ETHZ, D-MATH

Prüfung

Numerische Methoden D-PHYS, FS 2019

Dr. V. Gradinaru

13.08.2019

Prüfungsdauer : 180 Minuten.

1. *(8 Punkte) Polynomfit und die Runge Funktion*

Die Runge Funktion ist definiert als:

$$f(x) := \frac{1}{1+x^2}\,. \tag{1}$$

Wir wollen diese Funktion auf dem Interval $[-5,5]$ mit einem Polynom $P_n(x)$ von Grad n approximieren. Dazu schreiben wir ein lineares Ausgleichsproblem mit m gleichmässig in $[-5,5]$ verteilten Punkten $\{(x_i, f(x_i))\}_{i=1}^m$ wie folgt:

$$\mathbf{Ac} = \mathbf{b} \tag{2}$$

wobei $\mathbf{c}$ die $n+1$ Koeffizienten des Polynoms P_n sind.

a) Schreiben Sie auf Papier die Definition der Einträge der Matrix $\mathbf{A}$ und der rechten Seite $\mathbf{b}$.

b) Für Polynome mit Grad $2 \leq n \leq 14$ und jeweils $m = 20$ und $m = 40$:

bi) Plotten Sie die Konditionszahl von $\mathbf{A}$ und $\mathbf{A}^T\mathbf{A}$ in linlog-Skala im selben Bild;

bii) Lösen Sie das lineare Ausgleichsproblem und plotten Sie diese Lösungen. *Hinweis*: Die Funktionen `scipy.linalg.lstsq` und `numpy.polyval` können nützlich sein.

Bitte wenden!

2. *(8. Punkte) Potenzmethoden*

Im Template `PowerMethod.py` ist eine Matix $\mathbf{A}$ definiert. Die Eigenwerte von $\mathbf{A}$ sind im Folgenden so geordnet:

$$\lambda_1 < \lambda_2 < \ldots < \lambda_{n-1} < \lambda_n.$$

Verwenden Sie die Potenzmethoden, um die folgende Eigenwerte zu berechnen.

a) λ_n, den größten Eigenwert von $\mathbf{A}$;

b) λ_1, den kleinsten Eigenwert von $\mathbf{A}$;
Hinweis: Die Funktionen `linalg.lu` und `linalg.solve_triangular` in `scipy` können nützlich sein.

c) λ_α, den Eigenwert von $\mathbf{A}$, welcher am nächten an $\alpha = 45$ liegt.

Siehe nächstes Blatt!

3. *(8 Punkte) Nichtlineares System*

Das nichtlineare Gleichungssystem

$$F = \begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} \sin\left(\pi\sqrt{x^2+4y^2}\right) \\ x^2 + y - 4\sin^2(\pi x) + 1 \end{pmatrix} = \mathbf{0} \tag{3}$$

ist zu lösen.

a) Implementieren Sie im File `Nullstellensuche.py` die Funktion F, deren Nullstellen wir suchen, und die Ableitung DF, so dass diese mit `numpy.array` arbeiten können;

b) Implementieren Sie ein Newton-Verfahren in $\mathbb{R}^2$ in `newton(x0, F, DF, rtol, maxiter)` in `Nullstellensuche.py`, wobei

- `x0` der Initialwert,
- `F` die Funktion F,
- `DF` die Ableitung DF,
- `rtol` die relative Toleranz,
- `maxiter` die maximale Anzahl Iterationen,

sind. Die Funktion `newton` soll die approximierte Lösung und die Anzahl benötigten Iterationen zurückgeben.
Hinweis: Die Funktion `np.linalg.solve` kann nützlich sein.

Bitte wenden!

