Numerisches Rechnen mit LuaLaTeX II

Anwendungen mit Python und pyluatex

Jürgen Vorloeper

Hochschule Ruhr West

DANTE Sommertagung
24. Juni 2022

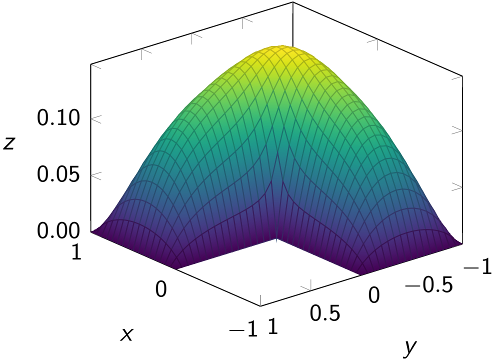
Motivation

- Erstellung mathematischer Schriftstücke, insb. Lehrmaterialien
- Grafiken und Datensätze/Tabellen als Ergebnis numerischer Berechnungsergebnisse
- Programmcode (C, Python, R, Matlab,...) in der Regel vorhanden
- Modulare Verwendung von Textbausteinen und Möglichkeit leichter Anpassungen
- Verwendung von LuaLaTeX

Motivation

Gleichung auf L-Gebiet:

$$-\Delta u = 1 \text{ in } \Omega \subset \mathbb{R}^2$$
$$u = 0 \text{ auf } \partial\Omega$$



- Numerische Berechnung der Lösung einer mathematischen Gleichung mit C, Python, Matlab/Octave,...
- Darstellung der Lösung direkt in LaTeX mit TikZ/pgfplots

LuaTeX

LuaTeX...

- ist eine Weiterentwicklung von pdfTeX
- bietet Unterstützung für Unicode
- bietet Zugriff auf Schriften im Format OpenType
- integriert METAPOST
- enthält Skriptsprache Lua

$\rightsquigarrow$ Numerisches Rechnen mit LuaLaTeX [Mon+14; Vos20; Vor21]

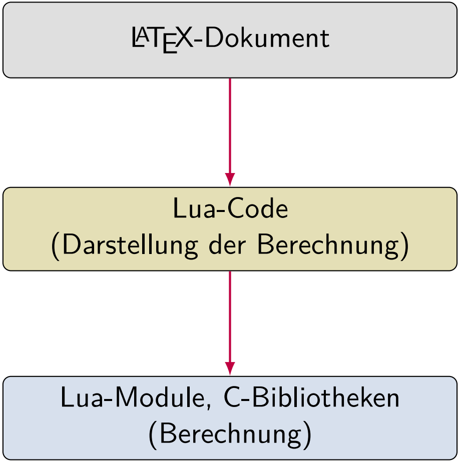
Lua

Lua...

- ist eine (weitgehend) plattformunabhängige Skriptsprache
- besitzt eine C-Schnittstelle
- ist einfach zu erlernen und zu nutzen
- ermöglicht Erstellung eigenständiger Programme als auch eingebetteter Programme
- besitzt math-Modul mit Basisfunktionalitäten

$\rightsquigarrow$ Numerisches Rechnen mit Lua grundsätzlich möglich
$\rightsquigarrow$ Verwendung von C-Bibliotheken möglich $\rightsquigarrow$ Vielfältige Möglichkeiten – im Prinzip

3-Schichten Struktur



Beispiel: Standardnormalverteilung

- Verteilungsfunktion $\Phi(x)$ der Standardnormalverteilung als Tabelle
- Numerische Berechnung mittels Reihenentwicklung [Mar04]

x_0	0	1	2	3	4	5	6	7	8	9
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891
1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87286	0.87493	0.87698	0.87900	0.88100	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91309	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.92220	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189

Beispiel:

$$\Phi(1.26) \approx 0.89617$$

Im LaTeX-Dokument:

```
\directlua{require "lua/luacode.lua"}

\begin{tabular}{ccccccccccc}\toprule
  $x_0$ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \midrule
  \directlua{Table_NormalCDF()}
\end{tabular}
```

Datei luacode.lua:

```
require 'lua/normcdf'

Table_NormalCDF = function()

    for j=0,14 do
        tex.print(string.format("%.1f & %.5f (...) & %.5f \\\\", 0.1*j, normcdf(0.1*j+0.00), normcdf(0.1*j
              +0.01), (...), normcdf(0.1*j+0.09)))

        if j == 14 then
            tex.print(" \\bottomrule")
        elseif (j+1) % 5 == 0 then
            tex.print(" \\midrule")
        end
    end

end
```

