ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROENERGETIKY A EKOLOGIE

DIPLOMOVÁ PRÁCE

Rozběhové charakteristiky asynchronního stroje

Bc. Konstantin Ryba **2014**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2013/2014**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Konstantin RYBA**
Osobní číslo:	**E12N0024K**
Studijní program:	**N2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Název tématu:	**Rozběhové charakteristiky asynchronního stroje**
Zadávající katedra:	**Katedra elektroenergetiky a ekologie**

Zásady pro vypracování:

1. Navrhněte vhodnou metodiku pro výpočet rozběhových charakteristik a provozních stavů asynchronního stroje.
2. Vytvořte model včetně uživatelského prostředí pro jejich výpočet.
3. Výsledky graficky a číselně interpretujte.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Pyrhönen, J., Jokinen, T., Hrabovcová, V.: Design of rotation electrical machines, John Wiley & Sons, Ltd, 2008**
2. **Kopylov, I., P.: Stavba elektrických strojů, SNTL/MIR, 1988**
3. **Nápověda SW použitého k analýze**

Vedoucí diplomové práce: **Ing. Vladimír Kindl, Ph.D.**
Katedra elektromechaniky a výkonové elektroniky

Datum zadání diplomové práce: **14. října 2013**

Termín odevzdání diplomové práce: **12. května 2014**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

Doc. Ing. Karel Noháč, Ph.D.
vedoucí katedry

V Plzni dne 14. října 2013

Abstrakt

Tato práce se zabývá výpočtem rozběhových charakteristik asynchronního stroje. K simulaci těchto charakteristik byl vytvořen program, který pomůže uživateli zohlednit vliv tvaru drážky při návrhu asynchronního stroje. Vzhledem k nutné znalosti MATLABu, obsahuje práce také vysvětlení prvků tohoto programu a popis postupu k získání požadovaných výstupů.

Klíčová slova

Asynchronní stroj, rozběhové charakteristiky, MATLAB, skinefekt, simulace.

Abstract

This thesis deals with an analytical method for calculation of run up characteristics of an induction machine. The simulation program was implemented to help the user to specify the effect of the shape of the rotor slot, during the machines design. While the knowledge of Matlab needed, the thesis contains an explanation of the elements of this program and description of the procedure for obtaining the required outputs.

Key words

Induction machine, torque/slip characteristics, MATLAB, skin effect, simulation

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

.....

podpis

V Plzni dne 29.4.2014 Konstantin Ryba

Poděkování

Na tomto místě bych rád poděkoval vedoucím práce, panu doktoru Vladimíru Kindlovi, za cenné rady a čas, který mi věnoval při konzultacích. Dále bych rád poděkoval celé své rodině a přítelkyni za podporu a trpělivost během celého mého studia. Děkuji.

Obsah

Seznam symbolů a zkratek

b_d	šířka drážky	[m]
b_t	šířka tyče	[m]
f_2	frekvence proudu v rotoru	[Hz]
h_t	výška tyče v drážce	[m]
I_1	statorový proud	[A]
I_2	rotorový proud	[A]
k_r	činitel změny odporu	[-]
k_x	činitel změny magnetické vodivost	[-]
n	otáčky rotoru	[Ot*min^{-1}]
n_s	synchronní otáčky točivého magnetického pole	[Ot*min^{-1}]
R_1	statorový odpor	[Ω]
R_2	rotorový odpor	[Ω]
R_{fe}	odpore v příčné větvi	[Ω]
r_i	odpor i-té vrstvy	[Ω]
R_t	odpor tyče bez vlivu skinefektu	[Ω]
$R_{t\xi}$	odpor tyče s vlivem skinefektu	[Ω]
s	skluz	[-]
S_i	průřez i-té vrstvy	[m^2]
U_1	napětí na statoru	[V]
U_2	napětí na rotoru	[V]
U_{i1}	indukované napětí	[V]
U_{i2}	indukované napětí	[V]
U_{i20}	přepočtené indukované napětí	[V]
$X_{1\sigma}$	rozptylová reaktance statoru	[Ω]
$X_{2\sigma}$	rozptylová reaktance rotoru	[Ω]
x_i	reaktance i-té vrstvy	[Ω]
Z_{1h}	impedance příčné větve	[Ω]
ω_1	synchronní úhlová rychlost statoru	[rad*s^{-1}]
ω_2	úhlová rychlost rotoru	[rad*s^{-1}]
λ'_d	magnetická vodivost bez vlivu skinefektu	[H]
$\lambda'_{d\xi}$	magnetická vodivost s vlivem skinefektu	[H]
λ_i	geometrickou vodivost magnetické trubice	[-]
μ_0	permeabilita vakua	[H*m^{-1}]
ξ	redukovaná výška vodiče	[m]
ρ_t	měrný elektrický odpor	[Ω*m]
$\rho_{t\vartheta}$	rezistivita materiálu tyče	[Ω*m]

Úvod

Asynchronní stroj, ačkoli byl vynalezen již před více než sto třiceti lety, je stále nejpoužívanějším elektrickým strojem. Princip tohoto stroje je vysvětlen v úvodní části této práce, na kterou navazuje vysvětlení pojmu náhradní schéma. V kapitole konstrukční řešení rotorového vinutí je popsán rozdíl a výhody asynchronních motorů podle typu vinutí, typu klece a prvně se zde zmiňuje jev zvaný skinefekt. Tím se autor dostává k rozběhovým charakteristikám.

Asynchronní stroj může být používán ve dvou pracovních režimech a to buď jako motor nebo jako generátor. Bude-li třeba použít například motor v nějakém pohonu, nestačí pouze zjistit hodnoty uvedené na štítku motoru, ale je třeba znát také rozběhové charakteristiky, jako jsou momentová a výkonová charakteristika, průběh proudů a také průběh účiníku. Důležitou hodnotou, kterou je třeba brát v úvahu při výběru motoru je tzv. záběrný moment, který se u jednotlivých typů motorů může výrazně lišit. Tvar momentové charakteristiky je ovlivněn mimo jiné i tvarem rotorové drážky. Při chybně navrženém tvaru drážky by mohlo dojít k nadměrnému ohřevu části zubu a tím i celkové poruše stroje.

Hlavním cílem této práce bylo zjistit, zda vůbec a pokud, tak jak velký vliv má tvar rotorové drážky na průběh rozběhových charakteristik. Za tímto účelem byl v programu MATLAB vytvořen program, který pomůže uživateli při návrhu asynchronního stroje zjistit, jak by drobná změna v tvaru drážky mohla přizpůsobit stroj daným požadavkům.

Vzhledem k tomu, že k použití vytvořeného programu je třeba alespoň základních znalostí práce s MATLABem, jsou níže v této práci popsány jednotlivé prvky MATLABU a především krok po kroku tak, jak by měl uživatel postupovat, aby se dostal k požadovaným výstupům.

Na závěr jsou pomocí programu porovnány vlivy konkrétních dvou tvarů drážek na momentovou a výkonovou charakteristiku, čímž dojde k ověření funkčnosti programu.

1 Asynchronní stroj

Asynchronní stroj byl vynalezen v roce 1883 americkým vynálezcem srbského původu Nikolou Teslou. Od té doby stroj prošel různými stadii vývoje a v dnešní době, díky své jednoduché a spolehlivé konstrukci, je nejpoužívanějším elektrickým točivým strojem. Asynchronní stroj je používán především jako motor, kde přeměňuje elektrickou energii na mechanickou práci.

1.1 Princip asynchronního stroje

Princip asynchronního stroje, v zahraniční literatuře překládán jako induction machine, spočívá ve vzájemném působení dvou polí - statorového a rotorového. Točivé magnetické pole statoru vzniká díky statorovému třífázovému vinutí, které je prostorově posunuté o 120°. Toto vinutí je napájeno třemi proudy, kde jednotlivé fáze jsou vzájemně časově posunuty o 120°. Statorové pole indukuje napětí do rotorového vinutí, které se v tomto poli nachází. Jelikož rotorové vinutí je ve většině případů spojeno do krátka, začne rotorem protékat proud, který zapříčiní vznik rotorového pole. Vzájemnou interakci těchto polí dochází k vyvolání vnitřního elektromagnetického momentu. Vzniklý moment otáčí rotorem ve smyslu otáčení pole statoru. Dochází k přeměně elektrické energie na mechanickou a asynchronní stroj (dále AS) je tak v režimu motor.

AS může pracovat i v režimu generátor. V tomto případě dodává do sítě činný výkon, čili mechanickou energii přeměňuje na elektrickou. Je nutné, aby byl AS připojen k síti, která musí stroj magnetovat a zároveň mechanické otáčky musí být vyšší než synchronní.

Veličina popisující rozdíl mezi úhlovými rychlostmi pole statoru a rotou se nazývá skluz (angl. Slip) a je označována jako s. Jedná se o bezrozměrnou veličinu, kdy v intervalu od 0 do 1 stroj pracuje v motorickém režimu. V oblasti nad-synchronních otáček je skluz $s < 0$ a stroj je v režimu generátor.

$$s = \frac{(\omega_1 - \omega)}{\omega_1} = \frac{(n_s - n)}{n_s} \quad [-] \tag{1.1}$$

Kde:

s – skluz [-],

ω_1 – synchronní úhlová rychlost statoru [rad/s],

ω – úhlová rychlost rotoru [rad/s],

n_s – synchronní otáčky točivého magnetického pole [ot/min],

n – otáčky rotoru [ot/min].

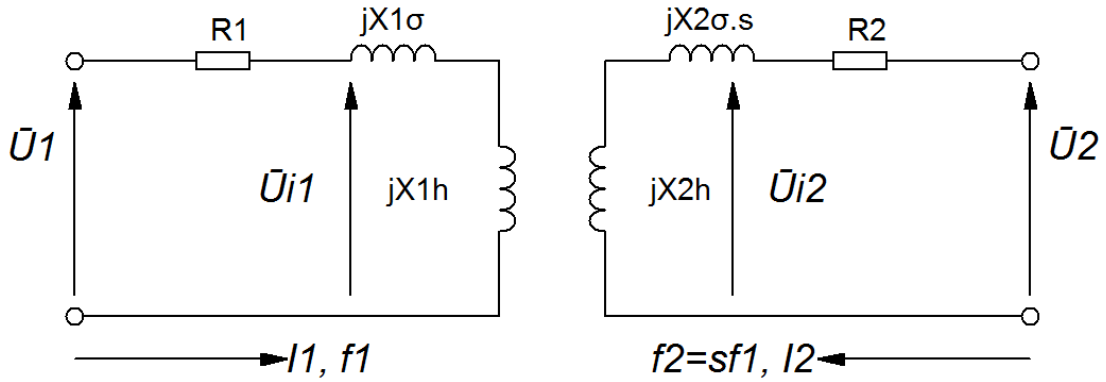
1.2 Náhradní schéma asynchronního stroje

Náhradním schématem je nazýván takový elektrický obvod, který nevysvětluje fyzikální jevy asynchronního stroje, ale pouze popisuje jeho chování při určitých omezeních. Zjednodušené předpoklady pro tvorbu náhradního schématu (dále jen NS), který má tvar T-článku jsou [1]:

- Harmonické průběhy elektrických veličin,
- Energie se přenáší ze sítě jen pracovní harmonickou,
- Parametry jsou konstantní a symetrické,
- Z počátku zanedbávány ztráty v železe.

Při respektování zjednodušených předpokladů a předpokladu symetrie, můžeme třífázový model redukovat na model jednofázový.

Takto získaný obvod lze bez obtíží analyticky řešit pomocí symbolicko-komplexní metody. Tato metoda byla před příchodem výpočetní techniky prakticky jediná, jež se dala použít pro ověření výsledků během návrhu stroje.



Obrázek 1.1- Obvodové znázornění asynchronního stroje
Zdroj: Vlastní zpracování podle [2]

Po analýze NS na *Obr. 1.1* dostaneme základní rovince 1.2:

$$\bar{U}_1 = (R_1 + jX_{1\sigma})\bar{I}_1 + \bar{U}_{i1}$$
$$\bar{U}_2 = (R_2 + jsX_{2\sigma})\bar{I}_2 + \bar{U}_{i2} \quad \textbf{1.2}$$

Kde:

$\bar{U}_1$ – napětí na statoru [V],

R_1 – statorový odpor [Ω],

$X_{1\sigma}$– rozptylová reaktance statoru[Ω],

$\bar{I}_1$ – statorový proud [A],

$\bar{U}_{\mathrm{i1}}$ – indukované napětí [V],

$\bar{U}_2$ – napětí na rotoru [V],

R_2 – rotorový odpor [Ω],

$X_{2\sigma}$– rozptylová reaktance rotoru[Ω],

$\bar{I}_2$ – rotorový proud [A],

$\bar{U}_{\mathrm{i2}}$ – indukované napětí [V],

Díky rozdílné frekvenci statoru a rotoru je třeba rovnici pro rotor v soustavě rovnic 1.2 vydělit skluzem *s*. Výsledkem jsou rovnice převedené na kmitočet statoru. Dále je pak třeba vzít v úvahu, že:

$$\bar{U}_{\mathrm{i2}} = \mathrm{s} * \bar{U}_{i20} \quad \textbf{1.3}$$

Získáváme:

$$\bar{U}_1 = (R_1 + jX_{1\sigma})\bar{I}_1 + \bar{U}_{i1}$$
$$\frac{\bar{U}_2}{s} = \left(\frac{R_2}{s} + jX_{2\sigma}\right)\bar{I}_2 + \bar{U}_{i20} \quad \textbf{1.4}$$

Kde:

$\bar{U}_{\mathrm{i20}}$ – přepočtené indukované napětí [V].

Jako další krok je třeba přepočítat rotorovou stranu na statorovou, k tomu je nutné vypočítat převod [2].

