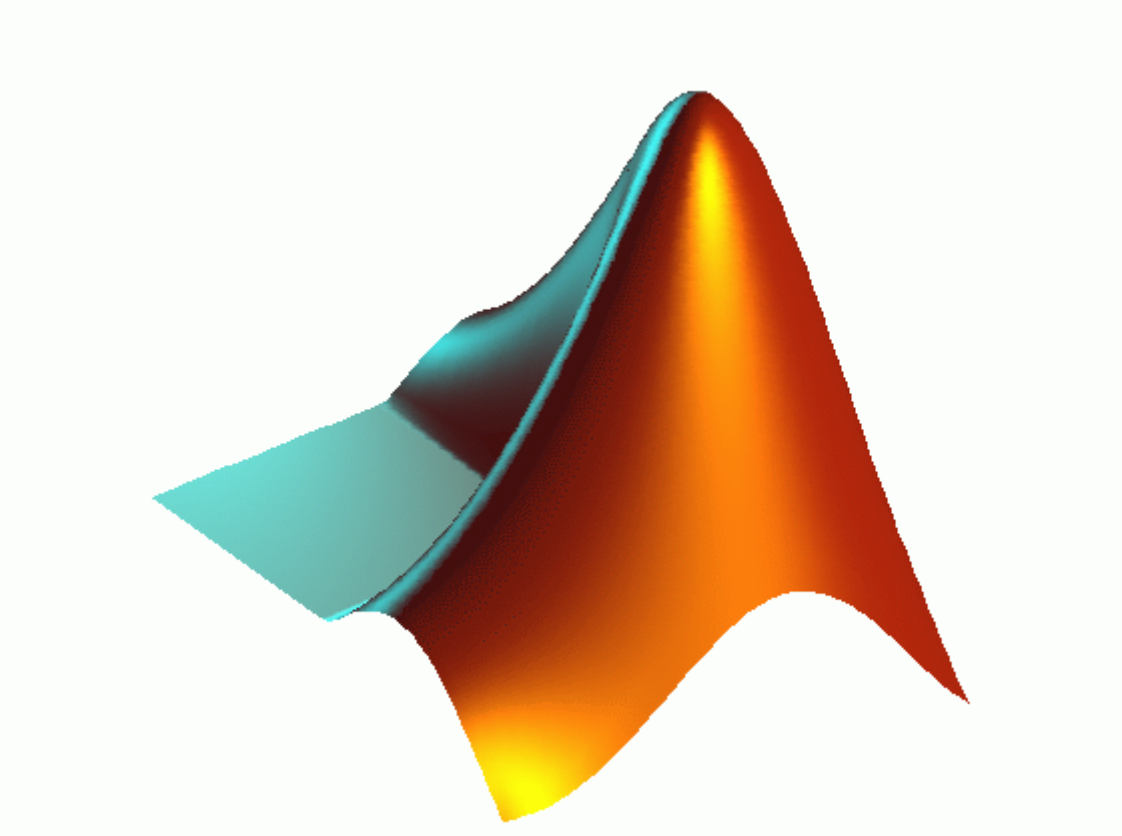

# MATLAB Compiler

# ab MATLAB R2008a

**Benutzerhinweise**

admin@informatik.fh-regensburg.de
Fakultät Informatik & Mathematik
Hochschule Regensburg

**Überarbeitet von:**

Martin Opel, Martin Sturm 2003, Daniela Krämer 2006, Thorsten Skrabal 2008

**Stand Dezember 2008**

**Zusammenfassung:**
Diese Anleitung beschreibt möglichst einfach die ersten Schritte mit dem MATLAB Compiler im **CIP-POOL**. Sie soll keine komplette Referenz dieses Produkts sein, sondern den Einstieg erleichtern. Die komplette Dokumentation im PDF-Format finden Sie unter:

MATLAB Help-Window:
► Contents ► MATLAB Compiler ► MATLAB Compiler User's Guide

# Inhaltsverzeichnis



# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1. Einleitung

Mit dem MATLAB Compiler ist es möglich, aus MATLAB Dateien verschiedene andere Dateitypen zu generieren.
Da mit CMEX-Dateien, Standalone-Anwendungen, C/C++ Programmcode oder dynamisch gelinkte Bibliotheken (DLLs) erstellt werden können, muss ein von MATLAB unterstützter Compiler installiert sein. Auf den CIP-Pool Rechnern ist dazu der Microsoft C/C++ Compiler installiert. Für die Erstellung von Standalone Anwendungen und DLLs braucht man außerdem die MATLAB C/C++ Bibliothek, die auf den CIP-Pool Rechnern installiert ist. Falls Sie in Ihrer MATLAB-Anwendung Grafik-Routinen verwenden, brauchen Sie auch die MATLAB C/C++ Graphik-Bibliotheken, die sie zu Ihrem Programm linken müssen, doch dazu später mehr.

