

Numerische Mathematik - Projektteil 2

Richard Weiss

Florian Schager

Christian Sallinger

Fabian Zehetgruber

Paul Winkler

Christian Göth

Random code snippet, damit alle checken, wie man code displayt:

```
1 print("Hello World!")
```

1 Eigenschwingungen

1.1 Aufgabestellung

Das Projekt beschäftigt sich mit den Eigenschwingungen einer fest eingespannten Saite. Sei dazu $u(t, x)$ die vertikale Auslenkung der Saite an der Position $x \in [0, 1]$ zur Zeit t . u wird näherungsweise durch die sogenannte Wellengleichung

$$\frac{\partial^2 u}{\partial x^2}(t, x) = \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2}(t, x) \quad (1)$$

für alle $x \in (0, 1)$ und $t \in \mathbb{R}$ beschrieben, wobei c die Ausbreitungsgeschwindigkeit der Welle ist. Wenn die Saite an beiden Enden fest eingespannt ist, so gelten die Randbedingungen

$$u(t, 0) = u(t, 1) = 0$$

für alle $t \in \mathbb{R}$.

Zur Berechnung der Eigenschwingungen suchen wir nach Lösungen u , die in der Zeit harmonisch schwingen. Solche erfüllen folgenden Ansatz

$$u(x, t) = \Re(v(x)e^{-i\omega t})$$

mit einer festen, aber unbekannten Kreisfrequenz $\omega > 0$ und einer Funktion v , welche nur noch vom Ort x abhängt. Durch Einsetzen erhalten wir für v die sogenannte Helmholtz-Gleichung

$$-v''(x) = \kappa^2 v(x), \quad x \in (0, 1), \quad (2)$$

mit der unbekannten Wellenzahl $\kappa := \frac{\omega}{c}$ und den Randbedingungen

$$v(0) = v(1) = 0. \quad (3)$$

1.2 Analytische Lösung

$$v_\kappa(x) = C_1 \cos(\kappa x) + C_2 \sin(\kappa x), \quad x \in [0, 1], \quad (4)$$

mit beliebigen Konstanten C_1, C_2 löst die Helmholtz-Gleichung (2). Das erkennt man durch stumpfes Einsetzen.

$$\begin{aligned} -v_\kappa''(x) &= -\frac{\partial^2}{\partial x^2}(C_1 \cos(\kappa x) + C_2 \sin(\kappa x)) = -\frac{\partial}{\partial x}(-C_1 \kappa \sin(\kappa x) + C_2 \kappa \cos(\kappa x)) \\ &= -(-C_1 \kappa^2 \cos(\kappa x) - C_2 \kappa^2 \sin(\kappa x)) = \kappa(C_1 \cos(\kappa x) + C_2 \sin(\kappa x)) = \kappa^2 v_\kappa(x) \end{aligned}$$

Wir fragen uns, für welche $\kappa > 0$, Konstanten C_1 und C_2 existieren, sodass v_κ auch die Randbedingungen (3) erfüllt.

$$0 \stackrel{!}{=} \begin{cases} v_\kappa(0) = C_1 \cos 0 + C_2 \sin 0 = C_1 \\ v_\kappa(1) = C_1 \cos \kappa + C_2 \sin \kappa = C_2 \sin \kappa \end{cases}$$

Nachdem $\cos 0 = 1$ und $\sin 0 = 0$, erhält man, aus der oberen Gleichung, $C_1 = 0$. Mit der unteren Gleichung folgt aber auch $C_2 \sin \kappa = 0$. Wenn nun auch $C_2 = 0$, dann erhielte man die triviale Lösung $v_\kappa = 0$. Für eine realistischere Modellierung, d.h. $v_\kappa \neq 0$, müsste $\sin \kappa = 0$, also $\kappa \in \pi\mathbb{Z}$.

Das sind die gesuchten $\kappa > 0$. Sei nun eines dieser κ fest. Offensichtlich ist $C_1 = 0$ eindeutig, $C_2 \in \mathbb{R}$ jedoch beliebig.

