6.4. Eigenwerte und Vektoriteration

Eigenvektor $v \neq 0$ und Eigenwert λ einer quadratischen Matrix A erfüllen die Gleichung

$$A\,v = \lambda\,v$$

Daher ist die durch den Vektor v bestimmte Gerade durch den Nullpunkt eine sog. Fixgerade der durch

$$x \rightarrow A\,x$$

definierten Abbildung.

Für eine symmetrische Matrix A gilt:

Die Eigenvektoren der n Eigenwerte von A bilden eine Orthonormalbasis des R^n.

Eigenvektormatrix V liefert eine Diagonalisierung der Matrix A.

Die durch A definierte Abbildung wird in dieser Basis trivial:

$$A\,v = \lambda\,v \quad \rightarrow \quad A\,V = V\,\Lambda,$$

Λ ist Diagonalmatrix, mit den Eigenwerten als Diagonaleinträge

$$V^T A V = \Lambda = \mathrm{diag}(\lambda_1 , \ldots , \lambda_n)$$

Ein Eigenwert erfüllt die Gleichung $\det(A - \lambda I) = 0$,
da $A - \lambda I$ singulär: $(A - \lambda I)\, v = 0$

Also sind Eigenwerte genau die Nullstellen des *charakteristischen Polynoms* $p(\lambda) := \det(A - \lambda I)$.

Eigenwerte von A ←→ Nullstellen eines Polynoms p(x)

Die Matrix $A := \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \ddots & \vdots & -a_1 \\ 0 & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & 1 & 0 & -a_{n-2} \\ 0 & \cdots & 0 & 1 & -a_{n-1} \end{pmatrix}$

hat das charakteristisches Polynom

$$p(x) = (-1)^n \left(a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + x^n \right)$$

Daher lassen sich Nullstellen von Polynomen berechnen, indem man die Eigenwerte der obigen Matrix A berechnet!

Eigenwertberechnung ist numerisch stabiler als Nullstellenberechnung bei Polynomen!

Vektoriteration

Vektoriteration ist eine einfache Fixpunktiteration zur Berechnung des betragsgrößten Eigenwerts einer Matrix:

Sei A symmetrisch positiv definit ($x \neq 0 \rightarrow x^T Ax > 0$);

Start mit $x_0 \neq 0$, $x_{k+1} := \frac{Ax_k}{\|Ax_k\|}$;

Die Iterationsfunktion: $\Phi(x) = Ax / \|Ax\|$

hat Fixpunkt v, wenn $v = Av / \|Av\|$

d.h. v ist Eigenvektor von A zu Eigenwert $\lambda = \|Av\|$,
denn dann gilt : $A v = \|Av\| * v$.

Offensichtlich ist dann $x_k = \text{const} * A^k x_0$

Außerdem kann x_0 in der Basis der Eigenvektoren dargestellt werden:

$$x_0 = c_1 v_1 + \ldots + c_m v_m$$

wobei λ_1 maximaler Eigenwert von A zu Eigenvektor v_1 ist.

Daher gilt $Av_j = \lambda_j v_j$ und

$$A^k x_0 = c_1 A^k v_1 + \ldots + c_m A^k v_m =$$

$$= c_1 \lambda_1^k v_1 + \ldots + c_m \lambda_m^k v_m =$$

$$= \lambda_1^k \left(c_1 v_1 + \ldots + c_m \left(\lambda_m / \lambda_1\right)^k v_m \right) \overset{k\to\infty}{\to} \lambda_1^k \left(c_1 v_1 \right)$$

$\lambda_m / \lambda_1 \to 0$

Normiert man,dann folgt $\dfrac{A^k x_0}{\left\| A^k x_0 \right\|} \to \dfrac{\lambda_1^k \left(c_1 v_1 \right)}{\left\| \lambda_1^k \left(c_1 v_1 \right) \right\|} = \dfrac{c_1}{|c_1|} v_1 = \pm v_1$

x_k ist normierter Vektor zu $A^k x_0$.
Dann bleibt in x_k nur die Komponente zum stärksten Eigenwert übrig, nämlich v_1 .

Also $x_k \rightarrow$ const $\cdot v_1$ konvergiert gegen Eigenvektor,
und daher auch

$\| Ax_k \| \rightarrow \lambda_1$ konvergiert gegen größten Eigenwert.

