Error control in scientific modelling (MATH 500, Herbst)

Sheet 2: Morse oscillator (10 P)

To be handed in via moodle by 05.10.2023

```
using PlutoTeachingTools
```

Introduction

In this exercise we want to explore the different error contributions based on the Morse oscillator. The Morse oscillator is a simple quantum-mechanical model for a chemical bond. Even though nowadays more accurate models exist, it is still used due to its simplicity. One remarkable feature of the model is that it is able to produce a realistic description of chemical bonding. At the same time the associated Schrödinger equation still has an analytic solution — even though we will not attempt this computation here: See wikipedia if you are curious.

In this exercise we will consider the model in one dimension, i.e. our computational domain is simply the real line $\mathbb{R}$. The Hamiltonian of the Morse potential has the usual form of

$$H^M = -\frac{1}{2}\Delta + V^M,$$

where V^M is the Morse potential energy function, which for all $x \in \mathbb{R}$ is defined as

$$V^M(x) = D\left(1 - \exp\left(-\frac{\omega}{\sqrt{2D}}\left(x - x_0\right)\right)\right)^2$$

with D is the dissociation energy and ω^2 the force constant and x_0 the equilibrium bond length. For our computations we consider the parameters

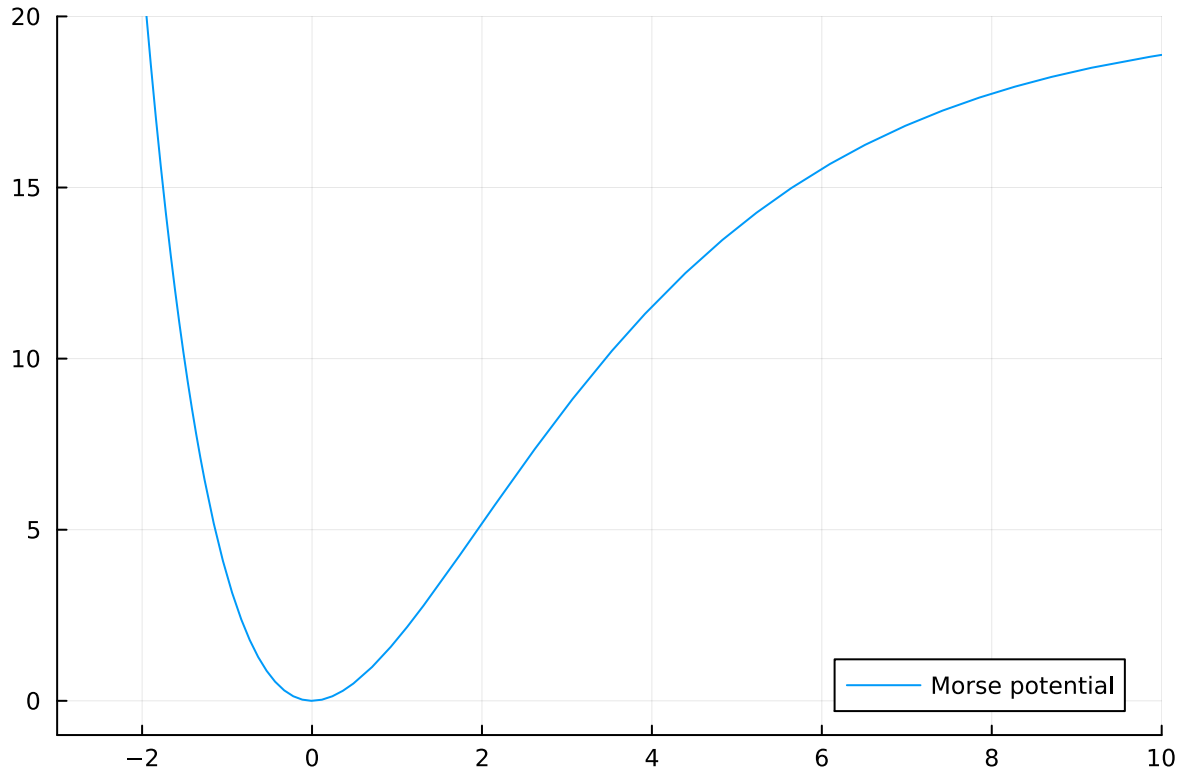
```
begin
    D  = 20.0
    ω  = 2.25
    x0 = 0.0
end;
```

for which the potential looks as such

```
using Plots
```

```
Vm (generic function with 1 method)
```

```
1 Vm(x) = D * (1 - exp( -ω / sqrt(2D) * (x - x0) ) )^2
```




```
1 plot(Vm, xlims=(-3, 10), ylims=(-1, 20), label="Morse potential")
```

