Peer Review

Aufgabe 1

Lösung gemäss Bewertung eingestuft

Schreiben Sie eine Funktion `[ ] = Name_Vorname_S5_Aufg1(f , xmin, xmax, ymin, ymax, hx, hy)`, welche Ihnen das Richtungsfeld der DGL $y'(x) = f(x, y(x))$ auf den Intervallen $[x_{min}, x_{max}]$ und $[y_{min}, y_{max}]$ plottet mit der Schrittweite h_x in $x-$Richtung und h_y in $y-$Richtung. Benutzen Sie dafür die MATLAB Funktionen meshgrid() und quiver().

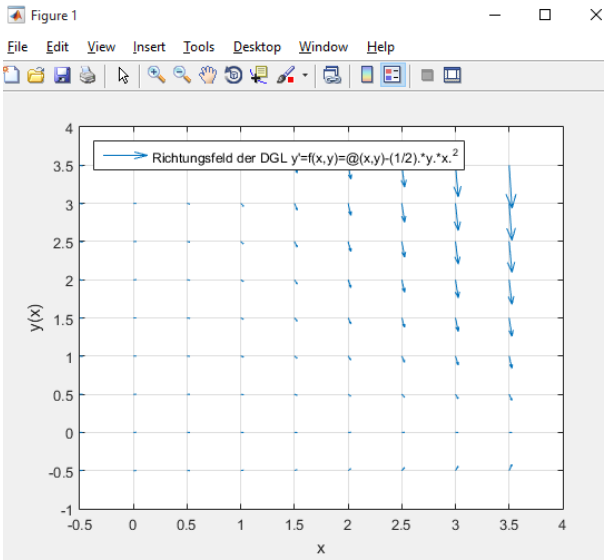
Gehen Sie dafür folgendermassen vor:

(i) Mit meshgrid() erzeugen Sie zuerst die Koordinaten des Punkterasters in der xy- Ebene, z.B. `[X,Y] = meshgrid(0:0.1:5,0:0.1:3)`

(ii) Mit Ihrer Funktion $f(x, y)$ berechnen Sie anschliessend für jeden dieser Punkte die Steigung, z.B. `Ydiff=f(X,Y)`. Die Funktion f muss also mit Vektoren rechnen können, also bei der Funktions-Definition unbedingt die Punkte nicht vergessen, z.B. `f = @(x,y) x.^2.*y.^2`

(iii) Damit quiver() die entsprechenden Steigungsvektoren für jeden Punkt zeichnen kann, erwartet es für jeden Punkt in der (x, y)- Ebene neben den Koordinaten X und Y auch die $x-$Komponenten der jeweiligen Steigungsdreiecke und die entsprechenden $y-$Komponenten. Sie erhalten das gewünschte Resultat, wenn Sie für die $y-$Komponente des Steigungsdreiecks Ydiff übergeben und für die $x-$Komponente eine Matrix mit lauter Einsen.

Lösung der Gruppe:



Die Gruppe hat die Aufgabe 1) in allen 3 Punkten erfüllt.

(i) Das Meshrgrid wird dynamisch aus den Werten **xmin, xmax, ymin, ymax, hx, hy** generiert.

(ii) Ebenfalls wurde die Funktion für den Beispielsaufruf korrekt für Vektorenberechnung übergeben.

(iii) Die Quiver Funktion ist ebenfalls richtig. Die 1er Matrix ‚**dum**' wird ebenfalls dynamisch mit erstellt in gleicher Dimension wie die zuvor aufgerufene **Ydiff = f(X,Y).**

Bewertung: Sehr gut

Besser/Schlechter an unserer Lösung

Die Lösung ist mit unserer Lösung fast identisch, die Aufgabe lässt auch nicht grossen Spielraum zu. Was die Gruppe sicherlich besser gemacht haben, ist die Beschriftung des Plots. Vor allem die dynamische Beschriftung der Funktion (**func2str(f)**).

Korrekt angewendete Algorithmen und Theorien

Die neuen Funktionen meshgrid und quiver wurden korrekt angewandt.

Verbesserungspotenzial

Kein Verbesserungspotenzial mehr möglich.

Aufgabe 2

Lösung gemäss Bewertung eingestuft

Betrachten Sie die folgende DGL

$$\frac{dy}{dx} = \frac{x^2}{y}$$

auf dem Intervall $0 \leq x \leq 2.1$ mit $y(0) = 2$. Lösen Sie die DGL manuell mit

(a) dem Euler-Verfahren mit $h = 0.7$.

