with(LinearAlgebra);
A := Matrix(4, datatype = complex(sfloat));
B := RandomMatrix(4, generator = 0 .. 1.0);
for j from 1 to 4 do
  for k from 1 to 4 do
    A(j, k) := B(j, k) + 0.*I;
  od:
od:
A;

We did the complex form of the Schur Lemma in class.
The real form differs slightly (due to the fact that eigenvalues of a
real matrix may be complex).

\[
\begin{align*}
\text{SchurForm}(A) ; \\
\begin{bmatrix}
1.82491377021677 & -3.28320558267607 & 10^{18} I & 0.367295500855236 + 1.38777878078145 & 10^{17} I, \\
0.0413085077236971 & -5.89805981832114 & 10^{17} I & 0.547348045752282 + 6.93889390390723 & 10^{17} I, \\
0. + 0. I & 0.34225005775036 & -7.07841101058017 & 10^{18} I & -0.180052284242930 \\
-1.52655655885959 & 10^{16} I & -0.331629585829796 & -2.42861286636753 & 10^{17} I, \\
0. + 0. I & 0. + 0. I & -0.222536246858923 & 5.05723412908172 & 10^{18} I, \\
-1.83880688453542 & 10^{16} I, \\
0. + 0. I & 0. + 0. I & 0. + 0. I & 0.0985782523629265 + 6.46076705521729 & 10^{19} I]
\end{bmatrix}
\end{align*}
\]

\[
T, Z := \text{SchurForm}(A, \text{output}=['T', 'Z']):
\]
\[
\begin{align*}
T; \\
Z; \\
\begin{bmatrix}
1.82491377021677 & -3.28320558267607 & 10^{18} I & 0.367295500855236 + 1.38777878078145 & 10^{17} I, \\
0.0413085077236971 & -5.89805981832114 & 10^{17} I & 0.547348045752282 + 6.93889390390723 & 10^{17} I, \\
0. + 0. I & 0.34225005775036 & -7.07841101058017 & 10^{18} I & -0.180052284242930 \\
-1.52655655885959 & 10^{16} I & -0.331629585829796 & -2.42861286636753 & 10^{17} I, \\
0. + 0. I & 0. + 0. I & -0.222536246858923 & 5.05723412908172 & 10^{18} I, \\
-1.83880688453542 & 10^{16} I, \\
0. + 0. I & 0. + 0. I & 0. + 0. I & 0.0985782523629265 + 6.46076705521729 & 10^{19} I]
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
(5) \\
\begin{bmatrix}
0.219036761512636 + 6.93889390390723 & 10^{18} I & 0.945299512555574 & 0. I & -0.216964094969587 \\
-1.38777878078145 & 10^{17} I & -0.106575373151025 & -2.42861286636753 & 10^{17} I, \\
0.455161838581246 & -1.04883408558608 & 10^{17} I & 0.102682443569910 & -3.64291929955129 & 10^{17} I \\
0.883146249743396 & + 0. I & 0.0483396115626368 & + 1.95156391047391 & 10^{17} I, \\
0.661431241889163 + 0. I & -0.287618457568129 & + 9.71445146547012 & 10^{17} I, \\
-2.0172597803089244 & + 1.38777878078145 & 10^{17} I, \\
0.554399954958834 & -2.08166817117217 & 10^{17} I, \\
0.314117367888374 & + 0. I & 0.762122167976819 & + 0. I]
\end{bmatrix}
\end{align*}
\]

\[
Z.\text{HermitianTranspose}(Z);
\]
\[
\begin{align*}
(6) \\
\begin{bmatrix}
1.000000000000000 + 0. I & 1.91686944095437 & 10^{-16} & -8.17806307158922 & 10^{-17} I, \\
-3.31534262854856 & 10^{-17} I & -1.1022302462516 & 10^{-16} & 2.03714197194047 & 10^{-17} I, \\
1.91686944095437 & 10^{-16} & + 8.17806307158922 & 10^{-17} I, \\
0.9999999999999999 & + 0. I & 6.90419943438769 & 10^{-16} & -3.00585298159696 & 10^{-17} I, \\
7.00828282494630 & 10^{-16} & + 2.13288663537235 & 10^{-17} I, \\
-4.44089209850063 & 10^{-16} & + 3.31534262854856 & 10^{-17} I, \\
+ 3.00585298159696 & 10^{-17} I, \\
1.000000000000000 & + 0. I & 1.11022302462516 & 10^{-16} & -1.35997836510819 & 10^{-17} I, \\
-1.11022302462516 & 10^{-16} & -2.03714197194047 & 10^{-17} I, \\
7.00828282494630 & 10^{-16} & -2.13288663537235 & 10^{-17} I, \\
1.11022302462516 & 10^{-16} & + 1.35997836510819 & 10^{-17} I, \\
1.000000000000000 & + 0. I]
\end{bmatrix}
\end{align*}
\]

\[
(7) \\
\begin{align*}
Z . T . \text{HermitianTranspose}(Z) - A;
\end{align*}
\]
\[
\begin{align*}
(6.66133814775094 & 10^{-16} & -1.02105458339470 & 10^{-18} I, \\
3.19189119570733 & 10^{-16}
\end{align*}
\]
L := SingularValues(A, output = 'list');
U, Vt := SingularValues(A, output = ['U', 'Vt']);
DiagL := DiagonalMatrix(L);
U . DiagL . Vt - A;

\[
\begin{align*}
L &= [1.94402988081538, 0.607037331707257, 0.221738830185565, 0.0523609676521310] \\
U, Vt &= \begin{bmatrix}
-0.404701142530598 + 0. I, 0.606705455105708 + 0. I, 0.651351998873087 + 0. I, 0.209442234383227 + 0. I, \\
-0.671486281758066 + 0. I, -0.279992900649665 + 0. I, 0.0632601445794775 + 0. I, -0.683160525136148 + 0. I, \\
-0.584293627930040 + 0. I, -0.336460372877923 + 0. I, -0.270591427773017 + 0. I, 0.687150386057493 + 0. I], \\
\text{DiagL} &= \begin{bmatrix}
1.94402988081538 & 0 & 0 & 0 \\
0 & 0.607037331707257 & 0 & 0 \\
0 & 0 & 0.221738830185565 & 0 \\
0 & 0 & 0 & 0.0523609676521310
\end{bmatrix} \\
U \cdot \text{DiagL} \cdot Vt - A &= \begin{bmatrix}
-6.66133814775094 10^{-16} + 0. I, -4.16333634234434 10^{-17} + 0. I, 1.38777878078145 10^{-15} + 0. I, \\
1.38777878078145 10^{-15} + 0. I, \\
-7.2164496606352 10^{-16} + 0. I, -3.33066907387547 10^{-16} + 0. I, 9.99200722162641 10^{-16} + 0. I, \\
5.55111512312578 10^{-17} + 0.1, \\
2.77555756156289 10^{-16} + 0. I, -1.11022302462516 10^{-16} + 0. I, -8.32667268468867 10^{-16} + 0. I, \\
-5.55111512312578 10^{-16} + 0. I, \\
5.55111512312578 10^{-16} + 0. I, 1.11022302462516 10^{-16} + 0. I, -4.99600361081320 10^{-16} + 0. I, \\
\end{bmatrix}
\end{align*}
\]
The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.

```maple
for j from 1 to 100 do
    B := R1 . Q1:
    Q1, R1 := QRDecomposition(B);
od:
```

The QR algorithm does not work for every matrix instantly (there are steps to reduce to a form on which the QR-algorithm works). So if you choose a different matrix $A$, the simple-minded version below will fail.