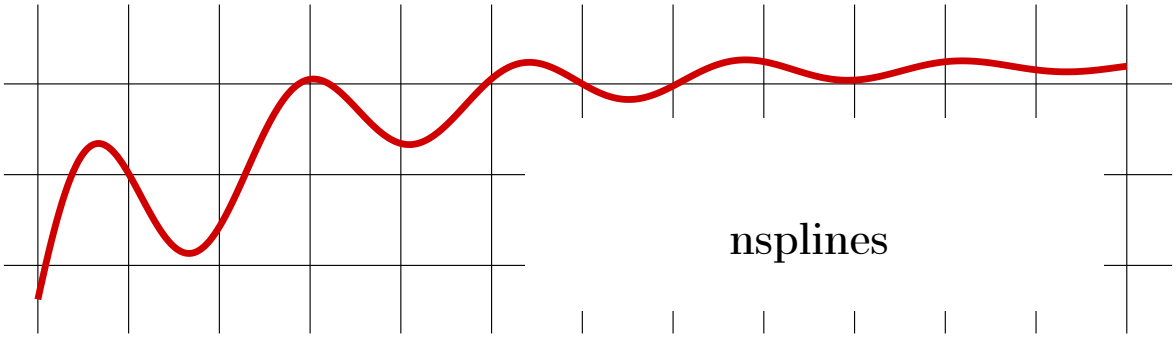

Dipl.-Ing. D. Krause

8. Dezember 2008

Inhaltsverzeichnis

1 Übersicht

1.1 Zweck des nsplines-Paketes

Das nsplines-Paket stellt einige *.m-Dateien bereit, um Messwerte mit GNU Octave und SciLab zu bearbeiten. Dazu werden die Messwerte mittels Natural Splines interpoliert.

1.2 Lizenzbedingungen

Die Software ist unter einer Lizenz im BSD-Stil an Sie lizensiert:

- Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
  - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above opyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  - Neither the name of the Dirk Krause nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
- This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed.
  In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

2 Installation

2.1 Installation für Octave

2.1.1 Systemweite Installation

Starten Sie GNU Octave, geben Sie am Prompt

```
1 DEFAULT_LOADPATH
```

ein.
Als Antwort erhalten Sie eine Liste mit den Verzeichnissen, die GNU Octave nach Funktions-Dateien durchsucht. Die einzelnen Verzeichnisse sind durch Doppelpunkte voneinander getrennt. Verzeichnisse, die mit zwei Slashes am Ende geschrieben sind, werden mitsamt Unterverzeichnissen durchsucht.
In der Liste wird ein Verzeichnis mit zwei Slashes am Ende gewählt, das im Namen weder eine Oktave-Versionsnummer, noch eine API-Versionsnummer, noch die aktuelle Hardwarearchitektur enthält. Ich bevorzuge hier „/usr/share/octave/site/m“, falls vorhanden.
In diesem Verzeichnis wird ein Unterverzeichnis „nsplines“ erstellt, die *.m-Dateien aus dem „m“-Verzeichnis werden dorthin kopiert.

2.1.2 Installation für einen Nutzer

Falls nicht schon vorhanden, wird zunächst ein nutzerspezifisches Octave-Verzeichnis im Home-Verzeichnis angelegt.
Für einen Nutzer „Erwin Unsinn“ mit dem Login-Namen „eunsinn“ und dem Home-Verzeichnis „/home/eunsinn“ wird also das Verzeichnis „/home/eunsinn/octave“ angelegt.
In der Datei „.octaverc“ im Home-Verzeichnis (Datei erstellen, falls noch nicht vorhanden) wird die Variable „LOADPATH“ so gesetzt, dass sie das neue Verzeichnis mit allen Unterverzeichnissen benutzt sowie die bisherigen Standardeinstellungen aus der Variable „DEFAULT_LOADPATH“.

```
1 LOADPATH='/home/eunsinn/octave//::';
```

Die zwei Slashes am Ende des Verzeichnisses bedeutet, dass dieses mitsamt Unterverzeichnissen durchsucht wird. Die zwei Doppelpunkte binden an dieser Stelle den Standardpfad ein.
Es wird nun im nutzerspezifischen Octave-Verzeichnis ein Unterverzeichnis „nsplines“ angelegt, die *.m-Dateien aus dem „m“-Verzeichnis werden dorthin kopiert.

```
1 mkdir /home/eunsinn/octave/nsplines
2 cp m/*.m /home/eunsinn/octave/nsplines
```

2.2 Installation für SciLab

2.2.1 Systemweite Installation

Im SciLab-Verzeichnis (z.B. c:\programme\scilab-4.0) legen Sie im Unterverzeichnis „macros“ ein neues Verzeichnis „nsplines“ an. Die Dateien sci*.sci werden in das neu angelegte Verzeichnis kopiert.
Als root bzw. Administrator wird SciLab gestartet und dann das Kommando

```
genlib('nsplines', 'SCI/macros/nsplines');
```

ausgeführt.
Optional fügen Sie in c:\programme\scilab-4.0\scilab.star eine Zeile

```
load('SCI/macros/nsplines/lib');
```

ein, damit die Funktionen nach dem Programmstart sofort für alle Benutzer verfügbar sind.

