SLOVENSKÁ POĽNOHOSPODÁRSKA UNIVERZITA
V NITRE
TECHNICKÁ FAKULTA

RIEŠENIE DIFERENCIÁLNYCH ROVNÍC NUMERICKÝMI METÓDAMI RUNGE - KUTTA

Diplomová práca

Študijný program:	Informačná a automatizačná technika v kvalite produkcie
Študijný odbor:	2386800 Kvalita produkcie
Školiace pracovisko:	Katedra konštruovania strojov
Školiteľ:	doc. Ing. Jozef Rédl, PhD.
Konzultant:	prof. Ing. Dušan Hrubý, PhD.

Nitra, 2011 **Veronika Váliková, Bc**

Čestné vyhlásenie

Podpísaná Bc. Veronika Váliková vyhlasujem, že som záverečnú prácu na tému „Riešenie diferenciálnych rovníc numerickými metódami Runge - Kutta“ vypracovala samostatne s použitím uvedenej literatúry.

Som si vedomá zákonných dôsledkov v prípade, ak uvedené údaje nie sú pravdivé.

V Nitre 20. apríla 2011

Bc. Veronika Váliková

Poďakovanie

Touto cestou by som sa chcela poďakovať všetkým, ktorí mi akýmkoľvek spôsobom pomohli a podporili ma pri spracovaní tejto diplomovej práce, ako aj počas celého vysokoškolského štúdia. Moje poďakovanie patrí hlavne vedúcemu diplomovej práce doc. Ing. Jozefovi Rédlovi, PhD. za odborné vedenie, pripomienky, návrhy a pomoc pri jej vypracovaní. Osobitné poďakovanie patrí mojej rodine a priateľom za ich neustálu morálnu podporu a porozumenie.

Abstrakt

Práca sa zaoberá problematikou riešenia diferenciálnych rovníc numerickými metódami Runge – Kutta. Cieľom je vytvoriť prehľad spomenutých numerických metód, ktoré sa využívajú v oblasti vedecko – technickej aplikačnej činnosti, zvoliť vhodné metódy Runge – Kutta a implementovať ich do počítačového programu prostredníctvom vývojového prostredia Delphi. Následne uviesť príklady riešenia diferenciálnych rovníc z vybranej oblasti použitia, výsledky riešenia porovnať s riešením vo vybranom CAD/CAM produkte a taktiež s presným riešením. V teoretickej časti práce je vysvetlený pojem diferenciálnej rovnice. Opísané sú základné metódy na jej riešenie, ako exaktné tak i približné. V práci je zahrnuté rozdelenie numerických metód Runge – Kutta na klasické metódy a na metódy vyšších rádov, ako aj odhad chýb metód Runge – Kutta. V poslednej podkapitole je predstavené vývojové prostredie Delphi, v ktorom sa bude aplikácia vytvárať. Praktická časť práce sa venuje naprogramovaniu aplikácie, ktorá implementuje dve metódy Runge – Kutta, a to klasickú metódu a metódu Runge – Kutta – Gill. Táto časť sa venuje opisu aplikácie a jej použitiu. Opísané sú použité údajové typy, úvodná obrazovka aplikácie, zadávanie vstupných hodnôt a taktiež menu s príslušnými funkciami. Ďalšia kapitola sa venuje technickej aplikácii, kde sa pomocou vytvoreného programu riešia diferenciálne rovnice získané z funkcie ohybových momentov zaťažených nosníkov. Sú uvedené dva druhy nosníkov, kde riešenie získané aplikáciou je porovnané s presným riešením. Uvedený je aj príklad výpočtu uhlu sklonu v programe MathCad, ktoré je porovnané s výsledkami aplikácie. Zistenia, ktoré vyplynuli z výsledkov sú zosumarizované v závere.

Kľúčové slová: diferenciálne rovnice, numerické metódy, metódy Runge – Kutta, numerická integrácia, Delphi.

Abstract

This work deals with solution of differential equations with Runge – Kutta numerical methods. The goal of this work is to create an overview of mentioned numerical methods, which are used in scientific - technical application activity, select the appropriate methods of Runge - Kutta and implement them into a computer program through Delphi development environment. Subsequently provide examples of differential equations of the selected field of application, the results of solutions given by application compare with solution of selected CAD / CAM product and with the exact solution, too. The theoretical part explains the concept of differential equations. There are described basic methods for its solution, exact methods as well as the approximate. The work includes the distribution of numerical methods Runge - Kutta on classical methods and methods of a higher order, as well as the estimation errors of Runge – Kutta methods. In the last subsection is presented the development environment Delphi in which is the application created. The practical part is devoted to programming of an application that implements two methods of Runge – Kutta, classical method and method Runge - Kutta - Gill. This section is devoted to the description of the application and its use. There are described used data types, splash screen, entering an input values and also menu with the appropriate functions. The next chapter is dedicated to technical applications, where there is created program used to solve differential equations obtained from function of bending moments of loaded beams. There are given two types of beams, where the solution given by application is compared with exact solution. There is also an example of calculating the angle of deflection line in the program MathCad. The findings which follow on the results are summarized in the conclusion.

Keywords: differential equations, numerical methods, Runge – Kutta methods, numerical integration, Delphi.

Obsah

Zoznam skratiek a značiek

L	dĺžka nosníka
E	modul pružnosti v ťahu tlaku
φ	uhol sklonu dotyčnice voči priehybovej čiare
M_{xi}	ohybový moment pre jednotlivé myslené rezy
J_Z	moment zotrvačnosti k osi z
F	pôsobiaca sila
q	spojité bremeno
SI	Medzinárodná sústava jednotiek (*Système International (d'Unités)*)

Úvod

Pre dnešnú dobu je charakteristický nástup výkonných počítačov s kvalitným hardvérovým aj softvérovým vybavením. Vďaka rôznym moderným technologickým vymoženostiam máme možnosť pristupovať k zložitým úlohám, ktorých riešenie bolo v minulosti komplikované a niekedy takmer nemožné. Vedné odbory matematika a informatika spolu veľmi úzko súvisia, a práve informačné prostriedky nám pomáhajú urýchliť výpočty náročných príkladov, ktorých riešenie by bez nich bolo časovo veľmi náročné. V tejto práci sú prepojené poznatky práve z týchto dvoch vedných odborov.

Na modelovanie javov z oblasti vedy a techniky sa najčastejšie využívajú diferenciálne rovnice ktoré môžeme zaradiť do časti matematiky nazývanej matematická analýza. V matematickej analýze sú predmetom skúmania funkcie a operácie s nimi. V technickej praxi ako aj vo vedecko – výskumnej činnosti sa často stretávame s rôznymi meraniami a procesmi, kedy treba spracovať práve experimentálne získané funkcie.

Keďže niekedy nie je možné nájsť analytické riešenie niektorých diferenciálnych rovníc, existujú aj numerické metódy, ktoré sú na ich riešenie vhodné. V tejto práci sa venujeme riešeniu diferenciálnych rovníc numerickými metódami Runge – Kutta. Patria medzi približné metódy a zaraďujú sa medzi jednokrokové metódy založené na Taylorovom rozvoji funkcie. Sú univerzálne a v technickej praxi veľmi užitočné.

Cieľom tejto práce je vytvoriť prehľad o súčasnom stave využitia metód Runge – Kutta pri riešení diferenciálnych rovníc. Tejto časti je venovaná prvá kapitola. V tejto kapitole sa čitateľ oboznámi aj s vývojovým prostredím *Delphi* firmy *Borland,* v ktorom je vytvorená aplikácia, kde je implementovaná vybraná metóda Runge – Kutta.

Otázka riešenia diferenciálnych rovníc numerickými metódami Runge – Kutta je zaujímavá v rôznych oblastiach vedecko – technickej aplikačnej činnosti, či už v dynamike sústavy hmotných bodov alebo napríklad v oblasti mechaniky prúdenia kvapalín. S diferenciálnymi rovnicami sa môžeme stretnúť aj v oblasti pružnosti a pevnosti – pri riešení úloh týkajúcich sa deformácie staticky určitých nosníkov. Práve na túto oblasť aplikačnej činnosti je zameraná táto diplomová práca.

Je dôležité poznať jednotlivé metódy Runge – Kutta, preto sa v teoretickej časti venujeme práve tomuto prehľadu.

Jednou z najpoužívanejších metód v technickej praxi je metóda Runge – Kutta štvrtého rádu a jej varianty. Preto sa v praktickej časti práce budeme venovať implementácii dvoch metód Runge – Kutta štvrtého rádu, konkrétne klasickej metódy a metódy Runge – Kutta – Gill, vo vývojovom prostredí programovacieho jazyka *Delphi*. Podrobne opíšeme vytvorenú aplikáciu, aké komponenty a techniky sme použili a prečo. Vysvetlíme, aké vstupné údaje sú potrebné zadať, objasníme jednotlivé položky hlavného menu a opíšeme aj rôzne ošetrenia, ktoré aplikácia obsahuje.

V závere tejto kapitoly sú uvedené príklady rôznych nosníkov, kde poznáme diferenciálne rovnice priehybových čiar, ktoré získame z funkcie ohybového momentu pre jednotlivé myslené rezy. Uskutočníme numerickú integráciu pomocou vytvorenej aplikácie, pričom výsledok riešenia diferenciálnej rovnice bude predstavovať hodnotu uhlu sklonu dotyčnice voči priehybovej čiare zaťaženého nosníka. Jednotlivé hodnoty následne porovnáme s riešením, ktoré uvádza literatúra. Uvedieme aj príklad nosníka, kde riešenie pomocou aplikácie porovnáme s riešením v programe *MathCad* firmy *PTC*.

Pri výbere témy diplomovej práce u mňa rozhodol fakt, že riešenie diferenciálnych rovníc je súčasťou matematiky. Keďže som sa venovala štúdiu aplikovanej informatiky, pri vypracovaní tejto diplomovej práce som mohla využiť aj tieto vedomosti. Matematika bola v mojom doterajšom štúdiu väčšinou obľúbený predmet. Páči sa mi na nej hlavne to, že človek pri nej dokáže využiť aj svoje logické myslenie.

1 Prehľad o súčasnom stave riešenej problematiky

1.1 Diferenciálne rovnice

Diferenciálne rovnice sú časťou matematiky, ktorá sa nazýva matematická analýza. Predmetom štúdia matematickej analýzy sú funkcie a operácie s nimi. Okrem riešenia diferenciálnych rovníc sú typickými úlohami aj výpočet integrálu z danej funkcie cez daný interval, výpočet hodnoty derivácie alebo len obyčajné stanovenie hodnoty funkcie v niektorom bode (Přikryl, 1988).

Rovnicu, ktorá vyjadruje vzťah medzi nezávislou premennou x z nejakej množiny $M \subset R$, neznámou funkciou f(x) a aspoň jednou z derivácií f′(x),f″(x),...,f(n)(x), $n \in N$ tejto funkcie nazývame obyčajnou diferenciálnou rovnicou.

Ak označíme $f(x) = y,\ f'(x) = y',..., f^{(n)}(x) = y^{(n)}$, potom môžeme diferenciálnu rovnicu symbolicky zapísať v tvare $F(x, y',..., y^{(n)}) = 0$.

Ak v rovnici vystupuje najviac n-tá derivácia $f^{(n)}(x) = y^{(n)}$ neznámej funkcie f(x) a nenachádza sa v nej žiadna derivácia $f^{(k)}(x), k > \mathrm{n}$, potom rovnicu nazývame obyčajná diferenciálna rovnica rádu n. Rád diferenciálnej rovnice je najvyšší rád derivácie, ktorá v rovnici vystupuje (Velichová, 2006).

Diferenciálne rovnice sú jedným z veľmi účinných prostriedkov modelovania javov vedy, techniky Využívajú sa v tejto oblasti hlavne preto, že v prírode, resp. spoločnosti sa často jednoduchšie pozoruje vývoj zmien v hodnotách ako vývoj hodnôt samotných. Preto sa aj jednoduchšie popíše nejaký jav pomocou vzťahov medzi týmito zmenami hodnôt a hodnotami ako nájsť priamo závislosť od stanovenej nezávislej premennej. a okolitého sveta (Nekvinda, 1976).

Diferenciálne rovnice, ako je spomenuté vyššie, nachádzajú svoje využitie v rôznych oblastiach vedy. Pri popise fyzikálnych javov môžeme uviesť ako príklad využívania diferenciálnych rovníc napríklad pri chladnutí telesa alebo pri pohybe oscilátorov a kyvadiel. Ďalšie uplatnenie je aj v chémii a to pri riedení roztokov, priebehy chemických reakcií a pod., v biológii množenie buniek, v ekológii rast populácie alebo v geológii určovanie veku hornín a skamenelín pomocou rádioaktívneho rozpadu (Kučerová, 2003).

Metódy riešenia diferenciálnych rovníc sa dajú rozdeliť na dve skupiny:

1. Exaktné metódy, pri ktorých ide o nájdenie všeobecného integrálu diferenciálnej rovnice ($y' = f(x, y)$) v takzvanom uzatvorenom vzťahu, t.j. ako konečnej kombinácie elementárnych funkcií, poprípade ich integrálov. Na základe počiatočnej podmienky ($y(a) = y_0$) sa potom určí jedno partikulárne riešenie.

2. Približné metódy. Pri ich použité sa kladie skromnejší cieľ v porovnaní s exaktnými metódami. Výhodou približných metód je však ich univerzálnosť, pretože exaktné metódy uspejú len pre niektoré typy diferenciálnych rovníc.

Tieto metódy sa dajú ďalej rozdeliť na metódy:

a) analytické, kde je výsledkom analytický výraz,

b) numerické, kde výsledkom býva zvyčajne tabuľka funkčných hodnôt, t.j. funkcia definovaná na diskrétnej množine,

c) *grafické*, ktoré sú spravidla založené na niektorej analytickej, resp. numerickej metóde.

Toto rozdelenie metód riešenia diferenciálnych rovníc je len veľmi hrubé. Často dochádza k prekrývaniu spomenutých skupín. Existujú napríklad približné metódy, ktoré sa dajú podľa tejto klasifikácie chápať ako metódy numerické, ale aj ako metódy analytické (Nekvinda, 1976).

V tejto práci sa venujeme iba numerickým metódam pre obyčajné diferenciálne rovnice so začiatočnými podmienkami (Cauchyho úloha). Ako vyplýva už z názvu tejto práce, zamerali sme sa na metódy Runge – Kutta, ktoré vychádzajú z Taylorovho rozvoja.

Predpokladajme diferenciálnu rovnicu v tvare:

$$y' = f(x, y(x)) \qquad (1)$$

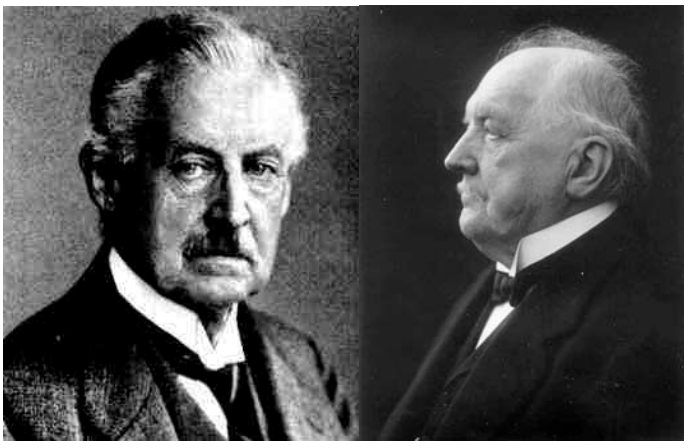
Riešením tejto rovnice je y, ktoré nadobúda v určitom bode $x_0 \in < a, b >$ dané hodnoty η, teda $y(x_0) = \eta, x_0 \in < a, b >$. Takouto počiatočnou úlohou je Cauchyho začiatočná úloha.

Aby bola zaručená jednoznačnosť riešenia je nutné Cauchyho jednoznačnosť rešenia úlohy $y' = f(x, y(x)), y(x_0) = \eta, x_0 \in < a, b >$ v oblasti spojitosti funkcie

nahradiť Lipschnitzovou podmienkou. Danú rovnicu prepíšeme na tvar $y' = f(x, y)$, kde $f(x, y)$ je reálna vektorová funkcia a je definovaná a spojitá na množine $S =< a, b > \times R^m$. Ak má byť splnená Lipschnitzova podmienka, tak musí platiť, že existuje konštanta L, ktorá je nezávislá na x, y tak, že platí: $\|f(x, y) - f(x, z)\| \leq L\|y - z\|$ pre všetky $x \in < a, b >$ a všetky vektory *y, z* (Rédl, 2010b).

1.2 Numerické metódy Runge - Kutta

Numerické metódy Runge – Kutta patria k všeobecným jednokrokovým metódam (Černá, 1987). Meno dostali po svojich autoroch C. Rungeho a W. Kuttu (obr.1), ktorí odvodili konkrétne vzorce (3. a 4. rád) na rozhraní predminulého a minulého storočia. Eulerova metóda, ktorá je pokladaná za vzorec typu Runge – Kutta prvého rádu je ešte staršieho dáta. W.Kutta odvodil aj vzorec piateho rádu, avšak s chybou. Túto chybu v roku 1925 opravil E. J. Nyström, ktorému bolo prisúdené prvenstvo v odvodení vzorca piateho rádu. Prvý vzorec šiesteho rádu odvodil v roku 1956 A. Huťa. Počnúc týmto krokom vzrástol počet vzorcov typu Runge – Kutta exponenciálne (Dávid, 1988). Najčastejšie používanou metódou na riešenie diferenciálnych rovníc ako aj sústav je metóda Runge – Kutta štvrtého stupňa a jej varianty (Rédl, 2010b).



Obr. 1 Carl David Tolmé Runge (vľavo) a Martin Wilhelm Kutta (vpravo) (zdroj: http://numericalmethods.eng.usf.edu)

Tieto metódy sa dajú použiť nielen k získaniu počiatočných hodnôt, ale samozrejme i celého riešenia. Nevyžadujú dodatočné počiatočné hodnoty a dajú sa ľahko naprogramovať pre samočinný počítač, ale tieto prednosti však nevyvážia

ich nevýhody, a to ťažké určenie chyby a malá rýchlosť v porovnaní s metódou prediktor – korektor (Ralston, 1978).

Metódy Runge - Kutta sú veľmi univerzálne a v technickej praxi užitočné. Sú založené na Taylorovom rozvoji funkcie, ale nepriamo tak, aby sme nemuseli určovať hodnoty derivácií funkcie - tieto sa aproximujú výpočtom samotnej funkcie vo vhodne zvolených strategických bodoch (Riešené úlohy z matematiky, 2000).

Prehľad o súčasnom stave využitia moderných numerických metód Runge – Kutta v oblasti vedecko-technickej aplikačnej činnosti

Technická prax a vedecko – výskumná činnosť je úzko spätá s meraním a spracovaním experimentálne získaných funkcií. Technické funkcie ($Y_i(t)$) sú často merané vo funkčnej závislosti na čase a sú spojité alebo nespojité. Tieto funkcie môžu byť vo vzťahu k požadovanej výslednej funkcii aj deriváciou vyšších stupňov napr. $y_i''(t) = f(Y_i(t))$, kde $0 < t < T, y(0) = Y_0$. V takomto prípade je nutné danú funkciu riešiť ako diferenciálnu rovnicu danú funkčnými hodnotami (tabuľkou), v ktorých sa hľadá riešenie (Rédl, 2010b).

Existujú rôzne technologické procesy, ktoré sú opísané napríklad sústavou nelineárnych diferenciálnych rovníc, ktoré je nutné riešiť numerickou integráciou. Na tento účel sa dá zvoliť už spomenutá metóda Runge - Kutta štvrtého rádu. Medzi jej hlavné výhody patrí prepracovanosť, presnosť, rýchlosť a možnosť jednoduchej implementácie. Medzi takéto technologické procesy môže patriť tvorba virtuálneho technologického procesu na príklade hydraulického systému alebo virtuálny regulátor (Božek, 2006).

Numerická metóda Runge – Kutta štvrtého rádu sa dá použiť aj na riešenie Navier – Stokesových rovníc v oblasti mechaniky prúdenia kvapalín. Jednou z ďalších aplikácií metódy Runge – Kutta je aplikácia v dynamike sústavy hmotných bodov.

Základom všetkých metód typu Runge – Kutta je vyjadrenie rozdielu medzi hodnotami y v bodoch x_{n+1} a x_n v tvare

$$y_{n+1} - y_n = \sum_{i=1}^{m} w_i k_i, \tag{2}$$

kde w_i sú konštanty a

$$k_i = h_n f(x_n + \alpha_i h_n, y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j), \qquad (3)$$

kde

$$h_n = x_{n+1} - x_n \quad \text{a} \quad \alpha_1 = 0.$$

Používame h_n namiesto h, pretože je možné, aj keď málokedy účelné, meniť interval Runge – Kuttovej metódy v každom štádiu. Ak sú dané w_i, α_i a β_i, je (2) metódou pre riešenie rovnice $y' = f(x, y), y(x_0) = y_0$, ktorá zrejme nevyžaduje dodatočné počiatočné hodnoty.

Našim cieľom je určiť konštanty w_i, α_i a β_i tak, aby (2) mala žiadané vlastnosti. Špeciálne je treba urobiť koeficienty pri h_n^r v Taylorovom rozvoji oboch strán (2) v bode $[x_n;y_n]$ totožné pre $r = 1,2,\ldots M$. Ako vidieť, nie je možné dosiahnuť lepší výsledok ako M = m (Ralston, 1978).

Toto riešenie môže byť prezentované aj formou Butcherovej tabuľky (tab.1). Ako uvádza Anastassi (2009), koeficienty α2, ..., αm musia spĺňať nasledujúcu podmienku:

$$\alpha_i = \sum_{j=1}^{i-1} \beta ij \; ; \; i = 2..m .$$

Butcherovu tabuľku spomínajú vo svojej práci napríklad aj Bruder (1996) alebo Skvortsov (2010).

Tab. 1 Butcherova tabuľka

0					
α_2	β_{21}				
α_3	β_{31}	β_{32}			
.					
.					
.					
α_m	β_{m1}	β_{m2}	…	$\beta_{m,m-1}$	
	w_1	w_2	...	W_{m-1}	w_m

Rozpísaním do Taylorovho rozvoja v tvare:

$$y_{i+1} = y_i + \left.\frac{dy}{dx}\right|_{x_i y_i} (x_{i+1} - x_i) + \left.\frac{1}{2!}\frac{d^2 y}{dx^2}\right|_{x_i y_i} (x_{i+1} - x_i)^2 + \frac{1}{3!}\frac{d^3 y}{dx^3}$$
$$\left|_{x_i y_i} (x_{i+1} - x_i)^3 + \left.\frac{1}{4!}\frac{d^4 y}{dx^4}\right|_{x_i y_i} (x_{i+1} - x_i)^4 \right. \tag{4}$$

a využitím vzťahov:

$$\frac{dy}{dx} = f(x, y),$$

$$x_{i+1} - x_i = h,$$

a dosadením do rovnice (4) dostávame:

$$y_{i+1} = y_i + f(x, y)h_i + \frac{1}{2!} f'(x_i, y_i){h_n}^2 + \frac{1}{3!} f''(x_i, y_i){h_n}^3$$
$$+ \frac{1}{4!} f'''(x_i, y_i){h_n}^4 \tag{5}$$

Dosadením pre obe strany rovnice (5) vznikne sústava rovníc pre k_i a h_n^i . Porovnaním dostávame nasledovných osem rovníc (sústavu parametrov):

$$w_1 + w_2 + w_3 + w_4 = 1,$$

$$w_2\alpha_2 + w_3\alpha_3 + w_4\alpha_4 = \frac{1}{2},$$

$$w_2\alpha_2^2 + w_3\alpha_3^3 + w_4\alpha_4^4 = \frac{1}{3},$$

$$w_3\alpha_2\beta_{32} + w_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) = \frac{1}{6}, \tag{6}$$

$$w_2\alpha_2^3 + w_3\alpha_3^3 + w_4\alpha_4^3 = \frac{1}{4},$$

$$w_2\alpha_2^2\beta_{32} + w_4(\alpha_2^2\beta_{42} + \alpha_3^2\beta_{43}) = \frac{1}{12},$$

$$w_3\alpha_2\alpha_3\beta_{32} + w_4(\alpha_2\beta_{42} + \alpha_3\beta_{43})\alpha_4 = \frac{1}{8},$$

$$w_4\alpha_2\beta_{32}\beta_{43}) = \frac{1}{24}.$$ (Rédl, 2010b)

Ak má byť týchto výsledných osem rovníc nezávislých na f(x,y), musí platiť:

$$\alpha_i = \sum_{j=1}^{i-1} \beta_{i,j} \ , \ i = 2,3,4 \, . \tag{7}$$

Sústava (7) a (6) má 11 rovníc a 13 neznámych, čo všeobecne stačí na určenie parametrov s dvoma stupňami voľnosti (Ralston, 1978).

