PVK Numerische Methoden
Tag 3

Lucas Böttcher

ETH Zürich
Institut für Baustoffe
Wolfgang-Pauli-Str. 27
HIT G 23.8
8093 Zürich

lucasb@ethz.ch

June 21, 2017

Übersicht

Kurstag	**Inhalt**
Tag 1	Differenzialgleichungen
Tag 2	Quadratur, Nichtlineare algebraische Gleichungen
Tag 3	Ausgleichsrechnung, Eigenwerte, Interpolation
Tag 4	Lineare ODEs, Exponentielle Integratoren, Splitting-Verfahren
Tag 5	Erweiterte Übungen

Kapitel IV
Ausgleichsrechnung (pp. 60–75 im Skript)

Ausgleichsrechnung

Für einen gegebenen Datensatz sollen die optimalen Parameterwerte für eine bestimmte funktionale Abhängigkeit gefunden werden.

Lernziele: Kennen und Anwenden von numerischen Methoden um lineare und nichtlineare Probleme der Ausgleichsrechnung zu lsen.

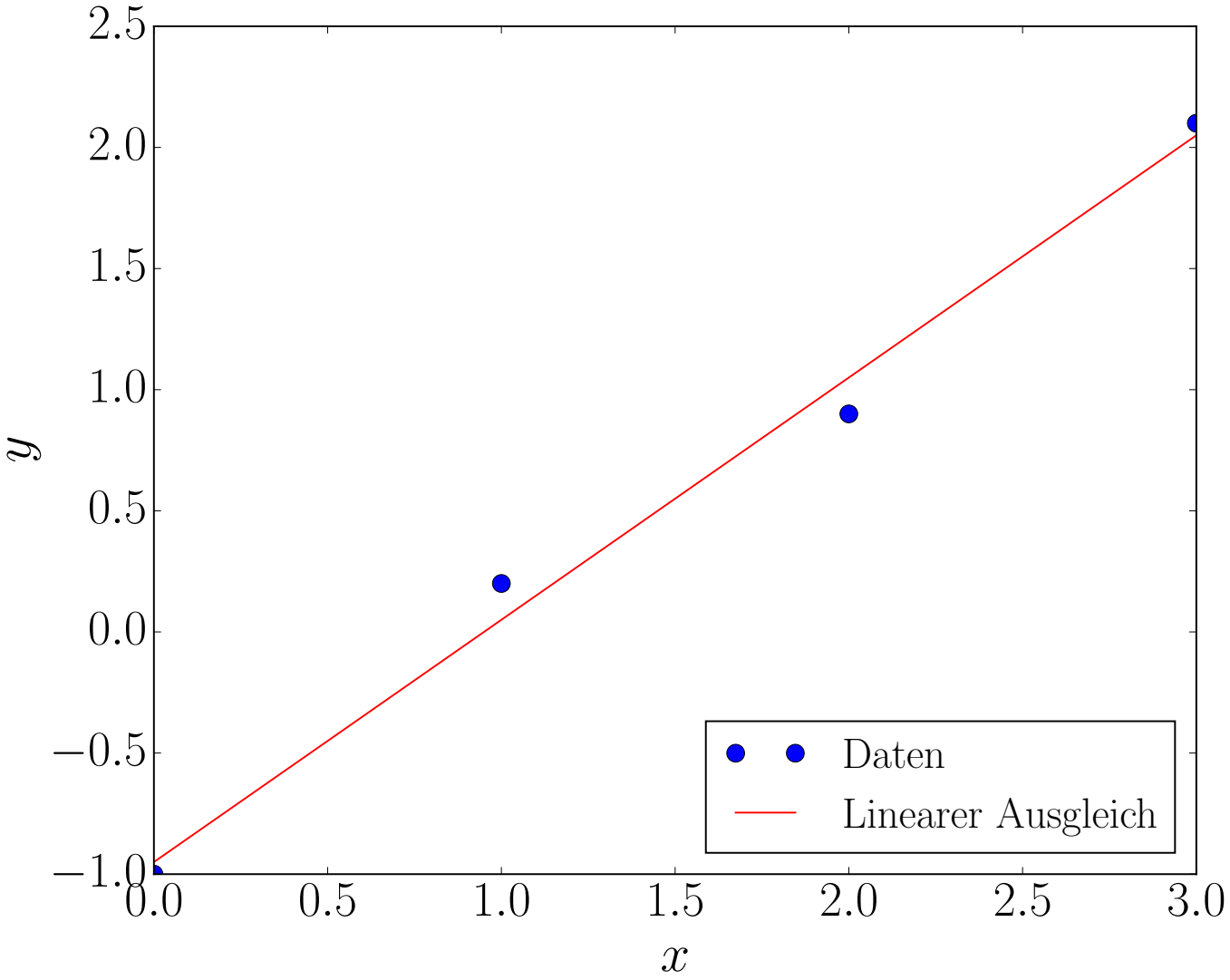
Lineare Ausgleichsrechnung

Grundidee: Für Messwerte (x_i, y_i) mit $i \in \{1, \ldots, N\}$ und einem erwarteten linearen Zusammenhang $y = mx + n$, kann man die Methode der kleinsten Quadrate anwenden, um die Parameter m und n zu bestimmen.

