UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Tvorba samostatně spustitelných aplikací v MATLABu

Jindřich Vlasák

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jindřich VLASÁK**
Osobní číslo:	**I07484**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Název tématu:	**Tvorba samostatně spustitelných aplikací v MATLABu**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

Cíl:
zjistit možnosti a omezení tvorby Stand Alone Application v MATLABu
Teoretická část:
a) princip lineární regrese (polynomiální aproximace metodou nejmenších čtverců)
b) nastudovat, popsat možnosti a omezení tvorby samostatně spustitelných aplikací v MATLABu včetně problematiky distribuce a licenčních omezení
Praktická část:
vytvořit ukázkovou aplikaci v MATLABu obsahující GUI rozhranní, která zpracuje data ze souboru a vykreslí do grafu výsledek. Grafické rozhranní bude realizovat
a) volbu souboru s daty
b) volbu stupně aproximačního polynomu
c) vykreslení výsledku

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Rektorys,K.: Přehled užité matematiky, či jiná literatura zabývající se lineární regresí
MATLAB on line dokumentace

Vedoucí bakalářské práce: **doc. Ing. František Dušek, CSc.**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

prof. Ing. Simeon Karamazov, Dr.
děkan

L.S.

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 5. 5. 2010 Jindřich Vlasák

Poděkování

Rád bych poděkoval vedoucímu práce, Doc. Ing. Františku Duškovi, CSc., za cenné rady a připomínky, které mi poskytl během vypracování mé bakalářské práce, a za jeho ochotný přístup.

Anotace

Tato práce je věnována popisu možností a omezení tvorby samostatně spustitelných aplikací v MATLABu včetně problematiky distribuce a licenčních omezení. Praktická část se zabývá tvorbou ukázkové samostatně spustitelné aplikace v MATLABu.

Klíčová slova

MATLAB, MATLAB kompilátor, MATLAB Compiler Runtime, samostatně spustitelná aplikace

Title

Build standalone executable application in MATLAB

Annotation

This work focused on description of posibility and limitation of production standalone executable applications in MATLAB, including problems of distribution and licence limitation. Practical part focused on production of speciment standalone executable applications in MATLAB.

Keywords

MATLAB, MATLAB Compiler, MATLAB Compiler Runtime, standalone application

Obsah

Seznam zkratek

GUI	Graphical User Interface
MCR	MATLAB Compiler Runtime
JIT	Just in Time adaptive compilation
VISA	Virtual Instrument Software Architecture
GPIB	Hewlett-Packard Interface Bus (sběrnice)

Seznam obrázků

Seznam tabulek

1 Úvod

Tato práce se zabývá tvorbou samostatně spustitelných aplikací v MATLABu. Nástroj MATLAB je výborným pomocníkem při řešení různých jednoduchých i složitých technických výpočtů, při kterých využívá předem nadefinované funkce. Uživateli pak stačí napsat pár programových řádků pro řešení složitých výpočtů. Problém nastane, kdybyste potřebovali využít některé MATLAB funkce v jiném programovacím jazyku např. C nebo C++, kde byste museli tu samou funkci složitě programovat. A právě MATLAB umožňuje řešit tento problém. Dokáže převádět MATLAB aplikace nebo funkce na C nebo C++ samostatně spustitelné aplikace nebo knihovny. K tomu potřebujete mít nainstalovaný MATLAB Compiler.

Vlastní práce spočívá ve vytvoření ukázkové samostatně spustitelné aplikace v MATLABu obsahující GUI rozhranní. Aplikace bude řešit problém, jak proložit načtená data ze souboru aproximačním polynomem pomocí metody nejmenších čtverců.

První část této práce je věnována základnímu seznámení s prostředím MATLAB. Druhá část se věnuje teorii regresní analýzy, kde se čtenář seznámí s metodou nejmenších čtverců. Ve třetí části jsou rozebírány možnosti a omezení tvorby samostatně spustitelné aplikace včetně problematiky distribuce a licenčních omezení. A poslední část obsahuje praktické řešení tvorby ukázkové samostatně spustitelné aplikace v MATLABu.

2 Seznámení s MATLABem

Celá druhá kapitola vychází z materiálů získaných z webových stránek (www.humusoft.cz) a dále uvedený text je zkrácenou verzí těchto materiálů.

2.1 MATLAB

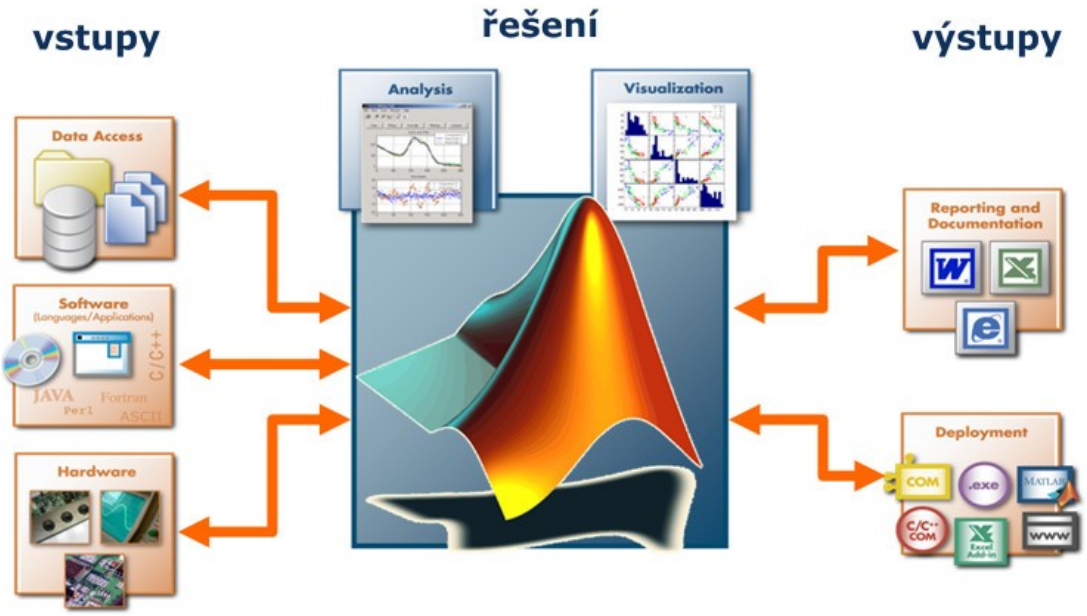
MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.

Výpočetní systém MATLAB se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací ve sféře vědy, výzkumu, průmyslu i v oblasti vzdělávání.

MATLAB poskytuje svým uživatelům nejen mocné grafické a výpočetní nástroje, ale i rozsáhlé specializované knihovny funkcí spolu s výkonným programovacím jazykem čtvrté generace. Knihovny jsou svým rozsahem využitelné prakticky ve všech oblastech lidské činnosti.

Díky své architektuře je MATLAB určen zejména těm, kteří potřebují řešit početně náročné úlohy a přitom nechtějí nebo nemají čas zkoumat matematickou podstatu problémů. Za nejsilnější stránku MATLABu je považováno mimořádně rychlé výpočetní jádro s optimálními algoritmy, které jsou prověřeny léty provozu na špičkových pracovištích po celém světě. MATLAB byl implementován na všech významných platformách (Windows, Linux, Solaris, Mac) [1].

MATLAB také obsahuje prostředky pro výměnu informací pomocí nejrůznějších technologií jak je naznačeno na následujícím obrázku 1 (převzato z [1]).



Obrázek 1 – Zpracování vstupních dat do výstupních souborů

Systém MATLAB nabízí:

- rychlé výpočetní jádro,
- podpora paralelních výpočtů,
- akcelerace výpočtů - JIT (Just in Time adaptive compilation),
- otevřený a rozšiřitelný systém,
- působivá 2D a 3D grafika,
- konfigurovatelné uživatelské rozhraní MATLAB Desktop,
- velké množství aplikačních knihoven,
- programovací jazyk 4. generace,
- objektové programování,
- integrace s jazykem Java,
- podpora vícerozměrných polí a uživatelsky definovaných datových struktur,
- interaktivní nástroje pro tvorbu grafického uživatelského rozhraní,
- podpora řídkých matic,
- interaktivní průvodce importem dat,
- zvukový vstup a výstup, animace,
- komunikace s externím přístrojovým vybavením (sériová linka, GPIB, VISA),
- výpočetní jádro pro programy psané ve Fortranu a jazyce C,
- distribuce nezávislých uživatelských aplikací: překlad do jazyka C, runtime modul, WWW technologie,
- rozsáhlá dokumentace v pdf nebo v on-line hypertextové formě.