Ähnliches gilt für allgemeine Matrizen, solange die Matrix einen eindeutigen betragsgrößten Eigenwert hat

(also z.B. nicht : $\pm\lambda_1$ sind beide Eigenwerte)

Beispiel:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1/2 \end{pmatrix}^k \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2^{-k} \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

ist Eigenvektor zu Eigenwert $\lambda = 1$

Beispiel:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}^k \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ (-1)^k \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ \pm 1 \end{pmatrix}$$

Im Eigenraum zu Eigenwerten $\lambda = 1$ und -1

Für innere Eigenwerte:

Wende Vektoriteration auf die Matrix $(A-\sigma I)^{-1}$ an.

Liefert deren größten Eigenwert.

Dies ist der Eigenwert von A, der am nächsten bei σ liegt.

Dies ist die sog. Inverse Iteration.

Problem: In jedem Schritt ist ein Gleichungssystem zu lösen
$(A-\sigma I)$ schlecht konditioniert, ev. singulär!

Anwendungen

Resonanzen, siehe Tacoma, London Millenium Bridge

Energieniveaus in der Quantenmechanik

Nullstellen von Polynomen

Biegen eines Balkens - Hauptträgheitsachsen

Stabilität eines Systems, Differentialgleichungen

Modellreduktion, Principal Component Analysis PCA, Eigenface, finger print

Analyse von Graphen, Pagerank

Unterschiedliche Aufgabestellung:

Alle Eigenwerte/vektoren oder nur einige!

Eigenwerte alleine oder mit Eigenvektoren

QR-Verfahren

Gesucht sind alle Eigenwerte/vektoren!

1. Schritt: transformiere A auf Tridiagonalform (obere Hessenberg) ohne die Eigenwerte zu verändern?

Am besten orthogonale Basistransformation mit orthonormalem Q:

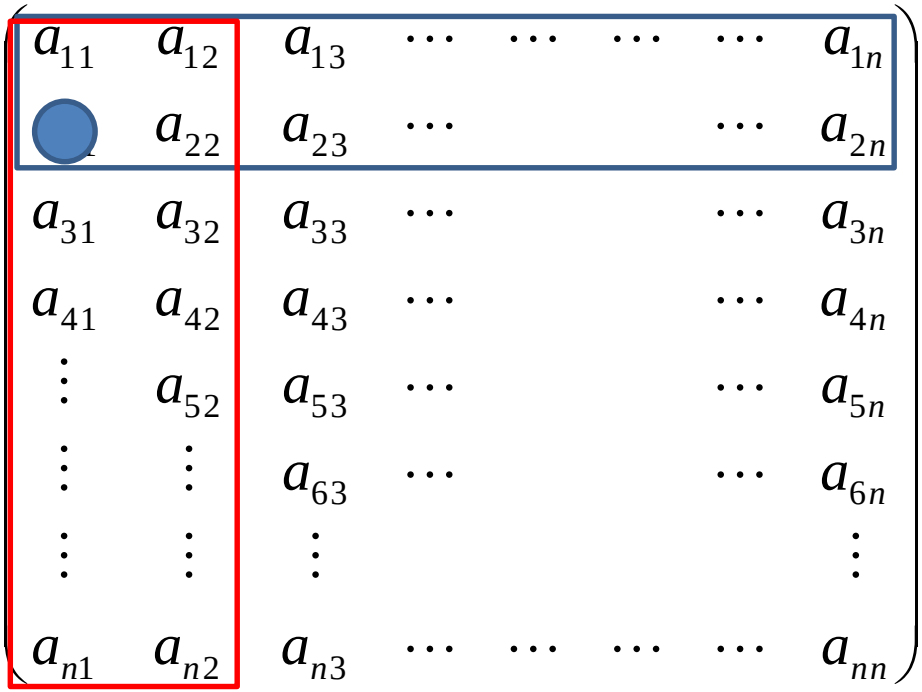
$$A_{neu} = Q*A*Q^T$$

$$Ax = \lambda x \Leftrightarrow AQ^T(Qx) = \lambda Q^T(Qx) \Leftrightarrow$$

$$\Leftrightarrow A_{neu}y = QAQ^T y = \lambda y, \qquad y = Qx$$

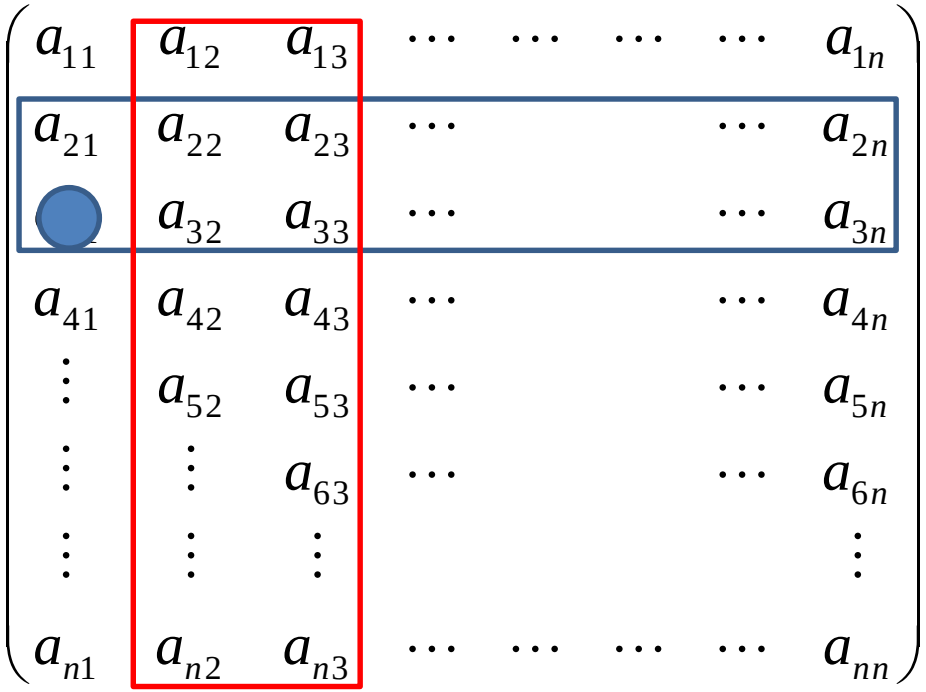
Matrix wie $Q*A*Q^T$, Eigenwert gleich, Eigenvektor wie Qx

Welches Q? Givens?



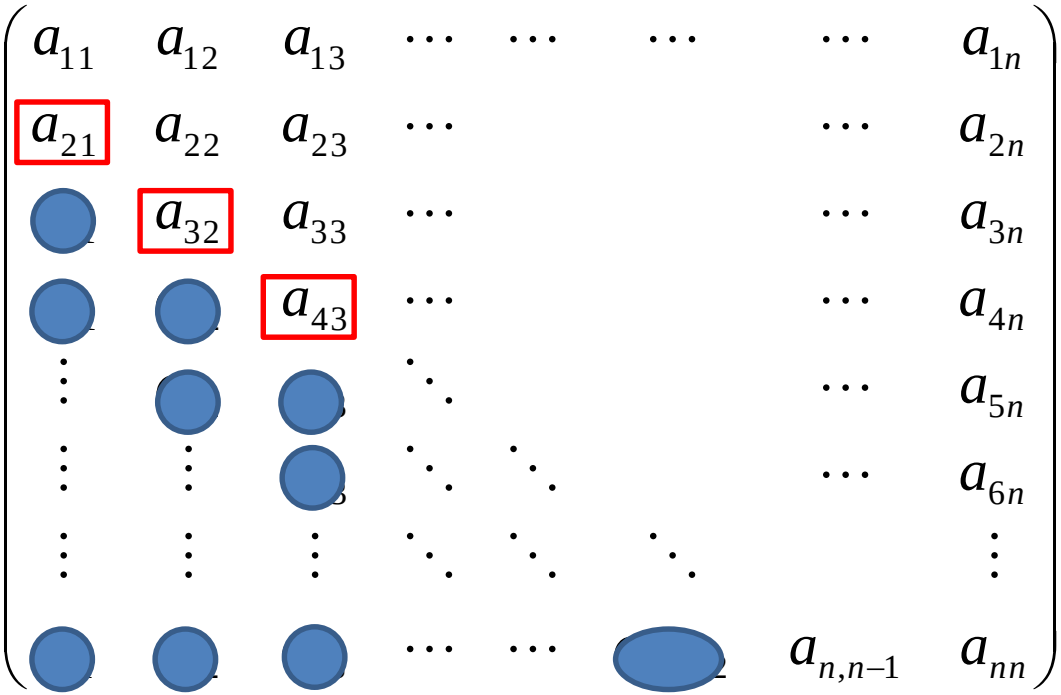
Von links Givens zur Elimination von a_{21}?
Anwendung von rechts füllt a_{21} wieder auf!

