Projektspezifikation – Abschlussprojekt für Wissenschaftliches Programmieren

Bálint Aradi
http://www.bccms.uni-bremen.de/cms/people/b_aradi

SoSe 2017

Spätester Abgabetermin: 20.09.2017

1 Die zeitunabhängige Schrödingergleichung

Das Verhalten makroskopischer Teilchen wird in der Quantenmechanik durch eine komplexe Wellenfunktion $\Psi(x)$ beschrieben, die von der Ortskoordinate als Variable abhängt. Das Betragsquadrat dieser Wellenfunktion gibt die Aufenthaltswahrscheinlichkeitsdichte des Teilchens an einem bestimmten Ort an. Im eindimensionalen stationären Fall, wenn also das äußere Potential nicht von der Zeit abhängt, gehorcht die Wellenfunktion der Schrödingergleichung

$$-\frac{1}{2M}\Psi''(x) + V(x)\Psi(x) = E\Psi(x), \tag{1}$$

wobei $\Psi''(x)$ die zweite Ableitung der Wellenfunktion nach dem Ort ist. Die Größen $V(x)$, M und E geben jeweils das Potential bzw. die Masse und die möglichen Energien des Teilchens an. Da das Betragsquadrat der Wellenfunktion die Aufenthaltswahrscheinlichkeitsdichte ist, muss das Integral der Wellenfunktion auf den gesammten Raum eins ergeben, d.h.

$$\int |\Psi(x)|^2 \, dx = 1. \tag{2}$$

Die Funktion $\Psi(x)$ sollte stetig und stetig differenzierbar sein.

Da es die Gleichungen wesentlich vereinfacht, werden diese in dieser Beschreibung in atomaren Einheiten aufgeschrieben. Bei diesem Einheitssystem wird die Masse in Elektronenmasse (m_e), die Ladung in Einheitsladung (e), die Wirkung in planckscher Konstante durch 2π ($\hbar$) und die Dielektrizität in $1/(4\pi\epsilon_0)$ angegeben, wobei ϵ_0 die Dielektritizitätskonstante ist. Es gilt also die

Relation

$$m_e = e = \hbar = 1. \tag{3}$$

Die Größen in diesem Einheitsystem werden als dimensionslos betrachtet. Die Einheit der Energie ist Hartree, das 27,211385 eV entspricht. Die Einheit der Länge ist Bohr (bohrscher Radius), das $0{,}529177 \times 10^{-10}$ m entspricht.

2 Numerisches Verfahren

Ein möglicher Weg zur Lösung der eindimensionalen zeitunabhängigen Schrödingergleichung (1) ist die Diskretisierung dieser Differentialgleichung. Die Wellenfunktion $\Psi(x)$ und das Potential $V(x)$ wird auf einem äquidistanten Punktgitter dargestellt:

$$\Psi_i = \Psi(x_i) \qquad \text{und} \qquad V_i = V(x_i) \qquad i = 1, ..., N, \tag{4}$$

wobei x_i die x-Koordinate der Gitterpunkte angibt, und N die Anzahl der Punkte ist. Die Ableitung des Potentials wird mit der Methode der finiten Differenzen gebildet:

$$\Psi_i' = \Psi'(x_i) \approx \frac{\Psi_{i+1} - \Psi_i}{\Delta}, \tag{5}$$

$$\Psi_i'' = \Psi''(x_i) \approx \frac{\Psi_{i+1} - 2\Psi_i + \Psi_{i-1}}{\Delta^2}, \tag{6}$$

wobei Δ den Abstand der Gitterpunkte angibt.

