Faculty of Civil Engineering — Faculty of Civil Engineering and Architecture
Belgrade — Niš
Master Study — Doctoral Study
COMPUTATIONAL ENGINEERING

LECTURES

LESSON X

10. Ordinary Differential Equations

10.1. Introduction

Problems involving ordinary differential equations (ODEs) can always be reduced to the set of first-order differential equations. For example the second order equation

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x) \tag{10.1.1}$$

can be rewritten as two first-order equations

$$\begin{aligned}\frac{dy}{dx} &= z(x)\\ \frac{dz}{dx} &= r(x) - q(x)z(x),\end{aligned} \tag{10.1.2}$$

where z is a new variable. This exemplifies the procedure for an arbitrary ODE. The usual choice for the new variables is to let them be just derivatives if each other, and, of course, of original variable. Occasionally, it is useful to incorporate into their definition some other factors in the equation, or some powers of the independent variable, for the purpose of the mitigating singular behavior that could result in overflows or increased roundoff error. Thus, involving new variables should be carefully chosen. The possibility of a formal reduction of a differential equation system to an equivalent set of first-order equations means that computer programs for the solution of differential equation sets can be directed toward the general form

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, \cdots, y_n) \qquad (i = 1, \ldots, n), \tag{10.1.3}$$

where the f_i functions are known and $y_1, y_2, \ldots, y_n$ are dependent variables.

A problem involving ODEs is not completely specified by its equations. Even more crucial in determining how to start solving problem numerically is the nature of the problem's boundary conditions. Boundary conditions are algebraic conditions on the values of the functions y_i in (10.1.3). Generally, they can be satisfied at discrete specified points, but do not hold between those points, i.e. are not preserved automatically by the differential equations. Boundary conditions can be as simple as requiring that certain variables have certain numerical values, or as complicated as a set of nonlinear algebraic equations among the variables. Usually, it is the nature of the boundary conditions that determines which numerical methods will be applied. Boundary conditions divide into two broad categories.

- **Initial value problems**, where all the y are given at some starting value x_s, and it is desired to find the the y_i's at some final point x_f, or at some discrete list of points (for example, to generate a table of results).

- **Two-point boundary value problems**, where the boundary conditions are specified at more than one x. Usually some conditions are specified at x_s and the remainder at x_f.

In considering methods for numerical solution of Cauchy problem for differential equations of first order, we will note two general classes of those methods:
a) Linear multi-step methods,
b) Runge-Kutta methods.

The first class of methods has a property of linearity, in contrary to Runge-Kutta methods, where the increasing of method order is realized by involving nonlinearity. The common "predecessor" of both classes is Euler's method, which belongs to both classes.

In newer times there appeared a whole series of methods, so known hybrid methods, which use good characteristics of mentioned basic classes of methods.

10.2. Euler's method

Euler's method is the simplest numerical method for solving Cauchy's problem

$$y' = f(x, y), \quad y(x_o) = y_0 \tag{10.2.1}$$

and is based on approximative equality

$$y(x) = y(x_0) + (x - x_0)y'(x_0),$$

i.e.

$$y(x) = y(x_0) + (x - x_0)f(x_0, y_0), \tag{10.2.2}$$

in regard to (10.2.1). If we denote with y_1 the approximate value for $y(x_1)$, based on (10.2.2) we have

$$y_1 = y_0 + (x_1 - x_0)f(x_0, y_0).$$

In general case, for arbitrary set of points $x_0 < x_1 < x_2 < \ldots$, the approximate values for $y(x_n)$, denoted as y_n, can be determined using

$$y_{n+1} = y_n + (x_{n+1} - x_n)f(x_n, y_n) \quad (n = 0, 1, \ldots). \tag{10.2.3}$$

The last formula defines Euler's method, which geometric interpretation is given in the Fig. 10.2.1.

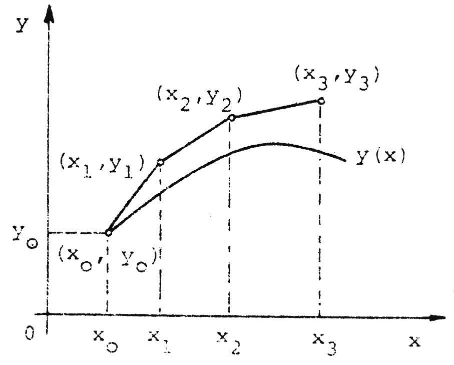

Figure 10.2.1

Polygonal line $(x_0, y_0) - (x_1, y_1) - (x_2, y_2) - \ldots$ is known as Euler's polygon.

The points x_n are usually chosen equidistantly, i.e. $x_{n+1} - x_n = h = const. (> 0)$ $(n = 0, 1, \ldots)$ in which case (10.2.3) reduces to

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (n = 0, 1, \ldots).$$

10.3. General linear multi-step method

In this and following sections a general method for solving Cauchy problem

$$y' = f(x, y), \quad y(x_0) = y_0 \quad (x_0 \le x \le b). \tag{10.3.1}$$

will be considered. If we divide the segment $[x_0, b]$ to N subsegments of length $h = \dfrac{b - x_0}{N}$, we get a string of points x_n determined with

$$x_n = x_0 + nh \qquad (n = 0, 1, \ldots, N).$$

Let y_n denotes sequence of approximate values of solutions of problem (10.3.1) in points x_n and let $f_n \equiv f(x_n, y_n)$. It is our task to determine a set y_n. In order to solve this problem a number of methods have been developed. One of them is Euler's method, which has been considered in previous section. At Euler's method series y_n is computed recursively using

$$y_{n+1} - y_n = hf_n \qquad (n = 0, 1, \ldots, N), \tag{10.3.2}$$

whereby the linear relation among y_n, y_{n+1} and f_n exists. In general case, for evaluation of series more complicated recurrence relations than (10.3.2) can be used. Among the methods originated from these relations, important role have the methods with linear relation between y_{n+i}, f_{n+i} $(i = 0, 1, \ldots k)$ and they form the class of linear multi-step methods.

General linear multi-step method can be represented in form

$$\sum_{i=0}^{k} \alpha_i y_{n+1} = h \sum_{i=0}^{k} \beta f_{n+i} \qquad (n = 0, 1, \ldots), \tag{10.3.3}$$

where α and β are constant coefficients determined by accuracy up to multiplicative constant. In order to obtain their uniqueness we will take $\alpha_k = 1$.

If $\beta_k = 0$, we say that method (10.3.3) is of open type or that is explicit; in counterpart we say that it is of closed type or implicit.

In general case (10.3.3) represents nonlinear difference equation, because of $f_{n+i} \equiv f(x_{n+i}, y_{n+i})$.

For determination of series y_n using method (10.3.3) it is necessary to know initial values y_i $(i = 0, 1, \ldots, k-1)$. Knowing in advance only value y_0, a particular problem in application of multi-step methods (10.3.3) is determination of other initial values. A special section will be devoted to this problem.

Supposing that initial values y_i $(i = 0, 1, \ldots, k-1)$ are known, at explicit methods are directly computed $y_k, y_{k+1}, \ldots, y_N$ using

$$y_{n+k} = h \sum_{i=0}^{k-1} \beta_i f_{n+i} - \sum_{i=0}^{k-1} \alpha_i y_{n+i} \qquad (n = 0, 1, \ldots, N-k).$$

Nevertheless, at implicit methods for determination of values y_{n+k} the equation

$$y_{n+k} = h\beta f(x_{n+k}, y_{n+k}) + \Phi, \tag{10.3.4}$$

where

$$\Phi = h\sum_{i=0}^{k-1}\beta_i f_{n+i} - \sum_{i=0}^{k-1}\alpha_i y_{n+i},$$

shell be solved. When $(x,y) \to f(x,y)$ nonlinear function which satisfies Lipschitz condition in y with constant L, the equation (10.3.4) can be solved by iterative process

(10.3.5) $$y_{n+k}^{[s+1]} = h\beta_k f(x_{n+k}, y_{n+k}^{[s]}) + \Phi,$$

starting from arbitrary value $y_{n+k}^{[0]}$ if

$$h|\beta_k|L < 1.$$

The condition given by this inequality ensures convergence of iterative process (10.3.5).

