Knihovna pro práci s maticemi pro mikropočítač Motorola

Pavel Doležel

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel DOLEŽEL**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Knihovna pro práci s maticemi pro mikropočítač Motorola**

Zásady pro vypracování:

1. **Porovnejte možnosti assembleru a CodeWarrioru při programování mikropočítačů**
2. **Popište možné využití modulu v automatizační technice**
3. **Vytvořte modul pro počítání s libovolným tvarem matic**
4. **Vytvořte funkce pro operace s maticemi**
5. **Zajistěte kompatibilitu s existujícími moduly**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Pack, J. D., Barrett, S. F.: 68HC12 Microcontroller – Tudory and application, Prentice Hall 2002. ISBN 0-13-033776-5
2. Morton, T. D.: Embedded Microcontrollers, Prentice Hall 2001. ISBN 80-7300-077-6
3. Burkhard, M.: C pro mikrokontroléry, Ben 2003. ISBN 80-7300-077-6

Vedoucí bakalářské práce: **Ing. Petr Doležel**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **14. února 2006**

Termín odevzdání bakalářské práce: **16. června 2006**

Ve Zlíně dne 14. února 2006

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Hlavním účelem této bakalářské práce byl požadavek na vytvoření knihovny pro práci s maticemi typu MxN, tzn. matic obdélníkových, pro mikropočítač Motorola HC12. Základními požadavky pro tuto knihovnu byla schopnost provádět jejich elementární úpravy jako je jejich vzájemné sčítání, odečítání, a násobení, dále pak výpočet determinantu a inverzní matice k matici původní. V teoretické části je proto popsána teorie práce s maticemi, včetně názorných příkladů, v části praktické pak popis prostředí Metrowerks CodeWarioru a popis deklarací všech funkcí použitých v projektu.

Klíčová slova: matice, determinant, inverzní matice, CodeWarrior, assembler

ABSTRACT

The main reason of this bachelor´s work was the assignment to make a module for working with matrixes of the type MxN for microcontrollers Motorola. The main idea for this module was that it could do the elementary functions of addition, subtraction, multiplication and also calculate the determinant and inverse matrix. In the whole theoretical part is therefore written the theory of working with matrixes with examples, and in the practical part the description of how Metrowerks CodeWarrior was used, and also a description of all the functions used in the project.

Keywords: matrix, determinant, inverse matrix, CodeWarrior, assembler

Děkuji mému vedoucímu bakalářské práce Ing. Petru Doleželovi za odborné vedení a pomoc v průběhu řešení této bakalářské práce

OBSAH

ÚVOD

Teorie matic a determinantů představuje úvod do lineární algebry. Pojem determinantu zavedl již v roce 1693 německý matematik W. G. Leibniz (1646–1716), ale jeho objev upadl v zapomenutí. V roce 1750 dospěl znovu k pojmu determinantu švýcarský matematik G. Cramer (1704–1752). Všeobecně se začalo v matematice používat determinantů až koncem 18. století. Zasloužili se o to zejména matematici A.T. Vandermonde (1735–1796) a A. L. Cauchy (1789–1857). Současně s teorií determinantů se rozvíjela teorie matic, jejímž zakladatelem je anglický matematik A.Cayley (1821–1895). Na dalším rozvoji teorie matic se podíleli zejména G. Frobenius (1849–1917), J. J. Sylvester (1814–1897) a K. Weierstrass (1815–1897).

Využití práce s maticemi, teorie matic a determinantů, počítání s nimi a uplatnění jejich výpočtů v praxi je velice široké. Nejrozsáhlejší aplikace mají matice a determinanty při řešení systémů lineárních rovnic, ale také v teorii automatického řízení při identifikaci soustav nebo v prediktivním řízení.

Tato bakalářská práce si klade za cíl nejprve teoreticky shrnout jejich vlastnosti, zavést matematické pojmy týkající se teorie matic a také vysvětlit základní matematické operace s maticemi.

Dalším, a hlavním cílem této práce bylo naprogramování knihovny pro mikropočítače Motorola, která bude obsahovat funkce pro operace s maticemi s libovolným tvarem, tzn. jak s maticemi čtvercovými tak obdélníkovými a zajistit kompatibilitu s moduly již existujícími. K dosažení tohoto cíle bylo možné použít buď přímo jazyka symbolických adres – assembleru, nebo jednoho z výborných nástrojů pro programování mikropočítačů Motorola, a to programu Metrowerks CodeWarrior. Díky výhodám, které jsou v projektu důkladněji popsány a mezi které patří především možnost vytváření zdrojového kódu v jazyce C a s tím související snadnější programování a především ladění programu, jsem zvolil právě tento vývojový nástroj.

I. TEORETICKÁ ČÁST

1 TEORIE MATIC

1.1 Základní maticové pojmy

1.1.1 Číselné těleso

Číselným tělesem je uspořádaná trojice $(T,*,\circ)$, kde T je podmnožina množiny komplexních čísel C taková, že $0 \in T$, $1 \in T$ a platí:

$(\forall x, y \in T)(x + y \in T \wedge x \cdot y \in T)$ (je uzavřená na sčítání a násobení)

$(\forall x \in T)(-1) \cdot (x) \in T$ (je uzavřená na opačné prvky)

$(\forall x, y \in T)(x \neq 0 \Rightarrow \frac{1}{x} \in T)$ (je uzavřená na převrácené hodnoty nenulových prvků)

Obecně se číselným tělesem rozumí každá uspořádaná trojice $(T,*,\circ)$, kde T je aspoň dvouprvková množina; $*$, $\circ$ jsou operace na T a platí $(x, y, z \in T)$:

$$x * y = y * x \qquad x \circ y = y \circ x \tag{1}$$

$$(x * y) * z = x * (y * z) \qquad (x \circ y) \circ z = x \circ (y \circ z) \tag{2}$$

$$(\exists 0 \in T)(\forall x)\ 0 * x = x \qquad (\exists 1 \in T)(\forall x)\ 1 \circ x = x \tag{3}$$

$$(\forall x)(\exists -x \in T)\ x * (-x) = 0 \qquad (\forall x \neq 0)(\exists x^{-1} \in T) 1 \circ x^{-1} = 1 \tag{4}$$

$$(x * y) \circ z = (x \circ z) * (y \circ z) \tag{5}$$

1.1.2 Matice

Obdélníková tabulka prvků z tělesa T sestavených do m řádků a n sloupců se nazývá matice typu (m, n) nad tělesem T. Jestliže $m = n$, hovoří se o čtvercové matici n-tého řádu.

Matice $\underline{A}$ přiřazuje každé dvojici (i,k), $i = 1,2,...,m$, $k = 1,2,...,n$ prvek z T, který se označuje a_{ik} a nazývá se prvkem matice $\underline{A}$ v i-tém řádku a k-tém sloupci. Matice $\underline{A}$ se zapisuje

$$\underline{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \tag{6}$$

nebo zkráceně

$$\underline{A} = (a_{ik})_{\substack{i=1\ldots m, \\ k=1\ldots n}} \tag{7}$$

pokud je z textu známo *m,n*, píše se pouze $\underline{A} = (a_{ik})$. Aritmetický vektor (a_{i1}, a_{i2}, a_{in}) se nazývá i-tý řádek, aritmetický vektor (a_{1k}, a_{2k}, a_{mk}) k-tý sloupec matice $\underline{A}$.

1.1.3 Diagonála a diagonální matice

$\underline{A} = (a_{ik})$ je matice typu *(m,n).* Aritmetický vektor $(a_{11},a_{22},\ldots,a_{rr})$, kde $r = min(m,n)$, se nazývá (hlavní) diagonála matice $\underline{A}$. Prvky a_{ii}, $i = 1,2,\ldots,r$, se nazývají diagonální prvky. Matice $\underline{A} = (a_{ik})$, která má mimo hlavní diagonálu samé 0, tj. $a_{ik} = 0$ pro $i \neq k$, je matice diagonální.

1.1.4 Jednotková matice

Jednotková matici $\underline{A}$ n-tého řádu označím jako diagonální matici $\underline{E} = (e_{ik})$ n-tého řádu, pro níž platí $e_{ii} = 1$ pro všechna $i = 1,2,\ldots,n$. (Jednotková matice stupně *n* je čtvercová matice $\underline{E} = (e_{ik})$ stupně *n* mající v hlavní diagonále všude prvek 1 a všude jinde prvek 0.)

1.1.5 Transponovaná matice

Transponovaná matice k matici $\underline{A} = (a_{ik})$ typu *(m,n)* je matice $\underline{B}^T = (b_{ik})$ typu *(n,m),* pro kterou platí $a_{ik} = b_{ik}$, $i = 1,2,\ldots,m$, $k = 1,2,\ldots,n$. (Transponovanou matici $\underline{A}^T$ dostaneme z matice $\underline{A}$ tak, že vzájemně vyměníme řádky a sloupce v matici $\underline{A}$).