Mit Hilfe dieser Diskretisierung lässt sich die Schrödingergleichung folgendermaßen umformen:

$$-\frac{1}{2M}\Psi_i'' + V_i\Psi_i = E\Psi_i \tag{7}$$

$$-\frac{1}{2M}\left(\frac{\Psi_{i+1} - 2\Psi_i + \Psi_{i-1}}{\Delta^2}\right) + V_i\Psi_i = E\Psi_i \tag{8}$$

$$-\frac{1}{2M\Delta^2}\Psi_{i-1} + \left(\frac{1}{M\Delta^2} + V_i\right)\Psi_i - \frac{1}{2M\Delta^2}\Psi_{i+1} = E\Psi_i. \tag{9}$$

Das Problem der Eigenfunktionsuche eines Operators wird somit in das viel einfachere Problem der Eigenvektorsuche einer Matrix transformiert. Legt man nämlich die Randbedingungen so fest, dass die Wellenfunktion an den Rändern des diskretisierten Raumbereiches verschwindet ($\Psi_0 = \Psi_{N+1} = 0$), so kann man Gleichung (9) für alle Punkte ($i = 1, ..., N$) gleichzeitig in der

Matrixform

$$\begin{pmatrix} a+V_1 & -\frac{1}{2}a & 0 & \dots & 0 \\ -\frac{1}{2}a & a+V_2 & -\frac{1}{2}a & \dots & 0 \\ 0 & -\frac{1}{2}a & a+V_3 & \dots & \vdots \\ \vdots & \vdots & & \ddots & -\frac{1}{2}a \\ 0 & 0 & \dots & -\frac{1}{2}a & a+V_N \end{pmatrix} \begin{pmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \end{pmatrix} = E \begin{pmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \end{pmatrix} \quad (10)$$

aufschreiben, wobei zur Abkürzung die Größe $a = \frac{1}{M\Delta^2}$ eingeführt wurde.

Die quadratische Matrix in dieser Eigenwertgleichung hat dank der gewählten Randbedingungen eine tridiagonale Struktur, da sie nur in der Hauptdiagonale und in der ersten oberen und in der ersten unteren Nebendiagonale von Null verschiedene Elemente enthält. Ferner ist diese Matrix auch symmetrisch. Für Eigenwertprobleme dieser Art gibt es sehr effiziente Algorithmen, die in diversen mathematischen Bibliotheken (z.B. LAPACK) implementiert sind.

Die gewählten Randbedingungen haben zur Folge, dass das Programm kein freistehendes System modelliert, sondern ein solches, das zwischen zwei unendlich hohen Barrieren eingeschlossen ist. Man muss sich deshalb immer vergewissern, dass die Randbedingungen das modellierte System nicht beeinflussen. Klingen die Wellenfunktionen schon weit von den Rändern von sich ab, stimmen die Ergebnisse mit jenen für ein freistehendes System überein. Wird jedoch das Verschwinden der Wellenfunktionen an den Rändern durch die Randbedingungen erzwungen, werden die Ergebnisse von denen eines freistehenden Systems abweichen.

3 Aufgabe

Jeweils zwei (eventuell in Ausnahmefällen *nach vorheriger Absprache* drei) Personen sollten ein Programm zur Lösung der eindimensionalen zeitunabhängigen Schrödingergleichung mit der oben beschriebenen Methode erstellen. Es muss das Repository abgegeben werden, das die Programmentwicklungsgeschichte enthält. Zusätzlich findet ein kurzes Auswertungsgespräch statt, bei dem die Entwickler eventuelle Fragen zu Ihrem Programm beantworten müssen. (Unabhängig davon, welche Teile des Projektes von wem umgesetzt wurden, müssen alle Entwickler mit allen Teilen des Programmes vertraut sein.)

3.1 Programmfunktionalität

- Das Programm soll die eindimensionale zeitunabhängige Schrödingergleichung in einem frei wählbaren Bereich lösen.
- Die Anzahl der zur Diskretisierung verwendeten Punkte wird vom Benutzer festgelegt. Ebenso legt der Benutzer die Masse des Teilchens und das externe Potential (V) fest.
- Das Potential wird durch Stützpunkte (beliebiger Anzahl) gegeben. Der Benutzer darf frei auswählen, ob zwischen den Stützpunkten stückweise linear interpoliert wird oder (mit Berücksichtigung aller Stützpunkten) eine Polynominterpolation verwendet wird. *Wird das Programm von drei Personen entwickelt, muss auch eine Interpolation mit natürlichen kubischen Splines (natural csplines) erlaubt sein.*
- Das aus den Stützpunkten erzeugte diskretisierte Potential soll in der Datei `discrpot.dat` in XY-Format gespeichert werden, sodass es mit graphischen Tools (z.B. xmgrace, gnuplot, etc.) dargestellt werden kann. Das XY-Format ist wie folgt:

  ```
  x1 y1
  x2 y2
  :
  ```

