) = 4.975049950
    (6) = 4.970074900
    (7) = 4.965104825
    (8) = 4.960139720
    (9) = 4.955179580
    (10) = 4.950224400
    ])

> repgraf(f,2,5,0.7,10);
```




```
> repgraf(f,2,5,0.5,10);
```




```
> repgraf(f,2,5,0.1,10);
```




```
> eroaremax(f,2,5,0.7,3);
```

```
[  1.5              .45             .135          ]
[2.482926521    1.232984820     .6122821415       ]

                eroare maxima, .982926521
```

```
> eroaremax(f,2,5,0.1,3);
```

```
[   4.5             4.05             3.645        ]
[4.524187090    4.093653768     3.704091104       ]
```

```
                eroare maxima, .059091104

> eroaremax(f,2,5,0.001,3);
```

$$\begin{bmatrix} 4.995 & 4.990005 & 4.985014995 \\ 4.995002501 & 4.990009996 & 4.985022480 \end{bmatrix}$$

```
                eroare maxima, .7485 10^-5
> compara(f,2,5,0.7,3);
```

$$\begin{bmatrix} 1.5 & .45 & .135 \\ 1.500000000 & .4500000000 & .1350000000 \end{bmatrix}$$

```
> compara(f,2,5,0.005,3);
```

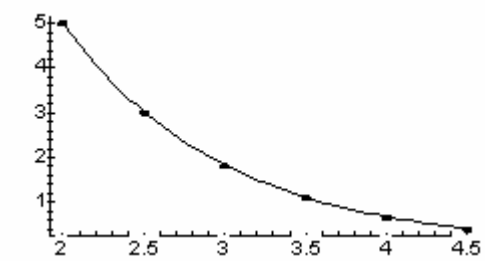
$$\begin{bmatrix} 4.975 & 4.950125 & 4.925374375 \\ 4.975000000 & 4.950125000 & 4.925374375 \end{bmatrix}$$

```
> yEP:=mEulerP(f,2,5,0.001,10,0.00001,4);
                        yEP := ypct
> print(yEP);
array(0 .. 10, [
    (0) = 5
    (1) = 4.995002500
    (2) = 4.990009995
    (3) = 4.985022480
    (4) = 4.980039950
    (5) = 4.975062400
    (6) = 4.970089825
    (7) = 4.965122220
    (8) = 4.960159580
    (9) = 4.955201901
    (10) = 4.950249177
    ])
> eroaremaxp(f,2,5,0.001,3,0.00001,4);
```

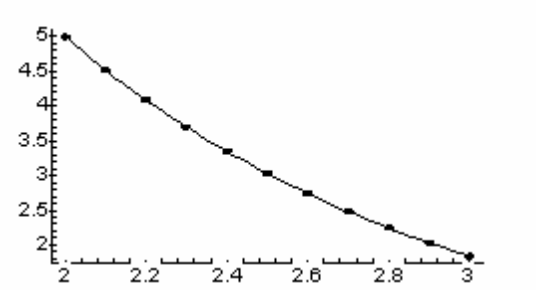
$$\begin{bmatrix} 4.995002500 & 4.990009995 & 4.985022480 \\ 4.995002501 & 4.990009996 & 4.985022480 \end{bmatrix}$$

```
                eroare maxima, .1 10^-8

> eroaremaxp(f,2,5,0.1,3,0.00001,4);
```

```
⎛ 4.523809375    4.092970252    3.703163440 ⎞
⎝ 4.524187090    4.093653768    3.704091104 ⎠

              eroare maxima, .000927664

> eroaremaxp(f,2,5,0.5,3,0.00001,4);
 Metoda Euler perfectionata nu converge la pasul, 1
 Metoda Euler perfectionata nu converge la pasul, 2
 Metoda Euler perfectionata nu converge la pasul, 3

⎛ 3.000488281    1.800585985    1.080527430 ⎞
⎝ 3.032653300    1.839397207    1.115650801 ⎠

              eroare maxima, .038811222

> eroaremaxp(f,2,5,0.5,3,0.00001,9);

⎛ 3.000001907    1.800002288    1.080002060 ⎞
⎝ 3.032653300    1.839397207    1.115650801 ⎠