1.3 Numerische Approximation

Häufig lassen sich solche Probleme nicht analytisch lösen, sodass auf numerische Verfahren zurückgegriffen wird, welche möglichst gute Näherungen an die exakten Lösungen berechnen sollen. Als einfachstes Mittel dienen sogenannte Differenzenverfahren. Sei dazu $x_j := jh$, $j = 0, \dots, n$ eine Zerlegung des Intervalls $[0, 1]$ mit äquidistanter Schrittweite $h = 1/n$. Die zweite Ableitung in (2) wird approximiert durch den Differenzenquotienten

$$v''(x_j) \approx D_h v(x_j) := \frac{1}{h^2}(v(x_{j-1}) - 2v(x_j) + v(x_{j+1})), \quad j = 1, \dots, n-1. \quad (5)$$

Es sei angemerkt, dass (??) tatsächlich einen Differenzenquotienten beschreibt. Um das einzusehen, verwenden wir den links- und rechts-seitigen Differenzenquotient erster Ordnung, sowie $x_{j-1} = x_j - h$, $x_{j+1} = x_j + h$. Wir erhalten $\forall j = 1, \dots, n-1$:

$$\begin{aligned} v''(x_j) &= \lim_{h \rightarrow 0} \frac{1}{h}(v'(x_j + h) - v'(x_j)) \\ &= \lim_{h \rightarrow 0} \frac{1}{h} \left(\frac{1}{h}(v(x_j + h) - v(x_j)) - \frac{1}{h}(v(x_j) - v(x_j - h)) \right) \\ &= \lim_{h \rightarrow 0} \frac{1}{h^2}(v(x_j + h) - 2v(x_j) + v(x_j - h)) \\ &= \lim_{h \rightarrow 0} D_h v(x_j) \end{aligned}$$

1.3.1 Approximationsfehler

Für hinreichend glatte Funktionen v mit einer geeigneten Konstanten $C > 0$ wird der Approximationsfehler quadratisch in h klein, d.h. dass

$$|v''(x_j) - D_h v(x_j)| \leq Ch^2. \quad (6)$$

Nachdem v hinreichend glatt ist, gilt nach dem Satz von Taylor, dass $\forall j = 1, \dots, n-1$:

$$\begin{aligned} v(x_j + h) &= \sum_{\ell=0}^{n+2} \frac{h^\ell}{\ell!} v^{(\ell)}(x_j) + \mathcal{O}(h^{n+3}), \\ v(x_j - h) &= \sum_{\ell=0}^{n+2} \frac{(-h)^\ell}{\ell!} v^{(\ell)}(x_j) + \mathcal{O}(h^{n+3}). \end{aligned}$$

Man beachte, dass sich die ungeraden Summanden, der oberen Taylor-Polynome, gegenseitig aufheben. Damit erhalten wir für den Differenzenquotient $D_h v(x_j)$, $j = 1, \dots, n-1$ eine asymptotische Entwicklung.

$$\begin{aligned} D_h v(x_j) &= \frac{1}{h^2} (v(x_j - h) + v(x_j + h) - 2v(x_j)) \\ &= \frac{1}{h^2} \left(2v(x_j) + h^2 v''(x_j) + \sum_{\substack{\ell=4 \\ \ell \in 2\mathbb{N}}}^{n+2} \frac{h^\ell}{\ell!} v^{(\ell)}(x_j) (1 + (-1)^\ell) - 2v(x_j) \right) + \mathcal{O}(h^{n+3}) \\ &= v''(x_j) + 2 \sum_{\ell=1}^{\lfloor \frac{n}{2} \rfloor} \frac{h^{2\ell}}{(2\ell+2)!} v^{(2\ell)}(x_j) + \mathcal{O}(h^{n+1}) \end{aligned}$$