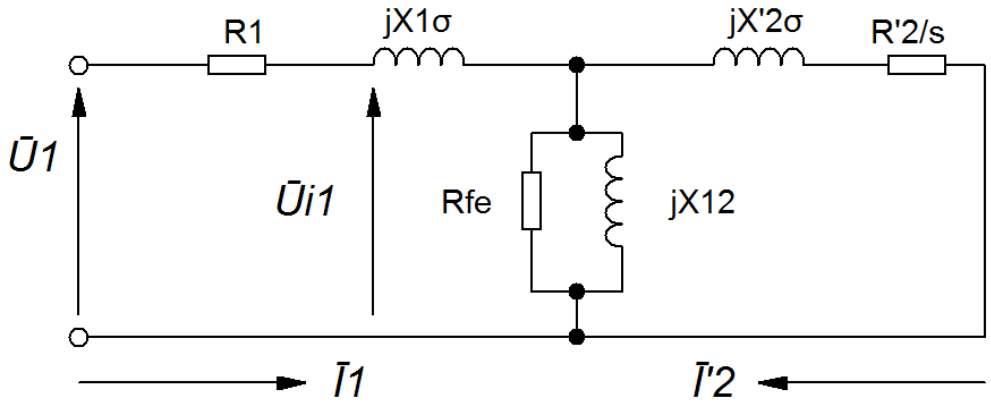
Za předpokladu, že rotorové vinutí je spojeno do krátka bude platit rovnice 1.5:

$$U_2 = 0 \qquad \textbf{1.5}$$

Tímto dostáváme konečné základní rovnice NS *1.6*. Díky provedeným úpravám lze v NS spojit stator a rotor a tím nakreslit dvojbran typu T *Obr. 1.2*.

$$\bar{U}_1 = (R_1 + jX_{1\sigma})\bar{I}_1 + \bar{U}_{i1}$$

$$0 = \left(\frac{R_2'}{s} + jX_{2\sigma}'\right)\bar{I}_2' + \bar{U}_{i1} \qquad \textbf{1.6}$$



Obrázek 1.2 - NS asynchronního stroje ve tvaru T
Zdroj: Vlastní zpracování podle [2]

Jak bylo už dříve uvedeno, jedním ze zjednodušovacích předpokladů je zanedbání ztrát v železe. Nyní je ale třeba tyto ztráty také započítat. Tyto ztráty v NS jsou reprezentovány odporem v příčné větvi R_{fe}. Z NS lze dopočítat také impedanci příčné větve, jak znázorňuje rovnice 1.7:

$$\bar{Z}_{1\mathrm{h}} = \frac{jX_{12} * R_{fe}}{R_{fe} + jX_{12}} \qquad \textbf{1.7}$$

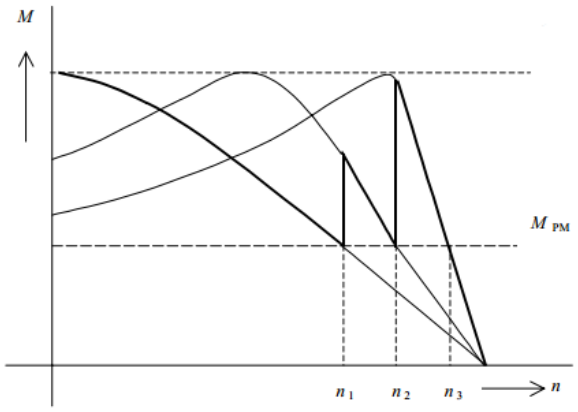
Kde:

$\bar{Z}_{1\mathrm{h}}$ – impedance příčné větve [Ω]

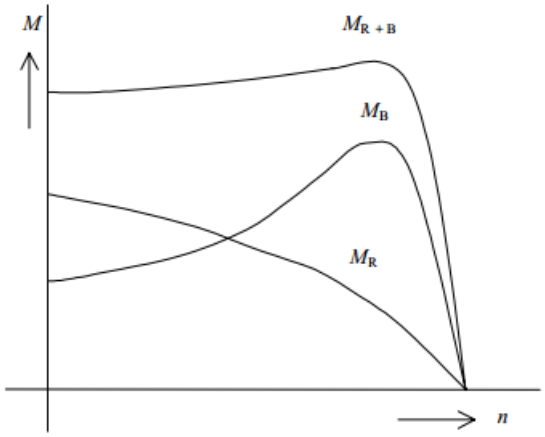
1.3 Konstrukční řešení rotorového vinutí

Asynchronní motor můžeme rozdělit podle provedení rotorového vinutí do dvou kategorií. V první řadě to je asynchronní motor (dále AM) s kroužkovým rotorem a AM s klecí na krátko. První typ má v drážkách rotoru třífázové vinutí stejné jako na statoru. Toto vinutí je spojeno do hvězdy a vyvedeno na nalisované kroužky na hřídeli. Na tyto kroužky doléhají kluzné uhlíkové kontakty, jinak zvané kartáče. Přes tyto kontakty je vinutí vyvedeno ven na svorkovnici. Největší výhodou tohoto konstrukčního řešení je

bezesporu možnost připojit do obvodu rotoru rozběhové odpory a tím upravit tvar momentové charakteristiky tak, aby byl maximální moment roven záběrnému. Při rozběhu z nuly, je třeba zařadit největší odpor a s rostoucími otáčkami postupně vyřazovat odporové stupně až do vyzkratování vinutí. Po vyzkratování se AM s kroužkovou kotvou začne chovat jako AM s kotvou do krátka. Na *Obr 1.3* můžeme vidět skokovou změnu momentu při vyřazování odporu z obvodu.



Obrázek 1.3 - Momentová char. AM s kotvou kroužkovou
Zdroj: [5]



Obrázek 1.4 - Momentová char AM s dvojitou klecí na krátko
Zdroj: [5]

Klecí nakrátko se rozumí soustava rotorových tyčí, které jsou uložené po obvodě rotoru a jsou vzájemně propojené na obou koncích kruhy. Zvláštními druhy těchto klecí jsou tzv. klec dvojitá a klec vírová. Klec vírová se od klasické liší hloubkou drážky, u této klece je využit povrchový jev, skinefekt, ke zvýšení záběrného momentu. Obdobně to je i s klecí dvojitou, kde hlavní myšlenkou je vytvoření jedné klece pro rozběh a druhé pro ustálený chod. Výsledný moment takového řešení je součtem momentů obou klecí a je zobrazen na *Obr. 1.4*

1.4 Rozběhové charakteristiky

Podle [1] je zvláštním druhem povrchových jevů jev zvaný skinefekt. Tento jev se projevuje v rotorových tyčích, které mají velkou výšku drážky a kde dochází ke zvýšení frekvence proudu. Vlivem skinefektu dochází k vytlačování proudu z celé plochy drážky k její části blízké vzduchové mezeře, tj. ve směru kolmém na směr rozptylových indukčních čar v drážce. To je zapříčiněné tím, že proudovodič vlivem vlastního magnetického pole indukuje sám do sebe vířivé proudy, které způsobí, že proudová hustota je vytlačovaná na povrch proudovodiče. Tím, že je proudová hustota rozdílná, tedy v horní části drážky vzrůstá a v dolní klesá, dochází k tomu, že odpor tyče roste a reaktance tyče klesá. Tyto změny mají nemalý vliv na rozběhové charakteristiky a je nutné s nimi počítat.

Při návrhu asynchronního motoru s klecovým vinutím se tento jev projevuje příznivě tím, že zvětšuje záběrný moment motoru. Na druhou stranu, špatným návrhem tvaru tyče může dojít k nadměrnému ohřevu části zubu a tím k jeho deformaci nebo dokonce vylomení a následné havárii stroje.

Dle [3] je výhodnější při výpočtech určovat místo odporu a reaktance tyčí jejich poměrnou změnu a to činitele k_r a k_x. Činitel k_r udává, kolikrát se zvětší odpor tyče s vlivem skinefektu oproti odporu tyče bez jeho působení.

$$k_{\mathrm{R}} = \frac{R_{t\xi}}{R_t} \qquad \textbf{1.8}$$

Kde:

k_{R} – činitel změny odporu [-]

$R_{t\xi}$ – odpor tyče s vlivem skinefektu [Ω]

R_t – odpor tyče bez vlivu skinefektu [Ω]

Obdobně k_x udává, jak se změní činitel magnetické vodivosti části drážky s procházejícím proudem a vlivem skinefektu oproti téže části drážky při rovnoměrném rozložení proudu hustoty v tyči.

$$k_{\mathrm{x}} = \frac{\lambda'_{d\xi}}{\lambda'_d} \qquad \textbf{1.9}$$

Kde:

k_{x}– činitel změny magnetické vodivost [-]

$\lambda'_{d\xi}$– magnetická vodivost s vlivem skinefektu [H]

λ'_d – magnetická vodivost bez vlivu skinefektu [H]

Pro analytické řešení činitelů k_x a k_r existují tyto vztahy 1.10:

$$k_R = \xi \frac{\sinh 2\xi + \sin 2\xi}{\cosh 2\xi - \cos 2\xi}$$
$$k_x = \frac{3}{2\xi} \frac{\sinh 2\xi - \sin 2\xi}{\cosh 2\xi - \cos 2\xi} \qquad \textbf{1.10}$$

Ovšem tyto vztahy jsou omezeny několika předpoklady, a to:

- Obdélníkový tvar tyče
- Konstantní rezistivita materiálu tyče na celé ploše průřezu.
- Permeabilita železa blížící se nekonečnu.
- Přímkové magnetické indukční čáry v drážce

V těchto vztazích figuruje ξ tzv. redukovaná výška vodiče. Jedná se o bezrozměrnou veličinu, která je definovaná vztahem 1.11:

$$\xi = 2\pi h_t \sqrt{\frac{b_t f_2}{b_d \rho_{t\vartheta}}} * 10^{-7} \qquad \textbf{1.11}$$

Kde:

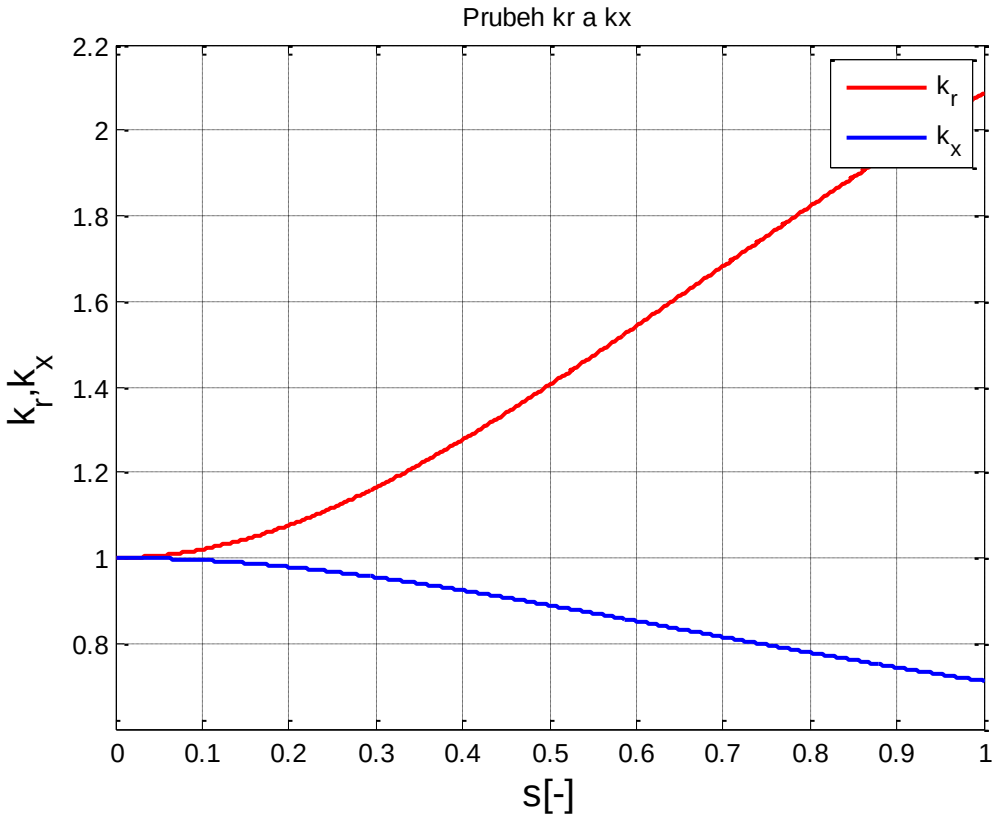
h_t - výška tyče v drážce [m]

b_t,b_d - šířka tyče a šířka drážky [m]

f_2 - frekvence proudu v rotoru [Hz]

$\rho_{t\vartheta}$ - rezistivita materiálu tyče [Ω*m]

Na následujícím obrázku Obr. 1.5 je znázorněn průběh činitelů k_r a k_x pro obdélníkový tvar drážky s výškou 25mm a šířkou 1mm. Tyto průběhy jsou počítány odlišnou metodou, než je uvedeno výše. Ta je popsána v následující kapitole.



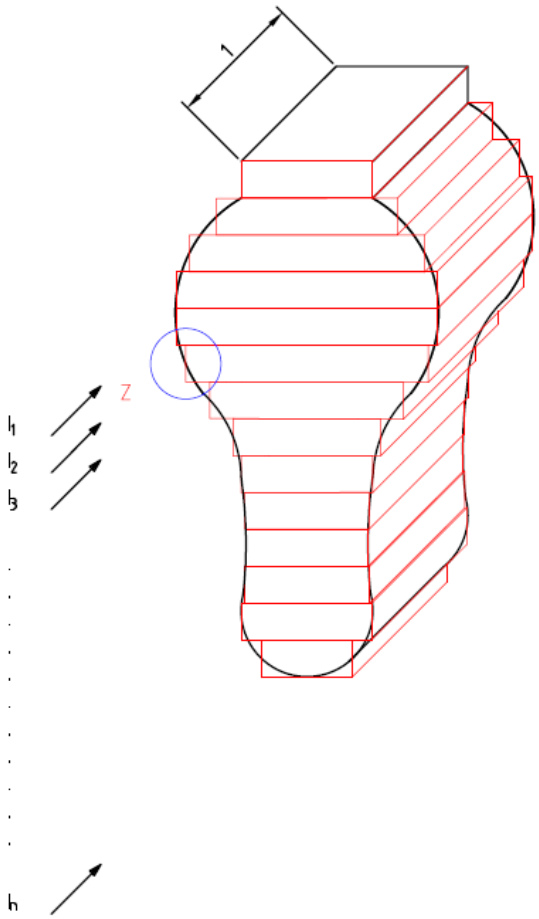
Obrázek 1.5 – Průběh k_r a k_x
Zdroj: Vlastní zpracování

1.5 Obecná metoda výpočtu činitelů k_r a k_x.

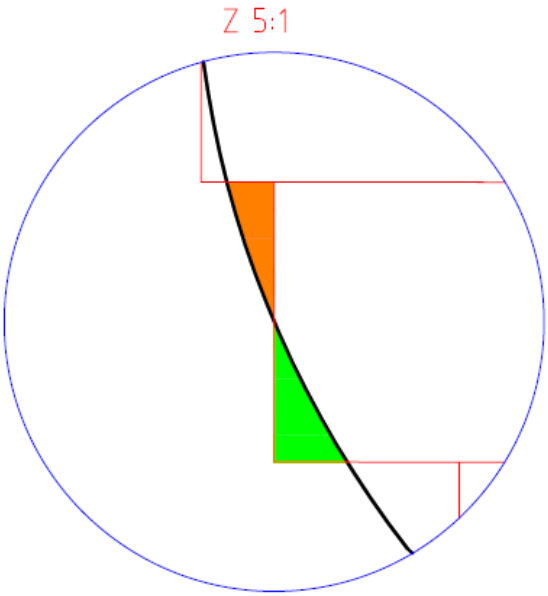
Hlavní myšlenkou této metody, která je popsaná v [3], je rozdělení zkoumané drážky na *n* dílčích drážek obdélníkového průřezu, pro které jsou následně určeny jejich odpor a magnetická vodivost. Výsledné hodnoty jsou uspořádány do impedanční matice. Aplikací Ohmova zákona lze snadno vypočítat celkový proud procházející tyčí a tím i celkový odpor a celkovou reaktanci celé tyče.

Touto metodou lze řešit drážku rotorové tyče libovolného tvaru, dokonce i drážku s proměnnou materiálovou rezistivitou. Díky těmto poznatkům, můžeme říct, že tato metoda je mnohem univerzálněji použitelná než metoda předcházející a proto byla použita i při výpočtech odporu a reaktance rotorové tyče ve skriptu, který je součástí této práce.

