Ü 7 Siebte Übungseinheit

Inhalt der siebten Übungseinheit:

- Eigenwertaufgaben
  - Anschauliche Erklärung: `eigshow`
  - Vektoriteration, QR-Iteration
  - Einige Anwendungen
- Einschrittverfahren für gewöhnliche Differentialgleichungen erster Ordnung
  - MATLAB-Löser
  - Klassisches Euler-Verfahren und andere einfache explizite Verfahren

Hinweis: Die Unterlagen verweisen auf Seiten im Vorlesungsskriptum. Seitenzahlen beziehen sich auf die aktualisierten Kapitel, die über die Homepage des Lehrstuhls (klick!) abrufbar sind.

Ü 7.1 Eigenwerte und Eigenvektoren

Was Sie an theoretischen Grundlagen für diese Einheit brauchen, fasst Abschnitt 9.1 (klick!) im aktuellen Skriptum zusammen. Siehe dazu auch die Folien der 8. Vorlesung (klick!). Informieren Sie sich dort über die Definitionen und grundlegenden Eigenschaften! Die Übungsaufgaben stellen den Zusammenhang zu typischen Anwendungen her.

MATLAB-Befehle:

`d = eig(A)` liefert Vektor von Eigenwerten

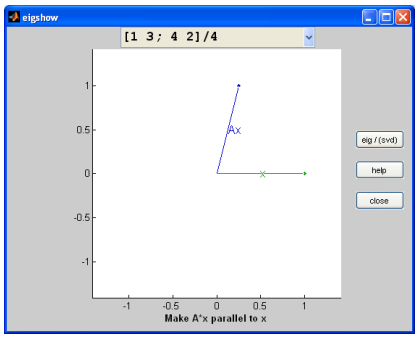
`[V,D] = eig(A)` Spalten von V sind Eigenvektoren, Diagonalelemente von D sind Eigenwerte.

Ein interaktives, recht lehrreiches Beispiel zu Eigenvektoren liefert das MATLAB-Demoprogramm `eigshow`. Laden Sie von der Übungs-Homepage (klick!) die Unterlagen zur 7. Übung herunter und starten Sie `eigshow.m`. Tipp: Unter `http://blogs.mathworks.com/cleve/2013/07/08/eigshow-week-1` erklärt Ihnen Cleve Moler, einer der MathWorks-Gründer, persönlich, worum es in eigshow geht.

Bewegen Sie den Vektor $\mathbf{x}$ und beobachten Sie, wie sich $A\mathbf{x}$ entsprechend ändert. Drehen Sie den Vektor $\mathbf{x}$ so, dass er zu $A\mathbf{x}$ parallel liegt. Jeder solche Vektor $\mathbf{x}$ ist ein *Eigenvektor* der Matrix A.
Das Ergebnis $A\mathbf{x}$ ist in diesem Fall ein Vielfaches der Ausgangsvektors, es gilt also die Gleichung

$$A\mathbf{x} = \lambda\mathbf{x} \ .$$

Der Proportionalitätsfaktor λ ist der zu $\mathbf{x}$ gehörende *Eigenwert* von A.



Die Matrix A können Sie im Fenster oben aus einer Liste wählen. Voreingestellt ist

$$A = \frac{1}{4}\begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix} \ .$$

- Finden Sie für die obige Matrix A Eigenvektoren. Lesen Sie die Komponenten von $\mathbf{x}$ ab und schätzen Sie λ. Wie viele verschiedene Werte von λ gibt es hier?
- Vergleichen Sie mit dem Resultat des Befehls `[V,D] = eig(A)`.
- Versuchen Sie andere Matrizen aus der Liste. Nicht immer können Sie Eigenvektoren finden. Es gibt Beispiele, wo in *gar keiner* Richtung $\mathbf{x}$ und $\mathbf{Ax}$ auf einer Linie liegen. Was liefert `[V,D] = eig(A)` in solchen Fällen?
- Im Normalfall (bei regulärer Matrix A) beschreibt $A\mathbf{x}$ eine Ellipse, wenn Sie $\mathbf{x}$ bewegen. Singuläre Matrizen bilden die Ausnahme. Sie haben $\lambda = 0$ als Eigenwert. Wie sieht die Bahn von $A\mathbf{x}$ dann aus? Finden Sie eine Beispielmatrix aus der Liste.

Aufgabe 58: Spannungstensor, Hauptachsentransformation

Diese Aufgabe testet, ob Sie die MATLAB-Befehle aufrufen und deren Ergebnisse interpretieren können.