Daraus folgt unmittelbar die quadratische Konvergenz (6), d.h. $\forall j = 1, \dots, n-1$:

$$D_h v(x_j) - v''(x_j) = \mathcal{O}(h^2), \quad h \rightarrow 0.$$

1.3.2 Eigenwertproblem

Wir wollen nun den Differenzenquotienten $D_h v(x_j)$ verwenden, um ein Eigenwertproblem der Form $A\vec{v} = \lambda\vec{v}$ mit einer Matrix $A \in \mathbb{R}^{(n-1) \times (n-1)}$ zu dem Eigenvektor $\vec{v} := (v(x_1), \dots, v(x_{n-1}))^T$ und dem Eigenwert $\lambda := \kappa^2$ herzuleiten. Das soll die Helmholtz-Gleichung (2), mit Tupeln und Eigenwerten für die Funktionen v_κ bzw. Vorfaktoren κ , approximieren.

Es wird eine Matrix $-A_n$, $n \geq 2$ gesucht, die den Differenzenquotienten $D_h v(x_j)$ auf den oberen Vektor $\vec{v}^{(n)}$ komponentenweise anwendet. Wir rufen in Erinnerung, dass $h = 1/n$ und definieren die naheliegende Matrix

$$-A_n := \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}.$$

Wenn nun die Randbedingungen (3) gelten sollen, d.h. $v(x_0), v(x_n) = 0$, leistet diese Matrix A_n tatsächlich das Gewünschte. Sie approximiert die linke Seite der Helmholtz-Gleichung (2).

$$A_n \vec{v}^{(n)} = -\frac{1}{h^2} \begin{pmatrix} v(x_0) - 2v(x_1) + v(x_2) \\ v(x_1) - 2v(x_2) + v(x_3) \\ \vdots \\ v(x_{n-3}) - 2v(x_{n-2}) + v(x_{n-1}) \\ v(x_{n-2}) - 2v(x_{n-1}) + v(x_{n-0}) \end{pmatrix} = - \begin{pmatrix} D_h v(x_1) \\ \vdots \\ D_h v(x_{n-1}) \end{pmatrix} \xrightarrow{n \rightarrow \infty} -v''$$

Die Matrix A_n wurde in der Funktion `my_numpy_matrix` implementiert.

```
1 def my_numpy_matrix(n):
2
3     assert n >= 2
4
5     h = 1/n
6
7     a = -2 * np.ones(n-1)
8     b = np.ones(n-2)
9
10    A = np.diag(b, -1) + np.diag(a, 0) + np.diag(b, 1)
11    A = -A/h**2
12
13    return A
```

Das Eigenwertproblem wurde mit `np.linalg.eig`, für beliebige $n \geq 2$, gelöst. `np.linalg.eig` retourniert die Eigenwerte und Eigenvektoren nicht zwangsläufig, der Größe der Eigenwerte nach, sortiert. Nachdem der Zusammenhang zwischen Eigenwert und Eigenvektor nicht verloren gehen soll, erfordert dies einigen logistischen Aufwand. Das ist aber nicht wesentlich und wird daher nicht weiter erklärt. Wir vergleichen lieber die Eigenwerte und Eigenvektoren mit den analytischen Ergebnissen.

Betrachtet man die, unten aufgelisteten, Eigenwerte, der ersten paar Matrizen A_2, \dots, A_{10} , so legen diese ein gewisses, laut (6) vielleicht sogar quadratisches, Konvergenzverhalten nahe.

n = 2:	-----	9.705050945562961
-----		36.89799941784412
	Eigen-Werte:	76.19294847228122
Eigen-Werte:	9.549150281252611	119.80705152771888
[8.]	34.54915028125264	159.10200058215582
	65.45084971874735	186.29494905443664
n = 3:	90.45084971874735	
-----		n = 8:
	n = 6:	-----
Eigen-Werte:	-----	Eigen-Werte:
9.0		9.743419838555344
27.0	Eigen-Werte:	37.49033200812192
	9.646170927520402	79.01652065726852
n = 4:	35.99999999999999	127.99999999999999
-----	71.99999999999997	176.98347934273144
	108.00000000000001	218.50966799187793
Eigen-Werte:	134.35382907247953	246.25658016144442
9.372583002030478		
31.999999999999996	n = 7:	
54.62741699796946	-----	n = 9:

n = 5:	Eigen-Werte:	