Besser:



Von links Givens zur Elimination von a_{31}?
Anwendung von rechts verändert a_{21} nicht!

Insgesamt:



Führt im Endeffekt auf Tridiagonalform für symmetrisches A, bzw. obere Hessenbergform für allgemeines A.

2. Schritt: QR-Verfahren:

Nachdem A auf Tridiagonal(Hessenberg)gestalt transformiert ist:

Berechne zu A die QR-Faktorisierung A=QR und setze

$$A_{neu} = RQ$$

Daher gilt $R=Q^TA$.

Damit ist $A_{neu} = Q^TAQ$ mit denselben Eigenwerten!

Wiederhole iterativ.

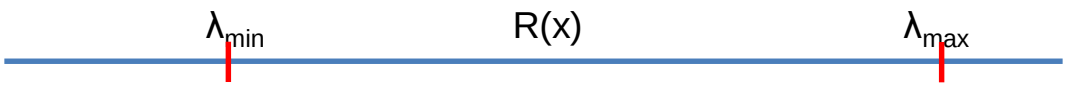
Im Endeffekt konvergiert A gegen Diagonalmatrix mit den Eigenwerten auf der Diagonalen.

Einige zusätzliche Tricks dabei!

Rayleigh Quotient

$$R(x) = \frac{x^T A x}{x^T x}$$ ist für symmetrisches A eine Funktion

$$R: \mathbb{R}^n \rightarrow \mathbb{R}$$



Das Maximum von R(x) ist der größte Eigenwert, das Minimum der kleinste.

R(x) ist der Range/Wertebereich der Matrix A, enthält alle Eigenwerte.

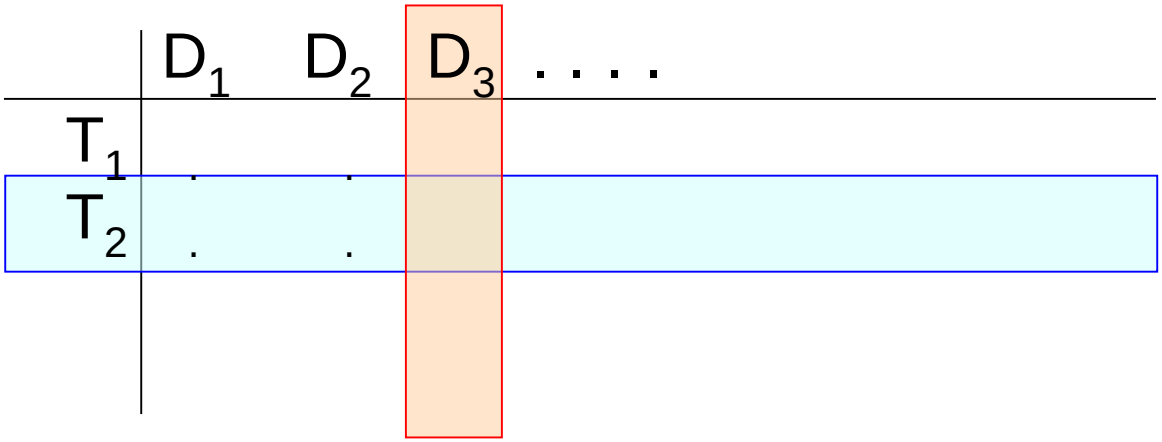
6.5.4. Information Retrieval und Vektoriteration:

Vektorraummodell für ‚Data Mining'

Liste von Dokumenten D_1 , ... , D_n , z.B. WWW-Seiten
Liste von Suchtermen T_1 , ... , T_m , z.B. alphabetisch geordnet:
Aachen, ABC, Auto, Bar, ... , Zug

Jeder Suchbegriff erscheint in jedem Dokument mit einem Bestimmten Gewicht, z.B. Gewicht ~ Häufigkeit.

Sammle diese Gewichte in der sog. Term-Dokument-Matrix (dünnbesetzte Matrix).
j-ter Spaltenvektor dieser Matrix listet für das j-te Dokument D_j die Darin enthaltenen Suchworte → *Vektor d_j .*
k-ter Zeilenvektor t_k listet für den k-ten Suchbegriff die relevanten Dokumente auf.