Reine Druck- oder Zugkräfte wirken normal auf ein Flächenelement, reine Scherkräfte parallel dazu. In einem elastisch deformierten Körper kann die Kraft auf ein Flächenelement irgendwie schräg wirken. Der Spannungstensor stellt den Zusammenhang zwischen Flächennormalrichtung und Kraftrichtung her:

Kraftvektor = Spannungstensor mal Normalenvektor des Flächenelements

In einem geeignet gedrehten Koordinatensystem nimmt der Spannungstensor Diagonalgestalt an. Die Achsen dieses Koordinatensystems sind die *Hauptachsen* des Spannungstensors. Sie zeigen in Richtung seiner Eigenvektoren. Der diagonalisierte Spannungstensor enthält die Eigenwerte in der Hauptdiagonalen.

Der Spannungstensor in einem Material sei (in willkürlichen Einheiten)

$$p = \begin{bmatrix} 5 & 10 & -8 \\ 10 & 2 & 2 \\ -8 & 2 & 11 \end{bmatrix}.$$

a) Welcher Kraftvektor wirkt auf ein Flächenelement parallel zur xy-Ebene? — zur Ebene $x + y + z = 0$? (Dazu brauchen Sie noch keine Eigenwerte.)

b) In welche Richtungen zeigen die Hauptachsen des Spannungstensors (`format rat` liefert „schöne“ Zahlen)? Wie lautet der auf Diagonalgestalt transformierte Tensor?

c) In zwei Richtungen wirkt reine Zug-, in einer reine Druckspannung (Vorzeichenkonvention bei den Diagonaltermen des Spannungstensors: Druck ist negativ[36]. In welcher Richtung wirkt
— die größere Zugspannung?
— die kleinere Zugspannung?
— die Druckspannung?

[36] wie auch sonst im Leben: Prüfungsdruck, Leistungsdruck, Erfolgsdruck,...

Aufgabe 59: Vektoriteration

Was passiert, wenn man einen Vektor wieder und wieder mit derselben Matrix multipliziert? Testen Sie für den Vektor $x = (1, 0, 0)^T$ und die Matrix A

$$A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

Wenn man den Befehl `x = A*x` zehnmal iteriert, werden die Zahlen in x schon recht groß.

Dividieren Sie nun zusätzlich in jedem Schritt den Vektor x nach der Multiplikation mit A durch seine erste Komponente. Das ändert nicht die Richtung von x, hält aber die Zahlenwerte der Komponenten in übersichtlichen Grenzen. Sie können das in der Form `>> y=A*x; x = y/y(1)` im Workspace durchspielen. Was beobachten Sie? Zu welchem Wert konvergiert y_1?

Iterieren Sie, beginnend mit dem Startvektor $x = (1, 0, 0)^T$,

```
>> y=A*x; x = y/y(1)
```

Zu welchem Wert konvergiert y_1?

Vergleichen Sie: Für diese Matrix berechnet das Skriptum (klick!) auf Seite 78 die Eigenwerte aus dem charakteristischen Polynom.

Im Skriptum auf Seite 80 ist Vektoriteration (in etwas allgemeinerer Fassung) als Pseudocode gegeben. Dabei ist noch berücksichtigt, dass die einfache Skalierung `x = y/y(1)` nicht funktioniert, falls $y(1) = 0$.

Die Beispielmatrix $A = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}$ im Skriptum auf Seite 76 ist so ein Fall, wo Skalierung mit der ersten Komponente nicht zielführend ist. Rechnen Sie wie oben einige Schritte, dann sehen Sie, warum.

Ein Computerprogramm skaliert am besten mit jener Komponente, die größten Absolutbetrag hat. Der MATLAB-Befehl `[m,i]=max(abs(x))` findet im Vektor x den Index i dieser Komponente.
Eine simple MATLAB-Implementierung dazu wäre

```
for k=1:100
    x1 = A*x0;
    [m,i]=max(abs(x1));
    lam = x1(i);

    x1 = x1/lam;
    if(norm(lam - lam0)<eps)
        break
    end
    x0=x1;
    lam0=lam;
end
```

Konkrete Aufgabe Schreiben Sie ein Skript, das eine 10×10-Matrix A mit zufällig gewählten ganzzahligen Elementen $a_{ij} \in \{0, \ldots, 9\}$ erzeugt[37]. Berechnen Sie mit Vektoriteration den betragsgrößten Eigenwert und einen zugehörigen Eigenvektor. Wählen Sie ein Abbruchkriterium mit Genauigkeit $\epsilon < 10^{-6}$.

Lassen Sie auch MATLABs `eig`-Befehl Eigenwert und Eigenvektor berechnen und vergleichen Sie die Ergebnisse. Beachten Sie, dass MATLAB Eigenvektoren als Einheitsvektoren in der 2-Norm, während die oben gezeigte Vektoriteration in der Unendlichnorm auf 1 skaliert. Zum Vergleich der Eigenvektoren müssen Sie MATLABs Eigenvektor gleich skalieren wie den aus der Vektoriteration berechneten!

