1. Kurze Fragen

(a) Wir lösen das Problem $\dot{y}(t) = -1000y(t) + 10000\sin(t) + 10\cos(t)$, $y(0) = 0$. Welches der folgenden Ergebnisse erhalten wir, wenn wir das *explizite Eulerverfahren* verwenden?

Antwort: ....

Begründen Sie *kurz* Ihre Antwort.

h	Fehler	Konv.-Ordnung
1.5625e-02	6.5475e-05	–
7.8125e-03	3.2793e-05	0.9975
3.9062e-03	1.6410e-05	0.9987
1.9531e-03	8.2087e-06	0.9993
9.7656e-04	4.1052e-06	0.9996
4.8828e-04	2.0528e-06	0.9998

(a)

h	Fehler	Konv.-Ordnung
1.5625e-02	1.2107e+68	–
7.8125e-03	2.8856e+99	-1.0423e+02
3.9062e-03	2.4215e+110	-36.2882
1.9531e-03	8.2156e-06	3.8358e+02
9.7656e-04	4.1069e-06	1.0003
4.8828e-04	2.0532e-06	1.0001

(b)

h	Fehler	Konv.-Ordnung
1.5625e-01	1.2107e+68	–
7.8125e-02	2.8856e+99	-1.0423e+02
3.9062e-02	2.4215e+110	-36.2882
1.9531e-02	8.2156e-06	3.8358e+02
9.7656e-03	4.1069e-06	1.0003
4.8828e-03	2.0532e-06	1.0001

(c)

h	Fehler	Konv.-Ordnung
7.8125e-01	1.2107e+68	–
3.9062e-01	2.8856e+99	-1.0423e+02
1.9531e-01	2.4215e+110	-36.2882
9.7656e-02	8.2156e-06	3.8358e+02
4.8828e-02	4.1069e-06	1.0003
2.4414e-02	2.0532e-06	1.0001

(d)

Siehe nächstes Blatt!

PDF-XChange Editor Click to BUY NOW! www.pdf-xchange.com

(b) Welche/s der folgenden Anfangswertprobleme erfüllt/en die Voraussetzungen von Picard-Lindelöf?

- ☐ $\dot{y}(t) = e^{\frac{1}{y(t)-2}},\ y(2) = -1$
- ☐ $\dot{y}(t) = e^{\frac{1}{y(t)-2}},\ y(4) = 2$
- ☐ $\dot{y}(t) = e^{\frac{1}{y(t)-2}},\ y(1) = 1$
- ☐ $\dot{y}(t) = e^{\frac{1}{y(t)-2}},\ y(2) = 2$

(c) Welche/s der folgenden Verfahren ist/sind A(0)-stabil?

- ☐ Implizite Trapezregel
- ☐ Implizite Mittelpunktsregel
- ☐ Explizite Eulerverfahren
- ☐ RK4
- ☐ BDF4

(d) Ist das folgende Problem (der mathematischen Definition aus der Vorlesung zufolge) steif?

$$\begin{cases} \dot{\mathbf{y}}(t) = \begin{pmatrix} -2 & 80 & 4+i \\ 0 & -5 & 0 \\ 0 & 0 & -800 \end{pmatrix} \mathbf{y}(t), \\ \mathbf{y}(0) = \mathbf{y}_0. \end{cases}$$

Begründen Sie *kurz* Ihre Aussage.
Antwort: ....

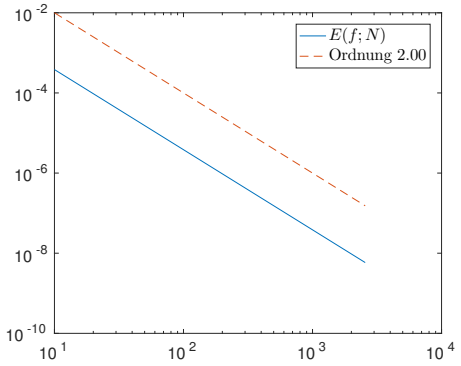
Bitte wenden!

