

Computeroriented methods for solving Differential- and Integrals equations

WS2001/2002

31st January 2002

## **EigenCalc - a tool for calculation of eigenvalues**

### **Documentation**

Thomas Soboll

Robert Puschnik

`{tsoboll, rpuschni}@cosy.sbg.ac.at`

Academic Supervisor Dr. Trobec Roman

Department of Computer Science  
University of Salzburg

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Some important mathematical definitions</b>	<b>3</b>
2.1	Properties of Matrices . . . . .	3
<b>3</b>	<b>Program description</b>	<b>4</b>
3.1	General program usage . . . . .	4
3.1.1	Installation and start . . . . .	4
3.1.2	Interaction . . . . .	4
3.2	Preprocessing and tolerance . . . . .	4
3.3	Provided functionality . . . . .	5
<b>4</b>	<b>Results</b>	<b>6</b>
<b>5</b>	<b>Conclusion</b>	<b>7</b>

# Chapter 1

## Introduction

Eigenvalue problems occur in many areas of science and engineering, such as structural analysis and are needed for solving problems such as finding solutions for a system of differential equations.

Goal of our project is to get a programming tool based on MatLab that can help students to compare different methods for the calculation of eigenvalues. We have implemented several algorithms for the calculation of eigenvalues of square matrices:

- Power iteration
- Inverse iteration
- Orthogonal iteration
- QR iteration
- Jacobi method

We also implemented several improvements:

- shifting
- rayleigh quotient
- preprocessing of a matrix for faster convergence

# Chapter 2

## Some important mathematical definitions

**Definition eigenvalue/eigenvector 2.1** *The number  $\lambda$  is named eigenvalue of the square matrix  $A$ , if vector  $g \neq 0$  exists, in such way that  $A * g = \lambda * g$ .  $g$  is named eigenvector of  $A$  to eigenvalue  $\lambda$*

**Definition characteristic polynomial 2.1**  *$\lambda$  is eigenvalue of matrix*

$$A \Leftrightarrow \det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & \dots & \dots & a_{nn} - \lambda \end{vmatrix} = 0$$

### 2.1 Properties of Matrices

Property	Defintion
Diagonal	$a_{ij} = 0 \text{ for } i \neq j$
Tridiagonal	$a_{ij} = 0 \text{ for }  i - j  \geq 1$
Triangular	$a_{ij} = 0 \text{ for } i > j (\text{upper})$ $a_{ij} = 0 \text{ for } i < j (\text{lower})$
Hessenberg	$a_{ij} = 0 \text{ for } i > j + 1 (\text{upper})$ $a_{ij} = 0 \text{ for } i < j - 1 (\text{lower})$
Orthogonal	$A^T A = A A^T = I$
Unitary	$A^H A = A A^H = I$
Symmetric	$A = A^T$
Hermitian	$A = A^H$
Normal	$A^H A = A A^H$

# Chapter 3

## Program description

### 3.1 General program usage

#### 3.1.1 Installation and start

1. Copy `eigenvalue_calc.m` to your preferred working directory
2. Set the path of working directory, if not already done, in your matlab environment with: `path('/working_directory/')`.
3. Start the script with: `eigenvalue_calc`

#### 3.1.2 Interaction

At the beginning you are asked to enter your matrix. You can do this either by entering the name of a predefined variable (containing the matrix) or entering the matrix directly in matlab syntax (Example: `rand(2)` for a random 2x2 matrix).

Follow the instructions of the program then to get the results you want to have. For yes/no questions: 1 means yes and 0 means no.

Because you can do multiple iterations of the program all the results you received in a former run are saved. You are able to clear these results at the end of every iteration of the program.

If you enter something wrong (i.e. an option which is not possible) you are asked to enter the value again.

### 3.2 Preprocessing and tolerance

Jacobi method requires a symmetric matrix. If you do not have one, you can preprocess your matrix using Lanczos iteration to gain a symmetric Hessenberg matrix.

You can also use preprocessing to achieve faster convergence for the QR Iteration (Arnoldi Iteration).

The choice of the value for the tolerance plays an important role on the convergence of the chosen algorithm.

### 3.3 Provided functionality

- Power Iteration: calculates greatest eigenvalue (norm) of a matrix.
  - Normal power iteration (enter 0)
  - Power iteration with shift (entered parameter represents shift)
- Inverse Iteration: calculates smallest eigenvalue (norm) of a matrix.
  - Normal inverse iteration (enter 0).
  - Inverse iteration with shift (entered parameter represents shift).
  - Inverse iteration with rayleigh (enter -1).
- .
- Orthogonal Iteration calculates first  $p$  eigenvalues. ( $p$  has to be entered,  $0 < p < n$  ( $n$ =rows of matrix))
- QR Iteration calculates all eigenvalues of a matrix. (if entered value for shift  $\neq 0$  then QR iteration with shifting is used )
- Jacobi method calculates all eigenvalues of a symmetric matrix. Therefore you have to preprocess a non symmetric matrix.

# Chapter 4

## Results

- If you need just one eigenvalue you should use power iteration or inverse iteration because of fast convergence. For more than one we would use orthogonal iteration. And if you need all eigenvalues of a matrix you should take the QR iteration.
- QR iteration is faster with preprocessing (Lanczos). In general all methods provide faster convergence if the matrix is preprocessed.
- Probably the slowest algorithm is the Jacobi method which we would not recommend to use.
- For calculation of an eigenvalue near to some scalar  $x$  someone should use power or inverse iteration with shifting.
- The choice of the value for the tolerance plays an important role on the convergence of the chosen algorithm. Sometimes taking a too small value can lead to no convergence. On the other side taking a too big value can lead to bad results.

# Chapter 5

## Conclusion

EigenCalc works well for matrices with real eigenvalues, but doesn't support complex eigenvalues. This would be a possible extension.

Another point that could be improved is the handling and maybe to provide a GUI or a graphical display of eigenvectors or eigenvalues.

Maybe it would be also a good extension to make it possible to view also the single steps of an iteration.

Another good extension could be to implement the eigenvalue calculation for not square matrices.