

---

MARTIN-LUTHER-UNIVERSITÄT  
HALLE-WITTENBERG  
FACHBEREICH MATHEMATIK UND INFORMATIK



---

**Arbeit im Mathematischen Praktikum**  
Anwendung linear-impliziter Zweischnitt-Methoden auf  
differential-algebraische Testgleichungen

**Eingereicht von** : cand. math. Patrick Reichert  
**Betreuer** : Prof. R. Weiner, H. Podhaisky

Halle (Saale), November 2001



# Vorwort

Diese Ausarbeitung entstand im Rahmen des mathematischen Praktikums im Studiengang Diplom-Mathematik an der Martin-Luther-Universität Halle. Das Projekt wurde betreut von Prof. Rüdiger Weiner und Helmut Podhaisky des Institutes für Numerische Mathematik.

Basierend auf dem vom Institut für Numerische Mathematik erstellten Programmpaket „ODETEST“ wurde das Lösungsverhalten verschiedener Differentialgleichungen getestet, die eine Forschungsgruppe des nationalen Forschungsinstitutes für Mathematik und Computerwissenschaft der Niederlande (CWI) zusammengestellt hat.

Weiterhin war die Neuentwicklung eines Programmteils nötig, welcher die Umformulierung impliziter Differentialgleichungen in eine für das bestehende Programmpaket lösbare Form durchführte.

Bedanken möchte ich mich vor allem bei Prof. Rüdiger Weiner und Helmut Podhaisky für die tatkräftige Unterstützung bei der Bearbeitung der Praktikumsaufgabe.

Mein besonderer Dank gilt meiner Lebensgefährtin Alena für die liebevolle Unterstützung bei der Entstehung dieser Arbeit sowie für das entgegengebrachte Verständnis und ihre Geduld.

Halle, November 2001



# Inhalt

<b>1 DIE EINLEITUNG</b>	<b>6</b>
<b>2 DIE GLEICHUNGEN</b>	<b>6</b>
<b>3 DIE UMFORMULIERUNG</b>	<b>6</b>
<b>4 DIE VERFAHREN</b>	<b>7</b>
<b>5 DIE PROGRAMMIERWEITERUNGEN</b>	<b>8</b>
5.1 AUFSPLITTUNG NACH EINZELNEN INDIZES	8
5.2 PROGRAMMTECHNISCHE UMSETZUNG DER UMFORMULIERUNG	10
5.3 ÜBERPRÜFUNG DER FUNKTIONSFÄHIGKEIT DES NEUEN MODULS	15
<b>6 DIE AUSWERTUNG</b>	<b>15</b>
6.1 BESTIMMUNG DES FEHLERS UND DER KONVERGENZORDNUNG	15
6.2 ERLÄUTERUNG DER ERGEBNISDARSTELLUNG	16
6.3 PROBLEM „KAPS“	16
6.4 PROBLEM „TBA“	17
6.5 PROBLEM „ANDREWS“	17
6.6 PROBLEM „CARAXIS“	18
6.7 PROBLEM „WHEEL“	19
6.8 PROBLEM „CHEMAKZO“	20
6.9 PROBLEM „TRANSAMP“	21
6.10 PROBLEM „WATER“	22
6.11 PROBLEM „CRANK“	23
6.12 PROBLEM „NAND“	24





Der Vorteil dieser Darstellung ist, dass das Programmpaket diese Art von Differentialgleichungssystemen behandeln kann. Es ergibt sich allerdings auch ein einschneidendes Problem aus dieser Umformulierung: das entstehende System besitzt keinen Anfangswert für die erste Ableitung, da hierfür rein formal die zweite Ableitung von  $y$  am Intervallanfang bekannt sein müsste. Setzt man für  $y_1'' = \dots = y_n'' = 0$ , so erhält man in den meisten Fällen nur noch Konvergenzordnung 1; die Wahl dieser willkürlichen Werte ist also nicht angemessen.

Als Ausweg aus dieser Misere wird mit RADAU [ODE00] am Intervallanfang ein Integrationsschritt über das Intervall  $[t_0, t_0 + 10^{-5}]$  durchgeführt und eine Näherungslösung für  $(y \ z)^T$  und  $(y' \ z')^T$  am Intervallende  $t_0 + 10^{-5}$  bestimmt. Mit diesen „numerischen Startwerten“ ist das umgeformte System über dem Intervall  $[t_0 + 10^{-5}, t_1]$  lösbar. Der Summand  $10^{-5}$  ist ein Kompromiß zwischen einer kleinen Abweichung des neuen Intervallstarts vom ursprünglichen Startpunkt und der Möglichkeit einer möglichst genauen Approximation von  $z' = y''$ .

Von den künstlich eingeführten Variablen  $z_1, \dots, z_n$  sind keine Vergleichslösungen am Intervallende  $t_1$  bekannt, somit ist keine Bestimmung der Abweichung einer Näherungslösung vom umgewandelten System zu einer Referenzlösung in den Variablen  $z_i$  möglich. Somit ist auch keine gesonderte Betrachtung nötig, welche Indizes die Hilfsvariablen im neuen Differentialgleichungssystem besitzen.

**Bemerkung:** Auch Gleichungen vom Typ (DAE) lassen sich in ähnlicher Weise umformulieren. Man erhält dabei aus

$$\mathbf{M} \mathbf{y}' = \mathbf{f}(\mathbf{t}, \mathbf{y}), \quad \mathbf{t}_0 \leq \mathbf{t} \leq \mathbf{t}_1$$

durch analoge Substitution das System

$$\begin{aligned} \mathbf{y}' &= \mathbf{z} \\ \mathbf{0} &= \mathbf{M} \mathbf{z} - \mathbf{f}(\mathbf{t}, \mathbf{y}), \quad \mathbf{t}_0 \leq \mathbf{t} \leq \mathbf{t}_1. \end{aligned}$$

Diese Umwandlung wird im folgenden als „implizite Behandlung“ einer DAE bezeichnet. Für ODE-Probleme ist dabei  $\mathbf{M}$  die Einheitsmatrix.

## 4 DIE VERFAHREN

Zur Lösung der einzelnen Differentialgleichungen werden linear-implizite Zweischrittmethoden mit  $s$  Stufen [PSW99] benutzt. Für das Anfangswertproblem (ODE) wird folgender Ansatz beim Zeitschritt  $t_m \rightarrow t_{m+1} = t_m + h_m$  benutzt:

$$Y_{mi} = u_m + h_m \sum_{j=1}^s a_{ij} k_{m-1,j}$$

$$(\mathbf{I} - h_m \gamma_i \mathbf{T}_m) k_{mi} = \mathbf{f}(t_m + c_i h_m, Y_{mi}) + h_m \mathbf{T}_m \sum_{j=1}^s \gamma_{ij} k_{m-1,j} \quad \text{für } i = 1, \dots, s$$

$$u_{m+1} = u_m + h_m \sum_{i=1}^s (b_i k_{mi} + v_i k_{m-1,i})$$

Wichtig ist die Wahl der Matrix  $\mathbf{T}_m$ . Um die besten Ergebnisse bezüglich Stabilität zu erhalten, ist  $\mathbf{T}_m = \mathbf{f}'(t_m, u_m)$  zu setzen. Um die Güte eines Verfahrens zu beschreiben, lassen sich die folgenden drei Kriterien benutzen. Der Parameter  $\sigma_m = h_m / h_{m-1}$  beschreibt dabei das Verhältnis von aufeinanderfolgenden Schrittweiten. Da hier keine Schrittweitensteuerung eingesetzt wird, besitzt dieses Verhältnis also stets einen konstanten Wert  $\sigma = 1$ .

$$B(p) : \sum_{j=1}^s b_j \sigma^{l-1} c_j^{l-1} + \sum_{j=1}^s v_j (c_j - 1)^{l-1} = \frac{\sigma^{l-1}}{1}, \quad l = 1, \dots, p$$

$$C(q) : \sum_{j=1}^s a_{ij} (c_j - 1)^{l-1} = \sigma^{l-1} \frac{c_i^l}{1}, \quad l = 1, \dots, q$$

$$\Gamma(q) : \sum_{j=1}^s \gamma_{ij} (c_j - 1)^{l-1} = -\gamma_i \sigma^{l-1} c_i^{l-1}, \quad l = 1, \dots, q$$

In [PSW99] werden für verschiedene Stufen  $s$  die Koeffizienten  $(a_{ij})$ ,  $(\gamma_{ij})$ ,  $(\gamma_i)$  und die Vektoren  $(b_i)$ ,  $(v_i)$  und  $(c_i)$  hergeleitet, so dass die entstehenden Verfahren möglichst hohe Konvergenzordnungen besitzen. Die folgende Übersicht zeigt die hier verwendeten Verfahren im Überblick.

- VNIL2:** Verfahren der Konvergenzordnung  $p = 2$  mit  $s = 2$  Stufen, L-stabil ( $\alpha = 90^\circ$ ), erfüllt die Bedingungen  $B(2)$ ,  $C(2)$  und  $\Gamma(2)$ , für die Parameter gilt  $\gamma = 0.6339$ ,  $c_1 = -0.4641$  und  $c_2 = 1$ .
- VNIL3:** Verfahren der Konvergenzordnung  $p = 3$  mit  $s = 3$  Stufen, L( $\alpha$ )-stabil ( $\alpha = 84.8^\circ$ ), erfüllt die Bedingungen  $B(3)$ ,  $C(3)$  und  $\Gamma(3)$ , mit Parameter  $\gamma = 0.5155$ ,  $c_1 = -2.0254$  und  $c_2 = -0.3347$ ,  $c_3 = 1$ .
- VSUPER2:** Verfahren der Konvergenzordnung  $p = 3$  (für feste Schrittweite  $h$ ) mit  $s = 2$  Stufen, L-stabil ( $\alpha = 90^\circ$ ), erfüllt die Bedingungen  $B(3)$ ,  $C(2)$  und  $\Gamma(2)$ , mit Parameter  $\gamma = 1.2$ ,  $c_1 = 2,2381$  und  $c_2 = 1$ .
- VSUPER3:** Verfahren der Konvergenzordnung  $p = 4$  (für feste Schrittweite  $h$ ) mit  $s = 3$  Stufen, L( $\alpha$ )-stabil ( $\alpha = 88.4^\circ$ ), erfüllt die Bedingungen  $B(4)$ ,  $C(3)$  und  $\Gamma(3)$ ,  $\gamma = 0.85$ ,  $c_1 = 1$  und  $c_2 = 2.3425$ ,  $c_3 = 1$ .

## 5 DIE PROGRAMMERWEITERUNGEN

### 5.1 Aufspaltung nach einzelnen Indizes

Das folgende Programmlisting zeigt die Veränderungen, die an der Datei „driver.f“ vorgenommen wurden. Schwerpunkte waren dabei die nach Indizes getrennten Ordnungs- und Fehlerberechnungen, sowie die Unterstützung der impliziten Behandlung von Differentialgleichungen. Da die Dimension des Systems durch die Einführung von Scheinvariablen  $z_k = y_k$  verdoppelt wird und am Intervallende keine Referenzlösung für die erste Ableitung gegeben ist, ist in der Routine *lsg\_from\_file* anhand der Datei, die die Referenzlösung enthält, zu unterscheiden, ob implizit gerechnet wird oder nicht. Dies ist jedoch der einzige gravierende Eingriff in das strenge Modulschema vom Programmpaket „ODETEST“.

Auszug aus „driver.f“	
<pre> subroutine odetest(fdgl, mas, jacv, pfac, psol, fdt, +               loes, rpar, ipar) [...]   real*8 errgem_index2, errlog_index2, eord_index2,          lerr_index2   real*8 errgem_index3, errlog_index3, eord_index3,          lerr_index3   character*80 col8_index2, col8_index3   logical ist_implicit   integer n_schleifenende    col8_index2 = ""   col8_index3 = ""   lerr_index2 = 1d0   lerr_index3 = 1d0 [...] c Lesen der Vergleichslösung nach u2, Flag setzen call lsg_from_file(name, n, u2, rpar, ipar, +               ist_implicit)  c Ersetze Lösungsvektor u durch u2 - u, c u ist somit jetzt Differenz von Lösung zu Referenzlsg. call daxpy(n, -1d0, u2, 1, u, 1)  c Ordnungsschätzung für Index-1-Variablen errgem = 0d0 do i = 1, nind1   errgem = errgem + (u(i) / (1 + dabs(u2(i))))**2 end do errgem = dsqrt(errgem) errlog = dlog10(errgem) eord = (lerr - errlog) / log10((naccpt / lschritte)) lerr = errlog </pre>	<p>Die Routine <i>odetest</i> stellt das Hauptprogramm der benutzten Testumgebung dar.</p> <p>Für die getrennte Behandlung der Index-2 und Index-3-Unbekannten werden zusätzliche Variablen im Quellcode benötigt, deren Nomenklatur sehr einfach ist.</p> <p>Beim Aufruf der Routine <i>lsg_from_file</i> wird jetzt ein zusätzlicher Parameter per Referenz übergeben, der speichert, ob implizit gerechnet wird.</p> <p>Die Bestimmung der numerischen Konvergenzordnung erfolgt für alle Unbekannten getrennt. Die Variablen <i>nind1</i> und <i>nind2</i> stehen dabei für die Anzahl für die Index-1- und Index-2-Variablen.</p>

```

c "_index2": Ordnungsschätzung für Index-2-Variablen
errgem_index2 = 0d0
do i = nind1 + 1, nind1 + nind2
  errgem_index2 = errgem_index2
    + (u(i) / (1 + dabs(u2(i))))**2
end do
errgem_index2 = dsqrt(errgem_index2)
errlog_index2 = dlog10(errgem_index2)
eord_index2 = (lerr_index2 - errlog_index2)
  / log10(naccpt / lschritte)
lerr_index2 = errlog_index2

c "_index3": Ordnungsschätzung für Index-3-Variablen
c Achtung: Der Vektors u besitzt Groesse n, aber im
c impliziten Fall werden die z-Variablen ignoriert.

if (ist_implicit .eqv. .true.) then
  n_schleifenende = n / 2
else
  n_schleifenende = n
end if

errgem_index3 = 0d0
do i = nind1 + nind2 + 1, n_schleifenende
  errgem_index3 = errgem_index3
    + (u(i) / (1 + dabs(u2(i))))**2
end do
errgem_index3 = dsqrt(errgem_index3)
errlog_index3 = dlog10(errgem_index3)
eord_index3 = (lerr_index3 - errlog_index3)
  / log10(naccpt / lschritte)
lerr_index3 = errlog_index3

c Jetzt erfolgt die Ausgabe der
c Fehler : errlog, errlog_index2, errlog_index3
c Ordnungen: eord, eord_index2, eord_index3
[... ]
return
end

subroutine lsg_from_file(name,n,u,rpar,ipar,ist_implicit)
  implicit none
  character*(*) name
  character*80 fn
  real*8 u(*), rpar(*)
  integer ipar(*), i, n, strlen
  logical ist_implicit

c Hier muss unterschieden werden: Wird eine implizite
c Datei verwendet (für interface_impl.f), so steht in der
c exakten Datei nur die Referenzlösung für die
c Funktionswerte und nicht für die Ableitungswerte.
c D.h. in diesem Fall darf nur die halbe Datei gelesen
c werden (n muss halbiert werden).

  integer n_lesen
  character*80 erste_zeile

  write (fn,99) "problem/exakt/", name(1:strlen(name))
99 format (a14,a)

```

Da die zusätzlich eingeführten Scheinvariablen  $z_k$  in der Routine *problem* in der Datei „interface\_impl.f“ rein formal den Index 3 zugeordnet bekommen, kann die Anzahl der wahren Index-3-Variablen nur über Differenzbildung zwischen Gesamtanzahl der eigentlichen Unbekannten und Summe der Anzahl der Variablen mit niedrigeren Index bestimmt werden. *n\_schleifenende* wird stets auf die Anzahl der ursprünglichen Unbekannten gesetzt, bei impliziter Behandlung muss die Dimension  $n$  des modifizierten Systems halbiert werden.

In der Routine *lsg\_from\_file* wird die Referenzlösung aus einer Datei gelesen. Steht dabei in der ersten Zeile die Markierung *NUR\_HALB\_LESEN*, so handelt es sich um ein Problem, welches durch implizite Rechnung gelöst werden soll.

<pre> open (unit=3,file=fn ,status='old') read (3,*) erste_zeile  if (erste_zeile .eq. "NUR_HALB_LESEN") then   ist_implicit = .true.   n_lesen = n / 2 else   ist_implicit = .false.   n_lesen = n end if  read (3,*) (u(i), i = 1, n_lesen) close (3)  return end </pre>	<p>Im Vektor <math>u</math> werden die Funktionswerte der Referenzlösung am Integrationsende gespeichert. Er wird später zur Fehlerberechnung herangezogen.</p>
--	---

## 5.2 Programmtechnische Umsetzung der Umformulierung

Es wurde die im folgenden abgedruckte Datei „interface\_impl.f“ erstellt, die analog zur Datei „interface\_dae.f“ in die implizit zu rechnenden Problembeschreibungsdateien eingebunden wird.

Auszug aus „water.f“	
<pre>include "interface_impl.f"</pre>	<p>Durch Einbindung dieses Moduls wird festgelegt, dass die Lösung der Differentialgleichung durch Umformulierung stattfinden soll.</p>

Das folgende Listing zeigt den Inhalt der Modul-Datei. Sie enthält die fünf Routinen *me\_back*, *fdgl*, *problem*, *antwort* und *mas*, die im globalen Hauptprogramm bei der Anwendung der einzelnen Lösungsalgorithmen aufgerufen werden. Diese Prozeduren sind wie die Algorithmen selbst generisch konstruiert, damit sind sie für alle zu behandelnden Arten von Differentialgleichungen einsetzbar. Diese Allgemeinheit wird dadurch erreicht, dass die in der Problembeschreibungsdatei der Differentialgleichung enthaltenen Routinen an den richtigen Programmstellen aufgerufen werden.

Der Aufbau des Moduls „interface\_impl.f“ gehorcht somit den allgemeinen Modulregeln. Bei deren Implementierung war die Fragestellung zu lösen, welche Indizes den neu eingeführten Scheinvariablen  $z_1, \dots, z_n$  zuzuordnen ist. Diese entsprechen analytisch der ersten Ableitung der Systemvariablen  $y_1, \dots, y_n$ , sollen aber in der Auswertung des Näherungsergebnisses keine Rolle spielen. (Es sind keine Ableitungswerte der Referenzlösung am Intervallende bekannt, somit ist keine Fehlerbestimmung möglich.) Um eine Berechnung der Fehlerwerte der Scheinvariablen zu verhindern, wurde diesen Variablen rein formal der Index 3 zugeordnet. Es soll hier ausdrücklich darauf hingewiesen werden, dass dies keine analytische Ungenauigkeit, sondern vielmehr eine programmtechnische Auslegung der Modulspezifikationen darstellt. Weiterhin soll bemerkt werden, dass der Index der Variablen nur bei der Schrittweitensteuerung eine entscheidende Rolle spielt, diese wird hier aber nicht untersucht. Die einzige Aufgabe ist hier also die Gruppierung der Variablen nach Indizes (ohne Betrachtung der Scheinvariablen) zur getrennten Fehler- und Ordnungsberechnung.

Inhalt von „interface_impl.f“	
<pre> subroutine me_back(odetest)   implicit none   external odetest, dummy, fdgl, loes_radau_dae, mas   real*8 rpar(20)   integer ipar(20)    call odetest(fdgl, mas, dummy, dummy, dummy, dummy, +             loes_radau_dae, rpar, ipar)   return end </pre>	<p><i>me_back</i> stellt die Callback-Routine dar, die vom globalen Hauptprogramm zur Lösung der Differentialgleichung aufgerufen wird. Sie verweist auf die weiter unten definierten Unterprogramme <i>fdgl</i> und <i>mas</i>.</p>

<pre> subroutine fdgl(n, t, u, fw, rpar, ipar)   implicit none   integer      n, ipar(*), ierr   real*8      t, rpar(*), u(*), fw(*)  c Variablen zur Bestimmung des Typs der DGL (ODE,IDE,DAE) character*80  temp_str integer      temp_int, temp_int2(n) real*8      temp_real(n) logical      temp_log character*80  type integer      mlas, mumas  c Temporäre Variablen zum Aufruf von mas im Fall DAE integer lmas  c Für am würde eigentlich (mlmas + mumas + 1, n/2) c ausreichen, aber die ersten beiden Variablen werden c unten erst bestimmt. double precision am(n / 2, n / 2) double precision x, y(n / 2), dy(n / 2)  c Variablen für Matrix-Vektor-Produkts im Fall DAE double precision MATRIX(n / 2, n / 2), VEKTOR(n / 2) integer j  c Momentan folgende Wertbelegung: c   n .. 2 * n_intern c   u .. Vektor der Größe n, bestehend aus c       y(1), ..., y(n_intern), z(1),..., z(n_intern) c Die Funktion feval benötigt die Vektoren: c   3. Parameter: (y(1) , ..., y(n_intern)) c   4. Parameter: (z(1) , ..., z(n_intern)) c               = (y'(1), ..., y'(n_intern)) c und gibt folgenden Vektor als 5. Parameter zurück: c   (f(t,y1(t)),f(t,y2(t)), ..., f(t,y[n_intern](t)))  c Anlegen von Variablen, die Werte zunächst aufnehmen real*8 fw_intern(5000), yprime_intern(5000) integer i, n_intern n_intern = n / 2  do i = 1, n_intern   yprime_intern(i) = u(n_intern + i) end do  call feval(n_intern, t, u, yprime_intern, fw_intern, +         ierr, rpar, ipar)  c Hier erfolgt Fallunterscheidung nach dem Typ der DGL. c Fälle:f(t,y,y')=0(IDE), y'=f(t,y)(ODE), My'=f(t,y)(DAE)  c Zur Typbestimmung prob mit Dummy-Variablen aufrufen  call prob(temp_str, temp_str, type, temp_int, temp_int, +        temp_real, temp_log, temp_int, temp_int, +        temp_log, mlas, mumas, temp_int2) c Code für y'=f(t,y) (ODE) if (type.eq.'ODE') then   do i = 1, n_intern     fw(i) = u(n_intern + i)     fw(n_intern + i) = u(n_intern + i) - fw_intern(i)   end do end if </pre>	<p><i>fdgl</i> stellt die rechte Seite der Differentialgleichung dar: <math>n</math> ist Dimension, <math>t</math> der Zeitparameter, <math>u</math> ist ein <math>n</math>-dimensionaler Vektor mit den Variablenwerten und <math>fw</math> ist ein <math>n</math>-dimensionaler Vektor, der das Ergebnis der Anwendung der rechten Seite der DGL aufnehmen soll.</p> <p><math>n\_intern</math> ist die eigentliche Dimension des gegebenen Problems. Die Vektoren besitzen hier alle die doppelte Dimension <math>n</math>, da jeder Variable <math>y_i</math> eine Scheinvariable <math>z_i = y_i'</math> zugeordnet wird.</p> <p><math>fw\_intern</math> ist der interne Funktionswert des eigentlichen Systems, <math>yprime\_intern</math> ist der interne Vektor der Ableitungswerte.</p> <p>Durch <i>feval</i> wird die rechte Seite des eigentlichen DGL-Systems ausgewertet.</p> <p><i>feval</i> und <i>prob</i> sind Routinen, die die Beschreibungsdatei des DGL-Problems bereitstellen muss.</p> <p>Im Falle, dass eine gewöhnliche Differentialgleichung implizit behandelt werden soll, ist <math>y' = z</math>  <math>0 = z - f(t, y)</math>  zu lösen.</p>
---	---

```

c Code für  $f(t,y,y') = 0$  (IDE):
c Ergebnisvektor: (z(1), ..., z(n_intern),
c               fw_intern(1),..., fw_intern(n_intern))
c   if (type.eq.'IDE') then
c     do i = 1, n_intern
c       fw(i) = u(n_intern + i)
c       fw(n_intern + i) = fw_intern(i)
c     end do
c   end if

c Code für  $My'=f(t,y)$  (DAE):
c   if (type.eq.'DAE') then
c     Zuerst Matrix-Vektor-Multiplikation: Massematrix mal
c      $y' = u(n\_intern + [1..n\_intern])$ 
c     (1) Zuerst Original-Masse-Matrix besorgen.
c     ierr = 0

c     if (mlmas .eq. n_intern) then
c       lmas = n_intern
c     else
c       lmas = mlmas + mumas + 1
c     end if

c     Für am würde eigentlich (lmas, n_intern) ausreichen,
c     aber hier MÜSSEN die Ausmaße von am angegeben werden,
c     die oben bei der Deklaration verwendet wurden.
c     call meval(n_intern, n_intern, x, y, dy, am, ierr,
+             rpar, ipar)
c     if (ierr.ne.0) then
c       write (*,*) "meval liefert Fehlercode ungleich 0!"
c       stop
c     end if

c     (2) Matrix auf Null setzen.
c     do i = 1, n_intern
c       do j = 1, n_intern
c         MATRIX(i, j) = 0
c       end do
c     end do

c     (3) Aus Bandmatrix volle Matrix erzeugen.
c     if (mlmas .eq. n_intern) then
c       do i = 1, n_intern
c         do j = 1, n_intern
c           MATRIX(i, j) = am(i, j)
c         end do
c       end do
c     else
c       do i = 1, mlmas + mumas + 1
c         do j = 1, n_intern
c           if ((i + j) .gt. (mumas + 1)) then
c             if ((i + j) .le. (mumas + 1 + n_intern)) then
c               MATRIX(i + j - mumas - 1, j) = am(i, j)
c             end if
c           end if
c         end do
c       end do
c     end if
c   end if

