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

$$\text{Numerisch:} \qquad -0.039788 \tag{1}$$

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

b)

Aufgabe 6 - Shooting-Methode

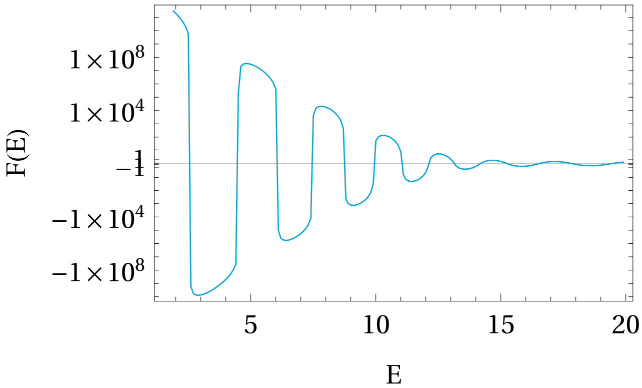

Abbildung 1: Fehlerfunktion für verschiedene Energien

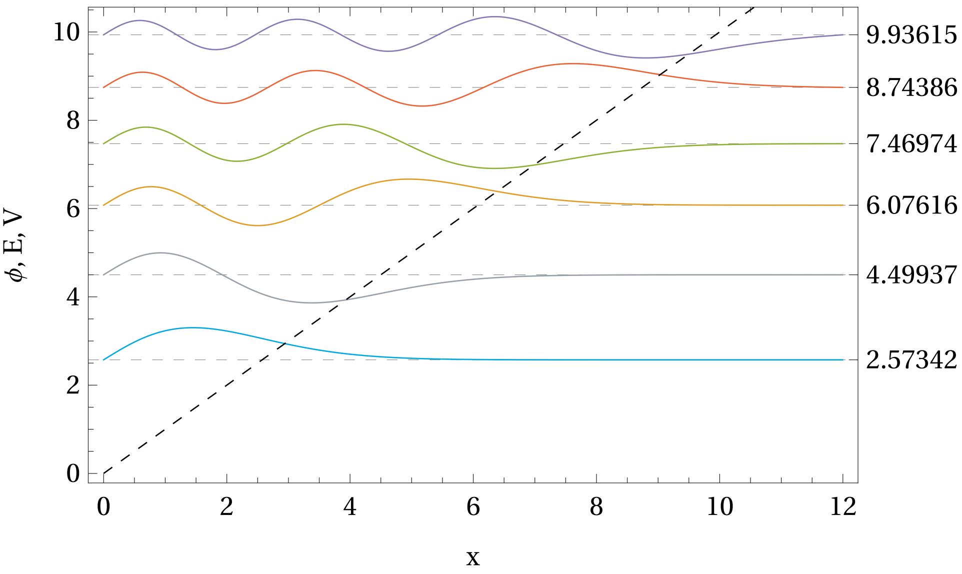

Abbildung 2: Potential mit verschiedenen Wellenfunktionen

Code

5 a)

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

```
#include <stdio.h>
#include <math.h>


#define H 0.001
#define rstart 100


double rho(double r);
void numerov(double* y1, double* y2, double r);


int main(){

    double y1=0, y2 = 0;

    for(double r = rstart; r > 0; r -= H){
        numerov(&y1, &y2, r);
        printf("%f\n", y2);
    }

    // Ergebnis:                        -0.039789
    // Analytisch: -1/(8 pi) \approx    -0.0397887

    return 0;
}


double rho(double r){
    double wert = exp(-r)/(8 * M_PI);
    return wert;
}

void numerov(double* y1, double* y2, double r){
    double y = 2 * *y2 - *y1 - H*H/12*(rho(r+H)+10*rho(r)+rho(r-H));
    *y1 = *y2;
    *y2 = y;
}
```

6

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

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define H 1.e-4
#define XMAX 15.

void integ(double E);

void numerov(double *y1, double *y2, double x, double E);

double V(double x);

FILE *file;

int main(){

    file = fopen("errorFkt.dat", "w+");


    for(double E = 0.0; E < 20; E += 0.1){
        integ(E);
    }

    //integ(5.0);



    fclose(file);

    return 0;
}


void integ(double E){

    double y1 = 0, y2 = H;
    for(double x = XMAX; x > 0; x -= H){
        numerov(&y1, &y2, x, E);
        //fprintf(file, "%f %f \n", x, y1);
    }
    fprintf(file, "%f %f \n", E, y2);
}

void numerov(double *y1, double *y2, double x, double E){

    double y = 2* *y2*(1 - H*H*5/12*(E-V(x))) - *y1*(1 - H*H/12*(E-V(x-H)));
    *y1 = *y2;
    *y2 = y;
}


double V(double x){
    return x;
}
```

Listing 3: . . .

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>

#define H .5e-5
#define X_MAX 12.
#define N ((int) (X_MAX/H))
#define EXPORT_STEPS 1000

#define UNCERT_END .5e-15

#define UNCERT_GUESS 0.5
//double guess[] = {2.5, 4.5, 6, 7.5, 8.5, 10, 12, 14, 16, 18.5};
double guess[] = {2.5, 4.5, 6, 7.5, 8.5, 10};

void integ(double E);
void numerov(double *y1, double *y2, double x, double E);
void findAndExport(int n);
double V(double x);

// has to be global else segmentation fault on stack (2 MB) with (8byte/double * N doubles)
double phi[N];

int main(){
  // loop over guesses
  for (size_t i = 0; i < sizeof(guess)/sizeof(guess[0]); i++){
    findAndExport(i);
  }

  return 0;
}

// find the root of the error function near the n-th guess and save the generated wavefunction
void findAndExport(int n){
  // bisection to find the root
  double E0 = guess[n]-UNCERT_GUESS;
  double E1 = guess[n]+UNCERT_GUESS;
  double Ex, tmp;
  do {
    Ex = (E1+E0)/2.0;

    integ(Ex);
    tmp = phi[N-1];
    integ(E0);

    if(tmp*phi[N-1]>0)
      E0=Ex;
    else
      E1=Ex;
  } while (fabs((E1-E0)/E0)>UNCERT_END);

  printf("guess: %.2f E: %f\n", guess[n], E1);

  // normalizing
  double norm = 0;
  for(int i=0; i<N; i++){
    norm += phi[i]*phi[i]*H;
  }
  norm = sqrt(norm);
  for(int i=0; i<N; i++){
    phi[i] /= norm;
  }

  // exporting
  FILE *file;
  char filename[210];
  snprintf(filename, 10, "out%02d.dat", n);
  printf("saving in: %s\n", filename);
  file = fopen(filename, "w+");
  fprintf(file, "%f\n", E1);
  for(int i=0; i<N; i+=EXPORT_STEPS){
    fprintf(file, "%f %f\n", i*H, phi[i]);
  }
  fclose(file);
}

// integrating the wavefunction
void integ(double E){

  double y1 = 0, y2 = H;
  for(int i=0; i < N; i++){
    numerov(&y1, &y2, i*H, E);
    phi[i] = y2;
  }
```

```
}

// numerov step
void numerov(double *y1, double *y2, double x, double E){

  double y = 2* *y2*(1 - H*H*5/12*(E-V(x))) - *y1*(1 - H*H/12*(E-V(x-H)));
  *y1 = *y2;
  *y2 = y;
}


double V(double x){
  return x;
}
```