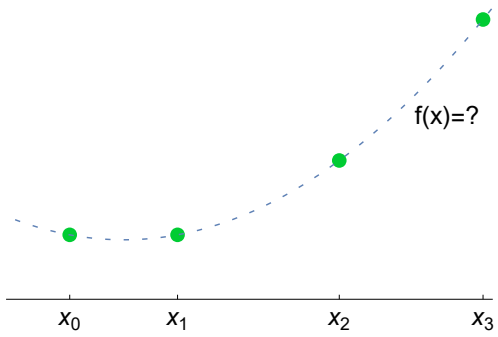
7 Interpolation

Interpolation, Kurzfassung
Gegeben: Datenpunkte
Gesucht:

- Eine Funktion, die *durch* die gegebenen Datenpunkte verläuft.
- Ein Wert *zwischen* den Datenpunkten
- Trend über den gegebenen Datenbereich hinaus: *Extrapolation*

Anwendungen: Zwischenwerte in Tabellen, „glatte“ Kurven für Graphik...



Sind die Werte einer Funktion f für einen Satz von x-Werten gegeben, also $f(x_i) = y_i$ für $i = 0, 1 \ldots, n$, aber kein analytischer (formelmäßiger) Ausdruck für f, so kann man ersatzweise eine geeignete, möglichst einfache Funktion suchen, die für alle gegebenen x_i-Werte – man nennt sie die *Stützstellen* – die entsprechenden Funktionswerte y_i annimmt.

Mit Hilfe dieser Funktion lassen sich dann leicht Werte an beliebigen anderen Stellen x berechnen. Sucht man einen Funktionswert für ein x zwischen x_0 und x_n, so spricht man von *Interpolation*; liegt x außerhalb des gegebenen Datenbereiches, heißt die Aufgabe *Extrapolation* und ist beträchtlich riskanter (was viele Pandemie-Prognose-Gurus inzwischen bestätigen können).

Beachten Sie den Unterschied zu den Approximationsaufgaben des vorigen Abschnittes: *Interpolation* legt eine Funktion *durch* gegebene Datenpunkte; *Approximation* findet eine Funktion, die sich an die Datenpunkte möglichst gut *annähert*.

Jede Formelsammlung bringt eine Vielzahl von Interpolationsformeln, verbunden mit klingenden Namen von Leuten, die schon lange tot sind (Newton, Bessel, Lagrange...). Nehmen Sie das als Indiz dafür, dass Interpolationsverfahren früher von großer Bedeutung waren. Zu einer Zeit, als mathematische Funktionen aus Tabellenwerken entnommen werden mussten, waren etwa trigonometrische Berechnungen ständig mit Interpolation in Tabellen verbunden. Winkelfunktionen bietet aber inzwischen jeder bessere Taschenrechner, und auch weniger elementare Funktionen stehen in gängigen Computerprogrammen zur Verfügung. Damit ist die Wichtigkeit der Interpolation in diesem Bereich ziemlich gesunken.
Interpolation und Extrapolation sind noch wichtig als Hilfsmittel zur Lösung anderer mathematischer Probleme (numerische Integration, numerisches Lösen von Differentialgleichungen). Sehr wichtig ist Interpolation bei Computergraphik und computerunterstütztem Design: Lara Croft existiert als ein Satz von Datenpunkten; damit sie am Bildschirm gut zur Geltung kommt, zeichnet der Rechner möglichst attraktive Kurven durch die Daten. Meist handelt es sich dabei um Spline-Interpolationsverfahren.

7.1 Polynomiale Interpolation

Durch zwei Punkte geht genau eine Gerade. Durch drei beliebige Punkte lässt sich eindeutig eine Parabel legen. Allgemein: Durch $n+1$ (verschiedene) Punkte ist ein Polynom n-ten Grades eindeutig bestimmt.

Polynom-Interpolation, Aufgabenstellung:
Gegeben: $n+1$ Wertepaare $(x_i, y_i), i = 0, \ldots, n$, wobei die x_i paarweise verschieden sind.
Gesucht: das eindeutig bestimmte Polynom n-ten Grades p, das durch die gegebenen Datenpunkte verläuft:

$$p(x_i) = y_i \quad \text{für } i = 0, \ldots, n$$

Oder: gesucht ist nicht die Form des interpolierenden Polynoms selbst, sondern dessen Wert an einer gegebenen Stelle x.

Lösung mit der Holzhammermethode: Ansatz des Polynoms mit unbestimmten Koeffizienten in der Standardform,

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \ .$$

Einsetzes der gegebenen Wertepaare führt auf ein System von $n+1$ linearen Gleichungen in den $n+1$ unbekannten Koeffizienten $a_0, a_1, \ldots, a_n$.

In Matrix-Vektor-Form haben die Gleichungen $p(x_i) = y_i$ eine einfache Struktur:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \ldots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Eine solche Matrix, in deren Spalten der Reihe nach die x_i hoch $0, 1, 2 \ldots$ stehen, heißt *Vandermonde-Matrix*.

Die gute Nachricht: Falls alle x_i-Werte verschieden sind, ist diese Matrix nicht singulär, das Gleichungssystem eindeutig lösbar.

Die schlechten Nachrichten: Die Vandermonde-Matrix kann sehr hohe Konditionszahl haben. Bei Polynomen höheren Grades entstehen gravierende Rundungsfehler. Der Rechenaufwand zur Lösung des Gleichungssystems steigt mit $O(n^3)$. Alle Numerik-Lehrbücher raten alternativ zu effizienteren Verfahren. Aber für kleine n und einen schnellen Computer sind diese Argumente nicht relevant, sogar MATLAB's Befehl `polyfit` verwendet diesen Ansatz.

Dabei lässt sich das Interpolationspolynom ganz einfach und elegant explizit hinzuschreiben. Die Formel kennen Sie aus den Mathematik-Einführungskursen:

Lagrangesche Interpolationsformel
Das Interpolationspolynom durch die $n+1$ Wertepaare (x_i, y_i), $i = 0, \ldots, n$ ist gegeben durch

$$p(x) = L_0(x)y_0 + L_1(x)y_1 + \ldots + L_n(x)y_n,$$

wobei

$$L_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

Diese Darstellung ist für eine Reihe weiterführender Überlegungen wichtig, aber zum Programmieren oder zur numerischen Auswertung nicht die günstigste.

Neben dem Ansatz mit der Vandermonde-Matrix und der Lagrange-Formel lernen Sie gleich noch den Ansatz nach Newton kennen.

Welchen Rechenweg sollen Sie wählen? *Alle Wege führen zum Polynom*, in diesem Fall zu *demselben* Interpolationspolynom, bloß in unterschiedlicher Schreibweise. Warum sollen Sie dann noch weitere Rechenwege kennenlernen? *Der Weg ist das Ziel*: neben der praktischen Anwendung stoßen Sie auf einen mathematisch tiefgründigen Zusammenhang: Polynome verhalten sich in vielfacher Weise wie Vektoren; für Mathematiker *sind* sie Vektoren (Elemente eines Vektorraums). Das würde hier zu weit führen, aber wir halten fest:

Polynome in verschiedenen Basis-Darstellungen Ein Polynom $p(x)$ lässt sich auf unterschiedliche Art als Summe von Termen der Form *Koeffizient mal Basisfunktion* anschreiben.