1 Příklad

Výpočet transponované matice z matice původní

$$\underline{A} = \begin{pmatrix} 2 & 4 & -1 \\ 5 & -2 & -3 \\ 4 & 0 & 1 \end{pmatrix}, \quad \underline{A}^T = \begin{pmatrix} 2 & 5 & 4 \\ 4 & -2 & 0 \\ -1 & -3 & 1 \end{pmatrix} \tag{8}$$

1.1.6 Matice symetrická a antisymetrická

Matice A je symetrická (antisymetrická), jestliže platí $\underline{A} = \underline{A^T}$ $(A = -A^T)$

1.1.7 Trojúhelníková matice

zobecněná : právě když matice $\underline{A} = (a_{ik})$ typu (m, n)

- má pouze nenulové řádky,
- jsou-li a_{ik}, a_{rs} vedoucí prvky takové, že $i < r$, pak $k < s$.

redukovaná : právě když je u zobecněné trojúhelníková matice $\underline{A} = (a_{ik})$ typu (m,n)

- každý vedoucí prvek je roven 1,
- nad každým vedoucím prvkem jsou ve sloupci pouze 0.

1.1.8 Čtvercová matice regulární x singulární

Čtvercová matice $\underline{A}$ n-tého řádu je **regulární** jestliže $hod\ \underline{A} = n$,

čtvercová matice $\underline{A}$ n-tého řádu je **singulární** jestliže $hod\ \underline{A} < n$.

(Regulární matice je matice čtvercová n-tého řádu, jejíž hodnost je rovna n. V opačném případě jde o singulární matici.)

Matice $\underline{A}$ je regulární je-li determinant $\det \underline{A} \neq 0$. K regulární matici existuje inverzní matice.

1.1.9 Řádkový prostor

Řádkový prostor matice $\underline{A}$ je podprostor vektorového prostoru $V_n\ (T)$ generovaný všemi řádky matice $\underline{A}$.

1.1.10 Hodnost matice

Hodnost matice $\underline{A} = (a_{ik})$ typu (n,m) je dimenze jejího řádkového prostoru

1.1.11 Elementární úpravy

Elementární řádkové (sloupcové) úpravy:

- změna pořadí řádků (resp.sloupců) matice $\underline{A}$
- nahrazení řádku (resp. sloupce) matice $\underline{A}$ jeho α-násobkem, kde $\alpha \in T$, $\alpha \neq 0$.
- nahrazení řádku (resp.sloupce) matice $\underline{A}$ jeho součtem s α-násobkem, $\alpha \in T$, jiného řádku matice $\underline{A}$.
- vynechání řádku (resp.sloupce), který je lineární kombinací ostatních řádků (resp.sloupců)

Dvě matice jsou ekvivalentní tehdy, když lze jednu z druhé získat konečným počtem elementárních úprav řádků.

2 Příklad

Určení hodnosti matice

$$\underline{A} = \begin{pmatrix} 3 & -2 & 1 & 0 \\ 4 & 0 & 2 & -3 \\ 11 & -4 & 13 & -1 \\ 2 & -1 & 5 & 1 \end{pmatrix} \tag{9}$$

Pomocí elementárních řádkových a sloupcových úprav se matice matici A převede na ekvivalentní zobecněnou trojúhelníkovou matici:

$$\begin{pmatrix} 3 & -2 & 1 & 0 \\ 4 & 0 & 2 & -3 \\ 11 & -4 & 13 & -1 \\ 2 & -1 & 5 & 1 \end{pmatrix} \approx \begin{pmatrix} -1 & -2 & 1 & 0 \\ 4 & 0 & 2 & -3 \\ 11 & -4 & 13 & -1 \\ 2 & -1 & 5 & 1 \end{pmatrix} \approx \begin{pmatrix} -1 & -2 & 5 & 1 \\ 0 & -1 & -9 & -2 \\ 0 & 4 & 2 & -3 \\ 0 & 5 & 11 & -1 \end{pmatrix} \approx \tag{10}$$

$$\begin{pmatrix} -1 & -2 & 5 & 1 \\ 0 & -1 & -9 & -2 \\ 0 & 0 & -34 & -11 \\ 0 & 0 & -34 & -11 \end{pmatrix} \approx \begin{pmatrix} -1 & 2 & 5 & 1 \\ 0 & -1 & -9 & -2 \\ 0 & 0 & -34 & -11 \end{pmatrix}$$

Hodnost matice A je 3

1.2 Matematické operace s maticemi

1.2.1 Rovnost matic

Matice $\underline{A} = (a_{ik})$ typu $()m,n$ a $\underline{B} = (b_{ik})$ typu (r,s) se sobě rovnají, právě když platí: $m = r$, $n = s$, $a_{ik} = b_{ik}$ pro všechna $i = 1,2,...,m$, $k = 1,2,\ldots,n$.

1.2.2 Součet matic

Jsou dány matice $\underline{A} = (a_{ik})$, $\underline{B} = (b_{ik})$ stejného typu *(m,n)* nad stejným tělesem *T*. Součtem matic $\underline{A}$ a $\underline{B}$ je matice $\underline{C} = (c_{ik})$ typu (m,n) definovaná předpisem

$$c_{ik} = a_{ik} + b_{ik},\ i=1,2,...,m,\ k=1,2,...,n \tag{11}$$

Píšeme $\underline{C} = \underline{A} + \underline{B}$.

Platí:

- $\underline{A} + \underline{B} = \underline{B} + \underline{A}$
- $\underline{A} + (\underline{B} + \underline{C}) = (\underline{A} + \underline{B}) + \underline{C}$
- $\underline{A} + 0 = 0 + \underline{A} = \underline{A}$
- $(\underline{A} + \underline{B})^T = \underline{A}^T + \underline{B}^T$
- $-\underline{A} = (-a_{ik})$, kde $-a_{ik}$ je opačný prvek k a_{ik} v tělese T, $-\underline{A}$ je matice opačná k matici $\underline{A}$, tj. platí $-\underline{A} + \underline{A} = 0$.

3 Příklad

Součet matic $\underline{A}$ a $\underline{B}$

$$\underline{A} = \begin{pmatrix} 2 & 4 & -1 \\ 5 & -2 & -3 \\ 4 & 0 & 1 \end{pmatrix}, \quad \underline{B} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -2 & 1 \\ -3 & 6 & -3 \end{pmatrix} \tag{12}$$

$$\underline{A} + \underline{B} = \begin{pmatrix} 2+1 & 4+2 & -1+3 \\ 5+0 & -2-2 & -3+1 \\ 4-3 & 0+6 & 1-3 \end{pmatrix} = \begin{pmatrix} 3 & 6 & 2 \\ 5 & -4 & -2 \\ 1 & 6 & -2 \end{pmatrix}$$

1.2.3 α-násobek matice

$\underline{A} = (a_{ik})$ je matice typu *(m,n)* nad tělesem *T*. Součinem prvků $\alpha \in T$ a matice $\underline{A}$ je matice $\underline{C} = (c_{ik})$ typu *(m,n)* definovaná předpisem

$$c_{ik} = \alpha \cdot a_{ik},\ i=1,2,...,m,\ k=1,2,...,n \tag{13}$$

Píšeme $\underline{C} = \alpha \cdot \underline{A}$

4 Příklad

Výpočet $\underline{D} = -2\underline{A} + 3\underline{B}$

$$\underline{A} = \begin{pmatrix} 2 & 4 & -1 \\ 5 & -2 & -3 \\ 4 & 0 & 1 \end{pmatrix}, \quad \underline{B} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -2 & -1 \\ -3 & 6 & -3 \end{pmatrix} \tag{14}$$

$$\underline{D} = \begin{pmatrix} -2\cdot 2 & -2\cdot 4 & -2\cdot(-1) \\ -2\cdot 5 & -2\cdot(-2) & -2\cdot(3) \\ -2\cdot 4 & -2\cdot 0 & -2\cdot 1 \end{pmatrix} + \begin{pmatrix} 3\cdot 1 & 3\cdot 2 & 3\cdot 3 \\ 3\cdot 0 & 3\cdot(-2) & 3\cdot 1 \\ 3\cdot(-3) & 3\cdot 6 & 3\cdot(-3) \end{pmatrix} =$$

$$= \begin{pmatrix} -4 & -8 & 2 \\ -10 & 4 & 6 \\ -8 & 0 & -2 \end{pmatrix} + \begin{pmatrix} 3 & 6 & 9 \\ 0 & -6 & 3 \\ -9 & 18 & -9 \end{pmatrix} = \begin{pmatrix} -1 & -2 & 11 \\ -10 & -2 & 9 \\ -17 & 18 & -11 \end{pmatrix}$$

1.2.4 Součin matic

Matice $\underline{A} = (a_{ik})$ je typu (m,n) a matice $\underline{B} = (b_{ik})$ typu (n,p), obě nad stejným tělesem T. Součinem matic $\underline{A}$, $\underline{B}$ (v tomto pořadí!) je matice $\underline{C} = (c_{ik})$ typu (m,p) definovaná předpisem

$$c_{ik} = a_{i1}b_{ik} + a_{i2}b_{2k} + \ldots + a_{in}b_{nk} = \sum_{j=1}^{n} a_{ij}b_{jk}, i = 1,2,\ldots,m, k = 1,2,\ldots,p \tag{15}$$

Píšeme $\underline{C} = \underline{A}\cdot\underline{B}$. (Podmínku pro typy matic při násobení si můžeme zapamatovat pomocí formálního vztahu $(m,n)(n,p) = (m,p)$). Násobení matic není komutativní !

