Numerisches Rechnen mit LuaLaTeX

Jürgen Vorloeper[1]

Hochschule Ruhr West

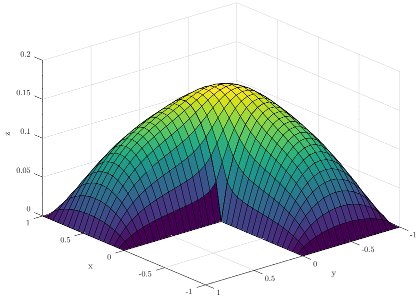
DANTE Frühjahrstagung
11. März 2021

[1] Mitarbeit: Thomas Flinkow (HRW)

Inhalt

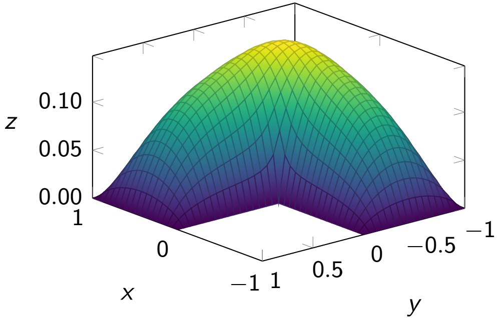
Inhalt

Motivation



- Numerische Berechnung der Lösung einer mathematischen Gleichung mit C, Matlab/Octave,...
- Grafische Darstellung der Lösung mit gnuplot, Matlab/Octave,...
- Einbindung in LaTeX mit `\includegraphics{Bild.pdf}`
- Berechnung und LaTeX-Dokument strikt getrennt

Motivation



- Numerische Berechnung der Lösung einer mathematischen Gleichung mit C, Matlab/Octave,...
- Berechnungsergebnisse in csv-Datei(en) speichern
- Darstellung der Lösung direkt in LaTeX mit TikZ/pgfplots
- Noch immer: Trennung von Berechnung und LaTeX-Dokument

LuaTEX

LuaTEX...

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

$\leadsto$ Numerisches Rechnen mit LuaLATEX [7, 8]

Lua

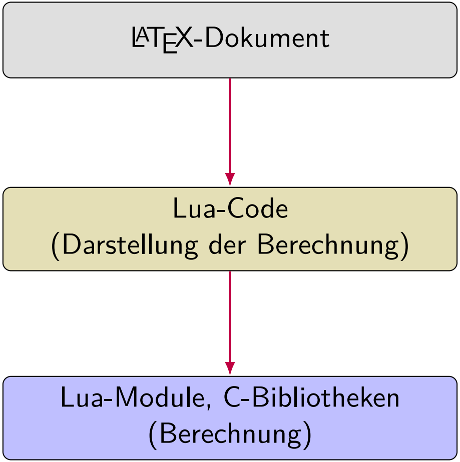
Lua…

- 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 und vielfach sinnvoll

Inhalt

3-Schichten Struktur



Beispiel 1: Standardnormalverteilung

- Verteilungsfunktion $\Phi(x)$ der Standardnormalverteilung als Tabelle
- Numerische Berechnung mittels Reihenentwicklung [6]

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:

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

Implementierung in Lua

Im LaTeX-Dokument:

```
\directlua{dofile("lua/Example_01.lua")}

\begin{tabular}{cccccccccc}
$x_0$ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\directlua{Example_2()}
\end{tabular}
```

Datei `Example_01.lua`:

```
dist = require("lua/share/lua_distributions")
function Example_1 ()
for j=0,14 do
    tex.print(string.format("%.1f & %.5f  \\\\", 0.1*j,
        dist.normcdf(0.1*j+0.00))) --(...)
end
end
```

Datei `lua_distributions.lua`

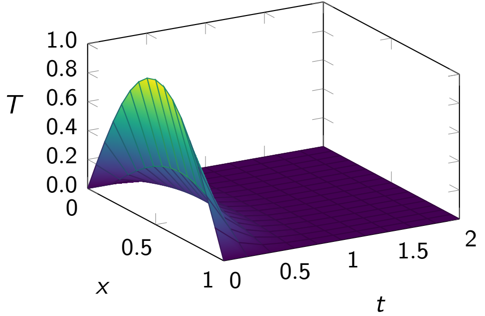
```
local lua_distributions =  {}

function lua_distributions.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

return lua_distributions
```