Lineare Ausgleichsrechnung

```
x = np.array([0, 1, 2, 3])
y = np.array([-1, 0.2, 0.9, 2.1])

# y = Ap mit A = [[x 1]] und p = [[m], [n]]
A = np.vstack([x, np.ones(len(x))]).T
m, n = np.linalg.lstsq(A, y)[0]
print m, n

# oder polyfit
m_poly, n_poly = np.polyfit(x, y, 1)
print m_poly, n_poly
```

Lineare Ausgleichsrechnung



Nichtlineare Ausgleichsrechnung

Grundidee: Für Messwerte (x_i, y_i) mit $i \in \{1, \ldots, N\}$ und einem erwarteten nichtlinearen Zusammenhang minimiert man $\Phi(\mathbf{a}) := \frac{1}{2}\|F(\mathbf{a})\|_2^2$ mit dem Parametervektor $\mathbf{a} \in \mathbb{R}^n$, wobei

$$F(\mathbf{a}) = \begin{bmatrix} f(x_1, \mathbf{a}) - y_1 \\ \vdots \\ f(x_n, \mathbf{a}) - y_n \end{bmatrix}.$$

Nichtlineare Ausgleichsrechnung

Grundidee: Für Messwerte (x_i, y_i) mit $i \in \{1, \dots, N\}$ und einem erwarteten nichtlinearen Zusammenhang minimiert man $\Phi(\mathbf{a}) := \frac{1}{2}\|F(\mathbf{a})\|_2^2$ mit dem Parametervektor $\mathbf{a} \in \mathbb{R}^n$, wobei

$$F(\mathbf{a}) = \begin{bmatrix} f(x_1, \mathbf{a}) - y_1 \\ \vdots \\ f(x_n, \mathbf{a}) - y_n \end{bmatrix}.$$

Man kann die Nullstelle von $\nabla\Phi = 0$ mittels eines Newton-Verfahrens suchen, cf. p. 69 im Skript.

Oder man löst die lineare Approximation von $F(\mathbf{a})$ mit dem Gauss-Newton-Verfahren, cf. p. 70 im Skript.

Nichtlineare Ausgleichsrechnung

Grundidee: Für Messwerte (x_i, y_i) mit $i \in \{1, \ldots, N\}$ und einem erwarteten nichtlinearen Zusammenhang minimiert man $\Phi(\mathbf{a}) := \frac{1}{2}\|F(\mathbf{a})\|_2^2$ mit dem Parametervektor $\mathbf{a} \in \mathbb{R}^n$, wobei

$$F(\mathbf{a}) = \begin{bmatrix} f(x_1, \mathbf{a}) - y_1 \\ \vdots \\ f(x_n, \mathbf{a}) - y_n \end{bmatrix}.$$

Man kann die Nullstelle von $\nabla\Phi = 0$ mittels eines Newton-Verfahrens suchen, cf. p. 69 im Skript.

Oder man löst die lineare Approximation von $F(\mathbf{a})$ mit dem Gauss-Newton-Verfahren, cf. p. 70 im Skript.

$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} - \mathbf{s} \text{ mit } \mathbf{s} := \operatorname*{argmin}_{\mathbf{z} \in \mathbb{R}^n} \|F\left(\mathbf{a}^{(k)}\right) - J_F\left(\mathbf{a}^{(k)}\right)\mathbf{z}\|_2^2.$$

Beispiel nichtlineare Ausgleichsrechnung

Beispiel nichtlineare Ausgleichsrechnung

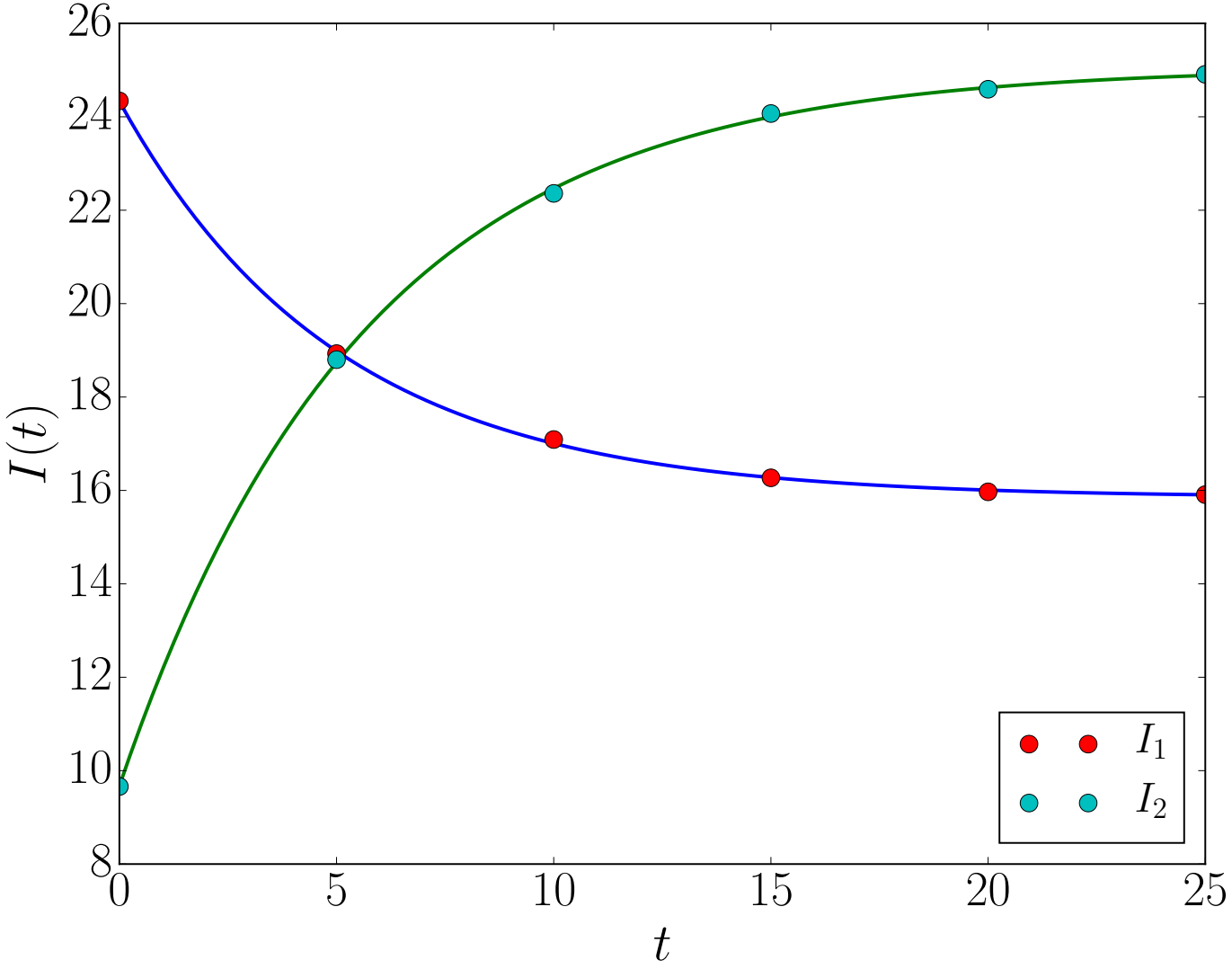
Finde die Parameter α, β und γ für den funktionellen Zusammenhang $I(t) = \alpha \pm \beta e^{-\gamma t}$ und die folgenden Messwerte mit dem Gauss-Newton-Verfahren, cf. p. 60 im Skript:

