FYS3150 Project 2

Fridtjof Ronge Gjengset,
Sebastian Ranum Tenmann and Cecilie Klarpås

September 27, 2021

GitHub

`https://github.com/cecilmkl/comp_phys/tree/main/project2`

Problem 1

We can write $\hat{x} = \frac{x}{L}$ as $x = \hat{x}L$, and then get the equation:

$$\gamma \frac{d^2 u(\hat{x}L)}{d(\hat{x}L)^2} = -Fu(\hat{x}L)$$

Which can be written as

$$\left(\frac{d\hat{x}}{dx}\frac{d}{d\hat{x}}\right)^2 Lu(\hat{x}) = -FLu(\hat{x})$$

where:

$$\frac{d\hat{x}}{dx} = \frac{d}{dx}\left(\frac{x}{l}\right) = \frac{1}{L}$$

giving:

$$\frac{L}{L^2}\frac{d^2 u(\hat{x})}{d\hat{x}^2} = -FLu(\hat{x})$$

Thus:

$$\frac{d^2 u(\hat{x})}{d\hat{x}^2} = -\frac{FL^2}{\gamma}u(\hat{x})$$

If $\lambda = \frac{FL^2}{\gamma}$ then we can write:

$$\frac{d^2 u(\hat{x})}{d\hat{x}^2} = -\lambda u(\hat{x}) \tag{1}$$

We have shown that equation (4) from the task can be written as equation (5).

Problem 2

Given the assumption that the vectors $\vec{v_i}$ is a set of orthonormal basis vectors, and having $\mathbf{U}$ being an orthogonal transformation matrix, i.e. $\mathbf{U}^T = \mathbf{U}^{-1}$. If we then look at another set of vectors:

$$\vec{w} = \mathbf{U}\vec{v_i}$$

Using the fact that $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

$$\vec{w}^T = \mathbf{U}^T\vec{v_i}^T$$
$$\vec{w}^T\vec{w} = \vec{v_i}^T\mathbf{U}^T\mathbf{U}\vec{v_i}$$

Since $\mathbf{U}$ is orthogonal we get: $\mathbf{U}^T\mathbf{U} = \mathbf{U}^{-1}\mathbf{U} = \mathbf{I}$

$$\vec{w}^T\vec{w} = \vec{v_i}^T\mathbf{I}\vec{v_i}$$
$$\vec{w}^T\vec{w} = \vec{v_i}^T\vec{v_i}$$

And using the assumption from the start:

$$\vec{w}^T\vec{w} = \delta_{ij}$$

So for these given assumptions $\vec{w_i}$ is also orthonormal, i.e. transformations with $\mathbf{U}$ preserves orthonormality.

Problem 3

Before solving the matrix equation, we set up the symmetric tridiagonal matrix $\mathbf{A} \in \mathbb{R}^{NxN}$ for $N = 6$ and with -1 on the sub- and superdiagonal and 2 along the diagonal. This is done using a function for symmetric tridiagonal matrices that calls a special case of a function for tridiagonal matrices with constant diagonals, which again calls a special case of a function of a general tridiagonal matrix.

We then solve $\mathbf{A}\vec{v} = \lambda\vec{v}$ by affirming that Armadillo's `arma::eig_sym` agrees with the analytical results given by

