Ernst-Abbe-Fachhochschule Jena
Hochschule für angewandte Wissenschaften

Beleg

Numerische Mathematik - Problem2: schlecht konditioniertes LGS

Marc Ludwig
Marvin Hohlfeld
Matthias Bukovsky
Matthias Springstein

Jena, 24. Mai 2013

Betreuer: Prof. P. Wilde
Gutachter: Dipl. Ing. Mustermann

Inhaltsverzeichnis

Abstract

Hier kommt das Geschwafel hin, welches als Einführung in unsere Problemstellung dient.

1. Entwickle die Funktion f aus obigem Problem in eine Taylorreihe

1.1. Die Taylorreihe

In der Analysis verwendet man Taylorreihen (auch Taylor-Entwicklungen, nach dem Mathematiker Brook Taylor[1]), um Funktionen in der Umgebung bestimmter Punkte durch Potenzreihen darzustellen (Reihenentwicklung). So kann ein komplizierter analytischer Ausdruck durch eine nach wenigen Gliedern abgebrochene Taylorreihe (oftmals gut) angenähert werden, z.B. in der Physik oder bei der Ausgleichung geodätischer Netze: So ist die oft verwendete Kleinwinkelnäherung des Sinus eine nach dem ersten Glied abgebrochene Taylorreihe dieser Funktion.

1.2. Definition

Sei $I \subset \mathrm{R}$ ein offenes Intervall, $f\colon I \to \mathrm{R}$ eine unendlich oft differenzierbare Funktion und a ein Element von I. Dann heißt die unendliche Reihe

$$\begin{aligned} P_f(x) &= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots && (1.1)\\ &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n && (1.2) \end{aligned}$$

die Taylor-Reihe von f mit Entwicklungsstelle a.

Hierbei bezeichnet

- $n!$ die Fakultät $n! = 1 \cdot 2 \cdot \cdots \cdot n$ und
- $f^{(n)}(a)$ die n-te Ableitung von f an der Stelle a, wobei man $f^{(0)} := f$ setzt.

[1] Brook Taylor (* 18. August 1685 in Edmonton, Middlesex; † 29. Dezember 1731 in Somerset House, London) war ein britischer Mathematiker. Nach ihm wurde u.a. die Taylorreihe benannt.

1.3. Eigenschaften

Die Taylorreihe $P_f(x)$ zur Funktion $f(x)$ ist eine Potenzreihe P in x. Im Fall einer analytischen Funktion f hat die Taylor-Reihe in jedem Punkt a einen positiven Konvergenzradius r und stimmt in ihrem Konvergenzbereich mit f überein, d.h. es gibt ein $r > 0$, sodass für $|x - a| < r$ gilt

$$P_f(x) = \sum_{n=0}^{\infty} a_n (x-a)^n = f(x) \tag{1.3}$$

. Außerdem stimmen die Ableitungen der Reihe im Entwicklungspunkt a mit den tatsächlichen Ableitungswerten überein:

$$P_f^{(k)}(a) = f^{(k)}(a) \tag{1.4}$$

.

1.4. Konstruktion

Damit die Ableitungen der Reihe im Entwicklungspunkt a mit den tatsächlichen Ableitungswerten übereinstimmen, sind die Koeffizienten a_n der Taylorreihe wie folgt konstruiert:

$$a_n = \frac{f^{(n)}(a)}{n!} \Rightarrow P_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{1.5}$$

.

1.5. Entwicklung von *f(x)* in eine Taylor Reihe

Bei gegebener Funktion

$$f(x) = \frac{1}{1+x} \tag{1.6}$$

und den Gleichungen 1.1 und 1.2 folgt für deren Ableitungen

$$f'(x) = -\frac{1}{(1+x)^2} \tag{1.7}$$

$$f''(x) = \frac{2(1+x)}{(1+x)^4} = \frac{2}{(1+x)^3} \tag{1.8}$$

$$f'''(x) = \frac{2(1+x)^4 - 8(1+x)(1+x)^3}{(1+x)^8} = -\frac{6}{(1+x)^4} \tag{1.9}$$