Zeit [min]	Stromstärke [mA]	
0	9.66	24.34
5	18.8	18.93
10	22.36	17.09
15	24.07	16.27
20	24.59	15.97
25	24.91	15.91

Beispiel nichtlineare Ausgleichsrechnung

```
def gauss_newton(x,F,J,tol):
    s = lstsq(J(x),F(x))[0]
    x = x-s
    while norm(s) > tol*norm(x):
        s = lstsq(J(x),F(x))[0]
        x = x-s
    return x
```

Beispiel nichtlineare Ausgleichsrechnung



Kapitel V
Eigenwerte (pp. 76–107 im Skript)

Eigenwerte

Eigenwerte treten in vielen Bereichen, wie zum Beispiel bei Schwingungsproblemen oder beim Entkoppeln von Differentialgleichungen auf. Deren (numerische) Bestimmung ist damit von grosser Bedeutung.

Lernziele: Kennen und Anwenden von numerischen Methoden um ein gegebenes Eigenwertproblem zu lösen.

Beispiel Vibration einer Saite

Beispiel Vibration einer Saite

Eine an beiden Enden fixierte Saite unter Spannung T wird beschrieben durch folgende Differentialgleichung:

$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{T}{m(x)} \frac{\partial^2 u(x,t)}{\partial x^2}$ mit der Masse $m(x)$.

Beispiel Vibration einer Saite

Beispiel Vibration einer Saite

Eine an beiden Enden fixierte Saite unter Spannung T wird beschrieben durch folgende Differentialgleichung:

$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{T}{m(x)} \frac{\partial^2 u(x,t)}{\partial x^2}$ mit der Masse $m(x)$.

Mit dem Ansatz $u(x,t) = \psi(x) e^{-i\omega t}$ findet man:

$$\frac{T}{m(x)} \frac{d^2 \psi(x)}{dx^2} + \omega^2 \psi(x) = 0.$$

Beispiel Vibration einer Saite

Beispiel Vibration einer Saite

Eine an beiden Enden fixierte Saite unter Spannung T wird beschrieben durch folgende Differentialgleichung:

$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{T}{m(x)} \frac{\partial^2 u(x,t)}{\partial x^2}$ mit der Masse $m(x)$.

Mit dem Ansatz $u(x,t) = \psi(x) e^{-i\omega t}$ findet man:

$\frac{T}{m(x)} \frac{d^2\psi(x)}{dx^2} + \omega^2 \psi(x) = 0.$

Unterteile das Interval der Länge $L = 2$ in $N = 513$ Stücke mit Länge h und stelle die Matrix A für das Eigenwertproblem $A\psi = -\omega^2 \psi$ auf.