Eigen-Werte:	Eigen-Werte:	n -> inf:
9.769795432682793	9.788696740969272	-----
37.90080021472559	38.19660112501045	
80.99999999999997	82.44294954150533	(1 * pi)^2 = 9.869604401089358
133.8689952179573	138.1966011250105	(2 * pi)^2 = 39.47841760435743
190.13100478204277	200.00000000000006	(3 * pi)^2 = 88.82643960980423
243.00000000000014	261.80339887498934	(4 * pi)^2 = 157.91367041742973
286.09919978527444	317.5570504584944	(5 * pi)^2 = 246.74011002723395
314.2302045673173	361.8033988749895	(6 * pi)^2 = 355.3057584392169
	390.2113032590302	(7 * pi)^2 = 483.61061565337855
n = 10:	.	(8 * pi)^2 = 631.6546816697189
-----	.	(9 * pi)^2 = 799.437956488238

Die Matrix A_n besitzt also scheinbar $n - 1$ paarweise verschiedene Eigenwerte $\lambda_{1,n} < \dots < \lambda_{n-1,n}$, welche jeweils gegen $\lambda_i := (i\pi)^2$, $i \in \mathbb{N}$ konvergieren. Nachdem $\{\sqrt{\lambda_i} : i \in \mathbb{N}_0\} = \pi\mathbb{N}_0 \subsetneq \pi\mathbb{Z}$, könnte man sich nun fragen, wo die andere Hälfte der analytischen Ergebnisse steckt. Die Erklärung ist ganz unspektakulär $\lambda := (\pm\kappa)^2$.

$$\begin{array}{ccccccc}
\lambda_{1,2} & \rightarrow & \lambda_{1,3} & \rightarrow & \dots & \rightarrow & \lambda_{1,n} \xrightarrow{n \rightarrow \infty} \lambda_1 = (1\pi)^2 \\
& & \lambda_{2,3} & \rightarrow & \dots & \rightarrow & \lambda_{2,n} \xrightarrow{n \rightarrow \infty} \lambda_2 = (2\pi)^2 \\
& & & & \ddots & & \vdots \\
& & & & & & \lambda_{i,n} \xrightarrow{n \rightarrow \infty} \lambda_i = (i\pi)^2
\end{array}$$

Wir bezeichnen mit $\epsilon_i(n) := |\lambda_i - \lambda_{i,n}|$, $i = 1, \dots, n - 1$ den absoluten Konvergenz-Fehler des i -ten Eigenwertes. In der folgenden Abbildung 1 wurde ϵ_i für $j = 1, 2, 3$ gegen id^2 , doppelt logarithmisch, geplottet.

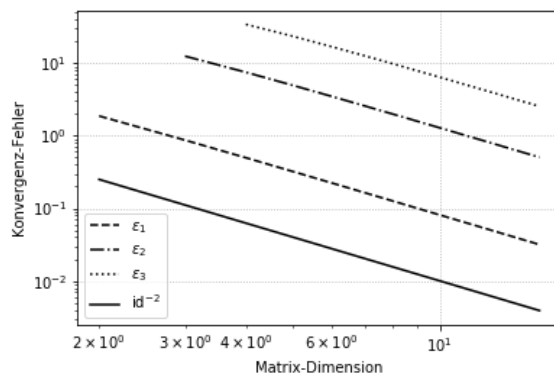
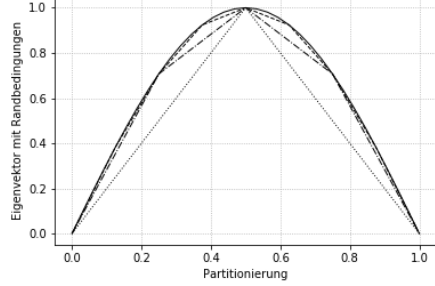


