

Exercise sheet 1

by Robin Heinemann (group 4), Paul Rosendahl (group 2) and Andreas Rall (group 1)

May 6, 2018

1 Numerical Integration

The integral

$$y_n = y_n(a) = \int_0^1 \left(\frac{x^n}{x+a} \right) dx = \frac{1}{n} - a y_{n-1}$$

is evaluated. First the integrand is plotted for $a = 5$ and $n \in \{1, 5, 10, 20, 30, 50\}$.

[htbp]

```
1 reset
2 set size 1,.75
3 set xrange [0:1]
4 f(x, n) = x**n / (x + 5)
5 set key left
6 plot f(x, 1) with lines title "y_1(5)", f(x, 5) with lines title "y_5(5)", f(x,
  ↪ 10) with lines title "y_{10}(5)", f(x, 20) with lines title "y_{20}(5)", f(x,
  ↪ 30) with lines title "y_{30}(5)", f(x, 50) with lines title "y_{50}(5)"
```

Listing 1: Gnuplot code for plotting the integrand

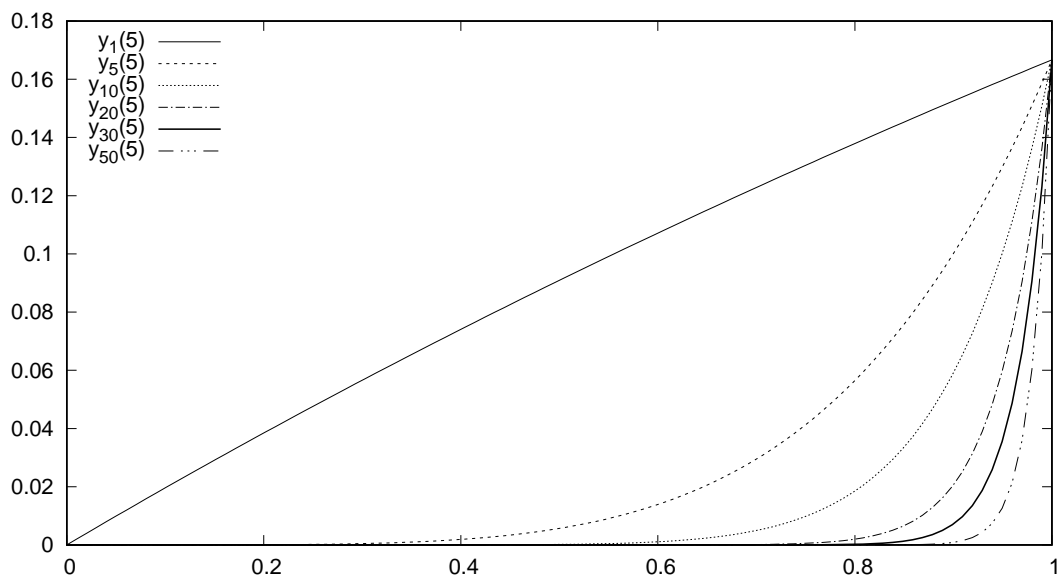


Figure 1: Plot of integrand for different values of n

The forward iteration is given by

$$y_n = \frac{1}{n} - ay_{n-1}$$

solving for y_{n-1} gives a backward iteration

$$y_{n-1} = \frac{1/n - y_n}{a}$$

The source code for implementation in *rust* is shown at listing 2. [htbp]

```

1 fn y(n0: u64, n1: u64, a: f64, yn0: f64) -> f64 {
2     if n0 < n1 {
3         let mut yn = yn0;
4
5         for i in (n0 + 1)..(n1 + 1) {
6             yn = 1.0 / (i as f64) - a * yn;
7         }
8
9         yn
10    } else if n0 > n1 {
11        let mut yn = yn0;
12
13        for i in ((n1 + 1)..(n0 + 1)).rev() {
14            yn = (-yn + 1.0 / (i as f64)) / a;
15        }
16
17        yn
18    } else {
19        yn0
20    }
21 }
22
23 fn main() {
24     let a: f64 = 5.0;
25
26     println!("| n | $y_n$");
27     println!("|--");
28
29     for n in 0..31 {
30         println!("| {} | {}", n, y(0, n, 5.0, ((1.0 + a) / a).ln()));
31     }
32
33     println!("|   | ");
34     println!("|   | ");
35
36     for n in (30..50).rev() {
37         println!("| {} | {}", n, y(50, n, 5.0, 42.0));
38     }
39 }