- Standardbasis

  $$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

  p ist Linearkombination der Basis-Polynome $1,\ x,\ x^2, \ldots, x^n$

- Lagrange-Basis

  $$p(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x)$$

  p ist Linearkombination der Lagrange-Polynome $L_0, L_1, \ldots, L_n$

- Newton-Basis

  $$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1})$$

  p ist Linearkombination der Basis-Polynome $1,\ (x - x_0),$
  $(x - x_0)(x - x_1),\ \ldots,\ (x - x_0)(x - x_1)\cdots(x - x_{n-1})$

Newtons Interpolations-Algorithmus ist eine schlauere Variante der anfangs vorgestellten Holzhammer-Methode. Newton macht ebenfalls auf einem Ansatz mit unbestimmten Koeffizienten, aber mit anderen Basispolynomen:

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_n)$$

Gesucht sind die Koeffizienten $c_0, \ldots, c_n$. Der Ansatz sieht auf den ersten Blick komplizierter aus als die Standardform, aber das resultierende Gleichungssystem wird einfacher: Die Gleichungen $p(x_i) = y_i$ ergeben nun ein Gleichungssystem mit unterer Dreiecksmatrix.

$$\begin{bmatrix} 1 & & & & 0 \\ 1 & (x_1 - x_0) & & & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & & \ddots & \\ 1 & (x_n - x_0) & \cdots & & \prod_{i=0}^{n-1}(x_n - x_i) \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{17}$$

Die Koeffizienten $c_0, \ldots, c_n$ lassen daraus der Reihe nach von oben nach unten berechnen:

$$c_0 = y_0, \quad c_1 = \frac{y_1 - y_0}{x_1 - x_0}, \ldots$$

Beim Weiterrechnen würden Sie bemerken: es treten ständig Differenzen von y-Werten, dividiert durch x-Wert-Differenzen auf. Das lässt sich als effizientes Rechenschema organisieren: das Verfahren der *dividierten Differenzen*. Darum geht es im nächsten Abschnitt.

7.2 Das Newtonsche Interpolationspolynom

Newtons Interpolations-Algorithmus, Kurzfassung:

- Ansatz mit *Newton-Basisfunktionen*
- Das *Schema der dividierten Differenzen* berechnet mit wenig Aufwand die Koeffizienten
- Auswertung des Polynoms effizient mit *Horner-Schema*.

Häufig sind Werte einer Funktion tabellarisch gegeben, wie im folgenden Beispiel:

Spezifische Wärmekapazität von kohlenstoffarmem Stahl in J/kg K für Temperaturen zwischen 0 und 600C

T	cp
0	460.8
100	471.1
200	496.4
300	537.0
400	593.3
500	666.8
600	760.8

Gesucht sind Interpolationspolynome, mit denen sich Zwischenwerte berechnen lassen, zum Beispiel ein kubisches Polynom für den Bereich $0 < T < 300$. (Dieses Beispiel wird dann später durchgerechnet. Es ist möglich, aber nicht unbedingt klug, ein einziges Interpolationspolynom für den gesamten Bereich $0 < T < 600$ aufzustellen. Aus praktischer Sicht ist vielleicht lineare Interpolation zwischen benachbarten Datenpunkten völlig ausreichend. Das hängt von der Qualität der Daten ab.)

Das Schema der dividierten Differenzen ist ein optimierter Rechenablauf zur Lösung des Gleichungssystems (17). Mit Papier und Stift organisiert man die Rechnung am besten in Tabellenform nach folgendem Schema

$$
\begin{array}{cccccccc}
 & & & x_0 & y_0 & & & \\
 & & x_1 - x_0 & & & [x_0, x_1] & & \\
 & x_2 - x_0 & & x_1 & y_1 & & [x_0, x_1, x_2] & \\
 & & x_2 - x_1 & & & [x_1, x_2] & & \\
\ldots & x_3 - x_1 & & x_2 & y_2 & & [x_1, x_2, x_3] & \ldots \\
 & & x_3 - x_2 & & & [x_2, x_3] & & \\
 & x_4 - x_2 & & x_3 & y_3 & & [x_2, x_3, x_4] & \\
 & & x_4 - x_3 & & & [x_3, x_4] & & \\
 & & & x_4 & y_4 & & &
\end{array}
$$

Darin bezeichnet das Symbol in eckigen Klammern, $[x_0, x_1]$ zwischen x_0 und x_1, eine dividierte Differenz, definiert durch

$$[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}$$

Das ist die Steigung der Geraden durch die beiden Datenpunkte.

Damit lässt sich auch schon die Formel für lineare Interpolation (Interpolationspolynom ersten Grades) angeben:

$$p(x) = y_0 + (x - x_0)[x_0, x_1]$$

Die höheren dividierten Differenzen erklärt man durch die niedrigeren. Eine zweite Differenz $[x_0, x_1, x_2]$ wird erklärt durch

$$[x_0, x_1, x_2] = \frac{[x_1, x_2] - [x_0, x_1]}{x_2 - x_0}$$

und allgemein ist für $k > i$

$$[x_i, x_{i+1}, ..., x_{k+1}] = \frac{[x_{i+1}, ..., x_{k+1}] - [x_i, ..., x_k]}{x_{k+1} - x_i}$$

Dann hat $p(x)$ folgende, durch Induktion leicht beweisbare Darstellung,

$$\begin{array}{rcl} p(x) = y_0 & + & (x - x_0)[x_0, x_1] + (x - x_0)(x - x_1)[x_0, x_1, x_2] + \\ & ... & +(x - x_0)(x - x_1)...(x - x_{n-1})[x_0, x_1, ..., x_n] \end{array}$$

Die Formeln sehen kompliziert aus, aber der eigentliche Rechengang ist sehr einfach: Man berechnet zuerst die Einträge in der linken Hälfte der Tabelle. In der rechten Tabellenhälfte erhält man die dividierten Differenzen nach der Regel „linker unterer minus linker oberer Nachbar, dividiert durch den symmetrisch gelegenen Eintrag aus der linken Tabellenhälfte“.

Die Koeffizienten des Newtonschen Interpolationspolynoms liegen in diesem Schema rechts in der oberen Schrägzeile (von der Mitte oben nach rechts schräg abwärts).

Das durchgerechnete Tableau für vier Datenpunkte aus dem obigen Beispiel sieht so aus:

```
          0    460,8
      100              0,103
  200     100  471,1           0,000750
300   100              0,253              0,000000050
  200     200  496,4           0,000765
      100              0,406
          300  537,0
```

Das Newton-Interpolationspolynom hat in diesem Fall die Form

$$p(x) = 460{,}8 + x \cdot 0{,}103 + x(x - 100) \cdot 0{,}000\,750 + x(x - 100)(x - 200) \cdot 0{,}000\,000\,050$$

Es ist nicht ratsam, diese Formel zu „vereinfachen“, indem man Produkte wie $x(x - 100)(x - 200)$ ausmultipliziert und gleiche Potenzen in x zusammenfasst.

Man setzt x-Werte in die obige Formel direkt ein oder schreibt sie noch rechengünstiger (Horner-Schema) in der Form

$$p(x) = 460{,}8 + x(0{,}103 + (x - 100)(0{,}000\,750 + (x - 200) \cdot 0{,}000\,000\,050))$$

Umformen auf $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ kann ganz drastische Rundungsfehler hervorrufen!

Bei äquidistanten x-Werten stehen in der linken Tabellenseite spaltenweise gleiche Werte, und der Rechengang könnte vereinfacht werden. Es gibt (z.B. im Bronstein) speziell optimierte Formeln für Interpolation mit äquidistanten x-Werten; hier beschränken wir uns auf den allgemeinen Fall.

Die x-Werte in dieser Rechentabelle brauchen auch nicht der Größe nach geordnet zu sein. Außerdem lässt sich die Tabelle leicht nachträglich ergänzen. Der Rest der Tabelle muss nicht neu berechnet werden, und auch im Interpolationspolynom kommt nur ein Term dazu, die anderen bleiben gleich. Das ist ein Vorteil des Newtonschen (und auch des Nevilleschen) Interpolationsverfahrens gegenüber dem Lagrangeschen Interpolationspolynom, bei dem sich alle Terme ändern, wenn ein Datenpunkt hinzugefügt oder geändert wird.