2. Quadratur

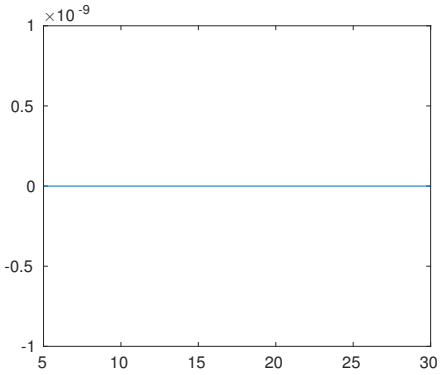
a) Gegeben sei eine Funktion $f : [0,1] \to \mathbb{R}$. Wir approximieren $\int_0^1 f(x)dx$ mithilfe der *summierten Trapezregel* $Q_N^T(f;0,1)$, basierend auf N Intervallen der Länge $h = \frac{1}{N}$. Die folgenden Graphiken zeigen das Verhalten von

$$E(f;N) = \left| \int_0^1 f(x)dx - Q_N^T(f;0,1) \right|$$

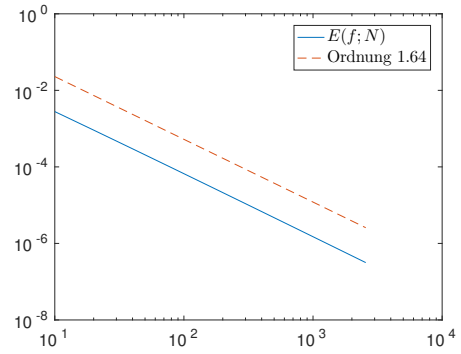
bezüglich N für 4 verschiedene Funktionen f, wobei sowohl x- als auch y-Achse geeignet skaliert wurden (, das heisst zum Teil eine log-Skalierung verwenden).



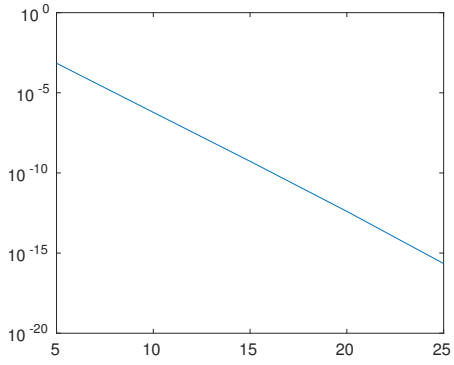
(a)



(b)



(c)



(d)

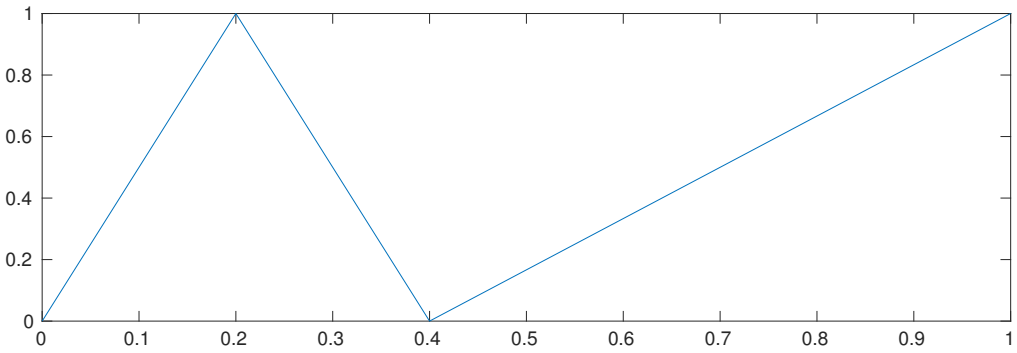
Ordnen Sie die Graphiken den folgenden Funktionen zu:

- Funktion $f(x) = x^{2/3}$ entspricht der Graphik: ...
- Funktion $f(x) = x$ entspricht der Graphik: ...
- Funktion $f(x) = \sin(x)$ entspricht der Graphik: ...
- Funktion $f(x) = \frac{1}{\sqrt{1+\frac{\sin(2\pi x-1)}{2}}}$ entspricht der Graphik: ...