- Der Benutzer sollte frei auswählen können, welche Eigenwerte und Wellenfunktionen gespeichert werden sollten (von bis). Die ausgewählten Eigenwerte sind in der Datei `energies.dat` (ein Eigenwert pro Zeile), die Wellenfunktionen in `wfuncs.dat` zu speichern. Zusätzlich sollten zu den ausgewählten Wellenfunktionen die zugehörenden Energiewerte addiert werden, und die mit Energie verschobenen Wellenfunktionen in `ewfuncs.dat` gespeichert werden. Die Dateien `wfuncs.dat` und `ewfuncs.dat` sollten NXY-Format haben, sodass sie mit graphischen Tools (z.B. xmgrace, gnuplot, etc.) dargestellt werden können:

  ```
  x1 wf1(x1) wf2(x1) wf3(x1) ...
  x2 wf1(x2) wf2(x2) wf3(x2) ...
  :
  ```

  Die entsprechenden Zeilen sollten Sie auf jeden Fall mit expliziter Formattierung – also *nicht* mit `write(*,*)` – ausgeben, da ansonsten bei manchen Compilern[1] ein Zeilenumbruch entsteht.

[1] Zum Beispiel beim Intel-Compiler, der zur Auswertung des Projektes eingesetzt wird!

- Die Benutzerangaben sollten aus der Datei `schrodinger.inp` eingelesen werden. Alle Größen in der Ein- und Ausgabe sollten in atomaren Einheiten angegeben sein.

3.2 Technische Angaben

- Das Programm muss in Fortran 2003 geschrieben werden und dem Fortran 2003 Standard entsprechen.

- Das Projekt muss in mehrere Module aufgeteilt werden. (Eine mögliche Aufteilung: Hauptprogramm, Modul für formattierte Ausgabe der Ergebnisse, Modul für Rechen- und Interpolationsroutinen, Modul für Präzisionseinstellungen.)

- Die verschiedenen Stufen der Programmentwicklung müssen mit dem Versionsverwaltungssystem Git festgehalten werden. Beide Entwickler müssen gleich nach dem Projektstart eigene Zweige (branches) anlegen, und ihre Veränderungen immer in den eigenen Zweig einchecken. Die Zweige müssen dann so oft wie erforderlich synchronisiert werden (merge). *Abzugeben ist nicht nur das fertige Programm, sondern auch eines der beiden git-Repositories*, das den Code in seinem Endzustand, mindestens 4-5 Revisionen (mit mindestens ein *merge*) enthalten muss.

- Das Programm muss ein Makefile enthalten, in dem die Abhängigkeiten der Projektteile korrekt abgebildet sind. Das Projekt muss mit der Ausführung des Befehls `make` compiliert (und gelinkt) werden können. Es sollten auch die Phony-Ziele `clean` und `realclean` (sinngemäß) implementiert werden.

- Die Schnittstellen (Module, Unterprogramme, etc.) sind mit Hilfe von Doxygen zu dokumentieren. Die Konfigurationsdatei für Doxygen muss auch im Repository enthalten sein. Mit Doxygen sollte aus dem Projekt automatisch eine HTML-Dokumentation erstellt werden können.

- Die Diagonalisierung der tridiagonalen Matrix muss mit einer LAPACK-Routine gelöst werden. Es sollte entweder manuell ein Interface für die entsprechende f77-LAPACK-Routine erstellt oder LAPACK95 benutzt werden. *Wird das Programm von drei Personen entwickelt, muss die Lösung des tridiagonalen Gleichungssystems für die Bestimmung der Splineparameter auch mit einer LAPACK-Routine gelöst werden.*

- Das Projekt muss auch ein `README` enthalten, in dem die wesentlichen Informationen zur Compilierung und Benutzung beschrieben werden.
- Das Projekt muss für die unten angegebenen Beispiele die unten angegebenen Resultate liefern. Diese sollten gleichzeitig als Autotest-Beispiele dienen, sodass mit `make test` nach dem Compilieren getestet werden kann, ob das erstellte Programm so funktioniert, wie das von den Programmierern vorgesehen war.