Zum besseren Verständnis seien hier noch die wichtigsten Begriffe erklärt:

**MEX-Dateien** sind Dateien, in denen externe Routinen implementiert sind, die von MATLAB verwendet werden können. **MEX-Funktionen** sind in der Regel um ein Vielfaches schneller als die MATLAB Skript M-Dateien.

**Stand-Alone Anwendungen** sind Anwendungen die ohne Hilfe von anderen Programmen, wie MATLAB oder C-Compiler laufen.

Anwendungen die mit dem MATLAB-Compiler erstellt werden, sind **32-Bit Anwendungen**. Der Aufruf des MATLAB-Compilers kann sowohl von der MS-DOS Eingabeaufforderung aus, als auch aus dem MATLAB-Befehlsfenster erfolgen (siehe rechtes Teilfenster in Abb. 1).

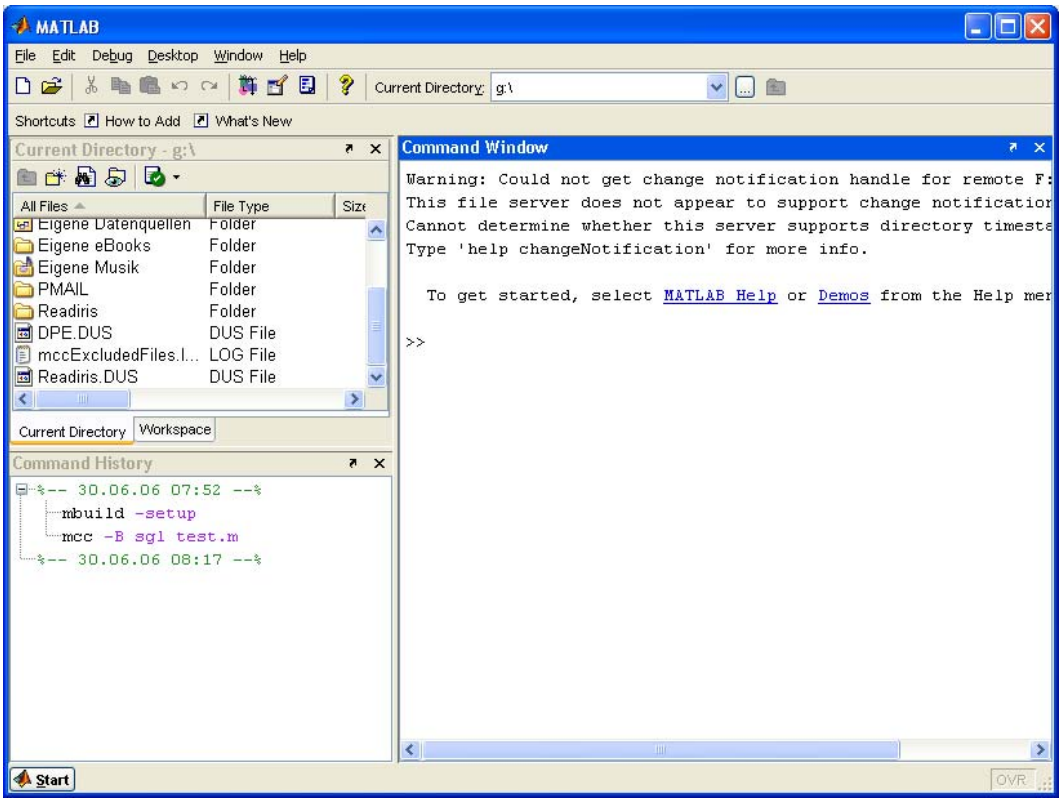


**Abbildung 1: MATLAB Haupt-Fenster**

Bei der Übersetzung von M-Dateien mit dem MATLAB Compiler werden folgende Dateien erzeugt:

- C oder C++ Code, abhängig davon, welche Zielsprache angegeben wurde
- Archiv Dateien
- Wrapper-Datei
- Zieldatei: entweder Standalone Exe-Anwendung, DLL, MEX-Datei, ...