Hieraus läßt sich folgende Bildungsvorschrifft für eine beliebige Ableitung bestimmen

$$f^n\left(x\right) = \left(-1\right)^n \frac{n!}{\left(1+x\right)^{n+1}} \tag{1.10}$$

hieraus folgt nach Gleichung 1.5

$$f\left(x\right) = P_f\left(x\right) = \sum_{n=0}^{\infty} \left(-1\right)^n \frac{n!}{n!\left(1+a\right)^{n+1}} \left(x-a\right)^n \tag{1.11}$$

$$= \sum_{n=0}^{\infty} \left(-1\right)^n \frac{\left(x-a\right)^n}{\left(1+a\right)^{n+1}} \tag{1.12}$$

2. Nachweis der hermiteschen Form von *H*

Eine hermitesche Matrix oder selbstadjungierte Matrix wird im mathematischen Teilgebiet der Linearen Algebra untersucht. Es handelt sich um eine spezielle Art von quadratischen Matrizen. Benannt sind diese nach dem Mathematiker Charles Hermite[1].

2.1. Definition - hermitesch

Eine $n \times n$-Matrix A mit Einträgen in C heißt hermitesch, wenn sie mit ihrer (hermitesch) Adjungierten A^*, also der transponierten und komplex konjugierten Matrix übereinstimmt. Das heißt, wenn

$$A = A^* = \overline{A}^{\mathrm{T}} = \overline{A^{\mathrm{T}}} \tag{2.1}$$

gilt.

Für die adjungierte Matrix finden sich auch die Bezeichnungen $A^\dagger$ und A^{H}. Die Schreibweise A^* wird auch für die komplex konjugierte Matrix verwendet.

Für die Einträge einer hermiteschen Matrix gilt also:

$$a_{jk} = \overline{a_{kj}} \tag{2.2}$$

Anders formuliert ist eine Matrix A genau dann hermitesch, wenn ihre Transponierte gleich ihrer komplex Konjugierten ist, d.h. $A^{\mathrm{T}} = \overline{A}$.

2.1.1. direkte Folgen aus der Definition

- Der Realteil einer hermiteschen Matrix ist symmetrisch, $\mathrm{Re}(a_{jk}) = \mathrm{Re}(a_{kj})$, der Imaginärteil ist schiefsymmetrisch, $\mathrm{Im}(a_{jk}) = -\mathrm{Im}(a_{kj})$.
- Ist A hermitesch, dann ist iA schiefhermitesch.
- Die Hauptdiagonalelemente einer hermiteschen Matrix sind reell.
- Für reelle Matrizen fallen die Begriffe hermitesch und symmetrisch zusammen.
- Hermitesche Matrizen sind normal, d. h $A^* \cdot A = A \cdot A^*$

[1] Charles Hermite (* 24. Dezember 1822 in Dieuze (Lothringen); † 14. Januar 1901 in Paris) war ein französischer Mathematiker.

2.2. Mathematischer Beweis für die gegebene Matrize 'H'

Mit der gegebenen Bildungsvorschrifft

$$H := \left(\frac{1}{j+k+1}\right)_{j,k=0}^{n} \tag{2.3}$$

Gleichung 2.2 und den Aussagen aus 2.1.1 folgt, dass für H der folgende Zusammenhang gillt:

$$\begin{aligned} H &= H^{\mathrm{T}} && \mid \text{da rein reelwertig} && (2.4)\\ H_{j,k} &= H_{k,j} && \mid \text{Indizees vertauschbar} && (2.5)\\ \left(\frac{1}{j+k+1}\right)_{j,k=0}^{n} &= \left(\frac{1}{j+k+1}\right)_{k,j=0}^{n} && && (2.6) \end{aligned}$$

somit ist H symmetrisch und auch hermitesch.

3. Eigenwerte und Kondition von H

Die Erstellung der Matrix H wird durch ein Octave-Script[1] organisiert. Siehe hierzu A.1 und A.2.

