VII. Numerische Behandlung von Differentialgleichungen

7.1. Gewöhnliche Diff'gleichungen (erster Ordnung)

Aufgabe: Funktion y(x) nur implizit gegeben durch Bedingungen an die Ableitung y' ?? → y ??

$y' = \varphi(y, x)$ → **y(x) ?**

Ableitung von y(x) nach x in jedem möglichen Punkt (x,y) ist gegeben durch Funktion φ(x,y).
Diff'gleichungsproblem allgemein: Aus Beziehungen zwischen den Änderungen (Ableitungen) und den Funktionswerten soll eine gesuchte Funktion bestimmt werden!

7.1.1. Beispiel: Freier Fall

Kugel wird aus Höhe h_0 losgelassen zum Zeitpunkt $t_0 = 0$ mit Anfangsgeschwindigkeit $v_0 = 0$, Erdbeschleunigung g.

$$\boxed{\begin{aligned} v(t) &= \dot{s}(t) = \frac{ds}{dt}, \\ -g &= \dot{v}(t) = \frac{dv}{dt} = \ddot{s}(t) = \frac{d^2 s}{dt^2}. \end{aligned}}$$

Diff'gleichung für v(t): $\dot{v}(t) = \varphi_1(v,t) = -g$

Integration → $v(t) = \int_0^t \dot{v}(\tau)d\tau = \int_0^t -g\,d\tau = -gt;$

Diff'gleichung für s(t):

$$\dot{s}(t) = \varphi_2(s,t) = v(t) = -gt$$

Integration →

$$s(t) = h_0 + \int_0^t -g\tau \, d\tau = h_0 - \frac{1}{2} g t^2 .$$

Aus Anfangswerten $t_0 = 0$ und h_0, und aus der Bedingung für die Ableitung wird die Funktion selbst bestimmt.

In dieser einfachen Form explizit lösbar durch Quadratur!

Im Allgemeinen ist das Integral nicht direkt lösbar,

oder

Diff'gleichung kann nicht auf Integral zurückgeführt werden.

7.1.2. Definition: Anfangswertproblem (AWP)

Aus Anfangswert $y(x_0) = y_0$ und
Differentialgleichung $y'(x) = \varphi(y,x)$ soll
die Funktion $y(x)$ für $x > x_0$ bestimmt werden.

Dabei ist die Diff'gleichung hier in expliziter Form gegeben,
d.h. in der Form: $y'(x) = \ldots$

Impliziter Fall: $\varphi(y',y,x) = 0 \rightarrow y$???

(Beispiel implizit: $y' * \exp(y') = y+x$)

7.1.3. Beispiel für explizit lösbare Diff'gleichung:

$$y'(x) = \varphi(y, x) = \alpha y \qquad und \qquad y(x_0) = y_0$$

Lösung durch Integration (Separation)

$$\frac{dy}{dx} = \alpha y \Leftrightarrow \frac{dy}{y} = \alpha \cdot dx$$

$$\int_{y(x_0)}^{y(x)} \frac{dy}{y} = \int_{x_0}^{x} \alpha dt$$

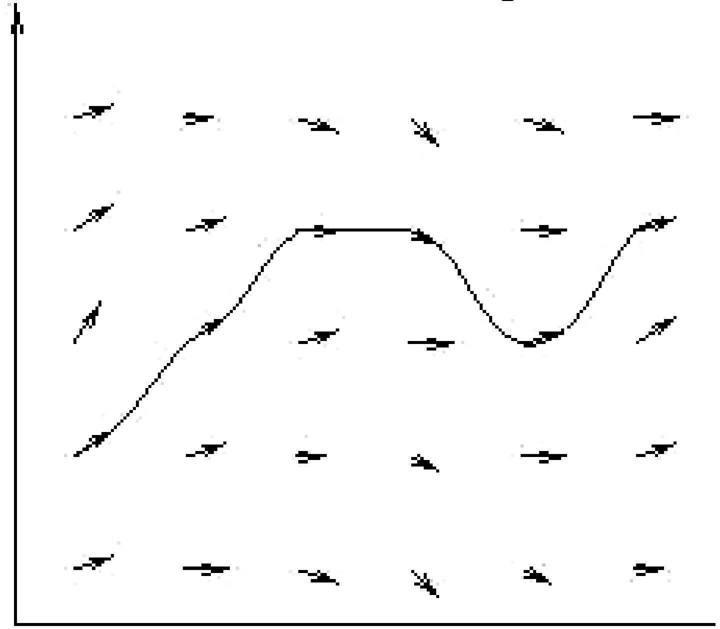
$$\ln(y(x)) - \ln(y(x_0)) = \alpha(x - x_0)$$

und daher $\quad y(x) = y_0 \; e^{\alpha(x - x_0)}$

7.1.4. Geometrische Problemstellung:

Von einer Funktion sind alle möglichen Ableitungswerte an allen möglichen Stellen (x,y) bekannt
(also an allen potentiellen Funktionswerten)
Dies entspricht einem Vektorfeld von Tangentenrichtungen
Weiterhin bekannt ist der Funktionswert an einer Stelle x_0 .