1.2.1 Klasické metódy Runge – Kutta

1.2.1.1 Metódy druhého rádu

Pre prípad $m = 2$ zostanú v sústave (6) len rovnice, ktoré sa vzťahujú k h_n^2. Spolu so (7) pre $i=2$ dostaneme:

$$\begin{aligned} & w_1 + w_2 = 1, \\ & \alpha_2 w_2 = \frac{1}{2}, \\ & \beta_{21} = \alpha_2 \, . \end{aligned} \tag{8}$$

Ak položíme za α_2 hodnoty $\frac{1}{2}, \frac{2}{3}, 1$, dostaneme tri metódy druhého rádu, ktoré majú rovnaký význam. Pre ne je potom (2) dané vzťahmi:

$$y_{n+1} - y_n = h_n f(x_n + \frac{1}{2} h_n, y_n + \frac{1}{2} h_n f_n),$$

$$y_{n+1} - y_n = \frac{1}{4} h_n \left[f(x_n, y_n) + 3 f(x_n \frac{2}{3} h_n, y_n + \frac{2}{3} h_n f_n) \right],$$

$$y_{n+1} - y_n = \frac{1}{2} h_n \left[f(x_n, y_n) + f(x_n + h_n, y_n + h_n f_n) \right].$$

1.2.1.2 Metódy tretieho rádu

Rovnice pre tento prípad sú:

$$
\begin{aligned}
&w_1 + w_2 + w_3 = 1, \\
&\alpha_2 w_2 + \alpha_3 w_3 = \frac{1}{2}, \\
&\alpha_2^2 w_2 + \alpha_3^2 w_3 = \frac{1}{3}, \\
&\alpha_2 \beta_{32} w_3 = \frac{1}{6}, \\
&\alpha_2 = \beta_{21}, \\
&\alpha_3 = \beta_{31} + \beta_{32}.
\end{aligned}
\tag{9}
$$

(9) je dvoj - parametrický systém, ktorý sa dá zapísať v tvare:

$$
\begin{aligned}
&w_1 = 1 + \frac{2 - 3(\alpha_2 + \alpha_3)}{6\alpha_2\alpha_3}, \\
&w_2 = \frac{3\alpha_2 - 2}{6\alpha_2(\alpha_3 - \alpha_2)}, \\
&w_3 = \frac{2 - 3\alpha_2}{6\alpha_2(\alpha_3 - \alpha_2)}, \qquad \alpha_2 \neq \alpha_3, \quad \alpha_2, \alpha_3 \neq 0, \\
&\beta_{21} = \alpha_2, \\
&\alpha_2 \neq \frac{2}{3}, \\
&\beta_{31} = \frac{3\alpha_2\alpha_3(1 - \alpha_2) - \alpha_3^2}{\alpha_2(2 - 3\alpha_2)}, \qquad \alpha_2 \neq \frac{2}{3}, \\
&\beta_{32} = \frac{\alpha_3(\alpha_3 - \alpha_2) - \alpha_3^2}{\alpha_2(2 - 3\alpha_2)}.
\end{aligned}
\tag{10}
$$

Z metód tretieho rádu majú význam nasledujúce dve metódy:

$$
\begin{aligned}
y_{n+1} - y_n &= \frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3, \\
k_1 &= h_n f(x_n, y_n), \\
k_2 &= h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}k_1), \\
k_3 &= h_n f(x_n + \frac{3}{4}h_n, y_n + \frac{3}{4}k_2),
\end{aligned}
\tag{11}
$$

a

$$
\begin{aligned}
y_{n+1} - y_n &= \frac{1}{6}(k_1 + 4k_2 + k_3), \\
k_1 &= h_n f(x_n, y_n), \\
k_2 &= h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}k_1), \\
k_3 &= h_n f(x_n + h_n, y_n - k_1 + 2k_2).
\end{aligned}
\tag{12}
$$

Ak je f(x,y) funkciou len premennej *x*, potom je (12) Simpsonovým pravidlom.

1.2.1.3 Metódy štvrtého rádu

V praxi sa najčastejšie používa metóda Runge – Kutta štvrtého rádu. Keď budeme riešiť dvoj - parametrickú sústavu (6) dostaneme:

$$
\begin{aligned}
w_1 &= \frac{1}{2} + \frac{1 - 2(\alpha_2 + \alpha_3)}{12\alpha_2\alpha_3}, \\
w_2 &= \frac{2\alpha_3 - 1}{12\alpha_2(\alpha_3 - \alpha_2)(1 - \alpha_2)}, \\
w_3 &= \frac{1 - 2\alpha_2}{12\alpha_3(\alpha_3 - \alpha_2)(1 - \alpha_3)}, \\
w_4 &= \frac{1}{2}1\frac{2(\alpha_2 + \alpha_3) - 3}{12(1 - \alpha_2)(1 - \alpha_3)}, \\
\beta_{32} &= \frac{\alpha_3(\alpha_3 - \alpha_2)}{2\alpha_2(1 - 2\alpha_2)}, \qquad \alpha_4 = 1,
\end{aligned}
\tag{13}
$$

$$\beta_{42} = \frac{(1-\alpha_2)\left[\alpha_2 + \alpha_3 - 1 - (2\alpha_3 - 1)^2\right]}{2\alpha_2(\alpha_3 - \alpha_2)\left[6\alpha_2\alpha_3 - 4(\alpha_2 + \alpha_3) + 3\right]},$$

$$\beta_{43} = \frac{(1-2\alpha_2)(1-\alpha_2)(1-\alpha_3)}{\alpha_3(\alpha_3 - \alpha_2)\left[6\alpha_2\alpha_3 - 4(\alpha_2 + \alpha_3) + 3\right]},$$

s výnimkou tých prípadov, keď je $\alpha_2, \alpha_3 = 0$, $\alpha_2, \alpha_3 = 1$, $\alpha_2 = \alpha_3$, alebo keď sa anulujú menovatele β_{32}, β_{42} a $a\beta_{43}$.

Najčastejšie používaná metóda Runge – Kutta štvrtého rádu je tá, kde je $\alpha_2 = \alpha_3 = \frac{1}{2}$. Príslušné rovnice klasickej metódy so štyrmi aproximáciami sú v tvare:

$$\begin{aligned} y_{n+1} - y_n &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= h_n f(x_n, y_n), \\ k_2 &= h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}k_1), \\ k_3 &= h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}k_2), \\ k_4 &= h_n f(x_n + h_n, y_n + k_3). \text{ (Ralston, 1978)} \end{aligned} \tag{14}$$

Táto metóda štvrtého rádu je tiež v špeciálnom prípade $f(x, y) \equiv f(x)$ totožná so Simpsonovým pravidlom (Přikryl, 1988).

Ďalšou známou metódou štvrtého rádu je metóda, ktorej rovnice sú tvaru:

$$\begin{aligned} y_{n+1} - y_n &= \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4), \\ k_1 &= h_n f(x_n, y_n), \\ k_2 &= h_n f(x_n + \frac{1}{3}h_n, y_n + \frac{1}{3}k_1), \\ k_3 &= h_n f(x_n + \frac{2}{3}h_n, y_n - \frac{1}{3}k_1 - k_2), \\ k_4 &= h_n f(x_n + h_n, y_n + k_1 - k_2 + k_3). \end{aligned} \tag{15}$$

Metóda (14) je oveľa používanejšia, a preto, keď sa hovorí o metóde Runge – Kutta, myslí sa tým práve táto konkrétna metóda. Metóda (15) sa často nazýva ako trojosminové pravidlo.

V staršej literatúre sa uvádza ešte jedna metóda štvrtého rádu, takzvaná Gillova metóda:

$$y_{n+1} - y_n = \frac{1}{6}\left[k_1 + 2(1-\frac{1}{2}\sqrt{2})k_2 + 2(1+\frac{1}{2}\sqrt{2})k_3 + k_4 \right],$$

$$k_1 = h_n f(x_n, y_n),$$

$$k_2 = h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}k_1), \qquad (16)$$

$$k_3 = h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2}(\sqrt{2}-1)k_1 + (1-\frac{1}{2}\sqrt{2})k_2),$$

$$k_4 = h_n f(x_n + h_n, y_n - \frac{1}{2}\sqrt{2}k_2 + (1+\frac{1}{2}\sqrt{2})k_3).$$

Gillova metóda má zložitejšie vyzerajúce koeficienty práve kvôli tomu, že túto metódu použitú pre riešenie *m* diferenciálnych rovníc možno upraviť na tvar, ktorý vyžaduje na prevedenie jedného kroku *3m+a* pamäťových miest, kde *a* nezávisí na *m*. Klasická metóda Runge - Kutta (14) si vyžaduje *4m+a* pamäte. Na moderných počítačoch sú však problémy spojené s úsporou pamäte v súvislosti s metódami Runge – Kutta málo dôležité, takže táto metóda stráca svoj predošlý význam. Jednou z predností tejto metódy je, že istým spôsobom minimalizuje nakopenie zaokrúhľovacích chýb (Vitásek, 1987).

1.2.2 Metódy Runge – Kutta vyššieho rádu

Metódy Runge – Kutta, ktoré sú vyššie ako štvrtého rádu, sa používajú veľmi zriedka. Medzi najznámejšie patrí metóda piateho rádu, nazývaná Nyströmová metóda a metóda šiesteho rádu pomenovaná Huťova metóda. Obidve metódy sa nepoužívajú tak často kvôli tomu, že k prevedeniu jedného kroku je potrebné mnohokrát vypočítavať hodnotu pravej strany danej diferenciálnej rovnice (čo je bezpochyby najväčší podiel v práci vynaloženej na prevedenie jedného kroku metódy). Tieto metódy sú však vysokého rádu, takže sa dajú použiť s väčším integračným krokom, čím môže byť spomenutá nevýhoda do značnej miery, aspoň v niektorých prípadoch, vykompenzovaná.

1.2.2.1 Metóda piateho rádu - Nyströmová metóda

Táto metóda vyžaduje šesť funkčných hodnôt a rovnice, ktorými je daná sú:

$$y_{n+1} - y_n = \frac{1}{192} h(23k_1 + 125k_3 - 81k_5 + 125k_6),$$

$$k_1 = h_n f(x_n, y_n),$$

$$k_2 = h_n f(x_n + \frac{1}{3}h_n, y_n + \frac{1}{3}h_n k_1),$$

$$k_3 = h_n f(x_n + \frac{2}{5}h_n, y_n + \frac{1}{25}h_n(4k_1 + 6k_2)),$$

$$k_4 = h_n f(x_n + h_n, y_n + \frac{1}{4}h_n(k_1 - 12k_2 + 15k_3)), \qquad (17)$$

$$k_5 = h_n f(x_n + \frac{2}{3}h_n, y_n + \frac{1}{81}h_n(6k_1 + 90k_2 - 50k_3 + 8k_4)),$$

$$k_6 = h_n f(x_n + \frac{4}{5}h_n, y_n + \frac{1}{75}h_n(6k_1 + 36k_2 - 10k_3 + 8k_4)).$$

1.2.2.2 Metóda šiesteho rádu – Huťova metóda

Huťova metóda vyžaduje v každom kroku osemkrát vypočítať hodnotu pravej strany danej diferenciálnej rovnice. Príslušné rovnice pre túto metódu šiesteho rádu sú nasledovné:

$$y_{n+1} - y_n = \frac{1}{840} h_n(41k_1 + 216k_3 + 27k_4 + 272k_5 +$$

$$+ 27k_6 + 216k_7 + 41k_8),$$

$$k_1 = h_n f(x_n, y_n),$$

$$k_2 = h_n f(x_n + \frac{1}{9}h_n, y_n + \frac{1}{9}h_n k_1),$$

$$k_3 = h_n f(x_n + \frac{1}{6}h_n, y_n + \frac{1}{24}h_n(k_1 + 3k_2)),$$

$$k_4 = h_n f(x_n + \frac{1}{3}h_n, y_n + \frac{1}{6}h_n(k_1 - 3k_2 + 4k_3)),$$

$$k_5 = h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{8}h_n(-5k_1 + 27k_2 - 24k_3 + 6k_4)),$$

$$k_6 = h_n f(x_n + \frac{2}{3}h_n, y_n + \frac{1}{9}h_n(221k_1 - 981k_2 + \quad (18)$$

$$+867k_3 - 102k_4 + k_5)),$$

$$k_7 = h_n f(x_n + \frac{5}{6}h_n, y_n + \frac{1}{48}h_n(-183k_1 + 678k_2 -$$

$$-472k_3 - 66k_4 + 80k_5 + 3k_6)),$$

$$k_8 = h_n f(x_n + h_n, y_n + \frac{1}{82}h_n(716k_1 - 2079k_2 + 1002k_3 +$$

$$+834k_4 - 454k_5 - 93k_6 + 72k_7)) \text{ (Vitásek, 1987).}$$

1.2.3 Odhad chýb metód Runge – Kutta

Predpokladajme, že vo vhodnom okolí bodu (x_n, y_n), v ktorom chceme odhadnúť lokálnu chybu, platí:

$$|f(x,y)| < M, \left|\frac{\partial^{i+j} f}{\partial x^i \partial y^i}\right| < \frac{L^{i+j}}{M^{m-1}}, \; i + j \le p \qquad (19)$$

(kde tvar druhého odhadu bol zvolený z dôvodu formálneho zjednodušenia ďalšieho zápisu).

Potom pre lokálnu chybu všeobecnej metódy Runge – Kutta druhého rádu používajúcej dve dosadenia platí:

$$|L(y(x_n);h)| < \left(4\left|\frac{1}{6} - \frac{1}{2}\alpha_2^2 w_2\right| + \frac{1}{3}\right)h^3 ML^2, \qquad (20)$$

Pre metódu tretieho rádu s troma dosadeniami je

$$\begin{aligned} |L(y(x_n);h)| < [8&\left|\frac{1}{24} - \frac{1}{6}\left(\alpha_2^3 w_2 + \alpha_3^3 w_3\right)\right| + \\ +4&\left|\frac{1}{24} - \frac{1}{2}\alpha_2^2 \beta_{32} w_3\right| + 4\left|\frac{1}{8} - \alpha_2\alpha_3\beta_{32} w_3\right| + \frac{1}{12}]h^4 ML^3 \end{aligned} \qquad (21)$$

a pre metódu štvrtého rádu platí:

$$\begin{aligned} |L(y(x_n);h)| < (&16|b_1| + 4|b_2| + |b_2 + 3b_3| + |2b_2 + 3b_3| + \\ &+ |b_2 + b_3| + |b_3| + 8|b_4| + |b_5| + |2b_5 + b_7| + \\ &|b_5 + b_6 + b_7| + |b_6| + |2b_6 + b_7| + |b_7| + 2|b_8|)h^5 ML^4, \end{aligned} \qquad (22)$$

kde

$$b_1 = \frac{1}{120} - \frac{1}{24}\left(\alpha_2^4 w_2 + \alpha_3^4 w_3 + w_4\right),$$

$$b_2 = \frac{1}{20} - \frac{1}{2}\left[\alpha_2 \alpha_3^2 \beta_{23} w_3 + \left(\alpha_2 \beta_{42} + \alpha_3 \beta_{43}\right) w_4\right],$$

$$b_3 = \frac{1}{120} - \frac{1}{6}\left[\alpha_2^3 \beta_{32} w_3 + \left(\alpha_2^3 \beta_{42} + \alpha_3^3 \beta_{43}\right) w_4\right],$$

$$b_4 = \frac{1}{30} - \frac{1}{2}\left[\alpha_2^2 \alpha_3 \beta_{32} w_3 + \left(\alpha_2^2 \beta_{42} + \alpha_3^2 \beta_{43}\right) w_4\right], \qquad (23)$$

$$b_5 = \frac{1}{120} - \frac{1}{2}\alpha_2^2 \beta_{43} w_4,$$

$$b_6 = \frac{1}{40} - \frac{1}{2}\left[\alpha_2^2 \beta_{32} w_3 + \left(\alpha_2 \beta_{42} + \alpha_3 \beta_{43}\right)^2 w_4\right],$$

$$b_7 = \frac{7}{120} - \alpha_2 \left(1 + \alpha_3\right) \beta_{32} \beta_{43} w_4,$$

$$b_8 = \frac{1}{120}.$$

Pre populárnu metódu štvrtého rádu (14) zo vzorca (22) sa stane vzorec:

$$\left|L(y(x_n); h)\right| < \frac{73}{720} h^5 L^4. \qquad (24)$$

Ak píšeme odhady funkcie f a ich derivácie v tvare

$$\left|f(x, y)\right| < M, \left|\frac{\partial^{i+j} f}{\partial x^i \partial y^j}\right| < \frac{N}{M^{j-1}}, \; i + j \le 4, \qquad (25)$$

dá sa napísať pre lokálnu chybu metódy (14) takzvaný Bieberchavov (dnes už klasický) odhad:

$$\left|L(y(x_n); h)\right| \le 6h^5 MN(1 + N + N^2 + N^3 + N^4). \qquad (26)$$

Odhady, ktoré sú uvedené vyššie sa javia dosť komplikované a nie je jednoduché ich získať. Okrem toho, sú dosť často pesimistické, najmä ak je okolie, v ktorom odhadujeme pravú stranu danej diferenciálnej rovnice, veľké (Vitásek, 1987).

1.3 Vývojové prostredie Delphi

V dnešnej dobe, v čase mohutného rozvoja informatiky a informačných technológii, sa čoraz viac ľudí venuje programovaniu. V súčasnosti asi najčastejšie používaným jazykom je jazyk *C++*. K frekventovanejšie používaným jazykom patrí aj *Delphi*, ktoré je založené na staršom jazyku *Pascal*. *Pascal* sa vyznačuje modrou obrazovkou s blikajúcim kurzorom. V súčasnosti však už nie je záujem o *DOS*ovské okná, a preto sa firma Borland podujala dokonale tento starší jazyk prispôsobiť potrebám moderných operačných systémov a vytvorili programovací jazyk prispôsobený dnešným grafickým nárokom (Hajnala, 2005).

Ako sme spomenuli *Delphi* je vývojovým nástrojom, ktorého prvá verzia bola vydaná firmou *Borland* v roku 1994. O sedem rokov neskôr patrilo *Delphi* k najpoužívanejším programovacím prostriedkom. V súčasnosti *Delphi* vlastní a je vyvíjané firmou *Embarcadero Technologies*. Keďže je založený na jazyku *Pascal*, ktorý sa v tej dobe už tak často nepoužíval, začalo byť *Delphi* stále viac populárnejšie. Aplikácia, ktorá je vytvorená v rámci tejto diplomovej práce, je naprogramovaná práve vo vývojom prostredí *Borland Delphi Enterprise* verzia 7 s využitím komponentu *Express* verzia 2.5. Napriek tomu, že *Delphi* nie je freeware, poskytuje sa však študentská verzia. Po stiahnutí z oficiálnej stránky sa treba zaregistrovať ako študent a na uvedenú e-mailovú adresu je poslané sériové číslo.

Delphi je komplexné vizuálne programovacie prostredie pre operačný systém Windows. Jeho hlavná sila spočíva predovšetkým vo význame slova vizuálny. Vzhľad, rozhranie a prostredie každej aplikácie, ktoré by sa v *Pascale* alebo podobnom programovacom jazyku tvorilo týždeň, sa zvládne v *Delphi* vytvoriť za hodinu. Sú dve vysvetlenia, prečo sa v *Delphi* programuje rýchlejšie. Prvým sú vlastnosti *Windows* a vôbec spôsob programovania pod týmto operačným systémom – o mnoho problémov sa nemusí starať programátor, zvládne to systém sám. Druhým vysvetlením je, že vytvorenie vizuálneho prostredia a maximálne zjednodušenie návrhovej fázy umožňuje programátorom venovať sa vlastnej algoritmizácií.

Vývojové prostredie *Delphi* má mnohé výhody, firma *Borland*, ktorá *Delphi* uviedla na trh, je známym výrobcom spoľahlivých, osvedčených kompilátorov pre *Pascal* a *C*, dá sa v ňom vytvoriť jednoduchý návrh vizuálneho prostredia aplikácie s využitím vlastností operačného systému, vyznačuje sa objektovo orientovaným

prístupom, podporuje databázy, čo je v poslednej dobe podstatná vlastnosť, obsahuje veľké množstvo integrovaných nástrojov, podporuje Internet (využívanie komponentu WebBrowser) a v neposlednom rade možnosť vytvárať vlastné komponenty (Kadlec, 2007).

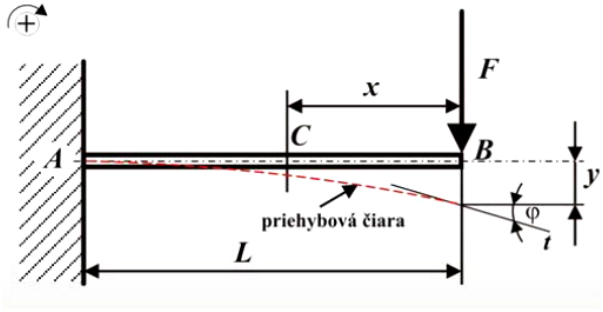
Skutočnosť, že *Delphi* je objektovo orientovaný jazyk znamená, že príkazy sa nepíšu do radu v jednom okne, ale kompozícia programu je úplne iná. V *Delphi* sa vytvárajú projekty, ktoré sa skladajú z dvoch častí: formulára a *unitu*. Formulárov a *unitov* môže byť v rámci jedného projektu viac (pri našej aplikácii vytvárame štyri formuláre a štyri unity). Formulár je okno, do ktorého sa vkladajú objekty. Tak, ako sa nastaví formulár s objektmi, tak aplikáciu uvidí aj koncový používateľ po reálnom spustení programu. *Unit* je miesto pre zdrojový kód, je to časť, kde je zapísaný zoznam príkazov, ktoré sa majú počas behu programu uskutočniť. Projekt vytvorený v *Delphi* môže správne pracovať, len ak má k dispozícii obe tieto jeho časti. (Hajnala, 2005)

2 Cieľ práce

Hlavnou témou diplomovej práce s názvom Riešenie diferenciálnych rovníc numerickými metódami Runge Kutta, ako už z názvu vyplýva, sú diferenciálne rovnice a ich možné riešenie pomocou metód Runge – Kutta. Diferenciálna rovnica je matematická rovnica, v ktorej ako premenné vystupujú derivácie funkcií. Využívajú sa vo väčšine oblastí ľudského poznania, ako napríklad vo fyzike, chémii, technike a podobne.

Cieľom tejto diplomovej práce je spracovať prehľad, ktorý sa týka súčasného stavu využitia moderných numerických metód Runge - Kutta na riešenie diferenciálnych rovníc v oblasti vedecko-technickej aplikačnej činnosti. Úlohou je popísať jednotlivé metódy Runge – Kutta druhého rádu až po metódy vyšších rádov. V práci by sa malo čerpať jednak zo zdrojov domácej literatúry (Rédl, Vitásek, Přikryl...) ale aj zo zdrojov zahraničnej literatúry (Ralston, Anastassi...).