2.2 Výpočetní jádro

Nejpodstatnější součástí numerického jádra MATLABu jsou algoritmy pro operace s maticemi reálných a komplexních čísel. MATLAB umožňuje provádět všechny běžné operace jako násobení, inverze, determinant atd. a v nejjednodušší podobě je možno jej použít jako maticový kalkulátor, protože všechny tyto operace se zapisují téměř tak, jako bychom je psali na papíře. Kromě datových typů jednodušších než tradiční matice podporuje MATLAB také typy složitější, jako jsou např. vícerozměrná pole reálných nebo komplexních čísel. Dalším datovým typem jsou tzv. pole buněk, tedy struktury podobné maticím, ve kterých ovšem každý prvek může být jiného typu. Podobně lze tvořit datové struktury, kde jsou prvky rozlišeny ne souřadnicemi, ale jménem, takže připomínají struktury známé z běžných programovacích jazyků. Skládáním těchto datových typů je pak možné vytvořit libovolně složité datové struktury. MATLAB ukládá všechna čísla v tzv. dvojité přesnosti. Vektor reálných čísel může v MATLABu představovat i polynom a operace s polynomy jsou v programu rovněž obsaženy. Vektory mohou také reprezentovat časové řady nebo signály a MATLAB obsahuje funkce pro jejich analýzu - výpočet střední hodnoty, hledání extrémů, výpočet směrodatné odchylky, korelačních koeficientů, rychlé Fourierovy transformace. MATLAB také podporuje speciální formát uložení tzv. řídkých matic, což jsou rozměrem velké matice, které obsahují většinu nulových prvků. Další významnou vlastností jazyka MATLABu je možnost práce s objekty. Ty uživateli umožňují rozšířit výpočetní prostředí o nové datové typy, na kterých je možno definovat libovolné funkce a operátory [1].

2.3 Grafický subsystém

Grafika v MATLABu umožňuje snadné zobrazení a prezentaci výsledků získaných výpočtem. Je možné vykreslit různé druhy grafů: dvourozměrné pro funkce jedné proměnné, třírozměrné pro funkce dvou proměnných, histogramy, koláčové grafy a další. Všem grafickým objektům je možné téměř libovolně měnit vzhled, a to jak při jejich vytváření, tak po jejich nakreslení. Tak je možné stínovat třírozměrné grafy s určením zdroje dopadajícího světla, animovat grafy včetně třírozměrných, zobrazovat kontury a transparentní objekty, používat pseudo-barevné zobrazení, a mnoho dalšího. Většinu těchto efektů je možné docílit jedním nebo několika málo příkazy a pro jejich rychlé vykreslení se používá algoritmus Z-buffer nebo technologie OpenGL, pokud ji použitý počítač podporuje.

Obrázky v grafických oknech MATLABu navíc nejsou statické - každý již nakreslený objekt má přiřazen identifikátor (handler), jehož prostřednictvím je možné měnit vlastnosti objektu a tím i jeho vzhled. Vzhled grafických objektů je také možno měnit interaktivně, pomocí lišty nástrojů umístěné pod záhlavím obrázku. Grafický systém MATLABu, nazvaný Handle Graphics, dovoluje vkládat do obrazů ovládací prvky (tlačítka, apod.) a vytvořit tak aktivní graficky ovládané uživatelské rozhraní [1].

2.4 Otevřená architektura

Vlastností, která patrně nejvíce přispěla k rozšíření MATLABu, je jeho otevřená architektura. MATLAB je úplný programovací jazyk, to znamená, že uživatelé v něm mohou vytvářet funkce "šité na míru" pro jejich aplikace. Tyto funkce se způsobem volání nijak neliší od vestavěných funkcí a jsou uloženy v souborech v čitelné formě. Dokonce většina funkcí s MATLABem dodávaných je takto vytvořena a opravdu vestavěné jsou jen funkce základní. To má dvě velké výhody: jazyk MATLABu je téměř neomezeně rozšiřitelný a kromě toho se uživatel může při psaní vlastních funkcí poučit z dodaných algoritmů. Navíc jsou takto koncipované funkce snadno přenosné mezi různými platformami, na kterých je MATLAB implementován. Všechny moduly systému doprovází rozsáhlá pdf i hypertextová on-line dokumentace, která uživatelům usnadňuje orientaci ve funkcích MATLABu. Otevřená architektura MATLABu inspirovala mnoho nezávislých firem k vývoji a distribuci vlastních produktů, které buď rozšiřují výpočetní prostředí MATLAB o další knihovny a nástroje nebo zajišťují propojení MATLABu s jinými specializovanými programy. Rodina MATLABu tak obsahuje kromě více než 90-ti modulů z autorské dílny firmy The MathWorks ještě dalších více než 300 komerčně distribuovaných "third-party" produktů a velké množství volně přístupných akademických aplikací [1].

2.5 Pracovní nástroje

MATLAB je koncipován tak, aby kromě pohodlné interaktivní práce umožňoval i programování aplikací. Programovací jazyk obsahuje všechny nezbytné příkazy pro psaní programů, jako jsou podmíněné příkazy, větvicí příkazy, cykly a podobně. Přes

jednoduchost a snadnou zvladatelnost je jazyk MATLABu úplným programovacím jazykem čtvrté generace, ve kterém je možné vyvíjet i velice složité aplikace.

Základním nástrojem výpočetního systému je uživatelské rozhraní MATLAB Desktop. Pracovní nástroje jako prohlížeč adresářů a souborů, prohlížeč pracovního prostoru, okno historie příkazů, interaktivní spouštěč aplikací, editor, debugger, profiler, hypertextová nápověda a příkazové okno jsou do prostředí plně integrovány. Uživatelské rozhraní je konfigurovatelné, takže si uživatel může přizpůsobit rozměry a počet zobrazených nástrojů tak, aby to maximálně vyhovovalo jeho potřebám. Je tak možné vytvořit pracovní plochu, která vyhovuje jak začátečníkům, tak pokročilým. K vytvoření dokonalé grafické podoby uživatelské aplikace pomáhá interaktivní nástroj pro vytváření uživatelských rozhraní (GUI rozhraní), ve kterém lze snadno a přehledně vytvořit a uspořádat ovládací prvky aplikace. Důležitým pomocníkem je interaktivní nástroj pro import dat, který výrazně usnadní načítání dat prakticky z jakéhokoli zdroje (text, tabulky, databáze, binární data, obrázky, animace, ...). MATLAB je koncipován tak, aby kromě pohodlné interaktivní práce umožňoval i programování aplikací.

Další významnou předností programovacího jazyka MATLABu je jeho těsná integrace s jazykem Java. Objekty jazyka Java mohou být přímo použity programem v MATLABu, což umožňuje jednak vytvářet složitá grafická rozhraní s použitím grafických objektů Javy, jednak využít velkého množství volně dostupných knihoven, které byly v jazyce Java vytvořeny. Kromě toho je možné k MATLABu připojovat také moduly napsané v jazyce C a ve Fortranu [1].

2.6 Toolboxy

Otevřená architektura MATLABu vedla ke vzniku knihoven funkcí, nazývaných toolboxy, které rozšiřují použití programu v příslušných vědních a technických oborech. Tyto knihovny, navržené v jazyce MATLABu, nabízejí předzpracované specializované funkce, které je možno rozšiřovat, modifikovat, anebo jen čerpat informace z přehledně dokumentovaných algoritmů [1].

Díky toolboxům MATLAB najde využití v oblastech aplikované matematiky, automatického řízení a regulace, zpracování signálu a komunikace, zpracování obrazu, měření a testování, výpočetní biologie, finančního modelování a analýzy, modelování fyzikálních soustav.

