

# Praktikum 1 : DGL

Oliver Steenbuck, Karolina Bernat

31.10.2012

## Inhaltsverzeichnis

<b>1</b>	<b>Steife Differentialgleichungen</b>	<b>2</b>
1.1	Gleichung . . . . .	2
1.2	Iterationslgleichungen . . . . .	2
1.2.1	Euler, explizit . . . . .	2
1.2.2	Euler, implizit . . . . .	2
1.2.3	Runge Kutta 2. Ordnung . . . . .	2
1.3	Matlab Programme . . . . .	3
<b>2</b>	<b>Van der Pol DGL</b>	<b>4</b>
2.1	Gleichung . . . . .	4
2.2	Gleichung als DGL 1. Ordnung . . . . .	4
2.3	Euler Verfahren . . . . .	4
2.4	Runge Kutta 2. Ordnung . . . . .	4
2.5	Ergebnisse . . . . .	5
2.6	Ergebnisse . . . . .	9
2.6.1	h=0.001 . . . . .	9
2.6.2	h=0.02 . . . . .	10
2.7	Matlab Programme . . . . .	11

## Abbildungsverzeichnis

1	Stiff (h=0.001) . . . . .	5
2	Stiff (h=0.002) . . . . .	6
3	Stiff (h=0.003) . . . . .	6
4	Stiff (h=0.004) . . . . .	7
5	Stiff (h=0.005) . . . . .	8
6	Van Der Pol DGL Y1 h=0.001 . . . . .	9
7	Van Der Pol DGL Y2 h=0.001 . . . . .	9

8	Van Der Pol DGL Y1 h=0.02 . . . . .	10
9	Van Der Pol DGL Y2 h=0.02 . . . . .	11

## Listings

1	Stiff . . . . .	3
2	Steife Differentialgleichung . . . . .	3
3	VanDerPol GDL . . . . .	11
4	VanDerPol . . . . .	11

## 1 Steife Differentialgleichungen

### 1.1 Gleichung

$$y(0) = 1 \quad (1)$$

$$y' = 10 - 500 \cdot y + 5000 \cdot x \quad (2)$$

### 1.2 Iterationsgleichungen

#### 1.2.1 Euler, explizit

$$y(0) = 1 \quad (3)$$

$$y_{j+1} = y_j + h \cdot (10 - 500 \cdot y_j + 5000 \cdot x_j) \quad (4)$$

#### 1.2.2 Euler, implizit

$$y(0) = 1 \quad (5)$$

$$y_{j+1} = y_j + h \cdot (10 - 500 \cdot y_{j+1} + 5000 \cdot x_{j+1}) \quad (6)$$

Wobei hier  $y_{j+1}$  mit dem Newton Verfahren Approximiert wird.

### 1.2.3 Runge Kutta 2. Ordnung

Es gelte  $f(x) = 10 - 500 \cdot y + 5000 \cdot x$

$$y(0) = 1 \quad (7)$$

$$y_{j+1} = y_j + \frac{h}{2} \cdot (f(x_{j+1}, y_j) + f(x_j, y_{j+1})) \quad (8)$$

## 1.3 Matlab Programme

Listing 1: Stiff

```

1 function [] = stiff(h)
2
3     vec_ana_x = 0:h:0.2;
4     vec_ana_y = mtp0101_ana(vec_ana_x);
5     [vec_eulereexpl_x, vec_eulereexpl_y] = eulerE(@f, 0.2, h, [1]);
6     [vec_eulerimpl_x, vec_eulerimpl_y] = euler_impl(1, h, 0.2, @f);
7     [vec_runge_x, vec_runge_y] = rungeKutta(1, h, 0.2, @f);
8
9
10    fh = figure('color','w');
11    screen_size = get(0, 'ScreenSize');
12    set(fh, 'Position', [0 0 screen_size(3) screen_size(4)]);
13    a = subplot(1,2,1);
14    %set(gca, 'fontName', 'Humor Sans111', 'fontSize', 14, 'lineWidth', 3, 'box', 'off')
15
16    lw = 1;
17
18
19    %annotation(fh, 'textarrow', [0.38 0.34], [0.42 0.55], ...
20    %    'string', sprintf('analytische Loesung', char(10)), 'headStyle', 'none',
21    %    'lineWidth', 1.5, ...
22    %    'fontName', 'Comic Sans MS', 'fontSize', 14, 'verticalAlignment', 'middle', 'horizontalAlignment', 'left')
23
24    hold on;
25    plot(vec_ana_x, vec_ana_y, 'r', 'lineWidth', lw);
26    plot(vec_eulereexpl_x, vec_eulereexpl_y, 'k', 'lineWidth', lw);
27    plot(vec_eulerimpl_x, vec_eulerimpl_y, 'g', 'lineWidth', lw);
28    plot(vec_runge_x, vec_runge_y, 'b', 'lineWidth', lw);
29    grid off;
30    title('Approximation');
31    legend('analytische Loesung', 'Expl Euler', 'Impl Euler', 'Runge-Kutta')
32    axis([0, 0.04, -1.5, 1.5]);
33
34    %xkcdify(a)
35
36
37
38    subplot(1,2,2)
39    %
40    hold on;
41    plot(vec_eulereexpl_x, vec_eulereexpl_y - vec_ana_y, 'k');
42    plot(vec_eulerimpl_x, vec_eulerimpl_y - vec_ana_y, 'g');
43    plot(vec_runge_x, vec_runge_y - vec_ana_y, 'b');

