

Creating worksheets with nice numbers and nice diagonalization

Mike May, S.J.

Saint Louis University

For teaching purposes we want to create matrices that are big enough to be make exercises nontrivial but nice enough to do and with nice enough numbers so that the students are not distracted by the arithmetic. For example, we may want 3 by 3 matrices that diagonalize nicely. (Randomly chosen 3 by 3 matrices often have problems with the step that solves the cubic characteristic polynomial. The answers may quite ugly.) The trick is to work backwards. Start with the matrix in nice form and then find a conjugation with small enough numbers.

```
> with(LinearAlgebra):
```

▼ Matrices with nice eigenvalues and eigenvectors

We start with the diagonal matrix that gives the eigenvalues.

```
> MatSize := 3:
MaxDiag := 6:
f1 := (i,j) -> rand(-MaxDiag..MaxDiag)():
D1 := Matrix(MatSize,MatSize,f1,shape=diagonal);
```

$$D1 := \begin{bmatrix} 6 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -5 \end{bmatrix} \quad (1.1)$$

We next create a nice matrix to conjugate D with. For the time being, I think a nice matrix has small integer valued entries and has determinant plus or minus 1 so that its inverse is also nice. We use the method of looking at random matrices until we find a nice one.

```
> test := 0:
count := 0:
MaxValP := 3:
while test = 0 do:
P1 := RandomMatrix(MatSize,MatSize,generator=rand(-MaxValP..
MaxValP)):
d1 := abs(Determinant(P1)):
if d1 = 1 then test := 1 end if:
count := count + 1:
end do:
count; P1; Q1 := P1^(-1);
```

$$\begin{matrix} 20 \\ \begin{bmatrix} 0 & 2 & -3 \\ 2 & -3 & 0 \\ 1 & -3 & 2 \end{bmatrix} \end{matrix}$$

$$Q1 := \begin{bmatrix} -6 & 5 & -9 \\ -4 & 3 & -6 \\ -3 & 2 & -4 \end{bmatrix} \quad (1.2)$$

Now we conjugate to get our matrix. Just to verify, we compute eigenvalues and eigenvectors. The eigenvalues should be the entries of the diagonal matrix. The eigenvectors are scalar multiples of the columns of P1.

```
> M1 := P1.D1.Q1;
   Eigenvectors(M1);
```

$$M1 := \begin{bmatrix} -37 & 24 & -48 \\ -84 & 69 & -126 \\ -18 & 19 & -32 \end{bmatrix}$$

$$\begin{bmatrix} -5 \\ 6 \\ -1 \end{bmatrix}, \begin{bmatrix} \frac{-3}{2} & 0 & \frac{-2}{3} \\ 0 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (1.3)$$

It is useful to put all the code in a single block that can be repeated until the answer is sufficiently pretty.

```
> MatSize := 3:   MaxDiag := 6:
   MaxValP := 3:

   f1 := (i,j) -> rand(-MaxDiag..MaxDiag)():
   D1 := Matrix(MatSize,MatSize,f1,shape=diagonal):
   test := 0:
   while test = 0 do:
     P1 := RandomMatrix(MatSize,MatSize,generator=rand(-MaxValP..
   MaxValP)):
     d1 := abs(Determinant(P1)):
     if d1 = 1 then test := 1 end if:
   end do:
   Q1 := P1^(-1):
   M1 := P1.D1.Q1:
   print("D1"=D1, "P1" = P1, "Q1"=Q1, "M1"=M1);
   Eigenvectors(M1);
```

$$"D1" = \begin{bmatrix} -4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}, "P1" = \begin{bmatrix} -1 & 1 & -1 \\ -1 & -2 & -3 \\ 1 & 0 & 2 \end{bmatrix}, "Q1" = \begin{bmatrix} -4 & -2 & -5 \\ -1 & -1 & -2 \\ 2 & 1 & 3 \end{bmatrix}$$

$$, "M1" = \begin{bmatrix} -23 & -13 & -32 \\ -22 & -8 & -26 \\ 24 & 12 & 32 \end{bmatrix}$$

(1.4)

$$\begin{bmatrix} 3 \\ 2 \\ -4 \end{bmatrix}, \begin{bmatrix} \frac{-1}{2} & \frac{-1}{2} & -1 \\ 1 & \frac{-3}{2} & -1 \\ 0 & 1 & 1 \end{bmatrix} \quad (1.4)$$

> The diagonal matrix is set as M1. Repeat until you get a matrix that looks nice.
>

Repeated and complex eigenvalues

We can vary the technique above if we are interested in matrices that are not diagonalizable, either because they have complex eigenvalues or because of non-trivial Jordan blocks. In those cases we can use the CompanionMatrix command to get the nice form that we start with.

```
> Poly1 := (x-1)^2*(x-2);
Poly2 := expand(Poly1);
deg := degree(Poly2,x);
N1 := CompanionMatrix(Poly2,x);
Eigenvectors(N1);
```

$$Poly1 := (x - 1)^2 (x - 2)$$

$$Poly2 := x^3 - 4x^2 + 5x - 2$$

$$deg := 3$$

$$N1 := \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & -5 \\ 0 & 1 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 1 \\ -3 & 0 & -2 \\ 1 & 0 & 1 \end{bmatrix}$$

