Dokumentation Create Gateway - CMEX

Lang Klaus

21. Februar 2010

Inhaltsverzeichnis

1 Allgemeines

MatLab bietet die Möglichkeit Funktionen oder Unterprogramme, welche in C geschrieben wurden in ein MatLab Programm als sog. MEX - Files einzubinden. Dies ist vor allem dann von Nutzen, wenn das erforderliche Unterprogramm nur als C - Programm zur Verfügung steht oder eine Ausführung dieses Codes in MatLab zu viel Zeit in Anspruch nehmen würde.
Create Gateway ist ein Programm um den Benutzer bei der Einbindung von `*.c` oder `*.cpp` Dateien in MatLab zu unterstützen.

1.1 Funktion

Die Funktion solcher MEX - Files lässt sich schematisch mit Hilfe von Abbildung 1 erklären.

Abbildung 1: Informationsfluss zwischen MatLab (`main.m`) und dem Unterprogramm (`master.c`)

`main.m`
In MatLab wird das Hauptprogramm ausgeführt. Von diesem M - File wird das Unterprogramm oder die Funktion in `master.c` aufgerufen.

`master.c`
In dieser Datei ist der C - Code des Unterprogrammes gespeichert.
Es ist nicht erforderlich, dass sich der gesamte Code in dieser Datei befindet, es können auch weitere Unterprogramme in anderen C - Files aufgerufen werden. Wichtig ist lediglich, dass sich das „Master“ Unterprogramm in dieser Datei befindet und entsprechend gekennzeichnet ist.
(vgl. Kapitel 2 *C Masterfile*)
Der Name „Master“ bezieht sich nicht auf irgendein Keyword in MatLab oder C sondern soll nur verdeutlichen, dass es sich um das Hauptprogramm in C handelt, ohne dabei den Ausdruck *main* zu verwenden.

`gateway.c`
Um das Matlab Programm mit der C Datei zu verbinden ist ein sog. Gateway

- File (auch mexFunktion genannt) erforderlich.
Dieses Gateway - File wird vom Programm *Create Gateway* automatisch erstellt.

Anmerkung:

- Die Namen der Dateien können, mit Ausnahme der Keywords von MatLab und C, frei gewählt werden und müssen **nicht** mit der Nomenklatur von Abbildung 1 übereinstimmen.
- In weiterer Folge wird in dieser Dokumentation die Richtung des Informationsflusses von MatLab nach C mit „In“ und von C nach Matlab mit „Out“ bezeichnet.

1.2 Installation

Um eine möglichst einfache Verwendung des Programms *Create Gateway* zu gewährleisten, ist es notwendig die drei Dateien

- `Create_Gateway.m`
- `get_data.m`
- `write_file.m`

in einen eigenen Ordner zu kopieren und diesen Ordner zur Matlab Variable `Path` hinzuzufügen.

1.2.1 Hinzufügen eines Ordners zur Variable `Path`

1. Öffnen des Dialogfensters Set Path durch `File --> Set Path...`
(vgl. Abbildung 2)

