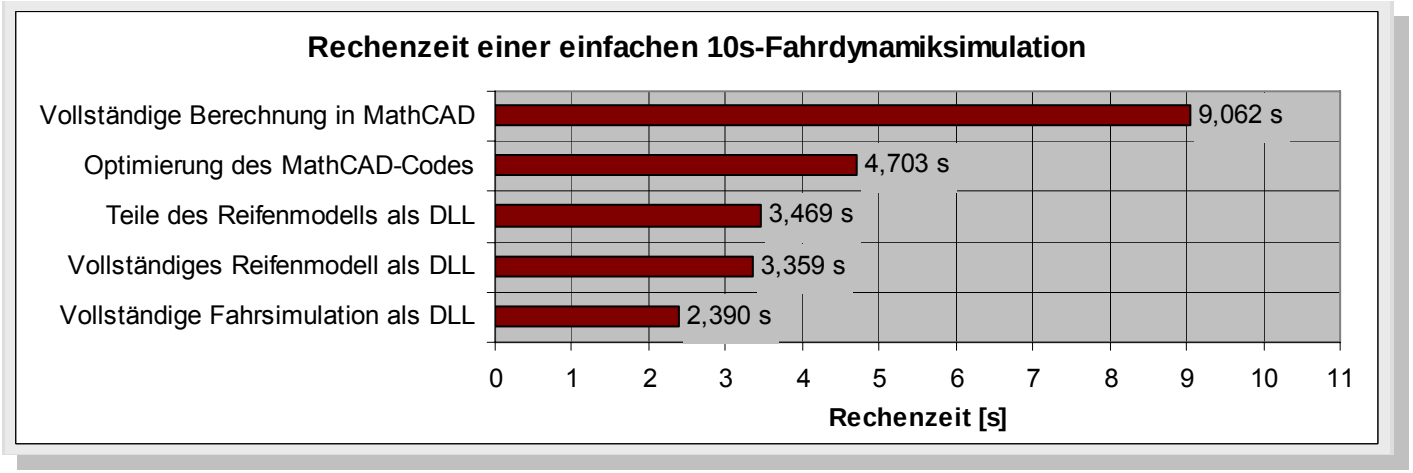
TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY

GERMANY

Entwicklung von dynamischen Bibliotheken (DLL) für Mathcad®

– Effizienzsteigerung, Quellcodeschutz, Codeportierung –



28.04.2009

Dipl.-Ing. Christian Meißner

TU Chemnitz

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY

GERMANY

Entwicklung von dynamischen Bibliotheken (DLL) für Mathcad®

– Effizienzsteigerung, Quellcodeschutz, Codeportierung –

Agenda:

- Einführung
- Effizienzsteigerung durch DLL‘s
- Zusatzfunktionen durch DLL‘s
- Quellcodeschutz durch DLL‘s
- Codeportierung – Erstellung von Mathcad®-DLL‘s
- Zusammenfassung und Ausblick

28.04.2009

Dipl.-Ing. Christian Meißner

TU Chemnitz

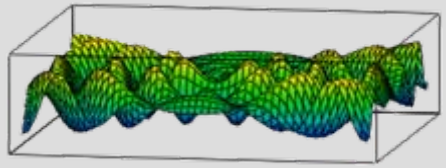
Einführung

Vorteile von Mathcad® (Auswahl)

Dateneingabe

Excel MATLAB®

Diagrammdarstellung



Große Funktionsbibliothek

Bildverarbeitung, Dateizugriff, Datenanalyse, DGL-Löser, Matrizen, Wahrscheinlichkeit, Zeichenfolgen, ...

Integrierte Dokumentation

1 Grundlagen

1.1 Kinematik und Kinetik

Drehzahl: $\omega_S := \omega_H \cdot i_0 + \omega_T \cdot (1 - i_0)$

$\omega_S = 261.5 \frac{1}{s}$

Mathcad®

Pro/E-Schnittstelle

Pro|ENGINEER

Gleichungen in Standdardnotation

$$f(x,a) := \frac{x^2 - \sin(a \cdot x)}{a - x}$$

Erweiterung durch DLL‘s

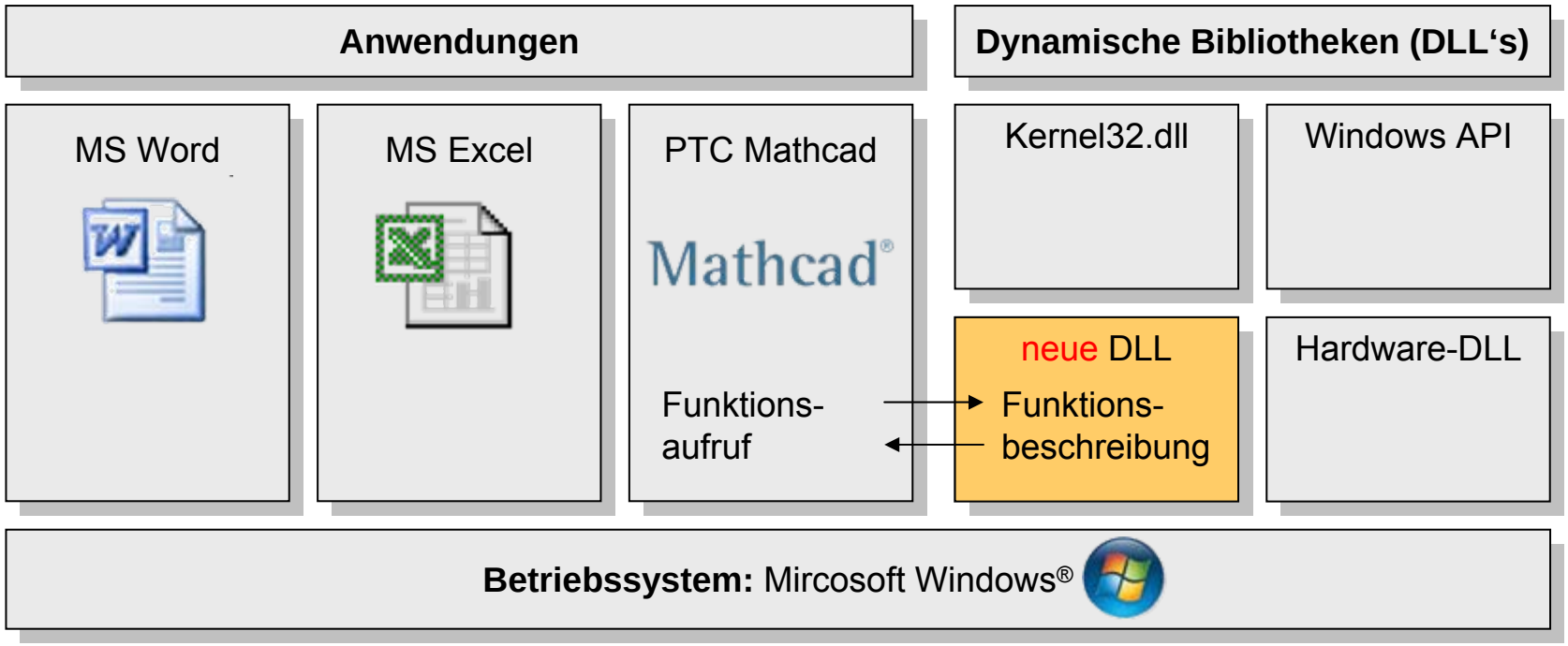
```
#include <stdio.h>
void main(char *text)
```

Erweiterungspakete (Extensionpacks)

Datenanalyse, Bildverarbeitung, Signalverarbeitung, Wavelets

Einführung

Das Windows DLL-Konzept



Aktionen:

- Beim Programmstart: Laden aller DLL‘s in den Speicher (z.B. in „C:\mathcad\userfi\“)
- Beim einem Funktionsaufruf in Mathcad: In Mathcad definierte Funktion?
  Mathcad – interne Funktion?
  DLL – Funktion?

Vorteile:

- einfache Erweiterung von bestehender Software (zusätzliche Funktionen)
- einfache Aktualisierung von Funktionen (Austausch der DLL)

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY

GERMANY

Entwicklung von dynamischen Bibliotheken (DLL) für Mathcad®

– Effizienzsteigerung, Quellcodeschutz, Codeportierung –

Agenda:

Einführung	✓
Effizienzsteigerung durch DLL‘s	
Zusatzfunktionen durch DLL‘s	
Quellcodeschutz durch DLL‘s	
Codeportierung – Erstellung von Mathcad®-DLL‘s	
Zusammenfassung und Ausblick	

28.04.2009

Dipl.-Ing. Christian Meißner

TU Chemnitz

Steigerung der Recheneffizienz durch DLL's

Einfache Berechnungen in Mathcad

z.B. Dimensionierungs-gleichungen für Maschinen-elemente (Wellen, Zahnräder, Passfedern, ...)

Pressverbindung:

Fugendurchmesser:	$D_F := 30mm$
Fugenlänge:	$l_F := 42mm$
Flächenpressung:	$p := 100MPa$
Haftbeiwert:	$\nu := 0.1$
Übertragbares Drehoment:	$T := \frac{\pi}{2} \cdot D_F^2 \cdot l_F \cdot \nu \cdot p$
	$T = 593.8\,N \cdot m$

Komplexere Berechnungen in Mathcad

z.B. Mehrkörpersimulation, numerisches Lösen nicht-linearer Differenzial-gleichungen, Verarbeitung großer Datenmengen

