

# Computerphysik I: Blatt 05

Aurel Müller-Schönau und Leon Oleschko

8. Juli 2022

## Aufgabe 5 - Numerov-Verfahren

a)

In Programm 1 ist die Implementierung von Analytisch

$$\text{Numerisch:} \quad -0.039788 \quad (1)$$

$$\text{Analytisch:} \quad -\frac{1}{8\pi} \approx -0.039788 \quad (2)$$

b)

## Aufgabe 6 - Shooting-Methode

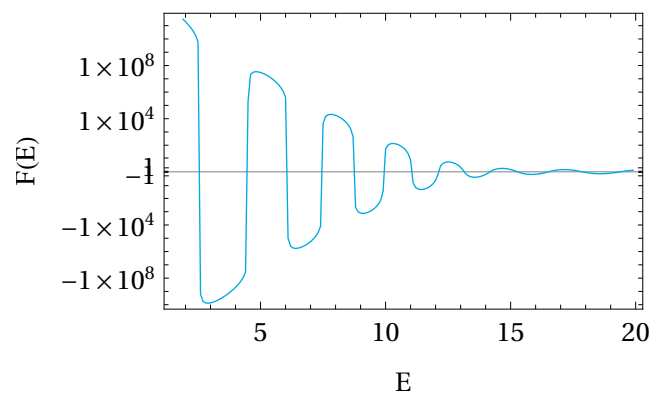
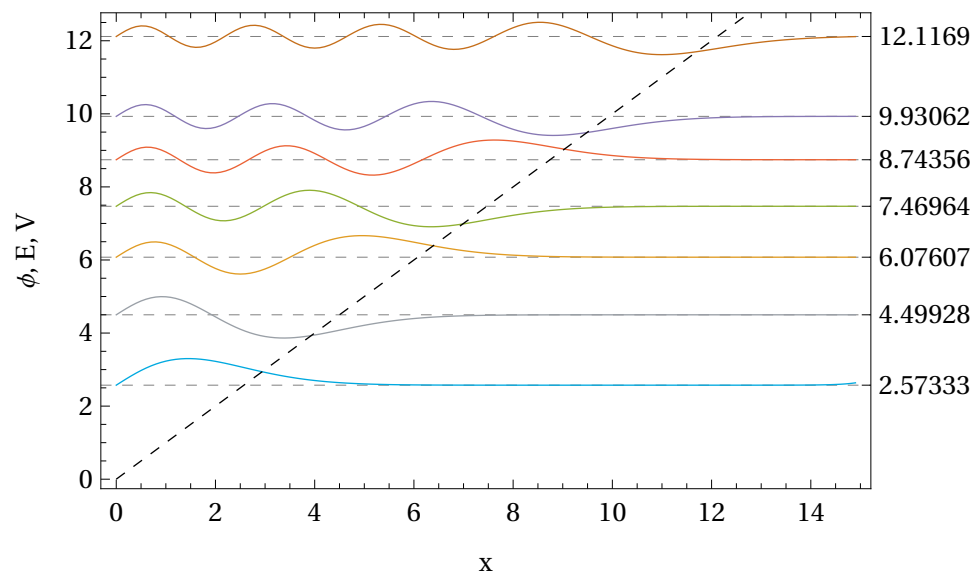


Abbildung 1: Fehlerfunktion für verschiedene Energien



**Abbildung 2:** Potential mit verschiedenen Wellenfunktionen

## Code

### 5 a)

**Listing 1:** Programm zum lösen der Poisson-Gleichung mit dem Numerov-Verfahren.

```
1 #include <stdio.h>
2 #include <math.h>
3
4
5 #define H 0.001
6 #define rstart 100
7
8
9 double rho(double r);
10 void numerov(double* y1, double* y2, double r);
11
12
13 int main(){
14
15     double y1=0, y2 = 0;
16
17     for(double r = rstart; r > 0; r -= H){
18         numerov(&y1, &y2, r);
19         printf("%f\n", y2);
20     }
21
22     // Ergebnis: -0.039789
23     // Analytisch: -1/(8 pi) \approx -0.0397887
24
25     return 0;
26 }
27
28
29 double rho(double r){
30     double wert = exp(-r)/(8 * M_PI);
31     return wert;
32 }
33
34 void numerov(double* y1, double* y2, double r){
35     double y = 2 * *y2 - *y1 - H*H/12*(rho(r+H)+10*rho(r)+rho(r-H));
36     *y1 = *y2;
37     *y2 = y;
38 }
```