Ďalším cieľom práce je zvoliť oblasť použitia a formulovať matematický problém. Za oblasť využitia numerických metód Runge – Kutta na riešenie diferenciálnych rovníc sme zvolili úlohy z predmetu Pružnosť a pevnosť, presnejšie úlohy ohybu. V úlohách, kde silou alebo spojitým bremenom zaťažíme nosník, môžeme uskutočniť numerickú integráciu funkcie ohybového momentu. Pomocou prvej integrácie tak hodnota riešenia diferenciálnej rovnice, bude predstavovať uhol sklonu dotyčnice φ voči priehybovej čiare zaťaženého nosníka (obr. 2).



Obr. 2 Zaťažený votknutý nosník dĺžky L so znázornením uhlu sklonu φ (Rédl, 2009)

Ako sme už spomenuli, cieľom je naprogramovať aplikáciu, ktorá bude využívať zvolené metódy Runge Kutta na riešenie používateľom zadanej diferenciálnej rovnice.

Pre aplikáciu vytvorenú v rámci tejto diplomovej práce sme zvolili dve metódy Runge – Kutta štvrtého rádu, a to klasickú metódu a metódu Runge – Kutta – Gill.

Aplikácia bude naprogramovaná vo vývojovom prostredí programovacieho jazyka *Borland Delphi 7*. Aj keď *Delphi* nie je freeware, poskytuje sa študentská verzia, kde je potrebné sa na oficiálnej stránke zaregistrovať a získať tak seriálové číslo.

Okrem vytvorenia aplikácie je cieľom praktickej časti ju aj opísať a uviesť príklady jej využitia. Keďže bude aplikácia zameraná na riešenie úloh ohybu, ďalším cieľom práce je uviesť aj príklady zaťažených nosníkov, pričom budeme vychádzať z funkcie ohybového momentu. Po prvej numerickej integrácii získame hodnotu uhlu sklonu dotyčnice φ. Získané výsledky treba porovnať s presným riešením podľa príslušných vzorcov uvádzaných v literatúre. Okrem porovnania s presným riešením, treba uskutočniť aj riešenie pomocou zvoleného CAD/CAM produktu, s ktorým sa riešenie pomocou aplikácie takisto porovná.

3 Metodika práce

Objektom skúmania tejto diplomovej práce sú diferenciálne rovnice a ich riešenie pomocou numerických metód Runge – Kutta. Pri písaní tejto práce budeme postupovať tak, aby sme v jej závere dospeli k stanoveným cieľom.

V prvom rade bude potrebné sa oboznámiť s problematikou súčasného stavu využitia moderných numerických metód Runge - Kutta na riešenie diferenciálnych rovníc v oblasti vedecko-technickej aplikačnej činnosti. V tejto časti postupne zhromaždíme potrebné informácie o danej problematike a taktiež vyhľadáme vhodné zdroje literatúry, odkiaľ sa potrebné poznatky budú čerpať. Literatúra, ktorú zhromaždíme, bude jednak z domácich, ale aj zo zahraničných zdrojov.

Po oboznámení sa s problematikou a vyhľadaní potrebných zdrojov informácii bude nasledovať vytvorenie samotného prehľadu o súčasnom stave využitia moderných numerických metód Runge - Kutta na riešenie diferenciálnych rovníc. V tejto časti sa vytvorí hlavná teoretická časť práce, v ktorej sa postupne objasnia pojmy ako diferenciálna rovnica, numerické metódy a taktiež tu budú bližšie popísané jednotlivé metódy Runge – Kutta, pričom každá metóda určitého rádu bude popísaná v osobitej podkapitole.

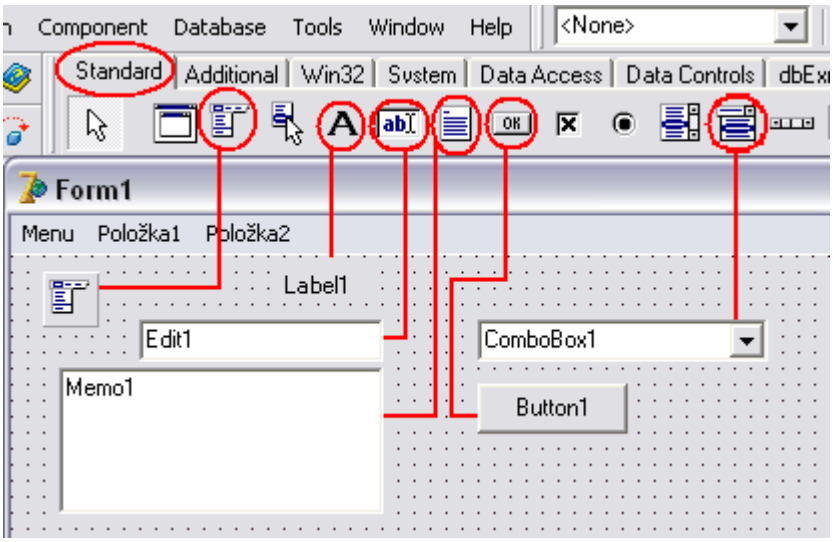
V ďalšej časti vyberieme dve metódy Runge – Kutta. Budú nimi metódy štvrtého rádu – klasická a Runge – Kutta – Gill. Následne zvolíme oblasť využitia, pre ktorú budeme vytvárať aplikáciu, kde budú spomenuté metódy Runge – Kutta implementované.

Pri úlohách ohybu, kde silou alebo spojitých bremenom zaťažujeme rôzne druhy nosníkov, vieme určiť ohybový moment pre jednotlivé myslené rezy. Ak uskutočníme numerickú integráciu funkcie ohybového momentu, dostávame hodnotu uhlu sklonu dotyčnice voči priehybovej čiare zaťaženého nosníka. Aplikácia bude vhodná pre riešenie takýchto úloh, kde z funkcie ohybového momentu (ktorá je spolu s dĺžkou nosníka L, krokom výpočtu h a metódou integrovania vstupnými hodnotami, ktoré používateľ bude musieť zadať) získame uskutočnením numerickej integrácie spomenutý uhol sklonu φ.

Aplikáciu budeme vytvárať vo vývojovom prostredí *Borland Delphi Enterprise*, verzia 7. Pri programovaní využijeme vedomosti získané v doterajšom štúdiu a pomôže nám pri tom aj offline verzia seriálov *„Umíme to s Delphi“* (Kadlec, 2007) a *„Tipy*

a triky v Delphi“ (Šindelář, 2007), kde nájdeme užitočné rady ohľadom programovania v tomto vývojovom prostredí.

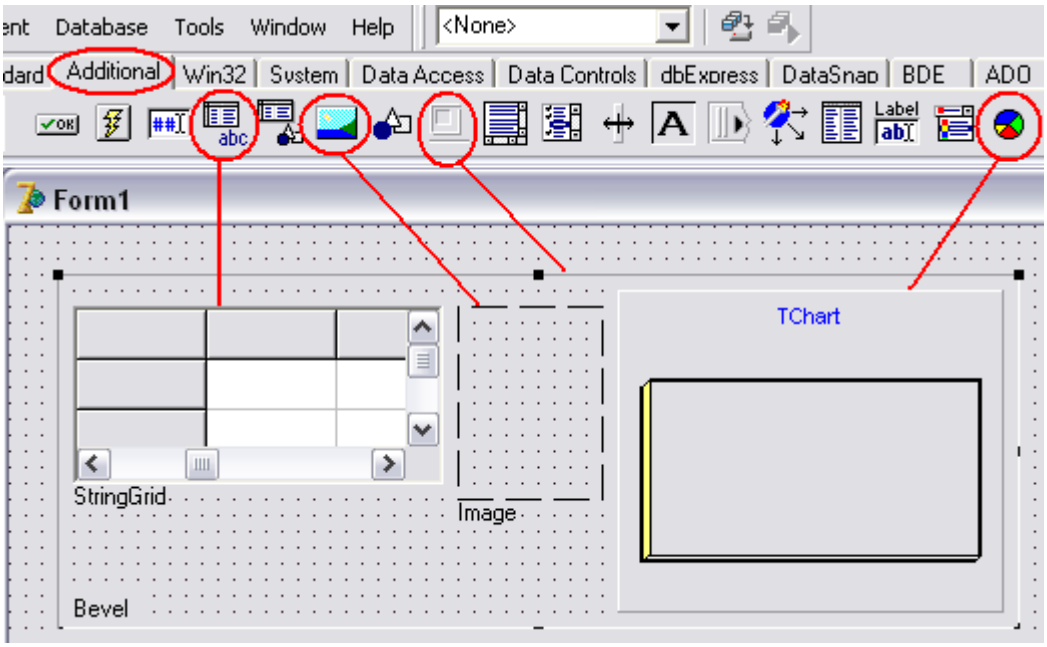
Pri tvorení aplikácie použijeme klasické komponenty, ktoré sa štandardne využívajú pri tvorbe aplikácií. Sú to komponenty ako textové pole (*Edit*), tlačidlo *(Button*), popisné pole (*Label*), textová plocha (*Memo*), rozbaľovacia lišta (*ComboBox*) alebo hlavné menu (*MainMenu*) nachádzajúce sa v palete *Standard* (obr. 3). Výhodou *Comboboxu* je hlavne úspora miesta. V našom programe ho využijeme na výber metódy, na základe ktorej sa uskutoční numerická integrácia. Komponent *MainMenu* použijeme pre vytvorenie hlavného menu, ktoré nám umožní sprehľadniť celú aplikáciu. Pomocou tohto menu bude aplikácia poskytovať možnosti, ktoré bude schopná uskutočniť (napríklad vytvorenie tabuliek, grafov alebo ukladanie so súboru).



Obr. 3 Zobrazenie palety Standard a komponentov, ktoré budú využité v aplikácii

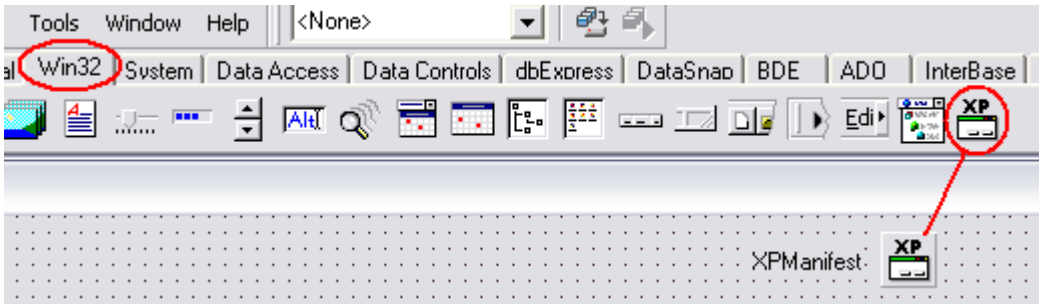
Ďalšie komponenty, ktoré budeme používať pri programovaní, sa nachádzajú v palete *Additional* (obr. 4). Veľký význam majú komponenty *StringGrid* a *Chart*, ktoré sa využijú pri tvorbe tabuliek a grafov. Zobrazovací význam komponentu *StringGrid* spočíva v prezentovaní textových údajov v tabuľkovom formáte. Komponent *Chart* môžeme používať pre grafickú prezentáciu akýchkoľvek údajov aj v aplikáciách, ktoré nepracujú so žiadnou databázou. Preto sme sa rozhodli použiť práve tento komponent na vykresľovanie grafov v našej aplikácii. V palete *Additional* nájdeme aj komponent *Bevel* (oddeľovacia čiara), ktorú využijeme pre lepšiu vizuálnu stránku aplikácie (rôzne orámovania a pod.). Posledným komponentom z tejto palety je komponent obrázok

(*Image*). Tento komponent umožňuje zobraziť v aplikácii obrázok, ktorý sa do neho nahrá.



Obr. 4 Zobrazenie palety Additional a komponentov, ktoré budú využité v aplikácii

Z palety *Win32* použijeme len jeden komponent nazvaný *XPManifest* (obr. 5).

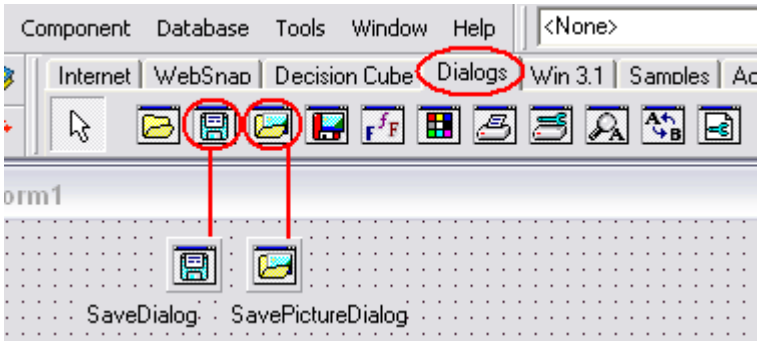


Obr. 5 Umiestnenie komponentu XPManifest v palete Win32

Pre samotný program nemá tento komponent špeciálnu úlohu. Použijeme ho len kvôli vzhľadu. Umožní totiž nášmu programu dosiahnuť modernejší vzhľad, ktorý je typický pre aplikácie v operačnom systéme *Windows XP*. Na príklade tlačidla na obr. 6 je znázornený rozdiel pred a po použití komponentu *XPManifest*.

Obr. 6 Tlačidlo pred a po použití komponentu XPManifest

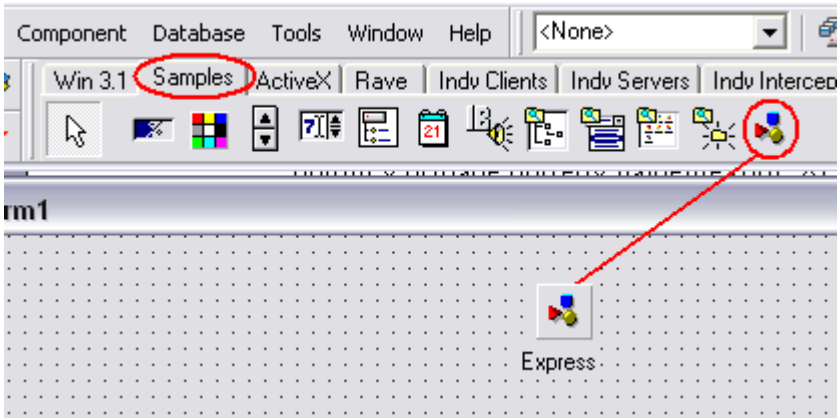
Nie všetky komponenty na formulári sú viditeľné aj po spustení aplikácie. Medzi takéto komponenty patria už spomenuté *MainMenu*, *XPManifest* a *Express*. Okrem týchto sú neviditeľné aj ďalšie dva komponenty, ktoré nájdeme v palete *Dialogs* – *SaveDialog* a *SavePictureDialog* (obr. 7).



Obr. 7 Komponenty SaveDialog a SavePictureDialog v palete Dialogs

Sú to štandardné dialógy, ktoré sa používajú na ukladanie súboru. Výhodou je, že sa nemusí vytvárať vlastné okno, pomocou ktorého užívateľ zvolí miesto a názov uloženia súboru. Po voľbe uložiť súbor sa otvorí preddefinované okno, kde sa zadá potrebný názov súboru.

Pri tvorbe aplikácie využijeme okrem štandardných komponentov aj komponent *Express*, verzia 2.5. Dôležitosť tohto komponentu spočíva v tom, že slúži na rýchle vyhodnocovanie výrazov. Je jedným z neviditeľných komponentov použitých v aplikácii. Tento komponent nepatrí medzi štandardnú výbavu *Delphi*. Skôr, ako ho chceme použiť, musíme ho do *Delphi* nainštalovať. Inštalačné súbory pre komponent *Express* budú súčasťou elektronického nosiča v Prílohe 1. Je potrebný súbor *Express.pas*, ktorý sa registruje medzi ostatné komponenty do palety *Samples*, kde ho potom v prípade potreby nájdeme (obr. 8).



Obr. 8 Umiestnenie komponentu Express na formulári

Tento komponent je voľný pre neobchodné a vzdelávacie použitie a je voľne dostupný aj na internete (napr. Parsers, mathematical expression evaluators, calculators, 1998), preto sme sa ho rozhodli použiť v našom programe.

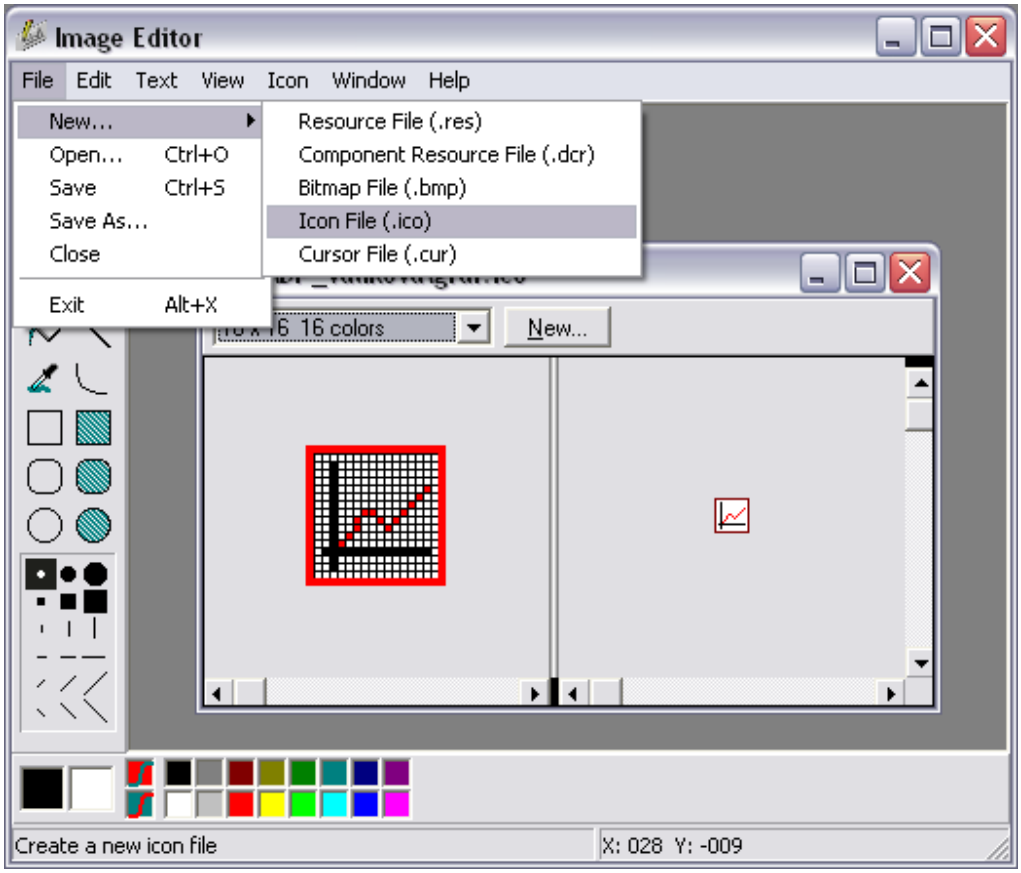
Výhodou tohto komponentu je to, že je určený pre opakované vyhodnocovanie toho istého výrazu s rôznymi hodnotami premenných. *Express* vyhodnocuje výrazy pre funkcie v závislosti od troch premenných a až od šiestich parametrov, čo je trochu neobvyklé. Namiesto neustáleho vykonávania vyhodnotenia rekurzívnej postupnosti pre výraz, tento komponent robí všetko v reálnom čase. Používa sa vtedy, ak by sme chceli použiť funkciu pre nejaké kritické veci, ako sú napríklad iterácie.

Komponent *Express* podporuje štandardné funkcie ako sú: *sin, cos, tan, cot, exp, ln, sqrt, arctan, sinh, cosh, tanh, arcsin, arccos, abs*. Zo zvláštnych konštánt je podporované len *„π“.* Namiesto *„e“* sa používa *exp(1).*

Komponent *Express* vytvorila autorka Renate Schaaf (obr. 9) ako hlavný základ pre interaktívne aplikácie matematiky (Pohuba, 2009).

Obr. 9 Renate Schaaf
(zdroj: http://owpdb.mfo.de/detail?photo_id=3643)

Keďže vývojové prostredie *Delphi* všetkým svojím aplikáciám automaticky priradí tú istú štandardnú ikonu, budeme pre našu aplikáciu vytvárať vlastné ikony. Na vytvorenie týchto ikon využijeme nástroj *Image Editor*, ktorým *Delphi* disponuje aj kvôli tomuto účelu. *Image Editor* nájdeme v hlavnom menu *Delphi*, v ponuke *Tools – Image Editor.* Po spustení *Image Editora* treba zvoliť z jeho hlavnej ponuky *File – New – Icon File (.ico)* (obr. 10).



Obr. 10 Nástroj Image Editor v Delphi

Ako sme spomenuli, našou úlohou bude pomocou spomenutých komponentov vytvoriť aplikáciu, kde po zadaní požadovaných vstupných hodnôt program ponúkne určité možnosti na prácu so zadanou rovnicou. Medzi tieto možnosti bude patriť vytvorenie troch tabuliek, ktoré budú obsahovať body x a k nim príslušné funkčné hodnoty. Počet bodov x bude závisieť od podielu spomenutých vstupných hodnôt L a h Používateľ si bude môcť zvoliť, ktorú metódu Runge – Kutta chce použiť, či klasickú metódu Runge – Kutta štvrtého rádu, alebo metódu Runge – Kutta – Gill, ktorá je takisto štvrtého rádu.

Prvá tabuľka bude rátať funkčné hodnoty z rovnice zadanej používateľom. Druhá tabuľka bude slúžiť pre zobrazenie funkčných hodnôt po prvej numerickej integrácii jednou zo spomenutých metód Runge – Kutta a tretia tabuľka funkčné hodnoty po druhej integrácii zvolenou metódou. Program bude schopný ku každej tabuľke vykresliť aj graf.

Aby si používateľ mohol jednotlivé výsledky zaznamenať, bude aplikácia poskytovať možnosť uložiť údaje z jednotlivých tabuliek vo forme súboru spustiteľného

v programe *Excel*. Okrem toho bude tiež možné uložiť vykreslený graf vo forme obrázku.

Pri ukladaní hodnôt z tabuľky do *Excelu* využijeme *technológiu OLE*. *OLE* je skratka z anglického *„Object Linking and Embeding“*, čo v preklade znamená *„Vkladanie a prepojovanie objektov*“. *OLE* je podporované samotným tvorcom operačného systému, teda spoločnosťou Microsoft, čiže nie je výmyslom firmy Borland. Technológia *OLE* je postavená na *modeli COM* (*„Component Object Model*“ – komponentový objektový model). *COM* model umožňuje interakciu medzi aplikáciami a komponentami. Je to špecifikácia popisujúca vytváranie objektov a spôsob prístupov k ich rozhraniam. Tým sú sprostredkovávané funkcie týchto objektov.

Mechanizmus *OLE* založený na modeli *COM* umožňuje vkladať do jednej aplikácie objekt definovaný v inej aplikácii. Pomocou serveru *OLE* tiež ponúka používanie funkcií inej aplikácie (serveru *OLE*). Typickým serverom *OLE* je napríklad Microsoft Word, pretože dokáže pomocou mechanizmu *OLE* poskytovať klientom *OLE* svoje funkcie. Podobným serverom je aj *Microsoft Excel*, ktorý budeme využívať v našej aplikácii (Kadlec, 2007), (Šindelář, 2007).