Following our discussion about the Schrödinger equation in the lectures, we are interested in the eigenvalues and eigenfunctions of this Hamiltonian, i.e. the $(E_n^M, \varphi_n^M) \in \mathbb{R} \times L^2(\mathbb{R})$ pairs satisfying

$$H^M \varphi_n^M = E_n^M \varphi_n^M.$$

As mentioned above, this system can actually be solved *analytically* with its eigenvalues given by

$$E_n^M = \omega\left(n + \frac{1}{2}\right) - \omega\chi\left(n + \frac{1}{2}\right)^2 \qquad n = 0, 1, 2, \ldots, \lfloor \tfrac{2D}{\omega} - \tfrac{1}{2} \rfloor$$

where

$$\chi = \frac{\omega}{4D}$$

is the so-called anharmonicity constant.

Exercise 1 (1 + 1 P)

While the Morse oscillator provides already a relatively simple model for a chemical bond, a yet even simpler model is often sufficent: The Harmonic oscillator. Its potential simply takes the form

$$V^H(x) = \frac{1}{2}\omega^2(x - x_0)^2.$$

Again we are able to find analytical eigenpairs (E_n^H, φ_n^H) of the Harmonic oscillator Hamiltonian

$$H^H = -\frac{1}{2}\Delta + V^H = -\frac{1}{2}\Delta + \frac{1}{2}\omega^2(x - x_0)^2,$$

where in particular

$$E_n^H = \omega\left(n + \frac{1}{2}\right) \qquad \text{for } n = 0, 1, 2, \dots$$

(a) Show that $V^H(x)$ is the leading-order (in $x - x_0$) term of $V^M(x)$, i.e. that the Harmonic osillator is indeed a simplified version of the Morse oscillator.

(a) Solution:

We have

$$V^H(x) = \frac{1}{2}\omega^2(x - x_0)^2$$

and

$$V^M(x) = D\left(1 - \exp\left(-\frac{\omega}{\sqrt{2D}}\,(x - x_0)\right)\right)^2$$

One can take the Taylor expansion of $e^{-\frac{\omega}{\sqrt{2D}}(x - x_0)}$ around x_0:

$$\begin{aligned} V^M \approx& \left(1 - 1 + \frac{\omega}{\sqrt{2D}}(x - x_0) + O((x - x_0)^2)\right)^2 \\ =& \frac{\omega^2}{2D}(x - x_0)^2 + O\left((x - x_0)^4\right). \end{aligned}$$

Therefore,

$$V^M(x) \approx V^H(x) + O\left((x - x_0)^4\right).$$

(b) Show that the ground state (lowest-eigenvalue eigenfunction) is given by

$$\tilde{\varphi}_0^H(x) = \exp\left(-\tfrac{1}{2}\omega(x - x_0)^2\right)$$

and normalise this function with respect to the standard $L^2(\mathbb{R})$ norm

$$\|f\| = \sqrt{\int_{\mathbb{R}} |f(x)|^2 dx}.$$

For the normalised function we will use the symbol φ_0^H in the following.

(b) Solution:

One need to test that the function $\tilde{\varphi}_0^H(x)$ satisfies the following condition: $H^H\tilde{\varphi}_0^H(x) = E_0^H\tilde{\varphi}_0^H(x)$.

$$H^H\tilde{\varphi}_0^H(x) = -\frac{1}{2}\Delta + V^H = -\frac{1}{2}\Delta\tilde{\varphi}_0^H(x) + \frac{1}{2}\omega^2(x-x_0)^2\tilde{\varphi}_0^H(x)$$

with

$$\Delta\tilde{\varphi}_0^H(x) = \frac{d^2}{dx^2}(\exp\left(-\tfrac{1}{2}\omega(x-x_0)^2\right)) = e^{-\frac{1}{2}\omega(x-x_0)^2}(\omega^2(x-x_0)^2 - \omega)$$

Therefore,

$$-\frac{1}{2}e^{-\frac{1}{2}\omega(x-x_0)^2}(\omega^2(x-x_0)^2 - \omega) + \frac{1}{2}\omega^2(x-x_0)^2e^{-\frac{1}{2}\omega(x-x_0)^2} = \frac{1}{2}\omega e^{-\frac{1}{2}\omega(x-x_0)^2}$$

after simplifications:

$$-\omega^2(x-x_0)^2 + \omega + \omega^2(x-x_0)^2) = \omega$$

$$\omega = \omega$$

Which is true.

