Von Euler-Cauchy zu Runge-Kutta

Themenbereich	
Numerik, Analysis	
Inhalte	Ziele
• Euler-Cauchy-Verfahren • Halbschritt-Verfahren • Runge-Kutta-Verfahren • Umsetzung numerischer Integrationsverfahren mit dem TI92.	• Wichtige Verfahren zur numerischen Integration kennenlernen. • Entwicklung von Algorithmen zur numerischen Integration. • Vergleich der Genauigkeit der einzelnen Verfahren.
Adressaten: An zwei Beispielen (einer quadratischen Funktion und eines Erwärmungsvorganges) sollen verschiedene häufig gebrauchte Verfahren zur numerischen Integration entwickelt werden. Dabei sollen die Schüler (in erster Linie wohl Wahlpflicht-Schüler aus Mathematik) bzw. Schüler des Realgymnasiums der 11. und 12.Jahrgangsstufe (bei der Behandlung von Wachstumsprozessen, beim Thema Systemdynamik oder bei einer genaueren Betrachtung der numerischen Integration) erkennen, dass die Güte des numerischen Verfahrens über die Qualität der Integration (bei numerischer Integration) entscheidet.	

1. Euler-Cauchy-Verfahren

Ist die Änderungsrate *r* (der „Zuwachs“) einer Größe *y* bekannt, so läßt sich der ungefähre Verlauf von *y* ermitteln. Die Funktion *r(t)* steuert dabei unseren „Zeichenstift“. Die folgende Abbildung soll diese Idee verdeutlichen:

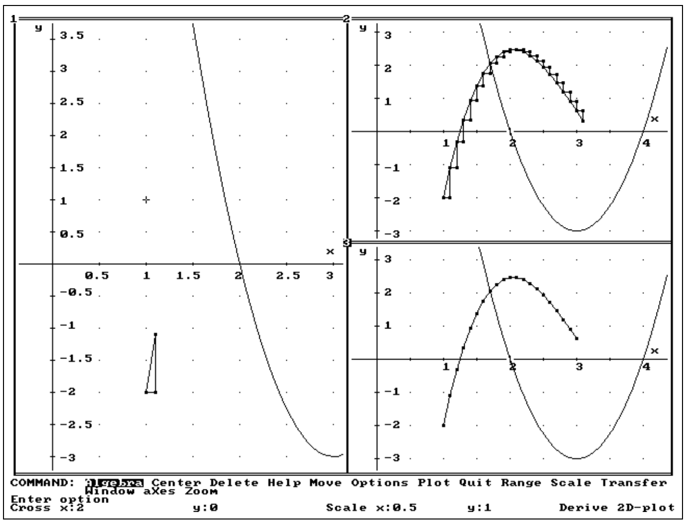
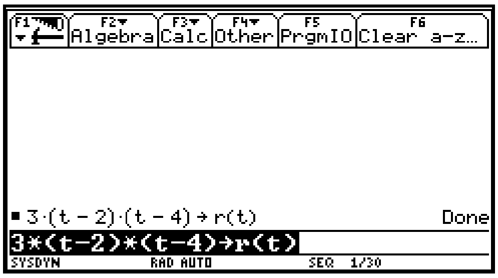


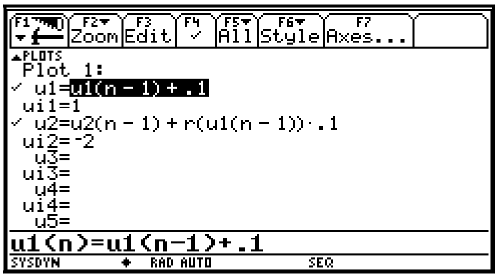
Abbildung 1: Aus (kleinen) Steigungsdreiecken entsteht die ursprüngliche Funktion

Die Umsetzung mit dem TI-92 ist gar nicht schwierig:

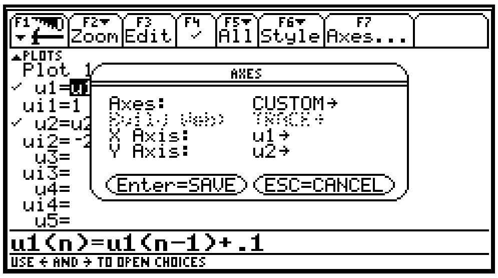
- Wir definieren zuerst die Funktion *r(t)*



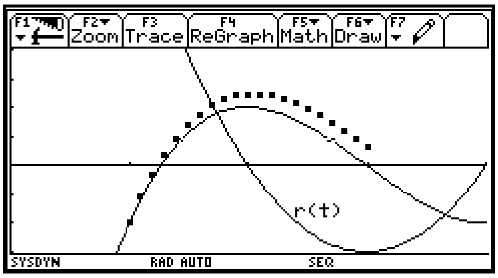
- Weiters definieren wir als Folge *u1(n)* jene Folge von *x*-Werten, die wir durchlaufen wollen. Als Folge *u2(n)* definieren wir die Summe von *altem Funktionswert + Zuwachs**Δt.