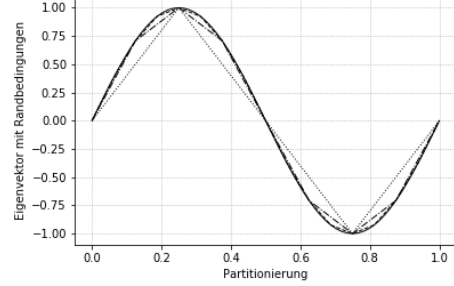
Abbildung 1: Konvergenz-Fehler der Eigenwerte von A_n

Allem Anschein nach, verschwindet ϵ_i quadratisch. Das korreliert mit dem Ergebnis (6). Man beachte, dass der i -te Eigenwert erst ab einer Matrix A_n , $n > i$ existiert. Daher fangen die plots von ϵ_i desto später an, je größer i ist. Für größeres i ist auch der initiale Fehler größer. Obwohl dieser ebenfalls quadratisch konvergiert, werden mehr Rechenoperationen für ein genaues Ergebnis benötigt.

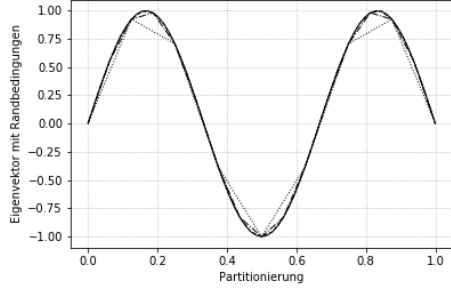
Seien $\vec{v}^{(1,n)}, \dots, \vec{v}^{(n-1,n)}$ die Eigenvektoren (eigentlich Eigenräume), zu den Eigenwerten $\lambda_{1,n} < \dots < \lambda_{n-1,n}$, der Matrix A_n . Diese sollten nun gegen die Funktionen v_{κ_i} , $\kappa_i = \sqrt{\lambda_i}$, vielleicht sogar quadratisch, konvergieren. Folgende Abbildungen sollen dies veranschaulichen.



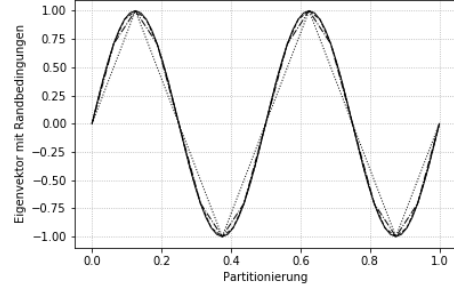
(a) Eigenvektoren $v^{(1,n)}$, $n = 2, 4, 8, 32$



(b) Eigenvektoren $v^{(2,n)}$, $n = 4, 8, 16, 64$



(c) Eigenvektoren $v^{(3,n)}$, $n = 8, 16, 32, 64$



(d) Eigenvektoren $v^{(4,n)}$, $n = 8, 16, 32, 64$

Abbildung 2: Eigenvektoren $v^{(i,n)}$, $i = 1, \dots, 4$ der Matrizen A_n

Erstaunlicherweise, gibt es keinen Konvergenz-Fehler, da die Eigenvektoren scheinbar direkt an den Grenzfunktionen liegen. Mit anderen Worten, $\forall n \in \mathbb{N}, \forall i, j = 1, \dots, n-1$:

$$\vec{v}_j^{(i,n)} = v_{\kappa_i}(x_j).$$

Gemeinsam mit der Polynom-Interpolation, liefert das eine hervorragende Approximation. Dies wird hier jedoch nicht weiter behandelt.