Will man zum Beispiel die gegebenen Daten im Bereich $200 \le x \le 600$ zuerst einmal nur linear approximieren, stellt man das Schema

```
      200   496,4
400                0,661
      600   760,8
```

auf und erhält

$$p(x) = 496{,}4 + (x - 200) \cdot 0{,}661 \ .$$

Will man doch quadratisch interpolieren, fügt man ein Wertepaar an, ergänzt die Tabelle

```
            200   496,4
       400                0,661
100         600   760,8          0,00085
     -300                0,746
            300   537,0
```

und hängt dem Interpolationspolynom einen Term an:

$$p(x) = 496{,}4 + (x - 200) \cdot 0{,}661 + (x - 200)(x - 600) \cdot 0{,}000\,85 \ .$$

Bei Polynomdarstellung in der Standard-Form ändern sich hingegen alle Koeffizienten, wenn Sie von linearer zu quadratischer Interpolation erweitern: Lineares und quadratisches Polynome lauten hier jeweils

$$\begin{aligned} p(x) &= 364{,}2 + 0{,}661 \cdot x \\ p(x) &= 466{,}2 - 0{,}019 \cdot x + 0{,}000\,85 \cdot x^2 \end{aligned}$$

Das Kapitel 7.3 illustriert an einem ähnlichen Beispiel das Verfahren von Neville, das sich gut eignet, wenn ein einziger Zwischenwert interpoliert werden soll und als Hilfsmittel nur Papier, Bleistift und ein billiger Taschenrechner zur Verfügung stehen.

Wichtige Zusatzinformationen in den Übungsunterlagen

Die Unterlagen zur 6. Übung erläutern Vandermonde-Ansatz und das Netwtonsche Interpolationsverfahren ausführlich. Auch der Ansatz mit Orthogonalpolynomen wird dort vorgestellt. Lesen Sie dort nach!

7.3 Das Verfahren von Neville

Das Verfahren von Neville berechnet den Wert des Interpolationspolynoms an einer gegebenen Stelle x. (Zum Verwechseln ähnlich ist das auch manchmal angegebene Aitken-Verfahren.)

Man setzt für $k = 0, 1, ..., n$

$$P_i(x) = y_i$$

Dann ist P_i der Wert des eindeutig bestimmten Polynoms vom Grad 0, das durch (x_i, y_i) geht.

Nun sei P_{01} der Wert (an der gegebenen Stelle x) des eindeutig bestimmten Polynoms ersten Grades (der Geraden), das durch die Punkte (x_0, y_0) und (x_1, y_1) geht. Und dementsprechend: $P_{12}, P_{23}, \ldots$. In gleicher Weise sei P_{012} der Wert der Parabel (an der gegebenen Stelle x) durch die Punkte (x_0, y_0), (x_1, y_1) und (x_2, y_2). Dieses Schema für Polynome höherer Ordnung weiterführend gelangt man zu $P_{0,1,2\ldots n}$, dem Wert des Polynoms durch alle gegebenen Punkte und damit der gesuchten Antwort.

Der Trick an der Sache ist, dass sich die einzelnen P-Werte leicht aus jeweils zwei vorherigen berechnen lassen. Am einfachsten schreibt sich die Formel mit der Determinante einer 2×2-Matrix,

$$P_{i(i+1)\ldots(i+m)} = \frac{1}{x_{i+m} - x_i} \begin{vmatrix} x - x_i & P_{i(i+1)\ldots(i+m-1)} \\ x - x_{i+m} & P_{(i+1)(i+2)\ldots(i+m)} \end{vmatrix}$$

Durch Induktion zeigt man leicht, dass die so definierten P genau die behaupteten Eigenschaften haben. Zur tatsächlichen Berechnung bedient man sich dann eines Schemas (Tableaus) der Form

$$\begin{array}{lllll} x_0: & y_0 = P_0 & & & \\ & & P_{01} & & \\ x_1: & y_1 = P_1 & & P_{012} & \\ & & P_{12} & & P_{0123} \\ x_2: & y_2 = P_2 & & P_{123} & \\ & & P_{23} & & \\ x_3: & y_3 = P_3 & & & \end{array}$$

Man füllt dieses Tableau spaltenweise von links nach rechts aus. Zum Beispiel ergibt sich der Wert P_{123} aus den beiden linken Nachbarn als

$$P_{123} = \frac{1}{x_3 - x_1} \begin{vmatrix} x - x_1 & P_{12} \\ x - x_3 & P_{23} \end{vmatrix} = \frac{(x - x_1)P_{23} - (x - x_3)P_{12}}{x_3 - x_1}$$

Will man die Interpolationsordnung erhöhen, kann man Werte für x_4 und y_4 unten an das Tableau anfügen und die fehlenden Werte $P_{34}, P_{234}, P_{1234}$ und P_{01234} ergänzen, ohne das restliche Tableau neu berechnen zu müssen (Vorteil gegenüber der Lagrangeschen Formel). Übrigens müssen die x_i weder äquidistant noch irgendwie geordnet sein.

Beispiel: Dichte von Wasser

Die Dichte von Wasser in Abhängigkeit von der Temperatur ist in folgender Tabelle für den Bereich von 0 bis 100 Celsius angegeben

T	ρ	T	ρ
0	999.840	10	999.699
1	999.899	20	998.203
2	999.940	30	995.645
3	999.964	40	992.212
4	999.972	50	988.030
5	999.964	60	983.191
6	999.940	70	977.759
7	999.901	80	971.785
8	999.848	90	965.304
9	999.781	100	958.345

Dichtewerte für Temperaturen, die nicht in der Tabelle angeführt sind, findet man durch Interpolation oder Approximation. Welches Verfahren ist angebracht?

- Einzelne Werte werden benötigt: Neville-Verfahren.
- Ein formelmäßiger Zusammenhang $T = T(\rho)$ in einem engen Temperaturintervall wird benötigt: Newtonsches Interpolationspolynom
- Ein formelmäßiger Zusammenhang über den gesamten Bereich ist gesucht: Approximation durch ein Polynom oder eine rationale Funktion.

Neville

Gesucht sei $\rho(24)$. Ein durchgerechnetes Tableau lautet

$$
\begin{array}{lllll}
10: & 999.699 & & & \\
 & & 997.605 & & \\
20: & 998.203 & & 997.307 & \\
 & & 997.180 & & 997.297 \\
30: & 995.645 & & 997.285 & \\
 & & 997.705 & & \\
40: & 992.212 & & &
\end{array}
$$

Die einzelnen Schritte dabei:

$$P_{01} = \frac{(x - x_0)P_1 - (x - x_1)P_0}{x_1 - x_0}, \quad P_{12} = \frac{(x - x_1)P_2 - (x - x_2)P_1}{x_2 - x_1}, \ldots$$

Das sind die Werte, die sich aus linearer Interpolation ergeben. In vielen Fällen wird lineare Interpolation ausreichend genau sein; man hätte sich dann mit dem Wert P_{12} zufriedengeben können. Die nächste Spalte entspricht quadratischer Interpolation gemäß den Formeln

$$P_{012} = \frac{(x - x_0)P_{12} - (x - x_2)P_{01}}{x_2 - x_0}, \quad P_{123} = \frac{(x - x_1)P_{23} - (x - x_3)P_{12}}{x_3 - x_1}.$$

Die daraus gewonnenen Werte unterscheiden sich nur mehr um 2 Einheiten in der Hundertstel-Stelle. Der letzte Wert entsteht aus kubischer Interpolation:

$$P_{0123} = \frac{(x - x_0)P_{123} - (x - x_3)P_{012}}{x_3 - x_0},$$

Rechenprogramme, welche Stoffgrößen benötigen, können entweder umfangreiche Tabellen abspeichern und darin linear interpolieren; alternativ können sie Speicherplatz sparen, Tabellen mit weniger Stützstellen und Interpolation höherer Ordnung verwenden. Sehr oft steckt ein beträchtlicher Teil der Rechenzeit in der Auswertung von Stoffgrößen; eine rechengünstige Implementierung ist dann bedeutsam.

