Miniprojekt 1
Beräkningsvetenskap 3
Fredrik Gustafsson F3C
2018-01-29

In this mini-project one was supposed to try two different methods to find the largest and smallest eigenvalue of an n by n matrix, the methods being the power method and inverse iteration respectively. These two methods had to be implemented in matlab, taking a general matrix A as input and giving the largest/smallest eigenvalue and the corresponding eigenvector as output.

To start of one had to create the general matrix A being defined as $A = M^{-}1K$ where M and K was two matrices given in the instructions.To solve this a small program, see appendix, was made in which one could easily change the size of the matrix. After having an easy way to change the size of the A matrix the implementation of the Power method started.

The power method works by taking an initial guess for the solution to the equation Ax=B and a guess for the eigenvalue. After which it iterates until the eigenvalue is within a set tolerance, in this case 10^{-4}. This is done by computing $xnew = A * x$ for each iteration meaning that the solution will become a bit better each time. The largest eigenvalue can then be approximated as the norm of the xnew vector and then updating the x-vector as $x = \frac{xnew}{lamda}$. Ending by calculating the difference between the two latest lamdas and if this difference is less than the tolerance ending the iteration. To then get the eigenvector the final x vector gets divided by it's norm. This was done for 3 different sizes 20x20, 40x40 and 80x80 giving corresponding eigenvalues of $4.778 * 10^3$, $1.9178 * 10^4$ and $7.6778 * 10^4$. These values can be compared with the values which the built in function eig() gives which are $4.7779 * 10^3$, $1.9178 * 10^4$ and $7.6778 * 10^4$. From this we can see that the power method gives a good approximation to the eigenvalues. One can also study the rate of convergence (roc) by having a counter built into the function. The resulting graph is shown below in figure 1.

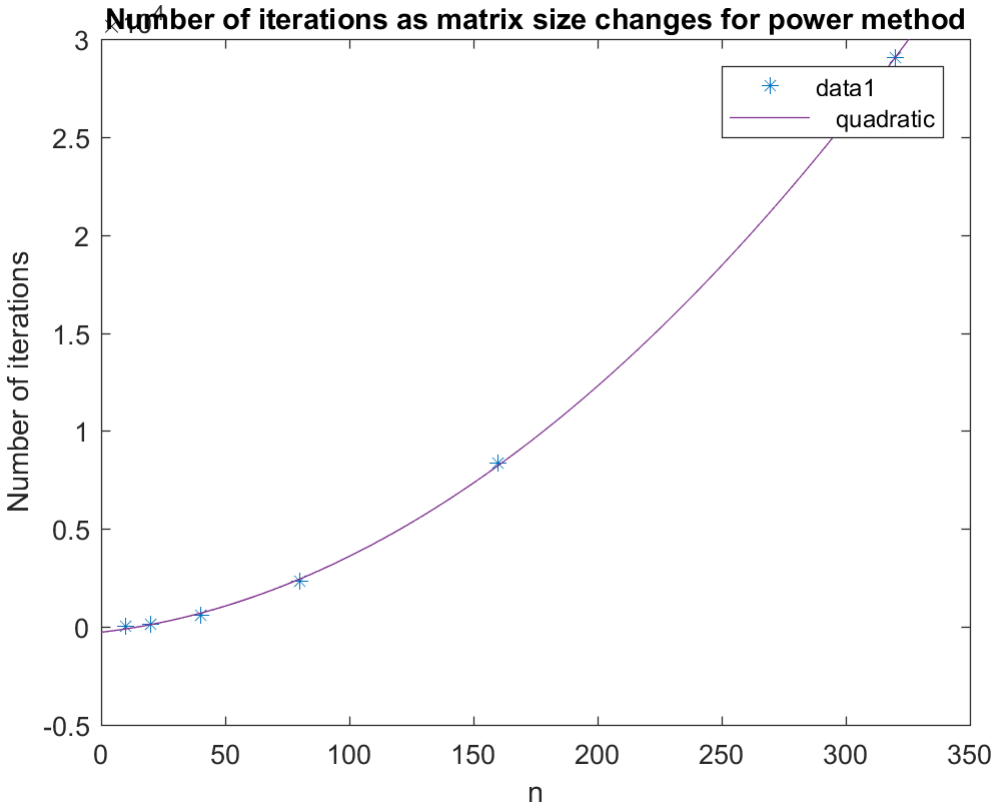

Figure 1. Number of iterations required as the matrix size increase.

From this we can see that as the size of the matrix increases the computational time also increases with a relation which seems to follow n^2. This should be expected since the convergence depends on the norm of the quotient between the second largest and largest eigenvalue and as this quotient approaches 1 the roc becomes slower and with larger matrices more eigenvalues will likely exist slowing down the roc. Lastly we were supposed to plot the eigenvector which is shown in figure 2.