- Abschließend legen wir noch die Achsen (über F7) geeignet fest.



- So erhalten wir schließlich die gewünschte Folge.



Da diese Definitionen doch etwas mühsam sind und auch kaum ausbaubar (für andere numerische Verfahren), empfiehlt es sich, ein kleines Programm zu schreiben. Dieses Programm benützt eine Hilfsfunktion, die den Aufbau der Folge im *y-Editor* (Sequence-Mode) bewerkstelligt.

Hilfsfunktion (EUler-CAuchy mit einer Bestandsgröße):

```
euca1(t,y,„t)
Func
  y+rate1*„t
EndFunc
```

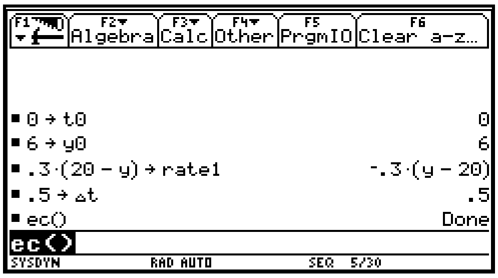
Programm:
```
ec()
Prgm
  t0»ui1:y0»ui2
  u1(n-1)+„t»u1(n)
  euca1(u1(n-1),u2(n-1),„t)»u2(n)
EndPrgm
```

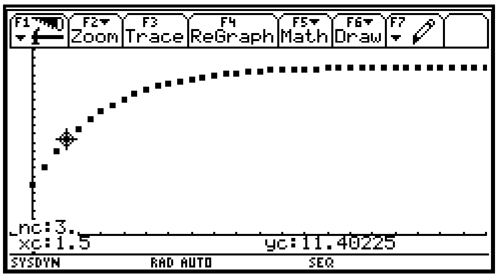
Im *Home-Screen* ist dann nur mehr folgendes zu definieren

```
3.(t-2).(t-4) » rate1
1    » t0
-2   » y0
0.1  » Δt
```

Damit können wir nun auch bequem Systeme mit einer Zusstandsgröße simulieren.

Beispiel 1: *Ein Getränk wird aus dem Kühlschrank genommen. Zu Beginn hat es eine Temperatur von 6°C. Die Umgebung habe eine Temperatur von 20°C. Pro Minute nimmt die Temperatur um 30% der Differenz zwischen Umgebungstemperatur und Flüssigkeitstemperatur zu.*
Wir wollen uns ansehen wie der Erwärmungsvorgang abläuft.





2. Improved-Euler-Cauchy-Verfahren (Verfahren von Heun, Halbschrittverfahren)

Verwenden wir als Steigung für das Steigungsdreieck im Intervall [t,$t+\Delta t$] nicht die Steigung an der Stelle t, also $r(t)$, sondern die Steigung an der rechten Intervallgrenze $t + \Delta t$ so erhalten wir eine veränderte Kurve (siehe Abb.2).

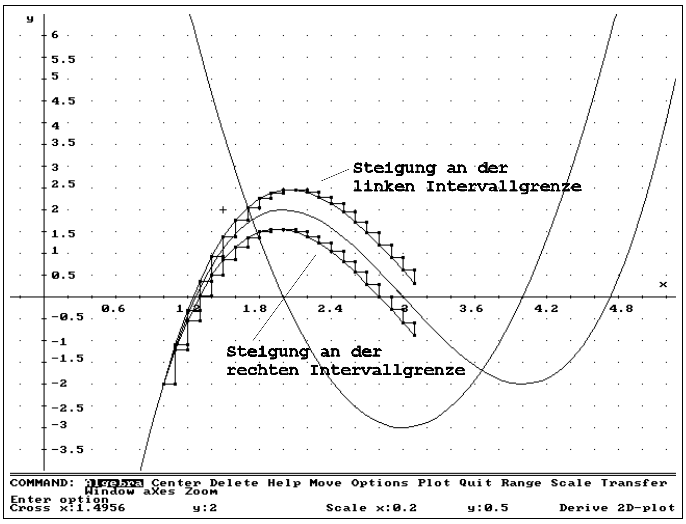


Abbildung 2: Steigungsdreiecke mit der Steigung am Intervallbeginn (an der Stelle t) und am Intervallende (t+Δt)

