Schwerpunktfach Physik und Angewandte Mathematik

Numerische Methoden: Approximation und Integration Programmierung TI-89

Bruggmann Roland

Luzern, Mai 2005

Maturitätsschule für Erwachsene MSE, Reussbühl

Diese Arbeit dokumentiert eine Implementation numerischer Methoden zur Approximation (Bisection, Regula falsi und Newton) und Integration (Rechteck-, Trapez-, Tangenten- und Simpson-Verfahren) für den Taschenrechner TI-89.

Als Programmiersprache kommt TI-BASIC zum Einsatz, den Quellcode zum Programm «nummeth() – Numerische Methoden» wurde auf einer Intel-x86-Maschine mit GNU/Linux als Betriebssystem und dem Editor «Kate» erstellt. Um das Programm auf den TI-89 zu laden, wurde die Software «TiLP» verwendet.

Als Nachschlagwerke und Grundlage zur Programmierung dienten ausschliesslich das dem TI-89 von Texas Instruments mitgelieferte Benutzerhandbuch zur Advanced Mathematics Software Version 2.0 [1] in gedruckter Form sowie die Hinweise zur Version 2.8 [2] als PDF.

Inhaltsverzeichnis

1. Bemerkungen zu Beginn

1.1. Allgemeines

Kenntnisse zur Bedienung des TI-89 werden voraussgesetzt und können z.B. im Benutzerhandbuch [1] nachgeschlagen werden, das zum Rechner mitgeliefert wurde.

Im Programm `nummeth` kann mit `ESC` jeweils zum vorherigen Fenster gewechselt werden. Numerische Eingaben werden jeweils auf ihre Gültigkeit geprüft. Bei einer ungültigen Eingabe erscheint eine Fehlermeldung. Danach besteht die Möglichkeit, die Eingabe zu korrigieren (siehe Abbildung 1.1).

1.2. Erfassen von Funktionen

Vor dem Programmstart

Funktionen können im Y-Editor eingegeben und im Bildschirm `Graph` zur Wahl des Intervalls untersucht werden.

Die im Y-Editor eingegebenen Funktionen `y1(x)` bis `y10(x)` stehen dem Programm `nummeth` zur Verfügung, können darin also optional ausgewählt werden – sowohl bei einer Approximation als auch bei einer Integration.

Nach dem Programmstart

Eine Funktion kann auch erst nach dem Programmstart mit den anderen Parametern (Intervall, Iterationen etc.) erfasst werden – dies wird in den folgende Kapitel erläutert.