Rechenaufwand steigt

Forderung nach höherer Rechengeschwindigkeit steigt

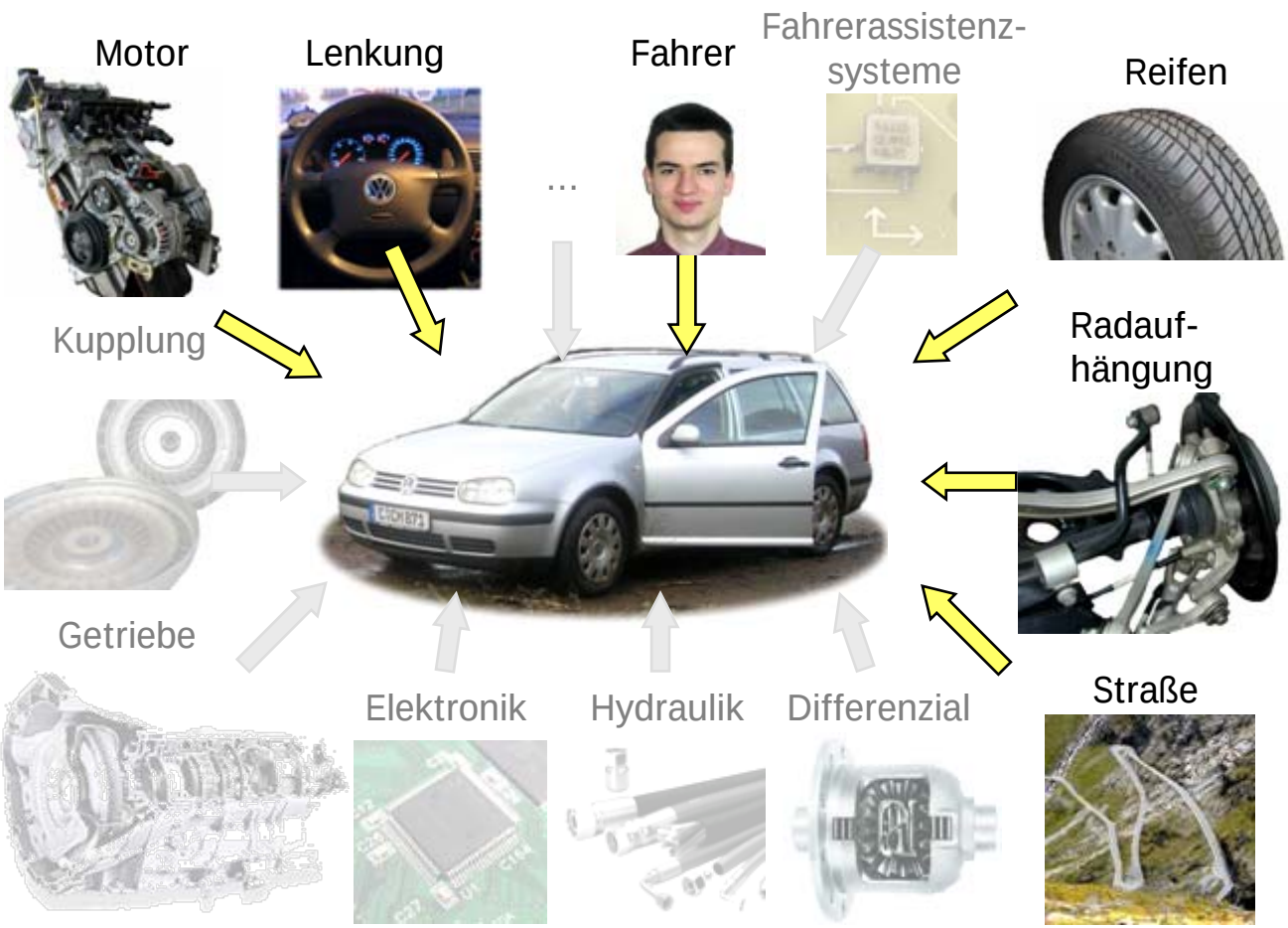
Lösungsmöglichkeiten:

- Schnellerer PC
- Batchbetrieb über Nacht
- Parallelberechnung (Cluster)
- Codeoptimierung
- Code-Compilierung
- Maschinennahe Programmierung

Steigerung der Recheneffizienz durch DLL's

Beispiel: Fahrzeugsimulation – vereinfachtes Modell

Komponenten des Simulationsmodells



Steigerung der Recheneffizienz durch DLL's

Beispiel: Fahrzeugsimulation – vereinfachtes Modell

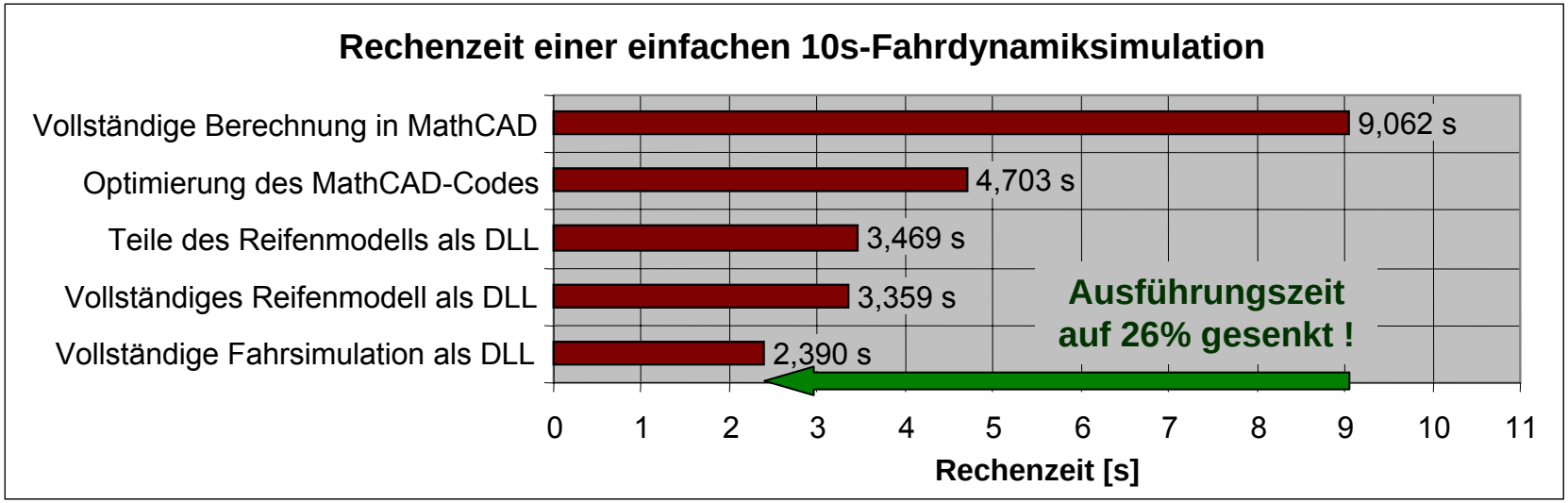
Ziel: Erhöhung der Rechengeschwindigkeit

Teil 1: Optimierung des Mathcad®-Programmcodes

- Zusammenfassen von Funktionen, Preprozessing, Interpolationsroutinen, ...

Teil 2: Auslagerung von rechenintensiven Funktionen in eine DLL

- Schrittweises Auslagerung von Funktionen
- Zwischentest der Funktionen
- Reduktion der Übergabeparameter durch Ablegen im DLL-internen Speicher



Geschwindigkeitssteigerung bei dieser Anwendung um Faktor 3,8 !

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY
GERMANY

Agenda:

Einführung	✓
Effizienzsteigerung durch DLL‘s	✓
Zusatzfunktionen durch DLL‘s	
Quellcodeschutz durch DLL‘s	
Codeportierung – Erstellung von Mathcad®-DLL‘s	
Zusammenfassung und Ausblick	

Dipl.-Ing.
C. Meißner

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY
GERMANY

Agenda:

Einführung	Erweiterte Dateifunktionen
Effizienzsteigerung durch DLL‘s	Shared Memory
Zusatzfunktionen durch DLL‘s	Onlinedarstellung von 3D-Grafiken
Quellcodeschutz durch DLL‘s	Hardwareschnittstelle
Codeportierung – Erstellung von Mathcad®-DLL‘s	Datum- und Zeitfunktion
Zusammenfassung und Ausblick	...

Dipl.-Ing.
C. Meißner

Zusatzfunktionen durch DLL‘s

Erweiterte Dateifunktionen

Vorteil von Mathcad®:

- MathCAD bietet sehr viele Möglichkeiten zum Dateiimport (z.B. aus EXEL, Matlab, ASCII-Text, ...)

Nachteil:

- Massenabfertigung vom Messwertdatein erfordert Dateinamenübergabe als Variable und sollte auch in einem Programm aufrufbar sein
- → dazu existieren die Funktionen

READFILE

⊖ vorher händische Umwandlung der Dezimaltrennzeichen

PRNLESEN

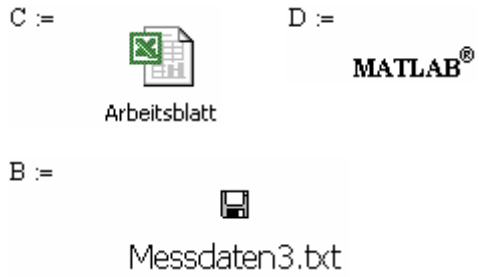
⊖ akzeptiert keine Spaltenbezeichnungen

Wunsch:

- tolerantere Datei-Lese-Funktion mit grundlegenden Konvertierungmöglichkeiten

Lösung:

- DLL mit Zusatzfunktion



Messdatenbeispiel:

ch0	ch1	ch2
Zeit	Spannung	Strom
[s]	[V]	[A]
0,00	7,21	1,23
0,01	7,98	1,15
0,02	8,34	1,01
...	...	...

Zusatzfunktionen durch DLL's

Erweiterte Dateifunktionen

DLL: „SimulationTools_MathCAD_Dateifunktionen.dll"

Quelle: www.simulationtools.de (frei zum Download)

Funktionen: Datei lesen: **`ST_load_datafile( Dateiname , manip )`**

Dateiname = zfinvek(„C:\Messwerte.txt")

manip = (Komma, Datenreduktion, Mittelwert)^T

Beispiel:

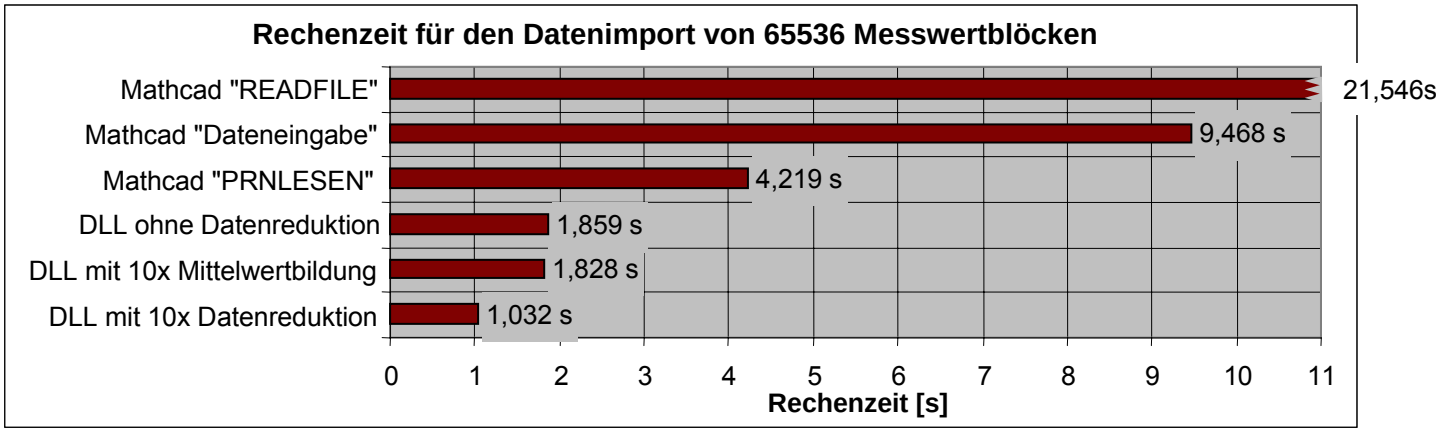
Dateiname := zfinvek("C:\Messwerte.txt")

$$manip := \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

… Kommas in Punkte umwandeln
… jede Zeile verarbeiten (keine Datenreduktion)
… keine Mittelwertbildung

DATA := ST_load_datafile(Dateiname,manip)

Geschwindigkeitsvergleich:



Zusatzfunktionen durch DLL‘s

Shared Memory

Nachteil von Mathcad®:

- Debugging-Funktion setzt die Rechengeschwindigkeit drastisch herab
- Keine globalen Variablen in- und außerhalb von Funktionen