1.4 verallgemeinerte Eigenschwingungen

Die Ausbreitungsgeschwindigkeit c in (1) hängt vom Material der Saite ab. Bisher haben wir sie als konstant angenommen, d.h. die Saite bestand aus einem Material. Sei nun für $c_0, c_1 \in \mathbb{R}$

$$c(x) := \begin{cases} c_0, & x \in (0, 1/2) \\ c_1, & x \in (1/2, 1) \end{cases}. \quad (7)$$

Zuerst leiten wir eine zur Helmholtz-Gleichung (2) ähnliche Gleichung her und geben einen (4) entsprechenden Lösungsansatz an, wenn die Lösung v auf $(0, 1)$ stetig differenzierbar sein soll. Dabei betrachten wir eine angepasste Version der Wellengleichung (1).

$$\frac{\partial^2 u}{\partial x^2}(t, x) = \frac{1}{c^2(x)} \frac{\partial^2 u}{\partial t^2}(t, x), \quad x \in (0, 1), \quad t \in \mathbb{R}$$

Wir verwenden jedoch den selben Ansatz, wie Vorher. Das war $u(x, t) = \Re(v(x)e^{-i\omega t})$, mit einer festen, aber unbekannten Kreisfrequenz $\omega > 0$ und einer Funktion v , welche nur noch vom Ort x abhängt.

Einsetzen und analoges Nachrechnen, gibt mit der unbekannten Wellenzahl $\kappa(x) := \frac{\omega}{c(x)}$, die Randbedingungen (3) und

$$-v''(x) = \kappa^2(x)v(x), \quad x \in (0, 1).$$

Um Probleme mit der Differenzierbarkeit von κ zu vermeiden, führen wir die Abkürzungen $\kappa_0 := \frac{\omega}{c_0}$, $\kappa_1 := \frac{\omega}{c_1}$ ein. Wir definieren den Lösungsansatz durch Fallunterscheidung und mit (vorerst) beliebigen Konstanten $C_{01}, C_{02}, C_{11}, C_{12}$.

$$v(x) := \begin{cases} C_{01} \cos(\kappa_0 x) + C_{02} \sin(\kappa_0 x), & x \in (0, 1/2) \\ C_{11} \cos(\kappa_1 x) + C_{12} \sin(\kappa_1 x), & x \in (1/2, 1) \end{cases}$$

Durch Berücksichtigung der Randbedingungen (3), erhält man (fast analog zu Vorher)

$$C_{01} = 0, \quad C_{11} \cos \kappa_1 + C_{12} \sin \kappa_1 = 0.$$

Soll v auf $1/2$ stetig fortgesetzt werden, so müssen dessen links- und rechts-seitiger Grenzwert übereinstimmen.

$$C_{02} \sin(\kappa_0/2) = \lim_{x \rightarrow 1/2-} v(x) = \lim_{x \rightarrow 1/2+} v(x) = C_{11} \cos(\kappa_1/2) + C_{12} \sin(\kappa_1/2)$$

Um stetige Differenzierbarkeit zu erhalten, muss auch die Ableitung

$$v'(x) = \begin{cases} C_{02} \kappa_0 \cos(\kappa_0 x), & x \in (0, 1/2) \\ -C_{11} \kappa_1 \sin(\kappa_1 x) + C_{12} \kappa_1 \cos(\kappa_1 x), & x \in (1/2, 1) \end{cases}$$

auf $1/2$ stetig fortgesetzt werden.

$$\kappa_0 C_{02} \cos(\kappa_0/2) = \lim_{x \rightarrow 1/2-} v'(x) = \lim_{x \rightarrow 1/2+} v'(x) = -C_{11} \kappa_1 \sin(\kappa_1/2) + C_{12} \kappa_1 \cos(\kappa_1/2)$$