              eroare maxima, .039394919

> graficEP(f,2,5,0.5,5,0.00001,9);
```




```
> graficEP(f,2,5,0.1,10,0.00001,4);
```



III.4. Metoda Runge-Kutta

Fie problema Cauchy

$$\begin{cases} y' = f(x,y),\ x \in [a, b] \\ y(x_0) = y_0 \end{cases}$$

unde $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$. Metoda Runge-Kutta constă în găsirea unui număr de puncte $y_1, y_2, \ldots, y_n$, care aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_n)$ ale soluției problemei Cauchy. $h = \frac{b-a}{n}$ este pasul discretizării şi $x_i = a + hi$, $i = 0,1,\ldots, n-1$. Se caută o funcție de h a cărei dezvoltare după puterile lui h să coincidă cu dezvoltarea în serie Taylor a soluției problemei Cauchy $y = y(x)$ în jurul lui x_0 în punctul $x_1 = x_0 + h$ până la o putere r. Se exprimă y_1 sub forma:

$$y_1 = y_0 + \sum_{j=1}^{r} p_{rj} k_j$$

unde p_{rj} sunt constante care urmează a fi determinate, iar k_j sunt valori înmulțite cu h ale funcției f în puncte vecine punctului (x_0, y_0) .

Pentru diverse valori ale lui r se obțin diverse formule de tip Runge – Kutta. Pentru r = 1 se regăseşte metoda Euler. Pentru r = 2 se obține

$$y_1 = y_0 + \frac{h}{2} f(x_0, y_0) + \frac{h}{2} f(x_0 + h, y_0 + hf(x_0, y_0))$$

Cunoscând y_1 putem determina y_2 cu o formulă asemănătoare. În general:

$$y_{i+1} = y_i + \frac{h}{2} f(x_i, y_i) + \frac{h}{2} f(x_i + h, y_i + hf(x_i, y_i)),\ i = 0,1,\ldots, n-1$$

Această formulă este utilizată în cadrul metodei Euler perfecționată pe post de formulă corector.

Pentru r =3 se obține:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_2 + k_3),\ i = 0,1, \ldots, n-1$$

unde

$$\begin{cases} k_1 = hf(x_i, y_i) \\ k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 = hf(x_i + h, y_i + 2k_2 - k_1) \end{cases}$$

Cazul cel mi important se obține pentru r = 4. Cea mai utilizată formulă Runge – Kutta este:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),\ i = 0,1, \ldots, n-1$$

unde

$$\begin{cases} k_1 = hf(x_i, y_i) \\ k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\ k_4 = hf(x_i + h, y_i + k_3) \end{cases}$$

Proceduri MAPLE

Procedura `mRungeKutta4` întoarce un tablou n+1 dimensional ce conține aproximațiile valorilor soluției problemei Cauchy y' = f(x,y), y(x0) = y0, obținute prin metoda Runge-Kutta de ordinul 4 . Procedura `graficRK` reprezintă grafic în acelaşi sistem de coordonate soluția adevărată a problemei Cauchy şi punctele aproximative obținute din procedura `mRungeKutta4`. Procedura `compgraficRK` reprezintă grafic soluția unei probleme Cauchy folosind DEplot cu opțiunea method=classical[rk4] (opțiunea implicită). Pe acelaşi grafic se reprezintă punctele obținute prin aplicarea procedurii `mRungeKutta4`. Toate aceste proceduri au drept parametri de intrare funcția f (se rezolva aproximativ ecuația y' = f(x,y)), x0, y0 ce dau condiția inițială (y(x0) = y0), pasul h, şi numărul de puncte aproximate n.

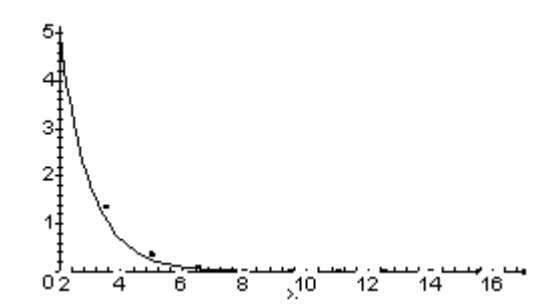
```
> mRungeKutta4 := proc(f, x0, y0, h, n)
local yp, i, k0, k1, k2, k3;
    yp := array(0 .. n);
```

```
    yp[0] := y0;
    for i from 0 to n - 1 do
        k0 := h*evalf(f(x0 + h*i, yp[i]));
        k1 := h*evalf(f(x0 + h*i + 1/2*h, yp[i] +
1/2*k0));
        k2 := h*evalf(f(x0 + h*i + 1/2*h, yp[i] +
1/2*k1));
        k3 := h*evalf(f(x0 + h*(i + 1), yp[i] +
k2));
        yp[i + 1] := yp[i] + 1/6*k0 + 1/3*k1 +
1/3*k2 + 1/6*k3
    od;
    RETURN(evalm(yp))
end;


>graficRK := proc(f, x0, y0, h, n)
local yR, sol, p1, p2, p3, y1, y2, ra, rb, i;
    yR := mRungeKutta4(f, x0, y0, h, n);
    y1 := min(seq(yR[i], i = 0 .. n));
    y2 := max(seq(yR[i], i = 0 .. n));
    ra := 1/200*h*n;
    rb := 1/200*y2 - 1/200*y1;
    p1 := seq(ellipse([x0 + h*i, yR[i]], ra, rb,
filled = true,
        color = black), i = 0 .. n);
    sol := rhs(dsolve({diff(y(x), x) = f(x, y(x)),
y(x0) = y0},
        y(x), explicit));
    p2 := plot(sol, x = x0 .. x0 + h*n, color =
black);
    p3 := p2, p1;
    display(p3)
end;