Die dazugehörige Diskretisierung ist (zur Vereinfachung $m_i = 1$):

$\frac{T}{m_i} \frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{h^2} + \omega^2 \psi_i = 0.$

Beispiel Vibration einer Saite

Beispiel Vibration einer Saite

Die dazugehörige Diskretisierung ist (zur Vereinfachung $m_i = 1$):

$$\frac{T}{m_i}\frac{\psi_{i-1}-2\psi_i+\psi_{i+1}}{h^2} + \omega^2\psi_i = 0.$$

In Matrixschreibweise findet man $A\psi = -\omega^2\psi$ wobei

$$A = \frac{T}{h^2}\begin{pmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & & \ddots & & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N-1\times N-1}.$$

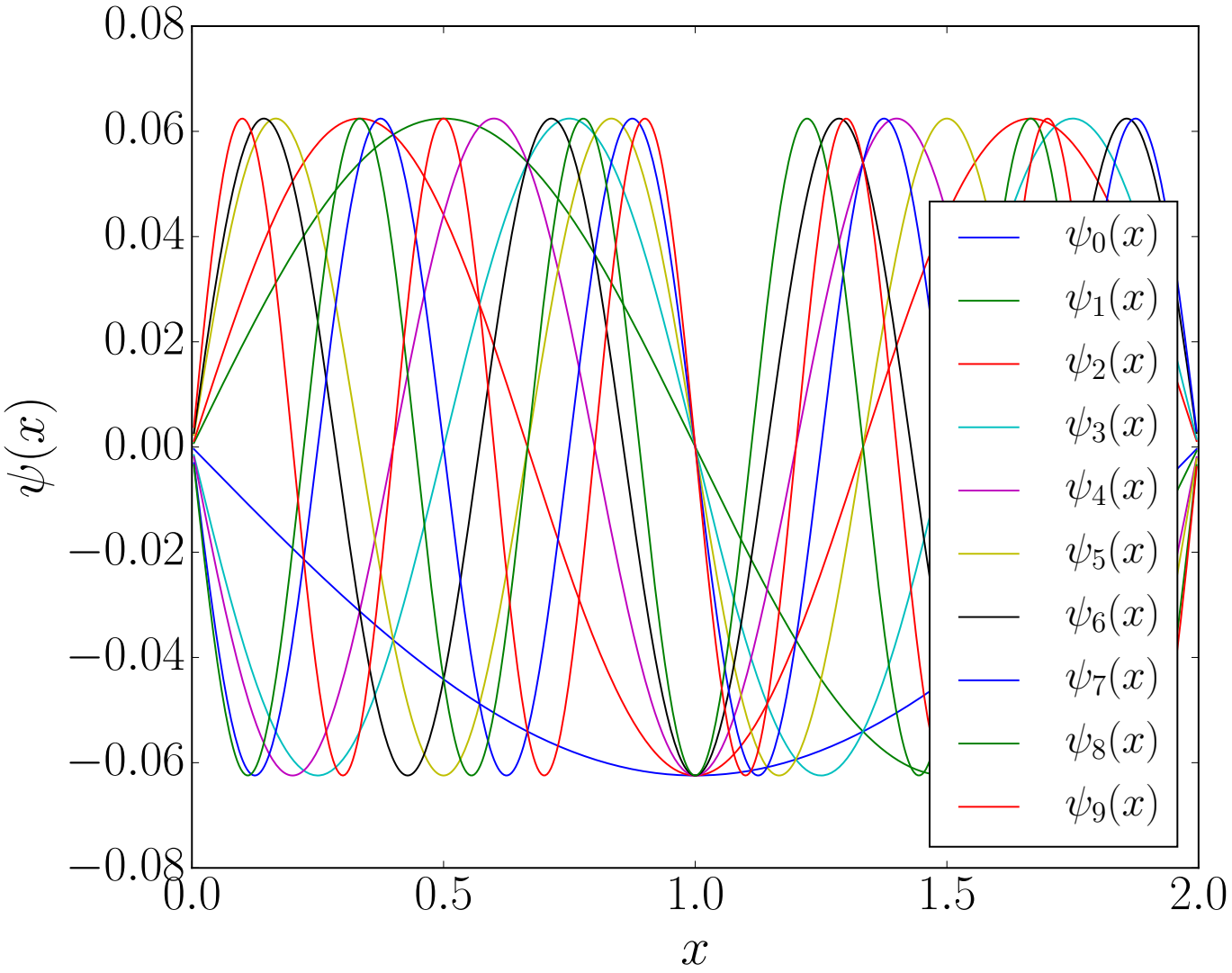
Finde den kleinsten und grössten Eigenwert mit numpy.linalg.eigh (für symmetrische Matrizen) und der Powermethode.

Plotte die Eigenfunktionen mit den 10 kleinsten Eigenwerten.

Beispiel Vibration einer Saite

```
def power_method(A, n_rep = 2000):
    x = np.random.random(A.shape[0])
    x = x / np.linalg.norm(x)

    for k in range(n_rep):
        x = np.dot(A, x)
        x = x / np.linalg.norm(x)

    eigenvalue = np.dot(x.transpose(), np.dot(A, x))

    return eigenvalue

ewh, evh = np.linalg.eigh(A)
```

Beispiel Vibration einer Saite



Krylov-Unterraum-Verfahren

Für grosse dünnbesetzte Matrizen sind Krylov-Verfahren gut geeignet, wenn man nicht an allen Eigenwerten interessiert ist. Weitere Informationen sind im Skript p. 93 zu finden.

Kapitel VI
Interpolation (pp. 108–157 im Skript)

Interpolation

Für eine gegebene, analytisch schwer zugängliche Funktion ist es in manchen Situationen angebracht diese durch einfachere Funktionen zu approximieren.

Lernziele: Kennen und Anwenden von gängigen Interpolationsverfahren.

Interpolation

Interpolation

Grundidee: Für eine Menge von N Funktionswerten $(x_i, y_i) \in \mathbb{R}^2$ mit $i \in \{1, \ldots, N\}$ einer Funktion f, soll eine Interpolationsfunktion $\tilde{f}$ gefunden werden.