(b) dem Mittelpunkt-Verfahren mit $h = 0.7$.

(c) dem modifizierten Euler-Verfahren mit $h = 0.7$.

Die exakte Lösung der DGL ist $y(x) = \sqrt{\frac{2x^3}{3} + 4}$. Berechnen Sie für (a)-(c) jeweils den absoluten Fehler $| y(x_i) - y_i |$ für jedes x_i.

Die Aufgabe wurde sehr übersichtlich und Korrekt gelöst. Alle zwischenschritte sind vollständig und nachvollziehbar angegeben. Die Resultate stimmen mit unseren überein. Die Kontrolle mit unserem Matlab Script ergab, dass die Aufgaben richtig gelöst wurden.

An einer Stelle hat sich ein Rundungsfehler eingeschlichen. Dieser könnte aber auch daher kommen, dass mit genaueren Zahlen gerechnet wurde als aufgeschrieben wurden.

Rundungsfehler

Lösung der Gruppe	Überprüfung mit Wolfram Alpha	Korrekt gerundet
$\left\lvert \sqrt{2 \cdot \frac{(2.1)^3}{3} + 4} - 2.8033 \right\rvert = 0.3863$	Input interpretation: $\sqrt{2 \times \frac{2.1^3}{3} + 4} - 2.8033$ Result: 0.386371...	0.3864

Bewertung: Sehr gut

Besser/Schlechter an unserer Lösung

Die Lösung in Form einer Tabelle in Word ist für den Betrachter sehr angenehm zu lesen. Da unsere Lösungen ansonsten überreinstimmen ist neben der besseren Darstellung kein Unterschied festzustellen.

Korrekt angewendete Algorithmen und Theorien

- Eulerverfahren
- Mittelpunktverfahren
- Modifiziertes Eulerverfahren

Verbesserungspotenzial

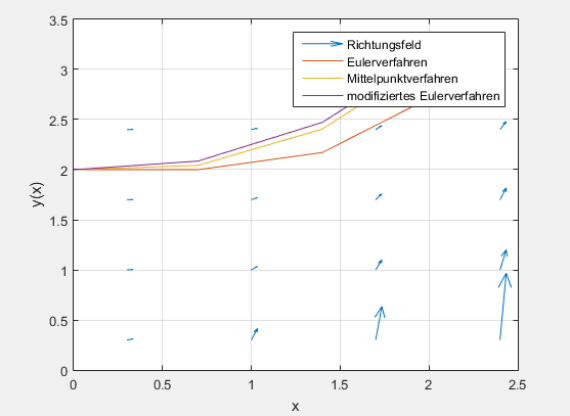
Ausser dem Rundungsfehler wurde kein Verbesserungspotenzial gefunden.

Aufgabe 3

Lösung gemäss Bewertung eingestuft

Schreiben Sie eine Funktion `[x, y_euler, y_mittelpunkt, y_modeuler ] = Name_Vorname_S5_Aufg3(f , a, b, n, y0)`, welche Ihnen das Anfangswertproblem $y'(x) = f(x, y(x)),\ y(a) = y_0$ auf dem Intervall $[a, b]$ mit n Schritten berechnet, sowohl mit dem Euler-Verfahren als auch mit dem Mittelpunkt-Verfahren und dem modifizierten Euler-Verfahren. Die Resultate werden in die Vektoren `y_euler`, `y_mittelpunkt`, `y_modeuler` geschrieben, `x` enthält die entsprechenden x_i-Werte. Ausserdem soll eine Grafik des Richtungsfeldes erzeugt (benutzen Sie dafür ihre Funktion aus Aufgabe 1) und die drei Lösungen eingezeichnet werden. Überprüfen Sie damit Ihre Resultate aus Aufgabe 2.

Grundsätzlich ist bei diesem Script ebenfalls sehr wenig auszusetzten. Der Funktionenkopf ist gemäss Aufgabenstellung aufrufbar. Die 3 Lösungen werden in einem gemeinsamen Plot mit dem Richtungsfeld geplottet.