```

```

44 grid on;
45 title('Error');
46 legend('Expl Euler','Impl Euler','Runge-Kutta');
47 axis([0,0.2,-2.5,2.5]);
48
49 end

```

Listing 2: Steife Differentialgleichung

```

1 function [ z ] = f( x,y )
2     z = 10-500*y+5000*x;
3 end

```

## 2 Van der Pol DGL

### 2.1 Gleichung

$$y(0) = 0 \quad (9)$$

$$\dot{y}(0) = 1 \quad (10)$$

$$\ddot{y} = 6 \cdot (1 - y^2) \cdot \dot{y} - y \quad (11)$$

### 2.2 Gleichung als DGL 1. Ordnung

$$\dot{z} = 6 \cdot (1 - y^2) \cdot z - y \quad (12)$$

$$\dot{y} = z \quad (13)$$

### 2.3 Euler Verfahren

$$z_{1_{n+1}} = z_{1_n} + h \cdot (6 \cdot (1 - z_{2_n}^2) \cdot z_{1_n} - z_{2_n}) \quad (14)$$

$$z_{2_{n+1}} = z_{2_n} + h \cdot z_{1_n} \quad (15)$$

### 2.4 Runge Kutta 2. Ordnung

Es gelte

$$g(t, y) = z \quad (16)$$

$$f(y, z) = 6 \cdot (1 - y^2) \cdot z - y \quad (17)$$

Dann können wir durch einsetzen von (16) und (17) in Runge Kutta 2. Ordnung die Iterationsgleichungen erstellen:

$$y_{j+1} = y_j + \frac{h}{2} \cdot [g(t_j, y_j) + g(t_{j+1}, y_j + h \cdot g(t_j, y_j))] \quad (18)$$

$$z_{j+1} = z_j + \frac{h}{2} \cdot [f(y_j, z_j) + f(y_{j+1}, z_j + h \cdot f(y_j, z_j))] \quad (19)$$

## 2.5 Ergebnisse

Im folgenden sind die Approximation durch alle 3 Verfahren mit Schrittweiten von (0.001 bis 0.005) graphisch dargestellt. Deutlich erkennbar wird hier wie die expliziten Verfahren (Expliziter Euler, Runge Kutta 2. Ordnung) gegenüber dem impliziten Euler Verfahren bei wachsender Schrittweite an Genauigkeit verlieren, wie dies auch zu erwarten war.