2.2.2 Installation für einen Nutzer

Kopieren Sie die Dateien sci*.sci in ein Verzeichnis, auf das Sie Schreibrechte besitzen. Das Verzeichnis sollte so gewählt werden, dass die vollständigen Dateinamen (z.B. „c:\erwin\...\ns_create.sci“) keine Leerzeichen enthalten.
Optional erstellen Sie in Ihrem Homeverzeichnis eine Datei „.scilab“ und tragen eine Zeile

```
exec('c:\jjames\octave\...\ns_create.sci');
exec('c:\jjames\octave\...\ns_polynom.sci');
exec('c:\jjames\octave\...\ns_value.sci');
exec('c:\jjames\octave\...\ns_gpfile.sci');
exec('c:\jjames\octave\...\nssp_create.sci');
exec('c:\jjames\octave\...\nssp_polynom.sci');
exec('c:\jjames\octave\...\nssp_value.sci');
exec('c:\jjames\octave\...\nssp_gpfile.sci');
exec('c:\jjames\octave\...\ns_nssp.sci');
exec('c:\jjames\octave\...\ns_psfile.sci');
exec('c:\jjames\octave\...\nssp_psfile.sci');
```

ein, um die Funktionen nach dem Programmstart sofort zur Verfügung zu haben.

3 Theorie

3.1 Natural Splines

Natural Splines sind ein altbekanntes mathematisches Mittel, um Kurven zu interpolieren, wenn einzelne Punkte der Kurve gegeben sind. Die Kurve wird dabei stückweise aus einzelnen Polynomen dritten Grades zusammengesetzt.
Das Polynom

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

beschreibt dabei den Kurvenverlauf zwischen den Punkten i und $i + 1$, d.h. für x-Werte mit $x_i \leq x \leq x_{i+1}$.
Zwischen n Punkten bestehen $n-1$ Zwischenräume, somit werden $n-1$ Polynome benötigt. Da jedes Polynom 4 Koeffizienten aufweist, müssen $4(n-1)$ Polynom-Koeffizienten bestimmt werden, hierzu werden $4(n-1)$ Gleichungen benötigt.
Die Gleichungen werden folgendermaßen gewonnen:

(a) $n-1$ Gleichungen $(1 \leq i \leq n-1)$
für den Messwert am jeweils linken Polynomrand

$$d_i = y_i$$

(b) $n-1$ Gleichungen $(1 \leq i \leq n-1)$
für den Messwert am jeweils rechten Polynomrand

$$(x_{i+1} - x_i)^3 a_i + (x_{i+1} - x_i)^2 b_i + (x_{i+1} - x_i)c_i + d_i = y_{i+1}$$

(c) $n-2$ Gleichungen $(2 \leq i \leq n-1)$
aus der folgenden Stetigkeitsbedingung: An den inneren Punkten ist die erste Ableitung des jeweils linken und rechten Polynomes gleich.

$$3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} = c_i$$
$$3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} - c_i = 0$$

(d) $n-2$ Gleichungen $(2 \leq i \leq n-1)$
aus der folgenden Stetigkeitsbedingung: An den inneren Punkten ist die zweite Ableitung des jeweils linken und rechten Polynomes gleich.

$$6(x_i - x_{i-1})a_{i-1} + 2b_{i-1} = 2b_i$$
$$6(x_i - x_{i-1})a_{i-1} + 2b_{i-1} - 2b_i = 0$$

(e) 2 Gleichungen
aus der folgenden Randbedingung: An den äußeren Punkten ist die zweite Ableitung des jeweiligen Polynomes 0.

$$\begin{aligned} 2b_1 &= 0 \\ 6a_{n-1}(x_n - x_{n+1}) + 2b_{n-1} &= 0 \end{aligned}$$

3.2 Modifizierte Natural Splines

Mitunter sind nicht nur einzelne Messpunkte vorgegeben sondern auch der Anstieg in diesen Punkten. In diesem Fall müssen für innere Punkte die Stetigkeitsbedingungen bzw. für äußere Punkte die Randbedingungen durch Gleichungen ersetzt werden, die den Anstieg im Punkt ausdrücken.