Wunsch:

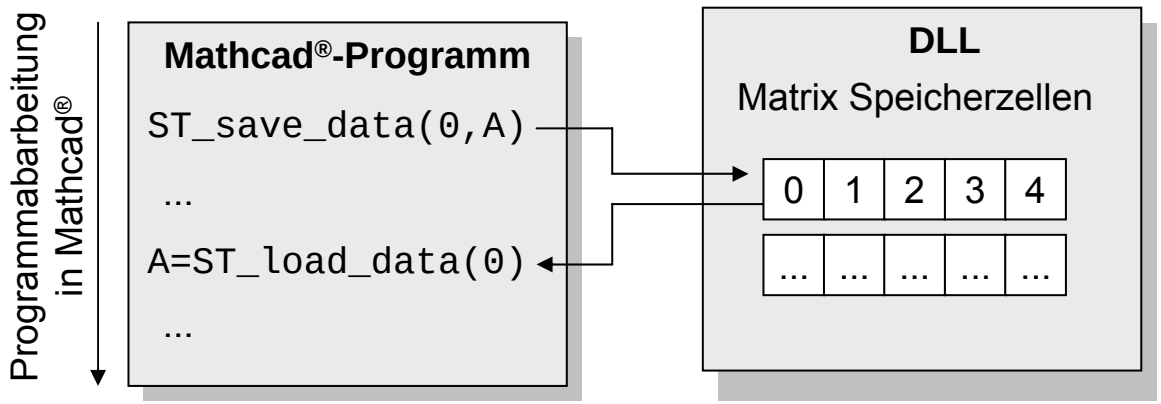
- globaler Speicherbereich, der von jeder Stelle im Programm aus zugänglich ist

Sinn:

- Fehler in aufwändigen Simulationen können schneller lokalisiert werden

Lösung:

- Speicherbereich einer DLL nutzen (Shared Memory = geteilter Speicher)



Zusatzfunktionen durch DLL's

Shared Memory

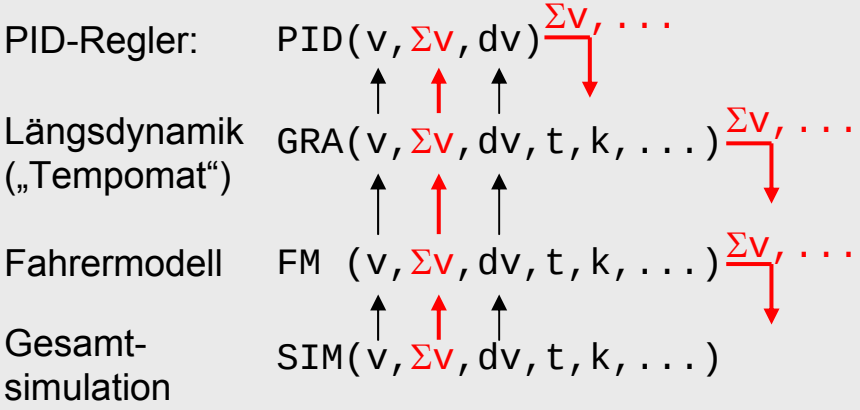
Beispielhafte Verwendung zum Einsatz eines PID-Reglers in einer Fahrdynamiksimulation:

PID-Regler: $y = \underbrace{K_P \cdot v}_{\text{Proportional-}} + \underbrace{\frac{1}{T_N} \cdot \int v \cdot dt}_{\text{Integral-}} + \underbrace{T_V \cdot \frac{dv}{dt}}_{\text{Differentzialglied}}$

- numerisch integrieren = schrittweise addieren
- Veränderung bei jedem Funktionsaufruf → globaler Speicher erforderlich

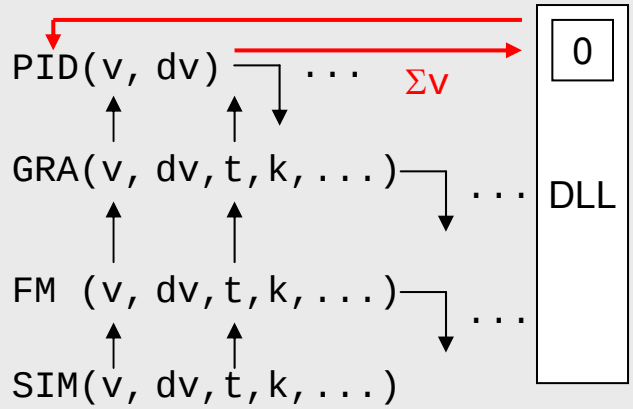
Klassische Programmierung

- „Durchschleifen" des Integrationswertes bis zur untersten Ebene



Programmierung mit DLL-Funktion

- Integrationswert im Shared Memory zwischenspeichern, beim nächsten Funktionsaufruf wieder laden



- DLL-Speicher verringert die Anzahl der Übergabeparameter → übersichtlicher, schneller
- Achtung! Falsche Speicherzellenzuordnung kann abgelegte Werte überschreiben!

Zusatzfunktionen durch DLL‘s

Online-Darstellung von 3D-Grafiken

Nachteil von Mathcad®:

- sequenzielle Abarbeitung der Befehle
- → Diagrammausgabe kann erst nach vollständiger Simulation erfolgen

Wunsch:

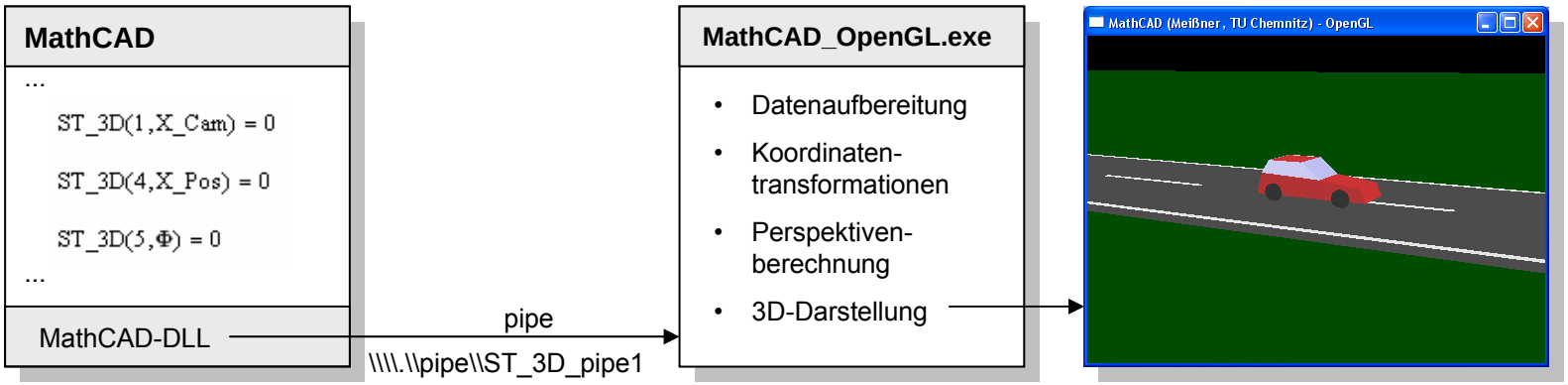
- 3D-Darstellung während der Berechnung

Sinn:

- Fehler in aufwändigen Simulationen können schneller erkannt werden

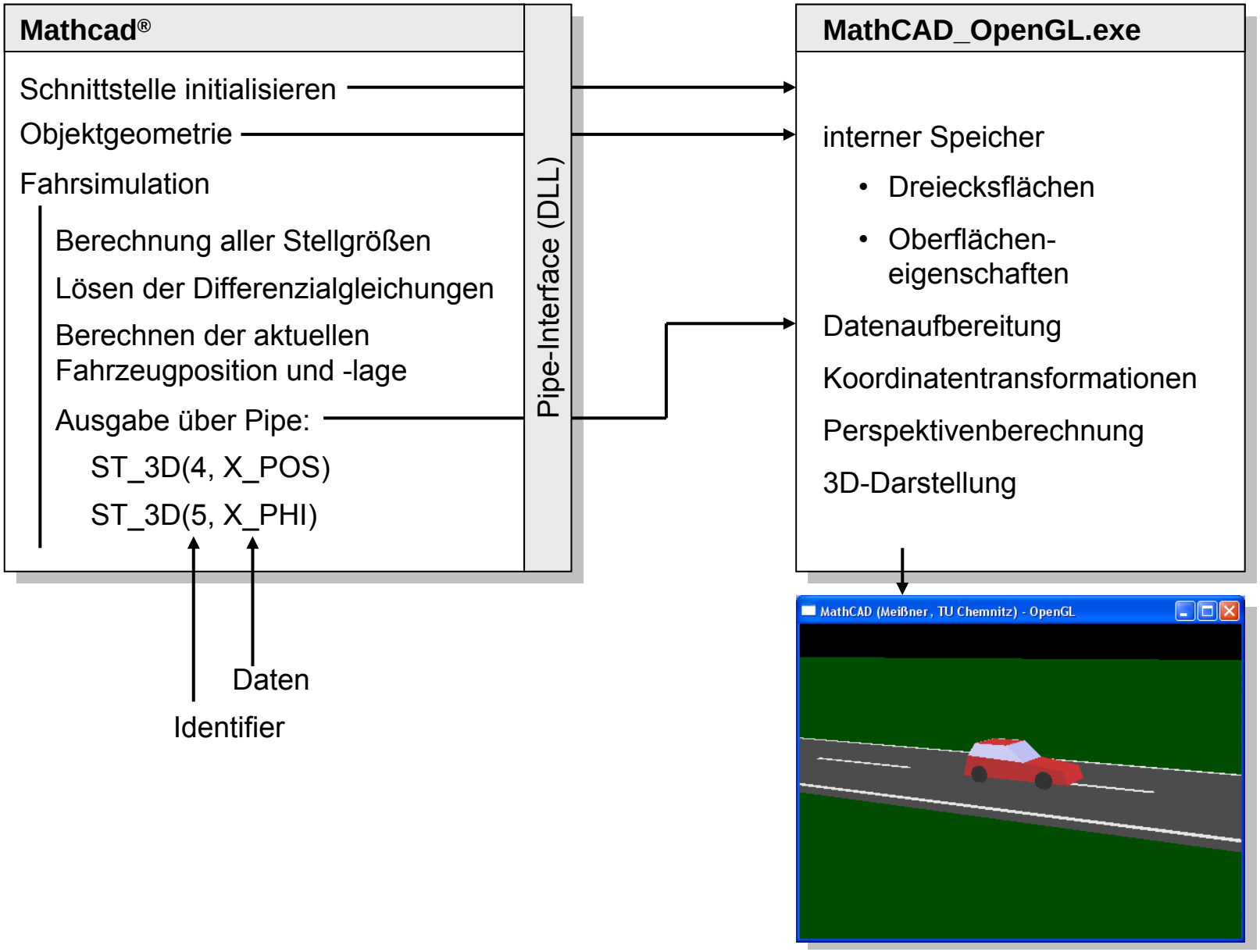
Lösung:

- externe Software zur 3D-Darstellung
- Kommunikation über DLL und Pipe
- Konzentration des Entwicklers auf das Wesentliche