Aufgabe 60: Ein Top-10-Algorithmus: Eigenwerte aus QR-Iteration

Eigenwertprobleme für 2×2- oder 3×3-Matrizen sind Ihnen bei der Hauptachsentransformation des Spannungstensors (vergleiche Aufgabe 58) begegnet. Darüber hinaus gab es bis etwa 1920 aus Sicht der technisch-wissenschaftlichen Anwendungen keine besondere Motivation, Eigenwerte größerer Matrizen zu berechnen. Aber dann nahm die Wichtigkeit von Eigenwertaufgaben dramatisch zu: es kamen Quantentheorie, Matrizenmechanik, Finite-Elemente-Methoden und, ganz aktuell, Data Science.

Der klassische Lösungsweg über das charakteristische Polynom ist unter anderem deswegen für größere Matrizen ungeeignet, weil die Nullstellen extrem empfindlich gegenüber kleinen Störungen in den Polynomkoeffizienten sein können (vergleiche Übungsbeispiel 13).

Um 1960 entstand der QR-Algorithmus, der es in die Top-10 Algorithmen von größtem Einfluss auf Wissenschaft und Technik im 20. Jahrhundert geschafft hat[38][39]. Was die Experten dabei auch begeistert, ist: es handelt sich um einen tatsächlich völlig neuen Lösungsansatz und nicht bloß (wie uns hier schon häufig begegnet) eine verfeinerte Ausarbeitung der Ideen von Gauss oder Newton.

Sie haben die QR-Zerlegung einer Matrix bereits bei der Lösung überbestimmter Gleichungssysteme kennengelernt. (Vergleiche Folien zur 5. Vorlesung). Für die rechnerische Ausführung haben die Unterlagen bis jetzt immer nur auf MATLABs Befehl `[Q R]=qr(A)` verwiesen. Aber die QR-Zerlegung lässt sich in wenigen Zeilen implementieren. Auch das soll in dieser Aufgabe angesprochen werden. (Vergleiche Übungsaufgabe 32, sie befasst sich mit den Codezeilen für LR-Zerlegung).

Bemerkung: Die Matrix R in der Zerlegung $A = Q \cdot R$ ist **nicht** dieselbe wie in der Zerlegung $A = L \cdot R$. (Die englischsprachigen Literatur schreibt die LR-Zerlegung als $A = L \cdot U$; dort kann diese Verwirrung gar nicht erst entstehen.)

Diese Aufgabe lässt Sie mit der Basis-Version des QR-Algorithmus experimentieren. In seiner einfachsten Form zerlegt er eine Matrix in das Produkt einer orthogonalen und einer rechten oberen Dreiecksmatrix, $A = Q \cdot R$. Anschließend multipliziert er die Faktoren in umgekehrter Reihenfolge. Diese beiden Schritte werden iteriert.

---

[37] Wir verwenden Zufallsmatrizen hier nur als Testprobleme für den Vektoriterations-Algorithmus. Das ist bei weitem nicht die interessanteste Anwendung. In einem zufälligen Gespräch in Princeton 1972 entdeckten der Mathematiker Hugh Montgomery und der Physiker Freeman Dyson eine überraschende Verbindung zwischen Nullstellen der Riemannschen Zetafunktion, Eigenwerten von Zufallsmatrizen und quantenphysikalischen Systemen *(Montgomery's pair correlation conjecture)*

[38] J. Dongarra and F. Sullivan, "Guest Editors Introduction to the top 10 algorithms" in Computing in Science & Engineering, vol. 2, no. 01, pp. 22-23, 2000.

[39] B. Parlett, "The QR Algorithmïn Computing in Science & Engineering, vol. 2, no. 01, pp. 38-42, 2000.

Konkrete Aufgabe Schreiben Sie ein Skript, das eine *symmetrische*[40] 10×10-Matrix A mit zufällig gewählten ganzzahligen Elementen $a_{ij} \in \{0, \ldots, 9\}$ erzeugt; eine einfache Möglichkeit:

```
A = randi(5,10)-1;
A=A+A';
```

