Plots mit LaTeX und pgfplots Kurzanleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

V1.0 26.11.2012 Markus Grün

Da langwierige und komplizierte Anleitungen nur selten gelesen werden, ist diese hier kurz und einfach gehalten.

Einführung

Mit pgfplots kann man direkt in LaTeX sehr schöne Plots erzeugen. Die Bedienung ist ausgesprochen einfach und das Handbuch sehr ausführlich mit Bild-Beispielen zu fast jedem Befehl. Es stehen mittlerweile eine Reihe von MATLAB-funktionen zur Verfügung, so dass ein Plot genauso einfach erzeugt werden kann, wie ein normaler MATLAB-plot.

Alle hier vorgestellten Code-Beispiele sind voll lauffähig.

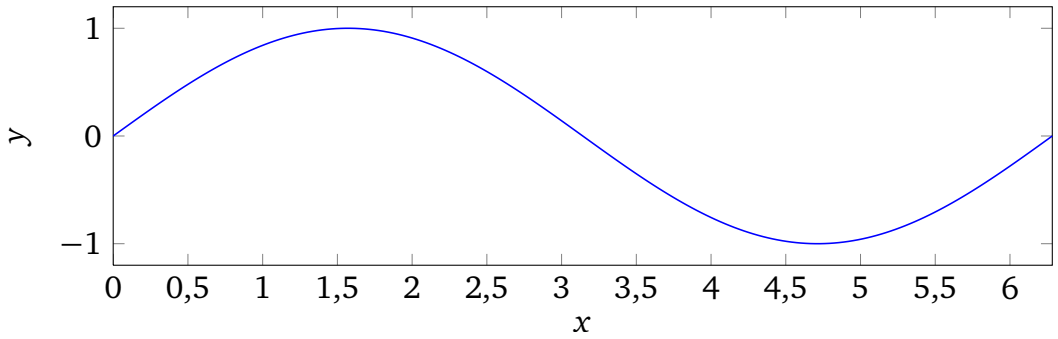
MATLAB

```
x = linspace(0,2*pi);
y = sin(x);


pgfplot(x,y,'myfile','d:\myfolder');
```

LaTeX

```
\documentclass{article}
\input{D:/TikZ/pgfplotssetup.tex}
\begin{document}
    \input{d:/myfolder/myfile.tikz}
\end{document}
```



Muss ich wirklich die komplette Anleitung lesen um `pgfplots` einsetzen zu können?

Du willst `pgfplots` *nur mal kurz ausprobieren?*

Auf den folgenden drei Seiten steht alles, was Du zum Erstellen einfacher Bilder wissen musst!

Du willst `pgfplots` *in Deiner Arbeit einsetzen?*

Dann schau Dir trotzdem die folgenden drei Seiten an um einen kurzen Einblick in die Funktion von `pgfplots` zu erhalten und lies dann noch die ausführliche Anleitung

Vorbereitung

Damit alle hier vorgestellten Beispiele funktionieren, muss der Ordner "`TikZ`" der sich in der Zip-Datei befindet, auf die Festplatte entpackt werden, zum Beispiel nach `D:\TikZ\`.

In MATLAB muss der Ordner `D:\TikZ\Matlab\` in den Suchpfad aufgenommen werden. `->File->Set Path->Add Folder...`

Los gehts:

Wir wollen jetzt also unseren ersten Plot mit `pgfplots` erstellen, und zwar wollen wir die Sprungantwort eines PT_1-Glieds plotten. Dazu tippen wir in MATLAB, ähnlich wie in dem Beispiel auf dem Deckblatt, folgendes ins `Command-Window` und ersetzen die gewohnte MATLAB-funktion „`step(G)`“ einfach durch „`pgfstep(G,...)`“:

```
G = tf([1],[0.1  1]);

pgfstep(G,'myPlot','D:\myfolder');
```

Falls das Verzeichnis „`myfolder`“ noch nicht existiert, kommt eine Fehlermeldung. Also Verzeichnis erstellen und Code nochmals ausführen. Ein Blick in das Verzeichnis zeigt, dass die Funktion `pgfstep(·)` die Datei „`myPlot.tikz`“, sowie den Ordner „`TikZdata`“ und darin einen Ordner „`myPlot`“ erstellt hat, in dem die Dateien „`G.txt`“ und „`G_end.txt`“ liegen. Diese Dateien enthält die x- und y-Werte der Sprungantwort und den stationären Endwert.

