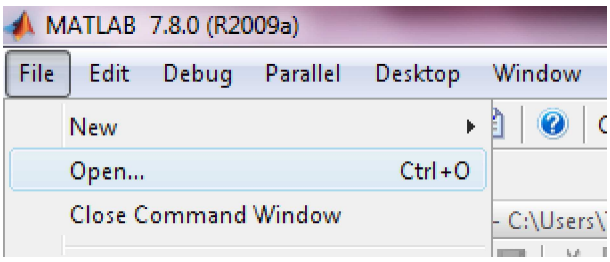
2 Riešenie úloh v simulačnom jazyku MATLAB s využitím skriptov a funkcií

2.1 Práca s m-súborom v programovom prostredí MATLAB

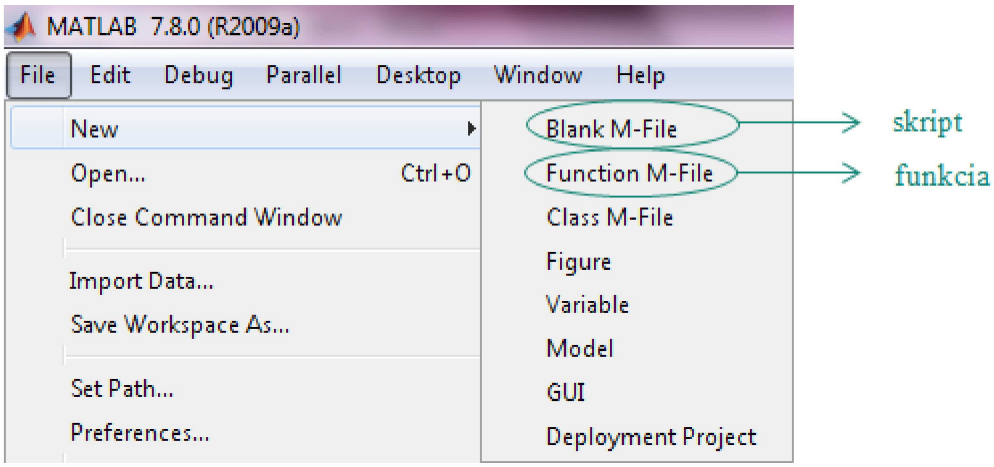
- **M – file / M – súbor** je textový súbor s príponou „ .m “ , ktorý slúži na ukladanie postupnosti príkazov – ***skripty***, alebo na ukladanie užívateľských funkcií – ***funkcie***. Využívajú sa predovšetkým vtedy, keď je potrebné zadať väčšie množstvo príkazov, čo by v príkazovom okne bolo neprehľadné, zdĺhavé a zložité meniť. Programové prostredie MATLAB obsahuje vlastný textový editor M – súborov, avšak môže sa použiť aj iný textový editor. Takto je k nim zabezpečený priamy prístup a môžu sa kedykoľvek upravovať nezávisle od hlavného programu.

Editor prostredia MATLAB sa spustí v novom okne po otvorení ***M- súboru***



Obrázok 2-1 Spustenie Editora po otvorení M - súboru

alebo po vytvorení nového ***M- súboru***



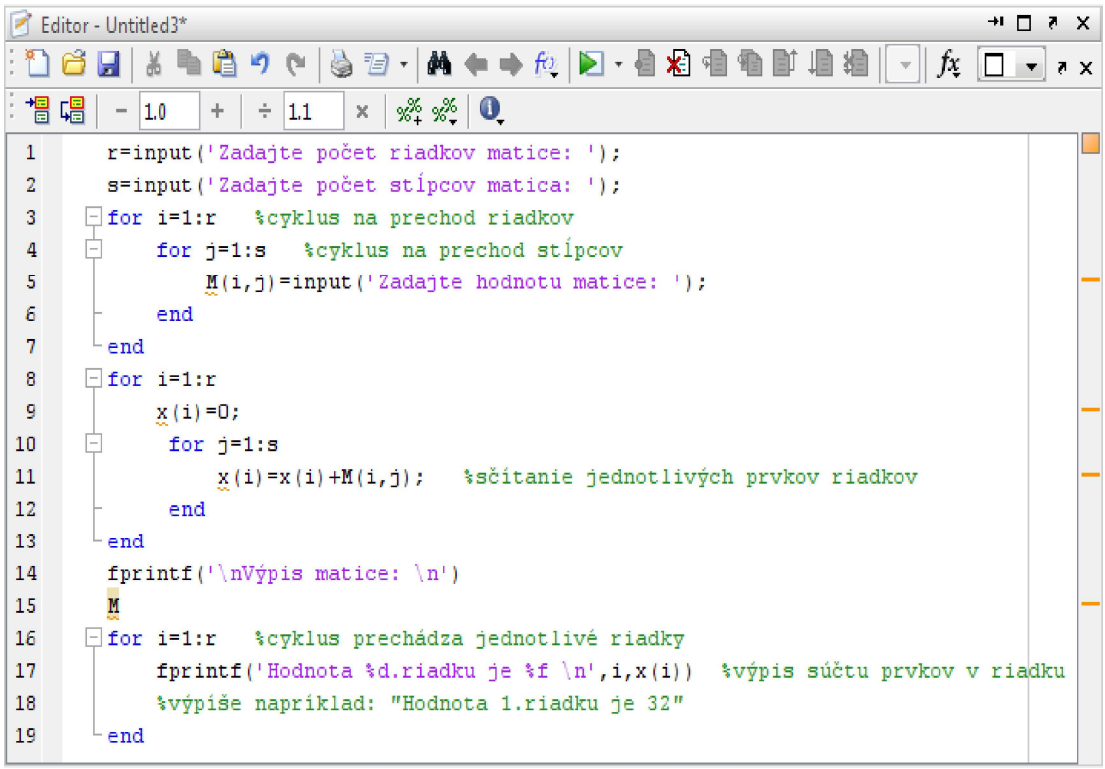
Obrázok 2-2 Spustenie Editora po vytvorení nového súboru funkcie

Ako už bolo spomenuté, existujú dva ***typy M- súborov*** a to : ***súbor so skriptom*** a ***súbor s funkciou***, v nasledujúcej tabuľke je ich porovnanie.