7.4 Fehlerabschätzung der Polynominterpolation

Nehmen wir an das eine Funktion $f : [a, b] \to \mathbb{R}$, $-\infty < a < b < \infty$, an den Stützpunkten $\pi = \{a = x_0 < x_2 < \ldots < x_n = b\}$ gegeben ist und $p_n(x)$ ei Polynom $n-1$ten Grades ist, das eben durch diese Stützpunkten läuft. Die Frage ist jetzt, wie groß ist jetzt der Fehler, oder

$$e(x) = \sup_{x \in [a,b]} |p_n(x) - f(x)|.$$

Der folgende Satz liefert eine Abschätzung des Fehlers, abhängig von der Funktion f und deren Ableitungen.

Fehlerabschätzung für das Interpolationspolynom
Angenommen f ist $(n+1)$– mal stetig differenzierbar und das Polynom p_n geht durch die Punkte $\{(x_i, f(x_i)) : i = 0, \ldots, n\}$. Dann gibt es für jedes $x^* \in (a, b)$ ein Punkt $\xi \in (a, b)$ mit

$$f(x^*) - p_n(x^*) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x^* - x_0)(x^* - x_1) \cdots (x^* - x_{n-1})(x^* - x_n).$$

Beweis des Satzes: Betrachten wir die Funktion

$$\Theta(x) := f(x) - p_n(x) - c(x^*)(x - x_0)(x - x_1) \cdots (x - x_{n-1})(x - x_n)$$

wobei

$$c(x^*) = \frac{f(x^*) - p_n(x^*)}{(x^* - x_0)(x^* - x_1) \cdots (x^* - x_{n-1})(x^* - x_n)}.$$

Die Funktion Θ ist so definiert, dass x^* eine Nullstelle ist, d.h. $\Theta(x^*) = 0$. Da $f(x_k) = p(x_k)$ gilt, hat aber die Funktion Θ noch zusätzlich die Nullstellen $x_0, x_1, \ldots, x_n$, also $n+2$ Nullstellen. Differenziert man Θ (dies ist erlaubt da f nach Voraussetzung $n+1$ mal differenzierbar ist), so findet man nach den Satz von Rolle zwischen den Nullstellen von Θ jeweils Zwischenwerte $\xi_0, \xi_2, \ldots, \xi_n$ mit $\Theta'(\xi_j) = 0$, $j = 1, \ldots, n$. Nennen wir diese Zwischenwerte $\xi_j^{(1)}$. Wieder mit den Satz von Rolle wissen wir, dass es n Zwischenwerte $\xi_0, \xi_2, \ldots, \xi_{n-1}$ mit $\Theta''(\xi_j) = 0$ gibt, $j = 0, \ldots, n-1$. Nennen wir diese Zwischenwerte $\xi_j^{(2)}$. Dies können wir immer weiter machen bis wir nur einen Zwischenwert $\xi = \xi_1^{(n+1)}$ haben sodass gilt

$$0 = \Theta^{(n+1)}(\xi) = f^{(n+1)}(\xi) - c(x^*)(n+1)!.$$

Daraus folgt nach einer einfachen Umformung

$$c(x^*) = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

also die Abschätzung aus den Satz.

Haben wir nun ein equidistantes Gitter und $0 \le a < b \le 1$, so gilt

$$|f(x^*) - p_n(x^*)| \le \sup_{\xi\in[a,b]} \frac{|f^{(n+1)}(\xi)|}{(n+1)!} \frac{(a-b)^n}{n^n} \le \sup_{\xi\in[a,b]} \frac{|f^{(n+1)}(\xi)|}{(n+1)!n^n}.$$

Hier haben wir folgende Abschätzung vorgenomen:

$$(x^* - x_0)(x^* - x_1)\cdots(x^* - x_{n-1})(x^* - x_n) \le \left(\frac{(a-b)}{n}\right)^n.$$

7.5 Interpolation oder Approximation?

Polynome hohen Grades neigen zu Oszillationen und sind bei der Auswertung in Standardform anfällig gegen Rundungsfehler. Ein Interpolationspolynom durch alle Datenpunkte im Beispiel zur Dichte von Wasser aus Abschnitt 7.3 liefert völlig sinnlose Werte.

> Das mit den Matlab-Befehlen `polyfit` und `polyval` berechnete und ausgewertete Interpolationspolynom weicht in diesem Fall selbst an den Original-Datenpunkten (wo es genau stimmen sollte!) deutlich ab: $\rho(0) = 1000.2$ statt 999.8. Zwischen den Datenpunkten liefert es Phantasiewerte im Bereich $\pm 10^8$.
> Numerisch genauere Berechnungen des Interpolationspolynoms würden zwar an den Original-Datenpunkten bessere Werte liefern, dazwischen aber trotzdem Schrott.

Eine formelmäßige Darstellung für den gesamten Datenbereich lässt sich mit Regressionspolynomen finden. Eine quadratische Approximation liefert Genauigkeit $\pm 0,5$,

$$\rho(T) = 1000.35 - 0.0614512\,T - 0.00364033\,T^2$$

ein Regressionspolynom vierten Grades

$$\rho(T) = 999.867 + 0.0545396\,T - 0.00765475\,T^2 + 0.0000434548\,T^3 - 1.38985\,10^{-7}\,T^4$$

weicht von den Datenpunkten maximal um 0.03 ab.

Rationale Funktionen sind oft zur Approximation besser geeignet. Beispielsweise zeigt die Näherung

$$\rho(T) = \frac{999.844 + 9.63049\,T - 0.0244379\,T^2}{1 + 0.00956721\,T - 0.0000163504\,T^2}$$

einen maximalen Fehler von 0.004. Auch solche Näherungen lassen sich nach der Methode der kleinsten Quadrate finden: Ansatz mit unbestimmten Koeffizienten, Einsetzen der Daten, Umformen auf ein überbestimmtes lineares System und Lösen der Normalengleichungen. (Je nach Ansatz können aber auch *nichtlineare* Ausgleichsprobleme entstehen, die viel schwieriger zu behandeln sind.) Die obige Näherung minimiert das Maximum des Fehlers (sogenannte *Minimax-Näherung*). Für die Konstruktion solcher Näherungen sei auf weiterführende Literatur verwiesen (Das Computer-Algebra-System Mathematica bietet z. B. dafür ein Paket an).

7.6 Weitere Interpolationsverfahren

Nicht alle Funktionen lassen sich in vorteilhafter Weise durch Polynome interpolieren. Andere wichtige Verfahren sind

- Rationale Interpolation
- Spline-Interpolation

- Trigonometrische Interpolation
- Interpolation in zwei oder mehr Dimensionen

Rationale Funktionen sind Quotienten zweier Polynome. Bulirsch und Stoer (siehe deren Buch Numerische Mathematik 1) haben einen Algorithmus gefunden, der mit einem Tableau ähnlich wie bei der Aitken-Neville-Interpolation arbeitet. Die meisten Funktionen lassen sich besser durch rationale Funktionen approximieren als durch Polynome. Auch Ihr Taschenrechner wertet mathematische Funktionen wie Cosinus und Sinus höchstwahrscheinlich in Form rationaler Ausrücke aus.

Spline-Funktionen spielen eine wichtige Rolle überall dort, wo Computer eine „glatte“ Kurve oder Oberfläche durch gegebene Punkte zeichnen sollen.