Die berechneten Daten werden in zwei Dateien geschrieben,

```
# öffnen und schreiben in eine Datei
# mögliche Modi: r−read, w−write, a−append
fileId = fopen ('fileName','mode');
# do something
    ...
# done!
# schließen der Datei nicht vergessen
fclose (fileId);
```

um sie später mit einem externen Programm (z.B. gnuplot) weiter verarbeiten zu können. Ihnen sind die Werte in Tabelle 3.1 entnommen.

Tabelle 3.1.: Eigenwerte & Konditionen für 3..10xn Matrizen

N											Kond.
3	0.00268	0.12232	1.40831								524.06
4	9.6702 e-05	6.7383 e-03	1.6914 e-01	1.5002 e+00							1.5514 e+04
5	3.2879 e-06	3.0590 e-04	1.1407 e-02	2.0853 e-01	1.5671 e+00						4.7661 e+05
6	1.0828 e-07	1.2571 e-05	6.1575 e-04	1.6322 e-02	2.4236 e-01	1.6189 e+00					1.4951 e+07
7	3.4939 e-09	4.8568 e-07	2.9386 e-05	1.0086 e-03	2.1290 e-02	2.7192 e-01	1.6609 e+00				4.7537 e+08
8	1.1115 e-10	1.7989 e-08	1.2943 e-06	5.4369 e-05	1.4677 e-03	2.6213 e-02	2.9813 e-01	1.6959 e+00			1.5258 e+10
9	3.4998 e-12	6.4609 e-10	5.3856 e-08	2.6730 e-06	8.7581 e-05	1.9789 e-03	3.1039 e-02	3.2163 e-01	1.7259 e+00		4.9315 e+11
10	1.0931 e-13	2.2667 e-11	2.1474 e-09	1.2290 e-07	4.7297 e-06	1.2875 e-04	2.5309 e-03	3.5742 e-02	3.4293 e-01	1.7519 e+00	1.6025 e+13

[1] GNU Octave ist eine freie Software zur numerischen Lösung mathematischer Probleme, wie zum Beispiel Matrizenrechnung, Lösen von (Differential-)Gleichungssystemen, Integration etc. Berechnungen können in Octave mit einer Skriptsprache durchgeführt werden, die weitgehend zu dem proprietären MATLAB kompatibel ist.

4. Numerische Lösung des LGS für eine fünfstellige Genauigkeit von R(0)

Falls H eine $n \times n$-Matrix ist, wird von Octave Gaußelimination verwendet, sonst wird via QR-Zerlegung eine Lösung im Sinne der kleinsten Fehlerquadrate berechnet. Ist A schlecht konditioniert oder singulär, wird eine Warnung ausgegeben. Um zu zeigen, dass das Eliminationsverfahren nach Gauß ungeeignet ist haben wir hierführ eine eigene Funktion gechrieben, welche im Listing 4.1 zu sehen ist.

```
function [x] = gaussElim(A,b)
% File gaussElim.m
%   This subroutine will perform Gaussian elmination
%   on the matrix that you pass to it.
%   i.e., given A and b it can be used to find x,
%        Ax = b
%
%   To run this file you will need to specify several
%   things:
%   A - matrix for the left hand side.
%   b - vector for the right hand side
%
%   The routine will return the vector x.
%   ex: [x] = gaussElim(A,b)
%     this will perform Gaussian elminiation to find x.
%
%

 N = max(size(A));

% Perform Gaussian Elimination

 for j=2:N,
     for i=j:N,
        m = A(i,j-1)/A(j-1,j-1);
        A(i,:) = A(i,:) - A(j-1,:)*m;
        b(i) = b(i) - m*b(j-1);
     end
 end

% Perform back substitution

 x = zeros(N,1);
 x(N) = b(N)/A(N,N);

 for j=N-1:-1:1,
   x(j) = (b(j)-A(j,j+1:N)*x(j+1:N))/A(j,j);
 end
```