5 Příklad

Výpočet součinu matic $\underline{C} = \underline{A}\cdot\underline{B}$

$$\underline{A} = \begin{pmatrix} 2 & 4 & -1 \\ 5 & -2 & -3 \\ 4 & 0 & 1 \end{pmatrix}, \quad \underline{B} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -2 & -1 \\ -3 & 6 & -3 \end{pmatrix} \tag{16}$$

$a_{11} = 2 \cdot 1 + 4 \cdot 0 + (-l) \cdot (-3) = 5$
$a_{12} = 2 \cdot 2 + 4 \cdot (-2) + (-l) \cdot 6 = -10$
$a_{13} = 2 \cdot 3 + 4 \cdot 1 + (-l) \cdot (-3) = 13$
$a_{21} = 5 \cdot 1 + (-2) \cdot 0 + (-3) \cdot (-3) = 14$
$a_{22} = 5 \cdot 2 + (-2) \cdot (-2) + (-3) \cdot 6 = -4$
$a_{23} = 5 \cdot 3 + (-2) \cdot 1 + (-3) \cdot (-3) = 22$
$a_{31} = 4 \cdot 1 + 0 \cdot 0 + l \cdot (-3) = 1$
$a_{32} = 4 \cdot 2 + 0 \cdot (-2) + 1 \cdot 6 = 14$
$a_{33} = 4 \cdot 3 + 0 \cdot 1 + l \cdot (-3) = 9,$ tedy

$$\underline{A} \cdot \underline{B} = \begin{pmatrix} 5 & -10 & 13 \\ 14 & -4 & 22 \\ 1 & 14 & 9 \end{pmatrix}$$

1.2.5 Adjungovaná matice

Adjungovanou matici *adj* $\underline{A}$ vypočítáme tak, že každý prvek matice $\underline{A}$ nahradíme jeho algebraickým doplňkem a takto získanou matici transponujeme:

6 Příklad

Určení adjungované matice *adj* $\underline{A}$ k matici $\underline{A}$

$$\underline{A} = \begin{pmatrix} 2 & 4 & -1 \\ 5 & -2 & -3 \\ 4 & 0 & 1 \end{pmatrix} \tag{17}$$

$$adj\, A = \begin{pmatrix} \begin{vmatrix} -2 & -3 \\ 0 & 1 \end{vmatrix} & \begin{vmatrix} 5 & -3 \\ 4 & 1 \end{vmatrix} & \begin{vmatrix} 5 & -2 \\ 4 & 0 \end{vmatrix} \\ -\begin{vmatrix} 4 & -1 \\ 0 & 1 \end{vmatrix} & \begin{vmatrix} 2 & -1 \\ 4 & 1 \end{vmatrix} & -\begin{vmatrix} 2 & 4 \\ 4 & 0 \end{vmatrix} \\ \begin{vmatrix} 4 & -1 \\ -2 & -3 \end{vmatrix} & \begin{vmatrix} 2 & -1 \\ 5 & -3 \end{vmatrix} & \begin{vmatrix} 2 & 4 \\ 5 & -2 \end{vmatrix} \end{pmatrix}^T = \begin{pmatrix} -2 & -17 & 8 \\ -4 & 6 & 16 \\ -14 & 1 & -24 \end{pmatrix}^T = \begin{pmatrix} -2 & -4 & -14 \\ -17 & 6 & 1 \\ 8 & 16 & -24 \end{pmatrix}$$

1.2.6 Determinant matice

$\underline{A} = (a_{ij})$ je čtvercová matice n-tého řádu nad tělesem *T*. Determinant matice $\underline{A}$ je prvek (číslo) det $\underline{A}$ z tělesa *T*, pro který platí:

$$\det \underline{A} = \sum_{\pi \in S_n} sign \begin{pmatrix} 1 & 2 & \dots & a_{1n} \\ s_1 & s_2 & \dots & s_n \end{pmatrix} a_{1s_1} a_{2s_2} \dots a_{ns_n} \tag{18}$$

Místo det $\underline{A}$ se někdy píše | A |. Pro n = 2,3 lze definiční vztah snadno rozepsat a upravit na tvar:

$$\det \underline{A} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \tag{19}$$

a

$$\det \underline{A} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \tag{20}$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - (a_{13}a_{22}a_{31} + a_{12}a_{21}a_{33} + a_{11}a_{23}a_{32})$$

Výpočet podle vztahu (19) resp. (20) se nazývá výpočet podle Sarrusova pravidla podle francouzského matematika P.F.Sarruse (1798 – 1858)

7 Příklad

Vypočet determinantu matice $\underline{A}$ podle Sarrusova pravidla:

$$A = \begin{pmatrix} 2 & -3 & 8 \\ 4 & 6 & -7 \\ -5 & 4 & -9 \end{pmatrix} \tag{21}$$

$$\det \underline{A} = \left[2 \cdot 6 \cdot (-9) + (-3)(-7)(-5) + 4 \cdot 4 \cdot 8\right] - \left[8 \cdot 6 \cdot (-5) + 4(-3)(-9) + 4(-7)3\right] =$$
$$= (-108 - 105 - 128) - (-240 - 56 + 108) = -85 + 188 = 109$$

Pro výpočet determinantu matice jejíž rozměr je n > 3 musíme použít elementární maticové transformace. Následující tvrzení popisují vlastnosti determinantu, které jsou důležité pro jeho výpočet. Zejména popisují, jaký vliv má provedení jednotlivých transformací na hodnotu determinantu:

- Determinant matice $\underline{A}$ je součet součinů; v každém součinu se vyskytuje z každého řádku i sloupce právě jeden prvek. Na druhé straně každý prvek řádku či sloupce se vyskytuje aspoň v jednom sčítanci.

- Determinant trojúhelníkové matice je roven součinu prvků na diagonále
- Vznikne-li matice B ze čtvercové matice $\underline{A}$ n-tého řádu výměnou dvou řádků, resp. sloupců, potom $\det \underline{B} = -\det \underline{A}$.
- Platí $\det \underline{A} = det\, \underline{A}^T$.
- Věta o součtu determinantů:

$$\det(\underline{a}_1, \underline{a}_2, ..., \underline{a}_{i-1}, \underline{a}_i + \underline{b}_i, \underline{a}_{i+1}, ..., \underline{a}_n) = \det(\underline{a}_1, ..., \underline{a}_i, ..., \underline{a}_n) + \det(\underline{a}_1, ..., \underline{b}_i, ..., \underline{a}_n) \quad (22)$$

- Věta o vytýkání konstanty ze řádku:

$$\det(\underline{a}_1, \underline{a}_2, ..., \underline{a}_{i-1}, \alpha\underline{a}_i, \underline{a}_{i+1}, ..., \underline{a}_n) = \alpha \det(\underline{a}_1, \underline{a}_2 ..., \underline{a}_{i-1}, \underline{a}_i, \underline{a}_{i+1}, ..., \underline{a}_n) \quad (23)$$

- $\underline{A}$ je čtvercová matice stupně n . Jestliže matice $\underline{B}$ vznikne z matice $\underline{A}$ vynásobením libovolného řádku prvkem $c \in T$, pak $\det \underline{B} = c \cdot \det \underline{A}$.
- Věta o součinu dvou determinantů

  $\det (\underline{AB}) = \det \underline{A} \cdot \det \underline{B}$

- Hodnota determinantu se nezmění, jestliže k danému řádku, resp. sloupci přičteme libovolnou lineární kombinaci ostatních řádků, resp. sloupců.
- Determinant regulární (singulární) matice je vždy různý od nuly (roven nule)

Tato tvrzení poskytují jednoduchý návod, jak hodnotu determinantu určit. Pomocí elementárních transformací se matice převede na dolní trojúhelníkovou matici. Jednotlivé transformace sice mohou měnit hodnotu determinantu, ale díky těmto tvrzením je zřejmé k jakým změnám dojde. Hodnota determinantu je pak rovna součinu prvků na hlavní digonále.

Výpočet determinantu lze provést i pomocí následující metody: $\underline{A} = (a_{ij})$ je matice typu (m,n). Každou matici $\underline{B}$, která vznikne z $\underline{A}$ vynecháním některých (libovolných) řádků a některých sloupců, se nazývá dílčí maticí matice $\underline{A}$. Determinant každé čtvercové dílčí matice se nazývá subdeterminantem matice $\underline{A}$. Je-li $\underline{A}$ čtvercová matice stupně n, pak vynecháním libovolných k řádků, $k < n$, a libovolných k sloupců z matice $\underline{A}$ dostaneme dílčí čtvercovou matici stupně $n - k$. Determinant každé takové dílčí matice se nazývá subdeterminantem matice $\underline{A}$ stupně $n - k$. Subdeterminant stupně $n - 1$ vzniklý vynecháním i-tého řádku a j-tého sloupce v matici $\underline{A}$ označíme A_{ij}. Prvek $D_{ij} = (-1)^{i+j} M_{ij}$ se nazývá algebraickým doplňkem prvku a_{ij} . Potom platí:

$$\det A = \sum_{k=1}^{n} a_{ik} D_{ik} = \sum_{k=1}^{n} (-1)^{i+k} a_{ik} A_{ik} \tag{24}$$

a

$$\det A = \sum_{k=1}^{n} a_{kj} D_{kj} = \sum_{k=1}^{n} (-1)^{k+j} a_{kj} A_{kj} \tag{25}$$

Vztah (24) se nazývá rozvoj podle *i*-tého řádku, a vztah (25) rozvoj podle *j*-tého sloupce. Celkově se tyto vztahy nazývají Laplaceovy věty.