Rozdělení drážky na *n* dílčích obdélníkových drážek navzájem od sebe izolovaných, tak jako je naznačeno na *Obr. 1.6* vnáší do výpočtu určitou chybu. Tato chyba je způsobena tím, že elementární obdélníková drážka nekopíruje přesně daný tvar drážky a proto při výpočtu můžeme zjistit, že suma obsahu průřezů všech elementárních tyčí nemusí odpovídat obsahu průřezů drážky původní. K minimalizaci této chyby nám dopomůže správná volba šířky elementární vrstvy, tak jak je to naznačeno na *Obr. 1.7.* Ačkoli se oranžová a zelená plocha zdají být stejné, není tomu tak. Proto je třeba volit výšku elementární vrstvy co nejmenší, aby se tento rozdíl minimalizoval a tím se chyba vykompenzovala. Můžeme tvrdit, že při počtu elementárních drážek jdoucích do nekonečna, budou oba obsahy průřezu stejné. Jelikož v dnešní době výpočetní technika už má dostatečný výkon, lze při návrhu volit velký počet elementárních vrstev a tak dosáhnout požadované přesnosti. V [1] se můžeme dočíst, že už rozdělení na 20 elementárních vrstev zajistí přesnost výpočtů stejnou jako při analytickém řešení.

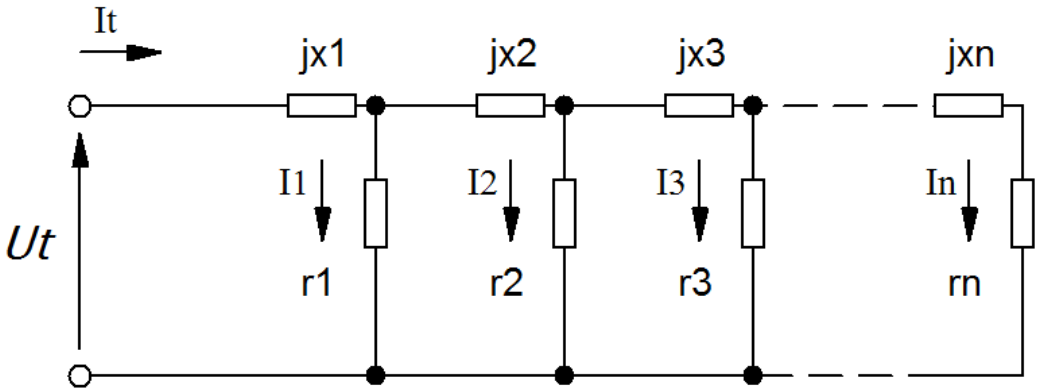


Obrázek 1.7 - Rozdělení rotorové tyče na n dílčích obdélníkových
Zdroj: Vlastní zpracování podle [3]



Obrázek 1.6 - Detail elementární vrstvy
Zdroj: Vlastní zpracování

Pro účely výpočtu můžeme vytvořit NS, které interpretuje rozdělení rotorové tyče na n elementárních vrstev. Toto NS může vypadat následujícím způsobem.



Obrázek 1.8 - NS pro výpočet odporu a reaktancí elementárních vrstev
Zdroj: Vlastní zpracování podle [3]

Dále je třeba zohlednit následující dva předpoklady:

- Pole drážky je plošně rovnoměrné
- Hustota proudu podél indukční čáry se nemění

Při dostatečně malé výšce vrstvy tyto předpoklady splněny jsou, a proto můžeme přistoupit k výpočtu odporu 1.12 a reaktance 1.13 elementární vrstvy *i* na jednotku délky.

$$r_{\mathrm{i}} = \frac{\rho_t}{S_i} \qquad \textbf{1.12}$$

Kde:

r_{i}– odpor i-té vrstvy [Ω*m^{-1}]

ρ_t– měrný elektrický odpor [Ω*m]

S_i– průřez i-té vrstvy [m^2]

$$x_{\mathrm{i}} = \omega_2 \mu_0 \lambda_i \qquad \textbf{1.13}$$

Kde:

x_{i}– reaktance i-té vrstvy [Ω*m^{-1}]

ω_2– uhlová rychlost [rad*s^{-1}]

μ_0– permeabilita vakua [H*m^{-1}]

λ_{i} značí geometrickou vodivost magnetické trubice, V našem případě počítaná jako poměr šířky i-té vrstvy k její délce.

Pro výpočet odporu tyče, s uvažováním povrchového jevu a činitele magnetické vodivosti části drážky, je zapotřebí spočítat proudy v jednotlivých elementárních vrstvách. Nejsnazší je vytvořit matici, 1.15, ze soustavy rovnic, 1.14. Pokud položíme I_n=1, bude možno najít zbývající proudy v poměrných jednotkách a tím získat celkový proud tyče.

$$jx_1\bar{I}_1 + \left(r_2 + j\sum_1^2 x_i\right)\bar{I}_2 + j\sum_1^2 x_i\bar{I}_3 + \cdots + j\sum_1^2 x_i\bar{I}_n = \bar{U}_t$$

$$jx_1\bar{I}_1 + j\sum_1^2 x_i\bar{I}_2 + j\sum_1^3 x_i\bar{I}_3 + \cdots + \left(r_n + j\sum_1^n x_i\right)\bar{I}_n = \bar{U}_t \qquad \textbf{1.14}$$

$$\bar{I}_1 + \bar{I}_2 + \bar{I}_3 + \cdots + \bar{I}_n = \bar{I}_t$$

$$\begin{bmatrix} r_1 + jx_1 & jx_1 & jx_1 & jx_1 & \dots & jx_1 \\ jx_1 & r_2 + j\sum_1^2 x_i & j\sum_1^2 x_i & j\sum_1^2 x_i & \dots & j\sum_1^2 x_i \\ jx_1 & j\sum_1^2 x_i & r_3 + j\sum_1^3 x_i & j\sum_1^3 x_i & \dots & j\sum_1^3 x_i \\ \dots & \dots & \dots & \dots & \dots & \dots \\ jx_1 & j\sum_1^2 x_i & j\sum_1^3 x_i & j\sum_1^4 x_i & \dots & r_n + j\sum_1^n x_i \end{bmatrix} \cdot \begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \bar{I}_3 \\ \dots \\ \bar{I}_n \end{bmatrix} = \begin{bmatrix} \bar{U}_t \\ \bar{U}_t \\ \bar{U}_t \\ \dots \\ \bar{U}_t \end{bmatrix} \qquad \textbf{1.15}$$

Po vypočtení jednotlivých proudů elementárních vrstev, je možno spočítat dle rovnice 1.16 celkový proud tyče.

$$\bar{I}_\mathrm{t} = \sum_1^n \bar{I}_i \qquad \textbf{1.16}$$

Nyní jsou známé všechny veličiny potřebné k výpočtu odporu tyče rotoru s uvažováním povrchového jevu, 1.17. Ten je třeba k určení činitele zvětšení odporu tyče, 1.18.

$$\mathrm{r}_{\mathrm{t}\xi} = \frac{\sum_{i=1}^n (\mathrm{r_i I_i^2})}{\mathrm{I_t^2}} \qquad \textbf{1.17}$$

$$\mathrm{k_r} = \frac{\sum_{i=1}^n (\mathrm{r_i I_i^2})}{\mathrm{r_t I_t^2}} \qquad \textbf{1.18}$$

Obdobným způsobem vypočítáme i činitel magnetické vodivosti části drážky s uvažováním povrchového jevu.

$$\lambda'_{\mathrm{d}2\xi} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\lambda_\mathrm{i} |\sum_{k=n}^n \mathrm{I_k}|^2)}{\mathrm{I_t^2}} \qquad \textbf{1.19}$$

Pro potřebu výpočtu činitele zmenšení magnetické vodivosti, je nutné spočítat i činitele magnetické vodivosti části drážky, bez uvažování povrchového jevu. Vzhledem k tomu, že zde popisovaná metoda má být univerzálně použitelná, musíme uvažovat

i případy, kdy se vodivosti jednotlivých elementárních vrstev mohou lišit. Hlavní předpoklad, kdy toto může nastat, je použití dvojité klece nakrátko, kde běhová klec, umístěná blíže ke středu rotoru, může být z jiného materiálu než klec rozběhová. U takovýchto případů je nutné uvažovat, že se jedná o jednu složitou drážku.

$$\lambda'_{d2} = \frac{\sum_{i=1}^{n}[\lambda_i(\sum_{k=n}^{i}\frac{1}{r_k})^2]}{(\sum_{i=1}^{n}\frac{1}{r_i})^2} \qquad \textbf{1.20}$$

Činitel zmenšení magnetické vodivosti se vypočítá podílem magnetické vodivosti části drážky s uvažováním povrchového jevu ku magnetické vodivosti bez tohoto jevu.

$$k_x = \frac{\lambda'_{d2\xi}}{\lambda'_{d2}} \qquad \textbf{1.21}$$

Z výpočtu *1.13* si můžeme všimnout, že při výpočtu činitelů k_r a k_x má velkou roli rotorová frekvence potažmo skluz. Z toho plyne, že hodnoty k_r a k_x se musí počítat pro každou frekvenci zvlášť. To je důvodem, který prakticky nutí uživatele k využití výpočetní techniky.

2 MATLAB

Název počítavého programu MATLAB je složeninou z anglických slov *Matrix,* neboli matice*,* a *Laboratory,* neboli laboratoř. Jedná se o programové prostředí předně určené pro vědecké a technické výpočty. Jak už název napovídá, základní filozofií programu MATLAB, je práce s maticemi, což výrazně usnadňuje práci ve srovnání s jinými programovacími jazyk jako je C, C++ nebo Java. Programovací jazyk MATLAB umožnuje uživateli kvalitně zpracovat data, zobrazit je a navíc poskytuje nástroje pro měření v reálném čase, práci s matematikou, grafikou, přenosem dat a zpracování obrázků a videí.

Program MATLAB se skládá z těchto pěti částí:

- Výpočetní jádro
- Grafický subsystém
- Otevřená architektura
- Pracovní nástroje
- Toolboxy

2.1.1 Výpočetní jádro

Jak již bylo uvedeno, výpočetní jádro pracuje s maticemi, a to jak s maticemi reálných čísel, tak i s maticemi čísel komplexních. Mimo to umožnuje i běžné matematické operace jako je sčítaní a odečítaní, násobení a dělení a tak pod. Nabízí také nepřeberné množství funkcí pro práci s vektory a polynomy.

2.1.2 Grafický subsystém

Grafický subsystém dovoluje prezentovat získané výsledky v mnoha podobách jako například 2D a 3D grafy, histogramy, koláčové grafy atd. Navíc tento subsystém nabízí i tvorbu grafických objektů jako jsou tlačítka, posuvníky, rozbalovací menu, což je základním předpokladem využití MATLABU pro tvorbu vlastních grafických aplikací.

2.1.3 Otevřená architektura

Tato část MATLABU zaručuje neomezenou rozšiřitelnost programu. Uživatel si sám může naprogramovat funkce, které potřebuje a uložit je do souboru pro pozdější použití. Takto získané funkce se nijak neliší od funkcí vestavěných.

2.1.4 Pracovní nástroje

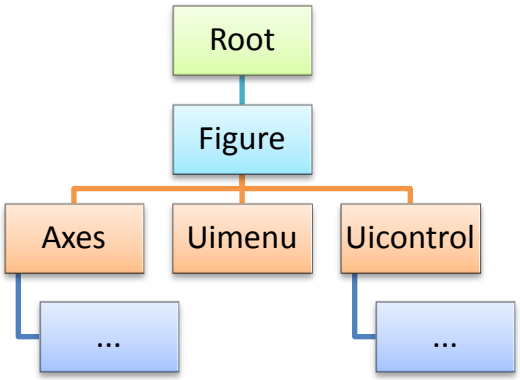
Mezi hlavní pracovní nástroje patří prohlížeč adresářů a souborů, okno historie příkazu, spouštěč aplikací, editor, debugger, příkazové okno a další. Všechny tyto části jsou do prostředí MATLAB Desktop plně integrovány a lze je uspořádat podle svého uvážení.

2.1.5 Toolboxy

Jsou to knihovny funkcí, které lze do MATLABU integrovat a které zajišťují jeho rozšiřitelnost pro daný vědní nebo technický obor. *Toolboxy* mimo potřebné funkce obsahují také detailní a přesnou dokumentaci.

2.2 Handles

Systém *Handles graphic* je součást programu MATLAB. Je určen k efektivnímu využití grafických objektů, které jsou v tomto systému obsažené. K pochopení práce s *handles* je nezbytně nutné znát hierarchii grafických objektů, jak je vidět na *Obr 2.1*, a jejich dědičnost, podobně jako u objektově orientovaného programování. Každý objekt má proto svoje unikátní *handles*.



Obrázek 2.1 - Hierarchie grafických objektů
Zdroj: Vlastní zpracování podle [6]

2.2.1 Dědičnost

U objektově orientovaného programování jsou objekty organizovány stromovým způsobem. Toto uspořádání umožňuje tzv. dědit vlastnosti z jednoho z druhů objektů do jiného a pouze přidat další své vlastní rozšíření. Objekt, který dědí vlastnosti se nazývá *dítě* (z angl. Child) a objekt, který své vlastnosti předává, se nazývá *rodič* (z angl. Parent).

Takto uspořádaný způsob programování nám usnadní práci v momentě, kdy např. chceme změnit font textu. Stačí nám změnit font pouze u objektu, který je *rodičem* a změna se projeví ve všech objektech, které jsou na něj vázány.

2.2.2 Root

Root je grafický objekt, který koresponduje s obrazovkou. Objekt *Root* je pouze jeden a nemá žádné tzv. *rodiče*. Je základním stavebním kamenem při práci s grafickými objekty. Tento objekt existuje již při spuštění MATLABU, proto ho nemusíme nikdy vytvářet a zároveň ho nelze odstranit. Pro práci s objektem *Root* můžeme využít dvou příkazů a to: *getappdata(0,'X')* pro získání vlastnosti *X* a příkazu *setappdata(0,'X',Y)* pro její nastavení.

Při vytváření grafické aplikace si mnohdy nevystačíme pouze s jedinou *Figure*. Proto je nutné zajistit předávání informací z *Figure 1* do *Figure 2* a to lze zajistit prakticky pouze přes *Root*.

2.2.3 Figure

Figure je grafický objekt v podobě okna, ve kterém je možné umístit všechny grafické objekty, které jsou potřebné pro správu funkcí grafické aplikace. Samozřejmostí také je, že do okna *Figure* lze umisťovat různé grafické výstupy jako je prostý text, grafy, obrázky atd.

Příkaz *figure* vytvoří nový objekt typu *Figure* za použití základních nastavení. Tento objekt se zároveň přesune nad všechna ostatní okna a jeho *handles* se umístí do proměnné *gcf*, tj. proměnné, která je pro *handles* aktivního objektu přímo rezervovaná.

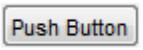
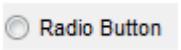
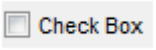
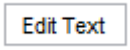
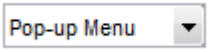
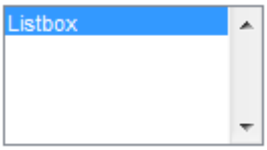
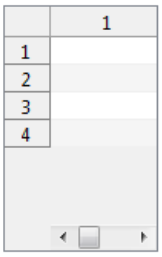
2.2.4 Axes

Axes je grafický objekt určený pro zobrazování grafických objektů jako jsou například grafy, čáry, obrázky atd. K zobrazení požadovaného grafu je nutné nejdříve určit v jakém objektu typu *Axes* se požadovaný výstup má zobrazovat. To se určí jednoduše pomocí *handles*. Jako příklad lze uvést zobrazení obrázku ze souboru.

```
axes(handles.axes1)
imshow('Obrazek.png')
```

2.2.5 Uicontrol

Prvky typu *Uicontrol* (pozn. User Interface Control) jsou základní grafické prvky pro ovládání grafického rozhraní aplikace. V tablce níže je konkrétní přehled těchto prvků.