Skript	**Funkcia**
Nezadávajú sa vstupné argumenty a ani sa nevypisujú výstupné	Môže obsahovať vstupné a vracať výstupné premenné
Pracuje s premennými vo Workspace	Vnútorné premenné sú lokálne pre funkciu, Workspace „nevidí“
Vhodné pre automatizáciu volania tých istých príkazov viac krát	Vhodné pre rozšírenie funkčnej základne matlabovských vstavaných funkcií

Príklad 1

Vytvorte postupnosť príkazov - skript, v ktorom vyzvete užívateľa na zadanie rozmerov matice a jej následné naplnenie. Vypíšte súčet riadkov matice.



2.2 Tvorba vlastných funkcií v jazyku MATLAB

V prostredí MATLAB existujú *vstavané funkcie* (bult-in functions), ako napr. ***sin***, ***sqrt*** či ***abs***, ktoré bežne používame pri matematických výpočtoch. Ich názov hovorí na čo slúžia, ale užívateľ nemusí vedieť nič o výpočte vykonávajúcom sa vo vnútri funkcie. Užívateľ jednoducho zadá *vstupný parameter* a následne mu je vrátený *výstupný parameter*.
Výhodou využívania funkcií je „skrytie" algoritmu vypočítavajúceho danú funkciu pod názov funkcie. Nie však všetky funkcie, ktoré používateľ potrebuje, sú vopred naprogramované, preto existujú ***používateľské funkcie***.

- **Funkcie** sú M-súbory s presne definovanou štruktúrou, ktoré obsahujú postupnosti príkazov a zároveň vstupné a výstupné premenné.

```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


end
```

Charakteristické znaky funkcie :

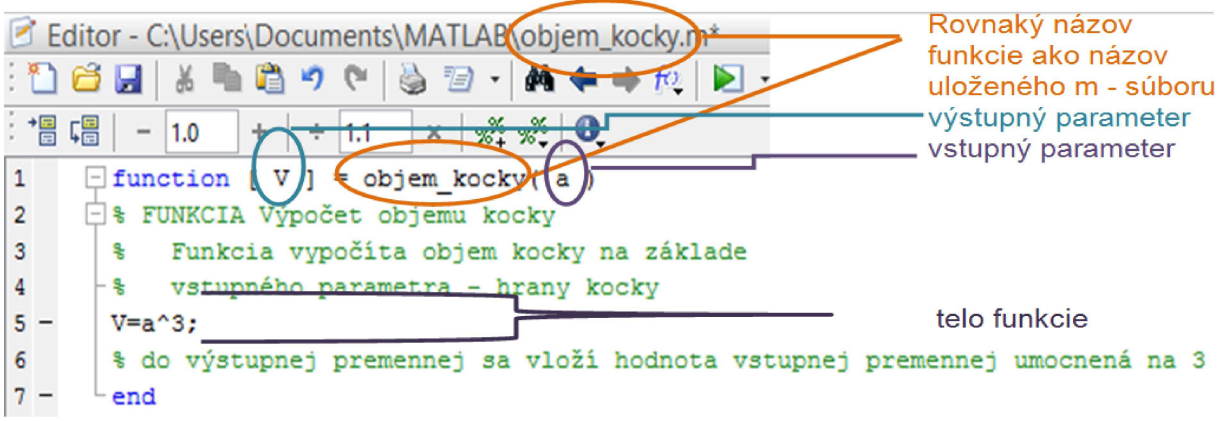
1. definícia funkcie – názov funkcie, vstupné a výstupné parametre
2. prvý riadok nápovedy – stručný popis funkcie, je prehľadávaný funkciou ***lookfor***
3. nápoveda – popis funkcie
4. vlastný kód

Poznámka :

⇨ nápoveda (body 2, 3) je nepovinná časť,
⇨ názov funkcie sa musí zhodovať s názvom súboru bez prípony - „.m"

Vstupné parametre môžu byť pri každom spustení funkcie iné. Výhodou funkcií je použitie jedného postupu - algoritmu pre rôzne hodnoty vstupných parametrov. V takomto prípade nie je vhodné použitie skriptov, keďže by sa hodnoty museli stále prepisovať.
Všetky premenné vo funkcii sú ***lokálne***, to znamená, že existujú len počas spustenia funkcie. Vo funkcii nie je možné použiť iné premenné ako tie, ktoré vložíme do vstupných parametrov, alebo tie, ktoré vo funkcii vytvoríme. Taktiež platí, že všetky premenné po skončení funkcie zaniknú. Vstupné parametre sú zadavané hodnotou, takže ak zmeníme ich hodnotu v tele funkcie nezmení sa tým hodnota premennej, s ktorou bola funkcia volaná.

Príklad 2
Vytvorte funkciu, ktorej vstupným parametrom bude strana kocky a výstupným objem kocky. Do tela funkcie napíšte kód, ktorým zrealizujete výpočet objemu kocky.



Obrázok 2-3 Zápis funkcie v Editore pre „výpočet objemu kocky"

V príkazovom riadku si pri spustení funkcie s názvom ***objem_kocky*** do premennej *a* vložíme hodnotu hrany kocky 12 a pomocou vytvorenej ***funkcie objem_kocky*** bude vypočitaný jej objem ***V***, ktorý uložíme do premennej ***X***.

```
Command Window
>> X=objem_kocky(12)

X =

        1728

fx >>
```

Obrázok 2-4 Spustenie funkcie objem_kocky pre definovaný parameter strany kocky *a* = 12

Príklad 3

Vytvorte funkciu, ktorej výstupná premenná bude obsahovať vypočitaný n – faktoriál čísla pričom vstupný parameter bude číslo, ktorého faktoriál sa počíta.