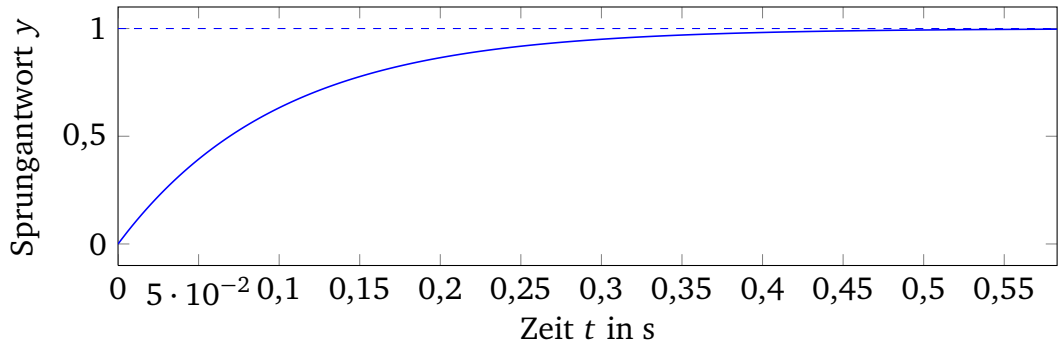
```
d:\myfolder\myPlot.tikz                  (enthält TikZ-code)
d:\myfolder\TikZdata\myPlot\G.txt        (enthält die x- und y-Werte)
d:\myfolder\TikZdata\myPlot\G_end.txt    (enthält den stat. Endwert)
```

Die `.tikz`-Datei kann direkt in LaTeX eingebunden werden. Wir erstellen also ein neues LaTeX-Projekt und binden die Datei mit dem `\input{.}`-Befehl ein:

```
\documentclass{article}
\input{D:/TikZ/pgfplotssetup.tex}

\begin{document}
    \begin{center}
        \input{d:/myfolder/myPlot.tikz}
    \end{center}
\end{document}
```

und kompilieren das Ganze:



Beim Betrachten des Ergebnisses wünschen wir uns jedoch, dass die Linienfarbe besser rot sein sollte. Des weiteren wäre ein Gitternetz. Wir öffnen also `myPlot.tikz` und schauen uns den Code an:

```
\renewcommand{\mywidth}{0.6\textwidth}
\renewcommand{\myheight}{50mm}

\begin{tikzpicture}
  \begin{axis}[xlabel=Zeit $t$ in s,
               ylabel=Sprungantwort $y$,
               enlarge x limits=false,
%               ymin=0, ymax=20,
%               minor tick num = 1,       % alternativ: minor x|y|z tick num
               grid=none,                % minor|major|both|none
               legend pos = north east,  % [outer] south west|south east|north west|north east
               legend cell align=left,   % Legendeneinträge linksbündig
               width=\mywidth, height=\myheight,
               ]
    \addplot[smooth, blue, semithick] table{d:/myfolder/TikZdata/myPlot/G.txt};
      \addplot[dashed, blue, forget plot] table{d:/myfolder/TikZdata/myPlot/G_end.txt};
  \end{axis}
\end{tikzpicture}
```

