Error control in scientific modelling (MATH 500, Herbst)

Sheet 3: Matrix eigenproblems

```
begin
    using LinearAlgebra
    using Plots
end
```

```
Precompiling Plots...
    770.6 ms  ✓ JLLWrappers
   1294.2 ms  ✓ Graphite2_jll
   1208.2 ms  ✓ libfdk_aac_jll
   1502.1 ms  ✓ EpollShim_jll
   1405.3 ms  ✓ LERC_jll
   1516.0 ms  ✓ Bzip2_jll
   1630.4 ms  ✓ Xorg_libICE_jll
   1523.0 ms  ✓ fzf_jll
   1700.1 ms  ✓ LLVMOpenMP_jll
   1753.2 ms  ✓ LAME_jll
   1897.9 ms  ✓ OpenSSL_jll
   1882.0 ms  ✓ Xorg_libXau_jll
   1898.8 ms  ✓ libpng_jll
   2100.2 ms  ✓ Libmount_jll
   1104.5 ms  ✓ Xorg_libXdmcp_jll
   1289.5 ms  ✓ Ogg_jll
   1583.3 ms  ✓ JpegTurbo_jll
   1434.6 ms  ✓ mtdev_jll
   1485.1 ms  ✓ gperf_jll
   1515.3 ms  ✓ x264_jll
   1582.6 ms  ✓ x265_jll
   1372.1 ms  ✓ Expat_jll
   1199.2 ms  ✓ Opus_jll
   2020.8 ms  ✓ XZ_jll
   1585.8 ms  ✓ libaom_jll
   1512.7 ms  ✓ LZO_jll
   1712.5 ms  ✓ Zstd_jll
   1106.1 ms  ✓ libevdev_jll
   1212.4 ms  ✓ Xorg_xtrans_jll
   1018.1 ms  ✓ Wayland_protocols_jll
   1391.3 ms  ✓ Libgpg_error_jll
   1564.7 ms  ✓ Libiconv_jll
   1597.5 ms  ✓ Libffi_jll
```

Exercise 1

In the lectures we saw that minimising the Rayleigh quotient provides a numerical tool for computing eigenvalues via optimisation problems. But even beyond that setting the Rayleigh quotient can be interpreted as a tool to obtain an approximation for the eigenvalue corresponding to an approximate eigenvector. We will explore this in this exercise.

We consider the setting where we want to compute the eigenpair $(\lambda, v) \in \mathbb{R} \times \mathbb{C}^n$ of the Hermitian matrix $A \in \mathbb{C}^{n \times n}$. As usual we take v to be normalised. Employing some numerical scheme we do not get the exact eigenvector v but only some approximation u of unit norm. The Rayleigh quotient $R_A(u)$ provides an approximation to λ.

(a) Let $\theta = \angle(u, v)$ be the angle between u and v, and let d be the unit component of u orthogonal to v. We can decompose $u = \cos(\theta)v + \sin(\theta)d$ with $d \perp v$ and $\|d\| = 1$.

Show that

$$R_A(u) = \lambda + \sin^2\theta(\langle d, Ad\rangle - \lambda).$$

Deduce that

$$R_A(u) = \lambda + (\langle d, Ad\rangle - \lambda)\theta^2 + O(\theta^4). \qquad (\theta \to 0)$$

solution.

We expand and simplify:

$$\begin{aligned} R_A(u) &= \langle u, Au\rangle \\ &= \lambda\cos^2\theta + 2\cos\theta\sin\theta\,\mathrm{Re}\langle d, Av\rangle + \sin^2\theta\langle d, Ad\rangle \\ &= \lambda\cos^2\theta + \sin^2\theta\langle d, Ad\rangle \\ &= \lambda(1 - \sin^2\theta) + \sin^2\theta\langle d, Ad\rangle \\ &= \lambda + \sin^2\theta(\langle d, Ad\rangle - \lambda). \end{aligned}$$

(b) Assume now a numerical scheme (e.g. power iteration) yields an approximation u to the exact eigenvector v, which is accurate to a tolerance ε, i.e. $\|u - v\| < \varepsilon$. We want to estimate the corresponding eigenvalue using $R_A(u)$. Based on (a), how does the error between this estimate and the true eigenvalue λ scale with ε?

solution. For u, v of unit norm we have $\|u - v\| = 2\sin\frac{\theta}{2}$. We have $|R_A(u) - \lambda| = O(\varepsilon^2)$.