Název prvku	Funkce	Náhled
Push Button	Tlačítko	
Slider	Posuvník	
Radio Button	Výběrové pole	
Check Box	Zaškrtávací pole	
Edit Text	Pole pro editaci textu	
Static Text	Zobrazení textu	
Pop-up Menu	Rozbalovací nabídka	
Listbox	Nabídka	
Table	Tabulka	
Frame	Rámeček	

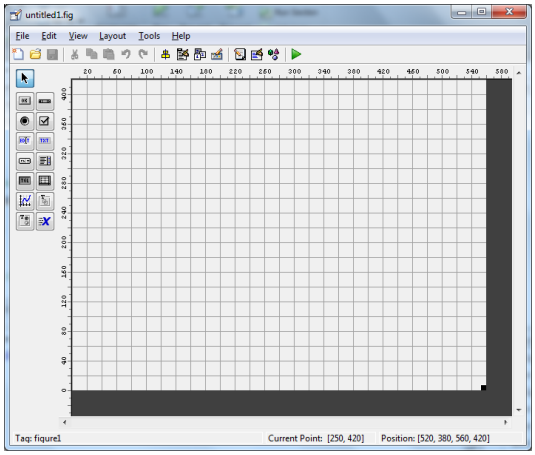
Tabulka 1 - Přehled prvků Uicontrol
Zdroj: Vlastní zpracování

2.2.6 GUIDE

V dnešní době je většina aplikací vytvářena ve formě grafických uživatelských rozhraní. Tyto aplikace tedy obsahuji grafické ovládací prvky, které jsou popsané výše. Tyto prvky jsou jedinou vazbou mezi programem a uživatelem. Z tohoto důvodu by měly být v GUI (pozn. Grafické uživatelské rozhraní, z angl. Graphical User Interface) prvky uspořádány přehledně a zároveň intuitivně.

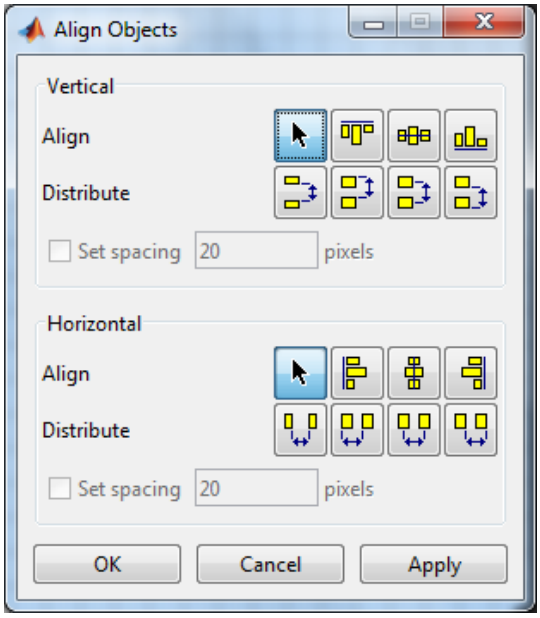
Při tvorbě GUI lze postupovat dvěma způsoby. První je více časově náročný a na programátorovi je vyžadována znalost každého grafického objektu do detailu. Princip prvního způsobu spočívá v tom, že se musí každý prvek naprogramovat samostatně včetně jeho vlastností pomocí příkazů *set*. Takto získaný kód je přehledný a lze ho spustit jednoduchým zkopírováním do editoru. Druhou metodou je využití nástroje GUIDE (pozn. Graphical User Interface Development Environment), který je implementován do MATLABU. Vytvoření aplikace pomocí GUIDE je založeno na jednoduchém *Drag&Drop* přetahovaní Uicontrol objektů z nabídky do okna *Figure*. Tento nástroj sám vytvoří zdrojový kód potřebný ke spuštění aplikace a vytvoření všech grafických objektů. Takto získaný kód není úplně optimální, jelikož obsahuje množství nepotřebných částí. Automaticky vygenerovaný kód je uložen v souboru *Nazev_souboru.fig*. Zároveň se souborem *Nazev_souboru.fig* se vytváří i soubor *Nazev_souboru.m,* do kterého se zapisují funkce aplikace. Výhodou této metody je menší časová náročnost a větší přehlednost při tvorbě grafického rozhraní. Na druhou stranu, aby aplikace fungovala i na jiném počítači, musí být k dispozici zmiňované soubory *Nazev_souboru.fig a Nazev_souboru.m.*

Na Obr. 2. 2 je vyobrazeno okno editoru GUIDE, které se objeví po zadání příkazu *guide* do příkazového řádku.

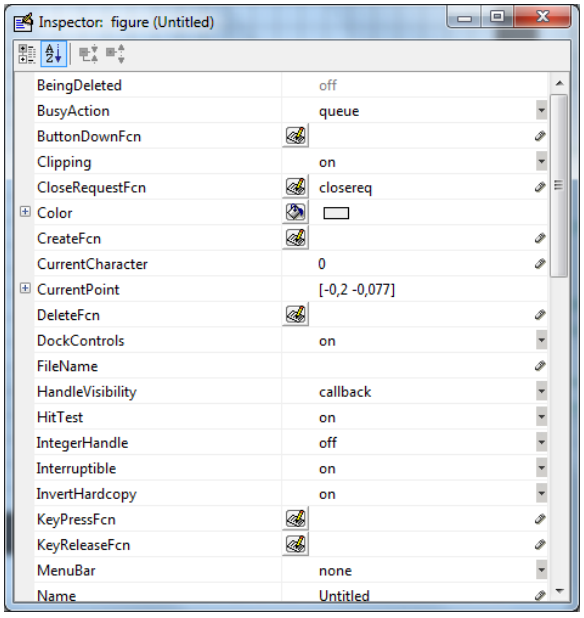


Obrázek 2.2 - Prostředí editoru GUIDE
Zdroj: [6]

Pro efektivní práci s grafickými objekty v editoru GUIDE, jsou k dispozici dva nástroje a to nástroj *Align Objects Obr. 2.3* a nástroj *Property Inspector Obr. 2.4*



Obrázek 2.3 Nástroj Align Objects
Zdroj:[6]



Obrázek 2.4 Nástroj Property Inspector
Zdroj:[6]

Při tvorbě grafického rozhraní je nutné dbát i na vzhled, proto je třeba dbát i na zarovnání všech objektů. K tomu slouží nástroj *Align Objects*, který umožnuje po vybrání referenčního objektu, zarovnat zbylé objekty. Zarovnání je možné provést jak ve vertikální ose tak i v horizontální. Tento nástroj nabízí zarovnaní nejen vlevo, vpravo nebo na střed, ale taky zarovnání s určitým rozestupem, kde si můžeme vybrat i osy mezi rozestupy.

Nástroj *Property Inspector* umožnuje nastavení vlastností daného objektu. Všechny vlastností jsou zde seřazeny do přehledné tabulky, kde si můžeme jednoduše změnit barvu pozadí, font a velikost písma atd. Mimo to, v této tabulce najdeme i vlastnost *Tag.* Tato vlastnost určuje *handles* daného objektu. *Tag* daného objektu se vygeneruje automaticky s pořadovým číslem při vytvoření. Pro lepší přehlednost si jej můžeme libovolně změnit, musíme ale mít na paměti, že musí být unikátní. Pozdější změna této vlastnosti není doporučována.

2.2.7 Zpětnovazební funkce Callback

Po vytvoření návrhu grafického rozhraní zjistíme, že ovládací prvky neplní žádnou funkci. Z tohoto důvodu je nutné pro dané objekty vytvořit zpětnovazební funkci *Callback*. Z názvu funkce je vidět, že jsou tyto funkce volány, až momentě kdy uživatel provede akci. Jaké akce se provedou po zavolání těchto funkcí, určuje programátor zápisem příkazů do daných funkcí. Ty jsou uloženy v souboru *Nazev_souboru.m. CallBack* funkce se nijak neliší od funkcí klasických, i když jsou tyto funkce generované automaticky editorem GUIDE. V případě potřeby vytvoření další zpětnovazební funkce pro daný objekt, ji lze dodatečně vygenerovat pomocí kontextové nabídky *View Callbacks*.

V následujícím seznamu jsou popsaný všechny zpětnovazební funkce.

Název funkce	Popis funkce
CreateFcn	Funkce je volána při startu aplikace
DeleteFcn	Funkce je volána při smazání okna
CloseRequestFcn	Funkce je volána před zavřením aplikace
KeyPressFcn	Funkce je volána při stisku tlačítka, pokud je kurzor v okně aplikace
ResizeFcn	Funkce je volána při změně velikostí okna
WindowButtonDownFcn	Funkce je volána při kliknutí myší na neaktivní prvky např. obrázek
WindowButtonMotionFcn	Funkce je volána pohybu myší s držením tlačítka myši
WindowButtonUpFcn	Funkce je volána při uvolnění tlačítka myši nad neaktivním prvkem např. obrázkem

Tabulka 2 - Seznám zpětnovazebných funkcí
Zdroj: [6]

3 Aplikace pro výpočet rozběhových charakteristik

3.1 Vlastní návrh aplikace

Úkolem aplikace je simulování rozběhových charakteristik asynchronního stroje. Uživatel si bude moci zobrazit průběh momentu, výkonu, proudů a účiníku. Zároveň uživateli bude umožněno zadat tvar drážky rotorové tyče pro výpočet činitelů k_r a k_x.

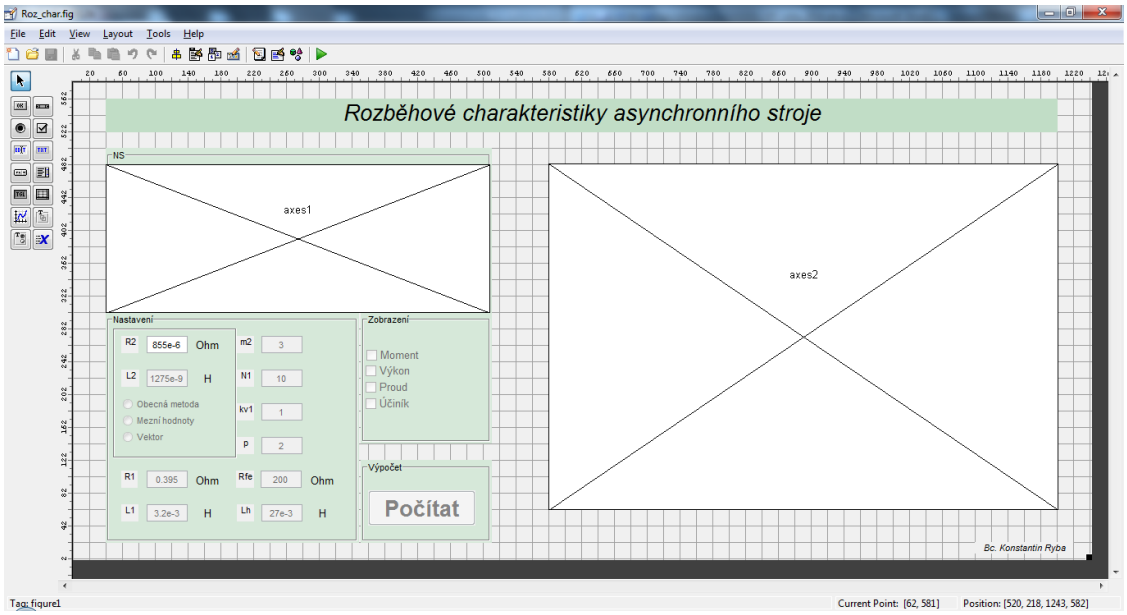
Po stanovení cíle je vhodné si rozmyslet vzhled aplikace. Je zapotřebí si uvědomit potřebný počet ovládacích, zobrazovacích a sdělovacích prvků. Před samotnou tvorbou aplikace byly připraveny i potřebné soubory, v našem případě se jedná především o obrázek náhradního schéma asynchronního stroje.

V editoru GUIDE byly rozmístěny grafické objekty a zároveň uspořádány do logických celků pomocí panelů. Celkově v aplikaci lze nalézt tři panely, a to panel s objekty potřebnými pro zadání základních parametrů náhradního schéma, panel s volbou zobrazení požadované charakteristiky a panel výpočet. Mimo jiné se v aplikaci nalézají objekty typu *axes*, které jsou určené pro zobrazení náhradního schéma a zobrazení rozběhových charakteristik.

Jelikož aplikace obsahuje i tři metody zadávání rotorových prvku, bylo nutné vytvořit i volbu metody pomocí *radio button.* Volba zobrazení požadované charakteristiky je realizována pomocí čtyř objektů typu *check button*. Pomocí již dříve popsaného nástroje *Align Objects, byly všechny prvky zarovnány*.

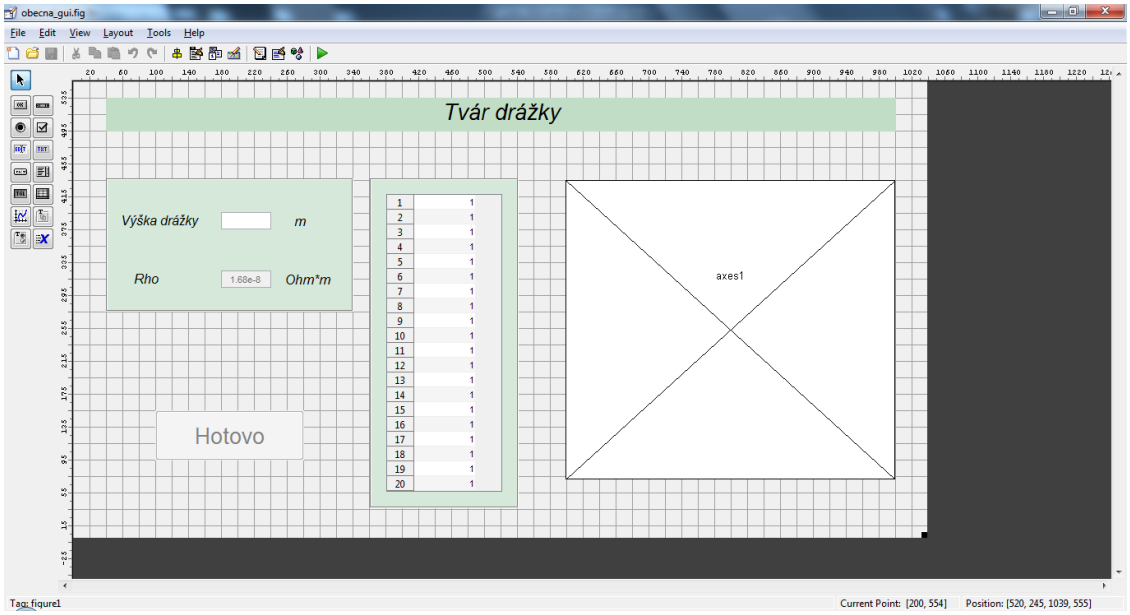
Když už jsou prvky na svých místech, použitím nástroje *Property Inspector,* se změní vlastnosti těchto prvků. Konkrétně se jednalo o změny barvy pozadí, změny velikostí a stylů textů, nastavení počátečních hodnot a v neposlední řadě také změny vlastností *Enable*, kde všechny prvky byly nastaveny na hodnotu *off.* Toto opatření bylo zavedeno, jak si dále ukážeme, z důvodu usnadnění orientace v aplikaci a hlavně, aby byl jednoznačně určen postup při zadávání hodnot.