Zusatzfunktionen durch DLL's

Online-Darstellung von 3D-Grafiken



Zusatzfunktionen durch DLL‘s

Hardwareschnittstelle

Nachteil von Mathcad®:

- Datenein- und –ausgaben erfolgen über Tastatur / Datei / Komponenten etc.
- → Keine Anbindung von Mess- und Regelungshardware

Wunsch:

- Ein- und Ausgabe von Daten über externe Hardware

Sinn:

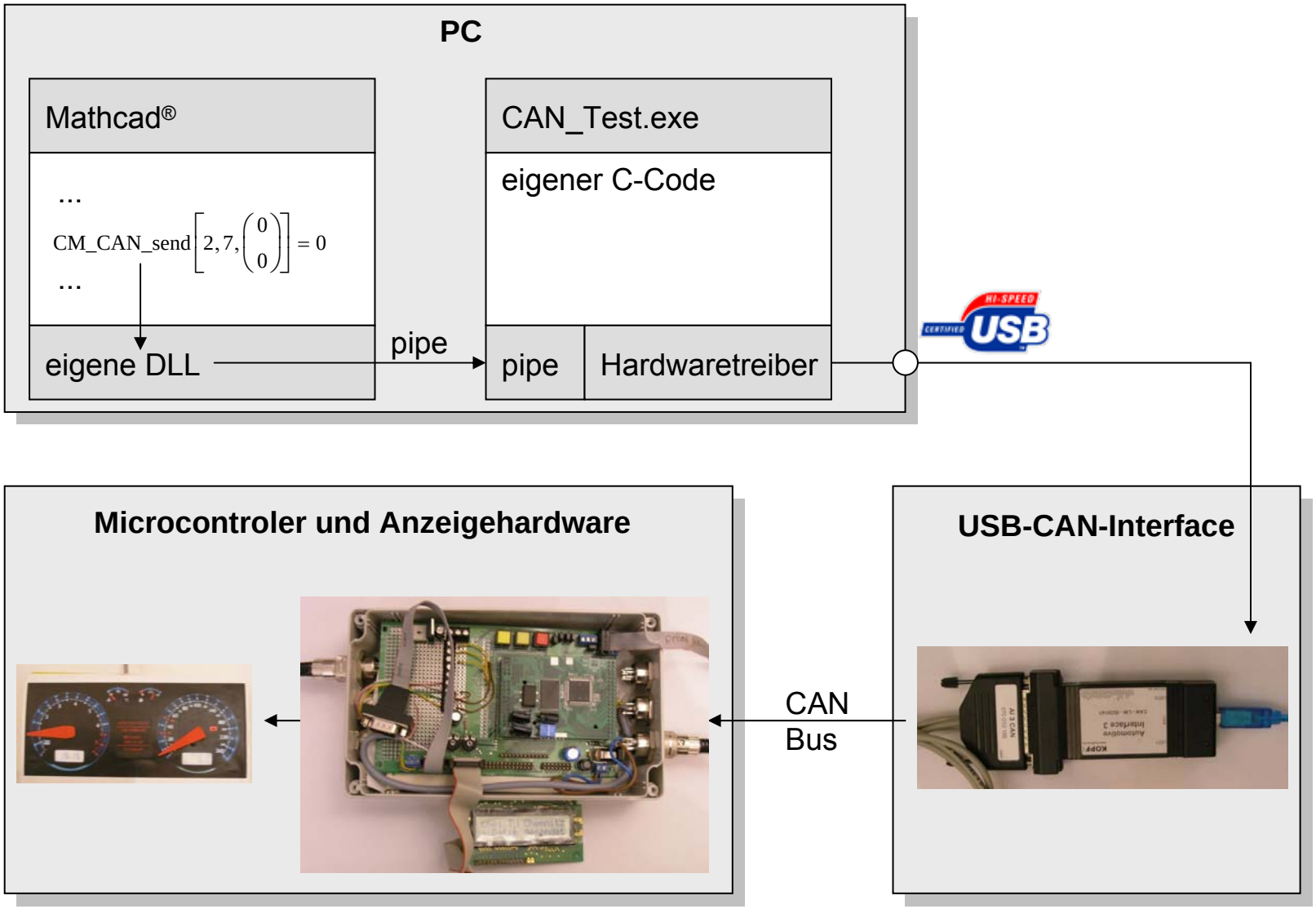
- Online-Auswertung von Messungen, Steuerung

Lösung:

- externe Software zur Hardwareansteuerung
- Kommunikation über DLL und Pipe

Zusatzfunktionen durch DLL‘s

Hardwareschnittstelle



TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY

GERMANY

Agenda:

Einführung	✓
Effizienzsteigerung durch DLL‘s	✓
Zusatzfunktionen durch DLL‘s	✓
Quellcodeschutz durch DLL‘s	
Codeportierung – Erstellung von Mathcad®-DLL‘s	
Zusammenfassung und Ausblick	

Dipl.-Ing.
C. Meißner

Quellcodeschutz

Ziel:

- Schutz des geistigen Eigentums
- Keine Einsicht in die Programmroutinen bei Weitergabe an Dritte

Möglichkeiten in Mathcad:

- passwortgeschütztes Sperren und Ausblenden von Regionen

Testregion - Fr Apr 17 12:30:46 2009

Problem:

- Suche bei GOOGLE liefert viele Einträge
  zum „Knacken“ der Passwörter gesperrter Regionen

→ Quellcode ist nicht so sicher, wie man denkt

Lösung:

- Funktionen als DLL schreiben
- DLL liegt als compilierter Maschinencode vor
  und kann nicht eingesehen werden

Kritik:

- auch DLL‘s lassen sich „decompilieren“

→ dieser decompilierte Code ist aber wesentlich schwerer zu durchdringen

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY
GERMANY

Agenda:

Einführung	✓
Effizienzsteigerung durch DLL‘s	✓
Zusatzfunktionen durch DLL‘s	✓
Quellcodeschutz durch DLL‘s	✓
Codeportierung – Erstellung von Mathcad®-DLL‘s	
Zusammenfassung und Ausblick	

Dipl.-Ing.
C. Meißner

Codeportierung – Erstellung von Mathcad®-DLL's

Kurzanleitung:

Compiler:
- Microsoft Studio Entwicklungsumgebung
- Open Watcom (www.openwatcom.org)

Codeaufbau:
- Sechs Funktionsblöcke:

1	Include-Anweisungen für andere verlinkte C-Dateien
2	Definition der Mathcad®-spezifischen Routinen
3	Definition der Fehlermeldungen
4	Benutzerdefinierte Funktionen als funktionaler Kern der DLL
5	Beschreibung der Funktionen und deren Übergabeparameter
6	DLL-Hauptroutine als Eintrittspunkt

Integration in Mathcad®:
- Einbinden/Linken der von PTC bereitgestellten Dateien „Mcadincl.h " und „Mcaduser.lib"
- optionaler Eintrag in USER_DE.xml

Hilfeeinträge:
- Link HTML-Dokument in USER_DE.xml

Hilfe:
- Mathcad Hilfe → Developer's reference
- Website www.simulationtools.de

Codeportierung – Erstellung von Mathcad®-DLL's

Problem:

- Mathcad bietet keine Schnittstelle z.B. zum Erzeugen con C-Code

übliche Lösung:

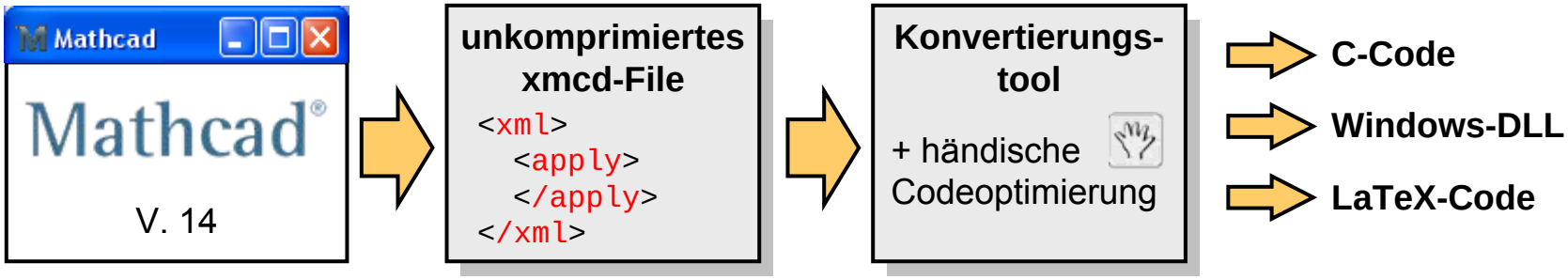
- Neu- bzw. Abprogrammieren des Mathcad-Sheetes

→ extrem zeit- und ressourcenaufwändig

Neue Lösung:

- Konvertierungstool von SimulationTools.de scannt Mathcad-Sheet und erstellt halbautomatisch C-Code
- einige händische Anpassungen notwendig
- Konvertierung wird als Dienstleistung zur Verfügung gestellt (Tool wird dabei genutzt)

→ günstigere Konvertierung, als bei eigener, manueller Konvertierung



Codeportierung – Erstellung von Mathcad®-DLL‘s

Nutzung der konvertierten Daten:

C-Code	• Nutzung in eigener Software • Einbettung in andere Simulationssoftware • Plattformunabhängiger Code z. B. für Simulationen unter LINUX
Windows-DLL	• Nutzung in Mathcad® zur Effizienzsteigerung • Quellcodeschutz: Sichere Weitergabe der DLL‘s an andere Mathcad®-Nutzer
LaTeX-Code	• Dokumentation wissenschaftlicher Arbeiten • Veröffentlichungen

TECHNISCHE UNIVERSITÄT CHEMNITZ

CHEMNITZ UNIVERSITY OF TECHNOLOGY

GERMANY

Agenda:

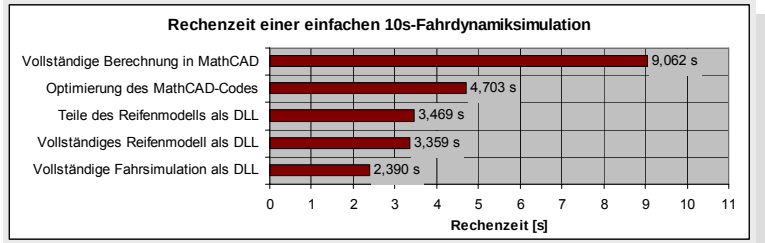
Einführung	✓
Effizienzsteigerung durch DLL‘s	✓
Zusatzfunktionen durch DLL‘s	✓
Quellcodeschutz durch DLL‘s	✓
Codeportierung – Erstellung von Mathcad®-DLL‘s	✓
Zusammenfassung und Ausblick	

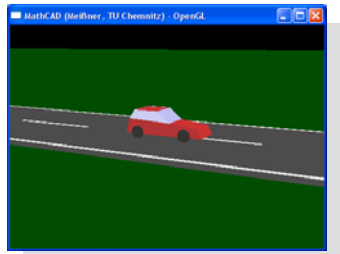
Dipl.-Ing.
C. Meißner

Zusammenfassung und Ausblick

Zusammenfassung:

- Erweiterung von Mathcad® durch DLLs (C/C++)
- Steigerung der Rechengeschwindigkeit
- Zusatzfunktionen (Messwertdatei lesen, Onlinedarstellung von 3D-Grafiken, Hardwareschnittstelle)
- Quellcodeschutz durch DLLs
- Codeportierung (www.simulationtools.de)





Ausblick:

Haben Sie Fragen zu DLLs?