Editor - C:\Users\ \Documents\MATLAB\faktorial.m

```
1  function [ nfaktorial ] = faktorial( x )
2  %FUNKCIA výpočet n - faktoriálu
3  %   vypočíta faktoriál vstupného argumentu x
4  nfaktorial=1;
5  for i=1:x
6      nfaktorial=nfaktorial*i;
7  end
8
9  end
```

Obrázok 2-5 Výpočet *n* – faktoriálu pre zadané číslo – riešené pomocou *funkcie*

V príkazovom riadku si do premennej *N* vložíme vypočitanú hodnotu faktoriálu čísla 5 (5!), ktorý bol vypočitaný pomocou vytvorenej funkcie ***faktorial*** pre zvolený parameter funkcie *x* = 5.

```
Command Window
>> N=faktorial(5)

N =

   120
```

Obrázok 2-6 Spustenie funkcie *faktorial(5)*

Príklad 4

Vytvorte funkciu na výpočet obsahu trojuholníka. V hlavnom programe vyzvite užívateľa na zadanie strany a výšky pre vytvorenú funkciu.

```
% HLAVNÝ PROGRAM
b=input('Zadajte stranu trojuholníka: ');
vb=input('Zadajte výšku na zadanú stranu: ');
St=obsah_trojuholnika(b,vb);
fprintf('Obsah trojuholníka je : %f',St);
```

```
% FUNKCIA
function [ S ] = obsah_trojuholnika( a,va )
S=a*va/2;
end
```

Príklad 5
Vytvorte funkciu, ktorá vypočíta hodnotu zlomku

$$\frac{(a^2-3a-2)}{(a-1)^2(a+2)}.$$

Túto funkciu použite v hlavnom programe, pomocou ktorého budete môcť vypočítať hodnotu funkcie pre premennú ***a*** zadanu z klávesnice. Skontrolujte či pre danú hodnotu ***a*** má výraz riešenie. Ak nie výsledok bude 0. Výstup zobrazte.

```
function [ H ] = zlomok( a )
if (a==1 || a==-2)
    H=0;
else
    H=(a^2-3*a-2)/((a-1)^2*(a+2));
end
end
```

Príklad 6
Vytvorte funkciu, ktorá bude využívať Pytagorovu vetu pre výpočet chýbajúcej strany trojuholníka. V hlavnom programe (skripte) ponúknite možnosť výberu chýbajúcej strany - prepona/odvesna a zohľadnite to aj vo funkcii. Dĺžky zvyšných dvoch strán zadá užívateľ na vstupe.

Kód v jazyku MATLAB

```
function [ x ] = pytagoras( o,a,b )
switch o
    case {1}
        x=sqrt(a*a+b*b);
    case {2}
        if a>b
            x=sqrt(a*a-b*b);
        else x=sqrt(b*b-a*a);
        end
end
end
```

Kód hlavného programu :

```
o=input('Pre výpočet prepony zadaj "1" \nPre výpočet odvesny zadaj "2"\n');
while (o>2 | o<1)
o=input('Pre výpočet prepony zadaj "1" \nPre výpočet odvesny zadaj "2"\n');
end
a=input('Zadajte hodnotu prvej strany: ');
b=input('Zadajte hodnotu druhej strany: ');
c=pytagoras(o,a,b)
```

Príklad 7
Vytvorte funkciu, ktorá bude vracať súčet vektorov, súčin vektorov po prvkoch a súčet absolútnych hodnôt jednotlivých prvkov vektorov. Vektory budú vstupnými argumentami funkcie.

Kód v jazyku MATLAB

```
function [ sucin, sucet, SAH ] = vektory( v1,v2 )
% sucin - výstupný vektor, ktorý predstavuje súčin vstupných vektorov
% sucet - výstupný vektor, ktorý predstavuje súčet vstupných vektorov
% SAH - výstupná hodnota - súčet absolútnych hodnôt jednotlivých prvkov vstupných vektorov
if length(v1)~=length(v2)
    error('vektory nemajú rovnakú dĺžku')
else
    sucet=v1+v2;
    SAH=0;
    for i=1:length(v1)
        SAH=SAH+abs(v1(i))+abs(v2(i));
        sucin(i)=v1(i)*v2(i);
    end
end
end
```

Príklad 8
Vytvorte funkciu ktorej vstupnými parametrami budú dve ľubovoľné matice. Funkcia porovná ich rozmery, v prípade nerovnosti rozmerov vypíše chybu, v opačnom prípade bude hodnotiť prvky matíc na rovnakých pozíciách na základe ich párnosti. Výsledky zapíše do výslednej matice, ktorá bude výstupným argumentom, a to tak, že do nej zapíše počet párnych prvkov na danej pozícii(0,1 alebo 2)

Kód v jazyku MATLAB

```
function [ V ] = matice( A,B )
if size(A)~=size(B)
    error('Zadané matice nemajú rovnaké rozmery')
else
    a=size(A);
    for i=1:a(1)    %cyklus na prehliadanie riadkov matíc
        for j=1:a(2)    %cyklus na prehliadanie stĺpcov matíc
            if rem(A(i,j),2)==0 %overenie párnosti prvku matice A na pozícii (i,j)
                if rem(B(i,j),2)==0 %overenie párnosti prvku matice B na pozícii (i,j)
                    V(i,j)=2;   %prípad, že obidve podmienky sú vyhodnotené ako pravdivé
                else
                    V(i,j)=1;   %prvá podmienka platí a druhá nie
                end
            else
                if rem(B(i,j),2)==0 %overenie párnosti prvku matice B na pozícii (i,j)
                                    %v prípade,že prvok matice A na danej pozícii nie je párny
                    V(i,j)=1;   %prvok matice A nepárny a prvok matice B párny
                else
                    V(i,j)=0;   %prípad, že obidve podmienky sú vyhodnotené ako nepravdivé
                end
            end
        end
    end

end
```

2.3 Príklady na riešenie

1. Vytvorte funkciu, ktorej výstupný parameter bude štvrtá mocnina vstupného parametra.

2. Vytvorte funkciu, ktorá vypočíta obsah trojuholníka so vstupnými premennými základňou a výškou.

3. Vytvorte funkciu, ktorá vráti skalárny súčin zadaných vektorov.

4. Máme funkciu výnosov *f(b)*=895b+64-2^b. Vytvorte funkciu v Matlabe, ktorá vytvorí vektor hodnôt funkcie výnosov na základe vstupného vektora a vyhľadá bod, v ktorom sú výnosy najvyššie. Použite funkciu v hlavnom programe, v ktorom si vyžiadate ručný vstup hodnôt, ktoré budú vstupnými hodnotami pre funkciu výnosov. Na vstup dajte kontrolu, či ste zadali vektor. Ak nie vytvorte cyklus, aby sa funkcia vykonala až po správnom zadaní hodnôt. Vo výsledku vypíšte hodnoty funkcie a aj najvyššiu hodnotu a bod.