Na obrázku Obr. 3.1 je snímek obrazovky po provedení popsaných změn. Zároveň toto grafické okno bylo uloženo pod názvem *Roz_char.fig*.

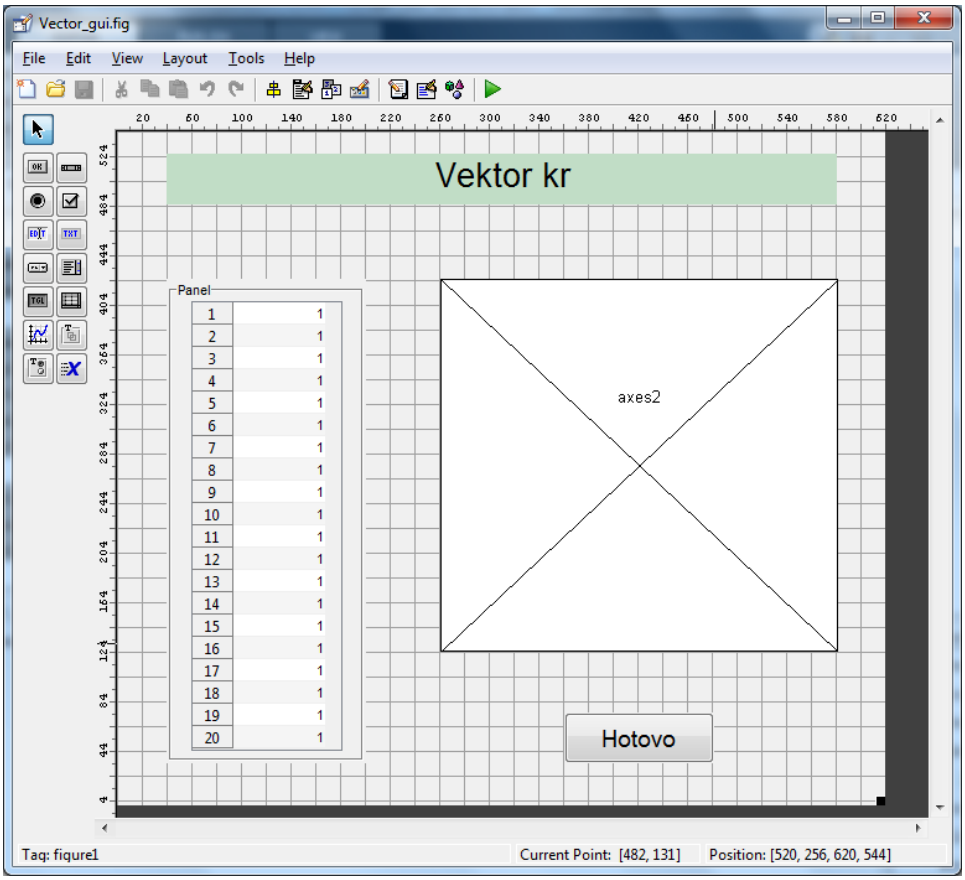


Obrázek 3.1 - Náhled navrhnutého grafického rozhraní
Zdroj: Vlastní zpracování

Pro potřebu zadávání tvaru drážky bylo vytvořeno další grafické okno pod názvem *obecna_gui.fig Obr. 3.2.* Taktéž bylo vytvořeno grafické rozhraní *Vector_gui.fig Obr. 3.3*, pro potřebu zadávání vektoru k_r. Tyto dvě *Figures* byly vytvořeny stejným způsobem, jako je popsáno výše.



Obrázek 3.2 - Náhled rozhraní obecna_gui
Zdroj: Vlastní zpracování



Obrázek 3.3 - Náhled rozhraní Vector_gui
Zdroj: Vlastní zpracování

Jelikož všechna tři okna jsou navzájem nezávislá, bylo zapotřebí vyřešit předávání dat mezi nimi. K tomuto účelu bylo využito objektu typu *root,* kde pomocí zpětnovazebních funkcí byla data ukládána. Následně tato data byla načítaná v případě potřeby. Nevýhodou tohoto řešení je poměrná nepřehlednost, jelikož takto uložená data se nezobrazují v seznamu *Workspace*. Bylo třeba klást důraz na pojmenování vlastností, které byly vytvářeny v *handles root*. Zároveň při vývoji aplikace byly tyto *handles* evidovány v externím seznamu.

Další nepříjemností s použitím tohoto způsobu předávání dat je fakt, že MATLAB nečeká, než se volaná funkce splní a rovnou pokračuje dle zadaného programu dále. Tento problém byl vyřešen pomocí příkazů *uiwait(x) a uiresume(gcf).* Příkaz *uiwait* se vloží do funkce, která volá další funkci a způsobí, že program se zastaví a čeká, dokud nepřijde příkaz *uiresume*, který byl vložen do volané funkce. Tato metoda byla využita, k získání dat z rozhraní *Vector_gui a obecna_gui* , pro případ že uživatel chce zadat tvar drážky nebo vektor sám.

Protože je uživateli umožněno si vybrat metodu zadávání parametrů, byla vytvořena funkce *jeden_vyber*. Kód této funkce je uveden níže. Úkolem této funkce je zajistit, že si uživatel vybral pouze jednu metodu zadávání parametrů. Zároveň byla tato funkce použita i k zamezení více výběrů pro zobrazení. Tato funkce je volána při změně vlastnosti *Value* jednoho z uvedených prvků. Jako vstupní proměna *off* se zadává *handles* zbylých objektů, kterým se tato vlastnost změní automaticky na hodnotu *0.*

```
function jeden_vyber (off)
set(off,'Value',0)
```

Obdobou k funkci *jeden_vyber* je funkce *odblokovat.* Tato funkce má za úkol aktivovat objekt, jehož *handles* je uloženo v proměnné *off.* Touto cestou je řešené postupné aktivování všech objektů, jelikož byly implicitně nastaveny jako inaktivní.

```
function odblokovat (off)
set(off,'Enable','on')
```

K zadávání vektoru změny rotorového odporu v *Figure Vector_gui* , byl využit objekt typu *uitable* o velikosti 20x1. Tato velikost byla zvolena jako kompromis mezi počtem zadaných hodnot a časovou náročností zadávání. Uživatel si zde může zadat jakákoliv bezrozměrná čísla a upravovat je tak, aby výsledný průběh odpovídal jeho představě. Aby tato čísla byla poměrná, je každé z daných čísel vyděleno minimální hodnotou obsaženou v této tabulce. Po přepočítání těchto čísel je automaticky vykreslován průběh k_r v závislosti na skluzu.

Pro pozdější výpočet bylo třeba upravit zadané hodnoty z tabulky interpolací. Tato úprava byla nutná kvůli zachování stejných velikostí vektorů. Program MATLAB obsahuje funkci *spline,* která byla použita k interpolaci. Jedná se o přibližný výpočet hodnot ležících v určitém intervalu, které se počítají na základě hodnot v krajích tohoto intervalu.

Pro ucelený náhled na vektor k_r je do objektu typu *axes* vykreslován průběh jak zadaných hodnot ve formě bodu, tak i průběh po interpolaci. Po stisku tlačítka HOTOVO dojde k uložení upravených hodnot do *root* a zavolaní příkazu *uiresume*, který způsobí opětovný rozběh programu, tak jak už bylo výše popsáno.

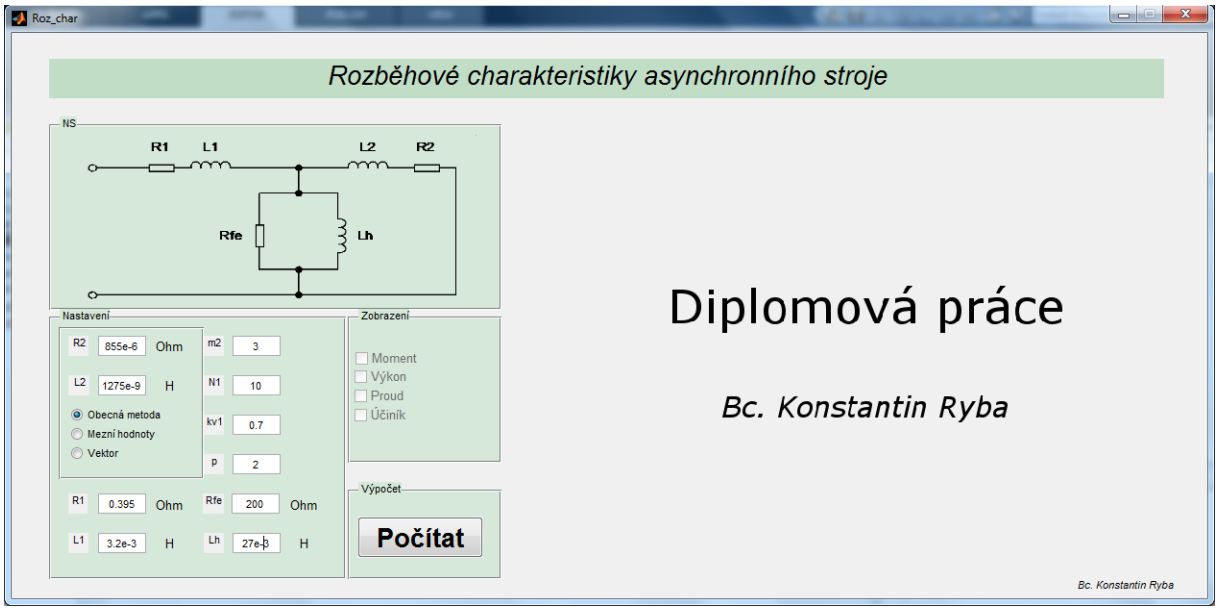
V *Figure obecna_gui* je zadávaní rozměrů drážky provedeno obdobným způsobem. Výška drážky se zadává v metrech do objektu *edit text* a šířka v milimetrech do objektu typu *uitable* o velikosti 20x1. Po potvrzení změny hodnoty buňky dochází k přepočtení hodnot na metry a zároveň k vykreslení v podobě grafu.

Vykreslení je řešeno dělbou šířky jednotlivé elementární obdélníkové drážky dvěma. Takto upravená šířka se vykresluje dvěma grafy v jednom okně, kde jeden je násoben *-1.* Jelikož tyto dva grafy jsou souměrné podél osy Y, lze tímto způsobem vykreslit pouze souměrné drážky.

Obě tyto *Figures* si uchovávají zadané hodnoty pro pozdější editaci. Tato vlastnost je zajištěna načtením uložených hodnot v *Root* při spuštění. Při prvním spuštění se načítají základní data, která byla zadána v GUIDE editoru.

3.2 Použití aplikace

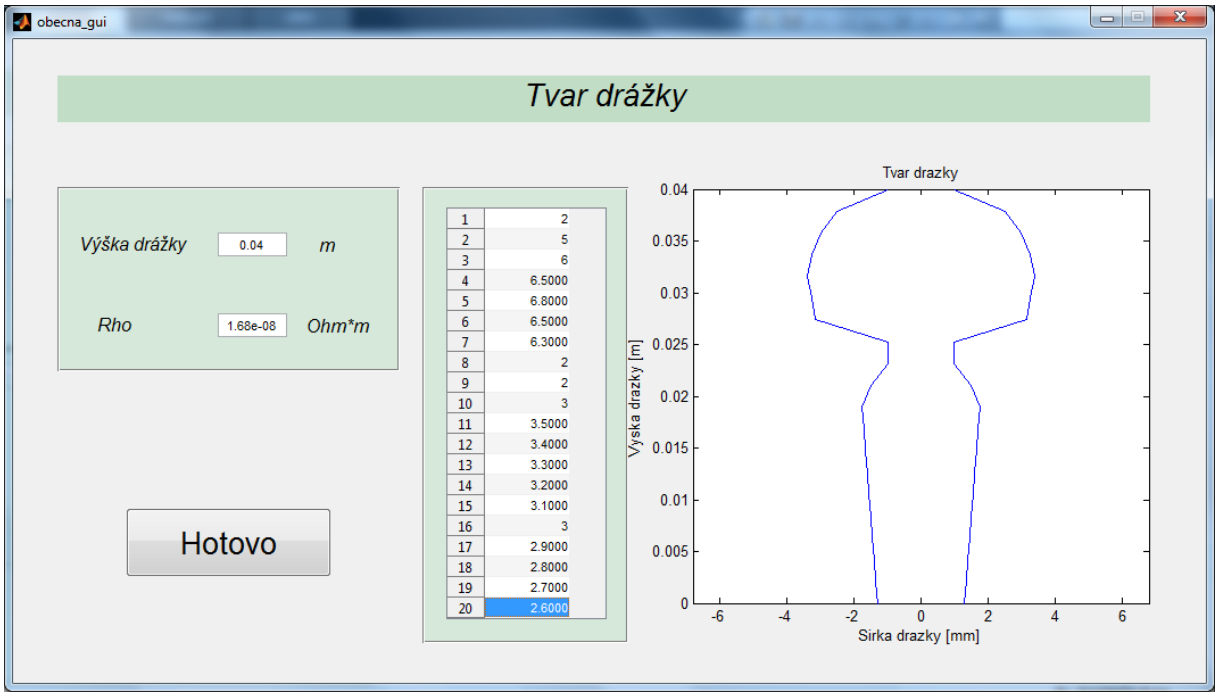
V následující kapitole je popsán postup použití aplikace v praxi. Po spuštění programu MATLAB vybere uživatel soubor *Roz_char.m* a spustí tento skript tlačítkem *Run*. Otevře se okno s grafickým rozhraním, *Obr. 3.4*. V tuto chvíli jsou všechny objekty inaktivní a postupným zadáním hodnot se aktivují. Není tedy možné spustit tlačítko *Počítat*, dokud nebudou vyplněny všechny požadované hodnoty a vybrána metoda výpočtu. Toto řešení bylo zvoleno kvůli robustnosti aplikace. Výběr možností zobrazení dat se aktivuje až po výpočtu.



Obrázek 3.4 - Grafické rozhraní aplikace
Zdroj: Vlastní zpracování

Po stisknutí tlačítka *Počítat*, pokud byla vybrána obecná metoda výpočtu nebo vektor, se spustí příslušné grafické okno. V případě výběru metody mezní hodnoty se tento krok vynechá a automaticky proběhne výpočet na pozadí. V tomto popisu práce s aplikací je vybrána obecná metoda výpočtu, jelikož je nejsložitější. V nově spuštěném okně, *Obr. 3.5,* je třeba zadat hodnoty a to nejprve výšku drážky a měrný elektrický odpor. Následně se aktivuje tabulka, do které se zadá dvacet hodnot šířky drážky. Jedná-li se tedy o drážku tvaru obdélníku, bude všech dvacet hodnot shodných.