Let us for method (10.3.3) define difference operator $L_h : C^1[x_0,b] \to C[x_0,b]$ by

(10.3.6) $$L_h[y] = \sum_{i=0}^{k}[\alpha_i y(x+ih) - h\beta_i y'(x+ih)].$$

Let function $g \in C^1[x_0,b]$. Then $L_h[g]$ can be presented in form

(10.3.7) $$L_h[g] = C_0 g(x) + C_1 h g'(x) + C_2 h^2 g''(x) + \cdots,$$

where C_j $(j = 0,1,\ldots)$ are constants not depending on h and g.

Definition 10.3.1. *Linear multi-step method* (10.3.3) *is of order p if in development* (10.3.7)

$$C_0 = C_1 = \ldots = C_p = 0 \text{ and } C_{p+1} \neq 0.$$

If $x \to y(x)$ is exact solution of problem (10.3.1) and y_n series of approximate values of this solution in points $x_n = x_0 + nh$ $(n = 0,1,\ldots,N)$ obtained by method (10.3.3), with initial values $y_i = s_i(h)$ $(i = 0,1,\ldots,k-1)$.

Definition 10.3.2. *For linear multi-step method* (10.3.3) *one says to be convergent if for every* $x \in [x_0,b]$

$$\lim_{\substack{x\to 0\\ x-x_0=nh}} y_n = y(x)$$

and for initial values hold

$$\lim_{h\to 0} s_i(h) = y_0 \quad (i = 0,1,\ldots,k-1).$$

Linear multi-step method (10.3.3) can be characterized by first and second characteristic polynomials given by

$$\rho(\xi) = \sum_{i=0}^{k}\alpha_i\xi^i \quad \text{and} \quad \sigma(\xi) = \sum_{i=0}^{k}\beta_i\xi^i,$$

respectively.

Two important classes of convergent multi-step methods, which are met in practice are:

1. Methods at which $\rho(\xi) = \xi^k - \xi^{k-1}$;
2. Methods at which $\rho(\xi) = \xi^k - \xi^{k-2}$.

Explicit methods of first class are called Adam-Bashforth methods, and the implicit Adam-Moulton methods. Similarly, explicit methods of second class are called Nystrom's methods and corresponding implicit methods Milne-Simpson's.

Of course, there are methods that do not belong to neither of these classes.

10.4. Choice of initial values

As earlier mentioned, at application of linear multi-step methods on solving problem 10.3.1), it is necessary knowledge on initial values $y_i = s_i(h)$, such that

$$\lim_{h\to 0} s_i(h) = y_0 \quad (i = 1, \ldots, k-1).$$

Certainly, this problem is stated when $k > 1$.

If method (10.3.3) is of order p, then initial values $s_i(h)$ are obviously to be chosen such that

$$s_i(h) - y(x_i) = \mathbf{O}(h^{p+1}) \quad (i = 1, \ldots, k-1),$$

where $x \to y(x)$ is exact solution of problem (10.3.1).

In this section we will show one class of methods for determination of necessary initial values.

Suppose that function f in differential equation (10.3.1) is enough times differentiable. Than, based on Tailor's method we have

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2!}y''(x_0) + \cdots + \frac{h^p}{p!}y^{(p)}(x_0) + \mathbf{O}(h^{p+1}).$$

Last equation points out that it can be taken

$$s_i(h) = y(x_0) + hy'(x_0) + \frac{h^2}{2!}y''(x_0) + \cdots + \frac{h^p}{p!}y^{(p)}(x_0),$$

because of $s_i(h) - y(x_1) = O(h^{p+1})$ $(x_1 = x_0 + h)$. The same procedure can be applied to determination of other initial values. Namely, in general case, we have

$$s_i(h) = y(x_{i-1}) + hy'(x_{i-1}) + \frac{h^2}{2!}y''(x_{i-1}) + \cdots + \frac{h^p}{p!}y^{(p)}(x_{i-1}) \qquad (i = 1, \ldots, k-1),$$

whereby for $y(x_{i-1})$ we take $s_{i-1}(h)$.

10.5. Predictor-corrector methods

As mentioned in section 10.3.3, application of implicit methods is in connection with solution of equation (10.3.4) in every integration step, whereby for this solution is used iterative process (10.3.5). Regardless to this difficulty in implicit method, they are often used for solving of Cauchy problem, because they have a number of advantages over explicit methods. (higher order, better numerical stability). The initial value $y_{n+k}^{[0]}$ is determined in practice using some explicit method, which is then called predictor. Implicit method (10.3.4) is then called corrector. Method obtained by such combination is called predictor-corrector method.

For determination y_{n+k}, the iterative procedure (10.3.5) should be applied until fulfilment of the condition

$$|y_{n+k}^{[s+1]} - y_{n+k}^{[s]}| < \varepsilon,$$

where ε tolerable error, usually of order of local round-off error. Then for y_{n+k} can be taken $y_{n+k}^{[s+1]}$.

Nevertheless, this method is usually not applied in practice, due to demanding large number of function f evaluations by step of calculation and, in addition, this number is varying from step to step. In order to reduce this number of calculations, number of iterations in (10.3.5) is fixed. Thus, one takes only $s = 0, 1, \ldots, m-1$.

10.6. Program realization of multi-step methods

In this section we will give program realization of explicit as well as implicit methods. The presented programs are tested on the example (with $h = 0.1$).

$$y' = x^2 + y, \quad y(1) = 1 \quad (1 \leq x \leq 2).$$

The exact solution of this problem is $y(x) = 6e^{x-1} - x^2 - 2x - 2$.

10.6.1. Euler's method

Euler's method is given by expression

$$y_{n+1} - y_n = hf_n \quad (n = 0, 1, \ldots),$$

of order $p = 1$, and Adams-Bashforth method of third order

$$y_{n+3} - y_{n+2} = \frac{h}{12}(23f_{n+2} - 16f_{n+1} + 5f_n) \quad (n = 0, 1, \ldots),$$

are realized by subroutines `EULER` i `ADAMS`, respectively.

```
      SUBROUTINE EULER (XP,XK,H,Y,FUN)
      DIMENSION Y(1)
      N=(XK-XP+0.00001)/H
      X=XP
      DO 11 I=1,N
      Y(I+1)=Y(I)+H*FUN(X,Y(I))
   11 X=X+H
      RETURN
      END
C
      FUNCTION FUN (X,Y)
      FUN=X*X+Y
      RETURN
      END
C
      SUBROUTINE ADAMS (XP, XK, H, Y, FUN)
      DIMENSION Y(1)
      N=(XK-XP+0.00001)/H
      X=XP
      F0=FUN (X,Y(1))
      F1=FUN (X+H,Y(2))
      N2=N-2
      DO 11 I=1,N2
      F2=FUN(X+2.*H,Y(I+2))
      Y(I+3)=Y(I+2)+H*(23.*F2-16.*F1+5.*F0)/12.
      F0=F1
      F1=F2
   11 X=X+H
      RETURN
      END
```

Parameters in list of subroutine parameters are of following meaning:

`XP` and `XK` - start and end point of integration interval;

`H` - step of integration;

`Y` - vector of approximate values of solution obtained by multi-step method, whereby at Euler's method `Y(1)` represents given initial value, whereas at Adam's method initial values are given by `Y(1)`, `Y(2)` i `Y(3)`;

`FUN` - name of function subroutine which defines right hand size of differential equation $f(x, y)$. Initial values for Adam's method we determine by using Taylor's method for $p = 3$ (see section 10.3.4). Namely, being

$$y(1) = 1, \quad y'(1) = 2, \quad y''(1) = 4, \quad y'''(1) = 6, \quad h = 0.1,$$

we get `Y(1)=1.`, `Y(2)=1.221`, `Y(3)=1.48836`.

