PROGRAM ROZWOJOWY
POLITECHNIKI WARSZAWSKIEJ

LECTURE 5

INTRODUCTION TO NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS (INITIAL VALUE PROBLEMS)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

The aim of this lecture is to provide an elementary introduction to the numerical methods designated for the initial value problems (IVP) formulated for the ordinary differential equations (ODE) and their systems. This is the class of the computational tasks which are particularly common in the engineering practice. The reason is twofold:

- The ordinary differential equations are used to describe the time evolution of various physical systems with lumped parameters (mechanical, electric, electronic, chemical, etc.)
- Most of the ODE encountered in practice cannot be solved analytically

Consider the initial value problem formulated for the single 1^{st} -order differential equation. The standard form of such problem is

$$\begin{cases} y'(t) = F[t, y(t)] \\ y(t_0) = y_0 \end{cases}$$

where the function F and the initial value y_0 are given. If F does not depend explicitly on time, i.e. $F = F(y)$, then the system is called **autonomous**.

We will not discuss here the mathematical results pertaining existence and uniqueness of a solution to this problem. Instead we will just point out to the possible problem by showing representative counterexamples.

First, let us remind that the problem may have multiply solutions. The standard example is

$$\begin{cases} y'(t) = 3y^{2/3}(t) \\ y(0) = 0 \end{cases}$$

Which is solved by two different functions, namely

$$Y_1(t) \equiv 0 \quad , \quad Y_2(t) = t^3$$

The reason the solution is not unique is that the right-hand side function is not Lipschitz-continuous at $y = 0$.

Next example shows that the solution may not exist for arbitrarily long time interval. Consider

$$\begin{cases} y'(t) = y^2(t) \\ y(0) = 1 \end{cases}$$

The exact solution is $Y(t) = \dfrac{1}{1-t}$ and we see that $\lim\limits_{t \to 1^-} Y(t) = +\infty$ i.e., the solution "blows up" in the finite time $t = 1$.

NUMERICAL METHODS – THE GENERAL CONCEPTS

In the numerical analysis of differential equations, time is discretized, i.e. approximate solutions are determined only for the finite (but usually large) number of discrete time instants. The distance between two subsequent time instants is called the **time step** (we will denote it by *h*). An approximate solution is determined by means of a certain **integration scheme.** Such scheme is simply the recipe explaining how to obtain the solution at the next time instant $t_{k+1} = t_k + h$, providing that up to the current time $t = t_k$ the solution is already known.

The **general form of the integration scheme** for our standard problem can be writes as follows

$$y_{k+1} = y_k + G[t_{k-(n-1)}, y_{k-(n-1)}; t_{k-(n-2)}, y_{k-(n-2)}; \ldots; t_k, y_k; t_{k+1}, y_{k+1}]$$

In the above formula, the symbols y_j, $j = k-(n-1), \ldots, k+1$ denotes the approximate value of the solution at the respective time instants. The function *G* may not be directly given as an explicit formula – it rather symbolizes a certain numerical recipe for determination of the solution increment in the time interval $[t_k, t_{k+1}]$.

In general, this recipe may be **multi-step**, i.e. it may use the solution from several previous time instants (the number n can be larger than 1). Obviously, only the **single-step** schemes (with $n = 1$) are **self-starting** (the initial condition is imposed for only one point in time).

The integration schemes can be **explicit** (open) or **implicit** (closed). In explicit schemes, the function G does not depend on y_{k+1} (and t_{k+1}), hence the formula for y_{k+1} is direct (explicit). In implicit schemes, the list of the arguments of the function G includes y_{k+1} (if the function F depends explicitly on time - also t_{k+1}). Thus, the formula is indirect, i.e., we deal with some kind of an algebraic (and usually nonlinear) problem, which has to be solved with respect to y_{k+1}.

THE EULER SCHEME. CONSISTENCE, STABILITY AND CONVERGENCE.