Ein kleiner, aber relevanter Fehler ist jedoch unterlaufen. Am Anfang wird der **x0**-Wert (bzw. eine Zeile weiter **x(1)**) auf 0 gesetzt. Das mag mit dem Beispielsaufruf im Skript nicht sofort auffallen, ist aber nicht korrekt. Der **x0** Wert müsste auf die Intervallsgrenze **a** gesetzt werden.

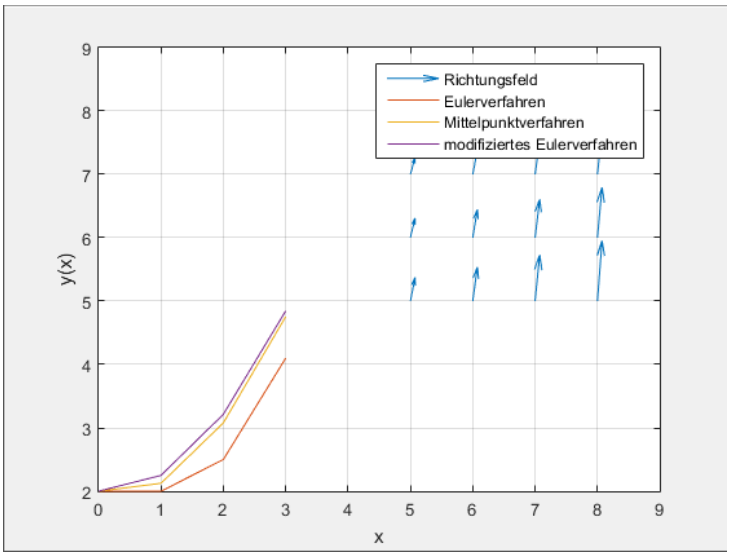
```
 8
 9 -   h = (b-a)/n;
10 -   x0 = 0;
11 -   x(1) = x0;
```

→

```
 8
 9 -   h = (b-a)/n;
10 -   x0 = a;
11 -   x(1) = x0;
```

Ruft man die Funktion mit einem anderen Intervall auf, ist der Fehler relativ schnell ersichtlich:

```
WIN05_IT13t_S5_Aufg3(@(x,y) (x.^2).*(y.^(-1)),5,8,3,2)
```



Das Richtungsfeld passt sich gemäss dem neuen Intervall an, jedoch nicht die geplotteten Funktionen.

Bewertung: Gut, da die Algorithmen richtig programmiert wurden. Abzug für den kleinen, aber relevanten Fehler

Besser/Schlechter an unserer Lösung

Was möglicherweise noch besser gemacht werden könnte, ist die Arbeit mit der Indexe. Anstatt bei 0 die Schleife zu beginnen könnte man ganz ‚matlab' üblich bei 1 beginnen, dies erspart die ganzen Indexkorrekturen in der Schleife (verbessert Leserlichkeit für andere Personen die den Code lesen müssen).

Lösung der Gruppe_

```
y_mittelpunkt(1) = y0; y_h_2 = 0;
for i=0:n-1
    x(i+2) = x(i+1) + h;
    x_h_2(i+1) = x(i+1) + (h/2);
    y_h_2(i+1) = y_mittelpunkt(i+1) + (h/2) * f(x(i+1),y_mittelpunkt(i+1));
    y_mittelpunkt(i+2) = y_mittelpunkt(i+1) + h * f(x_h_2(i+1),y_h_2(i+1));
end
```

Unsere Lösung:

```
h2 = h/2;
for i=1:n
    xh2 = x(i) + h2;
    yh2 = y_mittelpunkt(i) + h2 * f(x(i), y_mittelpunkt(i));
    x(i+1) = x(i) + h;
    y_mittelpunkt(i+1) = y_mittelpunkt(i) + h * f(xh2, yh2);
end
```

Korrekt angewendete Algorithmen und Theorien

Die 3 gelernten Euler-Verfahren wurden korrekt implementiert.

Verbesserungspotenzial

Grundsätzlich wurde die Aufgabe richtig gelöst. Da die Euler Funktionen nicht wirklich sehr komplex sind, bzw. in Matlab relativ einfach gelöst werden können, würden Kommentare im Code unserer Meinung nach eher störend wahrgenommen werden. Lieber in den Schleifen die Indexe nicht korrigieren, sondern in der Idee von Matlab die Indexe nutzten. Fremde Leser können so viel schneller den Code lesen.

Einzig den Initialisierungswert von x0 kann bei dieser Aufgabe vorgehalten werden.