Main program and output listing are of form:

```
C
C==================================================
C      RESAVANJE DIFERENCIJALNIH JEDNACINA
C            EKSPLICITNIM METODIMA
C==================================================
      EXTERNAL FUN
      DIMENSION Y(100),Z(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      OPEN(5,FILE='EULER.OUT')
      WRITE (5,10)
   10 FORMAT(3X,'RESAVANJE DIFERENCIJAL.JED.',
     1'EKSPLICITNIM METODIMA'//8X,'XN',8X,'YN(I)',
     15X,'GRESKA(%)',3X,'YN(II)',4X,'GRESKA (%)'/)
      XP=1.
      XK=2.
      H=0.1
      Y(1)=1.
      CALL EULER (XP,XK,H,Y,FUN)
      Z(1)=Y(1)
      Z(2)=1.221
      Z(3)=1.48836
      CALL ADAMS (XP,XK,H,Z,FUN)
      N=(XK-XP+0.00001)/H
      NN=N+1
      X=XP
      DO 22 I=1,NN
      G1=ABS((Y(I)-F(X))/F(X))*100.
      G2=ABS((Z(I)-F(X))/F(X))*100.
      WRITE (5,20)X,Y(I),G1,Z(I),G2
   22 X=X+H
   20 FORMAT (8X,F3.1,2(4X,F9.5,4X,F5.2))
      CLOSE(5)
      STOP
      END
```

```
RESAVANJE DIFERENCIJAL.JED.EKSPLICITNIM METODIMA
XN        YN(I)     GRESKA(%)   YN(II)    GRESKA (%)
1.0      1.00000       .00     1.00000       .00
1.1      1.20000      1.72     1.22100       .00
1.2      1.44100      3.19     1.48836       .00
1.3      1.72910      4.42     1.80883       .02
1.4      2.07101      5.47     2.19028       .03
1.5      2.47411      6.37     2.64126       .04
1.6      2.94652      7.13     3.17116       .05
1.7      3.49717      7.79     3.79040       .06
1.8      4.13589      8.36     4.51045       .06
1.9      4.87348      8.87     5.34403       .07
2.0      5.72183      9.32     6.30518       .07
```

10.6.2. Taking Euler's method as predictor and trapeze rule ($p = 2$)

$$y_{n+1} - y_n = \frac{h}{2}(f_n + f_{n+1}) \quad (n = 0, 1, \ldots),$$

as corrector (with number of iterations $m = 2$) the subroutine `PREKOR` is written. Main program, subprogram, and output results are of form:

```
C==================================================
C     RESAVANJE DIF.JED. METODOM PREDIKTOR-KOREKTOR
C==================================================
      EXTERNAL FUN
      DIMENSION Y(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      OPEN(5,FILE='PREKOR.OUT')
      OPEN(8,FILE='PREKOR.TXT')
      WRITE(5,10)
 10   FORMAT(8X,'RESAVANJE DIF. JED. METODOM',
     1' PREDIKTOR-KOREKTOR'//15X,'XN',13X,'YN'
     2,10X,'GRESKA(%)'/)
      READ(8,5)XP,XK,YP,H
    5 FORMAT(4F6.1)
      CALL PREKOR(XP,XK,YP,H,Y,FUN)
      N=(XK-XP+0.00001)/H
      NN=N+1
      X=XP
      DO 11 I=1,NN
      G=ABS((Y(I)-F(X))/F(X))*100.
      WRITE(5,15)X,Y(I),G
   15 FORMAT(15X,F3.1,8X,F9.5,8X,F5.2)
   11 X=X+H
      STOP
      END
C
C
      SUBROUTINE PREKOR(XP,XK,YP,H,Y,FUN)
      DIMENSION Y(100)
      N=(XK-XP+0.00001)/H
      X=XP
      Y(1)=YP
      DO 10 I=1,N
C  PROGNOZIRANJE VREDNOSTI
      FXY=FUN(X,Y(I))
      YP=Y(I)+H*FXY
C  KOREKCIJA VREDNOSTI
      DO 20 M=1,2
   20 YP=Y(I)+H/2.*(FXY+FUN(X+H,YP))
      Y(I+1)=YP
   10 X=X+H
      RETURN
      END
C
C
      FUNCTION FUN(X,Y)
      FUN=X*X+Y
      RETURN
      END
```

```
RESAVANJE DIF. JED. METODOM PREDIKTOR-KOREKTOR
       XN             YN          GRESKA(%)
       1.0          1.00000           .00
       1.1          1.22152           .04
       1.2          1.48952           .07
       1.3          1.81097           .10
       1.4          2.19363           .12
       1.5          2.64602           .14
       1.6          3.17760           .15
       1.7          3.79881           .17
       1.8          4.52118           .18
       1.9          5.35747           .18
       2.0          6.32177           .19
```

10.7. Runge-Kutta methods

In previous sections are considered linear multi-step methods for solving Cauchy problem (10.3.1). The order of these methods can be enlarged by increasing number of steps. Nevertheless, by sacrifice of linearity these methods posses, it is possible to construct single-step methods of arbitrary order.

For solving Cauchy problem of form (10.3.1) with enough times differentiable function f, it is possible to construct single-step methods of higher order (e.g. Taylor's method).

Consider general explicit single-step method

$$y_{n+1} - y_n = h\Phi(x_n, y_n, h) \tag{10.7.1}$$

Definition 10.7.1. *Method* (10.7.1) *is of order p if p is greatest integer for which holds*

$$y(x+h) - y(x) - h\Phi(x, y(x), h) = \mathbf{O}(h^{p+1}),$$

where $x \to y(x)$ is exact solution of problem (10.3.1).

Definition 10.7.2. *Method* (10.7.1) *is consistent if $\Phi(x, y, 0) \equiv f(x, y)$.*

Note that Taylor's method is special case of method (10.7.1). Namely, at Taylor's method of order p we have

$$\Phi(x, y, h) = \Phi_T(x, y, h) = \sum_{i=0}^{p-1} \frac{h^i}{(i-1)!}(\frac{\delta}{\delta x} + f\frac{\delta}{\delta y})^i f(x, y). \tag{10.7.2}$$

In special case, at Eulerov's method is $\Phi(x, y, h) = f(x, y)$.

In this section we will consider a special class of methods of form (10.7.1), which was proposed in 1895. year by C. Runge. Later on, this class of methods of form (10.7.1) was developed by W. Kutta i K. Heun.

As we will see later, all these methods contain free parameters. Considering time in which these methods appeared, the free parameters have been chosen in such a way to obtain as simple as possible formulas for practical calculation. Nevertheless, such values of parameters do not ensure optimal characteristics of observed methods. In further text these methods will be called classical. General explicit Runge-Kutta method is of form

$$y_{n+1} - y_n = h\Phi(x_n, y_n, h) \tag{10.7.3}$$

where

$$\Phi(x, y, h) = \sum_{i=1}^{m} c_i k_i,$$

$$\begin{aligned} k_1 &= f(x, y), \\ k_i &= f(x + a_i, y + b_i h) \quad (i = 2, \ldots, m). \end{aligned}$$

$$a_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad b_i = \sum_{j=1}^{i-1} \alpha_{ij} k_j. \tag{10.7.4}$$

Note that from the condition of consistence of method (10.7.3) it follows

$$\sum_{i=1}^{m} c_i = 1.$$

Unknown coefficients which appear in this method are to be determined from the condition that method has a maximal order. Here, we use the following fact: If $\Phi(x, y, h)$, developed by degrees of h, can be presented in form

$$\Phi(x, y, h) = \Phi_T(x, y, h) = \mathbf{O}(h^p),$$

where Φ_T is defined by (10.7.2), then method (10.7.3) is of order p.

