9. NUMERICAL INTEGRATION. SOLVING DIFFERENTIAL EQUATIONS AND SYSTEMS OF DIFFERENTIAL EQUATIONS

Objectives of the paper:

- Fixing the knowledge regarding the calculation of the defined integral of a function by the trapezoidal method, respectively Simpson's method,
- Familiarizing with the ways of calculating the defined integral of a function using the Matlab environment,
- Fixing knowledge about solving ordinary differential equations and ordinary differential equation systems using the Matlab environment,
- Acquiring the method to bring the ordinary differential equations or the systems of higher order ordinary equations to a form equivalent to the systems of first-order differential equations, by studying some examples and solving some problems.

It is recommended to go through Annex M9 before studying paragraphs 9.1 and 9.2.

9.1. Elements regarding numerical integration in Matlab. Elements regarding the solving of systems of differential equations in Matlab

Numerical integration

For the calculation of the integrals defined by ***the trapezoidal method***, in Matlab the `trapz` function is used. This function assumes that the function `f` that has to be integrated is specified as numerical values, $\{y_k = f(x_k)\}_{k=1,\dots,n}$, in equidistant points $\{x_k\}_{k=1,\dots,n}(x_1 = a, x_n = b)$ of the integration interval [`a,b`]. The syntax of the function `trapz` is:

```
I=trapz(x,y)
```

where:

- `x` represents the vector of the values $\{x_k\}$;
- `y` represents the vector of the values $\{y_k = f(x_k)\}$;
- `I` represents the approximation with the trapezoidal method of the defined integral $\int_a^b f(x)dx$

Comment: `n` here represents the number of points. Therefore, the integration step - which is implicitly calculated - is given by:

$$h = \frac{b-a}{n-1}$$

The Matlab *quad* function performs the calculation of the defined integral of a function by using ***the adaptive-recursive Simpson method*** (a more advanced version of Simpson's method, the step of going through the integration interval is implicitly calculated by the Matlab function). The *quad* function assumes that the integrated function *f* is known by its analytical expression, $y = f(x)$. Two *quad*

```
I=quad(file_name,a,b)
I=quad(file_name,a,b,precision)
```

function syntaxes are:

where:

- *file_name* represents a string that contains the name of the function-file in which the expression of the integrated function *f* was written;
- *a* and *b* represent the limits of integration (the ends of the interval [a,b] on which the integration is made);
- *precision* is an optional argument that can change the default precision 10^{-6};
- *I* represents the approximation of the defined integral $\int_a^b f(x)dx$.

Solving differential equations and systems of differential equations

Matlab provides the user with several Matlab functions for solving differential equations and systems of differential equations of different orders. These functions implement various numerical methods. Thus:

- the Matlab *ode23* function uses a combination of the Runge-Kutta methods of orders 2 and 3, respectively;
- the Matlab *ode45* function has implemented a combination of Runge-Kutta methods of orders 4 and 5, respectively;
- the Matlab *ode113* function implements a variant of the Adams-Bashforth-Moulton method.

The three Matlab functions mentioned above have the same syntax. Two of these syntaxes are:

```
[xval,yval] = Matlab_function(file_name,dom,y0)
[xval,yval] = Matlab function (file name,dom,y0,options)
```

where:

- *Matlab_function* is one of the functions: *ode23*,*ode45*,*ode113*;
- *file_name* represents a string that contains the name of the function-file in which the expression of the unknown functions derivative was defined, in the case of a first-order differential equation, respectively, the vector of the expressions of the first-order derivatives of the unknown functions, in the case of systems of first-order differential equations or equations and systems of higher order differential equations, which have been previously brought to a form equivalent to a first-order system;
- *dom* represents the vector of the ends of the interval [a,b] of the independent variable (see annex M9);
- *y0* represents the value of the unknown function of the initial condition in the case a first-order differential equation, respectively, the vector of values of the unknown functions of the initial conditions in other cases;

- *options* represents a structure that contains options for optimizing the calculation of the solution/solutions; is an optional argument; optimization options can be changed using Matlab *optimset* function (see paragraph 6.1);
- *xval* represents a vector that contains the values of the independent variable, in which the values of the solution/solutions are determined;
- *yval* represents the vector of the solution function values for the points *xval*, in the case of a first-order differential equation, respectively a matrix whose columns represent the values of the solution functions for the points *xval*, in the other cases.