The simplest possible integration scheme is the **Euler method**. It can be formally derive from the Taylor expansion theorem. Assume that $y = Y(t)$ is the exact solution of our IVP and let Y has continuous derivatives up to at least second order. Then, the one can write the equality

$$Y(t+h) = Y(t) + hY'(t) + \tfrac{1}{2}h^2 Y''(t+\theta\cdot h) \ , \ \theta \in (0,1)$$

For sufficiently small time increment (or time step) *h* the quadratic term can be dropped and we arrive at the following formula

$$y_{k+1} = y_k + h\underbrace{F[t_k, y_k]}_{F_k} = y_k + hF_k$$

where y_k, y_{k+1} are approximate values of $Y(t_k), Y(t_k + h)$, respectively.

What is the error made by this scheme during one time step? More precisely: what is the difference between the exact and numerical solutions at the time $t_k + h$, if we assume that both solutions ideally coincide at the time t_k ?

This difference – called the **local approximation error** (**LAE**) of the scheme – is the most basic characterization of any numerical scheme. Obviously, we demand that the LAE of each reasonable integration scheme shrinks to zero while the time step $h \to 0$. In other words, any integration scheme should be **consistent**.

The consistency of the Euler scheme can be easily established. From the derivation of this scheme it is clear that LAE can is equal

$$e(t_k, h) \equiv Y(t_{k+1}) - \hat{y}_{k+1} = \tfrac{1}{2} h^2 Y''(t_k + \theta \cdot h)$$

where $\hat{y}_{k+1} = Y(t_k) + hF[t_k, Y(t_k)]$. Clearly, $\lim_{h \to 0} e(t_k, h) = 0$.

An important characteristic of an integration scheme is its **order of accuracy**. We say that the **order of the integration scheme is equal *r*** if and only if the LAE of this scheme can be represented by the Taylor expansion as

$$e(h) = Ch^{r+1} + o(h^{r+1}) \quad , \quad C \neq 0 \quad , \quad \lim_{h \to 0} \frac{o(h^{r+1})}{h^{r+1}} = 0,$$

i.e., the leading term in the Taylor expansion contain the time step *h* in the power *r* +1.

We conclude immediately that **the Euler method is the 1st -order scheme**.

Yet another essential issue is the numerical **stability** of the integration schemes. This problem is rather complicated and its more detailed analysis will be postponed to more advance 2^{nd} – year course "Numerical methods". Still, the essence of the stability concept can be explained on the following example.

Consider the following test problem: $y'(t) = \lambda y(t)$, $y(0) = Y_0$, where $\lambda = \lambda_R + i\lambda_I$ is the given complex number. The exact solution is

$$Y(t) = Y_0 \exp(\lambda t) = Y_0 e^{\lambda_R t}(\cos \lambda_I t + i \sin \lambda_I t)$$

Note also that if $\lambda_R < 0$ then $\lim_{t\to\infty} |Y(t)| = 0$.

Let us apply the Euler scheme to the test problem. The numerical solution is the sequence of the complex numbers obtained from the recurrent formula

$$y_{k+1} = y_k + hF_k = y_k + h\lambda y_k = y_k(1 + h\lambda)$$

One can easily infer the general formula, namely

$$y_k = (1 + h\lambda)^k Y_0$$

Question arises: when does this sequence properly follow the asymptotic behavior of the true (i.e., exact) solution?

If $\lambda_R < 0$ then this sequence should be bounded and approach zero while $k \to \infty$. This will happen only when $|1 + h\lambda| < 1$, i.e., when the complex number $h\lambda$ is located inside the circle $|z - (-1,0)| = 1$. If $\lambda \in R$ and $\lambda = -\mu < 0$ then the **stability condition** is

$$-1 < 1 - h\mu < 1 \Leftrightarrow h < \frac{2}{\mu}$$

Note also that in such case the exact solution approaches zero monotonically. The numerical solution will share this property only when $0 < 1 - h\mu < 1$, i.e., if $h < 1/\mu$. We say that the **Euler scheme is conditionally stable, i.e., it is stable for sufficiently small time step**.