5. Vytvorte funkciu, ktorá vynásobí 2 vektory medzi sebou po prvkoch a zo súčinu vypíše všetky čísla deliteľné 3 alebo 4. Vektory zadajte z klávesnice a výsledky vypíšte.

6. Vytvorte funkciu, ktorej vstupné parametre budú dva vektory rovnakej dĺžky. Funkcia má umocniť každý *i*-ty prvok 1.vektora *i*-tym prvkom 2.vektora.

7. Vstupnými parametrami funkcie nech sú dve matice. Vytvorte funkciu na porovnanie týchto matíc. Funkcia zistí či sú matice rovnakých rozmerov. V prípade že nie sú výsledkom bude 0. V opačnom prípade bude funkcia postupne porovnávať prvky matíc na rovnakých pozíciách. Výsledky porovnania zapíše do výslednej matice nasledovne. Ak je prvok prvej matice väčší ako prvok druhej matice do výslednej matice sa zapíše 1, ak je prvok prvej matice menší ako prvok druhej matice do výslednej matice sa zapíše 2. V prípade, že sa prvky rovnajú zapíše sa 0.

2.4 Postup pre vytvorenie m-súboru, skriptu a funkcie v Editore programového prostredia MATLAB

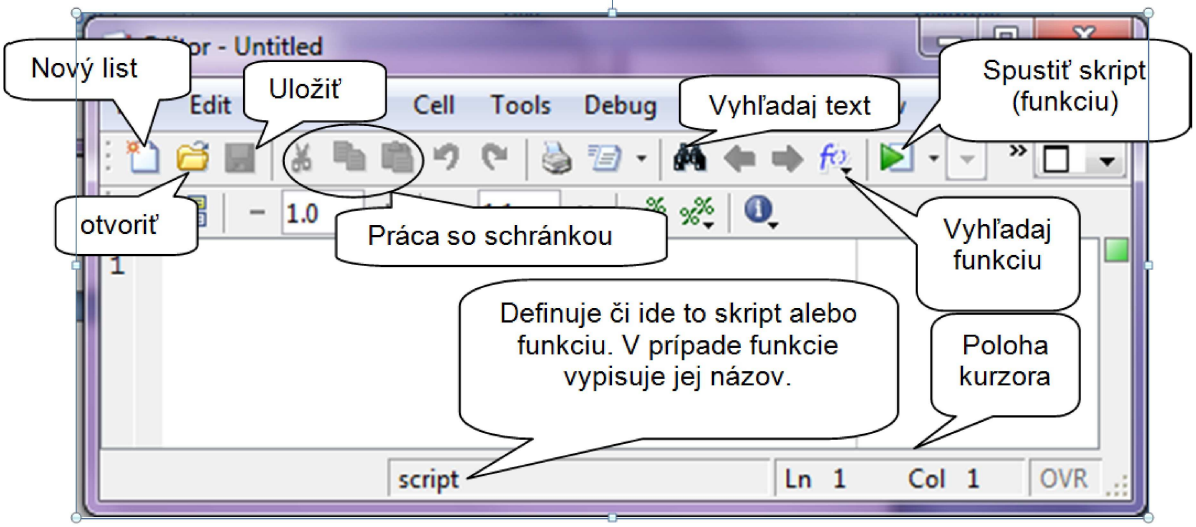
- **M-súbory**

⇨ ***m-súbory*** sú zapisované do editora, ktorý je súčasťou programového prostredia MATLAB

⇨ otvorenie nového m-súboru v prostredí MATLAB je možné po kliknutí na ikonu 

⇨ ***m-súbory*** môžu byť ***skripty*** alebo ***funkcie***

- **Popis Editora**



Obrázok 2-7 Editor programového prostredia MATLAB a práca v ňom

⇨ zvýrazňuje kľúčové slová simulačného jazyka MATLAB

⇨ umožňuje krokovať obsah *m-súborov*

- **Skripty**

⇨ skript je po obsahovej stránke postupnosť príkazov simulačného jazyka MATLAB zapísaných do súboru pod určitým menom

⇨ skripty neprijímajú vstupné a nevracajú výstupné argumenty, pracujú s dátami uloženými v pamäťovom okne *Workspace* (okno na vizualizáciu obsahu používaných premenných)

⇨ v skripte použité funkcie pracujú s údajmi v základnom pracovnom priestore

⇨ súbory sú ukladané s jedinečným **menom** a príponou **.m** ***(meno_suboru.m)***

⇨ premenné, ktoré sú pred použitím skriptu definované, môžeme v skripte použiť

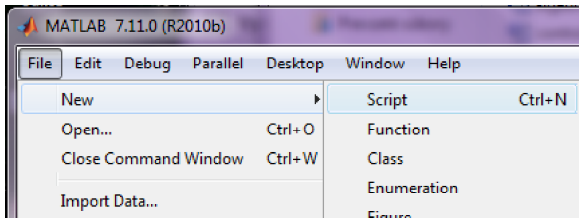
⇨ premenné, ktoré sú vytvorené počas vykonávania skriptu, zostanú po ukončení skriptu zachované

- **Tvorba skriptu**

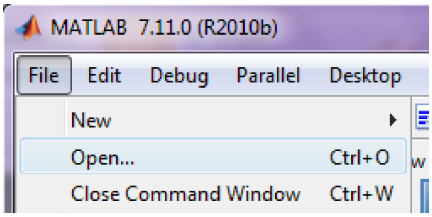
a. Najskôr si musíme v pracovnom priestore MATLAB-u nastaviť cestu k pracovnému adresáru

Otvorenie nového skriptu prebieha v Editore ikonou 

alebo



ak meníme existujúci skript, použijeme  alebo



b. do prázdneho listu editora sa napíše postupnosť príkazov, ktoré sa v tomto prípade nevykonávajú hneď po napísaní a stlačení klávesy ENTER

