



Einsatz von Maxima im Unterricht

Wilhelm Haager

HTL St. Pölten

Abteilung Elektrotechnik

Drei Meilensteine bei Berechnungen

Drei Meilensteine bei Berechnungen

Taschenrechner (1975)

Drei Meilensteine bei Berechnungen

Taschenrechner (1975)

Personalcomputer (1990)

Drei Meilensteine bei Berechnungen

Taschenrechner (1975)

Personalcomputer (1990)

Computeralgebra (2005)

Drei Meilensteine bei Berechnungen

Taschenrechner (1975)

- ▶ Irrtum: Die Schüler müssen nicht mehr händisch rechnen und kopfrechnen (können).

Personalcomputer (1990)

Computeralgebra (2005)

Drei Meilensteine bei Berechnungen

Taschenrechner (1975)

- ▶ Irrtum: Die Schüler müssen nicht mehr händisch rechnen und kopfrechnen (können).

Personalcomputer (1990)

- ▶ Irrtum: Die Schüler brauchen keine Diagramme mehr selber zeichnen (können).

Computeralgebra (2005)

ET
Drei Meilensteine bei Berechnungen**Taschenrechner (1975)**

- ▶ Irrtum: Die Schüler müssen nicht mehr händisch rechnen und kopfrechnen (können).

Personalcomputer (1990)

- ▶ Irrtum: Die Schüler brauchen keine Diagramme mehr selber zeichnen (können).

Computeralgebra (2005)

- ▶ Irrtum: Die Schüler brauchen keine Lösungswege mehr beschreiben (können).

Besondere Eigenschaften

Besondere Eigenschaften

- ▶ Maxima ist ein echtes Computeralgebra-System

Besondere Eigenschaften

- ▶ Maxima ist ein echtes **Computeralgebra-System**
- ▶ Maxima ist **plattform-unabhängig**

Besondere Eigenschaften

- ▶ Maxima ist ein echtes **Computeralgebra-System**
- ▶ Maxima ist **plattform-unabhängig**
- ▶ Maxima ist **frei verfügbar** (Nachhaltigkeit!)

Besondere Eigenschaften

- ▶ Maxima ist ein echtes **Computeralgebra-System**
- ▶ Maxima ist **plattform-unabhängig**
- ▶ Maxima ist **frei verfügbar** (Nachhaltigkeit!)
- ▶ Maxima ist **einfach** zu bedienen

Besondere Eigenschaften

- ▶ Maxima ist ein echtes **Computeralgebra-System**
- ▶ Maxima ist **plattform-unabhängig**
- ▶ Maxima ist **frei verfügbar** (Nachhaltigkeit!)
- ▶ Maxima ist **einfach** zu bedienen
- ▶ Datenfiles sind **ASCII-Files** (Kompatibilität!)

Besondere Eigenschaften

- ▶ Maxima ist ein echtes **Computeralgebra-System**
- ▶ Maxima ist **plattform-unabhängig**
- ▶ Maxima ist **frei verfügbar** (Nachhaltigkeit!)
- ▶ Maxima ist **einfach** zu bedienen
- ▶ Datenfiles sind **ASCII-Files** (Kompatibilität!)
- ▶ Maxima ist eine **umfangreiche Programmiersprache**, daher in der Funktionalität beliebig erweiterbar

Besondere Eigenschaften

- ▶ Maxima ist ein echtes Computeralgebra-System
- ▶ Maxima ist plattform-unabhängig
- ▶ Maxima ist frei verfügbar (Nachhaltigkeit!)
- ▶ Maxima ist einfach zu bedienen
- ▶ Datenfiles sind ASCII-Files (Kompatibilität!)
- ▶ Maxima ist eine umfangreiche Programmiersprache, daher in der Funktionalität beliebig erweiterbar
- ▶ Maxima spricht \TeX

Besondere Eigenschaften

- ▶ Maxima ist ein echtes Computeralgebra-System
- ▶ Maxima ist plattform-unabhängig
- ▶ Maxima ist frei verfügbar (Nachhaltigkeit!)
- ▶ Maxima ist einfach zu bedienen
- ▶ Datenfiles sind ASCII-Files (Kompatibilität!)
- ▶ Maxima ist eine umfangreiche Programmiersprache, daher in der Funktionalität beliebig erweiterbar
- ▶ Maxima spricht \TeX

Daher gibt es eigentlich *keinen* Grund, **Maxima** nicht zu verwenden.