Consider now the time interval of the prescribed length T. More precisely, let $t \in (t_0, t_0 + T)$. Next, divide this interval into *n* subintervals (time steps). It is sufficient to assume that all intervals have the same length $h = T/n$.

The **global approximation error (GAE)** of an integration scheme is defined as the difference between exact and numerical solutions at the time $t_0 + T$:

$$E(n,h) = Y(t_0 + nh) - y_n$$

Of course, it is assumed that at initial time t_0 $y_0 \equiv Y(t_0)$.

We say that the **integration scheme is convergent** if and only if $\lim_{\substack{n\to\infty \\ nh=T}} E(n,T) = 0$.

The convergence implies that for each value of *T* we can found a sufficiently small time step *h* such that the discrepancy between exact and numerical solution after time *t* = *T* can be made arbitrarily small. Or – putting things yet another and more practical way – the accumulation of the local errors during integration procedure can be kept under control.

It can be shown that the **Euler integration scheme is convergent**. If we consider the test problem, then this conclusion is immediate as

$$y_n = (1+h\lambda)^n Y_0 = (1+\tfrac{\lambda T}{n})^n Y_0 \xrightarrow[n\to\infty]{} Y_0 e^{\lambda T} = Y(T)$$

In general case, the proof relies of the assumption that the right-hand side function F is Lipschitz-continuous with respect to the second argument, i.e., the following estimate holds

$$\exists L > 0, \forall t \in I, \forall y_1, y_2 \in R : \left|F(t, y_1) - F(t, y_2)\right| \le L \cdot \left|y_1 - y_2\right|$$

Let $\hat{y}_n$ denotes the value of the numerical solution which the Euler scheme would yield if $y_{n-1} \equiv Y(t_{n-1})$. Then the GAE can be expressed as

$$E_n := \underbrace{Y_n - y_n}_{e(h)-LAE} = (Y_n - \hat{y}_n) + (\hat{y}_n - y_n) = e(h) + \underbrace{Y_{n-1} + hF[t_{n-1}, Y_{n-1}]}_{\hat{y}_n} - \underbrace{\{y_{n-1} + hF[t_{n-1}, y_{n-1}]\}}_{y_n}$$

Then, using the Lipschitz continuity of F, we obtain the estimate

$$\left|E_n\right| \le \left|e(h)\right| + (1+hL)\left|E_{n-1}\right|$$

By recursion on n, we find

$$\left|E_n\right| \le [1+(1+hL)+..+(1+hL)^{n-1}]e(h) = \frac{(1+hL)^n - 1}{hL} \underbrace{he(h)}_{\tau(h)} \le \frac{e^{L(t_n - t_0)} - 1}{L}\tau(h)$$

Note that for the Euler method the quantity $\tau(h) = \frac{1}{2}Y''(\xi)h$, $\xi \in (t_{n-1}, t_n)$. Hence, we can write

$$\left|E_n\right| \le \frac{e^{L(t_n - t_0)} - 1}{L}\frac{M}{2}h \quad , \quad \forall n \ge 0$$

where $M = \max_{\xi \in [t_0, t_n]} \left|Y''(\xi)\right|$. It follows that the GAE tends to zero as the integration step h shrinks to zero – the **Euler method is convergent**.

The following example show that **consistency alone is not sufficient for convergence**!

Consider the following integration scheme

$$y_{k+1} = 4y_k - 3y_{k-1} - 2hF(t_k, y_k)$$

This scheme is consistent with the order equal 1. To see this, we compare the exact power series expansion of the solution

$$Y(t_{n+1}) = Y(t_n) + hY'(t_n) + \tfrac{1}{2}h^2Y''(t_n) + o(h^2)$$

with the approximate expansion implied by the numerical formula …

$$Y_{k+1} \approx 4Y_k - 3Y_{k-1} - 2hY'_k = 4Y_k - 3(Y_k - hY'_k + \tfrac{1}{2}h^2Y''_k - \ldots) - 2hY'_k =$$
$$= Y + hY'_k - \tfrac{3}{2}h^2Y''_k + o(h^2)$$

Hence, the exact and approximate expansions agree up to the linear term meaning that the proposed scheme is of the 1st order.