Interpolation

Interpolation

Grundidee: Für eine Menge von N Funktionswerten $(x_i, y_i) \in \mathbb{R}^2$ mit $i \in \{1, \ldots, N\}$ einer Funktion f, soll eine Interpolationsfunktion $\tilde{f}$ gefunden werden.

Die Interpolationsfunktion erfüllt $\tilde{f}(x_i) = y_i$ dabei exakt.

Interpolation

Interpolation

Grundidee: Für eine Menge von N Funktionswerten $(x_i, y_i) \in \mathbb{R}^2$ mit $i \in \{1, \ldots, N\}$ einer Funktion f, soll eine Interpolationsfunktion $\tilde{f}$ gefunden werden.

Die Interpolationsfunktion erfüllt $\tilde{f}(x_i) = y_i$ dabei exakt.

Man findet für eine Basis b_j, dass $f(x) \approx f_n(x) = \sum_{j=1}^{n} \alpha_j b_j$.

Interpolation

Interpolation

Grundidee: Für eine Menge von N Funktionswerten $(x_i, y_i) \in \mathbb{R}^2$ mit $i \in \{1, \ldots, N\}$ einer Funktion f, soll eine Interpolationsfunktion $\tilde{f}$ gefunden werden.

Die Interpolationsfunktion erfüllt $\tilde{f}(x_i) = y_i$ dabei exakt.

Man findet für eine Basis b_j, dass $f(x) \approx f_n(x) = \sum_{j=1}^{n} \alpha_j b_j$.

Beispiele sind:

$b_j = x^{j-1}$ (Polynomiale Interpolation),
$b_j = \cos((j-1)\arccos(x))$ (Chebychev Interpolation),
$b_j = e^{2\pi i j x}$ (Trigonometrische Interpolation).

Interpolation

Interpolationsverfahren:

1. Polynomiale Interpolation
2. Chebychev Interpolation
3. Trigonometrische Interpolation

Polynomiale Interpolation

Grundidee: Wir nutzen als Beispiel die Rungefunktion $f(x) = \frac{1}{1+x^2}$ und interpolieren diese mit einem Polynom $P_n(x)$ im Interval $[-5, 5]$.

Polynomiale Interpolation

Grundidee: Wir nutzen als Beispiel die Rungefunktion $f(x) = \frac{1}{1+x^2}$ und interpolieren diese mit einem Polynom $P_n(x)$ im Interval $[-5, 5]$.

Es werden m Stützstellen verwendet $\{(x_i, f(x_i))\}_{i=1}^m$ und man findet das lineare Ausgleichsproblem $Ac = b$ mit dem Koeffizientenvektor c.

Polynomiale Interpolation

Grundidee: Wir nutzen als Beispiel die Rungefunktion $f(x) = \frac{1}{1+x^2}$ und interpolieren diese mit einem Polynom $P_n(x)$ im Interval $[-5, 5]$.

Es werden m Stützstellen verwendet $\{(x_i, f(x_i))\}_{i=1}^m$ und man findet das lineare Ausgleichsproblem $Ac = b$ mit dem Koeffizientenvektor c.

Oder in folgender Schreibweise

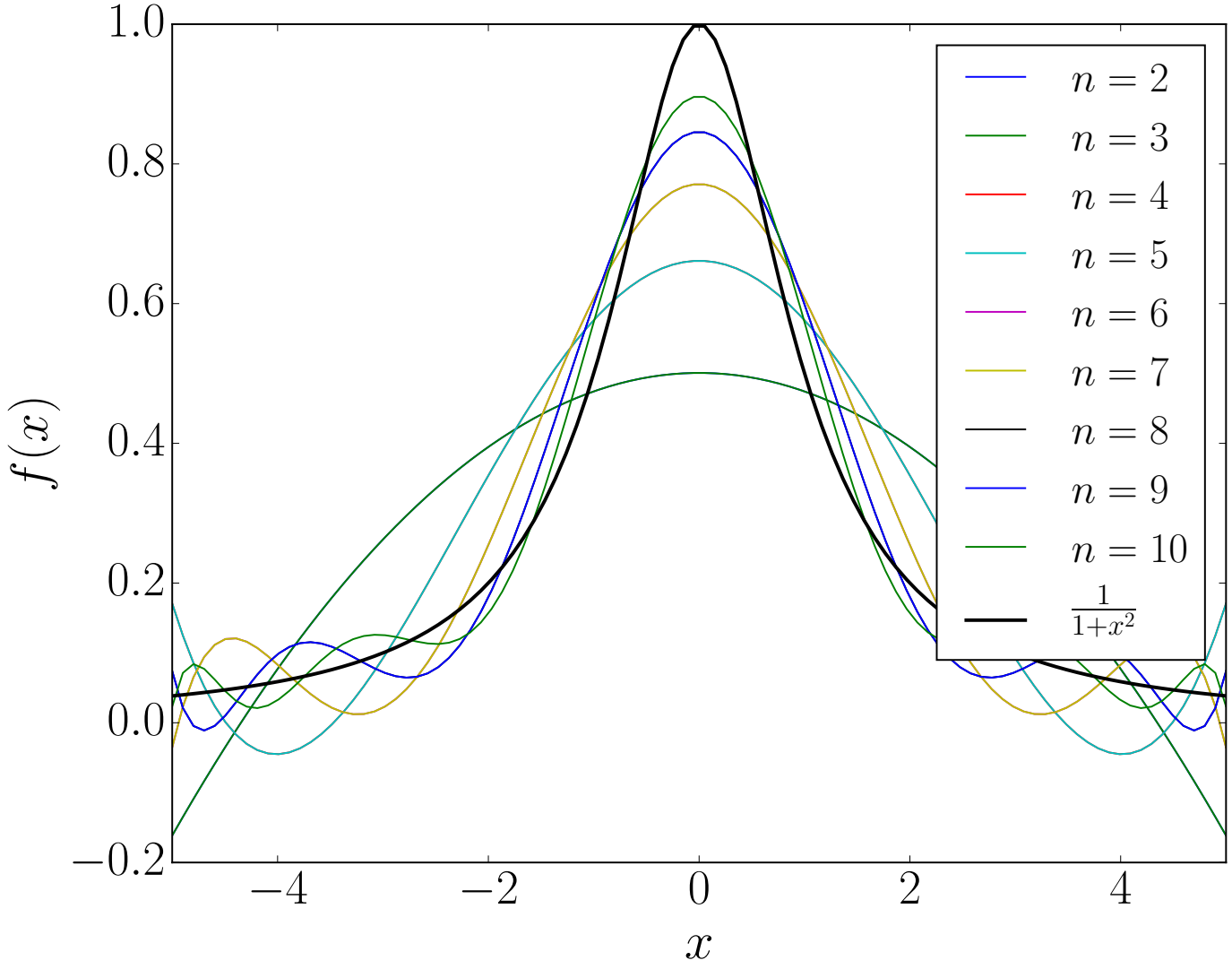
$$\begin{pmatrix} 1 & x_1 & \dots & x_1^n \\ \vdots & & & \vdots \\ 1 & x_m & \dots & x_m^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