Abbildung 2: Set Path

2. Klick auf den Button `Add Folder...` im Dialogfenster `Set Path` (vgl. Abbildung 3)

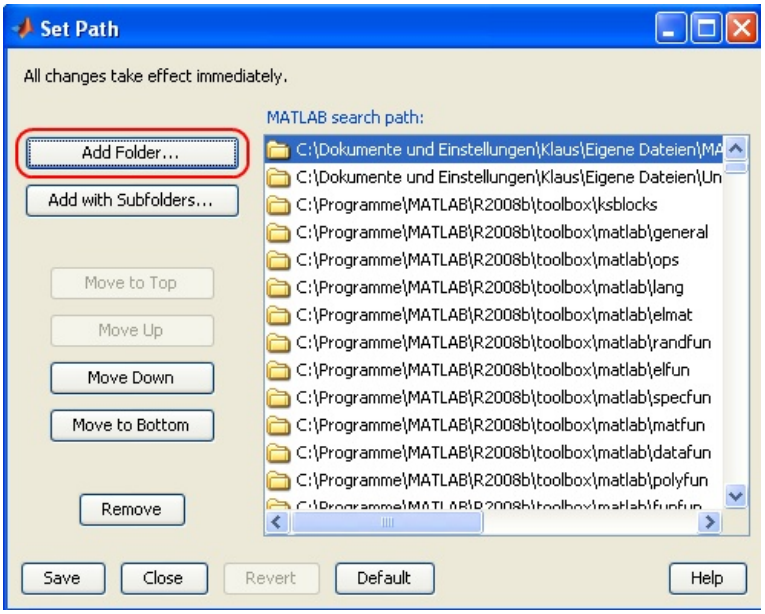

Abbildung 3: Ordner hinzufügen

3. Auswahl des Ordners mit den Dateien `Create_Gateway.m`, `get_data.m` und `write_file.m`
4. Speichern der Einstellungen mit dem Button `Save` im Dialogfenster Set `Path`

Durch diesen Vorgang kann *Create Gateway* durch die Eingabe des Befehls `Create_Gateway` in das MatLab Command Window gestartet werden.

1.2.2 Compiler

Durch Eingabe des Befehls `mex -setup` in das MatLab Command Window kann der Compiler für das Erstellen von MEX - Files ausgewählt werden. (Weitere Informationen findet man in der MatLab Hilfe.)

Achtung:
Create Gateway wurde nur für folgende Compiler getestet:

- Microsoft Visual C++ 2008 Express
- Microsoft Visual C++ 6.0

1.3 Bedienung

In diesem Unterkapitel wird die Bedienung von *Create Gateway* anhand des C - Masterfiles `times_two.c` erklärt.

C - Code von `times_two.c`:

```
/*
Dieses Programm multipliziert die Input Variable x mit dem Faktor 2
und gibt die Outputvariable y zurück
*/

// begin_marker()

void times_two(double x, double *y)
{
    *y = x * 2;
}
```

Bedienung:

1. Starten von *Create Gateway* durch Eingabe des Befehls

   `Create_Gateway`

   in das MatLab Command Window.
   Das Programm listet alle `*.c` und `*.cpp` Dateien (in diesem Fall nur `times_two.c`) des Current Directory auf.
   (vgl. Abbildung 4)

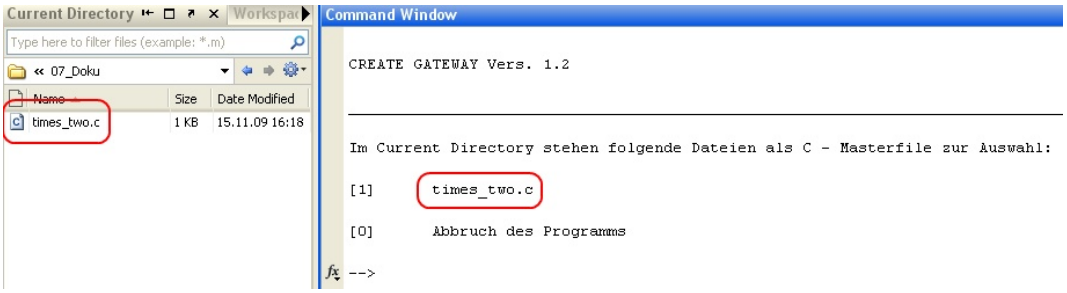

Abbildung 4: Current Directory und Command Window nach dem Starten von *Create Gateway*

2. Auswahl von `times_two.c` als Masterfile durch Eingabe von `1`
   Es erscheint eine Kontrollliste der Übergabeparameter die vom Benutzer entweder bestätigt (Eingabe von `y`) oder abgelehnt (Eingabe von `n`) werden kann.
   (vgl. Abbildung 5)
3. Im Current Directory erscheint das Gateway - File