3 Regresní analýza

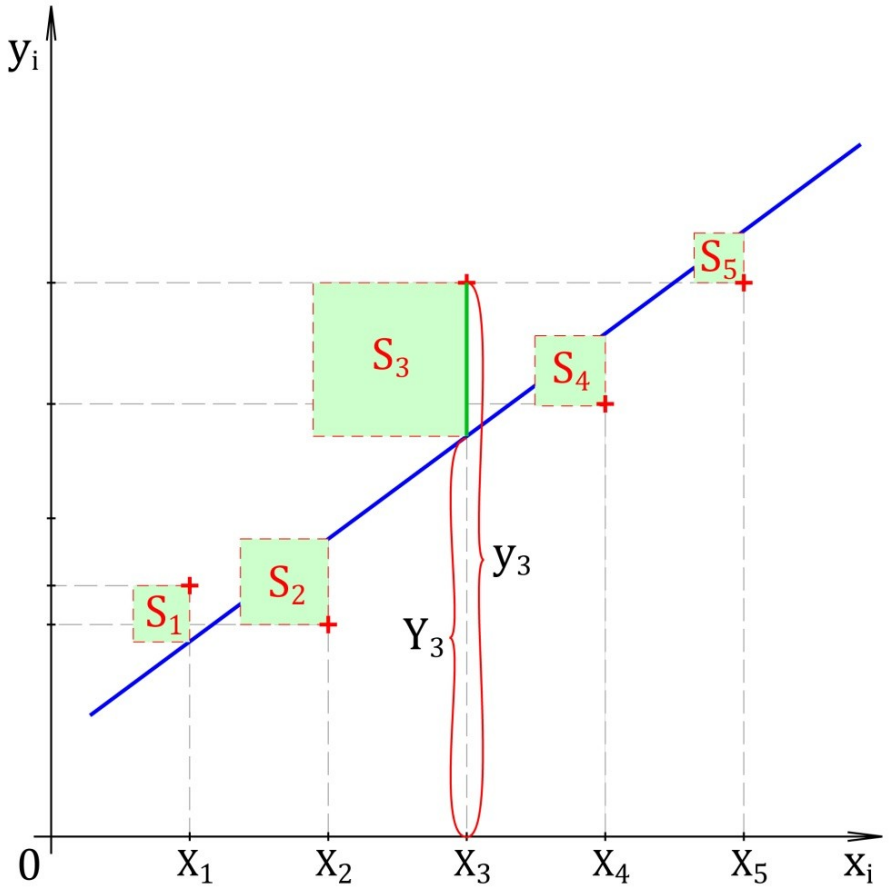
Tato kapitola vychází zejména z literatury *Sbírka příkladů z inženýrské ekonomiky a managementu*, která je dostupná na webových stránkách (vydavatelstvi.vscht.cz) a je doplněna o informace získané z *Wikipedie*.

Regresní analýza je označení statistických metod, pomocí nichž odhadujeme hodnotu jisté náhodné veličiny (takzvané závisle proměnné*)* na základě znalosti jiných veličin (nezávisle proměnných*)*. **Nezávisle proměnná veličina** je ta, jejíž změna má za následek změnu jiných veličin, které se proto nazývají **závislé proměnné**. Závislosti mezi uvedenými veličinami mohou být funkční (matematické) nebo korelační (statistické, stochastické). **Funkční závislosti jsou typické** v teoretických oblastech např. fyzice nebo matematice. Na základě známé resp. známých nezávisle proměnných veličin lze jednoznačně stanovit závisle proměnnou veličinu. Při praktickém používání se častěji vyskytují statistické závislosti, kde jedné konkrétní hodnotě nezávisle proměnné odpovídá celé rozdělení četností závisle proměnných veličin. Je tomu tak proto, že na závisle proměnnou veličinu mají vliv i jiné neměřené proměnné.

Úkolem regresní analýzy je vytvoření regresního modelu. Jde o určení vztahu mezi nezávisle a závisle proměnnými veličinami. Při jeho konstrukci je nutné nejprve odhadnout nejvhodnější empirickou funkci. Nejjednodušším a nejčastěji používaným regresním modelem je přímka, která předpokládá lineární závislost závislé proměnné na nezávislé proměnné. V tomto případě se s touto metodou lze setkat pod názvem **lineární regrese**, kde existuje jenom jedno přesné řešením. Pokud vybraný tvar regresního modelu není lineárně závislý na hledaných parametrech jedná se o **nelineární regresi**, kde neexistuje obvykle analytické řešení, ale k výsledku se dospěje pomocí numerických metod pouze přibližně.

Po zvolení empirické funkce je potřebné určit její konstanty tak, aby jí naměřené hodnoty co nejlépe odpovídaly. K tomu se používá **metody nejmenších čtverců (MNČ)**, která je založena na minimalizaci součtu čtverců odchylek empirických hodnot (y_i) od hodnot teoretických (Y_i). Princip použití MNČ je přiblížen na následujícím obrázku 2. Matematický zápis této metody je následující:

$$S = \sum_{i=1}^{n} [Y_i - y_i]^2 = min \tag{1}$$



Obrázek 2 – Proložení přímky skupinou bodů pomocí metody nejmenších čtverců

Při výpočtu se místo teoretických hodnot dosadí příslušný matematický předpis zvolené regresní funkce a dosadí se konkrétní empirické hodnoty. Minimalizace funkce se provede parciálními derivacemi podle regresních konstant. Jednotlivé parciální derivace se položí rovny nule a tím se získají tzv. normální rovnice, z nichž se vyjádřením spočtou regresní konstanty [2].

3.1 Lineární regrese

Termín **lineární regrese** může označovat dvě částečně odlišné věci. Za prvé může termín **lineární regrese** představovat aproximaci daných hodnot $[x_i, y_i]$ polynomem prvního řádu (přímkou) metodou nejmenších čtverců. Jinak řečeno, jedná se o proložení několika bodů v grafu takovou přímkou, aby **součet druhých mocnin odchylek** jednotlivých bodů od přímky byl **minimální**.

V obecnějším případě může **lineární regrese** znamenat aproximaci daných hodnot $[x_i, y_i]$ takovou funkcí $y = f(x, b_1, b_2, \dots, b_k)$, kterou lze vyjádřit jako lineární kombinaci $y = b_1 f_1(x) + \cdots + b_k f_k(x)$. Koeficienty $b_1, \dots, b_k$ jsou vypočteny metodou nejmenších čtverců [3].

3.1.1 Odvození vztahů pro lineární regresi

Uvažuje se nyní funkční závislost:

$$f(x) = ax + b \tag{2}$$

kde a, b - jsou parametry přímky.

Součet S čtverců odchylek empirických hodnot y_i od hodnot hledané funkce $y = ax + b$ bude mít tvar:

$$S(a,b) = \sum_{i=1}^{n} [f(x_i) - y_i]^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 \tag{3}$$

kde x_i, y_i – jsou souřadnice aproximovaných bodů.

Aby se našlo minimum součtu čtverců $S(a,b)$, položí se parciální derivace podle obou parametrů (a,b) rovny nule:

$$0 = \frac{\partial S}{\partial a} = 2\sum_{i=1}^{n} (ax_i + b - y_i)\, x_i \tag{4}$$

$$0 = \frac{\partial S}{\partial b} = 2\sum_{i=1}^{n} (ax_i + b - y_i) \tag{5}$$

Úpravami se obdrží soustava:

$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \tag{6}$$

$$a\sum_{i=1}^{n} x_i^2 + bn = \sum_{i=1}^{n} y_i \tag{7}$$

Jejím řešením pro konkrétní hodnoty x_i a y_i se získají hledané hodnoty parametrů a a b.

$$a = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{8}$$

$$b = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{9}$$

Podobný postup lze aplikovat na jakýkoliv vztah, který je lineární vzhledem k hledaným parametrům.

Pro případ, že se uvažuje obecný tvar lineární regrese, aproximační funkce bude mít tvar $y = f(x, b_1, b_2, \dots, b_p)$, přičemž $b_1, b_2, \dots, b_p$ jsou její parametry, které hledáme.

Součet S čtverců odchylek empirických hodnot y_i od hodnot hledané funkce $y = f(x_i, b_1, b_2, \dots, b_p)$ bude mít tvar:

$$S = \sum_{i=1}^{n} (y_i - f(x_i, b_1, b_2, \dots, b_p))^2 \tag{10}$$

kde x_i, y_i – jsou souřadnice aproximovaných bodů.