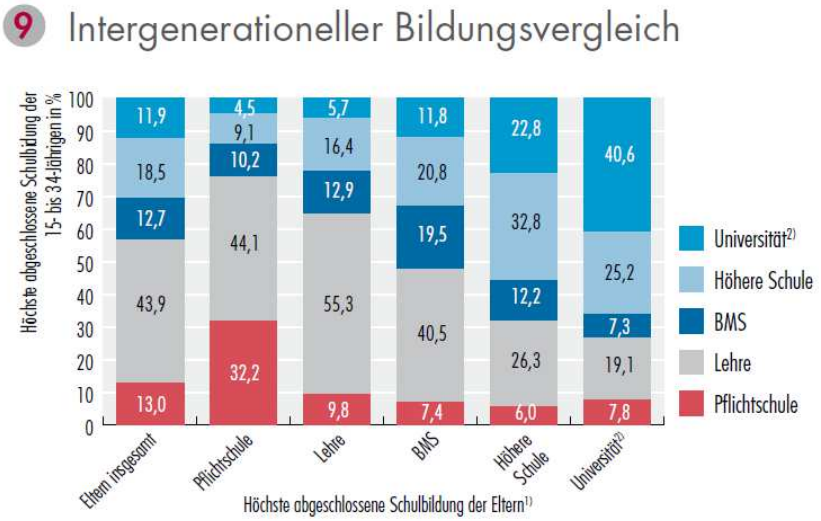
Lassen Sie Matlab die Eigenwerte von A bestimmen; merken Sie sich den Ergebnis-Vektor. Nun wiederholen Sie 100 mal die Schritte

```
[Q, R]=qr(A);
A=R*Q;
```

- Sehen Sie sich bei mehreren Versuchen mit unterschiedlichen Zufalls-Matrizen jeweils die Struktur der Ergebnis-Matrix an: In welchen Positionen treten Werte näherungsweise gleich Null auf? Wo immer, wo meistens?
- Vergleichen Sie die Diagonalelemente der Ergebnis-Matrix mit den Eigenwerten der Original-Matrix. Was können Sie erkennen?
- Ersetzen Sie MATLABs `qr`-Befehl durch den Aufruf von `myQR` (Im Material zur 7. Übung). Das ist eine einfache Implementierung der QR-Zerlegung. Prüfen Sie nach: An den Aussagen zu den zwei vorherigen Punkten sollte sich dadurch nichts ändern.

Aufgabe 61: Bildungsmobilität zwischen den Generationen

Es hängt stark von der sozialen Herkunft ab, welche Ausbildung Kinder und Jugendliche erhalten. Die Abbildung[41] zeigt Daten aus Österreich.



Q: STATISTIK AUSTRIA, Mikrozens.-Arbeitskräfteerhebung Ad-hoc-Modul „Eintritt junger Menschen in den Arbeitsmarkt" – 2. Quart. 2009. Bevölkerung in Privathaushalten (15- bis 34-Jährige). – Pers. nicht in Ausbild. – 1) Höchste abgeschl. Schulbildung d. Eltern: Haben Mutter und Vater nicht denselben Ausbildungsabschluss, wird jeweils die höhere Ausbildung verwendet. – 2) Inkl. hochschulverwandte Lehranstalten.

[40] Der QR-Algorithmus kann auch unsymmetrische Matrizen behandeln; unsere Basisversion funktioniert besserr für symmetrische Matrizen.

[41] Quelle: *Bildung in Zahlen 2010/11, Schlüsselindikatoren und Analysen.* Statistik Austria, Wien, 2012

Erstellen Sie aus diesen Daten eine Matrix $A = [a_{ij}]$, in der das Element a_{ij} angibt:

> Von Kindern, deren Eltern Ausbildung j haben, erreicht der Anteil a_{ij} die Ausbildung i.

Beispiel: Kinder aus Akademikerfamilien ($j = 5$) erreichen zu 19,1% eine Lehre ($i = 2$) als höchste Bildungsstufe; daher $a_{25} = 0{,}191$.

Der Balken ganz links in der Graphik gibt den Ist-Zustand (Bildungsniveau der Jugendlichen insgesamt) an. Setzen Sie diese Daten in einen Vektor $\mathbf{x}^{(1)}$ ein. Angenommen, das Bildungsniveau der Eltern insgesamt ist durch einen Vektor $\mathbf{x}^{(0)}$ beschrieben (diese Daten lassen sich nicht direkt aus der Grafik ablesen).

a) Begründen Sie: Das Bildungsniveau der nächsten Generation insgesamt errechnet sich durch Matrix-Vektor-Multiplikation $\mathbf{x}^{(1)} = A \cdot \mathbf{x}^{(0)}$.

b) Berechnen Sie $\mathbf{x}^{(0)}$ aus den gegebenen Daten für A und $\mathbf{x}^{(1)}$.

c) Berechnen Sie, ausgehend von $\mathbf{x}^{(1)}$, den Zustand nach 1, 2 und 3 weiteren Generationen;

d) Ein *stabiler Zustand* besteht, wenn sich von einer Generation zur nächsten nichts ändert. Begründen Sie: dabei handelt es sich um einen Eigenvektor von A. Berechnen Sie den stabilen Zustand.