```

Im Falle, dass eine implizite DGL zu lösen ist, wird

$$y' = z$$

$$0 = f(t, y, z)$$

geschrieben:

Im Falle, dass eine differential-algebraische Gleichung zu lösen ist, wird

$$y' = z$$

$$0 = Mz - f(t, y)$$

betrachtet. Dazu wird hier zunächst die Matrix-Vektor-Multiplikation vorgenommen.

```

c  (4) VEKTOR = MATRIX * y'
c          = MATRIX * u(n_intern + [1..n_intern])
  do i = 1, n_intern
    VEKTOR(i) = 0
    do j = 1, n_intern
      VEKTOR(i) = VEKTOR(i)+(MATRIX(i,j)*u(n_intern+j))
    end do
  end do

c  Jetzt:fw(1..n_intern) = u(n_intern + 1 .. n_intern*2)
c          fw(n_intern + 1 .. n_intern * 2)
c          = VEKTOR(1..n_intern) - fw_intern(1..n_intern)

  do i = 1, n_intern
    fw(i)          = u(n_intern + i)
    fw(n_intern + i) = VEKTOR(i) - fw_intern(i)
  end do
end if
end

subroutine problem(name, n, t, te, mljac, mujac, imas,
+               mlmas, mumas, nind1, nind2, ifcn, rpar, ipar)
  implicit none
  character*(*) name
  character*80 fullnm, type
  integer n, mljac, mujac, imas, mlmas, mumas, ipar(*), i
  integer nind1, nind2, nx, ifcn
  real*8 t, te, rpar(*)
  parameter (nx=5000)
  integer ind(nx)
  logical numjac, nummas

  integer ndisc
  real*8 tt(100)
  integer n_intern

  call prob(fullnm, name, type, n_intern, ndisc, tt,
+          numjac, mljac, mujac, nummas, mlmas, mumas,
+          ind)

  t = tt(1)
  te = tt(ndisc + 2)

c Hier wird n noch nicht verdoppelt,
c z-Variablen haben zunächst alle Index 3 (formal).
c nind1 und nind2 greifen nur bei Schrittweitensteuerung
c in die Berechnung der Lösung ein.

  nind1 = n_intern
  nind2 = 0

  do i = 1, n_intern
    if (ind(i).le.1) nind1 = i
    if (ind(i).eq.2) nind2 = i - nind1
  end do

c Dimension n des Systems verdoppeln
  n = n_intern + n_intern

c Jetzt Bandmatrix, nur Diagonalmatrix.
  mlmas = 0
  mumas = 0

```

Die Routine *problem* liefert allgemeine Informationen über das Problem. Im wesentlichen werden hier die Parameter, die die von der DGL bereitgestellte Routine *prob* zurückgegeben werden, übernommen. Änderungen der internen Parameter gibt es nur in der Verdopplung der Dimension durch Einführung neuer Scheinvariablen. Weiterhin werden wichtige Konstanten gesetzt, die unter anderem dafür sorgen, dass eine Massmatrix angegeben wird, die als Bandmatrix gespeichert wird.

*mlmas* und *mumas* bestimmen die Anzahl der besetzten Nebendiagonalen unter und über der Hauptdiagonalen der Massmatrix *M*, falls die Matrix als Bandmatrix gespeichert wird. Hier werden beide Parameter auf Null gesetzt, da die neue Matrix nur Diagonalgestalt besitzt.

```

mljac = n
mujac = n
c Wir sind hier im impliziten Fall und bieten eine
c Masseurmatrix an, somit wird das Flag imas gesetzt.
  imas = 1
  ifcn=1
  return
end

subroutine anwert (n, t, u, uprime, rpar, ipar, nind1,
+                 nind2)
  implicit none
  integer n, ipar(*)
  real*8 u(*), rpar(*), t, uprime(*)
  integer nind1, nind2
  logical consis
  external loes_radau_dae, fdgl, mas

c Lokale Variablen anlegen, die an Funktion init des
c Testsets weitergegeben werden.
  integer n_intern, i
  real*8 u_intern(1000), uprime_intern(1000)
  n_intern = n / 2
  call init(n_intern, t, u_intern, uprime_intern, consis)

c Der neue Vektor u besitzt die Groesse 2 * n_intern, die
c ersten n_intern Komponenten bestehen aus
c u_intern(1)..u_intern(n_intern), die letzten
c n_intern Komponenten bestehen aus
c uprime_intern(1)..uprime_intern(n_intern).
  do i = 1, n_intern
    u(i) = u_intern(i)
    u(n_intern + i) = uprime_intern(i)
  end do

c Die Belegung von uprime ist (y'(0),y''(0)),
c uprime besitzt Größe 2 * n_intern = n.
c Hier wird die Ableitung trickreich berechnet.
c (Benutzung der letzten Stufe von radau,
c Achtung: Das Zeitintervall wird verkürzt.)
c Der folgende Aufruf setzt uprime; und t auf t + 1D-5.
c Für Probleme mit sehr kleinem Integrationsintervall
c wird hier t + 1D-12 gewählt.
  call loes_radau_dae(n, fdgl, t, u, t + 1D-5, uprime,
+                 1D-10, 1D-10, 0, n, n, mas, 1,
+                 0, 0, nind1, nind2, rpar, ipar)
  return
end

subroutine mas(n, am, lmas, rpar, ipar)
  integer n, lmas, ipar(*)
  double precision am(lmas, n), rpar(*)
c Ziel ist folgende Gestalt: am = (I, 0)
  integer i, n_intern
  n_intern = n / 2
c Erzeugung einer Diagonalmatrix, am(1, i) = m(i, i)
  do i = 1, n_intern
    am(1, i) = 1
    am(1, n_intern + i) = 0
  end do
  return
end

```

*mljac* und *mujac* sind die analogen Parameter für die Jacobi-Matrix. Hier werden beide Parameter auf *n* gesetzt, somit wird mit einer vollbesetzten Matrix gerechnet.

Die Routine *anwert* soll die Anfangswerte der Variablen am Integrationsanfang bereitstellen. Hier erfolgt ein Aufruf der Hilfsroutine *loes\_radau\_dae*, um den Wert von *y*, *z*, *y'* und *z' = y''* an einer Nebenstelle ( $t+10^{-5}$ ) des eigentlichen Startpunktes zu bestimmen. Somit ist es möglich, ein neues DGL-Problem mit kleinerem Integrationsintervall anzugeben, für welches am Intervallanfang auch die Ableitungswerte der Scheinvariablen *z* bekannt sind.

Die Routine *mas* gibt die Masseurmatrix zurück. Da das Problem transformiert wird, wird die eigentliche Masseurmatrix des Problems an dieser Stelle nicht benötigt (diese fließt jetzt in die Berechnung der rechten Seite der DGL ein), es erfolgt also kein Aufruf der Funktion *meval*.

## 5.3 Überprüfung der Funktionsfähigkeit des neuen Moduls

Untersucht wird das Lösungsverhalten des nachfolgenden Differentialgleichungssystems über dem Integrationsintervall  $[0, 1]$ .

$$\begin{aligned} y_1' &= -3y_1 + y_2^2 \\ y_2' &= y_1 - y_2 - y_2^2 \\ y_1(0) &= y_2(0) = 1, y_1'(0) = -2, y_2'(0) = -1 \end{aligned}$$

Durch die Transformation  $z = y'$  erhält man folgendes System doppelter Dimension.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_1' \\ y_2' \\ z_1' \\ z_2' \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ 3y_1 - y_2^2 + z_1 \\ -y_1 + y_2 + y_2^2 + z_2 \end{pmatrix}$$

Durch Einsetzen wird leicht bestätigt, dass die exakte Lösung wie folgt lautet:

$$\begin{aligned} y_1 &= e^{-2t} & \Rightarrow y_1' &= -2e^{-2t} \\ y_2 &= e^{-t} & \Rightarrow y_2' &= -e^{-t} \\ z_1 &= -2e^{-2t} & \Rightarrow z_1' &= 4e^{-2t} \\ z_2 &= -e^{-t} & \Rightarrow z_2' &= e^{-t} \end{aligned}$$

Somit hat man als Vergleichslösung am Intervallende folgende Werte vorzugeben:  $y_1(1) = e^{-2}$ ,  $y_2(1) = e^{-1}$ . Die Ergebnisse der numerischen Behandlung dieser im folgenden als „Problem Kaps“ [KPB85] bezeichneten Aufgabe erfolgt im Abschnitt 6.3. Dort wird gezeigt, dass die Ergebnisse der Berechnungen mit und ohne Einführung von neuen Variablen bis auf bei kleinen Schrittweiten auftretende Rundungsfehler identisch sind. Somit kann davon ausgegangen werden, dass die Implementierung der Dimensionsverdopplung im Modul „interface\_impl.f“ korrekt ist.

## 6 DIE AUSWERTUNG

### 6.1 Bestimmung des Fehlers und der Konvergenzordnung

Für jede Differentialgleichung ist die exakte Lösung (bzw. eine sehr gute numerische Näherung an diese) am Intervallende bekannt. Somit kann durch Differenzbildung der Fehler der numerischen Lösung bestimmt werden. Wie im obigen Auszug aus „driver.f“ zu sehen ist, bestimmt sich der ausgegebene Fehler nach folgender Formel, wobei  $u$  die vom jeweiligen Algorithmus berechnete Lösung und  $y$  die aus der Datei gelesene Referenzlösung am Integrationsende bezeichnen soll (im Programm ist  $y = u_2$ ).

$$\text{Fehler} = \sqrt{\sum_{i=1}^n \left( \frac{u_i(t_{\text{end}}) - y_i(t_{\text{end}})}{1 + |y_i(t_{\text{end}})|} \right)^2}$$

Diese Formel wurde insofern im Programm so modifiziert, dass die Summation nicht über alle Einträge in den Vektoren vorgenommen wird, sondern gesplittet nach Index-1-, Index-2- und Index-3-Variablen.

Werden zwei Berechnungen mit dem gleichen Verfahren mit unterschiedlichen Schrittzahlen  $SZ_1$  und  $SZ_2$  durchgeführt, so bestimmt die Schätzung der numerischen Konvergenzordnung durch folgende Formel.

$$\text{Konvergenzordnung} = \frac{\log_{10}(\text{Fehler bei } SZ_1) - \log_{10}(\text{Fehler bei } SZ_2)}{\log_{10}\left(\frac{\text{Akzeptierte Schritte bei } SZ_2}{\text{Akzeptierte Schritte bei } SZ_1}\right)}$$

Somit bedeutet eine positive Konvergenzordnung stets eine Erhöhung der Genauigkeit bei  $SZ_2$  gegenüber  $SZ_1$ , und eine negative Konvergenzordnung deutet stets auf eine Verschlechterung bei Übergang von  $SZ_1$  zu  $SZ_2$ . Bei der Aufspaltung nach einzelnen Indizes sind die Komponenten im Zähler des Bruches die Einzelfehler der verschiedenen Indizes.

## 6.2 Erläuterung der Ergebnisdarstellung

Die betrachteten Probleme besitzen Unbekannte bis zum Index 3. Die nachstehende Tabelle zeigt beispielhaft, in welcher Form die numerischen Fehler- und Ordnungswerte nach Index-1-, Index-2- und Index-3-Variablen getrennt aufgeschlüsselt werden. Dabei gilt für alle Tabellen, dass stets der dekadische Logarithmus des Fehlers betrachtet wird. Die angegebenen Ordnungszahlen in einer Zeile beziehen sich immer auf die Schrittzahl der aktuellen und der direkt vorangehenden Zeile.

Schritte	Logarithmischer Fehler			Ordnungsschätzung		
	Index 1	Index 2	Index 3	Index 1	Index 2	Index 3
6400	-3.96	-1.31	-1.24	2.9	2.9	2.9
9050	-4.40	-1.75	-1.68	3.0	3.0	2.9

In jedem Testszenario wurde versucht, die erste Schrittzahl 100 zu wählen. Damit auch für diese Schrittzahl eine Ordnungsschätzung möglich war, wurde davor eine Hilfsrechnung mit 93 Schritten durchgeführt. In vielen Fällen zeigten sich für geringe Schrittzahlen noch keine auswertbaren Ergebnisse, somit beginnen viele Tabellen erst mit einer höheren Schrittzahl.

Es wurde versucht, die Schrittzahl stets durch Multiplikation mit  $\sqrt{2}$  zu erhöhen. Damit ganze Zahlen entstehen, wurde auf ein Vielfaches von Zehn gerundet. Somit entstand folgende Folge:

(93,) 100, 140, 200, 280, 400, 560, 800, 1.130, 1.600, 2.260, 3.200, 4.520, ...

Für die Probleme „Water“ und „Crank“ erfolgte für ausgewählte Intervalle eine Verfeinerung dieser groben Schrittweitenänderung. In diesen Intervallen wurde als Quotient aufeinanderfolgender Schrittzahlen nicht versucht,  $2^{1/2}$  anzunähern, sondern beispielsweise  $2^{1/8}$ . Somit bilden die einzelnen Schrittzahlen annähernd eine geometrische Zahlenfolge.

## 6.3 Problem „Kaps“

### Beschreibung des Problems

Die Aufgabenstellung dieses Differentialgleichungssystems ist schon in Abschnitt 5.3 erfolgt. Es ist ein ODE mit zwei Gleichungen, die beide den Index 1 besitzen. Das System besitzt die Gestalt

$$dy/dt = f(y), y(0) = y_0, y'(0) = y_0',$$

wobei  $0 \leq t \leq 1$ ,  $y \in \mathbb{R}^2$ ,  $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ .

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde sowohl mit dem herkömmlichen Modul „interface\_dae.f“ (Massematrix = Einheitsmatrix) als auch mit dem Modul „interface\_impl.f“ gerechnet. Zielstellung ist der Vergleich der erreichten Fehler- und Ordnungswerte für verschiedene Schrittweiten. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt erhält man sehr gute Ergebnisse, so werden beispielsweise die theoretisch erwarteten Ordnungszahlen stabil realisiert.

Bei der impliziten Rechnung werden die Ableitungswerte im Punkt  $10^{-12}$  wie folgt berechnet:

$$y_1'(10^{-12}) = -2, y_2'(10^{-12}) = -1, z_1'(10^{-12}) = 3.99999996, z_2'(10^{-12}) = 0.999999989.$$

Bei der Benutzung von VNIL2 wird im Bereich von 100 – 102.400 Schritten die theoretisch erwartete Konvergenzordnung von 2,0 erreicht. Die erste Abweichung der IMPL-Fehlerwerte von den DAE-Fehlerwerten tritt bei 102.400 Schritten auf, dort wird dann aber schon eine Genauigkeit von  $10^{-10}$  erreicht.

Bei der Verwendung von VNIL3 wird im Bereich von 280 – 1.600 Schritten die theoretisch erwartete Konvergenzordnung von 3,0 stabil realisiert, für kleinere Schrittweiten erhöht sich sogar die numerische Konvergenzordnung. Insgesamt ist das „stabile“ Intervall für die Schrittzahl hier kürzer als bei VNIL2; dies liegt sicher daran, dass die Genauigkeit von  $10^{-10}$  schon für 1.600 Schritte erreicht wird. Die erste Abweichung der IMPL-Fehlerwerte von den DAE-Fehlerwerten tritt bei 6.400 Schritten auf.

Die Methode VSUPER2 erreicht im Bereich von 140 – 9.050 Schritten die theoretisch erwartete Konvergenzordnung von 3,0. Bei 9.050 Schritten ist schon eine Genauigkeit von  $10^{-11.85}$  erreicht. Die erste Abweichung der IMPL-Fehlerwerte von den DAE-Fehlerwerten tritt bei 12.800 Schritten auf.

Löst man das Differentialgleichungssystem mit VSUPER3, so wird im Bereich von 70 – 200 Schritten eine numerische Konvergenzordnung von 3,8 – 3,9 erreicht. Die erste Abweichung der IMPL-Fehlerwerte von den DAE-Fehlerwerten tritt bei 6.400 Schritten auf, dort ist aber schon eine Genauigkeit von  $10^{-12.2}$  erreicht.

Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Fehler  $10^{-13.43}$  bei Benutzung von VNIL3, Beibehaltung der Dimension und Durchführung von 18.100 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

## 6.4 Problem „TBA“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Two bit adding unit“. Es ist ein steifes DAE-System vom Index 1 mit 175 differentiellen und 175 algebraischen Gleichungen. Das Differentialgleichungssystem besitzt die Gestalt

$$\begin{aligned} dy/dt &= f(t, x) \\ 0 &= y - g(x) \end{aligned}$$

wobei  $0 \leq t \leq 320$ ,  $y, x \in \mathbb{R}^{175}$ ,  $f: \mathbb{R}^{351} \rightarrow \mathbb{R}^{350}$ ,  $g: \mathbb{R}^{350} \rightarrow \mathbb{R}^{350}$ ,  $y(0) = y_0$ ,  $x(0) = x_0$ . Alle Variablen besitzen den Index 1. Die erste Ableitung der Funktion  $f$  hat Unstetigkeitsstellen für  $t = 0, 5, 10, \dots, 320$ . Die Funktion  $f$  enthält mehrere Quadratwurzeln, somit kann sie für einige Parametersätze nicht ausgewertet werden.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem Standardalgorithmus (interface\_dae.f) gerechnet als DAE, somit wurde keine Umformulierung durchgeführt. Größtes Problem bei der Berechnung stellte die hohe Dimension 350 dar, somit dauerten die Berechnungen extrem lange. Aus diesem Grund wurden nur maximal 72.400 Schritte durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt ergeben sich sehr schlechte Ordnungswerte in allen Verfahren.

VNIL2 liefert sehr stark schwankende Konvergenzordnungen, die sogar negativ werden.

VNIL3 zeigt sehr ähnliches Verhalten, nur die Fehlerweite liegen meist etwas unter den Fehlerwerten von VNIL2.

VSUPER2 und VSUPER3 liefern deutlich stabilere Ordnungswerte, die allerdings weit unter den theoretisch ermittelten liegen. Weiterhin treten bei Verwendung dieser Verfahren deutlich größere Fehler auf, die Ergebnisse sind also ungenauer.

## 6.5 Problem „Andrews“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Andrews' squeezing mechanism“. Es ist eine nicht-steife DAE zweiter Ordnung mit Index 3, bestehend aus 21 differentiellen und 6 algebraischen Gleichungen. Das Differentialgleichungssystem besitzt die Gestalt

$$K dy/dt = \phi(y), y(0) = y_0, y'(0) = y_0',$$

wobei  $0 \leq t \leq 0.03$ ,  $y \in \mathbb{R}^{27}$ , konstante Matrix  $K \in \mathbb{R}^{27 \times 27}$ ,  $\phi: \mathbb{R}^{27} \rightarrow \mathbb{R}^{27}$ . Die ersten sieben Unbekannten im Vektor  $y$  besitzen Index 1, weitere sieben Index 2 und die restlichen 13 sind Index-3-Variablen.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem Standardalgorithmus (interface\_dae.f) gerechnet als DAE, somit wurde keine Umformulierung durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt erhält man sehr gute Ergebnisse, so beispielsweise stabile Ordnungszahlen in Variablen aller Indizes. Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-9.81}$  bei Benutzung von VSUPER2 und Durchführung von 144.810 Schritten

Index-2: Fehler  $10^{-5.75}$  bei Benutzung von VSUPER2 und Durchführung von 144.810 Schritten

Index-3: Fehler  $10^{-3.34}$  bei Benutzung von VNIL3 und Durchführung von 204.800 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdigramme der Ergebnisse der vier Verfahren.

#### Auswertung der Berechnung mit VNIL2

Selbst für eine geringe Anzahl von Schritten wird eine annehmbare Genauigkeit erreicht. Bestimmt wird eine stabile Konvergenzordnung von 2,1 - 2,6 im Bereich 200 – 72.400 Schritte für Index-1- und Index-2-Komponenten. Zunächst erhält man eine stabile Ordnung für Index-3-Variablen, für eine größere Schrittzahl ergibt sich jedoch eine kontinuierliche Abnahme der Konvergenzordnung.

#### Auswertung der Berechnung mit VNIL3

Hier ist eine deutliche Ähnlichkeit zu VNIL2 erkennbar. Von 800 – 4.520 Schritten wurde für alle Variablen die (theoretisch ermittelte) Konvergenzordnung von 3,0 erreicht. Bei höherer Schrittzahl bleibt diese hohe Konvergenzordnung in den ersten beiden Variablen noch bestehen, bricht jedoch in den Index-3-Variablen rapide zusammen, ab 25.600 Schritten erfolgt keine Verkleinerung des Fehlers in den Index-3-Variablen mehr.

#### Auswertung der Berechnung mit VSUPER2

Ähnliche Resultate wie VNIL2, aber der Bereich, in dem eine hohe Konvergenzordnung erreicht wird, ist deutlich kürzer für die Index-3-Variablen: ab einer Schrittzahl von 1.130 nimmt die Ordnung fast linear ab, bis sie bei einer Durchführung von 204.800 Schritten schließlich den Wert 0,1 erhält.

#### Auswertung der Berechnung mit VSUPER3

Erst bei einer Schrittzahl von 1.130 tritt hier Konvergenz auf. Im Bereich bis 12.800 Schritte erhält man sehr stabile Ordnungswerte, die für die Index-3-Komponente jedoch etwas unter der 3,0-Marke liegen. Im Gegensatz zu den Ergebnissen, die mit VSUPER2 erreicht wurden, nimmt die Konvergenzordnung für die Index-3-Komponente ab 18.100 Schritten rasant ab.

## 6.6 Problem „Caraxis“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „The car axis problem“. Es ist eine steife DAE mit Index 3, bestehend aus 8 differentiellen und 2 algebraischen Gleichungen. Das Differentialgleichungssystem besitzt die Gestalt

$$\begin{aligned} p' &= q \\ Kq' &= f(t, p, \lambda), \quad p, q \in \mathbb{R}^4, \lambda \in \mathbb{R}^2, 0 \leq t \leq 3 \\ 0 &= \phi(t, p) \end{aligned}$$

wobei  $p(0) = p_0$ ,  $q(0) = p'(0) = q_0$ ,  $\lambda(0) = \lambda_0$ ,  $\lambda'(0) = \lambda_0'$ , konstante Matrix  $K \in \mathbb{R}^{4 \times 4}$ ,  $f: \mathbb{R}^9 \rightarrow \mathbb{R}^4$ ,  $\phi: \mathbb{R}^5 \rightarrow \mathbb{R}^2$ . Die Indizes der Variablen  $p$ ,  $q$  und  $\lambda$  sind 1, 2 und 3.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem Standardalgorithmus (interface\_dae.f) gerechnet als DAE, somit wurde keine Umformulierung durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt ist eine relativ hohe Schrittzahl (ab ca. 12.800) nötig, um stabile Ordnungszahlen zu erhalten. Außer bei der Benutzung von VNIL2 werden bei allen Berechnungen nicht die gewünschten Ordnungszahlen in der dritten Komponente erreicht.

Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-10.4}$  bei Benutzung von VNIL3 und Durchführung von 289.630 Schritten

Index-2: Fehler  $10^{-8.39}$  bei Benutzung von VNIL3 und Durchführung von 204.800 Schritten

Index-3: Fehler  $10^{-8.15}$  bei Benutzung von VNIL2 und Durchführung von 204.800 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

### Auswertung der Berechnung mit VNIL2

Man erhält stabile Ordnungswerte von 1,9 - 2,0 ab einer Schrittzahl von 4.520 in allen drei Komponenten. Die Ordnung bricht für die Index-3-Variablen erst ab einer Durchführung von 102.400 oder mehr Schritten zusammen.