Dále se omezí tvar funkce y na lineární regresní funkci:

$$y = b_1 f_1(x) + \cdots + b_p f_p(x) \tag{11}$$

Hledané minimum součtu čtverců S pak bude mít tvar:

$$S = \sum_{i=1}^{n} (y_i - b_1 f_1(x_i) - \dots - b_p f_p(x_i))^2 = min \tag{12}$$

Nutnou podmínkou existence minima funkce S je:

$$\frac{\partial S}{\partial b_j} = 0 \tag{13}$$

pro $j = 1, 2, \dots, p$.

Řešením parciální rovnice obdržíme:

$$\sum_{i=1}^{n} f_j(x_i) f_1(x_i)\, b_1 + \cdots + \sum_{i=1}^{n} f_j(x_i) f_p(x_i) b_p = \sum_{i=1}^{n} y_i\, f_j(x_i) \tag{14}$$

Když označíme:

$$\sum_{i=1}^{n} f_j(x_i) f_h(x_i) = a_{jh} \tag{15}$$

$$\sum_{i=1}^{n} y_i f_j(x_i) = z_j \tag{16}$$

Získáme soustavu rovnic:

$$\begin{aligned} a_{11}b_1 + a_{12}b_2 + \cdots + a_{1p}b_p &= z_1 \\ a_{21}b_1 + a_{22}b_2 + \cdots + a_{2p}b_p &= z_2 \\ &\vdots \\ a_{p1}b_1 + a_{p2}b_2 + \cdots + a_{pp}b_p &= z_p \end{aligned} \tag{17}$$

Řešením této soustavy tzv. normálních rovnic obdržíme hledané odhady $b_1, b_2, \dots, b_p$ [3].

Toto řešení se může zapsat pomocí maticového tvaru:

$$\mathbf{A} \cdot \mathbf{b} = \mathbf{z} \tag{18}$$

Kde konečné řešení vektoru **b** má tvar:

$$\mathbf{b} = \mathbf{A^{-1}}.\mathbf{z} \tag{19}$$

Na ukázku pro aproximační funkci polynomu m-tého stupně $y = b_0 + b_1 x + b_2 x^2 + \cdots + b_m x^m$ má konečná soustava rovnic tvar [4]:

$$\begin{aligned} nb_0 + \sum_{i=1}^{n} x_i b_1 + \cdots + \sum_{i=1}^{n} x_i^m b_m &= \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i b_0 + \sum_{i=1}^{n} x_i^2 b_1 + \cdots + \sum_{i=1}^{n} x_i^{m+1} b_m &= \sum_{i=1}^{n} x_i y_i \\ &\vdots \\ \sum_{i=1}^{n} x_i^m b_0 + \sum_{i=1}^{n} x_i^{m+1} b_1 + \cdots + \sum_{i=1}^{n} x_i^{2m} b_m &= \sum_{i=1}^{n} x_i^m y_i \end{aligned} \tag{20}$$

4 MATLAB kompilátor 4.13

Informace shrnuté v této kapitole byly získány v on-line dokumentaci k MATLABu (www.mathworks.com) a dále uvedený text je volným překladem z originální dokumentace.

MATLAB Compiler umožňuje automaticky konvertovat uživatelské programy vytvořené v MATLABu do **samostatných aplikací a softwarových komponent** a sdílet je s koncovými uživateli. Tento proces kompilování je někdy označovan jako „*building*“. Aplikace a komponenty vytvořené pomocí MATLAB Compileru nevyžadují ke svému chodu MATLAB, ale používají tzv. „runtime engine“ nebo-li MATLAB Compiler Runtime (MCR). MCR je samostatná sada sdílených knihoven, které umožní spustit MATLAB soubory na počítačích bez nainstalované (licencované) verze MATLABu. MCR je poskytován společně s MATLAB Compilerem a vytvořená aplikace je určena k volné distribuci bez nutnosti vlastnit licenci na MATLAB. Pro běh samostatně spustitelné aplikace nebo sdílené knihovny je zapotřebí, aby koncový uživatel si na svůj počítačích nainstaloval MCR [5].

MATLAB kompilátor se využívá k:

- vytváření samostatně spustitelných aplikací nebo sdílených knihoven z MATLAB aplikací
- zabalení samostatně spustitelných aplikací nebo sdílených knihoven,
- volné distribuci samostatně spustitelných aplikací a softwarových součástí,
- včlenění základních MATLAB algoritmů do aplikací vyvinutých za použití jiných jazyků a technologií,
- skrytí MATLAB kódu tak, že nelze zobrazit nebo upravit [6].

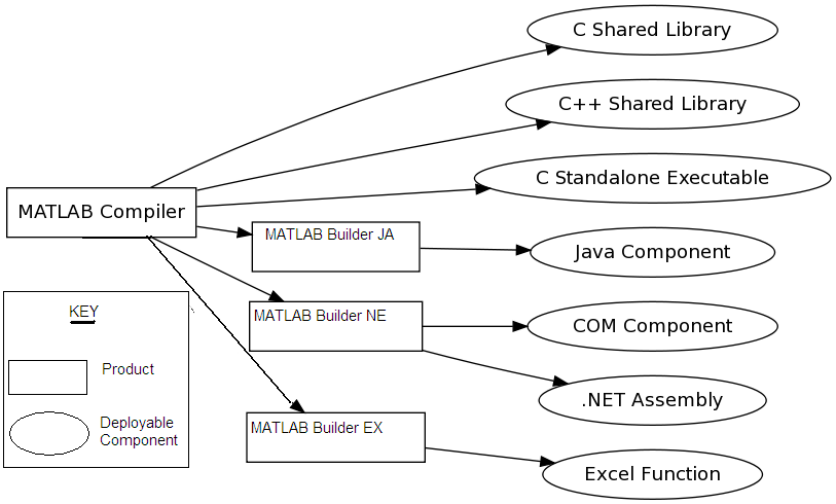
4.1 Builder produkty

Pokud je zapotřebí využívat MATLAB aplikace v dalších programovacích jazyků, nabízí se využití MATLAB Builder produktů (dostupné odděleně) s MATLAB Compilerem. Takto lze konvertovat aplikace a algoritmy v MATLABu do dalších typů softwarových komponent, jako např. Java tříd, .NET komponent nebo Excel doplňků, pro použití v jiných aplikacích. Všechny tyto builder produkty používají ke svému chodu runtime engine nazvaný MATLAB Compiler Runtime, který je poskytován společně s MATLAB Compilerem a je určen pro volnou distribuci [5].

Následující tabulka 1 a obrázek 3 demonstrují možnosti kompilace jednotlivých produktů (převzato z [7]).

Tabulka 1 – Přehled možností kompilace jednotlivých produktů

Produkt	Cíl	Vytvoří samostatně spustitelné soubory?	Vytvoří knihovny funkcí?	Vytvoří aplikace s grafikou?	Vytvoří webové aplikace?
MATLAB Compiler	C a C++ samostatně spustitelné aplikace a knihovny	Ano	Ano	Ano	Ne
MATLAB Builder NE	C# .NET komponenty Visual Basic COM komponenty	Ne	Ano	Ano	Ano
MATLAB Builder JA	Java komponenty	Ne	Ano	Ano	Ano
MATLAB Builder EX	Microsoft Excel doplňky	Ne	Ano	Ano	Ne



Obrázek 3 – Cílové soubory jednotlivých produktů

MATLAB Builder EX

MATLAB Builder EX je nadstavbou MATLAB Compileru, která umožňuje začlenění aplikací vytvořených v MATLABu do sešitů v Excelu ve formě maker nebo doplňků, které mohou být volně distribuovány.

MATLAB Builder NE

MATLAB Builder NE je nadstavbou MATLAB Compileru, která umožňuje tvorbu .NET a COM komponent z aplikací vytvořených v MATLABu. Komponenty lze volně šířit pro použití na osobních počítačích a Web serverech. Také mohou být začleněny do .NET a COM programů.

MATLAB Builder JA

MATLAB Builder JA je nadstavbou MATLAB Compileru, která umožňuje začlenění aplikací vytvořených v MATLABu do Java programů tvorbou Java tříd. Java třídy jsou určené k volnému šíření pro osobní počítače a Web servery.