> compgraficRK := proc(f, x0, y0, h, n)
local yR, p1, p2, p3, y1, y2, ra, rb, i;
    yR := mRungeKutta4(f, x0, y0, h, n);
    y1 := min(seq(yR[i], i = 0 .. n));
    y2 := max(seq(yR[i], i = 0 .. n));
    ra := 1/75*h*n;
    rb := 1/75*y2 - 1/75*y1;
```

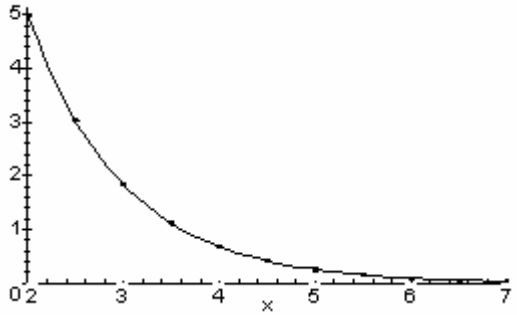
```
    p1 := seq(ellipse([x0 + h*i, yR[i]], ra, rb,
filled = true,
        color = black), i = 0 .. n);
    p2 := DEplot(diff(y(x), x) = f(x, y(x)), y(x),
x0 .. x0 + h*n,
        [[y(x0) = y0]], method = classical[rk4],
stepsize = h,
        startinit = TRUE, color = black, linecolor =
black);
    p3 := p2, p1;
    display(p3)
end;
```

Exemple

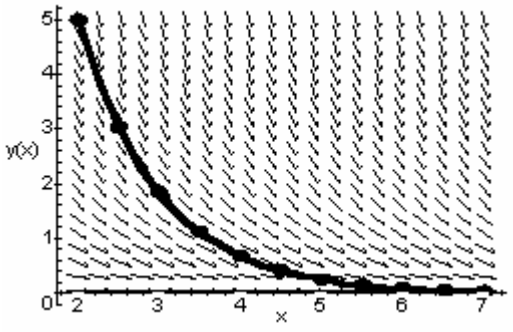
```
>with(linalg);
>with(plots);
>with(plottools);
>with(DEtools);
> f:=(x,y)->-y;
                    f := (x, y) -> -y
> yR:=mRungeKutta4(f,2,5,0.001,10);
                         yR := yp
> print(yR);
array(0 .. 10, [
    (0) = 5
    (1) = 4.995002500
    (2) = 4.990009995
    (3) = 4.985022480
    (4) = 4.980039949
    (5) = 4.975062398
    (6) = 4.970089823
    (7) = 4.965122218
    (8) = 4.960159578
    (9) = 4.955201898
    (10) = 4.950249172
    ])