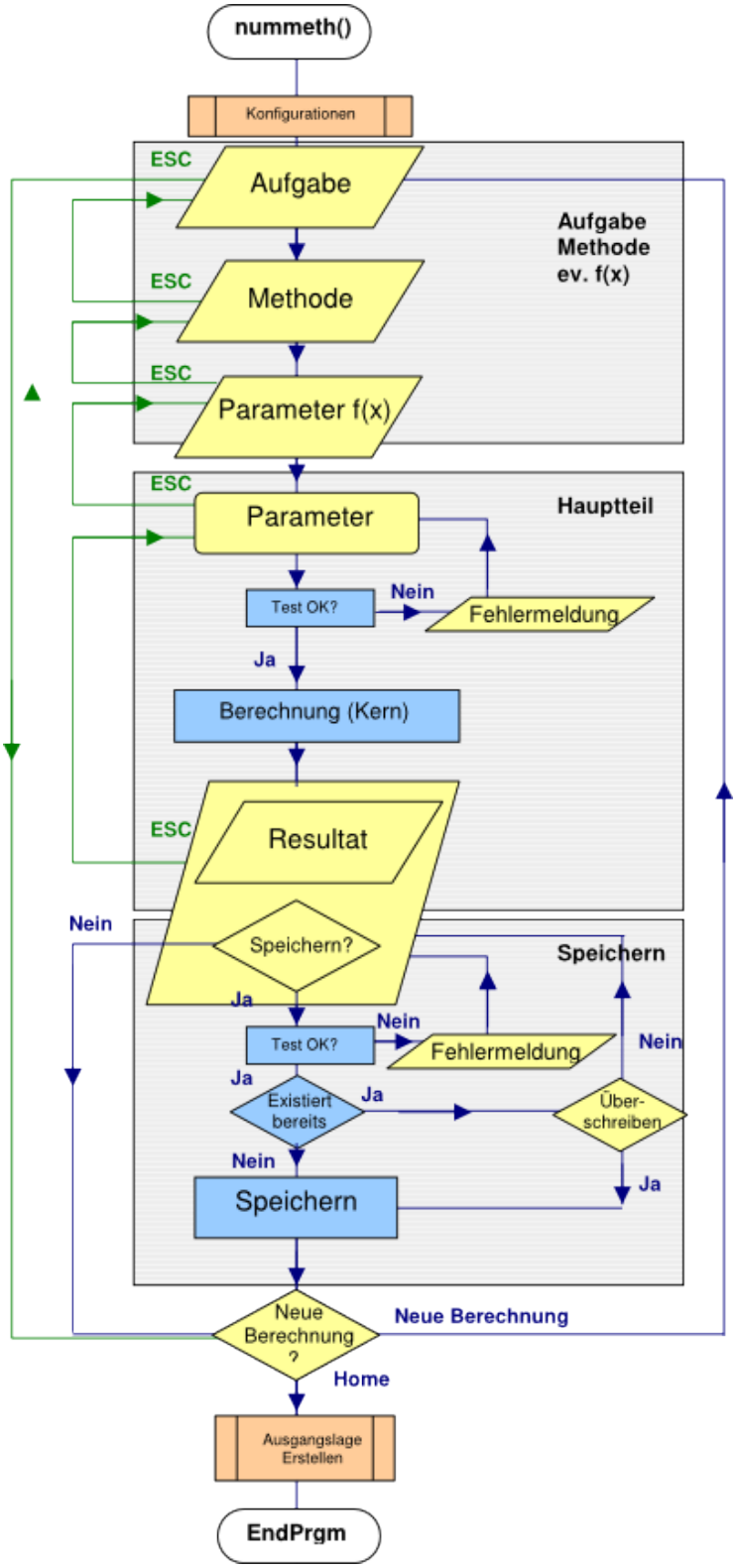

Abbildung 1.1.: Flussdiagramm zu nummeth

2. Das Programm

2.1. Programm starten

Mit dem Start des Programms werden einige Konfigurationen ausgelöst: Das Programm wird auf Fehler überprüft, aktuelles Verzeichnis wird nach Möglichkeit `NUMERIK`, aktuelle Einstellungen der Modi unter `MODE` werden gesichert und für die Dauer des Programmablaufs konfiguriert, eine Parameterliste wird erstellt und der Y-Editor wird auf erfasste Funktionen untersucht. Dies kann beim erstmaligen Gebrauch des Programms einige Sekunden dauern.

Start mit VAR-LINK

Mit `2nd` ▶ `VAR-LINK` kann der Variablen-Bildschirm aufgerufen werden. Im Folder `NUMERIK` ist die Programmvariable `nummeth` zu finden (siehe Abbildung 2.1). Diese ist mit dem Cursor zu markieren und mit der Taste `ENTER` zu bestätigen.

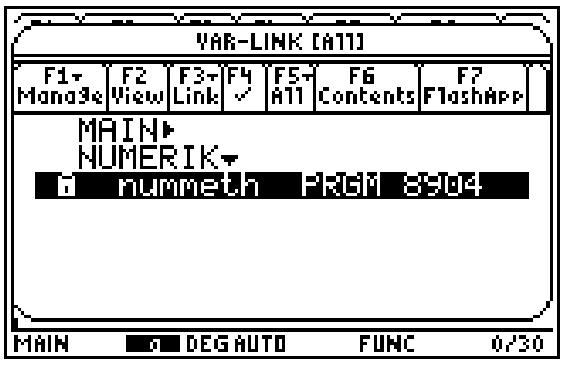

Abbildung 2.1.: Variablen-Bildschirm mit Ordner NUMERIK und Programmvariable nummeth

Der Pfad der Programmvariablen ist somit in die Eingabezeile kopiert worden. Dieser muss nun noch mit einer schliessenden runden Klammer ’) ’ ergänzt werden (siehe Abbildung 2.2). Mit ENTER wird das Programm gestartet.

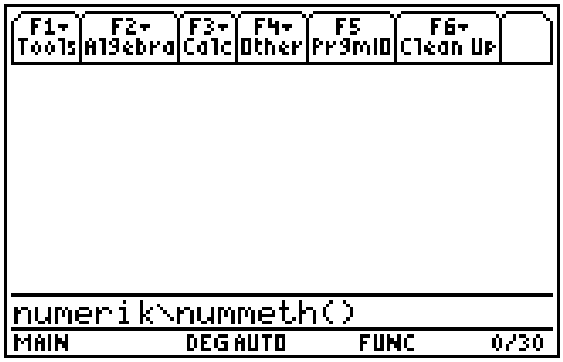

Abbildung 2.2.: Eingabezeile mit kopiertem Pfad

Start mit CATALOG

Mit CATALOG ▶ F4 User-Defined werden gespeicherte Funktions- und Programmvariablen angezeigt. Mit dem Cursor den Pfeil zu nummeth(bewegen (siehe Abbildung 2.3) und die Taste ENTER betätigen.

Der Pfad der Programmvariablen ist somit in die Eingabezeile kopiert worden. Dieser muss nun noch mit einer schliessenden runden Klammer ’) ’ ergänzt werden (siehe Abbildung 2.2). Mit ENTER wird das Programm gestartet.

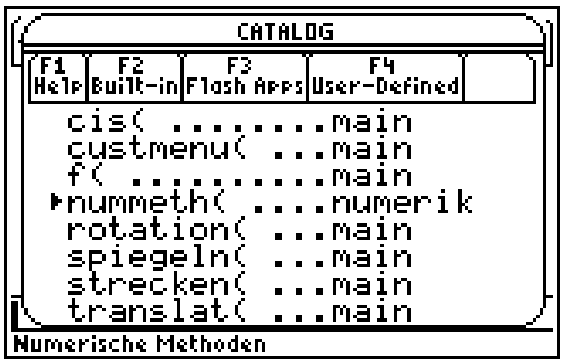

Abbildung 2.3.: Variablen-Katalog mit Eintrag nummeth

2.2. Aufgabe, Methode, Parameter f(x) (optional)

Aufgabe

Die Wahl der Aufgabe erfolgt per Cursor im Drop-Down-Menu, bestätigen mit ENTER (siehe Abbildung 2.4).

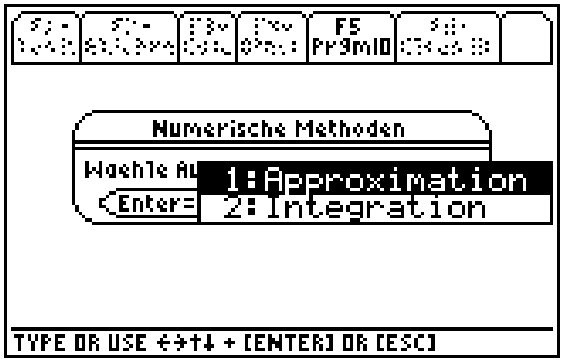

Abbildung 2.4.: Dialog Numerische Methoden mit Dropdown zur Wahl der Aufgabe

Methode

Die Wahl der Methode erfolgt per Cursor im Drop-Down-Menu, bestätigen mit ENTER (siehe Abbildung 2.5).