(Eine Begründung ist nicht notwendig.)

Siehe nächstes Blatt!

b) Gegeben sei eine Funktion $f : [0, 1] \to \mathbb{R}$, die durch den folgenden Graph definiert ist.



Welche Methode würden Sie wählen, um $\int_0^1 f(x)dx$ mit sehr hoher Genauigkeit *möglichst effizient* zu berechnen?

c) Berechnen Sie das Integral

$$\int_0^2 \left(x^2 + 3x - 1\right) dx$$

mithilfe der Gauss Quadraturformel mit 2 Stützstellen.

Hinweis: Sie dürfen nicht benutzen, dass die Quadraturformel exakt ist.

Bitte wenden!

3. Wir betrachten das skalare Anfangswertproblem

$$\dot{y}(t) = f(t, y(t)), \quad y(0) = y_0$$

und das Runge-Kutta Einschrittverfahren, das durch das folgende Butcher-Schema gegeben ist

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Berechnen Sie die Stabilitätsfunktion des Verfahrens.

4. Wir betrachten das skalare Anfangswertproblem

$$\dot{y}(t) = f(t, y(t)), \quad y(0) = y_0,$$

und das folgende Verfahren

$$y_{n+1} = y_n + hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n, y_n)\right).$$

Zeigen Sie, dass das Verfahren genau die Konsistenzordnung 2 (und nicht die Ordnung 3) besitzt.

Siehe nächstes Blatt!

5. (Lassen Sie sich von der langen Angabe nicht abschrecken. Die Lösung ist relativ kurz.)

Wir betrachten die lineare Advektionsgleichung

$$\frac{\partial u}{\partial t}(x,t) + a\frac{\partial u}{\partial x}(x,t) = 0$$

auf dem Orts-Zeit-Gebiet $[0,1] \times [0,T]$ mit periodischen Randbedingungen. Es bezeichne $u_j^n \approx u(x_j, t_n)$ die numerische Lösung zum Zeitpunkt t_n am Gitterpunkt $x_j = j\Delta x$, $j = 0, \ldots, N-1$. Zur numerischen Lösung verwenden wir das Lax-Friedrichs-Verfahren definiert durch

$$u_j^{n+1} = u_j^n - \frac{a\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) + \frac{1}{2}(u_{j-1}^n - 2u_j^n + u_{j+1}^n) \tag{1}$$

für innere Gitterpunkte. (An den Gitterpunkten x_0 und x_{N-1} muss das Verfahren angepasst werden, um die periodischen Randbedingungen zu berücksichtigen.)

Wir schreiben (1) um und erhalten

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) = \frac{\Delta x^2}{2\Delta t}\left(\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2}\right). \tag{2}$$

Mit der Definition

$$\mathbf{u}(t) = (u(x_0,t), u(x_1,t), \ldots, u(x_{N-1},t))^T$$

lässt sich (2) interpretieren als die Lösung von

$$\dot{\mathbf{u}}(t) = \mathbf{A}_\epsilon \mathbf{u}(t) \tag{3}$$

mithilfe des *expliziten Eulerverfahrens*, wobei $\epsilon = \frac{\Delta x^2}{2\Delta t}$ und

$$\mathbf{A}_\epsilon := -\frac{a}{2\Delta x}\begin{bmatrix} 0 & 1 & & & -1 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ 1 & & & -1 & 0 \end{bmatrix} + \frac{\epsilon}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}$$

und die periodischen Randbedingungen geeignet in $\mathbf{A}_\epsilon$ berücksichtigt wurden.