> graficRK(f,2,5,1.5,10);
```



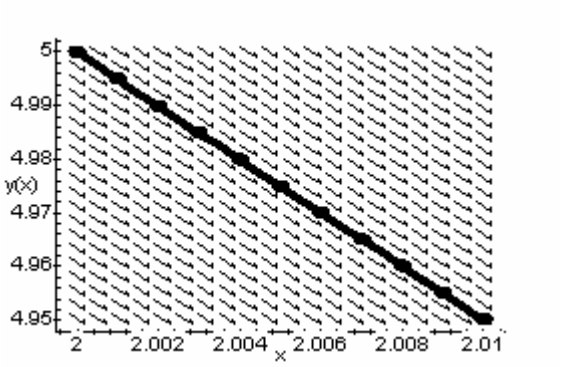

```
> graficRK(f,2,5,0.5,10);
```




```
> compgraficRK(f,2,5,0.5,10);
```




```
> compgraficRK(f,2,5,0.001,10);
```



III.5. Metoda predictor-corector cu paşi legați a lui Adams

Fie problema Cauchy

$$\begin{cases} y' = f(x,y),\ x \in [a,b] \\ y(x_0) = y_0 \end{cases}$$

unde $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$. Fie $h = \dfrac{b-a}{n}$ este pasul unei discretizări a intervalului $[a, b]$ şi $x_i = a + hi$, $i = 0,1,\ldots, n-1$.

Presupunem că printr-o anumită metodă numerică (cu paşi separaţi) s-a construit un tabel de tipul

x_0	x_1	…	x_i
y_0	y_1	…	y_i

unde $y_1, y_2, \ldots, y_i$, aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_i)$ ale soluției problemei Cauchy. Fie $k \le i$. Notăm $f_j(x_j, y_j) = f_j$, $j = 0, 1, \ldots, i$ şi fie $p_k(x)$ polinomul de interpolare generat de tabelul:

x_{i-k}	x_{i-k+1}	…	x_i
f_{i-k}	f_{i-k+1}	…	f_i

care aproximează funcția $x \to f(x, y(x))$. Formula "exactă":

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))dx$$

se înlocuieşte cu formula "aproximativă".

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} p_k(x)dx$$

care poartă numele de formula Adams – Bashforth. Se obțin următoarele formule Adams - Bashforth:

$$k = 1 \Rightarrow y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1})$$

$$k = 2 \Rightarrow y_{i+1} = y_i + \frac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2})$$

$$k = 3 \Rightarrow y_{i+1} = y_i + \frac{h}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3})$$

Dezavantajul acestei metode îl constituie faptul că nu se autoporneşte, adică nu poate determina valorile inițiale pentru pornire. Una din posibilitățile de a înlătura această dificultate este calculul valorilor inițiale de pornire cu ajutorul unei metode cu paşi separați, de exemplu metoda Runge – Kutta de ordinul patru.

Algoritmul utilizat de metoda Adams – Moulton este de tip implicit. Presupunem că $y_1, y_2, \ldots, y_i$, aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_i)$ ale soluției problemei Cauchy, iar y_{i+1}^p aproximează pe $y(x_{i+1})$. Fie $k \le i$. Notăm $f(x_j, y_j) = f_j$, $j = 0, 1, \ldots, i$, $f(x_{i+1}, y_{i+1}^p) = f_{i+1}$ şi şi fie $q_{k+1}(x)$ polinomul de interpolare generat de tabelul:

x_{i-k}	x_{i-k+1}	…	x_{i+1}
f_{i-k}	f_{i-k+1}	…	f_{i+1}

care aproximează funcția $x \to f(x, y(x))$. Formula "exactă":

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))dx$$

se înlocuieşte cu formula "aproximativă".

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} q_{k+1}(x)dx$$

care poartă numele de formula Adams – Moulton şi din care se determină corecția $y_{i+1}^c = y_i + \int_{x_i}^{x_{i+1}} q_{k+1}(x)dx$. În cazul metodei Adams – Bashforth – Moulton (predictor-corector) pentru determinarea lui y_{i+1} se procedează în felul următor

- Se calculează y_{i+1}^p prin formula Adams – Bashforth pentru un anumit k_1.
- Valoarea y_{i+1} este determinată iterativ. Pentru prima iterație se calculează y_{i+1}^c cu formula Adams – Moulton pentru un anumit k_2 cu ajutorul lui y_{i+1}^p, iar pentru următoarele iterații în locul lui y_{i+1}^p se ia y_{i+1}^c obținut în iterația precedentă. Aplicarea formulei Adams – Moulton se repetă până se ajunge la precizia impusă $\varepsilon > 0$, adică până când modul (sau norma în cazul m – dimensional) a două valori y_{i+1}^c consecutive devin mai mică decât ε.

Valorile inițiale de pornire se determină cu ajutorul unei metode cu pași separați, de obicei utilizându-se metoda Runge – Kutta. Cele mai utilizate metode corector – predictor sunt:

$$\begin{cases} y_{i+1}^p = y_i + \frac{h}{2}(3f_i - f_{i-1}), & k_1 = 1 \\ y_{i+1}^c = y_i + \frac{h}{2}(f_{i+1} - f_i), & k_2 = 0 \end{cases}$$

$$\begin{cases} y_{i+1}^p = y_i + \frac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2}), & k_1 = 2 \\ y_{i+1}^c = y_i + \frac{h}{12}(5f_{i+1} + 8f_i - f_{i-1}), & k_2 = 1 \end{cases}$$

$$\begin{cases} y_{i+1}^p = y_i + \frac{h}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}), & k_1 = 3 \\ y_{i+1}^c = y_i + \frac{h}{24}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2}), & k_2 = 2. \end{cases}$$

Proceduri Maple

Procedura `mABM` întoarce un tablou n+1 dimensional ce conține aproximațiile valorilor soluției problemei Cauchy y' = f(x,y), y(x0) = y0, aproximații obținute prin metoda predictor-corector Adams-Bashforth-Moulton, k_1=3 şi k_2 =2. Punctele de pornire sunt obținute prin metoda Runge-Kutta de ordinul 4. Procedura `grafic` reprezintă grafic în acelaşi sistem de coordonate soluția adevărată a problemei Cauchy şi punctele aproximative obținute aplicând metoda predictor-corector Adams-Bashforth-Moulton, k_1=3 şi k_2 =2. Sunt reprezentate deasemenea punctele aproximative obținute în etapa predictor (metoda Adams-Bashfoth, k=3). Punctele obținute prin metoda Adams-Bashforth-Moulton sunt reprezentate prin elipse umplute, iar punctele prezise (prin metoda Adams-Bashforth) sunt reprezentate prin elipse (neumplute). Cele două proceduri au drept parametri de intrare funcția f (se rezolva aproximativ ecuația y' = f(x,y)), x0, y0 ce dau condiția inițială (y(x0) = y0), pasul h, numărul de puncte aproximate n, numărul de puncte de pornire n0, eroarea eps ce stabileşte precizia pentru valorile corectate, şi numărul maxim de corecții Nmax.