- Innere Punkte
  Die unter (c) und (d) aufgestellten Gleichungen

  $$\begin{aligned} 3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} - c_i &= 0 \\ 6(x_i - x_{i-1})a_{i-1} + 2b_{i-1} - 2b_i &= 0 \end{aligned}$$

  werden ersetzt durch

  $$\begin{aligned} 3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} &= y'_i \\ c_i &= y'_i \end{aligned}$$

- Äußere Punkte
  Je nachdem, ob es sich um den ersten oder letzten Punkt handelt, wird die entsprechende Gleichung (eine der beiden)

  $$\begin{aligned} 2b_1 &= 0 \\ 6a_{n-1}(x_n - x_{n+1}) + 2b_{n-1} &= 0 \end{aligned}$$

  ersetzt durch

  $$\begin{aligned} c_1 &= y'_1 \\ 3(x_n - x_{n-1})^2 a_{n-1} + 2(x_n - x_{n-1})b_{n-1} + c_{n-1} &= y'_n \end{aligned}$$

Somit stehen für die $4(n-1)$ Polynomkoeffizienten wieder $4(n-1)$ Gleichungen zur Verfügung.

3.3 NSSP

In manchen Situationen (z.B. wenn die Polynome abgeleitet oder integriert werden sollen), ist es besser, einfache Polynome der Form

$$f_i(x) = p_i x^3 + q_i x^2 + r_i x + s_i$$

zu verwenden.

$$\begin{aligned} f_i(x) &= a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \\ &= a_i x^3 + (b_i - 3a_i x_i)x^2 + (c_i - 2b_i x_i + 3a_i x_i^2)x + d_i - c_i x_i + b_i x_i^2 - a_i x_i^3 \end{aligned}$$

Somit ergibt sich:

$$\begin{aligned} p_i &= a_i \\ q_i &= b_i - 3a_i x_i \\ r_i &= c_i - 2b_i x_i + 3a_i x_i^2 \\ s_i &= d_i - c_i x_i + b_i x_i^2 - a_i x_i^3 \end{aligned}$$

Die Abkürzung NSSP steht für natural splines, simple polynomials.

4 Nutzung

Folgende Funktionen werden bereitgestellt

- M = ns_create(mw)
  Die Funktion erwartet als Eingabe *mw* eine Messwerte-Matrix, die in der ersten Spalte die x-Werte und in der zweiten Spalte die y-Werte der Messpunkte enthält. Sind für einige Messpunkte Anstiege vorgegeben, besteht die Messwerte-Matrix aus 4 Spalten. Ist der Wert in der dritten Spalte größer als 0, so enthält die vierte Spalte den Anstieg $\frac{dy}{dy}$ im Messpunkt.

  Zurückgegeben wird eine Matrix, die in jeder Zeile die Kennwerte für ein Segment enthält: x-Startwert, x-Endwert, a, b, c und d.

- y = ns_value(M, x)
  Die Funktion erwartet als Eingabe M die mit *ns_ create()* erzeugte Matrix und einen x-Wert. Für den x-Wert wird ein Funktionswert ausgerechnet und zurückgegeben.

- ns_gpfile(mw, M, name)
  Die Funktion schreibt eine rudimentäre GnuPlot-Datei, die die Messpunkte und die Polynomfunktionen darstellt.

- ns_psfile(mw, M, name)
  Diese Funktion schreibt PostScript-Code zur Berechnung von Funktionswerten in eine Datei.
  Dieser Code kann dann in LaTeX-Quellen mit dem Paket pst-plot verwendet werden, um die Natural-Spline-Interpolation graphisch darzustellen.

- nssp_ . . . ()
  Die Funktionen sind äquivalent zu *ns_ . . . ()*, mit dem Unterschied, dass Polynomkoeffizienten für einfache Polynome in der Matrix gespeichert werden.
  Hinweis: Auf Matrizen, die mit *ns_ create()* erzeugt wurden, dürfen nur die *ns_ . . . ()*-Funktionen angewendet werden. Auf Matrizen, die mit *nssp_ create()* erzeugt wurden, dürfen nur die *nssp_ . . . ()*-Funktionen angewendet werden.

5 Beispiele

5.1 BH-Kennlinie eines Dauermagneten

Gegeben sei eine B-H-Kennlinie in Form von Messdaten (siehe Tabelle 1).