Then, we compute the norm of the function:

$$\|\tilde{\varphi}_0^H(x)\| = \sqrt{\int_{\mathbb{R}} |\tilde{\varphi}_0^H(x)|^2 dx} = \sqrt{\int_{\mathbb{R}} |e^{-\frac{1}{2}\omega(x-x_0)^2}|^2 dx},$$

where

$$\int_{\mathbb{R}} e^{-\omega(x-x_0)^2} dx = \frac{\sqrt{\pi}}{\omega}.$$

From which the normalizing constant can be deduced, giving:

$$\varphi_0^H(x) = \sqrt{\frac{\omega}{\sqrt{\pi}}}\tilde{\varphi}_0^H(x)$$

Exercise 2 (1 + 1 + 1 + 1 P)

We now want to solve the Harmonic oscillator problem numerically using finite differences. To avoid the infinite computational domain, which cannot be treated within a finite differences approach, we will solve the problem only within $\Omega = [-a, a]$ for $a > 0$.

We perform an N-point finite-difference approximation within Ω, i.e. we split the domain into N intervals of length $h = \frac{2a}{N-1}$. Recall that for this setup the finite-difference Laplacian is given by

```
using LinearAlgebra
```

```
fd_laplacian (generic function with 1 method)
```

```
    function fd_laplacian(N, a;T=Float64)
        h = 2a / (T(N-1))
        diagonal = -2ones(T, N) ./ h^2
        side_diagonal = ones(T, N-1) ./ h^2
        SymTridiagonal(diagonal, side_diagonal)
    end
```

(a) Compute the function values of the potential $V^H(x)$ and the normalised first eigenfunction $\varphi_0^H(x)$ on the grid points of a finite differences approximation with $a = 5$ and $N = 1000$ and use this to plot $V^H(x)$ and $\varphi_0^H(x)$ within our computational domain.

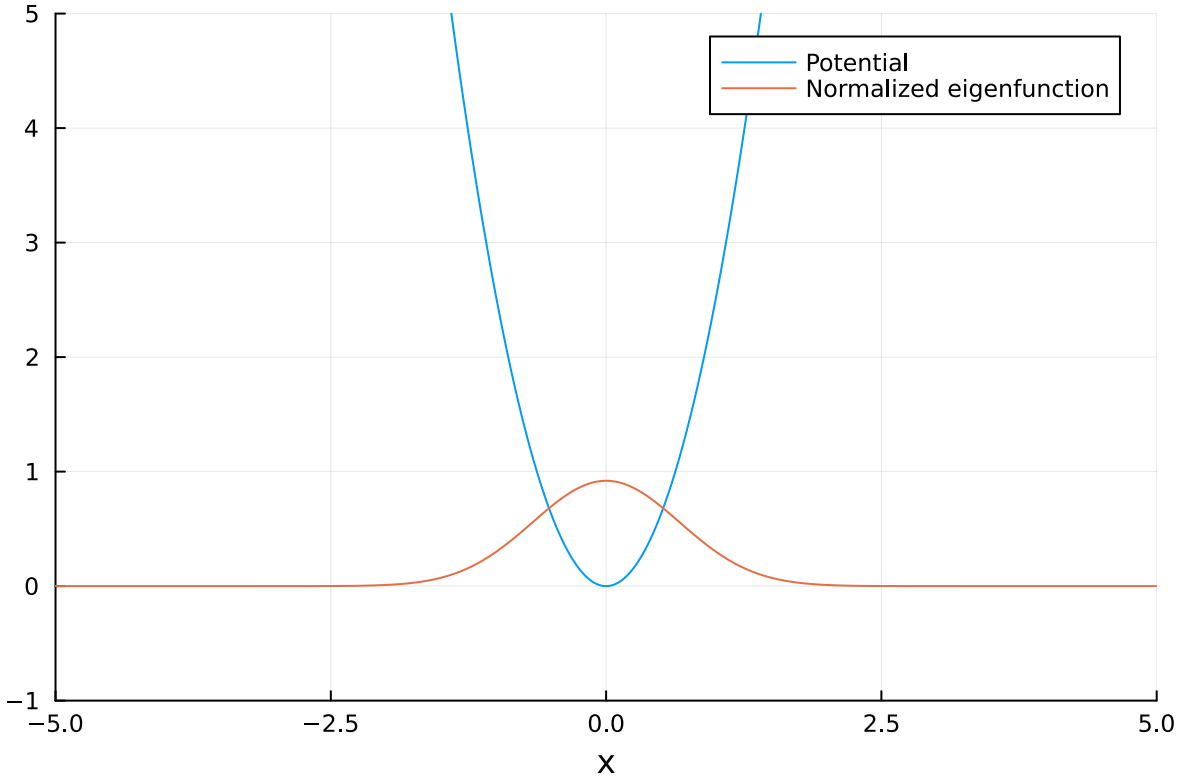
> **Hint**
>
> You can use the `range` function to generate a vector of grid points.