```
> mABM := proc(f, x0, y0, h, n, n0, eps, Nmax)
local ypA, i, k0, k1, k2, k3, k, yp, yc;
    ypA := array(0 .. n);
    ypA[0] := y0;
    for i from 0 to n0 - 1 do
        k0 := h*evalf(f(x0 + h*i, ypA[i]));
        k1 := h*evalf(f(x0 + h*i + 1/2*h, ypA[i] +
1/2*k0));
        k2 := h*evalf(f(x0 + h*i + 1/2*h, ypA[i] +
1/2*k1));
        k3 := h*evalf(f(x0 + h*(i + 1), ypA[i] +
k2));
        ypA[i + 1] := ypA[i] + 1/6*k0 + 1/3*k1 +
1/3*k2 + 1/6*k3
    od;
    for i from n0 to n - 1 do
        yp := ypA[i] + 1/24*(55*evalf(f(x0 + h*i,
ypA[i]))
            - 59*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
            + 37*evalf(f(x0 + h*(i - 2), ypA[i -
2]))
```

```
              - 9*evalf(f(x0 + h*(i - 3), ypA[i -
3])))*h;
        yc := ypA[i] + 1/24*(9*evalf(f(x0 + h*(i +
1), yp))
              + 19*evalf(f(x0 + h*i, ypA[i]))
              - 5*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
              + evalf(f(x0 + h*(i - 2), ypA[i -
2])))*h;
        k := 1;
        while eps <= abs(yc - yp) and k <= Nmax do
            yp := yc;
            yc := ypA[i] + 1/24*(9*evalf(f(x0 + h*(i
+ 1), yp))
                  + 19*evalf(f(x0 + h*i, ypA[i]))
                  - 5*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
                  + evalf(f(x0 + h*(i - 2), ypA[i -
2])))*h;
            k := k + 1
        od;
        if Nmax < k then print(
            `Metoda    Adams-Bashforth-Moulton    nu
converge la pasul`
            , i)
        fi;
        ypA[i + 1] := yc
    od;
    RETURN(evalm(ypA))
end;


>grafic := proc(f, x0, y0, h, n, n0, eps, Nmax)
local ypA, yB, k0, k1, k2, k3, k, yp, yc, i, y1, y2,
ra, rb, p1,
p2, p3, p4, sol;
    ypA := array(0 .. n);
    yB := array(0 .. n);
    ypA[0] := y0;
    for i from 0 to n0 - 1 do
        k0 := h*evalf(f(x0 + h*i, ypA[i]));
        k1 := h*evalf(f(x0 + h*i + 1/2*h, ypA[i] +
1/2*k0));
```

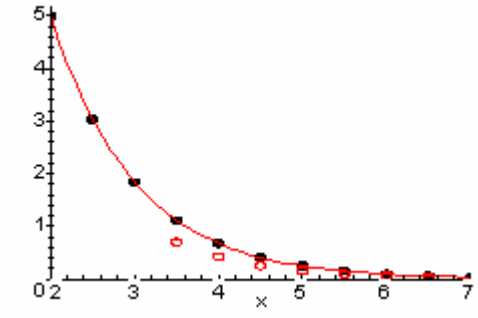
```
        k2 := h*evalf(f(x0 + h*i + 1/2*h, ypA[i] +
1/2*k1));
        k3 := h*evalf(f(x0 + h*(i + 1), ypA[i] +
k2));
        ypA[i + 1] := ypA[i] + 1/6*k0 + 1/3*k1 +
1/3*k2 + 1/6*k3
    od;
    for i from n0 to n - 1 do
        yp := ypA[i] + 1/24*(55*evalf(f(x0 + h*i,
ypA[i]))
             - 59*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
             + 37*evalf(f(x0 + h*(i - 2), ypA[i -
2]))
             - 9*evalf(f(x0 + h*(i - 3), ypA[i -
3])))*h;
        yB[i] := yp;
        yc := ypA[i] + 1/24*(9*evalf(f(x0 + h*(i +
1), yp))
             + 19*evalf(f(x0 + h*i, ypA[i]))
             - 5*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
             + evalf(f(x0 + h*(i - 2), ypA[i -
2])))*h;
        k := 1;
        while eps <= abs(yc - yp) and k <= Nmax do
            yp := yc;
            yc := ypA[i] + 1/24*(9*evalf(f(x0 + h*(i
+ 1), yp))
                 + 19*evalf(f(x0 + h*i, ypA[i]))
                 - 5*evalf(f(x0 + h*(i - 1), ypA[i -
1]))
                 + evalf(f(x0 + h*(i - 2), ypA[i -
2])))*h;
            k := k + 1
        od;
        ypA[i + 1] := yc
    od;
    y1 := min(min(seq(ypA[i], i = 0 .. n)),
        min(seq(yB[i], i = n0 .. n - 1)));
    y2 := max(max(seq(ypA[i], i = 0 .. n)),
        max(seq(yB[i], i = n0 .. n - 1)));
    ra := 1/75*h*n;
```