Let us apply this scheme to the test problem

$$\begin{cases} y'(t) = -y(t) \\ y(0) = 1 \end{cases}$$

which has the exact solution $Y(t) = e^{-t}$.

The application our 1st-order scheme yields the sequence of values which satisfy the following recurrent relation

$$y_{k+1} = 4y_k - 3y_{k-1} - 2h(-y_k) = (4 + 2h)y_k - 3y_{k-1}$$

We want to fins an explicit formula for the numerical solution. To this aim, we have to solve the following **linear difference** (not differential !) **equation**

$$y_{k+1} - (4 + 2h)y_k + 3y_{k-1} = 0\,,\ k = 1,2,..$$

We will use the following starting conditions based on the exact solution

$$y_0 = 1\,,\ y_1 = e^{-h}$$

In order to find the analytical solution, we assume that

$$y_k = s^k$$

where the number s is to be found. Insertion of this formula to the difference equation yields the following quadratic equation for s

$$s^2 - (4 + 2h)s + 3 = 0$$

which has two roots

$$s_1 = 2 + h - \sqrt{1 + 4h + h^2}) \ , \ s_2 = 2 + h + \sqrt{1 + 4h + h^2}$$

Using a power series expansion $\sqrt{1+\varepsilon} = 1 + \frac{1}{2}\varepsilon + O(\varepsilon^2)$, for we obtain for $h \ll 1$

$$s_1 = 1 - h + O(h^2) \, , \ s_2 = 3(1 + h) + O(h^2)$$

The solution of the finite difference equation is $y_k = C_1 s_1^k + C_2 s_2^k$

The coefficients C_1 and C_2 are chosen such that the starting conditions are satisfied. Note that typically both of them are different than zero.

Now, we claim that the proposed schemed is in fact completely useless. Indeed, the Reader should note that the second root s_2 is never smaller than 3, thus with k the second term is the solution will rise without limits, which is in clear contradiction to the behavior of true solution. In other words, $\lim_{k\to\infty}|y_k| = \infty$ no matter how small we choose the time step h, i.e. our scheme is **unconditionally unstable**.

Summarizing, any reasonable integration scheme has to be consistent and also stable. The latter property means essentially that the scheme should not generate artificial unbounded components of the approximate solution. Usually, the numerical schemes are stable only when the time integration step h is sufficiently small, meaning that they are **conditionally stable**. Some schemes (as a rule, they are implicit) can be unconditionally stable, i.e. they bring qualitatively correct behavior of approximate solution with arbitrary large time steps. The convergence to the exact solution can be obtained only when a numerical scheme is both consistent and stable:

Consistence + Stability = Convergence

The basic example of an unconditionally stable scheme is the **Implicit Euler method**.

The formula of this method is

$$y_{k+1} = y_k + h\underbrace{F[t_{k+1}, y_{k+1}]}_{F_{k+1}} = y_{k+1} + hF_{k+1}$$

The following properties hold (the **proof is left to the Reader** as an exercise!):

- The implicit Euler method is the 1st-order scheme
- The implicit Euler method is unconditionally stable scheme (hint: stability the stability analysis using as before the test problem $y'(t) = \lambda y(t)$, $y(0) = Y_0$. You should arrive at the conclusion that the sequence $\{|y_0|, |y_1|, .., |y_k|, ..\}$approaches monotonically zero, no matter how large is the time step h).

Other unconditionally stable scheme is the **trapezoidal method** (known also as the **Crank-Nicholson scheme**). The formula of this scheme is

$$y_{k+1} = y_k + \tfrac{1}{2}h(F_k + F_{k+1})$$

Clearly, the above formula is the "arithmetic average" of the explicit and implicit Euler formulae.

It is, again, lest to the Reader to show that:

- The trapezoidal method is 2^{nd}-order accurate.
- It is unconditionally stable, however it can produce **artificially oscillating numerical solution**s if the time step h is not sufficiently small (consider the test problem with the real and negative value of λ.