### Auswertung der Berechnung mit VNIL3

Hier ist nur eine geringe Ähnlichkeit zu VNIL2 erkennbar. Der Schrittbereich, in dem die gewünschten Ordnungszahlen erreicht werden, wird erst später erreicht (ab 12.800 Schritten), dann befinden sich allerdings die Ordnungszahlen für die Index-3-Komponente schon in der Nähe von 0. Insgesamt ist der Fehler der Näherungslösung zur Referenzlösung um jeweils ca. drei Zehnerpotenzen kleiner als bei Verwendung des Verfahrens VNIL2.

**Bemerkung:** Rechnet man dieses Problem implizit (durch Verwendung von „interface\_impl.f“ mit Bestimmung des Ableitungswertes an Stelle  $t_0 + 10^{-5}$ ), so ergeben sich bei Benutzung von VNIL3 im Bereich von 800 – 12.800 Schritten in allen Komponenten viel stabilere Ordnungswerte. Die Genauigkeit liegt etwas über den hier betrachteten Werten.

### Auswertung der Berechnung mit VSUPER2

Hierbei ist eine große Ähnlichkeit zu den Ergebnissen von VNIL2 erkennbar. Im Bereich von 2.260 - 72.400 Schritten treten stabile Ordnungswerte in den Index-2- und Index-3-Variablen auf. Dagegen erreicht man in der dritten Komponente nur eine stabile Ordnungszahl von 1,0 - 1,3. Die Berechnungsgenauigkeit ist generell höher als bei VNIL2.

### Auswertung der Berechnung mit VSUPER3

Lediglich in der ersten Komponente werden in dem kleinen Intervall von 6.400 – 51.200 Schritten die gewünschten Ordnungszahlen realisiert. Für die Index-2- und Index-3-Variablen ist die Existenz solcher Intervalle bei dem groben Schritt-Raster nicht erkennbar. Die erreichten Fehlerzahlen liegen über denen, die mit VSUPER2 berechnet wurden.

## 6.7 Problem „Wheel“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Wheelset“. Es ist eine IDE der Dimension 17 mit Index 2. Das Differentialgleichungssystem besitzt die Gestalt

$$\begin{aligned} \dot{p} &= v \\ M(p) \begin{pmatrix} \dot{v} \\ \dot{\beta} \end{pmatrix} &= \begin{pmatrix} f(u) - (\partial g_1(p, q) / \partial p)^T C \lambda \\ d(u) \end{pmatrix} \\ 0 &= (\partial g_1(p, q) / \partial p) v \\ 0 &= g_2(p, q) \end{aligned}$$

wobei  $u = (p, v, \beta, q, \lambda)^T \in \mathbb{R}^{17}$ ,  $p, v \in \mathbb{R}^5$ ,  $\beta \in \mathbb{R}$ ,  $q \in \mathbb{R}^4$ ,  $\lambda \in \mathbb{R}^2$ ,  $C \in \mathbb{R}$ ,  $C = \text{const}$ ,  $M: \mathbb{R}^5 \rightarrow \mathbb{R}^6 \times \mathbb{R}^6$ ,  $f: \mathbb{R}^{17} \rightarrow \mathbb{R}^5$ ,  $d: \mathbb{R}^{17} \rightarrow \mathbb{R}$ ,  $g_1: \mathbb{R}^9 \rightarrow \mathbb{R}^2$ ,  $g_2: \mathbb{R}^9 \rightarrow \mathbb{R}^4$ ,  $u(0) = u_0$ ,  $u'(0) = u_0'$ . Das Integrationsintervall ist  $t \in [0, 10]$ . Die Indizes der Variablen  $p, v, \beta, q$  und  $\lambda$  sind 1, 1, 1, 1 und 2.

## Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem modifizierten Algorithmus (interface\_impl.f) gerechnet, somit wurde durch Einführung neuer Variablen eine Umformulierung durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt ist der Fehler bei den Index-2-Variablen immer kleiner als der Fehler in den Index-1-Variablen, dieses Verhalten wurde bei den Beispielen vorher nicht betrachtet. Das stabilste Ergebnis wurde bei Benutzung der Methode VNIL2 erreicht, in den anderen Methoden sind die Schrittzahlen-Intervalle, in denen die Ordnungszahlen relativ konstant sind, extrem kleiner. Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-7.00}$  bei Benutzung von VNIL3 und Durchführung von 289.630 Schritten  
 Index-2: Fehler  $10^{-10.1}$  bei Benutzung von VNIL3 und Durchführung von 579.260 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdigramme der Ergebnisse der vier Verfahren.

### Auswertung der Berechnung mit VNIL2

Ab einer Schrittzahl von 72.400 tritt in beiden Komponenten der konstante Ordnungswert 2,0 auf; erst ab einer Durchführung von 819.200 Schritten fällt dieser (theoretisch erwartete) Wert langsam. Im Gegensatz zu den meisten anderen Beispielen fällt die Ordnung für die Index-2-Komponente langsamer als für die Index-1-Komponente.

### Auswertung der Berechnung mit VNIL3

Hier ist keine Ähnlichkeit zu VNIL2 erkennbar, die Ordnungszahlen schwanken in beiden Komponenten sehr stark. Die theoretische Ordnung von 3,0 wird nur im kleinen Bereich von 12.500 – 18.100 Schritten näherungsweise erreicht.

### Auswertung der Berechnung mit VSUPER2

Hier ist eine gewisse Ähnlichkeit zu VNIL2 vorhanden. Der Bereich, in dem die Ordnungswerte ungefähr konstant sind (18.100 – 144.810), ist deutlich kleiner als der Bereich bei Anwendung von VNIL2. Trotzdem werden in diesem Bereich Werte von 2,7 – 3,1 für die Konvergenzordnung erreicht. Der Fehler ist bei Benutzung von VSUPER2 in der Index-1-Komponente etwa um eine, in der Index-2-Komponente etwa um zwei Zehnerpotenzen kleiner.

### Auswertung der Berechnung mit VSUPER3

Hierbei treten sehr schwankende Ordnungswerte auf. Der annähernd konstanter Bereich liegt zwischen 25.600 und 36.200 Schritten, dort werden numerische Konvergenzordnungen von 4,4–4,7 in allen Variablen erreicht. Die Fehlerwerte liegen nur leicht unter den Fehlerwerten von VSUPER2.

## 6.8 Problem „Chemakzo“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Chemical Akzo Nobel problem“. Es ist ein steifes System von 6 nichtlinearen DAEs mit Index 1. Das Differentialgleichungssystem besitzt die Gestalt

$$M \, dy/dt = f(y), \quad y(0) = y_0, \quad y'(0) = y_0',$$

wobei  $0 \leq t \leq 180$ ,  $y \in \mathbb{R}^6$ , konstante Matrix  $M \in \mathbb{R}^{6 \times 6}$  mit  $\text{Rang}(M) = 5$ ,  $f: \mathbb{R}^6 \rightarrow \mathbb{R}^6$ . Alle Komponenten von  $y$  besitzen Index 1.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem Standardalgorithmus (interface\_dae.f) gerechnet als DAE, somit wurde keine Umformulierung durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt erhält man sehr gutes Lösungsverhalten bei allen Methoden, die theoretisch ermittelten Ordnungswerte lassen sich gut numerisch nachweisen.

Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-13,4}$  bei Benutzung von VNIL3 und Durchführung von 819.200 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

#### **Auswertung der Berechnung mit VNIL2**

Man erhält eine stabile Konvergenzordnung von 2,0 im Bereich von 72.400 – 579.260 Schritten, danach fällt die Ordnung langsam ab.

#### **Auswertung der Berechnung mit VNIL3**

Hier ergibt sich eine stabile Konvergenzordnung von 2,9 – 3,4 im Bereich von 25.600 – 409.600 Schritten. Bei Durchführung von mehr Schritten sind deutlich stärkere Schwankungen als bei VNIL2 aufgetreten.

#### **Auswertung der Berechnung mit VSUPER2**

Es tritt eine stabile Konvergenzordnung von 3,0 – 3,1 im Bereich von 6.400 – 204.800 Schritten auf. Der optimale Bereich beginnt also deutlich früher als bei den beiden vorhergehenden Verfahren. Der Fehler ist allerdings etwas kleiner als bei VNIL3.

#### **Auswertung der Berechnung mit VSUPER3**

Hierbei erhält man ein deutlich kürzeres optimales Schrittzahl-Intervall, in dem die Ordnungswerte annähernd konstant sind. Im Bereich von 72.400 – 289.630 Schritten ergibt sich eine Konvergenzordnung von 2,9 – 3,0. Der Fehler zur Referenzlösung ist noch kleiner als bei VSUPER2.

## 6.9 Problem „Transamp“

### **Beschreibung des Problems**

Die Originalbezeichnung dieses Problems in [CWI99] ist „Transistor amplifier“. Es ist eine steifes System von 8 DAEs mit Index 1. Das Differentialgleichungssystem besitzt die Gestalt

$$M \, dy/dt = f(y), \quad y(0) = y_0, \quad y'(0) = y_0',$$

wobei  $0 \leq t \leq 0,2$ ,  $y \in \mathbb{R}^8$ , konstante Matrix  $M \in \mathbb{R}^{8 \times 8}$  mit  $\text{Rang}(M) = 5$ ,  $f: \mathbb{R}^8 \rightarrow \mathbb{R}^8$ . Alle Komponenten von  $y$  besitzen Index 1.

### **Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem**

Dieses Problem wurde mit dem Standardalgorithmus (interface\_dae.f) gerechnet als DAE, somit wurde keine Umformulierung durchgeführt. Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Insgesamt erhält man sehr gutes Lösungsverhalten bei allen Methoden, die theoretisch ermittelten Ordnungswerte lassen sich gut numerisch nachweisen. Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-11,53}$  bei Benutzung von VSUPER3 und Durchführung von 409.600 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

#### **Auswertung der Berechnung mit VNIL2**

Man erhält eine extrem stabile Konvergenzordnung von 2,0 für 1.130 – 2.317.040 Schritte, hier konnte die Schrittweite also in hohem Maße verringert werden, ohne dass der Rechenfehlereinfluss die Näherungslösung allzu sehr verfälschte.

### Auswertung der Berechnung mit VNIL3

Im Gegensatz zu den Berechnungen mit der Methode VNIL2 ist hier das Intervall für die Schrittzahl, in dem sich stabile Ordnungen ergeben, viel kleiner. Für 4.520 – 102.400 Schritte ergeben sich Ordnungswerte von 2,9 – 3,0.

### Auswertung der Berechnung mit VSUPER2

Eine (fast) linear ansteigende Ordnung von 2,3 bis 3,0 ergibt sich für die Schrittzahlen 1.130 - 144.810, danach treten große Schwankungen auf.

### Auswertung der Berechnung mit VSUPER3

Hier ergibt sich nur ein sehr kleines Schrittzahl-Intervall, in dem die theoretisch erwarteten Ordnungszahlen stabil realisiert wurden: von 4.520 – 25.600 wurden Konvergenzordnungen im Bereich von 3,7 – 4,0 erreicht.

## 6.10 Problem „Water“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Water tube system“. Es ist ein Index-2-System von 49 nichtlinearen DAEs. Das Differentialgleichungssystem besitzt die Gestalt

$$M \, dy/dt = f(y), \quad y(0) = y_0, \quad y'(0) = y_0',$$

wobei  $0 \leq t \leq 17 \cdot 3600$ ,  $y \in \mathbb{R}^{49}$ , konstante Matrix  $M \in \mathbb{R}^{49 \times 49}$ ,  $f: \mathbb{R}^{49} \rightarrow \mathbb{R}^{49}$ . Die ersten 38 Komponenten von  $y$  besitzen Index 1, die letzten 11 sind Index-2-Variablen.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Diese Aufgabe wurde sowohl explizit(DAE, interface\_dae.f) als auch implizit (interface\_impl.f) gerechnet. Da die Ergebnisse der Berechnungen nach der Umformung deutlich besser sind, werden die impliziten Ergebnisse hier in den Vordergrund gestellt. Die Ergebnisse der expliziten Rechnung sind zum Vergleich angefügt. Im Modul „interface\_impl.f“ wurde die erste Ableitung an der Stelle  $t_0 + 10^{-5}$  numerisch bestimmt.

Im Gegensatz zu den vorhergehenden Problemen entsteht hier ein extrem instabiles Lösungsverhalten, d.h. nur mittlere Änderungen an den Schrittzahlen führen zu extremen Änderungen des Fehlers, somit treten viele Vorzeichenwechsel bei den Ordnungszahlen auf. Um trotzdem Qualitätsaussagen über die verwendeten Algorithmen treffen zu können, wurde nach der Berechnung mit einer großen Schrittzahlveränderung für jede Methode ein Intervall herausgesucht, in dem eine stabile Ordnungszahl vermutet wurde. In diesem Teilintervall wurde eine feinere Unterteilung vorgenommen und die jeweiligen Fehlerwerte ausgerechnet. Aus der Durchführung kleinerer Schritte folgt eine bessere Abschätzung der Ordnungswerte.

Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-8.56}$  bei Benutzung von VSUPER3 und Durchführung von 204.800 Schritten  
 Index-2: Fehler  $10^{-11.44}$  bei Benutzung von VSUPER3 und Durchführung von 409.600 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdigramme der Ergebnisse der vier Verfahren.

### Auswertung der Berechnung mit VNIL2

Zunächst wurden Berechnungen für 100 – 819.200 Schritte durchgeführt, dabei wurde als multiplikative Schrittweite etwa  $2^{1/2}$  gewählt. Für größere Schrittzahlen verringern sich zwar die Fehlerwerte, jedoch ist keine Monotonie erkennbar, so dass die Ordnungszahlen starken Schwankungen unterliegen. Durch Verfeinerung des Intervalls [280, 560] mit Hilfe von 7 Zwischenpunkten und einer multiplikativen Schrittweite von etwa  $2^{1/8}$  wird in der Index-2-Komponente hervorragende Ordnungszahlen von 1,9 – 2,1 erreicht. Die Ordnungszahlen der Index-1-Variablen zeigen dagegen keine Stabilität. Der Fehler in Index-2-Variablen ist bei diesem Problem stets kleiner als in den Index-1-Komponenten.

### Auswertung der Berechnung mit VNIL3

Die anfänglichen Rechnungen von 3.200 - 819.200 Schritten zeigen ein ähnliches stark schwankendes Bild wie die Ergebnisse von VNIL2. Auch eine Verfeinerung des Intervalls [3.200, 6.400] bringt hier nicht den gewünschten Erfolg: auch bei kleinerer Schrittzahländerung entstehen keine stabileren Ordnungszahlen. Trotzdem gilt auch hier, dass der Fehler in den Index-2-Komponenten kleiner ist als in den Index-1-Variablen. Der Genauigkeitsunterschied zwischen VNIL2 und VNIL3 ist nur gering.

### Auswertung der Berechnung mit VSUPER2

Hier lassen sich ähnliche Aussagen treffen wie für die Ergebnisse bei Benutzung von VNIL3. Während man in die Ordnungszahlen für Index-2-Variablen bei VNIL3 noch eine gewisse Monotonie erkennen kann, ist dies hier nicht mehr der Fall. Auch die Verfeinerung des Intervalls [25.600, 36.200] bringt keine Stabilität der Ordnungszahlen zutage.

### Auswertung der Berechnung mit VSUPER3

Auch hier zeigt sich ein ähnliches Bild. Der Gesamtfehler wird zwar mit Erhöhung der Schrittzahl global gesehen kleiner, es gibt jedoch kein gleichmäßiges monotonies Verhalten, welches eine relative Konstanz der Ordnungswerte nach sich ziehen würde. Die Betrachtung der Verfeinerung des Intervalls [1.130, 2.260] zeigt, dass selbst die Durchführung von kleinen Schrittzahlerhöhungen keine Stabilität der Ordnungswerte nach sich zieht. Auch hier gilt, dass der Fehler in den Index-2-Variablen geringer ist als in den Index-1-Komponenten.

## 6.11 Problem „Crank“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „Slider Crank“. Es ist ein DAE-System mit Index 2. Das Differentialgleichungssystem besitzt die Gestalt

$$M(p, q) \begin{pmatrix} \ddot{p} \\ \ddot{q} \end{pmatrix} = f(p, \dot{p}, q, \dot{q}) - G(p, q)^T \lambda$$

$$0 = \frac{d}{dt} (g(p, q) + r(t)) = G(p, q) \begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} + \dot{r}(t)$$

wobei  $0 \leq t \leq 0.1$ ,  $p \in \mathbb{R}^3$ ,  $q \in \mathbb{R}^4$ ,  $\lambda \in \mathbb{R}^3$ ,  $M: \mathbb{R}^7 \rightarrow \mathbb{R}^7 \times \mathbb{R}^7$ ,  $f: \mathbb{R}^{14} \rightarrow \mathbb{R}^7$ ,  $g: \mathbb{R}^7 \rightarrow \mathbb{R}^3$ ,  $r: \mathbb{R} \rightarrow \mathbb{R}^3$ ,  $G = \partial g / \partial (p, q)$ ,  $M(p, q)$  ist symmetrisch positiv definit mit  $\text{Rang}(M) = 3$ . Alle Komponenten von  $y$  besitzen Index 1. Die Variablen  $p$  und  $q$  besitzen den Index 1 und  $\lambda$  ist eine Index-2-Variable.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Das Problem „Crank“ kann natürlich nicht mit dem Standard-Algorithmus für DAE-Gleichungen behandelt werden, da die Matrix  $M$  nicht konstant ist. Testrechnungen mit einer konstanten Wahl von  $M(p, q) = M(p_0, q_0)$  haben in der Tat sehr schlechte Fehler- und Ordnungsdaten bzw. keine Konvergenzergebnisse ergeben. Rechnet man „Crank“ dagegen als implizites System, so ergeben sich vor allem für VSUPER2 und VSUPER3 hervorragende Konvergenzraten – nicht nur in Teilintervallen. Diese Ergebnisse wurden für einen Initialpunkt erreicht, der sehr nah am eigentlichen Startpunkt liegt ( $t_0 + 10^{-12}$ ). Für  $t_0 + 10^{-5}$  sind die Ergebnisse nicht so deutlich. Dies liegt wohl vor allem am sehr kleinen Gesamtintegrationsintervall  $[0, 0.1]$ .

Untersucht wurden dabei die vier Methoden VNIL2, VNIL3, VSUPER2 und VSUPER3. Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

Index-1: Fehler  $10^{-5.17}$  bei Benutzung von VNIL2 und Durchführung von 102.400 Schritten

Index-2: Fehler  $10^{-2.05}$  bei Benutzung von VSUPER2 und Durchführung von 144.810 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

### Auswertung der Berechnung mit VNIL2

Bei der Durchführung der Berechnung im Gesamtintervall [1.130, 289.630] ergeben sich sehr stark schwankende Ordnungswerte, die sogar negativ werden. Bei der Betrachtung der Verfeinerung des Intervalls [51.200, 102.400] lässt sich dagegen eine Monotonie der Ordnungswerte erkennen. Für die Index-1-Variablen liegen sie über 2,0 und für die Index-2-Komponente darunter.

### Auswertung der Berechnung mit VNIL3

Hier sind im Gegensatz zu der Berechnung mit VNIL2 die Ordnungswerte der Index-2-Variablen im betrachteten Bereich viel stabiler. Die Ordnungszahlen, die bei der Verfeinerung des Intervalls [12.800, 25.600] ermittelt wurden, sind zwar insgesamt zu hoch für dieses Verfahren, lassen aber eine gewisse Stabilität erkennen.

### Auswertung der Berechnung mit VSUPER2

Die Ordnungszahlen für die Index-2-Variablen liegen im Bereich von 9.050 – 144.810 stabil über 2,0 liegen. Vor allem bei der Verfeinerung des Intervalls [12.800, 25.600] ist die Stabilität der Ordnungswerte aller Variablen deutlich zu erkennen. Es soll weiterhin bemerkt werden, dass die Maximalanzahl der Schritte, bis keine Verbesserung der Genauigkeit mehr eintritt, bei den Index-1-Variablen bei ca. 72.400 liegt; bei der Index-2-Komponente ist diese mit etwa 204.800 deutlich größer.

### Auswertung der Berechnung mit VSUPER3

Hier zeigt sich vor allem bei der Betrachtung der Index-2-Variablen eine stabile Ordnung im Bereich von 2,5. Betrachtet man die Verfeinerung des Intervalls [12.800, 25.600], so besitzen die Index-1-Komponenten eine stabile numerische Konvergenzordnung, die deutlich über 3,0 liegt.

## 6.12 Problem „NAND“

### Beschreibung des Problems

Die Originalbezeichnung dieses Problems in [CWI99] ist „NAND gate“. Es ist ein steifes IDE-System vom Index 1 mit 14 Gleichungen. Das Differentialgleichungssystem besitzt die Gestalt

$$C(y(t)) \, dy/dt = f(t, y(t)), \quad y(0) = y_0, \quad y'(0) = y_0',$$

wobei  $0 \leq t \leq 80$ ,  $y \in \mathbb{R}^{14}$ . Alle Variablen besitzen den Index 1.

### Durchgeführte Rechnungen, Lösungsverhalten der Verfahren für dieses Problem

Dieses Problem wurde mit dem Modul „interface\_impl.f“ und somit mit Einführung neuer Variablen behandelt. Leider ergibt sich bei der Benutzung der Verfahren VNIL2, VNIL3, VSUPER2 und VSUPER3 stets nur eine stabile Konvergenzordnung von 1,0. Da vermutet wurde, dass diese schlechten Ordnungswerte aus der ungenauen Bestimmung der Ableitungswerte aller Unbekannten am modifizierten Intervallanfang  $t_0 + d$  ( $d$  ist geeigneter Parameter) folgen, wurden für den Parameter  $d$  verschiedene Zehnerpotenzen gewählt. Numerische Testrechnungen ergaben jedoch, dass bei Variation von  $d$  im Bereich von  $10^0$  bis  $10^{-13}$  die erreichte numerische Konvergenzordnung für die Schritte 12.800, 18.100 und 25.600 stets 1,0 ist. Man erhält also auch bei Variation von  $d$  keine nennenswerte Verbesserung der Rechengenauigkeit.

Für die nachfolgenden ausführlichen Rechnungen wurde  $d = 10^{-12}$  gewählt.

Bei der Benutzung von VNIL2 ergibt sich im Bereich von 1.130 – 51.200 eine stabile numerische Konvergenzordnung von 1,0, danach liegt sie stabil bei 0,9 – 0,8.

Bei Verwendung von VNIL3 erhält man bei der Durchführung von 2.260 – 36.200 Schritten die stabile Ordnung 1,0. Für größere Schrittweiten liegt sie im Bereich von 0,8 – 1,0.

Löst man dieses Problem mit VSUPER2, so erhält man für 1.130 – 36.200 Schritte eine Konvergenzordnung von 1,0 – 1,1, bei VSUPER3 ist dieser Bereich von 3.200 – 36.200 deutlich kleiner.

Die beste Annäherung an die exakte Referenzlösung erhält man für folgende Parameter:

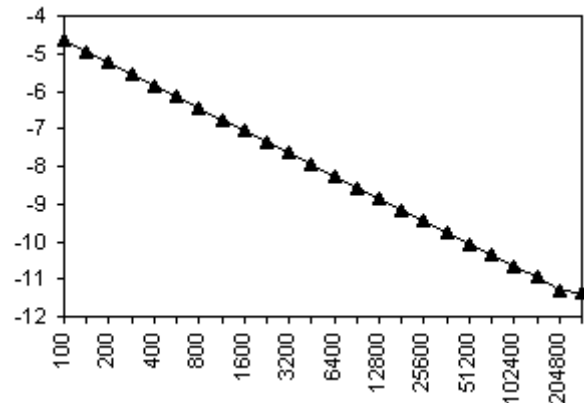
Fehler  $10^{-4,73}$  bei Benutzung von VNIL3 und Durchführung von 3.276.800 Schritten

Im Anhang befinden sich Tabellen, Fehler- und Ordnungsdiagramme der Ergebnisse der vier Verfahren.