Find previously development $\Phi_T(x, y, h)$ by degrees of h. Using Monge's notations for partial derivative, we have

$$(\frac{\partial}{\partial x} + \frac{\partial}{\partial y}) = f_x + f f_y = F$$

and

$$(\frac{\partial}{\partial x} + f\frac{\partial}{\partial y})^2 = (\frac{\partial}{\partial x} + f\frac{\partial}{\partial y})F = G + f_yF,$$

where we put $G = f_{xx} + 2ff_{xy} + f^2 f_{yy}$. Then from (10.7.2) it follows

$$\Phi_T(x, y, h) = f + \frac{1}{2}hF + \frac{1}{6}h^2(G + f_yF) + \mathbf{O}(h^3). \tag{10.7.5}$$

Consider now only Runge-Kutta methods of order $p \le 3$. One shows that for obtaining method of third order it is enough to take $m = 3$. In this case, formulas (10.7.3) reduce to

$$\begin{aligned}\Phi(x, y, h) &= c_1k_1 + c_2k_2 + c_3k_3\\ k_1 &= f(x, y)\\ k_2 &= f(x + a_2h, y + b_2h),\\ k_3 &= f(x + a_3h, y + b_3h)\end{aligned}$$

and

$$\begin{aligned}a_2 &= \alpha_{21}, \quad b_2 = \alpha_{21}k_1,\\ a_3 &= \alpha_{31} + \alpha_{32}, \quad b_3 = \alpha_{31}k_1 + \alpha_{32}k_2.\end{aligned}$$

By developing of function k_2 in Taylor's series in neighborhood of point (x, y), we get

$$k_2 = f + a_2Fh + \frac{1}{2}a_2^2Gh^2 + \mathbf{O}(h^3).$$

Because of

$$b_3 = \alpha_{31}k_1 + \alpha_{32}k_2 = \alpha_{31}f + \alpha_{32}(f + a_2Fh + \frac{1}{2}a_2^2gh^2) + \mathbf{O}(h^3)$$

we have

$$b_3 = a_3f + a_2\alpha_{32}Fh + \mathbf{O}(h^2)$$

and

$$b_3^2 = a_3^2f^2 + \mathbf{O}(h).$$

By developing of function k_3 in neighborhood of point (x, y) and by using last equalities we have

$$k_3 = f + a_3Fh + \frac{1}{2}(2a_3\alpha_{32}Ff_y + a_3^2G)h^2 + \mathbf{O}(h^3).$$

Finally, by substituting the obtained expressions for k_1, k_2, k_3 in expression for $\Phi(x, y, h)$ we get

$$\begin{aligned}\Phi(x, y, h) =& (c_1 + c_2 + c_3)f + (c_2a_2 + c_3a_3)Fh\\ &+ (c_2a_2^2G + 2c_3a_2\alpha_{32}Ff_y + c_3a_3^2G)\frac{h^2}{2} + \mathbf{O}(h^3).\end{aligned}$$

Last equality enables construction of methods for $m = 1, 2, 3$.

Case m=1. Being $c_2 = c_3 = 0$, we have

$$\Phi(x, y, h) = c_1f + \mathbf{O}(h^3).$$

By comparison with (10.7.5) we get

$$\Phi_T(x, y, h) - \Phi(x, y, h) = (1 - c_1)f + \frac{1}{2}h^2(G + f_yF) + \mathbf{O}(h^3),$$

wherefrom we conclude that for $c_1 = 1$ the method

$$y_{n+1} - y_n = hf_n,$$

of order $p = 1$ is obtained. Considering that it is Euler's method, we see that it belongs to the class of Runge-Kutta methods too.

Case m=2. Here is $c_3 = 0$ and

$$\Phi_T(x, y, h) = (c_1 + c_2)f + c_2a_2Fh + \frac{1}{2}c_2a_2^2Gh^2 + \mathbf{O}(h^3).$$

Because of

$$\begin{aligned}\Phi(x, y, h) - \Phi_T(x, y, h) = (c_1 + c_2 - 1)f + (c_2a_2 - \frac{1}{2})Fh \\ + \frac{1}{6}[(3c_2a_2^2 - 1)G - f_yF]h^2 + \mathbf{O}(h^3),\end{aligned}$$

we conclude that under condition

$$c_1 + c_2 = 1 \quad \text{and} \quad c_2a_2 = \frac{1}{2}, \tag{10.7.6}$$

one obtains method of second order with one free parameter. Namely, from system of equations (10.7.6) it follows

$$c_2 = \frac{1}{2a_2} \quad \text{and} c_1 = \frac{2a_2 - 1}{2a_2},$$

where $a_2 (\neq 0)$ is free parameter. Thus, with $m = 2$ we have single-parametric family of methods

$$\begin{aligned}y_{n+1} - y_n &= \frac{h}{2a_2}((2a_2 - 1)k_1 + k_2),\\ k_1 &= f(x_n, y_n),\\ k_2 &= f(x_n + a_2h, y_n + a_2k_1h).\end{aligned}$$

In special case, for $a_2 = \frac{1}{2}$, we get Euler-Cauchy method

$$y_{n+1} - y_n = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(x_n, y_n)).$$

Similarly, for $a_2 = 1$, we get so known improved Euler-Cauchy method

$$y_{n+1} - y_n = \frac{h}{2}[f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n)].$$

On geometric interpretation of obtained methods see, e.g. [6].

Case m=3. According to

$$\begin{aligned}\Phi(x, y, h) - \Phi_T(x, y, h) = (c_1 + c_2 + c_3 - 1)f + (c_2a_2 + c_3a_3 - \frac{1}{2})Fh \\ + [(c_2a_2^2 + c_3a_3^2 - \frac{1}{3}G + (2c_3a_2\alpha_{32} - \frac{1}{3})Ff_y]\frac{h^2}{2} + \mathbf{O}(\frac{h}{3}),\end{aligned}$$

we conclude that for obtaining of methods of third order the satisfactory conditions are

$$\begin{aligned}c_1 + c_2 + c_3 &= 1,\\ c_2a_2 + c_3a_3 &= \frac{1}{2},\\ c_2a_2^2 + c_3a_3^2 &= \frac{1}{3},\\ c_3a_2\alpha_{32} &= \frac{1}{6}.\end{aligned} \tag{10.7.7}$$

Having four equations with six unknowns, it follows that, in case $m = 3$, we have two-parametric family of Runge-Kutta methods. One can show that among methods of this family does not exists not single method with order greater than three.

In special case, when $a_2 = \frac{1}{3}$ i $a_3 = \frac{2}{3}$, from (10.7.7) it follows $c_1 = \frac{1}{4}, c_2 = 0, c_3 = \frac{3}{4}, \alpha_{32} = \frac{2}{3}$ Thus, we obtained the method

$$\begin{aligned} y_{n+1} - y_n &= \frac{h}{4}(k_1 + 3k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{h}{3}, y_n + \frac{h}{3}k_1), \\ k_3 &= f(x_n + \frac{2h}{3}, y_n + \frac{h}{3}k_2), \end{aligned}$$

which is known in bibliography as Heun's method.

For $a_2 = \frac{1}{2}, a_3 = 1 (\Rightarrow c_1 = c_3 = \frac{1}{6}, c_2 = \frac{2}{3}, \alpha_{32} = 2)$ we get the method

$$\begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 4k_2 + k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \\ k_3 &= f(x_n + h, y_n - hk_1 + 2hk_2), \end{aligned}$$

which is most popular among the methods of third order from the point of view of hand calculations.

In case when $m = 4$, we get two-parameter family of methods of fourth order. Namely, here, analogously to system (10.7.7), appears system of 11 equations in 13 unknowns.

Now we quote, without proof, Runge-Kutta method of fourth order.

$$\begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \\ k_3 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2), \\ k_4 &= f(x_n + h, y_n + hk_3, \end{aligned} \tag{10.7.8}$$

which is traditionally most used in applications.

From methods of fourth order it is often used so known Gill's variant, which can be expressed as the following recursive procedure:

$$\begin{aligned} &n := 0, \quad Q_0 := 0 \\ (*)\ &Y_0 := y_n, \\ &k_1 := hf(x_n, Y_0), \quad Y_1 := Y_0 + \frac{1}{2}(k_1 - 2Q_0), \\ &Q_1 := Q_0 + \frac{3}{2}(k_1 - 2Q_0) - \frac{1}{2}k_1, \\ &k_2 := hf(x_n + \frac{h}{2}, Y_1), \quad Y_2 := Y_1 + (1 - \sqrt{1/2})(k_2 - Q_1), \\ &Q_2 := Q_1 + 3(1 - \sqrt{1/2})(k_2 - Q_1) - (1 - \sqrt{1/2})k_2, \\ &k_3 := hf(x_n + \frac{h}{2}, Y_2), \quad Y_3 := Y_2 + (1 + \sqrt{1/2})(k_3 - Q_2), \\ &Q_3 := Q_2 + 3(1 + \sqrt{1/2})(k_3 - Q_2) - (1 + \sqrt{1/2})k_3, \end{aligned}$$

$$k_4 := hf(x_n + h, Y_3), \quad Y_4 := Y_3 + \frac{1}{6}(k_4 - 2Q_3),$$
$$Q_0 := Q_3 + \frac{1}{2}(k_4 - 2Q_3) - \frac{1}{2}k_4,$$
$$y_{n+1} := Y_4,$$
$$n := n + 1$$
skip to $(*)$.