Möchten Sie eigene Mathcad®-Funktionen als DLL einsetzen?

Möchten Sie selbst DLLs erstellen?

Wenden Sie sich an uns!

TECHNISCHE UNIVERSITÄT CHEMNITZ

Dipl.-Ing.
Christian Meißner

Technische Universität Chemnitz
Institut für Konstruktions- und Antriebstechnik
Reichenhainer Str. 70
09126 Chemnitz
Tel.: 0371 – 531 – 33715
Fax: 0371 – 531 – 23329
Email: christian.meissner@mb.tu-chemnitz.de

Entwicklung von dynamischen Bibliotheken (DLL) für Mathcad® – Effizienzsteigerung, Quellcodeschutz, Codeportierung –

Dipl.-Ing. Christian Meißner
Institut für Konstruktions- und Antriebstechnik, Technische Universität Chemnitz,
Professur Maschinenelemente, Email: christian.meissner@mb.tu-chemnitz.de
Internet: www.simulationtools.de

Abstract

One advantage of Mathcad® is the well arranged worksheet. Mathematic equations are not displayed in confusing chains of characters but in clear mathematical notation. The equations normally are checked of plausibility including units after writing. Many functions – even complex solving algorithms – are already defined. Furthermore formatted comments lead to a good documentation, e.g. for calculation reports. Hence the calculation sheets are developed in a very short time and they are easy to comprehend for new users.
However every advantage is bought by a disadvantage. Therefore there is a less calculation speed, limited possibilities for code protection and only few possibilities of communication with other programs. These disadvantages can be removed by using dynamic link libraries (DLL's).
This article gives an introduction of programming DLL's in Mathcad® and shows some examples to increase calculation efficiency, code protection and code transfer.

Zusammenfassung

Ein Vorteil von Mathcad® ist das übersichtliche Arbeitsblatt. Mathematische Gleichungen werden nicht als verwirrende Kette von Buchstaben und Zahlen dargestellt, sondern in eindeutiger mathematischer Notation. Die Gleichungen werden standardmäßig nach der Eingabe auf Plausibilität geprüft. Viele Funktionen – auch komplexe Lösungsalgorithmen – werden bereits mitgeliefert. Weiterhin können formatierte Textfelder für eine gute Dokumentation genutzt werden, z.B. für Berechnungsnachweise. Daher kann das Arbeitsblatt in sehr kurzer Zeit entwickelt werden und ist für neue Anwender leicht nachzuvollziehen.
Jedoch wird jeder Vorteil durch einen Nachteil erkauft. So liegen eine geringere Rechengeschwindigkeit, eingeschränkter Codeschutz und nur wenige Möglichkeiten zur Kommu-

nikation mit anderen Programmen vor. Diese Nachteile können durch dynamische Bibliotheken verringert werden.
Dieser Artikel stellt die Vorteile der DLL-Programmierung für Mathcad® dar und zeigt einige Beispiele zur Erhöhung der Rechengeschwindigkeit, zum Quellcodeschutz und zur Codeportierung.

Grundlagen zu Windows-DLL's

Basics on Windows DLL's

Das DLL-Konzept (The DLL-Concept)

Bei der Softwareentwicklung werden einige Programmteile für viele verschiedene An–wendungen benötigt. Um den Speicheraufwand zu reduzieren, werden diese Programmteile in einer separaten Datei – der DLL – als Maschinencode abgelegt. Beim Starten der eigent–lichen Anwendung wird dann diese DLL mit in den Speicher geladen, wodurch die darin enthaltenen Funktionen von der Anwendung aufgerufen werden können als stünden sie mit im Programmcode.
Neben der dadurch möglichen Speicherplatzreduktion besteht ein weiterer Vorteil der Verwendung von DLL's in der einfacheren Aktualisierung oder Erweiterung von bestehenden Anwendungen. So können bestehende Funktionen verbessert bzw. neue hinzugefügt werden. Dies geschieht einfach durch den Austausch der alten DLL durch eine neuere.

Sicherheitsaspekte (Safety aspects)

Da die ausgelagerten Funktionen einer Anwendung durch den einfachen Austausch einer DLL verändert werden können, besteht die Gefahr zum Missbrauch. Das kann bei vorhandenen Programmrechten soweit führen, dass beim Aufruf Lese-, Schreib- und Löschaktionen auf dem Datenträger ohne Rückmeldung durchgeführt werden oder vertrauliche Daten über eine Internetverbindung versandt werden können.

Erstellung von DLL's für Mathcad®

Creation of DLL's for Mathcad®

Programmiersprache, Compiler und Linker (Programming language, compiler and linker)

Die DLL's werden vorrangig in C oder C++ geschrieben. Diese Programmiersprache ist sehr weit verbreitet, wodurch man zahlreiche Dokumentationen und zusätzliche Bibliotheken im

Internet findet. In Mathcad® selbst findet man unter dem Menüpunkt „Hilfe → Developer's Reference" eine englischsprachige Kurzanleitung zum Erstellen von DLL's.
Zur Compilierung des C-Quellcodes stehen zahlreiche Compiler und Linker zur Verfügung. Die Mathcad®-Hilfe z. B. beschreibt die Verwendung der Visual Studio Entwicklungsumgebung, ein frei verfügbarer Compiler ist z. B. „OpenWatcom"[1].

Automatische Erzeugung des C-Quellcodes (Automatic code generation)

Mathcad® bietet keine Möglichkeit, aus einem bestehenden Mathcad®- Dokument direkt C-Code zu erzeugen, wie es von anderen Simulationstools bekannt ist (z. B. Matlab®). Daher muss die Erstellung des C-Codes händisch durch Neu- bzw. Abprogrammierung erfolgen. Dieser z. T. sehr aufwändige Vorgang rechtfertigt nicht in jedem Fall das Auslagern von zeitkritischen Funktionen in eine DLL.
Jedoch wurde von dem Unternehmen „SimulationTools"[2] eine Software entwickelt, welche Mathcad®-Dokumente halbautomatisch in C-Code umwandelt. Aufgrund des unterschiedlichen Programmierstils in Mathcad® und C sind händische Anpassungen notwendig. Diese bieten auch die Möglichkeit zur weiteren Optimierung. Daher wird dieses Konvertierungstool bei dem Unternehmen „SimulationTools" nur für interne Zwecke verwendet und dem Kunden so eine günstige Konvertierungsdienstleistung angeboten.

Einführendes Beispiel (Indroduction example)

Um grundlegende Aspekte einer DLL zu zeigen, wird im Folgenden eine sehr einfache theoretische Beispielfunktion gewählt. Darin wird die Summe aller Elemente einer Matrix berechnet. Zwei Implementierungsmöglichkeiten in Mathcad® sind in Bild 1 dargestellt.

a)
$$\text{Elementsumme\_1}(in) := \left| \begin{array}{l} sum \leftarrow 0 \\ \text{for } i \in 0..\text{zeilen}(in) - 1 \\ \quad \text{for } j \in 0..\text{spalten}(in) - 1 \\ \quad\quad sum \leftarrow sum + in_{i,j} \\ sum \end{array} \right.$$

b)
$$\text{Elementsumme\_2}(in) := \sum_{i=0}^{\text{zeilen}(in)-1} \sum_{j=0}^{\text{spalten}(in)-1} in_{i,j}$$

Bild 1: Realisierungsmöglichkeiten zur Elementsumme in Mathcad®

Fig. 1: Possibilities to implement the matrix sum in Mathcad®

Zur Realisierung dieser Funktion als DLL sind sechs gedankliche Funktionsblöcke im C-Quellcode notwendig:

---

[1] www.openwatcom.org

[2] www.simulationtools.de

1. Include-Anweisungen für andere verlinkte C-Dateien
2. Definition der Mathcad®-spezifischen Routinen
3. Definition der Fehlermeldungen,
   welche als Textmeldung auf dem Mathcad®-Arbeitsblatt erscheinen sollen
4. Benutzerdefinierte Funktionen als funktionaler Kern der DLL
5. Beschreibung der Funktionen und deren Übergabeparameter
   für die Einbindung der benutzerdefinierten Funktionen in Mathcad®
6. DLL-Hauptroutine als Eintrittspunkt (entry point) für das Betriebssystem

Die Umsetzung dieser Funktionen in eine DLL zeigt der Quellcode in Anhang 1.

Verwendung von DLL's in Mathcad®

Using the DLL's in Mathcad®

Aufruf der DLL-Funktion in Mathcad® (Calling of DLL function in Mathcad®)

Alle in dem Installationsverzeichnis „...\userfi\" stehenden DLL's werden beim Programmstart von Mathcad® geladen und die darin enthaltenen Funktionen registriert. Wurde eine DLL aktualisiert muss daher Mathcad® vollständig geschlossen und neu gestartet werden.
Der Aufruf der benutzerdefinierten Funktion erfolgt mit dem Namen, mit dem sie in der DLL definiert wurde (Bild 2).

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{DLL\_Elementsumme}(A) = 45$$

Bild 2: Verwendung der beispielhaften DLL-Funktion in Mathcad®
Fig. 2: Using the exemplary DLL function in Mathcad®

Nachteilig auf den Bedienkomfort der Funktionen wirkt sich die fehlende Rückverfolgbarkeit aus. Während Mathcad® bei eigenen Funktionen eine Neuberechnung veranlasst wenn Eingabewerte bzw. Abhängigkeiten zu anderen Funktionen geändert wurden, muss hier häufig die Berechnung mit F9 bzw. Strg-F9 erzwungen werden. Mathcad® kann natürlich nicht wissen, wie evtl. mehrere in einer DLL definierte Funktionen sich gegenseitig beeinflussen.

Beispielhafter Vergleich der Rechengeschwindigkeit (Exemplary comparison of calculation speed)

Ein entscheidender Vorteil bei der Verwendung von DLL's ist die höhere Rechengeschwindigkeit beim Ausführen des C-Codes. Bild 3 zeigt die Rechenzeit der Beispielfunktion im

Vergleich zu den Mathcad®-Funktionen aus Bild 1 bis zu einer Matrixgröße von 1 Mio. Elemente.