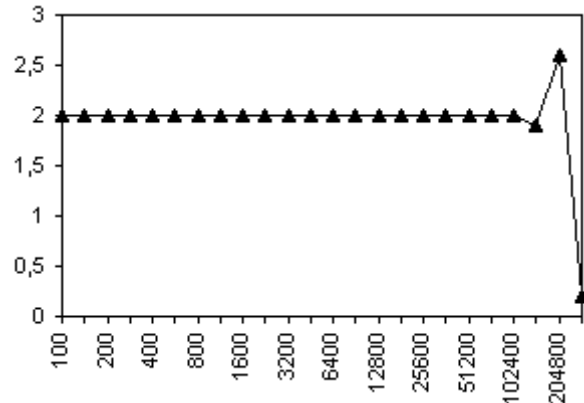
**Problem „Kaps“, Methode VNIL2  
( $d = 10^{-12}$  bei impliziter Rechnung)**

Schritte	Fehler		Ordnung	
	DAE	IMPL	DAE	IMPL
93	-4.60	-4.60	-	-
100	-4.66	-4.66	2.0	2.0
140	-4.95	-4.95	2.0	2.0
200	-5.25	-5.25	2.0	2.0
280	-5.54	-5.54	2.0	2.0
400	-5.85	-5.85	2.0	2.0
560	-6.14	-6.14	2.0	2.0
800	-6.45	-6.45	2.0	2.0
1130	-6.75	-6.75	2.0	2.0
1600	-7.05	-7.05	2.0	2.0
2260	-7.35	-7.35	2.0	2.0
3200	-7.65	-7.65	2.0	2.0
4520	-7.95	-7.95	2.0	2.0
6400	-8.26	-8.26	2.0	2.0
9050	-8.56	-8.56	2.0	2.0
12800	-8.86	-8.86	2.0	2.0
18100	-9.16	-9.16	2.0	2.0
25600	-9.46	-9.46	2.0	2.0
36200	-9.76	-9.76	2.0	2.0
51200	-10.06	-10.06	2.0	2.0
72400	-10.36	-10.36	2.0	2.0
102400	-10.65	-10.66	2.0	2.0
144810	-10.93	-10.94	1.9	1.9
204800	-11.33	-11.36	2.6	2.8
289630	-11.35	-11.35	0.2	0

Fehlerdiagramm



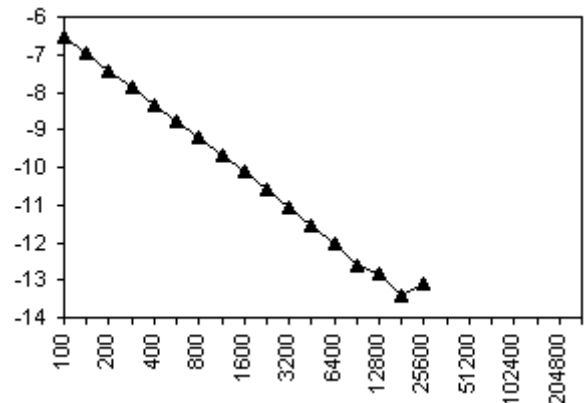
Ordnungsdiagramm



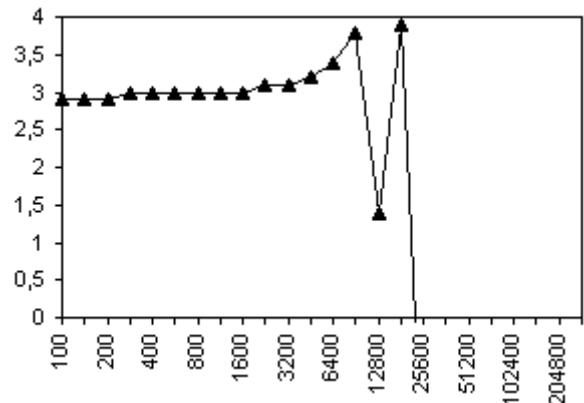
**Problem „Kaps“, Methode VNIL3  
( $d = 10^{-12}$  bei impliziter Rechnung)**

Schritte	Fehler		Ordnung	
	DAE	IMPL	DAE	IMPL
93	-6.45	-6.45	-	-
100	-6.55	-6.55	2.9	2.9
140	-6.97	-6.97	2.9	2.9
200	-7.43	-7.43	2.9	2.9
280	-7.86	-7.86	3.0	3.0
400	-8.32	-8.32	3.0	3.0
560	-8.76	-8.76	3.0	3.0
800	-9.22	-9.22	3.0	3.0
1130	-9.67	-9.67	3.0	3.0
1600	-10.13	-10.13	3.0	3.0
2260	-10.58	-10.58	3.1	3.0
3200	-11.05	-11.05	3.1	3.1
4520	-11.54	-11.54	3.2	3.2
6400	-12.04	-12.08	3.4	3.6
9050	-12.62	-12.62	3.8	3.6
12800	-12.84	-13.20	1.4	3.8
18100	-13.43	-12.84	3.9	-2.4
25600	-13.09	-12.57	-2.3	-1.8

Fehlerdiagramm



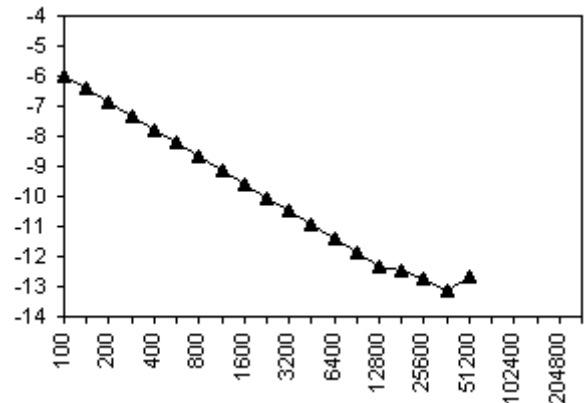
Ordnungsdiagramm



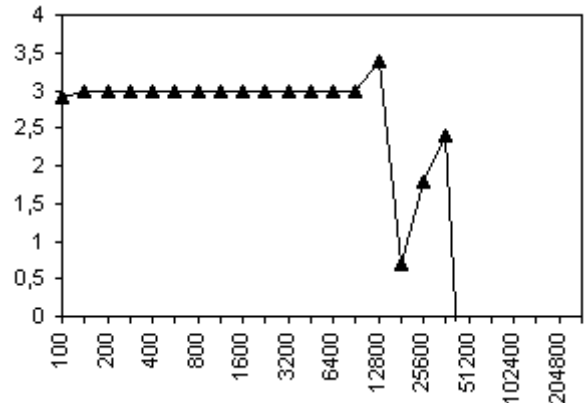
**Problem „Kaps“, Methode VSUPER2  
( $d = 10^{-12}$  bei impliziter Rechnung)**

Schritte	Fehler		Ordnung	
	DAE	IMPL	DAE	IMPL
93	-5.91	-5.91	–	–
100	-6.00	-6.00	2.9	2.9
140	-6.43	-6.43	3.0	3.0
200	-6.89	-6.89	3.0	3.0
280	-7.33	-7.33	3.0	3.0
400	-7.79	-7.79	3.0	3.0
560	-8.23	-8.23	3.0	3.0
800	-8.69	-8.69	3.0	3.0
1130	-9.14	-9.14	3.0	3.0
1600	-9.59	-9.59	3.0	3.0
2260	-10.04	-10.04	3.0	3.0
3200	-10.49	-10.49	3.0	3.0
4520	-10.94	-10.94	3.0	3.0
6400	-11.39	-11.40	3.0	3.0
9050	-11.85	-11.85	3.0	3.0
12800	-12.36	-12.32	3.4	3.1
18100	-12.47	-12.76	0.7	2.9
25600	-12.75	-13.37	1.8	4.1
36200	-13.11	-12.42	2.4	-6.3
51200	-12.65	-13.43	-3.1	6.7

Fehlerdiagramm



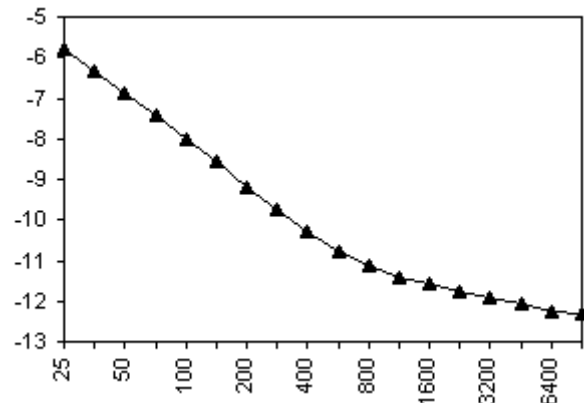
Ordnungsdiagramm



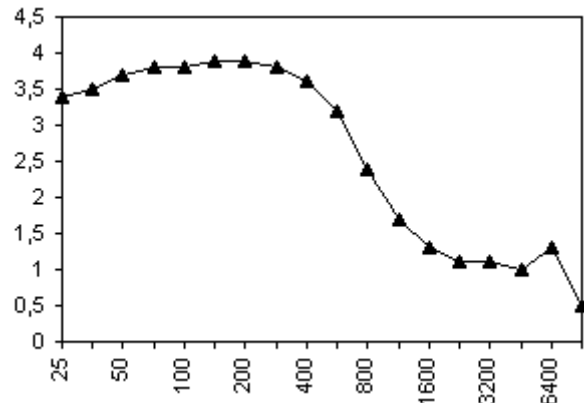
**Problem „Kaps“, Methode VSUPER3  
( $d = 10^{-12}$  bei impliziter Rechnung)**

Schritte	Fehler		Ordnung	
	DAE	IMPL	DAE	IMPL
18	-5.31	-5.31	–	–
25	-5.79	-5.79	3.4	3.4
35	-6.31	-6.31	3.5	3.5
50	-6.88	-6.88	3.7	3.7
70	-7.43	-7.43	3.8	3.8
100	-8.02	-8.02	3.8	3.8
140	-8.58	-8.58	3.9	3.9
200	-9.18	-9.18	3.9	3.9
280	-9.74	-9.74	3.8	3.8
400	-10.30	-10.30	3.6	3.6
560	-10.76	-10.76	3.2	3.2
800	-11.14	-11.14	2.4	2.4
1130	-11.40	-11.40	1.7	1.7
1600	-11.59	-11.59	1.3	1.3
2260	-11.76	-11.76	1.1	1.1
3200	-11.92	-11.92	1.1	1.1
4520	-12.07	-12.07	1.0	1.0
6400	-12.26	-12.23	1.3	1.1
9050	-12.33	-12.34	0.5	0.7

Fehlerdiagramm Gesamtintervall



Ordnungsdiagramm Gesamtintervall

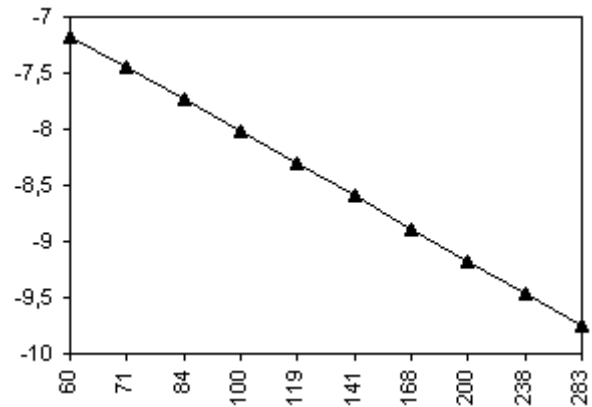


**Problem „Kaps“, Methode VSUPER3  
( $d = 10^{-12}$  bei impliziter Rechnung)**

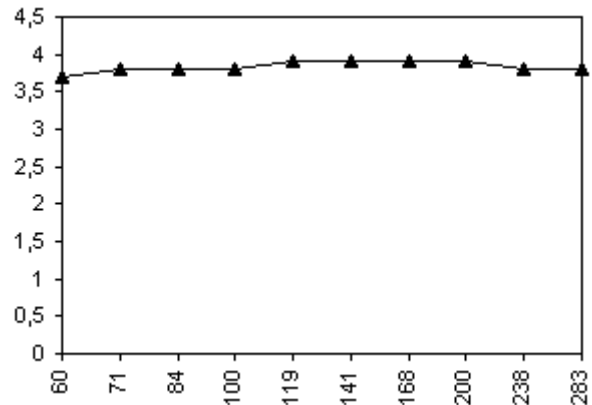
Verfeinerung von [50, 280]

Schritte	Fehler		Ordnung	
	DAE	IMPL	DAE	IMPL
50	-6.88	-6.88	–	–
60	-7.17	-7.17	3.7	3.7
71	-7.45	-7.45	3.8	3.8
84	-7.73	-7.73	3.8	3.8
100	-8.02	-8.02	3.8	3.8
119	-8.31	-8.31	3.9	3.9
141	-8.59	-8.59	3.9	3.9
168	-8.89	-8.89	3.9	3.9
200	-9.18	-9.18	3.9	3.9
238	-9.47	-9.47	3.8	3.8
283	-9.75	-9.75	3.8	3.8

Fehlerdiagramm Teilintervall



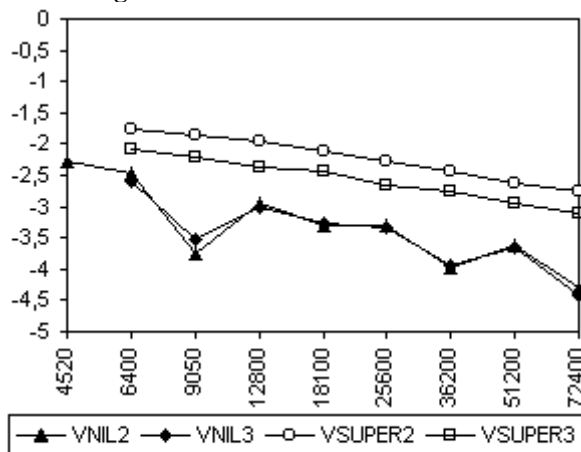
Ordnungsdiagramm Teilintervall



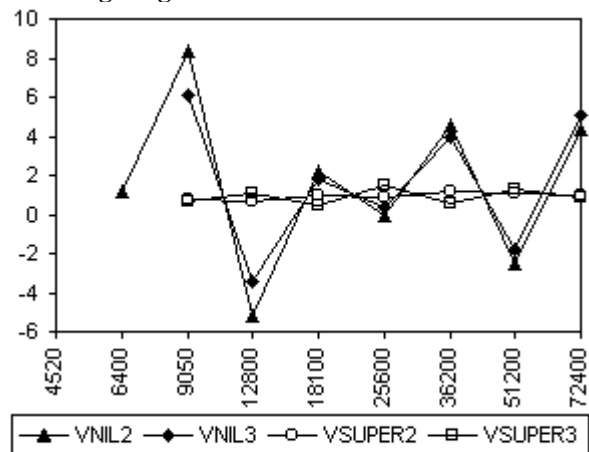
**Problem „TBA“, Methoden VNIL2, VNIL3, VSUPER2, VSUPER3**

Schritte	VNIL2		VNIL3		VSUPER2		VSUPER3	
	Fehler	Ordnung	Fehler	Ordnung	Fehler	Ordnung	Fehler	Ordnung
4520	-2.29	–	–	–	–	–	–	–
6400	-2.47	1.2	-2.60	–	-1.75	–	-2.09	–
9050	-3.74	8.4	-3.51	6.1	-1.87	0.8	-2.20	0.7
12800	-2.96	-5.2	-3.00	-3.4	-1.97	0.7	-2.37	1.1
18100	-3.30	2.2	-3.27	1.8	-2.13	1.0	-2.44	0.5
25600	-3.30	0	-3.34	0.4	-2.27	0.9	-2.66	1.5
36200	-3.99	4.6	-3.94	4.0	-2.45	1.2	-2.76	0.6
51200	-3.62	-2.5	-3.66	-1.8	-2.62	1.1	-2.96	1.3
72400	-4.28	4.4	-4.43	5.1	-2.76	1.0	-3.10	0.9

Fehlerdiagramm



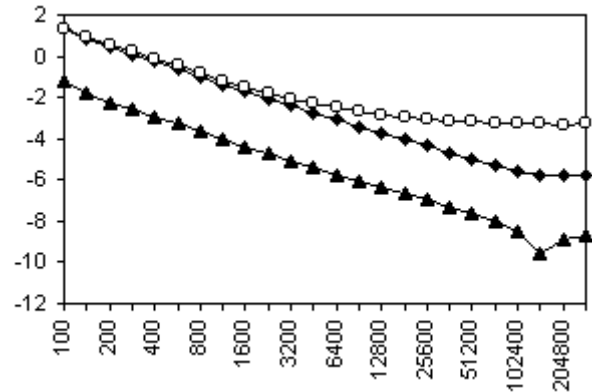
Ordnungsdiagramm



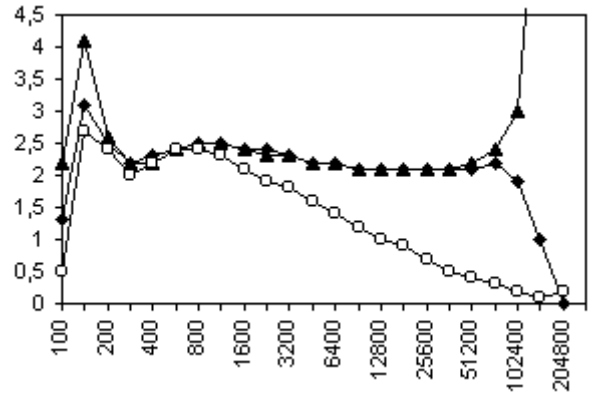
**Problem „Andrews“, Methode VNIL2**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
93	-1.17	1.32	1.30	-	-	-
100	-1.24	1.28	1.28	2.2	1.3	0.5
140	-1.84	0.82	0.89	4.1	3.1	2.7
200	-2.24	0.43	0.52	2.6	2.5	2.4
280	-2.56	0.10	0.23	2.2	2.2	2.0
400	-2.91	-0.25	-0.11	2.2	2.3	2.2
560	-3.26	-0.60	-0.46	2.4	2.4	2.4
800	-3.64	-0.99	-0.83	2.5	2.5	2.4
1130	-4.01	-1.36	-1.17	2.5	2.5	2.3
1600	-4.37	-1.72	-1.49	2.4	2.4	2.1
2260	-4.72	-2.08	-1.78	2.3	2.4	1.9
3200	-5.07	-2.42	-2.05	2.3	2.3	1.8
4520	-5.40	-2.76	-2.28	2.2	2.2	1.6
6400	-5.73	-3.08	-2.49	2.2	2.2	1.4
9050	-6.05	-3.40	-2.67	2.1	2.1	1.2
12800	-6.36	-3.72	-2.83	2.1	2.1	1.0
18100	-6.68	-4.03	-2.96	2.1	2.1	0.9
25600	-6.99	-4.35	-3.06	2.1	2.1	0.7
36200	-7.31	-4.66	-3.14	2.1	2.1	0.5
51200	-7.64	-4.98	-3.20	2.2	2.1	0.4
72400	-8.01	-5.30	-3.24	2.4	2.2	0.3
102400	-8.46	-5.60	-3.27	3.0	1.9	0.2
144810	-9.55	-5.74	-3.28	7.2	1.0	0.1
204800	-8.89	-5.75	-3.30	-4.4	0	0.2
289630	-8.68	-5.73	-3.28	-1.4	-0.1	-0.1

**Fehlerdiagramm**



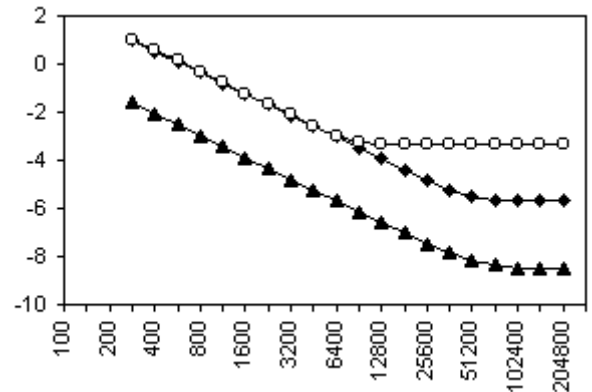
**Ordnungsdiagramm**



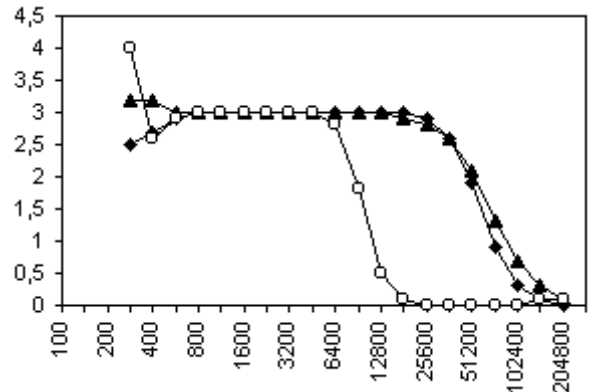
**Problem „Andrews“, Methode VNIL3**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
280	-1.60	0.96	0.97	3.2	2.5	4.0
400	-2.09	0.54	0.57	3.2	2.7	2.6
560	-2.54	0.11	0.15	3.0	2.9	2.9
800	-3.00	-0.35	-0.31	3.0	3.0	3.0
1130	-3.45	-0.80	-0.76	3.0	3.0	3.0
1600	-3.90	-1.24	-1.21	3.0	3.0	3.0
2260	-4.35	-1.69	-1.65	3.0	3.0	3.0
3200	-4.80	-2.14	-2.10	3.0	3.0	3.0
4520	-5.25	-2.59	-2.55	3.0	3.0	3.0
6400	-5.70	-3.04	-2.96	3.0	3.0	2.8
9050	-6.15	-3.49	-3.24	3.0	3.0	1.8
12800	-6.59	-3.94	-3.32	3.0	3.0	0.5
18100	-7.04	-4.39	-3.32	2.9	3.0	0.1
25600	-7.46	-4.82	-3.32	2.8	2.9	0
36200	-7.85	-5.22	-3.32	2.6	2.6	0
51200	-8.16	-5.51	-3.31	2.1	1.9	0
72400	-8.36	-5.64	-3.32	1.3	0.9	0
102400	-8.46	-5.68	-3.31	0.7	0.3	0
144810	-8.51	-5.69	-3.33	0.3	0.1	0.1
204800	-8.53	-5.69	-3.34	0.1	0	0.1

**Fehlerdiagramm**



**Ordnungsdiagramm**

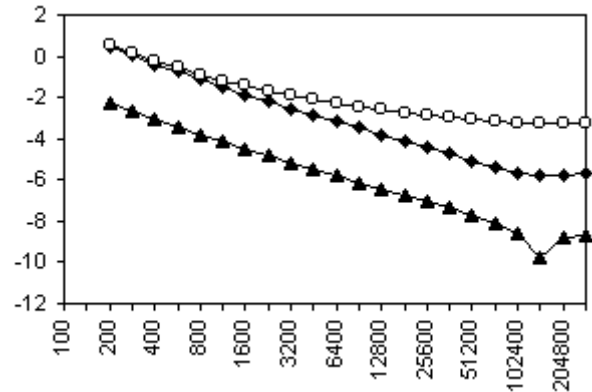


▲ Index-1    ◆ Index-2    □ Index-3

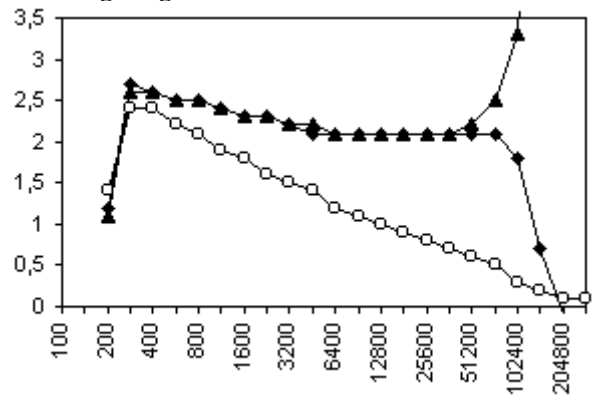
**Problem „Andrews“, Methode VSUPER2**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
200	-2.26	0.41	0.50	1.1	1.2	1.4
280	-2.64	0.01	0.15	2.6	2.7	2.4
400	-3.04	-0.40	-0.22	2.6	2.6	2.4
560	-3.41	-0.77	-0.54	2.5	2.5	2.2
800	-3.79	-1.15	-0.87	2.5	2.5	2.1
1130	-4.15	-1.51	-1.16	2.4	2.4	1.9
1600	-4.50	-1.86	-1.43	2.3	2.3	1.8
2260	-4.84	-2.20	-1.67	2.3	2.3	1.6
3200	-5.17	-2.53	-1.90	2.2	2.2	1.5
4520	-5.49	-2.85	-2.10	2.2	2.1	1.4
6400	-5.81	-3.17	-2.29	2.1	2.1	1.2
9050	-6.13	-3.49	-2.46	2.1	2.1	1.1
12800	-6.44	-3.80	-2.61	2.1	2.1	1.0
18100	-6.75	-4.11	-2.76	2.1	2.1	0.9
25600	-7.06	-4.42	-2.88	2.1	2.1	0.8
36200	-7.38	-4.73	-2.99	2.1	2.1	0.7
51200	-7.72	-5.05	-3.08	2.2	2.1	0.6
72400	-8.09	-5.38	-3.15	2.5	2.1	0.5
102400	-8.59	-5.65	-3.21	3.3	1.8	0.3
144810	-9.81	-5.75	-3.24	8.1	0.7	0.2
204800	-8.82	-5.74	-3.26	-6.5	-0.1	0.1
289630	-8.66	-5.72	-3.28	-1.1	-0.1	0.1