Tabelle 1: Beispiel-Messdaten

H/Acm^{-1}	B/T
-525	0
-470	0,50
-420	0,72
-330	0,94
-220	1,12
-120	1,22
0	1,30

Gesucht wird der magnetische Fluss B bei einer Feldstärke von $H = -262\,\mathrm{Acm}^{-1}$.
Die Datei ns0001.m[1]

```
# Messwerte / measurement results
#         H         B
mw = [   -525.0   0.0
         -470.0   0.5
         -420.0   0.72
         -330.0   0.94
         -220.0   1.12
         -120.0   1.22
         0.0      1.3  ];

global M;
M = ns_create(mw);

Hvalue = -262.0;
Bvalue = ns_value(M, Hvalue);
printf('B = %g\n', Bvalue);
```

erzeugt bei der Verarbeitung mit

```
octave -q ns0001.m
```

die Ausgabe

[1] Die SciLab-Datei ns0001.sce finden Sie im Anhang A.1.1 auf Seite 25

```
B = 1.06071
```

Es tritt also ein magnetischer Fluss von $B = 1,06\,\mathrm{T}$ auf.

Um die Polynom-Funktion zu drucken, wird zunächst eine Octave-Datei ns0002.m[2] erstellt:

```
# Messwerte / measurement results
#         H         B
mw = [   -525.0    0.0
         -470.0    0.5
         -420.0    0.72
         -330.0    0.94
         -220.0    1.12
         -120.0    1.22
          0.0      1.3  ];

M = ns_create(mw);
ns_gpfile(mw, M, "ns0002.gp", "native");
```

Diese wird mit

```
octave -q ns0002.m
```

verarbeitet und erzeugt ns0002.gp. An den Anfang dieser Datei werden zwei Anweisungen eingefügt, die den Ausgabetyp und den Dateinamen für die Ausgabedatei bestimmen:

```
set terminal mp c latex psnfss
set output "ns0002.mp"
```

Weitere Änderungen (z.B. Setzen von Labels und Beschriftungen, Verschiebung der Legende...) sind ebenfalls möglich.
Diese Datei wird dann mit den Kommandos

```
gnuplot ns0002.gp
mpost -tex=latex ns0002
mv ns0002.0 ns0002.mps
```

in eine *.mps-Datei umgewandelt, die dann hier in das Dokument eingebunden werden kann (siehe Abb. 1 auf der nächsten Seite).

[2] Anhang A.1.2 auf Seite 26 enthält die SciLab-Datei ns0002.sce

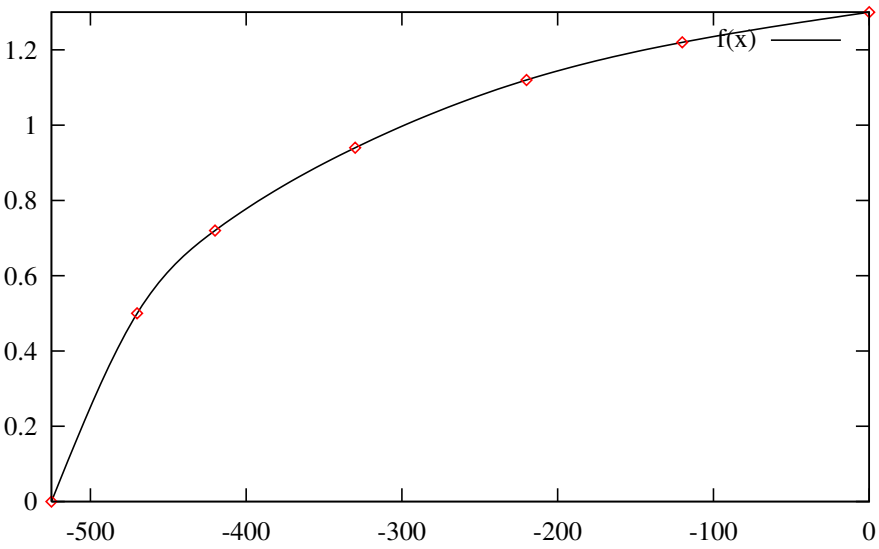

Abbildung 1: Kurvenverlauf

Gesucht wird nun der H-Wert H_x, für den sich $B_x = 0,8\,\mathrm{T}$ einstellt. Hierzu wird zunächst wieder eine Natural-Spline-Interpolation durchgeführt. Mit Hilfe der Interpolations-Matrix $\mathcal{M}$ wird eine Funktion

