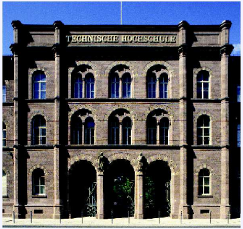
Übung zum Fach Rechnerunterstütze Mechanik I

J. Ruck, H. Erdle, T.-A. Langhoff, T. Böhlke

Chair for Continuum Mechanics
Institute of Engineering Mechanics
Department of Mechanical Engineering
Karlsruhe Institute of Technology (KIT)

WS 2017/2018



Karlsruher Institut für Technologie

Ü5: Differentialoperatoren

Themen der 5. Übung

- Numerisches Differenzieren von Funktionen mit einer Variablen:
  - Forward-Euler
  - Backward-Euler
  - Mid-Point-Rule
  - ZUSATZ: Numerische Lösung gewöhnlicher Differentialgleichungen:
    - explizites Einschrittverfahren
    - implizites Einschrittverfahren
- Darstellung zweidimensionaler Plots

Idee

Taylorreihen-Entwicklung

$$
\begin{aligned}
f(t+\delta t) &= f(t) + \delta t f'(t) + \frac{\delta t^2}{2} f''(t) + \dots \\
&= f(t) + \sum_{i=1}^{N} \frac{\delta t^i}{i!} \frac{\partial^i f(t)}{\partial t^i} + O(\delta t^{N+1})
\end{aligned}
$$

Definition: **Konsistenzordnung**

Ein Verfahren $\mathcal{D}f(t)$ zur numerischen Differenziation heißt konsistent von der Ordnung k, falls es für Polynome vom Grade k exakt ist:

$$
E(\mathcal{D}p(t)) = 0 \qquad\qquad (\forall p \in \mathcal{P}_k).
$$

Im Folgenden gilt $t_n = t_0 + n\delta t \;\; (n = 0, \dots, N;\; \delta t = (T - t_0)/N)$. Man bezeichnet eine solche Unterteilung als äquidistant.

Forward-Euler

Approximiere die Ableitung zum Zeitpunkt t aus $[t, t + \delta t]$ durch

$$\frac{\partial}{\partial t} f(t) \approx \frac{f(t + \delta t) - f(t)}{\delta t};$$

Backward-Euler

Approximiere die Ableitung zum Zeitpunkt t aus $[t - \delta t, t]$ durch

$$\frac{\partial}{\partial t} f(t) \approx \frac{f(t) - f(t - \delta t)}{\delta t};$$

Mittelpunktregel

Approximiere die Ableitung zum Zeitpunkt t aus $[t - \delta t, t + \delta t]$ durch

$$\frac{\partial}{\partial t} f(t) \approx \frac{f(t + \delta t) - f(t - \delta t)}{2\delta t};$$

Konsistenzordnungen

- Forward-Euler, Backward-Euler: 1
- Mittelpunktregel: 2

Zusatzaufgabe

Konstruieren Sie aus der Taylorreihenentwicklung um den Punkt t ein Verfahren 4.Ordnung. Greifen Sie dabei auf Funktionswerte aus dem Intervall $[t - 2\delta t, t + \delta t]$ zu.

Hinweise:

- Gehen Sie von einer äquidistanten Partitionierung aus (d.h. $t_i - t_{i-1} = \delta t$ ist unabhängig von i).
- Eliminieren Sie die (unbekannten) Ableitungen erster und höherer Ordnung aus den Gleichungen.
- Das Gleichungssystem sieht (schematisch) so aus:

  Finde $\alpha_{-2}, \ldots, \alpha_1$, so dass

  $$\alpha_{-2} f(t - 2\delta t) + \cdots = f'(t) + \mathcal{O}(\delta t^4). \qquad (1)$$

- Wie groß ist der Koeffizient vor $f^{IV}(t)$ im Fehlerterm? Was kann dieser Wert aussagen?

Rechnerunterstütze Mechanik I

J. Ruck, H. Erdle, T.-A. Langhoff, T. Böhlke

Dünnbesetzte Matrizen in Matlab

Idee: Nur wenige Komponenten sind $\neq 0$

$\rightarrow$ Die Matrix A kann über drei Listen definiert werden:

- $i_\mathrm{r}, i_\mathrm{c}$: Index von Zeile (**r**ow) bzw. Spalte (**c**olumn)
- v: Komponente der Matrix

Beispiel

```
ix = [1, 1, 2, 2, 2, 3, 3];
iy = [1, 2, 1, 2, 3, 2, 3];
v = [1, -1, -1, 2, -1, -1, 1];
A = sparse(ix, iy, v);
```

$$A = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

Zusatzmaterial beachten (Beispiel)

Vorteile bei der Implementierung

- Gezielte Manipulation einzelner Komponenten einfach möglich
- Zeilen/Spaltenoperationen leicht möglich; insbesondere:
  - Spalte/Zeile löschen (Reduktion der Systemmatrix)
  - Spalten/Zeilen vertauschen
- Modifikation von Komponenten durch Addition/Subtraktion:
  Keine direkte Manipulation von `v` notwendig, sondern:
  Neuen Eintrag hinzufügen; Komponenten werden addiert; Beispiel:

```
ix = [ 1,1 ];
iy = [ 1,1];
v = [2; -3];
A = sparse(ix, iy, v)
```

$$A = (2 - 3) = (-1)$$

- Vektor-Matrix-Multiplikation deutlich *kostengünstiger*
  $\rightarrow$ Wichtig für iterative Gleichungslöser