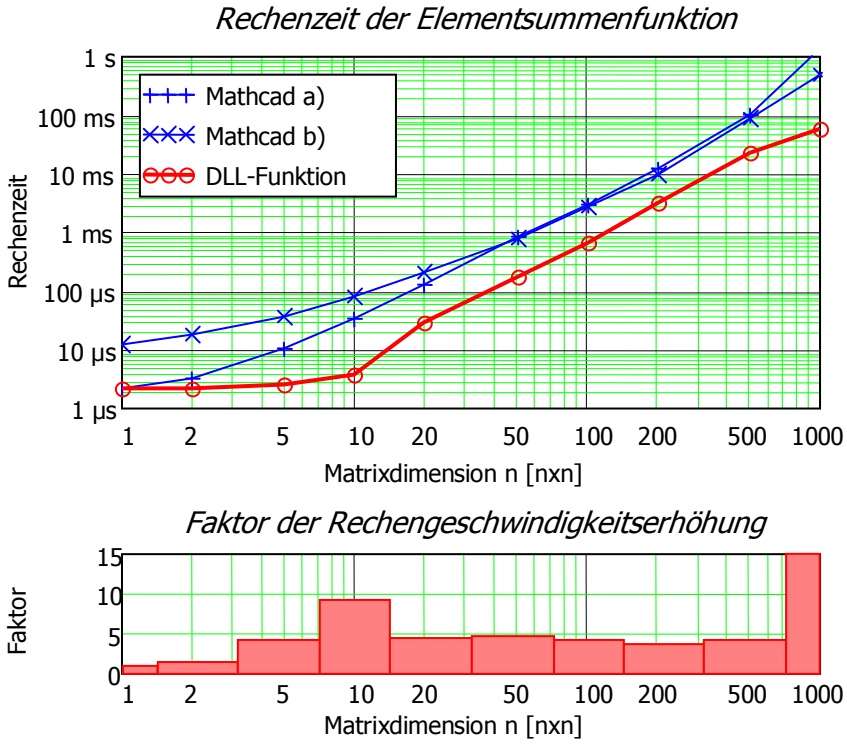


Bild 3: Vergleich der Rechenzeit verschiedener Funktionen[3]

Fig. 3: Comparison of the calculation time of different functions

Da auch die Datenübergabe an die DLL-Funktion eine gewisse Rechenzeit erfordert, ist der Geschwindigkeitsvorteil umso größer, je weniger Daten übergeben werden und je höher der Rechenaufwand innerhalb der Funktion ist. Selbst bei einer so einfachen Rechenoperation – zwei Schleifen und eine Addition – ist die DLL-Implementierung bei mittlerer Matrizengröße bis zu einem Faktor von 10 schneller als die Implementierung in Mathcad®.

Matrizenoperationen wie Multiplikationen oder die Bildung der Inversen sind in Mathcad® schon sehr weit optimiert. Eine Auslagerung in eine DLL bringt in vielen Fällen keinen weiteren Geschwindigkeitsvorteil.

Beispiel für die Effizienzsteigerung

Examples for increasing efficiency

Als Beispiel für Steigerung der Recheneffizienz wird im Folgenden die Simulation der Fahrzeugdynamik verwendet.

[3] Testumgebung: CPU: AMD 3800+ singlecore, RAM: 1,5 GB, Betriebssystem: Windows XP SP2, Mathcad-Version: 14.0 M020

Technischer Hintergrund (Technical background)

Bei der Entwicklung von Fahrzeugantrieben gewinnt die Simulation der Fahrzeugdynamik immer mehr an Bedeutung. In der Konstruktionsphase sind die im realen Fahrbetrieb zu erwartenden mechanischen Belastungen für die Auslegung der Bauteile von besonderer Bedeutung. Eine Dauerfestigkeitsauslegung auf die physikalischen Grenzbelastungen würde zu überdimensionierten Bauteilen und damit zu höheren Herstellkosten, Gewicht und Kraftstoffverbrauch führen. Mit entsprechenden Dynamiksimulationen können für jede Fahrzeugkonfiguration und jede Straßensituation die zu erwartenden Belastungen berechnet und die Fahrzeugreaktion simuliert werden. Dazu ist eine Modellierung aller Komponenten des Fahrzeuges notwendig (Bild 4).

Komponenten des Simulationsmodells

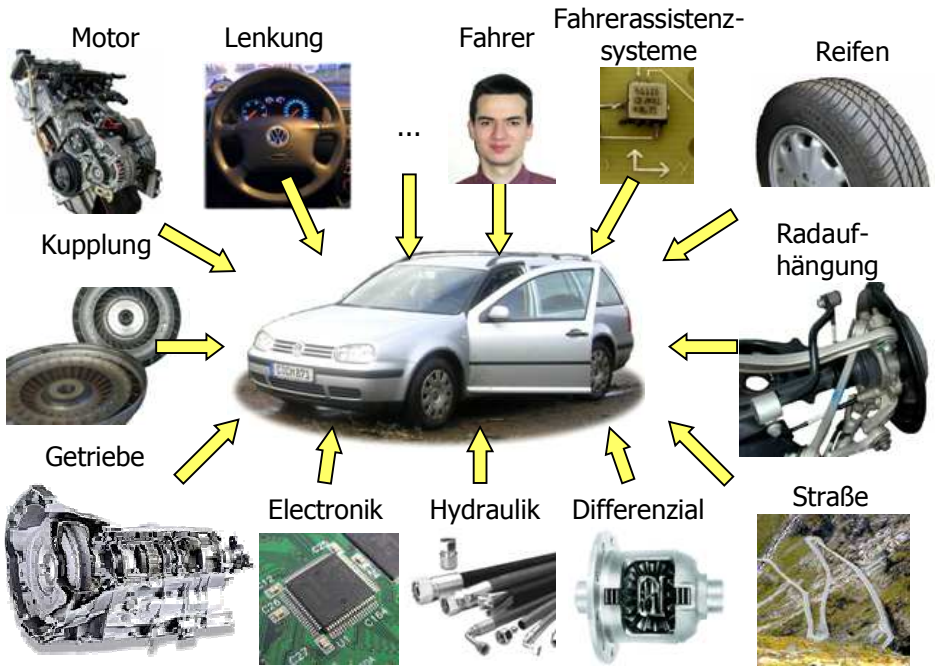


Bild 4: Komponenten des Simulationsmodells zur Fahrzeugdynamik

Fig. 4: Components of a simulation model for the vehicle dynamics

DLL-Implementierung (DLL implementation)

Im Folgenden wird eine einfache Fahrdynamiksimulation[4], die nur einige der in Bild 4 dargestellten Komponenten beinhaltet, schrittweise von der Mathcad®-Programmierung in eine DLL überführt. Diese inkrementelle Konvertierung ermöglicht zum einen die Reduzierung der Fehler beim Übertragen von Mathcad® nach C durch Kontrolle der Zwischen-

[4] Ebenes Zweispurmodell mit drei Freiheitsgraden, Magic-Formula-Reifenmodell nach dem Äquivalenzschlupfprinzip, Vorgabe von Antriebsmoment, -verteilung und Lenkwinkel, Runge-Kutta-Integrationsverfahren vierter Ordnung mit 2 ms Schrittweite

ergebnisse und zum anderen die Überprüfung der Effektivitätssteigerung durch die Ermitt–lung der Rechenzeit. Bild 5 zeigt die schrittweise Reduzierung der Rechenzeit mit steigender DLL-Integration. Als Resultat ergibt sich bei diesem Beispiel fast eine Geschwindigkeits–steigerung um den Faktor 4 im Vergleich zur Ausgangssituation.

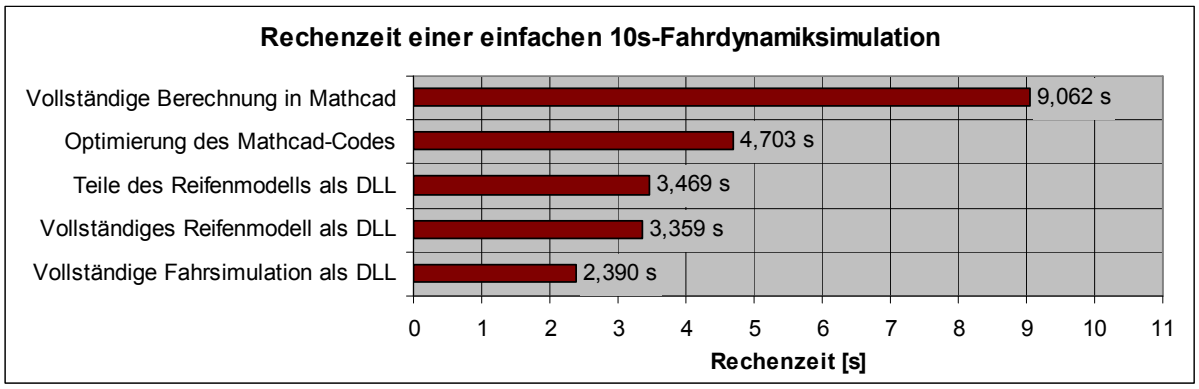


Bild 5: Rechenzeit einer einfachen 10s-Fahrdynamiksimulation

Fig. 5: Calculation time of a 10 s vehicle dynamics simulations

Beispiele für Zusatzfunktionen

Examples for additional functions

Bei der Installation von Mathcad® werden bereits sehr viele Funktionen mitgeliefert. Des Weiteren können Erweiterungen in der Form von „Extension Packs" über PTC bezogen werden. Dazu gehören

- Extension Pack „Data Analysis"
- Extension Pack „Image Processing"
- Extension Pack „Signal Processing"
- Extension Pack „Wavelets"

Darüber hinaus können mit DLL's weitere benutzerdefinierte Funktionen generiert werden, von denen einige im Folgenden vorgestellt werden. Ein Teil davon kann zur Evaluierung über die Internetseite www.simulationtools.de herunter geladen werden.

Zeit- und Datumsfunktionen (Time and date functions)

Bei umfangreichen Berechnungen und vielen Ausgabedateien kann es sinnvoll sein, auch das aktuelle Datum und die Zeit der Berechnung mit abzulegen. Dafür lässt sich z. B. eine DLL-Funktion entwickeln, welche diese Angaben liefert.

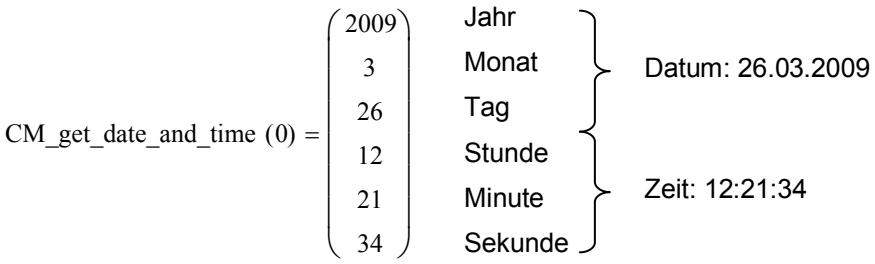


Bild 6: Aufruf der Datumsfunktion in Mathcad

Shared Memory (Shared memory)

Bei vielen verschachtelten Funktionen ist es v. a. zur Fehlersuche erforderlich, Zwischenergebnisse unmittelbar vor einem Fehler anzuzeigen. Die Mathcad®-interne Debuggingfunktion führt jedoch zu einer drastischen Verringerung der Rechengeschwindigkeit, sodass Fehler wie z.B. eine Division durch Null in aufwändigen Simulationen nur sehr schwer gefunden werden kann. Dafür ist eine DLL-Funktion erforderlich, welche während des Rechenlaufs Zwischenergebnisse in einem DLL-internen Speicher ablegt und z. B. nach dem Auftreten des Fehlers ausgelesen werden kann.