(a) Solution:

```
begin
    N  = 1000
    a  = 5
    Vh(x)= 0.5 * ω ^2 * (x-x0)^2
end;
```

```
begin
    norm_cte= (ω / π)^(0.25)
    phi_0(x)= norm_cte * exp(-0.5 * ω * (x - x0)^2)

    grid_points = range(-a, stop=a, length=N)
    potential_values = [Vh(x) for x in grid_points]
    eigenfunction_values = [phi_0(x) for x in grid_points]
end;
```



(b) Within the finite-differences approximation the potential is just a diagonal term (see the `Diagonal` Julia object). Use Julia's `eigen` function to compute an approximation to the ground state eigenvalue and eigenfunction for $a = 5$ and $N = 1000$.

Hint

To extract the first column of a matrix `A` as a vector, you can use `A[:, 1]`.

```
begin
    fd_Hh = - 0.5 * fd_laplacian(N, a) + 0.5 * ω^2 * Diagonal(grid_points.^2)
    evalues_H, evectors_H = eigen(fd_Hh)
end;
```

```
[1.13097e-14, 2.27604e-14, 3.44933e-14, 4.66524e-14, 5.93856e-14, 7.28465e-14, 8.71963e-14
```

```
begin
    # Eigenvalues are sorted in increasing order
    low_idx = 1
    # Get the lowest eigenvalue
    energy_approx_H = evalues_H[low_idx]
    # Get the corresponding eigenfunction
    efunction_approx_H = evectors_H[:, low_idx]
end
```

(c) Use the expressions for the exact ground state energy and ground state eigenfunction φ_0 from *Exercise 1* to verify that the eigenvalue is converged to 10^{-4} and the $L^2(\mathbb{R})$-error of the eigenvector is also around 10^{-4}.

Hint: Note, that `eigen` does indeed return the finite difference approximation $(f(x_1), \dots, f(x_n))^T$ to the eigenfunction f, where $\{x_i\}_{i=1}^N$ are the points of your finite differences grid. However, since it normalises the vectors in $\mathbb{R}^N$ using the Euclidean norm, the resulting approximate function f is *not* appropriately $L^2(\mathbb{R})$-normalised. To see this, consider the finite differences approximation to the integral

$$\int_\Omega f(x)dx,$$

which itself is an approximation to $\int_{\mathbb{R}} f(x)dx$.

Hint

It might be useful to write two helper functions

1. A function `discretized_l2_norm(f, a)` to approximate the $L^2(\mathbb{R})$ norm of a vector `f` of length N representing function discretized to the equispaced grid of N points on the interval $[-a, a]$.
2. A function `discretized_l2_error(f1, f2, a)` to approximate the $L^2(\mathbb{R})$ error between two discretized functions, also using

`discretized_l2_norm (generic function with 1 method)`

```
function discretized_l2_norm(f, a)
    dx = 2a / (length(f) - 1)
    l2_norm = sqrt(sum(abs.(f) .^ 2) * dx)
    return l2_norm
end
```