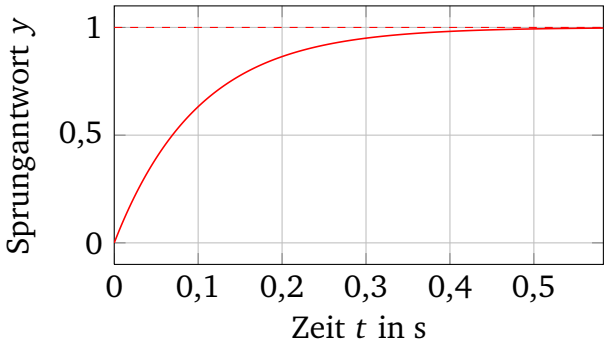
Die gewünschten Änderungen können ganz einfach vorgenommen werden. in Zeile 10 werden die Gitternetzlinien eingeschaltet, indem man statt „`none`“ entweder „`major`“ oder „`both`“ wählt. Die Linienfarbe finden wir in Zeile 15 und 16.

Mit den beiden Variablen `\mywidth` und `\myheight` in den ersten beiden Zeilen kann die Größe des Bildes angepasst werden. Die Syntax zur Anpassung des Aussehenes ist sehr einfach. Das Handbuch ist wirklich sehr gut, ausführlich und bietet zu fast jedem Befehl ein Bild mit Code-Beispiel.

Jetzt kompilieren wir unser LaTeX Projekt erneut und können die Änderungen direkt sehen:

```
\renewcommand{\mywidth}{0.6\textwidth}
\renewcommand{\myheight}{50mm}

\begin{tikzpicture}
  \begin{axis}[xlabel = Zeit $t$ in s,
    ylabel = Sprungantwort y,
    enlarge x limits=false,
%     ymin=0, ymax=20,
%     minor tick num = 1,
    grid=major, % minor|major|both|none
    legend pos = north east,
        legend cell align=left,   % Legendeneinträge →
            ←linksbündig
    width=\mywidth, height=\myheight,
    ]
    \addplot[smooth, red, semithick] table{D:/myfolder/...
      \addplot[dashed, red, forget plot] table{D:/myfo...
  \end{axis}
\end{tikzpicture}
```

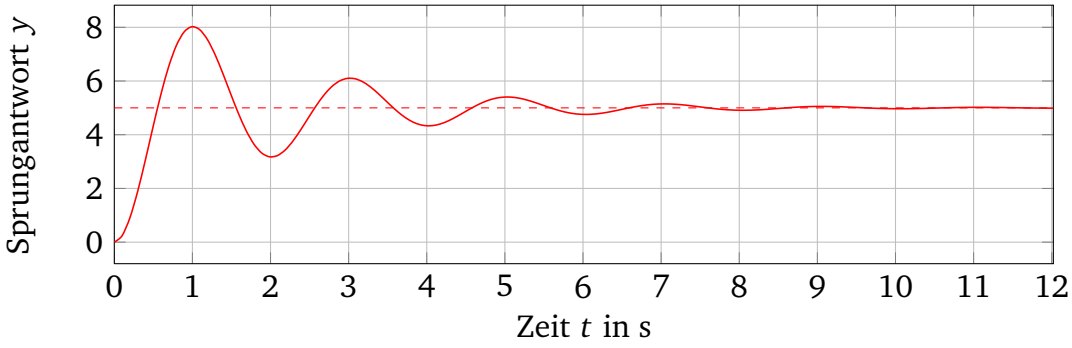


Wir stellen jetzt jedoch fest, dass wir eigentlich doch die Sprungantwort eines Systems zweiter Ordnung mit einer anderen Verstärkung plotten wollten. Kein Problem, wir gehen einfach zurück nach Matlab, ändern die Übertragungsfunktion `G` und führen den Befehl ein zweites Mal aus

```
G = tf([5],[0.1  0.1  1]);

pgfstep(G,'myPlot','D:\myfolder');
```