Es bieten sich **S** auf ganz elementargeometrischem Wege **S** zwei Möglichkeiten an, zu einem Mittelwert zu gelangen:

α) wir nehmen den Mittelwert der beiden Steigungen an den Intervallgrenzen,

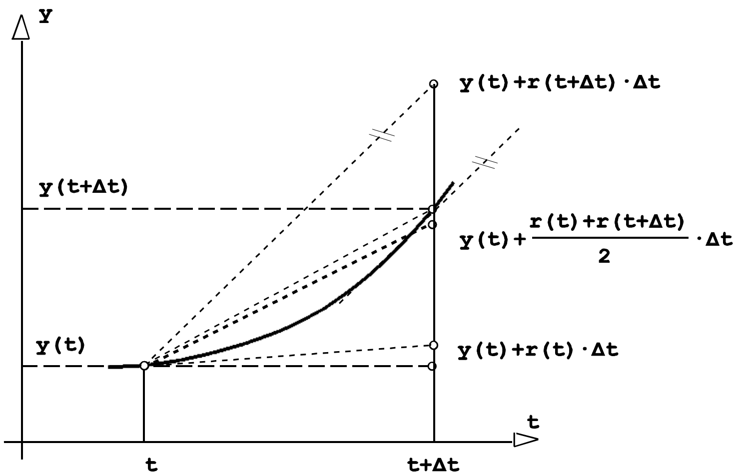


Abbildung 3: Skizze 2

β) wir nehmen die Steigung in der Mitte des Intervalles.

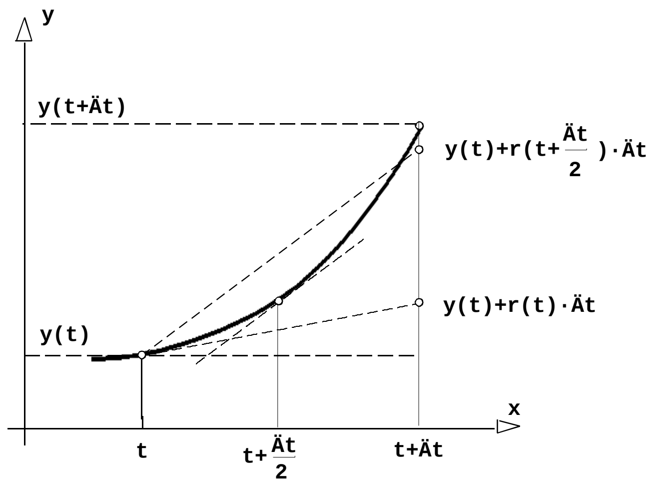
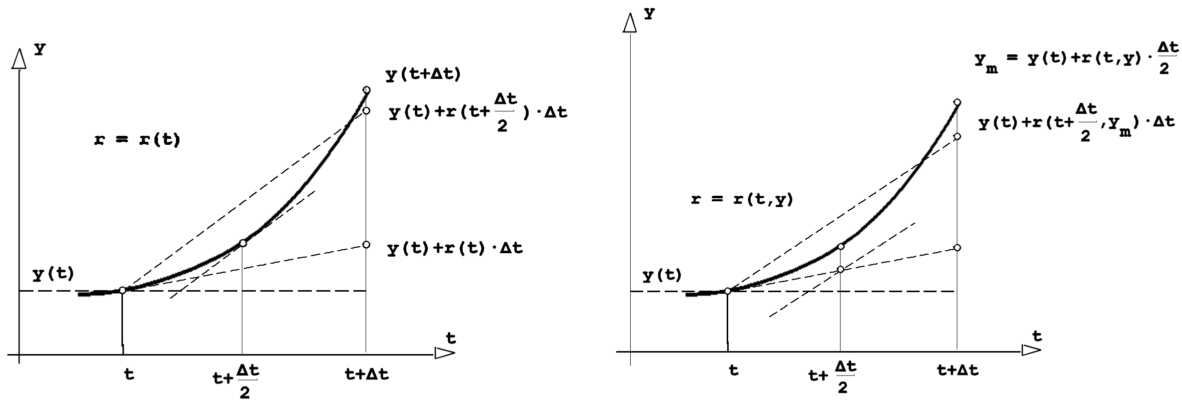


Abbildung 4: Skizze 3

Beide Varianten bringen uns dem Ziel ein großes Stück näher und bringen i.w. gleich gute Ergebnisse. Beide Methoden lassen sich problemlos implementieren und liefern damit eine TI-92-Implementation des sogenannten Halbschrittverfahrens. Für den Fall, daß die Änderungsrate nicht nur von *t* sondern auch von *y* (also vom aktullen Bestand) abhängt, müssen wir das Verfahren noch geringfügig modifizieren.