`discretized_l2_error (generic function with 1 method)`

```
function discretized_l2_error(f1, f2, a)
    l2_error = discretized_l2_norm(f1 - f2, a)
    return l2_error
end
```

```
begin
    # The ground state eigenvalue
    μ_exact_H = ω * 0.5

    # Normalizing the eigenfunction
    efunc_norm = discretized_l2_norm(efunction_approx_H, a)
    efunc_approx_normalized = efunction_approx_H/efunc_norm

    # Computing the errors
    efunc_error = discretized_l2_error(efunc_approx_normalized, eigenfunction_values,
    a)
    evalue_error = abs(energy_approx_H - μ_exact_H)

    println("Eigenvalue error: $evalue_error")
    println("Eigenfunction error: $efunc_error")
end
```

```
Eigenvalue error: 1.5852219709344695e-5
Eigenfunction error: 1.0370831859818148e-5
```

We can see that indeed the eigenvalue is converged to 10^{-4} and the $L^2(\mathbb{R})$-error of the eigenvector is also around 10^{-4}.

(d) Use the same computational setup to determine an approximation for the first eigenpair of the Morse oscillator Hamiltonian H^M. What is the deviation in the numerically obtained ground state eigenvalue between the Morse and Harmonic oscillator model and how does it compare to the expected deviation considering the analytical eigenvalues? Plot the numerical ground state eigenfunctions of H^M and H^H. Where do these functions differ?

(d) Solution:

```
begin
    fd_Hm = - 0.5 * fd_laplacian(N, a) + Diagonal(Vm.(grid_points) )
    evalues_M, evectors_M = eigen(fd_Hm)
end;
```

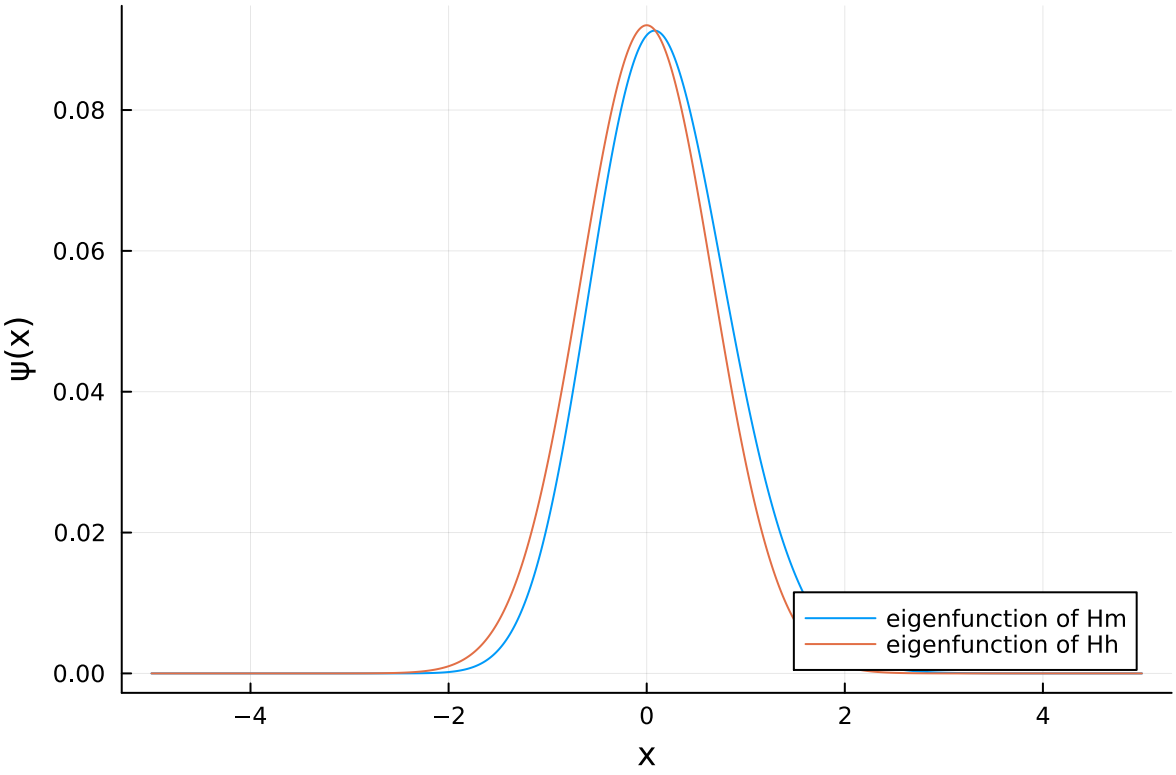
```
begin
    energy_approx_M = evalues_M[low_idx]
    efunction_approx_M = evectors_M[:, low_idx]

    # Deviation in the numerically obtained ground state eigenvalue
    energy_deviation_numerical = abs(energy_approx_M - energy_approx_H)

end;
```

```
begin
    μ_exact_M = 0.5 * ω - 0.25 * ω^2 / (4 * D)
    energy_deviation_expected = abs(μ_exact_M - μ_exact_H)
    println("Expected deviation in ground state eigenvalue:
    $energy_deviation_expected")
    println("Numerical deviation in ground state eigenvalue:
    $energy_deviation_numerical")
    difference = abs(energy_deviation_expected - energy_deviation_numerical)
    println("Difference: $difference")
end
```

```
Expected deviation in ground state eigenvalue: 0.015820312500000044
Numerical deviation in ground state eigenvalue: 0.01581972223205197
Difference: 5.902679480751516e-7
```




```
begin
    plot(grid_points, efunction_approx_M, label="eigenfunction of Hm", xlabel="x",
    ylabel="ψ(x)")
    plot!(grid_points, efunction_approx_H, label="eigenfunction of Hh",
    legend=:bottomright)
end
```