Die Eigenwerte von $\mathbf{A}_\epsilon$ sind gegeben durch

$$\begin{aligned} \lambda_k &:= -\frac{ia}{\Delta x}\sin(2\pi k\Delta x) - \frac{4\epsilon}{\Delta x^2}\sin^2(k\pi\Delta x) \\ &= -\frac{ia}{\Delta x}\sin(2\pi k\Delta x) - \frac{2\epsilon}{\Delta x^2}(1 - \cos(2\pi k\Delta x)), \qquad k = 0, \ldots, N-1. \end{aligned}$$

Leiten Sie die CFL Bedingung für das Lax-Friedrichs-Verfahren her, d.h., bestimmen Sie die Zeitschrittweitenbeschränkung für Δt (in Abhängigkeit von a und Δx), so dass das explizite Eulerverfahren angewendet auf (3) stabil ist.

Bitte wenden!

6. In dieser Aufgabe betrachten wir eine (sehr vereinfachte) Form des sogenannten "Trust-Region"-Verfahrens. Ziel ist es, das nicht-lineare Optimierungsproblem

$$\min_{(x_1,x_2)\in\mathbb{R}^2} g(x_1,x_2) := \min_{(x_1,x_2)\in\mathbb{R}^2} \left(x_1^4 + 3x_2^2 - 2x_1x_2 + 3\right)$$

zu lösen. Statt aber direkt dieses Problem zu lösen, lösen wir iterativ quadratische Optimierungsprobleme, welche wir erhalten, indem wir das gegebene Optimierungsproblem lokal durch quadratische Optimierungsprobleme ersetzen. Der grobe Algorithmus ist demnach gegeben durch:
für k=1,2,...

1. finde ein quadratisches Optimierungsproblem $m^{(k)}(x_1,x_2)$, das in der Umgebung der Iterierten $(x_1^{(k)}, x_2^{(k)})$ eine gute Approximation für das ursprüngliche Optimierungsproblem $g(x_1,x_2)$ darstellt;
2. löse das quadratische Optimierungsproblem $m^{(k)}(x_1,x_2)$, was auf die neue Iterierte $(x_1^{(k+1)}, x_2^{(k+1)})$ führt.

Lösen Sie die folgenden Aufgaben.

a) Bestimmen Sie geeignete Werte für $c \in \mathbb{R}$, $\mathbf{b} \in \mathbb{R}^2$ und $\mathbf{A} \in \mathbb{R}^2 \times \mathbb{R}^2$ (in Abhängigkeit der Iterierten $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)})$), so dass für $\mathbf{x} = (x_1, x_2)$ in der Umgebung von $\mathbf{x}^{(k)}$

$$m^{(k)}(x_1,x_2) = c + \mathbf{b}^T(\mathbf{x} - \mathbf{x}^{(k)}) + (\mathbf{x} - \mathbf{x}^{(k)})^T\mathbf{A}(\mathbf{x} - \mathbf{x}^{(k)})$$

eine gute Approximation von $g(x_1,x_2)$ darstellt.

Siehe nächstes Blatt!

b) Vervollständigen Sie den folgenden Code.

```
function trust_region

maxit= 50;    % maximale Anzahl Iterationen
% Funktion g
g = @(x1,x2) ...
% Gradient von g
dg = @(x1,x2) ...
% Hessematrix von g
d2g = @(x1,x2) ...

% Iterierte fuer k=0
x = 0.1*ones(2,1);

for k = 1:maxit
    % werte Komponenten des Ersatzproblems m aus:
    c = ...
    b = ...
    A = ...

    % loese das Ersatzproblem m und berechne neue Iterierte
       xnew
    ...
    ...

    % Stoppe falls xnew der alten Iterierten sehr aehnlich
       ist
    if (norm(xnew-x) < 1.e-8)
        fprintf(1,'Haben minimum gefunden! Stop.\n');
        break;
    end

    % Mache das Update
    x = xnew;

    if (k >= maxit)
        fprintf(1,'Haben maximale Anzahl an Iterationen
           erreicht.Stop.\n');
    end

end
```

Bitte wenden!