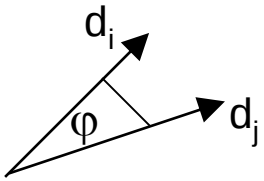
Vergleich zweier Dokumente auf Ähnlichkeit:

Dokument D_i repräsentiert durch Vektor der darin enthaltenen Suchbegriffe d_i , entsprechend d_j .

Vektoren ähnlich (kollinear), wenn Winkel zwischen ihnen sehr klein, d.h.

$$\cos(\sphericalangle(d_i, d_j)) = \frac{d_i^T d_j}{\|d_i\|_2 \cdot \|d_j\|_2}$$

nahe bei 1.



Eine Suchanfrage ist ein Spaltenvektor d, entspricht quasi einem Dokument, das mit anderen Dokumenten auf Ähnlichkeit verglichen werden soll.

Berechne Vektor $t = d^T*D = (d^T*D_1 \;\; d^T*D_2 \;\; ..)$.

Große Komponenten t_j entsprechen Dokumenten D_j , die mit der Suchanfrage am besten übereinstimmen.

Matrix D enthält sehr viel ‚Rauschen', (zufälliges Auftreten von Suchtermen in Dokumenten, Wortwahl, ...)

Modellreduktion:
Ersetze D durch System, das die *wesentlichen* Eigenschaften repräsentiert:

Berechne QR-Zerlegung von D : D=QR
Partitioniere R in der Form

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

dabei sei R_{22} gleich Null oder mit kleinen Einträgen. Dann repräsentieren die Zeilen von R_{22} ‚linear abhängige' Such-begriffe, die eigentlich überflüssig sind.

Ersetze D durch die Approximation kleinen Ranges:

$$\tilde{D} := Q\left(\begin{array}{cc} R_{11} & R_{12} \\ \hline 0 & 0 \end{array}\right) = \left(Q_1 \mid Q_2\right)\left(\begin{array}{cc} R_{11} & R_{12} \\ \hline 0 & 0 \end{array}\right) = Q_1 \cdot \left(R_{11} \quad R_{12}\right)$$

und untersuche $t^T = d^T \cdot \tilde{D}$ anstatt $d^T \cdot D$

Verbesserung durch Pivotsuche: D*P=Q*R mit allgemeiner QR-Zerlegung incl. Permutation liefert $\tilde{D} := Q \cdot \tilde{R} \cdot P^T$ und neues t^T

Andere Methode des ‚Denoising' mittels Eigenwerten:

Betrachte Singulärwertzerlegung von A:

$$A = U\,D\,V,$$

dabei sind U und V Eigenvektormatrizen von A^TA und AA^T, und D ist eine rechteckige Diagonalmatrix, deren Diagonaleinträge >= 0 sind:

$$A = U \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots \\ 0 & \ddots & \ddots & \vdots & \\ \vdots & \ddots & \sigma_k & 0 & \\ 0 & \cdots & 0 & 0 & \ddots \\ \vdots & & & \ddots & \ddots \end{pmatrix} V$$

mit Singulärwerten $\sigma_j > 0$, $\mathrm{Rang}(A)=k$.

Ersetze D durch $\tilde{D}$, indem kleine σ_j durch 0 ersetzt werden.

Ergibt neue Matrix (Term-Dokument-Tabelle)

$$\tilde{A} = U\tilde{D}V$$ mit kleinerem Rang:

Diese neue Matrix enthält wieder die wesentliche Information von A .

A = UDV heißt Singulärwertzerlegung von A.
Und ergibt sich aus der Eigenwert-Zerlegung von A^TA, bzw. AA^T .

VII. Numerische Behandlung von Differentialgleichungen

7.1. Gewöhnliche Diff'gleichungen (erster Ordnung)

Frage: Warum Differentialgleichungen?

Diff'gleichungsproblem allgemein: Aus Beziehungen zwischen den Änderungen (Ableitungen) und den Funktionswerten soll eine gesuchte Funktion bestimmt werden!

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

$$y' = \varphi(y, x) \quad \rightarrow \quad \text{y(x) ?}$$

Ableitung von y(x) nach x in jedem möglichen Punkt (x,y) ist gegeben durch Funktion φ(x,y).

