Notizen Vertiefungspraktikum

# allg./Aufgaben/Fragen

1. mass & force auf 1
2. technische Parameter
   1. mehr Stützpunkte und/oder mehr Funktionen
   2. besonders bei kleinen Abständen = ?
3. Variation von mass & force
   1. bei konstanten testen später

! bei odist == 0 können Probleme durch unendliche Werte entstehen (s. )

# Effekte

## Quadrupol-Quadrupol

## Coulomb (Ladung)

🡪 besonders bei weiten Abständen entscheidend

## Quadrupol-Ladung

## Dispersion

# main\_qho1d\_matrix.jl-Datei

* „ho“ = harmonischer Oszillator
* 1-dimensional

|  |  |
| --- | --- |
| **Code**  *geändert* | **Erläuterung** |
| using LinearAlgebra | Bibliotheken laden;  lineare Algebra für Matrizen & deren Inversion  z.B. dot()-Funktion |
| using FastGaussQuadrature | Tool für Funktionsintegration |
|  |  |
| # physical parameters: | „Physik“ |
| pdist = 0.0 | # parallel distance between the oscillators (in a.u.) |
| odist = 20.0 | # orthogonal distance between the oscillators (in a.u.)  ! kann bei 0 Probleme erzeugen |
| mass = 1.0 | # mass of oscillating particle (in a.u.)  für 1 e- |
| force =1.0 | # force constant (in a.u.) |
|  |  |
| # technical parameters: |  |
| nbpoints = 200 # number of Gauss-Hermite integration points | Punkte der Gewichtung  Integration bzw. Funktionen ca. auf doppelten Wert von nbpoints genau |
| nbfcts = 50 # number of QHI eigenfunctions to be used | **Anzahl** der benutzten Funktionen für die Summe der Gaußquadratur (! höchste Quantenzahl wird daher (beginnend bei 0) die 49 sein) |
|  |  |
| points, weights = gausshermite(nbpoints) | # obtain Gauss-Hermite quadrature points and weights  Ermitteln der Hermite-Polynome mittels der Bibliothek und direktes Abspeichern in 2 Vektoren  Genauigkeit: 16 Stellen  z.B.  points = [-19.339248667911406, -18.82289598056473]  weights = [2.2290934962740562e-163, 6.171630370185669e-155]  Funktionswerte an den Punkten & Gewichtungsfaktoren |
|  |  |
| # calculate (normalized) Hermite polynomials at quadrature points: |  |
| include("norm\_hermite\_polynomials\_at\_x.jl") | Datei eingefügt (s. dort) |
| hvalues=norm\_hermite\_polynomials\_at\_x.(nbfcts,points) | konkreter Funktionswert!? = Summe? |
|  |  |
| # calculate potential at quadrature points: |  |
| len = 1/sqrt(force\*mass) | zur Normierung |
| pot(x) = 1/sqrt(odist^2+(pdist-len\*x)^2) | damit x einheitenlos sein kann, muss len zur Normierung einbezogen werden 🡪 bringt Abstand x auf a.u.  Funktionsdefinition: potentielle Coulomb-Energie = 1/r  Abstand r durch horizontale & vertikale Verschiebung der Oszillatoren; Einbezug der Auslenkung nötig für pdist |
| potval=pot.(points) | . = Schleife 🡪 über alle Werte im Array points 🡪 Berechnen der Coulomb-Energie und speichern in Array potval |
| file\_pot=open("pot\_at\_int\_points.dat", "w") | Datei öffnen zum Speichern der Energien an den Integrationspunkten |
| for i = 1:nbpoints  println(file\_pot, len\*points[i], " ", potval[i], " ", potval[i]+0.5\*force\*(len\*points[i])^2)  end | Schleife über alle Gewichtspunkte nbpoints  Schreiben in die Datei: 3 Spalten mit Zahlen  z.B.:  -19.339248667911406 0.04916602458269113 187.05243554423922  „un“normierte Punkte, berechnete Werte in potval = pot. Energie durch Coulomb, pot. Energie (Coulomb & Oszillation () |
| close(file\_pot) | File schließen (! selbe Datei für alle Rechnungen; wird überschrieben) |
|  |  |
| # build potential part of Hamilton matrix: | Matrix aufstellen: |
| perpotmat = zeros(nbfcts,nbfcts) | nbfcts = 50 (s.o.) = Gaußfunktionsanzahl – bleibt unverändert  Erstellen einer 50x50-Matrix mit 0en:   * Typ perpotmat = Array of Floats (64) * Länge = 2500 * perpotmat[i] 🡪 Länge 1, Float64 = 0.0 * Abtrennung per ; bzw. Leerzeichen |
| for i = 1:nbfcts  for j = 1:i  integrand = zeros(nbpoints)  for k = 1:nbpoints  integrand[k] = hvalues[k][i] \* hvalues[k][j] \* potval[k]  end  perpotmat[i,j] = dot(weights,integrand)  perpotmat[j,i] = perpotmat[i,j]  end  end | Schleife über alle Anzahlen der Funktionen & je von 1 bis zur aktuellen Funktionszahl  Anlegen eines 0-Vektors der Länge nbpoints (Anzahl Gewichtspunkte) ; z.B. [0.0, 0.0, 0.0]  Schleife über alle Punkte nbpoints:   * Ersetzen der Werte im Vektor integrand * z.B. [0.2535974307898293, 0.5641895835477557, 2.5103557649808534] * hvalues (s.o.) für Bra & Ket; potval (s.o.) für Coulomb-Energie   Integrand-Werte in Matrix perpotmat eingesetzt als Punktprodukt mit dem Gewichtsfaktorenvektor   1. eine Hälfte 2. Matrix symmetrisieren |
|  |  |
| # build full Hamilton matrix: |  |
| hamiltonmat = copy(perpotmat) | Kopie der perpot-Matrix |
| omega = sqrt(force/mass) | Kreisfrequenz |
| for i = 1:nbfcts  hamiltonmat[i,i] = hamiltonmat[i,i] + ((i-1)+0.5)\*omega  end | Schleife über alle Gewichtspunkte  Eingeben von Werten (Ersetzen) in die Hamiltonmatrix:   * nur Diagonalwerte verändert, da 🡪 -Term * i-1, da Quantenzahlen bei 0 beginnend, Indizes der Matrix bei 1 * Summe aus Coulomb-Energie & kin. Energie durch Schwingung = [a.u.] |
|  |  |
| # determine eigenvalues of Hamilton matrix: | Eigenwerte bestimmen  Vektor der Eigenwerte = Energien (der Art [1, 2]) |
| energies = eigvals(hamiltonmat) |
| file\_en=open("energies.dat", "w")  for i = 1:nbfcts  println(file\_en, i-1, " ", energies[i])  end  close(file\_en) | Speichern der Energien in Abhängigkeit von ihrer Quantenzahl (= i-1) |
|  |  |
| # determine eigenvectors of Hamilton matrix: | Eigenvektoren, z.B.  [0.8675654984443372 -0.3586774066955762; -0.4895038222578973 -0.5686821855318818; 0.07342253966322135 0.6629913600200636] |
| coefficients = eigvecs(hamiltonmat) |

# Gaußsches Quadraturverfahren

Wikipedia

Douglas etc.

Genauigkeit bis auf 2\*n

<https://en.wikipedia.org/wiki/Gauss%E2%80%93Hermite_quadrature>

# Linux

* top für laufende Prozesse
* ping lewis 🡪 IP: 132.252.87.28
* hostname lewis
* Judith; minischwein