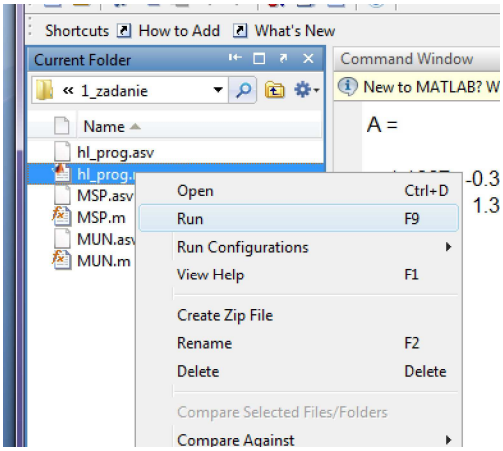
c. napísanú postupnosť príkazov je potrebné uložiť pod nejakým menom na disk, pre uloženie môžeme použiť ikonu  alebo *File/Save As...*

Ak sa pokúsime spustiť neuložený skript, vyskočí nám rovno okno pre uloženie, t.j. neuložený skript nevieme spustiť

d. Volanie/spustenie skriptu sa vykonáva prostredníctvom zápisu mena skriptu v príkazovom okne (Command Window),

V tomto prípade sa musí skript nachádzať v adresári, v ktorom sa aktuálne nachádzame alebo tam musi byť nastavená cesta .

Spustenie skriptu sa môže vykonať kliknutím na ikonu  v Editore, alebo kliknutím na skript, ktorý chceme otvoriť pravým tlačidlom myši a vybrať možnosť *Run*



Obrázok 2-8 Spustenie vytvoreného skriptu

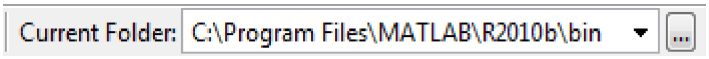
- **Funkcie**

⇨ funkcie sú najefektívnejším nástrojom pre modulárne programovanie a automatizáciu úlohy
⇨ prijímajú vstupné a vracajú výstupné argumenty

⇨ premenné novo vytvorené pri behu funkcie sú lokálne a po ukončení posledného príkazu zanikajú (ak nechceme, aby zanikli, musíme ich definovať ako globálne – príkazom ***global nazov_premennej***)

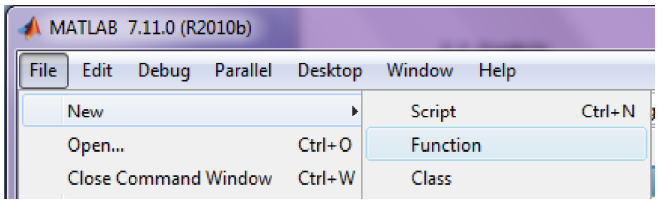
- **Tvorba vlastných funkcií**

a. Najskôr si musíme v pracovnom priestore Matlab-u nastaviť cestu k pracovnému adresáru



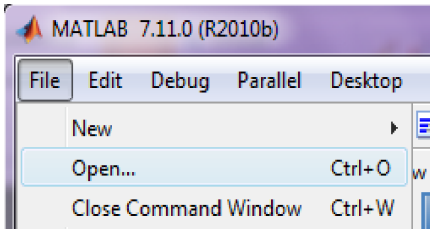
b. Otvorenie nového listu Editora ikonou 

alebo



ak chceme zmeniť už existujúcu funkciu použijeme 

alebo



c. Ako prvé musíme napísať hneď na začiatku deklaráciu funkcie a to :

function [výstup_1, výstup_2, ...] = meno_funkcie (vstup1, vstup2).

- funkcie nemusia mať žiaden vstup ani výstup
- ak existuje len jeden výstup nemusí byť v zátvorkách
- ak existuje viac vstupov (výstupov), oddeľujeme ich čiarkou
- názov funkcie by mal vystihovať jej funkčnosť
- názov sa nesmie zhodovať s názvom už existujúcej funkcie vstavanej v Matlabe alebo uživateľom predtým vytvorenej funkcie.

d. Ďalej môžeme pokračovať podobne ako pri tvorení skripu postupnosťou príkazov, ktoré však nie sú vykonávané hneď a samostatne ale až po **zavolaní funkcie**. xxx

e. Takúto funkciu musíme uložiť pod rovnakým názvom ako je ***meno_funkcie***. Ak ju uložíme aj s malým rozdielom, pri spustení ju nebude vedieť nájsť a systém nám vypíše chybu.

f. Volanie funkcie sa vykonáva v pracovnom priestore, kde funkciu zavoláme jej menom, alebo ju použijeme a voláme v skripte.

g. Ak pri tvorení funkcie si ju popíšeme pomocou komentárov, začínajúcich za deklaráciou funkcie a končiacich prvým príkazom, tieto komentáre si vieme zobraziť po zadaní príkazu ***help meno_funkcie*** alebo ***lookfor kľúčove_slovo*** v príkazovom riadku

```
>> help MSP
  Funkcia pre výpočet metódou slučkových prúdov
```

Príklad 1

a. Vytvor v simulačnom jazyku MATLAB skript pre výpočet odvesny pomocou Pytagorovej vety $a^2 + b^2 = c^2$.

```
a=input('Zadaj dĺžku prvej odvesny:  ');
b=input('Zadaj dĺžku druhej odvesny:  ');
c = sqrt(a^2 + b^2)
```

b. Vykonajte ten istý výpočet avšak s použitím funkcie pre výpočet odvesny Pytagorovou vetou.

prepona.m

```
function c=prepona(a,b)

c = sqrt(a^2 + b^2);
```

hl_prog.m **%skrip hlavného programu**

```
a=input('zadaj hodnotu prvej odvesny:  ');
b=input('zadaj hodnotu druhej odvesny:  ');
c=prepona(a,b)
```

2.5 Zadanie č. 1: Riešenie lineárnych algebraických rovníc s aplikáciou na elektrické obvody metódou slučkových prúdov a uzlových napätí v prostredí MATLAB