Gesucht: Kurve, deren Tangenten in allen Punkten dem vorgegebenen Vektorfeld entspricht:



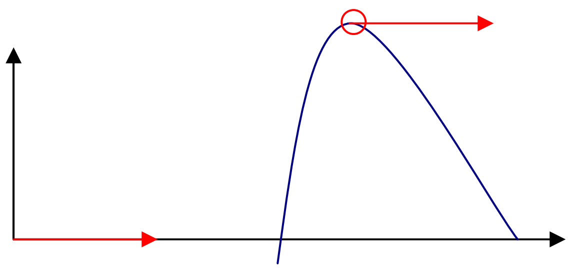
Im Beispiel ‚Freier Fall' ist dieses Vektorfeld trivial:

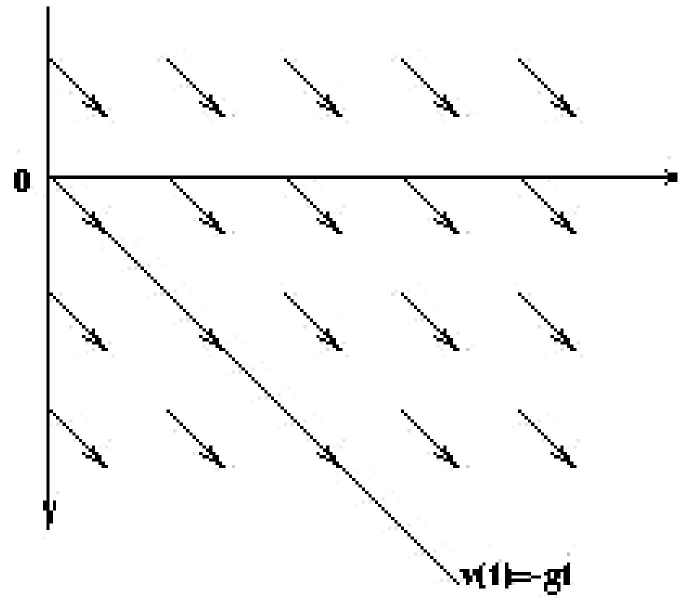
$$\dot{v}(t) = -g, \qquad v(t_0) = 0.$$

In der (v,t)-Ebene ist jede Tangentenrichtung durch den Vektor $(1,v')^T$ gegeben, also hier durch den Vektor $(1,-g)^T$.

Die Lösungskurve ist dann die Gerade mit Steigung -g durch den Nullpunkt, also v(t)=-gt .

v'=0: Tangentenvektor $(1,0)^T$





7.1.5. Das Eulerverfahren:

Gegeben AWP Anfangswertproblem

$$y'(x) = \varphi(y, x) \qquad und \qquad y(x_0) = y_0$$

Gesucht: y(x) für $x \geq x_0$

Wir wollen y(x) bei x_0 lokal als lineare Funktion g(x) betrachten, und einen kleinen Schritt der Länge h zu $x_1 = x_0 + h$ entlang dieser linearen Näherung gehen.