   In diesem Fall: `c2m_times_two.c`

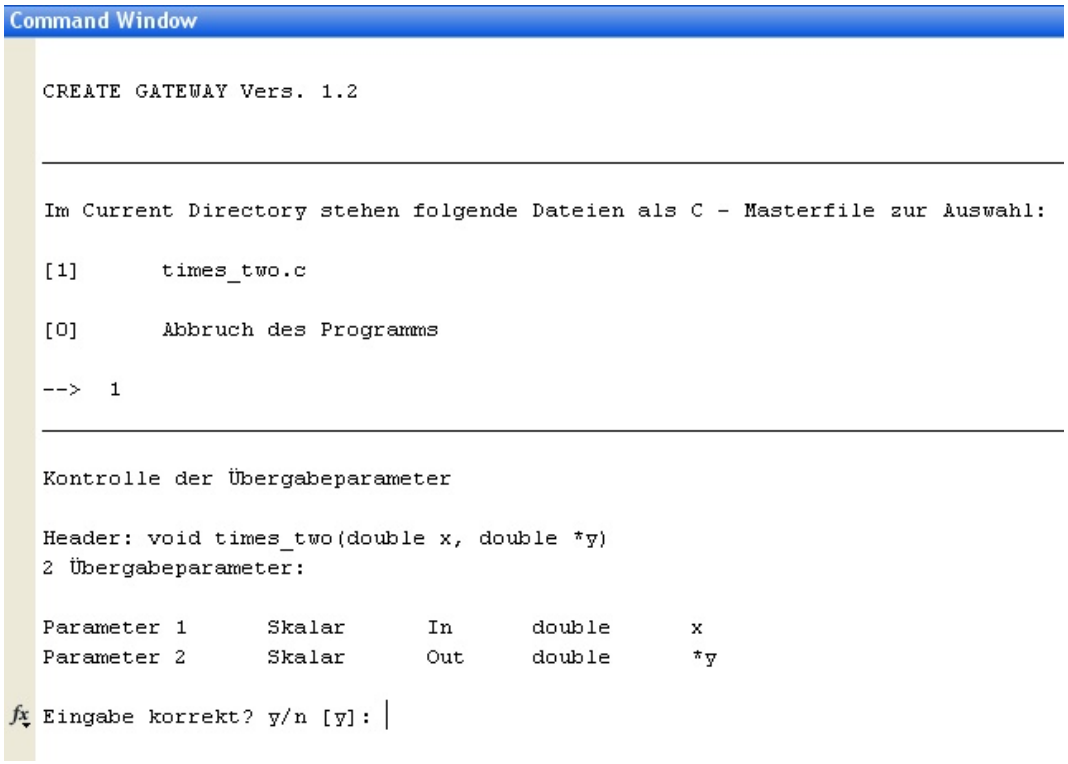

Abbildung 5: Kontrollliste der Übergabeparameter

4. Im Command Window wird der Befehl ausgegeben mit dem das Programm nach dem Compilieren gestartet werden kann

   In diesem Fall: `[y] = c2m_times_two(x);`

5. Es folgt eine Aufforderung ob der Compilierungsprozess gestartet werden soll. Diese kann wieder mit `y` bestätigt oder mit `n` abgelehnt werden.