By analyzing the plots we can notice that the peak of the Morse oscillator ground state eigenfunction is **shifted** in comparison with the Harmonic oscillator ground state eigenfunction. The heights of the two curves also look slightly different.

Exercise 3 (1 + 2 + 1 P)

We return to the problem of finding numerically an approximate eigenfunction of the Harmonic oscillator H^H and replace the diagonalisation using `eigen` in *Exercise 2(b)* by the inverse power method discussed in the lecture:

```
inverse_power_method (generic function with 1 method)
```

```
function inverse_power_method(A; u=randn(eltype(A), size(A, 2)),
                              tol=1e-6, maxiter=500)
    norm_Δu = NaN
    Afac = factorize(A)  # Factorise A to make A \ x more economical
    for i in 1:maxiter
        u_prev = u
        u = Afac \ u
        normalize!(u)
        norm_Δu = min(norm(u - u_prev), norm(-u - u_prev))
        norm_Δu < tol && break
    end
    μ = dot(u, A, u)
    norm_Δu ≥ tol && @warn "Inverse power not converged $norm_Δx"
    (; μ, u)
end
```

(a) For the numerical setup of *Exercise 2(b)* tune the tolerance `tol` such that we obtain an L^2-error in the eigenvector and an error in the eigenvalue below 10^{-4}. Run your computation in both double and single precision. Is single precision sufficiently accurate to not impact the quality of the result?

(a) Solution:

```
begin
    fd_Hh_64 = - 0.5 * fd_laplacian(N, a; T=Float64) + 0.5 * ω^2 *
    Diagonal(grid_points.^2)
    fd_Hh_32 = - 0.5 * fd_laplacian(N, a; T=Float32) + 0.5 * ω^2 *
    Diagonal(grid_points.^2)

    tol=1e-4

    μ_64, u_64=inverse_power_method(fd_Hh_64,tol=tol)
    μ_32, u_32=inverse_power_method(fd_Hh_32,tol=tol)

    efunc_approx_64 = u_64/discretized_l2_norm(u_64, a)
    efunc_approx_32 = u_32/discretized_l2_norm(u_32, a)
end;
```

```
begin
    # Computing the errors
    efunc_error_64 = discretized_l2_error(efunc_approx_64, eigenfunction_values, a)
    evalue_error_64 = abs(μ_64 - μ_exact_H)

    efunc_error_32 = discretized_l2_error(efunc_approx_32, eigenfunction_values, a)
    evalue_error_32 = abs(μ_32 - μ_exact_H)

    println("\nComparing the algorithm error for both types\n")

    println("\nSingle  Precision:")
    println("  Eigenvalue Error: $evalue_error_64")
    println("  Eigenfunction Error: $efunc_error_64")


    println("\nDouble Precision:")
    println("  Eigenvalue Error: $evalue_error_32")
    println("  Eigenfunction Error: $evalue_error_32")
end
```

```
Comparing the algorithm error for both types


Single  Precision:
  Eigenvalue Error: 1.5850975597198058e-5
  Eigenfunction Error: 1.9999999998352884

Double Precision:
  Eigenvalue Error: 1.5808150617413474e-5
  Eigenfunction Error: 1.5808150617413474e-5
```

We can conclude that single precision is sufficiently accurate for the problem not to impact the quality of the computed solution.