$$f(H) = B(H) - B_x$$

definiert. Für deren Nullstelle H_0 gilt $B(H) = B_x$. Auf diese Funktion wird das Verfahren zum Lösen nichtlinearer Gleichungssysteme angewendet:[3]

```
# Messwerte / measurement results
#          H          B
mw = [   -525.0    0.0
         -470.0    0.5
         -420.0    0.72
         -330.0    0.94
         -220.0    1.12
         -120.0    1.22
         0.0       1.3  ];

global M;
M = ns_create(mw);

global wishValue;
wishValue = 0.8;

function y = f(x)
  global M; global wishValue;
  y = ns_value(M, x) - wishValue;
endfunction

printf('For H = -262A/cm we have B = %gT\n', ns_value(M, -262.0));

[x, info] = fsolve("f", [-330.0]);
if (info == 1)
  printf('Successfully finished iteration.\n');
  printf('For B = 0.8T we need H = %g\n', x);
else
  perror("fsolve", info);
endif
ns_gpfile(mw, M, "ns0007.gp");
```

Aus der Ausgabe

```
Success.
H = -391.629
```

[3] auch wenn hier nur eine Gleichung vorhanden ist

ergibt sich $H_x = -391,6\,\mathrm{Acm}^{-1}$.

5.2 Beispiel für Messdaten mit Anstieg

Gegeben seien fiktive Messwerte einer Größe y in Abhängigkeit von x, siehe Tabelle 2. Es sei bekannt, dass immer

$$0 \leq y \leq 1$$

gilt.

Tabelle 2: Beispiel-Messdaten

x	y
1	0
2	0
3	0
4	0,5
5	1
6	1
7	1

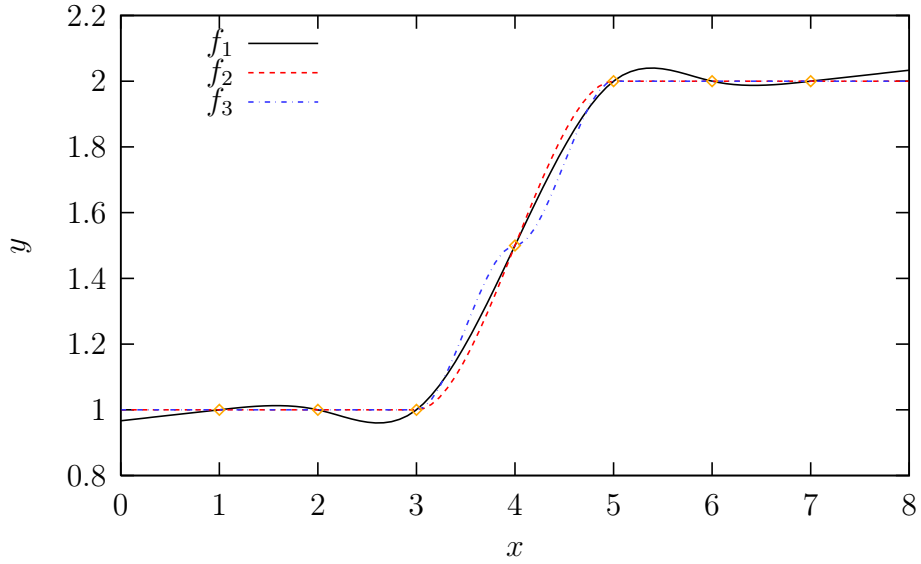

Abbildung 2: Verschiedene Kurven für dieselben Messwerte

```
mw = [
        1       1
        2       1
        3       1
        4       1.5
        5       2
        6       2
        7       2
];
[mwr, mwc] = size(mw);
M = ns_create(mw);
ns_gpfile(mw, M, "ns0004.gp", "native");
```

Werden GNU Octave nur die Messwerte (ohne Angabe von Anstiegen) vorgegeben, ergibt die Natural-Spline-Interpolation die Kurve f_1 in Abb. 2.
Deutlich sichtbar sind die Oszillationen, die Interpolation verläßt den vorgegebenen Wertebereich.

Um die Oszillationen zu unterbinden, wird ein Anstieg für die Punkte $x = 3$ und $x = 5$ vorgegeben. In den Zeilen für diese Punkte wird in der dritten Spalte ein positiver Wert eingetragen, die vierte Spalte enthält den Anstieg.

```
mw = [
        1       1       0       0
        2       1       0       0
        3       1       1       0
        4       1.5     0       0
        5       2       1       0
        6       2       0       0
        7       2       0       0
];
```

Es entsteht die Kurve f_2, die keine Oszillationen mehr aufweist.
Durch Angabe weiterer Anstiege kann die Interpolationskurve beeinflusst werden.

```
mw = [
        1       1       0       0
        2       1       0       0
        3       1       1       0
        4       1.5     1       0
        5       2       1       0
        6       2       0       0
        7       2       0       0
];
```

Wird beispielsweise auch der Anstieg im Punkt $x = 4$ auf 0 gesetzt, so entsteht die Interpolationsfunktion f_5.
Für eine optimale Darstellung wurden alle drei Funktionen in eine GnuPlot-Datei übernommen.