6. Wird der Compilierungsvorgang gestartet erscheint das zugehörige MEX - File im Current Directory.
   (vgl. Abbildung 6)

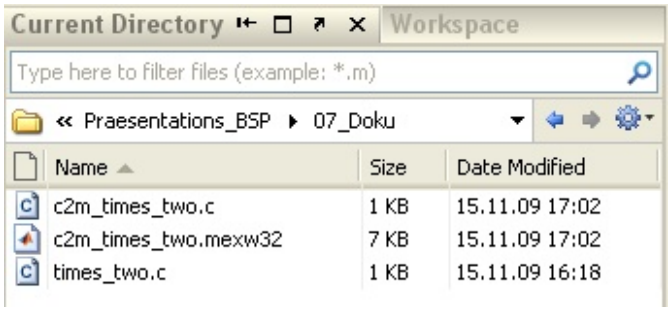

Abbildung 6: Current Directory nach Ausführen von *Create Gateway*

7. Die Eingabe in das Command Window von

   ```
   c2m_times_two(10)
   ```

   ergibt

   ```
   ans =

   20
   ```

Achtung!
Für die Ausführung des Programms wird nur mehr das MEX - File benötigt!

2 C - Masterfile

Create Gateway benötigt zur Erstellung des Gateway - Files bestimmte Informationen über die Übergabeparameter, wie Datentype, Richtung (In,Out) etc. .
Diese Informationen werden automatisch aus dem C - Masterfile ausgelesen. Dabei sucht das Programm nach einer Markierung, welche in jedem C - Masterfile folgendermaßen lauten muss:

```
// begin_marker()
```

Die nächste nicht leere Zeile im C - Masterfile, nach dieser Markierung, muss der Programmaufruf des Master - Unterprogramms sein.
(vgl. Kapitel 1.3 `times_two.c`)
Aus dieser Zeile werden die Informationen bezüglich der Übergabeparameter gewonnen.
Für bestimmte Parameter (im konkreten Fall Arrays) genügt die Programmaufrufzeile dafür nicht und es muss zusätzlich noch die Richtung (In, Out oder IO) des Informationsflusses angegeben werden.
Dies geschieht in der Markerzeile zwischen den runden Klammern nach folgendem Schema:

```
In / Out / IO Parametername
```

Achtung!

- Die Reihenfolge der Parameter in der Pogrammaufrufzeile und der Markerzeile müssen übereinstimmen.
- Sind mehrere Einträge in der Markerzeile notwendig müssen diese durch einen „ , “ getrennt werden.

Genauere Informationen siehe Kapitel 3 *Übergabeparameter*.

2.1 Schema eines C - Masterfiles

```
//Header
#include <...>
...
#define ...
...

// Prototypen, Globale Variablen, ...
```

```
...
C - Code
...

// begin_marker(In x[], Out y[])

void x_times_y(double x[], double y[])
{

...
C - Code
...

}
```

Achtung!
Der Header eines C - Masterfiles muss alle für das C - Programm notwendigen includes, defines, Prototypen etc. enthalten!

3 Übergabeparameter

Das Programm *Create Gateway* unterstützt für den Austausch zwischen MatLab und C folgende Arten von Übergabeparametern:

- Numerische 1 x 1 Matrizen sog. **Skalare**
- Numerische 1 x n oder n x 1 Matrizen sog. **Vektoren**
- Numerische n x m **Matrizen**
- **Strings**

Wie bereits in Kapitel 2 *C Masterfile* erwähnt, durchsucht *Create Gateway* die Programmaufrufzeile nach Information, um welche Art von Übergabeparameter es sich handelt. Dabei wird neben dem Datentype (int, double, char) auch nach bestimmten Keywords gesucht. (siehe Tabelle 1)

Tabelle 1: Keywörter der Übergabeparameter

Übergabeparameter	unterstützter Datentype	Richtung	Keyword
Skalar	double, int	In	-
Skalar	double, int	Out	*
Vektor	double, int	In, Out, IO	[]
Matrix	double, int	In, Out, IO	**
String	char	In, Out, IO	*, []

Anmerkung:
In Tabelle 1 wird als mögliche Richtung der Übergabeparameter auch „IO“ angegeben.
Diese Richtungsbezeichnung bedeutet, dass ein Vektor oder String sowohl als Input als auch als Output für das C - Programm verwendet werden kann.
(vgl. Kapitel 5.3 `square.c`)

3.1 Skalare

Achtung!

