Serie 10

Abgabedatum: Di. 18.05. / Mi. 19.05, in den Übungsgruppen

Koordinatoren: Bei Fragen zu den Übungen kontaktieren Sie bitte

- Francesca Bartolucci `francesca.bartolucci@sam.math.ethz.ch`
- Luc Grosheintz `luc.grosheintz@sam.math.ethz.ch`

Webpage: http://metaphor.ethz.ch/x/2021/fs/401-1662-10L

1. Kernaufgabe: Adaptive Methoden für steife Systeme

Aufgabenstellung

a) Implementieren Sie die Rosenbrock-Wanner Methoden der Ordnung 2 und 3. Es sollen Funktionen `row_2_step(f, Jf, yi, h)` und `row_3_step(f, Jf, yi, h)` geschrieben werden, die ausgehend vom Wert $y_i(t_i)$ genau einen Zeitschritt h der entsprechenden Methode berechnen und die Propagierte $y_{i+1}(t_i + h)$ zurück geben.

Hinweis: Die Parameter sind im Template `stiff_row_Template.py` erklärt.

b) Lösen Sie die logistische Differentialgleichung:

$$\dot{y}(t) = \lambda y(t)(1 - y(t))$$

mit dem Anfangswert $y(0) = c = 0.01$ und $\lambda = 25$ bis zum Zeitpunkt $T = 2$. Benutzen Sie $N = 100$ Zeitschritte. Plotten Sie die numerischen Lösungen $y(t)_{\texttt{ROW}}$ sowie die Fehler $y(t)_{\texttt{ROW}} - y(t)$ beider Methoden gegen die Zeit. Wie gross kann λ sein, bevor der Fehler der `ROW-2` Methode einen maximalen Wert von 0.05 überschreitet? Verwenden Sie zudem die Methode `solve_ivp` aus `scipy.integrate` und vergleichen Sie Ihre numerische Lösung und deren Approximationsfehler mit dem von `solve_ivp`. Benutzen Sie für `solve_ivp` die Methoden 'RK45' (default), 'Radau', 'BDF' und 'LSODA'.

c) Messen Sie die Konvergenzordnung beider Methoden. Benutzen Sie hierfür obige Gleichung und Anfangswerte mit $\lambda = 10$. Wählen Sie $N = [2^4, \ldots, 2^{12}]$ und berechnen Sie den Fehler zum Endzeitpunkt $T = 2$ gegenüber der exakten Lösung:

$$y(t) = \frac{ce^{\lambda t}}{1 - c + ce^{\lambda t}}$$

Plotten Sie den Fehler gegen die Anzahl Schritte doppelt logarithmisch.

d) Implementieren Sie eine adaptive Strategie basierend auf den `ROW-2` und `ROW-3` Methoden. Verwenden Sie als Fehlerschätzer die Norm:

$$\varepsilon_i := \|y(t_i)_{\texttt{ROW-2}} - y(t_i)_{\texttt{ROW-3}}\|_2$$

Bitte wenden!

Wählen Sie den initialen Zeitschritt als $h_0 = T/(100\,(\|f(y_0)\|_2 + 0.1))$ und passen Sie die Grösse des nächsten Zeitschritts durch Verkleinern ($h_{j+1} = \frac{h_j}{2}$) oder Vergrössern ($h_{j+1} = 1.1 h_j$) an.

e) Testen Sie die Implementation wiederum an der logistischen Differentialgleichung mit $\lambda = 50$. Wie viele Zeitschritte werden insgesamt zur Lösung benötigt? Plotten Sie die numerische Lösung $y(t)_{\texttt{ADA}}$ sowie die Fehler $y(t)_{\texttt{ADA}} - y(t)$ gegen die Zeit.

f) Lösen Sie das folgende gekoppelte System:

$$\dot{y}_0(t) = -76\, y_0(t) - 25\sqrt{3}\, y_1(t)$$
$$\dot{y}_1(t) = -25\sqrt{3}\, y_0(t) - 26\, y_1(t)$$

mit Anfangswerten $y_0(0) = 1$ und $y_1(0) = 1$ bis zum Zeitpunkt $T = 1$ mit dem adaptiven Verfahren und einer Anfangsschrittweite von $h = 0.1$, plotten Sie $y(t)$.

g) Lösen Sie die folgende sehr steife Gleichung:

$$\dot{y}(t) = \lambda y^2(t)(1 - y^2(t))$$

mit dem Anfangswert $y(0) = 0.01$ und $\lambda = 500$. Plotten Sie die numerische Lösung $y(t)_{\texttt{ADA}}$ sowie die Grösse der Zeitschritte gegen die Zeit. Wie viele Zeitschritte benötigt dieses Verfahren und was ist der kleinste Zeitschritt? Wie viele Zeitschritte dieser Grösse würde ein nicht-adapives Verfahren benötigen?

2. *Gram–Schmidt-Verfahren und Householder-Transformation*

In der Vorlesung haben wir die Householder-Transformation verwendet um die **QR**-*Zerlegung* einer Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ zu bestimmen. Ein weiteres, sehr intuitives Verfahren, das sukzessive die Spalten $\underline{a}_1, \ldots, \underline{a}_n$ mit $\underline{a}_i \in \mathbb{R}^m$, von $\mathbf{A}$ orthogonalisiert, ist das Gram–Schmidt-Verfahren. Das Gram–Schmidt-Verfahren ist ein Standardwerkzeug in Beweisen der Linearen Algebra. Die folgenden Algorithmen (in Pseudo-Code) liefern eine **QR**-Zerlegung nach dem *Gram–Schmidt-Verfahren* und dem *modifizierten Gram–Schmidt-Verfahren*:

Gram–Schmidt:

for $j = 1, \ldots, n$ **do**
  $\mathbf{v_j} = \mathbf{A}_{:j}$
  for $i = 1, \ldots, j-1$ **do**
    $\mathbf{R}_{ij} = \mathbf{q_i}^{\mathrm{T}} \mathbf{a_j}$
    $\mathbf{v_j} = \mathbf{v_j} - \mathbf{R}_{ij}\mathbf{q_i}$
  end for
  $\mathbf{R}_{jj} = \|\mathbf{v_j}\|_2$
  $\mathbf{Q}_{:j} = \frac{\mathbf{v_j}}{\mathbf{R}_{jj}}$
end for

Modifiziertes Gram–Schmidt:

for $j = 1, \ldots, n$ **do**
  $\mathbf{v_j} = \mathbf{A}_{:j}$
  for $i = 1, \ldots, j-1$ **do**
    $\mathbf{R}_{ij} = \mathbf{q_i}^{\mathrm{T}} \mathbf{v_j}$
    $\mathbf{v_j} = \mathbf{v_j} - \mathbf{R}_{ij}\mathbf{q_i}$
  end for
  $\mathbf{R}_{jj} = \|\mathbf{v_j}\|_2$
  $\mathbf{Q}_{:j} = \frac{\mathbf{v_j}}{\mathbf{R}_{jj}}$
end for

a) Implementieren Sie die beiden Gram–Schmidt-Verfahren in `ortho.py` und verwenden Sie beide Verfahren, um die **QR**-Zerlegung der Matrix $\mathbf{Z} \in \mathbb{R}^{m\times m}$ mit den Einträgen:

$$\mathbf{Z}_{ij} = 1 + \min(i, j), \quad 0 \le i, j < m$$

für $m = 50$ zu bestimmen.

Siehe nächstes Blatt!

b) Vergleichen Sie die Güte der beiden Gram–Schmidt-Verfahren in Bezug auf die Orthogonalität der Spalten von **Q**.

c) Warum sind die Gram–Schmidt-Verfahren im Gegensatz zur Householder-Transformation (siehe `ortho.py`) ungeeignete numerische Methoden zur Berechnung von **QR**-Zerlegungen?

d) Schreiben Sie einen Code, der die Matrizen $R_1, \ldots, R_m$ aus der Vorlesung zur Umschreibung des Gram–Schmidt-Verfahrens als Multiplikation mit oberen Rechteckmatrizen berechnet. Verwenden Sie die Matrix **Z** aus Aufgabe a) mit $m = 4$ und geben Sie Matrizen R_k und die Skalarprodukte der orthonomisierten Vektoren in jedem Schritt aus.

3. *Konditionszahl*

Für $A \in \mathbb{R}^{m\times n}$, $m \geq n$, ist die Konditionszahl definiert durch

$$\mathrm{cond}(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|}.$$

Sei $A = QR$ die QR-Zerlegung von A mit $R = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$. Zeigen Sie, dass für die zur euklidischen Norm gehörende Konditionszahl cond_2 gilt:

1. $\mathrm{cond}_2(A) = \mathrm{cond}_2(R) = \mathrm{cond}_2(\tilde{R}) \geq \frac{\max_{i=1,\ldots,n} |r_{ii}|}{\min_{k=1,\ldots,n} |r_{kk}|}$
2. $\mathrm{cond}_2(A^T A) = \mathrm{cond}_2(A)^2$

4. *Zerlegung einer reellen Matrix*

Gegeben seien $M \in \mathbb{R}^{n\times n}$ und eine Matrix $G \in \mathbb{R}^{m\times n}$, $m \leq n$, die vollen Rang besitzt. Zeigen Sie:

1. Falls $v^T M v > 0$ für alle $v \neq 0$ mit $Gv = 0$, so ist die Matrix $A = \begin{bmatrix} M & G^T \\ G & 0 \end{bmatrix}$ invertierbar.
2. Falls M symmetrisch und positiv definit ist, existiert eine Zerlegung der Form

$$\begin{bmatrix} M & G^T \\ G & 0 \end{bmatrix} = \begin{bmatrix} L & 0 \\ GL^{-T} & R^T \end{bmatrix} \begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} L^T & L^{-1}G^T \\ 0 & R \end{bmatrix}$$

Wieviele Operationen sind zur Lösung eines Gleichungssystems $Ax = b$ mit einer derartigen Matrix nötig?
<u>Hinweis</u>: Cholesky-Zerlegung von M.

5. *Die Normalengleichungen sind schlecht konditioniert*

Wir betrachten die Matrix:

$$\mathbf{A} = \begin{pmatrix} 1+\varepsilon & 1 \\ 1-\varepsilon & 1 \\ \varepsilon & \varepsilon \end{pmatrix}. \tag{1}$$

In exakter Arithmetik ist die Normalengleichung:

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\underline{x} = \mathbf{A}^{\mathrm{T}}\underline{b} \tag{2}$$

Bitte wenden!

äquivalent zu

$$\mathbf{B}_\alpha \begin{pmatrix} \underline{r} \\ \underline{x} \end{pmatrix} := \begin{pmatrix} -\alpha\mathbf{I} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \underline{r} \\ \underline{x} \end{pmatrix} = \begin{pmatrix} \underline{b} \\ \underline{0} \end{pmatrix}. \tag{3}$$

Schreiben Sie ein `Python`-Skript, das die Kondition von $\mathbf{A}$, $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, $\mathbf{B}_1$ und $\mathbf{B}_\alpha$ mit $\alpha = \varepsilon\|\mathbf{A}\|_2/\sqrt{2}$ für $10^{-5} < \varepsilon < 1$ plottet. Das `Python`-Modul `numpy.linalg` hat eine Funktion `cond`.

Hinweis: Verwenden Sie das Template `condi.py`.