Project 05: Matrix Diagonalization
Using the Power Method and Gram-Schmidt Orthogonalization

1. What are we trying to do?

In this project, **we must write our own C++ program** to calculate the **eigenvalues** and **eigenvectors** of a real symmetric matrix. They describe important properties of physical systems, for example vibration modes, quantum states and differential operators. Our central equation is

$$\boxed{A\mathbf{x}_k = \lambda_k \mathbf{x}_k,}$$

where A is the *known* matrix, λ_k is an *unknown eigenvalue*, and $\mathbf{x}_k$ is its corresponding *eigenvector*. The matrix changes the vector's length (and possibly its sign) but not its direction as a line. We calculate these quantities by an iterative method rather than a ready-made eigensolver.

2. What matrix do we use, and where does it come from?

We approximate a second derivative on equally spaced points x_i using

$$\left.\frac{d^2 f}{dx^2}\right|_{x_i} \approx \frac{f_{i-1} - 2f_i + f_{i+1}}{\Delta^2}.$$

With grid spacing $\Delta = 1$ and zero external boundary values, we obtain the following **symmetric tridiagonal matrix** for the official $n = 6$ problem:

$$A = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}.$$

We store all its entries in a **two-dimensional C++ array**. The eigenvalues of this discrete second-derivative matrix are negative; a matrix approximating $-d^2/dx^2$ would have the opposite sign.

3. How does the Power Method work?

We start with a nonzero vector $\mathbf{x}^{(0)}$. At every step, we multiply by A and normalize the result so that the vector does not grow or shrink without bound:

$$\boxed{\mathbf{y}^{(k)} = A\mathbf{x}^{(k)}, \qquad \mathbf{x}^{(k+1)} = \frac{\mathbf{y}^{(k)}}{\|\mathbf{y}^{(k)}\|_2}.}$$

Under the usual conditions, the method approaches the eigenvector whose eigenvalue has the **largest absolute value** (not necessarily the most positive eigenvalue). A negative dominant eigenvalue may cause alternating signs; this is normal.

A tiny example. Take $B = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ and $\mathbf{x}^{(0)} = (1,0)^T$. Then $B\mathbf{x}^{(0)} = (2,1)^T$ and $\mathbf{x}^{(1)} = (2,1)^T/\sqrt{5}$. To estimate the eigenvalue, we use the **Rayleigh quotient**:

$$\boxed{R(\mathbf{x}) = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}},} \qquad R\big((2,1)^T\big) = \frac{14}{5} = 2.8 \longrightarrow 3.$$

Here 3 is the exact dominant eigenvalue of B. We can also check the **residual** $\|A\mathbf{x} - \lambda\mathbf{x}\|_2$: a small change in λ alone does not prove that the eigenvector is accurate.

From Gram-Schmidt to Our Laboratory Work

4. Why do we need Gram-Schmidt orthogonalization?

The ordinary Power Method usually returns only one dominant eigenvector. We need **six different eigenvectors**. To prevent the next vector from pointing in directions already found, we remove its projections onto the previously calculated *normalized* eigenvectors $\mathbf{q}_0, \ldots, \mathbf{q}_{k-1}$:

$$\boxed{\mathbf{y} \leftarrow \mathbf{y} - \sum_{m=0}^{k-1} (\mathbf{q}_m^T \mathbf{y}) \mathbf{q}_m .}$$

In simple words, Gram-Schmidt makes the new vector **perpendicular** to the earlier ones. For example, subtracting the component of $\mathbf{y} = (2, 0)^T$ along $\mathbf{q}_0 = (1, 1)^T/\sqrt{2}$ gives $(1, -1)^T$, which becomes $(1, -1)^T/\sqrt{2}$ after normalization. We apply the projection during *every* power iteration, not only once.

5. What exactly must we do during the laboratory?

We must develop our **own complete C++ program** and perform these tasks:

1. **Build and display the matrix:** create the official 6×6 matrix (-2 on the diagonal; $+1$ immediately above and below; all other entries zero) using two-dimensional storage.
2. **Implement the numerical tools:** dot product, vector norm, normalization, matrix–vector multiplication, the Power Method, the Rayleigh quotient and Gram–Schmidt orthogonalization.
3. **Calculate all six eigenpairs:** use repeated iterations with Gram–Schmidt to find six eigenvalues and their eigenvectors. **Save the successive eigenvalue approximations** to files for later plots.
4. **Reconstruct and save the original matrix** from our computed, normalized eigenvectors $\mathbf{q}_k$ and eigenvalues λ_k:
$$A_{\text{num}} = \sum_{k=0}^{5} \lambda_k \mathbf{q}_k \mathbf{q}_k^T .$$
5. **Calculate and save the inverse matrix** using the spectral formula (all $\lambda_k \neq 0$):
$$A_{\text{num}}^{-1} = \sum_{k=0}^{5} \frac{1}{\lambda_k} \mathbf{q}_k \mathbf{q}_k^T .$$
6. **Verify the results:** check the eigenpair residuals, that different eigenvectors are perpendicular, that $A_{\text{num}} \approx A$, and that $A A_{\text{num}}^{-1} \approx I$.

Official numerical settings	$n = 6, \quad \Delta = 1, \quad \text{itmax} = 150, \quad \varepsilon = 10^{-8}$
Stopping rule	Monitor successive eigenvalue estimates; report the iteration cap honestly.

6. What must we investigate in our report?

1. Plot all six eigenvalue estimates versus iteration; compare convergence rates and explain whether their ordering is universal.
2. Repeat for a larger matrix, $n = 200$, and investigate the ten eigenvalues with largest *absolute magnitude*. Check convergence rather than assuming 150 iterations are enough.
3. Compare the reconstructed matrix with the original; repeat with the looser tolerance $\varepsilon = 10^{-4}$ and discuss any change in accuracy.
4. Explain why the inverse of an invertible symmetric matrix is symmetric: $(A^{-1})^T = (A^T)^{-1} = A^{-1}$.
5. Turn off Gram–Schmidt and investigate whether the method still returns distinct eigenvectors.

7. How do our small teaching examples help?

E01–E03: exact 2×2 eigenpairs, tridiagonal matrix and vector operations. **E04–E06:** one power step, repeated iterations, eigenvalue estimates and residuals. **E07–E08:** Gram–Schmidt and a projected-power toy. **E09:** matrix reconstruction and inverse. **Optional E10:** why a stable eigenvalue estimate may be misleading. These are complete *small learning examples*; **we must implement the official $n = 6$ project independently.**

8. What do we learn from this laboratory?

We learn to calculate eigenpairs using our own iterative C++ code, obtain different eigenvectors by orthogonalization, and judge numerical accuracy through residuals, reconstruction and inverse checks.

Official sources: Power-method laboratory assignment · Iterative-diagonalization lecture.