5.3 Optmierter Dauermagnet

Für das in Abschnitt 5.1 auf Seite 10 gegebene Material soll der optimale Arbeitspunkt bestimmt werden, d.h. der Punkt der Kennlinie, an dem das Produkt $|B \cdot H|$ den Maximalwert erreicht. Hierfür gibt es zwei Methoden:

- Ein graphisches Näherungsverfahren.
  Es wird ein Rechteck mit den Kantenlängen B_r und H_c eingezeichet. Der Schnittpunkt der Rechteck-Diagonalen mit der Kurve liefert die Näherung für den optimalen Arbeitspunkt.
  Für die rechnerische Lösung bedeutet dies:

  $$B = f(H) = H \cdot \frac{B_r}{H_c}$$

  Es muss also die Nullstelle von

  $$g(H) = f(H) - H \cdot \frac{B_r}{H_c}$$

  gesucht werden.

- Numerische Lösung.

  $$\begin{aligned} BH &= f_i(H) \cdot H \\ &= (p_i H^3 + q_i H^2 + r_i H + s_i) \cdot H \\ &= p_i H^4 + q_i H^3 + r_i H^2 + s_i H \end{aligned}$$

  Die Ableitung des Produktes ergibt

  $$\frac{\mathrm{d}}{\mathrm{d}H}(BH) = 4p_i H^3 + 3q_i H^2 + 2r_i H + s_i$$

  Bei der Suche nach dem Maximalwert des Produktes muss also die Nullstelle von

  $$h_i(H) = 4p_i H^3 + 3q_i H^2 + 2r_i H + s_i$$

  gesucht werden.

  Auch diese Funktion ist als Natural Spline darstellbar, die Interpolationsmatrix mit den Polynomkoeffizienten kann erzeugt werden, indem zunächst eine Kopie der Interpolationsmatrix von $f(H)$ erstellt wird, in der anschließend die Koeffizientenspalten mit 4, 3 und 2 multipliziert werden.

Wichtig ist, dass die *nssp_ . . . ()*-Funktionen verwendet werden, da hier mit einfachen Polynomen gearbeitet wird.

Die Eingabedatei ns0010.m[4] wendet beide Verfahren an:

```
mw = [   -525.0   0.0
         -470.0   0.5
         -420.0   0.72
         -330.0   0.94
         -220.0   1.12
         -120.0   1.22
         0.0      1.3  ];

global M1;
M1 = nssp_create(mw);

global M2;
M2 = M1;

[r,c] = size(M2);

for i=1:r
  M2(i,3) = 4.0*M2(i,3);
  M2(i,4) = 3.0*M2(i,4);
  M2(i,5) = 2.0*M2(i,5);
endfor

function back = g(H)
  global M1;
  back = nssp_value(M1, H) - 1.3 * H / (-525.0);
endfunction

function back = h(H)
  global M2;
  back = nssp_value(M2, H);
endfunction

[H, info] = fsolve("g", [-525.0/2.0]);
if (info == 1)
  printf('Grapical approximation solved successfully.\n');
  printf('H    = %g\n', H);
  printf('B    = %g\n', nssp_value(M1, H));
  printf('|BH| = %g\n', abs(H*nssp_value(M1,H)));
else
  perror("fsolve", info);
endif
```

[4] Anhang A.1.3 auf Seite 27 enthält die SciLab-Datei ns0007.sce

```
42
43 [H, info] = fsolve("h", [-525.0/2.0]);
44 if (info == 1)
45   printf('Optimizatzion solved successfully.\n');
46   printf('H    = %g\n', H);
47   printf('B    = %g\n', nssp_value(M1, H));
48   printf('|BH| = %g\n', abs(H*nssp_value(M1,H)));
49 else
50   perror("fsolve", info);
51 endif
```

Die entsprechende Ausgabe von GNU Octave

```
1 Grapical approximation solved successfully.
2 H    = -356.844
3 B    = 0.883614
4 |BH| = 315.313
5 Optimizatzion solved successfully.
6 H    = -368.153
7 B    = 0.857926
8 |BH| = 315.848
```

zeigt, dass die graphische Lösung eine recht gute Näherung liefert.