Trigonometrische Interpolation verwendet Summen von Sinus- und Cosinusfunktionen, oder komplexen Exponentialfunktionen, um eine Kurve durch einen Satz von Datenpunkten zu legen. Eng verwandt und überaus wichtig ist das Verfahren zur schnellen Fourier-Transformation (*Fast Fourier Transform, FFT*).

7.7 Spline-Interpolation

Interpolationspolynome hohen Grades sind rechnerisch aufwendig, oft sehr empfindlich gegenüber geringfügigen Änderungen der Daten (Koordinaten der Stützpunkte) und neigen zum Überschwingen. Will man – etwa für zeichnerische Zwecke – Punkte durch eine „möglichst glatte“ Kurve verbinden, bietet sich die Idee an, einzelne Polynome niedrigeren Grades aneinanderzustückeln. Derartige Funktionen werden allgemein als Splines bezeichnet.

Splines
Unter einer Spline-Funktion versteht man eine stückweise auf Teilintervallen definierte Funktion, deren Teile an den Intervallgrenzen stetig oder sogar ein- bzw. mehrmals stetig differenzierbar aneinanderstoßen.

Kubische (natürliche) Splines
Ein kubischer Spline $s(x)$ durch die $n+1$ Wertepaare (x_i, y_i), $i = 0, \ldots, n$ ist folgendermaßen charakterisiert:

In den einzelnen Intervallen (x_{i-1}, x_i) ist $s(x)$ jeweils ein kubisches Polynom

An den Intervallgrenzen stimmen die Funktionswerte, die ersten und die zweiten Ableitungen rechts- und linksseitig überein.

Zusatzbedingung: (natürlicher Spline) Am linken und rechten Rand ist $s''(x) = 0$.

Die natürlichen kubischen Splines sind unter allen zweimal stetig differenzierbaren, interpolierenden Funktionen $f(x)$ diejenigen mit dem kleinsten

$$\int_{x_0}^{x_n} \left(f''(x)\right)^2 dx$$

Dieses Integral misst aber im gewissen Sinn die Welligkeit oder Gesamtkrümmung der Funktion. Dadurch kann es natürlich kaum zum Überschwingen kommen.

Mit natürlichen kubischen Splines lassen sich aber keine schleifen- oder kreisförmigen Kurven zeichnen. Dazu dienen *parametrische* Splines, die sowohl x- als auch y-Werte einer Kurve als Splinefunktion eines Parameters darstellen. Die Methode zum Kurvenzeichnen in Excel arbeitet mit solchen Splines.

Nicht alle Arten von Splines interpolieren Punkte. *Bézier*-Splines verwenden einen Teil der Punkte, um die Gestalt der Kurve zu beeinflussen. Kurvenzeichnen in MS-Paint funktioniert in dieser Weise.

Fehlerabschätzung:

Nehmen wir an, dass eine Funktion $f : [a, b] \to \mathbb{R}$, $-\infty < a < b < \infty$, an den Stützpunkten $\Pi = \{a = x_0 < x_2 < \ldots < x_n = b\}$ gegeben ist, und $s_n(x)$ sei ein kubisches Spline mit genau diesen Stützpunkten $\{(x_i, f(x_i)) : i = 0, \ldots, n\}$. Die Frage ist jetzt, wie groß ist

$$e(x) = \sup_{x \in [a,b]} |s_n(x) - f(x)|.$$

Angenommen, f ist vier mal stetig differenzierbar, mit $f''(a) = f''(b) = 0$.

- dann existiert genau ein natürlicher kubischer Spline s_n.
- es gelten folgende Fehlerabschätzungen:

$$\sup_{x \in [a,b]} |f(x) - s_n(x)| \leq cM_4h^4, \tag{18}$$

$$\sup_{x \in [a,b]} |f'(x) - s_n'(x)| \leq cM_4h^3, \tag{19}$$

$$\sup_{x \in [a,b]} |f''(x) - s_n''(x)| \leq cM_4h^2, \tag{20}$$

$$\sup_{x \in [a,b]} |f'''(x) - s_n'''(x)| \leq cM_4h, \tag{21}$$

  wobei $h = \max(|x_j - x_{j-1}|)$ und $M_4 = \sup_{x \in [a,b]} |f^{IV}(x)|$.

8 Numerische Integration

Seit dem Altertum beschäftigt sich die Mathematik mit der Berechnung des Inhalts krummlinig berandeter Flächen. Solche Aufgaben erfordern – in heutiger Sprechweise – die Auswertung bestimmter Integrale.

Die *Quadratur des Kreises* ist mit Zirkel und Lineal, den Werkzeugen der antiken Geometer, undurchführbar. Der Begriff „Quadratur“ als Synonym für numerische Flächen- oder Integralberechnug ist geblieben. Numerischen Integration heißt also auch *numerische Quadratur*; die dazu geeigneten Formeln heißen *Quadraturformeln* (engl.: *quadrature formulas*).

Die analytisch integrierbaren Funktionen mit elementaren Stammfunktionen lassen sich auf wenigen Seiten einer Formelsammlung auflisten. Darüber hinaus können Integrale nur näherungsweise numerisch berechnet werden[12].

Auch wenn eine Funktion nicht als formelmäßiger Ausdruck, sondern als Tabelle oder Resultat eines Rechenprogrammes gegeben ist, kommt numerische Integration zum Einsatz.

8.1 Die Integrationsformeln von Newton-Cotes-Typ

Newton-Cotes-Formeln
Gegeben: Eine Funktion $f(x)$ in einem Intervall (a, b) durch ihre Werte f_i an $n + 1$ äquidistanten Stützstellen,

$$f_i = f(a + ih), \quad \text{mit } h = \frac{b-a}{n}, \quad i = 0, \ldots, n.$$

Gesucht: Ein Näherungswert für das Integral $\int_a^b f(x)dx$.
Prinzip: Interpoliere $f(x)$ durch ein Polynom $p(x)$. Nähere das Integral von f durch das Integral von p. Die Näherung ist gegeben als gewichtete Summe der f_i,

$$\int_a^b f(x)dx \approx (b-a) \sum_{i=0}^{n} \alpha_i f_i$$

mit fixen Gewichten α_i

Beispiele: Trapezregel

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \left(f(a) + f(b)\right)$$

Quadratische Interpolation (Lokal: „Keplersche Fassregel“, international: „Simpson-Regel“)

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right)$$

Diese Näherungsformel unter Verwendung von Funktionswerten an drei äquidistanten Stützstellen ist Johannes Kepler eingefallen, als er 1612 in Linz beim Kauf einiger Fässer Wein über deren Rauminhalt nachdachte (wieviel er dabei gekostet hat, ist nicht überliefert).

[12] Allerdings: wenn ein Integral oft genug auftritt, wird es kurzerhand als eigene Funktion *definiert*, tabelliert, und man findet spezielle Näherungsformeln und Reihenentwicklungen zur Auswertung. Beispiele: Das Integral $\int_0^\infty e^{-x^2} dx$ hat keine elementare Stammfunktion, ist aber so wichtig, dass dafür eine spezielle Funktion, die Gaußsche Fehlerfunktion, definiert ist. Auch für die Bogenlänge einer Ellipse oder die Schwingungsdauer eines mathematischen Pendels bei großen Amplituden sind Integrale auszuwerten, für die spezielle Funktionen (elliptische Funktionen) definiert sind.

Die Trapezregel liefert natürlich für lineare Funktionen exakte Werte. Die Keplersche Fassregel ist logischerweise für Parabeln, aber sogar noch für Polynome dritten Grades exakt. Die Fehler lassen sich durch Taylorreihenentwicklung abschätzen und hängen von den höheren Ableitungen der Funktion f ab.