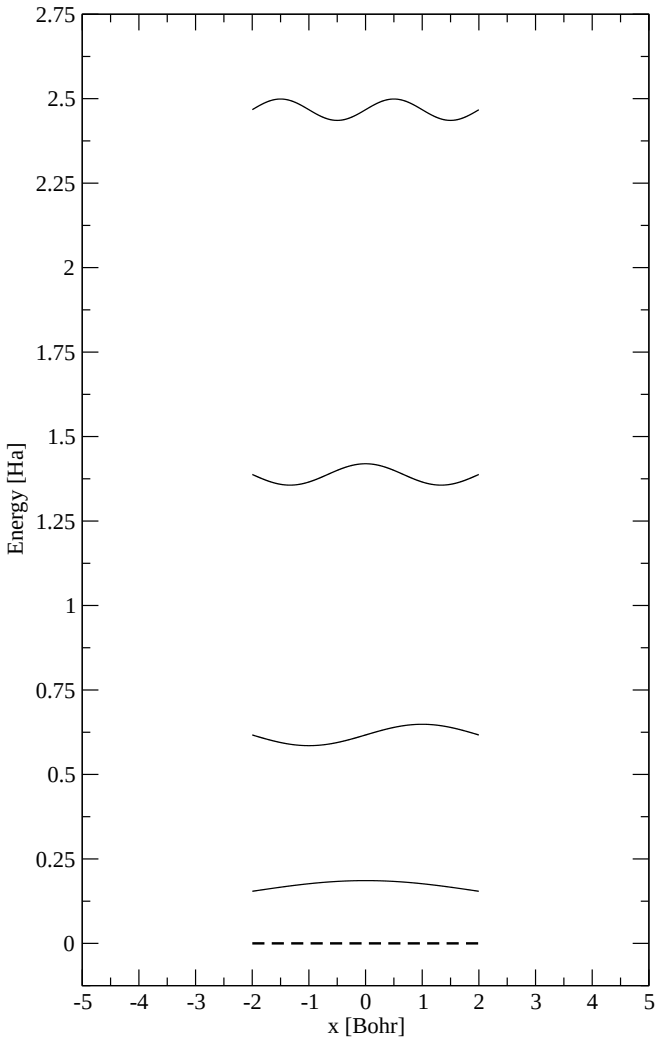
3.3 Anwendungsbeispiele

Nachfolgend werden Anwendungsbeispiele angegeben, jeweils mit dem Inhalt der Eingabedatei `schrodinger.inp` und der graphischen Darstellung der Ergebnisse im `discrpot.dat` und `ewfuncs.dat`. *Das abgegebene Programm muss die aufgeführten Eingabeinhalte, so wie sie hier stehen, lesen können*[2] *und aus den Ergebnissen müssen die gezeigten Graphiken mit* `xmgrace` *oder* `gnuplot` *erzeugt werden können.*

[2] Bitte beachten Sie, dass kein spezieller Parser für die Behandlung der Kommentare in den Beispielen nötig ist, sofern Sie die Daten in den kommentierten Zeilen zeilenweise einlesen.