In contrary to linear multi-step methods, Runge-Kutta methods do not demand knowledge of initial values (except $y(x_0) = y_0$, what, by the way, defines Cauchy problem), but for practical application are more complicated, because they demand m calculations of function f values in every step.

10.8. Program realization of Runge-Kutta methods

In this section we present program realization of Euler-Cauchy method, improved Eueler-Cauchy method, as well as method of fourth order (10.7.8) and Gill's variant of Runge-Kutta method. The obtained software will be tested on the example from section 10.3.6.

Program 10.8.1.

By subroutine EULCAU are realized Euler-Cauchy and improved Euler-Cauchy method. Parameters in parameter list have the following meaning:

XP - start point of integration interval;

H - integration step;

N - integer, such that $N + 1$ is lenght of vector Y;

M - integer which defines way of construction of vector Y. Namely, in vector Y is stored in turn every M-th value of solution obtained during integration process.

Y - vector containing solutions of length N+1, whereby Y(1) is given initial condition y_0, Y(2) is value of solution obtained by integration in point XP + M*H, etc.

FUN - name of function subroutine, which defines right-hand side of differential equation $f(x, y)$;

K - integer with values K=1 and K=2, which governs integration according to Euler-Cauchy and improved Euler-Cauchy method, respectively.

Subroutine EULCAU is of form:

```
      SUBROUTINE EULCAU(XP,H,N,M,Y,FUN,K)
      DIMENSION Y(1)
      X=XP
      Y1=Y(1)
      NN=N+1
      DO 10 I=2,NN
      DO 20 J=1,M
      Y0=Y1
      Y1=FUN(X,Y0)
      GO TO (1,2),K
 1    Y1=Y0+H*FUN(X+0.5*H,Y0+0.5*H*Y1)
      GO TO 20
 2    Y1=Y0+H*(Y1+FUN(X+H,Y0+H*Y1))/2.
 20   X=X+H
10    Y(I)=Y1
      RETURN
      END
C
      FUNCTION FUN(X,Y)
      FUN=X*X+Y
      RETURN
      END
```

Main program and output listing are given in further text. As input parameters for integration we have taken `H=0.1, N=10, M=1`, and in second case `H=0.05, N=10, M=2`. Columns `Y1N` and `Y2N` in output listing give values for solution of given Cauchy problem, according to regular and improved Euler-Cauchy method, respectively. In addition to those columns, in output listing are given columns with corresponding errors (as relation to exact solution, expressed in %)

```
C=====================================================
C RESAVANJE DIF. JED. EULER-CAUCHYEVIM
C      I POBOLJSANIM METODOM
C=====================================================
      EXTERNAL FUN
      DIMENSION Y(100), Z(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      OPEN(5,FILE='EULCAU.OUT')
      OPEN(8,FILE='EULCAU.IN')
      WRITE(5,10)
10    FORMAT(10X,'RESAVANJE DIF.JED.EULER-CAUCHYEVIM'
     1 ' I POBOLJSANIM METODOM')
20    READ(8,25,END=99)XP,Y(1),H,N,M
25    FORMAT(3F6.1,2I3)
      CALL EULCAU(XP,H,N,M,Y,FUN,1)
      Z(1)=Y(1)
      CALL EULCAU(XP,H,N,M,Z,FUN,2)
      WRITE(5,30)H
30    FORMAT(1H0,30X,'(H=',F6.4,')'//15X,'XN',8X,
     1'Y1N',4X,'GRESKA(%)',5X,'Y2N',4X,'GRESKA(%)'/)
     NN=N+1
     X=XP
     DO 11 I=1,NN
     G1=ABS((Y(I)-F(X))/F(X))*100.
     G2=ABS((Z(I)-F(X))/F(X))*100.
     WRITE(5,15)X,Y(I),G1,Z(I),G2
15    FORMAT(15X,F3.1,3X,F9.6,2X,F7.5,3X,F9.6,2X,
     1 F7.5)
11    X=X+H*M
      GO TO 20
99    CLOSE(5)
      CLOSE(8)
      STOP
      END
```

```
          RESAVANJE DIF.JED.EULER-CAUCHYEVIM I POBOLJSANIM METODOM
0                               (H= .1000)
               XN        Y1N    GRESKA(%)     Y2N    GRESKA(%)
               1.0    1.000000   .00000    1.000000   .00000
               1.1    1.220250   .06352    1.220500   .04304
               1.2    1.486676   .11693    1.487203   .08157
               1.3    1.806227   .16173    1.807059   .11576
               1.4    2.186581   .19934    2.187750   .14599
               1.5    2.636222   .23109    2.637764   .17274
               1.6    3.164526   .25808    3.166479   .19650
               1.7    3.781851   .28125    3.784260   .21773
               1.8    4.499645   .30138    4.502557   .23685
               1.9    5.330558   .31907    5.334026   .25422
               2.0    6.288567   .33483    6.292649   .27013
0                               (H= .0500)
               XN        Y1N    GRESKA(%)     Y2N    GRESKA(%)
               1.0    1.000000   .00000    1.000000   .00000
               1.1    1.220824   .01655    1.220888   .01130
               1.2    1.487963   .03046    1.488098   .02140
               1.3    1.808391   .04213    1.808604   .03034
               1.4    2.189811   .05192    2.190111   .03824
               1.5    2.640738   .06019    2.641133   .04523
               1.6    3.170581   .06721    3.171082   .05143
               1.7    3.789740   .07324    3.790357   .05696
               1.8    4.509705   .07848    4.510451   .06195
               1.9    5.343177   .08309    5.344066   .06647
```

```
 2.0    6.304192   .08719    6.305238   .07061
```

Program 10.8.2.

According to formulas (10.7.8) for standard Runge-Kutta method of fourth degree, the following subroutine RK4 is written:

```
      SUBROUTINE RK4(X0,Y0,H,M,N,YVEK,F)
C==============================================
C     METOD RUNGE-KUTTA CETVRTOG REDA
C==============================================
      DIMENSION YVEK(1)
      T=H/2.
      X=X0
      Y=Y0
      DO 20 I=1,N
      DO 10 J=1,M
      A=F(X,Y)
      B=F(X+T,Y+T*A)
      C=F(X+T,Y+T*B)
      D=F(X+H,Y+H*C)
      X=X+H
 10   Y=Y+H/6.*(A+2.*B+2.*C+D)
 20   YVEK(I)=Y
      RETURN
      END
```

Parameters in list of subroutine parameters are of following meaning:
X0,Y0 - define given initial condition (Y0=y(X0);
H - step of integration;
M, N - integers with meanings similar to ones in subroutine EULCAU;
YVEK - vector of length N which is obtained as result of numerical integration, whereby Y(1) is value obtained in point X0+M*H, Y(2) value in point X0+2M*H, etc.
F - name of function subroutine which defines right-hand side of differential equation $f(x, y)$.
Main program is of form:

```
C===========================================
C    RESAVANJE DIF.JED. METODOM RUGE-KUTTA
C===========================================
      EXTERNAL FUN
      DIMENSION Y (100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      OPEN(5,FILE='RK4.OUT')
      OPEN(8,FILE='RK4.IN')
      WRITE(5,10)
 10   FORMAT (14X,'RESAVANJE DIF.JED. METODOM',
     1 ' RUNGE-KUTTA')
 20   READ (8,5,END=99)X0,Y0,H,N,M
 5    FORMAT (3F6.1,2I3)
      CALL RK4(X0,Y0,H,M,N,Y,FUN)
      G=0.
      WRITE (5,25) H,X0,Y0,G
25    FORMAT( 28X,'(H=',F6.4,')'//15X,'XN',13X,'YN',
     110X,'GRESKA(%)'//15X,F3.1,8X,F9.6,7X,F7.5)
      X=X0
      DO 11 I=1,N
      X=X+H*M
      G=ABS((Y(I)-F(X))/F(X))*100.
11    WRITE (5,15)X,Y(I),G
15    FORMAT (15X,F3.1,8X,F9.6,7X,F7.5)
      GO TO 20
99    CLOSE(5)
```

```
      CLOSE(8)
      STOP
      END
C
      FUNCTION FUN(X,Y)
      FUN=X*X+Y
      RETURN
      END
```

Taking `H=0.1`, `N=10`, `M=1` the following results are obtained:

```
RESAVANJE DIF.JED. METODOM RUNG-KUTTA
              (H= .1000)
 XN             YN          GRESKA(%)
 1.0         1.000000        .00000
 1.1         1.221025        .00002
 1.2         1.488416        .00005
 1.3         1.809152        .00007
 1.4         2.190946        .00009
 1.5         2.642325        .00011
 1.6         3.172709        .00012
 1.7         3.792512        .00014
 1.8         4.513240        .00015
 1.9         5.347611        .00017
 2.0         6.309682        .00018
```

Program 10.8.3.