Aus den Randbedingungen und stetigen Fortsetzungen, ergibt sich also das homogene lineare Gleichungssystem $R\vec{C} = 0$, mit

$$R := \begin{pmatrix} \sin(\kappa_0/2) & -\cos(\kappa_1/2) & -\sin(\kappa_1/2) \\ \kappa_0 \cos(\kappa_0/2) & \kappa_1 \sin(\kappa_1/2) & -\kappa_1 \cos(\kappa_1/2) \\ 0 & \cos \kappa_1 & \sin \kappa_1 \end{pmatrix} \in \mathbb{R}^{3 \times 3}, \quad \vec{C} := \begin{pmatrix} C_{02} \\ C_{11} \\ C_{12} \end{pmatrix} \in \mathbb{R}^{3 \times 1}.$$

Sei $R \in \text{GL}_3(\mathbb{R})$ regulär, so ist deren Kern trivial, d.h. $\ker R = \{0\}$, und somit auch die Lösung $\vec{C} = 0$. Dieser Fall wäre jedoch wieder der trivialfall, darum betrachten wir den anderen Fall, also dass $\det R = 0$ gilt. Mit SymPy berechnet man

$$\det R = \sin\left(\frac{\kappa_0}{2}\right) \cos\left(\frac{\kappa_1}{2}\right) \kappa_1 + \sin\left(\frac{\kappa_1}{2}\right) \cos\left(\frac{\kappa_0}{2}\right) \kappa_0.$$

Über $\kappa_0 = \frac{\omega}{c_0}$, $\kappa_1 = \frac{\omega}{c_1}$ lässt sich ω , für gegebene c_0, c_1 als (nicht eindeutige) Nullstelle einer Funktion f charakterisieren. Diese Überlegung, wird, in Form von folgender Funktion, implementiert.

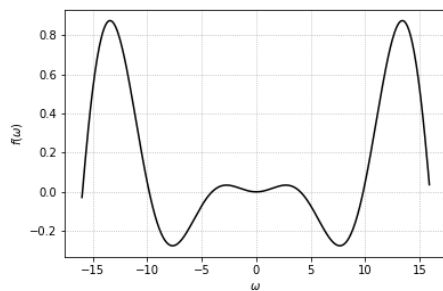
```

1 # c ... pair of propagation speeds
2
3 def get_zero_function(c):
4
5     # allocate some sympy symbols:
6     omega = sp.Symbol('\omega')
7     kappa = sp.IndexedBase('\kappa')
8
9     # implement the matrix R (properly):
10    R = sp.Matrix([[ sp.sin(kappa[0]/2),  kappa[0]*sp.cos(kappa[0]/2), 0],
11                  [-sp.cos(kappa[1]/2),  kappa[1]*sp.sin(kappa[1]/2), sp.cos(kappa[1])],
12                  [-sp.sin(kappa[1]/2), -kappa[1]*sp.cos(kappa[1]/2), sp.sin(kappa[1])]])
13    R = R.T
14
15    # calculate R's determinant (properly):
16    det = sp.det(R)
17    det = sp.simplify(det)
18
19    # substitute for kappa_0 and kappa_1:
20    kappa_0 = omega/c[0]
21    kappa_1 = omega/c[1]
22    substitution = {kappa[0]: kappa_0, kappa[1]: kappa_1}
23    det = det.subs(substitution)
24
25    # transform expression det into proper numpy function:
26    zero_function = sp.lambdify(omega, det, 'numpy')
27
28    return zero_function

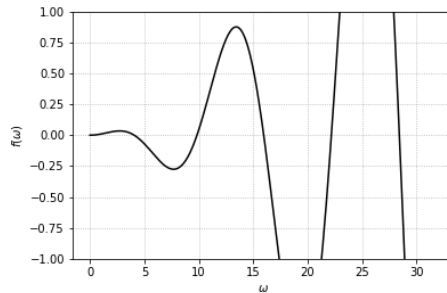
```

Nun können wir f für fixe c_0, c_1 plotten lassen. Dann bekommen wir ein besseres Verständnis dafür, welchen Startwert wir wählen sollen, um die Gleichung $f(\omega) = 0$, mit `fsolve` aus `scipy.optimize`, lösen zu lassen.