In der Regel benutzt man diese Formeln nicht für das gesamte Intervall $[a, b]$, sondern unterteilt es in eine Reihe kleinerer Intervalle, wendet dort jeweils die Formel an und addiert die Teilergebnisse. Man spricht dann von *zusammengesetzten* Newton-Cotes-Formeln.

Gegeben: Von einer Funktion $f(x)$ in einem Intervall (a, b) die Werte f_i an $n+1$ äquidistanten Stützstellen,

$$f_i = f(a + ih), \quad \text{mit } h = \frac{b-a}{n}, \quad i = 0, \ldots, n.$$

Zusammengesetzte Trapezregel

$$\int_a^b f(x)dx = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n) + E$$

Zusammengesetzte Simpson-Regel

$$\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n) + E$$

Nur für gerades n möglich!

Die Fehlerterme sind

$$E = \frac{a-b}{12}h^2 f''(\xi) \quad \text{für ein } \xi \in [a, b]$$

bei der zusammengesetzten Trapezregel und

$$E = \frac{a-b}{180}h^4 f''''(\xi) \quad \text{für ein } \xi \in [a, b]$$

bei der zusammengesetzten Simpson-Regel

8.2 Romberg-Verfahren, Gauß-Quadraturformeln

Das Romberg-Verfahren wendet die zusammengesetzte Trapezregel mehrfach, mit unterschiedlichen Schrittweiten an. Aus den Ergebnissen extrapoliert es einen noch genaueren Wert. Ausführliche Behandlung in den Übungsunterlagen, 6. Einheit.

Gauß-Quadraturformeln werden heuer (2023) nicht behandelt. Im Unterschied zu Formeln vom Newton-Cotes-Typ, die mit äquidistanten Stützstellen arbeiten, wählen Gauß-Formeln die Stützstellen an optimalen Positionen so, dass möglichst hohe Fehlerordnung erzielt wird.

9 Eigenwerte und Eigenvektoren

9.1 Einführung, Definition, Grundlagen

Eigenwertproblem
Gegeben ist eine $n \times n$-Matrix A. Gesucht sind

- ein vom Nullvektor verschiedener Vektor $\mathbf{x}$ und
- ein Skalar λ (auch $\lambda = 0$ ist erlaubt),

welche die Gleichung

$$A\mathbf{x} = \lambda\mathbf{x} \tag{22}$$

erfüllen.
Ein solches λ heißt *Eigenwert* von A, ein passendes $\mathbf{x}$ heißt *Eigenvektor* von A zum Eigenwert λ.

Die Situation „Matrix mal Eigenvektor ist Null mal Vektor“, also $A\mathbf{x} = 0\mathbf{x}$, kann durchaus auftreten. In so einem Fall ist $\lambda = 0$ ein Eigenwert von A.

Wir verlangen aber ausdrücklich, dass $\mathbf{x}$ nicht der Nullvektor 0 sein darf, weil die Gleichung $A \cdot 0 = \lambda \cdot 0$ für jedes beliebige λ erfüllt wäre – das wäre keine sinnvolle Definition von λ.

Anschauliche Interpretation

Der Hauptberuf einer Matrix ist, Vektoren zu multiplizieren. Wenn sie das macht, hat der Ergebnisvektor gewöhnlich eine andere Länge und zeigt in eine andere Richtung als der Ausgangsvektor. Jeder Matrix hat aber ganz spezielle „eigene“ Vektoren, bei denen sie zwar die Länge ändert, die Richtung aber gleich lässt (falls $\lambda > 0$) oder genau umkehrt (falls $\lambda < 0$). Es kann auch passieren (falls $\lambda = 0$), dass ein Eigenvektor von der Matrix zum Nullvektor gemacht wird. Reguläre Matrizen tun das aber nicht, nur singuläre Matrizen sind so fies.

Beispiel:
Die Matrix $A = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}$ macht aus $\mathbf{u} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$$A\mathbf{u} = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 5 \end{bmatrix},$$

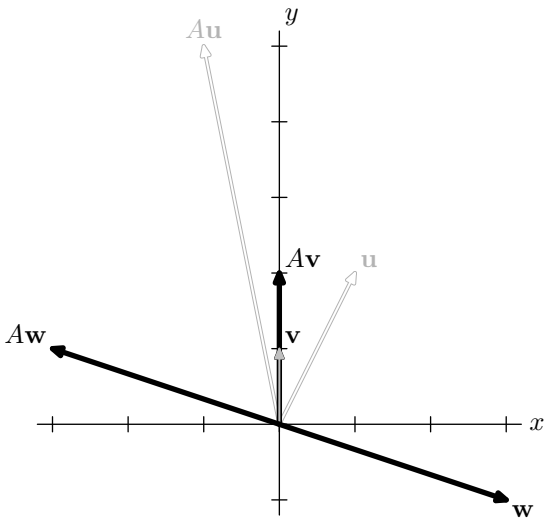
andere Länge und Richtung – kein Eigenvektor.
Der Vektor $\mathbf{v} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ wird hingegen zu

$$A\mathbf{v} = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix},$$

gleiche Richtung, doppelte Länge –
Eigenvektor zum Eigenwert 2.
Der Vektor $\mathbf{w} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ wird zu

$$A\mathbf{w} = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$$

gleiche Länge, umgekehrte Richtung –
Eigenvektor zum Eigenwert -1.



Zu einem Eigenwert gibt es viele Eigenvektoren

Ist $\mathbf{x}$ ein Eigenvektor von A zum Eigenwert λ , so ist auch jedes Vielfache $\mu\mathbf{x}$, $\mu \neq 0$, ein Eigenvektor von A zum selben Eigenwert λ, denn da A, λ und $\mathbf{x}$ die Gleichung (22) erfüllen, gilt auch

$$A(\mu\mathbf{x}) = \mu(A\mathbf{x}) = \mu(\lambda\mathbf{x}) = \lambda(\mu\mathbf{x}).$$

Eigenvektoren sind also nur bis auf einen skalaren Faktor eindeutig bestimmt.

Eine $n \times n$-Matrix hat n Eigenwerte

Durch Umformung von (22) erhalten wir

$$\begin{aligned} A\mathbf{x} &= \lambda\mathbf{x} \\ A\mathbf{x} - \lambda\mathbf{x} &= 0 \\ (A - \lambda I)\mathbf{x} &= 0 \end{aligned} \tag{23}$$

wobei I die Einheitsmatrix ist. Der Vektor $\mathbf{x} = 0$ (der Nullvektor) ist immer eine Lösung von Gleichung (23), aber der interessiert uns nicht. Wir wollen, dass es noch andere, nicht triviale Lösungen gibt. Das ist nur genau dann der Fall, wenn

$$\det(A - \lambda I) = 0. \tag{24}$$

Entwickeln der Determinante liefert ein Polynom n-ten Grades in λ. Dieses Polynom heißt das *charakteristische Polynom* von A. Jedes Polynom n-ten Grades hat genau n reelle oder komplexe Nullstellen (sagt der Fundamentalsatz der Algebra; mehrfache Nullstellen zählt er dabei entsprechend ihrer Vielfachheit). Daraus folgt, dass jede $n \times n$-Matrix genau n (reelle oder komplexe, unter Umständen mehrfach gezählte) Eigenwerte hat.

Die Eigenwerte einer Matrix A lassen sich als Wurzeln (Nullstellen) ihres charakteristischen Polynoms berechnen. Dies ist ein klassische Verfahren, aber im allgemeinen nur für kleine Matrizen ($2 \times 2,\ 3 \times 3$) sinnvoll.