- Basiert auf C-Code aus [6]
- weitere Funktionen in Lua-Modul

Anfangsrandwertaufgabe (Wärmeleitungsgleichung)

$$\frac{\partial T}{\partial t}(x,t) - \frac{\partial^2 T}{\partial x^2}(x,t) = 0, \quad t > 0,\ x \in (0,1)$$

Anfangswerte $T(x,0) = \sin(\pi x)$, Randwerte $T(0,t) = T(1,t) = 0$



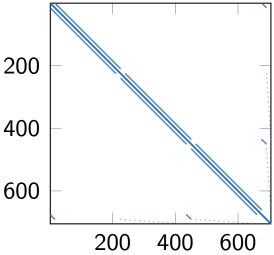
Lösung mit Lua (Mathematisch: Linienmethode, impliziter RK1-Löser)

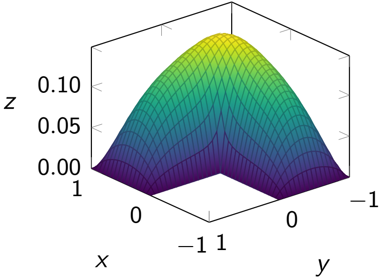
Beispiel 3: Poisson-Gleichung

- 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





- Darstellung in LuaLaTeX mittes Lua-Coroutinen

Coroutinen

```latex
\directlua{dofile("lua/Example_03.lua")}

% Latex text...

% startup coroutine
% and print sparsity pattern
\directlua{co = coroutine.create(Example_3)}

% do other Latex commands


% print solution
\directlua{coroutine.resume(co)}
% complete output in Latex
```

Vortrag.tex

```lua
function Example_3 ()

n=16 -- h=1/n spatial resolution

local A = setup_AL(n)

-- tex.print(...) print sparsity pattern

coroutine.yield()

local b = setup_bL(n)

-- solve A y = b
-- tex.print(...)  output solution

end
```

Example_03.lua

Inhalt

Verwendung von C-Bibliotheken

- Implementierung robuster und effizienter numerischer Verfahren aufwändig
- Zahlreiche C-Bibliotheken zum wissenschaftlichen Rechnen verfügbar
  - LAPACK (numerische lineare Algebra) [1, 4]
  - GLPK (lineare Optimierung) [2]
  - GSL (scientific library, allgemeine Funktionen) [3]
  - ...
- Implementierung mit LuaLATEX
  - Verwendung von `make` und `gcc`
  - Lua C-Interface stackbasiert [5, Abschnitt 4]
  - Suchpfade für shared objects (dynamische Bibliotheken) beachten
  - Texterstellung in LATEX mit `--shell-escape`

Beispiel 4: Gaußsche Glockenkurve

Berechnung der Tabellenwerte mit C-Funktion [6]

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

Lua C-Interface

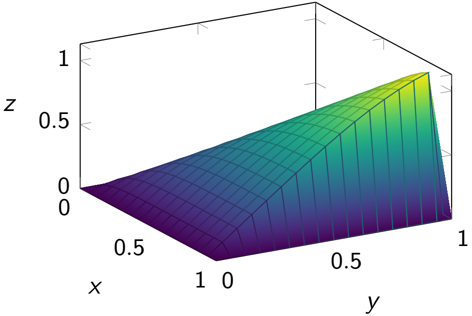
```
#include <lua.h>
#include <lauxlib.h>
#include <lualib.h>
#include <math.h>

double normcdf (double x)
{
long double s=x,t=0,b=x,q=x*x,i=1;

while (s!=t)
     s=(t=s)+(b*=q/(i+=2));

return .5+s*exp(-.5*q-.91893853320467274178L);
}

static int inormcdf (lua_State *L)
{
double x = lua_tonumber(L, -1); /* Get the single number arg from stack*/
double z = normcdf(x);
lua_pushnumber(L,z);      /* Push the return */

return 1;                 /* One return value */
}

int luaopen_luaC_distributions (lua_State *L)
{
     lua_register(L,"normcdf",inormcdf);
     return 0;
}
```

Beispiel 5: (Konvektions-Diffusions-Gleichung)

- Konvektions-Diffusions-Gleichung auf $\Omega = (0,1)^2$

$$\begin{aligned} -\varepsilon\Delta u + (\cos(\beta), \sin(\beta)) \cdot \nabla u &= 1 \quad \text{in } \Omega \\ u &= 0 \quad \text{auf } \partial\Omega \end{aligned}$$