CONSTRUCTION OF THE HIGHER-ORDER INTEGRATION SCHEMES

We will discuss shortly the issue of a design of higher-order integration schemes.

The natural approach is the application of the higher-order Taylor expansion. We will show how this approach works for 2^{nd} –order schemes. We begin with the Taylor's expansion formula

$$Y(t+h) = Y(t) + hY'(t) + \tfrac{1}{2}h^2 Y''(t) + O(h^3)$$

The function $Y = Y(t)$ is the solution of the differential equation. Hence, one can express two first derivatives of Y in terms of the given function F and its derivatives as follows

$$Y'(t) = F[t, Y(t)]$$
$$Y''(t) = \partial_t F[t, Y(t)] + \partial_y F[t, Y(t)] \cdot Y'(t) = \partial_t F[t, Y(t)] + \partial_y F[t, Y(t)] \cdot F[t, Y(t)]$$

After insertion, we obtain the **generic 2^{nd} –order integration scheme**

$$y(t_k + h) = y(t_k) + hF(t_k, y_k) + \tfrac{1}{2}h^2 (\partial_t F + \partial_y F \cdot F)(t_k, y_k)$$

The above procedure can be extended easily to the higher-order schemes. However, such approach **is not practical** because it involves higher-order partial derivatives of *F*, which can be difficult (sometimes impossible) for explicit evaluation. We would prefer to deal some alternative schemes which use only the values of *F* but not its derivatives!

Such alternative approach to construct higher-order schemes which are convenient for practical use is exemplified by the **Runge-Kutta family** of methods. The general procedure is pretty complicated – here we discuss the derivation of the 2nd-order methods.

The general formula of the 2nd-order Runge-Kutta method is

$$\begin{cases} y(t_k + h) = y(t_k) + w_1 K_1 + w_2 K_2 \\ K_1 = hF(t_k, y_k) \\ K_2 = hF[t_k + \alpha h, y_k + \beta K_1] \end{cases}$$

We need to find such parameters w_1, w_2, α and β that the above formula is of the second order. To this aim, we will transform the above formula using the Taylor expansions to the form which can be directly compared to the generic 2nd -order scheme.

The calculations go as follows …

$$Y(t_k + h) = Y(t_k) + w_1 hF[t_k, Y(t_k)] + w_2 hF\{t_k + \alpha h, Y(t_k) + \beta hF[t_k, Y(t_k)]\} =$$
$$= Y(t_k) + w_1 hF[t_k, Y(t_k)] + w_2 h\{F[t_k, Y(t_k)] + \partial_t F[t_k, Y(t_k)] \cdot \alpha h +$$
$$+ \partial_y F[t_k, Y(t_k)] \cdot \beta hF[t_k, Y(t_k)] + O(h^2)\}$$

After a simple algebra we get

$$Y(t_k + h) = Y(t_k) + (w_1 + w_2) hF[t_k, Y(t_k)] + h^2 (w_2 \alpha \partial_t F + w_2 \beta \partial_y F \cdot F)[t_k, Y(t_k)] + O(h^3)$$

The obtained formula is equivalent to the generic 2nd-order scheme only if

$$w_1 + w_2 = 1 \ , \ \alpha = \beta \ , \ \alpha w_2 = \tfrac{1}{2}$$

We have obtained 3 conditions for 4 parameters – the solution is not unique. In fact, we have infinitely many 2nd-order methods of this sort.

Two most popular variants are:

The Heun method: $w_1 = w_2 = \frac{1}{2}$, $\alpha = \beta = 1$

$$\begin{cases} y(t_k + h) = y(t_k) + \frac{1}{2}(K_1 + K_2) \\ K_1 = hF(t_k, y_k) \\ K_2 = hF[t_k + h, y_k + K_1] \end{cases}$$

The modified Cauchy-Euler: $w_1 = 0$, $w_2 = 1$, $\alpha = \beta = \frac{1}{2}$

$$\begin{cases} y(t_k + h) = y(t_k) + K_2 \\ K_1 = hF(t_k, y_k) \\ K_2 = hF[t_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1] \end{cases}$$

Higher-order methods can be derived in the similar manner. The calculation are, however, rather complicated. For instance, in order to design a 4^{th} –order method one has to determine 13 coefficients. Again, the choice is not unique as the number of equation to satisfy is “only” 11.