Einsatzmöglichkeiten

Einsatzmöglichkeiten

Maxima als Programmiersprache

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

- ▶ Anschauliche Darstellung komplexer Zusammenhänge

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

- ▶ Anschauliche Darstellung komplexer Zusammenhänge
- ▶ Animationen

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

- ▶ Anschauliche Darstellung komplexer Zusammenhänge
- ▶ Animationen

Maxima als Werkzeug

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

- ▶ Anschauliche Darstellung komplexer Zusammenhänge
- ▶ Animationen

Maxima als Werkzeug

- ▶ Anwendung im Labor, bei Konstruktionsübungen und begleitend in allen Theoriegegenständen.

Einsatzmöglichkeiten

Maxima als Programmiersprache

- ▶ Programmier-Paradigmen: prozedural, funktional, regelbasiert.
- ▶ Kein Unterschied zwischen Anweisungen und Daten („Homoikonizität“), Sprache beliebig erweiterbar.

Maxima zum Veranschaulichen

- ▶ Anschauliche Darstellung komplexer Zusammenhänge
- ▶ Animationen

Maxima als Werkzeug

- ▶ Anwendung im Labor, bei Konstruktionsübungen und begleitend in allen Theoriegegenständen.
- ▶ Schwerpunkt: **Formulieren** von Lösungswegen (statt *Beschreiten*).

Maxima als Programmiersprache

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen

$y = f(x)$

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen

$y = f(x)$

$z = g(y)$

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen

$y = f(x)$

$z = g(y)$

z ausgeben

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen
 $y = f(x)$
 $z = g(y)$
z ausgeben

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen
 $y = f(x)$
 $z = g(y)$
z ausgeben

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

einlesen

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

x einlesen
 $y = f(x)$
 $z = g(y)$
z ausgeben

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

f(einlesen)

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

```
x einlesen  
y = f(x)  
z = g(y)  
z ausgeben
```

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

```
g(f(einlesen))
```

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

```
x einlesen  
y = f(x)  
z = g(y)  
z ausgeben
```

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

```
ausgeben ( g ( f ( einlesen ) ) )
```

ET
Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

```
x einlesen  
y = f(x)  
z = g(y)  
z ausgeben
```

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

```
ausgeben(g(f(einlesen)))
```

- **Erweiterung der Syntax** am Beispiel neuer Operatoren:

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

```
x einlesen  
y = f(x)  
z = g(y)  
z ausgeben
```

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

```
ausgeben(g(f(einlesen)))
```

- **Erweiterung der Syntax** am Beispiel neuer Operatoren:
Parallelschalt-Operator: `R1 // R2`

ET Maxima als Programmiersprache

- **Prozedurales Programmieren:** Ein Programm besteht aus einer Abfolge von Anweisungen.

```
x einlesen  
y = f(x)  
z = g(y)  
z ausgeben
```

- **Funktionales Programmieren:** Ein Programm besteht aus ineinandergeschachtelten Funktionsaufrufen.

```
ausgeben(g(f(einlesen)))
```

- **Erweiterung der Syntax** am Beispiel neuer Operatoren:

Parallelschalt-Operator: $R1 \ // \ R2$

Versor-Operator („cis“): $r \ /_{\phi}$

Maxima zum Veranschaulichen

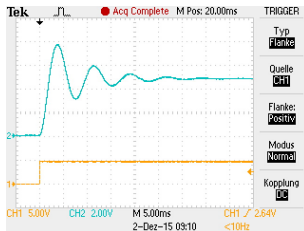
- ▶ Entstehung der Zeitverläufe sinusförmiger Größen aus rotierenden Zeigern:

Maxima als Werkzeug

Maxima als Werkzeug

Messwertverarbeitung

- Sprungantwort eines Schwingkreises

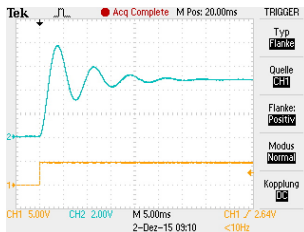


Oszilloskop

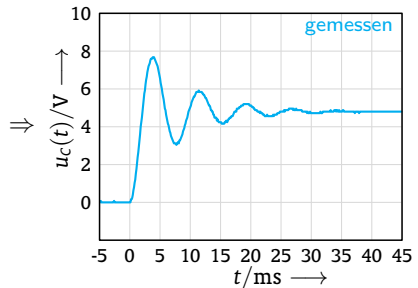
Maxima als Werkzeug

Messwertverarbeitung

- Sprungantwort eines Schwingkreises



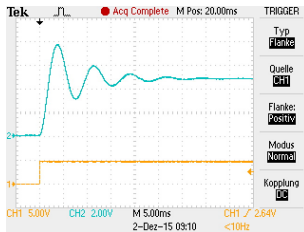
Oszilloskop



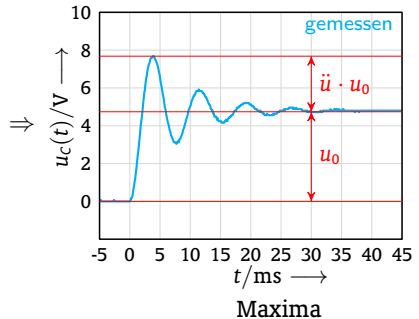
Maxima als Werkzeug

Messwertverarbeitung

- Sprungantwort eines Schwingkreises



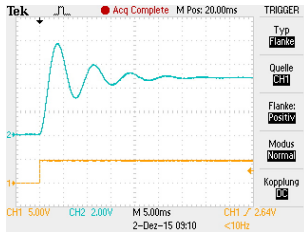
Oszilloskop



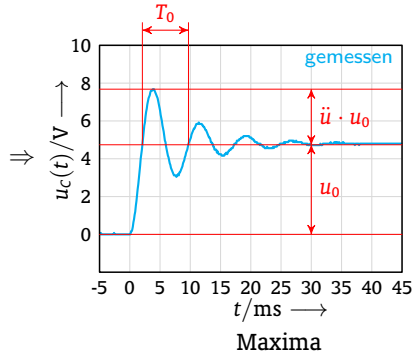
Maxima als Werkzeug

Messwertverarbeitung

- Sprungantwort eines Schwingkreises



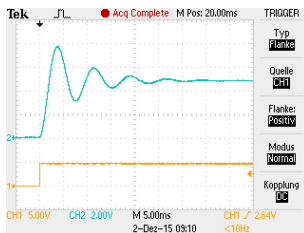
Oszilloskop



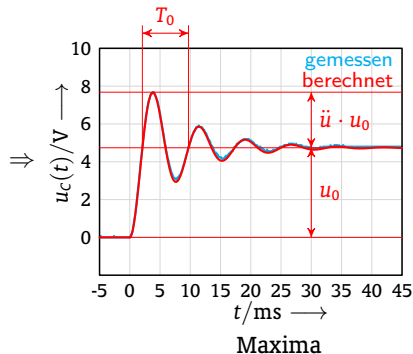
Maxima als Werkzeug

Messwertverarbeitung

- Sprungantwort eines Schwingkreises



Oszilloskop



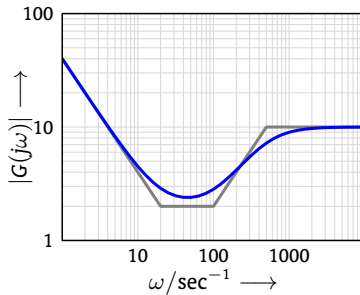
Maxima als Werkzeug

Schaltungsdimensionierung

Maxima als Werkzeug

Schaltungsdimensionierung

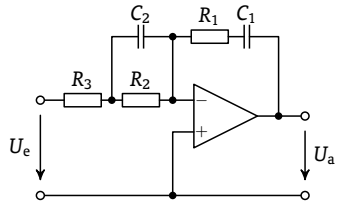
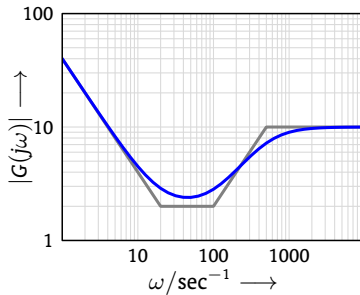
- Entwurf eines PIDT1-Reglers



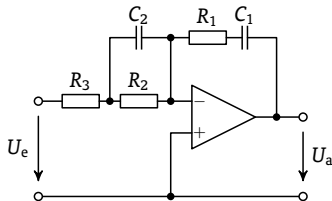
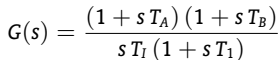
Maxima als Werkzeug

Schaltungsdimensionierung

► Entwurf eines PIDT1-Reglers



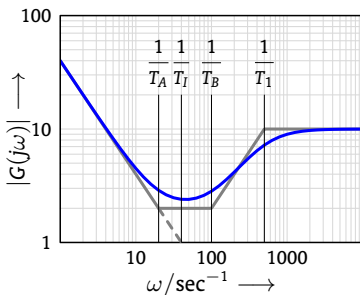
► Entwurf eines PIDT1-Reglers



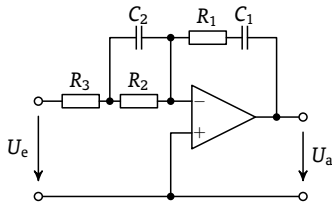
Maxima als Werkzeug

Schaltungsdimensionierung

► Entwurf eines PIDT1-Reglers



$$G(s) = \frac{(1 + s T_A)(1 + s T_B)}{s T_I (1 + s T_1)}$$



$$G(s) = - \frac{R_1 + \frac{1}{s C_1}}{R_3 + R_2 \parallel \frac{1}{s C_2}}$$



Jetzt geht's los ...