Project 2

Benedicte Allum Pedersen, Emil Heland Broll
Fredrik Oftedal Forr

1 Abstract

Eigenvalue problems, from the equations of a buckling beam to Schroedinger's equation for two electrons in a three-dimensional harmonic oscillator well

2 Introduction

In this project we will solve eigenpair-problems using numerical calculations. We will also implement unit testing in our code to avoid mathematical and programming errors.

3 Methods

3.1 Mathematics

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:

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

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.

$$B_2 = S_2^T B S_2$$

When this is achieved, we will have a diagonal matrix, where all non-diagonal matrix elements are approaching 0.

$$D = S_n^T S_{n-1}^T \cdots S_1^T A S_1 \cdots S_{n-1} S_n$$

Since we have only performed orthogonal transformations, the eigenvalues for D and A will be the same. Since diagonal matrices have their eigenvalues on the diagonal, we can easily read off the eigenvalues.

3.2 Programming

To avoid errors in our code, we implement unit tests at the following points: * test * Test

4 Results

For our eigenproblem-solver, we tested and found a reasonable tolerance, 10^{-8}, so that the algorithm converges without reaching the maximum number of iterations, n^3. We test this using a tridiagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & -1 \\ 0 & 0 & 0 & \dots & -1 & 2 \end{bmatrix}$$

Tabel 1 shows the number of similarity transformations were needed to reach a point where all non-diagonal matrix elements approach 0, for matrix dimensionalities.

Dimensionality	# of transformations
4	6
10	141
20	551
50	3529

We can see that it looks like $i = 1.49n^2 - 1.63n - 4.23$, where i is number of iterations and n is dimensinality of the matix.

Table 2 shows a comparison of eigenvalues found with Armadillo's eigensystem-solver, on our tridiagonal matrix A.

λ_{Jacobi}	$\lambda_{\text{Armadillo}}$	Diff
0.2679	0.2679	0.0
1.0000	1.0000	0.0
2.0000	2.0000	0.0
3.0000	3.0000	0.0
3.7321	3.7321	0.0

Table 3 shows a comparison of runtimes of the Armadillo eigensystem-solver and our diagonalisation, for a 50x50-matrix.

Jacobi runtime	Armadillo runtime	Diff
0.090051s	0.000405s	0.089646s

5 Conclusions

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