Fehlerdiagramm



Ordnungsdiagramm

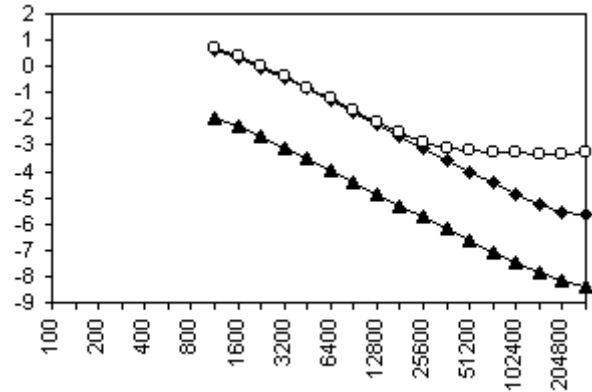


**Problem „Andrews“, Methode VSUPER3**

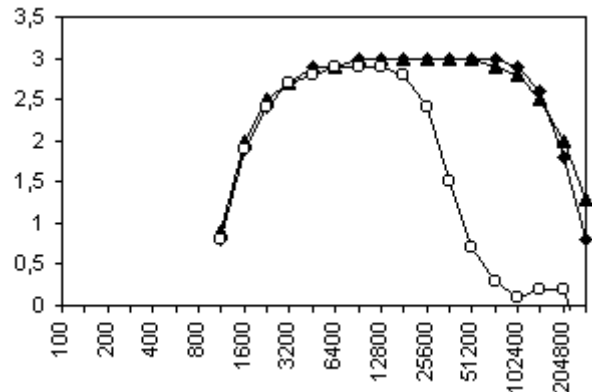
Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
1130	-1.99	0.63	0.68	0.9	0.8	0.8
1600	-2.30	0.34	0.40	2.0	1.9	1.9
2260	-2.67	-0.03	0.03	2.5	2.4	2.4
3200	-3.09	-0.44	-0.37	2.7	2.7	2.7
4520	-3.52	-0.86	-0.80	2.9	2.9	2.8
6400	-3.96	-1.31	-1.24	2.9	2.9	2.9
9050	-4.40	-1.75	-1.68	3.0	3.0	2.9
12800	-4.85	-2.20	-2.12	3.0	3.0	2.9
18100	-5.30	-2.65	-2.53	3.0	3.0	2.8
25600	-5.75	-3.10	-2.89	3.0	3.0	2.4
36200	-6.20	-3.55	-3.12	3.0	3.0	1.5
51200	-6.65	-4.00	-3.23	3.0	3.0	0.7
72400	-7.09	-4.44	-3.28	2.9	3.0	0.3
102400	-7.51	-4.88	-3.29	2.8	2.9	0.1
144810	-7.89	-5.27	-3.32	2.5	2.6	0.2
204800	-8.19	-5.53	-3.34	2.0	1.8	0.2
289630	-8.38	-5.65	-3.25	1.3	0.8	-0.6

—▲— Index-1 —◆— Index-2 —□— Index-3

Fehlerdiagramm



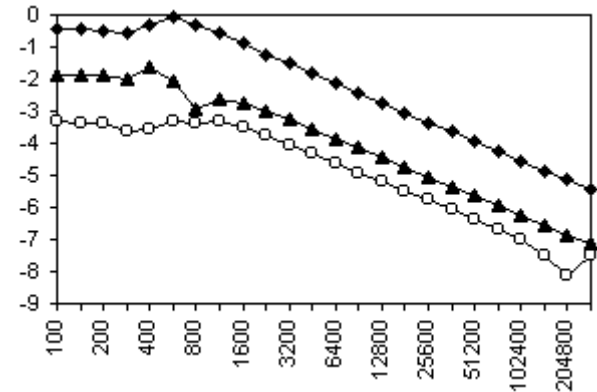
Ordnungsdiagramm



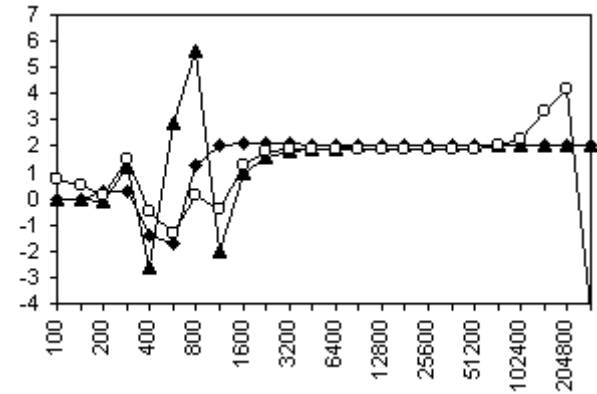
**Problem „Caraxis“, Methode VNIL2**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
93	-1.86	-0.46	-3.30	-	-	-
100	-1.86	-0.46	-3.32	0	0	0.7
140	-1.86	-0.46	-3.39	0	0	0.5
200	-1.85	-0.51	-3.40	-0.1	0.3	0.1
280	-2.03	-0.55	-3.62	1.2	0.3	1.5
400	-1.62	-0.33	-3.54	-2.6	-1.4	-0.5
560	-2.05	-0.09	-3.34	2.9	-1.7	-1.3
800	-2.91	-0.29	-3.36	5.6	1.3	0.1
1130	-2.61	-0.59	-3.30	-2.0	2.0	-0.4
1600	-2.76	-0.90	-3.51	1.0	2.1	1.3
2260	-3.00	-1.22	-3.77	1.6	2.1	1.8
3200	-3.27	-1.53	-4.05	1.8	2.1	1.9
4520	-3.56	-1.83	-4.34	1.9	2.0	1.9
6400	-3.85	-2.14	-4.63	1.9	2.0	1.9
9050	-4.15	-2.44	-4.92	2.0	2.0	1.9
12800	-4.45	-2.74	-5.21	2.0	2.0	1.9
18100	-4.75	-3.04	-5.50	2.0	2.0	1.9
25600	-5.05	-3.35	-5.78	2.0	2.0	1.9
36200	-5.35	-3.65	-6.07	2.0	2.0	1.9
51200	-5.65	-3.95	-6.35	2.0	2.0	1.9
72400	-5.95	-4.25	-6.66	2.0	2.0	2.0
102400	-6.25	-4.55	-7.01	2.0	2.0	2.3
144810	-6.55	-4.85	-7.51	2.0	2.0	3.3
204800	-6.85	-5.15	-8.15	2.0	2.0	4.2
289630	-7.15	-5.45	-7.49	2.0	2.0	-4.4

**Fehlerdiagramm**



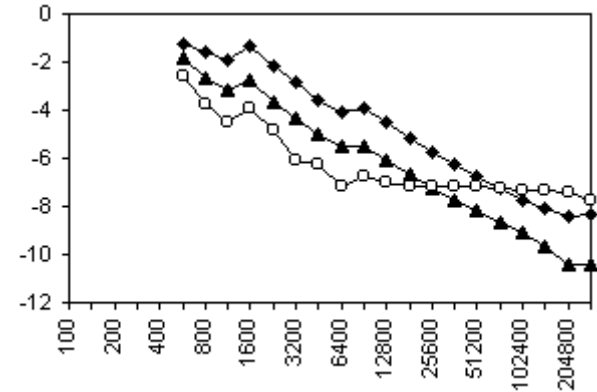
**Ordnungsdiagramm**



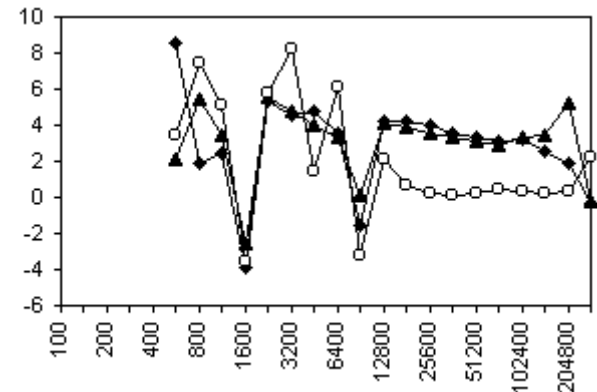
**Problem „Caraxis“, Methode VNIL3**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
560	-1.83	-1.28	-2.59	2.1	8.6	3.5
800	-2.66	-1.57	-3.73	5.4	1.9	7.4
1130	-3.18	-1.95	-4.50	3.5	2.5	5.1
1600	-2.79	-1.36	-3.95	-2.5	-3.9	-3.6
2260	-3.64	-2.16	-4.82	5.6	5.3	5.8
3200	-4.36	-2.85	-6.06	4.8	4.6	8.2
4520	-4.97	-3.58	-6.28	4.0	4.8	1.5
6400	-5.47	-4.12	-7.20	3.3	3.6	6.1
9050	-5.49	-3.88	-6.72	0.1	-1.6	-3.2
12800	-6.11	-4.51	-7.03	4.1	4.2	2.1
18100	-6.70	-5.14	-7.14	3.9	4.2	0.7
25600	-7.23	-5.74	-7.16	3.6	4.0	0.2
36200	-7.72	-6.29	-7.18	3.3	3.6	0.1
51200	-8.19	-6.78	-7.20	3.1	3.3	0.2
72400	-8.63	-7.24	-7.26	2.9	3.1	0.4
102400	-9.12	-7.71	-7.31	3.3	3.2	0.3
144810	-9.66	-8.11	-7.34	3.5	2.6	0.2
204800	-10.4	-8.39	-7.39	5.2	1.9	0.3
289630	-10.4	-8.35	-7.72	-0.2	-0.3	2.2

**Fehlerdiagramm**



**Ordnungsdiagramm**

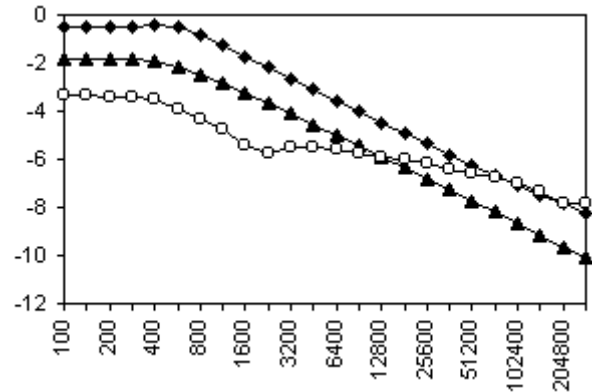


—▲— Index-1 —◆— Index-2 —□— Index-3

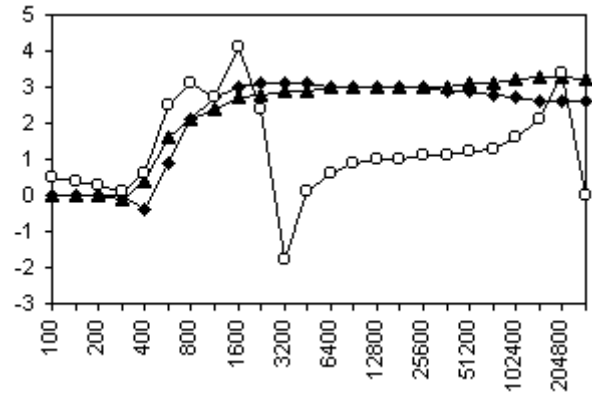
**Problem „Caraxis“, Methode VSUPER2**

Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
93	-1.86	-0.46	-3.30	-	-	-
100	-1.86	-0.46	-3.31	0	0	0.5
140	-1.86	-0.46	-3.37	0	0	0.4
200	-1.86	-0.46	-3.42	0	0	0.3
280	-1.85	-0.46	-3.43	-0.1	0	0.1
400	-1.91	-0.40	-3.52	0.4	-0.4	0.6
560	-2.15	-0.53	-3.88	1.6	0.9	2.5
800	-2.48	-0.86	-4.37	2.1	2.1	3.1
1130	-2.84	-1.27	-4.78	2.4	2.7	2.7
1600	-3.25	-1.72	-5.40	2.7	3.0	4.1
2260	-3.67	-2.18	-5.77	2.8	3.1	2.4
3200	-4.11	-2.65	-5.49	2.9	3.1	-1.8
4520	-4.55	-3.11	-5.51	2.9	3.1	0.1
6400	-5.00	-3.57	-5.61	3.0	3.0	0.6
9050	-5.45	-4.02	-5.74	3.0	3.0	0.9
12800	-5.90	-4.47	-5.88	3.0	3.0	1.0
18100	-6.35	-4.92	-6.04	3.0	3.0	1.0
25600	-6.81	-5.37	-6.20	3.0	3.0	1.1
36200	-7.27	-5.81	-6.38	3.0	2.9	1.1
51200	-7.73	-6.24	-6.56	3.1	2.9	1.2
72400	-8.19	-6.66	-6.76	3.1	2.8	1.3
102400	-8.67	-7.08	-7.00	3.2	2.7	1.6
144810	-9.16	-7.48	-7.31	3.3	2.6	2.1
204800	-9.66	-7.86	-7.83	3.3	2.6	3.4
289630	-10.1	-8.26	-7.83	3.2	2.6	0

**Fehlerdiagramm**



**Ordnungsdiagramm**

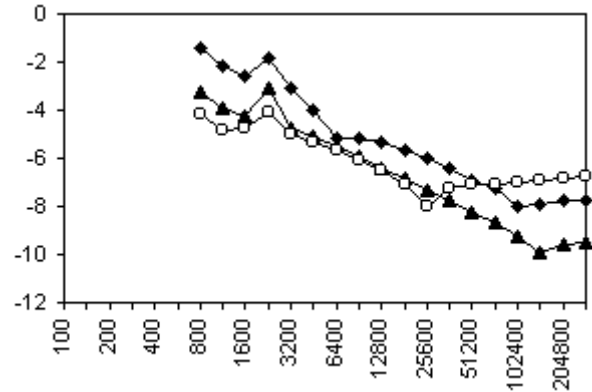


**Problem „Caraxis“, Methode VSUPER3**

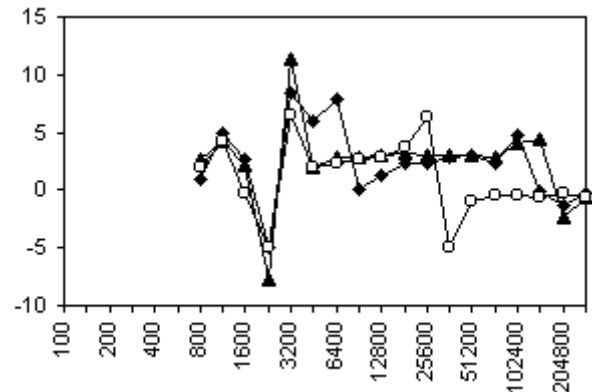
Schritte	Fehler			Ordnung		
	1	2	3	1	2	3
800	-3.27	-1.40	-4.19	2.6	1.0	2.0
1130	-3.91	-2.15	-4.82	4.2	5.0	4.2
1600	-4.24	-2.56	-4.78	2.2	2.7	-0.3
2260	-3.08	-1.82	-4.06	-7.8	-4.9	-4.9
3200	-4.79	-3.09	-5.04	11.3	8.4	6.5
4520	-5.09	-3.98	-5.32	2.0	5.9	1.9
6400	-5.51	-5.15	-5.68	2.8	7.8	2.4
9050	-5.94	-5.15	-6.06	2.9	0	2.6
12800	-6.40	-5.33	-6.50	3.0	1.2	2.9
18100	-6.87	-5.68	-7.05	3.2	2.3	3.7
25600	-7.32	-6.03	-8.01	3.0	2.3	6.4
36200	-7.78	-6.44	-7.27	3.1	2.8	-4.9
51200	-8.25	-6.89	-7.12	3.1	3.0	-1.0
72400	-8.68	-7.25	-7.06	2.8	2.4	-0.5
102400	-9.28	-7.96	-6.98	4.0	4.7	-0.5
144810	-9.95	-7.94	-6.90	4.4	-0.1	-0.6
204800	-9.59	-7.75	-6.86	-2.4	-1.3	-0.3
289630	-9.49	-7.71	-6.76	-0.6	-0.3	-0.7

—▲— Index-1    —◆— Index-2    —□— Index-3

**Fehlerdiagramm**



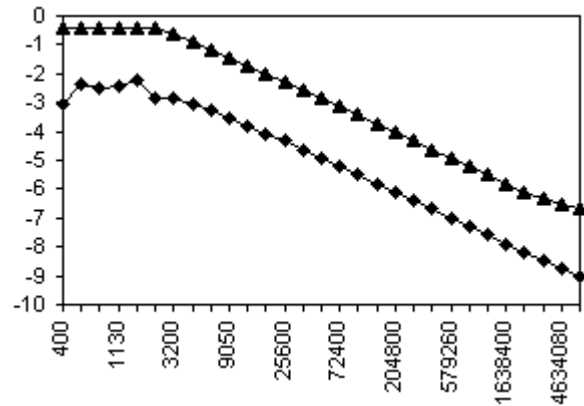
**Ordnungsdiagramm**



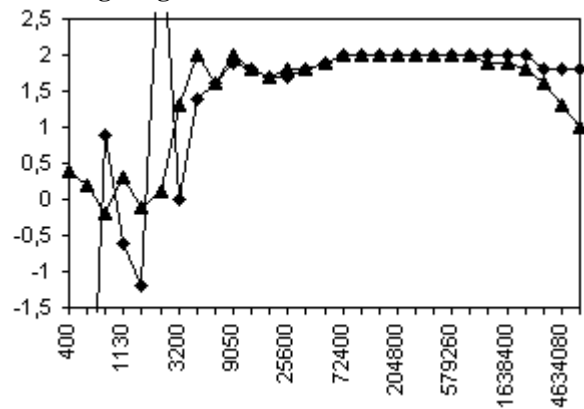
**Problem „Wheel“, Methode VNIL2**

Schritte	Fehler		Ordnung	
	1	2	1	2
400	-0.39	-3.07	0.4	2.8
560	-0.43	-2.38	0.2	-4.7
800	-0.39	-2.52	-0.2	0.9
1130	-0.44	-2.43	0.3	-0.6
1600	-0.42	-2.25	-0.1	-1.2
2260	-0.44	-2.83	0.1	3.9
3200	-0.63	-2.83	1.3	0
4520	-0.93	-3.04	2.0	1.4
6400	-1.17	-3.28	1.6	1.6
9050	-1.48	-3.56	2.0	1.9
12800	-1.75	-3.83	1.8	1.8
18100	-2.01	-4.08	1.7	1.7
25600	-2.27	-4.34	1.8	1.7
36200	-2.55	-4.62	1.8	1.8
51200	-2.84	-4.91	1.9	1.9
72400	-3.13	-5.20	2.0	2.0
102400	-3.43	-5.50	2.0	2.0
144810	-3.73	-5.80	2.0	2.0
204800	-4.03	-6.10	2.0	2.0
289630	-4.33	-6.40	2.0	2.0
409600	-4.63	-6.70	2.0	2.0
579260	-4.93	-7.00	2.0	2.0
819200	-5.23	-7.30	2.0	2.0
1158520	-5.52	-7.60	1.9	2.0
1638400	-5.81	-7.90	1.9	2.0
2317040	-6.08	-8.20	1.8	2.0
3276800	-6.31	-8.47	1.6	1.8
4634080	-6.51	-8.75	1.3	1.8
6553600	-6.66	-9.02	1.0	1.8

**Fehlerdiagramm**



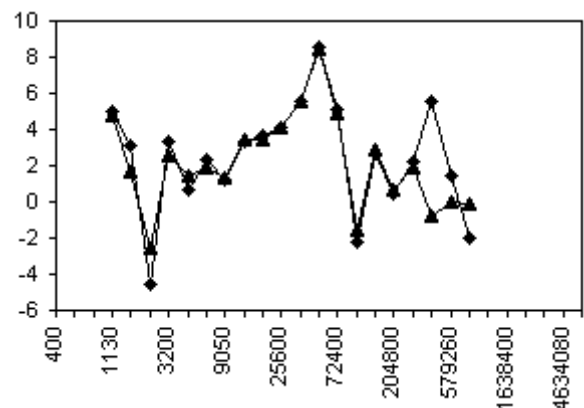
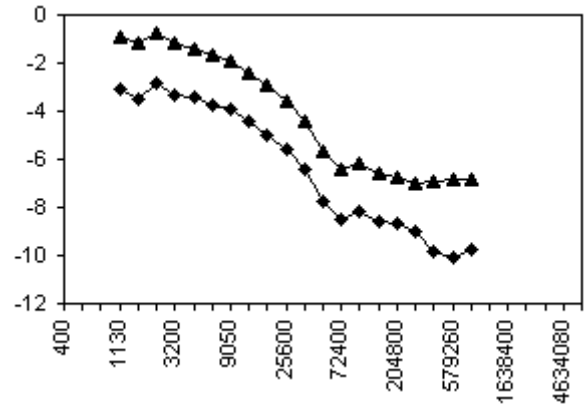
**Ordnungsdiagramm**



**Problem „Wheel“, Methode VNIL3**

Schritte	Fehler		Ordnung	
	1	2	1	2
1130	-0.90	-3.05	4.8	5.0
1600	-1.16	-3.51	1.7	3.1
2260	-0.79	-2.82	-2.5	-4.6
3200	-1.18	-3.32	2.6	3.3
4520	-1.40	-3.43	1.5	0.7
6400	-1.69	-3.77	1.9	2.3
9050	-1.89	-3.95	1.3	1.2
12800	-2.41	-4.44	3.4	3.3
18100	-2.94	-5.00	3.5	3.7
25600	-3.56	-5.61	4.1	4.1
36200	-4.40	-6.45	5.6	5.6
51200	-5.68	-7.75	8.5	8.6
72400	-6.42	-8.51	4.9	5.1
102400	-6.18	-8.17	-1.6	-2.2
144810	-6.61	-8.58	2.9	2.7
204800	-6.71	-8.66	0.7	0.5
289630	-7.00	-8.99	1.9	2.2
409600	-6.88	-9.83	-0.8	5.6
579260	-6.87	-10.1	0	1.4
819200	-6.86	-9.75	-0.1	-2.0

**Fehler- und Ordnungsdiagramm**

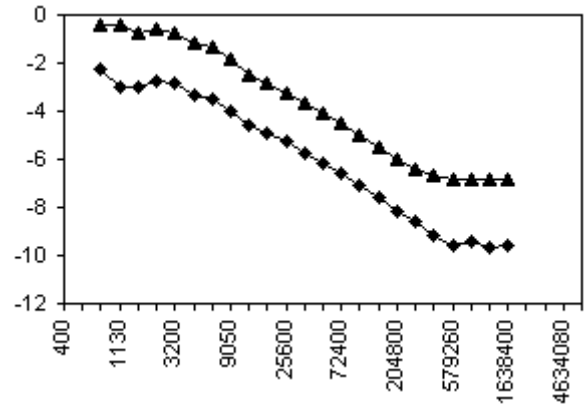


▲ Index-1    ◆ Index-2

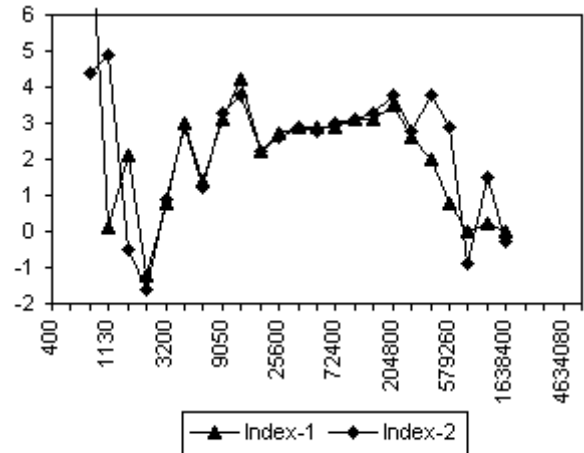
**Problem „Wheel“, Methode VSUPER2**

Schritte	Fehler		Ordnung	
	1	2	1	2
800	-0.43	-2.29	8.8	4.4
1130	-0.45	-3.03	0.1	4.9
1600	-0.77	-2.96	2.1	-0.5
2260	-0.60	-2.72	-1.2	-1.6
3200	-0.72	-2.85	0.8	0.9
4520	-1.17	-3.30	3.0	2.9
6400	-1.37	-3.47	1.4	1.2
9050	-1.84	-3.97	3.1	3.3
12800	-2.48	-4.55	4.2	3.8
18100	-2.82	-4.89	2.2	2.2
25600	-3.23	-5.29	2.7	2.6
36200	-3.66	-5.72	2.9	2.9
51200	-4.09	-6.15	2.9	2.8
72400	-4.53	-6.60	2.9	3.0
102400	-5.00	-7.07	3.1	3.1
144810	-5.46	-7.57	3.1	3.3
204800	-5.99	-8.14	3.5	3.8
289630	-6.38	-8.57	2.6	2.8
409600	-6.68	-9.14	2.0	3.8
579260	-6.81	-9.57	0.8	2.9
819200	-6.81	-9.43	0	-0.9
1158520	-6.84	-9.65	0.2	1.5
1638400	-6.84	-9.61	0	-0.3