8 Příklad

Výpočet determinantu pomocí elementárních transformací

$$A = \begin{pmatrix} 2 & -3 & 8 \\ 4 & 6 & -7 \\ -5 & 4 & 9 \end{pmatrix} \tag{26}$$

$$\det A = \begin{vmatrix} 2 & -3 & 8 \\ 4 & 6 & -7 \\ -5 & 4 & 9 \end{vmatrix} = \frac{1}{2} \begin{vmatrix} 2 & -3 & 8 \\ 2 & 12 & -23 \\ 0 & -7 & -22 \end{vmatrix} = \frac{1}{2}\frac{1}{12} \begin{vmatrix} 2 & -3 & 8 \\ 0 & 12 & -23 \\ 0 & 0 & 103 \end{vmatrix} =$$

$$= \frac{1}{24}(2 \cdot 12 \cdot 103) = 103$$

9 Příklad

Výpočet determinantu podle Laplaceovy věty

$$A = \begin{pmatrix} 2 & -3 & 8 \\ 4 & 6 & -7 \\ -5 & 4 & 9 \end{pmatrix} \tag{27}$$

$$\det A = \begin{vmatrix} 2 & -3 & 8 \\ 4 & 6 & -7 \\ -5 & 4 & 9 \end{vmatrix} = (-4) \begin{vmatrix} -3 & 8 \\ 4 & -9 \end{vmatrix} + 6 \begin{vmatrix} 2 & 8 \\ -5 & -9 \end{vmatrix} + 7 \begin{vmatrix} 2 & -3 \\ -5 & 4 \end{vmatrix} =$$

$$= (-4)(27 - 32) + 6(-18 + 40) + 7(8 - 15) = 20 + 132 - 49 = 103$$

1.2.7 Inverzní matice

$\underline{A} = (aij)$ je čtvercová matice typu (n,n). Čtvercová matice $\underline{B}$ typu (n,n) se nazývá inverzní k matici $\underline{A}$, když $\underline{A} \cdot \underline{B} = \underline{B} \cdot \underline{A} = \underline{E}$. Na první pohled není zřejmé, zda matice $\underline{B}$ inverzní k $\underline{A}$ existuje vždy, zda je určena jednoznačně a jak tuto matici vypočítat. K formulaci odpo-

vědí nám pomůže teorie determinantů. K matici $\underline{A} = (a_{ij})$ typu (n,n) vždy existuje matice adj $\underline{A}$, tzv. adjungovaná matice k matici $\underline{A}$. Inverzní matice k $\underline{A}$ existuje právě tehdy, když $\det(\underline{A}) \neq 0$. V případě, že inverzní matice existuje, je určena jednoznačně a označuje se A^{-1}. Inverzní matici lze pak vypočítat podle vzorce

$$\underline{A}^{-1} = \frac{1}{\det \underline{A}} \cdot adj\, \underline{A} \tag{28}$$

Pro inverzní matici platí řada zákonitostí. Uvádím aspoň dvě nejpoužívanější. Platí $(\underline{A}^{-1})^{-1} = \underline{A}$, $(A \cdot B)^{-1} = \underline{B}^{-1} \cdot \underline{A}^{-1}$.

Rovněž existuje alternativní možnost výpočtu inverzní matice. Je založena na teorii elementárních transformací matic.

$[\underline{A}|\underline{E}]$ je matice typu $(n,2n)$ vzniklá tak, že za matici $\underline{A}$ napíšeme jednotkovou matici $\underline{E}$. Pak existuje posloupnost řádkových elementárních transformací, která převede matici $[\underline{A}|\underline{E}]$ na matici $[\underline{E}|\underline{B}]$ a platí $\underline{B} = \underline{A}^{-1}$.

10 Příklad

Výpočet inverzní matici $\underline{A}^{-1}$ k matici $\underline{A}$ pomocí matice adjungované

$$\underline{A} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \end{pmatrix} \tag{29}$$

$$\underline{A}^{-1} = \frac{1}{\det \underline{A}} \cdot adj\, \underline{A} = \frac{1}{-5} \begin{vmatrix} \begin{vmatrix} 1 & 2 \\ 2 & 0 \end{vmatrix} & -\begin{vmatrix} 1 & 2 \\ 1 & 0 \end{vmatrix} & \begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix} \\ -\begin{vmatrix} 1 & 1 \\ 2 & 0 \end{vmatrix} & \begin{vmatrix} 2 & 1 \\ 1 & 0 \end{vmatrix} & -\begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix} \\ \begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix} & -\begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix} & \begin{vmatrix} 2 & 1 \\ 1 & 1 \end{vmatrix} \end{vmatrix}^{T} =$$

$$= \frac{1}{-5} \begin{vmatrix} -4 & 2 & 1 \\ 2 & -1 & -3 \\ 1 & -3 & 1 \end{vmatrix}^{T} = \begin{vmatrix} \frac{4}{5} & -\frac{2}{5} & -\frac{1}{5} \\ -\frac{2}{5} & \frac{1}{5} & \frac{3}{5} \\ -\frac{1}{5} & \frac{3}{5} & -\frac{1}{5} \end{vmatrix}$$

11 Příklad

Výpočet inverzní matici $\underline{A}^{-1}$ k matici $\underline{A}$ pomocí matice jednotkové

$$\underline{A} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \end{pmatrix} \tag{30}$$

$$\left[\underline{A}\middle|\underline{E}\right] = \left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 0 & 1 \\ 1 & 1 & 2 & 0 & 1 & 0 \\ 2 & 1 & 1 & 1 & 0 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 0 & 1 \\ 0 & -1 & 2 & 0 & 1 & -1 \\ 0 & -3 & 1 & 1 & 0 & -2 \end{array}\right] \rightarrow$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 0 & 1 \\ 0 & -1 & 2 & 0 & 1 & -1 \\ 0 & 0 & -5 & 1 & -3 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 0 & 1 \\ 0 & -1 & 2 & 0 & 1 & -1 \\ 0 & 0 & 1 & -\frac{1}{5} & \frac{3}{5} & -\frac{1}{5} \end{array}\right] \rightarrow$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & -\frac{2}{5} & -\frac{1}{5} & -\frac{3}{5} \\ 0 & 0 & 1 & -\frac{1}{5} & -\frac{3}{5} & -\frac{1}{5} \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{4}{5} & -\frac{2}{5} & -\frac{1}{5} \\ 0 & 1 & 0 & -\frac{2}{5} & \frac{1}{5} & \frac{3}{5} \\ 0 & 0 & 1 & -\frac{1}{5} & \frac{3}{5} & -\frac{1}{5} \end{array}\right] = \left[\underline{E}\middle|\underline{A}^{-1}\right]$$

Zkoušku správnosti lze provést ověřením platnosti vztahu $\underline{A} \cdot \underline{A}^{-1} = \underline{A}^{-1} \cdot \underline{A} = \underline{E}$

2 VYUŽITÍ MATIC

2.1 Identifikace

Velice významného užití nacházejí matice při identifikaci procesů, např. v průběžné identifikaci metodou nejmenších čtverců, kde se využívají matice čtvercové, symetrické (tzn.je nutné vypočítat matici transponovanou k matici původní), maticemi upravenými na dolní trojúhelníkový tvar a také maticemi inverzními.

2.2 Určení stability dynamického systému pomocí algebraického Hurwitzova kritéria

Další využití nacházejí také v jednom z kritérií k určení stability dynamického systému. Stabilita dynamického systému ve je schopnost vrátit se po vychýlení zpět do původního stavu. Toto vychýlení je vždy způsobeno nenulovými počátečními podmínkami. Schéma kritéria vychází z tzv. Hurwitzovy matice, v níž se počítají všechny hlavní subdeterminanty odpovídající hlavním minorům matice $\underline{H}_n$, tedy det $\underline{H}_{n-1}$, det $\underline{H}_{n-2}$,…,det $\underline{H}_2$, det $\underline{H}_1$. Tato matice má tvar

$$\underline{H}_n = \begin{pmatrix} a_{n-1} & a_{n-3} & a_{n-5} & \cdots & 0 & \cdots & 0 \\ a_n & a_{n-2} & a_{n-4} & \cdots & 0 & \cdots & 0 \\ 0 & a_{n-1} & a_{n-3} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & a_2 & a_0 \end{pmatrix} \tag{31}$$

a rozměr (*n*,*n*)

Při splnění nutné podmínky stability (ve formě kladnosti koeficientů ve vyšetřovaném polynomu) zní Hurwitzovo kritérium:

Polynom je stabilní právě tehdy, jestliže všechny hlavní subdeterminanty jsou větší než nula.