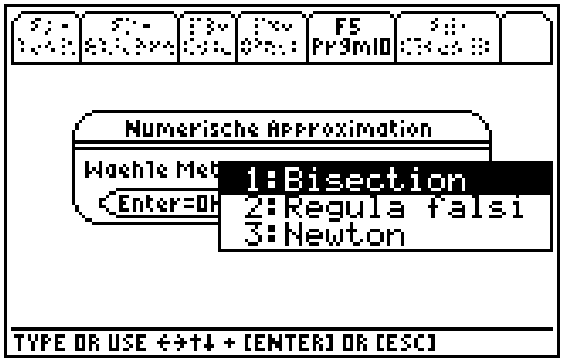

Abbildung 2.5.: Dialog Numerische Approximation mit Dropdown zur Wahl der Methode

Parameter f(x)

Sind im Y-Editor Funktionen erfasst worden, erscheint nach der Wahl einer Methode ein Dialog 'Parameter' mit der Möglichkeit, eine der erfassten Funktionen zur weiteren Bearbeitung auszuwählen. Dies erfolgt im Drop-Down-Menu 'Funktion aus dem Y-Editor uebernehmen?' durch die Wahl des Eintrages 'Ja' (siehe Abbildung 2.6).

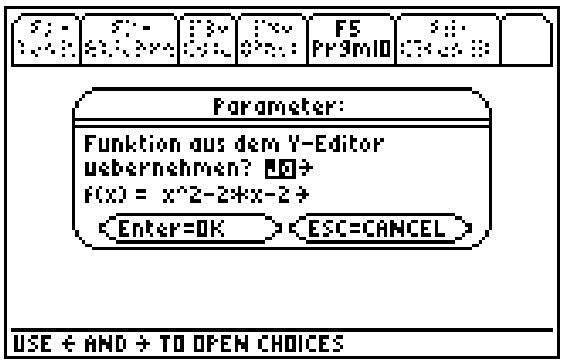

Abbildung 2.6.: Dialog Parameter zum übernehmen einer Funktion aus dem Y-Editor

Die Wahl der Funktion erfolgt per Cursor im Drop-Down-Menu 'f(x) = ' (siehe Abbildung 2.7). Hat eine der erfassten Funktionen mehr als 18 Zeichen, wird die Variable der Funktion angezeigt (z.B. `y3(x)`; analog der Bezeichnung im Y-Editor).

Abbildung 2.7.: Dialog Parameter mit Dropdown zur Wahl der Funktion aus dem Y-Editor

2.3. Erfassen der Parameter

Der Wechsel zwischen dem Eingabemodus für Ziffern und jenem für Buchstaben dient die alpha-Taste. In folgenden Dialogen kann eine Funktion f(x) erfasst werden, falls nicht schon im Y-Editor eingegeben.

2.3.1. Approximation

Bisection oder Regula falsi

Zu beachten: Für diese Methoden muss das Intervall so gewählt werden, dass der Funktionswert f(a) kleiner, der Funktionswert f(b) grösser ist als Null. Der Wert für 'n Iterationen' muss eine ganze, positive Zahl sein.

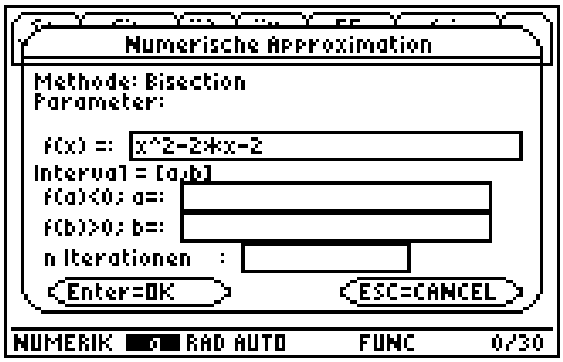

Abbildung 2.8.: Dialog Numerische Approximation für die Methode Bisection

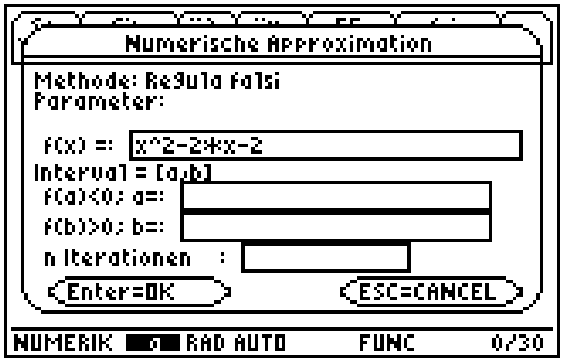

Abbildung 2.9.: Dialog Numerische Approximation für die Methode Regula falsi

Newton

Der Startwert für die Methode Newton muss wie folgt gewählt werden:

- Der Funktionswert darf nicht Null betragen (d.h. f(startwert) $\neq 0$).
- Die Steigung der Tangente durch den Startwert darf nicht Null sein (d.h. f'(startwert) $\neq 0$).

Für die Methode nach Newton kann zwischen der Anzahl Iterationen und der Genauigkeit gewählt werden.

- n Iterationen: Die Berechnung wird nach n Iterationen abgebrochen.
- Genauigkeit eps: Die Berechnung wird abgebrochen, sobald der Betrag der Differenz der letzten zwei Resultate kleiner als der in eps angegebene Wert ist.