für das Polynom $P_n(x) = \sum_{i=0}^n a_i x^i$,

Wir lösen das Interpolationsproblem für $2 \leq n \leq 10$ für $m = 40$.

Polynomiale Interpolation

Wir lösen das Interpolationsproblem für $2 \leq n \leq 10$ für $m = 40$.



Polynomiale Interpolation

```
def build_matrix(degree, n_samples):
    exponent = np.arange(0, degree+1)
    x = np.linspace(-5, 5, n_samples)

    A = x[:,np.newaxis] ** exponent[np.newaxis,:]
    b = f(x)
    return A, b

for i in range(2, 11):
    A, b = build_matrix(i, 40)
    c = scipy.linalg.lstsq(A, b)[0]
    # from highest to lowest degree
    y = np.polyval(c[::-1], x)
```

Interpolation

Interpolationsverfahren:

1. Polynomiale Interpolation
2. Chebychev Interpolation
3. Trigonometrische Interpolation

Chebychev Interpolation

Chebychev Interpolation

Grundidee: Um den Interpolationsfehler zu verringern nutzt man auch nicht-äquidistant verteilte Stützstellen. Hier betrachten wir das Beispiel der Chebychev Interpolation.

Chebychev Interpolation

Chebychev Interpolation

Grundidee: Um den Interpolationsfehler zu verringern nutzt man auch nicht-äquidistant verteilte Stützstellen. Hier betrachten wir das Beispiel der Chebychev Interpolation.

Für eine Funktion $f: [-1,1] \to \mathbb{C}$ sind die Chebychevknoten gegeben durch $t_k := \cos\left(\frac{2k+1}{n+1}\frac{\pi}{2}\right)$ mit $k = 0, \ldots, n$, cf. p. 152 im Skript. Und es gilt $f(t_k) = p(t_k)$ für das Chebychev-Interpolationspolynom.

Chebychev Interpolation

Chebychev Interpolation

Grundidee: Um den Interpolationsfehler zu verringern nutzt man auch nicht-äquidistant verteilte Stützstellen. Hier betrachten wir das Beispiel der Chebychev Interpolation.

Für eine Funktion $f: [-1,1] \to \mathbb{C}$ sind die Chebychevknoten gegeben durch $t_k := \cos\left(\frac{2k+1}{2(n+1)\pi}\right)$ mit $k = 0, \ldots, n$, cf. p. 152 im Skript. Und es gilt $f(t_k) = p(t_k)$ für das Chebychev-Interpolationspolynom.

Für ein allgemeines Intevall $[a,b]$ sind die Chebychevknoten gegeben durch $t_k = a + \frac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)\pi}\right) + 1\right)$, cf. p. 121 im Skript.

Chebychev Interpolation

Grundidee: Um den Interpolationsfehler zu verringern nutzt man auch nicht-äquidistant verteilte Stützstellen. Hier betrachten wir das Beispiel der Chebychev Interpolation.

Für eine Funktion $f: [-1,1] \to \mathbb{C}$ sind die Chebychevknoten gegeben durch $t_k := \cos\left(\frac{2k+1}{2(n+1)\pi}\right)$ mit $k = 0, \ldots, n$, cf. p. 152 im Skript. Und es gilt $f(t_k) = p(t_k)$ für das Chebychev-Interpolationspolynom.

Für ein allgemeines Intevall $[a,b]$ sind die Chebychevknoten gegeben durch $t_k = a + \frac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)\pi}\right) + 1\right)$, cf. p. 121 im Skript.

Die Chebychevpolynome sind für $x \in [-1,1]$ definiert durch $T_0(x) = 1$, $T_1(x) = x$ und $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$, cf. p. 120 im Skript.

Beispiel Chebychev Interpolation

Beispiel Chebychev Interpolation

Wir betrachten erneut die Rungefunktion $f(x) = \frac{1}{1+x^2}$.