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.

$$v(t) = \dot{s}(t) = \frac{ds}{dt},$$

$$-g = \dot{v}(t) = \frac{dv}{dt} = \ddot{s}(t) = \frac{d^2 s}{dt^2}.$$

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

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

Diff'gleichung für s(t):

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

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

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: $y'(x) = ...$

Impliziter Fall: $\varphi(y',y,x) = 0$ → 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 der Variablen)

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

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

$$\ln(y(x)) - \ln(y(x_0)) = \ln(y(x))\Big]_{x_0}^{x} = \alpha \cdot t\Big]_{x_0}^{x} = \alpha(x - x_0)$$

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

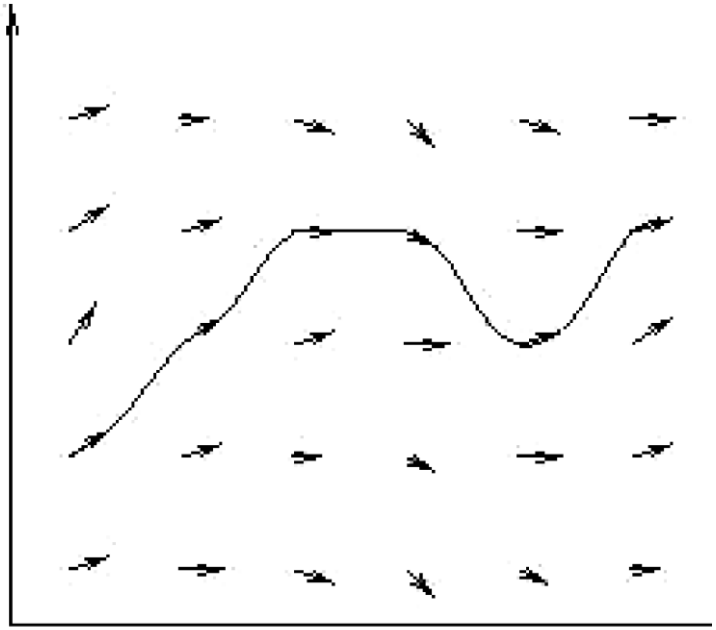
Beachte: y manchmal als einfache Variable,
manchmal als Funktion y(x).

7.1.4. Geometrische Problemstellung:

Von einer Funktion sind alle möglichen Ableitungswerte an allen möglichen Stellen (x,y) bekannt
(also überall an allen potentiellen Funktionswerten)
Dies entspricht einem Vektorfeld von Tangentenrichtungen

Weiterhin bekannt ist der Funktionswert an einer Stelle x_0 .

Gesucht: Kurve durch (x_0,y_0) , deren Tangenten in allen Punkten dem vorgegebenen Vektorfeld entspricht!



Im Beispiel ‚Freier Fall' ist dieses Vektorfeld trivial:

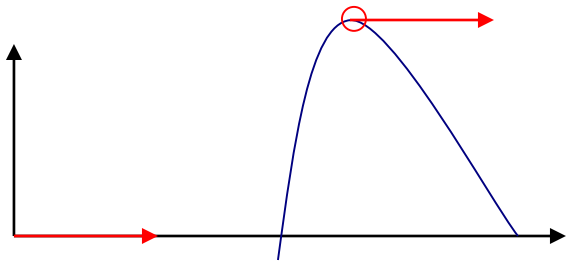
$$\dot{v}(t) = -g, \qquad v(t_0) = 0.$$

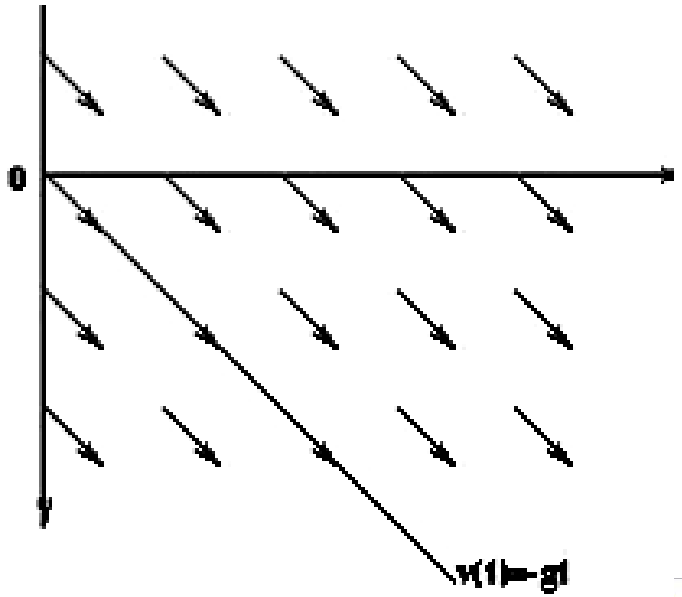
Kurve (t,v)←→v(t)