The most popular variant is the **RK4 scheme** defined follows

$$y(t_k + h) = y(t_k) + \tfrac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$K_1 = hF(t_k, y_k)$$

$$K_2 = hF[t_k + \tfrac{1}{2}h, y_k + \tfrac{1}{2}K_1]$$

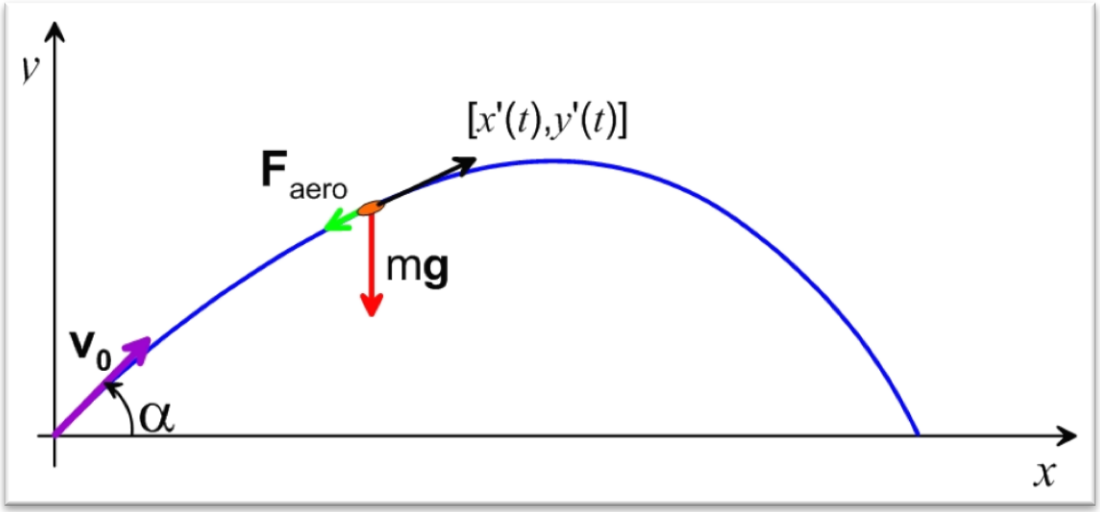
$$K_3 = hF[t_k + \tfrac{1}{2}h, y_k + \tfrac{1}{2}K_2]$$

$$K_4 = hF[t_k + h, y_k + K_3]$$

GENERALIZATION FOR THE SYSTEMS OF ODES

The numerical integration schemes design for the single 1st-order ordinary differential equation can be easily extended for the systems of such equations. All we have to do is to replace the scalar quantities by the vector ones. But the real issue is that in practice we often obtain differential system which are not in the standard form, i.e., this are not the systems of the 1st-order equations solved with respect to the first derivatives of the unknown functions.

Consider the following example. The simple two-dimensional model of the ballistic flight in the uniform air and gravity field can be written as



$$\begin{cases} mx'' = -cx'\sqrt{x'^2 + y'^2} \\ my'' = -mg - cy'\sqrt{x'^2 + y'^2} \end{cases}$$

In the above, m stands for the mass of the projectile, c is the coefficient of the aerodynamic drag and g is the gravity acceleration. The position of the projectile is identified by means of two coordinates: the horizontal x and the vertical y.

The initial condition for the ballistic flight of the projectile are

$$\begin{cases} x(0)=0 \ , \ y(0)=0 \\ x'(0)=u_0\cos(\alpha) \ , \ y'(0)=u_0\sin(\alpha) \end{cases}$$

where u_0 is the initial velocity and α is the angle of the initial inclination of the trajectory with respect to the ground.