- Wie bereits in Tabelle 1 ersichtlich, werden Skalare als Pointer von C zurück an MatLab übergeben.
- Für Skalare Übergabeparameter ist kein zusätzlicher Eintrag im Marker notwendig!
  (vgl. Kapitel 1.3 `times_two.c`)

3.2 Vektoren (1D Arrays)

Vektoren benötigen einen zusätzlichen Eintrag in die Marker Zeile des C - Masterfiles. (vgl. Tabelle 2)

Tabelle 2: Markereintrag für Vektoren

Bsp. Parametername	Richtung	Eintrag Programmaufrufzeile	Eintrag Markerzeile
x	In	double x[]	In x[]
y	Out	double y[]	Out y[]
z	IO	double z[]	IO z[]
oder			
x	In	int x[]	In x[]
y	Out	int y[]	Out y[]
z	IO	int z[]	IO z[]

Beispiel:

```
// Header
...
C - Code
...

// begin_marker(In x[], Out y[], IO z[])

void vektoren(double x[], double y[], double z[])
{
...
C - Code
...
}
```

Achtung!
Vektoren mit der Richtungsbezeichnung „Out“...

- ... werden immer als Zeilenvektor an MatLab zurück gegeben!
- ... benötigen in MatLab einen zusätzlichen Input Parameter der die Anzahl der Elemente festlegt!
  (vgl. Kapitel 5.1 `xtimesy.c`)

Achtung!

Vektoren vom Datentype `int`, welche von MatLab an C übergeben werden,[1] müssen schon in MatLab in den Datentype `int32` umgewandelt werden. (vgl. MatLab Hilfe und Kapitel 5.4 `sum_ele.c`)

3.3 Matrizen (2D Arrays)

Auch Matrizen benötigen einen Eintrag in die Marker Zeile des C - Masterfiles. (vgl. Tabelle 3)

Tabelle 3: Markereintrag für Matrizen

Bsp. Parametername	Richtung	Eintrag Programmaufrufzeile	Eintrag Markerzeile
x	In	double **x	In **x
y	Out	double **y	Out **y
z	IO	double **z	IO **z
oder			
x	In	int **x	In **x
y	Out	int **y	Out **y
z	IO	int **z	IO **z

Beispiel:

```
// Header
...
C - Code
...

// begin_marker(In **x, Out **y, IO **z)

void matrizen(double **x, double **y, double **z)
{
...
C - Code
...
}
```

Achtung!
Matrizen mit der Richtungsbezeichnung „Out“ benötigen in MatLab einen

[1] Betrifft die Richtungsbezeichnungen „In“ und „IO“

zusätzlichen Input Parameter der die Anzahl der Zeilen und Spalten der Matrix festlegt!
(vgl. Kapitel 5.5 `,mat_out.c`)

Achtung!
Matrizen vom Datentype `int`, welche von MatLab an C übergeben werden, müssen wie bei den Vektoren, schon in MatLab in den Datentype `int32` umgewandelt werden.
(vgl. MatLab Hilfe und Kapitel 5.4 `sum_ele.c`)

3.4 Strings

Auch Strings benötigen einen zusätzlichen Eintrag in die Markerzeile des C - Masterfiles. (vgl. Tabelle 2)

Tabelle 4: Markereintrag für Strings

Bsp. Parametername	Richtung	Eintrag Programmaufrufzeile	Eintrag Markerzeile
x	In	char x[]	In x[]
y	Out	char y[]	Out y[]
z	IO	char z[]	IO z[]
oder			
x	In	char *x	In *x
y	Out	char *y	Out *y
z	IO	char *z	IO *z

Beispiel:

```
// Header
...
C - Code
...

// begin_marker(In x[], Out y[], IO z[])

void strings(char x[], char y[], char z[])
{
...
C - Code
...
}
```

Achtung!