$$\lambda^{(i)} = d + 2acos\Big(\frac{i\pi}{N+1}\Big)$$

$$\vec{v}_j^{(i)} = sin\Big(\frac{j * i\pi}{N+1}\Big) \qquad j = 1, ..., N$$

for each $i = 1, ..., N$. The eigenvectors are arranged in a matrix where each column $i-1$ is given by $\vec{v}^{(i)}$ and $j-1$ indicate row (since our code has indices 0, ..., N-1). The eigenvectors are scaled to unit norm and are found using functions `analytical_eigenvalues()` and `analytical_eigenvectors()` which give:

```
Eigenvalues with eig_sym:
   9.7051e+00
   3.6898e+01
   7.6193e+01
   1.1981e+02
   1.5910e+02
   1.8629e+02

Eigenvector with eig_sym:
  -0.2319  -0.4179   0.5211  -0.5211   0.4179  -0.2319
  -0.4179  -0.5211   0.2319   0.2319  -0.5211   0.4179
  -0.5211  -0.2319  -0.4179   0.4179   0.2319  -0.5211
  -0.5211   0.2319  -0.4179  -0.4179   0.2319   0.5211
  -0.4179   0.5211   0.2319  -0.2319  -0.5211  -0.4179
  -0.2319   0.4179   0.5211   0.5211   0.4179   0.2319

Analytical eigenvalues:
   9.7051e+00
   3.6898e+01
   7.6193e+01
   1.1981e+02
   1.5910e+02
   1.8629e+02

Analytical eigenvectors:
   0.2319   0.4179   0.5211   0.5211   0.4179   0.2319
   0.4179   0.5211   0.2319  -0.2319  -0.5211  -0.4179
   0.5211   0.2319  -0.4179  -0.4179   0.2319   0.5211
   0.5211  -0.2319  -0.4179   0.4179   0.2319  -0.5211
   0.4179  -0.5211   0.2319   0.2319  -0.5211   0.4179
   0.2319  -0.4179   0.5211  -0.5211   0.4179  -0.2319
```

With this level of precision, the eigenvalues are the same for both methods. The same is true for the absolute values of the eigenvectors, which is acceptable since the scaled eigenvector $c\vec{v}$ is equally good an eigenvector as $\vec{v}$. In this case, the analytical eigenvectors $\vec{v}^{(1)}$, $\vec{v}^{(2)}$, $\vec{v}^{(4)}$ and $\vec{v}^{(6)}$ at column 0, 1, 3 and 5 can be scaled with $c = -1$ to be equal to the eigenvectors given by `arma::eig_sym`.

Problem 4

a)

We write the following function

```
double find_max_value(arma::mat A, int& k, int& l){

  double max_value = 0;
  int N = arma::size(A)(0); //i is dimension N of matrix A

  //for loop runs through the non-diagonal matrix elements under
    the diagonal.
  for (int j=0; j<=N-1; j++){

      for (int i=1+j; i<=N-1; i++){
        if(abs(A(i,j))>abs(max_value)){
          max_value= A(i,j);
          k = j, l=i; //column k and row l

        }
      }
    }

  return max_value;
}
```

Listing 1: Algorithm for finding the largest off-diagonal element

The function takes an arbitrary matrix A as input as well as references to the integer k and l, representing the columns and rows respectively. Under the assumption that the matrix is symmetric, it locates the largest off-diagonal element in the lower triangular matrix of A. When the largest value is located, the references to the location (l,k) is updated, and the *max_value* is returned by the function as a double.

b)

We build the matrix from task 4b):

$$A = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & -0.7 & 0 \\ 0 & -0.7 & 1 & 0 \\ 0.5 & 0 & 0 & 1 \end{bmatrix}$$

We call the function above(listing(1)), with matrix A form task 4 b). After calling the function we print the return value of the function as well as the updated indices and get:

```
max value: -0.7 row: 2 column: 1
```

Listing 2: The result of calling the function from 4 a) with the matrix A from 4b) as argument and printing the updated indices

The matrix locates -0.7 in row 2 and column 1(this is index rotation and would normally be row 3 and column 2). The result is as expected as -0.7 is the largest absolute off-diagonal element($|-0.7| > |0.5|$).