Für die folgenden Plots in Abbildung 3, wurden die arbiträren Werte $c_0 = 100$, $c_1 = 1$ gewählt. Die Funktion f ist scheinbar gerade, d.h. symmetrisch bzgl. der y -Achse.



(a) auf dem Intervall $(-16, 16)$



(b) auf dem Intervall $(0, 64)$ und herangezoomt

Abbildung 3: Plots von f für $c_0 = 100$, $c_1 = 1$

Dementsprechend, können passende Startwerte für iterative Verfahren gewählt werden. Die Ergebnisse vom,

bereits erwähnten, `fsolve` folgen. Warum die Quadrate dieser Ergebnisse eine Rolle spielen, wird später noch erwähnt.

Wir wollen nun den Differenzenquotienten $D_h v(x_j)$, um ein verallgemeinertes Eigenwertproblem der Form $A\vec{v} = \lambda B\vec{v}$ mit Matrizen $A, B \in \mathbb{R}^{(n-1) \times (n-1)}$ herzuleiten.

Sei abermals $x_j := jh$, $j = 0, \dots, n$ unsere Zerlegung des Intervalls $[0, 1]$ mit äquidistanter Schrittweite $h = 1/n$. Die Matrix A_n , für den Differenzenquotienten $D_h v(x_j)$ und der Vektor $\vec{v} := (v(x_1), \dots, v(x_{n-1}))^T$, bleiben ebenfalls nach wie vor so, wie sie waren.

$B_n \lambda$ soll nun, analog zu Vorher, $-\kappa^2$ repräsentieren. Diesmal jedoch, ist κ als Funktion zu verstehen. Also wird die Matrix B_n deren Fallunterscheidungen übernehmen und λ bleibt konstant.

$$B_n := \text{diag}^{-2}(\underbrace{c_0, \dots, c_0}_{\lfloor \frac{n-1}{2} \rfloor\text{-mal}}, \underbrace{c_1, \dots, c_1}_{\lceil \frac{n-1}{2} \rceil\text{-mal}}), \quad \lambda := -\omega^2.$$

Die Wahl von B_n lässt sich wie folgt begründen: Für zwei Vektoren a, b , ist die Matrix-Vektor-Multiplikation \cdot , mit der Diagonalmatrix von a , äquivalent zur komponentenweisen Multiplikation \odot , d.h. $\text{diag}(a) \cdot b = a \odot b$. Bei dem vorherigen Eigenwertproblem wäre B_n als die Einheits-Matrix I_n zu betrachten, nachdem der Eigenwert λ gleich ganz $-\kappa^2$ (konstant) übernehmen konnte. Da $-\kappa^2$ nun zwei Werte

$$-\left(\frac{\omega}{c_0}\right)^2, -\left(\frac{\omega}{c_1}\right)^2$$

annimmt, muss diese Eigenschaft von Diagonalmatrizen ausgenutzt werden, um die Unterscheidung zwischen $x_j < 1/2$ und $x_j \geq 1/2$ zu realisieren.

Wenn wir davon ausgehen, dass $c_0, c_1 \neq 0$, so ist B_n wohldefiniert als Diagonalmatrix deren quadratische Kehrwerte. Damit lässt sich dieses verallgemeinerte Eigenwertproblem $A_n \vec{v} = \lambda B_n \vec{v}$, mit nahezu keinem Aufwand, in ein konkretes $B_n^{-1} A_n \vec{v} = \lambda \vec{v}$ umformulieren.

Dieses Eigenwertproblem wurde ebenfalls mit `np.linalg.eig`, für beliebige $n \geq 2$, gelöst. Wir vergleichen die Eigenwerte und Eigenvektoren mit den oberen semi-analytischen Ergebnissen mit $c_0 = 100$, $c_1 = 1$. Dass die Eigenwerte gegen $-\omega^2$ konvergieren, überrascht wenig.

```
n = 1000
-----
Eigenvalues:
-16.44054324040394
-96.3725475394107
-254.1214683324469
```