```
    rb := 1/75*y2 - 1/75*y1;
    p1 := seq(ellipse([x0 + h*i, ypA[i]], ra, rb,
filled = true,
        color = black), i = 0 .. n);
    p2 := seq(ellipse([x0 + h*i, yB[i]], ra, rb,
filled = false,
        color = red), i = n0 .. n - 1);
    sol := rhs(dsolve({diff(y(x), x) = f(x, y(x)),
y(x0) = y0},
        y(x), explicit));
    p3 := plot(sol, x = x0 .. x0 + h*n, color =
red);
    p4 := p3, p2, p1;
    display(p4)
end;
```

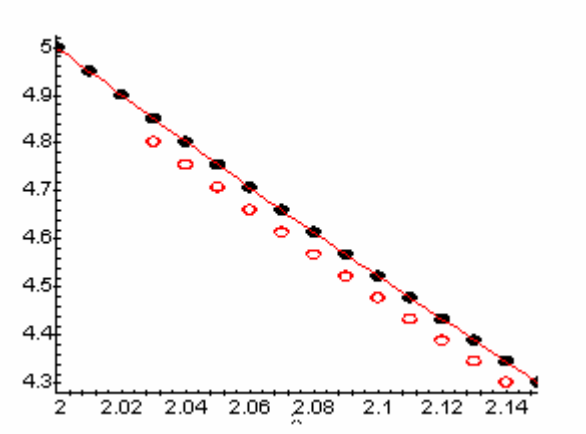
Exemple

```
>with(plots);
>with(plottools);
>with(DEtools);
> f:=(x,y)->-y;
                    f := (x, y) -> -y
> yA:=mABM(f,2,5,0.001,10,3,0.0001,4);
                        yA := ypA
> print(yA);
array(0 .. 10, [
    (0) = 5
    (1) = 4.995002500
    (2) = 4.990009995
    (3) = 4.985022480
    (4) = 4.980039949
    (5) = 4.975062398
    (6) = 4.970089822
    (7) = 4.965122216
    (8) = 4.960159576
    (9) = 4.955201896
    (10) = 4.950249171
    ])
> yA1:=mABM(f,2,5,0.5,10,3,0.0001,4);
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              3
```

```
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                                4
                      yA1 := ypA

> print(yA1);
array(0 .. 10, [
     (0) = 5
     (1) = 3.033854167
     (2) = 1.840854220
     (3) = 1.116976650
     (4) = .6765341520
     (5) = .4098830016
     (6) = .2482954649
     (7) = .1504177341
     (8) = .09112145418
     (9) = .05518625432
     (10) = .03342395473
     ])
> grafic(f,2,5,0.5,10,3,0.0001,4);
```