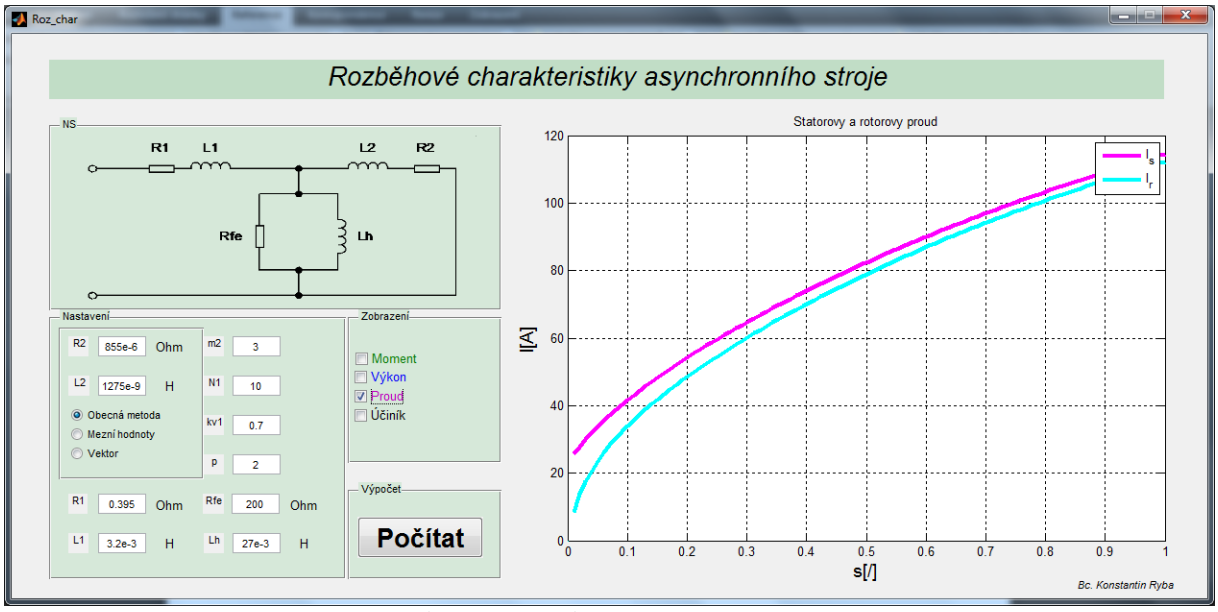
V pravé části okna se pro kontrolu vykresluje průřez drážky. Po zadání všech hodnot se zaktivní tlačítko *Hotovo* a je možné potvrdit výběr.



Obrázek 3.5 - Grafické rozhraní - obecná metoda
Zdroj: Vlastní zpracování

Po stisknutí tlačítka *Hotovo*, se toto okno zavře, uživatel se automaticky vrátí do původního okna, ale nyní už budou aktivní možnosti výběru zobrazení dat. Při výběru jednotlivých metod zobrazení se v pravé části okna automaticky vykreslí příslušné grafy, *Obr. 3.6,* tedy momentová charakteristika, výkonová charakteristika, průběhy proudů anebo průběh účiníku.

Výhodou této aplikace je fakt, že po zadání hodnot a proběhlém výpočtu je možné změnit jeden z parametrů a výpočet opakovat bez nutnosti znovu zadávání všech hodnot. To může práci poměrně urychlit.



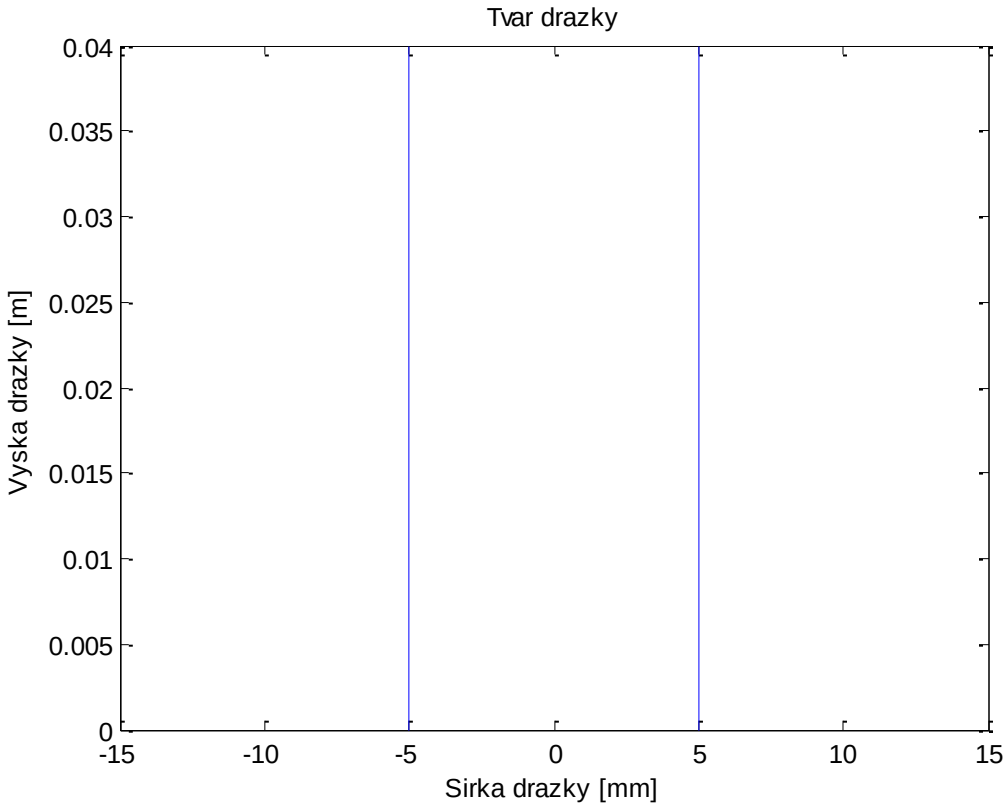
Obrázek 3.6 - Grafické rozhraní - průběhy proudů
Zdroj: Vlastní zpracování.

Na závěr této kapitoly by bylo vhodné zjistit, zda opravdu má tvar drážky vliv na výkon a moment asynchronního stroje a především zda nám to pomůže zjistit vytvořená aplikace.

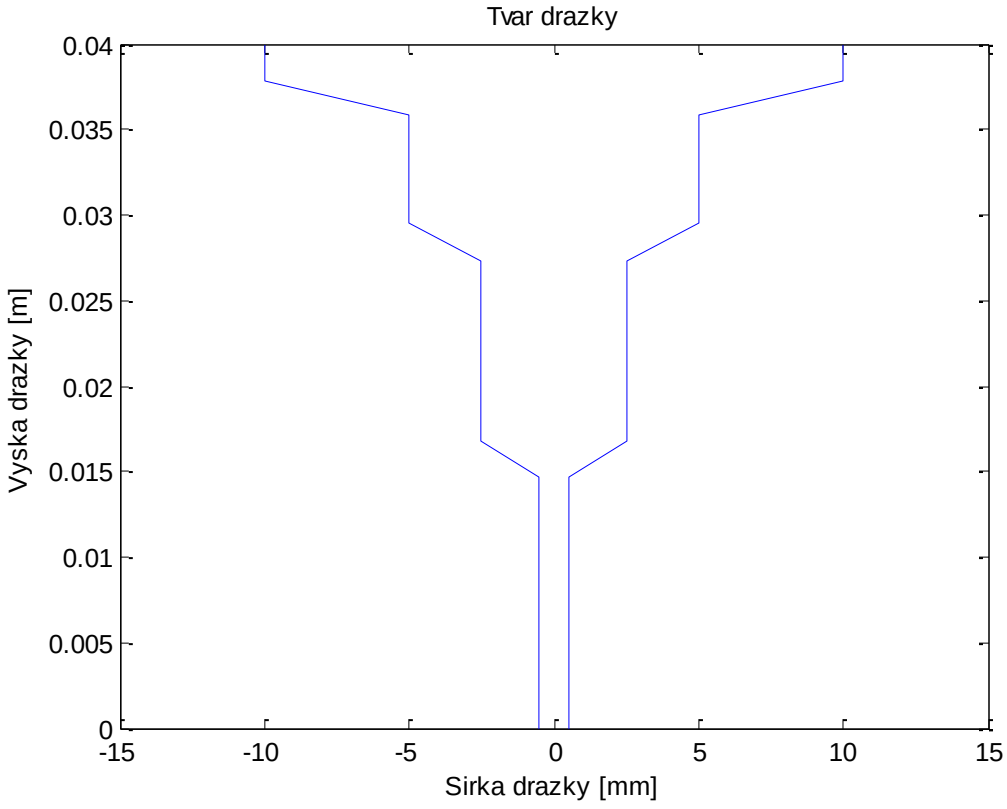
Pro porovnání použijeme dva typy drážek a to tvaru obdélníka a drážky smyšlené, kde průřez připomíná tvar trychtýře. Tato drážka se v praxi nepoužívá, ale byla vybrána pro zjištění vlivu na charakteristiky stroje. Výpočet probíhal se stejnými parametry, výška i materiál byly zachovány a měnil se pouze tvar drážky, tedy hodnoty v druhém okně aplikace.

Tvar, respektive průřez drážkami je znázorněn na obrázcích *Obr. 3.7 a Obr. 3.8.* Po výpočtech bylo možné zobrazit grafické znázornění charakteristik asynchronního stroje s použitím těchto drážek. Z obrázků *Obr. 3.9 a Obr. 3.10* je vidět vliv na tvaru drážky na momentovou charakteristiku. U obdélníkového tvaru drážky je záběrný moment větší o 70 Nm než u asynchronního motoru s drážkou druhého typu při zachovávání stejného maximálního momentu.

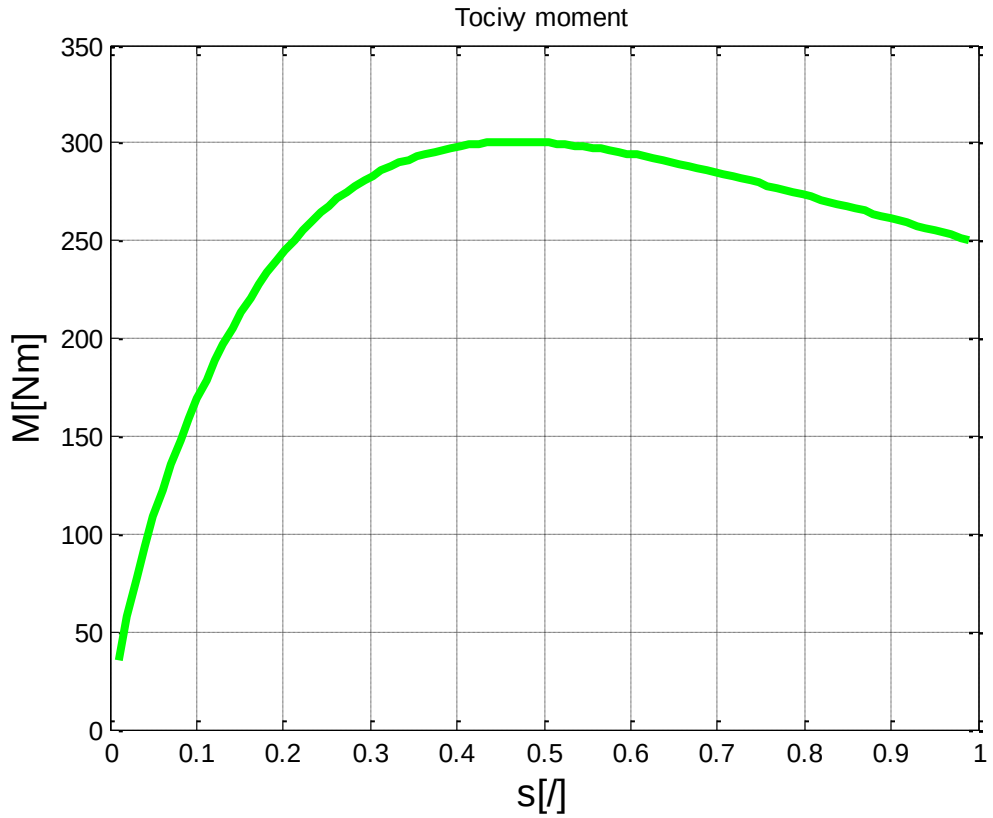
Na obrázcích *Obr. 3.11 a Obr 3.12* je vidět i mírná změna výkonu při změně tvaru drážky.



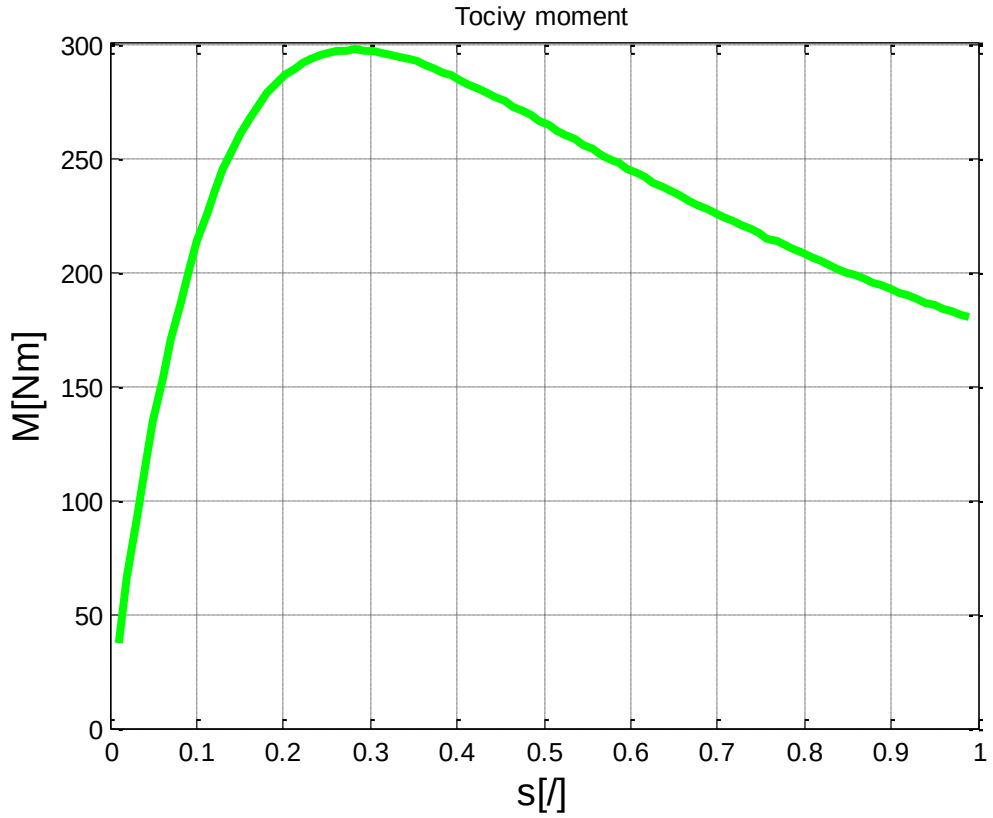
Obrázek 3.7 - Průřez (Obdélníková drážka)
Zdroj: Vlastní zpracování