9.2. Examples

Example 9.1: Having the function *f* given through points by the relations:

$f(x_i)=\dfrac{\sin(x_i)}{i^2+1}\cdot\cos\left(\dfrac{i}{x_i}\right)$, $x_i=\pi+i\cdot\dfrac{\pi}{30}$, $i=1,2,\ldots,150$. Calculate: $\displaystyle\int_{\pi+\frac{\pi}{30}}^{6\cdot\pi} f(x)dx$.

Solution: Because the function is known by points, the trapezoidal method will be used to calculate the required integral. The following Matlab program sequence is executed:

```
% generating the vectors x and y
for i=1:150
    x(i)=pi+i*pi/30;
    y(i)=sin(x(i))/(i^2+1)*cos(i/x(i));
end

% calculating the integral using the trapezoidal method
I=trapz(x,y)
```

resulting in the following integer value:

```
I=    -0.0025
```

Example 9.2: Calculate: $\displaystyle\int_0^{\pi}\ln(x+1)\cdot\sin(x)dx$.

Solution: Because the expression of the function is known, its integral will be calculated using the Simpson method. The expression of the function is defined in a function-file (e.g., `f.m`):

```
function y=f(x)
y=log(x+1).*sin(x);
```

In order to calculate the integral the Matlab *quad* function is used:

```
I=quad('f',0,pi)
```

resulting value is:

```
I=     1.8113
```

Example 9.3: Solve by using the Runge-Kutta method of order 2-3 the first-order differential equation: $y'=x^2\cdot(y+1)$ with the initial condition $y(1)=1$, over the interval [1,2].

Solution: The following two steps have to be performed:

- the expression of the derivative of the unknown function y is defined in a function-file, e.g.eqdif1.m:

```
function dy=eqdif1(x,y)
dy=x.^2.*(y+1);
```

- the differential equation is solved using theMatlab *ode23* function, by executing the following Matlab program sequence (e.g., script file):

```
% initial condition
y0=1;
% domain (interval)
dom=[1,2];
% solving the differential equation
[xval,yval]=ode23('eqdif1',dom,y0)
% graphical representation of the solution
plot(xval,yval)
```

The solution is obtained in the form of sets of values:

```
xval =
     1.0000     1.0400     1.1400     1.2400     1.3400     1.4368
     1.5280     1.6140     1.6950     1.7717     1.8443     1.9133
     1.9789     2.0000
yval =
     1.0000     1.0850     1.3482     1.7055     2.1955     2.8512
     3.7058     4.8171     6.2622     8.1423    10.5908    13.7830
    17.9494    19.6011
```

Solution is graphically represented in figure 9.1.

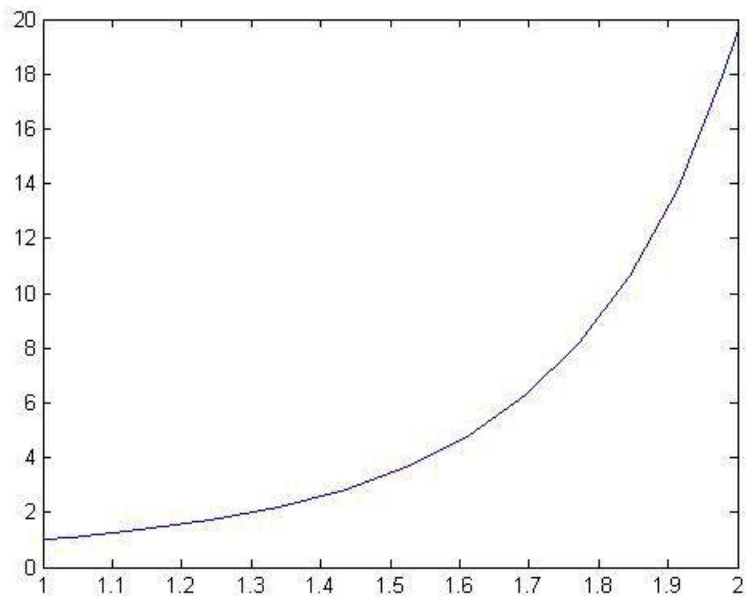

Fig.9.1. *Graph for the solution of the differential equation from example 9.3.*

Example 9.4: Solve by using the Adams-Bashforth-Moulton method the following second-order differential equation: $y''=2\cdot y'-3\cdot x^2\cdot y$ with the initial conditions: $y(0)=-1$, $y'(0)=2$, over the interval [1,2.5].