Implementierung in Lua

Datei `normcdf.lua`

```lua
function normcdf(x)
  if x <= -8 then
    return 0
  elseif x >= 8 then
    return 1
  else
    local s, b, q = x, x, x^2
    for i=3,1/0,2 do
      b = b*q/i
      local t = s
      s = t + b
      if s == t then break end
   end
  return 0.5 + s*math.exp(-0.5*q - 0.91893853320467274178)
  end
end
```

- Basiert auf C-Code aus [Mar04]
- Fachwissen für qualitativ *hochwertige numerische Ergebnisse* erforderlich

Eigene praktische Erfahrungen

- Zahlreiche Lehrmaterialien erstellt (Skripte, Vortragsfolien, Arbeitshefte)
- Abschlussarbeit zur teilautomatischen Erstellung von Aufgaben und Lösungen [Fli22]
- Verwendung von C-Bibliotheken erfordert Low-Level-Programmierung in C und Lua
- Aufwändige Rechnungen mit zahlreichen verschiedenen C-Bibliotheken praktisch nicht durchführbar
- Zunehmende Beliebtheit von Python im Wissenschaftlichen Rechnen, z. B. [MRS16; pyM22]
- Exzellente Python-Pakete für numerische Berechnungen verfügbar, z. B. NumPy [Har+20], Scipy [Vir+20]

LaTeX und Python

- Mehrere Pakete zur Verwendung von Python in LaTeX:
  - seit 2013 `pythontex` [Poo21; Poo13; Poo15]
  - seit 2021 `pyluatex` [End22]
- Vielfältige Anwendungen mit `pythontex`, z. B. [Stu22]
- Python-Paket `sympy` [Meu+17] für symbolisches Rechnen
- Mehrere wichtige Python-Pakete unterstützen LaTeX-Ausgabe, z. B. `sympy` [Meu+17], `matplotlib` [Hun07]

Verwendung von pyluatex

```
\usepackage[executable=/home/juergen/anaconda3/envs/dev/bin/python]{pyluatex}

\begin{python}
import math
import random
random.seed(0)
from scipy.stats import norm

text = 'Hallo Dante! '
\end{python}

\begin{itemize}
  \item \py{text}
  \item Wurzel 2 ist $\sqrt{2} = \py{math.sqrt(2)}$
  \item Eine Zufallszahl: \py{random.randint(2,5)}
  \item Standardnormalverteilung $\Phi(1.26)\approx\py{round(norm.cdf(1.26),5)}$
\end{itemize}
```

- Hallo Dante!
- Wurzel 2 ist $\sqrt{2} = 1.4142135623730951$
- Eine Zufallszahl: 5
- Standardnormalverteilung $\Phi(1.26) \approx 0.89617$

Verwendung von `pyluatex`

- LuaLaTeX erforderlich
- Einfache Verwendung (wenige Paketoptionen, nur *ein* LuaLaTeX-Durchlauf erforderlich)
- Verwendung von `-shell-escape` erforderlich
- In `beamer`: Frame mit Option `fragile` erforderlich
- Befehl
  py leitet Python-Ausgabe an LaTeX Weiterentwicklung
- In Python können beliebige Pakete verwendet werden
- Verwendung verschiedener Python-Sessions möglich

```
\begin{python}
    from scipy.stats import norm

    def norm_tabular (N):
       res = "\\begin{tabular}{cccccccccc}\\toprule\n"
       res += "$x_0$ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\\\ \\midrule \n"

       for j in range (0,N):
          res += "{:.1f}".format(0.1*j) + " & "
          for k in range (0,9):
             res += "{:.5f}".format(norm.cdf(0.1*j+k*0.01)) + " & "

          if ((j+1)%5>0):
             res += "{:.5f}".format(norm.cdf(0.1*j+9*0.01)) + " \\\\ \n"
          else:
             if (j==N-1):
                res += "{:.5f}".format(norm.cdf(0.1*j+9*0.01)) + " \\\\ \\bottomrule \n"
             else:
                res += "{:.5f}".format(norm.cdf(0.1*j+9*0.01)) + " \\\\ \\midrule \n"

       res += "\\end{tabular}"

       return res
\end{python}


\begin{center}
\py{norm_tabular(40)}
\end{center}
```

Und ein letztes Mal Standardnormalverteilung

- Einfache Verwendung von Funktionen aus C-Bibliotheken
- Beispiel `double normcdf (double x)` in `normcdf.so`