4.2 Vytváření a balení samostatné aplikace nebo knihovny

Při vytváření své aplikace se může použít vysokoúrovňový "maticově optimalizovanný" MATLAB jazyk a zabudované matematické, grafické, datové funkce a funkce pro analýzu pro rychlé vytvoření, vyvinutí a testování aplikací a funkcí. Z těchto MATLAB aplikací se pak s pomocí MATLAB Compileru vytvoří samostatně spustitelné aplikace nebo sdílené knihovny (dynamicky linkované knihovny – DLL na Microsoft Windows) užitím jediného příkazu nebo předpřipraveného grafického rozhraní, snadno je zabalit a distribuovat ke koncovým uživatelům. Při psání C nebo C++ programů, je možné volat MATLAB funkce ze sdílených knihoven, stejně jako když se volá funkce z MATLAB příkazové řádky.

S pomocí MATLAB Compileru se může automaticky zkompilovat vaše MATLAB aplikace do:

- samostatné aplikace,
- C nebo C++ knihovny (DLL ve Windows, sdílené knihovny v Linux a UNIX),
- softwarové součásti, jako například Java třídy, .NET komponent nebo Excel doplňky,pro použití uvnitř dalších aplikací (s produktem MATLAB builder).

S pomocí Deployment Tool lze zabalit MCR spolu s vaší aplikací a jejími podporujícími soubory. Deployment Tool je grafické uživatelské rozhraní, které je součástí MATLAB Compileru. Nabízí grafickou alternativu pro užívání příkazového řádku v MATLABu k sestavení a zabalení komponent pro umístění na jiném počítači. Deployment Tool umožňuje [8]:

- stanovit vaši hlavní MATLAB funkci,
- zabalit vaši aplikaci společně s MCR a podporujícími soubory jako např. datové soubory nebo obrázky,
- uložit kompilaci a předvolby zabalení.

4.3 Distribuce aplikace nebo komponent

Vytvořené aplikace pomocí MATLAB Compileru se můžou distribuovat ke koncovým uživatelům, kteří MATLAB nevlastní. Pokud je zakoupená plná licence, lze tyto aplikace volně šířit. Distribuci vaší aplikace vám usnadní grafické uživatelské rozhraní Deployment Tool, které zabalí aplikaci společně s MCR a dalšími podpůrnými soubory.

MATLAB generované samostatné aplikace se můžou distribuovat na jakýkoli cílový stroj, který má stejný operační systém jako stroj, na kterém byli samostatné aplikace vytvořeny. Například, pokud se instaluje aplikace na počítač s operačním systémem Windows, musí být samostatná aplikace vytvořena MATLAB Compilerem na počítači také s operačním systémem Windows [9].

4.4 Kompilování MATLAB funkcí a toolboxů

MATLAB Compiler 4.13 podporuje plně MATLAB jazyk a většinu MATLAB toolboxů až na nějaké výjimky.

4.4.1 Kompilace toolboxů

Následující tabulka 2 ukazuje MATLAB toolboxy, které můžete použít s MATLAB Compilerem 4.13 a popisuje, co může nebo nemůže být zkompilováno [10].

Tabulka 2 – Kompilace toolboxů

Produkt	**Může být zkompilováno**	**Nemůže být zkompilováno**
Aerospace Toolbox 2.5	Všechno s výjimkou animačních funkcí	Animační funkce: rozhraní FlightGear, letecký simulátor a MATLAB Handle Graphics animation objects
Bioinformatics Toolbox 3.5	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • affyread, proteinplot, phytreetool
Communications Toolbox 4.5	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Control System Toolbox 8.5	• LTI objekty • Příkazy analýzy a syntézy • grafy	• LTI Viewer • SISO Design GUI
Curve Fitting Toolbox 2.2	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Data Acquisition Toolbox 2.16	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Database Toolbox 3.7	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • querybuilder
Datafeed Toolbox 3.5	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • dftool
Econometrics Toolbox 1.3	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Filter Design Toolbox 4.7	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • Kódování generační funkčnosti • Schopnosti Fixed-point
Financial Derivatives Toolbox 5.5.1	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Financial Toolbox 3.7.1	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • ftstool • ftsgui • uicalendar
Fixed-Income Toolbox 1.9	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem

Produkt	Může být zkompilováno	Nemůže být zkompilováno
Fuzzy Logic Toolbox 2.2.11	FIS struktura, manipulace, a simulace	• Celé rozhranní poskytnuté s toolboxem • ANFIS tréninkový algoritmus
Global Optimization Toolbox 3.0	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Image Acquisition Toolbox 3.5	Veškerá funkčnost příkazové řádky	imaqtool
Image Processing Toolbox 7.0	Veškerá funkčnost příkazové řádky včetně modulárních interaktivních nástrojů	• cpselect • imtool • implay
Instrument Control Toolbox 2.10	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • Test & Measurement Tool
Mapping Toolbox 3.1	Veškerá funkčnost příkazové řádky	• maptool • mapview
MATLAB 7.10	Většina funkčnosti příkazové řádky	Příkazové okno, editor, GUIDE a další vývojové nástroje
MATLAB Report Generator 3.8	Většina příkazové řádky, včetně: • Generování sestav skrz zprávu • Konverze dokumentu skrz rptconvert	• Funkčnost příkazové řadky, která porovnává XML soubory skz xmlcomp • Celé rozhranní poskytnuté s toolboxem
Model Predictive Control Toolbox 3.2	MPC objekt, MPC controler design a simulace	• Celé rozhranní poskytnuté s toolboxem • Simulink block
Neural Network Toolbox 6.0.4	Pre-trained síť funce příkazové řádky	• Všechny další funkčnost příkazové řádky • Celé rozhranní poskytnuté s toolboxem • Simulink blocks • gensim
OPC Toolbox 2.1.5	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Optimization Toolbox 5.0	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Parallel Computing Toolbox 4.3	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem, a lokální workers
Partial Differential Equation Toolbox 1.0.16	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
RF Toolbox 2.7	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Signal Processing Toolbox 6.13	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • Kódování generační funkčnosti
Spline Toolbox 3.3.8	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem
Statistics Toolbox 7.3	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem

Produkt	Může být zkompilováno	Nemůže být zkompilováno
System Identification Toolbox 7.4	• vytváření dat a modelů objektů • Předzpracování a zacházení s daty • Simulační modely, výpočtový čas a frekvenční odezvy, a vyhodnocování odpovědí • Transformace modelů, včetně konverze mezi nepřetržitým a diskrétním časem a modelovým zmenšením	• System Identification Tool (GUI) • Simulink blocks pro toolbox (slident) • Modelový odhad, včetně neparametrické analýzy
Vehicle Network Toolbox 1.2	Veškerá funkčnost příkazové řádky	• Celé rozhranní poskytnuté s toolboxem • CAN Tool
Wavelet Toolbox 4.5	Veškerá funkčnost příkazové řádky	Celé rozhranní poskytnuté s toolboxem

4.4.2 Kompilování funkcí

Nějaké funkce nejsou podporované v samostatném módu. Nemůžete je proto zkompilovat s MATLAB Compilerem. Tyto funkce jsou v následujících kategorií [11]:

- Funkce tisku nebo zprávy MATLAB kódu z funkce, např. MATLAB help funkce nebo debug funkce, nebudou pracovat.
- Simulink funkce, obecně nebudou pracovat.
- Funkce, které požadují příkazovou řádku, např. MATLAB lookfor funkce, nebudou pracovat.
- clc , home a savepath nebudou dělat cokoli v deployed módu.

Kvůli množství seznamů MathWorks produktů a funkcí, toto není kompletní seznam funkcí, které nemohou být zkompilovány.

Seznam některých nezajištěných funkcí

add_block	dbtype	keyboard	propedit
add_line	dbup	linmod	propertyeditor
applescript	delete_block	mislocked	publish
close_system	delete_line	mlock	rehash
colormapeditor	depfun	more	restoredefaultpath
createClassFromWsdl	doc	munlock	run
dbclear	echo	new_system	segment
dbcont	edit	open_system	set_param
dbdown	fields	pack	sim
dbquit	figure_palette	plotbrowser	simget
dbstack	get_param	plotedit	simset
dbstatus	help	plottools	sldebug
dbstep	home	profile	type
dbstop	inmem	profsave	

5 Praktická část

Vlastní práce spočívá ve vytvoření ukázkové samostatně spustitelné aplikace v MATLABu obsahující GUI rozhraní. Aplikace bude řešit problém, jak proložit načtená data ze souboru aproximačním polynomem volitelného stupně pomocí lineární regrese (metodou nejmenších čtverců). V první část se zabývá tvorbou této aplikace v MATLABU a v druhá část se zabývá kompilováním aplikace do samostatně spustitelné aplikace.