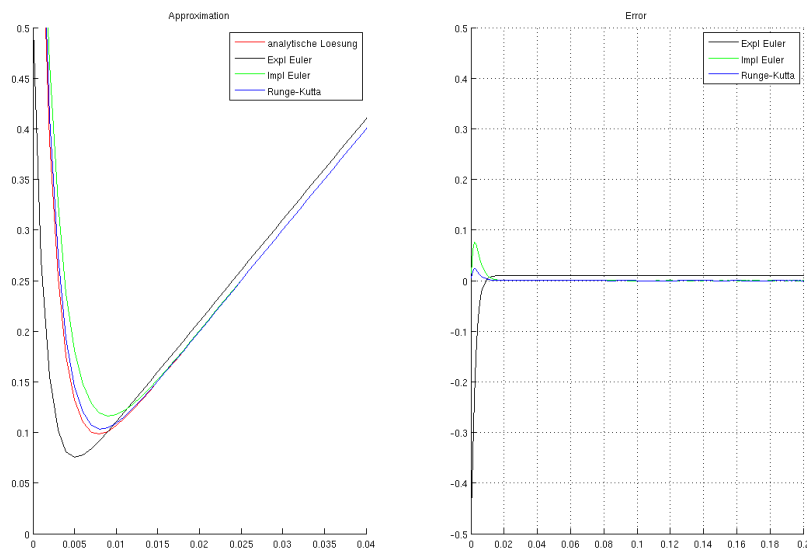
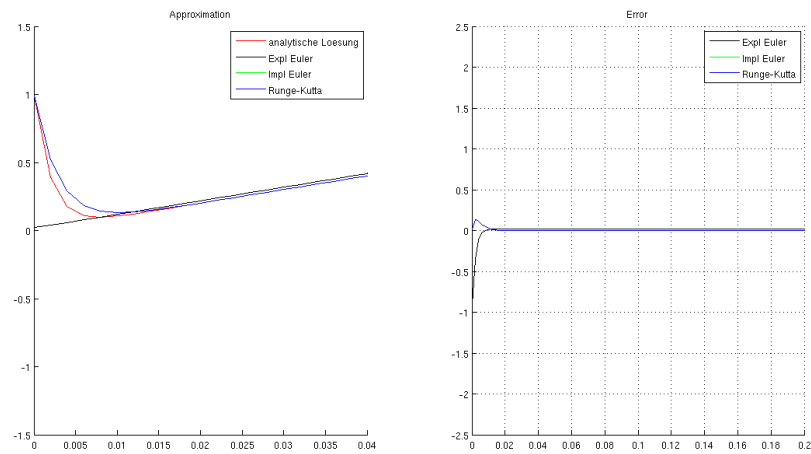
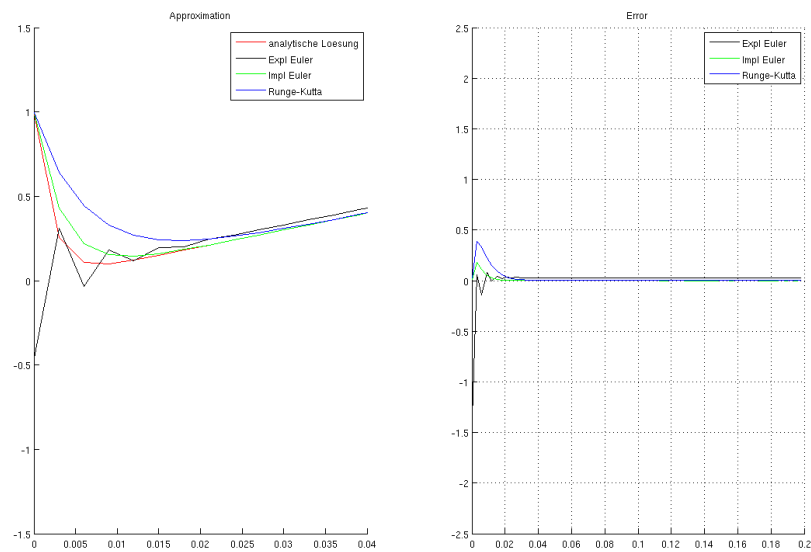
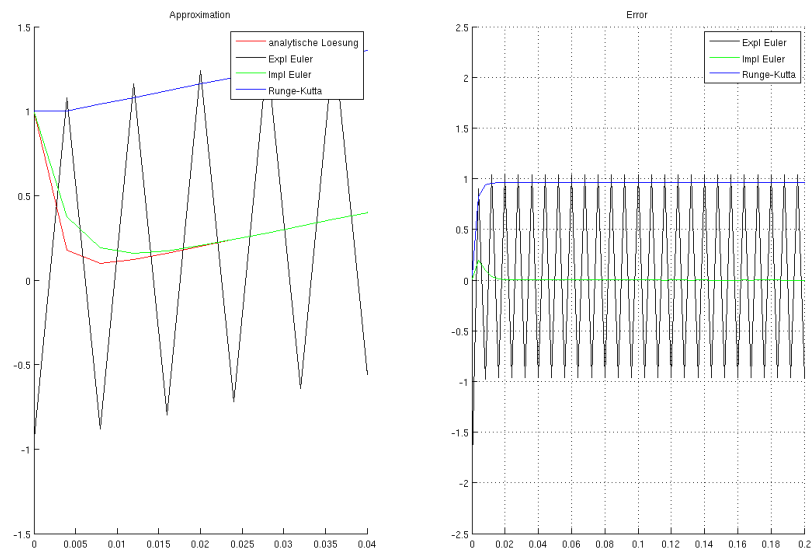
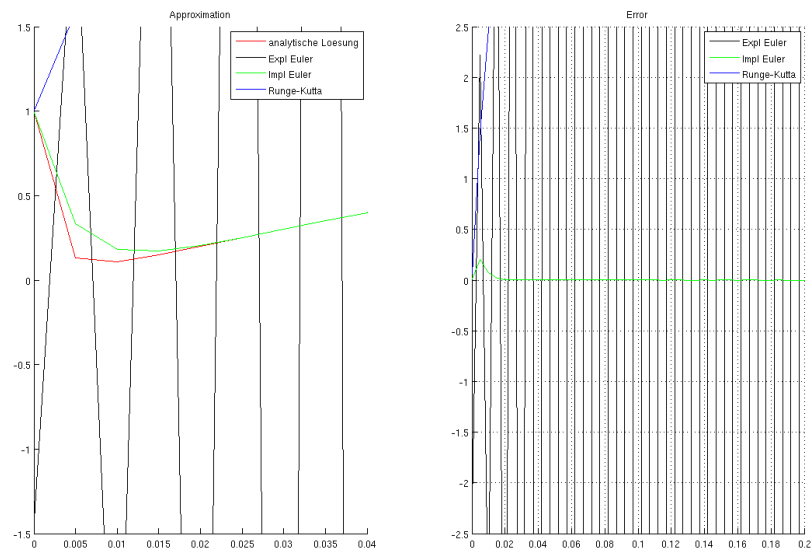