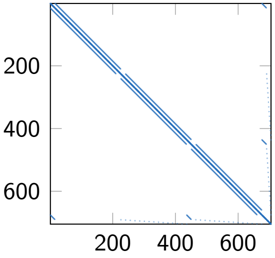
```
\begin{python}
    import ctypes
    import os

    fun = ctypes.cdll.LoadLibrary(os.getcwd()+'/normcdf.so')
    fun.normcdf.argtypes = [ctypes.c_double]
    fun.normcdf.restype = ctypes.c_double
\end{python}

      single value $\Phi(1.26) \approx \py{"{:.5f}".format(fun.normcdf(1.26))}$
```

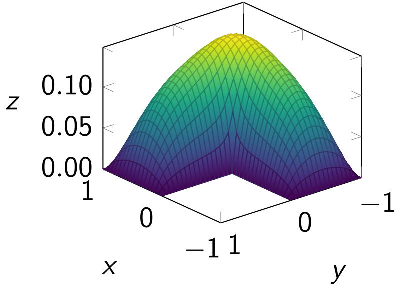
Lösung einer PDE

- Gesucht: Lösung u der *Poisson-Gleichung* auf $\Omega \subset \mathbb{R}^2$

$$-\Delta u = 1 \quad \text{in } \Omega, \qquad u = 0 \quad \text{auf } \partial\Omega$$

- FD-Diskretisierungen führen auf dünnbesetzte Matrizen
- Darstellung der Besetzungsstruktur einer Matrix und der Lösung





Lösung einer PDE

```
\pyfileq{pde.py}

\begin{python}
n = 16

A, b = setup_Ab(n)
z = spsolve(A,b)

s1 = plot_A(A)
s2 = plot_L_tex(z,n)
\end{python}
```

- Datei `pde.py` enthält Python-Code zur Lösung der PDE
- Strings `s1` und `s2` enthalten LaTeX-Code zur Ausgabe in Ti*k*Z/pgfplots
- Ausgabe mit `\py{s1}` bzw. `\py{s1}`

Symbolisches Rechnen mit SymPy

```
\begin{python}
    import sympy as sym
    from mpmath import mp

    n = 24
    s1 = sym.factorial(n)

    mp.dps = 50   # set number of digits
    s2 = mp.pi    # print pi
\end{python}

Fakultät $\py{n}! = \py{s1}$

Kreiszahl: $\pi = \py{s2}$
```

Fakultät $24! = 620448401733239439360000$

Kreiszahl: $\pi = 3.1415926535897932384626433832795028841971693993751$

Symbolisches Rechnen und Numerische Integration

- Approximation von Integralen durch Quadraturformeln
- Newton-Cotes-Formeln

$$\int_0^1 f(x)\,\mathrm{d}x \approx \sum_{j=0}^{m} c_j \cdot f(x_j)$$

mit Stützstellen $x_j = \frac{j}{m}$ und Gewichten c_j mit

$$c_j = \int_0^1 \prod_{\substack{k=0\\k\neq j}} \frac{x - x_k}{x_j - x_k}\,\mathrm{d}x, \quad j = 0, \ldots, m$$

- Exakte (symbolische) Berechnung von x_j und c_j und Darstellung in Tabellenform

Newton-Cotes-Formeln

m	Bezeichnung	x_j	c_j
0	Mittelpunktsregel	$\frac{1}{2}$	1
1	Trapezregel	$0,\ 1$	$\frac{1}{2},\ \frac{1}{2}$
2	Simpson-Regel	$0,\ \frac{1}{2},\ 1$	$\frac{1}{6},\ \frac{2}{3},\ \frac{1}{6}$
3	$\frac{3}{8}$-Regel	$0,\ \frac{1}{3},\ \frac{2}{3},\ 1$	$\frac{1}{8},\ \frac{3}{8},\ \frac{3}{8},\ \frac{1}{8}$
4	Milne-Regel	$0,\ \frac{1}{4},\ \frac{1}{2},\ \frac{3}{4},\ 1$	$\frac{7}{90},\ \frac{16}{45},\ \frac{2}{15},\ \frac{16}{45},\ \frac{7}{90}$
5	6-Punkt-Regel	$0,\ \frac{1}{5},\ \frac{2}{5},\ \frac{3}{5},\ \frac{4}{5},\ 1$	$\frac{19}{288},\ \frac{25}{96},\ \frac{25}{144},\ \frac{25}{144},\ \frac{25}{96},\ \frac{19}{288}$
6	Weddle-Regel	$0,\ \frac{1}{6},\ \frac{1}{3},\ \frac{1}{2},\ \frac{2}{3},\ \frac{5}{6},\ 1$	$\frac{41}{840},\ \frac{9}{35},\ \frac{9}{280},\ \frac{34}{105},\ \frac{9}{280},\ \frac{9}{35},\ \frac{41}{840}$