PDF-XChange Editor Click to BUY NOW! www.pdf-xchange.com

7. Wir betrachten das Anfangswertproblem

$$\begin{cases} \dot{\mathbf{y}}(t) = & \mathbf{f}(t, \mathbf{y}(t)), \\ \mathbf{y}(a) = & \mathbf{y}_0, \end{cases} \tag{4}$$

mit $\mathbf{y}(t) \in \mathbb{R}^n$, $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ und $n > 1$. Wir wollen (4) mithilfe eines adaptiven Einschrittverfahrens basierend auf der *expliziten Trapezregel* lösen. Den Fehler schätzen wir durch *Schrittweitenhalbierung*, das heisst wir machen, jeweils ausgehend von y_k, einen Schritt mit Schrittweite H, resultierend in y_{k+1}^H, und zwei Schritte mit Schrittweite $\frac{H}{2}$, resultierend in $y_{k+1}^{H/2}$, und schätzen den Fehler mithilfe der Formel

$$EST_{k+1} = \frac{|y_{k+1}^H - y_{k+1}^{H/2}|}{1 - 2^{-p}}.$$

Als Vorschlag für die neue Schrittweite verwenden wir die optimale Schrittweite für den gerade akzeptierten Schritt, den Ideen aus der Vorlesung folgend.

Wir haben versucht, dies in der folgenden MATLAB-Funktion umzusetzen. Leider erhalten wir schlechte Ergebnisse und vermuten, dass sich einige Fehler in unseren Code eingeschlichen haben. *Finden* Sie diese Fehler und *korrigieren* Sie sie.

Hinweise:

- Die Anzahl der Fehler liegt zwischen 5 und 10.
- Korrigieren Sie fälschlicherweise korrekte Stellen, führt dies zu Punktabzug.
- Es ist *NICHT* erlaubt, Zeilen hinzuzufügen, zu streichen oder zu vertauschen.

Erklärungen zum Code:

Die Funktion `ynew = Explizit_Trapez(f,yn,tn,h)` berechnet einen Schritt der expliziten Trapezregel mit Schrittweite h ausgehend von t_n, wobei `ynew` und `yn` Zeilenvektoren sind.

Argumente für Funktion `adaptive_fehler`:

- `f`: rechte Seite der ODE
- `a`,`b`: ODE soll für Zeitintervall $[a, b]$ gelöst werden
- `y0`: Zeilenvektor mit Startwerten für $\mathbf{y}$
- `h`: Startwert für Schrittweite
- `tol`: Toleranz
- `hmin`: minimale Schrittweite
- `maxstep`: maximale Anzahl an versuchten Schritten

Siehe nächstes Blatt!

```
function [y tt] = adaptive_fehler(f,a,b,y0,h,tol,hmin,maxstep)
p  = 4;     % Konvergenzordnung
t  = a;     % t: aktuelle Zeit
tt = zeros(maxstep,1);
y  = zeros(maxstep,1);
tt(1)  = a;
y(1,:) = y0;
i = 1;     % Zaehler fuer akzeptierte Schritte

for nstep = 1:maxstep
    ytemp1 = Explizit_Trapez(f,y(nstep,:),t,h);
    yhalf  = Explizit_Trapez(f,y(nstep,:),t,h/2);
    ytemp2 = Explizit_Trapez(f,yhalf,t,h/2);
    e = norm(ytemp2 - ytemp1);

    while e <= tol
        t = t + h;
        i = i + 1;
        tt(i) = t;
        y(i,:) = ytemp1;
        h = max(hmin,min(0.8*h,2*h*(tol/e)^(1/(p+1))));
        if (t + h > b - 1.E-12)
            t = b - h;
        end;
        if (t > b - 1.E-12)
            fprintf(1,'Schrittweitensteuerung erfolgreich.\n');
            break;
        end
    else
        h = h/2;
    end;
    if (h < hmin - 1.E-12)
        fprintf(1,'h zu klein.\n'); break;
    end
end;

if (nstep >= maxstep)
    fprintf(1,'Haben maximale Anzahl an Schritten erreicht.\n');
    fprintf(1,'Brechen Rechnung jetzt ab.\n');
    fprintf(1,'Haben die Zeit T=%f erreicht.\n',t);
end
```

Bitte wenden!