Dadurch erhält man den Näherungswert

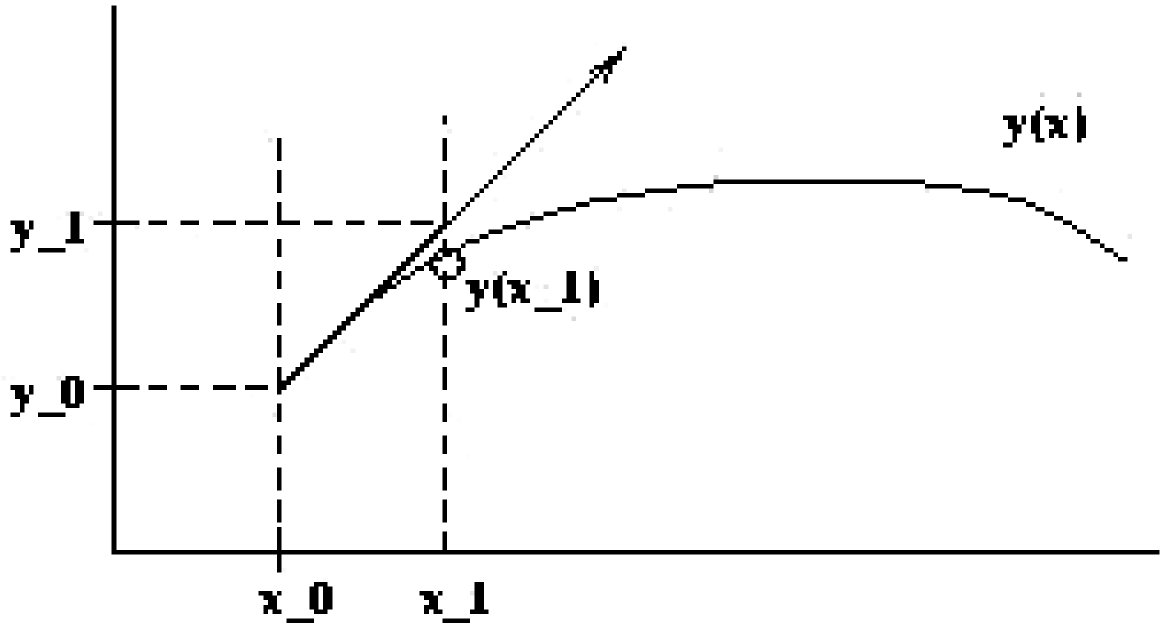
$$y(x_1) \approx y_1 = g(x_1) = y_0 + y'(x_0)(x_1 - x_0) = y_0 + h \cdot \varphi(y_0, x_0)$$

Ersetze wieder Funktion lokal durch Tangentengleichung!

Dies ergibt die Iterationsvorschrift des Eulerverfahrens (Vorwärts-Euler):

$$\boxed{y_0 = y(x_0); \quad x_{k+1} = x_k + h; \quad y_{k+1} = y_k + h \cdot \varphi(y_k, x_k) \qquad \text{für } k=0,1,...}$$

Der Einfachheit halber wählen wir die Schrittweite h konstant (muss aber nicht sein).



Euler aus Integration:

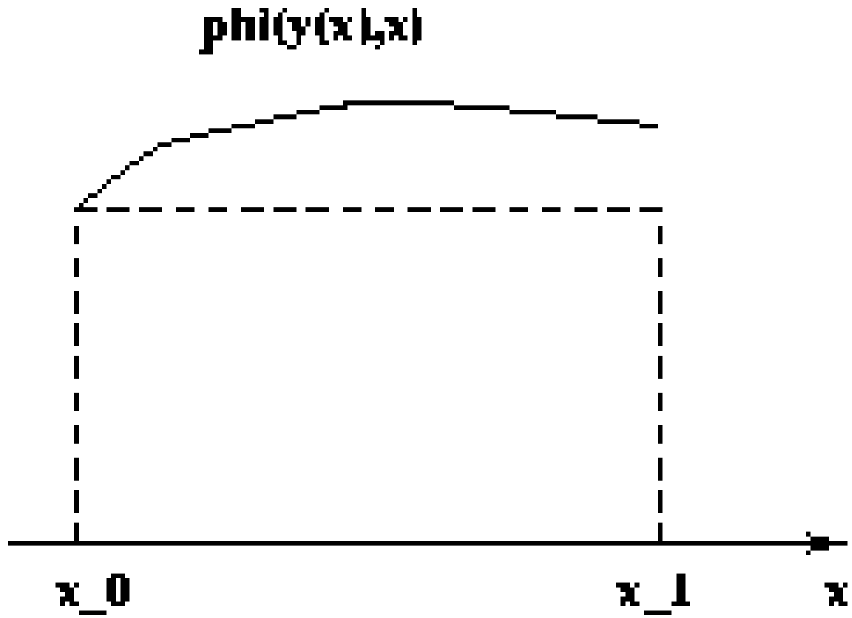
Betrachte die Diff'gleichung zwischen den Stellen x_0 und x_1 :

$$y_1 - y_0 = y(x)\Big]_{x_0}^{x_1} = \int_{x_0}^{x_1} y'(x)dx = \int_{x_0}^{x_1} \varphi(y(x),x)dx \approx (x_1 - x_0)\cdot\varphi(y_0,x_0).$$

aus Rechteckregel, indem die Fläche unter der Kurve $\varphi(y(x),x)$ angenähert wird durch die Fläche des Rechtecks mit den Ecken

$(x_0,0)$, $(x_1,0)$, und

$(x_1,\varphi(y_0,x_0))$, $(x_0,\varphi(y_0,x_0))$



Eulerverfahren aus Taylorentwicklung :

$$y(x_1) = y(x_0 + h) = y(x_0) + hy'(x_0) + \cancel{\frac{h^2}{2} y''(z_0)}$$

Bekannt sind $y(x_0) = y_0$, $y'(x_0) = \varphi(y_0, x_0)$, mit Zwischenstelle z_0. Der h^2-Term ist klein und wird vernachlässigt → Eulerverfahren.