Abbbildung 4, 5: Halbschrittverfahren bei *r=r(t)* und *r=r(t,y)*

Hilfsfunktion (Improved EUler CAuchy mit einer Bestandsgröße)

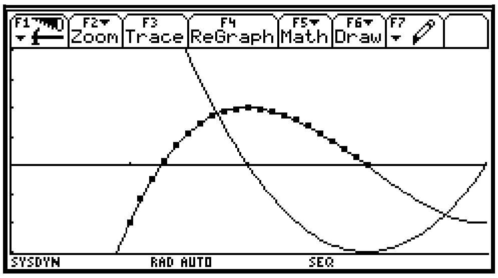
```
ieuca1(t,y,„t)
Func
  Local ym
  y+rate1*„t/2»ym
  y+(rate1|t=t+„t/2 and y=ym)*„t
EndFunc
```

Programm

```
iec()
Prgm
```

```
  t0»ui1:y0»ui2
  u1(n-1)+„t»u1(n)
  ieuca1(u1(n-1),u2(n-1),„t)»u2(n)
EndPrgm
```

Der Erfolg ist verblüffend:



3. Runge-Kutta-Verfahren

Wenn das Halbschrittverfahren auch in der Regel gute bis sehr gute Ergebnisse liefert, so läßt es sich durchaus noch weiter verbessern: eine weitere Verbesserung stellt hier das RUNGE-KUTTA-Verfahren dar, das zur Berechnung des nächsten Schrittes einen gewichteten arithmetischen Mittelwert aus vier Änderungsraten verwendet. Die erste Änderungsrate r_1 ist die Änderungsrate beim Startpunkt $r_1 = r(t,y)$. Mit die Änderungsrate r_1 wird genauso wie beim Halbschrittverfahren ein Stützpunkt in der Intervallmitte berechnet. Die Änderungsrate in diesem Stützpunkt nennen wir r_2. Mit Hilfe von r_2 wird in gleicher Art und Weise vom Startpunkt aus ein zweiter Stützpunkt berechnet, die Änderungsrate in diesem Punkt ist dann r_3. Schließlich wird noch ein dritter Stützpunkt dadurch ermittelt, daß mit r_3 ein ganzer Schritt durchgeführt wird. Für die Berechnung des neuen Punktes wird schließlich der Mittelwert herangezogen

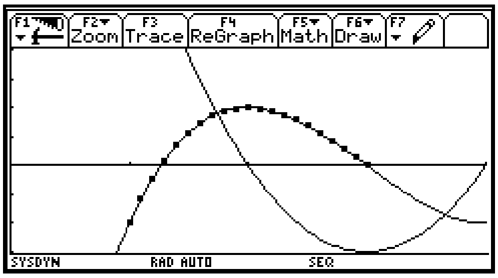
$$\frac{r_1+2\cdot r_2+2\cdot r_3+r_4}{6} \quad .$$

Hilfsfunktion (RUnge-KUtta mit einer Bestandsgröße)

```
ruku1(t,y,„t)
Func
  Local ra1,ra2,ra3,ra4
  rate1*„t»ra1
  (rate1|t=t+„t/2 and y=y+ra1/2)*„t»ra2
  (rate1|t=t+„t/2 and y=y+ra2/2)*„t»ra3
  (rate1|t=t+„t and y=y+ra3)*„t»ra4
  y+(ra1+2*ra2+2*ra3+ra4)/6
EndFunc
```

Programm

```
rk()
Prgm
  t0»ui1:y0»ui2
  u1(n-1)+„t»u1(n)
  ruku1(u1(n-1),u2(n-1),„t)»u2(n)
EndPrgm
```



Bei unserem Beispiel ist kaum ein Unterschied in der Graphik zwischen dem Improved-Euler-Cauchy und dem Runge-Kutta-Verfahren zu sehen. Der Vergleich der Wertetabellen macht uns aber sicher: Während das Runge-Kutta-Verfahren praktisch exakt arbeitet, treten beim Improved-Euler-Cauchy-Verfahren doch noch merkbare Fehler auf.

Runge-Kutta

n	u1	u2
0.	1.	-2.
5.	1.5	1.125
10.	2.	2.
15.	2.5	1.375
20.	3.	0.
25.	3.5	-1.375
30.	4.	-2.
35.	4.5	-1.125

n=0.

Improved-Euler-Cauchy

n	u1	u2
0.	1.	-2.
5.	1.5	1.1238
10.	2.	1.9975
15.	2.5	1.3713
20.	3.	-.005
25.	3.5	-1.381
30.	4.	-2.008
35.	4.5	-1.134

n=0.