12 Příklad

Určete zda polynom $s^3 + 2s^2 + 2s + 40$ je stabilní či nikoliv.

$$\underline{H}_3 = \begin{pmatrix} 2 & 40 & 0 \\ 1 & 2 & 0 \\ 0 & 2 & 40 \end{pmatrix} \tag{32}$$

det $\underline{H}_1 = 2>0$

det $\underline{H}_2 = 4\text{-}40 = -36 \Rightarrow$ nestabilní

2.3 Prediktivní řízení

Prediktivní řízení se rozvíjí od sedmdesátých let minulého století. Důvodem rozvoje prediktivního řízení je fakt, že představuje nejobecnější cestu návrhu řízení procesu podle zadaného kritéria. Jsou zejména vhodné k nasazení v oblasti omezené znalosti o řízení.

Základní myšlenkou prediktivního řízení je použití přesného modelu k predikci výstupu procesu v budoucím časovém okamžiku. Známe tedy výstup procesu několik kroků dopředu (výhled predikce). Známe-li vývoj žádané hodnoty, můžeme vypočítat hodnoty akčního zásahu (výhled řízení). Akční zásah je vypočítán optimalizací daného kritéria, aby výstup procesu co nejlépe sledoval žádanou hodnotu. Toto kritérium má obvykle podobu kvadratické funkce rozdílu mezi predikovaným výstupem a žádanou hodnotou (účelová funkce). Ve většině případů je v účelové funkci obsažen také řídicí signál, jehož hodnoty je třeba penalizovat. Rozhodující roli v regulátoru hraje model procesu. Musí být schopný zachytit dynamiku a přesně predikovat výstupy. Návrh modelu není jedinečný, ale existuje řada různě formulovaných typů modelů. Článek [1] blíže seznamuje s prediktivním algoritmem DMC, který využívá matic k ukládání hodnot odezvy do řídící matice, která je po identifikaci využívána k výpočtu řídícího zásahu

II. PRAKTICKÁ ČÁST

3 POROVNÁNÍ MOŽNOSTÍ ASSEMBLERU A CODEWARRIORU

Na začátku projektu bylo nutné se rozhodnout, zda k vytvoření modulu bude použit výhradně jazyk symbolických adres, nebo programu CodeWarrior firmy Metrowerks, která je samostatnou dceřinou firmou Motoroly. Mezi neoddiskutovatelné výhody CodeWarrioru patří především to, že práce v něm je založena na programovacím jazyce C, včetně veškerých jeho matematických funkcí a s tím související usnadnění ladění programu a zvýšení jeho přehlednosti. Mezi další výhody patří také kompatibilita modulů mezi všemi mikropočítači Motorola, což byl také jeden z požadavků kladených na tento projekt, jednoduchá implementace modulů do výsledné aplikace a také v neposlední řadě snadný přístup k funkcím mikropočítače a snadná práce s čísly s pohyblivou desetinnou čárkou. Tyto jednoznačné výhody oproti programování v jazyce assembler, jehož předností je větší přehlednost obsazenosti paměti RAM, jeho mírně větší rychlost a potřeba menšího paměťového prostoru vedly k závěru použít vývojové prostředí Metrowerks CodeWarrior.

4 METROWERKS CODEWARRIOR

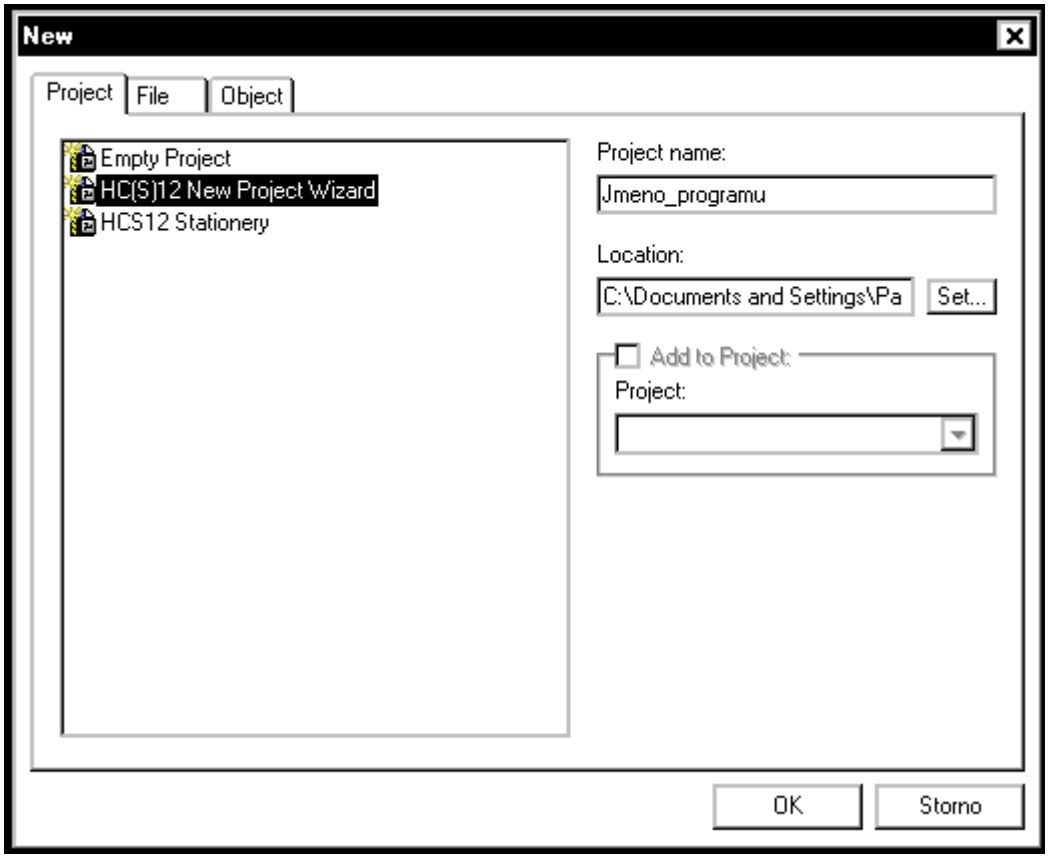
4.1 Základní informace

Kompletní zdrojový kód pro knihovnu pro mikropočítač Mototrola HC12 jsem navrhl v programu CodeWarrior 12 Program CodeWarrior od firmy Metrowerks umožňuje vývoj aplikací určených pro 8-mi, 16-ti bitové mikroprocesory Motorola řady HC08 a HC12. Pro psaní zdrojového kódu aplikačního softwaru lze použít jak asembler příslušného procesoru, tak programovacího jazyka C podle normy ISO ANSI C. Prostředí CodeWarrior krom nástrojů pro psaní a editaci zdrojových kódu, obsahuje nástroj pro nahrávání spustitelného programu do cílového systému a dále také velmi užitečný simulátor, který umožňuje velmi jednoduché ladění aplikačního software bez nutnosti nahrání do cílového sytému. Pro detailní popis prostředí a postupu při vývoji aplikace doporučuji [2] a [3]. Integrované vývojové prostředí (IDE - Integrated Development Environment) sady CodeWarrior nabízí intuitivní grafické uživatelské rozhraní, které je společné pro všechny mikrokontrolery Motorola a umožňuje tak snadnou migraci. Vývojáři mohou tvořit, kompilovat, sestavovat i ladit v jediném IDE.