(b) Assume our goal is to approximate the ground state eigenvalue of the Morse oscillator using the single-precision procedure in (a), i.e. the computation of the ground state eigenvalue of the harmonic oscillator using the inverse power method in single precision. Compute the total error against the analytical ground state eigenvalue E_0^M. Split this total error into error contributions (model error, discretisation error, algorithm error, arithmetic error) as discussed in the lecture and indicate their respective sizes in each case. Use `BigFloat` as a proxy for estimating the result for "exact" floating-point arithmetic.

(b) Solution:

Let's start by doing a recap of all the error types:

- the arithmetic error due to the storage of the data: $|\mu_1^{(\text{fp32})} - \mu_1^{(\text{big})}|$;
- the algorithm error due to the non-null tolerance: $|\mu_1^{(\text{big})} - \mu_1|$;
- the discretization error due to the finite number of mesh points: $|\mu_1 - \lambda_1|$;
- the model error due to the simplifying assumptions of our model: $|\lambda_1 - \lambda_*|$.

The total error is $|\mu_1^{(\text{fp32})} - \lambda_*|$.

In our case, without knowing the analytical formula of eigenvalue μ_1 for the discretized problem we can calculate the difference between numerical and analytical solutions, $\mu_1^{(\text{big})}$ and λ_1:

$$|\mu_1^{(\text{big})} - \lambda_1| \leq e_{\text{algorithm}} + e_{\text{discretization}}$$

```
begin
    fd_Hh_big = - 0.5 * fd_laplacian(N, a; T=BigFloat) + 0.5 * ω^2 *
    Diagonal(grid_points.^2)

    μ_big, u_big=inverse_power_method(fd_Hh_big,tol=tol)
end;
```

e_arithmetic = 4.21048930999646948386643480314820351187454260917868484765899968324579176146418e-08

```
# Arithmetic error
e_arithmetic = abs(μ_big - μ_32)
```

e_discrit_plus_algorithm = 1.58502555105134391872617843027963250221792923010917868484765899968324579176146 4e-05

```
# Computing the discretization and algorithm errors together
e_discrit_plus_algorithm = abs(μ_big - μ_exact_H)
```

e_model = 0.015820312500000044

```
# Model error
e_model = abs(μ_exact_M - μ_exact_H)
```

e_total = 0.01580450434938263

```
# The total error against the analytical ground state eigenvalue
e_total = abs(μ_32 - μ_exact_M)
```

(c) Use the numerical parameters of your setup to balance all error contributions with the model error, i.e. tune `N`, `tol` and decide between `Float32` and `Float64`, such that each of the error contributions you computed in (b) are roughly on the order of the model error.

```
begin
    tolerance = 0.26
    N_points = 100
    points = range(-a, stop=a, length=N_points)

    Hh = - 0.5 * fd_laplacian(N, a; T=Float32) + 0.5 * ω^2 * Diagonal(grid_points.^2)
    μ, _=inverse_power_method(Hh,tol=tolerance)

    Hh_big = - 0.5 * fd_laplacian(N, a; T=BigFloat) + 0.5 * ω^2 *
    Diagonal(grid_points.^2)
    μbig, _=inverse_power_method(Hh_big,tol=tolerance)
end;
```

0.015445442397400150220395524724022963877779212719862804566116531462985825046994 84

```
abs(μbig - μ) # arithmetic error
```

0.018694625307581653068386011494891265269380775219862804566116531462985825046994 84

```
abs(μbig - μ_exact_H) # discretization and algorithm errors together
```

0.015820312500000044

```
abs(μ_exact_M - μ_exact_H) # model error
```

0.019069495410181547

```
abs(μ - μ_exact_M) # total error
```

After tuning the number of points in the grid and the tolerance we can see that each of the error contributions is roughly on the order of the model error.