Rechenbeispiel: Wir berechnen die Eigenwerte der Matrix A vom vorigen Beispiel (Seite 76).

$$\begin{aligned} \det(A - \lambda I) &= \det\left(\begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = \det\begin{bmatrix} -1-\lambda & 0 \\ 1 & 2-\lambda \end{bmatrix}, \\ &= (-1-\lambda)\cdot(2-\lambda) - 0 \cdot 1 = \lambda^2 - \lambda - 2 \end{aligned}$$

Die Nullstellen des charakteristischen Polynoms $p(\lambda) = \lambda^2 - \lambda - 2$ und daher die Eigenwerte von A sind $\lambda_1 = -1\,,\ \lambda_2 = 2$.

Noch ein Beispiel: Wir berechnen die Eigenwerte der Matrix $A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$ als Wurzeln des charakteristischen Polynoms.

$$\det(A - \lambda I) = \det\left(\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) = \det \begin{bmatrix} 8-\lambda & 1 & 6 \\ 3 & 5-\lambda & 7 \\ 4 & 9 & 2-\lambda \end{bmatrix}$$

Berechnen der Determinante, z.B. mit der Regel von Sarrus, liefert

$$\begin{aligned} \det(A - \lambda I) &= (8-\lambda)(5-\lambda)(2-\lambda) + 28 + 162 - 24(5-\lambda) - 63(8-\lambda) - 3(2-\lambda) \\ &= -360 + 24\,\lambda + 15\,\lambda^2 - \lambda^3 \end{aligned}$$

Das charakteristische Polynom von A lautet daher $p(\lambda) = -360 + 24\,\lambda + 15\,\lambda^2 - \lambda^3$, und seine Wurzeln (Nullstellen) können mit Verfahren des Kapitels 1 gefunden werden. Die drei Wurzeln, und damit die drei Eigenwerte von A sind $\lambda_1 = 15, \lambda_{2,3} = \pm 4,89898$.
Für größere Matrizen ist dieses Rechenverfahren zu umständlich und zu anfällig gegenüber Rundungsfehlern.

Kennt man einen Eigenwert λ, dann findet man einen dazu passenden Eigenvektor als Lösung des Gleichungssystems (23).

Für $\lambda = 15$ im obigen Beispiel suchen wir also eine Lösung des Systems

$$\begin{bmatrix} -7 & 1 & 6 \\ 3 & -10 & 7 \\ 4 & 9 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

In diesem System ergibt die Summe aus erster und zweiter Gleichung genau das Negative der dritten. Wenn Sie sich nun wundern, wie Sie aus nur zwei linear unabhängigen Gleichungen die drei Unbekannten $x_1,\ x_2,\ x_3$ bestimmen sollen, haben Sie die Pointe nicht verstanden: Das Gleichungssystem *darf nicht eindeutig lösbar sein!* Eben deswegen haben wir mit der Determinantenbedingung (24) dafür gesorgt, dass es außer der trivialen Lösung $x_1 = x_2 = x_3 = 0$ noch andere, sinvolle Lösungen gibt. Wir können zum Beispiel $x_1 = 1$ frei wählen und in die ersten beiden Gleichungen einsetzen, dann lassen sich x_2 und x_3 bestimmen:

$$\begin{array}{rcrcr} x_2 & + & 6x_3 & = & 7 \\ -10x_2 & + & 7x_3 & = & -3 \end{array} \qquad \text{Lösung: } x_2 = 1,\ x_3 = 1$$

9.2 Eigenwertaufgabe $Ax = \lambda x$: weitere Eigenschaften

Matrizen in physikalischen Anwendungen sind zumeist symmetrisch (Steifigkeitsmatrix, Spannungstensor, Trägheitstensor...). Die zugehörigen Eigenwerte entsprechen physikalischen Größen (Schwingungsfrequenzen, Hauptspannungen, Trägheitsmomenten...), von denen man erwartet, dass sie reell sind. Da ist es beruhigend zu wissen, dass die Mathematik garantiert:

Alle Eigenwerte einer symmetrischen Matrix sind reell.

Für eine symmetrische Matrix A vereinfacht sich die Eigenwertaufgabe beträchtlich. Dazu gibt es eine reichhaltige und elegante mathematische Theorie und eine Fülle von Verfahren.

Die Eigenvektoren einer symmetrischen Matrix bilden ein orthogonales System.

Sind die Eigenvektoren einer symmetrischen Matrix A zusätzlich auch auf Länge 1 normiert, dann lassen sie sich in einer orthogonalen Matrix Q zusammenfassen. Dann gilt:

$$Q^T A Q = D \quad \text{mit } D \text{ Diagonalmatrix aus Eigenwerten}\ .$$

Eigenwerte lassen sich „verschieben“

Ist λ ein Eigenwert von A und $\mathbf{x}$ ein zugehöriger Eigenvektor, dann ist für beliebiges $s \in \mathbb{R}$ der Wert $\lambda + s$ Eigenwert von $A + sI$ und dasselbe $\mathbf{x}$ ist zugehöriger Eigenvektor.

Eigenwerte der Inversen Matrix sind Inverse der Eigenwerte

Ist λ ein Eigenwert von A mit zugehorigem Eigenvektor $\mathbf{x}$, dann ist $1/\lambda$ ein Eigenwert von A^{-1} mit demselben Eigenvektor $\mathbf{x}$.

Ähnlichkeitstransformation

Ist X eine invertierbare Matrix[13] , dann haben A und $X^{-1}AX$ die gleichen Eigenwerte. Die Transformation von A zu $X^{-1}AX$ heißt *Ähnlichkeitstransformation* .

Moderne Rechenverfahren nützen Ähnlichkeitstransformationen, die Verschiebung von Eigenwerten und das Berechnen inverser Eigenwerte geschickt aus. Ohne solche Tricks wären die Verfahren deutlich langsamer.

Diagonalisieren

Hat eine $n \times n$-Matrix A genau n linear unabhängige Eigenvektoren, dann kann man sie als Spaltenvektoren zu einer $n \times n$-Matrix V zusammenfassen. Die Ähnlichkeitstransformation mit V erzeugt eine Diagonalmatrix D; in deren Haupdiagonale stehen die Eigenwerte.

$$V^{-1}AV = D$$

Beispiel: die Matrix $A = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}$ vom Beispiel auf Seite 76 hat Eigenvektoren $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ und $\begin{bmatrix} 3 \\ -1 \end{bmatrix}$.
Hier sind also

$$V = \begin{bmatrix} 0 & 3 \\ 1 & -1 \end{bmatrix} \text{ und } V^{-1} = \begin{bmatrix} 1/3 & 1 \\ 1/3 & 0 \end{bmatrix}, \quad V^{-1}AV = \begin{bmatrix} 1/3 & 1 \\ 1/3 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 3 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$$

Der MATLAB-Befehl zur Berechnung von Eigenwerten und -vektoren lautet `[V,D] = eig(A)`. Das im Hintergrund laufende Rechenverfahren führt iterativ Ähnlichkeitstransformationen durch, so dass sich die transformierte Matrix schrittweise einer Diagonalmatrix annähert. Als Endergebnis liefert es die Diagonalmatrix D und die Matrix V der Ähnlichkeitstransformation. Dann gilt: in der Hauptdiagonale von D stehen die Eigenwerten von A, die zugehörigen Eigenvektoren stehen in den Spalten von V.

Es kann sein, dass eine $n \times n$-Matrix A nicht n linear unabhängige Eigenvektoren hat. Das verkompliziert die Situation. Zum Glück treten – wie schon erwähnt – in wichtigen Anwendungen symmetrische Matrizen auf. Für sie gibt es immer eine Transformationsmatrix aus linear unabhängigen Eigenvektoren. Als zusätzlicher Bonus gilt: die Transformationsmatrix lässt sich als *orthogonale* Matrix Q wählen.

Kurzfassung:

> Symmetrische Matrizen lassen sich durch orthogonale Transformationen diagonalisieren:
>
> $$Q^T AQ = D \ .$$
>
> Die Spalten von Q sind Eigenvektoren von A mit den entsprechenden Diagonalelementen von D als Eigenwerten.