**Fehlerdiagramm**



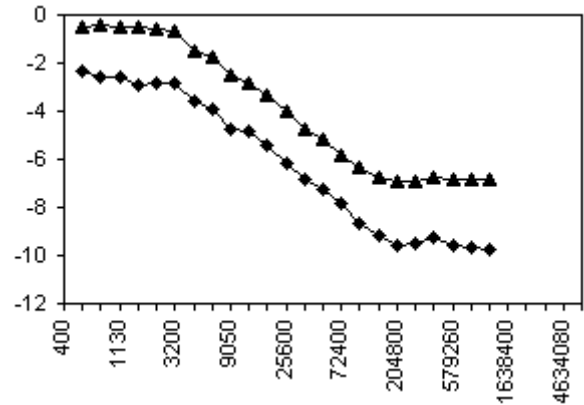
**Ordnungsdiagramm**



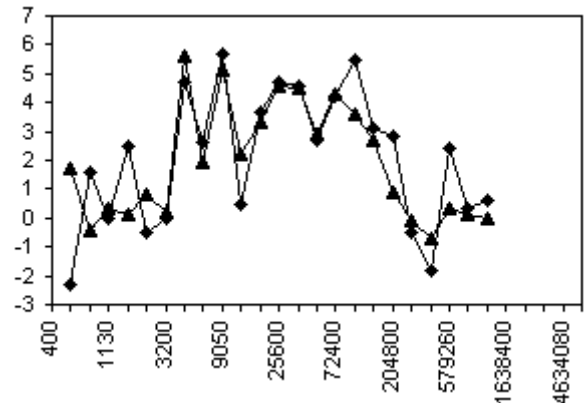
**Problem „Wheel“, Methode VSUPER3**

Schritte	Fehler		Ordnung	
	1	2	1	2
560	-0.46	-2.30	1.7	-2.3
800	-0.41	-2.55	-0.4	1.6
1130	-0.46	-2.55	0.3	0
1600	-0.48	-2.93	0.1	2.5
2260	-0.60	-2.85	0.8	-0.5
3200	-0.63	-2.84	0.2	0
4520	-1.47	-3.55	5.6	4.7
6400	-1.75	-3.94	1.9	2.6
9050	-2.52	-4.79	5.1	5.7
12800	-2.86	-4.87	2.2	0.5
18100	-3.35	-5.42	3.3	3.7
25600	-4.04	-6.13	4.6	4.7
36200	-4.72	-6.81	4.5	4.6
51200	-5.16	-7.23	2.9	2.7
72400	-5.81	-7.86	4.3	4.2
102400	-6.36	-8.68	3.6	5.5
144810	-6.76	-9.15	2.7	3.1
204800	-6.90	-9.57	0.9	2.8
289630	-6.89	-9.49	-0.1	-0.5
409600	-6.78	-9.22	-0.7	-1.8
579260	-6.82	-9.59	0.3	2.4
819200	-6.84	-9.64	0.1	0.3
1158520	-6.85	-9.73	0	0.6

**Fehlerdiagramm**



**Ordnungsdiagramm**

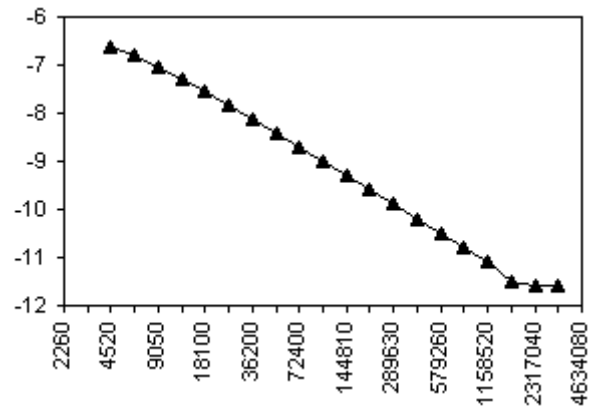


**Problem „Chemakzo“, Methode VNIL2**

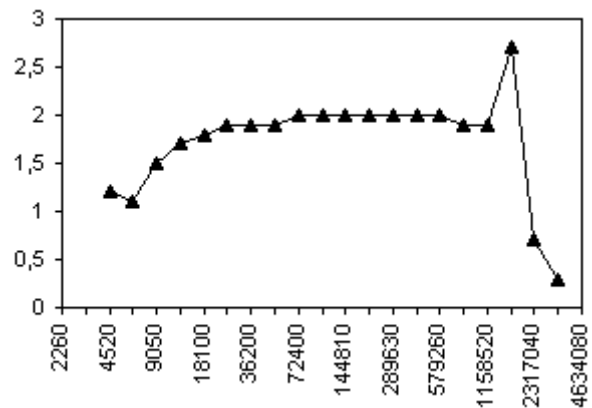
Schritte	Fehler	Ordnung
	1	1
4520	-6.64	1.2
6400	-6.81	1.1
9050	-7.03	1.5
12800	-7.28	1.7
18100	-7.55	1.8
25600	-7.83	1.9
36200	-8.11	1.9
51200	-8.41	1.9
72400	-8.70	2.0
102400	-9.00	2.0
144810	-9.29	2.0
204800	-9.59	2.0
289630	-9.89	2.0
409600	-10.2	2.0
579260	-10.5	2.0
819200	-10.8	1.9
1158520	-11.1	1.9
1638400	-11.5	2.7
2317040	-11.6	0.7
3276800	-11.6	0.3

▲ Index-1

**Fehlerdiagramm**



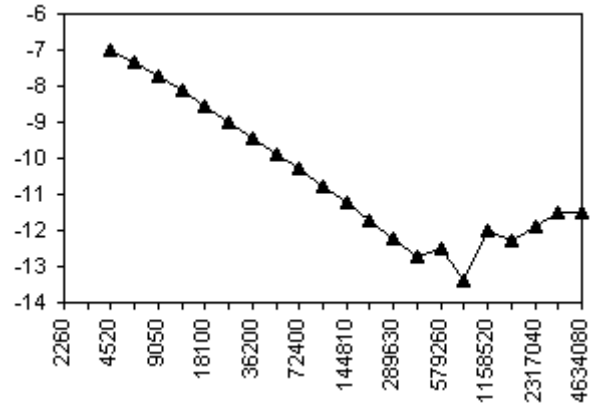
**Ordnungsdiagramm**



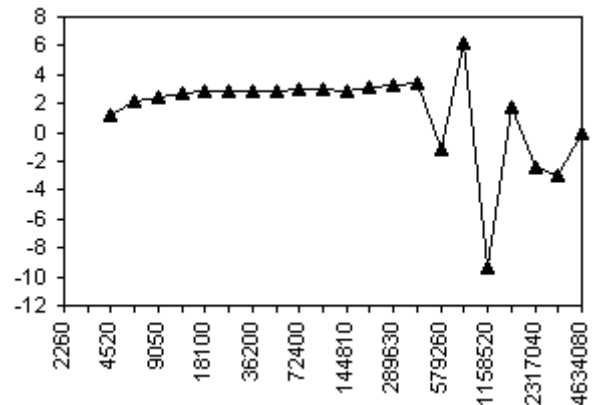
**Problem „Chemakzo“, Methode VNIL3**

Schritte	Fehler	Ordnung
	1	1
4520	-7.01	1.2
6400	-7.34	2.2
9050	-7.72	2.5
12800	-8.13	2.7
18100	-8.55	2.8
25600	-8.98	2.9
36200	-9.42	2.9
51200	-9.87	2.9
72400	-10.3	3.0
102400	-10.8	3.0
144810	-11.2	2.9
204800	-11.7	3.1
289630	-12.2	3.3
409600	-12.7	3.4
579260	-12.5	-1.2
819200	-13.4	6.2
1158520	-12.0	-9.4
1638400	-12.3	1.7
2317040	-11.9	-2.4
3276800	-11.5	-3.0
4634080	-11.5	0

**Fehlerdiagramm**



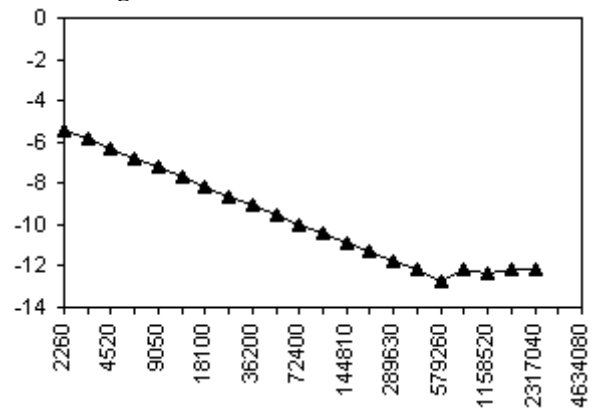
**Ordnungsdiagramm**



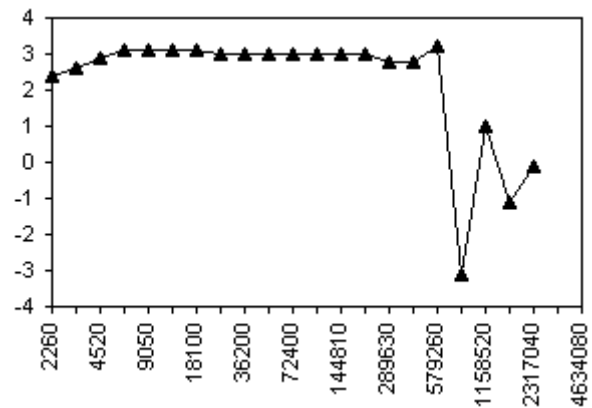
**Problem „Chemakzo“,  
Methode VSUPER2**

Schritte	Fehler	Ordnung
	1	1
2260	-5.46	2.4
3200	-5.85	2.6
4520	-6.29	2.9
6400	-6.76	3.1
9050	-7.23	3.1
12800	-7.69	3.1
18100	-8.15	3.1
25600	-8.61	3.0
36200	-9.06	3.0
51200	-9.52	3.0
72400	-9.97	3.0
102400	-10.4	3.0
144810	-10.9	3.0
204800	-11.3	3.0
289630	-11.8	2.8
409600	-12.2	2.8
579260	-12.7	3.2
819200	-12.2	-3.1
1158520	-12.3	1.0
1638400	-12.2	-1.1
2317040	-12.2	-0.1

**Fehlerdiagramm**



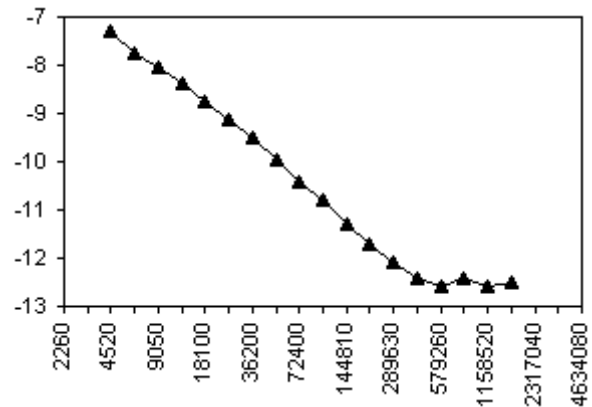
**Ordnungsdiagramm**



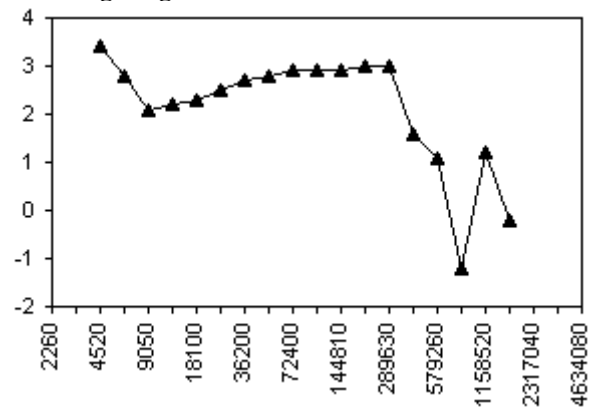
**Problem „Chemakzo“,  
Methode VSUPER3**

Schritte	Fehler	Ordnung
	1	1
4520	-7.30	3.4
6400	-7.73	2.8
9050	-8.05	2.1
12800	-8.38	2.2
18100	-8.73	2.3
25600	-9.11	2.5
36200	-9.52	2.7
51200	-9.94	2.8
72400	-10.4	2.9
102400	-10.8	2.9
144810	-11.3	2.9
204800	-11.7	3.0
289630	-12.1	3.0
409600	-12.4	1.6
579260	-12.6	1.1
819200	-12.4	-1.2
1158520	-12.6	1.2
1638400	-12.5	-0.2

**Fehlerdiagramm**



**Ordnungsdiagramm**

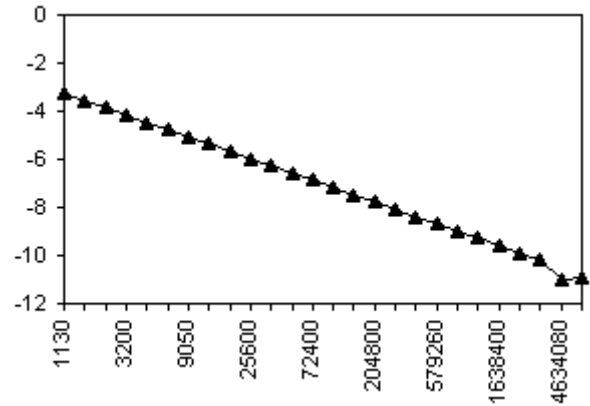


—▲— Index-1

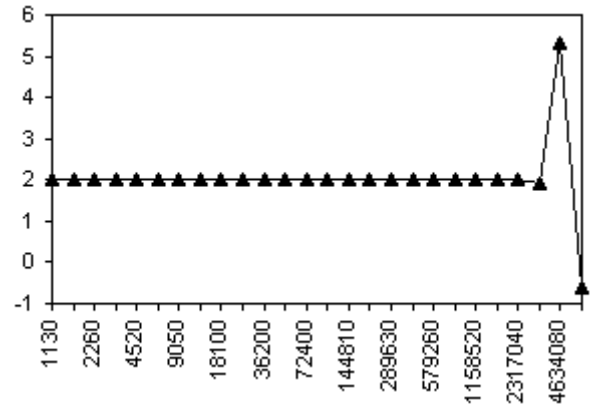
**Problem „Transamp“, Methode VNIL2**

Schritte	Fehler	Ordnung
	1	1
1130	-3.26	2.0
1600	-3.56	2.0
2260	-3.86	2.0
3200	-4.16	2.0
4520	-4.46	2.0
6400	-4.76	2.0
9050	-5.07	2.0
12800	-5.37	2.0
18100	-5.67	2.0
25600	-5.97	2.0
36200	-6.27	2.0
51200	-6.57	2.0
72400	-6.87	2.0
102400	-7.17	2.0
144810	-7.48	2.0
204800	-7.78	2.0
289630	-8.08	2.0
409600	-8.38	2.0
579260	-8.68	2.0
819200	-8.98	2.0
1158520	-9.28	2.0
1638400	-9.58	2.0
2317040	-9.88	2.0
3276800	-10.17	1.9
4634080	-10.97	5.3
6553600	-10.88	-0.6

**Fehlerdiagramm**



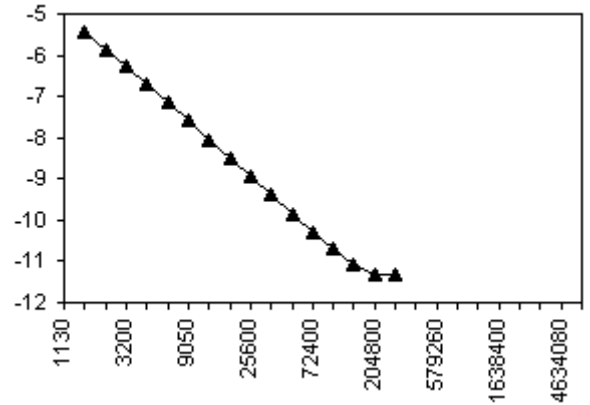
**Ordnungsdiagramm**



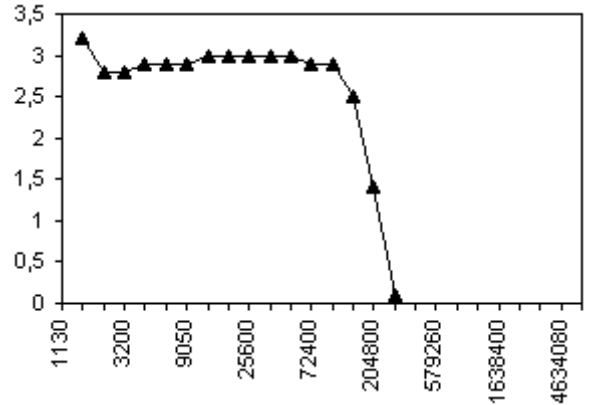
**Problem „Transamp“, Methode VNIL3**

Schritte	Fehler	Ordnung
	1	1
1600	-5.44	3.2
2260	-5.86	2.8
3200	-6.28	2.8
4520	-6.72	2.9
6400	-7.16	2.9
9050	-7.60	2.9
12800	-8.05	3.0
18100	-8.49	3.0
25600	-8.94	3.0
36200	-9.39	3.0
51200	-9.84	3.0
72400	-10.28	2.9
102400	-10.71	2.9
144810	-11.09	2.5
204800	-11.30	1.4
289630	-11.32	0.1

**Fehlerdiagramm**



**Ordnungsdiagramm**

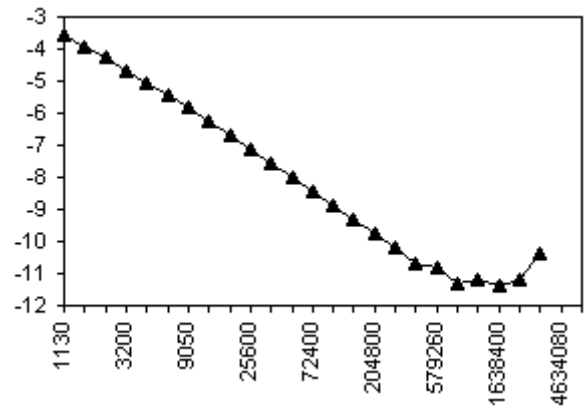


—▲— Index-1

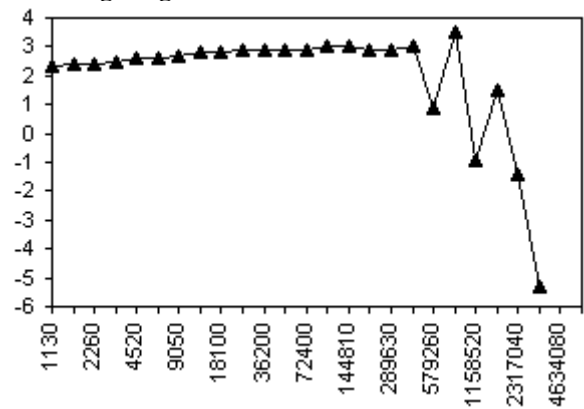
**Problem „Transamp“,  
Methode VSUPER2**

Schritte	Fehler	Ordnung
	1	1
1130	-3.56	2.3
1600	-3.92	2.4
2260	-4.28	2.4
3200	-4.66	2.5
4520	-5.04	2.6
6400	-5.44	2.6
9050	-5.84	2.7
12800	-6.26	2.8
18100	-6.68	2.8
25600	-7.11	2.9
36200	-7.55	2.9
51200	-7.99	2.9
72400	-8.44	2.9
102400	-8.88	3.0
144810	-9.33	3.0
204800	-9.77	2.9
289630	-10.20	2.9
409600	-10.66	3.0
579260	-10.79	0.9
819200	-11.31	3.5
1158520	-11.18	-0.9
1638400	-11.40	1.5
2317040	-11.20	-1.4
3276800	-10.39	-5.3

**Fehlerdiagramm**



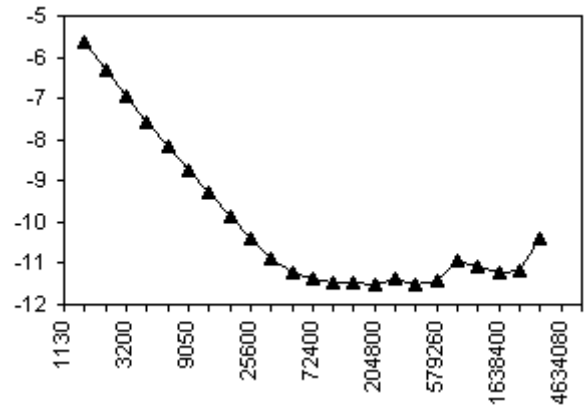
**Ordnungsdiagramm**



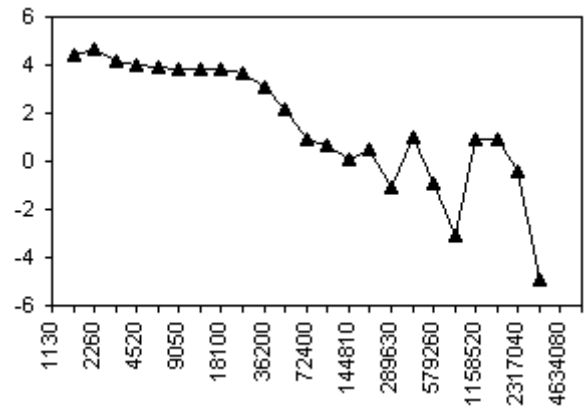
**Problem „Transamp“,  
Methode VSUPER3**

Schritte	Fehler	Ordnung
	1	1
1600	-5.62	4.4
2260	-6.33	4.7
3200	-6.96	4.2
4520	-7.56	4.0
6400	-8.14	3.9
9050	-8.72	3.8
12800	-9.29	3.8
18100	-9.86	3.8
25600	-10.41	3.7
36200	-10.88	3.1
51200	-11.21	2.2
72400	-11.35	0.9
102400	-11.45	0.7
144810	-11.46	0.1
204800	-11.53	0.5
289630	-11.37	-1.1
409600	-11.53	1.0
579260	-11.40	-0.9
819200	-10.93	-3.1
1158520	-11.07	0.9
1638400	-11.21	0.9
2317040	-11.15	-0.4
3276800	-10.41	-4.9

**Fehlerdiagramm**



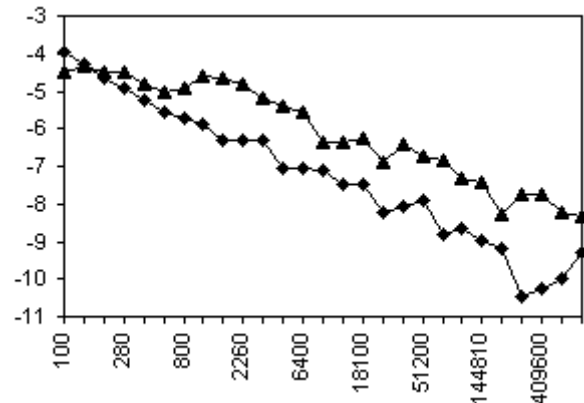
**Ordnungsdiagramm**



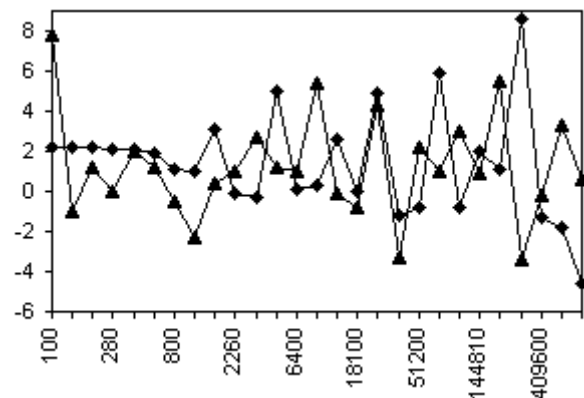
**Problem „Water“, Methode VNIL2  
(Gleiche Ergebnisse bei Berechnung durch  
explizite und implizite Beschreibung)**