5.1 Tvorba aplikace s grafickým uživatelským rozhraním v MATLABu

5.1.1 Teorie

MATLAB má implementovánu „velmi silnou“ grafiku nejen pokud se týká zobrazování jakýchkoliv funkčních průběhů, ale navíc umožňuje tvorbu Grafického uživatelského rozhraní (GUI). Tedy jistého panelu, kterým je aplikace překryta, takže uživatel nemusí o MATLABu vědět vůbec nic. Toto GUI lze v MATLABu vytvořit dvěma způsoby a to:

1. přímým naprogramováním,
2. pomocí vestavěného nástroje **GUIDE**.

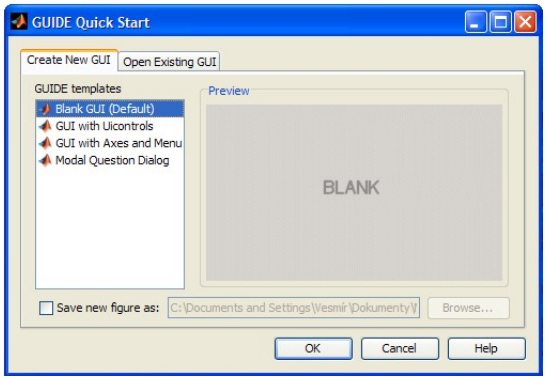
První způsob je obtížnější. Programátor už musí mít zvládnutu filosofii grafiky MATLABu na poměrně vysoké úrovni. O to jednodušší je způsob druhý. Spuštěním vestavěného nástroje, zvaného GUIDE, který vytváří soubor s příponou .fig, kde jsou zapsány informace o grafickém okně, a soubor s príponou .m, který lze editovat, se otevře plocha budoucího rozhraní se čtvercovou sítí a předpřipravenými všemi ovládacími a zobrazovacími objekty (tlačítka, posuvníky, rozbalovací nabídky, zaškrtávací políčka, radiobuttony, zobrazovací plochy s osami,…). Potřebné objekty v potřebném množství uživatel myší „natahá“ na plochu budoucího rozhraní. U objektů může libovolně upravovat jejich vlastnosti, jako je např. velikost, barva atd. Až je spokojen se vzhledem, nechá MATLAB automaticky vygenerovat patřičný program (soubor s příponou. m). Pak už stačí jen do takto předpřipraveného programu dopsat několik programových řádků a nezbytných zpětných vazeb mezi jednotlivými prvky a rozhraní může fungovat. Takto vygenerovaný program sice není optimální, neboť je automaticky složen z robustních programových modulů, které musí fungovat za všech podmínek. Z hlediska požadované funkce je zcela spolehlivý [12].

Ve vygenerovaném programu pak bude mít každý objekt svou subfunkci, která je označená podle typu objektu (push button, radio button,text edit…). U objektů, které mají po kliknutí vykonat nějakou akci (např. push button), mají subfunkce v názvu označení **CallBack**. Do těchto subfunkcí se píšou různé příkazy z MATLABu, které se po vybrání objektu vykonají. Obecně každý objekt má své jedinečné číslo (handle) a své vlastnosti. Pomocí funkce **get** lze tyto vlastnosti objektu zjišťovat a pomocí funkce **set** je lze nastavovat na různé hodnoty.

5.1.2 Postup tvorby ukázkové aplikace

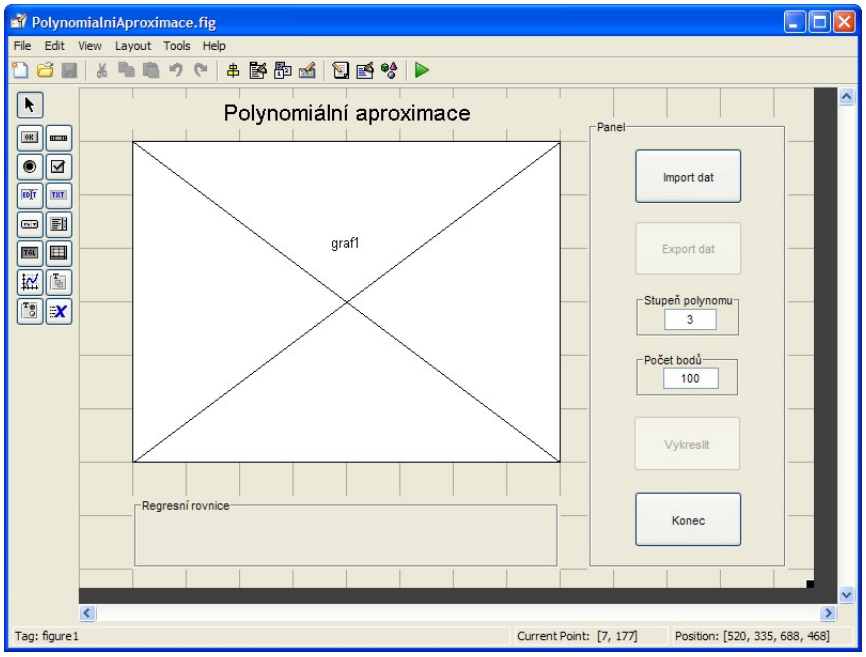
Nyní bude ukázán postup při tvorbě aplikace s rozhraním s pomocí vestavěného nástroje GUIDE. Aplikace obsahuje GUI rozhranní, které načte data z textového souboru, které proloží aproximačním polynomem, jehož stupeň se bude libovolně volit.

Nejdříve se spustí nástroj GUIDE buď pomocí zadáním příkazu **guide** do příkazové řádky nebo přes nabídku menu **File→New→GUI**. Zobrazí se dialogové okno, kde zvolíme Blank GUI (Default) a potvrdí se tlačítkem OK (Obrázek 4). Tím se vytvoří prázdný soubor s příponou .fig a otevře se dialogové okno, kde se vytváří budoucí vzhled grafického rozhraní aplikace (Obrázek 5).



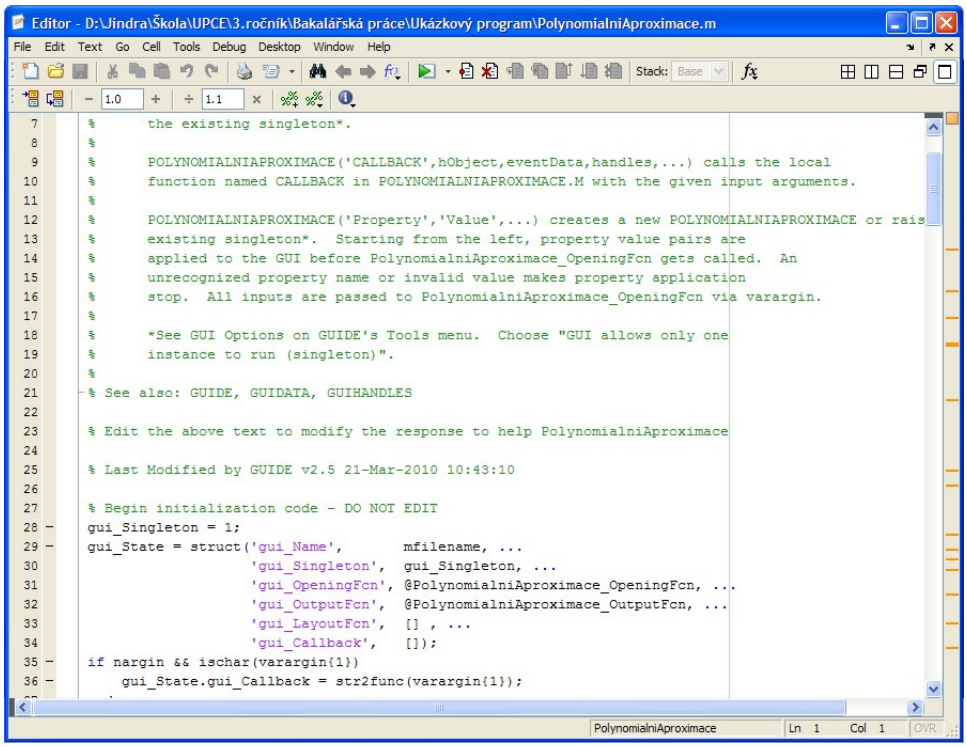
Obrázek 4 – Založení nového projektu

Konkrétně tato aplikace bude osahovat jeden graf, kde se budou zobrazovat načtená data a vykreslený aproximační polynom, čtyři tlačítka push button (Import dat, Export dat, Vykreslit, Konec), která po kliknutí vykonají sled požadovaných akcí, a dvě editační políčka pro určení stupně polynomu a počtu vzorků, ze kterých bude vykreslen daný aproximační polynom. Konečná podoba grafického rozhraní je zobrazena na následujícím obrázku 5.