- Strings mit der Richtungsbezeichnung „Out“ dürfen nicht mehr als 500 Zeichen enthalten!
- Strings mit der Richtungsbezeichnung „In“ und „IO“ müssen als Spaltenvektoren an C übergeben werden!
  (vgl. Kapitel 5.2 `hello_world.c`)

4 Anmerkungen

4.1 GCC

Bei der Verwendung des GCC - Compilers kann beim Compilieren folgende Fehlermeldung auftauchen:

```
error: expected expression before '/' token
```

Unter Umständen kann dieser Fehler behoben werden, in dem man das Gateway - File von `*.c` in `*.cpp` umbenennt und noch einmal mit diesem Befehl compiliert.

```
mex c2m_filename.cpp
```

Genauere Informationen über das Compilieren von MEX Funktionen findet man in der MatLab Hilfe unter dem Suchbegriff `mex`.

4.2 Debuggen

Informationen zum Debuggen von MEX - Files findet man ebenfalls in der MatLab Hilfe unter dem Suchbegriff `Debugging C Language MEX-Files`.

5 Beispiele

5.1 xtimesy.c

C - Code:

```
#include <stdio.h>

/*
 * Multipliziert den Input Vektor "x" mit der Variable "y" und gibt das
 * Ergebnis im Vektor "z" zurück!
 */

// begin_marker(In x[],Out z[])

void xtimesy(int n, double x[], double y, double z[])
{
    int k;

    for(k=0;k<n;k++)
    {
        z[k] = x[k] * y;
    }
}
```

Programmaufruf in MatLab:

```
[z] = c2m_xtimesy(n,x,y,numel(z));
```

Achtung!
Dieses Programm benötigt einen zusätzlichen Input Parameter der die Anzahl der Elemente des Outputvektors festlegt!

5.2 hello_world.c

C - Code:

```
#include <stdio.h>

/*
 * Gibt den String "word" mit dem printf - Befehl im Command Window aus!
 */

// gibt den String word mit dem printf Befehl aus

// begin_marker(In word[])
```

```
void hello_world(char word[])
{
    printf("\n%s\n",word);
}
```

Programmaufruf in MatLab:

```
word = 'Hello World';
c2m_hello_world(word);
```

5.3 square.c

C - Code:

```
#include <stdio.h>

/*
 * Quadriert den Vektor "x" und gibt das Ergebnis zurück!
 */

// begin_marker(IO x[])

void square(int n, double x[])
{
    int k;
    for(k=0;k<n;k++)
    {
        x[k] = x[k]*x[k];
    }
}
```

Programmaufruf in MatLab:

```
[x] = c2m_square(n,x);
```

5.4 sum_ele.c

C - Code:

```
#include <stdio.h>

/*
 * Summiert über die Elemente des Vektors "x"!
 */

// begin_marker(IO x[])
```

```
void sum_ele(int n, int x[])
{
    int k;
    for(k=1;k<n;k++)
    {
        x[k] = x[k] + x[k-1];
    }
}
```

Programmaufruf in MatLab:

```
[x] = c2m_sum_ele(n,int32(x));
```

Achtung!
Der Inputparameter `x` muss bereits in MatLab mit dem Befehl `int32` in den Datentype `int32` umgewandelt werden!

5.5 mat_out.c

C - Code:

```
#include <stdio.h>

/*
 * Gibt eine Matrix mit Nullen zurück!
 */

// begin_marker(Out **Y)

void mat_out(int row, int col, double **Y)
{
    int k,l;

    for(k=0;k<row;k++)
        for(l=0;l<col;l++)
            Y[k][l] = 0;
}
```

Programmaufruf in MatLab:

```
[Y] = c2m_mat_out(row,col,size(Y));
```

Achtung!
Dieses Programm benötigt einen zusätzlichen Input Parameter der die Anzahl der Zeilen und Spalten der Outputmatrix festlegt!