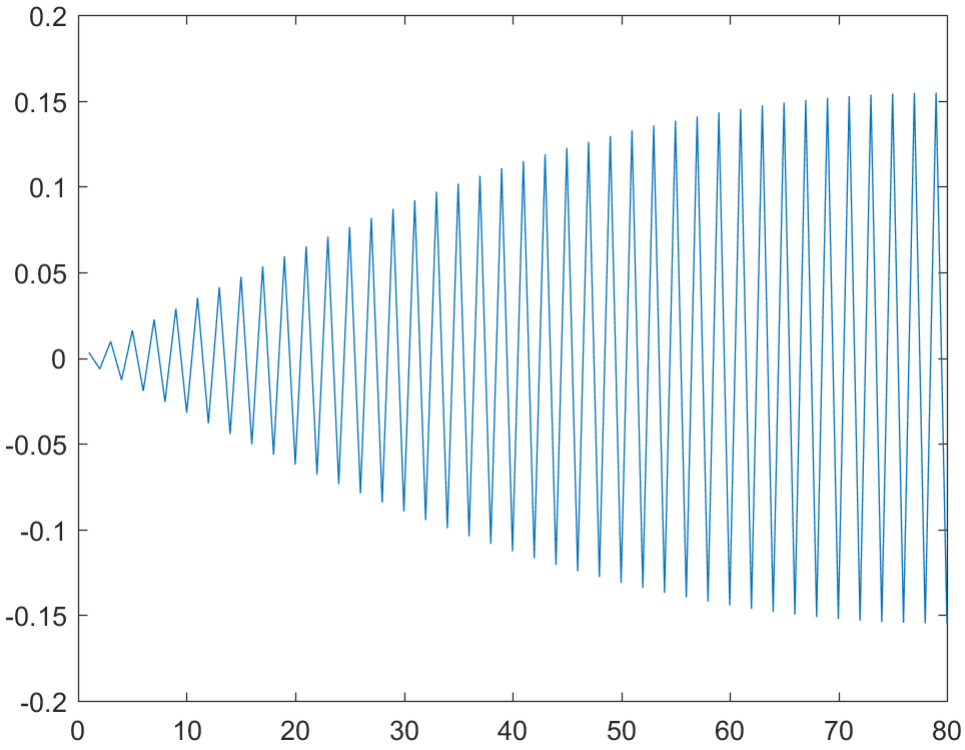

Figure 2. The eigenvector for a matrix with size 80 plotted.

For the second part the inverse power method was supposed to be used to find the smallest eigenvalue. This works similar to the power method with the difference being that instead of calculating the product xnew=Ax one calculates xnew=A\x and in the end lamda is given by dividing 1 with the norm of xnew. Once again we can see that the implmented function gives values close to the $eig()$ function with n being 20, 40 and 80 giving lamdas of 2.4683, 2.4675 and 2.4675 while eig(A) gives 2.4686697, 2.4677 and 2.46748. A big difference however between the two methods is that the inverse power method is quicker taking just 3 steps for n being 20 and 40 and 4 steps for n being 80. This is due to the roc often being quicker for the inverse iteration since the quotient depends on how close the largest and second largest calculated eigenvalues are.

1 Appendix

Matlab code

```
    clear all
clc
close all
%Skapar matris A som används i miniproject 1
n=80 %antal columner
h=1/n

a=ones(1,n-1); %vektor med ettor för övre och under diagonal


b=ones(1,n) ;
%diagonal
for i=1:n-1;
b(i)=2;
end
b(n)= 1;

c=diag(b); %diagonal K matris

d=ones(1,n);
for i=1:n-1;
d(i)=4;
end
d(n)= 2;
%diagonal M matris


K=1/h.*(diag(-a,1)+diag(-a,-1)+diag(b));

M=h/6.*(diag(a,1)+diag(a,-1)+diag(d));

A = M^-1 * K;
```

Power method code

```
function [ x,lamda ] = eig_power(A)
%Calculate largest eigenvalue with the power function to the matrix A
% and its corresponding eigenvector
tol=1e-4; %how close before stopping iterations
n=length(A);
x=ones(n,1); %Initial solution
x=x/norm(x);
```

```
lamda=0; %initial eigenvalue
dd=1; %Value larger than tol to start while loop
counter=0;
while dd¿tol
counter=counter+1;
lamdaold=lamda; %Set previous lamda to current value
xnew=A*x; %Solve equation
lamda=norm(xnew); %Update lamda
x=xnew/lamda;
dd= (lamda-lamdaold);
end
x = x/norm(x)
lamda
counter
plot(x)
end
```

Inverse iteration code

```
function [ x,lamda ] = inv_eig_power(A)
% Inverse iteration to find smalles eigenvalue to matrix A and
% the coresponding eigenvector

tol=1e-4;
n=length(A);
x=ones(n,1);
x=x/norm(x);

lamda=0; %% Initial guess for eigenvalue
dd=1; % Value larger than tol in to start while loop
counter=0;
while dd¿tol
;
lamdaold=lamda; %Set previous lamda to current value
xnew=A; %Solve equation
lamda=norm(xnew); % Update lamda
x=xnew/lamda; % Calculate eigenvector
dd= (lamda-lamdaold); %Calculate difference between eigenvalues
counter=counter+1
end
x = x/norm(x) %Final eigenvector
lamda=1/lamda %Final eigenvalue
counter %Number of steps

end
```