Der erzeugte C oder C++ Code und die erzeugte Archiv-Datei sind unabhängig vom Zieltyp (z.B. Standalone Anwendung) und von der Zielplattform.

Die erzeugten Dateien landen immer in dem Verzeichnis, das als Arbeitsverzeichnis konfiguriert wurde. Nach dem Start des MATLAB-Compilers ist das das Unterverzeichnis `\work` des Installationsverzeichnisses von MATLAB.

Da dieses Verzeichnis im CIP-Pool nicht beschreibbar ist, sollten Sie sich zuerst Ihr eigenes Arbeitsverzeichnis einstellen. Im MATLAB Hauptfenster ist dies mit dem Auswahldialog *Current Directory* oben rechts möglich. Beim Aufruf des Compilers in einem DOS-Fenster werden die Dateien selbstverständlich im aktuellen Verzeichnis erzeugt.

# 2. Benutzung des Compilers

## *2.1 Aufruf des Compilers*

Der Standardaufruf des MATLAB Compilers lautet `mcc`. Damit können durch Angabe von verschiedenen Parametern verschiedene Typen von Zieldateien erstellt werden. Die genaue Syntax lautet:

```
mcc [optionaleParameter] <Datei1> [<Datei2>...]
```

`mcc` übersetzt die angegebene Datei `<Datei1>.m` in `<Datei2>.c` oder `<Datei2>.cpp`, je nachdem welcher Zieltyp (hier C bzw. C++) mit den optionalen Parametern eingestellt wird. Die erzeugten Dateien landen im aktuellen Verzeichnis bzw. im aktuell konfigurierten *Current Directory*.

Wenn mehr als eine Datei angegeben wird, dann wird für jede M-Datei eine Zieldatei erzeugt. Es ist ebenfalls möglich C/C++ Dateien beim Aufruf von `mcc als <Datei1> [<Datei2>...]` anzugeben, um diese ebenfalls zu übersetzen.

Wenn gegensätzliche Optionen als Parameter angegeben werden, werden die am weitesten rechts liegenden benutzt. In Tabelle 2 sind Beispiele für die gebräuchlichsten Compiler-Aufrufe erklärt.

| Befehl | | Beschreibung |
|---|---|---|
| mcc | -p <Dateiname> | Übersetzen in C++ und Erstellen der ausführbaren Datei für M-Dateien ohne Grafik. |
| mcc | -m <Dateiname> | Übersetzen in C und Erstellen der ausführbaren Datei ohne Grafik. |
| mcc | -B sglcpp <Dateiname> | Übersetzen in C++ und Erstellen der ausführbaren Datei für M-Dateien mit Grafik. |
| mcc | -B sgl <Dateiname> | Übersetzen in C und Erstellen der ausführbaren Datei für M-Dateien mit Grafik. |
| mcc | -x <Dateiname> | Erzeugen einer MEX-Datei. |

**Tabelle 1: Wichtige Compiler Optionen**

Eine komplette Liste aller MATLAB Compiler-Optionen inklusive Beschreibung erhalten Sie, wenn Sie den Befehl

```
mcc -?
```

Im *Command Window* eingeben. Auch im DOS-Fenster lassen sich so die verfügbaren Optionen anzeigen, allerdings muss man sich die Ausgabe seitenweise anzeigen lassen, indem man folgenden Befehl im DOS-Fenster verwendet:

```
C:\> mcc -? | more
```

## *2.2 Beispiel: Erstellung einer Standalone-Anwendung*

Das folgende Beispiel soll zeigen, wie man eine in MATLAB erstellte M-Datei mit dem MATLAB-Compiler übersetzt, in eine eigenständige Datei umwandelt, und diese dann aufruft.

**Wichtiger Hinweis**

Damit die Übersetzung mit Hilfe des MATLAB-Compilers problemlos funktioniert, müssen Sie ggf. **VORHER** die Umgebungsvariable für den Pfad (PATH) entsprechend setzen. Weitere Einzelheiten dazu finden Sie im Kapitel 2.2.4.2. Erst dann sollten Sie MATLAB bzw. das DOS-Fester mit dem MATLAB-Compiler aufrufen.