Quelltext 4.1: Gauß-Eliminationsverfahren als Matlab-/Octavescript

5. Visualisierung der Koeffezienten sowie des Taylorpolynoms

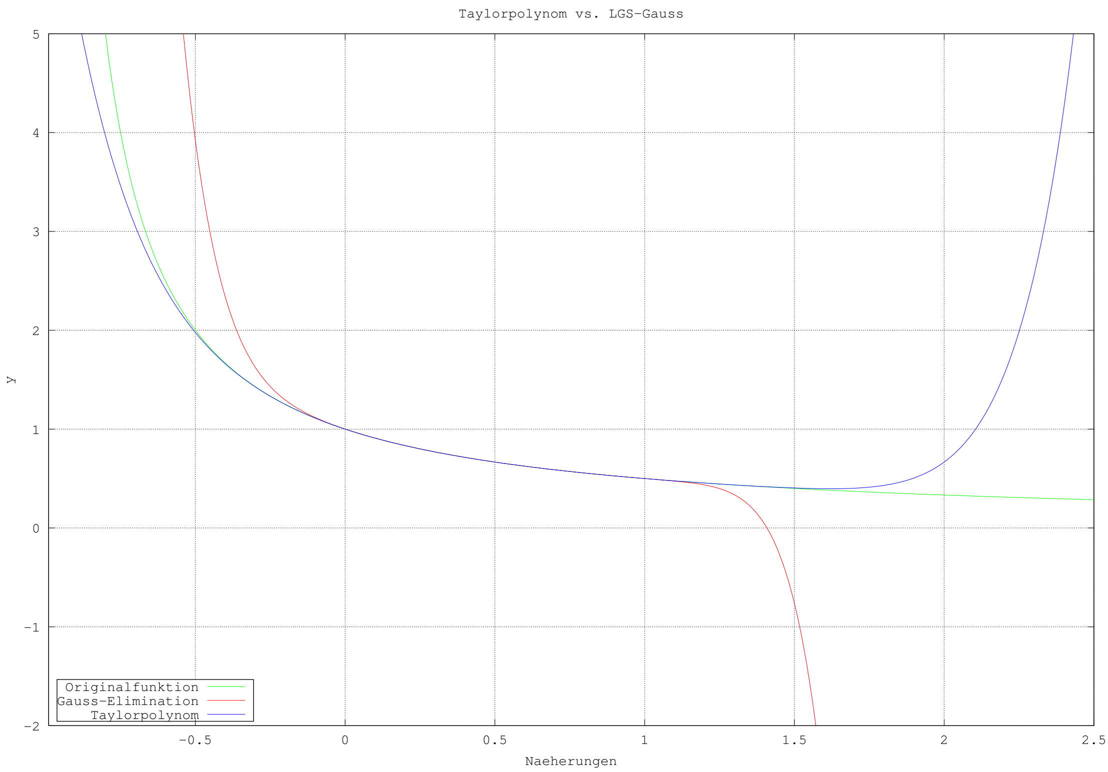

Abbildung 5.1.: Plot des Taylorpolynoms vs. Gauss-Elimination

Glossar und Abkürzungsverzeichnis

Begriff Erklärung des Begriffs

Abbildungsverzeichnis

Quelltextverzeichnis

Literaturverzeichnis

KARP, RICHARD M. (1972). *Reducibility Among Combinatorial Problems*. In: MILLER, R. E. und J. W. THATCHER, Hrsg.: *Complexity of Computer Computations*, S. 85–103. Plenum Press.

PRECHELT, LUTZ (2000). *An Empirical Comparison of Seven Programming Languages*. Foobar book of lazyness, S. 23–29.

SEIFART, MANFRED (1990). *Analoge Schaltungen*. Huethig Buch Verlag Heidelberg, Heidelberg, Baden-Wuerttemberg, Germany.