Problem 5

To use the Jacobi rotation method for solving $\mathbf{A}\vec{v} = \lambda\vec{v}$ we first write a function that rotates an element in the matrix $\mathbf{A}$, and modifies it. To do this we first compute the value τ:

$$\tau = \frac{a_{ll}^{(m)} - a_{kk}^{(m)}}{2a_{kl}^{(m)}}$$

Which we then use to find the tan (t) value:

$$t = \tau \pm \sqrt{1+\tau^2}$$

with:

$$\text{If } \tau > 0, \text{ then } t = \frac{1}{\tau + \sqrt{1+\tau^2}}$$
$$\text{If } \tau < 0, \text{ then } t = \frac{-1}{-\tau + \sqrt{1+\tau^2}}$$

We can then use tan to equate the sin (s) and cos (c) values:

$$c = \frac{1}{\sqrt{1+t^2}}$$
$$s = ct$$

We then use these values to update the matrix elements in A using the following formulas:

$$a_{kk}^{(m+1)} = a_{kk}^{(m)}c^2 - 2a_{kl}^{(m)}cs + a_{ll}^{(m)}s^2$$
$$a_{ll}^{(m+1)} = a_{ll}^{(m)}c^2 - 2a_{kl}^{(m)}cs + a_{kk}^{(m)}s^2$$
$$a_{kl}^{(m+1)} = 0$$
$$a_{kl}^{(m+1)} = 0$$

and for all $i \neq l, k$:

$$\begin{aligned} a_{ik}^{(m+1)} &= a_{ik}^{(m)} c - a_{il}^{(m)} s \\ a_{ki}^{(m+1)} &= a_{ik}^{(m+1)} \\ a_{il}^{(m+1)} &= a_{il}^{(m)} c - a_{ik}^{(m)} s \\ a_{li}^{(m+1)} &= a_{il}^{(m+1)} \end{aligned}$$

and to update the overall rotation matrix R we use:

$$\begin{aligned} r_{ik}^{(m+1)} &= r_{ik}^{(m)} c - r_{il}^{(m)} s \\ r_{il}^{(m+1)} &= r_{il}^{(m)} c - r_{ik}^{(m)} s \end{aligned}$$

Then we write a function to solve this matrix equation, that runs the rotate function until the off-diagonal elements are close to zero, at a tolerance (ϵ) that we have set to $\epsilon = 10^{-8}$. It will then write out the eigenvalues as entries in a vector and eigenvectors as columns in a matrix. The results of which were printed to the terminal, and shown below:

```
Eigenvalues with jacobi:
   9.7051e+00
   3.6898e+01
   7.6193e+01
   1.1981e+02
   1.5910e+02
   1.8629e+02

Eigenvector with jacobi:
   0.2319  -0.4179  -0.5211   0.5211   0.4179  -0.2319
   0.4179  -0.5211  -0.2319  -0.2319  -0.5211   0.4179
   0.5211  -0.2319   0.4179  -0.4179   0.2319  -0.5211
   0.5211   0.2319   0.4179   0.4179   0.2319   0.5211
   0.4179   0.5211  -0.2319   0.2319  -0.5211  -0.4179
   0.2319   0.4179  -0.5211  -0.5211   0.4179   0.2319
```

As we can see, these results with this level of precision, gives identical values to the ones calculated with the analytical approach given in section "Problem 3".

Problem 6

a)