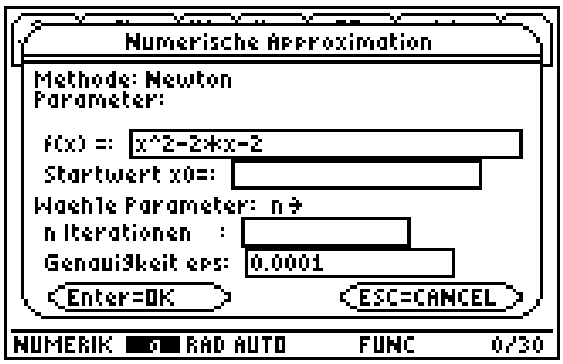

Abbildung 2.10.: Dialog Numerische Approximation für die Methode Newton

2.3.2. Integration

Zu beachten: Für diese Methoden muss das Intervall so gewählt werden, dass die Funktionswerte f(a) und f(b) grösser als Null sind, wobei b > a ist. Der Wert für 'n Schritte' muss eine ganze, positive Zahl sein.

Rechteck- oder Trapez-Verfahren

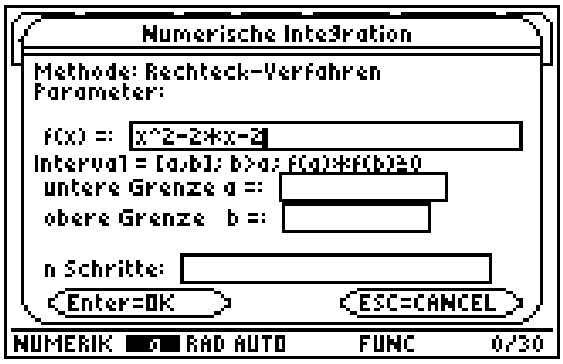

Abbildung 2.11.: Dialog Numerische Integration für die Methode Rechteck-Verfahren

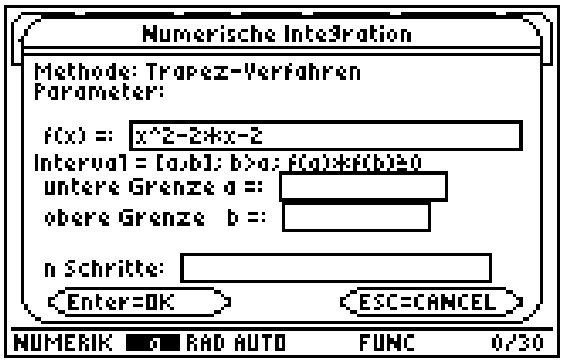

Abbildung 2.12.: Dialog Numerische Integration für die Methode Trapez-Verfahren

Tangenten- oder Simpson-Verfahren

Zu beachten: Bei diesen Methoden muss der Wert für die Variable n gerade sein.

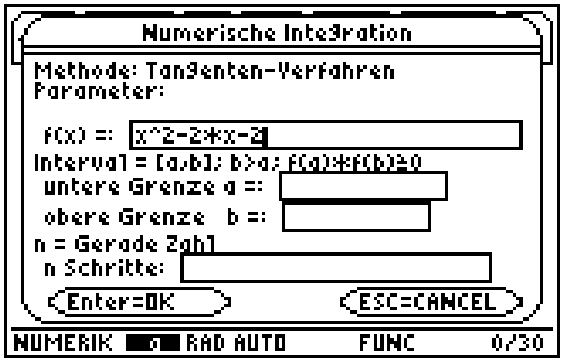

Abbildung 2.13.: Dialog Numerische Integration für die Methode Tangenten-Verfahren

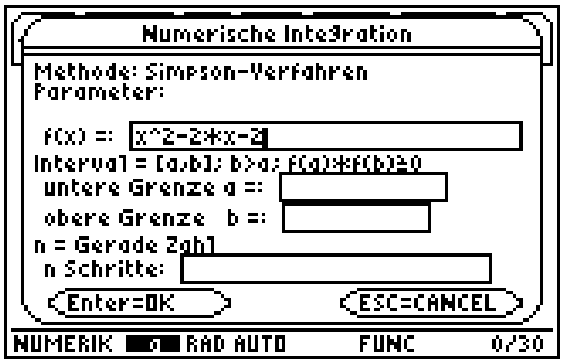

Abbildung 2.14.: Dialog Numerische Integration für die Methode Simpson-Verfahren

2.4. Resultate speichern

2.4.1. Variablenname

Um die Resultate zu speichern, werden Variablennamen wie folgt generiert:

- Matrizenvariable für die Ergebnisse:
  - die ersten drei Buchstaben der Aufgabe (**App**roximation oder **Int**egration)
  - die ersten vier Buchstaben der Methode (**Bise**ction, **Regu**la falsi, **Newt**on, **Rech**teck, **Trap**ez, **Tang**enten oder **Simp**son)
  - die Ziffer **1** als möglicher Zähler
  - Beispiele: **AppRegu1**, **IntTang1**
- Variable für die Nullstellen:
  - **NS** als Kurzform für **N**ull**s**telle
  - die ersten vier Buchstaben der Methode (**Bise**ction, **Regu**la falsi, **Newt**on)
  - die Ziffer **1** als möglicher Zähler
  - Beispiele: **NSBise1**, **NSNewt1**

Diese Variablennamen können auch geändert werden, wobei im TI-89 die Länge der Variablennamen auf acht Zeichen beschränkt ist. Existiert im Verzeichnis Numerik bereits eine Variable mit gewähltem Variablennamen, wird dies im Dialog 'Matirzenvariable' gemeldet (siehe Abbildung 2.15). Mit `ENTER` wird die bestehende Variable überschrieben, mit `ESC` kann zurück in die Resultat-Anzeige gewechselt und die Variable editiert werden.