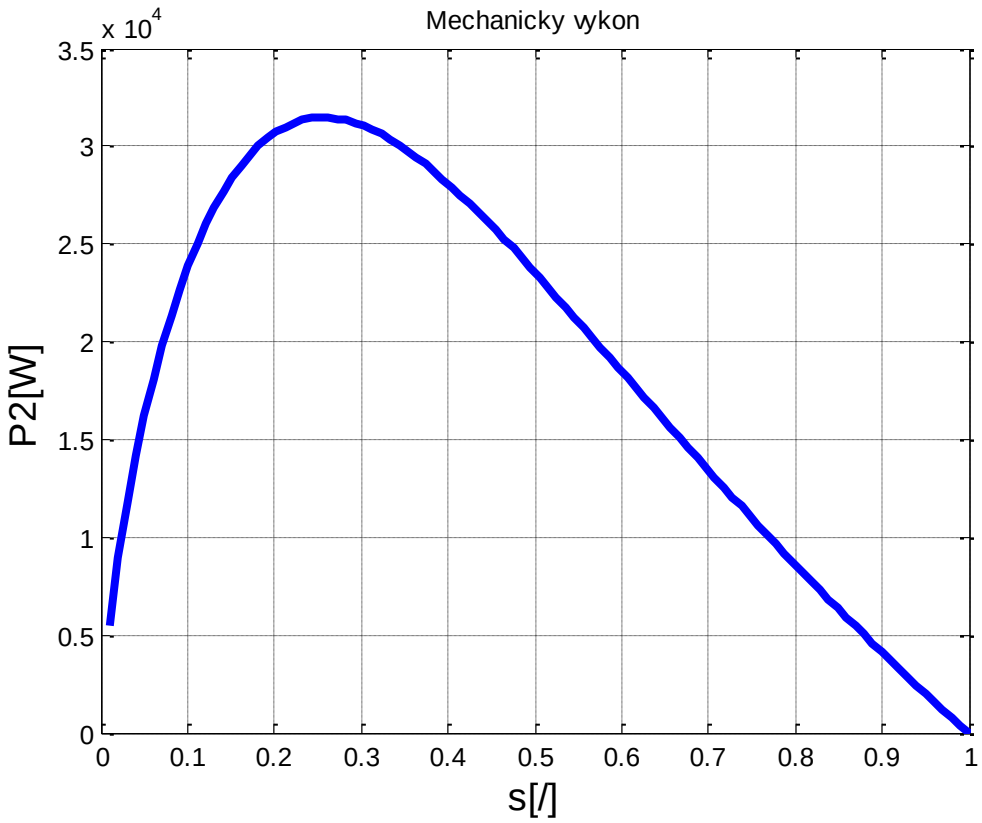
Obrázek 3.8 - Průřez (Trychtýřovitá drážka)
Zdroj: Vlastní zpracování



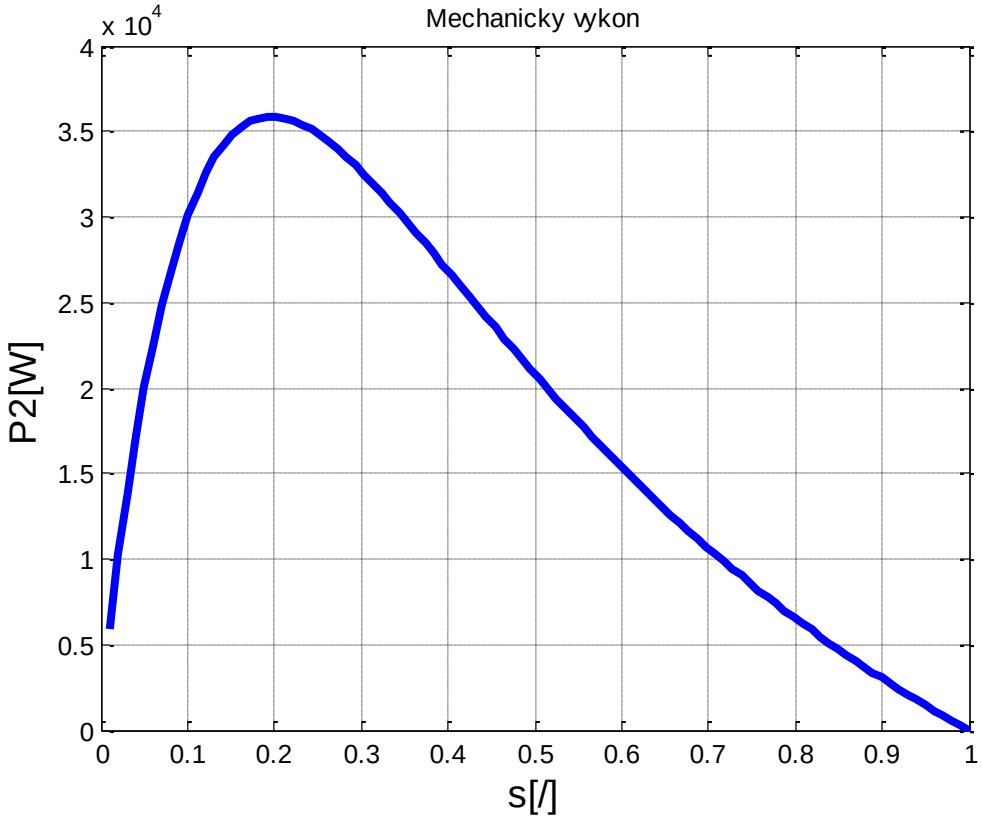
Obrázek 3.9 - Průběh momentu (Obdélníková drážka)
Zdroj: Vlastní zpracování.



Obrázek 3.10 - Průběh momentu (Trychtýřovitá drážka)
Zdroj: Vlastní zpracování.



Obrázek 3.11 - Průběh výkonu (Obdélníková drážka)
Zdroj: Vlastní zpracování.



Obrázek 3.12 - Průběh výkonu (Trychtýřovitá drážka)
Zdroj: Vlastní zpracování.

Závěr

V úvodu této práce byla stanovena hypotéza, že tvar rotorové drážky má vliv na celkové charakteristiky, tedy především na výkonovou a momentovou charakteristiku, asynchronního stroje.

Princip fungování asynchronního stroje, náhradní schéma i konstrukční řešení již byly vysvětleny. Hlavním cílem práce bylo vytvoření programu, který by uživateli pomohl při konstrukci asynchronního stroje. Vzhledem k tomu, že program byl vytvořen v programu MATLAB, je na uživateli požadována alespoň jeho základní znalost. Proto práce obsahuje i vysvětlení prvků potřebných pro použití této aplikace a přesný popis postupu k získání požadovaných výstupů.

Při vývoji programu došlo k několika zjednodušením z důvodu usnadnění používání programu pro uživatele. Jedná se především o měrný elektrický odpor rotorové tyče, kde v programu lze zadat jeho hodnotu pouze pro celý průřez drážky. Toto zjednodušení znemožňuje použití programu pro výpočet rozběhových charakteristik dvojité klece na krátko s různými materiály rotorových tyčí. Dalším zjednodušení je, že při zadávání vektoru $k_{r,}$ se při výpočtu zanedbává vliv činitelé $k_{x.}$

V práci byly popsány dvě různé metody výpočtu rozběhových charakteristik, přičemž jako ta, která byla vhodná pro použití v aplikaci, byla vybrána obecná metoda výpočtu vlivu povrchového jevu v rotorových tyčích libovolného tvaru. Tato metoda je popsána v [3] a byla vybrána pro použití aplikace, jelikož splňuje její hlavní předpoklad a to právě libovolný tvar drážky.

V závěru práce byly vybrány dva tvary drážky a to obdélníkový a tzv. trychtýřovitý tvar. Po výpočtu pomocí aplikace se tak potvrdila hypotéza, že změna tvaru této drážky ovlivňuje v nemalé míře záběrný moment asynchronního stroje a tím pádem také ostatní rozběhové charakteristiky, např. graficky znázorněný průběh točivého momentu a průběh mechanického výkonu.

Seznam literatury a informačních zdrojů

[1]. Kohout, J. Metody matematického modelování. Plzeň : ZČU V Plzni, 2007.

[2]. Bartoš, V. Elektrické stroje I,II. Plzeň : VŠSE v Plzni - ediční středisko, 1986.

[3]. Kopylov, I.P. a kol. Stavba elektrických stroj. Praha : SNTL, 1988.

[4]. Šimek, J. Příprava výukového materiálu pro tvorbu GUI v MATLABu. České Budějovice Jihočeská univerzita v Českých Budějovicích, 2012.

[5]. Kocman, S. Asynchronní stroje. Ostrava : FEI VŠV-TU, 2002.

[6]. MATLAB. Napověda programu MATLAB - MATLAB Help.