Die Berechnung des stabilen Zustandes lässt sich als Vektoriteration (Aufgabe 59) durchführen oder mit MATLABs `eig` Befehl.

Hinweis: Die Prozentangaben in der Grafik sind gerundet, deswegen summieren sich nicht alle Spalten auf exakt 100%, woraus sich kleine Ungenauigkeiten ergeben.

Aufgabe 62: Schwingungsgleichung

Die Eigenschwingungsformen und zugehörigen Frequenzen einer schwingenden Saite lassen sich näherungsweise aus dem Eigenwertproblem

$$A\mathbf{x} = \lambda\mathbf{x}$$

bestimmen. Man denkt sich die Saite dazu in n Einzelmassen, wie Perlen auf einer Schnur, zerlegt. Für grosse n nähert sich die Perlenschnur einer kontinuierlichen Massenverteilung an. Die $n \times n$ Matrix A hat eine einfache Tridiagonal-Form.

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix}.$$

Ist die Saite mit Länge ℓ und Gesamtmasse m mit Kraft τ gespannt, dann ist die zum Eigenwert λ dieser Matrix zugehörige Eigenschwingungsfrequenz ν (in Hz) näherungsweise bestimmt durch:

$$\nu = \frac{n+1}{2\pi}\sqrt{\lambda\frac{\tau}{m\ell}}$$

Dem kleinsten Eigenwert von A entsprichtdie Frequenz der Grundschwingung, die weiteren Eigenwerte entsprechen den Obertönen. In der Praxis sind Grundschwingung und einige

Oberschwingungen niedriger Ordnung relevant. Je größer n, desto genauer sind die aus der „Perlenschnur-Näherung“ berechneten Frequenzen.

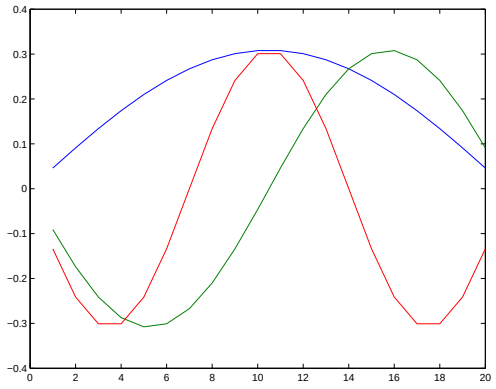
In diesem einfachen Fall lassen sich Eigenfrequenzen und Schwingungsformen auch durch exakte Formeln angeben. Aber schon geringfügig allgemeinere Aufgabenstellungen, zum Beispiel ungleichförmige Massenverteilung entlang der Saite, sind nicht analytisch lösbar. Die Analyse des Schwingungsverhaltens von Bauteilen ist eine typische Anwendung für numerische Simulation.

Siehe dazu auch die Vorlesungsfolie 10 der 8. Vorlesung, sie zeigt ein Perlenschnur-Modell einer frei pendelnden Kette.

Ihre Aufgabe: Für die E-Saite einer Konzertgitarre gilt: freier Länge $\ell = 65\,\mathrm{cm}$, Masse $m = 3{,}58\,\mathrm{g}$, Spannkraft $\tau = 63{,}11\,\mathrm{N}$. Mit diesen Daten lautet die obige Frequenzgleichung

$$\nu = 164{,}7\frac{n+1}{2\pi}\sqrt{\lambda}$$

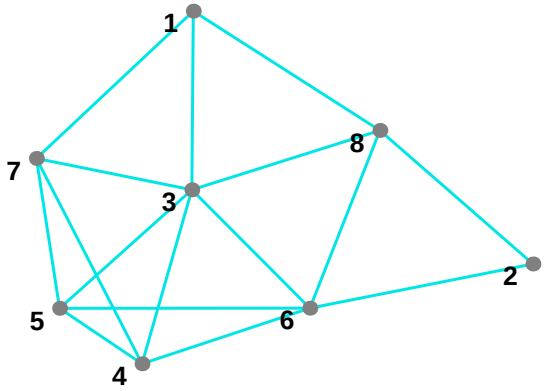
Erstellen Sie für $n = 5, 10, 20$ die Matrix A und berechnen Sie die Frequenzen der Grundschwingung. Sie sollten erkennen: die berechneten Näherungen konvergieren rasch. (Für eine Genauigkeit von zwei Nachkommastellen müssten Sie aber $n = 80$ wählen!)