Abbildung 2.15.: Dialog Matirzenvariable

2.4.2. Approximation

Nullstelle sowie Ergebnisse können im Dialog 'Resultat' ins angegebene Verzeichnis gespeichert werden (siehe Abbildung 2.16).

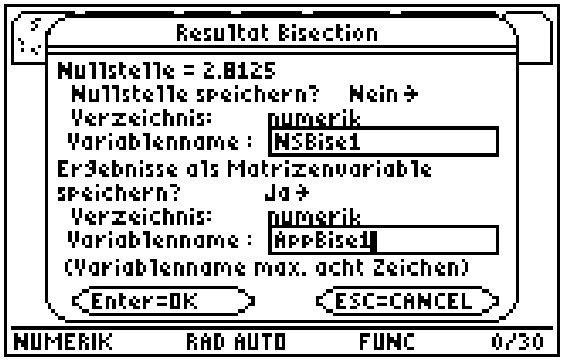

Abbildung 2.16.: Dialog Resultat für die Methode Bisection

2.4.3. Integration

Die Ergebnisse können per Dialog 'Resultat' ins angegebene Verzeichnis gespeichert werden (siehe Abbildung 2.17).

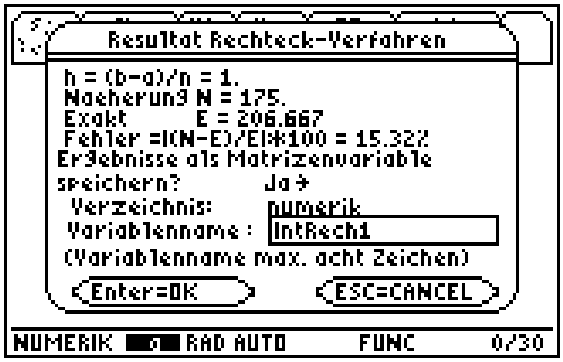

Abbildung 2.17.: Dialog Resultat für die Methode Rechteck-Verfahren

2.5. Ende der Berechnungen

Abbildung 2.18.: Pop-Up zur Wahl einer neuen Berechnung oder für die Rückkehr zum Home-Bildschirm

- Neue Berechnung: Die Routine kehrt zurück zur Auswahl der Aufgabe.
- Home: Der Bildschirm `Home` wird eingeblendet, aktuelles Verzeichnis wird das aktuelle Verzeichnis vor Programmstart, `MODE`-Einstellungen werden zurückgesetzt auf den Stand wie vor dem Programmstart.

  Achtung: `ESC` sollte hier nicht verwendet werden, da sonst ein möglicher Fehler (Undefined Variable) das Zurückstellen der `MODE` -Einstellungen auf die zu Beginn gesicherten Parameter umgeht.

3. Resultate als Matrizenvariable

Die als Matrizenvariable gespeicherten Resultate können mit `APPS` ▶ `6:Data/Matrix Editor` ▶ `2:Open...` geöffnet werden (siehe Abbildung 3.1). Folgende Einträge in den Drop-Down-Menus müssen gewählt werden (siehe Abbildung 3.2):

- Als 'Type' muss 'Matrix' gewählt werden.
- 'Folder' ist das Verzeichnis, in das die Matrize gespeichert wurde.
- 'Variable' ist der Name der Matirzenvariable.

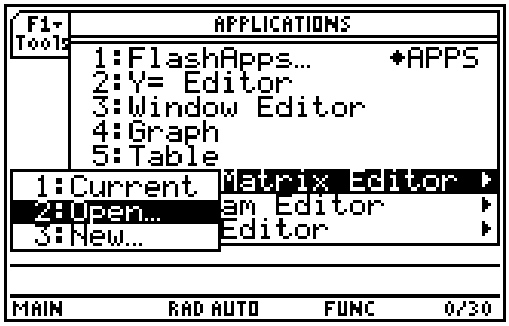

Abbildung 3.1.: Applikationen

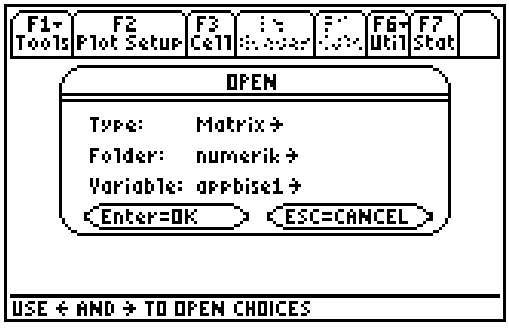

Abbildung 3.2.: Öffnen-Dialog

In die Matrize werden jeweils auch Aufgabe, Methode und Funktion gespeichert, damit ersichtliche ist, wie die Resultate zustande kamen. In der Tabelle kann mit den Pfeiltasten gearbeitet werden: Der Inhalt der aktiven Zelle wird jeweils in der Eingabezeile eingeblendet (siehe Abbildung 3.3).

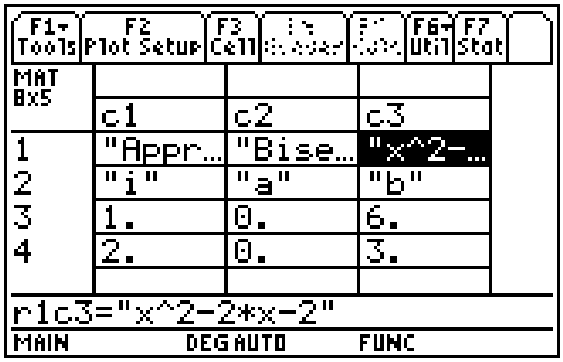

Abbildung 3.3.: Resultate als Matrize

Abbildungsverzeichnis

Literaturverzeichnis

[1] TEXAS INSTRUMENTS: *TI-89/TI-92 Plus Benutzerhandbuch. Advanced Mathematics Software Version 2.0*, 1999.