Plotte die Chebychev Interpolation der Rungefunktion im Intervall $[-1, 1]$ für $n = 30$ Stützsstellen.

Studiere den Fehler der Chebychev Interpolation als Abweichung von der Rungefunktion für $n \in \{2, \ldots, 200\}$.

Folgt die Konvergenz einem exponentiellen Verlauf? Wenn ja, bestimme die dazugehörige funktionelle Abhängigkeit.

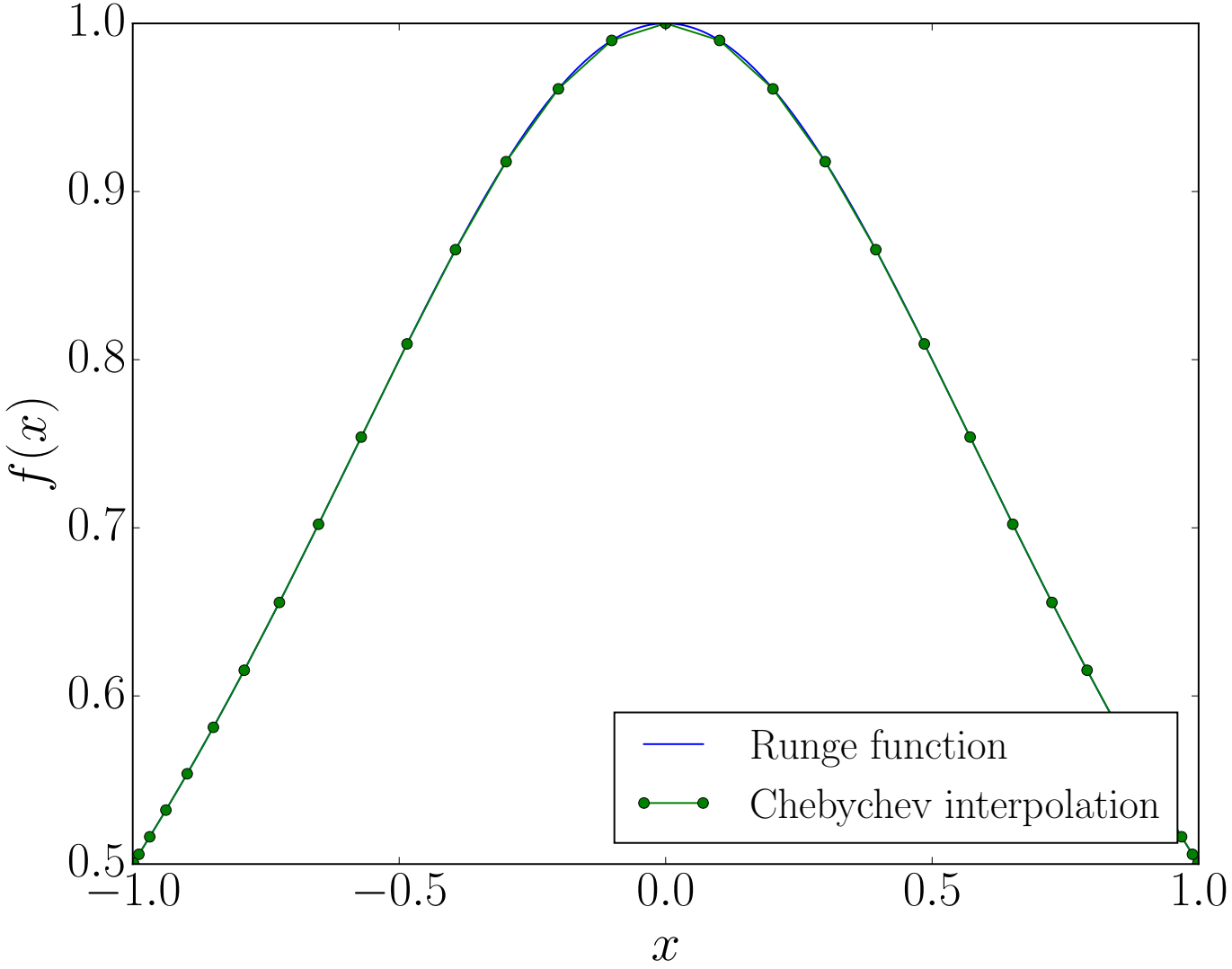
Beispiel Chebychev Interpolation

```
# Clenshaw Algorithmus
def clenshaw(a,x):
    # Polynomgrad
    n = a.shape[0] - 1
    d = tile( reshape(a,n+1,1), (x.shape[0], 1) )
    d = d.T

    for j in range(n, 1, -1):
        d[j-1,:] = d[j-1,:] + 2.0*x*d[j,:]
        d[j-2,:] = d[j-2,:] - d[j,:]

    y = d[0,:] + x*d[1,:]

    return y
```