(c) Reconsider your power iteration implementation from Sheet 1. Extend it, such that it employs the iterated eigenvector $x^{(i)}$ as well as the Rayleigh quotient to estimate the eigenvalue at each step. For the procedure computing the largest eigenvalue of the matrix

$$A = \begin{pmatrix} 30\,000 & -10\,000 & 10\,000 \\ -10\,000 & -30\,000 & 0 \\ 10\,000 & 0 & 1 \end{pmatrix}$$

record both the approximate eigenvalue as well as the approximate eigenvector in each iteration in two separate arrays. Use this data to plot the error in the approximate eigenvalue as well as the error in the eigenvector as the iteration proceeds. You should find numerical confirmation to your analysis of (a) and (b).

Some hints:

- For computing the eigenvalue error just take the modulus of the absolute error, for the eigenvector error take the l_2-norm (`norm` function in Julia).
- You can compute the exact eigenvalue and eigenvector using the `eigen` routine of Julia. Note, however, that (real) eigenvectors are only determined up to the sign, so you have to ensure that

the same sign convention is used in `eigen` as well as your own algorithm. The easiest is to determine the sign of the first element of the vector returned by your routine as well as `eigen` and multiply one of the vectors by `-1` in case these differ.

- Usually it is best to employ a `log`-scale on the y-axis for such error plots.

```
A = 3×3 Matrix{Float64}:
  30000.0  -10000.0  10000.0
 -10000.0  -30000.0      0.0
  10000.0       0.0      1.0
```

```
A = [ 30_000  -10_000  10_000;
     -10_000  -30_000     0.0;
      10_000      0.0     1.0]
```

```
# split: statement
function power_method(A, x0=randn(eltype(A), size(A, 2)); tol=1e-8, maxiter=500)
    x_history = []
    λ_history = []
    x = x0
    for i in 1:maxiter
        xprev = x
        x = A * x
        x /= norm(x)

        λ = 1.0  # TODO: Estimate λ by a Rayleigh quotient

        push!(x_history, x)
        push!(λ_history, λ)

        norm_Δx = min(norm(x - xprev), norm(-x - xprev))
        if norm_Δx < tol
            break
        end
    end
    λ = dot(x, A, x)
    (; λ, x, λ_history, x_history)
end
```

```
power_method (generic function with 2 methods)
```

```
    # split: solution
    function power_method(A, x0=randn(eltype(A), size(A, 2)); tol=1e-8, maxiter=500)
        x_history = []
        λ_history = []
        x = x0
        for i in 1:maxiter
            xprev = x
            x = A * x
            x /= norm(x)

            λ = dot(x, A, x)  # Estimate λ by a Rayleigh quotient

            push!(x_history, x)
            push!(λ_history, λ)

            norm_Δx = min(norm(x - xprev), norm(-x - xprev))
            if norm_Δx < tol
                break
            end
        end
        λ = dot(x, A, x)
        (; λ, x, λ_history, x_history)
    end
```

results =

```
(λ = 34454.0, x = [-0.949879, 0.147373, -0.275703], λ_history = [-30225.1, -30058.1, -29769.4
```

```
    results = power_method(A)
```

```
[-30225.1, -30058.1, -29769.4, -29431.2, -29036.3, -28576.4, -28041.9, -27422.3, -26706.3, -
```

```
    results.λ_history
```