Zeichnen Sie für $n = 20$ die Eigenvektoren zu den niedrigsten drei Eigenwerten; sie geben die entsprechenden Schwingungsformen der Saite (Grundschwingung, erste und zweite Oberschwingung) an. Siehe Abbildung!

Aufgabe 63: Erreichbarkeit in einem Netzwerk

Ein Netzwerk (Verkehrsverbindungen, verlinkte Seiten im Internet, soziales Netz. . . , mathematisch: ein Graph) lässt sich durch seine Adjazenzmatrix beschreiben.



$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ \vdots & & & & & & & \vdots \end{bmatrix}$$

(a) Stellen Sie zum hier gezeigten Netzwerk die (01)-Adjazenzmatrix A auf (siehe Vorlesungsfolie 8, 8. Vorlesung 2023) und berechnen Sie einen Eigenvektor zum größten Eigenwert. Welcher Knoten im Netz ist (jedenfalls laut Eigenvektor-Daten) am besten, welcher am schlechtesten vernetzt?

(b) Gegeben ist ein Vektor $\mathbf{x}^{(0)}$ aus lauter Einsen. Welches Maß für „Vernetzung“ oder „Erreichbarkeit ist durch $\mathbf{x}^{(1)} = A\mathbf{x}^{(0)}$ definiert?

(c) Führen Sie, beginnend mit dem Vektor $\mathbf{x}^{(0)}$ aus (b), zehn Schritte der Vektoriteration durch. Was erhalten Sie dadurch näherungsweise?

Die Idee, Links zwischen Internet-Seiten auf diese Art mathematisch zu modellieren, untersuchte 1997 ein Doktorand (ein gewisser Larry Page) an der Stanford University. Seine Dissertation hat er zwar bis heute nicht abgeschlossen, aber wenn es Sie interessiert, googeln Sie, was aus ihm und seinem Forschungsprojekt geworden ist…

Die Folien zur 8. Vorlesung zeigen ein ähnliches Netzwerk. Für ein praktisch relevantes Beispiel sehen Sie in Wikipedia zum Stichwort Page Rank nach. Im Material zur 7. Übung finden Sie die Datei `pagerank.m`, sie vollzieht die Rechnungen im Wikipedia-Artikel nach.

Ü 7.2 Gewöhnliche Differentialgleichungen erster Ordnung

Aufgabenstellung:
Explizite gewöhnliche Differentialgleichung 1. Ordnung mit Anfangsbedingung
Gegeben ist eine Funktion $f(x, y)$. Gesucht ist eine Funktion $y(x)$. Sie soll erfüllen

$$y'(x) = f(x, y(x)) \qquad \text{Differentialgleichung}$$
$$y(x_0) = y_0 \qquad \text{Anfangsbedingung}$$

Die Folien der 9. Vorlesung (klick!) zeigen die geometrische Interpretation dieser Aufgabe als Richtungsfeld im $\mathbb{R}^2$ mit Lösungsfunktionen, die dem Richtungsfeld folgen.

In den aktuellen Übungsunterlagen finden Sie die MATLAB-Dateien `BeispieleGDG.m` und `GDGDemo.m`. Laden Sie sich diese Skript-Dateien herunter. Übersichtlich formatiert und gut lesbar werden diese Dateien, wenn Sie MATLABs Publish-Funktion nützen oder die Dateien als Live-Scripts öffnen!

Die Datei `BeispieleGDG.m` zeigt zuerst auch die symbolische Lösung von Differentialgleichungen. Wir konzentrieren uns auf die numerische Lösung, denn brauchbare symbolische Lösungen existieren nur in einfachen Fällen.

Ü 7.2.1 MATLAB-Löser für gewöhnliche Differentialgleichungen

MATLAB bietet fertige Löser für Anfangswertprobleme an.

Es sind sind sieben Solver verfügbar: `ode23`, `ode23s`, `ode23t`, `ode23tb`, `ode45`, `ode113`, `ode15s`. Eine genauere Beschreibung können Sie in der MALAB-Hilfe nachlesen.

Die Funktion `ode45` ist das Standardverfahren, welches man für ein „neues“ Anfangswertproblem zuerst probieren wird. Sie ist die Implementierung eines expliziten Einschritt-Runge-Kutta-Verfahrens (Fehlerordnung 5 und Kontrollrechnung mit Fehlerordnung 4).