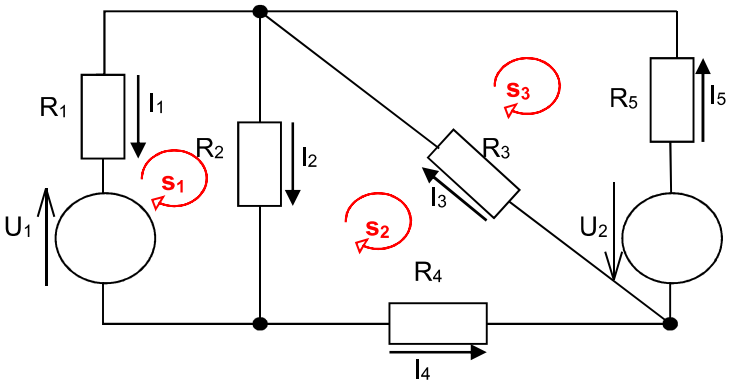
*Nasledujúce uvedené poznatky z oblasti riešenia elektrických obvodov pomocou **metódy slučkových prúdov** a **uzlových napätí** je potrebné využiť pri riešení Zadania č. 1. Vzorové zadanie číslo 1 sa nachádza nižšie.*

Metóda slučkových prúdov

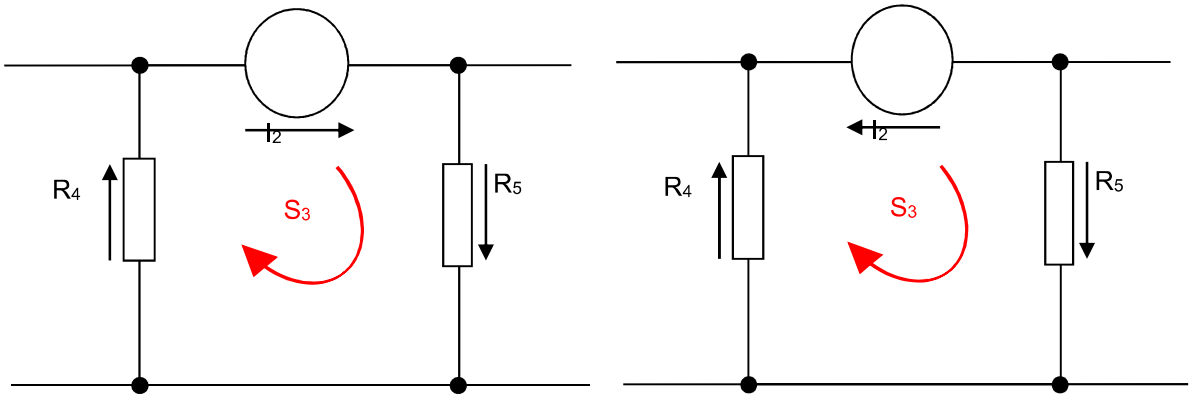
- je založená na poznatku, že prúdy vo vetvách stromu grafu sú jednoznačne určené prúdmi v nezávislých vetvách grafu
- spočíva v aplikácii 2. Kirchhoffovho zákona na všetky základné slučky grafu za predpokladu, že nimi tečie fiktívny, tzv. slučkový prúd, čím získame podmienkové rovnice pre daný obvod.

Postup:

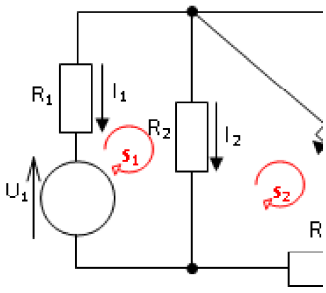
1. Zvolíme si smery slučkových prúdov v jednotlivých slučkách, smery napätia na zdrojoch a prúdy pretekajúce rezistormi.



2. Zistíme či sa v niektorej vetve nenachádza prúdový zdroj, potom sa hodnota slučkového prúdu bude rovnať prúdu zdroja s kladným alebo záporným znamienkom závisiacim od orientácie týchto prúdov.



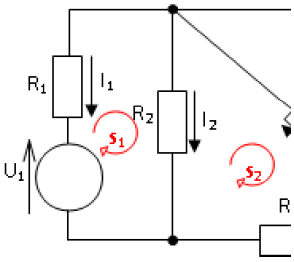
3. Rovnice zostavujeme nasledovne: slučkovým prúdom danej slučky vynásobíme súčet odporov danej slučky, ak niektorý rezistor susedí s ďalšími slučkami, potom odčítame ich súčin (odporu a slučkového prúdu), a nakoniec ak sa nachádza v slučke aj napäťový zdroj pričítame ho s kladným alebo záporným znamienkom podľa jeho smeru prúdenia.



$$\Rightarrow I_{s1} * (R_1 + R_2) - I_{s2} * R_2 + U_1 = 0$$

4. Za jednotlivé odpory a napätia a prúdy zdrojov dosadíme ich hodnoty a zapíšeme do rozšírenej matice (prúdy, ktoré sme dostali v 2.bode už do tejto matice nezapisujeme), z ktorej vypočítame hodnoty zvyšných slučkových prúdov.Konštanty dávame na pravú stranu matice. Ak sme zostavili rovnice správne matica by mala byť zrkadlová vzhľadom na hlavnú diagonálu.

5. Vytvoríme si podmienkové rovnice, z ktorých dostaneme konkrétne prúdy na jednotlivých rezistoroch.

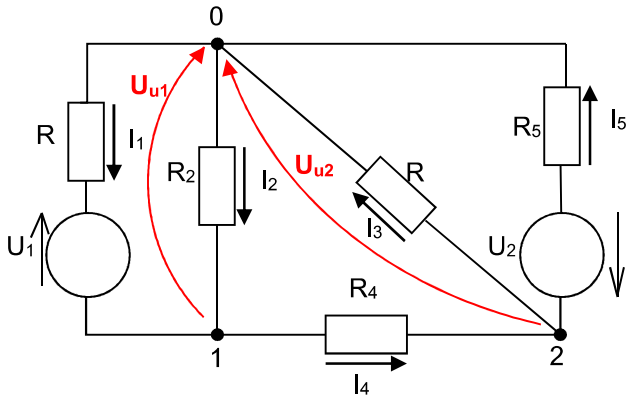


$$\Rightarrow I_1 = -I_{s1}$$
$$\Rightarrow I_2 = I_{s1} - I_{s2}$$
$$\Rightarrow \ldots$$