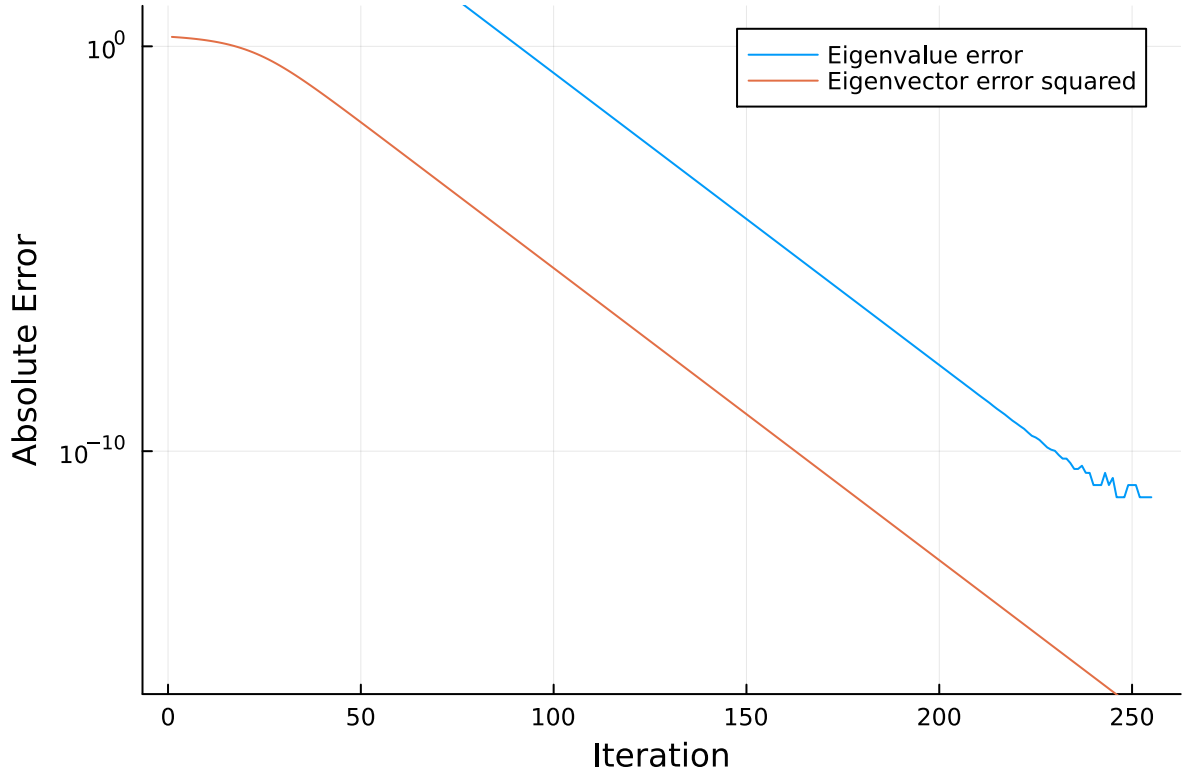
```
[[0.013738, 0.999085, -0.0405144], [-0.314727, -0.949172, 0.00432943], [0.00293331, 0.995079
```

```
    results.x_history
```

```
    # split: statement
    begin
        λ_errors = ones(200)         # TODO: Compute the true λ errors
        x_errors = 1e-3 * ones(200)  # TODO: Compute the true x errors squared
    end
```

```
[1.70928, 1.68425, 1.65767, 1.629, 1.59811, 1.56489, 1.52922, 1.49099, 1.45011, 1.40652, 1.3
```

```
# split: solution
begin
    λ, X = eigen(A)
    _, i = findmax(abs, λ)
    λmax = λ[i]
    xmax = X[:,i]
    λ_errors = abs.(results.λ_history .- λmax)
    x_errors = [min(norm(x - xmax), norm(-x - xmax))^2 for x in results.x_history]
end
```




```
# Here is some helpful plotting code with a log-scale
let
    plt = plot(xlabel="Iteration", ylabel="Absolute Error",
               yscale=:log, ylims=(1e-16, 10))
    plot!(λ_errors, label="Eigenvalue error")
    plot!(x_errors, label="Eigenvector error squared")
end
```

Exercise 2

We saw in the lecture that if $A \in \mathbb{C}^{n\times n}$ is Hermitian (i.e. $A = A^H$), then the Rayleigh quotient

$$R_A(x) = \frac{\langle x, Ax\rangle}{\langle x, x\rangle}$$

is real for all $x \in \mathbb{C}^n$. The point of this exercise is to prove the converse, i.e.