A. Anhang

A.1. Quelltexte

```
#
# Öffnen der Datei und merken der File-Id, um später in Sie zu schreiben.
# ACHTUNG! Wird 'fopen' mit dem Parameter 'w' aufgerufen, so werden existente
# Dateien überschrieben.
#
fileIdData = fopen ('data', 'w');
fileIdTask3 = fopen ('aufgabe3', 'w');
#
# Iterationen um 3x3 bis 10x10 Matrizzen zu berechnen
#
for i = 10:10
  if (!numa (i, fileIdData, fileIdTask3))
    break;
  endif
endfor
# Schließen der Dateien
fclose (fileIdData);
fclose (fileIdTask3);
```

Quelltext A.1: Foo

```
function retVal = numa (N, fileIdData, fileIdTask3)
  retVal = false;      # default Zuweisung

  if (isnumeric (N) && is_valid_file_id (fileIdData) && is_valid_file_id (
      fileIdTask3))
    clc;
    N--;
    taylorsize = 10;     # Länge des Taylor Polynoms
    entwPkt = 0.5;       # und dessen Entwicklungspunkt
    xmin = -0.99;
    xmax =  2.5;
    xval = [xmin:0.01:xmax];
    ymin = -2;
    ymax = 5.0;
    #
    # Zähler und Nenner der gegebenen Funktion
    #
    numerator = [1];
    denominator = [1, 1];
    func = deconv (numerator, denominator);
    #
    # Erzeugen der Matrix
    #
    for i = 0:N
      for j = 0:N
        H(i+1, j+1) = 1/(1 + i + j);
      endfor
    endfor
    #
    # Anlegen des Lösungsvektors
    #
```

```
R(1) = log(2);     # r0 ist gleich ln2
for i = 1:N
  sum = 0;
  for j = 1:i
    sum = sum + ((-1)^j)/j;
  endfor
   R(i+1) = (-1)^i * (log(2) + sum);
endfor
Rtrim = R;
Rtrim(1) = 0.69315; # 5-Stellen Genauigkeit
#
# Lösen des LGS
#
solved = H\R'
solvedTrim = gaussElim (H, Rtrim)
#
# Ausgabe
#
printf ("Matrix␣'H'␣%ix%i:\n" ,N+1 ,N+1)
disp (H)
commentLine = strcat ("#\n#␣'H'␣Matrix-", num2str (N+1), "x", num2str (N+1), "\n#");
fdisp (fileIdData, commentLine);
fdisp (fileIdData, H);
printf ("zugehörige␣Lösungsvektoren␣'r'␣und␣'rTrim':\n")
disp (R)
disp (Rtrim)
commentLine = "#\n#␣'R'␣Lösungsvektor\n#";
fdisp (fileIdData, commentLine);
fdisp (fileIdData, R);
commentLine = "#\n#␣Lösung␣des␣LGS\n#";
fdisp (fileIdData, commentLine);
fdisp (fileIdData, solved');
printf ("Eigenwerte:\n")
lambda = eig (H);
disp (lambda)
commentLine = strcat ("#\n#␣Eigenwerte␣der␣Matrizze␣'H'␣-", num2str (N+1), "x", num2str (N+1), "\n#");
fdisp (fileIdTask3, commentLine);
fdisp (fileIdTask3, lambda');   # lambda ist transponiert, um besser lesbar in eine Datei zu schreiben
printf ("Kondition:\n")
cond = cond (H);
disp (cond)
commentLine = "#\n#␣Kondition\n#";
fdisp (fileIdTask3, commentLine);
fdisp (fileIdTask3, cond);
printf ("\n")
#
# Berechnen des Taylorpolynoms
#
taylor = zeros (1, size (xval, 2));
for i = 1:size (xval, 2)
  for n = 0:taylorsize
    taylor(i) += (-1)^n * ( (xval(i)-entwPkt)^n / ((1+entwPkt)^(n+1)) );
  endfor
endfor
#
# Plot des Polynoms
#
solved = solved(end:-1:1);  # Potenzen umsortieren
```

```
    solvedTrim = solvedTrim(end:-1:1);
    # berechne die ursprüngliche Funktion
    ref = zeros (1, N);
    for n = 1:size (xval, 2)
      ref(n) = 1/(1+xval(n));
    endfor
    plotPdf (xval, ref, solvedTrim, taylor, xmin, xmax, ymin, ymax); # Aufruf des externen Plot Scripts
    #
    # Return bool 'true'
    #
    retVal = true;
  else
    printf ("Fehlerhafter Funktionsaufruf!\nSyntax: function (numericValue, fileId, fileId)\n")
  endif
endfunction
```

Quelltext A.2: Berechnungen für eine nxn Matrize