Project 2

Benedicte Allum Pedersen, Emil Heland Broll
Fredrik Oftedal Forr

In this project we will solve eigenpair-problems using numerical calculations.

We will start off by proving that orhogonal or unitary transformations preserves the orthogonality/dot product of the vectors.

Starting with an orthogonal basis of vectors, $\mathbf{v}_i$, we know that:

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \vdots \\ v_{in} \end{bmatrix}, \qquad \mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$$

An orthogonal transformation gives us the following:

$$\mathbf{w}_i = U\mathbf{v}_i, \text{where } U^T U = UU^T = \mathbf{I}$$

We want to prove that orthogonality is preserved:

$$\mathbf{w}_i^T \mathbf{w}_j = (U\mathbf{v}_i)^T (U\mathbf{v}_j) = \mathbf{v}_i^T U^T U \mathbf{v}_j = \mathbf{v}_i^T \mathbf{v}_j$$

Now that we have proved that orthogonal transformations preserve orthogonality, we can move on to performing our Jacobi iterations. To do so, we use an orthogonal transformation matrix, S:

$$S = \begin{bmatrix} 1 & 0 & \cdots & & & & \\ 0 & 1 & 0 & & & & \\ \vdots & 0 & \ddots & & & & \\ & & & \cos\theta & 0 & \cdots & \sin\theta \\ & & & 0 & 1 & \cdots & 0 \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & -\sin\theta & 0 & \cdots & \cos\theta \end{bmatrix}, \qquad S^T = S^{-1}$$

We simpltfy by assigning:

$$c = \cos\theta, s = \sin\theta, t = \tan\theta = \frac{s}{c}$$

For our positive definite matrix A, we perform the transformation $S^T AS = B$, giving us the general expression:

$$b_{ii} = a_{ii}, i \neq k, i \neq l$$
$$b_{ik} = a_{ik}c - a_{il}s, i \neq k, i \neq l$$
$$b_{il} = a_{il}c + a_{ik}s, i \neq k, i \neq l$$
$$b_{kk} = a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2$$
$$b_{kl} = (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2)$$

By choosing the largest non-diagonal element in the original matrix A, we can fix θ by choosing to set the largest non-diagonal element to zero. From this, we can deduce the required values of c and s:

$$c = \frac{1}{\sqrt{(1+t^2)}}, \qquad s = tc$$
$$t = -\tau \pm \sqrt{1+\tau^2}, \qquad \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}$$

This will result in a new matrix, B. We can then find the largest non-diagonal element of the new matrix and repeat the algorithm until this value is less than a given tolerance.
We have tested multiple tolerances and adjusted the maximum iterations to make sure the algorithm gets to finish properly. A table of our findings is below.
Number of similarity iterations required to reach tolerance
Estimate of number of iterations as a func of matrix dimensionality.
TABLE HERE
Compared to Armadillos *eigsys*-solver: ...
Time needed for running it...

Project 2a)

We have for three dimentions

$$\hat{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \qquad \mathbf{v}_i = \begin{bmatrix} v_{i1} \\ v_{i2} \\ v_{i3} \end{bmatrix}$$

This gives us

$$\mathbf{w}_i = \hat{U}\mathbf{v}_i$$
$$\begin{bmatrix} w_{i1} \\ w_{i2} \\ w_{i3} \end{bmatrix} = \begin{bmatrix} u_{11}v_{i1} + u_{12}v_{i2} + u_{13}v_{i3} \\ u_{21}v_{i1} + u_{22}v_{i2} + u_{23}v_{i3} \\ u_{31}v_{i1} + u_{32}v_{i2} + u_{33}v_{i3} \end{bmatrix}$$

If we then do the same for $\mathbf{w}_j$ we get

$$\mathbf{w}_j = \hat{U}\mathbf{v}_i$$
$$\begin{bmatrix} w_{j1} \\ w_{j2} \\ w_{j3} \end{bmatrix} = \begin{bmatrix} u_{11}v_{j1} + u_{12}v_{j2} + u_{13}v_{j3} \\ u_{21}v_{j1} + u_{22}v_{j2} + u_{23}v_{j3} \\ u_{31}v_{j1} + u_{32}v_{j2} + u_{33}v_{j3} \end{bmatrix}$$

wtf.....