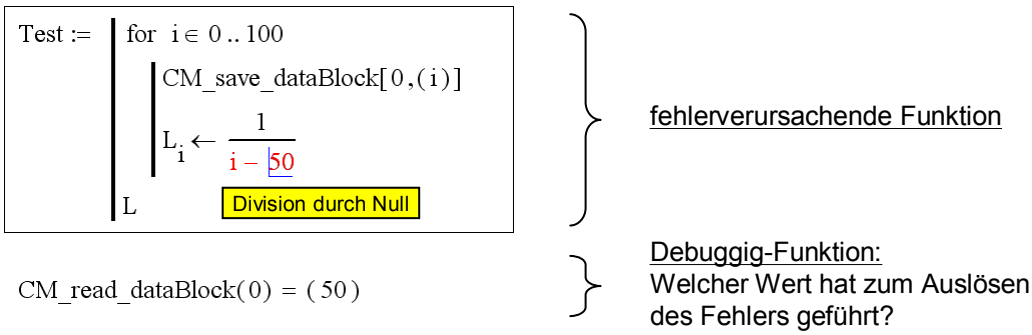


Bild 7: Beispielhafte Nutzung der Shared-Memory-Funktion in Mathcad®

Fig. 7: Exemplary using of the shared memory function in Mathcad®

Dateifunktionen (File operations)

Mathcad® bietet zahlreiche Möglichkeiten zur Datenein- und -ausgabe. Darunter z. B. auch Excel, Matlab oder eine Textdatei. Messwerte von einem Prüfstand z. B. liegen häufig als Textdatei im ASCII-Code vor. Damit die Daten auch später noch zugeordnet werden können, werden oft die ersten Zeilen als Beschreibung und Spaltentitel verwendet. Solche Dateien können mit der *Dateneingabe*-Funktion von Mathcad® eingelesen werden.

Zum automatisierten Analysieren vieler Messwertdateien muss die Übergabe des Dateinamen in einer Variablen erfolgen. Das ist mit der *Dateneingabe*-Funktion nicht möglich. Eine

Alternativlösung ist der Befehl *PRNLESEN*, der aber keine Spaltentitel in der Messwertdatei akzeptiert. Diese vorher für jede Messwertdatei händisch zu löschen wäre nicht nur zu aufwendig, sondern dadurch geht auch eine wichtige Information, die Semantik des Spalteninhaltes, verloren.

Der Befehl READFILE liest zwar die Spaltentitel mit ein, erlaubt aber keine automatische Konvertierung der Dezimaltrennzeichen von Komma in Punkt und ist zudem wesentlich langsamer als PRNLESEN.

Die Lösung des Problems kann mit einer DLL-Funktion erfolgen, welche die ersten text-enthaltenden Zeilen ignoriert. Zudem ist diese DLL-Funktion mehr als doppelt so schnell. Bild 8 zeigt dazu die Rechengeschwindigkeit für eine Datei mit 1,9 Mio. Messwerten[5].

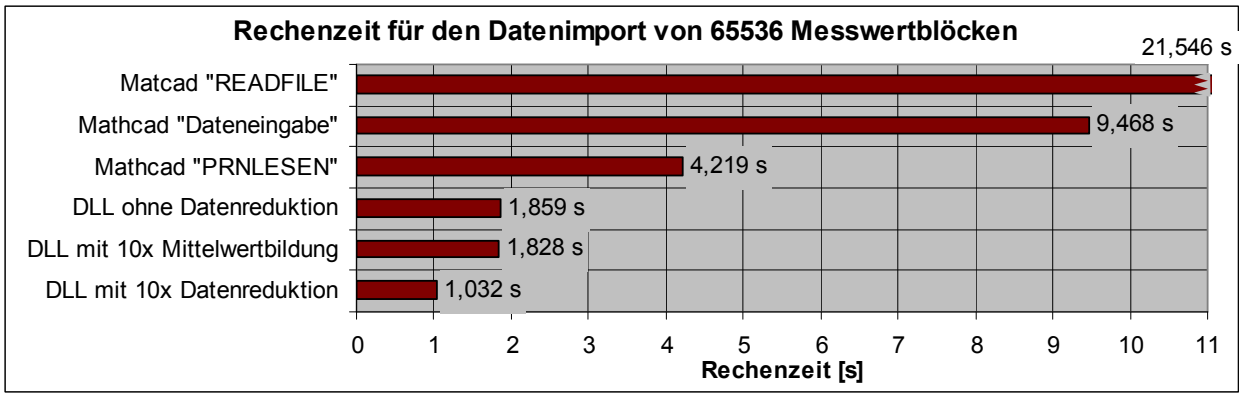


Bild 8: Rechenzeit für den Datenimport

Fig. 8: Calculation time for the file input

Praktischerweise lässt sich in dieser DLL-Funktion auch noch eine erste Datenselektierung vornehmen. So kann zur Datenreduktion z. B. immer der Mittelwert aus mehreren aufeinanderfolgenden Werten bestimmt werden oder eine gewisse Anzahl von Zeilen übersprungen werden.

Diese DLL ist auch im Downloadbereich unter www.simulationtools.de verfügbar. Nach dem Kopieren der DLL „SimulationTools_Mathcad_Dateifunktionen.dll" in das Instalationsverzeichnis von Mathcad® „...\userfi\" steht u. a. folgende Funktion zur Verfügung:

Funktionsname: **`ST_load_datafile(Dateiname, Manipulation)`**

Dateiname: Der Vektor „Dateiname" enthält den vollständigen Pfad der Datei als ASCII-Code. Dieser kann über die Mathcad®-Funktion `zfinvek` aus der Zeichenkette ermittelt werden.

Manipulation: Der Vektor „Manipulation" enthält in seinen Zeilen Parameter zur Steuerung der Dateikonvertierung:

[5] 65536 Messwertblöcke mit je 29 Messwerten; Testumgebung: CPU: AMD 3800+ singlecore, RAM: 1,5 GB, Betriebssystem: Windows XP SP2, Mathcad-Version: 14.0 M020

Zeile 0: Kommas in Punkte umwandeln (1...ja, 0...nein)

Zeile 1: Datenreduktion (x Werte zu einem Wert zusammenfassen) (1... jeden Wert einlesen)

Zeile 2: Art der Datenreduktion (0...Zeilen überspringen, 1...Mittelwertbildung)

Beispiel: `ST_load_datafile(zfinvek(„C:\Messwerte.txt"),(1,10,1)^T)`

Aus der Messwertdatei werden nun immer 10 Zeilen zu einer durch Mittelwertbildung zusammengefasst.

Netzwerkfunktionen (Network functions)

Die Lösung einiger technischer Problemstellungen lässt sich hervorragend durch Parallelisierung beschleunigen. Werden z. B. sehr viele Getriebestrukturen auf deren Eignung für eine bestimmte Anwendung untersucht, so kann dies von mehreren Rechnern gleichzeitig erfolgen. Ein zentraler Rechner übernimmt dabei das Datenmanagement (Bild 9).

Steigerung der Rechengeschwindigkeit durch Parallelisierung

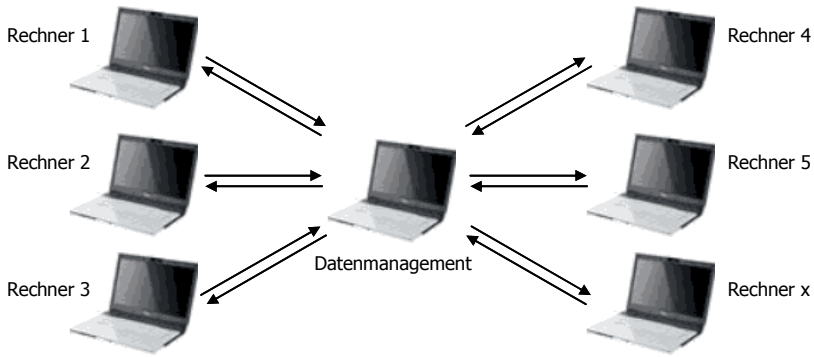


Bild 9: Parallelisiertes Rechnen in einem (heterogenen) Cluster

Fig. 9: Parallel computing in a (heterogen) cluster

Die DLL zur Kommunikation zwischen den Rechnern muss dabei über folgende grundlegende Funktionen verfügen:

- Auslesen der eigenen Netzwerkkonfiguration (z.B. IP)
- Aufbau der Verbindung zu einem anderen Rechner (über sog. Sockets)
- Reparatur defekter Verbindungen (v. a. bei WLAN-Verbindung)
- Verschlüsselung und Entschlüsselung der Daten
- Kontrolle der übertragenen Daten durch Antwort vom Zielrechner mit Prüfsumme
- Dynamisches Einbinden bzw. Entfernen von Clients (z.B. Steigerung der Rechenleistung des Clusters während der Mittagspause)

Da bei dieser Art von Clusterbildung die Kommunikation zwischen den einzelnen Rechnern einer starken Geschwindigkeitsbeschränkung unterliegt, eignet sich dieser Cluster nur für weniger kommunikationsintensive Aufgaben. Die Rechenleistung kann nicht mit der von massiv parallelen Systemen verglichen werden.

Lizenzverwaltung (License administration)

Wenn ein Mathcad®-Dokument an Dritte weitergegeben wird, kann eine zeitliche oder funktionsmäßige Einschränkung – z. B. für eine Testphase – notwendig sein. Diese Beschränkung muss natürlich in einem Bereich erfolgen, der vom Nutzer nicht verändert, und damit aufgehoben werden kann. In Mathcad® eignen sich dafür Regionen besonders gut, da diese ausgeblendet und durch ein Passwort geschützt werden können. Leider findet man im Internet Hinweise, diesen Passwortschutz zu umgehen.
Dieses kann dadurch verhindert werden, dass Schlüsselfunktionen in eine DLL ausgelagert werden und darin z. B. auch die Abfrage der Nutzungsdauer erfolgen. Über eine Netzwerkfunktion kann auch eine Lizenzabfrage auf einem eigenen Server erfolgen und so z. B. eine Verlängerung der Lizenz sehr kurzfristig erfolgen.

Online-Darstellung von 3D-Grafiken (Online presentation of 3D graphics)

Ein Grundprinzip von Mathcad® ist die sequenzielle Abarbeitung der Befehle. So kann erst nach einer Berechnung die Darstellung der Ergebnisse z. B. in einem Diagramm erfolgen. Für die meisten Anwendungen ist dieses auch völlig ausreichend. In einigen Fällen ist jedoch eine Visualisierung der Ergebnisse bereits während der Berechnung sinnvoll. So könnte man z. B. bei der Entwicklung einer aufwändigen Fahrsimulation schon während der Berechnung eventuelle Fehler erkennen und den Rechenlauf unterbrechen. Bisher kann eine Bewertung der Berechnungsergebnisse erst nach dem Abschluss der vollständigen Simulation erfolgen.
Zur Realisierung dieser Funktionalität kann eine Mathcad®-DLL programmiert werden, welche die 3D-Darstellung übernimmt. Um die zu entwickelnde Software möglichst allgemein gültig zu gestalten, ist eine Funktionstrennung sinnvoll. Dazu wird eine zweite Software z. B. als Windows-Applikation in C++ entwickelt, welche die 3D-Darstellung übernimmt. Zwischen der Mathcad®-DLL und der neuen Software sorgt eine sog. *Pipe* für den Datenaustausch. Bild 10 zeigt dazu die prinzipielle Kommunikationsstruktur.
Der Vorteil liegt u. a. darin, dass diese neue Software ihre Befehle zur Darstellung über eine standardisierte Schnittstelle erhält und somit an jede beliebige Simulationssoftware gekoppelt werden kann. Die Mathcad®-DLL nimmt lediglich die Zahlenwerte z. B. der

aktuellen Fahrzeugposition entgegen und sendet diese über die Pipe zur 3D-Darstellung an die neue Software.