[13] gleichbedeutend sind: nichtsinguläre Matrix, reguläre Matrix

9.3 Vektoriteration

Die Berechnung von Eigenwerten und -vektoren ist in vielen Anwendungen von Bedeutung, beginnend mit der Mechanik (Eigenschwingungen) bis zur Ökonomie (Durchschnittspreise mit maximalem Gewinn) und zur Bewertung der Wichtigkeit von Internetseiten (PageRank einer Homepage als Eigenvektor der Google-Matrix, ein einfaches Beispiel findet sich in den Übungsaufgaben).

Dabei ist es nicht immer notwendig, alle Eigenwerte zu kennen, oft genügt es, den größten Eigenwert oder jenen mit größten Absolutbetrag zu berechnen.

Es kann in Ausnahmefällen mehrere Eigenwerte mit gleichem Absolutbetrag geben. Gibt es aber unter allen Eigenwerten *genau einen* mit maximalem Betrag, so nennen wir ihn *dominanten* Eigenwert. Ein solcher Eigenwert λ und ein zugehöriger Eigenvektor lassen sich durch ein einfaches iteratives Verfahren als Fixpunkt des Systems

$$\mathbf{x} = \frac{1}{\lambda} A\mathbf{x}$$

finden: die sogenannte Vektoriteration (englisch: *power iteration*).

Die Vektoriteration geht (ausgehend von früheren Ansätzen) auf Richard von Mises[14] und Hilda Geiringer[15] zurück. Kurzfassung: Man nehme irgendeinen Vektor $\mathbf{x}^{(0)}$ und multipliziere ihn mit A. Das Ergebnis skaliere man geeignet durch Division mit einem Faktor $\lambda^{(1)}$ und nenne es $\mathbf{x}^{(1)}$. Dies Verfahren setze man fort, bis sich die $\mathbf{x}^{(k)}$ nur mehr vernachlässigbar ändern. Dann ist $\lambda^{(k)}$ der Eigenwert mit größtem Absolutbetrag und $\mathbf{x}^{(k)}$ ein zugehöriger Eigenvektor.

Vektoriteration für den betragsgrößten Eigenwert
(Potenzmethode nach v. Mises und Geiringer, *power iteration*)
Gegeben eine $n \times n$-Matrix A, ein Startvektor $\mathbf{x}^{(0)} \neq 0$ und ein fix gewähltes $i \in \{1, \ldots, n\}$

Iteriere für $k = 1, 2, \ldots$
berechne $\mathbf{y}^{(k)} = A\mathbf{x}^{(k-1)}$
setze $\lambda^{(k)} = y_i^{(k)}$ (i-te Komponente von $\mathbf{y}^{(k)}$)
setze $\mathbf{x}^{(k)} = \mathbf{y}^{(k)}/\lambda^{(k)}$ (Skalierung)

Mit Papier und Bleistift ist die Multiplikation $A\mathbf{x}$ von tödlicher Kompliziertheit; Computer hingegen finden diese Methode recht einfach. Sie konvergiert, wenn es einen dominanten Eigenwert gibt und der Startvektor eine Komponente in Richtung des zugehörigen Eigenvektors hat.

Normaler Weise ist es irrelevant, welche Komponente von $\mathbf{x}^{(k)}$ zur Skalierung gewählt wird, außer der Eigenvektor hat genau für diese Komponente den Wert 0. Eine alternative Skalierunsvorschrift vermeidet die Festlegung auf eine bestimmte Komponente; das ist für Rechenprogramme einfacher:

Finde die betragsgrößte Komponente $y_i^{(k)}$ und setze $\lambda^{(k)} = y_i^{(k)}$.

[14] Richard von Mises, 1883 (Lemberg, Österreich; heute Lwiw, Ukraine) –1953 (Boston, USA). Mathematiker, studiert in Wien, ist Professor in Straßburg, Dresden, Berlin, Istanbul und ab 1939 an der Harvard University; richtungsweisende Arbeiten auf fast allen Gebieten der angewandten Mathematik, sowie der Strömungslehre, speziell unter Berücksichtigung des Flugzeugbaus (hält 1913 die erste Vorlesung über Motorflug!).

[15] Hilda Geiringer (Wien, 1893 – Santa Barbara, Kalifornien, 1973), studiert in Wien, arbeitet in Berlin als Assistentin von v. Mises am Inst. f. Angewandte Mathematik über Wahrscheinlichkeitstheorie und Statistik, emigriert 1933, lehrt in Brüssel, Istanbul und verschiedenen amerikanischen Universaitäten

Durch Anwenden der Potenzmethode auf die Matrix A^{-1} kann man den *betragskleinsten* Eigenwert ermitteln. Allgemein kann man jenen Eigenwert, welcher einem Wert μ am nächsten liegt, durch die Potenzmethode, angewandt auf $(A-\mu I)^{-1}$ finden (*inverse Iteration*). Dabei werden A^{-1} oder $(A-\mu I)^{-1}$ allerdings nicht explizit berechnet. Vielmehr wird der Iterationsschritt $\mathbf{y}^{(k)} = A^{-1}\mathbf{x}^{(k-1)}$ umformuliert zu $A\mathbf{y}^{(k)} = \mathbf{x}^{(k-1)}$ und $\mathbf{y}^{(k)}$ als Lösung dieses Systems bestimmt. Falls μ eine gute Näherung an ein λ_i ist, konvergiert das Verfahren rasch.

Für große, schwach besetzte symmetrische Matrizen gibt es ein besonders effizientes Verfahren, wenn nur der größte oder einige von den größten (oder kleinsten) Eigenwerten gesucht sind, das Lanczos-Verfahren. Im Kern stecken dort, wie bei der Potenzmethode, Iterationen der Form $\mathbf{y}^{(k)} = A\mathbf{x}^{(k-1)}$.

Auch dazu gibt es MATLAB-Befehle. Zum Beispiel liefert `d = eigs(A)` für schwach besetzte Matrizen die sechs betragsgrößten Eigenwerte und zugehörige Eigenvektoren.

9.4 Einfache Formen

Viele wichtige Verfahren zur Eigenwertberechnung bringen eine Matrix A durch eine Folge von Ähnlichkeitstransformationen $X^{-1}AX$ auf eine einfache Form und berechnen dann die Eigenwerte. (Die Eigenwerte ändern sich bei Ähnlichkeitstransformationen nicht.) Standard-Verfahren ist der sogenannte QR-Algorithmus.

Einfache Formen

- Diagonalmatrix
- Dreiecksmatrix
- Tridiagonalmatrix

Für Diagonal- und Dreiecksmatrizen sind die Eigenwerte genau die Diagonalelemente.

Für eine Tridiagonalmatrix lässt sich das charakteristische Polynom leicht auswerten, ohne dass man es explizit anschreiben müsste. Für eine symmetrische Tridiagonalmatrix T,

$$T = \begin{bmatrix} a_1 & b_1 & & \cdots & 0 \\ b_1 & a_2 & b_2 & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & b_{n-1} \\ 0 & \cdots & & b_{n-1} & a_n \end{bmatrix},$$

zeigt Entwickeln der Determinante $\det(T - xI)$, dass der Wert p(x) des charakteristischen Polynoms rekursiv so bestimmt werden kann:

setze $p_0(x) = 1, p_{-1}(x) = 0$.
Für $k = 1, \ldots, n$
  $p_k(x) = (a_k - x)p_{k-1}(x) - b_{k-1}^2 p_{k-2}(x)$
Ergebnis $p(x) = p_n(x)$.

Ein Verfahren zur Nullstellenbestimmung, zum Beispiel Intervallhalbierung, liefert mit dieser Methode die Eigenwerte als Nullstellen des charakteristischen Polynoms.