[2] TEXAS INSTRUMENTS: *TI-89/TI-92 Plus/Voyage 200 PLT Version 2.08 - Versionshinweise. Advanced Mathematics Software Version 2.8*, 2002.

A. Quellcode

Autor: Roland Bruggmann (mailto:roland.bruggmann@gmx.net)

Date: April 29, 2005

Version: 1.02

Synopsis: nummeth() is a program written in TI-BASIC for the pocket calculator TI-89 and serves with well known methods for numeric approximation and numeric integration.

A.1. Programmstart

```
:nummeth()
:Prgm
:@Numerische Methoden
    Numerische Approximation
    (Bisection, Regula falsi, Newton) und
    Numerische Integration
    (Rechteck−, Trapez−, Tangenten−, Simpson−Verfahren)
    Hinweis:
    y1 bis y10 des Y-Editors koennen
    ins Programm uebernommen werden
:@ Version 1.02, 29. April 2005
:@ Roland Bruggmann
:@ Email: roland.bruggmann@gmx.net
:ClrIO
:Disp ""
:Disp "..."
```

A.2. Konfigurationen

```
:@Konfigurationen
:Local i,oldfoldr,yeditor,z,gmlist
:@Verzeichnis
:getFold()→oldfoldr
:Lbl neuverz
:Try
:  setFold(numerik)
:Else
:  ClrErr
:  Try
:    NewFold numerik
:  Else
```

```
:    ClrErr
:    Goto modus
:  EndTry
:  Goto neuverz
:EndTry
:@Modus
:Lbl modus
:getMode("0")→gmlist
:setMode({
  "1","1",
  "3","1",
  "4","1",
  "5","1",
  "6","1",
  "7","1",
  "8","1",
  "12","1",
  "13","1",
  "14","1"})
:@Parameterliste
:Try
:  dim(numlist)
:Else
:  ClrErr
:  {"","","",""}→numlist
:EndTry
:@Y-Editor:y1 bis y10
:DelVar x
:0→z
:For i,1,10
:  expr("y"&string(i)&"(x)")→yeditor
:  If string(yeditor)≠"y"&string(i)&"(x)" Then
:    1+z→z
:    If dim(string(yeditor))>18 Then
:      "y"&string(i)&"(x)"→ylist[z]
:    Else
:      string(yeditor)→ylist[z]
:    EndIf
:  EndIf
:EndFor
:Try
:  dim(ylist)
:Else
:  ClrErr
:  {"no"}→ylist
:EndTry
```

A.3. Erfassen von Aufgabe, Methode und Funktion

```
:@Aufgabe,Methode,Parameter1
:Local auf,meth,msf,msn,msm,mspa,msw,
  pml,auflist,m1list,m2list
:"f(x) ="→msf
:"Numerische "→msn
:"Methode: "→msm
:"Parameter:"→mspa
:"Waehle "→msw
:{"Approximation","Integration"}→auflist
:{"Bisection","Regula falsi","Newton"}→m1list
:{"Rechteck","Trapez","Tangenten","Simpson"}→m2list
:{"Ja","Nein"}→qlist
:Lbl aufgabe
:Dialog
:  Title msn&"Methoden"
:  DropDown msw&"Aufgabe:",auflist,auf
:EndDlog
:If ok=0
:  Goto neub
:Lbl methode
:Dialog
:  Title msn&auflist[auf]
:  DropDown msw&msm,expr("m"&string(auf)&"list"),meth
:EndDlog
:If ok=0
:  Goto aufgabe
:Lbl pam1
:If ylist[1]≠"no" Then
:  2→pml
:  Dialog
:    Title mspa
:    Text "Funktion aus dem Y-Editor"
:    DropDown "uebernehmen?",qlist,pml
:    DropDown msf,ylist,yl
:  EndDlog
:  If ok=0
:    Goto methode
:  If pml=1
:    ylist[yl]→numlist[1]
:EndIf
```

A.4. Hauptteil

```
:@Hauptteil
:Local a,b,k,msfa,msi,msnt,n,titl,w
:" failed"→msfa
:"Interval = [a,b]"→msi
:" isn't true"→msnt
:Lbl anfang
```

A.4.1. Integration

```
:If auf=2 Then
:  @Integration
:  Local h,s,msger,msv,ms21,ms22
:  "-Verfahren"→msv
:  "b>a"→ms21
:  "f(a)*f(b)≥0"→ms22
:  numlist[1]→fstring
:  numlist[2]→n
:  numlist[3]→a
:  numlist[4]→b
:  If meth>2 Then
:    "n = Gerade Zahl "→msger
:  Else
:    ""→msger
:  EndIf
:  @Parameter
:  Dialog
:    Title msn&auflist[auf]
:    Text msm&m2list[meth]&msv
:    Text mspa
:    Text ""
:    Request msf,fstring,0
:    Text msi&"; "&ms21&"; "&ms22
:    Request "untere Grenze a =",a
:    Request "obere Grenze
:    b =",b
:    Text msger
:    Request "n Schritte",n
:  EndDlog
:  If ok=0 Then
:    If ylist[1]="no"
:    Goto methode
:    Goto pam1
:  EndIf
:  {fstring,n,a,b}→numlist
:  For i,1,dim(numlist)
:    If numlist[i]=""
:      Goto anfang
```

```
:  EndFor
:  @Test n=gerade
:  If meth>2 and remain(expr(n),2)>0 Then
:    Text msger&msnt
:    Goto anfang
:  EndIf
:  @Test Interval
:  expr(fstring)→f(x)
:  expr(a)→a
:  expr(b)→b
:  For i,1,2
:    If not expr(#("ms2"&string(i))) Then
:      Text expr("ms2"&string(i))&msnt
:      Goto anfang
:    EndIf
:  EndFor
:  @Kern
:  expr(n)→n
:  (b-a)/n→h
```