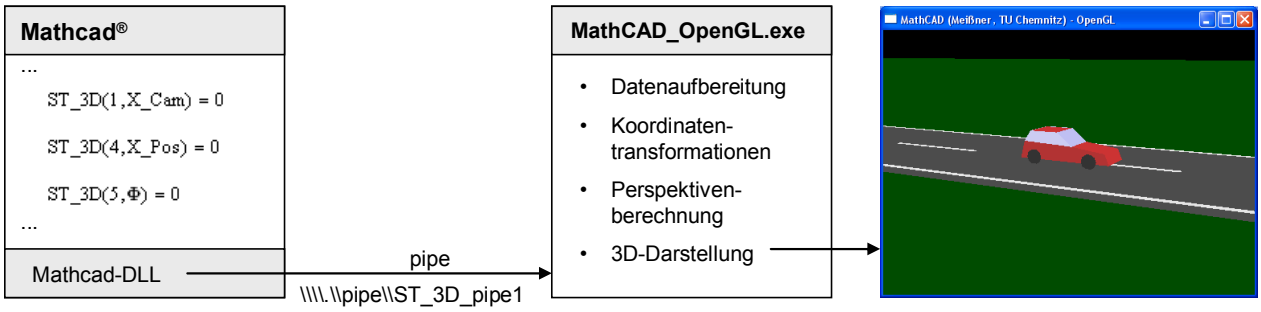


Bild 10: Kommunikationsstruktur zur Online-Darstellung von 3D-Grafiken

Fig. 10: communication structure for visualisation von 3D graphics

Zur Reduzierung des Datenaustausches über die Pipe kann z. B. die Fahrzeuggeometrie als Preprozess einmal in den Speicher der Software geladen werden. Zur Darstellung während der Simulation genügt das Übertragen der Fahrzeugposition, der Drehbewegung um die körperfesten Achsen und die Position der Kamera. Die neue Software wird daraufhin die Daten aufbereiten, alle Koordinatentransformationen durchführen, die aktuelle Blickrichtung der Kamera ermitteln und die 3D-Darstellung vornehmen.

Ein positiver Nebeneffekt dieser Modellierungsart liegt in der teilweise parallelen Abarbeitung der Befehle auf der CPU. In der aktuellen Version nutzt Mathcad® maximal zwei Prozessorkerne. Bei einem Rechner mit mehr als zwei Prozessorkernen wird bei dieser Modellierungsart die Rechengeschwindigkeit von Mathcad® durch die 3D-Darstellung nicht vermindert. Die neue Software kann dann über einen anderen *Windows-Thread* einen der noch freien Prozessorkerne nutzen.

Hardwareschnittstellen (Hardware interface)

Mathcad® ist so konzipiert, dass klar definierte Eingangsgrößen in der Form von Werten oder Wertetabellen im Dokument selbst oder in anderen Dateien gespeichert sind, welche bei der Ausführung geladen werden. Einige Anwender würden jedoch auch gern eine angeschlossene Hardware als Datenein- oder Datenausgabe nutzen. Dafür ist es analog zu dem obigen Abschnitt sinnvoll, den Datenaustausch über eine *Pipe* als Interprozesskommunikation (IPC) durchzuführen. Bei einer allgemein gültigen Programmierung der DLL können dieselben Grundfunktionen sowohl für die Ausgabe in einem 3D-Fenster als auch auf eine angeschlossene Hardware verwendet werden. Bild 11 zeigt dafür eine mögliche Kommunikationsstruktur.

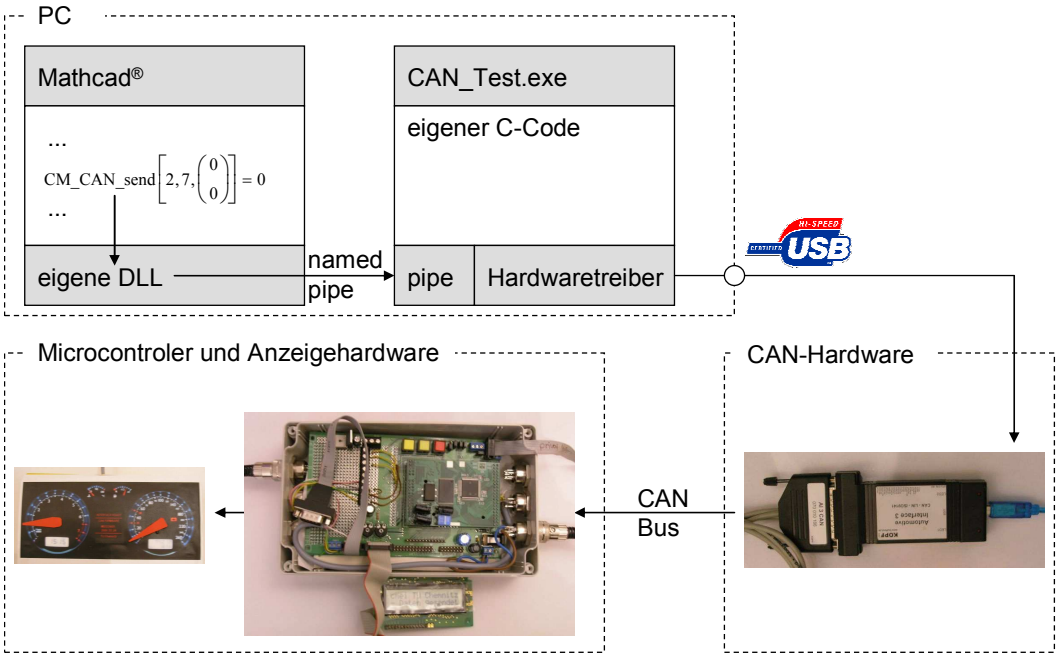


Bild 11: Kommunikationsstruktur für eine Hardwareansteuerung aus Mathcad®

Fig. 11: communication structure for hardware output in Mathcad®

Ausblick

Prospectus

Die in diesem Dokument vorgestellten Methoden und beispielhaften Ergebnisse zeigen den Vorteil von DLLs für Mathcad®-Anwendungen. Dabei wurden besonders die Effizienzsteigerung, der Quellcodeschutz und die Codeportierung hervorgehoben. Anhand eines Code-Beispiels und der notwendigen Downloadquelle für den Compiler kann der Leser sich in die Erstellung eigener DLLs einarbeiten bzw. die Konvertierungsdienstleistung von Mathcad in eine DLL oder in einen C-Quelltext von dem Unternehmen *SimulationTools* in Anspruch nehmen.

Dieser Artikel hilft dem Leser sowohl über die in Mathcad® vorhandenen Funktionen als auch die hier im Überblick dargestellten Routinen hinaus eigene Funktionalitäten zu entwickeln.

Literatur

Literature

[1] Schuschk, Michael; Soltendick, Wolfgang; Borland C++: Einführung, Ressourcen, Programmierung. München. 2. Aufl. 1992.

[2] Mathsoft. Developer's Reference. Mathcad® 14.

Anhang 1

Vollständiger Beispielcode zur Berechnung der Elementsumme einer Matrix

```
// Funktionsblock 1
#include "mcadincl.h"

// Funktionsblock 2
CUFPROC CreateUserFunction;
CUEMTPROC CreateUserErrorMessageTable;
MAAPROC MathcadArrayAllocate;
MAFPROC MathcadArrayFree;
MAPROC MathcadAllocate;
MFPROC MathcadFree;
ISIPROC isUserInterrupted;

// Funktionsblock 3
#define  NUMBER_OF_ERRORS       3
char * myErrorMessageTable[NUMBER_OF_ERRORS] = {"Rechenlauf unterbrochen", "zu
   wenig Speicher", "Wert muss reell sein"};

// Funktionsblock 4
LRESULT Elementsumme(COMPLEXSCALAR *const out, const COMPLEXARRAY *in)
{
   long i,j;                     // Zählvariablen für Zeilen und Spalten
   double sum=0;           // Elementsumme

   for(i=0 ; i < in->rows ; i++)        // alle Zeilen durchlaufen
      for(j=0 ; j < in->cols ; j++)            // alle Spalten durchlaufen
         sum = sum + in->hReal[j][i];          // Summierung durchführen

   out->real=sum;                // „sum“ ist Ausgabewert der Funktion
   return 0;                     // Funktion ohne Fehler beenden
}

// Funktionsblock 5
FUNCTIONINFO DLL_Elementsumme_ = {"DLL_Elementsumme","A,R","Testfunktion für
   SAXIM",(LPCFUNCTION)Elementsumme,COMPLEX_SCALAR,1,{COMPLEX_ARRAY}};

// Funktionsblock 6
int APIENTRY LibMain(HINSTANCE hDLL, DWORD dwReason, LPVOID lpReserved)
{
switch (dwReason)
{
   case DLL_PROCESS_ATTACH:
   {
```

```
      HINSTANCE  hinstLib=LoadLibrary("mcaduser.dll");

      if(hinstLib!=NULL)
      {
         CreateUserFunction =
            (CUFPROC) GetProcAddress(hinstLib,"CreateUserFunction");
         CreateUserErrorMessageTable =
            (CUEMTPROC)GetProcAddress(hinstLib,"CreateUserErrorMessageTable");
         MathcadArrayAllocate =
            (MAAPROC)GetProcAddress(hinstLib,"MathcadArrayAllocate");
         MathcadArrayFree=(MAFPROC)GetProcAddress(hinstLib,"MathcadArrayFree");
         MathcadAllocate=(MAPROC)GetProcAddress(hinstLib,"MathcadAllocate");
         MathcadFree=(MFPROC)GetProcAddress(hinstLib,"MathcadFree");
         isUserInterrupted=(ISIPROC)GetProcAddress(hinstLib,"isUserInterrupted");

         if (CreateUserErrorMessageTable == NULL ||
            CreateUserFunction == NULL ||
            MathcadArrayAllocate == NULL ||
            MathcadArrayFree == NULL ||
            MathcadAllocate == NULL ||
            MathcadFree  == NULL ||
            isUserInterrupted == NULL) break;

         if(CreateUserErrorMessageTable(hDLL,NUMBER_OF_ERRORS,myErrorMessageTable))
         {
            CreateUserFunction( hDLL, &DLL_Elementsumme_ );
         }
      }
   }
   break;
   case DLL_THREAD_ATTACH:
   case DLL_THREAD_DETACH:
   case DLL_PROCESS_DETACH:
   break;
   }
return TRUE;
}

#undef NUMBER_OF_ERRORS
```