The Gill's variant of Runge-Kutta method is realized in double precision. Parameters in parameter list of subroutine `GILL, X0, H, N, M, Y, FUN` have the same meaning as the parameters `HP, H, N, M, Y, FUN` in subroutine `EULCAU`, respectively. Note that this subroutine is realized in such a way that the optimization of number of variables has been performed.

Input parameters are taken like in program **10.8.1.**

```
C==================================================
C   RESAVANJE DIF.JED. METODOM RUNGE-KUTTA
C            (GILLOVA VARIJANTA)
C==================================================
      EXTERNAL FUN
      REAL*8 Y(100),F,FUN,X0,X,H,G
      F(X)=6.*DEXP(X-1.)-X*X-2.*X-2.
      OPEN(8,FILE='GILL.IN')
      OPEN(5,FILE='GILL.OUT')
      WRITE(5,10)
   10 FORMAT(8X,'RESAVANJE DIF.JED.METODOM'
     1' RUNGE-KUTTA (GILLOVA VARIJANTA)' )
   20 READ(8,25,END=99)X,Y(1),H,N,M
   25 FORMAT(3F6.1,2I3)
      X0=X
      CALL GILL(X0,H,N,M,Y,FUN)
      WRITE(5,30)H
   30 FORMAT(/28X,'(H=',F6.4,')'//15X,'XN',13X,'YN',
     1 10X,'GRESKA(%)'/)
      NN=N+1
      DO 11 I=1,NN
      G=DABS((Y(I)-F(X))/F(X))*100.
      WRITE(5,15)X,Y(I),G
   15 FORMAT(15X,F3.1,8X,F9.6,6X,D10.3)
   11 X=X+H*M
      GO TO 20
   99 CLOSE(5)
      CLOSE(8)
      STOP
      END
C
```

```
C
      SUBROUTINE GILL(X0,H,N,M,Y,FUN)
      REAL*8 Y(1),H,FUN,X0,Y0,Q,K,A,B
      B=DSQRT(0.5D0)
      Q=0.D0
      Y0=Y(1)
      NN=N+1
      DO 10 I=2,NN
      DO 20 J=1,M
      K=H*FUN(X0,Y0)
      A=0.5*(K-2.*Q)
      Y0=Y0+A
      Q=Q+3.*A-0.5*K
      K=H*FUN(X0+H/2.,Y0)
      A=(1.-B)*(K-Q)
      Y0=Y0+A
      Q=Q+3.*A-(1.-B)*K
      K=H*FUN(X0+H/2,Y0)
      A=(1.+B)*(K-Q)
      Y0=Y0+A
      Q=Q+3.*A-(1.+B)*K
      K=H*FUN(X0+H,Y0)
      A=(K-2.*Q)/6.
      Y0=Y0+A
      Q=Q+3.*A-K/2.
   20 X0=X0+H
   10 Y(I)=Y0
      RETURN
      END
C
      FUNCTION FUN(X,Y)
      REAL*8 FUN,X,Y
      FUN=X*X+Y
      RETURN
      END
```

```
RESAVANJE DIF.JED.METODOM RUNGE-KUTTA (GILLOVA VARIJANTA)
                   (H= .1000)
       XN             YN           GRESKA(%)
       1.0         1.000000         .000D+00
       1.1         1.221025         .246D-04
       1.2         1.488416         .460D-04
       1.3         1.809152         .647D-04
       1.4         2.190946         .808D-04
       1.5         2.642325         .949D-04
       1.6         3.172709         .107D-03
       1.7         3.792512         .118D-03
       1.8         4.513240         .128D-03
       1.9         5.347611         .136D-03
       2.0         6.309682         .144D-03
                   (H= .0500)
       XN             YN           GRESKA(%)
       1.0         1.000000         .000D+00
       1.1         1.221025         .162D-05
       1.2         1.488417         .303D-05
       1.3         1.809153         .425D-05
       1.4         2.190948         .531D-05
       1.5         2.642327         .623D-05
       1.6         3.172713         .704D-05
       1.7         3.792516         .775D-05
       1.8         4.513245         .838D-05
       1.9         5.347618         .894D-05
       2.0         6.309690         .946D-05
```

10.9. Solution of system of equations and equations of higher order

Methods considered in previous sections can be generalized in that sense to be applicable in solution of Cauchy problem for system of p equations of first order

$$y_i' = f_i(x; y_1, \ldots, y_p), \quad y_i(x_0) = y_{i0} \quad (i = 1, \ldots, p). \tag{10.9.1}$$

In this case, system of equations (10.9.1) shell be represented in vector form

$$\vec{y}\,' = \vec{f}(x, \vec{y}), \quad \vec{y}(x_0) = \vec{y}_0, \tag{10.9.2}$$

where

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \vec{y}_0 = \begin{bmatrix} y_{10} \\ y_{20} \\ \vdots \\ y_{p0} \end{bmatrix}, \quad \vec{f}(x, \vec{y}) = \begin{bmatrix} f_1(x; y_1, \ldots, y_p) \\ \vdots \\ f_p(x; y_1, \ldots, y_p) \end{bmatrix}.$$

It is of our interest the solution of Cauchy problem for differential equations of higher order. Note, nevertheless, that this problem can be reduced to previous one. Namely, let be given the differential equation of order p

$$y^{(p)} = f(x, y, y', \ldots, y^{(p-1)}) \tag{10.9.3}$$

with initial conditions

$$y^{(i)}(x_0) = y_{i0} \quad (i = 0, 1, \ldots, p-1). \tag{10.9.4}$$

Then, by substitution

$$z_1 = y, \quad z_2 = y', \ldots, z_p = y^{(p-1)},$$

equation (10.9.3) with conditions (10.9.4), reduces to system

$$\begin{aligned} z_1' &= z_2 \\ z_1' &= z_2 \\ &\vdots \\ z_{p-1}' &= z_m \\ z_p' &= f(x; z_1, z_2, \ldots, z_p), \end{aligned}$$

with conditions $z_i(x_0) = z_{i0} = y_{i0} \quad (i = 1, \ldots, p)$.

Linear multi-step methods considered up to now, can be formally generalized to vector form

$$\sum_{i=0}^{k} \alpha_i \vec{y}_{n+i} = h \sum_{i=0}^{k} \beta_i \vec{f}_{n+i},$$

where $\vec{f}_{n+i} = \vec{f}(x_{n+i}, \vec{y}_{n+i})$, and then as such can be applied to solution of Cauchy problem (10.9.2).

Also, the Runge-Kutta methods for solution of Cauchy problem (10.9.2) are of form

$$\vec{y}_{n+1} - \vec{y}_n = h\vec{\Phi}(x_n, \vec{y}_n, h),$$

where

$$\begin{aligned} &\vec{\psi}(x, \vec{y}, h) = \sum_{i=1}^{m} c_i \vec{k}_i, \\ &\vec{k}_1 = \vec{f}(x, \vec{y}), \\ &\vec{k}_i = \vec{f}(x + a_i h, \ \vec{y} + \vec{b}_i h) \\ &a_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad \vec{b}_i = \sum_{j=1}^{i-1} \alpha_{ij} \vec{k}_j \quad (i = 2, \ldots, m). \end{aligned}$$

All analyses given in previous sections can formally be translated to noted vector methods.