## 6

**Listing 2:** Programm zum aufzeichnen der Error-Funktion für verschiedene Energien.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 #define H 1.e-4
6 #define XMAX 15.
7
8 void integ(double E);
9
10 void numerov(double *y1, double *y2, double x, double E);
11
12 double V(double x);
13
14 FILE *file;
15
16 int main(){
17
18     file = fopen("errorFkt.dat", "w+");
19
20
21     for(double E = 0.0; E < 20; E += 0.1){
22         integ(E);
23     }
24
25     //integ(5.0);
26
27
28
29     fclose(file);
30
31     return 0;
32 }
33
34
35 void integ(double E){
36
37     double y1 = 0, y2 = H;
38     for(double x = XMAX; x > 0; x -= H){
39         numerov(&y1, &y2, x, E);
40         //fprintf(file, "%f %f \n", x, y1);
41     }
```

```
42     fprintf(file, "%f %f \n", E, y2);
43 }
44
45 void numerov(double *y1, double *y2, double x, double E){
46
47     double y = 2* *y2*(1 - H*H*5/12*(E-V(x))) - *y1*(1 - H*H/12*(E-V(x-H)
48     ));
49     *y1 = *y2;
50     *y2 = y;
51 }
52
53 double V(double x){
54     return x;
55 }
```

## Listing 3: ...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <stdbool.h>
5 #include <pthread.h>
6
7 #define H 1.e-4
8 #define X_MAX 15.
9 #define N ((int) (X_MAX/H))
10 #define EXPORT_STEPS 1000
11
12 #define UNCERT_END .5e-15
13
14 #define UNCERT_GUESS 0.5
15 //double guess[] = {2.5, 4.5, 6, 7.5, 8.5, 10, 12, 14, 16, 18.5};
16 double guess[] = {2.5, 4.5, 6, 7.5, 8.5, 10, 12};
17
18 void integ(double E, double* phi);
19 void numerov(double *y1, double *y2, double x, double E);
20 void *findAndExport(void *argument);
21 double V(double x);
22
23
24 int main(){
25     const uint NUM_THREADS = sizeof(guess)/sizeof(double);
26     int thread_args[NUM_THREADS];
27     pthread_t threads[NUM_THREADS];
28
29     // crate a thread for each guess
30     for (int i = 0; i < NUM_THREADS; i++){
31         thread_args[i] = i;
32         pthread_create(&threads[i], NULL, findAndExport, &thread_args[i]);
33     }
34
35     //wait for each thread to complete
36     for (size_t i = 0; i < NUM_THREADS; i++) {
37         pthread_join(threads[i], NULL);
38     }
39
40
41     return 0;
42 }
43
```

```
44 // find the root of the error function near the n-th guess and save the
    generated wave function
45 void *findAndExport(void *argument){
46     // get the index of the thread
47     const int n = *((int *)argument);
48
49     // create array for the wave Function
50     double phi[N];
51
52     // bisection to find the root
53     double E0 = guess[n]-UNCERT_GUESS;
54     double E1 = guess[n]+UNCERT_GUESS;
55     double Ex, tmp;
56     do {
57         Ex = (E1+E0)/2.0;
58
59         integ(Ex, &phi);
60         tmp = phi[N-1];
61         integ(E0, &phi);
62
63         if(tmp*phi[N-1]>0)
64             E0=Ex;
65         else
66             E1=Ex;
67     } while (fabs((E1-E0)/E0)>UNCERT_END);
68
69     printf("guess: %.2f E: %f\n", guess[n], E1);
70
71     // normalizing
72     double norm = 0;
73     for(int i=0; i<N; i++){
74         norm += phi[i]*phi[i]*H;
75     }
76     norm = sqrt(norm);
77     for(int i=0; i<N; i++){
78         phi[i] /= norm;
79     }
80
81     // exporting
82     FILE *file;
83     char filename[210];
84     snprintf(filename, 10, "out%02d.dat", n);
85     printf("saving in: %s\n", filename);
86     file = fopen(filename, "w+");
```

```
87  fprintf(file, "%f\n", E1);
88  for(int i=0; i<N; i+=EXPORT_STEPS){
89      fprintf(file, "%f %f\n", i*H, phi[i]);
90  }
91  fclose(file);
92
93  // end the task successfully
94  return 0;
95 }
96
97 // integrating the wave function
98 void integ(double E, double *phi){
99
100     double y1 = 0, y2 = H;
101     for(int i=0; i < N; i++){
102         numerov(&y1, &y2, i*H, E);
103         phi[i] = y2;
104     }
105 }
106
107 // numerov step
108 void numerov(double *y1, double *y2, double x, double E){
109
110     double y = 2* *y2*(1 - H*H*5/12*(E-V(x))) - *y1*(1 - H*H/12*(E-V(x-H)))
111     ;
112     *y1 = *y2;
113     *y2 = y;
114 }
115
116 double V(double x){
117     return x;
118 }
```