Pri programovaní budeme využívať rôzne údajové typy. Jedným z najdôležitejších bude použitie dynamických polí. Ich výhoda oproti statickým poliam spočíva v tom, že nemusia mať danú pevne stanovenú veľkosť hneď od začiatku programovania. Veľkosť dynamických polí sa nastavuje až počas behu programu. V našom prípade je nutné použiť dynamické polia hlavne preto, lebo ich veľkosť zistíme až potom, ako sa zadajú vstupné hodnoty *L* a *h* z ktorých sa táto veľkosť určí.

Aby sa dala aplikácia opätovne použiť pre ďalšiu rovnicu, bude vytvorené tlačidlo, pre vynulovanie aplikácie. V tejto časti sa nielen vymažú údaje z textových polí pre zadávanie vstupných hodnôt, ale sa aj vynulujú použité dynamické polia a potrebné premenné a program sa pripraví na ďalšie použitie.

Vytvorený program bude ošetrený tak, aby nebolo možné zadať neplatné vstupné údaje. Tými môže byť napríklad rovnica, ktorá nebude spĺňať požadovaný tvar alebo zadanie nečíselného údaju namiesto číselného. Ďalšie ošetrenia sa budú týkať tvorby tabuliek a grafov. Pokiaľ sa nevytvorí jedna z troch tabuliek, nebude možné k nim vykresliť graf. V prípade, že používateľ nebude mať ešte vytvorené prvé dve tabuľky

a zvolí možnosť vytvorenia tretej tabuľky s hodnotami po druhej numerickej integrácii, automaticky program vytvorí predchádzajúce dve a aj ich zobrazí.

Súčasťou tejto práce bude aj podrobný opis vytvorenej aplikácie. V ňom postupne uvedieme použité údajové typy, opíšeme úvodnú obrazovku spolu so zadávaním vstupných hodnôt, aké ošetrenia aplikácia bude obsahovať, hlavné menu, s ktorým súvisí vytváranie jednotlivých tabuliek, grafov, ukladanie hodnôt do súboru a podobne. Kompletný zdrojový kód vytvorenej aplikácie sa bude nachádzať na elektronickom nosiči v Prílohe 1.

Posledná časť bude zameraná na riešenie príkladov pre rôzne zaťažené nosníky. S princípmi riešenia úloh ohybu je dobré sa oboznámiť v príslušnej literatúre (Rédl, 2010a). Výsledky, ktoré pomocou aplikácie získame, budú porovnané s presným riešením získaným podľa vzorcov uvádzaných v literatúre. Uvedené bude aj riešenie v programe *MathCad*, ktorého výsledky sa takisto porovnajú s výsledkami, ktoré poskytne naprogramovaná aplikácia. Hodnoty, ktoré sa budú používať pri výpočtoch, budú udávané v základných jednotkách SI.

4 Vlastná práca a dosiahnuté výsledky

Vlastná práca je zameraná na vytvorenie aplikácie vo vývojovom prostredí *Delphi*. Zdrojový kód aplikácie sa kompiluje priamo do strojového kódu počítača a aplikácia je spustiteľná vo všetkých operačných systémoch *Windows*. Pre použitie vytvorenej aplikácie nie je potrebné rozhranie . NET.

Aplikácia bude využívať numerické metódy Runge – Kutta štvrtého rádu na riešenie diferenciálnej rovnice zadanej používateľom. Diferenciálna rovnica sa bude zvolenou metódou integrovať, pričom aplikácia ponúka vytvoriť ako prvú, tak aj druhú numerickú integráciu. Pri ladení aplikácie budeme ako testovaciu funkciu využívať funkciu $y'' = \sin(x)$. Ďalej uvedieme vzorové riešenia pre dva typy nosníkov, pričom využijeme odladenú aplikáciu. Riešenia pomocou aplikácie porovnáme s presným riešením. Ďalšiu rovnicu porovnáme s riešením, ktoré nám poskytne program *MathCad*.

4.1 Opis aplikácie

4.1.1 Použité údajové typy

Pri programovaní aplikácie sme využili jednoduché štandardné údajové typy ordinálne – *integer*, *boolean*, ale aj neordinálne – *extended*, ale aj zložitejšie štruktúry, a to dynamické polia.

V aplikácii máme globálne zadeklarované nasledovné štandardné údajové typy:

```
Var L,h:extended;
    n, metóda:integer;
    OK, naplnene, naplneneRK, naplneneRK2, vytvorene:boolean;
```

Názvy premenných väčšinou charakterizujú ich použitie v programe. Premenné typu *extended* ukladajú zadanú dĺžku nosníka *L* a krok výpočtu *h*, premenné typu *integer* ukladajú okrem poradového čísla metódy (1 alebo 2) aj hodnotu *n*, ktorá udáva na koľko bodov sa rozdelí dĺžka *L*. Premenné typu *boolean* sú v programe využívané väčšinou pri overovaní, či boli isté údaje alebo operácie uskutočnené alebo nie. Všetky tieto premenné majú po spustení aplikácie (vytvorení formuláru – jeho udalosť *OnCreate*) nastavenú hodnotu *false* (nepravda), ktoré počas behu programu podľa potreby menia svoju hodnotu.

Dynamické polia

Ako sme spomenuli, využili sme aj dynamické polia. Nevyužívame klasické statické polia preto, lebo veľkosť poľa vieme určiť až po tom, ako sa spustí aplikácia a používateľ zadá vstupné údaje. Preto sme na ukladanie hodnôt použili práve dynamické pole, ktorého veľkosť sa nastaví počas behu programu.

V programe sme si najskôr zadefinovali nový typ *pole* nasledovne:

```
Type pole= Array of extended;
```

V časti deklarácií máme potom deklarované premenné, ktoré sú typu *pole*:

```
Var rk_vstupX, rk_vystupY, rk_vystupY1, rk_vystupY2:pole;
```

Veľkosť týchto polí sa nastavuje pomocou procedúry *SetLength* v priebehu programu. Používateľ najskôr zadá rovnicu, dĺžku L a hodnotu kroku h a potvrdí nastavenie týchto hodnôt. Vtedy sa určí veľkosť takéhoto dynamické poľa ako podiel dĺžky L a kroku h a uloží sa do premennej n. Hodnoty premenných typu pole, ktoré sa v programe používajú, sa nastavia na požadovanú veľkosť, keď používateľ bude chcieť vyrátať a zobraziť hodnoty v tabuľke.

Premenná *rk_vstupX* obsahuje hodnoty z intervalu 0 až L, pričom rozdiel dvoch susedných bodov je rovný kroku h. Táto premenná sa využíva pri všetkých troch tabuľkách, ktoré aplikácia poskytuje. Vo všetkých troch prípadoch obsahuje premenná *rk_vstupX* hodnoty, ktoré budú nanášané na os x. Hodnoty osi y sa menia podľa toho, o ktorú tabuľku pôjde.

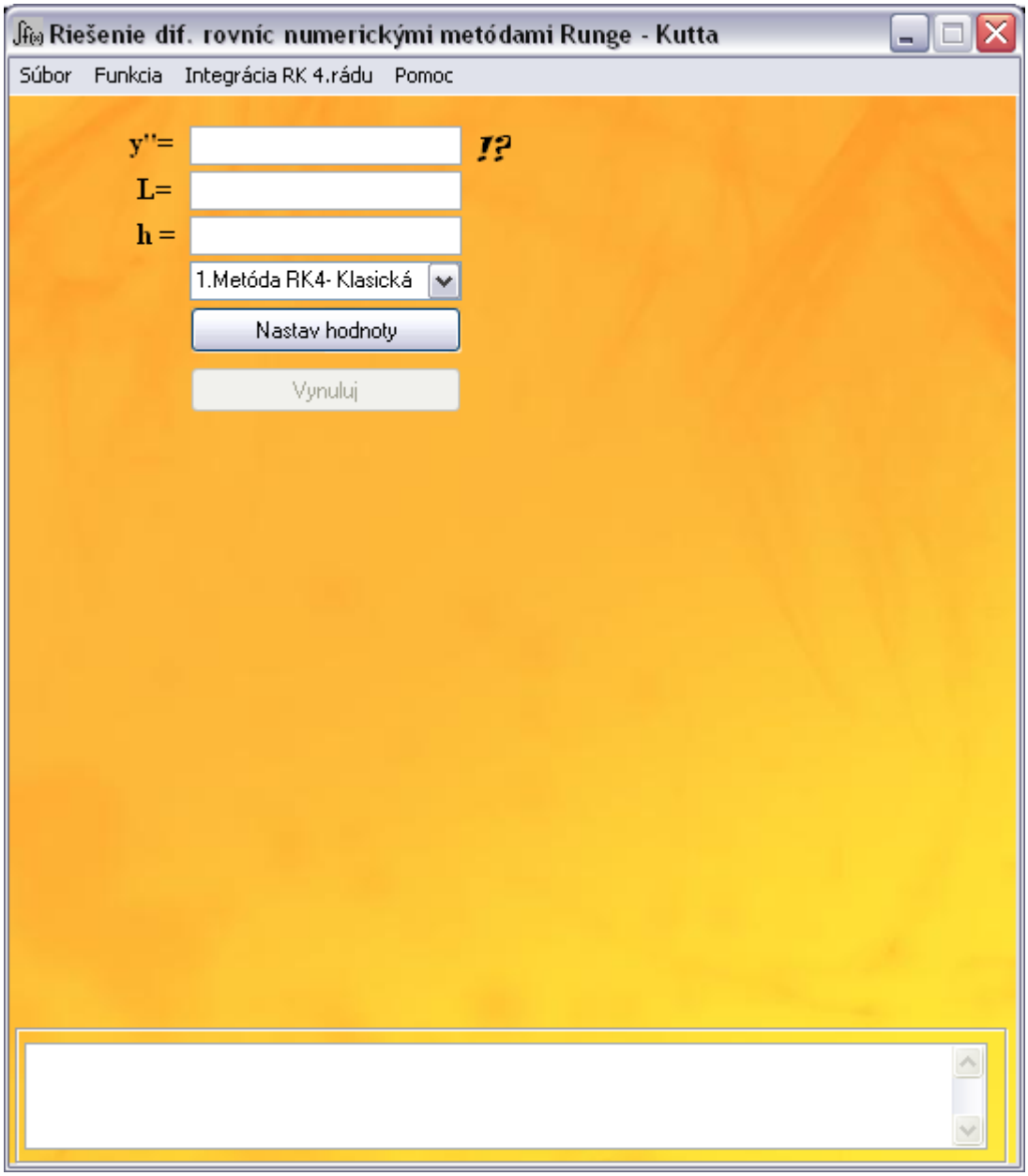
Ak sa vyžiada vytvorenie prvej tabuľky, uskutoční sa pomocou komponentu *Express* výpočet funkčných hodnôt pre každý prvok poľa *rk_vstupX*. Príslušné funkčné hodnoty sa uložia do dynamického poľa *rk_vystupY*. Pre vytvorenie prvej tabuľky a jej príslušného grafu sa využívajú práve tieto dve premenné.

Pre druhú tabuľku sa na prvky poľa *rk_vystupY* aplikuje procedúra s vybranou numerickou metódou Runge – Kutta. Všetky hodnoty, ktoré boli integrované, sa následne uložia do premennej *rk_vystupY1*. Pri vykresľovaní tabuľky a grafu pre prvú numerickú integráciu sa teda použijú dynamické polia *rk_vstupX* (os x) a *rk_vystupY2* (os y).

Pri tvorení tretej tabuľky je postup rovnaký ako pri tvorení druhej, avšak integrovať sa budú hodnoty poľa *rk_vystupY1* a výsledky sa uložia do premennej *rk_vystupY2*.

4.1.2 Úvodná obrazovka a zadávanie vstupných hodnôt

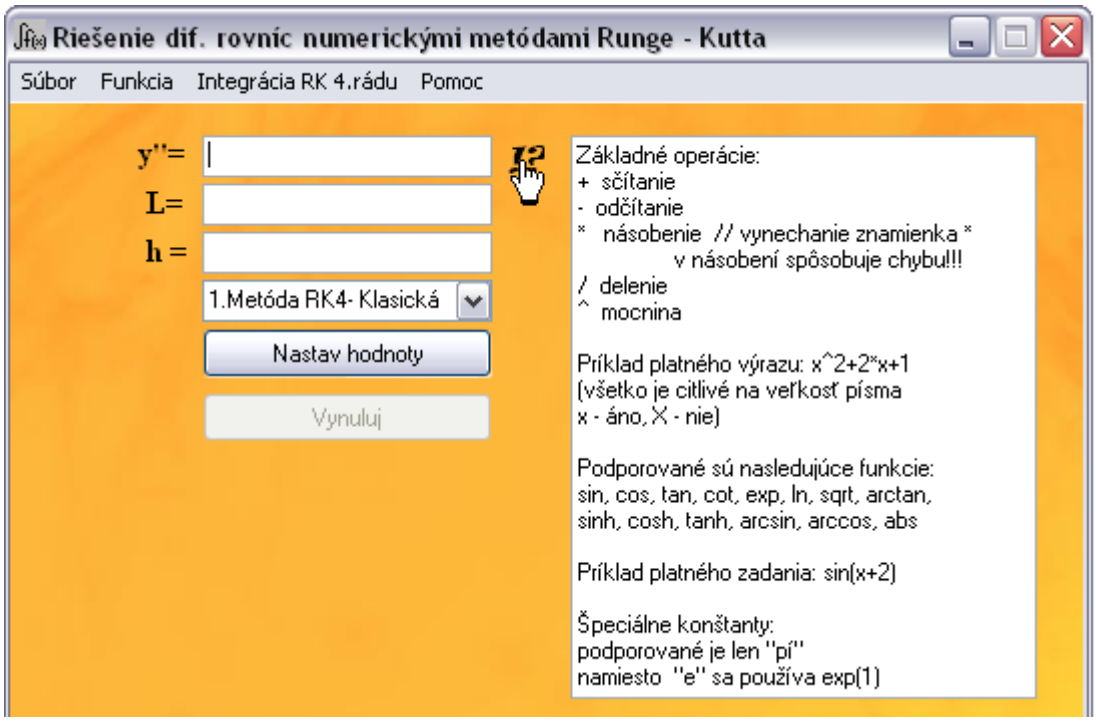
Na obr. 11 je znázornená úvodná obrazovka programu, ktorý je vytvorený vo vývojovom prostredí Delphi. Na tomto obrázku vidíme polia, kde sa vstupné údaje majú zadať, ako aj hlavné menu aplikácie.



Obr. 11 Úvodná obrazovka vytvorenej aplikácie

Keďže aplikácia slúži na riešenie diferenciálnych rovníc, používateľ zadá pri spustení programu ako prvú práve rovnicu, s ktorou chce pracovať. Okrem tejto rovnice, zadáva používateľ aj ďalšie hodnoty. Prvou je hodnota *L*, ktorá predstavuje (pri aplikačných výpočtoch uvedených v kap. 4.2) dĺžku nosníka (obr. 2) a hodnotu *h*, ktorá predstavuje veľkosť kroku, ktorý budú metódy Runge Kutta využívať. Čím menší krok zadáme, tým sa rozdelí dĺžka *L* na viac častí a vytvorí sa tak viac bodov, v ktorých sa budú uskutočňovať výpočty a dosiahneme presnejšie výsledky.

Aby sa používateľ lepšie orientoval pri zadávaní rovnice, je vytvorený pomocný popis. Na vytvorenie pomocného popisu sme využili komponent *Memo*. Jeho vlastnosť *Visible* je nastavená na hodnotu *false* (nepravda), čo znamená, že ho nevidieť hneď po spustení, ale až keď si to používateľ vyžiada. Obsahom tohto komponentu je pomocný popis, ktorý sa týka tvaru zadávanej rovnice – aké operácie a funkcie sú podporované a na čo si treba pri zadávaní dať pozor (obr.12).



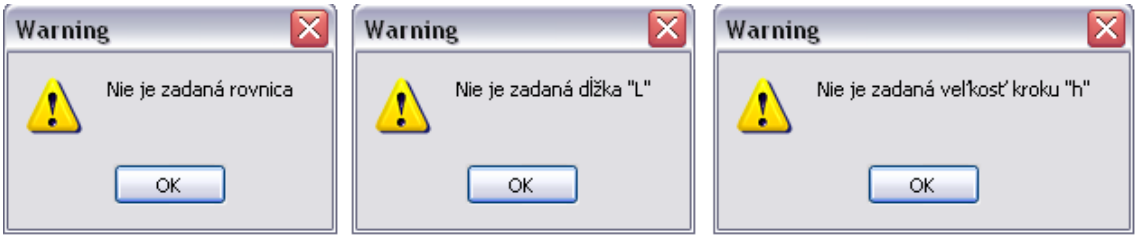
Obr. 12 Pomocný popis k zadaniu platného tvaru rovnice

Keď používateľ kurzorom myši prejde ponad znaky *„!?“* nachádzajúci sa napravo od textového poľa určeného na zadanie rovnice, objaví sa tento pomocný popis v pravej časti úvodného okna. Znaky *„!?“* sú vytvorené pomocou komponentu *Label* (popisné pole). Pri tomto popisnom poli využívame jeho udalosti *OnMouseMove* a *OnMouseLeave*. Tieto dve udalosti slúžia na zistenie, či sa kurzor myši nachádza

(*OnMouseMove*) alebo nenachádza (*OnMouseLeave*) nad popisným poľom. Ak sa kurzor myši bude nachádzať nad poľom, napravo od neho sa zviditeľní textová plocha s pomocným popisom, ktorý doteraz používateľ nevidel.

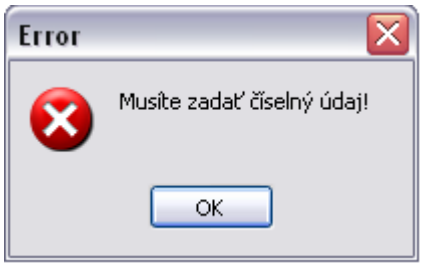
Ďalšou úlohou používateľa je zvoliť si v *ComboBoxe* metódu Runge – Kutta, pomocou ktorej sa uskutoční integrácia. Na výber má dve metódy, a to klasickú metódu Runge – Kutta štvrtého rádu alebo metódu Runge – Kutta – Gill, ktorá je tiež štvrtého rádu.

Tlačidlo *„Nastav hodnoty“* slúži na potvrdenie zadaných hodnôt. Po stlačení tohto tlačidla sa ako prvé zisťuje, či textové polia (*edity*) nie sú prázdne. Táto kontrola sa uskutočňuje pomocou podmieneného príkazu. Ak sa zistí, že používateľ nezadal rovnicu, zobrazí sa varovanie (obr. 13).



Obr. 13 Upozornenie pre používateľa ak nenastaví základné hodnoty a údaje

Ak používateľ zadá do polí *L* a *h* nečíselný údaj, program ho tiež upozorní, tento krát chybovým hlásením. Ak by táto kontrola nebola zabezpečená, mohlo by sa stať, že pri konvertovaní reťazca zadaného v *edite* na číslo by nastala chyba a program by zostal nefunkčný. Ak používateľ nezadá číselnú hodnotu do prvého a druhého *editu*, zobrazí sa nasledovné upozornenie:



Obr. 14 Chybové hlásenie v prípade, ak sa nezadá číselná hodnota

Toto ošetrenie je uskutočnené pomocou procedúry *Kontrola*, ktorej hlavička je nasledovná:

```
procedure TForm1.Kontrola(ret:string);
```

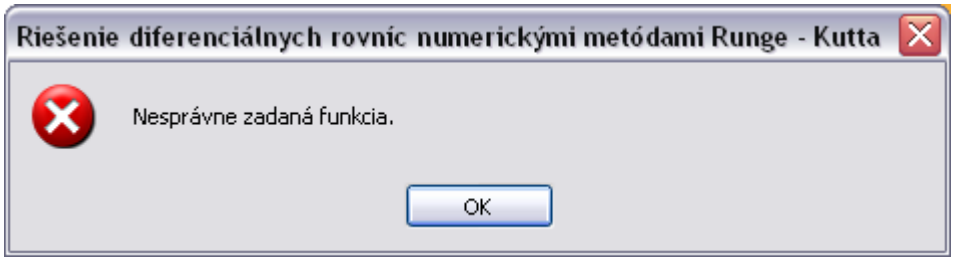
Parameter *ret* typu *string* sa pri volaní procedúry nahradí textom z *editu*, pre ktorý chceme previesť kontrolu. V prípade, že používateľ nezadá požadované údaje ako číselné hodnoty, dostane o tom chybové hlásenie a program mu neumožní uskutočniť ďalšie operácie až pokým nezadá správne údaje.

V rámci tejto procedúry sa využívajú funkcie pre prácu so *stringom* uvedené v tab. 2.

Tab. 2. Funkcie pre prácu so stringom(Skalka, 2007)

Funkcia	Popis	Syntax príklad
Pos	zistí či sa jeden reťazec nachádza v inom	`pozicia:=Pos(hladany,retazec);` ak sa obsah premennej *hladany* nachádza v obsahu premennej *retazec*, do premennej *pozicia* sa uloží poradové číslo (index) znaku, na ktorom začína zadaný hľadaný reťazec v zadanom prehľadávanom reťazci, pokiaľ sa nevyskytuje, *Pos* vráti hodnotu 0,
Delete	vymaže z reťazca podreťazec	`Delete(retazec,zaciatocnaPozicia,pocetZnakovNaVymazanie);` z premennej *retazec* vymaže časť začínajúcu na danej začiatočnej pozícii a pozostávajúci zo zadaného počtu znakov,
Val	skonvertuje reťazec (string) na číslo, v prípade neúspechu nepreruší vykonávanie programu, ale uloží pozíciu nepreložiteľného znaku do premennej	`Val(stringSCislom,ciselnaPremenna,kod);` *ciselnaPremenna* – obsahuje konvertovanú hodnotu, *kod* - vracia pozíciu c reťazci, na ktorej ak bol priebeh bezchybový nadobúda 0.

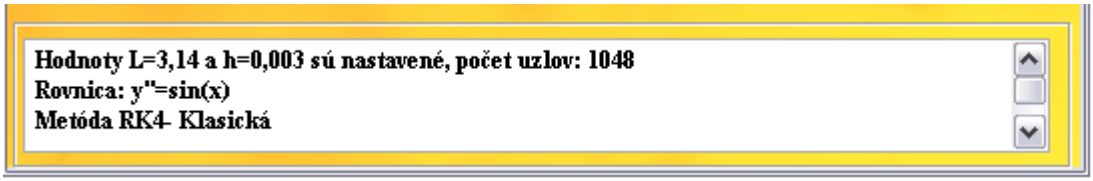
Ak sú splnené všetky doteraz spomenuté podmienky, program pokračuje načítaním rovnice. Taktiež je zabezpečené, aby bola táto zadávaná rovnica platného tvaru. Ak používateľ spraví chybu pri zadávaní, bude na to upozornený (obr. 15).



Obr. 15 Chybové hlásenie v prípade, že sa zadá neplatný výraz

Kvôli tomuto ošetreniu sme nemuseli vytvárať vlastnú procedúru, pretože komponent *Express* uskutoční túto kontrolu sám. Stačí, ak v časti, kde sa kontrolujú hodnoty zadávané do jednotlivých *editov* zadáme príkaz priradenia výrazu v *Edit1* do vlastnosti *Expression* komponentu *Express*. *Express* vyhodnotí, či je rovnica požadovaného tvaru ak nie, upozorní používateľa a pokiaľ nezadá platný tvar rovnice, neumožní mu uskutočniť žiadne iné funkcie.

Ak všetky spomenuté ošetrenia sú vyhodnotené kladne, potvrdenie tlačidlom je úspešné a zadané údaje sa nastavia a uložia do príslušných premenných. Následne sa v dolnej časti aplikácie zobrazí potvrdenie o nastavení hodnôt (obr. 16) a tlačidlo *„Nastav hodnoty“* sa zablokuje.