```

Listing 2: Source code of iteration of y_n

Table 1: Output data produced by forward iteration from zero to 30 and backward iteration from 50 to 30

n	y_n
0	0.1823215567939546
1	0.08839221603022707
2	0.05803891984886467
3	0.04313873408900998
4	0.03430632955495011
5	0.02846835222524946
6	0.024324905540419356
7	0.02123261515504607
8	0.018836924224769652
9	0.016926489987262844
10	0.015367550063685786
11	0.01407134059066198
12	0.012976630380023432
13	0.012039925022959766
14	0.011228946313772595
15	0.010521935097803692
16	0.00989032451098154
17	0.009371906856857001
18	0.008696021271270546
19	0.009151472591015689
20	0.00424263704492156
21	0.026405862394439816
22	-0.08657476651765363
23	0.47635209345783336
24	-2.3400938006225003
25	11.740469003112501
26	-58.66388347710097
27	293.35645442254184
28	-1466.746557826995
29	7333.7672718935955
30	-36668.80302613464
49	-8.395999999999999
48	1.683281632653061
47	-0.3324896598639456
46	0.07075325112172529
45	-0.009802824137388536
44	0.006405009271922152
43	0.0032644526910701153
42	0.003998272252483651
41	0.003962250311408032
40	0.004085598718206199
39	0.0041828802563587605
38	0.004291629076933376
37	0.004404832079350167
36	0.0045244389895353725
35	0.004650667757648481
34	0.004784152162756019
33	0.004925522508625267
32	0.005075501558881007
31	0.005234899688223799
30	0.005404632965581047

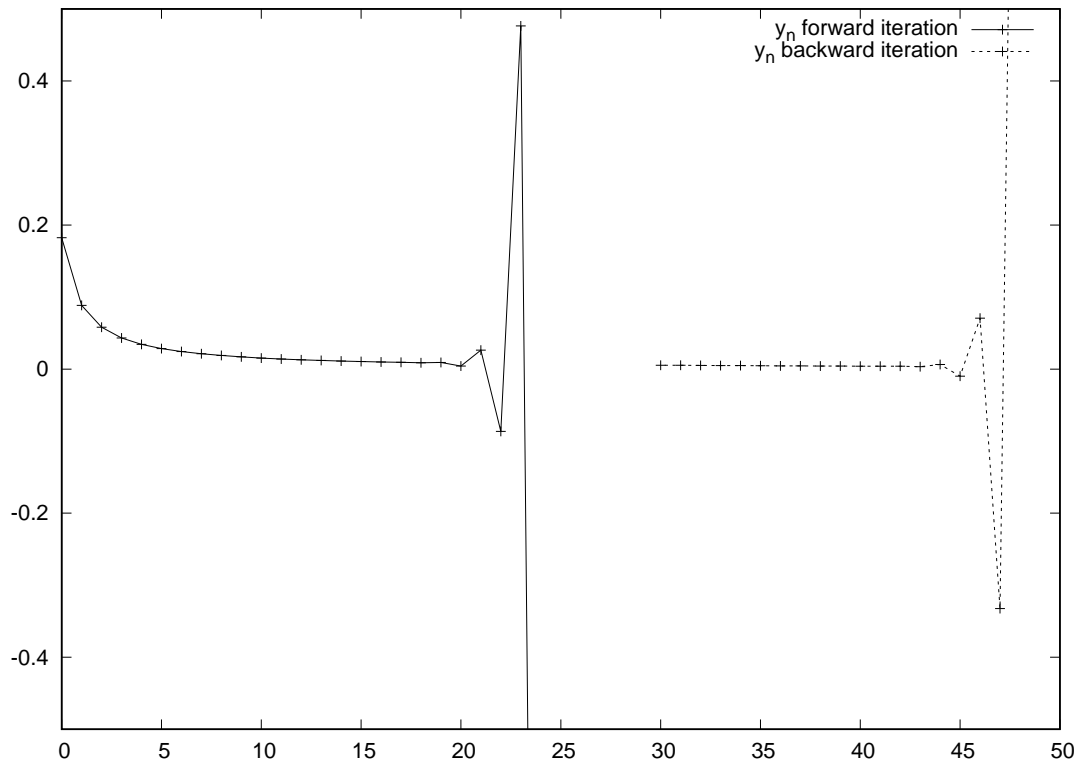
[htbp]

```

1 reset
2 set yrange [-.5:.5]
3 plot data index 0 using 1:2 with linespoints title "y_n forward iteration", data
  ↪ index 1 using 1:2 with linespoints title "y_n backward iteration"

```

Listing 3: Gnuplot code for plotting the iteration

Figure 2: Plot of y_n for forward and backward integration, some values are intentionally left out to show more details of the small values

For small n the forward iteration works fine, but at some point it starts to give wrong values, such as negative and very big values, due to roundoff errors. However using the backward iteration works fine, and the error of an arbitrary (random) initial condition vanishes after some steps.

2 Additional notes

All programs written are written using the programming language *rust*. Extra dependencies (*rust crates*) will be listed in a comment in the first line. To get the source files of each program just unzip this *pdf* file. You will find directories for every program in this file. To execute one of the programs run `cargo run` in it's directory. All plots are made with *gnuplot*. This document was written in *org-mode* and converted to *pdf*. The corresponding *org-mode* sources can also be found by unzipping this *pdf* file.