Solution: The equation is rewritten in the form of a system of two first-order differential equations by introducing the notations $y_1 = y$, $y_2 = y'$. The obtained system is:

$$\begin{cases} y_1' = y_2 \\ y_2' = 2 \cdot y_2 - 3 \cdot x^2 \cdot y_1 \end{cases}$$

with the initial conditions: $y_1(0) = -1$, $y_2(0) = 2$. Solving the above system involves going through the two stages described in the example 9.3.:,

- the vector of the expression of the derivatives of the unknown functions y_1 and y_2 are defined in a function-file (e.g. `eqdif2.m`):

```
function dy=eqdif2(x,y)
dy=zeros(2,1); %initializing the vector
dy(1)=y(2);
dy(2)= 2*y(2)-3*x.^2.*y(1);
```

- the differential equation is solvedby executing the following Matlab program sequence (e.g., script file):

```
% initial conditions
y0=[-1; 2];
% domain (interval)
dom=[1,2.5];
% solving the differential equation
[xval,yval]=ode113('eqdif2',dom,y0)
% graphical representation of the solution
plot(xval,yval(:,1))
```

The obtained solution (first column of the `yval` matrix) and its derivative (second column of the `yval` matrix) as sets of values:

```
xval =                    yval =
    1.0000                   -1.0000    2.0000
    1.0023                   -0.9955    2.0158
    1.0068                   -0.9863    2.0478
    1.0158                   -0.9675    2.1124
    1.0339                   -0.9281    2.2451
    1.0700                   -0.8420    2.5236
    1.1423                   -0.6381    3.1282
    1.2869                   -0.0908    4.4597
    1.3591                    0.2558    5.1217
    1.4314                    0.6480    5.7155
    1.5037                    1.0787    6.1732
    1.6483                    1.9980    6.3409
    1.7928                    2.8318    4.8694
    1.9229                    3.2723    1.5840
    1.9880                    3.2996   -0.8232
    2.0530                    3.1549   -3.6957
    2.1181                    2.8109   -6.9292
    2.2482                    1.4658  -13.7100
    2.3783                   -0.6870  -18.9046
    2.5000                   -3.0793  -19.5740
```

The solution is graphically represented in figure 9.2.

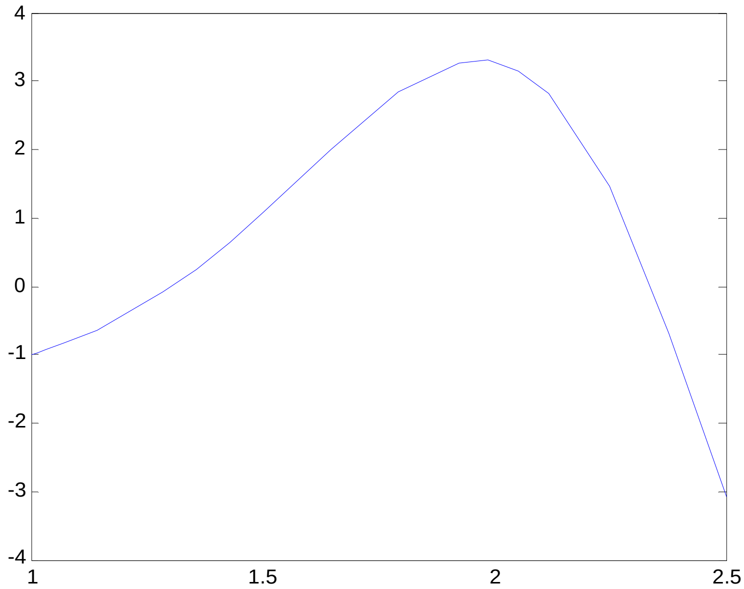

Fig.9.2. *Graph for the solution of the differential equation from example 9.4.*

Example 9.5: Solve by using the Runge-Kuttaof order 4-5 method the following system of first-order differential equations:

$$\begin{cases} y_1' = y_1 + y_2 \\ y_2' = x - y_1 \end{cases}$$

with the initial conditions: $y_1(0) = 0.1$, $y_2(0) = 0.2$, over the interval [0,10].