Schritte	Fehler		Ordnung	
	1	2	1	2
93	-4.23	-3.91	-	-
100	-4.47	-3.98	7.8	2.2
140	-4.33	-4.30	-1.0	2.2
200	-4.51	-4.63	1.2	2.2
280	-4.51	-4.94	0	2.1
400	-4.81	-5.26	2.0	2.1
560	-5.00	-5.54	1.2	1.9
800	-4.92	-5.71	-0.5	1.1
1130	-4.58	-5.86	-2.3	1.0
1600	-4.65	-6.33	0.4	3.1
2260	-4.80	-6.32	1.0	-0.1
3200	-5.21	-6.29	2.7	-0.3
4520	-5.39	-7.03	1.2	5.0
6400	-5.55	-7.05	1.0	0.1
9050	-6.36	-7.09	5.4	0.3
12800	-6.35	-7.49	-0.1	2.6
18100	-6.23	-7.50	-0.8	0
25600	-6.88	-8.24	4.3	4.9
36200	-6.39	-8.05	-3.3	-1.2
51200	-6.71	-7.93	2.2	-0.8
72400	-6.86	-8.81	1.0	5.9
102400	-7.31	-8.68	3.0	-0.8
144810	-7.44	-8.99	0.9	2.0
204800	-8.27	-9.16	5.5	1.1
289630	-7.76	-10.45	-3.4	8.6
409600	-7.73	-10.26	-0.2	-1.3
579260	-8.23	-9.99	3.3	-1.8
819200	-8.32	-9.30	0.6	-4.6

Fehlerdiagramm für Gesamtintervall



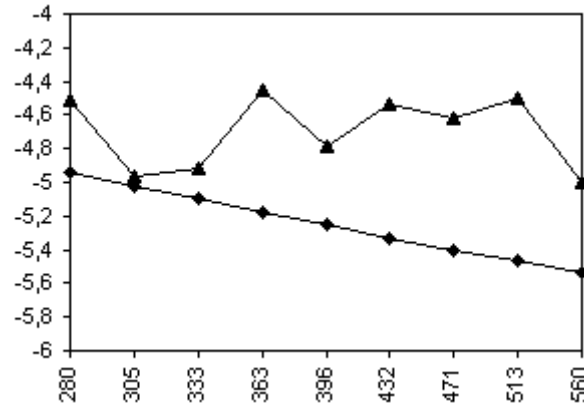
Ordnungsdiagramm für Gesamtintervall



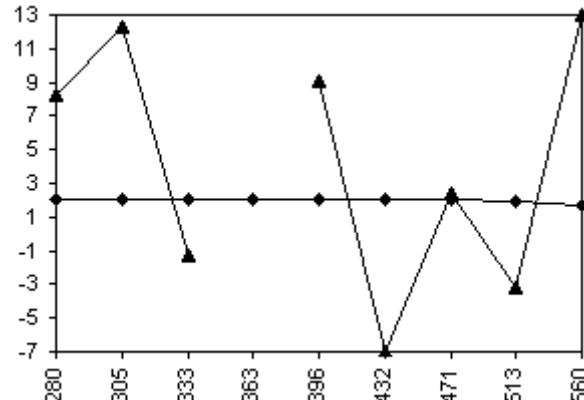
**Verfeinerung des Intervalls [280, 560]**

Schritte	Fehler		Ordnung	
	1	2	1	2
257	-4.21	-4.87	-	-
280	-4.51	-4.94	8.2	2.1
305	-4.97	-5.02	12.3	2.1
333	-4.92	-5.10	-1.3	2.1
363	-4.45	-5.18	-	2.0
396	-4.79	-5.25	9.1	2.0
432	-4.53	-5.33	-7.0	2.0
471	-4.62	-5.40	2.4	2.0
513	-4.50	-5.47	-3.2	1.9
560	-5.00	-5.54	13.0	1.7

Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



▲ Index-1    ◆ Index-2

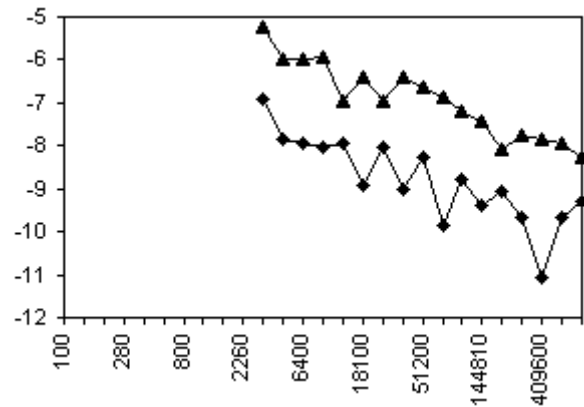
**Problem „Water“, Methode VNIL3  
(Gleiche Ergebnisse bei Berechnung durch  
explizite und implizite Beschreibung)**

Schritte	Fehler		Ordnung	
	1	2	1	2
3200	-5.21	-6.89	2.2	9.7
4520	-5.99	-7.86	5.2	6.5
6400	-5.98	-7.95	0	0.6
9050	-5.91	-8.02	-0.5	0.5
12800	-6.94	-7.95	6.8	-0.5
18100	-6.39	-8.93	-3.6	6.5
25600	-6.98	-8.05	3.9	-5.8
36200	-6.40	-9.01	-3.9	6.4
51200	-6.61	-8.29	1.4	-4.8
72400	-6.87	-9.86	1.7	10.4
102400	-7.21	-8.78	2.3	-7.2
144810	-7.42	-9.40	1.4	4.1
204800	-8.08	-9.08	4.3	-2.1
289630	-7.73	-9.68	-2.3	3.9
409600	-7.83	-11.05	0.7	9.1
579260	-7.95	-9.68	0.8	-9.1
819200	-8.27	-9.31	2.1	-2.4

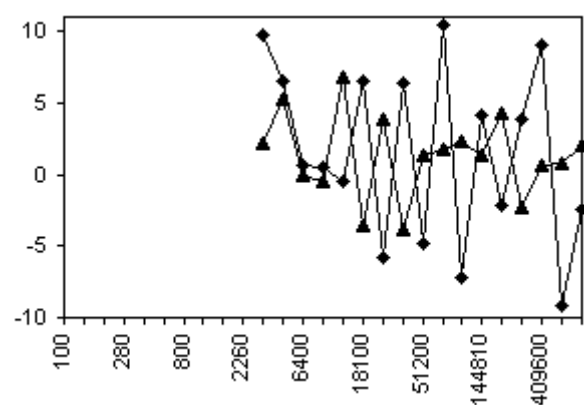
**Verfeinerung des Intervalls [3200, 6400]**

Schritte	Fehler		Ordnung	
	1	2	1	2
2934	-5.04	-6.62	-	-
3200	-5.21	-6.89	4.5	7.0
3490	-5.39	-6.94	4.9	1.3
3805	-5.28	-7.18	-2.9	6.4
4150	-5.51	-8.34	6.0	30.9
4525	-5.45	-7.33	-1.6	-
4935	-5.77	-7.43	8.4	2.7
5382	-6.21	-7.01	11.7	-
5869	-5.74	-7.91	-	24.1
6400	-5.98	-7.95	6.5	1.1

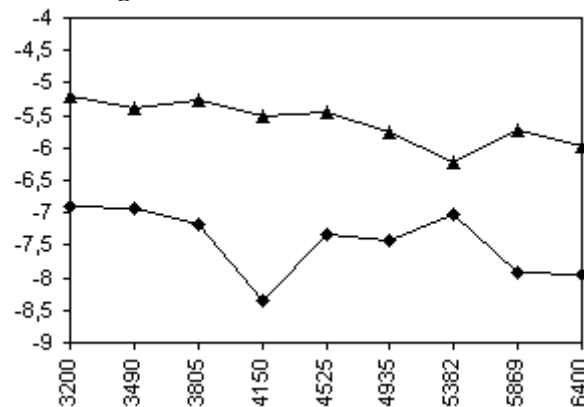
Fehlerdiagramm für Gesamtintervall



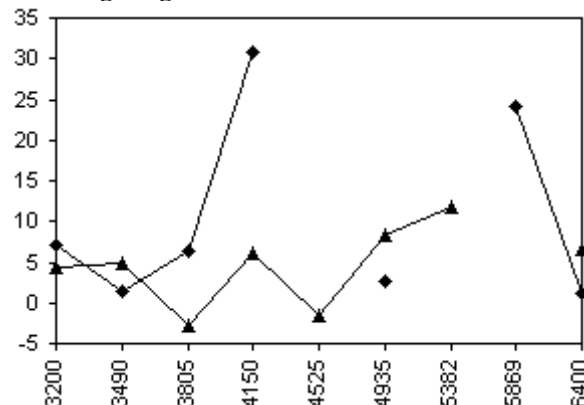
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



▲ Index-1    ◆ Index-2

**Problem „Water“, Methode VSUPER2  
(Gleiche Ergebnisse bei Berechnung durch  
explizite und implizite Beschreibung)**

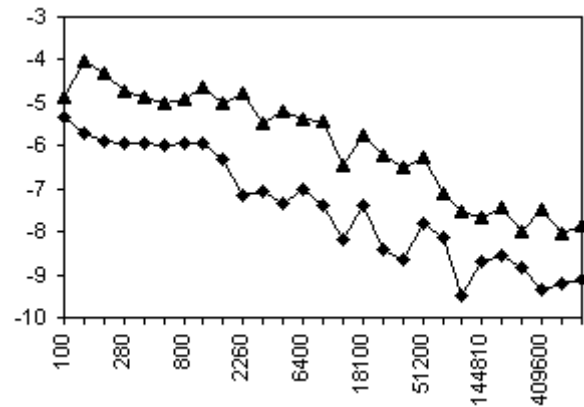
Schritte	Fehler		Ordnung	
	1	2	1	2
93	-4.74	-5.24	-	-
100	-4.86	-5.32	3.9	2.8
140	-4.04	-5.70	-5.6	2.6
200	-4.32	-5.90	1.8	1.3
280	-4.73	-5.96	2.8	0.4
400	-4.86	-5.95	0.9	-0.1
560	-5.01	-5.97	1.0	0.1
800	-4.89	-5.92	-0.8	-0.3
1130	-4.65	-5.95	-1.6	0.2
1600	-4.99	-6.31	2.3	2.3
2260	-4.77	-7.17	-1.4	5.7
3200	-5.45	-7.07	4.5	-0.6
4520	-5.18	-7.32	-1.8	1.7
6400	-5.37	-6.99	1.3	-2.2
9050	-5.44	-7.39	0.5	2.7
12800	-6.46	-8.17	6.8	5.2
18100	-5.75	-7.40	-4.7	-5.1
25600	-6.20	-8.41	3.0	6.7
36200	-6.49	-8.63	2.0	1.5
51200	-6.27	-7.82	-1.5	-5.3
72400	-7.10	-8.15	5.5	2.2
102400	-7.53	-9.50	2.9	9.0
144810	-7.66	-8.67	0.8	-5.6
204800	-7.45	-8.56	-1.4	-0.7
289630	-7.99	-8.85	3.6	1.9
409600	-7.48	-9.35	-3.4	3.3
579260	-8.02	-9.19	3.6	-1.1
819200	-7.85	-9.10	-1.1	-0.6

**Verfeinerung des Intervalls [25600, 36200]**

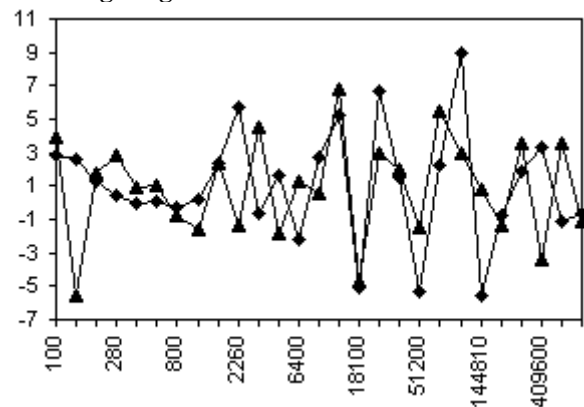
Schritte	Fehler		Ordnung	
	1	2	1	2
23475	-6.04	-7.46	-	-
25600	-6.20	-8.41	4.1	25.1
27917	-6.17	-7.89	-0.9	-
30444	-6.81	-8.09	17.2	5.3
33199	-6.50	-7.80	-8.4	-7.7
36204	-6.11	-8.30	-	13.2

▲ Index-1    ◆ Index-2

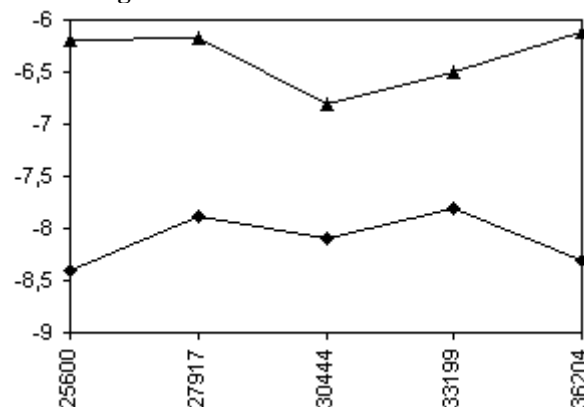
Fehlerdiagramm für Gesamtintervall



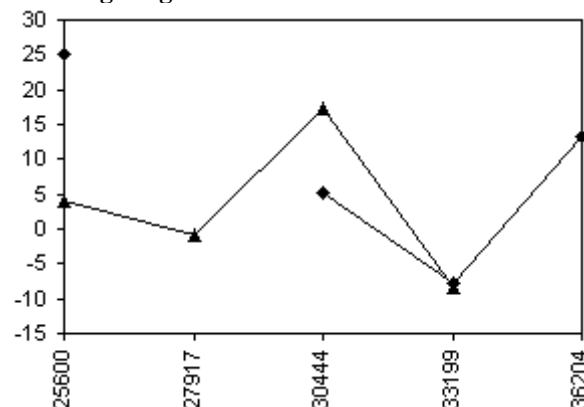
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



**Problem „Water“, Methode VSUPER3  
(Gleiche Ergebnisse bei Berechnung durch  
explizite und implizite Beschreibung)**

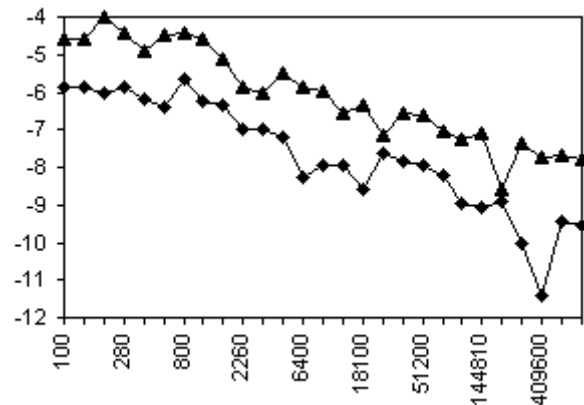
Schritte	Fehler		Ordnung	
	1	2	1	2
93	-4.43	-5.83	-	-
100	-4.59	-5.86	5.3	1.0
140	-4.58	-5.86	-0.1	0
200	-4.01	-6.00	-3.7	0.9
280	-4.41	-5.88	2.7	-0.8
400	-4.91	-6.16	3.3	1.8
560	-4.50	-6.39	-2.8	1.5
800	-4.44	-5.65	-0.4	-4.8
1130	-4.59	-6.26	1.0	4.1
1600	-5.10	-6.32	3.4	0.4
2260	-5.86	-6.96	5.1	4.3
3200	-6.05	-6.98	1.2	0.2
4520	-5.49	-7.22	-3.8	1.6
6400	-5.84	-8.29	2.3	7.1
9050	-5.95	-7.94	0.8	-2.3
12800	-6.57	-7.92	4.1	-0.1
18100	-6.33	-8.57	-1.6	4.3
25600	-7.16	-7.60	5.5	-6.5
36200	-6.57	-7.85	-3.9	1.7
51200	-6.61	-7.95	0.2	0.6
72400	-7.06	-8.21	3.0	1.7
102400	-7.26	-8.96	1.3	5.0
144810	-7.08	-9.07	-1.2	0.7
204800	-8.56	-8.90	9.9	-1.1
289630	-7.37	-10.04	-8.0	7.6
409600	-7.72	-11.44	2.3	9.3
579260	-7.66	-9.44	-0.4	-
819200	-7.78	-9.53	0.8	0.6

**Verfeinerung des Intervalls [1130, 2260]**

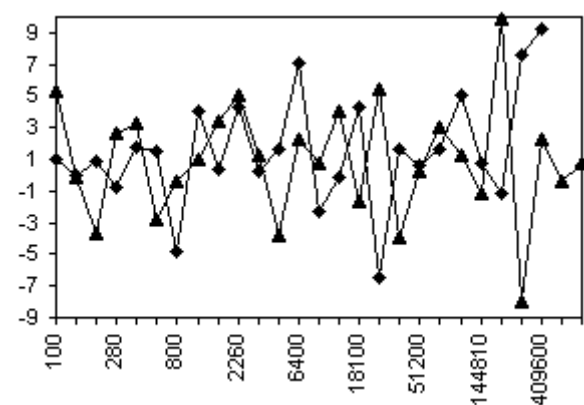
Schritte	Fehler		Ordnung	
	1	2	1	2
1036	-4.48	-6.72	-	-
1130	-4.59	-6.26	2.9	-
1232	-4.46	-6.16	-3.4	-2.7
1344	-5.03	-6.15	15.1	-0.3
1465	-5.18	-6.08	4.0	-1.8
1598	-5.32	-6.17	3.6	2.3
1743	-5.32	-6.63	-0.1	12.3
1900	-5.09	-6.94	-6.2	8.3
2072	-5.34	-7.16	6.8	5.7
2260	-5.86	-6.96	13.8	-5.3

▲ Index-1    ◆ Index-2

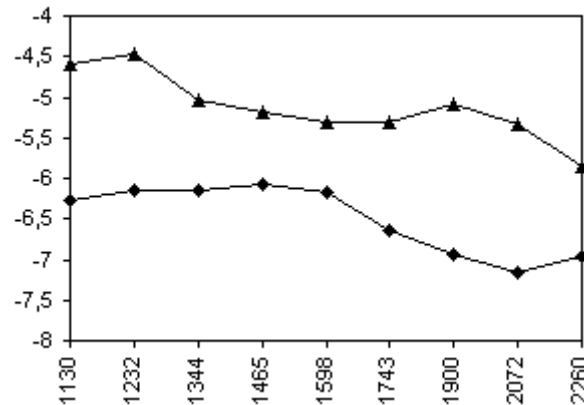
Fehlerdiagramm für Gesamtintervall



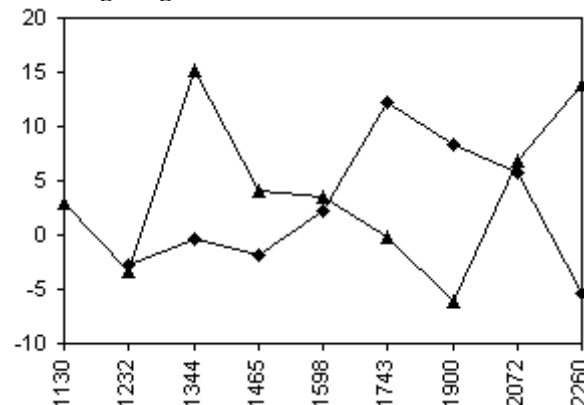
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



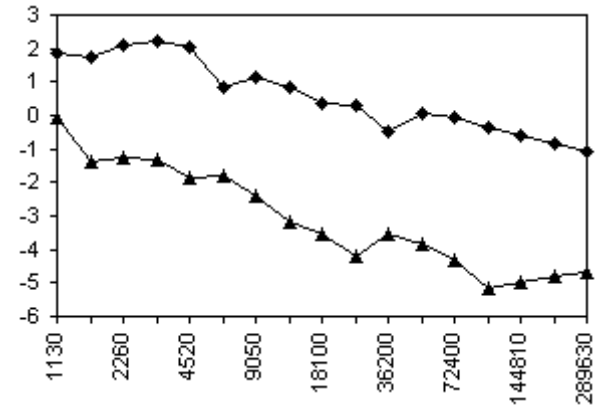
**Problem „Crank“, Methode VNIL2  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler		Ordnung	
	1	2	1	2
1130	-0.09	1.85	6.6	3.0
1600	-1.36	1.77	8.4	0.5
2260	-1.25	2.07	-0.7	-2.0
3200	-1.33	2.21	0.6	-0.9
4520	-1.84	2.04	3.4	1.2
6400	-1.82	0.84	-0.2	7.9
9050	-2.37	1.14	3.7	-2.0
12800	-3.21	0.82	5.6	2.1
18100	-3.55	0.33	2.3	3.2
25600	-4.18	0.30	4.1	0.2
36200	-3.54	-0.51	-4.2	5.4
51200	-3.85	0.04	2.0	-3.7
72400	-4.33	-0.09	3.2	0.9
102400	-5.17	-0.33	5.5	1.6
144810	-5.01	-0.59	-1.0	1.8
204800	-4.78	-0.85	-1.6	1.7
289630	-4.70	-1.08	-0.5	1.5

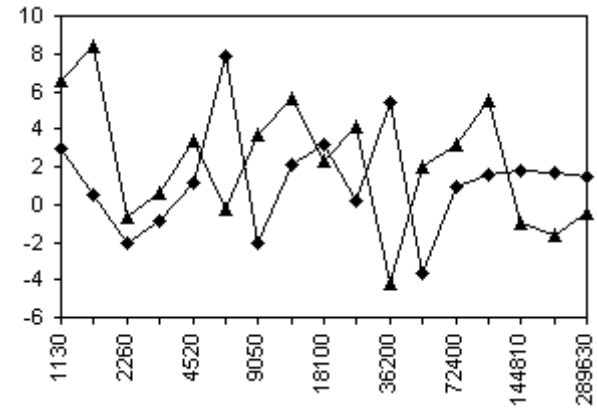
**Verfeinerung von [51200, 102400]**

Schritte	Fehler		Ordnung	
	1	2	1	2
46951	-3.75	0.02	-	-
51200	-3.85	0.04	2.6	-0.6
55834	-3.96	0.03	2.9	0.3
60887	-4.07	0.01	3.1	0.8
66398	-4.20	-0.04	3.3	1.1
72408	-4.33	-0.09	3.6	1.3
78961	-4.48	-0.14	4.0	1.5
86108	-4.66	-0.20	4.6	1.6
93901	-4.87	-0.26	5.6	1.7
102400	-5.17	-0.33	7.9	1.7

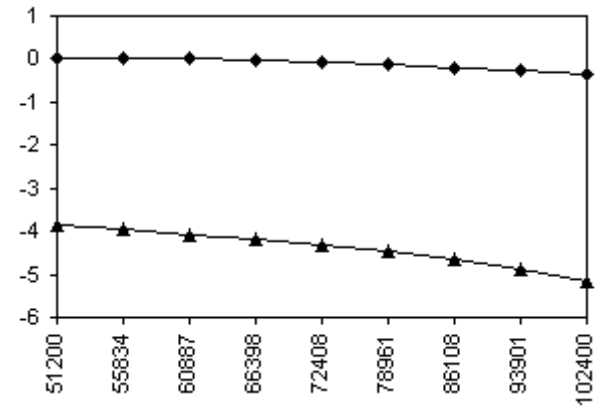
Fehlerdiagramm für Gesamtintervall



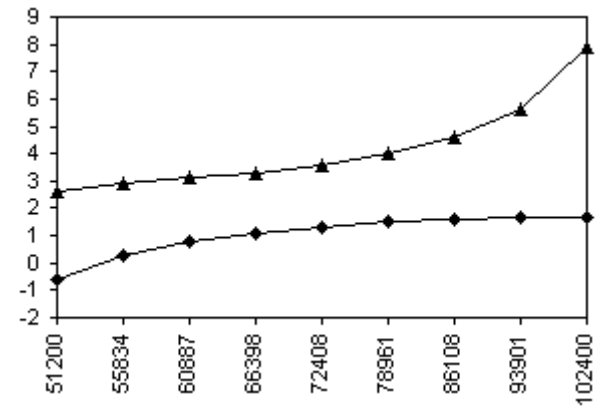
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



