MTH5110 **Introduction to Numerical Computing**

Problem Sheet 8

Problem 15
Consider the matrix

$$A = \begin{bmatrix} 4.0 & 1.0 & 1.0 \\ 0.0 & 2.0 & 1.0 \\ -2.0 & 0.0 & 9.0 \end{bmatrix}$$

a) Calculate the inverse and the transpose of *A* using Maple.

```
> A:=<<4.0|1.0|1.0>,<0.0|2.0|1.0>,<-2.0|0.0|9.0>>;
```

$$A := \begin{bmatrix} 4.0 & 1.0 & 1.0 \\ 0. & 2.0 & 1.0 \\ -2.0 & 0. & 9.0 \end{bmatrix} \quad (1)$$

```
> with(LinearAlgebra):
> Transpose(A);
```

$$\begin{bmatrix} 4.0 & 0. & -2.0 \\ 1.0 & 2.0 & 0. \\ 1.0 & 1.0 & 9.0 \end{bmatrix} \quad (2)$$

```
> A^(-1);
```

$$\begin{bmatrix} 0.243243243243243 & -0.121621621621622 & -0.0135135135135135 \\ -0.0270270270270270 & 0.513513513513513 & -0.0540540540540541 \\ 0.0540540540540541 & -0.0270270270270270 & 0.108108108108108 \end{bmatrix} \quad (3)$$

b) Find the eigenvalues of *A* using the Maple procedure *Eigenvalues*, which is part of the *LinearAlgebra* package.

```
> Eigenvalues(A);
```

$$\begin{bmatrix} 8.48534949329733 + 0.\,I \\ 4.63182668375403 + 0.\,I \\ 1.88282382294864 + 0.\,I \end{bmatrix} \quad (4)$$

By default, Maple looks for eigenvalues in the complex plane; hence the output.
The eigenvalues of A are real though: they all have a zero imaginary part.

c) Find the eigenvectors of *A* using the Maple procedure *Eigenvectors*. Where are the eigenvectors in Maple's output?

```
> Eigenvectors(A);
```

$$\begin{bmatrix} 8.48534949329733 + 0.\,I \\ 4.63182668375403 + 0.\,I \\ 1.88282382294864 + 0.\,I \end{bmatrix},$$

$$\begin{bmatrix} 0.246473798094566 + 0.\,I & -0.898063212561229 + 0.\,I & -0.382631934381108 + 0.\,I \\ 0.147691317070026 + 0.\,I & -0.156235511989670 + 0.\,I & 0.917622740792180 + 0.\,I \\ 0.957829808324506 + 0.\,I & -0.411184789404387 + 0.\,I & -0.107523524741419 + 0.\,I \end{bmatrix} \quad \textbf{(5)}$$

The command *Eigenvectors* computes both, eigenvalues and eigenvectors. The eigenvectors are the second part of the output, the three columns are the three eigenvectors.

To obtain the second part of the output, specify the second "component" of the output using [2]

```
> Eigenvectors(A)[2];
```

$$\begin{bmatrix} 0.246473798094566 + 0.\,I & -0.898063212561229 + 0.\,I & -0.382631934381108 + 0.\,I \\ 0.147691317070026 + 0.\,I & -0.156235511989670 + 0.\,I & 0.917622740792180 + 0.\,I \\ 0.957829808324506 + 0.\,I & -0.411184789404387 + 0.\,I & -0.107523524741419 + 0.\,I \end{bmatrix} \quad \textbf{(6)}$$

To obtain an element of this matrix output, use [n,m], e.g. [1,2] for the first component of the second eigenvector. To obtain the first column, i.e., the first eigenvector specifiy a range from 1 to 3, i.e., [1..3,1]

```
> Eigenvectors(A)[2][1..3,1];
```

$$\begin{bmatrix} 0.246473798094566 + 0.\,I \\ 0.147691317070026 + 0.\,I \\ 0.957829808324506 + 0.\,I \end{bmatrix} \quad \textbf{(7)}$$

d) Show the result of *A* multiplied by the vector $u = (1.0,\ 3.0,\ 1.0)^T$.

```
> u:=<1.0,3.0,1.0>;
```

$$u := \begin{bmatrix} 1.0 \\ 3.0 \\ 1.0 \end{bmatrix} \quad \textbf{(8)}$$

```
> A.u;
```

$$\begin{bmatrix} 8. \\ 7. \\ 7. \end{bmatrix} \quad \textbf{(9)}$$

e) Write a loop that implements the recursive relation $v_{n+1} = A\,v_n$ for *n=0,1,...,9* . Use the vector *u* as initial (seed) vector v_0. Display the iterates v_n and the normalised vectors $\frac{v_n}{|v_n|}$. How does the result relate to part c).

```
> v:=u:
```

```
print(0,v,v/VectorNorm(v,Euclidean)):
for k from 0 to 9 do
   v:=A.v:
   print(k+1,v,v/VectorNorm(v,Euclidean)):
end do:
```

$$0,\begin{bmatrix}1.0\\3.0\\1.0\end{bmatrix},\begin{bmatrix}0.301511344600000\\0.904534033800000\\0.301511344600000\end{bmatrix}$$