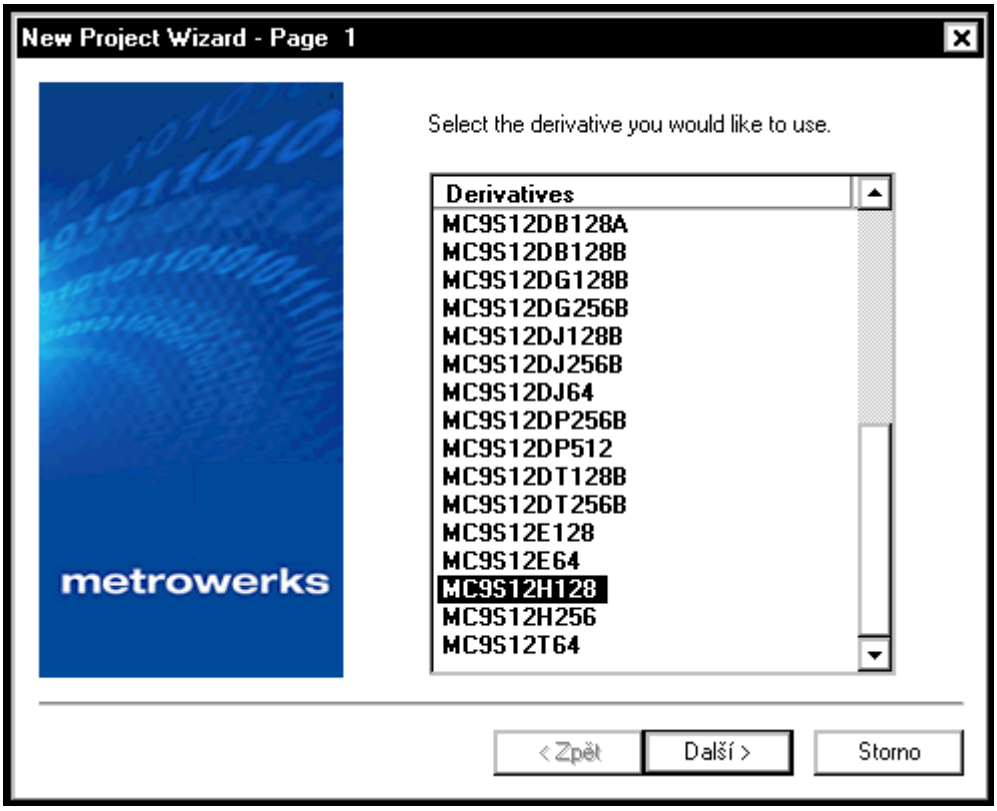
4.2 Vytváření aplikace

- Spustíme program Metrowerks CodeWarrior IDE
- Z menu jsem zvolíme "File->New" pro vytvoření nového projektu. Objeví se okno (Obr.1), ze kterého zvolíme „HC(S)12 New Project Wizard“ a zadáme název nového projektu (Project Name) a cestu, kde budou soubory projektu uloženy.
- V dalším okně zvolíme typ mikrokontroléru, se kterým budeme pracovat – MC9S12E128 a stiskneme tlačítko "Další" (Obr.2).
- V dalším okně zaškrtneme "C" (Obr.3) (můj projekt byl programován v jazyku C)
- V následujícím okně volbu ponecháme a pokračujeme stiskem tlačítka další kde zatrhneme volbu „float is IEEE32, double is IEEE32“ (Obr.4).
- V dalším okně opět volbu ponecháme, pokračujeme tlačítkem „Další“

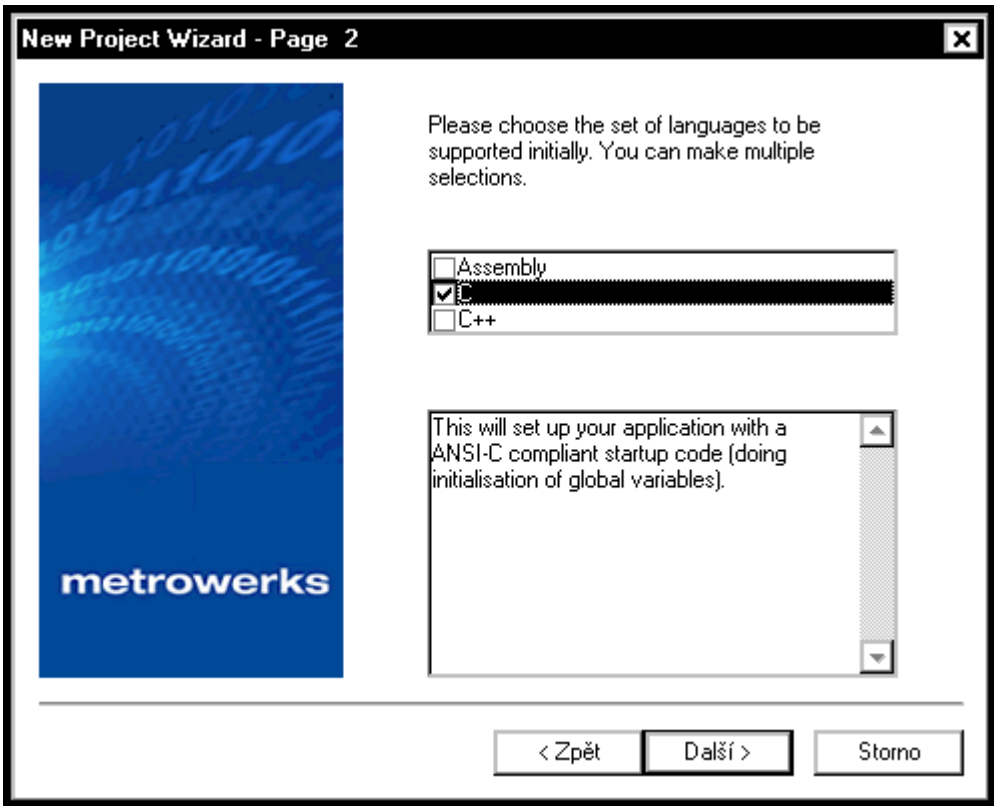
- V dalším okně (Obr.5) zvolíme „Metrowerks Full Chip Simulation“, tato volba znamená, že budeme ladit aplikaci v SW simulátoru. Tlačítkem "Dokončit" se vygenerujeme projekt.



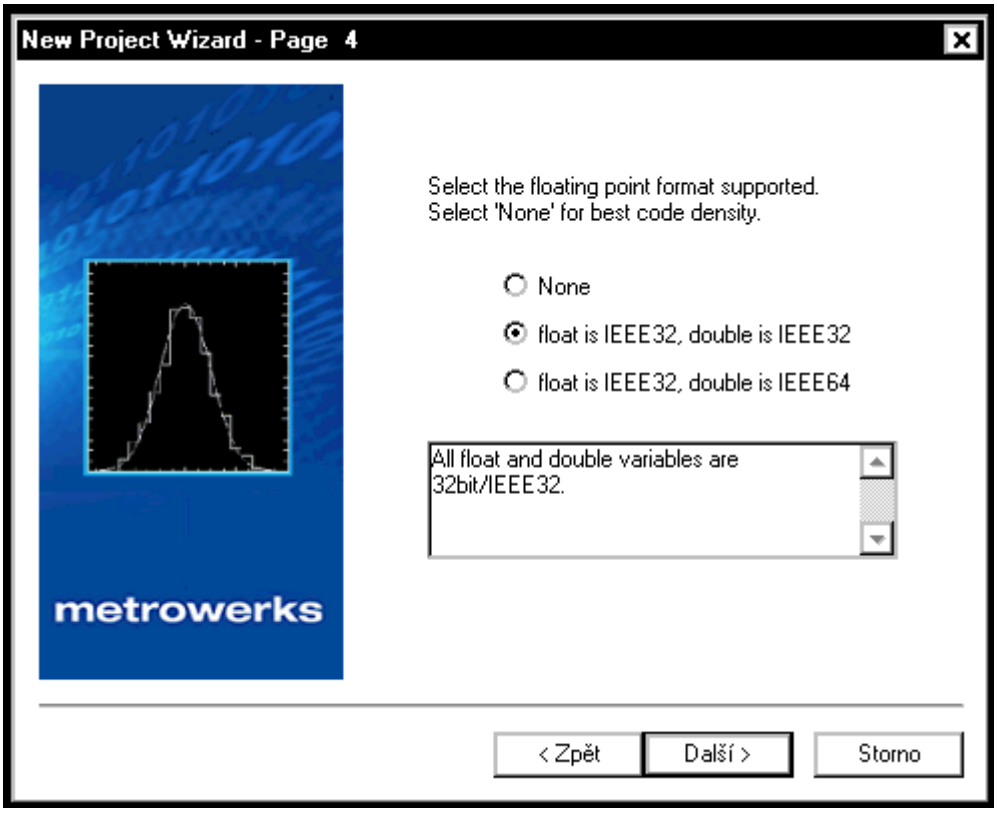
Obr. 1. Vytvoření nového projektu



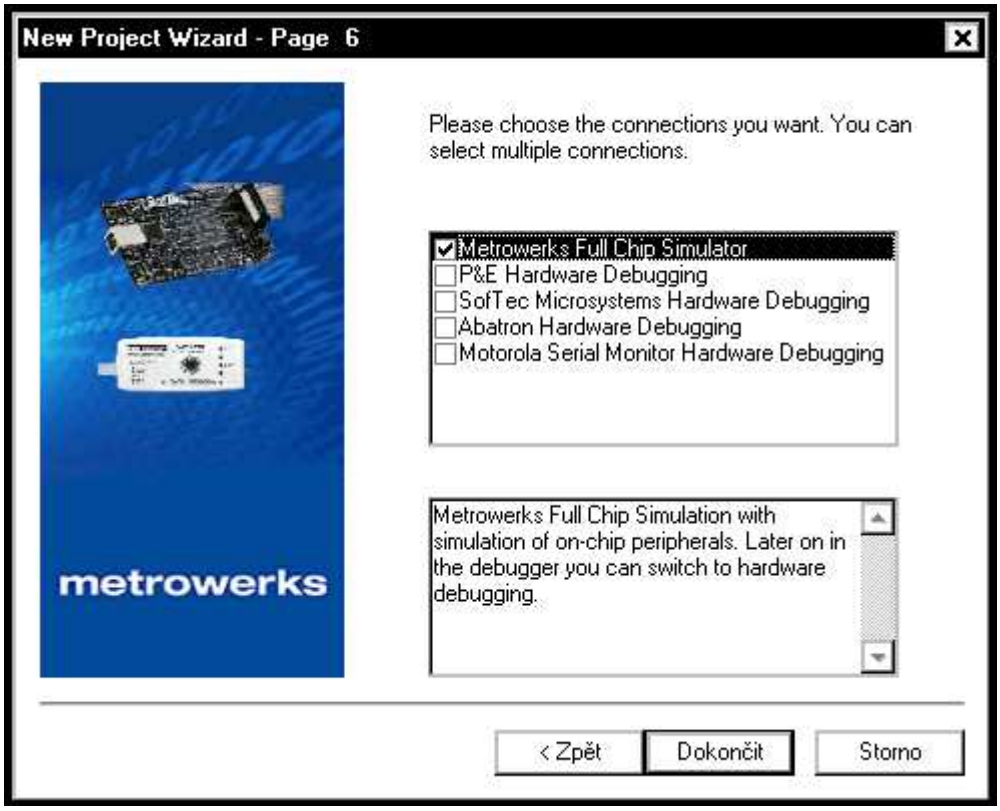
Obr. 2. Výběr typu mikrokontroléru



Obr. 3. Výběr programovacího jazyka



Obr. 4. Zvolení formátu typu float a double



Obr. 5. Dokončení vytváření aplikace

4.3 Vývojové prostředí

Vidíme před sebou okno (Obr.6) se stromem souborů, které náležejí projektu (záložka „Files“). Nás zajímá větev „Sources“, v níž je soubor „main.c“ - soubor obsahující kód budoucí aplikace a soubor „datapage.c“. Při poklikání na soubor „main.c“ se nám otevře okno, do kterého již můžeme zapisovat samotný kód.