Euler aus der Diskretisierung des Differentialquotienten:

$$\varphi(y_0, x_0) = y_0' = \left.\frac{dy}{dx}\right]_{x_0} \approx \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_1 - y_0}{h}$$

ergibt wieder das Eulerverfahren!

7.1.6. Verbesserte Verfahren:

7.1.6.1. Rückwärts-Euler:

Der Diff'quotient kann natürlich mit derselben Berechtigung angenähert werden durch

$$\varphi(y_1, x_1) = y_1' = \left.\frac{dy}{dx}\right]_{x_1} \approx \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_1 - y_0}{h}$$

Dies führt zu der Vorschrift $\boxed{y_{k+1} = y_k + h \cdot \varphi(y_{k+1}, x_{k+1})}$

Im Unterschied zum einfachen Euler taucht hier die Unbekannte y_{k+1} auch noch in der Funktion φ auf;
das macht die Sache komplizierter.

Um y_{k+1} zu erhalten, ist die Nullstelle einer Funktion zu bestimmen, nämlich von

$$f(z) = z - h \cdot \varphi(z, x_{k+1}) - y_k \overset{!}{=} 0$$

Die berechnete Nullstelle $\overline{z}$ ist dann die nächste Näherung

$$\bar{z} = y_{k+1} \approx y(x_{k+1})$$

Solche Verfahren heißen implizite Verfahren, im Gegensatz zum einfachen Eulerverfahren, das ein explizites Verfahren ist.
Zur Bestimmung von y_{k+1} kann man iterative Verfahren wie z.B. das Newtonverfahren verwenden.

7.1.6.2. Weitere Taylorpolynom-Verfahren:

Berücksichtigung höherer Terme in der Taylorentwicklung:

$$y(x_1) = y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2} y''(x_0) + \cancel{\frac{h^3}{6} y'''(z_0)}$$

mit Zwischenstelle z_0.

Hier taucht aber die unbekannte Ableitung $y''(x_0)$ auf.
Sie kann berechnet werden aus

$$y''(x) = \frac{dy'}{dx} = \frac{d}{dx}\varphi(y(x), x) = \frac{\partial}{\partial y}\varphi(y, x) \cdot \frac{dy}{dx} + \frac{\partial}{\partial x}\varphi(y, x) =$$

$$= \frac{\partial \varphi}{\partial y}\varphi + \frac{\partial \varphi}{\partial x}$$

Benötigt: Partielle Ableitungen, Nachdifferenzieren (Kettenregel)

Also
$$y''(x_0) = \frac{\partial}{\partial y}\varphi(y,x)\Big]_{(y_0,x_0)} \cdot \varphi(y_0,x_0) + \frac{\partial}{\partial x}\varphi(y,x)\Big]_{(y_0,x_0)}$$

und damit

$$\boxed{y_{k+1} = y_k + h\varphi(y_k,x_k) + \frac{h^2}{2}\left(\frac{\partial\varphi}{\partial y}(y_k,x_k)\cdot\varphi(y_k,x_k) + \frac{\partial\varphi}{\partial x}(y_k,x_k)\right)}$$

Auf diese Art können beliebig hohe Ableitungen der Lösungsfunktion y an einer Stelle (y_k, x_k) auf Ableitungen der Funktion φ zurückgeführt werden.

Die nächste Iterierte erhält man aus dem Anfang der Taylorentwicklung an der letzten Stelle x_k.
Man spricht hier auch von Einschrittverfahren da stets
$y_k \rightarrow y_{k+1}$
Modifizierung: Runge-Kutta; vermeide höhere Ableitungen.

7.1.6.3. Mehrschrittverfahren:

Hier wird aus mehreren schon berechneten Iterierten das nächste y_{k+1} gewonnen, also

$$y_{k+1-m}\ ,\ \dots\ ,\ y_{k-1}\ ,\ y_k \ \rightarrow\ y_{k+1}\ .$$

Solche Verfahren können sehr einfach aus Quadraturregeln hergeleitet werden, z.B.

$$y(x_{k+1}) - y(x_{k-1}) = y(x)\Big]_{x_{k-1}}^{x_{k+1}} = \int_{x_{k-1}}^{x_{k+1}} y'(x)\ dx =$$

$$= \int_{x_{k-1}}^{x_{k+1}} \varphi(y(x), x)\ dx \approx$$

$$\approx (x_{k+1} - x_{k-1}) \cdot \varphi(y_k, x_k)$$

unter Verwendung der Mittelpunktsregel.