Příloha A

Úplný výpis aplikace

Roz_char.m

```
% ------------------------------------------------------------------
----- %
%|                              Diplomova Prace
|%
%|               Rozbehove charakteristiky asynchronniho stroje
|%
%|                            Bc. Konstantin Ryba
|%
%|                                 2014
|%
% ------------------------------------------------------------------
----- %

function varargout = Roz_char(varargin)
% ROZ_CHAR MATLAB code for Roz_char.fig
%      ROZ_CHAR, by itself, creates a new ROZ_CHAR or raises the
existing
%      singleton*.
%
%      H = ROZ_CHAR returns the handle to a new ROZ_CHAR or the
handle to
%      the existing singleton*.
%
%      ROZ_CHAR('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in ROZ_CHAR.M with the given input
arguments.
%
%      ROZ_CHAR('Property','Value',...) creates a new ROZ_CHAR or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Roz_char_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Roz_char_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Roz_char

% Last Modified by GUIDE v2.5 28-Mar-2014 19:18:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Roz_char_OpeningFcn, ...
                   'gui_OutputFcn',  @Roz_char_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
function varargout = Roz_char_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
function Roz_char_OpeningFcn(hObject, eventdata, handles, varargin)
    handles.output = hObject;

    setappdata(0,'f',50);                    %Nastaveni f
    setappdata(0,'m1',3);                    %Nastaveni m
    setappdata(0,'U',230);                   %Nastaveni U
    a=ones(20,1);
    setappdata(0,'Data',a);
    setappdata(0,'Data_vec',a);
    setappdata(0,'vyska',0);
    setappdata(0,'Rho',1.68e-8);
    axes(handles.axes1)                      %Aktivace axes1
    imshow('schema.png')                     %Nacteni obrazku schema
    axes(handles.axes2)                      %Aktivace axes1
    imshow('Titulka.png')                     %Nacteni obrazku schema

    guidata(hObject, handles);


%-------------------------------------------------------------------
-------

function editR2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editLh_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editL2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editRfe_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editm2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editN1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editkv1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editR1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editL1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editp_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%-----------------------------uzivatelske funkce----------------------
-------

function jeden_vyber (off)
set(off,'Value',0)                          %Nastaveni par Value na 0

function odblokovat (off)
set(off,'Enable','on')                      %Nastaveni par Enable na 0

function s_linear
    vektor_slin=linspace(0,1,100);              %Vytvoreni vektoru
od 0 do 1
    setappdata(0, 'vektor_slin', vektor_slin);  %Ulozeni vektoru do
root

function vypocet
    f      =  getappdata(0,'f');            %Nacteni hodnot
    p      =  getappdata(0,'p');
    m1     =  getappdata(0,'m1');
    m2     =  getappdata(0,'m2');
    U      =  getappdata(0,'U');
    R1     =  getappdata(0,'R1');
    R2     =  getappdata(0,'R2');
    Rfe    =  getappdata(0,'Rfe');
    L1     =  getappdata(0,'L1');
    L2     =  getappdata(0,'L2');
    Lh     =  getappdata(0,'Lh');
    Rho    =  getappdata(0,'Rho');
    s_vec  =  getappdata(0,'vektor_s');
    s_lin  =  getappdata(0,'vektor_slin');
    ai     =  getappdata(0,'drazka_X');
    bi     =  getappdata(0,'drazka_Y');
    kv1    =  getappdata(0,'kv1');
    N1     =  getappdata(0,'N1');
    roz    =  getappdata(0,'roz');
    %-------------------------------------------------------------

    ws    =  2*pi*f;                        %Synchronni uhlova rychlost
    X1    =  ws*L1;                         %Reaktance statoru
    X2    =  ws*L2;                         %Reaktance rotoru
    Xu    =  ws*Lh;                         %Podelna reaktance
    ku    =  2*N1*kv1;                      %Napetovy prevod
    kz    =  (m1/m2)*ku^2;                  %Prevod impedanci
    R21   =  R2*kz;                         %Prevedeny odpor rotoru na
stator
    X21   =  X2*kz;                         %Prevedena reaktance rotoru
na stator
```

```
    if roz==1
        R21=  R21*s_vec;                %Rotorovy odpor
        s  =  s_lin;                    %Vektor skluzu
    elseif roz==2
        s  =  s_lin;                    %Vektor skluzu
    else
        vypocet_skinefekt               %Volani fce pro vypocet kr a
kx
        kR = getappdata(0,'kR');        %Prevzeti hodnot
        kX = getappdata(0,'kX');
        R21  =  R21.*kR;                %Rotorovy odpor
        X21  =  X21.*kX;                %Rotorova reaktance
        s  =  s_lin;                    %Vektor skluzu
    end
    Z1h   = (j*Xu*Rfe)/(Rfe+j*Xu);      %Pricna impedance
    Z1    =  R1+j*X1;                   %Statorova impedance
    Z2    =  R21./s+j*X21;              %Rotorova impedance
    Zi    =  Z1h.*Z2./(Z1h+Z2);         %Impedance pro vypocet
indukovaneho napeti
    Zc    =  Z1+Zi;                     %Celkova impedance
    Ui    =  U.*Zi./(Zi+Z1);            %Nap. delic  pro
induk.napeti
    I1    =  U./Zc;                     %Statorovy proud
    I2    =  abs(Ui./Z2);               %Rotorivy proud
    P2    =  m1.*R21.*(1-s)./s.*I2.^2;  %Vykon
    M     =  (m1.*R21.*(1-s)./s.*I2.^2)./(ws/p.*(1-s));%Moment
    cos_fi= real(I1)./abs(I1);          %Ucinik

    setappdata(0,'I1',I1);              %Ulozeni hodnot do root
    setappdata(0,'I2',I2);
    setappdata(0,'P2',P2);
    setappdata(0,'M',M);
    setappdata(0,'cos_fi',cos_fi);

function vypocet_skinefekt
    f     =  getappdata(0,'f');             %Nacteni hodnot
    Rho   =  getappdata(0,'Rho');
    ai    =  getappdata(0,'drazka_X');
    bi    =  getappdata(0,'drazka_Y');

    ws=2*pi*f;                              %Synchronni uhlova
rychlost
    mi=4*pi*10^-7;                          %Permiabilita vakua
    lami=ai./bi;                            %Geometricka vodivost
magneticke trubice
    si=ai.*bi;                              %Prurez i-te vrstvy
    vel=20;                                 %Velikost vektoru
    len=100;                                %velikost vystupnich
cinitelu
    %-------------------------------------------------------------
-------
    n=0;
    for a=f/len:f/len:f

        for i=1:vel
```

```
            xi(i)=2*pi*a*mi*lami(i);          %reaktance i-te vrstvy
            ri(i)=Rho/si(i);                  %Odpor i-te vrstvy
        end
        I=zeros(vel,1);                       %Vektor pro Ik
        I(vel,1)=1;
        suma_I=0;
        for j=vel-1:-1:1                      %Vypocet elementarnich
proudu
            suma_I=suma_I+I(j+1,1);
            I(j,1)=I(j+1,1)*ri(j+1)/ri(j)+i*(xi(j+1)/ri(j)*suma_I);
        end
        It=sum(I);                            %Celkovy proud tyce
        suma_rI=0;                            %Vypocet odporu tyce s
uvazovanim povrchoveho jevu
        for j=1:vel
            suma_rI=suma_rI+(ri(j)*abs(I(j)^2));
        end
        rt_xi=suma_rI/abs(It)^2;
        rt1=0;                                %vypocet odporu tyce pri
rovnomernem rozlozeni proudu
        for j=1:vel
            rt1=rt1+1/ri(j);
        end
        rt=1/rt1;
        suma_xI=0;                            %Vypocet cinitele mag
vodivosti drazky s uvazovanim povrchoveho jevu
        for j=1:vel
            suma_Ina2=0;
            for jj=j:vel
                suma_Ina2=suma_Ina2+I(jj);
            end
            suma_xI=suma_xI+xi(j)*(abs(suma_Ina2))^2;
        end
        xt_xi=suma_xI/abs(It)^2;
        suma_xr=0;                            %Vypocet cinitele mag
vodivosti drazky bez uvazovanim povrchoveho jevu
        for j=1:vel
            suma_rna2=0;
            for jj=j:vel
                suma_rna2=suma_rna2+1/ri(jj);
            end
            suma_xr=suma_xr+xi(j)*(suma_rna2)^2;
        end
        xt=suma_xr*rt^2;
        n=n+1;
        if xt==0
            kX(n)=1;
        else
            kX(n)=xt_xi/xt;                   %Vypocet cinitelu kX a
kR
        end
        kR(n)=rt_xi/rt;
    end
    kR=kR./min(kR);                           %Pu
    kX=kX./max(kX);
    setappdata(0,'kR',kR);                    %Predani parametru
```

```
    setappdata(0,'kX',kX);

%----------------------------uicontrol funkce-------------------------
-------

function radiobutton1_Callback(hObject, eventdata, handles)
    set(handles.radiobutton1,'Value',1)                          %
Blokace odznaceni
    off    = [handles.radiobutton2,handles.radiobutton3];        %
Vypnuti zbyvajicich radiobuttonu
    jeden_vyber(off)
    off    = [handles.editR1];                                   %
Aktivuje zadani parametru
     odblokovat(off)

function radiobutton2_Callback(hObject, eventdata, handles)
    set(handles.radiobutton2,'Value',1)                          %
Blokace odznaceni
    off    = [handles.radiobutton1,handles.radiobutton3];        %
Vypnuti zbyvajicich radiobuttonu
    jeden_vyber(off)
    off    = [handles.editR1];                                   %
Aktivuje zadani parametru
    odblokovat(off)

function radiobutton3_Callback(hObject, eventdata, handles)
    set(handles.radiobutton3,'Value',1)                          %
Blokace odznaceni
    off    = [handles.radiobutton1,handles.radiobutton2];        %
Vypnuti zbyvajicich radiobuttonu
    jeden_vyber(off)
    off    = [handles.editR1];                                   %
Aktivuje zadani parametru
    odblokovat(off)

function pushbutton1_Callback(hObject, eventdata, handles)
    if get(handles.radiobutton1,'Value')==1
%Podminka vyberu metody
        setappdata(0,'roz',1);
        v  =  Vector_gui;                                        %Volani
metody
        uiwait(v);                                               %Pause
    elseif get(handles.radiobutton2,'Value')==1
%Podminka vyberu metody
        setappdata(0,'roz',2);
        s_linear                                                 %Volani
metody
    else
        setappdata(0,'roz',3);
        o  =  obecna_gui;                                        %Volani
metody
        uiwait(o);                                               %Pause
        s_linear
    end
```

```
    vypocet                                                    %Volani
funkce
    off    =
[handles.checkbox1,handles.checkbox2,handles.checkbox3,handles.check
box4];      % Aktivuje Vyper char
    odblokovat(off)

function editR2_Callback(hObject, eventdata, handles)
    R2    = get(handles.editR2,'String');
    R2    = str2double(R2);
    setappdata(0,'R2',R2);
    off    = [handles.editL2];              % Aktivuje zadani
parametru
    odblokovat(off)

function editL2_Callback(hObject, eventdata, handles)
    L2 = get(handles.editL2,'String');
    L2 = str2double(L2);
    setappdata(0,'L2',L2);
    off    =
[handles.radiobutton1,handles.radiobutton2,handles.radiobutton3];
% Aktivuje zadani parametru
    odblokovat(off)

function editR1_Callback(hObject, eventdata, handles)
    R1     = get(handles.editR1,'String');
    R1     = str2double(R1);
    setappdata(0,'R1',R1);
    off    = [handles.editL1];              % Aktivuje zadani
parametru
    odblokovat(off)

function editL1_Callback(hObject, eventdata, handles)
    L1     = get(handles.editL1,'String');
    L1     = str2double(L1);
    setappdata(0,'L1',L1);
    off    = [handles.editm2];              % Aktivuje zadani
parametru
    odblokovat(off)

function editm2_Callback(hObject, eventdata, handles)
    m2      =   get(handles.editm2,'String');
    m2      =   str2double(m2);
    setappdata(0,'m2',m2);
    off    =   [handles.editN1];            % Aktivuje zadani
parametru
    odblokovat(off)

function editN1_Callback(hObject, eventdata, handles)
    N1     = get(handles.editN1,'String');
    N1     = str2double(N1);
    setappdata(0,'N1',N1);
    off    = [handles.editkv1];             % Aktivuje zadani
parametru
    odblokovat(off)
```

```
function editkv1_Callback(hObject, eventdata, handles)
    kv1     = get(handles.editkv1,'String');
    kv1     = str2double(kv1);
    setappdata(0,'kv1',kv1);
    off     = [handles.editp];                % Aktivuje zadani
parametru
    odblokovat(off)

function editp_Callback(hObject, eventdata, handles)
    p      = get(handles.editp,'String');
    p      = str2double(p);
    setappdata(0,'p',p);
    off     = [handles.editRfe];              % Aktivuje zadani
parametru
    odblokovat(off)

function editRfe_Callback(hObject, eventdata, handles)
    Rfe     = get(handles.editRfe,'String');
    Rfe     = str2double(Rfe);
    setappdata(0,'Rfe',Rfe);
    off     = [handles.editLh];               % Aktivuje zadani
parametru
    odblokovat(off)

function editLh_Callback(hObject, eventdata, handles)
    Lh = get(handles.editLh,'String');
    Lh = str2double(Lh);
    setappdata(0,'Lh',Lh);
    off     = [handles.pushbutton1];           % Aktivuje zadani
parametru
    odblokovat(off)

function checkbox4_Callback(hObject, eventdata, handles)
    axes(handles.axes2);
    s_lin=getappdata(0,'vektor_slin');              %Nacteni hodnot
    cos_fi    =getappdata(0,'cos_fi');
    plot([0,s_lin],[0,cos_fi],'k','LineWidth',3)    %Graf
    title('Ucinik')
    xlabel('s[/]','FontSize',14)
    ylabel('cos(\phi)','FontSize',14)
    grid on
    set(handles.checkbox4,'Value',1)                % Blokace
odznaceni
    off     =
[handles.checkbox1,handles.checkbox2,handles.checkbox3]; % Vypnuti
zbyvajicich radiobuttonu
    jeden_vyber(off)

function checkbox3_Callback(hObject, eventdata, handles)
    axes(handles.axes2);
    s_lin=getappdata(0,'vektor_slin');              %Nacteni hodnot
    I1     =getappdata(0,'I1');
    I2     =getappdata(0,'I2');
```

```
plot([0,s_lin],[0,abs(I1)],'m',[0,s_lin],[0,I2],'c','LineWidth',3)
%Graf
    title('Statorovy a rotorovy proud')
    xlabel('s[/]','FontSize',14)
    ylabel('I[A]','FontSize',14)
    legend('I_s','I_r')
    grid on
    set(handles.checkbox3,'Value',1)                % Blokace
odznaceni
    off    =
[handles.checkbox1,handles.checkbox2,handles.checkbox4]; % Vypnuti
zbyvajicich radiobuttonu
    jeden_vyber(off)

function checkbox2_Callback(hObject, eventdata, handles)
    axes(handles.axes2);
    s_lin=getappdata(0,'vektor_slin');              %Nacteni hodnot
    P2    =getappdata(0,'P2');
    plot(s_lin,P2,'b','LineWidth',3)                %Graf
    title('Mechanicky vykon')
    xlabel('s[/]','FontSize',14)
    ylabel('P2[W]','FontSize',14)
    grid on
    set(handles.checkbox2,'Value',1)              % Blokace
odznaceni
    off    =
[handles.checkbox1,handles.checkbox3,handles.checkbox4]; % Vypnuti
zbyvajicich radiobuttonu
    jeden_vyber(off)

function checkbox1_Callback(hObject, eventdata, handles)
    axes(handles.axes2);
    s_lin=getappdata(0,'vektor_slin');              %Nacteni hodnot
    M    =getappdata(0,'M');
    plot(s_lin,M,'g','LineWidth',3)                 %Graf
    title('Tocivy moment')
    xlabel('s[/]','FontSize',14)
    ylabel('M[Nm]','FontSize',14)
    grid on
    set(handles.checkbox1,'Value',1)                % Blokace
odznaceni
    off    =
[handles.checkbox2,handles.checkbox3,handles.checkbox4]; % Vypnuti
zbyvajicich radiobuttonu
    jeden_vyber(off)
```

Obecna_gui.m

```
% ------------------------------------------------------------------
----- %
%|                              Diplomova Prace
|%
%|               Rozbehove charakteristiky asynchronniho stroje
|%
%|                            Bc. Konstantin Ryba
|%
%|                                 2014
|%
% ------------------------------------------------------------------
----- %
function varargout = obecna_gui(varargin)
% OBECNA_GUI MATLAB code for obecna_gui.fig
%      OBECNA_GUI, by itself, creates a new OBECNA_GUI or raises the
existing
%      singleton*.
%
%      H = OBECNA_GUI returns the handle to a new OBECNA_GUI or the
handle to
%      the existing singleton*.
%
%      OBECNA_GUI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in OBECNA_GUI.M with the given input
arguments.
%
%      OBECNA_GUI('Property','Value',...) creates a new OBECNA_GUI
or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before obecna_gui_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to obecna_gui_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help obecna_gui

% Last Modified by GUIDE v2.5 27-Apr-2014 11:51:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @obecna_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @obecna_gui_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function obecna_gui_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to obecna_gui (see VARARGIN)

% Choose default command line output for obecna_gui
handles.output = hObject;
Vyska    = getappdata(0,'vyska');                    %Nastaveni
predchozich hodnot
set(handles.edit1,'String',Vyska);
Data     = getappdata(0,'Data');
set(handles.uitable2,'data',Data);
Rho      = getappdata(0,'Rho');
set(handles.edit2,'String',Rho);
% Update handles structure
guidata(hObject, handles);

function varargout = obecna_gui_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------
-------
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%----------------------------uicontrol funkce-----------------------
-------

function uitable2_CellEditCallback(hObject, eventdata, handles)
    set(handles.pushbutton1,'Enable','on')
%Odblokovani tlacitka
    drazka_X =  get(handles.uitable2,'data');
%Nacteni dat z tabulky
    drazka_X =  drazka_X(end:-1:1);
%Obraceni poradi pro zobrazeni
    h        =  get(handles.edit1,'String');
%Nacteni vysky drazky
    h        =  str2double(h);
    Y_pom    =  linspace(0,h,20);                                %Y
souradnice pro vykresleni drazky
    drazka_Y =  ones(1,length(drazka_X))*h/length(drazka_X); %Y
souradnice pro vypocet
    R        =  max(drazka_X);
%Meritko pro osu X pri vykresleni
    X_pom    =  drazka_X'./2;                                    %X
souradnice pro zobrazeni s osou uprostred
    drazka_X =  drazka_X(end:-1:1);
%Obraceni poradi pro vypocet
    drazka_X =  drazka_X./1000;
%Prevedeni vektoru na m
    setappdata(0,'drazka_X',drazka_X);
%Ulozeni dat do root
    setappdata(0,'drazka_Y',drazka_Y');
%Ulozeni dat do root
    setappdata(0,'Data',drazka_X*1000);
    axes(handles.axes1);
%Vkresleni
    plot(X_pom,Y_pom,'b',-X_pom,Y_pom,'b')
    axis([-R R 0 h])
%Měritko os a popis
    title('Tvar drazky')
    xlabel('Sirka drazky [mm]')
    ylabel('Vyska drazky [m]')

function edit1_Callback(hObject, eventdata, handles)
```

```
    hodnota  = get(handles.edit1,'String');
%Nacteni hodnot z pole textu
    hodnota  = str2double(hodnota);
    if hodnota>0
        set(handles.edit2,'Enable','on')
%Odblokovani Rho
        set(handles.uitable2,'Enable','on')
%Odblokovani tabulky
        setappdata(0,'vyska',hodnota)
    else
        set(handles.edit2,'Enable','off')
%Zablokovani Rho
        set(handles.uitable2,'Enable','off')
%Zablokovani tabulky
        set(handles.pushbutton1,'Enable','off')
%Zablokovani Tlacitka
    end

function pushbutton1_Callback(hObject, eventdata, handles)
    uiresume(gcf)                   %Pokracovat za volanim teto gui
    close obecna_gui                %Zavreni teto gui

function edit2_Callback(hObject, eventdata, handles)
    hodnota  = get(handles.edit2,'String');
%Nacteni hodnoty pole textu
    hodnota  = str2double(hodnota);
    if hodnota<=0
        Rho  = getappdata(0,'Rho');
        set(handles.edit2,'String',Rho);
%Nastaveni puvodni hodnoty
    else
    setappdata(0,'Rho',hodnota);
%Ulozeni rho do root
    end
```

Vector_gui.m

```
% ------------------------------------------------------------------
----- %
%|                              Diplomova Prace
|%
%|               Rozbehove charakteristiky asynchronniho stroje
|%
%|                            Bc. Konstantin Ryba
|%
%|                                 2014
|%
% ------------------------------------------------------------------
----- %
function varargout = Vector_gui(varargin)
% VECTOR_GUI MATLAB code for Vector_gui.fig
%      VECTOR_GUI, by itself, creates a new VECTOR_GUI or raises the
existing
%      singleton*.
%
%      H = VECTOR_GUI returns the handle to a new VECTOR_GUI or the
handle to
%      the existing singleton*.
%
%      VECTOR_GUI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in VECTOR_GUI.M with the given input
arguments.
%
%      VECTOR_GUI('Property','Value',...) creates a new VECTOR_GUI
or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Vector_gui_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Vector_gui_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Vector_gui

% Last Modified by GUIDE v2.5 28-Mar-2014 22:26:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn', @Vector_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @Vector_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Vector_gui_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Vector_gui (see VARARGIN)

% Choose default command line output for Vector_gui
handles.output = hObject;
 vec=getappdata(0,'Data_vec');
 set(handles.uitable1,'Data',vec);
% Update handles structure
guidata(hObject, handles);

function varargout = Vector_gui_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%-----------------------------uicontrol funkce-----------------------
-------

function uitable1_CellEditCallback(hObject, eventdata, handles)
    Y  = get(handles.uitable1,'data');      %Nacteni dat z tabulky
    setappdata(0,'Data_vec',Y);
    X  = linspace(0,1,20);                  %Vtvoreni pomocneho
vyktoru
    R  = min(Y);                            %Zjisteni min hodnoty
    Y  = Y./R;                              %Pomerna hodnota s
    xx = linspace(0,1,100);                 %Pomocny vektor pro
prolozeni
    yy = spline(X,Y,xx);                    %Prolozeni
    setappdata(0,'X_vector',X);             %Aktualizace dat v
handles
    setappdata(0,'vektor_s',yy);
```

```
    axes(handles.axes2);
    plot(xx,yy,'r',X,Y,'.')                    %Vykresleni
    title('Průběh skluzu s')
    xlabel('t [-]')
    ylabel('s [-]')

function pushbutton2_Callback(hObject, eventdata, handles)
    uiresume(gcf)                       %Pokracovat za volanim teto gui
    close                               %Zavreni teto gui
```