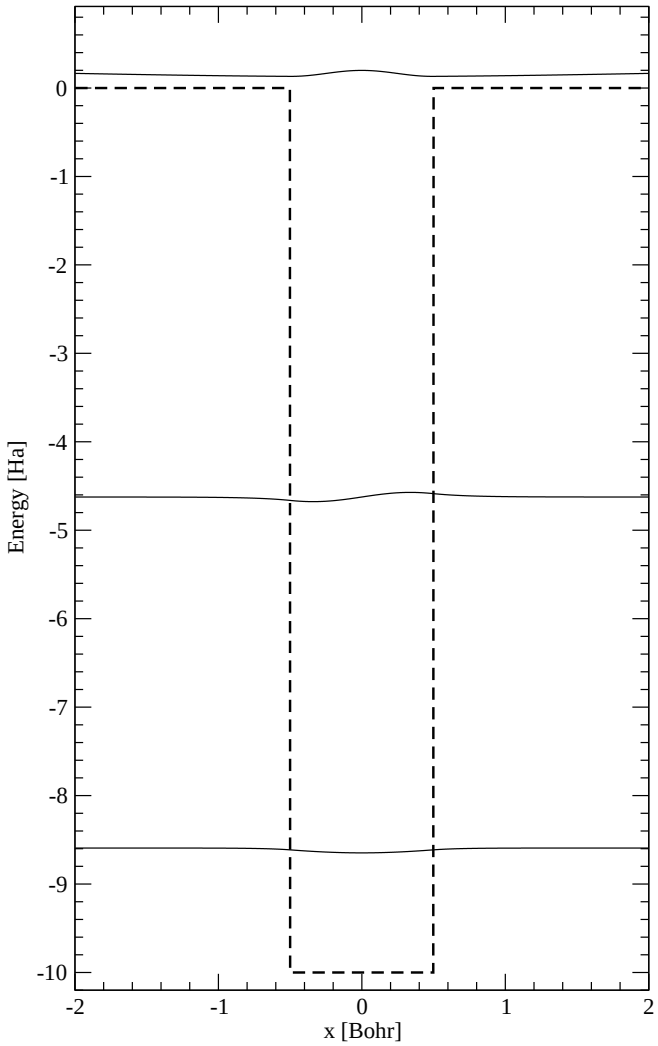
3.3.1 Unendlich tiefer Potentialtopf

```
2.0             # mass
-2.0 2.0  1999  # xMin xMax nPoint
1 15            # first and last eigenvalue to print
linear          # interpolation type
2               # nr. of interpolation points and xy declarations
-2.0 0.0
 2.0 0.0
```



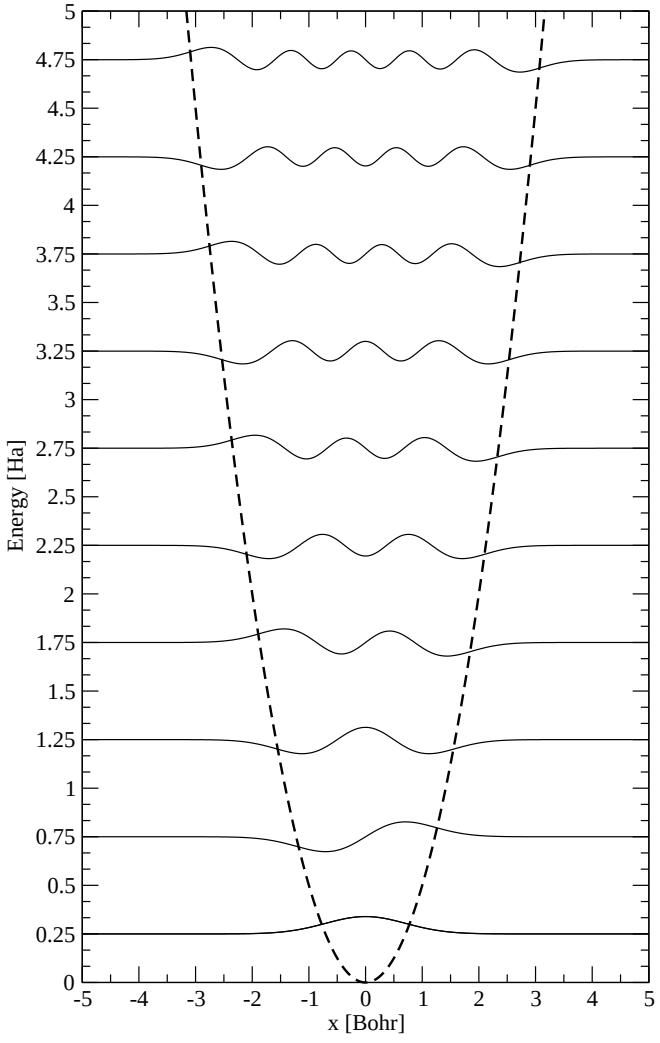
3.3.2 Endlich tiefer Potentialtopf

```
2.0             # mass
-2.0 2.0  1999  # xMin xMax nPoint
1 15            # first and last eigenvalue in output
linear          # interpolation type
6               # nr. of interpolation points and xy declarations
-2.0  0.0
-0.5  0.0
-0.5 -10.0
 0.5 -10.0
 0.5  0.0
 2.0  0.0
```