Es folgt ein kleines Beispiel, das die Verwendung von `ode45` demonstriert. Angenommen, Sie wollen für $y = y(x)$ das Anfangswertproblem

$$y' = g(x, y) = 2x(1 + y^2), \quad y(0) = 0,$$

im Intervall $[0, 1]$ mit diesem Solver lösen. Sie speichern dafür die Funktion g in einer Datei `g.m` ab

```
function ydot=g(x,y)
ydot=2*x*(1+y^2);
end
```

und fassen dann die Lösung des Anfangswertproblems in einer Scriptdatei (`test.m`) zusammen:

```
y0=0.0;
xspan=[0, 1];
ode45(@g,xspan,y0)
```

So einfach ist das! Sie erhalten eine graphische Darstellung der Lösung. (Noch kürzer ist die Implementierung in der Beispieldatei `BeispieleGDG.m`!)

Sie erhalten Vektoren von x- und y-Werten statt der grafischen Ausgabe, wenn Sie folgende Syntax verwenden:

```
[X, Y] = ode45(@g,xspan,y0);
```

MATLAB wählt sich aber die Lage der x-Werte selbst aus (um die Fehler unterhalb einer bestimmten Schranke zu halten). Wenn Sie die Lösung numerisch für bestimmte, von Ihnen gewählte x-Werte berechnen wollen, schreiben Sie beispielsweise

```
sol=ode45(@g,xspan,y0);
x = [0 0.25 0.5 0.75 1];
y = deval(sol,x)
```

Aufgabe 64:

Berechnen Sie mit MATLAB numerisch für die nachfolgend gegebenen Funktionen $y = y(x)$ die Lösung des Anfangswert-Problems. Stellen Sie die Lösung graphisch dar.

a) $y' = y$ mit $y(0) = 1$, Lösung gesucht im Bereich $0 \le x \le 2$.

Das ist die klassische Differentialgleichung der Exponentialfunktion, exakte Lösung ist $y = e^x$. Stellen Sie auch (in einem separaten Diagramm) den Fehler zwischen numerischer und exakter Lösung dar.

b) $y' = xy/4 - 1$ mit drei verschiedenen Anfangsbedingungen: $y(0) = 2;\ 2,5;\ 3$, Lösungen gesucht im Bereich $0 \le x \le 4$.

Vergleichen Sie: im Skriptum sind Lösungen dieser Gleichung im Richtungsfeld dargestellt.

c) $y' = \dfrac{1}{x^2}$ mit $y(-1) = 1$, Lösungen gesucht im Bereich $-1 \le x \le 1$. Prüfen Sie nach: Die Funktion $y = -\dfrac{1}{x}$ erfüllt in ihrem Definitionsbereich Differentialgleichung und Anfangsbedingung. Bei der numerischen Lösung müssen Sie allerdings auf Probleme gefasst sein. Die Funktion y' hat hier eine spezielle Eigenschaft, die eine Lösung im gesamten Intervall $-1 \le x \le 1$ nicht zulässt – welche Eigenschaft ist das? Vergleichen Sie mit der numerischen Lösung.

Ü 7.2.2 Explizite Einschrittverfahren

Bei der numerischen Lösung einer DG bestimmt man ausgehend von den Anfangsbedingungen eine Folge von Wertepaaren $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots$, die den Verlauf der gesuchten Funktion $y = y(x)$ annähern sollen. Schema:

Wähle Schrittweite h und maximale Schrittzahl N;
setze x_0 und y_0 laut Anfangsbedingung;
für $i = 0, 1, \ldots, N-1$
$\quad x_{i+1} = x_i + h$;
$\quad y_{i+1} = y_i + hF(x_i, y_i, h)$.

Die Funktion $F(x, y, h)$ heißt die *Verfahrensfunktion* des jeweiligen Verfahrens. Geometrisch interpretiert gibt $F(x, y, h)$ die *Fortschreit-Richtung*. Die verschiedenen Verfahren unterscheiden sich in der Verfahrensfunktion, also in der Definition der Fortschreit-Richtung. Nur beim expliziten Euler-Verfahren ist die Fortschreit-Richtung im Punkt $(x; y)$ auch gleich der Steigung $y'(x, y)$, also

$$F(x, y, h) = f(x, y),$$

Andere Verfahrensfunktionen versuchen, die Fortschreit-Richtung besser an den Verlauf der Lösung anzupassen. Die Folien der 8. Vorlesung stellen verschiedene Einschrittverfahren und deren Verfahrensfunktionen bildlich dar.

Beim modifizierten Euler-Verfahren ist

$$F(x, y, h) = f\left(x + \frac{h}{2}, y + \frac{h}{2} f(x, y)\right),$$

beim Verfahren von Heun

$$F(x, y, h) = \frac{1}{2}(k_1 + k_2)$$

mit

$$\begin{aligned} k_1 &= f(x, y) \\ k_2 &= f(x + h, y + hf(x, y)). \end{aligned}$$

Dazu gibt es ein Musterprogramm `GDGdemo.m` zum Herunterladen!