Obrázek 5 – Vzhled grafického rozhraní

Až bude rozhraní dokončeno, kliknutím na „zelenou šipku“ Run Figure se zobrazí dialogové okno Uložit jako, kde se zvolí místo uložení MATLAB aplikace. Uložená aplikace je m-funkce, která obsahuje automaticky naprogramované grafické rozhraní. Po uložení se dále otevře okno Editoru (Obrázek 6), kde stačí už jen naprogramovat jednotlivé subfunkce, které reprezentují dané objekty (tlačítka).



Obrázek 6 – Editor

V tomto případě stačí naprogramovat čtyři subfunkce a to Import dat, Vykreslit, Export dat a Konec, které se patřičným způsobem propojí, aby aplikace na venek mohla správně fungovat.

Pomocí tlačítka Import dat se data načtou z textového souboru a vykreslí se do grafu. Tlačítkem Vykreslit, které se poté zviditelní, se proloží načtenými daty aproximační polynom stupněm, který se zadá do editačního políčka Stupeň polynomu. Poté tlačítkem Export dat, které se poté zviditelní, se exportuje aproximační polynom do zvoleného souboru. Tlačítkem Konec se aplikace ukončí.

Pro ukázku budou ukázány nejdůležitější subfunkce této aplikace. Jedná se o subfunkci Import dat, Export dat a Vykreslit.

V subfunkci Import_dat se pomocí funkce importdata načtou data ze souboru. Tyto načtená data se zobrazí do grafu a uloží se do vlastnosti User data objektu Import_dat, které slouží jako úložný prostor pro načtená data.

Kód subfunkce Import_dat:

```
function Import_dat_Callback(hObject, eventdata, handles)
% hObject    handle to Import_dat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%vyskakovací okno pro import dat
[jmenosouboru, jmenocesty, filterindex] = uigetfile({'*.mat','MAT-files (*.mat)';
'*.m','M-files(*.m)';'*.*','All Files (*.*)'},'Výběr souboru s daty');

if jmenosouboru~=0
data=importdata(strcat(jmenocesty,jmenosouboru),'\t');%do data se importují data ze
zvoleného souboru
x=data(:,1);%do vektoru x se uloží 1.sloupec z matice data
y=data(:,2);

%zobrazeni grafu
plot(x,y,'.r')%vykreslení grafu
xlabel('x')%popisek osy x
ylabel('y =f(x)')
axis(axis+[0 0 0 1]);

set(findobj('Tag','Import_dat'),'UserData',data);%uloží výsledek do úložného prostoru
UserData
set(findobj('Tag','Export_dat'),'Enable','off');%nastavení viditelnosti tlačítka
set(findobj('Tag','Vykreslit'),'Enable','on');
end
```

V subfunkci Export dat se pomocí funkce get načtou vykreslená data z úložného prostoru objektu Vykreslit a pomocí funkce save se tyto data uloží do zvoleného souboru.

Kód subfunkce Export_dat:

```
function Export_dat_Callback(hObject, eventdata, handles)
% hObject    handle to Export_dat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%vyskakovací okno pro export dat
[jmenosouboru2, jmenocesty2, filterindex2] = uiputfile({ '*.m', 'M-files (*.m)';
'*.mat','MAT-files (*.mat)';'*.*',  'All Files (*.*)'},'Uložit data','DefaultName');

if jmenosouboru2~=0
A=get(findobj('Tag','Vykreslit'),'UserData');%načtení výsledků z úložného prostoru UserData
do A
save(strcat(jmenocesty2,jmenosouboru2),'-ascii','-double','-tabs','A');%exportování
výsledků do zvoleného souboru
end
```

V subfunkci Vykreslit se po načtení a kontrole parametrů, pomocí funkce get načtou vykreslená data z úložného prostoru objektu Import_dat. Do vektoru x a y se uloží celé jednotlivé sloupce textového souboru. Z jednotlivých bodů, které jsou tvořeny dvojicí vektorů x a y, se funkcí polyfit najdou koeficienty polynomu pro daný stupeň aproximace. Z vektoru x se pomocí funkce pro hledání maxima a minima vytvoří vektor hodnot, který je složen z počtu vorků, který uživatel zadá. Tento vektor spolu s koeficienty polynomu tvoří vstupní hodnoty pro funkci polyval, která proloží načtená data aproximačním polynomem. Tyto výsledky se zobrazí do grafu a uloží se do vlastnosti User data objektu Vykreslit, které slouží jako úložný prostor pro vykreslená data. V MATLABU jsou vektory hodnot řazeny do řádků, proto se musí před uložením zaměnit řádky za sloupce, protože v textovém souboru se vektory hodnot řadí pro přehlednost do sloupců.

Kód subfunkce Vykreslit:

```
function Vykreslit_Callback(hObject, eventdata, handles)
% hObject    handle to Vykreslit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%načtení textu z editačních políček
pbstr=get(findobj('Tag','Pocet_bodu'),'String');
spstr=get(findobj('Tag','Stupen_polynomu'),'String');

%převod textu na číslo
[pb, status] = str2num(pbstr);
[sp, status2] = str2num(spstr);

%kontrola správnosti zadávaných parametů aproximace(zda se jedná o číslo)
if status~=1 || pb<=0
set(findobj('Tag','Pocet_bodu'),'String','Nelze');
end
if status2~=1 || sp<0
set(findobj('Tag','Stupen_polynomu'),'String','Nelze');
end

if status==1 && pb>0 && status2==1 && sp>=0 %je-li vše v pořádku provede se vykreslení

    data=get(findobj('Tag','Import_dat'),'UserData');%do data se uloží importovaná data
    ze zvoleného souboru
    x=data(:,1);%do vektoru x se uloží 1.sloupec z matice data
    y=data(:,2);
    p=polyfit(x,y,sp);%výpočet koeficientů polynomu
    xmin=min(x);
    xmax=max(x);
    xp=xmin:(xmax-xmin)/(pb):xmax;%vektor hodnot osy x, který je složen z pb vzorků
    yp=polyval(p,xp);%výpočet funkčních hodnot aproximačního polynomu pro hodnoty
    vektoru xp

    %zobrazeni grafů
    axes(handles.graf1);
    plot(xp,yp,'b',x,y,'.r')%vykreslení grafů
    xlabel('x')%popisek osy x
    ylabel('y =f(x)')

    A=[xp(:),yp(:)];%vytvoření matice A se záměnou řádků za sloupců
    set(findobj('Tag','Vykreslit'),'UserData',A);%uložení matice A do úložného prostoru
    UserData

    %vytvoření popisků regresní rovnice
    str='y=';
    str=strcat(str,num2str(p(sp+1),3));
    for k=sp:-1:1
        if p(k)>=0
        str=strcat(str,'+');
        end
        str=strcat(str,num2str(p(k),3));
        str=strcat(str,'x');
        str1=num2str(sp+1-k);
        if sp+1-k~=1
        str=strcat(str,'^',str1);
        end
    end

set(findobj('Tag','Regresni_rovnice'),'String',str)%zápis regresní rovnice
set(findobj('Tag','Export_dat'),'Enable','on');%nastavení viditelnosti tlačítka

end
```

Celý kód programu je uveden v příloze, kde v komentářích je zhruba vysvětlena funkce jednotlivých příkazů. A nakonec se aplikace odzkouší kliknutím na „zelenou šipku“ Run PolynomialniAproximace.m.