Also $$\boxed{y_{k+1} = y_{k-1} + 2h\varphi(y_k, x_k)}$$

7.1.7. AWP für Diff'gleichungen erster Ordnung im R^n:

Sei
$$y(x) = \begin{pmatrix} y_1(x) & \cdots & y_n(x) \end{pmatrix}^T$$
Also
$$y'(x) = \begin{pmatrix} y_1'(x) & \cdots & y_n'(x) \end{pmatrix}^T = \varphi(y, x) =$$
$$= \begin{pmatrix} \varphi_1(y_1, \ldots, y_n, x) & \cdots & \varphi_n(y_1, \ldots, y_n, x) \end{pmatrix}^T$$
mit vorgegebenem Startvektor
$$y(x_0) = \begin{pmatrix} y_1(x_0) & \cdots & y_n(x_0) \end{pmatrix}^T$$
Lösung genauso durch Euler:
$$\boxed{y^{k+1} = \begin{pmatrix} y_1^{k+1} \\ \vdots \\ y_n^{k+1} \end{pmatrix} = \begin{pmatrix} y_1^{k} \\ \vdots \\ y_n^{k} \end{pmatrix} + h \cdot \begin{pmatrix} \varphi_1(y_1^k, \ldots, y_n^k, x_k) \\ \vdots \\ \varphi_n(y_1^k, \ldots, y_n^k, x_k) \end{pmatrix}}$$

7.1.8. AWP für Diff'gleichungen höherer Ordnung:

Gegeben eine Bedingung für die j-te Ableitung der Funktion y(x)

$$\frac{d^j}{dx^j} y(x) = y^{(j)}(x) = \Psi\left(y^{(j-1)}, \ldots, y, x\right)$$

mit Anfangswerten

$$\begin{pmatrix} y^{(j-1)}(x_0) \\ \vdots \\ y(x_0) \end{pmatrix} = \begin{pmatrix} y_0^{(j-1)} \\ \vdots \\ y_0 \end{pmatrix}$$

Umformulieren in Diff'gleichung erster Ordnung im R^j :
Definiere dazu Vektorfunktion $u(x) = (u_1(x), \ldots , u_j(x))^T$
und setze

$$\begin{aligned} u_1(x) &:= y(x)\,, \\ u_2(x) &:= y'(x)\,, \\ &\;\;\ldots \\ u_j(x) &:= y^{(j-1)}(x)\,. \end{aligned}$$

Damit erhalten wir für u(x)

$$u'(x) = \begin{pmatrix} u_1'(x) \\ \vdots \\ u_{j-1}'(x) \\ u_j'(x) \end{pmatrix} = \begin{pmatrix} u_2(x) \\ \vdots \\ u_j(x) \\ \Psi(u_j, \ldots, u_1, x) \end{pmatrix} = \varphi(u, x)$$

mit Anfangsbedingungen

$$u(x_0) = \begin{pmatrix} u_1(x_0) \\ \vdots \\ u_j(x_0) \end{pmatrix} = \begin{pmatrix} y(x_0) \\ \vdots \\ y^{(j-1)}(x_0) \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_0^{(j-1)} \end{pmatrix}$$

Eulerverfahren auf Vektor: $u^{k+1} = u^k + h \cdot \varphi(u^k, x_k)$

7.1.9. Fehleranalyse:

Start bei $a = x_0$,
Gesuchter Wert $b = x_n$,

Bei äquidistanter Einteilung ist Schrittweite $h := (b-a)/n$
Damit $x_j = x_0 + jh$, $j=0,1,...,n$

Definition lokaler Diskretisierungsfehler:

An der Stelle x_k ist der lokale Dis.-Fehler gegeben durch

$$|y_{k+1} - y(x_{k+1})| = |e_k|$$

also der Fehler der in einem Schritt von $(y(x_k), x_k)$ nach (y_{k+1}, x_{k+1}) entsteht.

$y(x_k)$ ist dabei der exakte Wert einer Lösung, nicht y_k!

Definition globaler Diskretisierungsfehler:

Für das AWP zur Bestimmung der Lösung an der Stelle b ist der globale Dis.-Fehler gegeben durch