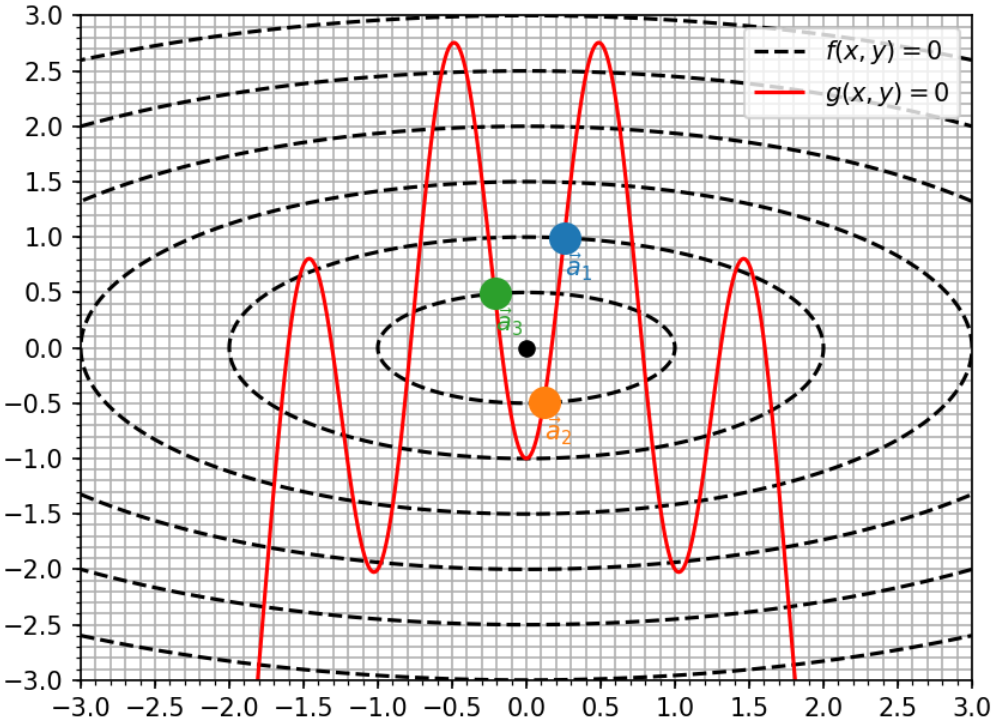

Abbildung 1 – Die Nullstellen von f und g.

c) Schauen Sie sich die Figur 1 an. Hier sind drei Nullstellen eingezeichnet: betrachte a_2.

(i) Finden Sie diese mit der Funktion `fsolve` aus `scipy.optimize`;

(ii) Finden Sie einen Startwert x_0^2, so dass das Newton-Verfahren mit Initialwert x_0^2 in maximal 100 Schritten mit relativer Toleranz tol $= 10^{-12}$ zur eingezeichneter Nullstelle a_2 konvergiert;

(iii) Für solchen Startwert x_0^2 geben Sie a_2^{fsolve}, a_2^{newton}, $F(a_2^{\text{newton}})$ und die verwendete Anzhal Newton-Schritte N_{it} in folgender Form aus

`Start` : x_0^2
`fsolve` : a_2^{fsolve}
`Newton` : a_2^{newton}
`F` : $F(a_2^{\text{newton}})$
`Iterationen` : N_{it}
`-----------------------------------`

Hinweis: zur Verfügung steht neben das Template `Nullstellensuche.py` auch das File `plot_contour.py`, das die Abbildung 1 generiert und nicht modifiziert werden muss.

Siehe nächstes Blatt!

4. *(8 Punkte): Federpendel*

Die Differentialgleichung für den Auslenkungswinkel θ des invertierten Federpendels im Bild ist

$$\ddot{\theta} = -\frac{c}{mL^2}\theta + \frac{g}{L}\sin\theta\,. \qquad (4)$$

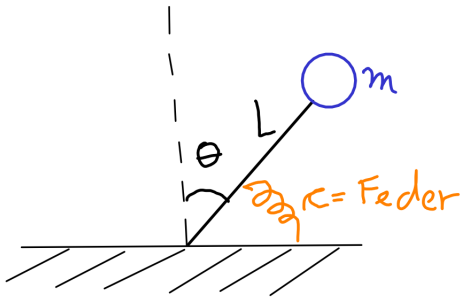

Abbildung 2 – Federpendel.

Die physikalischen Konstanten sind im File angegeben. Die Endzeit ist im Folgenden $T = 1000$.

a) Schreiben Sie die Gleichung als Hamilton-System und geben Sie die Hamilton Funktion an.

b) Benutzen Sie das Hamilton-System, um die Gleichung mit $N = 5000$ Zeitschritten des Störmer-Verlet Verfahrens zu lösen.
Hinweis: das Strömer-Verlet Verfahren kann auch als Splitting Verfahren verstanden werden. Damit können die implementierte Funktionen `Approximate`, `splitting`, `splitting_step` und `integrate` nützlich sein.

c) Benutzen Sie das Hamilton-System, um die Gleichung mit $N = 2500$ Zeitschritten eines symplektischen partitionierten Runge-Kutta (spRK) Verfahren der Ordnung 6 zu lösen.

d) Plotten Sie in linlog-Skala die Abweichungen der Totalenergie für die Approximationen aus (b), (c) und (d). Erklären Sie das beobachtete Verhalten.

Hinweis: Die Funktion `semilogy` unter `matplotlib.pyplot` kann nützlich sein.

e) Schreiben Sie welcher dieser drei Lösungen Sie am meisten und welcher Sie am wenigsten trauen und begründen Sie Ihre Wahl.

Bitte wenden!