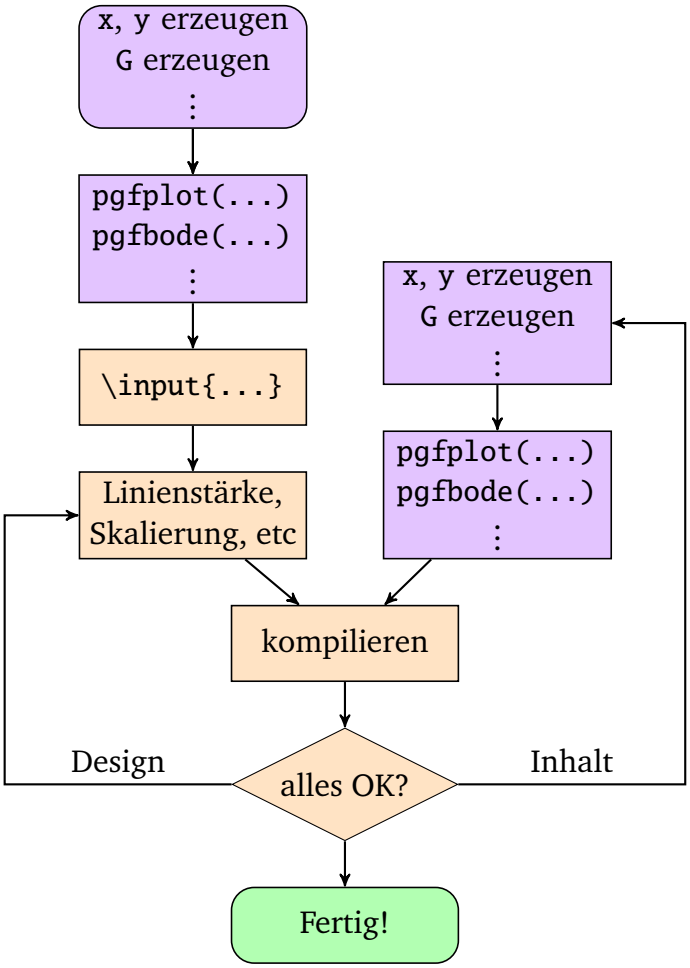
Ohne weitere Aktionen in LaTeX kompilieren wir das Dokument erneut:



Wie zu sehen ist, sind die ganzen Einstellungen, die wir vorher in der `.tikz`-Datei gemacht haben, erhalten geblieben. Sobald diese Datei einmal erzeugt wurde, wird sie von der Matlab-Funktion nicht mehr überschrieben. Nur die `.txt`-Dateien, die die Daten enthält werden überschrieben. Die Skalierung der Achsen erfolgt automatisch, sofern sie nicht fest vorgegeben wird (Zeile 8).

Zusammenfassung

Genauso wie bisher in MATLAB erzeugt man nun auf diese Weise einen Plot in LaTeX. Anstatt jedoch in der MATLAB-`figure` Einstellungen vorzunehmen, wie Linienart und -stärke, Achsenskalierung, etc., tut man dies nun in der erstellten `.tikz`-Datei. Die Syntax ist etwas anders, aber trotzdem sehr leicht zu erlernen und das Handbuch ist ausgesprochen gut.



MATLAB-funktion	Funktionsweise
`pgfplot(x,y,...)`	wie `plot(x,y)`
`pgfplotyy(x1,y1,x2,y2,...)`	wie `plotyy(x1,y1,x2,y2)`
`pgfbode(G,...)`	wie `bode(G)`
`pgfbodeLOG(G,...)`	wie `bode(G)`, Ausgabe doppelt logarithmisch
`pgfasymp(G,...)`	wie `bode(G)`, mit Asymptoten
`pgfasympLOG(G,...)`	wie `bode(G)`, mit Asymptoten, doppelt logarithmisch
`pgfloglog(x,y,...)`	wie `loglog(x,y)`
`pgfsemilogx(x,y,...)`	wie `semilogx(x,y)`
`pgfsemilogy(x,y,...)`	wie `semilogy(x,y,)`
`pgfstep(G,...)`	wie `step(G)`
`pgfplot3(x,y,z,...)`	wie `plot3(x,y,z)`