As an example, realize standard Runge-Kutta method of forth order (10.7.8) for solving of system of two differential equations

$$y' = f_1(x, y, z), \quad z' = f_2(x; y, z),$$

with conditions $y(x_0) = y_0$ and $z(x_0) = z_0$.

The corresponding subroutine is of form:

```
      SUBROUTINE RKS(XP,XKRAJ,YP,ZP,H,N,YY,ZZ)
      REAL KY1,KY2,KY3,KY4,KZ1,KZ2,KZ3,KZ4
      DIMENSION YY(1),ZZ(1)
      K=(XKRAJ-XP)/(H*FLOAT(N))
      N1=N+1
      X=XP
      Y=YP
      Z=ZP
      T=H/2.
      YY(1)=Y
      ZZ(1)=Z
      DO 6 I=2,N1
      DO 7 J=1,K
      KY1=FUN(1,X,Y,Z)
      KZ1=FUN(2,X,Y,Z)
      KY2=FUN(1,X+T,Y+T*KY1,Z+T*KZ1)
      KZ2=FUN(2,X+T,Y+T*KY1,Z+T*KZ1)
      KY3=FUN(1,X+T,Y+T*KY2,Z+T*KZ2)
      KZ3=FUN(2,X+T,Y+T*KY2,Z+T*KZ2)
      KY4=FUN(1,X+H,Y+H*KY3,Z+H*KZ3)
      KZ4=FUN(2,X+H,Y+H*KY3,Z+H*KZ3)
      Y=Y+H*(KY1+2.*(KY2+KY3)+KY4)/6.
      Z=Z+H*(KZ1+2.*(KZ2+KZ3)+KZ4)/6.
    7 X=X+H
      YY(I)=Y
    6 ZZ(I)=Z
      RETURN
      END
```

Using this subroutine we solved system of equations

$$y' = xyz, \quad z' = xy/z,$$

under conditions $y(1) = 1/3$ and $z(1) = 1$ on segment $[1, 2.5]$ taking for integration step $h = 0.01$, and printing on exit x with step 0.1 and corresponding values of y, y_T, z, z_T, where y_T and z_T are exact solutions of this system, given with

$$y_T = \frac{72}{(7 - x^2)^3} \quad \text{and} \quad z_T = \frac{6}{7 - x^2}.$$

The corresponding program and output listing are of form:

```
C==================================================
C   RESAVANJE SISTEMA DIF. JED. METODOM RUNGE-KUTTA
C==================================================
      DIMENSION YT(16),ZT(16),YY(16),ZZ(16),X(16)
      YEG(P)=72./(7.-P*P)**3
      ZEG(P)=6./(7.-P*P)
      OPEN(8,FILE='RKS.IN')
      OPEN(5,FILE='RKS.OUT')
      READ(8,15)N,XP,YP,ZP,XKRAJ
15    FORMAT(I2,4F3.1)
      YP=YP/3.
      H=0.1
      N1=N+1
      DO 5 I=1,N1
      X(I)=XP+H*FLOAT(I-1)
      YT(I)=YEG(X(I))
    5 ZT(I)=ZEG(X(I))
      WRITE(5,22)
      H=0.01
      CALL RKS(XP,XKRAJ,YP,ZP,H,N,YY,ZZ)
      WRITE(5,18)H,(X(I),YY(I),YT(I),ZZ(I),ZT(I),
     1      I=1,N1)
   18 FORMAT(//7X,'KORAK INTEGRACIJE H=',F6.3//7X,
     1'X',11X,'Y',10X,'TACNO',11X,'Z',10X,'ZTACNO'//
     2(F10.2,4F14.7))
   22 FORMAT(1H1,9X,'RESAVANJE SISTEMA SIMULTANIH',
     1' DIFERENCIJALNIH JEDNACINA'//33X,'Y''=XYZ'//
     1 33X, 'Z''=XY/Z')
      CLOSE(5)
      CLOSE(8)
      STOP
      END
C
      FUNCTION FUN(J,X,Y,Z)
      GO TO (50,60),J
   50 FUN=X*Y*Z
      RETURN
   60 FUN=X*Y/Z
      RETURN
      END
```

```
1          RESAVANJE SISTEMA SIMULTANIHDIFERENCIJALNIH JEDNACINA
                                  Y'=XYZ
                                  Z'=XY/Z
         KORAK INTEGRACIJE H=  .010
         X           Y          TACNO           Z          ZTACNO
        1.00      .3333333      .3333333     1.0000000     1.0000000
        1.10      .3709342      .3709342     1.0362690     1.0362690
        1.20      .4188979      .4188979     1.0791370     1.0791370
        1.30      .4808936      .4808935     1.1299430     1.1299430
        1.40      .5623943      .5623943     1.1904760     1.1904760
        1.50      .6718181      .6718181     1.2631580     1.2631580
        1.60      .8225902      .8225904     1.3513510     1.3513510
        1.70     1.0370670     1.0370680     1.4598540     1.4598540
        1.80     1.3544680     1.3544680     1.5957440     1.5957450
        1.90     1.8481330     1.8481340     1.7699110     1.7699110
        2.00     2.6666650     2.6666670     2.0000000     2.0000000
        2.10     4.1441250     4.1441260     2.3166020     2.3166020
        2.20     7.1444800     7.1444920     2.7777760     2.7777780
        2.30    14.3993600    14.3993900     3.5087690     3.5087710
        2.40    37.7628900    37.7631300     4.8387000     4.8387110
        2.50   170.6632000   170.6667000     7.9999230     8.0000000
```

10.10. Contour problems

In this section we will point out to difference method for solution contour problem

$$y'' + p(x)y' + q(x)y = f(x); \quad y(a) = A, \ y(b) = B, \tag{10.10.1}$$

where functions p, q, f are continuous on $[a, b]$.

Let us divide segment $[a, b]$ to $N + 1$ subsegments of length $h = \dfrac{b-a}{N+1}$, so that $x_n = a + nh$ $(n = 0, 1, ..., N + 1)$. In points x_n $(n = 1, \ldots, N)$ we approximate the differential equation from (10.10.1) with

$$\frac{y_{n+1} - 2y_n + Y_{n-1}}{h^2} + p_n \frac{y_{n+1} - y_{n-1}}{2h} + q_n y_n = f_n \qquad (n = 1, \ldots, N), \tag{10.10.2}$$

where $p_n \equiv p(x_n)$, $q_n \equiv q(x_n)$, $f_n \equiv f(x_n)$.

If we involve substitutions

$$a_n = 1 - \frac{h}{2}p_n,\ b_n = h^2 q_n - 2,\ c_n = 1 + \frac{h}{2}p_n,$$

(10.10.2) can be represented in form

$$a_n y_{n-1} + b_n y_n + c_n y_{n+1} = h^2 f_n \quad (n = 1, \ldots, N). \tag{10.10.3}$$

In regards to contour conditions $y_0 = A$ and $Y_{N+1} = B$, we have the problem of solving the system of linear equations $\mathbf{T}\vec{y} = \vec{d}$, where

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \vec{d} = \begin{bmatrix} h^2 f_1 - Aa_1 \\ h^2 f_2 \\ \vdots \\ h^2 f_N - Bc_N \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} b_1 & c_1 & 0 & \ldots & 0 \\ a_2 & b_2 & c_2 & 0 & \\ \vdots & & & & \\ 0 & 0 & 0 & \ldots & b_N \end{bmatrix}.$$

System matrix is tri-diagonal. For solving of this system it is convenient to perform decomposition of matrix **T** as **T**=**LR** (see chapter 2), whereby the problem is reduced to successive solution of two triangular systems of linear equations. This procedure for solution contour problem (10.10.1), is known as matrix factorization.