If Rayleigh quotient $R_A(x)$ is real for all $x \in \mathbb{C}^n$, then A is a Hermitian matrix.

To show this we proceed as follows:

(a) Given an arbitrary matrix S, show that if $\langle x, Sx\rangle = 0$ for all $x \in \mathbb{C}^n$, then we must have

$$\langle y, Sz\rangle + \langle z, Sy\rangle = 0 \qquad \forall y, z \in \mathbb{C}^n$$

Hint: Expand $\langle (y+z), S(y+z)\rangle$.

(b) Given an arbitrary matrix A show that $A - A^H$ is anti-Hermitian, i.e. that

$$\langle x, (A - A^H)y\rangle = \overline{-\langle y, (A-A^H)x\rangle} \qquad \forall x, y \in \mathbb{C}^n$$

(c) Use the results of (a) and (b) to show if $\langle x, Ax\rangle$ is real for all $x \in \mathbb{C}^n$, then A must be Hermitian.
Hints: First relate $\langle x, Ax\rangle$ to $\langle x, A^H x\rangle$, then use (a). From the resulting expression show that $\langle x, Ay\rangle = 0 \, \forall x, y \in \mathbb{C}^n$. You will only need (b) for one part of your reasoning in this step.

(d) Prove the above statement, i.e. if $R_A(x)$ is real for all $x \in \mathbb{C}^n$, then A is Hermitian.

solution.

(a) For all $x, y \in \mathbb{C}^n$ we have

$$\begin{aligned} 0 &= \langle (x+y), S(x+y)\rangle \\ &= \langle x, Sx\rangle + \langle x, Sy\rangle + \langle y, Sx\rangle + \langle y, Sy\rangle \\ &= \langle x, Sy\rangle + \langle y, Sx\rangle. \end{aligned}$$

(b) Note that $\forall x \in \mathbb{C}^n$

$$\begin{aligned} \langle x, (A - A^H)y\rangle &= \overline{\langle (A-A^H)y, x\rangle} \\ &= \overline{\langle y, (A-A^H)^H x\rangle} \\ &= \overline{-\langle y, (A-A^H)x\rangle}. \end{aligned}$$

(c) First note $\forall x \in \mathbb{C}^n$

$$\langle x, Ax\rangle = \overline{\langle x, Ax\rangle} = \langle Ax, x\rangle = \langle x, A^H x\rangle$$

From this we deduce $\langle x, (A - A^H)x\rangle = 0$ for all $x \in \mathbb{C}^n$. Employing (a) we obtain for arbitrary $x, y \in \mathbb{C}^n$

$$0 = \langle x, (A-A^H)y\rangle + \langle y, (A-A^H)x\rangle \qquad \forall x, y \in \mathbb{C}^n.$$

This gives us

$$\begin{aligned} \overline{\langle y, (A-A^H)x\rangle} &= \overline{-\langle x, (A-A^H)y\rangle} \\ &= \langle y, (A-A^H)x\rangle. \end{aligned}$$

where we have used property (a) in the first and property (b) in the second step. This shows that

$$\langle y, (A - A^H)x \rangle \in \mathbb{R} \qquad \forall x, y \in \mathbb{C}^n.$$

This gives us the freedom to also replace $y \to iy$ and observe

$$\langle iy, (A - A^H)x \rangle = i\langle y, (A - A^H)x \rangle \in \mathbb{R} \qquad \forall x, y \in \mathbb{C}^n.$$

We deduce that

$$\langle y, (A - A^H)x \rangle = 0 \qquad \forall x, y \in \mathbb{C}^n.$$

From this we conclude

$$\langle x, Ay \rangle = \langle x, A^H y \rangle \qquad \forall x, y \in \mathbb{C}^n$$

or $A = A^H$ is Hermitian.