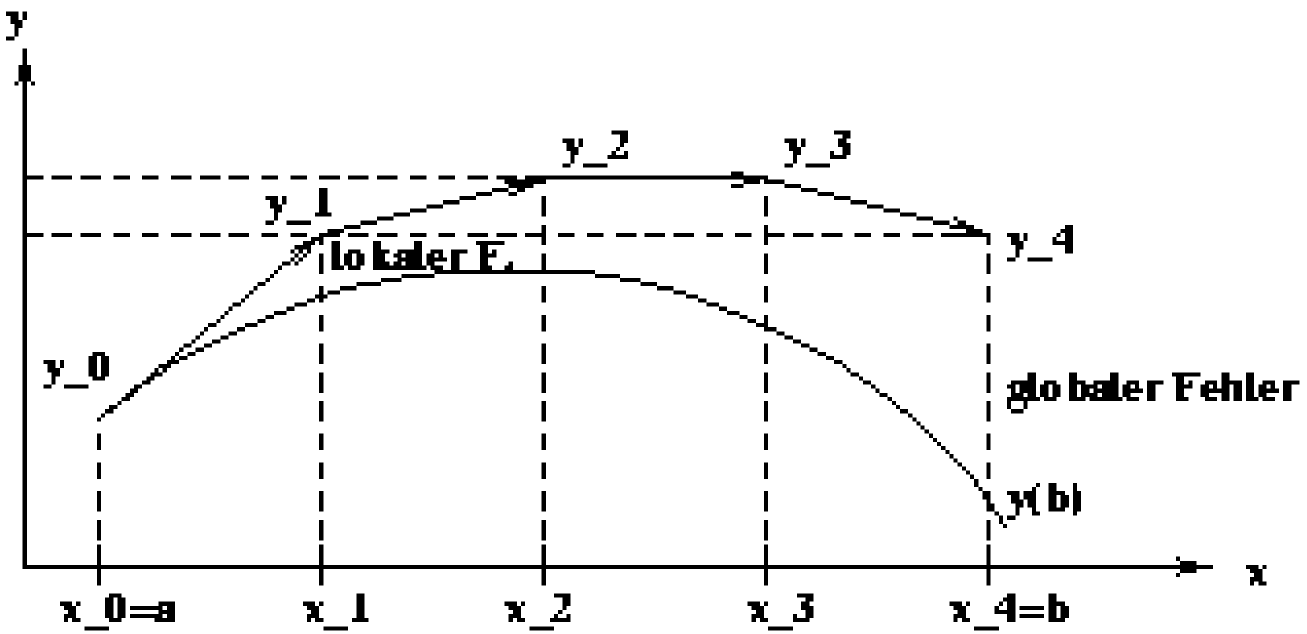
$$\left|y_n - y(b)\right| = \left|y_n - y(x_n)\right| = \left|g_n\right|$$

Definition der Konvergenz:

Die durch unser Lösungsverfahren erzeugte Folge heißt konvergent, wenn gilt

$$\lim_{h \to 0, n \to \infty} y_n = y(b)$$

Wenn also in exakter Arithmetik bei immer feineren Unterteilungen der Näherungswert gegen den exakten Wert konvergiert.
Das bedeutet natürlich auch, dass der globale Fehler gegen 0 gehen muß!



kmdiff.ppt

An jeder Stelle x_k treten neue lokale Fehler auf und addieren sich zum globalen Fehler.
Um insgesamt Konvergenz zu erhalten, muss also das Verfahren so sein, dass

- lokal ein genügend kleiner Fehler entsteht (Konsistenz) und
- sich diese lokalen Fehler nur zu einem kleinen globalen Fehler aufsummieren (Stabilität).

Definition Konsistenz:

Ein Verfahren heißt konsistent, wenn der lokale Diskretisierungsfehler mindestens von der Ordnung h^2 ist, also

$$\left| y_1 - y(x_1) \right| = O(h^2)$$

Nur so ist für den globalen Fehler $O(h^2)n = O(h)$ erreichbar!

Ist $\left|y_1 - y(x_1)\right| = O(h^{p+1})$ mit $p \geq 1$, so heißt das Verfahren von der Ordnung p.
Offensichtlich ist das Eulerverfahren konsistent von Ordnung 1, da gilt

$$\left|y(x_1) - y_1\right| = \left|y(x_1) - y_0 - h\varphi(y_0, x_0)\right| = \frac{1}{2}h^2\left|y''(z)\right|$$

mit einer Zwischenstelle z.

Zur Untersuchung der Stabilität beschränken wir uns auf den ‚linearen' Spezialfall

$y' = \varphi(y,x) = \lambda y$ mit Lösung $y(x)=y_0\exp(\lambda x)$

7.1.10. Stabilität bei Diff'gleichungen im R^n:

Für die berechneten Näherungslösungen gilt beim Eulerver.

auf $\qquad y'(x) = \varphi(y,x) = \lambda y \qquad und \qquad y(x_0) = y_0$

mit Lösung $\qquad y(x) = y_0 \; e^{\lambda(x-x_0)}$:

$$y_1 = y_0 + h\varphi(y_0, x_0) = (1+h\lambda)y_0,$$

$$y_2 = y_1 + h\lambda y_1 = (1+h\lambda)y_1 = (1+h\lambda)^2 y_0,$$

$$y_k = (1+h\lambda)y_{k-1} = (1+h\lambda)^k y_0.$$

Für $|1+h\lambda|>1$ wird daher die Näherungslösung y_k immer größer.