```
> grafic(f,2,5,0.01,15,3,0.0001,4);
```



III.6. Alegerea metodei numerice de rezolvarea a ecuațiilor diferențiale

După cum am văzut metodele numerice pentru rezolvarea problemei Cauchy:

$$\begin{cases} y' = f(x,y),\ x \in [a, b] \\ y(x_0) = y_0 \end{cases}$$

unde $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$, constau în găsirea unui număr de puncte $y_1, y_2, \ldots, y_n$, care aproximează valorile adevărate $y(x_1), y(x_2), \ldots, y(x_n)$ ale soluției problemei Cauchy. Diferența dintre valoarea adevărată $y(x_i)$ şi valoarea aproximativă y_i reprezintă eroarea totală pe pasul de calcul. Eroarea totală este compusă din:

- eroarea de aproximare datorată metodei numerice
- eroarea de rotunjire datorată limitării numărului de cifre semnificative
- eroarea de propagare datorată erorii din paşii anteriori

Eroarea de aproximare depinde de metoda numerică utilizată:

Denumirea metodei	Ordinul de mărime al erorii de aproximare (trunchiere) (h pas)
Euler	$O(h^2)$
Euler perfecționată	$O(h^2)$
Runge-Kutta de ordinul 2	$O(h^3)$
Runge-Kutta de ordinul 3	$O(h^4)$
Runge-Kutta de ordinul 5	$O(h^5)$
Adams – Bashforth, k =1	$O(h^3)$
Adams – Bashforth, k =2	$O(h^4)$
Adams – Bashforth, k =3	$O(h^5)$
Adams-Bashforth-Moulton, $k_1=1$, $k_2=0$	$O(h^3)$
Adams-Bashforth-Moulton, $k_1=2$, $k_2=1$	$O(h^4)$
Adams-Bashforth-Moulton, $k_1=3$, $k_2=2$	$O(h^5)$

Micşorarea erorii de aproximare este posibilă dacă se face o micşorare corespunzătoare a pasului h.

Eroarea de rotunjire depinde de numărul de cifre semnificative cu care se lucrează (simplă precizie, dublă precizie, şamd). Mărirea numărului de cifre semnificative are ca efect diminuarea erorii de rotunjire. Posibilitatea măririi numărului de cifre semnificative depinde de calculator dar şi de softul utilizat. În MAPLE numărul de cifre semnificative poate fi controlat cu ajutorul variabilei globale Digits. Eroarea de rotunjire creşte pe măsura creşterii numărului de paşi. De aceea micşorarea pasului pentru diminuarea erorii de trunchiere (aproximare) are ca efect creşterea erorii de rotunjire, deoarece creşte numărul de paşi. Deci există o valoare optimă a pasului pentru care eroarea totală este minimă. Pe de altă parte micşorarea pasului deci creşterea numărului de paşi are ca efect o creştere a volumului de calcul şi în consecință a timpului de execuție. Această creştere a volumului de calcul este nejustificată dacă problema pe care încercăm să o rezolvăm nu necesită limitarea erorii pe pas.

Dacă numărul de puncte pe care le determinăm este redus, se poate alege o metodă simplă Euler sau Euler perfecționată, cu un pas relativ mic, dar cu o valoare absolută superioară lui ε_{mach}. Dacă numărul de puncte este mare, este recomandabil să se aleagă o metodă cu eroare mică pe pas, de exemplu Adams-Bashforth-Moulton (predictor-corector). Dacă facem abstracție de timpul de calcul este recomandabilă metoda Runge – Kutta de ordinul 4.

Pentru rezolvarea unei ecuații de ordin superior se deduce sistemul de ecuații diferențiale echivalent cu ecuația inițială. Considerăm ecuația diferențială de ordinul m:

$$y^{(m)} = f(x, y(x), y'(x), \ldots, y^{(m-1)}(x))$$

unde $f : [a, b] \times D \to \mathbf{R}^m$, $D \subset \mathbf{R}^m$. Considerăm condițiile inițiale:

$$y^{(i)}(x_0) = y_0^i \, , \; i = 0, 1, \ldots, m\text{-}1$$

cu $x_0 \in [a, b]$. Notăm $z_i(x) = y^{(i-1)}(x)$, $i = 1, 2 \ldots, m$. Ecuația diferențială este echivalentă cu sistemul:

$$\begin{cases} z_1'(x) = z_2(x) \\ z_2'(x) = z_3(x) \\ \vdots \\ z_{m-1}'(x) = z_m(x) \\ z_m'(x) = f(x, z_1(x), z_3(x), \ldots, z_m(x)) \end{cases}$$

cu condițiile inițiale

$$z_1(x_0) = y_0^0 \, , \; z_2(x_0) = y_0^1 \, , \ldots, \; z_m(x_0) = y_0^{m-1} \, .$$

Procedură Maple pentru compararea erorilor

Procedura `eroaremaxcomp` întoarce un tabel ce conține erorile maxime în cazul aplicării diverselor metode de rezolvare numerică a ecuațiilor diferențiale (se presupune că problemă Cauchy ce poate fi rezolvată prin metode exacte). Parametri de intrare sunt: funcția f (se rezolva aproximativ ecuația y' = f(x,y)), x0, y0 ce dau condiția inițială (y(x0) = y0), pasul h, numărul de puncte aproximate n, numărul de puncte de pornire n0 (acest parametru este utilizat doar de metodele cu pași legați, în cazul nostru de metoda Adams-Bashforth-Moulton), eroarea eps ce stabilește precizia pentru valorile corectate, și numărul maxim de corecții Nmax (eps și Nmax sunt utilizați doar de metodele predictor –corector, în cazul nostru metoda Euler perfecționată și metoda Adams-Bashforth-Moulton). Această procedură folosește procedurile definite în secțiunile precedente.

```
>eroaremaxcomp := proc(f, x0, y0, h, n, n0, eps,
Nmax)
local i, yp, y1, er, sol, erori;
```

```
   erori := table([Euler = 0, Euler_Perfectionata =
0,Runge_Kutta4 = 0, Adams_Basforth_Moulton = 0]);
    y1 := array(1 .. n);
    sol := rhs(dsolve({diff(y(x), x) = f(x, y(x)),
y(x0) = y0},y(x), explicit));
    for i to n do y1[i] := evalf(subs(x = x0 + h*i,
sol)) od;
    yp := mEuler(f, x0, y0, h, n);
    er := y1[1] - yp[1];
    for i from 2 to n do
        if abs(er) < abs(y1[i] - yp[i]) then er :=
y1[i] - yp[i]
        fi
    od;
    erori[Euler] := er;
    yp := mEulerP(f, x0, y0, h, n, eps, Nmax);
    er := y1[1] - yp[1];
    for i from 2 to n do
        if abs(er) < abs(y1[i] - yp[i]) then er :=
y1[i] - yp[i]
        fi
    od;
    erori[Euler_Perfectionata] := er;
    yp := mRungeKutta4(f, x0, y0, h, n, eps, Nmax);
    er := y1[1] - yp[1];
    for i from 2 to n do
        if abs(er) < abs(y1[i] - yp[i]) then er :=
y1[i] - yp[i]
        fi
    od;
    erori[Runge_Kutta4] := er;
    yp := mABM(f, x0, y0, h, n, n0, eps, Nmax);
    er := y1[1] - yp[1];
    for i from 2 to n do
        if abs(er) < abs(y1[i] - yp[i]) then er :=
y1[i] - yp[i]
        fi
    od;
    erori[Adams_Basforth_Moulton] := er;
    RETURN(evalm(erori))
end;
```

Exemple