Symbolische und Numerische Berechnung

```
\begin{python}
    x = sym.symbols('x')

    def nc(m,j):
      s = '1 '
      for k in range (m+1):
        if (k!=j):
          s += f' * (x-{k}/{m})/({j}/{m}-{k}/{m})'
      return sym.latex(sym.integrate(sym.S(s),(x,0,1)))

    def ncx(m,j):
      if (m==0 and j==0):
        return '\\frac{1}{2}' #sym.latex(sym.S('1/2'))

      if (m==0 and j==0):
        return sym.latex(sym.S('1/2'))
      if (m==j):
         return 1
      if (j==0):
         return 0
      s = f'{j}/{m}'
      return sym.latex(sym.S(s))
\end{python}
```

- Erstellung der Tabelle unter Verwendung von `\py{ncx(4,1)}` und `\py{nc(4,1)}`

- (Wettbewerbs-) Aufgabe: Gegeben ist die Funktion

$$u(x) = \frac{\sqrt[3]{x^2}\, e^{-\frac{1}{(0.99-x)^2}}}{e^{-\frac{1}{(x-0.01)^2}} + e^{-\frac{1}{(0.99-x)^2}}} \quad \text{für } 0.01 < x < 0.99$$

  An welcher Stelle $x^* \in \mathbb{R}$ nimmt u'' sein Maximum an? Berechnen Sie x^* mit mindestens 5 Stellen Genauigkeit.

- Die Funktion u'' lautet

$$u''(x) = \frac{2\left(-\frac{3-\frac{2}{(x-0.99)^2}}{(x-0.99)^4} + \frac{\frac{4\left(\frac{e^{-\frac{1}{(x-0.01)^2}}}{(x-0.01)^3}+\frac{e^{-\frac{1}{(x-0.99)^2}}}{(x-0.99)^3}\right)^2}{e^{-\frac{1}{(x-0.01)^2}}+e^{-\frac{1}{(x-0.99)^2}}} + \frac{3e^{-\frac{1}{(x-0.01)^2}}}{(x-0.01)^4} - \frac{2e^{-\frac{1}{(x-0.01)^2}}}{(x-0.01)^6} + \frac{3e^{-\frac{1}{(x-0.99)^2}}}{(x-0.99)^4} - \frac{2e^{-\frac{1}{(x-0.99)^2}}}{(x-0.99)^6}}{e^{-\frac{1}{(x-0.01)^2}}+e^{-\frac{1}{(x-0.99)^2}}} - \frac{4\left(\frac{e^{-\frac{1}{(x-0.01)^2}}}{(x-0.01)^3}+\frac{e^{-\frac{1}{(x-0.99)^2}}}{(x-0.99)^3}\right)}{(x-0.99)^3\left(e^{-\frac{1}{(x-0.01)^2}}+e^{-\frac{1}{(x-0.99)^2}}\right)} - \frac{4\left(\frac{e^{-\frac{1}{(x-0.01)^2}}}{(x-0.01)^3}+\frac{e^{-\frac{1}{(x-0.99)^2}}}{(x-0.99)^3}\right)}{3x\left(e^{-\frac{1}{(x-0.01)^2}}+e^{-\frac{1}{(x-0.99)^2}}\right)} + \frac{4}{3x(x-0.99)^3} - \frac{1}{9x^2}\right)\sqrt[3]{x^2}\, e^{-\frac{1}{(x-0.99)^2}}}{e^{-\frac{1}{(x-0.01)^2}}+e^{-\frac{1}{(x-0.99)^2}}}$$

Symbolische und Numerische Berechnung

- Händische Berechnung praktisch unmöglich
- Numerische Ungenauigkeiten bei Differenzenquotienten
- $\rightsquigarrow$ symbolische Berechnung von u'' mit `sympy` und Maximumsuche mit `fmin` (angewandt auf $-u''$) aus `scipy.optimize`
- Lösung: $x^* = 0.54535$

Symbolische und Numerische Berechnung

```
\begin{python}
    import sympy as sym
    from scipy import optimize
    from sympy.functions.elementary.miscellaneous import cbrt

    a = 0.01
    b = 0.99

    x = sym.symbols('x')

    f = sym.exp(-1/(b-x)**2)/(sym.exp(-1/(x-a)**2) + sym.exp(-1/(b-x)**2)) * cbrt(x**2)

    s1 = sym.latex(f)

    f2 = sym.diff(f,x,2)
    s2 = sym.latex(f2)

    fnp = sym.lambdify(x,-f2,'numpy')

    minimum = optimize.fmin(fnp, 0.5,xtol=1e-8,full_output=False,disp=False)
    s3 = "{:.5f}".format(minimum[0])
\end{python}
```