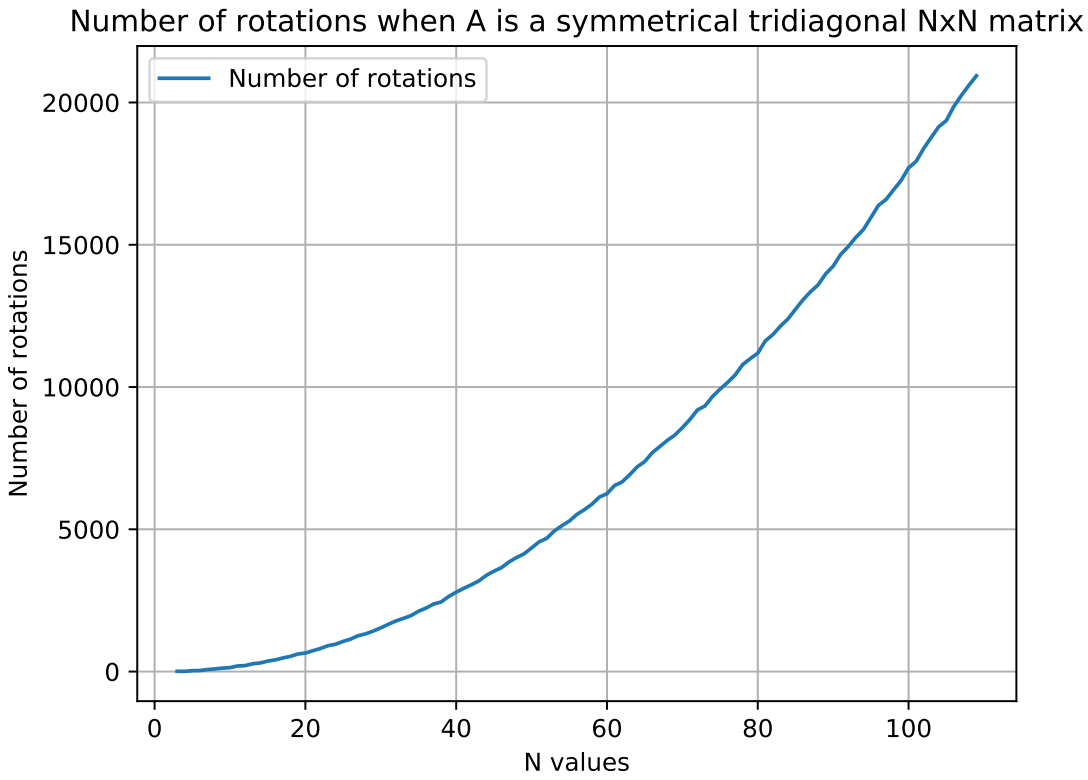

Figure 1: Shows a plot of how the amount of rotations/transformations done by jacobi eigensolver scales with N, when $\mathbf{A}$ is an NxN matrix. N values run from N=3 to N=109

When plotting the number of rotations/transformations that are done by the Jacobi rotation method for an NxN matrix A, we get the plot above(figure 1). The number of rotations required to get the off-diagonal elements in $\mathbf{A}$ to zero(or rather smaller than ϵ), seems to scale exponentially with N.

b)

If $\mathbf{A}$ was a dense matrix, we would expect greater difficulty scaling. As our aim is to bring all off-diagonal elements in $\boldsymbol{A}$ to 0(or close to 0), having a

sparse matrix where most off-diagonal elements are already 0 is a benefit. A dense matrix would require a lot more rotations to bring all off-diagonal elements to zero, hence greatly slowing down Jacobi's rotation algorithm.

Problem 7

We solve the eigenvalue problem $\mathbf{A}\vec{v} = \lambda\vec{v}$ with $\mathbf{A} \in \mathbb{N}\curvearrowleft\mathbb{N}$ for $N = 10$ and $N = 100$ using the Jacobi rotational algorithm from Problem 5. Then, we plot the approximate eigenvectors corresponding to the three smallest eigenvalues as functions of the dimensionless variable $\hat{x}$ and compare these to the analytical results.

The eigenvectors $\vec{v}^{(i)}$ are approximate solutions to the scaled equation (Equation 1) describing the situation with an applied force into a one-dimensional beam. Therefore, the graphs of the approximate eigenvectors as functions of scaled x variables, $\hat{x}$, represent the possible static shapes the beam may take.

Unsurprisingly, the discretization with $n = 100$ steps in Figure 2 gives a smoother curve than the one in Figure 3 due to the increased number of data points. The plots suggest that the approximations are so similar to the analytical solutions that they're difficult to tell apart without zooming in a fair amount. Our approximates are therefore following sine waves with varying wavelengths and equilibrium points in the endpoints $\hat{x} = 0$ and $\hat{x} = 1$.