The above system is not in the standard form, however it can be transformed to such. To this end, we defined a set of new functions

$$z_1(t)=x(t)\,,\ z_2(t)=x'(t)$$
$$z_3(t)=y(t)\,,\ z_4(t)=y'(t)$$

Then, the following standard system is equivalent to the original one (here $k=c/m$) …

$$\begin{cases} z_1' = z_2 \\ z_2' = -k\, z_2 \sqrt{z_2^2 + z_4^2} \\ z_3' = z_4 \\ z_4' = -g - k\, z_4 \sqrt{z_2^2 + z_4^2} \end{cases}$$

The above system can be also written conveniently in the vector form $\boldsymbol{z}'(t) = \boldsymbol{F}[t, \boldsymbol{z}(t)]$, where the vector right-hand side function **F** is

$$\boldsymbol{F}[t, \boldsymbol{z}(t)] \equiv \begin{cases} f_1(t, z_1, .., z_4) = z_2 \\ f_2(t, z_1, .., z_4) = -k\, z_2 \sqrt{z_2^2 + z_4^2} \\ f_3(t, z_1, .., z_4) = z_4 \\ f_3(t, z_1, .., z_4) = -k\, z_4 \sqrt{z_2^2 + z_4^2} \end{cases}$$

Finally, also the initial conditions are transformed as well

$$z_1(0) = 0 \;\; , \;\; z_2(0) = u_0 \cos(\alpha) \;\; , \;\; z_3(0) = 0 \;\; , \;\; z_4(0) = u_0 \sin(\alpha)$$

In order to deal with the standard differential systems the numerical schemes are re-formulated to the vector forms.
For instance, we have:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{F}[t_k, \boldsymbol{y}_k] \qquad 1^{st} - order\ Euler$$

$$\begin{cases} \boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \frac{1}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2) \\ \boldsymbol{k}_1 = h\boldsymbol{F}[t_k, \boldsymbol{y}_k] \\ \boldsymbol{k}_2 = h\boldsymbol{F}[t_k + h, \boldsymbol{y}_k + \boldsymbol{k}_1] \end{cases} \qquad 2^{nd} - order\ Heun\ (RK2)$$

$$\begin{cases} \boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \\ \boldsymbol{k}_1 = h\boldsymbol{F}[t_k, \boldsymbol{y}_k] \\ \boldsymbol{k}_2 = h\boldsymbol{F}[t_k + \frac{1}{2}h, \boldsymbol{y}_k + \frac{1}{2}\boldsymbol{k}_1] \\ \boldsymbol{k}_3 = h\boldsymbol{F}[t_k + \frac{1}{2}h, \boldsymbol{y}_k + \frac{1}{2}\boldsymbol{k}_2] \\ \boldsymbol{k}_4 = h\boldsymbol{F}[t_k + h, \boldsymbol{y}_k + \boldsymbol{k}_3] \end{cases} \qquad 4^{th} - order\ Runge - Kutta\ (RK4)$$

In practice, “raw” forms of the high-order Runge-Kutta methods are seldom in use. Usually, well design numerical codes use some kind of the **time step adaption** and **error control**. It means that the magnitude of the **time step** *h* **is automatically tuned** in the integration process. In such case, some **strategy of the step adjustment** is necessary.

Typically, such strategy can be based on the comparison between numerical solutions obtained (for the same time instant) by the same scheme, but with different time steps. For example, one can use the 4^{th} –order Runge-Kutta scheme for the time step *h* and then compared it with the solution obtained when two steps with the length *h*/2 are performed instead. However, much more efficient way is to compare approximate solutions obtained by two schemes of different order and such that the **number of evaluations of derivatives** (i.e., references to the right-hand side function $\boldsymbol{F}$) is as small as possible.

The example of such approach is the **Runge-Kutta-Fehlberg algorithm**.