(2.1)

Once again we put this in a block of code to give us a nice but disguised example.

```
> Poly1 := (x-1)^2*(x-2):
MaxValP := 6:

Poly2 := expand(Poly1):
MatSize := degree(Poly2,x):
N1 := CompanionMatrix(Poly2,x):
test := 0:
while test = 0 do:
  P1 := RandomMatrix(MatSize,MatSize,generator=rand(-MaxValP..
MaxValP)):
  d1 := abs(Determinant(P1)):
  if d1 = 1 then test := 1 end if:
end do:
Q1 := P1^(-1):
M1 := P1.N1.Q1:
print("N1"=N1, "P1" = P1, "Q1"=Q1, "M1"=M1);
Eigenvectors(M1);
```

```
CharacteristicPolynomial(M1,x);
```

$$\begin{aligned}
\text{"N1"} &= \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & -5 \\ 0 & 1 & 4 \end{bmatrix}, \text{"P1"} = \begin{bmatrix} 5 & -4 & -6 \\ -1 & -2 & 1 \\ 4 & -5 & -5 \end{bmatrix}, \text{"Q1"} = \begin{bmatrix} 15 & 10 & -16 \\ -1 & -1 & 1 \\ 13 & 9 & -14 \end{bmatrix} \\
\text{"M1"} &= \begin{bmatrix} 24 & 20 & -26 \\ 125 & 87 & -135 \\ 99 & 72 & -107 \end{bmatrix} \\
& \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{7}{9} & \frac{8}{9} & 0 \\ \frac{4}{9} & \frac{5}{18} & 0 \\ 1 & 1 & 0 \end{bmatrix} \\
& x^3 - 4x^2 + 5x - 2 \tag{2.2}
\end{aligned}$$

Executing the block above with Poly1 = (x-1)^2*(x-2) gives a matrix M1 where Rank(M1-I)=2 and Rank((M1-I)^2)=1.

Executing the block above with Poly1 = (x^2-x+1)*(x-2) gives a matrix M1 with complex eigenvectors

>

Special construction tools

There are a number of commands for Matrix construction that are quite useful for special cases. The commands are gathered here. They are simple enough to understand that an example will suffice.

```
> I3 := IdentityMatrix(3);
```

$$I3 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

```
> Z3 := ZeroMatrix(3);
```

$$Z3 := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.2}$$

```
> D1 := DiagonalMatrix([2,3,4]);
```

$$D1 := \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \tag{3.3}$$

```
> ScalarMatrix(x,2,3);
```

(3.4)

$$\begin{bmatrix} x & 0 & 0 \\ 0 & x & 0 \end{bmatrix} \quad (3.4)$$

```
> ScalarMatrix(x,2);
```

$$\begin{bmatrix} x & 0 \\ 0 & x \end{bmatrix} \quad (3.5)$$

```
> M2 := JordanBlockMatrix([[1,2],[3,1]]);
```

$$M2 := \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad (3.6)$$

```
> RandomMatrix(3,4);
```

$$\begin{bmatrix} 82 & 55 & 30 & 50 \\ -30 & 44 & 80 & -64 \\ 57 & 10 & 35 & 51 \end{bmatrix} \quad (3.7)$$

```
> RandomMatrix(3,4,generator=rand(10));
```

$$\begin{bmatrix} 1 & 4 & 5 & 6 \\ 2 & 2 & 6 & 8 \\ 6 & 3 & 6 & 5 \end{bmatrix} \quad (3.8)$$

```
> RandomMatrix(3,4,generator=rand(-10..10));
```

$$\begin{bmatrix} -10 & 9 & 1 & 7 \\ 5 & 2 & 3 & -10 \\ -1 & -6 & 1 & 2 \end{bmatrix} \quad (3.9)$$

```
>
```

The general matrix constructor is quite powerful in that you can specify a function of i and j to fill in the entries.

```
> f1 := (i,j)-> rand(-5..5)():
Matrix(3,3,f1);
```

$$\begin{bmatrix} 5 & 4 & -5 \\ 0 & -2 & -3 \\ 3 & -2 & 4 \end{bmatrix} \quad (3.10)$$

With a modification of $f1$ we can produce triangular and unitriangular matrices

```
> f1 := (i,j)-> if i>j then 0
else rand(-5..5)()end if:
Matrix(3,3,f1);
f1 := (i,j)-> if i>j then 0
elif i=j then 1
else rand(-5..5)()end if:
Matrix(3,3,f1);
```



$$\begin{bmatrix} 4 & -4 & 0 \\ 0 & -3 & -1 \\ 0 & 0 & 0 \\ 1 & -4 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

(3.11)