The following program is written in accordance to explained procedure.

```
       DIMENSION A(100),B(100),C(100),D(100)
C===================================================
C  MATRICNA FAKTORIZACIJA ZA RESAVANJE
C  KONTURNIH PROBLEMA KOD  LINEARNIH
C  DIFERENCIJALNIH JEDNACINA II REDA
C  Y''+ P(X)Y'+ Q(X)Y = F(X)
C  Y(DG) = YA, Y(GG) = YB
C ==================================================
      OPEN(8,FILE='KONTUR.IN')
      OPEN(7,FILE='KONTUR.OUT')
      READ(8,5) DG,YA,GG,YB
 5    FORMAT(4F10.5)
C UCITAVANJE BROJA MEDJUTACAKA
 10   WRITE(*,14)
 14   FORMAT(1X,'UNETI BROJ MEDJUTACAKA'
     1' U FORMATU I2'/ 5X,'(ZA N=0 => KRAJ)')
      READ(5,15) N
 15   FORMAT(I2)
      N1=N+1
      IF(N.EQ.0) GO TO 60
      H=(GG-DG)/FLOAT(N1)
      HH=H*H
      X=DG
      DO 20 I=1,N
      X=X+H
      Y=H/2.*PQF(X,1)
      A(I)=1.-Y
      C(I)=1.+Y
      B(I)=HH*PQF(X,2)-2.
 20   D(I)=HH*PQF(X,3)
      D(1)=D(1)-YA*A(1)
      D(N)=D(N)-YB*C(N)
      C(1)=C(1)/B(1)
      DO 25 I=2,N
      B(I)=B(I)-A(I)*C(I-1)
 25   C(I)=C(I)/B(I)
      D(1)=D(1)/B(1)
      DO 30 I=2,N
 30   D(I)=(D(I)-A(I)*D(I-1))/B(I)
      NM=N-1
      DO 35 I=1,NM
      J=NM-I+1
```

```
   35   D(J)=D(J)-C(J)*D(J+1)
        WRITE(7,40)N,(I,I=1,N1)
   40   FORMAT(///5X,'BROJ MEDJUTACAKA N='
      1 ,I3//5X,'I',6X,'0',9I10)
        DO 45 I=1,N
        C(I)=DG+H*FLOAT(I)
   45   B(I)=PQF(C(I),4)
        WRITE(7,50)DG,(C(I),I=1,N),GG
        WRITE(7,55)YA,(D(I),I=1,N),YB
        WRITE(7,65)YA,(B(I),I=1,N),YB
   50   FORMAT(/5X,'X(I)',10(F6.2,4X))
   55   FORMAT(/5X,'Y(I)',10F10.6)
   65   FORMAT(/5X,'YEGZ',10F10.6)
        GO TO 10
   60   CLOSE(7)
        CLOSE(8)
        STOP
        END
```

Note that this program is so realized that number of inner points N is read on input. In case when $N = 0$ program ends. Also, in program is foreseen tabulating of exact solution in observing points, as control. It is clear that last has meaning for scholastic examples when solution is known. So, for example, for contour problem

$$y'' - 2xy' - 2y = -4x; \quad y(0) = 1, \; y(1) = 1 + e \cong 3.7182818,$$

the exact solution is $y = x + exp(x^2)$.

For this contour problem function subroutine for defining functions p, q, f, as for as for exact solution, is named `PQF`. In case `N=4`, we got the results given in continuation.

```
        FUNCTION PQF(X,M)
        GO TO (10,20,30,40),M
   10   PQF=-2.*X
        RETURN
   20   PQF=-2.
        RETURN
   30   PQF=-4.*X
        RETURN
   40   PQF=X+EXP(X*X)
        RETURN
        END
```

```
 BROJ MEDJUTACAKA N=  4
  I     0         1         2         3         4         5
X(I)  .00       .20       .40       .60       .80      1.00
Y(I) 1.000000 1.243014 1.576530 2.035572 2.695769 3.711828
YEGZ 1.000000 1.240811 1.573511 2.033329 2.696481 3.711828
```

10.11. Packages for ODEs

Numerous libraries and software packages are available for integrating initial-value ordinary differential equations. Many work stations and main frame computers have such libraries attached to their operating systems.

Many commercial software packages contain routines for integrating initial-value ODEs. One of the oldest and very known among senior scientist is SSP (Scientific Subroutine Package) of IBM. For ODEs it has subroutines `RK1` (integral of first-order differential equation by Runge-Kutta method), `RK2` (integral of first-order differential equation by Runge-Kutta method in tabulated form) using in both subroutines fourth order Runge-Kutta method, and `RKGS` (solution of system of first-order differential equations with given initial values by the Runge-Kutta method) using evaluation by means of fourth order Runge-Kutta formulae in the modification due to Gill. Some of the more prominent packages are `Matlab` and `Mathcad`. More sophisticated packages, such as IMSL, Mathematica, and Macsyma contain also algorithms for integrating initial-value ODEs. The book *Numerical Recipes*([12]) contains numerous subroutines for integrating initial-value ordinary differential equations and the book *Numerical Methods for Engineers and Scientists*([3]) program code for solving single first-order ODEs, higher order ODEs, and systems of first-order ODEs, by using single-point methods, extrapolation methods, and multi-point methods (see Chapter 7, One-Dimensional Initial-Value Ordinary Differential Equations).

Bibliography (Cited references and further reading)

[1] Milovanović, G.V., *Numerical Analysis III*, Naučna knjiga, Beograd, 1988 (Serbian).

[2] Milovanović, G.V., *Numerical Analysis I*, Naučna knjiga, Beograd, 1988 (Serbian).

[3] Hoffman, J.D., *Numerical Methods for Engineers and Scientists.* Taylor & Francis, Boca Raton-London-New York-Singapore, 2001.

[4] Milovanović, G.V. and Djordjević, Dj.R., *Programiranje numeričkih metoda na FORTRAN jeziku.* Institut za dokumentaciju zaštite na radu "Edvard Kardelj", Niš, 1981 (Serbian).

[5] Milovanović, G.V., *Numerical Analysis II*, Naučna knjiga, Beograd, 1988 (Serbian).

[6] Bertolino, M., *Numerička analiza*, Beograd, 1977.

[7] Runge, C., *Über die numerische Auflösung von Differentialgleichungen*, Math. Ann. 46(1985), 167-178.

[8] Kutta, W., *Beitrag zur Näherungsweisen Integration totaler Differentialgleichungen*, Z. Math. Phys. 46(1901), 435-453.

[9] Neun, K., *Neue Methode zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen*, Z. Math. Phys. 45(1900), 23-38.

[10] Gill, S., *A Process for the step-by-step integration of differential equations in an automatic computing machine*, Proc. Cambridge Phil. Soc. 47(1951), 96-108.

[11] Stoer, J., and Bulirsch, R., *Introduction to Numerical Analysis*, Springer, New York, 1980.

[12] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., *Numerical Recepies - The Art of Scientific Computing.* Cambridge University Press, 1989.

[13] Milovanović, G.V. and Kovačević, M.A., *Zbirka rešenih zadataka iz numeričke analize.* Naučna knjiga, Beograd, 1985. (Serbian).

[14] Ralston,A., *A First Course in Numerical Analysis.* McGraw-Hill, New York, 1965.

[15] Hildebrand, F.B., *Introduction to Numerical Analysis.* Mc.Graw-Hill, New York, 1974.

[16] Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[17] Acton, F.S., *Numerical Methods That Work* (corrected edition). Mathematical Association of America, Washington, D.C., 1990.

[18] Lambert, J., *Computational Methods in Ordinary Differential Equations*, Wiley, New York, 1973.

[19] Lapidus, L., and Seinfeld, J., *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York, 1971.

[20] Abramowitz, M., and Stegun, I.A., *Handbook of Mathematical Functions.* National Bureau of Standards, Applied Mathematics Series, Washington, 1964 (reprinted 1968 by Dover Publications, New York).

[21] Rice, J.R., *Numerical Methods, Software, and Analysis.* McGraw-Hill, New York, 1983.

[22] Forsythe, G.E., Malcolm, M.A., and Moler, C.B., *Computer Methods for Mathematical Computations.* Englewood Cliffs, Prentice-Hall, NJ, 1977.

[23] Kahaner, D., Moler, C., and Nash, S., 1989, *Numerical Methods and Software.* Englewood Cliffs, Prentice Hall, NJ, 1989.

[24] Hamming, R.W., *Numerical Methods for Engineers and Scientists.* Dover, New York, 1962 (reprinted 1986).

[25] *IMSL Math/Library Users Manual* , IMSL Inc., 2500 City West Boulevard, Houston TX 77042

[26] *NAG Fortran Library*, Numerical Algorithms Group, 256 Banbury Road, Oxford OX27DE, U.K., Chapter F02.