Solution: Following steps have to be taken:

- the vector of the expressions of the derivatives is defined in a function-file (e.g. `systdif.m`):

```
function dy=systdif(x,y)
dy=zeros(2,1); %initializing the vector
dy=[y(1)+y(2); x-y(1)];
```

- the system of differential equation is solved by executing the following Matlab set of instructions (script file):

```
% initial conditions
y0=[0.1; 0.2];
% domain (interval)
dom=[0,10];
% solving the differential equation
[xval,yval]=ode45('systdif',dom,y0)
% graphical representation of the solution
plot(xval,yval(:,1),'b',xval,yval(:,2),'r--')
legend('y1','y2')
```

obtaining solutions in the form of points (first column of the matrix `yval` represents the solution function y_1, the second column represents the solution function y_2):

```
xval=    0              yval =     0.1000     0.2000
     0.0167                         0.1051     0.1984
     0.0335                         0.1102     0.1970
     0.0502                         0.1153     0.1959
     0.0670                         0.1206     0.1949
     0.1507                         0.1480     0.1927
     0.2344                         0.1778     0.1952
     0.3182                         0.2107     0.2021
     0.4019                         0.2472     0.2131
     0.5658                         0.3317     0.2452
     0.7296                         0.4381     0.2886
     0.8935                         0.5718     0.3393
     1.0573                         0.7386     0.3922
     1.2445                         0.9770     0.4479
     1.4317                         1.2747     0.4886
     1.6188                         1.6396     0.5024
     1.8060                         2.0787     0.4763
     2.0402                         2.7407     0.3648
     2.2744                         3.5344     0.1375
     2.5087                         4.4600    -0.2361
     2.7429                         5.5095    -0.7860
     2.9548                         6.5508    -1.4585
     3.1667                         7.6588    -2.3146
     3.3787                         8.8049    -3.3654
     3.5906                         9.9509    -4.6147
     3.8182                        11.1274    -6.1710
     4.0458                        12.1810    -7.9312
     4.2734                        13.0335    -9.8593
     4.5010                        13.5968   -11.8983
     4.7510                        13.7696   -14.1717
     5.0010                        13.3529   -16.3547
     5.2510                        12.2286   -18.2881
     5.5010                        10.2933   -19.7796
     5.7510                         7.4709   -20.6124
     6.0010                         3.7225   -20.5593
     6.2510                        -0.9331   -19.3946
     6.5010                        -6.4021   -16.9022
     6.7027                       -11.2909   -13.7945
     6.9043                       -16.4650    -9.6271
     7.1060                       -21.7488    -4.3594
     7.3077                       -26.9177     2.0060
     7.5085                       -31.6833     9.3849
     7.7094                       -35.7634    17.6997
     7.9103                       -38.8186    26.7816
     8.1111                       -40.4739    36.3831
     8.3315                       -40.2090    47.1212
     8.5518                       -37.2497    57.5623
     8.7721                       -31.0862    67.0658
     8.9924                       -21.2531    74.8667
     9.2185                        -6.9471    80.1954
     9.4446                        11.9206    81.8218
     9.6706                        35.4431    78.7156
     9.8967                        63.4540    69.8396
     9.9225                        66.9222    68.4123
     9.9484                        70.4427    66.8955
     9.9742                        74.0143    65.2878
    10.0000                        77.6362    63.5879
```

The solution graph is shown in the figure 9.3.:

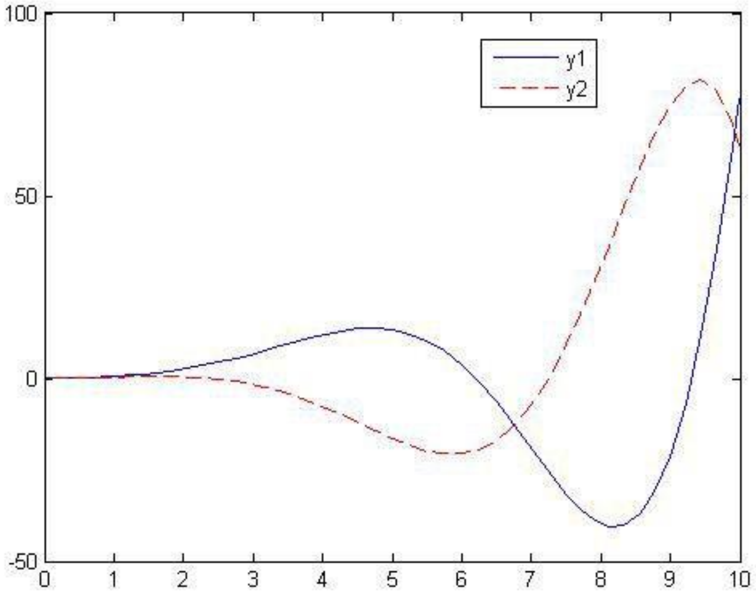

Fig.9.3. *Graph for the solutions of the system of differential equations from example 9.5.*

9.3. Problems to solve

P9.1. Calculate : $\int_{-1}^{0} f(x)dx$, where the function f is given by the following relations:

$$f(x_j)=\frac{j\cdot x_j^2}{x_j-1}-\frac{2}{j+1},\; x_j=-1.1+0.1\cdot j,\; j=1,2,\ldots,11$$

P9.2. Calculate the following integral:

$$\int_{\frac{\pi}{3}}^{\frac{\pi}{2}} \frac{1}{\sin(x)+\cos(x)}dx \,.$$

P9.3. Solve the following Cauchy problems on the mentioned intervals. The solution/ solutionswill be graphically represented (in the case of systems, the representation of the solutions will be done in the same graphical window):

a) $y'+y^2=3\cdot x$, $y(-1)=2$, over [-1,5];

b) $y''-y'=2\cdot y\cdot\sin(t)$, $y(0)=-1$, $y'(0)=2$, over [0,6];

c) $-y'''+y''-x\cdot y'+2\cdot y\cdot\sin(x)-x^3=0$, $y(1)=0.5$, $y'(1)=-0.5$, $y''(1)=0.3$ over [1,4];

d) $\begin{cases} x'+2x=y-2z+\sin(t), & x(0)=0 \\ y'+2y=x+2z-\cos(t), & y(0)=0.2 \\ z'-5z=3x-3y, & z(0)=-0.1 \end{cases}$, over [0,3].

P9.4. Approximate the values of the function-solutions obtained in the problem P9.3. for the following points:

a) -1, -0.5, 0, 1, 2.3, 5for the solution from P9.3.a);

b) 0, 1.5, 2.3, 3.7, 4, 5.45, 6for the solution from P9.3.b);

c) 1, 2.2, 3.5, 4 for the solution from P9.3.c);

d) 0, 0.75, 1.1, 1.16, 2, 3 for the solution from P9.3.d).

P9.5. Consider a robot with three degrees of freedom (translation-translation-rotation), whose dynamic equations of motion are:

$$\begin{cases} (m_1 + m_2 + m_3)\ddot{q}_1 = F_1 \\ (m_2 + m_3)\ddot{q}_2 = F_2 - m_2 g - m_3 g \\ J_3 \ddot{q}_3 = M_3 \end{cases},$$

in which the following notations were used:

- q_1, q_2, q_3 - generalized coordinates (time functions,t);
- m_1, m_2, $m_3$the masses of the robot coupling-element assemblies;
- F_1, F_2 - the forces that produce the translational movement of the couplings;
- M_3 - the moment that causes the movement of the rotation coupling;
- J_3 - the axial moment of inertia of the assembly coupling 3 - element3.