Abbildung 1: Stiff (h=0.001)

Abbildung 2: Stiff ( $h=0.002$ )Abbildung 3: Stiff ( $h=0.003$ )

Abbildung 4: Stiff ( $h=0.004$ )

Abbildung 5: Stiff ( $h=0.005$ )



## 2.6 Ergebnisse

### 2.6.1 $h=0.001$

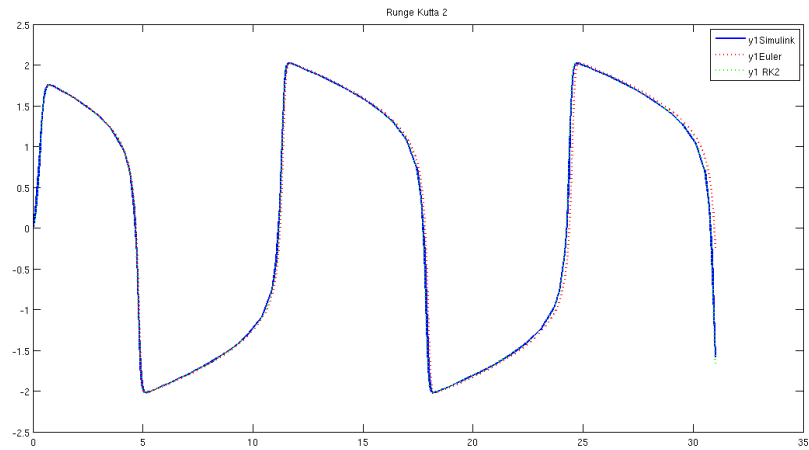


Abbildung 6: Van Der Pol DGL Y1  $h=0.001$

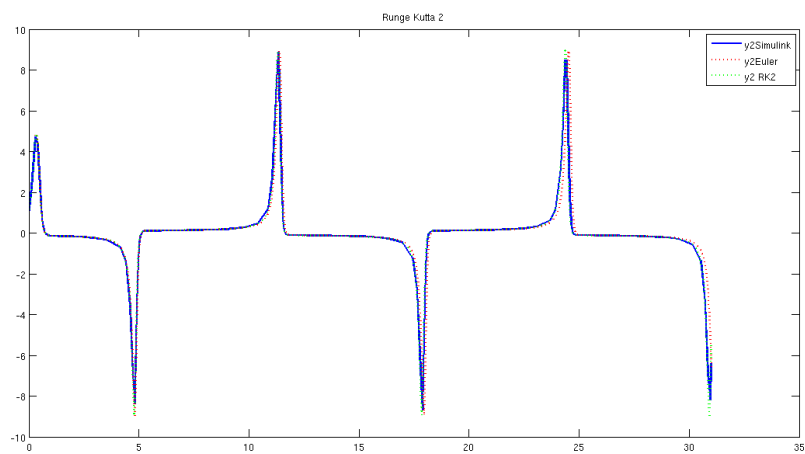


Abbildung 7: Van Der Pol DGL Y2  $h=0.001$

Bei einer Schrittweite  $h$  von 0.001 ist zu erkennen das beide Approximationsverfahren (Expliziter Euler und Runge Kutta 2. Ordnung) mit der aus Simulink extrahierten Approximation (Dormand-Prince, Variable Step Size) übereinstimmen.