Rechteck- und Trapez-Methode

```
:  If meth<3 Then
:    @Kern rechteck u. trapez
:    f(a)/meth→s
:    setMode({"14","3"})
:    [[0,a,f(a),s,h*s]]→loesung
:    For i,1,n-1
:      a+i*h→w
:      s+f(w)→s
:      augment(loesung;[[i,w,f(w),s,h*s]])→loesung
:    EndFor
:    If meth=2 Then
:      @trapez
:      s+f(b)/2→s
:      augment(loesung;[[n,b,f(b),s,h*s]])→loesung
:    EndIf
:    h*s→res
```

Tangenten-Methode

```
:  ElseIf meth=3 Then
:    @Kern tangenten
:    a+h→w
:    f(w)→s
:    setMode({"14","3"})
:    [[1,w,f(w),s,2*h*s]]→loesung
:    For i,3,n-1,2
:      a+i*h→w
:      s+f(w)→s
```

```
:      augment(loesung;[[i,w,f(w),s,2*h*s]])→loesung
:    EndFor
:    2*h*s→res
```

Simpson-Methode

```
:  ElseIf meth=4 Then
:    @Kern simpson
:    f(a)→s
:    2→k
:    setMode({"14","3"})
:    [[0,a,f(a),s,h/3*s]]→loesung
:    For i,1,n-1
:      6-k→k
:      a+i*h→w
:      s+k*f(w)→s
:      augment(loesung;[[i,w,f(w),s,h/3*s]])→loesung
:    EndFor
:    s+f(b)→s
:    augment(loesung;[[n,b,f(b),s,h/3*s]])→loesung
:    h/3*s→res
:  EndIf
```

Exakt, Fehler, Titelmatrize

```
:  @Exakt,Fehler,Titelmatrize
:  Local exakt,fehler
:  Try
:    string(approx(∫(f(x),x,a,b)))→exakt
:  Else
:    ClrErr
:    msfa→exakt
:    msfa→fehler
:    Goto titel
:  EndTry
:  string(approx(round(abs((res-expr(exakt))
      /(expr(exakt)))*100,2)))&"%"→fehler
:  Lbl titel
:  setMode({"14","1"})
:  [[auflist[auf]&";"&expr("m"&string(auf)&"list")[meth]&msv,
      fstring,
      "h="&string(approx(h)),
      "Exakt="&exakt,
      "Fehler="&fehler]
      ["i","xi","f(xi)","","A"]]→titl
```

A.4.2. Approximation

```
:Else
:  @Approximation
:  Local msit,stel
:  "n Iterationen     "→msit
:  If meth=3 Then
:    @Newton
:    Local bed,bed1,bed2,eps,ms13,pa,palist,x0,x1
:    "Startwert "→ms13
:    {"n","eps"}→palist
:    numlist[1]→fstring
:    numlist[2]→n
:    "0.0001"→eps
:    numlist[4]→x0
:    @Parameter
:    Dialog
:      Title msn&auflist[auf]
:      Text msm&m1list[meth]
:      Text mspa
:      Text ""
:      Request msf,fstring,0
:      Request ms13&"x0=",x0
:      DropDown msw&mspa,palist,pa
:      Request msit,n
:      Request "Genauigkeit eps",eps
:    EndDlog
:    If ok=0 Then
:      If ylist[1]="no"
:        Goto methode
:      Goto pam1
:    EndIf
:    {fstring,n,eps,x0}→numlist
:    If fstring="" or numlist[pa+1]="" or x0=""
:      Goto anfang
:    @Test Startwert
:    expr(fstring)→f(x)
:    expr(x0)→x0
:    d(f(x),x)→g(x)
:    If f(x0)*g(x0)=0 Then
:      Text ms13&msfa&": f(x0)*f'(x0)=0"
:      Goto anfang
:    EndIf
```

Newton

```
:    @Kern Newton
:    expr(n)→n
:    expr(eps)→eps
:    "i=n"→bed1
```

```
:     "abs(x1-x0)<eps"→bed2
:     expr("bed"&string(pa))→bed
:     [[auflist[auf],expr("m"&string(auf)&"list")[meth],
       fstring,""]
       ["i","xi","f(xi)","f'(xi)"]]→titl
:     setMode({"14","3"})
:     [[0,x0,f(x0),g(x0)]]→loesung
:     1→i
:     Loop
:       x0-f(x0)/(g(x0))→x1
:       augment(loesung;[[i,x1,f(x1),g(x1)]])→loesung
:       If expr(bed)
:         Exit
:       x1→x0
:       i+1→i
:     EndLoop
:     setMode({"14","1"})
:     If pa=1 Then
:       7→stel
:     Else
:       dim(string(fPart(eps)))+1→stel
:     EndIf
:     x1→res
```