$$1,\begin{bmatrix}8.\\7.\\7.\end{bmatrix},\begin{bmatrix}0.628539361120000\\0.549971940980000\\0.549971940980000\end{bmatrix}$$

$$2,\begin{bmatrix}46.\\21.\\47.\end{bmatrix},\begin{bmatrix}0.666316875620000\\0.304188138870000\\0.680802025090000\end{bmatrix}$$

$$3,\begin{bmatrix}252.\\89.\\331.\end{bmatrix},\begin{bmatrix}0.592349532264000\\0.209202810998000\\0.778046409442000\end{bmatrix}$$

$$4,\begin{bmatrix}1428.\\509.\\2475.\end{bmatrix},\begin{bmatrix}0.492007659469200\\0.175372478060100\\0.852744367777500\end{bmatrix}$$

$$5,\begin{bmatrix}8696.\\3493.\\19419.\end{bmatrix},\begin{bmatrix}0.403302460409680\\0.161998101910190\\0.900612980530770\end{bmatrix}$$

$$6,\begin{bmatrix}57696.\\26405.\\1.57379\ 10^{5}\end{bmatrix},\begin{bmatrix}0.340011167168256\\0.155608618779080\\0.927458012301944\end{bmatrix}$$

$$7,\begin{bmatrix}4.14568\ 10^{5}\\2.10189\ 10^{5}\\1.301019\ 10^{6}\end{bmatrix},\begin{bmatrix}0.300073290066088\\0.152139346900149\\0.941705707552179\end{bmatrix}$$

$$8,\begin{bmatrix}3.169480\ 10^{6}\\1.721397\ 10^{6}\\1.0880035\ 10^{7}\end{bmatrix},\begin{bmatrix}0.276513806620745\\0.150179220937040\\0.949202990401244\end{bmatrix}$$

$$9,\ \begin{bmatrix} 2.5279352\ 10^{7} \\ 1.4322829\ 10^{7} \\ 9.1581355\ 10^{7} \end{bmatrix},\ \begin{bmatrix} 0.263107793912354 \\ 0.149072173241382 \\ 0.953179823499989 \end{bmatrix}$$

$$10,\ \begin{bmatrix} 2.07021592\ 10^{8} \\ 1.20227013\ 10^{8} \\ 7.73673491\ 10^{8} \end{bmatrix},\ \begin{bmatrix} 0.255624460435527 \\ 0.148452946530815 \\ 0.955310346034562 \end{bmatrix} \quad \textbf{(10)}$$

We see that the normalised vectors converge quite slowly towards some limit. The limit seems to be the leading eigenvector computed in part c)
see eq.(7). The non normalised vectors increase in length (as the corresponding eigenvalue is larger than one).

Problem 16

a) Amend the procedure *simple_pw* for the power method. Add *imax* as an input parameter which specifies the maximum number of iteration steps. Display an error message if the method fails to converge. Specify the types of the input parameters (use the types *Matrix* and *Vector*; you can even specify the type of entries using, e.g., *Matrix(float)*).

* Change the while loop to a for loop with counter from 1 to imax
* Check the termination condition within the loop
* Display an error message if the end of the loop has been reached

```
> my_pm:=proc(A::Matrix,v::Vector,tol::float,imax::integer)
    local w,u,lambda,k;
      # transcribe input vector
     w:=v;
    # the following part is not needed any more because of
    # the changed loop structure
       # assign values to lambda and u so that loop starts
     # lambda:=0;
      # u:=v;
     # iteration loop with termination condition
     # loop with counter
    for k from 1 to imax do
          # normalisation
      u:=w/LinearAlgebra[VectorNorm](w,Euclidean);
          # iteration of vector
       w:=A.u;
          # estimate of eigenvalue
      lambda:=w.u;
      # termination condition
      if LinearAlgebra[VectorNorm](w-lambda*u,Euclidean)<tol then
          return lambda,u;
       end if;
    end do;
    # error message
    error "no convergence within", imax, "steps";
```

```
   end proc;
my_pm := proc(A::Matrix, v::Vector, tol::float, imax::integer)
    local w, u, λ, k;
    w := v;
    for k to imax do
        u := w / LinearAlgebra[LinearAlgebra:-VectorNorm](w, Euclidean);
        w := `.`(A, u);
        λ := `.`(w, u);
        if LinearAlgebra[LinearAlgebra:-VectorNorm](w − λ * u, Euclidean) < tol then
            return λ, u
        end if
    end do;
    error "no convergence within", imax, "steps"
end proc
```

(11)

Test the procedure with some input

```
> my_pm(<<2.0|0.5>,<-0.5|-1.0>>,<1.0,1.0>,0.001,50);
```

$$1.91393536567372169,\ \begin{bmatrix} 0.985551514938360 \\ -0.169375947291432 \end{bmatrix} \tag{12}$$

```
> my_pm(<<2.0|0.5>,<-0.5|-1.0>>,<1.0,1.0>,0.001,10);
Error, (in my_pm) no convergence within, 10, steps
```