Exercise 3

Recall that the Frobenius norm of a matrix $A \in \mathbb{C}^{n \times n}$ is given by

$$\|A\|_F = \sqrt{\operatorname{tr}(A^H A)} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{ij}|^2}$$

(a) What is the Frobenius norm of a diagonal matrix? What is the p-norm of a diagonal matrix? Conclude whether the Frobenius norm can be associated to any vector p-norm via the usual induced matrix norm expression

$$\max_{0 \neq x \in \mathbb{C}^n} \frac{\|Ax\|_p}{\|x\|_p}.$$

(b) Show that

$$\|A\|_2 \leq \|A\|_F$$

and use this to conclude for a Hermitian matrix B and any nonzero $x \in \mathbb{C}^n$ we have

$$R_B(x) \leq \|B\|_F. \qquad (*)$$

(c) Keeping in mind our corollary of Courant-Fisher, namely that for the **maximum** eigenvalue λ_n we have

$$\lambda_n = \max_{0 \neq x \in \mathbb{C}^n} R_B(x)$$

as well as your result from (a) on diagonal matrices, argue why $(*)$ is a crude bound, in particular for large matrices.

solution.

(a) For any diagonal matrix $D = \mathbf{diag}(d_1, d_2, \ldots, d_n)$ we have

$$\|D\|_F = \sum_{i=1}^{n} |d_i|^2$$

$$\|D\|_p = \max_{v \in \mathbb{C}^n, \|v\|_p = 1} \sqrt[p]{\sum_{i=1}^{n} |d_i|^p |v_i|^p} = \max_{i=1,\ldots,n} |d_i|$$

Since this shows that for a diagonal matrix the Frobenius norm does not agree with any of the matrix norms induced by a p-norm, it cannot be associated to any of the vector p norms.

(b) For any matrix A we can consider the matrix $A^H A$ which is Hermitian (and positive semidefinite) and thus has an eigendecomposition with nonnegative eigenvalues μ_i

$$A^H A = \sum_{i=1}^{n} \mu_i u_i u_i^H.$$

Let $\mu_{\max}$ denote the largest eigenvalue. We then have

$$\|A\|_2^2 = \mu_{\max} \leq \sum_i \mu_i = \mathrm{tr}(A^H A) = \|A\|_F^2$$

For the case that we have a Hermitian matrix B, it has n real eigenvalues that we can order as

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n.$$

Starting from the Courant-Fisher corollary and using above argument we thus get

$$R_B(x) \leq \max_{0 \neq x \in \mathbb{C}^n} R_B(x) = \lambda_n = \max_i \lambda_i \leq \max_i |\lambda_i| = \|B\|_2 \leq \|B\|_F$$

as desired.

(c) It is a crude bound since we have the sequence from (b)

$$R_B(x) \leq \lambda_n \leq \|B\|_2 \leq \|B\|_F,$$

and each of these three inequalities can easily have a large gap:

- $$R_B(x) \ll \lambda_n$$

  : For example, when the minimum eigenvalue of B is much smaller than the maximum eigenvalue (and corresponding directions x are considered)

- $$\lambda_n \ll \|B\|_2$$

  : This happens only if the largest eigenvalue is negative (i.e.~the full spectrum is negative and B is negative definite) and the right hand side discards the sign. (Otherwise, for $\lambda_n \geq 0$ we have

equality $\lambda_n = \|B\|_2$)

- $$\|B\|_2 \ll \|B\|_F$$

  : Whenever there are multiple eigenvalues that have large absolute value. Let us denote $\lambda_{\mathbf{maxabs}} = \mathbf{max}_i\, |\lambda_i|$ and we get

$$\|B\|_2^2 = \lambda_{\mathbf{maxabs}}^2 \ll \lambda_{\mathbf{maxabs}}^2 + \sum_{\lambda_i \neq \lambda_{\mathbf{maxabs}}} \lambda_i^2 = \|B\|_F^2.$$

As an aside, from the same argument we also see an upper bound $\|B\|_F \leq \sqrt{n}\|B\|_2$ which is also true for any matrix, not necessarily Hermitian (e.g. by an analogous proof using singular values).