```
> erori1:=eroaremaxcomp(f,2,5,0.9,10,3,0.1,4);
 Metoda Euler perfectionata nu converge la pasul, 1
                    erori1 := erori
> print(erori1);
table([
    Adams_Basforth_Moulton = -.0215078915

    Euler_Perfectionata = .1193550665

    Euler = 1.532848299

    Runge_Kutta4 = -.021339201
    ])
> erori2:=eroaremaxcomp(f,2,5,0.9,10,3,0.01,4);
 Metoda Euler perfectionata nu converge la pasul, 1

 Metoda Euler perfectionata nu converge la pasul, 2

 Metoda Euler perfectionata nu converge la pasul, 3

 Metoda Euler perfectionata nu converge la pasul, 4

Metoda Adams-Bashforth-Moulton nu converge la pasul,
                          3


                    erori2 := erori

> print(erori2);
table([
    Adams_Basforth_Moulton = -.021339201
    Euler_Perfectionata = .110526268
    Euler = 1.532848299
    Runge_Kutta4 = -.021339201
    ])
> erori3:=eroaremaxcomp(f,2,5,0.5,10,3,0.1,4);
                    erori3 := erori
> print(erori3);
table([
    Adams_Basforth_Moulton = .0058363234
    Euler_Perfectionata = .055279402
```

```
    Euler = .589397207
    Runge_Kutta4 = -.001457013
    ])
> erori4:=eroaremaxcomp(f,2,5,0.5,10,3,0.0001,4);
 Metoda Euler perfectionata nu converge la pasul, 1
 Metoda Euler perfectionata nu converge la pasul, 2
 Metoda Euler perfectionata nu converge la pasul, 3
 Metoda Euler perfectionata nu converge la pasul, 4
 Metoda Euler perfectionata nu converge la pasul, 5
 Metoda Euler perfectionata nu converge la pasul, 6
 Metoda Euler perfectionata nu converge la pasul, 7
 Metoda Euler perfectionata nu converge la pasul, 8
 Metoda Euler perfectionata nu converge la pasul, 9
   Metoda Adams-Bashforth-Moulton nu converge la
                        pasul,3
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              4
                    erori4 := erori
> print(erori4);

table([
    Adams_Basforth_Moulton = -.001457013
    Euler_Perfectionata = .038811222

    Euler = .589397207
    Runge_Kutta4 = -.001457013
    ])
>erori5:=eroaremaxcomp(f,2,5,0.0005,10,3,0.00001,4);
                             erori5 := erori
> print(erori5);
table([
      Adams_Basforth_Moulton = .2 10^-8
      Euler_Perfectionata = .3 10^-8
      Euler = .6222 10^-5
      Runge_Kutta4 = -.3 10^-8
    ])
>erori6:=eroaremaxcomp(f,2,5,0.0005,1000,3,0.00001,4
);
                    erori6 := erori
> print(erori6);
table([
      Adams_Basforth_Moulton = .10 10^-7
      Euler_Perfectionata = -.60 10^-7
```

```
      Euler = .000379193
      Runge_Kutta4 = .18 10^-7
    ])
> Digits:=25;
                              Digits := 25
> erori7:=eroaremaxcomp(f,2,5,0.5,1000,3,0.00001,4);
 Metoda Euler perfectionata nu converge la pasul, 1
 Metoda Euler perfectionata nu converge la pasul, 2
 Metoda Euler perfectionata nu converge la pasul, 3
 Metoda Euler perfectionata nu converge la pasul, 4
 Metoda Euler perfectionata nu converge la pasul, 5
 Metoda Euler perfectionata nu converge la pasul, 6
 Metoda Euler perfectionata nu converge la pasul, 7
 Metoda Euler perfectionata nu converge la pasul, 8
 Metoda Euler perfectionata nu converge la pasul, 9
 Metoda Euler perfectionata nu converge la pasul, 10
 Metoda Euler perfectionata nu converge la pasul, 11
 Metoda Euler perfectionata nu converge la pasul, 12
 Metoda Euler perfectionata nu converge la pasul, 13
 Metoda Euler perfectionata nu converge la pasul, 14
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              3
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              4
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              5
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              6
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              7
Metoda Adams-Bashforth-Moulton nu converge la pasul,
                              8
                    erori7 := erori

> print(erori7);
table([
 Adams_Basforth_Moulton = -.001457015062927280911270
 Euler_Perfectionata = .038811220673495787665119
 Euler = .58939720585721160797 7619
 Runge_Kutta4 = -.001457015062927280911270
 ])
>erori8:=eroaremaxcomp(f,2,5,0.0005,1000,3,0.00001,4
);
```

```
erori8 := erori
```

```
> print(erori8);
table([
 Adams_Basforth_Moulton = .2498647441 10^-14
 Euler_Perfectionata = -.63203975054436380 10^-7
 Euler = .00037918436285929699640 0
 Runge_Kutta4 = -.790082605 10^-15
 ])
```