Knowing the masses (m_1=10 kg, m_2=4.15 kg, m_3=0.5 kg), the axial moment of inertia (J_3=0.015 kgm^2), analytical expressions of forces and of the moment:

$F_1(t)$=-58.6·sin(2·t) $F_2(t)$=23.25·e^{-t}·(sin(4·t)-3cos(4·t))+45.601 $M_3(t)$=0.004·t^2

and the initial conditions:$q_1(0)$=0, $q'_1(0)$=2, $q_2(0)$=1, $q'_2(0)$=-1, $q_3(0)$=-0.5, $q'_3(0)$=0, determine and graphically represent the variation of the kinematic coupling coordinates over the time interval [0,3] (seconds).

ANNEX M9. ELEMENTS REGARDING NUMERICAL INTEGRATION. ELEMENTS REGARDING THE SOLVING OF DIFFERENTIAL EQUATIONS AND OF SYSTEMS OF DIFFERENTIAL EQUATIONS

M9.1. Numerical integration

The problem of numerical integration is expressed as follows: considering a real function with a real variable $f : [a,b] \to \boldsymbol{R}$, integrable over the interval $[a,b]$. It is required to calculate the defined integral:

$$I_f = \int_a^b f(x)dx.$$

The numerical calculation methods of the above integral are usually based on the approximation of the function f by another function g,whose integral can be easily calculated. Next,two methods based on this technique will be presented: the trapezoidal method and Simpson's method.

Trapezoidal method

The interval on which the integral is calculated, $[a,b]$, is divided into n equal sections, the step and the current abscissa being:

$$h = \frac{b-a}{n}, \quad x_k = a + k \cdot h, \quad k = 0,1,\ldots,n$$

In the case of the trapezoidal method, the integral function f is approximated by a piecewise affine function g, having the property $g(x_k)=f(x_k)$, $k=0,1,\ldots,n$. The trapezoidal method consists of using the approximation formula:

$$\int_{x_k}^{x_{k+1}} f(x)dx \approx \int_{x_k}^{x_{k+1}} g(x)dx = \frac{[f(x_{k+1}) + f(x_k)] \cdot h}{2}$$

Finally, the following expression is obtained:

$$I_f = \frac{h}{2}\left[y_a + y_b + 2 \cdot \sum_{k=1}^{n-1} y_k \right],$$

where $y_k= f(x_k)$, $k=1,\ldots,n-1$, $y_a= f(a)$, $y_b= f(b)$.

Simpson's method

The interval on which the integral is calculated, $[a,b]$, is divided into $2 \cdot n$ equal sections, the step and the current abscissa being:

$$h = \frac{b-a}{2 \cdot n}, \quad x_k = a + k \cdot h, \quad k = 0,1,\ldots,2 \cdot n$$

In the case of ***Simpson's method***, the approximation of the function f is done with a piecewise square function g, having the property $g(x_k)=f(x_k)$, $k=0,1,\ldots,2 \cdot n$. Simpson's method consists of using the approximation formula:

$$\int_{x_{2\cdot k}}^{x_{2\cdot k+2}} f(x)dx \approx \int_{x_{2\cdot k}}^{x_{2\cdot k+2}} g(x)dx = \frac{h}{3}\cdot\left[f(x_{2\cdot k}) + 4\cdot f(x_{2\cdot k+1}) + f(x_{2\cdot k+2})\right]$$

Finally, the integral can be expressed by the following expression, known as ***Simpson's generalized formula***:

$$I_f = \frac{h}{3}\left[y_a + y_b + 4\cdot(y_1 + y_3 + \ldots + y_{2\cdot n-1}) + 2\cdot(y_2 + y_4 + \ldots + y_{2\cdot n-2})\right],$$

where $y_k= f(x_k)$, $k=1,\ldots,n-1$, $y_a= f(a)$, $y_b= f(b)$.

From the two methods presented, Simpson's method has a better accuracy than the trapezoidal method for the same number of sections of the interval $[a,b]$.

M9.2. Differential equations

The problem of solving a first-order differential equation with an initial condition is formulated as follows:

Given the differential equation:

$y'= f(x, y)$, $f : [a,b]\times I \to \mathbf{R}$, $[a, b], I \subset \mathbf{R}$

(I being an interval) and the initial condition:

$y(x_0) = y_0$, $x_0 = a$

determine the function $y : [a,b] \to \mathbf{R}$, $x\to y(x)$, which verifies the above relationships. The problem of solving a differential equation with an initial condition is also called a ***Cauchy problem***.