Obr. 16 Potvrdenie nastavených hodnôt

Na výpis potvrdenia o nastavení hodnôt sme využili ďalšie *Memo*. Toto *Memo* ako aj *Memo* s pomocným popisom (obr. 12), majú nastavenú vlastnosť *ReadOnly* na hodnotu *true* (pravda), čo znamená, že používateľ do nich nemôže zapisovať žiadne údaje a slúžia len na čítanie.

Ak sú všetky kontrolné ošetrenia vyhodnotené správne, uskutoční sa v rámci udalosti *OnClick* tlačidla *„Nastav hodnoty“* uloženie potrebných vstupných údajov do príslušných premenných:

```
Express1.Expression:=Edit1.Text;
metoda:=strtoint(ComboBox1.Text[1]);
L:=StrToFloat(Edit2.Text);
h:=StrToFloat(Edit3.Text);
n:=round(L/h)+1;
```

Na uloženie vybranej metódy v tomto prípade zvolíme len zapamätanie prvého znaku textu, ktorý sa nachádza v *ComboBoxe*. Keďže sú metódy Runge – Kutta zadávané do *ComboBoxu* v tvare *poradové číslo + názov metódy*, uloží sa nám takto len poradové číslo metódy (1 alebo 2), ktoré sa neskôr v programe bude využívať pri rozhodovaní, ktorú procedúru zavolať a podobne.

Aplikácia obsahuje aj druhé tlačidlo s názvom *„Vynuluj“*, ktoré nie je prístupné hneď po štarte aplikácie, ale až po potvrdení hodnôt sa odblokuje. Ak bude chcieť používateľ pracovať s novou rovnicou, po stlačení tohto tlačidla sa mu textové polia pre zadávanie vstupných údajov vyprázdnia, tlačidlo *„Nastav hodnoty“* sa odblokuje a druhé tlačidlo sa opäť zneprístupní. Po udalosti *Click*, ktorá sa viaže k tlačidlu *„Vynuluj“*, sa vyprázdnia textové polia na zadávanie vstupných hodnôt a zavolá sa procedúra *Nuluj*, ktorá slúži na vynulovanie potrebných premenných. Premenné typu *boolean* sa nastavia na hodnotu *false* a aplikácia sa pripraví na zadanie novej rovnice. Táto procedúra sa volá aj v prípade, ak už boli nastavené zadané vstupné údaje a zmení sa niektorá z hodnôt v *editoch.* Volanie procedúry sa uskutočňuje v udalosti *OnChange* každého *editu*. Pre *Edit1* je kód nasledovný:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
nuluj;
end;
```

Udalosť *OnChange* použitého *ComboBoxu* je ošetrená tak isto ako *Edity* – zavolá sa procedúra *Nuluj* v prípade, že sa zmení výber metódy.

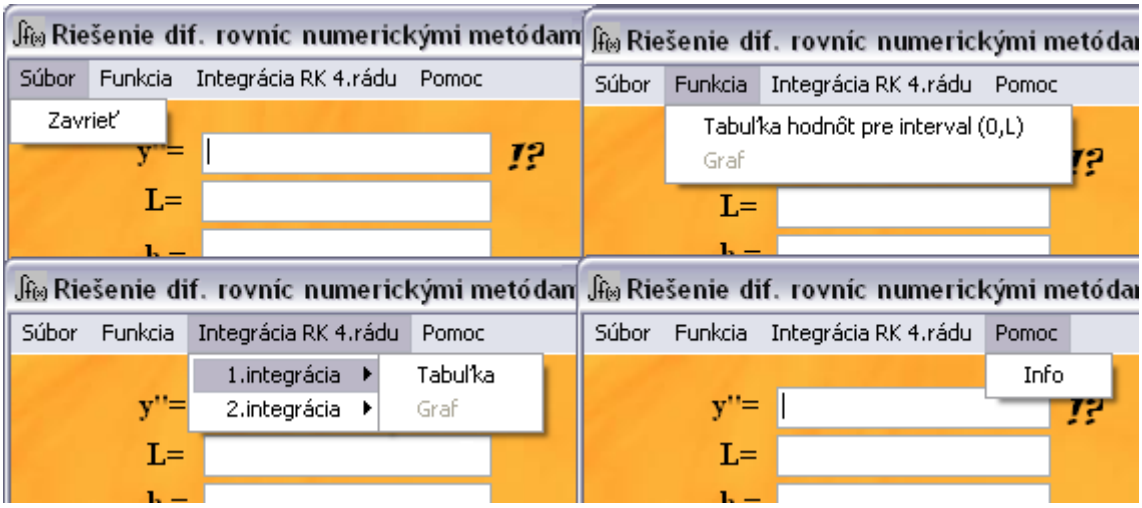
4.1.3 Opis Menu programu

Na sprehľadnenie aplikácie sme využili komponent *MainMenu*. Ten umožňuje vytvoriť pre používateľa prijateľné hlavné menu s rôznymi možnosťami, ktoré bude aplikácia schopná uskutočniť. Delphi má k dispozícií takzvaný *Menu Designer* (obr. 17), v ktorom sa dá ľahko previesť detailný návrh položiek menu.

Po dvojkliku na každú z položiek sa automaticky vytvorí procedúra, kde sa potom jednoducho naprogramuje, čo sa má stať, ak používateľ danú položku zvolí. Menu vytvorené pomocou *Menu Designera* vyzerá rovnako, ako bude vyzerať výsledné menu. Jediný rozdiel je v tom, že ak niektorej položke nastavíme vlastnosť *Visible* alebo *Enabled* na nepravda (*false*), používateľovi sa buď nezobrazí (*visible*) alebo sa zobrazí zablokované (*enabled*). V našom programe využívame druhú spomenutú vlastnosť, a to *Enabled*. Blokovaním určitých položiek zabezpečíme, aby používateľ napríklad nemohol dať vykresliť graf skôr, ako sú nadobudnuté hodnoty v tabuľke, alebo aby sa znova nevytvorila tabuľka, keď už raz vytvorená bola.

Obr. 17 Menu Designer s položkami vytvorenej aplikácie

Pomocou takto vytvoreného menu sa aplikácia zároveň aj ovláda. Jednotlivé položky menu sú zobrazené na obr. 18. Používateľovi menu poskytuje možnosť vytvoriť a zobraziť tabuľku s hodnotami prislúchajúcimi rovnici, ktorá sa zadá a následne tabuľky hodnôt prvej a druhej integrácie. Po vytvorení tabuliek sa používateľovi sprístupní aj možnosť vykresliť grafy patriace príslušným tabuľkám. Prvá položka menu s názvom „*Súbor*“ obsahuje len jedinú možnosť „*Zavrieť*“. Ostatné položky zahŕňajú v sebe viacero možností. V nasledujúcich podkapitolách si jednotlivé položky takto vytvoreného hlavného menu opíšeme podrobnejšie.



Obr. 18 Položky hlavného menu, ktoré slúži na ovládanie aplikácie

4.1.3.1 Položka *„Funkcia“*

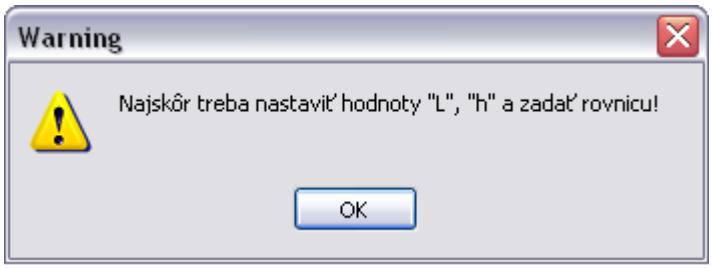
V rámci tejto položky má používateľ k dispozícii dve možnosti, ktoré sa vzťahujú k rovnici, ktorú zadáva na začiatku, konkrétne vytvorenie tabuľky hodnôt pre interval $< 0, L >$ a vytvorenie grafu.

Po zvolení prvej možnosti sa objaví tabuľka (obr. 19), v ktorej budú hodnoty zo spomenutého intervalu, pričom rozdiel dvoch susedných hodnôt je rovný kroku *h*. Počet riadkov tabuľky závisí od zadaných hodnôt *L* a *h*, z ktorých sa vyráta počet bodov a uloží do premennej *n*.

Druhá možnosť položky *„Funkcia*“ umožňuje vykresliť graf k vytvorenej tabuľke hodnôt, avšak táto možnosť je zablokovaná až pokým sa nevytvorí tabuľka hodnôt.

Vytvorenie tabuľky

Pri tvorení tabuliek je zabezpečené, že používateľovi aplikácia neumožní vytvoriť tabuľku skôr, ako nastaví potrebné vstupné hodnoty. V tomto prípade sa mu zobrazí varovné hlásenie (obr. 19).



Obr. 19 Varovné hlásenie pri vytvorení tabuľky skôr ako sa nastavia hodnoty

Táto kontrola je zabezpečená pomocou premennej typu *boolean,* ktorá svoju pôvodne nastavenú hodnotu (*false*) mení na *true*, ak sú po stlačení tlačidla *„Nastav hodnoty“* všetky údaje zadané správne. Ošetrenie je teda uskutočnené tak, že pokiaľ premenná nenadobudne hodnotu *true*, neumožní aplikácia vytvoriť nielen túto tabuľku, ale ani tabuľku pre prvú a druhú numerickú integráciu.

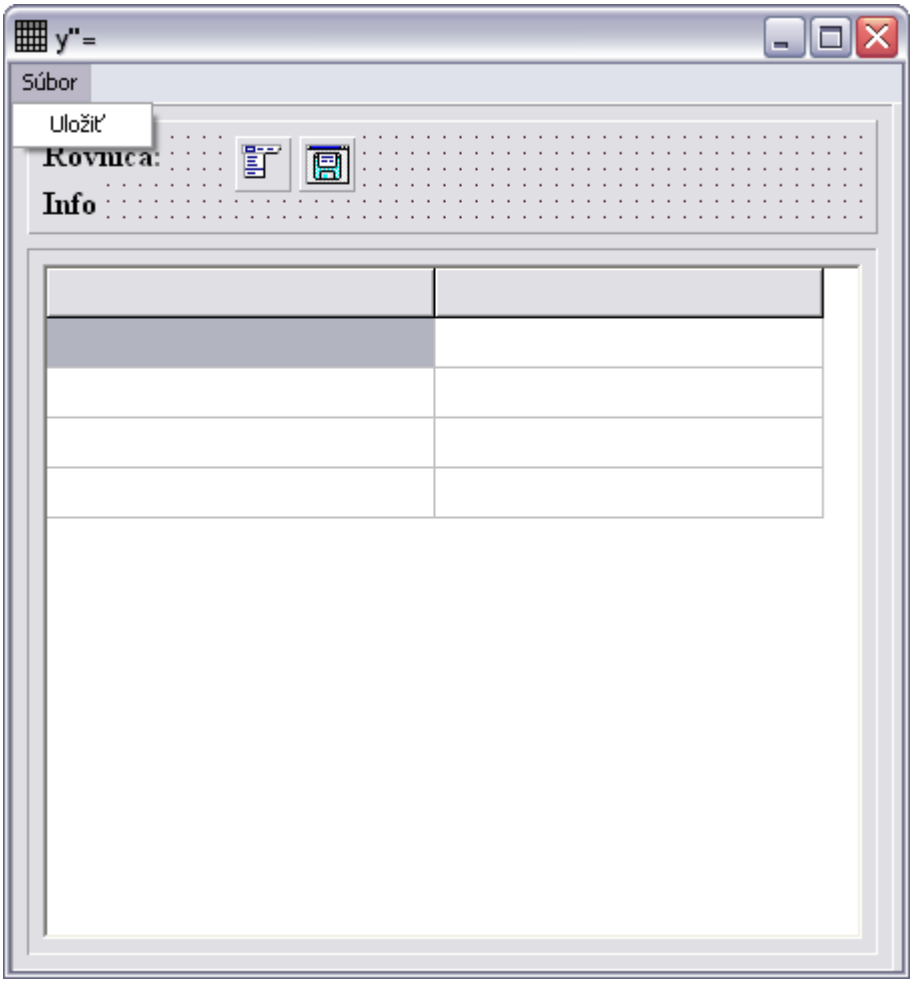
Tabuľky, ktoré bude aplikácia poskytovať sú vytvorené pomocou nového formuláru, na ktorý sme umiestnili komponent *StringGrid.* Počet riadkov *Stringridu* nie je v tomto prípade pevne nastavený (*StringGrid* na obr. 20 obsahuje zatiaľ len päť riadkov), pretože túto informáciu zisťujeme až v priebehu programu.

Pomocou vlastnosti *RowCount* nastavíme požadovaný počet riadkov až keď sa tabuľka bude vytvárať:

```
Form3.StringGrid1.RowCount:=n+1;
```

Hodnota *n* je zvýšená o jedna preto, lebo prvý riadok *StringGridu* bude slúžiť ako záhlavie tabuľky, kde bude informácia o názve stĺpca tabuľky .

Počet stĺpcov je pevne stanovený na dva už od začiatku. Jeden stĺpec reprezentuje hodnoty *x* a druhý príslušné funkčné hodnoty.



Obr. 20 Formulár pre vytváranie tabuliek

Pri napĺňaní tabuľky funkčnými hodnotami sme využili komponent *Express* slúžiaci na rýchle vyhodnocovanie výrazov. Pri práci s *Expressom* ako prvé uskutočníme priradenie výrazu, v našom prípade rovnice zadanej v *Edite1*.

Toto priradenie uskutočníme pomocou vlastnosti *Expression*:

```
Express1.Expression:=Edit1.Text;
```

Komponent *Express* obsahuje funkciu nazvanú *TheFunction*, ktorá je volaná, kedykoľvek je potrebné vyhodnotiť daný výraz. Práve túto funkciu využívame v našej aplikácii. Ak je zadaná rovnica platná, môže sa vyhodnotiť práve funkcia *TheFunction*. Táto funkcia sa využíva v procedúre *Napln_pole*. Zdrojový kód tejto procedúry má nasledujúci tvar:

```
procedure Tform1.Napln_pole(var hodnotaX, hodnotaY:pole);
var i:integer;
    x:extended;
begin
x:=0;
for i:=low(hodnotaX) to high(hodnotaX) do
  begin
    hodnotaX[i]:=x;
    hodnotaY[i]:=Express1.TheFunction(x,0,0);
    x:=x+h;
  end;
end;
```

Parametre *hodnotaX* a *hodnotaY* sú dynamické polia, ktoré sa pri volaní nahradia premennými *rk_vstupX* a *rk_vystupY*. Tieto premenné budú obsahovať vypočítané hodnoty, ktoré sa následne zobrazia do buniek *StringGridu*. Na takýto výpis je v programe vytvorená ďalšia pomocná procedúra, ktorej hlavička vyzerá:

```
procedure tform1.vypis_pole(hodnotaX, hodnotaY:pole; c:integer);
```

Parametre *hodnotaX* a *hodnotaY* sa pri volaní tejto procedúry takisto nahradia premennými *rk_vstupX* a *rk_vytupY*. Táto procedúra sa bude využívať aj pri tvorení tabuliek prvej a druhej integrácie, kde sa pri volaní použijú premenné, ktoré obsahujú požadované integrované hodnoty .

Parameter *c* procedúry *Vypis_pole* sa pri volaní nahradí celočíselnou hodnotou, pričom každá tabuľka obsahuje svoju hodnotu. Táto hodnota sa potom pripočíta k vlastnosti *Left* formulára:

```
Form3.Left:=Form3.Left+c;
```

To znamená, že tabuľky sa nebudú prekrývať, ale každá sa vykreslí na inej pozícii, ktorej hodnota je uložená v spomenutej vlastnosti *Left*.

Telo tejto procedúry obsahuje nasledujúci algoritmus, ktorý slúži na výpis hodnôt dynamického poľa do buniek *StringGridu*:

```
for i:=low(hodnotaX) to high(hodnotaX)  do
  begin
    Form3.StringGrid1.Cells[0,i+1]:='  '+Floattostr(hodnota X[i]);
    Form3.StringGrid1.Cells[1,i+1]:='  '+Floattostr(hodnotaY [i]);
  end;
```

Low(hodnotaX) a *high(hodnotaX)* určujú rozsah dynamického poľa – v tomto prípade označeného *hodnotaX*. *Low* určuje prvú a *high* poslednú zložku poľa. Táto procedúra sa v programe volá v rámci udalosti *Click* príslušnej položky menu na vytvorenie tabuľky.

Okrem *StringGridu* obsahuje formulár aj dve popisné polia (*label)* a taktiež oddeľovacie čiary (*bevel*). Na obr. 20 vidieť jeden *label* s názvom *Rovnica* a druhý s názvom *Info*, do ktorých sa vždy zapíše rovnica zadaná používateľom a informácia o tom, o ktorú tabuľku a numerickú metódu ide. V prípade, že tvar popisného poľa s rovnicou by bol dlhší ako je šírka formulára, je zabezpečené, aby sa tomu tabuľka prispôsobila. Vytvorili sme procedúru *Prispôsob*, ktorá sa vzťahuje k tomuto formuláru. Volá sa vždy pri vykresľovaní tabuľky a zabezpečí, aby sa šírka oddeľovacích čiar, formuláru a *StringGridu* prispôsobila šírke popisného poľa s rovnicou. Pri *StringGride* je to nielen šírka celej tabuľky, ale aj šírka jednotlivých buniek.

Zdrojový kód tejto procedúry je nasledovný:

```
procedure TForm3.prisposob;
begin
  Bevel1.Width:=Label1.Width+14;
  Bevel2.Width:=Bevel1.Width+14;
  Bevel3.Width:=Bevel1.Width;
  StringGrid1.Width:=Bevel3.Width-14;
  StringGrid1.DefaultColWidth:=(StringGrid1.Width-23)div 2;
  Form3.Width:=Bevel2.Width+25;
end;
```

Tabuľky sú flexibilné, to znamená, že používateľ si môže zvoliť veľkosť, aká mu vyhovuje. Využili sme udalosť formuláru *OnResize*, kde sme podobne

ako v procedúre *Prisposob* nastavili ako sa má meniť šírka a výška jednotlivých oddeľovacích čiar, *StringGridu* a jeho buniek pri zmene veľkosti formuláru.

Pri tvorení tabuliek sme využili aj komponenty *SaveDialog* a *MainMenu.* Oba komponenty nám slúžia pre uloženie hodnôt zapísaných v *StringGride* do súboru, ktorý bude mať príponu *.xls*. Vytvorí sa nám tak dokument s hodnotami, ktoré si bude môcť používateľ otvoriť pomocou programu *Microsoft Office Excel*.

Komponent *MainMenu* na obr. 20, obsahuje len jednu položku, a to *„Uložiť“*. V udalosti *OnClick* tohto komponentu sa využíva volanie funkcie *UlozExcel*, ktorá je typu *boolean*. Hlavička tejto funkcie má tvar:

```
function tform3.UlozExcel(Tabulka:TStringGrid;
            Nazov_suboru:string):Boolean;
```

Prvý parameter *Tabulka* určuje, ktorá tabuľka sa má vyexportovať do súboru s názvom, ktorý je uložený v parametri *Nazov_suboru*.

Jadrom tejto funkcie je využitie technológie *OLE*. Pomocou tejto funcie sa budú hodnoty uložené v príslušnej tabuľke exportovať do *Excelu*. Predpokladáme, že ak si chce používateľ uložiť údaje do *Excelu*, má ho aj nainštalovaný v počítači. Ak by sa stalo, že v počítači sa tento program nenachádza, export bude neúspešný a aplikácia o tom podá hlásenie. Pre tento prípad je vytvorená funkcia *NainstalovanyExcel,* pri ktorej sa využíva práca s registrami. Z nich sa na príslušnom mieste zistí potrebná informácia o prítomnosti alebo neprítomnosti aplikácie *Excel* v systéme:

```
function tform3.NainstalovanyExcel: Boolean;
var Reg: TRegistry;
begin
Reg := TRegistry.Create;
try
  Reg.RootKey := HKEY_CLASSES_ROOT;
  Result := Reg.KeyExists('Excel.Application');
finally
  Reg.Free;
end;
end;
```

Keďže je táto funkcia typu *boolean* pri volaní zisťujeme, či súbor existuje alebo nie. Ak je výsledná hodnota kladná, môže aplikácia pokračovať v exportovaní hodnôt, ak nie, okrem chybového hlásenia sa nevykoná žiadna iná úloha.

Aby sme mohli využiť spomenutú technológiu *OLE*, museli sme pripojiť knižnicu *ComObj* do zoznamu knižníc v *Unite*, ktorý patrí formuláru, kde sa vytvára tabuľka a kde je zapísaná aj spomenutá funkcia *UlozExcel*. Pre túto funkciu sme najskôr zadeklarovali lokálne premenné:

```
var XLS,Harok,Hodnoty:OLEVariant;
    j:integer;
    nazovH:string;
```

Premenné *XLS*, *Harok a Hodnoty* sú typu *OLEVariant*. *XLS* nám bude prezentovať vytvorený súbor (zošit) v *Exceli*, *Harok*, ako vyplýva z názvu, prezentuje hárok v zošite a do premennej *Hodnoty* sa načítajú údaje, ktoré sa majú vyexportovať do *Excelu*. Najskôr sa určí rozsah, aký premenná *Hodnoty* bude mať a následne sa zapíšu hodnoty zo stĺpca pre *x* spolu s údajmi, ktoré budú slúžiť pre identifikovanie o ktorú metódu a integráciu ide:

```
Hodnoty:=VarArrayCreate([1,Tabulka.RowCount+1,1,Tabulka.ColCount],
varVariant);
  Hodnoty[1,1]:=Label2.Caption+'>'+Label1.Caption;
  Hodnoty[2,1]:='Os x';
  Hodnoty[2,2]:='Os y';
for j := low(rk_vstupX) to high(rk_vstupX)do
      Hodnoty[j + 3, 1] := rk_vstupX[j] ;
```

Keďže formulár pre vytváranie tabuliek je jednotný pre všetky tri tabuľky, ktoré aplikácia poskytuje, treba pri exportovaní do súboru rozlíšiť, hodnoty ktorej tabuľky dal používateľ uložiť. Na základe prvého znaku textu, ktorý obsahuje *label2* formuláru pre vykreslenie tabuľky zisťujeme, ktorá tabuľka je zvolená. Podľa toho, ktorá tabuľka je zvolená, sa uskutoční zapísanie hodnôt zo stĺpca *y* a taktiež sa do premennej *nazovH* uloží príslušný názov hárku:

```
nazovH:=Label2.caption;
case nazovH[1] of 'F':begin
                        nazovH:='Pôvodná rovnica';
                        for j:=low(rk_vstupX) to high(rk_vstupX) do
                           Hodnoty[j+3,2]:=rk_vystupY[j];
                      end;
                  '1':begin
                        NazovH:='1.integrácia';
```

```
                        For j:=low(rk_vstupX) to high(rk_vstupX) do
                          Hodnoty[j+3,2] := rk_vystupY1[j] ;
                      end;
                  '2':begin
                        nazovH:='2.integrácia';
                        for j:=low(rk_vstupX) to high(rk_vstupX) do
                          Hodnoty[j+3,2]:=rk_vystupY2[j] ;
                      end;
end;
```

Ak bude premenná *Hodnoty* naplnená, môže sa vytvoriť súbor v *Exceli* s hárkom, ktorý sa pomenuje názvom uloženým v premennej *nazovH* a pomocou funkcie *Bunky* sa vymedzí rozsah buniek, do ktorých sa v *Exceli* zapíšu údaje uložené v premennej *Hodnoty*. Ak sa nevyskytne žiadna chyba, uloží sa súbor pod názvom, ktorý obsahuje parameter *Nazov_suboru*, funkcia *UlozExcel* nadobudne hodnotu *true* (správne) a objekt *XLS* sa zavrie:

```
Result := False;
XLS := CreateOleObject('Excel.Application');
Try   XLS.Visible := False;
      XLS.Workbooks.Add();
      Harok := XLS.Workbooks[1].WorkSheets[1];
      Harok.Name := NazovH;
      Harok.Range[Bunky(1,1),Bunky(Tabulka.RowCount+1,Tabulka.
      ColCount)].Value := Hodnoty;
      Try   XLS.Workbooks[1].SaveAs(Nazov_suboru);
            Result := True;
      except
      ShowMessage('Chyba pri ukladaní hodnôt!')
      end;
finally XLS.quit;
end;
```

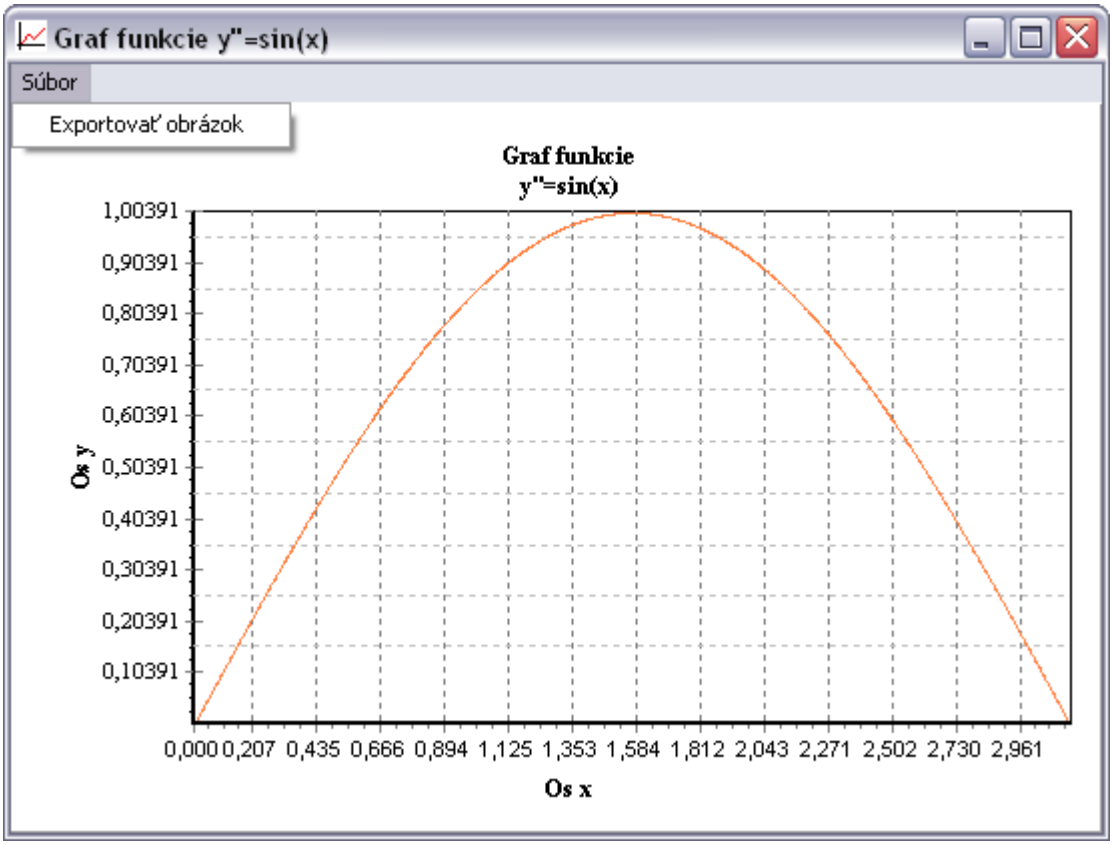
Spomenutá funkcia *Bunky* slúžia na vymedzenie rozsahu buniek, kde sa údaje vyexportujú. Zdrojový kód tejto funkcie je nasledovný:

```
function tform3.Bunky(riadok, stlpec: Integer): string;
begin
  Result := Chr(Ord('A') + stlpec - 1) + IntToStr(riadok);
end;
```

Výsledkom tejto funkcie v našej aplikácii je vždy rozpätie od *A1* až po *BX*, kde *X* je číslo reprezentujúce počet riadkov príslušnej tabuľky (*StringGridu*) zväčšený o 1.

Vytvorenie grafu

Druhá možnosť položky *„Funkcia“* nazvaná *„Graf“*, na základe vypočítaných hodnôt z tabuľky zostrojí a vykreslí graf (obr. 21), ktorý sa zobrazí v samostatnom okne. Keďže k vytvoreniu grafu sú potrebné údaje z tabuľky, možnosť vykresliť graf je spočiatku zablokovaná a sprístupní sa až po tejto časti. Veľkosť grafu je podobne ako veľkosť tabuľky flexibilná, takže používateľ si ju môže nastaviť tak, ako mu vyhovuje.



Obr. 21 Príklad grafu, ktorý vytvára aplikácia

Keďže sa graf zobrazí v samostatnom okne, vytvorili sme nový formulár, na ktorý sme umiestnili komponent *Chart* slúžiaci na grafickú prezentáciu hodnôt z tabuľky. Na napĺňanie grafu, resp. komponentu *Chart* údajmi, je vytvorená procedúra *Nakresli_graf* a jej hlavička vyzerá nasledovne:

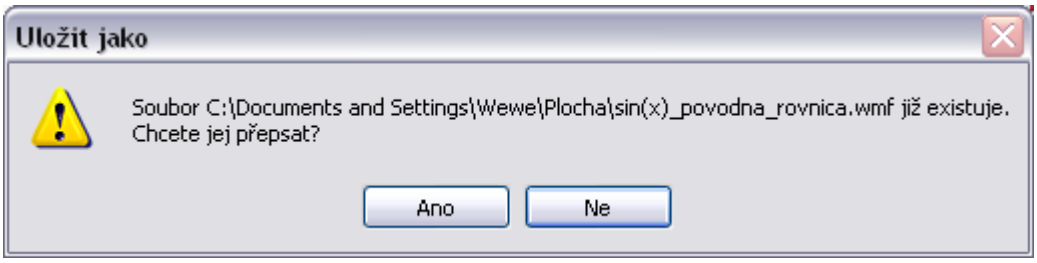
```
procedure TForm1.Nakresli_graf(hodnotaX,hodnotaY:pole;c:integer);
```

Parametre *hodnotaX* a *hodnotaY* sú pri volaní procedúry nahradené premennými typu pole, z ktorých sa má graf zostrojiť. Parameter *c*, podobne ako v procedúre *Vypis_pole* určuje, na akej pozícii sa má formulár s grafom zobraziť.

Algoritmus na zadanie údajov pre graf, ktorý je súčasťou tela procedúry *Nakresli_graf* je nasledovný:

```
with Form2.Series1 do
    begin
        for i:=low(hodnotaX) to high(hodnotaX) do
         Add(hodnotaY[i],FormatFloat('0.000',hodnotaX[i]));
    end;
```

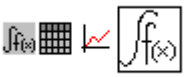
Okrem komponentu *Chart* sme využili aj komponent *MainMenu*. Pomocou neho sa používateľovi poskytne možnosť zobrazený graf vyexportovať ako obrázok (obr. 20). Pri exportovaní obrázku využívame komponent *SavePictureDialog*. Pomocou tohto komponentu a jeho funkcie *SaveToMetafile* program ponúkne používateľovi možnosť zvoliť miesto uloženia a názov obrázku vo formáte *.wmf*. Využili sme vlastnosť dialógu *ofOverwritePrompt*, ktorá zobrazí upozornenie, ak sa používateľ pokúsi prepísať už existujúci súbor (obr. 22).



Obr. 22 Upozornenie v prípade zadania názvu už existujúceho súboru

Vytvorenie vlastných ikon

Keďže aplikácia bude počas svojho behu otvárať rôzne okná (tabuľky, grafy), využili sme nástroj vývojového prostredia *Delphi Image Editor* na vytvorenie ikon, ktoré odlíšia jednotlivé formuláre. Prostredie *Delphi* všetkým svojím aplikáciám automaticky priradí tú istú štandardnú ikonu. Pre vytvorenie vlastných ikon k našej aplikácii sme sa rozhodli hlavne kvôli tomu, aby mala atraktívnejší vzhľad pre používateľa. Formulár v *Delphi* obsahuje vlastnosť *Icon*, pomocou ktorej sme každému z použitých formulárov nastavili príslušnú ikonu. Vytvorili sme tri ikony veľkosti 16x16 a jednu 32x32 (obr. 23). Programy v operačnom systéme *Windows* majú obvykle dve ikony. Jednu (veľkú) pre zobrazenie napríklad na ploche a druhú (malú) napríklad kvôli panelu nástrojov alebo záhlaviu okna. Preto máme vytvorené spomenuté malé ikony pre záhlavia okien hlavného formuláru a formulárov pre vytvorenie tabuliek a grafov a jednu veľkú pre *exe súbor*.



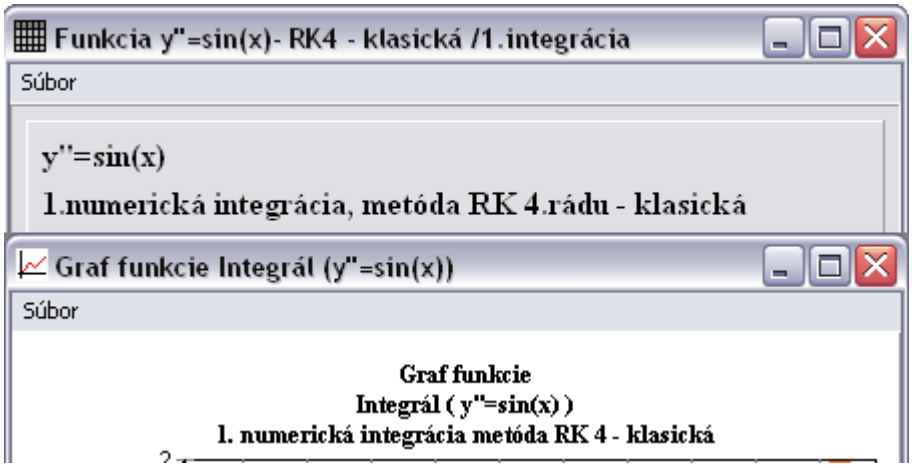
Obr. 23 Vlastné ikony použité v aplikácii

Pomocou *Image Editora* sme vytvorili aj *Resource File (.res)*, ktorý sme nazvali *ikony.res*, kde sme zadali názvy vytvorených ikon a pripojili sme ho v časti *implementation:*

```
implementation
{$R *.dfm}
{$R ikony.res}
```

4.1.3.2 Položka *„Integrácia RK – 4.rádu“*

Táto možnosť v sebe skrýva dve časti – prvá a druhá integrácia numerickou metódou Runge – Kutta štvrtého rádu. Podobne ako v prípade položky *„Funkcia“*, v každej z týchto častí sa dá zostrojiť tabuľka príslušných hodnôt a graf. Tabuľky a grafy sú takého istého formátu ako na obr. 20 a 21, líšia sa len záhlavím a označením, ktoré nesie informáciu o tom, ku ktorej integrácii tabuľka patrí a o akú metódu ide (obr. 24).



Obr. 24 Príklad označenia tabuľky a grafu (1. integrácia klasickou metódou Runge – Kutta)

Podľa toho, ktorú z metód používateľ zvolí pri zadávaní vstupných hodnôt, využije sa pri vytváraní oboch tabuliek buď procedúra *RK4_klasicka alebo RK4_Gill*. Tieto

procedúry využívajú numerické metódy Runge Kutta štvrtého rádu, pričom sa tak vytvára integrácia diferenciálnej rovnice zadanej v *Edite1*.

Zdrojový kód procedúry *RK4_klasicka* má nasledovný tvar:

```
procedure TForm1.RK4_klasicka(y:pole; var Y1:pole);
var k1,k2,k3,k4:extended;i:integer;
begin
for i:=low(y)+1 to high(y) do
  begin
    Y1[i]:=y1[i-1]+h;
    k1:=y[i-1];
    k2:=y[i-1]+(y[i]-y[i-1])/2;
    k3:=k2;
    k4:=y[i];
    Y1[i]:=Y1[i-1]+h/6*(k1+2*k2+2*k3+k4);
  end;
end;
```

Pre Gillovu metódu má procedúra zdrojový kód:

```
procedure TForm1.RK4_Gill(y:pole; var Y1:pole);
var k1,k2,k3,k4:extended;i:integer;
begin
for i:=low(y)+1 to high(y) do
  begin
    Y1[i]:=Y1[i-1]+h;
    k1:=y[i-1]*h;
    k2:=(y[i-1]+(k1/2))*h;
    k3:=(y[i-1]+((1/Sqrt(2)-0.5)*k1)+((1-(1/Sqrt(2)))*k2))*h;
    k4:=(y[i-1]-((1/Sqrt(2))*k2+((1+(1/Sqrt(2))))*k3))*h;
    Y1[i]:=Y1[i-1]+(1/6*(k1+((2-Sqrt(2))*k2)+((2+Sqrt(2))*k3)+k4));
   end;
end;
```

Pri prvej numerickej integrácii sa pri volaní oboch procedúr využijú premenné *rk_vystupY* a *rk_vystupY1*. Znamená to, že sa uskutoční numerická integrácia metódou Runge – Kutta pre hodnoty premennej *rk_vystupY*, ktorá obsahujú funkčné hodnoty rovnice zadanej používateľom, a uložia do premennej *rk_vystupY1*.

Pri druhej integrácii sa volanie uskutoční s premennými *rk_vystupY1* a *rk_vystupY2* a podobne ako v prvom prípade, sa integrované hodnoty premennej *rk_vystupY1* uložia do rk*_vystupY2*.

Na volanie procedúry na základe vybranej metódy využívame príkaze *case*:

```
case metoda of 1:RK4_klasicka(rk_vystupY, rk_vystupY1);
               2:RK4_Gill(rk_vystupY, rk_vystupY1);
end;
```

Keď sa uskutočnia tieto výpočty, nasleduje volanie procedúry *Vypis_pole* s prvou premennou *rk_vstupX* v oboch prípadoch a druhou premennou *rk_vystupY1* alebo *rk_vystupY2* podľa toho, o ktorú integráciu ide. Týmto sa vytvoria tabuľky a odblokuje sa aj zákaz vytvoriť graf ku každej z tabuliek.

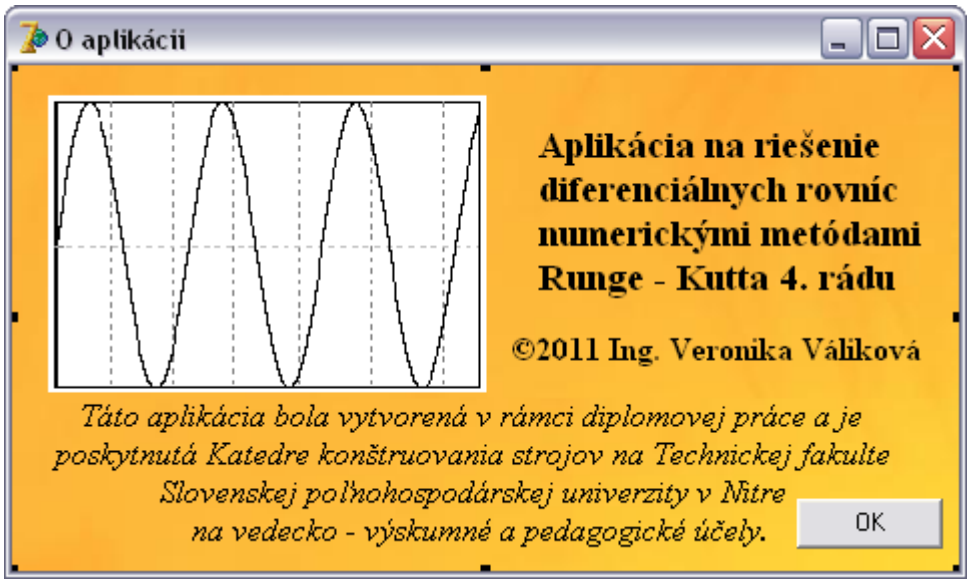
Ak používateľ zvolí vytvorenie druhej integrácie predtým, ako by sa vytvorila prvá integrácia alebo tabuľka s funkčnými hodnotami zadanej rovnice, je zabezpečené, že sa uskutočnia výpočty súvisiace s naplnením prvých dvoch tabuliek a až potom výpočet pre naplnenie tabuľky pre druhú integráciu. Ak sa stane takýto prípad, aplikácia zobrazí všetky tri príslušné tabuľky. V tomto prípade využívame premenné *naplnene, naplneneRK* a *naplneneRK2* typu *boolean,* ktoré zisťujú, či už bola vytvorená prvá, druhá alebo tretia tabuľka. Tieto premenné nadobúdajú hodnotu *true* v prípade, ak sa príslušná tabuľka vytvorí.

Ak sa jednotlivá tabuľka vytvorí a následne zavrie, hodnoty, ktoré sa v nej zobrazili zostanú naďalej uložené v príslušných dynamických poliach. Aby sa zbytočne pri jej ďalšom otvorení znova nevypočítavali hodnoty, zaviedli sme premennú *vytvorene,* na základe ktorej sa rozlišuje, či sa zavolá už len vypísanie dynamických premenných do *StringGridu* tabuľky alebo celý výpočet, ako je tomu pri prvom vytvorení tabuľky.

Aplikácia tiež blokuje vytvoriť tabuľku, ktorá už raz vytvorená bola (to znamená, že sa nezobrazí dvakrát tá istá tabuľka s rovnakými hodnotami).

4.1.3.3 Položka *„Pomoc“*

Po kliknutí na *„Info“* v tejto položke sa zobrazia informácie o aplikácii ako napríklad názov aplikácie, jej autor a rok vytvorenia. Pre účely našej aplikácie máme vytvorený obrázok *info.bmp,* ktorý pomocou vlastnosti *Picture* načítame do komponentu *Image*. Na obr. 25 je znázornený formulár s týmto komponentom.



Obr. 25 Umiestnenie komponentu SavePictureDialog a SaveDialog na formulári

Použité je aj jedno tlačidlo s názvom *„OK“*, ktoré slúži na zatvorenie tohto formuláru. Keby sme nepoužili toto tlačidlo, nebolo by ho možné vypnúť, pretože tento formulár má nastavené jednotlivé vlastnosti *BorderIcons* na *false*. Vlastnosti v rámci *BorderIcons* špecifikujú ikony, ktoré sa objavia v záhlaví formuláru (napr. minimalizačná, maximalizačná a pod.). Keďže sú vypnuté, po spustení formuláru sa nezobrazia (obr. 26).

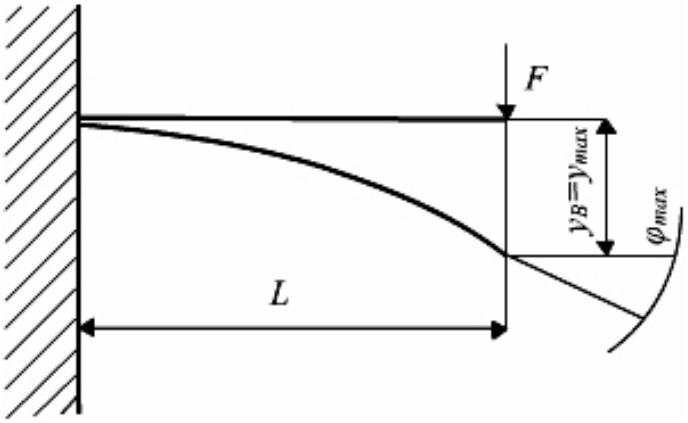
Obr. 26 Záhlavie formuláru *„Info“* bez systémových ikon

4.2 Technická aplikácia

Riešenie prvých dvoch nosníkov budeme porovnávať s presným riešením uvádzaným v literatúre (Leinveber, 2005). V ďalšom príklade budeme porovnávať výsledky aplikácie s riešením, ktoré získame pomocou programu *MathCad* od firmy *PTC*.

4.2.1 Nosník č. 1

Majme nosník zaťažený silou *F* ako to je na obr. 29. Zistite uhol sklonu dotyčnice v koncovom bode B.



Obr. 27 Nosník č.1 zaťažený silou F

Zadané hodnoty:

$$F = 10^4\,N,\ L = 1m,\ E = 2{,}1.10^{11}\,Pa,\ J_Z = 54{,}7.10^{-8}\,m^4$$

Funkcia ohybového momentu:

$$M_o = F.x \Rightarrow y'' = \frac{F.x}{EJ_Z} = \frac{10^4.x}{2{,}1.10^{11}.54{,}7.10^{-8}}$$

4.2.1.1 Presné riešenie podľa literatúry

Podľa Strojníckych tabuliek (Leinveber, 2005) je presné riešenie diferenciálnej rovnice nasledovné:

Uhol sklonu dotyčnice:

$$\varphi = \frac{FL^2}{2EJ_Z} = \frac{10^4.1^2}{2*2{,}1.10^{11}.54{,}7.10^{-8}} = \mathbf{0{,}043527466\ rad} \qquad (5)$$

4.2.1.2 Riešenie pomocou aplikácie

Vstupné údaje aplikácie:

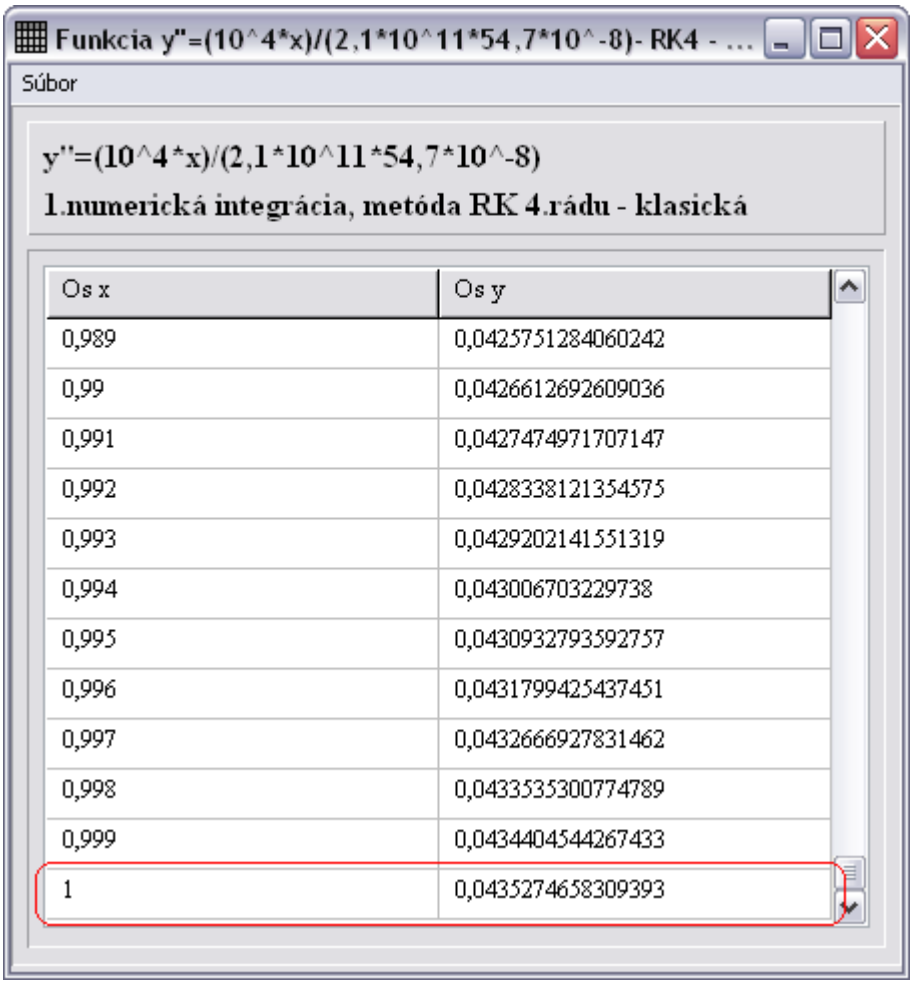
$y'' = (10\text{^}4*x)/(2,1*10\text{^}11*54,7*10\text{^}-8)$,

$L = 1, h = 0,001$.

Ďalej treba zvoliť numerickú metódu, pomocou ktorej sa uskutoční integrácia:

A: Numerická metóda Runge – Kutta štvrtého rádu - klasická

Z tabuliek, ktoré poskytuje aplikácia nás zaujíma druhá tabuľka – pre prvú numerickú integráciu. Pre jej vytvorenie zvolíme v hlavnom menu položku *Integrácia RK 4.rádu*, následne možnosť *1. integrácia* a keďže je položka *Graf* zatiaľ zablokovaná, môžeme zvoliť len položku *Tabuľka.* Vytvorí sa nám tabuľka s hodnotami prvej numerickej integrácie, pričom pre nás je zaujímavá posledná hodnota v tabuľke (kde $x=L=1$ – obr. 28), ktorá je riešením diferenciálnej rovnice. Táto hodnota reprezentuje uhol sklonu dotyčnice voči priehybovej čiare (φ).

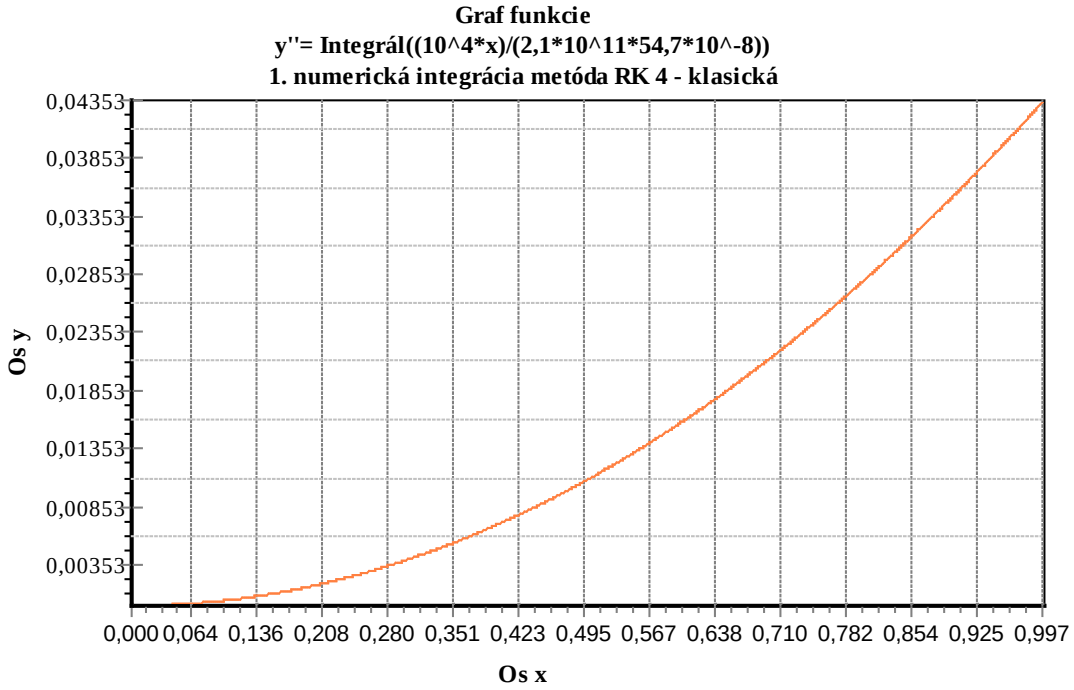


Os x	Os y
0,989	0,0425751284060242
0,99	0,0426612692609036
0,991	0,0427474971707147
0,992	0,0428338121354575
0,993	0,0429202141551319
0,994	0,043006703229738
0,995	0,0430932793592757
0,996	0,0431799425437451
0,997	0,0432666927831462
0,998	0,0433535300774789
0,999	0,0434404544267433
1	0,0435274658309393

Obr. 28 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie klasickou metódou Runge – Kutta pre nosník č. 1

Riešením diferenciálnej rovnice pre $x=L=1$ (obr. 28) je hodnota uhlu sklonu dotyčnice rovná **φ= 0,0435274658309393 rad**.

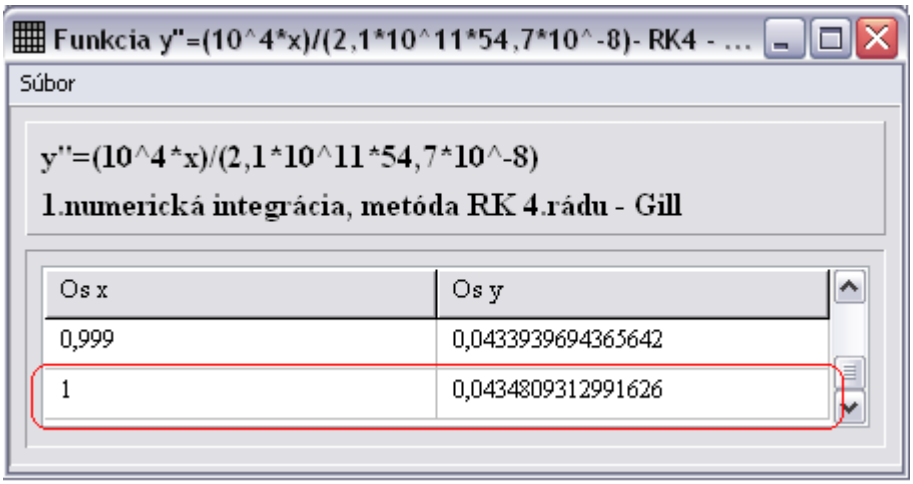
Po vytvorení tejto tabuľky aplikácia umožní vytvorenie grafu sprístupnenm dovtedy blokovanej položky „*Graf*" (obr.29).



Obr. 29 Graf hodnôt 1.numerickej integrácie klasickou metódou Runge – Kutta pre nosník č. 1

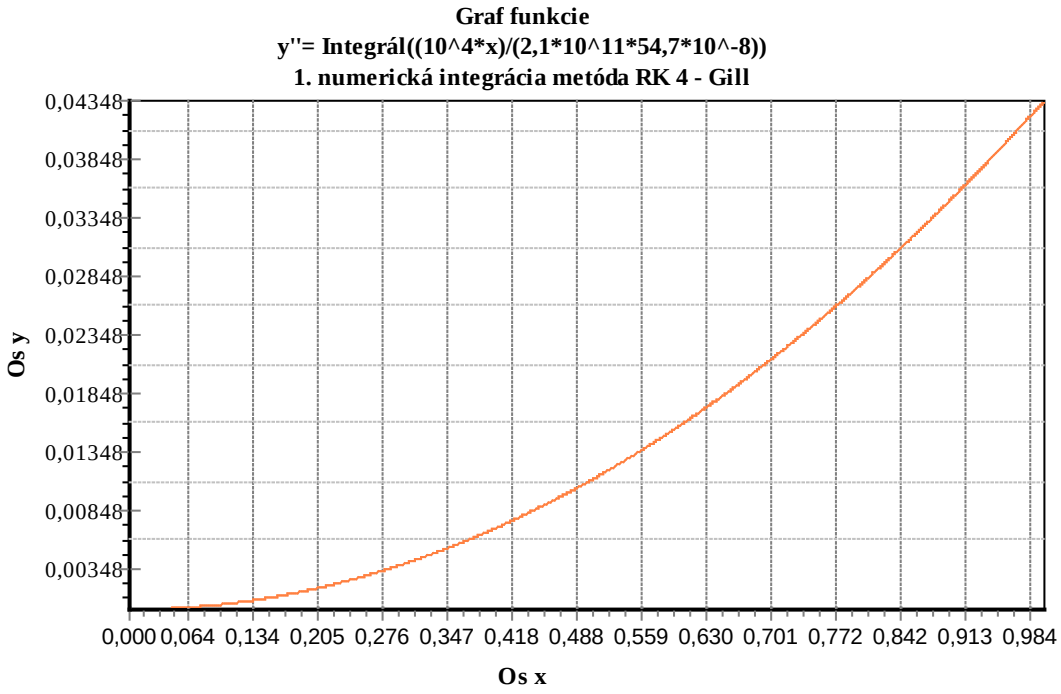
B. Numerická metóda Runge – Kutta – Gill štvrtého rádu

Riešením diferenciálnej rovnice pomocou druhej implementovanej metódy je hodnota uhlu sklonu **φ=0,043393969435642 rad** (v bode $x=L=1$ - obr. 30).



Obr. 30 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie metódou Runge – Kutta – Gill pre nosník č. 1

Príslušný graf k tabuľke, ktorý aplikácia vytvorí po uskutočnení prvej numerickej integrácie je zobrazený na obr. 31.



Obr. 31 Graf hodnôt 1.numerickej integrácie metódou Runge – Kutta – Gill pre nosník č. 1

4.2.1.3 Porovnanie riešení pre nosník č. 1

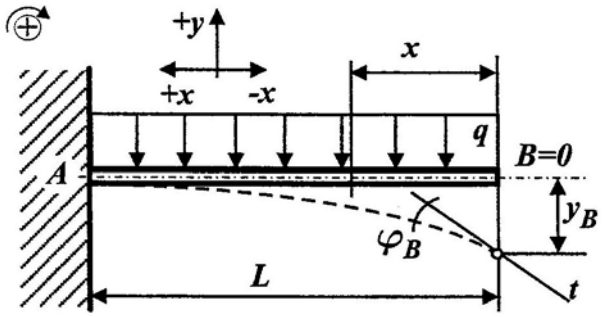
Tab. 3 Porovnanie získaných riešení

Nosník č.1		**Uhol sklonu φ [rad]**	**Rozdiel voči presnému riešeniu**	
			[rad]	**%**
Presné riešenie		0,0435274658309393		
Numerická integrácia	**RK4 klasická**	0,0435274658309393	0,000E+00	0,000E+00
	RK4 Gill	0,0434809312991626	4,653E-05	1,069E-01

Z tabuľky 3 vyplýva, že výsledok numerickej integrácie pomocou klasickej numerickej metódy Runge – Kutta štvrtého rádu je rovnaký ako výsledok presného riešenia. Integrácia pomocou numerickej metódy Runge – Kutta – Gill má hodnotu približne o 0,11 % menšiu ako presné riešenie. Z porovnania jednotlivých metód vyplýva, že klasická numerická metóda poskytuje v tomto prípade presnejšie výsledky ako druhá zvolená metóda.

4.2.2 Nosník č. 2

Majme votknutý nosník zaťažený spojitým bremenom, ako to je na obr. 32. Zistite uhol sklonu dotyčnice v koncovom bode B.



Obr. 32 Nosník č.2 zaťažený spojitým bremenom

Zadané hodnoty:

$q = 20.10^3\, Nm^{-1},\ L = 1{,}5m,\ E = 2{,}1.10^{11}\, Pa,\ J_Z = 54{,}7.10^{-8}\, m^4$

Funkcia ohybového momentu :

$$M_o = \frac{q.x^2}{2} \Rightarrow y'' = \frac{qx^2}{2EJ_Z} = \frac{20.10^3.x^2}{2.2{,}1.10^{11}.54{,}7.10^{-8}}$$

4.2.2.1 Presné riešenie podľa literatúry

Pre druh nosníku, ktorý je zobrazený na obr. 36, uvádza literatúra (Leinveber, 2005) nasledovné presné riešenie pre výpočet uhlu sklonu dotyčnice:

$$\varphi = \frac{qL^3}{6EJ_Z} = \frac{20.10^3.1{,}5^3}{6.2{,}1.10^{11}.54{,}7.10^{-8}} = \mathbf{0{,}097936798\ rad}$$

4.2.2.2 Riešenie pomocou aplikácie

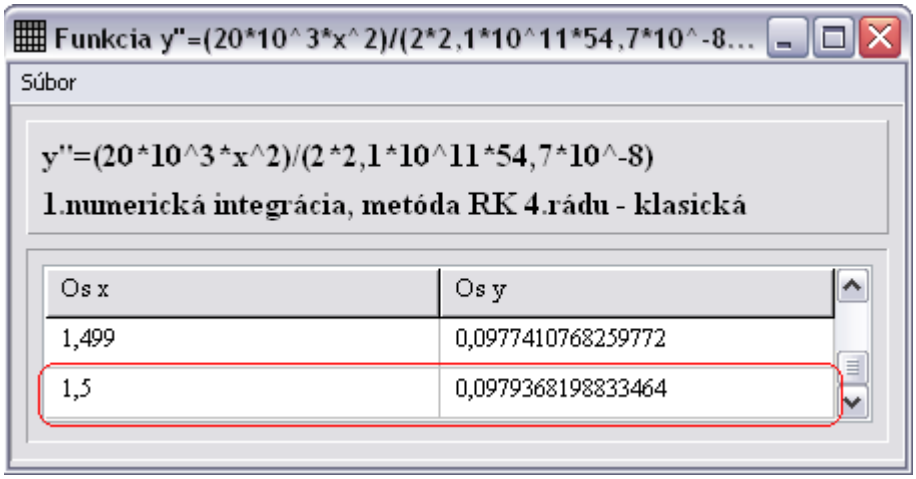
Tvar rovnice a hodnoty zadávané do programu:

y'' =(20*10^3*x^2)/(2*2,1*10^11*54,7*10^-8),

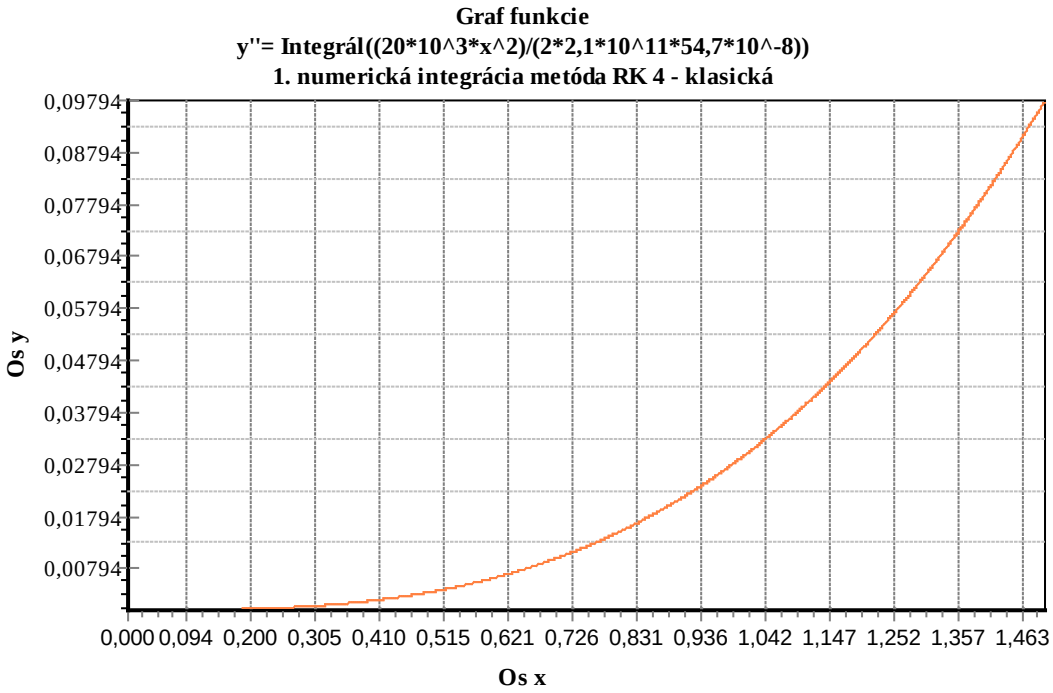
L = 1,5 , h = 0,001.

A. Numerická metóda Runge – Kutta 4. rádu - klasická

Po uskutočnení numerickej integrácie sa zobrazí príslušná tabuľka, kde hodnota pre *x=L=1,5* predstavuje riešenie zadanej diferenciálnej rovnice. Hodnota rišenia je **φ= 0,0979368198833464 rad** (obr. 33). Príslušný graf je zobrazený na obr. 34.



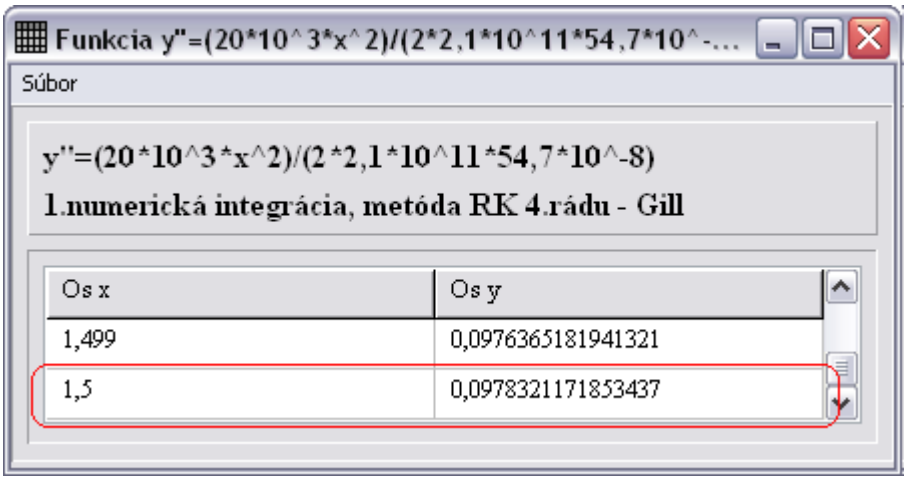
Obr. 33 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie klasickou metódou Runge – Kutta pre nosník č. 2



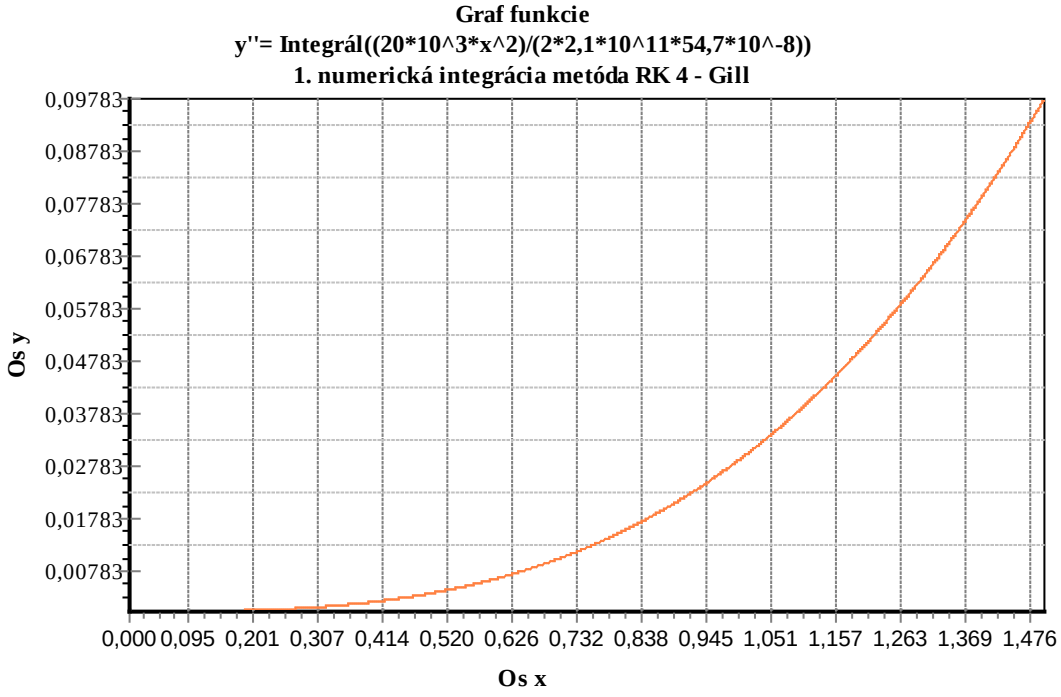
Obr. 34 Graf hodnôt 1.numerickej integrácie klasickou metódou Runge – Kutta pre nosník č. 2

B. Numerická metóda Runge – Kutta – Gill štvrtého rádu

Výsledok riešenia diferenciálnej rovnice druhou zvolenou metódou, a to Runge – Kutta –Gill, je zobrazený na obr. 35. Graf hodnôt pre túto tabuľku s hodnotami prvej numerickej integrácie je na obr. 36.



Obr. 35 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie metódou Runge – Kutta – Gill pre nosník č. 2



Obr. 36 Graf hodnôt 1.numerickej integrácie metódou Runge – Kutta – Gill pre nosník č. 2

4.2.2.3 Porovnanie riešení pre nosník č. 2

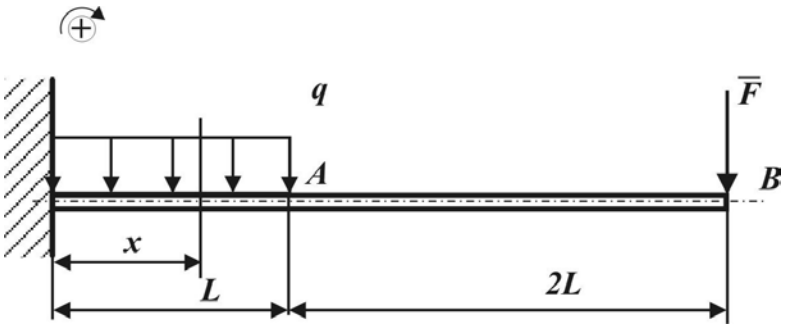
Tab. 4 Porovnanie získaných riešení

Nosník č.2		Uhol sklonu φ [rad]	Rozdiel voči presnému riešeniu	
			[rad]	%
Presné riešenie		0,0979367981196135		
Numerická integrácia	RK4 klasická	0,0979368198833464	+2,176E-08	+2,222E-05
	RK4 Gill	0,0978321171853437	-1,047E-04	-1,069E-01

Pri riešení druhého nosníka presnejšie riešenie poskytla znovu klasická metóda Runge – Kutta. Od presného riešenia je približne len o 0,00002 % vyššia, zatiaľ čo riešenie pomocou metódy Runge – Kutta – Gill je o 0,107 % nižšie (podobne ako pri prvom nosníku).

4.2.3 Nosník č. 3

Na obr. 37 je znázornený druh nosníka, ktorý je zaťažený spojitým bremenom. Zistite uhol sklonu dotyčnice v bode A .



Obr. 37 Nosník č. 3 zaťažený spojitým bremenom

Zadané hodnoty:

$$F = 0{,}6kN, L = 4m, q = \frac{F}{a} = \frac{0{,}6kN}{4m} = 150Nm^{-1},\ E = 2{,}1.10^{11}Pa,\ J_Z = 2{,}485.10^{-5}m^4,$$
$$R = 1{,}2.10^3 N, M_O = -8{,}4.10^3 Nm$$

4.2.3.1 Riešenie pomocou programu *MathCad*

Funkcia ohybového momentu:

$$M_1(x) := M_o + R \cdot x - q \cdot \frac{x^2}{2}$$

Prvá integrácia funkcie ohybového momentu:

$$\psi(x) = \int \frac{-M1(x)}{E \cdot J_z} dx$$

Rovnica pre výpočet uhlu sklonu (po uskutočnení integrácie):

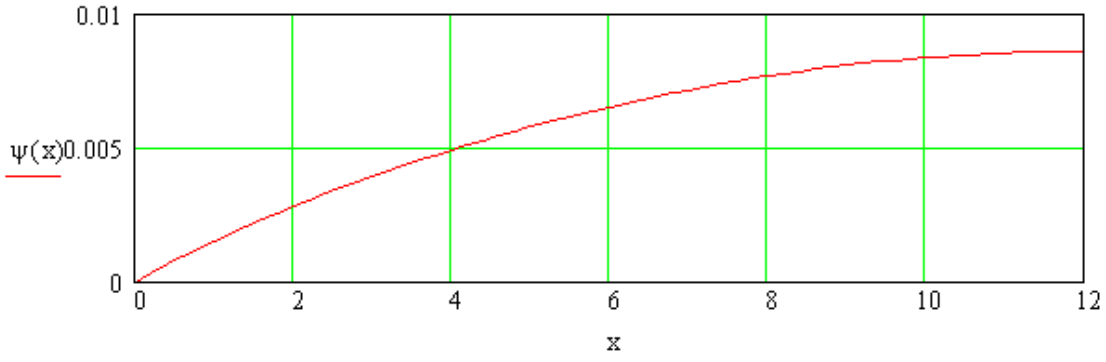
$$\psi_1(x) := \frac{1}{E \cdot J_z} \cdot \left(-M_o \cdot x - R \cdot \frac{x^2}{2} + q \cdot \frac{x^3}{6} + C_1 \right)$$

Z integrácie diferenciálnej rovnice dostávame riešenie: Ψ (L) = 0.281 deg.

Keďže výsledok, ktorý poskytuje *MathCad* je uvádzaný v stupňoch (deg) a aplikácia poskytuje výsledky v radiánoch (rad), uskutočníme prevod:

$$\phi = \frac{\psi(a).\pi}{180} = \frac{0,281.\pi}{180} = 0,0049043\ rad$$

Na obr. 38 je zobrazený graf, ktorý bol vytvorený v prostredí *MathCad*. Funkčná hodnota v bode *x=4* je približne 0,005 rad, čo zodpovedá hodnote, ktorú získame po prevedení vypočítanej hodnoty uhlu sklonu v stupňoch na radiány.



Obr. 38 Priebeh uhla sklonu dotyčnice pre nosník č. 3

4.2.3.2 Riešenie pomocou aplikácie

Funkcia ohybového momentu:

$$M_1 = M_O + R.x - q.\frac{x^2}{2} \Rightarrow y'' = \frac{-(M_O + Rx - q.\frac{x^2}{2})}{EJ_Z} = \frac{8{,}4.10^3 - 1{,}2.10^3 x + 150.\frac{x^2}{2}}{2{,}1.10^{11}.2{,}485.10^{-5}}$$

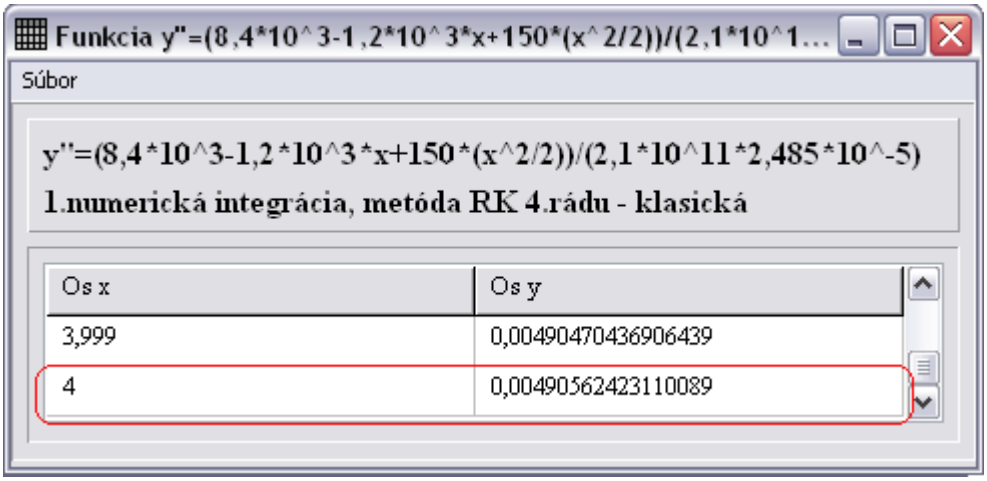
Tvar a hodnoty, ktoré sa zadávajú do aplikácie:

y'' = (8,4*10^3-1,2*10^3*x+150*(x^2/2))/(2,1*10^11*2,485*10^-5),

L=4, h=0,001

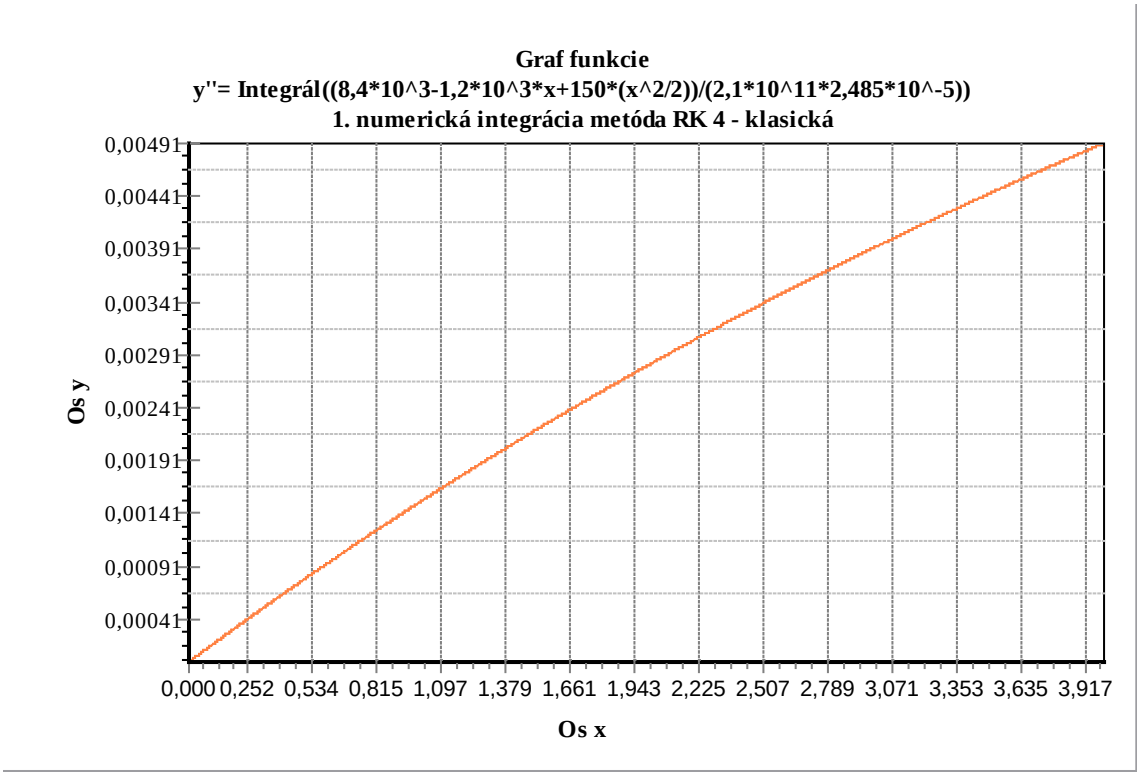
A. Numerická metóda Runge – Kutta 4. rádu - klasická

Riešením prvej numerickej integrácie zadanej diferenciálnej rovnice v bode *x=L=4* je hodnota **φ= 0,00490562423110089** (obr. 39).



Obr. 39 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie klasickou metódou Runge – Kutta pre nosník č. 3

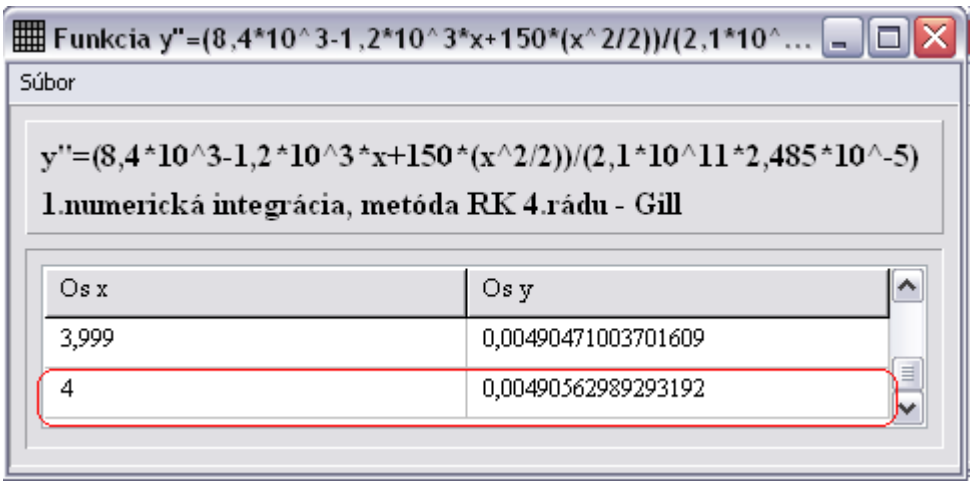
Na obr. 40 je znázornený graf zobrazujúci priebeh uhla sklonu dotyčnice φ pre nosník č. 3.



Obr. 40 Graf hodnôt 1.numerickej integrácie klasickou metódou Runge – Kutta pre nosník č. 3

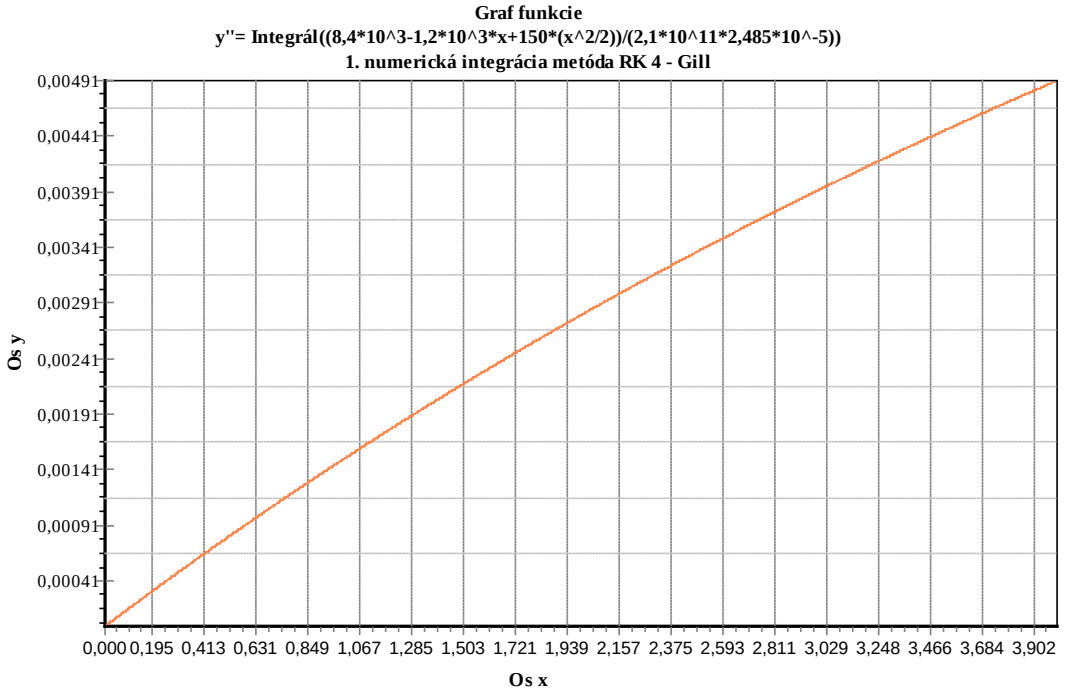
B. Numerická metóda Runge – Kutta – Gill štvrtého rádu

Po uskutočnení výpočtu pomocou metódy Runge – Kutta – Gill získame výsledok **φ= 0,00490562989293192** (obr. 41).



Obr. 41 Hodnota uhlu sklonu dotyčnice φ z 1. numerickej derivácie metódou Runge – Kutta – Gill pre nosník č. 3

Graf vytvorený z hodnôt prvej numerickej integrácie diferenciálnej rovnice na intervale $< 0,4 >$ je znázornený na obr. 42.



Obr. 42 Graf hodnôt 1.numerickej integrácie metódou Runge – Kutta – Gill pre nosník č. 3

4.2.3.3 Porovnanie riešení pre nosník č. 3

Tab. 5 Porovnanie získaných riešení pre nosník č. 3

Nosník č.3		**Uhol sklonu**		**Rozdiel oproti riešeniu v programe MathCad**	
		φ [rad]	**φ [deg]**	**[rad]**	**%**
Riešenie v programe MathCad		0,0049043751981040	0,28100		
Numerická integrácia	**RK4 klasická**	0,0049056242311008	0,28107	+1,249E-06	+2,547E-02
	RK4 Gill	0,0049056298929319	0,28107	+1,255E-06	+2,558E-02

Z výsledkov v tabuľke 5 vyplývam, že všetky výsledky sú takmer zhodné. Líšia sa len zanedbateľnými percentami. Pri oboch použitých metódach aplikácia poskytla výsledky, ktoré sú len o 0,025 % vyššie ako výsledky získané pomocou programu *MathCad*. Z hodnôt v tabuľke 5 usudzujeme, že presnejšie je riešenie pomocou klasickej numerickej metódy Runge – Kutta.

5 Návrh na využitie výsledkov

Výsledkom predkladanej diplomovej práce je aplikácia na riešenie diferenciálnych rovníc numerickými metódami Runge – Kutta štvrtého rádu. Pri riešení sa využíva zníženie stupňa derivácie v diferenciálnej rovnici pomocou numerickej integrácie.

Hlavný prínos tejto aplikácie spočíva v jej využití v pedagogickom procese pri výučbe predmetov Pružnosť a pevnosť a súvisiacich disciplín. Takisto je možné vytvorený program aplikovať aj na výpočty v praxi. Pomocou aplikácie sa dá uskutočniť numerická integrácia funkcie ohybového momentu pre jednotlivé myslené rezy. Hodnota v poslednom riadku tabuľky predstavuje riešenie zadanej diferenciálnej rovnice. Táto hodnota bude reprezentovať hodnotu uhlu sklonu dotyčnice voči priehybovej čiare zaťaženého nosníka. Ak by sa stanovili okrajové podmienky, bolo by možné vypočítať integračné konštanty. Po sčítaní výsledku druhej integrácie pre bod $x=L$ s integračnými konštantami, by takto získaná hodnota predstavovala veľkosť priehybu zaťaženého nosníka.

Na základe uvedených príkladoch v kapitole 4.2 môžeme povedať, že výsledky získavané aplikáciou sú na rovnakej úrovni ako pri klasickom integrovaní, čo umožňuje využitie aplikácie na zjednodušenie potrebných výpočtov.

Záver

Diplomová práca sa zaoberá numerickými metódami Runge – Kutta a ich použitím pri riešení diferenciálnych rovníc. Práca poskytuje prehľad o súčasnom stave riešenej problematiky, opis jednotlivých rovníc pre metódy Runge – Kutta rôznych rádov a taktiež oboznámenie s vývojovým prostredím, v ktorom sme naprogramovali aplikáciu na riešenie diferenciálnych rovníc. Táto aplikácia implementuje dve metódy Runge – Kutta štvrtého rádu, a to klasickú metódu a metódu Runge – Kutta – Gill.

Aplikácia slúži na vytvorenie prvej a druhej integrácie zadanej diferenciálnej rovnice. Pomocou aplikácie sa dá zistiť uhol sklonu dotyčnice voči priehybovej čiare zaťaženého nosníka, pričom vychádzame z ohybového momentu pre jednotlivé myslené rezy. Podľa zvolenej metódy sa uskutoční integrácia funkčných hodnôt bodov z intervalu $< 0, L >$, ktorý je rozdelený podľa kroku *h*. Po prvej integrácii hodnota poslednej položky v tabuľke reprezentuje spomenutý uhol sklonu φ.

Na dvoch zaťažených nosníkoch sme porovnávali riešenie poskytnuté oboma naprogramovanými metódami s presným riešením. Taktiež sme uviedli riešenie pre nosník v programe *MathCad* a takto získané výsledky sme porovnali s výsledkami naprogramovanej aplikácie. Pri aplikačnom riešení zvolených druhov nosníkov sme zistili, že riešenia, ktoré poskytuje aplikácia boli vo všetkých prípadoch takmer zhodné s presným riešením alebo s riešením prostredníctvom programu *MathCad*.

Riešenie klasickou numerickou metódou Runge – Kutta bolo v prípade prvého nosníka úplne rovnaké ako presné riešenie, zatiaľ čo hodnota výsledku riešenia pomocou druhej metódy, Runge – Kutta – Gill, bola o 0,11 % menšia ako hodnota presného riešenia.

Pri nosníku číslo dva bola hodnota získaná numerickou integráciou pomocou klasickej metódy len o 0,00002 % vyššia ako pri presnom riešení. Táto metóda sa ukázala opäť presnejšia ako metóda Runge – Kutta – Gill, kde rozdiel voči presnému riešeniu bol 0,000104 rad, čo predstavuje hodnotu o 0,107 % nižšiu ako pri presnom riešení.

Pri riešení tretieho nosníka oboma implementovanými metódami sa hodnota líšila od riešenia v programe *MathCad* približne o 0,025 %. Ak však porovnáme percentuálne rozdiely v tabuľke 5, aj v tomto prípade je výsledok riešenia klasickou

numerickou metódou presnejšie voči riešeniu v programe *MathCad* ako metóda Runge – Kutta – Gill.

Z uskutočnených riešení, ktoré boli zamerané na výpočet uhlu sklonu dotyčnice φ vyplýva, že vo všetkých troch prípadoch sa klasická numerická metóda Runge – Kutta ukázala presnejšia ako metóda Runge – Kutta – Gill, aj keď rozdiely boli minimálne.

Zoznam použitej literatúry

ANASTASSI, Z.A. – SIMOS, T.E. – VLACHOS, D.S. 2009. A family of Runge-Kutta methods with zero phase-lag and derivatives for the numerical solution of the Schrödinger equation and related problem. In *Journal of Mathematical Chemistry* [online], roč. 46, 2009, č. 4, s. 1158 – 1171 [cit. 2010-11-06]. Dostupné na <http://www.springerlink.com/content/592024n118075777/ >.ISSN: 1572-8897.

ANISZEWSKA, D. 2007. Multiplicative Runge – Kutta methods. In *Nonlinear Dynamics* [online], roč. 50, 2007, č. 1, s. 265 – 272 [cit. 2010-12-05]. Dostupné na <http://www.springerlink.com/content/tj7t5g1623480442/ >. ISSN: 1573-269X.

BOŽEK, P. – KUNÍK, S. 2006. Virtuálna realita vo vyučovaní a riadiacích procesoch. In *Konferencia: Medzinárodné vedecké dni 2006 „Konkurencieschopnosť v EÚ – výzva pre krajiny V4“* [online]. Nitra: SPU, 2006, s. 1495 – 1502 [cit. 2010-09-28]. Dostupné na <http://www.fem.uniag.sk/mvd2006/zbornik/sekcia8/s8_bozek_pavol_388.pdf>

BRUDER, J. 1996. Numerical results for a parallel linearly – implicit Runge – Kutta Method. In *Computing* [online], roč. 59, 1996, č.1 [cit. 2010-12-05]. Dostupné na <http://www.springerlink.com/content/w0310263130g7466/ > ISSSN: 1436-5057.

ČERNÁ, R. – MACHALICKÝ, M. – VOGEL, J. a i. 1987. *Základy numerické matematiky a programování*. Praha: SNTL, 1987. 448 s.

DÁVID, A. 1988. *Numerické metódy na osobnom počítači.* Bratislava: Alfa, 1988. 184 s.

HAJNALA, J. 2005. *Delphi 7 - Základy objektového programovacieho jazyka* [online]. Nitra: Gymnázium Párovská 1, 2005 [cit. 2011-02-28]. Dostupné na < http://www.gymparnr.edu.sk/obsah/predmety/subory/informatika/prostrdelphi.pdf>.

Holistic numerical methods institute [online]. Florida: University of South Florida [cit. 2010-10-05]. Dostupné na <http://numericalmethods.eng.usf.edu>

KADLEC, V. 2007. *Umíme to s Delphi* [online počítačový program]. Ver. 1.1.12.574. 2007 [cit. 2010-12-05] Požiadavky na systém: Windows 2000 a vyšší, Internet Explorer ver. 6 a vyššia. Dostupné na <http://umime-to-s-delphi.wz.cz/download.php>

KUČEROVÁ, I. 2003. *Diferenciálne rovnice v praktických príkladoch*: Diplomová práca. Bratislava: UK, 2003. 49 s.

LEINVEBER, J. – VÁVRA, P. 2005. *Strojnícke tabuľky*. 2.vyd. Úvaly: ALBRA, 2005. 905 s. ISBN 80-7361-011-6.

NEKVINDA, M. – ŠRUBAŘ, J. – VILD, J. 1976. *Úvod do numerickej matematiky*. Praha: SNTL, 1976. 288 s.

Oberwolfach Photo Collection. Renate Schaaf. 1981 [online]. Oberwolfach: MFO 1981 [cit. 2011-02-21]. Dostupné na: <http://owpdb.mfo.de/detail?photo_id=3643>

Parsers, mathematical expression evaluators, calculators. 1998 [online]. B.m.: b.v, 1998 [cit. 2011-02-21]. Dostupné na: <http://www.efg2.com/Lab/Library/Delphi/MathFunctions/Parsers.htm>

POHUBA, S. 2009. *Využitie komponentu Express v programovaní v Delphi*: bakalárska práca. Nitra: UKF, 2009. 38 s.

PŘIKRYL, P.1988. *Numerické metody matematické analýzy*. 2.vyd. Praha: SNTL, 1988. 192 s.

RALSTON, A. 1978. *Základy numerické matematiky*. 2.vyd. Praha: Academia, 1978. 636 s.

RÉDL, J. 2009. *Návody na riešenie úloh pružnosti a pevnosti* [online]. Nitra: SPU, 2009 [cit. 2011-02-21]. Dostupné na < http://www.tf.uniag.sk/e_sources/katmech/PP/Navody_PaP_2007.pdf>

RÉDL, J. 2010a. *Pružnosť a pevnosť: Návody na cvičenia*. Nitra: SPU, 2010.175 s. ISBN 978-80-552-0400-0.

RÉDL, J. 2010b. *Vývoj technických aplikácií pre vedu a výskum*. Nitra: SPU, 2010. 146 s. ISBN 978-80-552-0395-9.

Riešené úlohy z matematiky II. 2000 [online]. Bratislava: STU, 2000. [cit. 2010-09-29]. Dostupné na <http://www.math.sk/skripta2/>.

SKALKA, J. – CÁPAY, M. – LOVÁSZOVÁ, G. a i. 2007. *Algoritmizácia a úvod do programovania.* Nitra: UKF, 2007. 158 s. ISBN 978-80-8094-217-5

SKVORTSOV, L.M. 2010. Diagonally implicit Runge—Kutta methods for differential algebraic equations of indices two and three. In *Computational Mathematics and Numerical Analysis* [online], roč. 50, 2010, č. 6, s. 993 - 1005 [cit. 2010-12-05].

Dostupné na <http://www.springerlink.com/content/e21p33x5w1kk4427/ > ISSN 0965_5425.

ŠINDELÁŘ, J. 2007. *Tipy a triky v Delphi* [online počítačový program]. Ver. 1.1.13.574. 2007 [cit. 2011-04-03]. Požiadavky na systém: Windows 2000 a vyšší, Internet Explorer ver. 6 a vyššia. Dostupné na <http://tipy-a-triky-v-delphi.wz.cz/download.php>

VELICHOVÁ, D. 2006. *Matematika II.* [online]. Bratislava: STU, 2006 [cit. 2010-09-28]. Dostupné na <http://evlm.stuba.sk/~velichova/Matematika2/Kniha/kniha.html>

VITÁSEK, E. 1987. *Numerické metody*. Praha: SNTL, 1987. 516 s.

Prílohy

Príloha 1 – Elektronický nosič CD

Priložené CD obsahuje**:**

- Diplomová práca vo formáte PDF.
- Aplikácia v *Delphi* .