- Lösung u für $\varepsilon = 10^{-4}$, $\beta = \frac{\pi}{4}$ mit upwind-Differenzen:



- Lösung mit Lapack-Funktion `dgbsv` (Löser für Bandmatrizen)

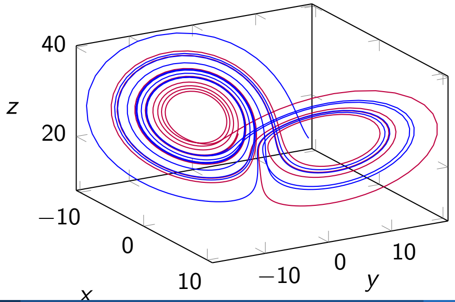
Beispiel 6: (Lorenz-Attraktor)

- Nichtlineares DGL-System

$$\begin{aligned} x'(t) &= \sigma(y(t) - x(t)) \\ y'(t) &= -x(t)z(t) + \rho x(t) - y(t) \\ z'(t) &= x(t)y(t) - \beta z(t) \end{aligned}$$

mit $\sigma = 3$, $\rho = 26.5$, $\beta = 1$ und Anfangswerten $(x_0, y_0, z_0) \approx (0, 1, 0)$

- Lösung mit C-Bibliothek GSL [3], siehe auch [7]



Beispiel 7: (Lineare Optimierung)

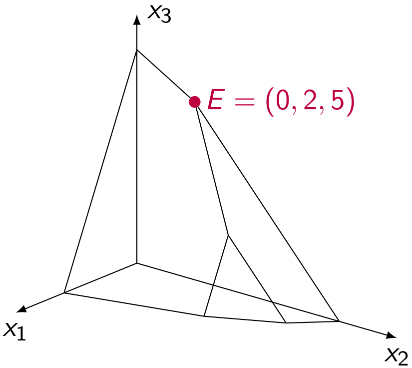
Maximiere

$$z(x) = 2x_1 + 3x_2 + 5x_3$$

unter Nebenbedingungen

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 4 & 1 & 2 \end{pmatrix} x \leq \begin{pmatrix} 7 \\ 8 \\ 12 \end{pmatrix}$$

und $x \geq 0$



- Maximum wird in Ecke $E = (0, 2, 5)$ mit $z^* = 31$ angenommen
- Verwendung von glpk [2]

Implementierung in Lua

```lua
require("luaC_glpk")

function Example_7 ()

c = {2,3,5}
A = {}
A[1] = {1,1,1}
A[2] = {2,1,1}
A[3] = {4,1,2}

b = {7,8,12}

lb = {0,0,0}

ub = {1000,1000,1000}

ctype = "UUU"
vartype = "CC"
s = -1
param = ""

x, z, status = glpk(c, A, b, lb, ub, ctype, vartype, s, param)

tex.print("\\draw [purple,fill] (" .. string.format('%.4f', 0.5*x[1]) .. "," ..  string.format('%.4f', 0.5*
    x[2]) .. "," ..  string.format('%.4f', 0.5*x[3]) .. ") circle (2pt) node [right] {$E=(" ..
    string.format('%1.0f', x[1]) .. "," .. string.format('%1.0f', x[2]) .. "," .. string.format('%1.0f',
    x[3]) .. ")$};")

end
```

Inhalt

Nächste Schritte...

- Robuste Implementierung (Fehlerhandling)
- Einbindung weiterer Bibliotheken/Funktionalitäten
- Implementierung auch für Windows
- Randomisierte Aufgaben (mit Lösungen) für Lehrzwecke

Literatur I

[1] E. Anderson u. a. *LAPACK Users' Guide*. Society for Industrial und Applied Mathematics, 1999.

[2] GLPK Project Contributors. *GLPK - GNU Scientific Library - GNU Project - Free Software Foundation (FSF)*. 2020. URL: https://www.gnu.org/software/glpk/.

[3] GSL Project Contributors. *GSL - GNU Scientific Library - GNU Project - Free Software Foundation (FSF)*. 2019. URL: https://www.gnu.org/software/gsl/.

[4] LAPACK Project Contributors. *LAPACK – Linear Algebra PACKage*. 2019. URL: http://www.netlib.org/lapack/.

[5] PUC-Rio Lua.org. *Lua 5.3 Reference Manual*. 2020. URL: https://www.lua.org/manual/5.3/.

Literatur II

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

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

[8] 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`