Máme-li hotový zdrojový program, můžete jej přeložit („Project/Make“ z menu či klávesa F7).

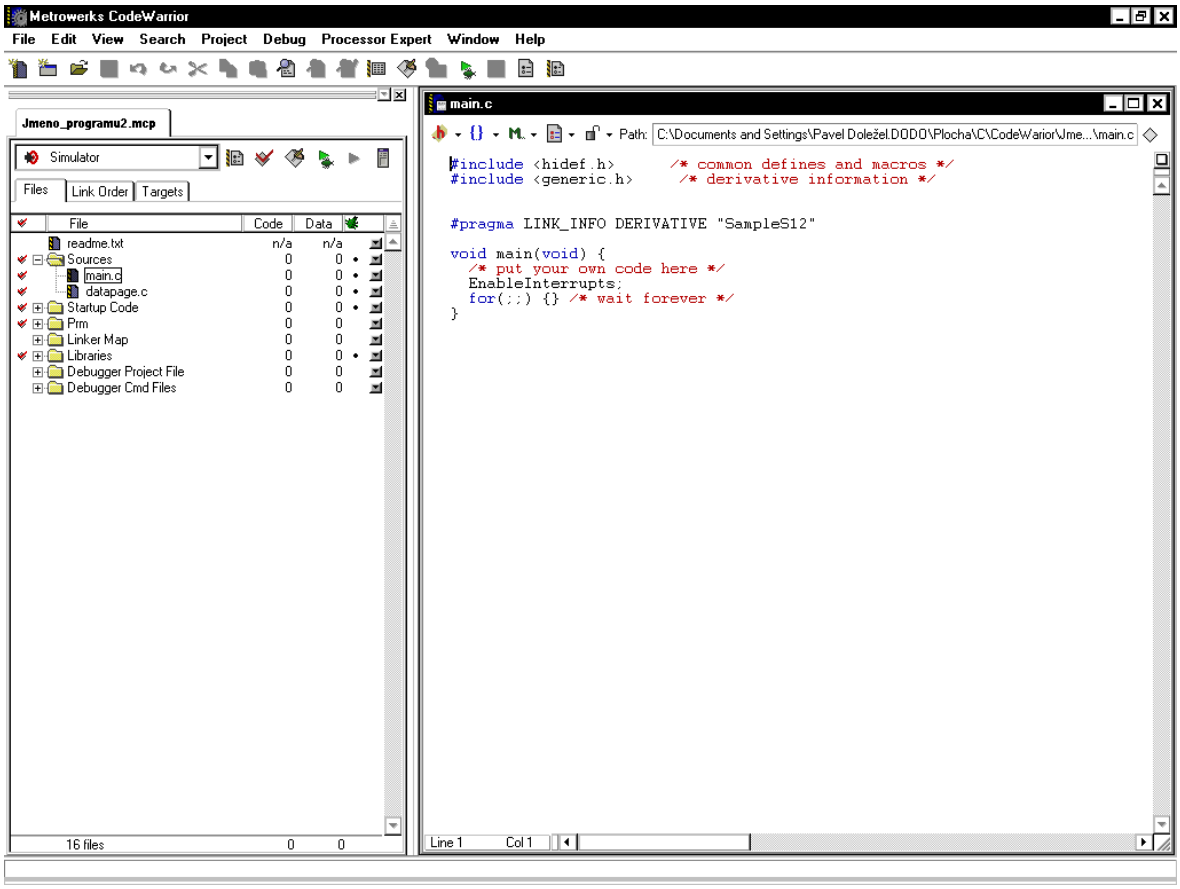
Po úspěšném překladu spustíme debugger („Project/Debug“ z menu či klávesa F5). Poté se nám otevře okno „True-Time Simulator @ Real Time Debugger“ (Obr.7), jehož jednotlivé části jsou následující.

Horní lišta je stejná jako u většiny programů s tím, že nejdůležitějšími tlačítky pro moji práci byly „Start/Continue“ a „Single Step“. Pomocí prvního spouštíme program, který proběhne buď celý, nebo pouze po tzv.breakpoint, který nastavujeme kliknutím pravým tlačítkem myši do požadovaného řádku zdrojového kódu v okně „Sources“ a vybráním

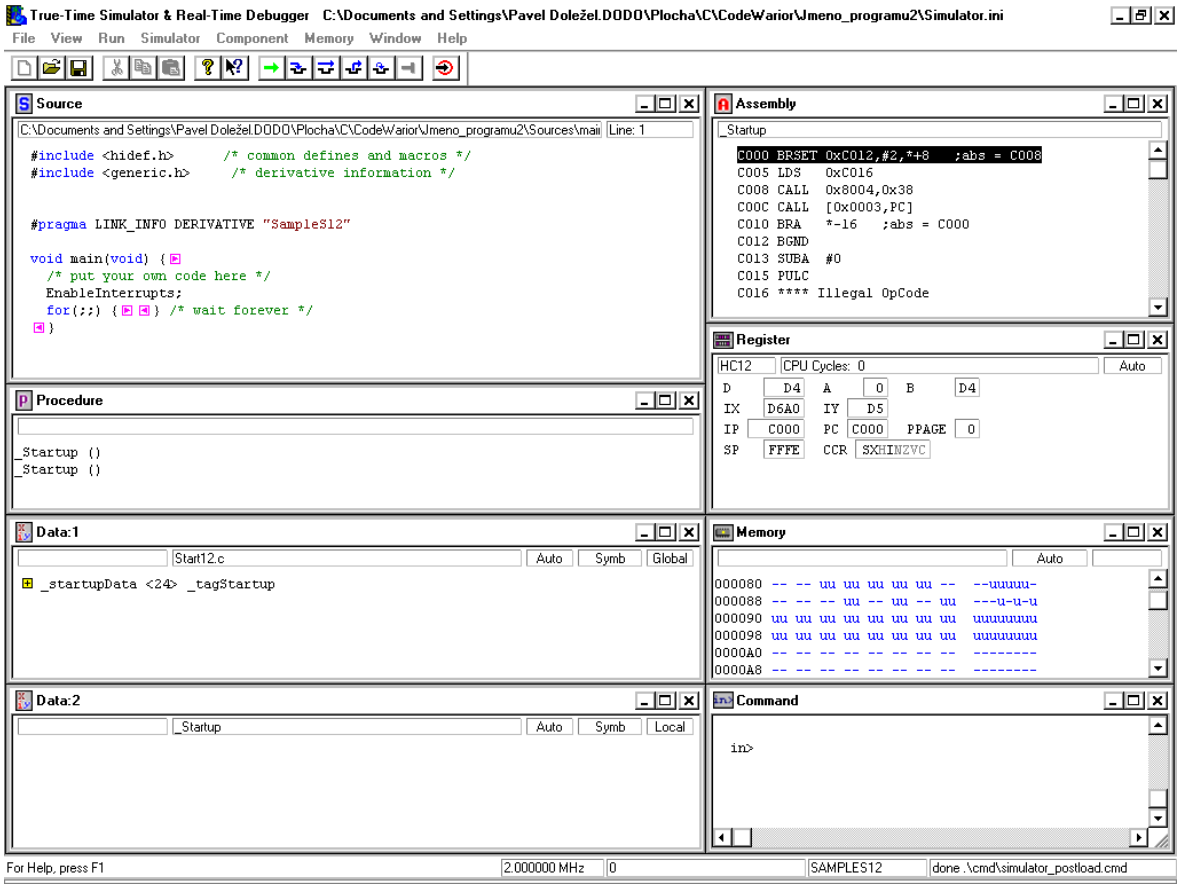
položky „Set Breakpoint“. Po opětovné kliknutí na toto tlačítko program program pokračuje v běhu buď na konec, nebo k dalšímu „breakpointu“. Pomocí druhého tlačítka můžeme běh programu spouštět krok po kroku.

V okně „Sources“ vidíme náš zdrojový kód, okno Data:1 (Data:2) zobrazuje globální (lokální) proměnné a jejich hodnoty. Velice užitečnou pomůckou při ladění programu je jeho krokování a následné ověřování hodnot proměnných po jednotlivých instrukcích.

Okno „Assembly“ zobrazuje kód v assembleru, okno „Register“ hodnoty v jednotlivých registrech, včetně pomocných a konečně okno „Command“ vypisuje jednotlivé akce a příkazy, které právě proběhly.