It uses six $\boldsymbol{F}$ evaluations in each integration step. Then, a pair of the numerical solutions are detrermined for the time instant $t = t_{k+1}$:

- y_{k+1} - by means if the 4^{th}-order formula,
- $\hat{y}_{k+1}$ - by means of the 5^{th} –order formula.

If these two solutions do not agree to a specified accuracy, the step size is reduced by the factor s. It the agreement is better than required, the time step is increased (s >1).

The complete set of the formulae describing the Runge-Kutta-Fehlberg algorithm (often referred to as **RKF45**) can be written as follows:

$$\begin{cases} \boldsymbol{k}_1 = h\boldsymbol{F}[t_k, \boldsymbol{y}_k] \\ \boldsymbol{k}_2 = h\boldsymbol{F}[t_k + \frac{1}{4}h, \boldsymbol{y}_k + \frac{1}{4}\boldsymbol{k}_1] \\ \boldsymbol{k}_3 = h\boldsymbol{F}[t_k + \frac{3}{8}h, \boldsymbol{y}_k + \frac{3}{32}\boldsymbol{k}_1 + \frac{9}{32}\boldsymbol{k}_2] \\ \boldsymbol{k}_4 = h\boldsymbol{F}[t_k + \frac{12}{13}h, \boldsymbol{y}_k + \frac{1932}{2197}\boldsymbol{k}_1 - \frac{7200}{2197}\boldsymbol{k}_2 + \frac{7296}{2197}\boldsymbol{k}_3] \\ \boldsymbol{k}_5 = h\boldsymbol{F}[t_k + h, \boldsymbol{y}_k + \frac{439}{216}\boldsymbol{k}_1 - 8\boldsymbol{k}_2 + \frac{3680}{513}\boldsymbol{k}_3 - \frac{845}{4104}\boldsymbol{k}_4] \\ \boldsymbol{k}_6 = h\boldsymbol{F}[t_k + \frac{1}{2}h, \boldsymbol{y}_k - \frac{8}{27}\boldsymbol{k}_1 + 2\boldsymbol{k}_2 - \frac{3544}{2565}\boldsymbol{k}_3 + \frac{1859}{4104}\boldsymbol{k}_4 - \frac{11}{40}\boldsymbol{k}_5] \end{cases}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \frac{25}{216}\boldsymbol{k}_1 + \frac{1408}{2565}\boldsymbol{k}_3 + \frac{2197}{4101}\boldsymbol{k}_4 - \frac{1}{5}\boldsymbol{k}_5$$

$$\hat{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_k + \frac{16}{35}\boldsymbol{k}_1 + \frac{6656}{12825}\boldsymbol{k}_3 + \frac{28561}{56430}\boldsymbol{k}_4 - \frac{9}{50}\boldsymbol{k}_5 + \frac{2}{52}\boldsymbol{k}_6$$

$$h_{opt} = sh \quad , \quad s = \sqrt[4]{\frac{\delta_h}{2\left\|\hat{\boldsymbol{y}}_{k+1} - \boldsymbol{y}_{k+1}\right\|}}$$

More recent and popular algorithms, belonging to so called “embedded” Runge-Kutta schemes , are:

- Bogacki-Shampine (order 2/ order 3 embedded pair) – implemented in MATLAB as *ode23* command,
- Dorman-Prince (order 4/ order 5 embedded pair) - implemented in MATLAB as *ode45* command.

Final remarks:

- Alternative approach to achieve higher-order methods (and step control) – the multistep methods (to be discussed in the “Numerical Methods” course). The **multistep methods** achieve higher-order accuracy by using the “history” of the right-hand side function **F**. These methods can be explicit or implicit. Smart combination of both leads to the predictor-corrector methods. Also, automatic step adaption may be incorporated, although it may be rather tricky.
- Special implicit methods are design to deal with so called **stiff systems**. These are the systems where very different characteristic time scales coexist in the system dynamic response. Typical example: coupled oscillation of the elements which flexibility properties are very much different (think about a big mass attached to a bending beam by means of a pliant spring). The problem of stiffness and the design of appropriate methods is among the topics of the “Numerical Methods” course.