In an effort to verify that the approximates and the analytical results actually do differ, we increased the precision when writing the results to file and looked manually at the values written to file. This quick survey of the values suggest that the difference lies in about 10^{-9}-th place. We conclude that our approximation method gives fairly accurate results.

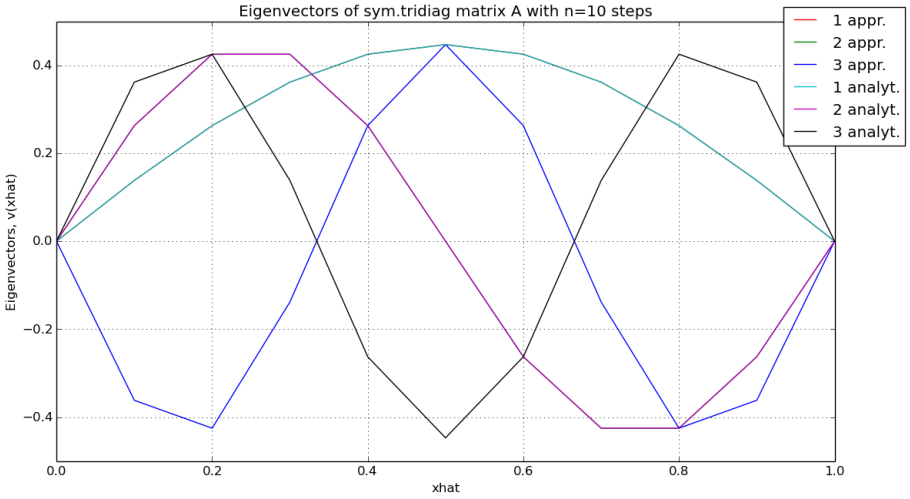

Figure 2: Shows the relation between the dimensionless $\hat{x}$ variable and the approximate eigenvectors corresponding to the first, second and third smallest eigenvalues with $n = 10$ discretization steps where eigenvectors are of size $N = n - 1 = 9$.

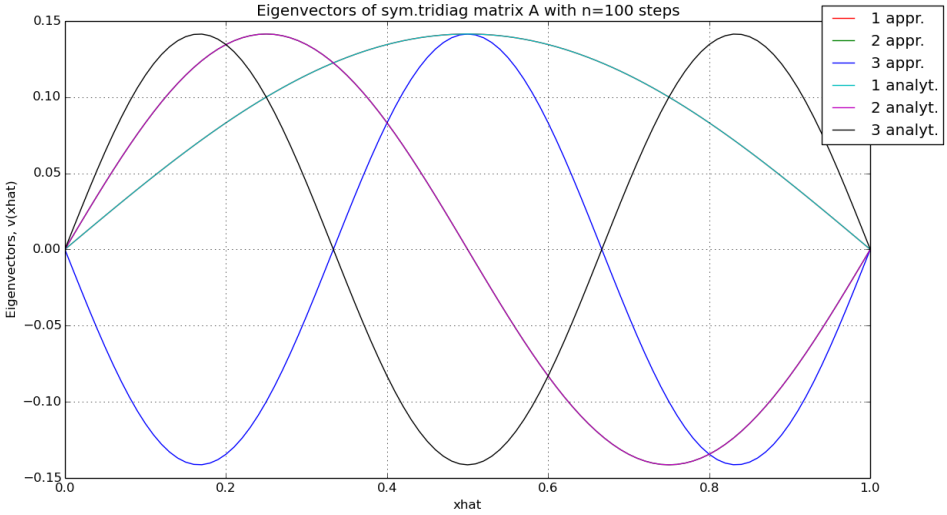

Figure 3: Shows the relation between the dimensionless $\hat{x}$ variable and the approximate eigenvectors corresponding to the first, second and third smallest eigenvalues with $n = 100$ discretization steps where eigenvectors are of size $N = n - 1 = 99$.