Bisection und Regula falsi

```
:  Else
:    @Bisection/Regula falsi
:    Local kern,kern1,kern2,ms11,ms12
:    "f(a)<0"→ms11
:    "f(b)>0"→ms12
:    numlist[1]→fstring
:    numlist[2]→n
:    numlist[3]→a
:    numlist[4]→b
:    @Parameter
:    Dialog
:      Title msn&auflist[auf]
:      Text msm&m1list[meth]
:      Text mspa
:      Text ""
:      Request msf,fstring,0
:      Text msi
:      Request ms11&"; a=",a
:      Request ms12&"; b=",b
:      Request msit,n
:    EndDlog
:    If ok=0 Then
:      If ylist[1]="no"
```

```
:        Goto methode
:      Goto pam1
:    EndIf
:    {fstring,n,a,b}→numlist
:    For i,1,dim(numlist)
:      If numlist[i]=""
:        Goto anfang
:    EndFor
:    @Test Interval
:    expr(fstring)→f(x)
:    expr(a)→a
:    expr(b)→b
:    For i,1,2
:      If not expr(#("ms1"&string(i))) Then
:        Text expr("ms1"&string(i))&msnt
:        Goto anfang
:      EndIf
:    EndFor
:    @Kern Bisection/Regula falsi
:    expr(n)→n
:    @Formel bisection
:    "(a+b)/2"→kern1
:    @Formel regula-falsi
:    "a-(b-a)/(f(b)-f(a))*f(a)"→kern2
:    expr("kern"&string(meth))→kern
:    [[auflist[auf],expr("m"&string(auf)&"list")[meth],
      fstring,"","",
      ["i","a","b","xi","f(xi)"]]→titl
:    expr(kern)→w
:    setMode({"14","3"})
:    [[1,a,b,w,f(w)]]→loesung
:    For i,2,n
:      If f(w)<0 Then
:        w→a
:      Else
:        w→b
:      EndIf
:      expr(kern)→w
:      augment(loesung;[[i,a,b,w,f(w)]])→loesung
:    EndFor
:    7→stel
:    w→res
:  EndIf
:  setMode({"14","1"})
:EndIf
:augment(titl;loesung)→loesung
:@Hauptteil Ende
```

A.4.3. Resultat

```
:@Resultat
:Local msa,mse,msma,msns,msr,mssp,msvn,msvz,
    fldr,namelist,reslist,splist,varlist,ma,ns
:getFold()→fldr
:"max. acht Zeichen)"→msa
:"Ergebnisse als "→mse
:"Matrizenvariable "→msma
:"Nullstelle "→msns
:"Resultat "→msr
:"speichern? "→mssp
:"Variablenname "→msvn
:" Verzeichnis:"→msvz
:left(auflist[auf],3)
    &left(expr("m"&string(auf)&"list")[meth],4)
    &"1"→name1
:2→ns
:Lbl resultat
:If auf=1 Then
:  @Approximation
:  "NS"&left(m1list[meth],4)&"1"→name2
:  Dialog
:    Title msr&m1list[meth]
:    Text msns&"= "&string(round(res,stel))
:    DropDown " "&msns&mssp,qlist,ns
:    Text msvz&fldr
:    Request msvn,name2
:    Text mse&msma
:    DropDown mssp&" ",qlist,ma
:    Text msvz&fldr
:    Request msvn,name1
:    Text " ("&msvn&msa
:  EndDlog
:Else
:  @Integration
:  ""→name2
:  Dialog
:    Title msr&m2list[meth]&msv
:    Text " h = (b-a)/n = "&string(round(h,3))
:    Text " Naeherung N = "&string(round(res,6))
:    Text " Exakt
:    E = "&exakt
:    Text " Fehler =|(N-E)/E|*100 = "&fehler
:    Text mse&msma
:    DropDown mssp&" ",qlist,ma
:    Text msvz&fldr
:    Request msvn,name1
:    Text " ("&msvn&msa
```

```
:  EndDlog
:EndIf
:If ok=0
:  Goto anfang
:{ma,ns}→splist
:If splist={2,2}
:  Goto neub
:@Test Variablennamen
:{name1,name2}→namelist
:For i,1,2
:  If splist[i]=1 and (namelist[i]="" or dim(namelist[i])>8)
:    Goto resultat
:EndFor
:If splist[2]=1 and namelist[2]="x" Then
:  Dialog
:    Title msns
:    Text msvn&"x unzulaessig"
:  EndDlog
:  If ok=0
:    Goto resultat
:  Goto resultat
:EndIf
```

A.5. Speichern

```
:@Speichern
:{msma,msns}→varlist
:0→z
:For i,1,2
:  Lbl neuname
:  If z=1 Then
:    Dialog
:      Title varlist[i]
:      Text msvn&namelist[i]
:      Text "wird bereits verwendet."
:      Text "Variable ueberschreiben?"
:    EndDlog
:    If ok=0
:      Goto resultat
:    DelVar #(expr("name"&string(i)))
:    0→z
:  EndIf
:  If i=1 and splist[1]=1 Then
:    Try
:      @Matirzenvariable
:      Rename loesung,#name1
:    Else
:      If errornum=270 Then
:        ClrErr
:        1→z
:        Goto neuname
:      Else
:        PassErr
:      EndIf
:    EndTry
:  ElseIf i=2 and splist[2]=1 Then
:    Try
:      @Nullstelle
:      Rename res,#name2
:    Else
:      If errornum=270 Then
:        ClrErr
:        1→z
:        Goto neuname
:      Else
:        PassErr
:      EndIf
:    EndTry
:  EndIf
:EndFor
```

A.6. Neue Berechnung

```
:@Neue Berechnung?
:Lbl neub
:Local nb
:PopUp {"Neue Berechnung","Home"},nb
:If nb=1 Then
:  Goto aufgabe
:ElseIf nb=2 Then
:  DispHome
:EndIf
```

A.7. Ausgangslage erstellen

```
:@Ausgangslage erstellen
:DelVar fstring,f,g,loesung,name1,name2,res,yl,qlist,ylist
:setMode(gmlist)
:Try
:  setFold(#oldfoldr)
:Else
:  ClrErr
:EndTry
:EndPrgm
```