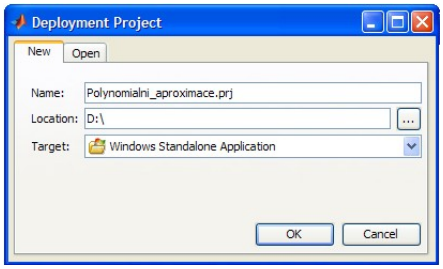
5.2 Tvorba samostatně spustitelné aplikace

Z hotové aplikace se může udělat samostatně spustitelná aplikace dvěma způsoby:

1. pomocí říkazu **mcc**,
2. pomocí **Deployment Tool** rozhraní.

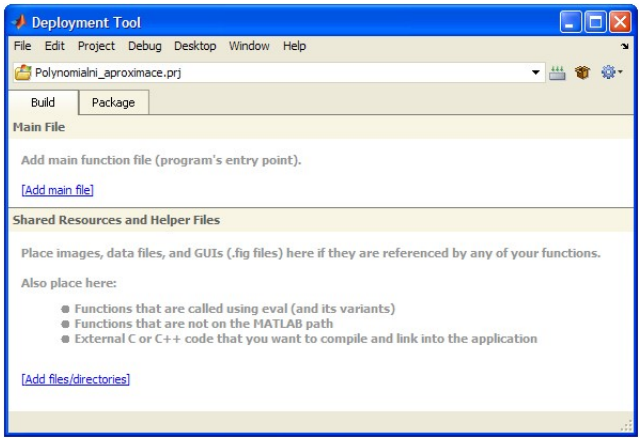
Druhý způsob je snadnější a přehlednější. Umožňuje aplikaci jednoduše zkompilovat do vybraného formátu a snadno aplikaci s dalšími podporujícími soubory, včetně MCR, zabalit do jednoho souboru.

Nyní bude popsána tvorba samostatně spustitelné aplikace pomocí rozhraní Deployment Tool. Nejdříve se otevře dialogové okno Deployment Project (Obrázek 7), které se zobrazí pomocí zadáním příkazu **deploytool** do příkazové řádky nebo přes nabídku menu **File→New→ Deploymentool project**. Zde zadáme jméno projektu, umístění projektu a vybereme cíl kompilace.



Obrázek 7 – Deployment projekt

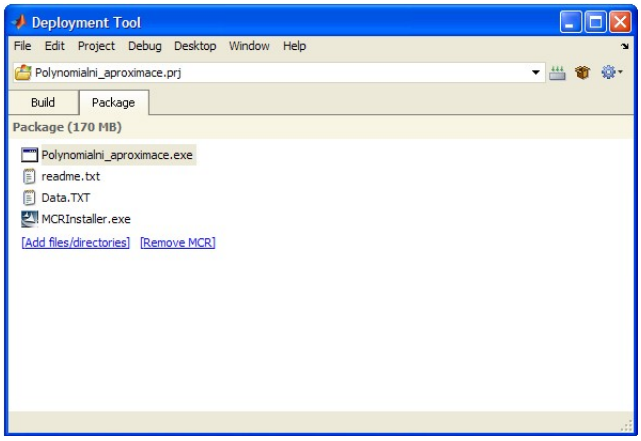
Potvrzením tlačítka OK se zobrazí dialogové okno Deployment Tool (Obrázek 8). V záložce „Build“ kliknutím na odkaz [Add main file] se vybere hlavní m-funkce, která má být zkompilována. Odkazem [Add files/directories] se vybírají obrázky, datové soubory a GUIs (.fig soubory), které jsou zmíněny v jakékoli z vašich kompilovaných funkcí. A kliknutím na tlačítko Build se tyto soubory zkompilují do jedné aplikace.



Obrázek 8 – Deployment Tool – Build

Nyní se může přistoupit k zabalení zkompilované aplikace společně s dalšími soubory. V záložce „Package“ (Obrázek 9) kliknutím na odkaz [Add files/directories] se vyberou další podpůrné soubory k zabalení. Odkazem [Add MCR] nebo [Remove MCR] se

může přidat respektive odebrat soubor MCRInstaller.exe k zabalení, kterým si koncový uživatel nainstaluje na svůj počítač potřebný MCR. Soubor readme.txt, ve kterém najdeme potřebné informace k instalaci samostatné aplikace, a zkompilovaná aplikace (Polynommialni_aproximace.exe) se automaticky přidá k ostatním souborům. Vše se nakonec zabalí kliknutím na tlačítko Package a aplikace je připravena k distribuci.



Obrázek 9 – Deployment Tool – Package

6 Závěr

MATLAB je užitečný nástroj při tvorbě samostatně spustitelných aplikací nebo sdílených knihoven, které spustíme bez nutnosti instalovat samotný MATLAB, který zabírá okolo 4 GB (při plné instalaci verze 2009b). MATLAB usnadňuje práci při vytváření svých aplikací díky maticově orientovaném" MATLAB jazyku a vestavěným matematickým, grafickým, datovým funkcím a funkcím pro analýzu. Pak stačí mít nainstalovaný MATLAB Compiler, který převede aplikaci nebo funkci na samostatně spustitelnou aplikaci nebo sdílenou knihovnu, kterou můžeme využít při psaní programů v jiném programovacím jazyce. Zkompilovat se dá většina MATLAB funkcí a toolboxů. Nevýhodou je potřeba instalovat Matlab Compiler Runtime, který umožní spustit samostatné MATLAB aplikace na počítačích bez nainstalované verze MATLABu. MCR po rozbalení dosahuje velikosti 465 MB.

MCR se nemusí instalovat, stačí když si instalaci zkopírujeme na přenosný flash disk nebo CD-ROM. Potom musíme přidat MCR adresář do systémových cest otevřením příkazového řádku a vykonáním příkazu:

```
set PATH=<cesta k MCR adresáři>\v711\runtime\win32;%PATH%
```

Aby se tímto uživatel nemusel zabývat, může se vytvořit jednoduchý dávkový soubor (textový soubor s příponou .BAT), obsahující řádkové příkazy DOSu nebo jiného operačního systému. Tyto příkazy se provedou po spuštění daného souboru. Takový soubor může vypadat nějak takto:

```
set PATH=X:\MCR\v711\runtime\win32;%PATH%
PolynomialniAproximace.exe
```

Literatura

[1] *Humusoft : MATLAB* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.humusoft.cz/produkty/matlab/matlab/>.

[2] BOTEK, M. *Sbírka příkladů z inženýrské ekonomiky a managementu* [online]. Praha : Vysoká škola chemicko-technologická v Praze, 2004 [cit. 2010-04-17]. Korelační a regresní analýza, s. 16. Dostupné z WWW: <http://vydavatelstvi.vscht.cz/knihy/uid_isbn-80-7080-544-7/pages-img/016.html>. ISBN 80-7080-544-7.

[3] *Wikipedia : Lineární regrese* [online]. 2010-21-3 [cit. 2010-04-17].. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Line%C3%A1rn%C3%AD_regrese>.

[4] REKTORYS, Karel. *Přehled užité matematiky II*. Praha : Prometheus, 1995. 874 s. ISBN 80-85849-62-3.

[5] *Humusoft : MATLAB Compiler* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.humusoft.cz/produkty/matlab/aknihovny/compiler>.

[6] *The Mathworks : MATLAB Compiler 4.13* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.mathworks.com/products/compiler/description1.html>.

[7] *The Mathworks : The MATLAB Application Deployment Products* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/bselcf1-1.html>.

[8] *The Mathworks : MATLAB Compiler 4.13* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.mathworks.com/products/compiler/description3.html>.

[9] *The Mathworks : MATLAB Compiler 4.13* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.mathworks.com/products/compiler/description4.html>.

[10] *The Mathworks : MATLAB Compiler 4.13* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: < http://www.mathworks.com/products/compiler/compiler_support.html>.

[11] *The Mathworks : Unsupported Functions* [online]. 2010 [cit. 2010-04-17]. Dostupné z WWW: <http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/br2cqa0-20.html>.

[12] ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB – tvorba uživatelských aplikací*. Praha : BEN, 2004. 133 s. ISBN 80-7300-133-0.

Přílohy

Všechny přílohy, včetně zdrojových kódů a bakalářské práce v elektronické podobě jsou k dispozici na přiloženém CD.