Für $|1+h\lambda|<1$ geht die Näherungslösung y_k gegen 0.

Stimmt dieses Verhalten mit der tatsächlichen Lösung überein?

Für die tatsächliche Lösung $y(x) = y_0 \exp(\lambda x)$ gilt

$$\lambda > 0 :\Rightarrow \lim_{x\to\infty} y(x) = \infty$$

$$\lambda < 0 :\Rightarrow \lim_{x\to\infty} y(x) = 0$$

Im Fall $\lambda > 0$ zeigen also exakte und Näherungslösung dasselbe Verhalten.

Kritisch ist der Fall $\lambda < 0$:

Die exakte Lösung geht gegen 0,

die Näherungslösung aber nur dann, wenn $|1+h\lambda| < 1$ gilt,

d.h. nur wenn $\boxed{h < 2 / |\lambda|}$

Dies ist die Stabilitätsbedingung, die garantiert, dass eine beschränkte exakte Lösung auch durch eine beschränkte Näherungslösung approximiert wird, und der globale Fehler nicht zu stark anwachsen kann.

Also ist für $\lambda < 0$ das Eulerverfahren nur konvergent, wenn die Schrittweite h so gewählt ist, dass $h < 2 / |\lambda|$ gilt !

Führen wir dieselbe Untersuchung für das implizite (Rückwärts) – Euler –Verfahren durch, so erhält man die Näherungslösungen von der Form

$$y_k = \frac{y_0}{(1 - h\lambda)^k}$$

Im kritischen Fall λ < 0 gilt hier unabhängig von h

$$\lim_{k\to\infty} y_k = \lim_{k\to\infty} \frac{y_0}{(1-h\lambda)^k} = 0$$

Das implizite Eulerverfahren erfüllt für alle h die Stabililtätsbedingung, nämlich dass exakte und Näherungs-lösung für λ < 0 beschränkt bleiben!

Beispiel 2-dim linear:

$$\frac{d}{dx}\begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix} = A\begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}$$

oder

$$\begin{pmatrix} y_1'(x) \\ y_2'(x) \end{pmatrix} = \begin{pmatrix} \lambda_1 y_1(x) \\ \lambda_2 y_2(x) \end{pmatrix}$$

Dies entspricht zwei unabhängigen Diff'gleichungen, die hier zusammen in vektorieller Form z.B. mittels Euler gelöst werden:

$$\begin{pmatrix} y_1^{k+1} \\ y_2^{k+1} \end{pmatrix} = \begin{pmatrix} y_1^k \\ y_2^k \end{pmatrix} + h \cdot \begin{pmatrix} \lambda_1 y_1^1 \\ \lambda_2 y_2^k \end{pmatrix} = \begin{pmatrix} (1 + h\lambda_1) y_1^k \\ (1 + h\lambda_2) y_2^k \end{pmatrix}$$

Die Stabilitätsbedingung hängt hier also von beiden λ's ab:

$$h < 2 / |\lambda_i| \quad \text{für } i=1,2$$

oder

$$h < \min\{ 2 / |\lambda_1| , \; 2 / |\lambda_2| \}$$

Gilt nun $\lambda_2 << \lambda_1 < 0$, so wird die erlaubte Schrittweite h bestimmt von λ_2 , also

$$h < 2 / |\lambda_2| \; .$$

Andererseits ist die dazugehörige Komponente in der Lösung

$$const \cdot \exp(\lambda_2 x)$$

schon nach wenigen Schritten verschwindend klein und spielt dann für die eigentliche Lösung keine Rolle mehr!

λ_2 bestimmt dann also die Lösung nur in einem kleinen Bereich, die Schrittweite h beim Eulerverfahren aber überall!

Man spricht hier von ‚steifen' Diff.-gleichungen.

Offensichtlich ist in solchen Fällen das implizite Eulerverfahren wesentlich besser geeignet, da es keine Einschränkung der Schrittweite h durch die $|\lambda_i|$ beinhaltet.

7.2. Anwendungsbeispiele:

7.2.1. Räuber-Beute-Modell

Einfaches Modell, das die Populationsänderung in einem ‚Primitiv-Ökosystem', bestehend aus genau einer Räuberspezies und einer Beutespezies, beschreibt.