### *2.2.1 Starten von MATLAB bzw. Start eines DOS-Fensters*

Starten Sie entweder die MATLAB-Umgebung mit dem Desktop-Symbol oder dem Quicklaunch-Symbol in Ihrer Taskleiste oder öffnen Sie ein DOS-Fenster aus dem Startmenü.

### *2.2.2 Setzen des Arbeitsverzeichnisses*

Falls Sie die MATLAB-Umgebung gestartet haben, wechseln Sie in das Verzeichnis, in dem auch Ihre M-Datei liegt. Wie oben schon erwähnt, können Sie das Arbeitsverzeichnis mit dem *Current Directory Dialog* setzen, der sich oben rechts im Hauptfenster befindet (siehe Abbildung 1, Seite 4).

Im DOS-Fenster wechseln Sie einfach in das gewünschte Verzeichnis mit dem `cd` Kommando.

### *2.2.3 Kompilieren der M-Datei*

Als Beispiel verwenden wir die Datei `test.m`, die Grafik-Befehle enthält. Wir entscheiden uns, eine ausführbare Datei zu erzeugen und nehmen den Weg über C. Vorher existiert im eingestellten Arbeitsverzeichnis nur die M-Datei `test.m`:

```
C:\> dir

Verzeichnis von G:\tmp

02.04.06   10:40      <DIR>           .
02.04.06   10:40      <DIR>           ..
02.04.2006 10:40                1.192   test.m
02.04.2006 10:40                   68   f.m
               2 Datei(en)
               2 Verzeichnis(se)
```

Als erstes muss der jeweilige Compiler, der genutzt werden soll, eingestellt werden. Dies muss nur einmalig getan werden und geschieht wie folgt, mit der `mbuild –setup` Anweisung.

```
C:\> mbuild -setup
Please choose your compiler for building standalone MATLAB
applications:

Would you like mbuild to locate installed compilers [y]/n? y
```

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in F:\M\Matlab\R2008a\CIP\sys\lcc
[2] Microsoft Visual C++ 2005 in C:\Programme\Microsoft Visual
Studio 8

[0] None

Compiler: 1

Please verify your choices:

Compiler: Lcc-win32 C 2.4.1
Location: F:\M\Matlab\R2008a\CIP\sys\lcc

Are these correct [y]/n? y

Trying to update options file: C:\Dokumente und
Einstellungen\Standard\Anwendungsdaten\MathWorks\MATLAB\R2008a\compo
pts.bat
From template:
F:\M\Matlab\R2008a\CIP\bin\win32\mbuildopts\lcccompp
.bat

Done . . .
```

Es werden folgende Compiler von MATLAB unterstützt:

- Microsoft Visual C / C++
- LCC – Win32
- Borland C / C++ Builder
- Borland C / C++ Compiler

**Hinweis:** [2] Microsoft Visual C/C++ 2005 steht nur dann zur Verfügung, wenn es vorher installiert wurde.

Jetzt führen wir den MATLAB Compiler aus und zwar mit der Option `-B sgl` zum Umwandeln der M-Datei in C und Verwenden der C Grafik-Bibliotheken.
Die Option `-B` lädt eine Datei `sgl`, die zusätzliche Optionen enthält, so genannte Bundles. Die Datei sgl in unserem Beispiel enthält die Optionen

```
-p -W mainhg libmwsglm.mlib
```

Die Optionen, die in dieser Datei enthalten sind, werden ganz einfach zusätzlich vom Compiler ausgeführt. Man könnte diese Optionen auch alle einzeln angeben. Den Befehl können Sie entweder im MATLAB Command Window (rechtes Fenster in der MATLAB Umgebung) oder im DOS-Fenster direkt eingeben.

```
C:\> mcc -B sgl test.m
```

Nach dem Kompilieren sollten zusätzlich die Header-Dateien, die C Dateien, die Wrapper-Datei und die erzeugte Exe-Datei vorhanden sein:

```
C:\> dir

02.04.2006 10:40      <DIR>          .
02.04.2006 10:40      <DIR>          ..
```

```
02.04.2006 10:40                68   f.m
02.04.2006 10:40               838   mccExcludedFiles.log
02.04.2006 10:40            35.623   test.ctf
02.04.2006 10:40            10.948   test.exe
02.04.2006 10:40             1.192   test.m
02.04.2006 10:40             2.724   test_main.c
02.04.2006 10:40             6.370   test_mcc_component_data.c
7 Datei(en)                  57.763 Bytes
```

Der Compiler hat zuerst aus der M-Datei die C-Datei `test.c` und das Archiv `test.ctf` und die Wrapper-C-Datei `test_main.c` erzeugt. Zum Schluss hat er dann daraus das ausführbare Programm **`test.exe`** erzeugt und die C Grafik-Bibliothek dazugelinkt.

**Hinweis:** Zum Ausführen des Programms `test.exe` benötigen Sie unbedingt die Archivdatei `test.ctf`!

### 2.2.4 Starten der ausführbaren kompilierten Programms (Datei-.exe)

#### 2.2.4.1 Fall 1 - MATLAB ist nicht auf dem Rechner installiert

Das Starten dieser Exe-Datei endet jedoch mit einer Fehlermeldung (siehe Abbildung 2). Schuld daran sind die fehlenden DLLs der MATLAB Laufzeit-Umgebung.

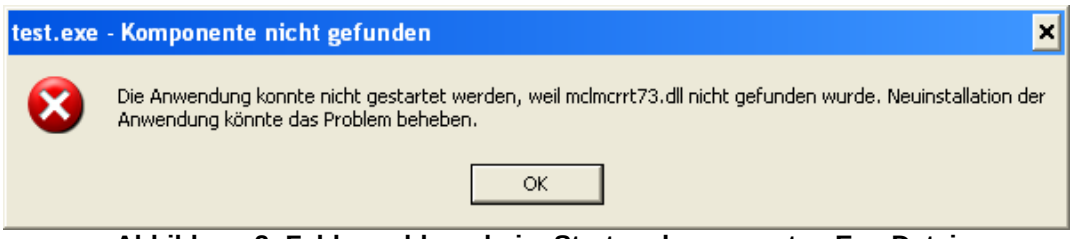


**Abbildung 2: Fehlermeldung beim Starten der erzeugten Exe-Datei**

Will man die erzeugte Exe-Datei auf einem System starten, auf dem MATLAB nicht installiert ist, dann muss vorher das MATLAB Runtime Component installiert werden.

Dazu wird aus dem Verzeichnis `$MATLAB\toolbox\compiler\deploy\win32`, d.h. im CIP-Pool

`F:\M\Matlab\R2008a\CIP\toolbox\compiler\deploy\win32`

die Datei `MCRInstaller.exe` (ca. 230MB groß) auf den Zielrechner kopiert. Diese muss dann ausgeführt und den Installationsanweisungen folge geleistet werden.

Danach kann `test.exe` (+ `text.ctf`) ausgeführt werden.

#### 2.2.4.2 Fall 2 - MATLAB ist auf dem Rechner installiert

Um `test.exe` (+ `test.ctf`) ablaufen lassen zu können, muss folgender Pfad in die Enviroment-Variable PATH aufgenommen werden:

`$MATLAB\bin\win32`, wobei `$MATLAB` das Root-Verzeichnis der MATLAB-Installation ist.

Beispiel für den CIP-Pool:
`F:\M\Matlab\R2008a\CIP\bin\win32;F:\M\Matlab\R2008a\CIP\bin`

Hierzu wird in den Systemeigenschaften unter Umgebungsvariablen (Abbildung 3) der neue Pfad als Benutzervariable festgelegt (Abbildung 4).

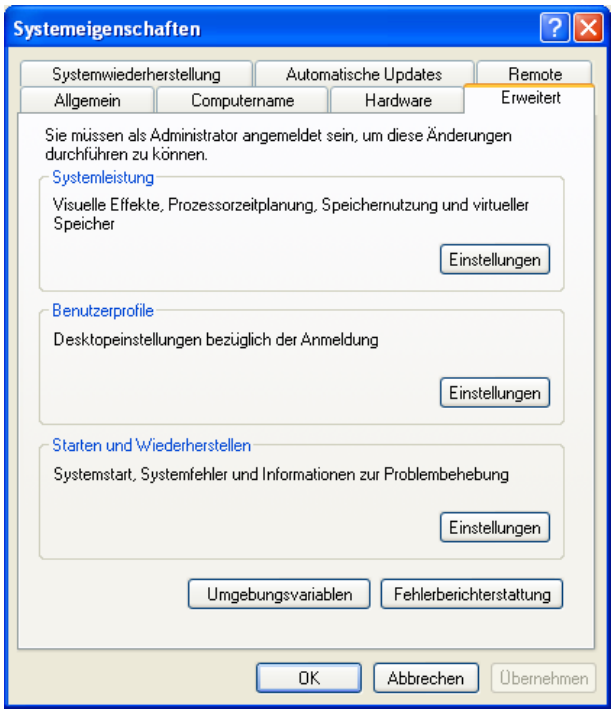


**Abbildung 3: Systemeigenschaften**

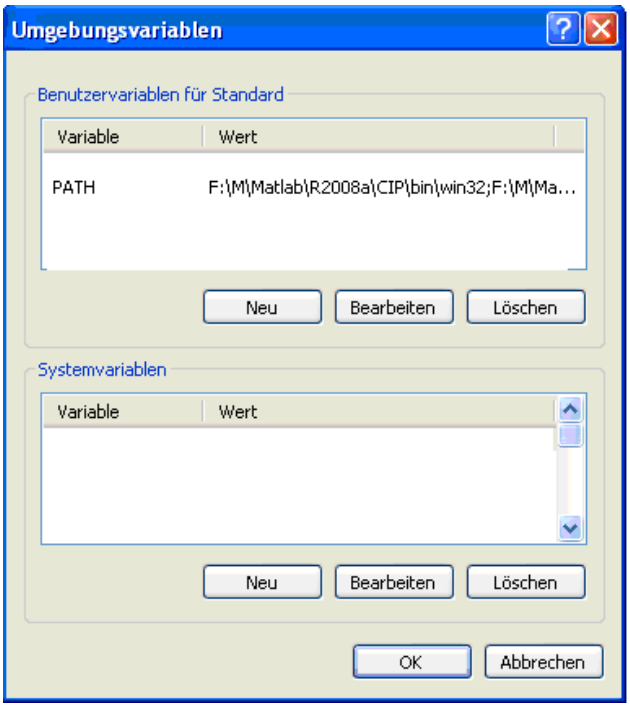


**Abbildung 4: Umgebungsvariablen**

Eine Alternative hierzu ist das Kommandozeilen-Skript `set-matlab-path.cmd`, zu finden im Verzeichnis `K:\WR\MATLAB-COMPILER`. Das Ausführen der Kommandozeilen-Skript erledigt die oben genannten Arbeiten der Pfadeinstellung automatisch.

# 3. Einschränkungen

## *3.1 MATLAB Code*

Der MATLAB Compiler unterstützt fast alle Funktionen von MATLAB. Es gibt allerdings einige Einschränkungen die man beachten muss.

- Skript M-Dateien, also M-Dateien die keine Funktionen beinhalten sondern einfach sequentielle Anweisungen, können nicht kompiliert werden
- Funktionen die nur MEX-Funktionen sind können nicht kompiliert werden
- M-Dateien die Objekte benutzen können nicht kompiliert werden

Außerdem kann der Compiler folgendes nicht kompilieren:

- M-Dateien, die `eval` oder `input` beinhalten. Diese Funktionen erzeugen und benutzen interne Variable, die nur der MATLAB Interpreter verwenden kann.
- Eingebettete MATLAB-Funktionen (Funktionen wie `eig` haben keine eigene M-Datei und können deshalb nicht kompiliert werden). Sie werden in der Regel als Bibliotheksfunktionen der MATLAB Math-Build-in Bibliothek (libmath) dazugelinkt.
- Aufrufe zum Laden (`load`) oder zum Speichern (`save`), die nicht die Namen der Variablen enthalten, in die das Ergebnis geladen oder gespeichert werden soll. Die `load` und `save` Funktionen werden nur mit Variablen-Listen unterstützt.

  Zum Beispiel wird akzeptiert:

```
load(Dateiname,'a','b','c'); % lädt a, b, c aus der Datei
x=load(Dateiname);           % lädt Matrix aus Datei
```

  Nicht erlaubt wäre dagegen:

```
load(Dateiname,var1,var2,var3); % Dies ist nicht erlaubt
```

- Es gibt auch keine Unterstützung für die load und save Optionen –ascii und –mat sowie den variablen Platzhalter (*).

## *3.2 Standalone-Anwendungen*

Die Einschränkungen für den MATLAB Code gelten auch für die Standalone-Anwendungen. Die Funktionen aus Tabelle 3 werden im MEX-Modues unterstützt, nicht aber im Standalone-Modus. Zusätzlich haben Standalone-Anwendungen keinen Zugriff auf:

| | | | |
|---|---|---|---|
| add_block | add_line | applescript | clc |
| close_system | debug | doc | dbclear |
| dbcont | dbdown | dbquit | dbstack |
| dbstatus | dbstep | dbstop | dbtype |
| dbup | delete_block | delete_line | echo |
| edit | fields | get_param | help |
| home | inmem | keyboard | linmod |
| lookfor | mislocked | mlock | more |
| munlock | new_system | open_system | pack |
| publish | rehash | savepath | set_param |
| sim | simget | simset | sldebug |
| type | | | |

**Tabelle 2: Nicht unterstützte Funktionen im Standalone-Modus**

- Aufrufe von MEX-Datei Funktionen, denn die MEX-Datei Funktionen benötigen MATLAB, was nicht der Sinn von Standalone-Anwendungen ist.
- Simulink Funktionen
- Obwohl der MATLAB Compiler M-Dateien die diese Funktionen aufrufen, kompilieren kann, werden sie nicht von der MATLAB C/C++-MathBibiliothek unterstützt. Deshalb wird man bei der Ausführung einen Runtime Error bekommen, außer man schreibt eigene Versionen der nicht unterstützten Routinen.
- Auch globale Variablen sollten vermieden werden. Diese werden zwar übersetzt, aber beim Ausführen einer Stand-Alone-Anwendung können sie zum Absturz führen.
- Beim Aufruf einer Stand-Alone-Anwendung mit Parametern sollte bedacht werden, dass Parameter als Zeichenketten (Strings) eingelesen werden und vor der Verwendung in einer Rechnung in ein Zahlenformat, z.B. Double (str2double), umgewandelt werden müssen, da sonst mit den ASCII-Code-Nummern der Zeichen gearbeitet wird.

# 4. M-Dateien: Konvertierung von Skript- in Funktionsdateien

MATLAB unterstützt zwei Formen von Unterprogrammen:

- Funktions-M-Dateien
- Skript-M-Dateien

Diese Kategorien unterscheiden sich in zweierlei Hinsicht:

- Man kann Argumente an Funktion M-Dateien übergeben, aber nicht an Skript M-Dateien.
- Variablen, die innerhalb einer Funktion M-Datei benutzt werden, sind nur lokal in dieser Funktion; man kann nicht auf diese Variablen vom MATLAB Interpreter aus zugreifen. Im Gegensatz dazu stehen Variablen, die innerhalb von Skript M-Dateien im allgemeinen Workspace von MATLAB benutzt werden. Man kann auf diese Variablen von der MATLAB Befehlszeile auszugreifen.

Der MATLAB Compiler kann weder Skript M-Dateien übersetzen, noch Funktion M-Dateien, die Skript-Dateien aufrufen.

Das Konvertieren einer Skript- in eine Funktions-M-Datei ist sehr einfach:
Um ein Skript in eine Funktion zu konvertieren, fügt man einfach eine function-Zeile am Anfang der M-Datei hinzu.

Zum Beispiel erzeugt die Skript M-Datei houdini.m

```
m=magic(4); % Zuweisen einer 4x4 Matrix zu m
t=m.^3;     % Potenziere jedes Element von m
disp(t);    % Gib das Ergebnis von t aus
```

die Variablen m und t im MATLAB Workspace.
Der MATLAB Compiler kann houdini.m deshalb nicht übersetzen. Konvertieren dieser Skript-Datei in eine Funktion-M-Datei geschieht durch das Hinzufügen einer function-Kopfzeile:

```
function t = houdini
m=magic(4); % Zuweisen einer 4x4 Matrix zu m
t=m.^3;     % Potenziere jedes Element von m
disp(t);    % Gib das Ergebnis von t aus
```

Der MATLAB Compiler kann nun `houdini.m` übersetzen und beim Ausführen von `houdini.mex` wird keine Variable m mehr im MATLAB-Workspace erzeugt. Wenn es z.B. notwendig wäre, auch auf m vom MATLAB-Workspace aus zuzugreifen, dann können Sie die erste Zeile der Funktion folgendermaßen ändern:

```
function [m,t] = houdini;
```

# 5. Online-Dokumentation

Die Online-Dokumentation ist auf folgender Seite zu finden:

http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/