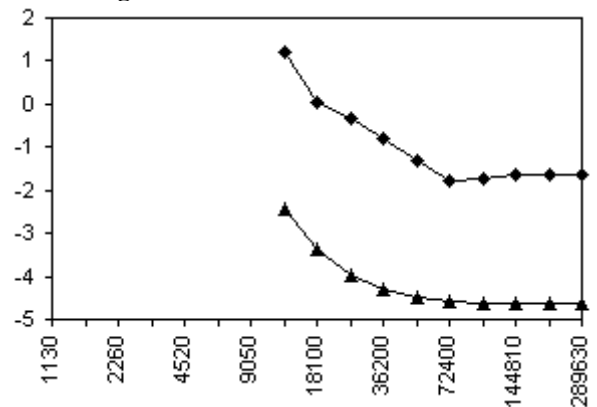
**Problem „Crank“, Methode VNIL3  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler		Ordnung	
	1	2	1	2
12800	-2.43	1.22	10.2	15.9
18100	-3.37	0.05	6.3	7.8
25600	-3.96	-0.33	3.9	2.5
36200	-4.32	-0.80	2.4	3.1
51200	-4.51	-1.30	1.3	3.3
72400	-4.59	-1.77	0.5	3.1
102400	-4.62	-1.73	0.2	-0.2
144810	-4.63	-1.66	0.1	-0.5
204800	-4.64	-1.63	0	-0.2
289630	-4.64	-1.62	0	-0.1

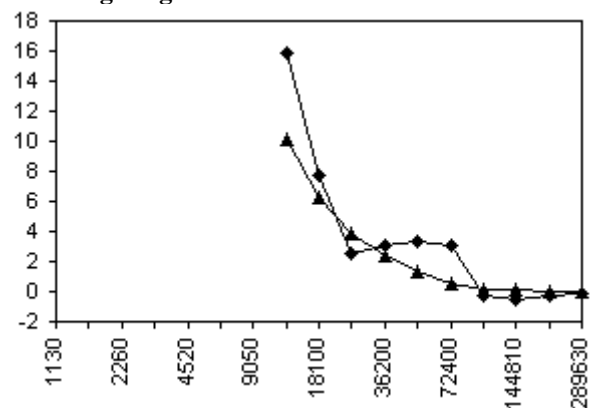
**Verfeinerung von [12800, 25600]**

Schritte	Fehler		Ordnung	
	1	2	1	2
11738	-2.11	1.76	-	-
12800	-2.43	1.22	8.3	14.5
13958	-2.71	0.70	7.5	13.6
15222	-2.96	0.30	6.7	10.8
16600	-3.18	0.13	5.9	4.5
18102	-3.37	0.04	5.1	2.3
19740	-3.55	-0.03	4.6	2.0
21527	-3.70	-0.13	4.0	2.7
23475	-3.84	-0.23	3.7	2.6
25600	-3.96	-0.33	3.3	2.7

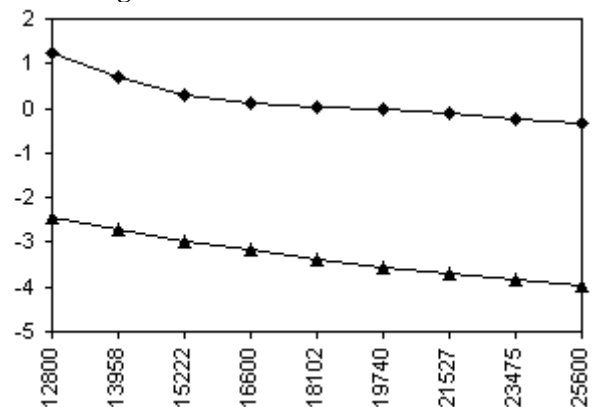
Fehlerdiagramm für Gesamtintervall



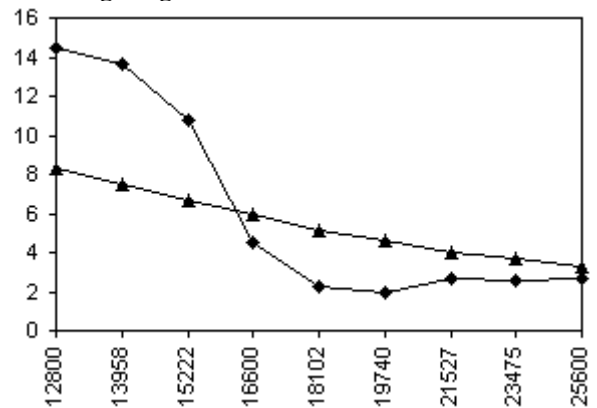
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



Ordnungsdiagramm für Teilintervall



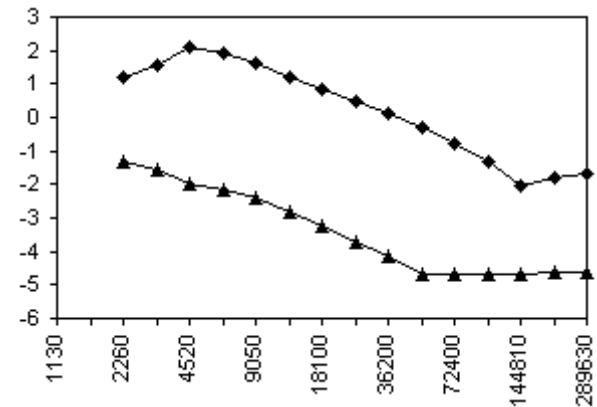
**Problem „Crank“, Methode VSUPER2  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler		Ordnung	
	1	2	1	2
2260	-1.31	1.21	-1.5	4.3
3200	-1.56	1.58	1.7	-2.4
4520	-1.99	2.09	2.8	-3.4
6400	-2.15	1.94	1.1	1.0
9050	-2.40	1.60	1.6	2.2
12800	-2.84	1.20	2.9	2.7
18100	-3.25	0.81	2.7	2.6
25600	-3.71	0.47	3.1	2.2
36200	-4.15	0.11	2.9	2.4
51200	-4.68	-0.31	3.5	2.8
72400	-4.68	-0.76	0	3.0
102400	-4.68	-1.33	0	3.8
144810	-4.65	-2.05	-0.2	4.8
204800	-4.64	-1.79	-0.1	-1.8
289630	-4.64	-1.67	0	-0.8

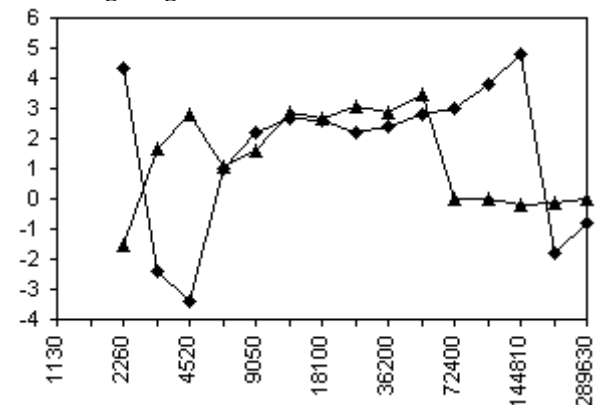
**Verfeinerung von [12800, 25600]**

Schritte	Fehler		Ordnung	
	1	2	1	2
11738	-2.68	1.31	-	-
12800	-2.84	1.20	4.2	2.8
13958	-2.66	1.10	-4.7	2.7
15222	-3.01	1.01	9.4	2.6
16600	-3.14	0.90	3.4	2.7
18102	-3.26	0.80	3.1	2.7
19740	-3.35	0.72	2.3	2.1
21527	-3.47	0.64	3.2	2.2
23475	-3.60	0.56	3.5	2.3
25600	-3.71	0.47	3.1	2.2

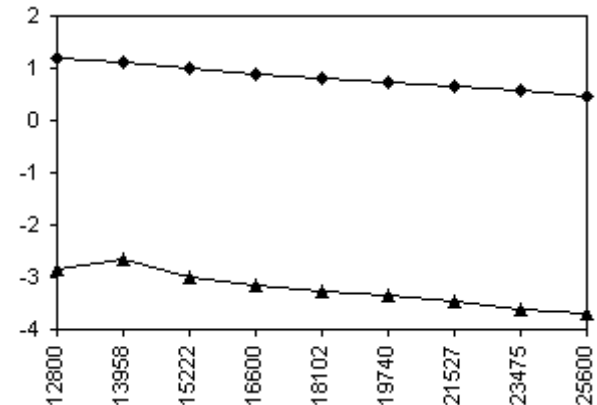
**Fehlerdiagramm für Gesamtintervall**



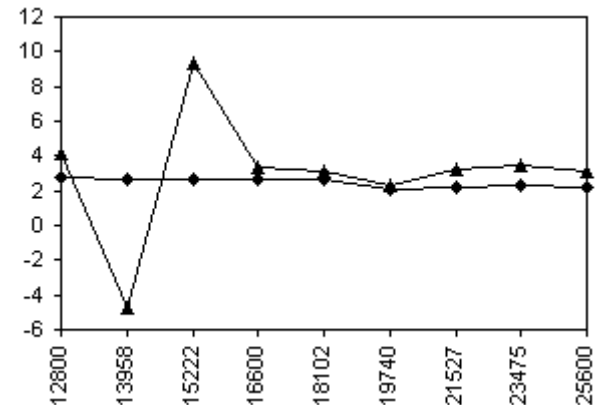
**Ordnungsdiagramm für Gesamtintervall**



**Fehlerdiagramm für Teilintervall**



**Ordnungsdiagramm für Teilintervall**



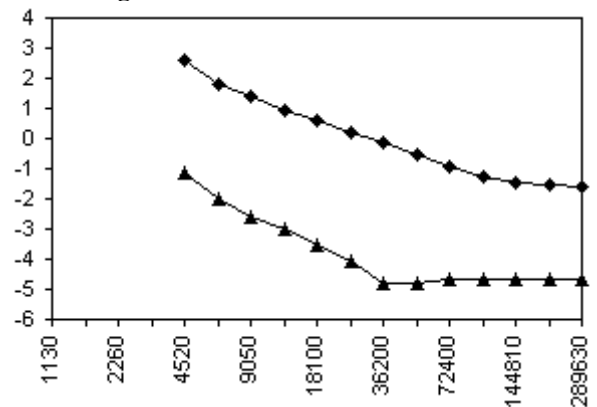
**Problem „Crank“, Methode VSUPER3  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler		Ordnung	
	1	2	1	2
4520	-1.14	2.63	9.9	4.2
6400	-1.97	1.83	5.5	5.3
9050	-2.58	1.37	4.0	3.0
12800	-3.01	0.91	2.9	3.1
18100	-3.50	0.58	3.2	2.2
25600	-4.07	0.22	3.8	2.4
36200	-4.80	-0.16	4.8	2.5
51200	-4.81	-0.55	0.1	2.6
72400	-4.68	-0.93	-0.9	2.5
102400	-4.65	-1.24	-0.2	2.1
144810	-4.64	-1.45	-0.1	1.4
204800	-4.64	-1.55	0	0.7
289630	-4.64	-1.59	0	0.3

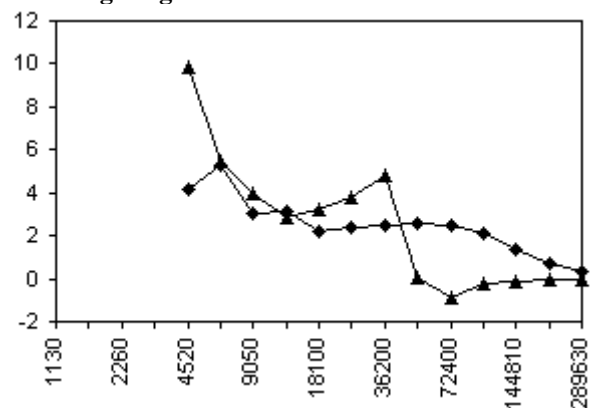
**Verfeinerung von [12800, 25600]**

Schritte	Fehler		Ordnung	
	1	2	1	2
11738	-2.92	1.01	-	-
12800	-3.01	0.91	2.5	2.5
13958	-3.12	0.83	2.8	2.3
15222	-3.24	0.75	3.2	2.1
16600	-3.37	0.67	3.3	2.2
18102	-3.50	0.58	3.5	2.2
19740	-3.63	0.50	3.6	2.3
21527	-3.77	0.41	3.8	2.3
23475	-3.92	0.32	3.9	2.4
25600	-4.07	0.22	4.0	2.5

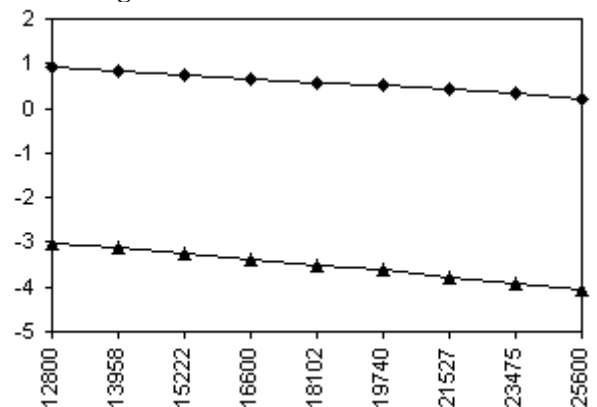
Fehlerdiagramm für Gesamtintervall



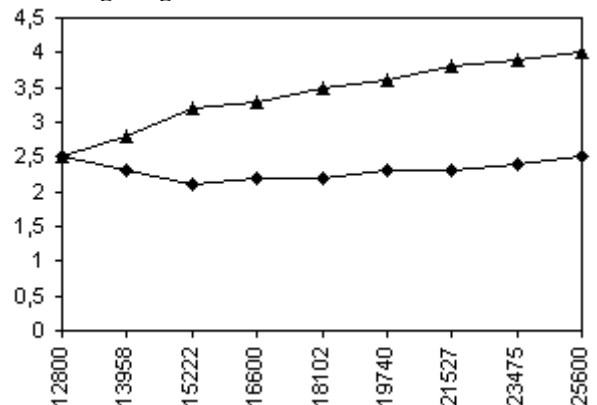
Ordnungsdiagramm für Gesamtintervall



Fehlerdiagramm für Teilintervall



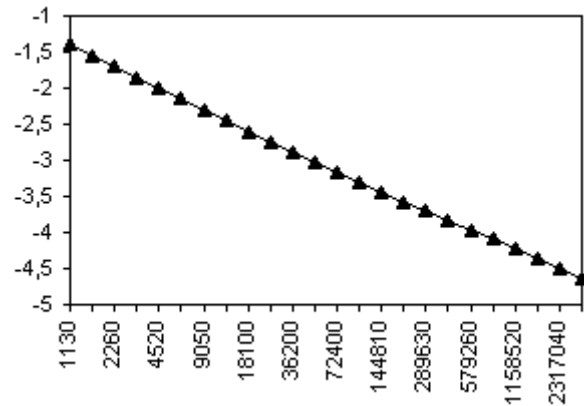
Ordnungsdiagramm für Teilintervall



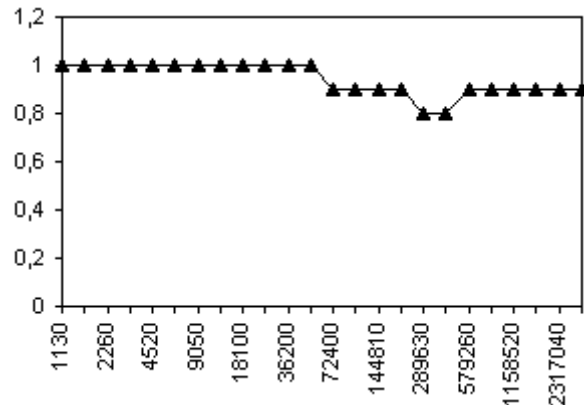
**Problem „NAND“, Methode VNIL2  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler	Ordnung
	1	1
800	-1.24	-
1130	-1.40	1.0
1600	-1.55	1.0
2260	-1.70	1.0
3200	-1.85	1.0
4520	-2.00	1.0
6400	-2.15	1.0
9050	-2.30	1.0
12800	-2.45	1.0
18100	-2.60	1.0
25600	-2.75	1.0
36200	-2.89	1.0
51200	-3.04	1.0
72400	-3.18	0.9
102400	-3.31	0.9
144810	-3.45	0.9
204800	-3.57	0.9
289630	-3.70	0.8
409600	-3.83	0.8
579260	-3.96	0.9
819200	-4.09	0.9
1158520	-4.22	0.9
1638400	-4.36	0.9
2317040	-4.50	0.9
3276800	-4.64	0.9

Fehlerdiagramm



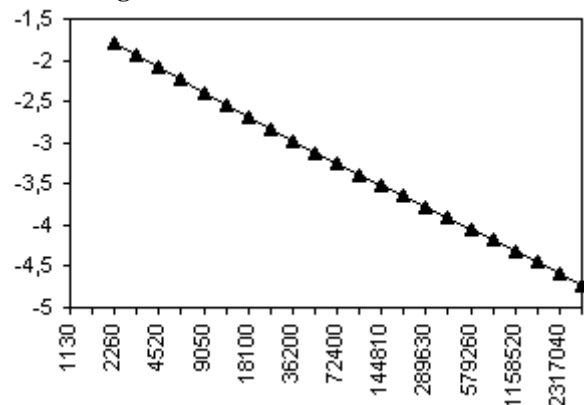
Ordnungsdiagramm



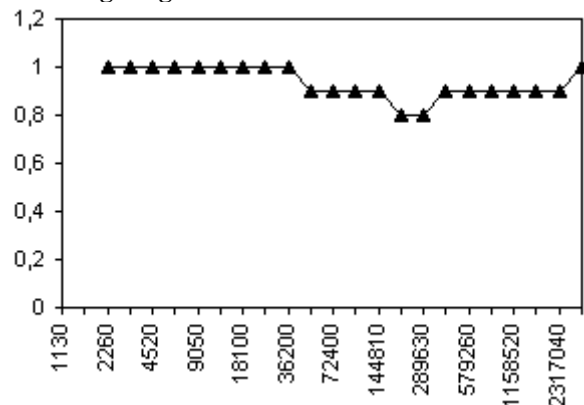
**Problem „NAND“, Methode VNIL3**

Schritte	Fehler	Ordnung
	1	1
1600	-1.64	-
2260	-1.79	1.0
3200	-1.94	1.0
4520	-2.09	1.0
6400	-2.24	1.0
9050	-2.39	1.0
12800	-2.54	1.0
18100	-2.69	1.0
25600	-2.83	1.0
36200	-2.98	1.0
51200	-3.12	0.9
72400	-3.26	0.9
102400	-3.39	0.9
144810	-3.52	0.9
204800	-3.65	0.8
289630	-3.78	0.8
409600	-3.91	0.9
579260	-4.04	0.9
819200	-4.17	0.9
1158520	-4.31	0.9
1638400	-4.45	0.9
2317040	-4.59	0.9
3276800	-4.73	1.0

Fehlerdiagramm



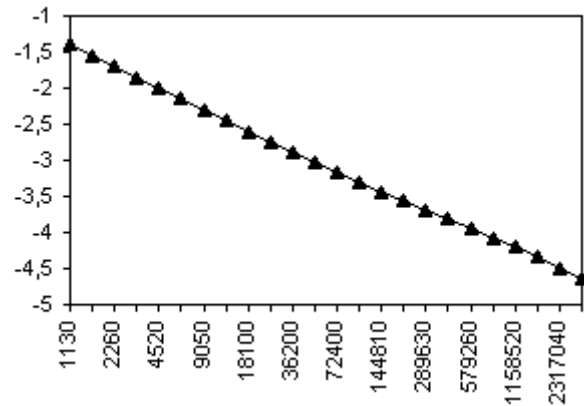
Ordnungsdiagramm



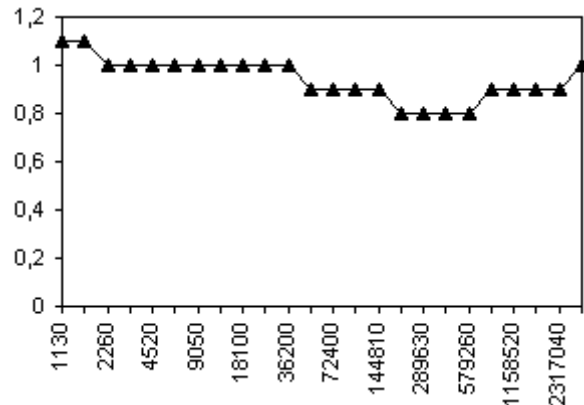
**Problem „NAND“, Methode VSUPER2  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler	Ordnung
	1	1
800	-1.22	-
1130	-1.38	1.1
1600	-1.55	1.1
2260	-1.70	1.0
3200	-1.85	1.0
4520	-2.00	1.0
6400	-2.15	1.0
9050	-2.30	1.0
12800	-2.45	1.0
18100	-2.60	1.0
25600	-2.75	1.0
36200	-2.89	1.0
51200	-3.04	0.9
72400	-3.17	0.9
102400	-3.31	0.9
144810	-3.44	0.9
204800	-3.56	0.8
289630	-3.69	0.8
409600	-3.81	0.8
579260	-3.94	0.8
819200	-4.07	0.9
1158520	-4.20	0.9
1638400	-4.34	0.9
2317040	-4.49	0.9
3276800	-4.63	1.0

Fehlerdiagramm



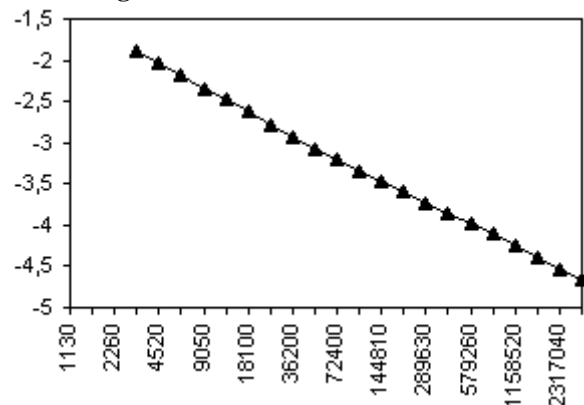
Ordnungsdiagramm



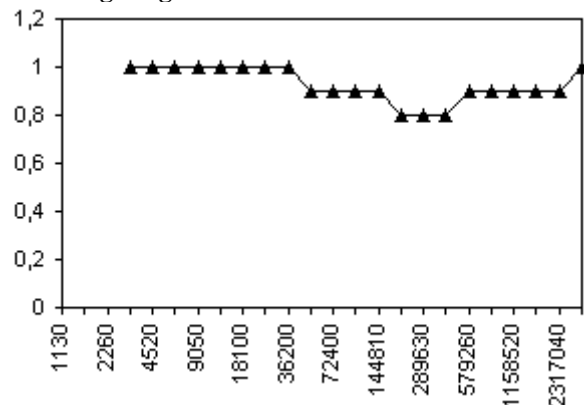
**Problem „NAND“, Methode VSUPER3  
(Implizite Rechnung,  $10^{-12}$ )**

Schritte	Fehler	Ordnung
	1	1
2260	-1.74	-
3200	-1.89	1.0
4520	-2.04	1.0
6400	-2.19	1.0
9050	-2.34	1.0
12800	-2.48	1.0
18100	-2.63	1.0
25600	-2.78	1.0
36200	-2.93	1.0
51200	-3.07	0.9
72400	-3.21	0.9
102400	-3.34	0.9
144810	-3.47	0.9
204800	-3.60	0.8
289630	-3.73	0.8
409600	-3.85	0.8
579260	-3.98	0.9
819200	-4.11	0.9
1158520	-4.25	0.9
1638400	-4.39	0.9
2317040	-4.53	0.9
3276800	-4.67	1.0

Fehlerdiagramm



Ordnungsdiagramm



## Literatur

- [AP98] Uri M. Ascher, Linda R. Petzold: „Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations”, Society for Industrial and Applied Mathematics, Philadelphia, 1998
- [CWI99] Walter M. Lioen, Jacques J.B. de Swart: „Test Set for Initial Value Problem Solvers”, Release 2.1 September 1999, <http://www.cwi.nl/cwi/projects/IVPtestset/>, <http://hilbert.dm.uniba.it/~testset/>
- [KPB85] P. Kaps, S. Poon, T.D. Bui: „Rosenbrock methods for stiff ODEs. A comparison of Richardson extrapolation and embedding technique”, Computing 34 (1985), 17–40
- [ODE00] H. Podhaisky, ...: „ODETEST - an environment for testing PTSW methods“, Quellcode in FORTRAN vorliegend, Versionsstand 22.11.2000
- [PSW99] H. Podhaisky, B.A. Schmitt, R. Weiner: „Two-step W-methods with parallel stages“, Report No. 22 (1999) of the Institute of Numerical Mathematics, Martin-Luther-Universität Halle-Wittenberg, Fachbereich für Mathematik und Informatik, Halle, 1999