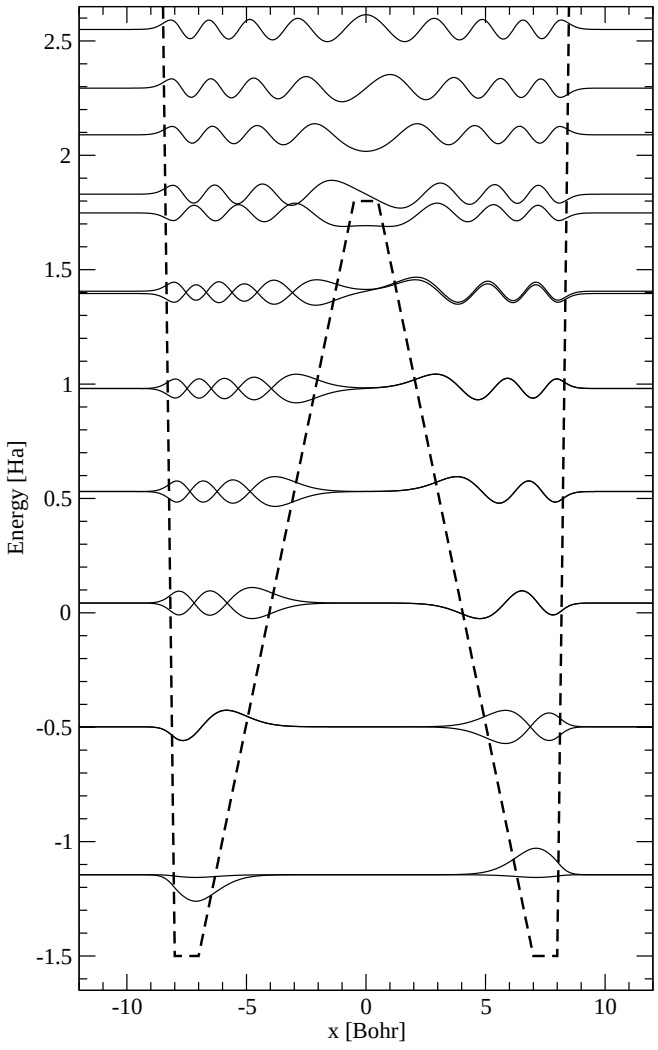
3.3.3 Harmonischer Oszillator

```
4.0                # mass
-10.0 10.0  1999 # xMin xMax nPoint
1 15               # first and last eigenvalue in output
polynomial         # interpolation type
3                  # nr. of interpolation points and xy declarations
-1.0  0.5
 0.0  0.0
 1.0  0.5
```



3.3.4 Doppelter Potentialtopf (linear)

```
2.0                 # mass
-20.0 20.0 1999     # xMin xMax nPoint
1 30                # first and last eigenvalue to print
linear              # interpolation type
8                   # nr. of interpolation points and xy declarations
-20.0    100.0
 -8.0     -1.5
 -7.0     -1.5
 -0.5      1.8
  0.5      1.8
  7.0     -1.5
  8.0     -1.5
 20.0    100.0
```



3.3.5 Doppelter Potentialtopf (spline)

(Nur wenn das Program zu dritt entwickelt wird.)

```
4.0              # mass
-20.0 20.0  1999 # xMin xMax nPoint
1 30             # first and last eigenvalue to include in the output
cspline          # interpolation type
5                # nr. of interpolation points and xy declarations
-20.0 35.0
-10.0  0.0
  0.0  2.0
 10.0  0.0
 20.0 35.0
```