Metóda uzlových napätí

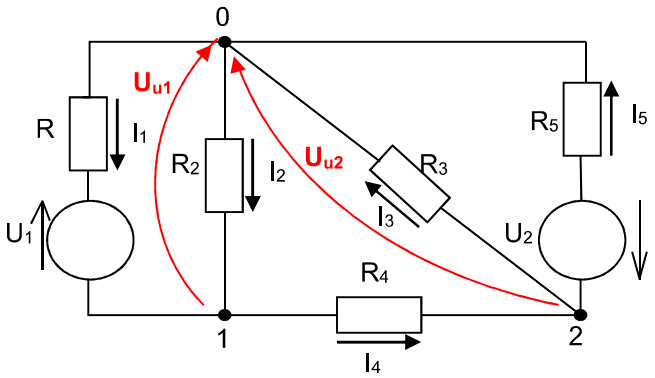
- je založená na poznatku, že napätia na nezávislých vetvách grafu sú jednoznačne určené napätiami na vetvách stromu grafu.
- spočíva v aplikácii 1. Kirchhoffovho zákona na všetky nezávislé uzly grafu za predpokladu, že na vetvách stromu grafu sú fiktívne, tzv. uzlové napätia, čím získame podmienkové rovnice pre daný obvod

1. Zvolíme si smery napätia na zdrojoch, prúdy pretekajúce odpormi a referenčný uzola označíme si aj zvyšné uzly.



2. Rovnice pre metódu uzlového napätia tvoríme nasledovne: pre každý uzol okrem referenčného vytvoríme rovnicu tak, že napätie od uzla vynásobíme súčtom prevrátených hodnôt odporov na prislúchajúcich vetvách a od toho odčítame súčin napätí a prevrátených

hodnôt z susedných uzlov. Nakoniec ešte pričítame súčin napätia na napäťovom zdroji a súčet prevrátených hodnôt na spoločnej vetve. Vo vetva s ideálnym napäťovým zdrojom je uzlové napätie rovné napätiu na zdroji.



$$U_{u1} * \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_4}\right) - U_{u2} * \frac{1}{R_4} - U_1 * \frac{1}{R_1} = 0$$

3. Zo zostavených rovníc zostavíme rozšírenú maticu dosadením za odpory a známe napätia ich hodnoty. Konštanty dávame na pravú stranu matice. Ak sme správne zostavili rovnice mala by byť matica zrkadlová vzhľadom na hlavnú diagonálu.

4. Vypočítaním matice dostaneme hodnoty uzlových napätí z ktorých si následne vytvoríme podmienkové rovnice pre výpočet jednotlivých prúdov.

$$I_1 = \frac{U_1 - U_{u1}}{R_1}, \quad I_3 = \frac{U_{u2}}{R_3}$$

ZADANIE 1:

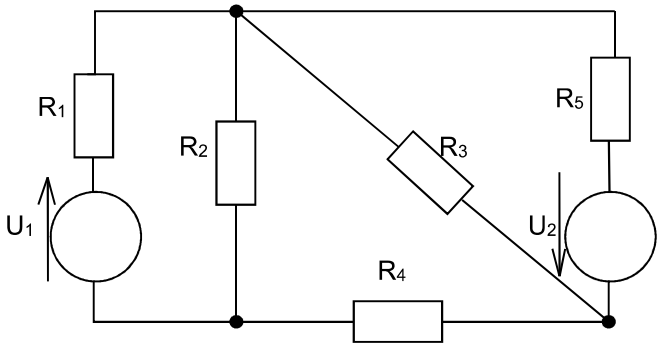
Z navrhnutej topológie elektrického obvodu vypočítajte prúdy vo vetvách metódou slučkových prúdov (MSP) a metódou uzlových napätí (MUN).

1. *Zvoliť topológiu obvodu a vykonať analytický výpočet pre obidve metódy a skúšku správnosti.*
2. *Algoritmicky vyriešiť a simulačne overiť v programovom prostredí MATLAB výpočet prúdov s využitím funkcií (MSP a MUN), [minimálne požiadavky na elektrický obvod: 3 slučky, 5 rezistorov, 2 zdroje]*

ÚLOHA:

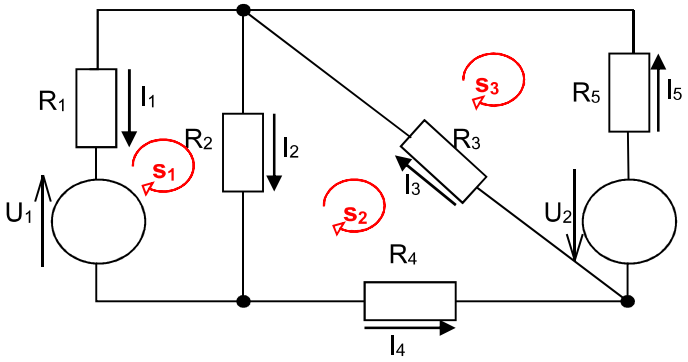
Vyriešiť analyticky zadaný obvod pomocou metódy slučkových prúdov a uzlových napätí a vytvor v programovom prostredí MATLAB program pre výpočet prúdov tohto obvodu.

$R_1 = 2\Omega$, $R_2 = 3\Omega$, $R_3 = 2\Omega$, $R_4 = 3\Omega$, $R_5 = 2\Omega$, $U_2 = 10$ V, $U_1 = 10$ V



1.a) **Metóda slučkových prúdov**

S1: $I_{s1} * (R_1 + R_2) - I_{s2} * R_2 = -U_1$
S2: $I_{s2} * (R_2 + R_3 + R_4) - I_{s1} * R_2 - I_{s3} * R_3 = 0$
S3: $I_{s3} * (R_3 + R_5) - I_{s2} * R_3 = -U_2$