### 2.6.2 $h=0.02$

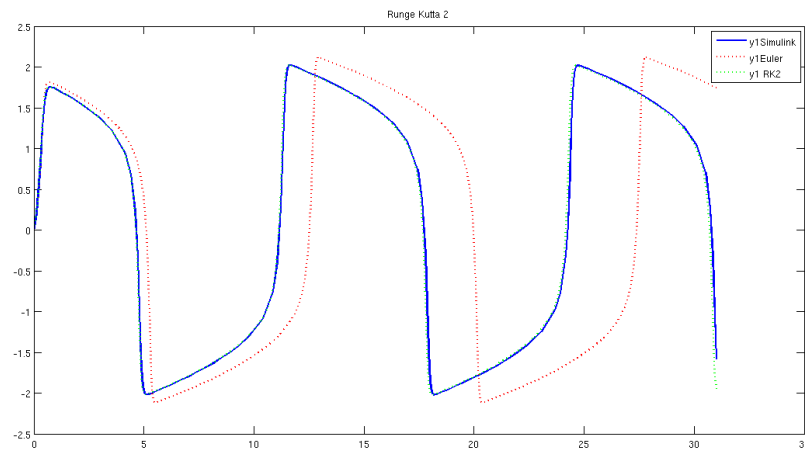


Abbildung 8: Van Der Pol DGL Y1  $h=0.02$

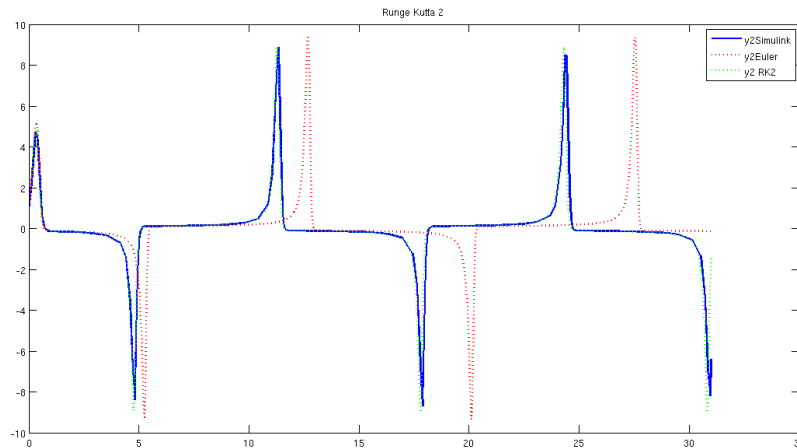


Abbildung 9: Van Der Pol DGL Y2 h=0.02

Bei einer Schrittweite  $h$  von 0.02 ist zu erkennen, dass das einfachere Approximationsverfahren (Expliziter Euler) deutlich von der aus Simulink extrahierten Approximation (Dormand-Prince, Variable Step Size) abweicht, während das komplexere Verfahren (Runge Kutta 2. Ordnung) auch hier noch sehr dicht an Simulink liegt.

## 2.7 Matlab Programme

Listing 3: VanDerPol GDL

```

1 function [ res ] = vdp( x, y )
2 % Van-Der-Pol-Gleichung zu Aufgabe 2
3
4 res = [ y(2); 6 * ( 1 - y(1)^2 ) * y(2) - y(1) ];
5
6 end

```

Listing 4: VanDerPol

```

1 function [ ] = vanDerPolZeigen( y1Sim, y2Sim, h )
2     sum(Prak1Aufg2Simulink)
3     [~, Y_E] = eulerE(@vdp, 31, h, [0;1]);
4     [X, Y_RK] = rk2(@vdp, 31, h, [0;1]);
5
6
7     %hold on;
8     plot(1:21);

```

```
9      p1=plot(y2Sim.time, y2Sim.signals.values, 'b-', X, Y_E(:,1), 'r:', X, Y_RK←  
10      (:,1), 'g:');  
11      set(p1, 'LineWidth', 2);  
12      title('Runge Kutta 2');  
13      legend('y1Simulink', 'y1Euler', 'y1 RK2');  
14  
15      figure  
16      plot(122);  
17      p2=plot(y1Sim.time, y1Sim.signals.values, 'b-', X, Y_E(:,2), 'r:', X, Y_RK←  
18      (:,2), 'g:');  
19      set(p2, 'LineWidth', 2);  
20      title('Runge Kutta 2');  
21      legend('y2Simulink', 'y2Euler', 'y2 RK2');  
22  end
```