Seems to work.

b) Consider the matrix

$$A = \begin{bmatrix} 2.0 & 2.5 & 2.0 \\ 3.0 & -1.0 & 0.5 \\ 3.0 & -2.0 & 1.5 \end{bmatrix}$$

Apply the power method to *A* using the vector $v = (1.0, 1.0, 1.0)^T$ as your initial vector and $\varepsilon = 10^{-4}$ as the error threshold, to compute the leading eigenvalue and the leading eigenvector. Compare your output with the one obtained with Maple's *Eigenvectors* (part of the *LinearAlgebra* package).

```
> A:=<<2.0|2.5|2.0>,<3.0|-1.0|0.5>,<3.0|-2.0|1.5>>;
```

$$A := \begin{bmatrix} 2.0 & 2.5 & 2.0 \\ 3.0 & -1.0 & 0.5 \\ 3.0 & -2.0 & 1.5 \end{bmatrix} \tag{13}$$

```
> v:=<1.0,1.0,1.0>;
```

$$v := \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix} \tag{14}$$

```
> lambda,u:=my_pm(A,v,10.0^(-4),50);
```

$$\lambda, u := 4.63103232685195998, \begin{bmatrix} 0.770663368157923 \\ 0.450598475690449 \\ 0.450598475690449 \end{bmatrix} \quad \textbf{(15)}$$

We have stored the eigenvalue in the variable *lambda* and the eigenvector in the variable *u*, for later use !

```
> Eigenvectors(A);
```

$$\begin{bmatrix} 4.63104367406501 + 0.\,I \\ -3.13104367406501 + 0.\,I \\ 1.00000000000000 + 0.\,I \end{bmatrix}, \quad \textbf{(16)}$$

$$\begin{bmatrix} 0.770670184027170 + 0.\,I & 0.527027414040391 + 0.\,I & -0.431456249888387 + 0.\,I \\ 0.450592647216711 + 0.\,I & -0.600933484193508 + 0.\,I & -0.452001785597358 + 0.\,I \\ 0.450592647216711 + 0.\,I & -0.600933484193508 + 0.\,I & 0.780730356940890 + 0.\,I \end{bmatrix}$$

The output of the power method looks quite accurate. If we want to compare the two outputs quantitatively take the difference
(see above how the select a particulr element of the output of *Eigenvectors* !)

```
> lambda-Eigenvectors(A)[1][1];
```

$$-0.0000113470650058289 - 0.\,I \quad \textbf{(17)}$$

```
> u-Eigenvectors(A)[2][1..3,1];
```

$$\begin{bmatrix} -0.00000681586924722311 + 0.\,I \\ 0.00000582847373731266 + 0.\,I \\ 0.00000582847373731266 + 0.\,I \end{bmatrix} \quad \textbf{(18)}$$

The differences are in both cases within the tolerance of 10^{-4}.

c) Perform a single deflation step using, e.g., the procedure *simple_deflat* of the lecture notes (or otherwise).

```
> pindex:=proc(v)
    local i,ndim;
     # length of input vector
    ndim:=LinearAlgebra[Dimension](v);
     # loop over all components
    for i from 1 to ndim do
        # checks for nonzero component
      if abs(v[i])>0 then
          # output
        return i;
      end if;
    end do;
  end proc:
> simple_deflat:=proc(A,u)
    local p,ap;
     # computes component
```

```
    p:=pindex(u);
     # pth row of input matrix
    ap:=A[p];
     # output, deflated matrix
    return A-(u.ap)/u[p];
  end proc:
```

Run the deflation procedure with input A and u (leading eigenvector, not the seed v!)

```
> B:=simple_deflat(A,u);
```

$$B := \begin{bmatrix} 0. & 0. & 0. \\ 1.83062178816805 & -2.46172276478993 & -0.669378211831946 \\ 1.83062178816805 & -3.46172276478993 & 0.330621788168054 \end{bmatrix} \quad \textbf{(19)}$$

d) Apply the power method again to the matrix *B* computed in part c).

```
> lambda_two,u_tmp:=my_pm(B,v,10.0^(-4),50);
```

$$lambda\_two,\ u\_tmp := -3.13110097398589060,\ \begin{bmatrix} 0. \\ -0.707106780888901 \\ -0.707106780888900 \end{bmatrix} \quad \textbf{(20)}$$

e) Confirm that the two eigenvalues you have obtained coincide (within the margin of error) with the output of the Maple procedure *Eigenvalues*.

```
> lambda,lambda_two;
```

$$4.63103232685195998,\ -3.13110097398589060 \quad \textbf{(21)}$$

Seems to be the leading and the subleading eigenvalue of A (see eq.(16)). Compare quantitatively:

```
> lambda-Eigenvectors(A)[1][1], lambda_two-Eigenvectors(A)[1][2];
```

$$-0.0000113470650058289 - 0.\,I,\ -0.0000572999349937042 - 0.\,I \quad \textbf{(22)}$$

Of course the second eigenvector of A (see eq.(16)) and the leading eigenvector of B (see eq.(20)) differ (i.e. deflation changes the eigenvectors!).