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

Tangentenvektor (t,v)‘ = (1,v‘)

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





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

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

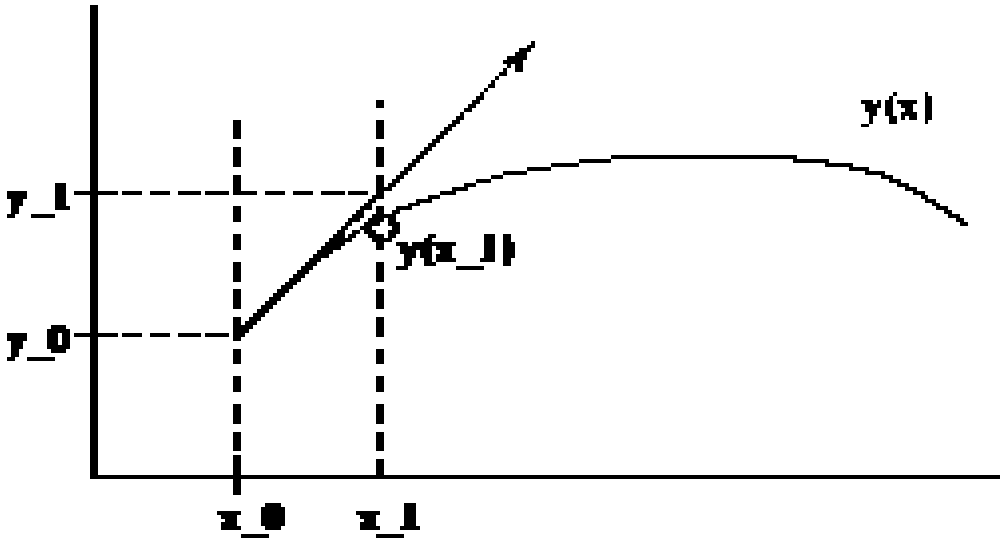
$$y(x_1) \approx y_1 = g(x_1) = y_0 + y'(x_0)(x_1 - x_0) = y_0 + h \cdot \varphi(y_0, x_0)$$

Ersetze wieder Funktion lokal durch Tangentengerade!

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

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

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

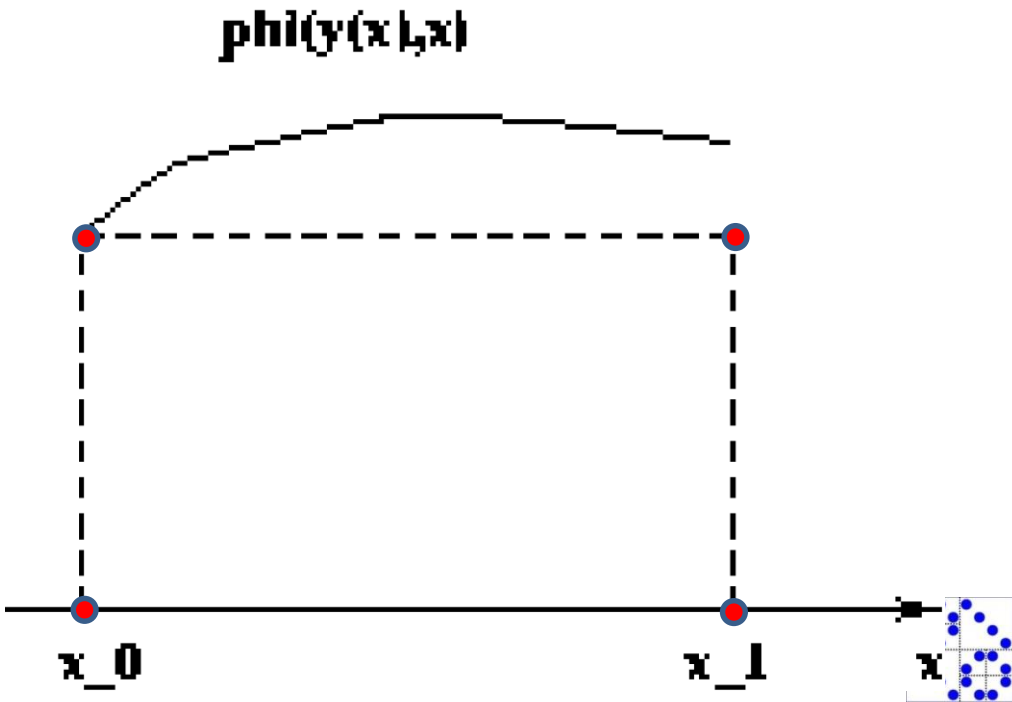


Euler aus Integration:

Betrachte die Diff'gleichung zwischen x_0 und $x_1=x_0+h$:

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

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 :

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

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:

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

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 $$y_{k+1} = y_k + h \cdot \varphi(y_{k+1}, x_{k+1})$$

Vorteile? Nachteile?