Viele Räuber → Beute nimmt ab
Wenig Räuber → Beute nimmt zu

Viel Beute → Räuber nehmen zu
Wenig Beute → Räuber nehmen ab

Änderung der Anzahl Beutetiere x(t) ist zum einen durch die Geburtenrate a proportional der Anzahl Beutetiere,
und zum anderen führt eine Begegnung eines Beutetiers mit einem Räuber (y(t)) zu einer Verringerung (mit Sterberate b):

$$\dot{x}(t) = ax - byx$$

Andererseits erhöht sich die Räuber-Geburtenrate durch viele Beutetiere,
während die Abnahme der Räuber von ihrer Sterberate d abhängt:

$$\dot{y}(t) = cxy - dy$$

Man erhält also zur Beschreibung der zeitlichen Änderung der Population von Räuber und Beute ein DGL-System:

$$\dot{x}(t) = ax(t) - bx(t)y(t)$$

$$\textit{für} \quad t_0 \le t \le t_n$$

$$\dot{y}(t) = cx(t)y(t) - dy(t)$$

mit Startwerten

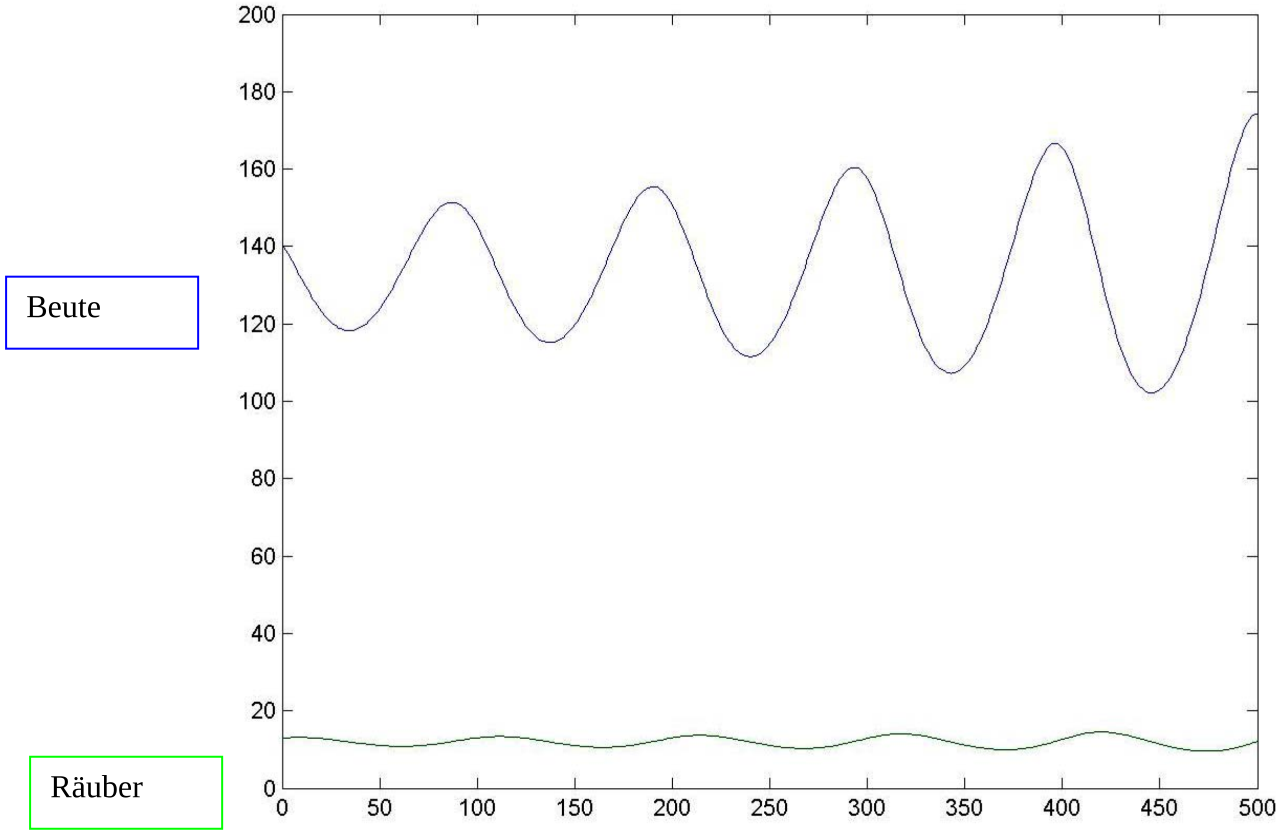
$$x(t_0) = x_0 \qquad \textit{und} \qquad y(t_0) = y_0$$

Eulerverfahren mit Schrittweite τ liefert dafür die Formel:

$$\frac{x_{i+1} - x_i}{\tau} = ax_i - bx_i y_i$$

$$\frac{y_{i+1} - y_i}{\tau} = cx_i y_i - dy_i$$

Fixpunkte dieser Vektoriteration: (x,y)=(d/c, a/b)
Populationsverlauf:
Parameter: a=.3, b=.025, c=.0015, d=.2



http://www.leinweb.com/snackbar/wator/