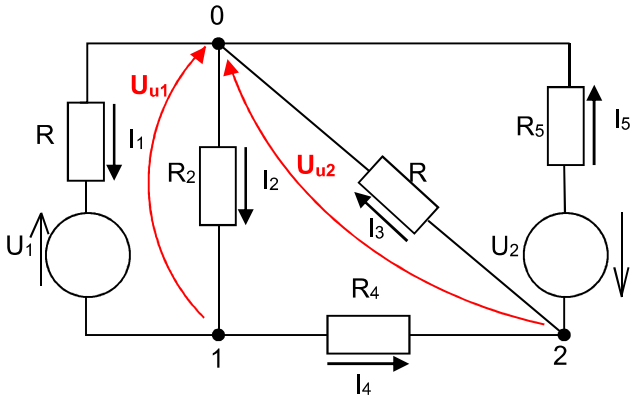
Zo získaných algebraických rovníc ***Ax = B*** vieme zostaviť maticu odporov ***A*** vektor naätí ***B***, a vypočítať prvky vektora x čo sú neznáme slučkové prúdy.

$$\begin{pmatrix} 5 & -3 & 0 & -10 \\ -3 & 8 & -2 & 0 \\ 0 & -2 & 4 & -10 \end{pmatrix} \Rightarrow \begin{matrix} I_{s1} = -\frac{85}{26} \\ I_{s2} = -\frac{55}{26} \\ I_{s3} = -\frac{185}{52} \end{matrix} \Rightarrow \begin{matrix} I_1 = -I_{s1} = \frac{85}{26} \\ I_2 = I_{s1} - I_{s2} = -\frac{15}{13} \\ I_3 = I_{s3} - I_{s2} = -\frac{75}{52} \\ I_4 = -I_{s2} = \frac{55}{26} \\ I_5 = -I_{s3} = \frac{185}{52} \end{matrix}$$

1.b) **Metóda uzlových napätí**

1: $U_{u1} * \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_4}\right) - U_{u2} * \frac{1}{R_4} = U_1 * \frac{1}{R_1}$

2: $U_{u2} * \left(\frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_5}\right) - U_{u1} * \frac{1}{R_4} = -U_2 * \frac{1}{R_5}$



Upravíme na tvar ***Ax = B***:

$$\left(\begin{array}{cc|c} \frac{6}{7} & -\frac{1}{3} & 5 \\ -\frac{1}{3} & \frac{4}{3} & -5 \end{array}\right) \Rightarrow \begin{array}{l} U_{u1} = \frac{45}{13} \\ U_{u2} = -\frac{75}{26} \end{array} \Rightarrow \begin{array}{l} I_1 = \frac{U_1 - U_{u1}}{R_1} = \frac{85}{26} \\ I_2 = \frac{U_{u1}}{R_2} = \frac{15}{13} \\ I_3 = \frac{U_{u2}}{R_3} = \frac{75}{52} \\ I_4 = \frac{U_{u1} - U_{u2}}{R_4} = \frac{55}{26} \\ I_5 = \frac{U_{u2} + U_2}{R_5} = \frac{185}{52} \end{array}$$

Pre jednotlivé výpočty si vytvoríme samostatné funkcie a pre hlavný program vytvoríme skript, v ktorom naše vytvorené funkcie použijeme.

2a. Programový návrh funkcie pre výpočet slučkových prúdov a prúdov v jednotlivých vetvách metódou slučkových prúdov

MSP.m

```
function I= MSP(U,R)
% Funkcia pre výpočet prúdov metódou slučkových prúdov

% vytvorte maticu A, ktorá bude obsahovať hodnoty ľavých strán rovníc
A = [R(1)+R(2)              -R(2)                          0;
-R(2)           R(2)+R(3)+R(4)           -R(3);
0                    -R(3)                   R(3)+R(5)]
% vytvorte maticu B, ktorá bude obsahovať hodnoty pravých strán rovníc
B = [ - U(1);
0;
- U(2)]

%výpočet slučkových prúdov realizujte ľavým delením
IS = A\B

%dopočítajte zvyšné hodnoty prúdov
I(1) = -IS(1);
I(2) = IS(1) - IS(2);
I(3) =  IS(3) -  IS(2);
I(4) =  -IS(2);
I(5) = - IS(3);
```

2b Programový návrh funkcie pre výpočet uzlových napätí a prúdov v jednotlivých vetvách

MUN.m

```
function I=MUN(U,R)
%metóda uzlových napätí

%vytvorte maticu A, ktorá bude obsahovať hodnoty ľavých strán rovníc
A = [(1/R(1)+1/R(2)+1/R(4)) -1/R(4); -1/R(4)  (1/R(3)+1/R(4)+1/R(5)) ]

%vytvorte maticu B, ktorá bude obsahovať hodnoty pravých strán rovníc
B = [ U(1)*(1/R(1));         -U(2)*(1/R(5))]
```

```
% výpočet uzlových napätí realizujte ľavým delením
Uu = A\B

%dopočítajte zvyšné hodnoty prúdov
I(1) = (U(1)-Uu(1))/R(1);
I(2) = Uu(1)/R(2);
I(3) = Uu(2)/R(3);
I(4) = (Uu(1)-Uu(2))/R(4);
I(5) = (Uu(2)+U(2))/R(5);
```

Skúška správnosti výpočtu

skuska.m

```
function skuska(I)

if I(1)-I(2)-I(4)<1e-6
if I(4)+I(3)-I(5)<1e-6
if I(2)-I(1)-I(3)+I(5)<1e-6
            disp('Prúdy vo všetkých vetvách vyhovujú 1.KZ.')
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
```

Hlavný program – načítanie hodnôt odporov a napätí a použitie vytvorených uživateľských funkcií MUN a MSP

hl_prog.m

```
% zadávanie hodnôt odporov
R1 = input ('Zadaj hodnotu R1 = ');
R2 = input ('Zadaj hodnotu R2 = ');
R3 = input ('Zadaj hodnotu R3 = ');
R4 = input ('Zadaj hodnotu R4 = ');
R5 = input ('Zadaj hodnotu R5 = ');
disp('*************************************************************')
% zadávanie hodnôt prúdov
U1 = input ('Zadaj hodnotu napäťového zdroja U1 = ');
U2 = input ('Zadaj hodnotu napäťového zdroja U2 = ');

disp('*************************************************************')
disp('')
disp('')
R = [R1, R2, R3, R4, R5 ];
U = [U1, U2 ];
disp ('metoda sluckovych prudov')
I=MSP(U,R)
disp('***************skúška*****************')
skuska(I)
disp('*************************************************************')
disp('metoda uzlovych napati')
I=MUN(U,R)
disp('***************skúška*****************')
skuska(I)
```