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

$$y_{k+1} - \left(y_k + h \cdot \varphi(y_{k+1}, x_{k+1})\right) = 0$$

Um y_{k+1}(=z) 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 $\bar{z}$ 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.

Explizites Euler als „Predictor“ liefert Startwert für implizites Euler / Newtonverfahren als „Corrector“

7.1.6.2. Weitere Taylorpolynom-Verfahren:

Berücksichtigung höherer Terme in der Taylorentwicklung:

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

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

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

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 durch Mehrfachauswertung von φ.
(vgl. Newton/Sekanten-verfahren)

Beispiel:

$$y'(x) = \varphi(y, x) = yx;$$

$$x_0 = 0; y(0) = 1;$$

Lösung:

$$\frac{dy}{y} = x \cdot dx \Rightarrow \ln(y(x)) - \ln(y(x_0)) = \frac{x^2}{2}\Big]_{x_0}^{x}$$

$$\Rightarrow y(x) = y(x_0)\exp\left(\frac{x^2 - x_0^2}{2}\right);$$

Euler:

$$y_{k+1} = y_k + h\varphi(y_k, x_k) = y_k + hy_k x_k = (1 + hx_k)y_k$$

Rückwärts-Euler:

$$y_{k+1} = y_k + h\varphi(y_{k+1}, x_{k+1}) = y_k + hy_{k+1}x_{k+1};$$

$$y_{k+1} = \frac{y_k}{1 - hx_{k+1}};$$

Beispiel:

$$y'(x) = \varphi(y, x) = yx;$$

$$x_0 = 0; y(0) = 1;$$

$$\frac{\partial}{\partial y}\varphi(y,x) = \frac{\partial}{\partial y}(yx) = x; \qquad \frac{\partial}{\partial x}\varphi(y,x) = \frac{\partial}{\partial x}(yx) = y;$$

$$\frac{d}{dx}\varphi(y(x),x) = \frac{\partial}{\partial y}\varphi \cdot y' + \frac{\partial}{\partial x}\varphi = x(yx) + y = y(x^2+1);$$

Taylor 2. Ordnung:

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

$$= y_k + hy_k x_k + \frac{h^2}{2}\left(y_k\left(x_k^2+1\right)\right) = y_k\left(1 + hx_k + \frac{h^2}{2} + \frac{h^2}{2}x_k^2\right);$$

MATLAB: gewdiff_tayl.m

7.1.6.3. Mehrschrittverfahren:

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

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

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

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

Idee von Mehrschrittverfahren:

Vermeide Berechnung höherer Ableitungen und benutze die berechneten Funktionswerte selbst, um den weiteren Verlauf der Lösung anzunähern.
Also finde beste Gerade/Parabel …, die durch die bisher berechneten Punkte geht.

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 $$y'(x) = \begin{pmatrix} y_1'(x) & \cdots & y_n'(x) \end{pmatrix}^T = \varphi(y, x) =$$

$$= \begin{pmatrix} \varphi_1(y_1,...,y_n,x) & \cdots & \varphi_n(y_1,...,y_n,x) \end{pmatrix}^T$$

mit vorgegebenem Startvektor

$$y(x_0) = \begin{pmatrix} y_1(x_0) & \cdots & y_n(x_0) \end{pmatrix}^T$$

Lösung genauso durch Euler im IR^n:

$$y^{k+1} = \begin{pmatrix} y_1^{k+1} \\ \vdots \\ y_n^{k+1} \end{pmatrix} = \begin{pmatrix} y_1^{k} \\ \vdots \\ y_n^{k} \end{pmatrix} + h \cdot \begin{pmatrix} \varphi_1(y_1^k,...,y_n^k,x_k) \\ \vdots \\ \varphi_n(y_1^k,...,y_n^k,x_k) \end{pmatrix}$$