Obr. 6. Hlavní okno programu



Obr. 7. Okno True-Time Simulatoru

5 POPIS DEKLARACÍ VYTVOŘENÝCH FUNKCÍ POUŽITÝCH V PROJEKTU

Pro každou matematickou operaci s maticemi byla naprogramovaná samostatná funkce v programu Metrowerks Codewarrior, který jsem zvolit kvůli jeho výhodám popsaným výše v textu. Každá z těchto funkcí vrací hodnotu *int* a to z toho důvodu, že v závislosti na úspěšném či neúspěšném proběhnutí funkce vrací hodnotu 1(vše v pořádku) nebo 0 (např. špatně volaná funkce, nemožnost provést požadovanou funkci). V modulu byl použit pro čísla s pohyblivou desetinnou čárkou typ *float*. Tento čtyřbytový typ je kompatibilní s tvarem IEEE, který je využíván ve starších modulech. Další typ pro čísla s pohyblivou desetinnou čárkou je typ *double*, který zabírá 8 bytů. Výpočetní operace s tímto typem jsou velmi přesné, avšak současně značně časově náročné. Navíc různé matematické funkce pro tento typ zabírají mnohem více místa v paměti. Proto byl využit výhradně jen typ první – *float*. Všechny funkce pro zajištění jejich univerzálního použití využívají dynamické přidělování paměti pro maticové proměnné.

5.1 Funkce pro výpočet součtu matic

*int soucet(float **matice1, int r1, int s1, float **matice2, int r2, int s2, float **vsoucet);*

Vstupními parametry funkce jsou *matice1* typu (*r1,s1*) a *matice2* typu (*r2,s2*). Výstupním parametrem je matice *vsoucet*, do které je uložen výsledek operace součtu. Ve funkci nejprve dochází k ověření, zda *matice1* a *matice2* mají stejný rozměr pomocí funkce *stejrozm()*, která jako vstupní parametry používá právě rozměry obou matic. Funkce vrací nulu v případě že se rozměry matic neshodují, v opačném případě vrací jedničku.

5.2 Funkce pro výpočet rozdílu matic

*int rozdil(float **matice1, int r1, int s1, float **matice2, int r2, int s2, float **vrozdil);*

Vstupními parametry funkce jsou *matice1* typu (*r1,s1*) a *matice2* typu (*r2,s2*). Výstupním parametrem je matice *vrozdil*, do které je uložen výsledek operace rozdílu. Stejně jako u funkce *soucet()* je rozměr obou matic ověřen pomocí funkce *stejrozm()*a návratové hodnoty jsou také stejné.

5.3 Funkce pro výpočet matice transponované

*int transpozice(float **matice1, int r1, int s1, float **vtransp);*

Vstupním parametrem funkce je *matice1* typu (*r1*,*s1*). Výstupním parametrem je matice *transp*, do které je uložen výsledek operace transpozice. Při úspěšném proběhnutí programu funkce vrací jedničku, v opačném případě nulu.

5.4 Funkce pro vynásobení matice konstantou

*int nasobkonst(float **matice, int r1, int s1, float k, float **vnasob);*

Vstupními parametry funkce jsou *matice1* typu (*r1*,*s1*) a konstanta *k*, kterou je vynásoben každý prvek matice. Výstupním parametrem je matice *vnasob*, do které je uložen výsledek operace vynásobení konstantou. Návratové hodnoty jsou stejné jako u funkce předešlé.

5.5 Funkce pro součin matic

*int nasobenimatic(float **matice1, int r1, int s1, float **matice2, int r2, int s2,*

*float **vroznasob);*

Vstupními parametry funkce jsou *matice1* typu (*r1*,*s1*) a *matice2* typu (*r2*,*s2*). Výstupním parametrem je matice *vroznasob* typu (*r1*,*s2*), do které je uložen výsledek operace součinu matic. Ve funkci nejprve dochází k ověření, zda *s1=r2*, v opačném případě by k provedení operace nemohlo dojít a funkce by vracela nulu.

5.6 Funkce pro výpočet determinantu

*int determ(float **matice, int r1, int s1, float *determinant);*

Vstupním parametrem funkce je *matice* typu (*r1*,*s1*). Výstupním parametrem je proměnná *determinant*, do které je uložen výsledek výpočtu determinantu. V této funkci je použita funkce *LUDCMP()*, která matici rozloží na matici trojúhelníkovou a jejíž algoritmus, přisuzovaný matematiku Croutovi, je blíže popsán v publikaci [4], popř. článku [5]. Ve funkci také dochází k ověření, zda matice *matice* je čtvercová pomocí funkce *ctvercova()*. Jestliže matice není čtvercová, popřípadě je singulární, funkce vrací nulu, v opačném případě je návratovou hodnotou jednička.

5.7 Funkce pro výpočet inverzní matice

*int inverze(float **matice, int r1, int s1, float **imatice);*

Vstupním parametrem funkce je *matice* typu (*r1,s1*). Výstupním parametrem je matice *imatice*, do které je uložen výsledek operace inverze matice. V této funkci jsou použity funkce *LUDCMP()* a *LUBKSB()*, jejichž algoritmy jsou popsány v publikaci [5], popř. článku [6]. Stejně jako u výpočtu determinantu funkce vrací nulu v případě že matice není čtvercová, nebo je singulární, navíc vrací nulu také v případě že determinant je roven nule.

ZÁVĚR

Tato bakalářská práce se zabývá problematikou matic a návrhem knihovny pro práci s maticemi libovolného typu.

Úvodní kapitola práce je věnována teoretickému rozboru práce s maticemi. Jsou zde vysvětleny pojmy jako číselné těleso, matice, diagonála a diagonální matice, jednotková matice, trojúhelníková matice, transponovaná matice, symetrická a antisymetrická matice, hodnost matice, regulární a singulární matice. Také jsou zde popsány veškeré elementární úpravy matic. Ke všem těmto pojmům jsou pro názornost uvedeny konkrétní případy.

Ve druhé kapitole jsou uvedeny a popsány některé z možností využití matic a práce s nimi jako jsou identifikace soustavy, určení stability dynamického systému Hurwitzovým kritériem, popřípadě využití matic v prediktivním řízení.

V praktické části jsou nejprve stručně popsány a srovnány výhody a nevýhody návrhu kódu v jazyce symbolických adres – assembleru a naproti tomu za použití vývojového prostředí Metrowerks CodeWarrior, a poté zmíněny důvody vedoucí k použití právě druhého ze zmíněných nástrojů.

V další kapitole jsou opět stručně shrnuty základní vlastnosti Metrowerks Codewarrioru, dále je zde uveden názorný postup při vytváření aplikace v tomto vývojovém prostředí, kdy programátor volí z několika možných nastavení. Na závěr této kapitoly jsou zde popsána jednotlivá okna True-Time Simulatoru díky kterým máme přehled nejen nad naším zdrojovým kódem, ale také nad globálními i lokálními proměnnými, včetně hodnot v jednotlivých registrech.

Závěrečná kapitola obsahuje popis deklarací veškerých funkcí použitých v projektu. Ke každé funkci jsou zde popsány případy, kdy se jejich návratová hodnota rovná nule, popřípadě jedničce. Stejně tak jsou zde vysvětleny důvody a způsoby použití jednotlivých vstupních a výstupních parametrů.

SEZNAM POUŽITÉ LITERATURY

[1] Malík, K.: Predictive Controllers [online]. Dostupný z URL http://www.feec.vutbr.cz

[2] OSEK builder v. 2.2.1, user's manual,Metrowerks, 2001

[3] CodeWarrior user's manual,Metrowerks,2001

[4] Press, W.H.: Numerical Recipes in C: the art of scientific computing

[5] Zábrodský, V.: LU rozklad [online]. Dostupný z URL http://www.geocities.com/zabrodskyvlada/cz_aat/a_lude.html

[6] Cifrik: Elementy lineární algebry [online]. Dostupný z URL http://www.matematika.webz.cz/algebra/linealg/algebra.doc

[7] Burkhard, M.: C pro mikrokontroléry, Ben 2003

[8] Krákora, M.: Diplomová práce [online]. Dostupná z URL dce.felk.cvut.cz/dolezilkova/diplomky/2003/dp_2003_krakora_michal

[9] Bobál, V., Böhm, J., Prokop, R., Fessl, J.: Praktické aspekty samočinně se nastavujících regulátorů: algoritmy a implementace. Brno 1999. ISBN 80-214-1299-2

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ANSI American National Standards Institute – americká standardizační organizace

ASCII American Standard Code for Information – kódovací tabulka pro základní znaky

DMC Prediktivní algoritmus

HC08 Typ mikropočítače Motorola

HC12 Typ mikropočítače Motorola

IEEE Institute of Electrical and Electronics Engineers

IDE Integrated Development Environment

ISO International Organization for Standardization – Mezinárodní organizace pro normalizaci

RAM Random Access Memory – paměť s náhodným přístupem

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

P I: CD ROM

Obsahuje:

- Samotnou bakalářskou práci v PDF a MS Word formátu
- Soubor *matice.h*
- Soubor *matice.c*
- Kompletní projekt v Metrowerks Codewarrior ve složce *Projekt*