6 Interne Arbeitsweise

6.1 Erzeugung der Polynome

Wie bereits festgestellt, liegt ein Gleichungssystem mit $4(n-1)$ Gleichungen und $4(n-1)$ Unbekannten (den Polynom-Koeffizienten) vor. Die Gleichungen aus (a) führen dabei direkt zu Lösungen für die d_i.
Es verbleibt ein Gleichungssystem mit $3(n-1)$ Gleichungen und $3(n-1)$ Unbekannten. Die Lösung dieses Gleichungssystemes wird GNU Octave überlassen. Hierzu muss jedoch zunächst eine Koeffizientenmatrix und ein Ergebnisvektor bereitgestellt werden.
Unser Lösungsvektor soll ein Spaltenvektor werden, der der Reihe nach a_1, b_1, c_1, $a_2 \ldots c_{n-1}$ enthält. Die Position der a_i innerhalb des Vektors ergibt sich als $3(i-1)+1$, die der b_i zu $3(i-1)+2$ und die der c_i zu $3(i-1)+2$.
In die Koeffizientenmatrix werden der Reihe nach folgende Gleichungen eingetragen:[5]

(b) $n-1$ Gleichungen für den jeweils rechten Polynomrand. Es gilt $1 \leq i \leq (n-1)$. Die Zeilennumer entspricht dem jeweiligen i.

$$(x_{i+1}-x_i)^3 a_i + (x_{i+1}-x_i)^2 b_i + (x_{i+1}-x_i)c_i + d_i = y_{i+1}$$

wird umgeformt zu

$$(x_{i+1}-x_i)^3 a_i + (x_{i+1}-x_i)^2 b_i + (x_{i+1}-x_i)c_i + y_i = y_{i+1}$$
$$(x_{i+1}-x_i)^3 a_i + (x_{i+1}-x_i)^2 b_i + (x_{i+1}-x_i)c_i + = y_{i+1} - y_i$$

(c) $n-2$ Gleichungen für die Stetigkeitsbedingungen der inneren Punkte (Gleichheit der ersten Ableitung des jeweils linken und rechten Polynomes). Es gilt $2 \leq i \leq (n-1)$. Für jedes i ergibt sich die Zeilennumer z zu $z = n + i - 2$.

$$3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} - c_i = 0$$

(d) $n-2$ Gleichungen für die Stetigkeitsbedingungen der inneren Punkte (Gleichheit der zweiten Ableitung des jeweils linken und rechten Polynomes). Es gilt $2 \leq i \leq (n-1)$. Für jedes i ergibt sich die Zeilennummer z zu $z = 2n + i - 4$.

$$6(x_i - x_{i-1})a_{i-1} + 2b_{i-1} - 2b_i = 0$$

[5] Die Aufzählung beginnt mit (b) wegen der Übereinstimmung mit Abschnitt 3.1 auf Seite 6

(e) 2 Gleichungen für die Randbedingungen am ersten und letzten Punkt, die zweite Ableitung des jeweiligen Polynomes soll 0 werden. Die Zeilennummern sind $z = 3n - 4$ und $z = 3n - 3$.

$$\begin{aligned} 2b_1 &= 0 \\ 6a_{n-1}(x_n - x_{n+1}) + 2b_{n-1} &= 0 \end{aligned}$$

Falls Anstiege vorgegeben sind, werden folgende Änderungen am Gleichungssystem vorgenommen:

- Handelt es sich um den ersten Punkt ($i = 1$), wird Gleichung $z = 3n - 4$ aus (e) durch die Gleichung

  $$c_1 = y'_1$$

  ersetzt.

- Handelt es sich um den letzten Punkt ($i = n$), wird die Gleichung $z = 3n - 3$ aus (e) durch

  $$3(x_n - x_{n-1})a_{n-1}^2 + 2(x_n - x_{n-1})b_{n-1} + c_{n-1} = y'_n$$

  ersetzt.