- Ausgabe mittels Strings `s1`, `s2`, `s3`

Symbolische und Numerische Berechnung

```
\begin{itemize}
  \item Händische Berechnung praktisch unmöglich
  \item Numerische Ungenauigkeiten bei Differenzenquotienten
  \item $\leadsto$ symbolische Berechnung von $u''$ mit \texttt{sympy} und Maximumsuche mit \texttt{fmin} (
       angewandt auf $-u''$) aus \texttt{scipy.optimize}
  \item Lösung: $x^\ast=\py{s3}$
\end{itemize}
```

Abschließende Bemerkungen

- Symbolisches Rechnen zur (teil-) automatischen Erstellung von mathematischen Aufgaben (und Lösungen) mit SymPy möglich
- Preprint Server arXiv verwendet *kein* LuaLaTeX
- Umfangreiche und zeitaufwändige Rechnungen in eigene Dokumente auslagern

Literatur I

[End22] Tobias Enderle. *pyluatex*. 2022. URL: https://www.ctan.org/pkg/pythontex (besucht am 13. 06. 2022).

[Fli22] Thomas Flinkow. „A Framework for Individualised Mathematical Assignments with Solutions in LaTeX“. Bachelorarbeit. Hochschule Ruhr West, 2022.

[Har+20] Charles R. Harris u. a. „Array programming with NumPy“. In: *Nature* 585.7825 (Sep. 2020), S. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[Hun07] John D Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in science & engineering* 9.3 (2007), S. 90–95.

[Mar04] George Marsaglia. „Evaluating the Normal Distribution“. In: *Journal of Statistical Software* 11 (4 Juli 2004).

Literatur II

[Meu+17] Aaron Meurer u. a. „SymPy: symbolic computing in Python“. In: *PeerJ Computer Science* 3 (2017), e103.

[Mon+14] Juan I. Montijano, Mario Pérez, Luis Rández und Juan Luis Varona. „Numerical methods with LuaLaTeX“. In: *TUGboat* 35.1 (2014).

[MRS16] René Milk, Stephan Rave und Felix Schindler. „pyMOR – Generic Algorithms and Interfaces for Model Order Reduction“. In: *SIAM Journal on Scientific Computing* 38.5 (2016), S194–S216. DOI: `10.1137/15M1026614`. eprint: `https://doi.org/10.1137/15M1026614`. URL: `https://doi.org/10.1137/15M1026614`.

[Poo13] Geoffrey M Poore. „Reproducible Documents with PythonTeX“. In: *Proceedings of the 12th Python in Science Conference*. Hrsg. von Stéfan van der Walt, Jarrod Millman und Katy Huff. 2013, S. 74–80. DOI: `10.25080/Majora-8b375195-00d`.

Literatur III

[Poo15] Geoffrey M Poore. „PythonTeX: reproducible documents with LaTeX, Python, and more“. In: *Computational Science & Discovery* 8.1 (Juli 2015), S. 014010. DOI: 10.1088/1749-4699/8/1/014010. URL: https://doi.org/10.1088/1749-4699/8/1/014010.

[Poo21] Geoffrey M. Poore. *pythontex*. 2021. URL: https://www.ctan.org/pkg/pythontex (besucht am 13. 06. 2022).

[pyM22] pyMOR developers and contributors. *pyMOR – model order reduction with python*. 2022. URL: https://pymor.org (besucht am 22. 06. 2022).

[Stu22] Bernhard Esslinger und Studerende. *PythonTeX and LaTeX. Familiarize us with PythonTeX in order to build the CrypTool Book*. 2022. URL: https://www.cryptool.org/download/ctb/PythonTex-by-Examples.pdf (besucht am 13. 06. 2022).

Literatur IV

[Vir+20] Pauli Virtanen u. a. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. In: *Nature Methods* 17 (2020), S. 261–272. DOI: 10.1038/s41592-019-0686-2.

[Vor21] Jürgen Vorloeper. *Numerisches Rechnen mit LuaLaTeX*. 2021. URL: https://github.com/dante-ev/Vortraege_Tagungen/blob/master/2021-Fruehjahr/Vorloeper_RechnenLuaTeX.pdf (besucht am 13. 06. 2022).

[Vos20] Herbert Voss. „Chaotische Symmetrien mit Lua berechnen“. In: *DTK* 32.3 (2020).

Kontakt

Prof. Dr. Jürgen Vorloeper

Hochschule Ruhr West
Duisburger Straße 100
45479 Mülheim an der Ruhr

E-Mail: `juergen.vorloeper at hs-ruhrwest.de`