5. *(8 Punkte): Modifizierte Euler Verfahren*

a) Begründen Sie das modifizierte explizite Euler-Verfahren

$$\begin{aligned} w &= y_n + \frac{h}{2} f(t_n, y_n) \\ y_{n+1} &= y_n + h f(t_n + \frac{h}{2}, w) \end{aligned} \tag{5}$$

und das modifizierte implizite Euler-Verfahren

$$\begin{aligned} w &= y_n + \frac{h}{2} f(t_n + \frac{h}{2}, w) \\ y_{n+1} &= y_n + h f(t_n + \frac{h}{2}, w) \end{aligned} \tag{6}$$

als Polygonzugverfahren zur numerischen Lösung einer gewöhnlicher Differentialgleichung

$$\dot{y} = f(t, y)\,.$$

b) Beweisen Sie, dass das modifizierte implizite Euler-Verfahren identisch mit der impliziten Mittelpunktsregel ist.

c) Finden Sie die Stabilitätsfunktionen der beiden modifizierten Euler-Verfahren.

d) Bestimmen Sie empirisch die Konvergenzrate der beiden Euler-Verfahren indem Sie die Differentialgleichung

$$\dot{y} = -\sin(t) + \lambda(y - \cos(t))\,, \qquad t \in [0, 2] \tag{7}$$

mit dem Startwert $y(0) = 1$ bis zuer Endzeit $T = 2$ für $\lambda = -10$ lösen. Die exakte Lösung ist $y(t) = \cos(t)$.
Hinweis: Sie können die Funktion `compute_error` im File `Modified_Euler.py` benutzen, um den Fehler zu berechnen.

e) Betrachten Sie die mit dem modifizierten expliziten und modifizierten impliziten Euler Verfahren berechneten Lösungen dieser Gleichung für $\lambda = -2100$ mit N äquidistanten Schritten bis zur Endzeit $T = 2$.

(i) Plotten Sie die zwei Lösungen für $N = 1000$ und erklären Sie das beobachtete Verhalten,

(ii) Ab welchen N beobachten Sie einen Fehler zur Endzeit $T = 2$ kleiner als 10^{-4} für beide Verfahren?

Siehe nächstes Blatt!

6. *(8 Punkte): Quadraturformeln*

Sei die Quadraturformel auf $[-1, 1]$ gegeben durch die Knoten und Gewichte im File `Quadrature.py`.

a) Welche Ordnung hat diese Quadraturformel? Schreiben Sie ein Programm, mit dem Sie Ihre Antwort begründen.

b) Zu welcher Klasse von Quadraturformeln gehört diese Quadraturformel? Begründen Sie Ihre Aussage.

c) Schreiben Sie ein Programm, das eine auf diesen Knoten und Gewichten basierte zusammengestetzte Quadraturformel berechnet. Wenden Sie dies für die Funktionen

$$f_1(x) = \frac{1}{1+5x^2}, \qquad f_2(x) = \sqrt{x} \tag{8}$$

auf dem Interval $[0, 1]$. Plotten Sie die Fehler in doppellogarithmischer Skala. Die exakten Werte der zwei Integrale sind:

$$\int_0^1 f_1(x)\,dx = \frac{\arctan(\sqrt{5})}{\sqrt{5}}, \qquad \int_0^1 f_2(x)\,dx = \frac{2}{3}. \tag{9}$$

Welche Ordnungen beobachten Sie für die zwei Fehler? Plotten Sie auf dem Bild mit den Fehlern die Graphen der Polynome, die diesen Ordnungen entsprechen.

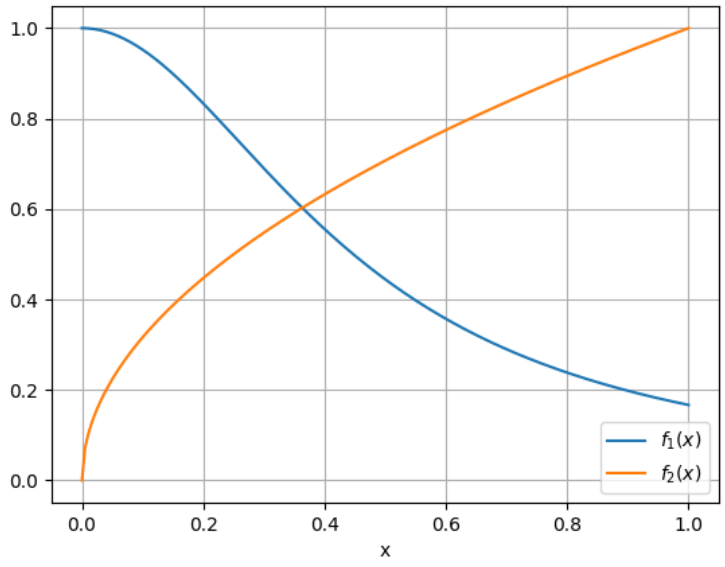

Abbildung 3 – Funktionen f_1 und f_2.

d) Begründen Sie auf Papier das beobachtete oder erwartete Verhalten der Fehler in den zwei Fälle.