By using numerical methods to solve the problem, the obtained values are y_1, $y_2,\ldots, y_n$ that approximate the values $y(x_1)$, $y(x_2),\ldots, y(x_n)$ of the solution function y in a set of n points of the interval $[a,b]$, $x_1<x_2<\ldots<x_n$, $x_1 =a$, $x_n= b$.

Depending on the number of previously calculated points used to determine the current point (x_i,y_i), the numerical methods for solving differential equations are divided into two categories:

i) single step methods (also called separate steps methods) that use only the values corresponding to the preceding point (x_{i-1},y_{i-1});

ii) multistep methods(also called chained steps methods), which use the values corresponding to several predetermined points, (x_{i-1},y_{i-1}), $(x_{i-2}, y_{i-2}),\ldots$.

The Runge-Kutta method is one of the first category.

The second category includes ***the Adams-Bashforth-Moulton method***, various improved versions of the Runge-Kutta method, et al.

M9.3. Systems of differential equations

The problem of solving a system of n first-order differential equations with initial conditions is:

Given the system of differential equations:

$$\begin{cases} y_1' = f_1(x, y_1, y_2, \ldots, y_n) \\ y_2' = f_2(x, y_1, y_2, \ldots, y_n) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_n' = f_n(x, y_1, y_2, \ldots, y_n) \end{cases}$$

f_i:$[a,b]\times I_1 \times I_2 \times \ldots \times I_n \to \mathbf{R}$, $\quad i=1,2,\ldots,n$, $\quad [a, b], I_1, I_2 \ldots, I_n \subset \mathbf{R}$

(I_i being intervals, i=1,2,...,n) and the initial conditions:

$y_1(x_0) = y_{01}$, $y_2(x_0) = y_{02}$, ..., $y_n(x_0)=y_{0n}$, $\quad x_0 = a$

determine the functions $y_1 : [a,b] \to \mathbf{R}$, $x \to y_1(x)$, $y_2 : [a,b] \to \mathbf{R}$, $x \to y_2(x)$, ..., y_n: $[a,b] \to \mathbf{R}$, $x \to y_n(x)$, that verify the above relationships.

To solve the first-order differential equation systems with initial conditions we use numerical methods having as starting point the solving methods of the first order differential equations with initial conditions.

M9.4. Higher order differential equations. Systems of higher order differential equations

The problems of solving the higher order differential equations, respectively of the higher order differential equation systems, are similar to the corresponding problems formulated in points 2 and 3, with the addition of the initial conditions related to the higher-order derivatives of unknown-functions.

To solve these problems, the higher order differential equation, respectively the system of higher order differential equations, must be brought into the form of a system of first order differential equations.

Next, the bringing of a differential equation of order n into the form of a first order differential equation system is exemplified:

Having the differential equation of order n:

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

$$f:[a,b]\times I\times I_1\times I_2\times\ldots\times I_{n-1}\to\mathbf{R}, \qquad [a,b],I,I_1,I_2,\ldots,I_{n-1}\subset\mathbf{R}$$

and the initial conditions:

$$y'(x_0) = y_{01}, \quad y''(x_0) = y_{02};\ldots, \quad y^{(n-1)}(x_0) = y_{0n}, \quad x_0 = a$$

In order solve the problem, the following notations have been introduced: $yi = y^{(i)}, \quad i = 1,2,\ldots,n.$

It can be observed that $y_i' = \left(y^{(i)}\right)' = y^{(i+1)} = y_{i+1}$, $i = 1,2,\ldots,n-1$.Therefore, the differential equation of order n with initial conditions is equivalent to the following system of first order differential equations:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_{n-1}' = y_n \\ y_n' = f(x, y_1, y_2, \ldots, y_n) \end{cases}$$

with the initial conditions: $y_1(x_0) = y_{01}$, $y_2(x_0) = y_{02}$, ...,$y_n(x_0) = y_{0n}$.