Tipp: Lassen Sie sich von Cleve Moler, einer der MathWorks-Gründer, persönlich erklären, wie das Runge-Kutta-Verfahren das Anwachsen einer Flamme berechnet!

Ü 7.2.3 Handrechnung und einfache Programme fürs Verständnis

„Ich habe ein numerisches Verfahren erst dann verstanden, wenn ich mich selbst dabei verrechnet habe“

Die numerische Lösung einer Differentialgleichung durch händische Rechnung dient heutzutage nur mehr der Illustration der Rechenverfahren. Die praktische Durchführung überlassen Sie dem Computer. Ihr Verständnis für die Rechenverfahren müssen Sie aber zumindest bei der Vorlesungsprüfung dadurch demonstrieren, dass Sie einige Rechenschritte selbst auf dem Papier ausarbeiten.

Die Folien der 9. Vorlesung (klick!) zeigen sehr ausführlich die einzelnen Rechenschritte der Verfahren, die sie hier durchrechnen sollen. Das Musterprogramm `GDGdemo.m` implementiert die Rechenschritte in MATLAB.

Ein Beispiel sei das Anfangswertproblem für die Funktion $y(x)$

$$\begin{aligned} y' &= 4xy + 3 \\ y(0) &= 0 \end{aligned}$$

In diesem Fall ist also $f(x, y) = 4xy + 3$ und $x_0 = y_0 = 0$. Wir wollen mit Schrittweite $h = 0{,}2$ den Wert der Funktion $y(x)$ an den Stellen $x_1 = 0{,}2$; $x_2 = 0{,}4$; $x_3 = 0{,}6$ näherungsweise bestimmen. Die exakte Lösung ist bekannt, aber nicht durch elementare Funktionen gegeben; auf sechs Nachkommastellen genau beträgt $y(0{,}6) = 2{,}973\,414$.

Ein tabellarisches Rechenschema ist hilfreich. Beachten Sie: Die vorletzte Spalte berechnet immer die Verfahrensfunktion, die letzte Spalte entspricht dem allgemeinen Schritt $y_{i+1} = y_i + hF(x_i, y_i, h)$, in einfachen Worten: „neuer y-Wert = alter Wert plus h mal Fortschreit-Richtung F“

Für das explizite Euler-Verfahren:

i	x	y	$F = f(x,y) = 4xy + 3$	$y + hF$
0				
1				
2				

Modifiziertes Euler-Verfahren:

i	x	y	$k_1 = f(x,y)$	$y + \frac{h}{2}k_1$	$F = f(x + \frac{h}{2}, y + \frac{h}{2}k_1)$	$y + hF$
0						
1						
2						

Verfahren von Heun:

x	y	$k_1 = f(x,y)$	$y + hk_1$	$k_2 = f(x + h, y + hk_1)$	$F = (k_1 + k_2)/2$	$y + hF$

Aufgabe 65: Einfache Einschrittverfahren programmieren, Fehlerordnung erkennen

Verwenden Sie das Musterprogramm `GDGdemo.m` und berechnen Sie für das oben gegebene Beispiel den Wert $y(0{,}6)$ mit verschiedenen Schrittweiten und verschiedenen Verfahren:

1. Rechnen Sie mit $h = 0{,}2$ und den drei obigen Verfahren. (Vergleichen Sie mit den Werten aus der Handrechnung.) Implementieren Sie auch das dreistufige Verfahren

$$\begin{aligned} k_1 &= f(x,y) \\ k_2 &= f(x + \frac{h}{2}, y + \frac{h}{2}k_1) \\ k_3 &= f(x + h, y - hk_1 + 2hk_2) \\ F &= \frac{1}{6}(k_1 + 4k_2 + k_3) \end{aligned}$$

2. Rechnen Sie nun mit feinerer Schrittweite $h = 0{,}05$. Wie groß ist jeweils der Fehler (Differenz zwischen Näherungswert und exakter Lösung $y(0,6) = 2,973414$?

3. Halbieren Sie die Schrittweite, rechnen Sie also mit $h = 0{,}025$ und vergleichen Sie die Fehler. Um welchen Faktor hat sich jeweils der Fehler reduziert?

Die verscheidenen MATLAB-Löser für Anfangswertprobleme arbeiten im Prinzip wie das Demo-Programm `GDGdemo.m`, aber

- die Verfahrensfunktion bestimmt die Fortschreit-Richtung aus mehreren, optimal gewählten Zwischenpunkten,
- der (globale) Fehler wird abgeschätzt,
- die Schrittweite wird automatisch angepasst.
- für *steife Probleme* gibt es Methoden (`ode23s, ode15s`) mit besserem Stabilitätsverhalten.