- Handelt es sich um einen inneren Punkt ($1 < i < n - 1$), werden die Gleichungen $z = n + i - 2$ aus (c) und $z = 2n + i - 4$ aus (d) durch

  $$\begin{aligned} 3(x_i - x_{i-1})^2 a_{i-1} + 2(x_i - x_{i-1})b_{i-1} + c_{i-1} &= y'_i \\ c_i &= y'_i \end{aligned}$$

Dieses Gleichungssystem wird mit GNU Octave gelöst, aus dem Lösungsvektor werden die Polynomkoeffizienten a_i, b_i und c_i ausgelesen und in die Matrix eingetragen, die von *ns_ create()* zurückgegeben wird. Die bereits bekannten d_i werden ebenfalls eingetragen. Die zurückgegebene Matrix hat die Form

$$\begin{pmatrix} x_1 & x_2 & a_1 & b_1 & c_1 & d_1 \\ x_2 & x_3 & a_2 & b_2 & c_2 & d_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & x_n & a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \end{pmatrix}$$

6.2 Berechnung von Werten

Zur Berechnung von Werten wird zunächst geprüft, ob der vorgegebene x-Wert innerhalb des Definitionsbereiches der Interpolationskurve liegt ($x_1 \leq x \leq x_n$).
Außerhalb des Definitionsbereiches wird die Interpolationskurve von den äußeren Punkten ausgehend linear fortgesetzt.
Liegt x im Definitionsbereich, so wird die Matrix-Zeile für das Intervall gesucht, das x enthält. Mit den x_i, a_i, b_i, c_i und d_i aus dieser Zeile wird dann der Funktionswert berechnet.

A SciLab-Dateien

A.1 BH-Kennlinie eines Dauermagneten

A.1.1 Berechnung

```
// Messwerte / measurement results
//        H         B
mw = [   -525.0    0.0
         -470.0    0.5
         -420.0    0.72
         -330.0    0.94
         -220.0    1.12
         -120.0    1.22
         0.0       1.3  ];

global M;
M = ns_create(mw);

Hvalue = -262.0;
Bvalue = ns_value(M, Hvalue);
printf('B = %g\n', Bvalue);
```

A.1.2 GnuPlot-Datei erzeugen

```
1 mw = [    -525.0    0.0
2           -470.0    0.5
3           -420.0    0.72
4           -330.0    0.94
5           -220.0    1.12
6           -120.0    1.22
7           0.0       1.3   ];
8
9 M = ns_create(mw);
10 ns_gpfile(mw, M, "ns0002.gp");
```

A.1.3 H-Wert für vorgegebenes B berechnen

```
mw = [   -525.0   0.0
         -470.0   0.5
         -420.0   0.72
         -330.0   0.94
         -220.0   1.12
         -120.0   1.22
         0.0      1.3  ];

global M;
M = ns_create(mw);

global wishValue;
wishValue = 0.8;

function y = f(x)
  global M; global wishValue;
  y = ns_value(M, x) - wishValue;
endfunction

printf('For H = -262A/cm we have B = %gT\n', ns_value(M, -262.0));

[x, v, info]=fsolve([-330.0],f);
if (info == 1)
  printf('For B = 0.8T we need H = %g\n', x);
else
  printf('Failed to solve problem!\n');
end

ns_gpfile(mw, M, "ns0007.gp");
```

A.2 Beispiel für Messdaten mit Anstieg

```
1  mw = [
2              1          1
3              2          1
4              3          1
5              4          1.5
6              5          2
7              6          2
8              7          2
9  ];
10 [mwr, mwc] = size(mw);
11 M = ns_create(mw);
12 ns_gpfile(mw, M, "ns0004.gp");
```

A.3 Optimierter Dauermagnet

```
mw = [   -525.0   0.0
         -470.0   0.5
         -420.0   0.72
         -330.0   0.94
         -220.0   1.12
         -120.0   1.22
         0.0      1.3  ];

global M1;
M1 = nssp_create(mw);

global M2;
M2 = M1;
[r,c] = size(M2);
for i=1:r
  M2(i,3) = 4.0*M2(i,3);
  M2(i,4) = 3.0*M2(i,4);
  M2(i,5) = 2.0*M2(i,5);
end

function back = g(H)
  global M1;
  back = nssp_value(M1, H) - 1.3 * H / (-525.0);
endfunction

function back = h(H)
  global M2;
  back = nssp_value(M2, H);
endfunction

[H, v, info] = fsolve([-525.0/2.0], g);
printf('Grapical approximation solved successfully.\n');
if (info == 1)
  printf('H    = %g\n', H);
  printf('B    = %g\n', nssp_value(M1, H));
  printf('|BH| = %g\n', abs(H*nssp_value(M1,H)));
else
  printf('Error: Failed to solve problem!\n');
end

[H, v, info] = fsolve([-525.0/2.0], h);
if (info == 1)
```

```
43    printf('Optimizatzion solved successfully.\n');
44    printf('H     = %g\n', H);
45    printf('B     = %g\n', nssp_value(M1, H));
46    printf('|BH| = %g\n', abs(H*nssp_value(M1,H)));
47  else
48    printf('Error: Failed to solve problem!\n');
49  end
```