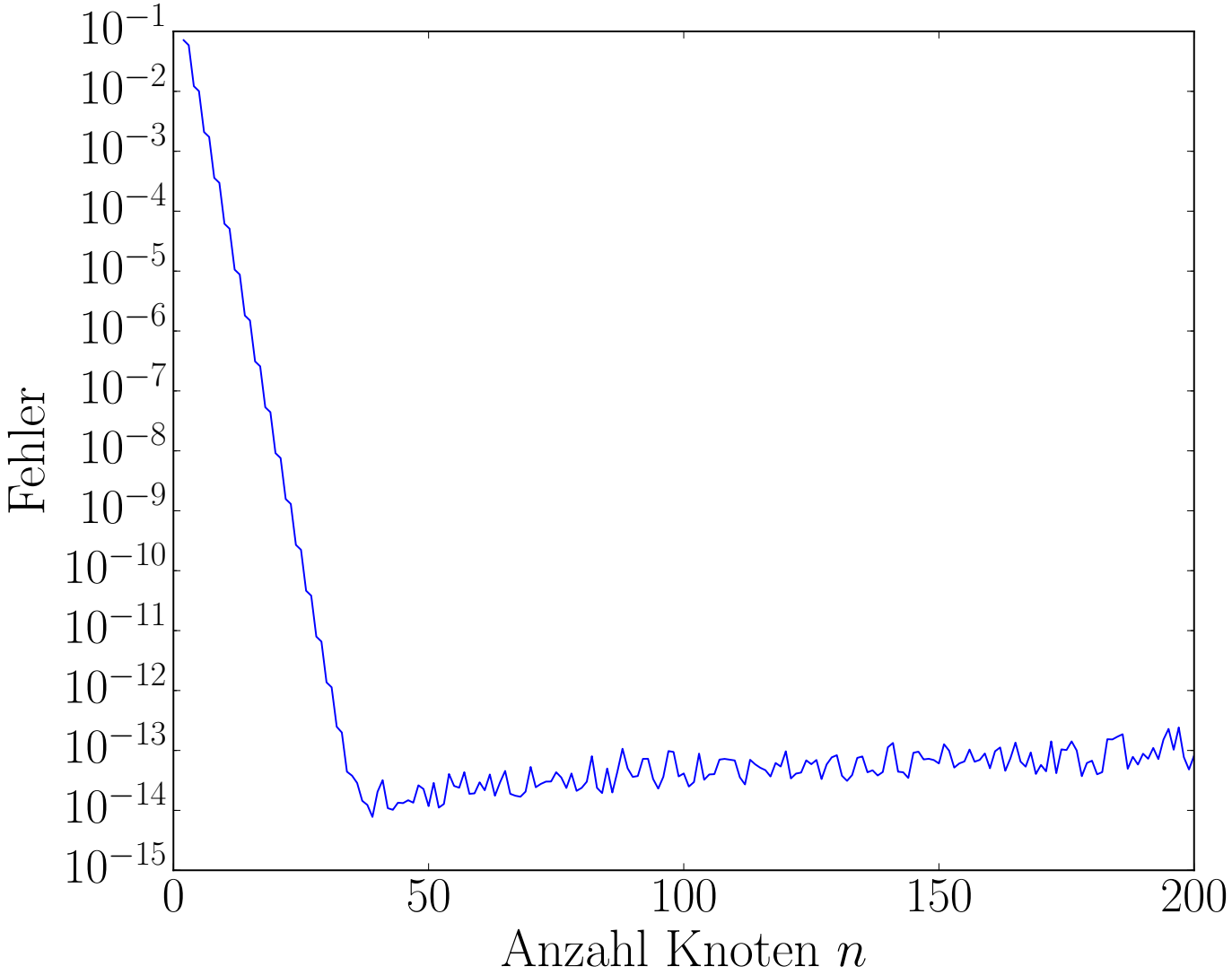
Beispiel Chebychev Interpolation

```
# Chebyshev Interpolation
def chebexp(y):
    # Polynomgrad
    n = y.shape[0] - 1
    # RHS Vektor
    t = arange(0, 2*n+2)
    z = exp(-pi*1.0j*n/(n+1.0)*t) * hstack([y,y[::-1]])
    # Loese lineares System
    c = ifft(z)
    # Berechne b_j
    t = arange(-n, n+2)
    b = real(exp(0.5j*pi/(n+1.0)*t) * c)
    # Berechne a_j
    a = hstack([ b[n], 2*b[n+1:2*n+1] ])
    return a
```

Beispiel Chebychev Interpolation



Beispiel Chebychev Interpolation



Interpolation

Interpolationsverfahren:

1. Polynomiale Interpolation
2. Chebychev Interpolation
3. Trigonometrische Interpolation

Trigonometrische Interpolation

Trigonometrische Interpolation

Grundidee: Für eine Funktion $f \in L^2(0,1)$ ist die dazugehörige Fourierreihe $f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{2\pi ikt}$ mit den dazugehörigen Fourierkoeffizienten $\hat{f}(k) = \int_0^1 f(t) e^{-2\pi ikt} dt$.

Trigonometrische Interpolation

Trigonometrische Interpolation

Grundidee: Für eine Funktion $f \in L^2(0,1)$ ist die dazugehörige Fourierreihe $f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{2\pi ikt}$ mit den dazugehörigen Fourierkoeffizienten $\hat{f}(k) = \int_0^1 f(t) e^{-2\pi ikt} dt$. Wir benutzen im nächsten Beispiel die FFT, um eine Stufenfunktion trigonometrisch zu interpolieren.

Trigonometrische Interpolation

```
def evaliptrig(y,N):
    n = len(y)
    if (n%2) == 0:
        c = fft.ifft(y)
        a = zeros(N, dtype=complex)
        a[:n/2] = c[:n/2]
        a[N-n/2:] = c[n/2:]
        v = fft.fft(a);
        return v
    else: raise TypeError, 'odd length'
```

Trigonometrische Interpolation