Zur Einbindung diskreter Differentialoperatoren in Matrixform optimal

Weitere sparse-Befehle

- `speye(N);`
  Einheitsmatrix; gleichbedeutend zu
  `ix=1:N; iy=1:N; v=ones(1,N); A=sparse(ix, iy, v);`
- `Asp = sparse(A); Afull = full(Asp);`
  Umwandlungen: vollbesetzt $\rightarrow$ sparse (und umgekehrt)
- `[ix,iy,v]=find(A);`
  Auslesen der Felder `ix,iy,v`, die A definieren
- `[V, D] = eigs(A,k);`
  Berechnet die k größten Eigenwerte und Eigenvektoren von A;
  (s. Matlab-Hilfe)
- `condest(A);`
  Schätzung der Kondition von A bzgl. $\| \bullet \|_1$ (schnell!)
  Achtung! Nicht exakte Konditionszahl!
- `density(A);`
  Berechnet n_{nonzero}/N^2; (dünnbesetzt $\rightarrow \ll 1$)
- `spy(A);`
  Visualisierung der Struktur von A

Lösungs-Visualisierung

```
close all;                                        % schliesse alle figures
figure;                                           % neue figure erstellen
clf;                                              % figure löschen
plot(x, f);                                       % f(x) über x plotten (s.u.)
legend('leg1', 'leg2')                            % Legende ausgeben
title('TITEL','FontWeight','bold','FontSize',16)  % Titel
ausgeben
xlabel('Beschriftung der x-Achse')                % x-Achsen Beschriftung
```

Der plot-Befehl

Prototyp: `plot(x,y,OPTIONEN)`

- `'Color','c'`
  Linienfarbe ('blue', 'yellow', 'magenta', 'cyan', 'red', 'green', 'black')
- `'Linewith',2`
  Liniendicke auf 2 setzen
- `'MarkerEdgeColor', 'c', 'MarkerFaceColor', 'c', 'MarkerSize', 10`
  Größe, sowie Farbe von Rand und Fläche der Punktsymbole setzen (Befehle müssen nicht gemeinsam verwendet werden)
- `'-','--',':','-.'`
  Linie/Gestrichelt/Gepunktete Linie/Punkt-Strich-Punkt-. . .
- `'+','d','*','s','.','x','^','<','>','o','p','h'`
  Symboltyp: Plus, Diamant, Stern, Quadrat, Punkt, X, , ◁, ▷, Kreis, Pentagramm ( ), Hexagramm ( )
- Linienzüge (Polygone) zeichnen:
  `px = [0, 1, 0, 0]; py = [0, 0, 1, 0]; plot(mx,my);`
  Zeichnet ein Dreieck: $(0,0) \rightarrow (1,0) \rightarrow (0,1) \rightarrow (0,0)$

Lösung gewöhnlicher Differentialgleichungen

Definition: **gewöhnliche Differentialgleichungen (ODE)**

$$\dot{u}(t) = f(t, u(t), \dot{u}(t))$$

Ziel: Approximation der (exakten) Lösung durch numerische Verfahren

Annahmen:

$$\begin{aligned} f(t, u(t), \dot{u}(t)) &= f(t, u(t)), \\ t_{i+1} - t_i &= \delta t = \text{konstant} \qquad && (0 \leq i < N), \\ u(t_0) &= u_0 && \text{(Anfangswert gegeben)} \end{aligned}$$

Lösung mit *explizitem-Einschritt-Verfahren*

Approximationsvorschrift

$$f(t,u(t)) = \dot{u}(t) \approx \frac{u(t+\delta t)-u(t)}{\delta t} \quad \Rightarrow \quad u(t+\delta t) \approx u(t) + \delta t f(t,u(t))$$

Für bekanntes $u(t)$ kann *explizit* der Folgewert $u(t+\delta t)$ angegeben werden
$\Rightarrow$ Rechenaufwand ist *a priori* bekannt

Zusatzaufgabe 1 (a) $(t \in [0,100])$

$$\varrho c\dot{\theta}(t) = h_0 \exp\left(-k/\theta(t)\right),$$
$$\theta(0) = \theta_0 = 293 \text{ K}$$

Zusatzaufgabe 1 (b) $(t \in [0,100])$

Zusätzliche Kühlung:

$$\varrho c\dot{\theta}(t) = h_0 \exp\left(-k/\theta(t)\right) \underbrace{\pm}_{?} q_{\mathrm{ab}},$$

$$\theta(0) = \theta_0 = 293 \text{ K}$$

Implizite Verfahren

Implizites Verfahren: Für gegebenes $u(t)$ ist $u(t+\delta t)$ (i.A.) *nicht* explizit anzugeben

Implizites Euler-Verfahren

Approximationsvorschrift

$$f(t+\delta t, u(t+\delta t)) = \dot{u}(t) \approx \frac{u(t+\delta t)-u(t)}{\delta t}$$

$$\Rightarrow \ u(t+\delta t) \approx u(t) + \delta t f(t+\delta t, \underbrace{u(t+\delta t)}_{\text{kritisch}})$$

Im Allgemeinen ist die Lösung der (i.A. nicht-linearen) Gleichung unbekannt;

Ansatz

Löse das Nullstellenproblem $f(t+\delta t, u_{i+1}) - \frac{u_{i+1}-u_i}{\delta t} = 0$
$\Rightarrow$ Gleichung für u_{i+1} (i.A. nichtlinear)

Vorteile

- Stabiles Verfahren (oft notwendig für realistische Schrittweiten)
- Lösung wird nicht blind aus dem vorigen Schritt extrapoliert (vgl. Forward-Euler)
- Zeitschrittweiten impliziter Verfahren bei FEM häufig mehrere Größenordnungen über expliziten Verfahren

Nachteile

- Rechenzeit nicht *a priori* bekannt und i.A. sehr problemspezifisch
- Lösung nicht-linearer Gleichungen oft zeitaufwändig; Implementierung aufwändiger

Beispiel: Zusatzaufgabe 2