

## Linear Systems and Matrices

> **restart;**

We need the [LinearAlgebra](#) package for some extra commands.

> **with(LinearAlgebra);**

[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CARE, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct, LA\_Main, LUdecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRdecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]

We illustrate **Gaussian elimination with back substitution** with Problem 7c on Page 357. This is a system with a unique solution. We enter the 4 equations.

> **eq1:=x[1]+1/2\*x[2]+1/3\*x[3]+1/4\*x[4]=1/6;**

$$eq1 := x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 = \frac{1}{6}$$

> **eq2:=1/2\*x[1]+1/3\*x[2]+1/4\*x[3]+1/5\*x[4]=1/7;**

$$eq2 := \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 = \frac{1}{7}$$

> **eq3:=1/3\*x[1]+1/4\*x[2]+1/5\*x[3]+1/6\*x[4]=1/8;**

$$eq3 := \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 = \frac{1}{8}$$

> **eq4:=1/4\*x[1]+1/5\*x[2]+1/6\*x[3]+1/7\*x[4]=1/9;**

$$eq4 := \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 = \frac{1}{9}$$

But we also add a fifth equation for later use.

```
> eq5:=x[1]+2*x[2]+3*x[3]+4*x[4]=5;
```

$$eq5 := x_1 + 2x_2 + 3x_3 + 4x_4 = 5$$

After making lists of our equations and variables, we use [GenerateMatrix](#) to create our **augmented matrices** for the systems of equations.

```
> EqList:=[eq1,eq2,eq3,eq4];
```

$$EqList := \left[ x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 = \frac{1}{6}, \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 = \frac{1}{7}, \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 = \frac{1}{8}, \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 = \frac{1}{9} \right]$$

```
> Vars:=[x[1],x[2],x[3],x[4]];
```

$$Vars := [x_1, x_2, x_3, x_4]$$

```
> AM:=GenerateMatrix(EqList,Vars,augmented=true);
```

$$AM := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{7} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{9} \end{bmatrix}$$

```
> EqList2:=[eq1,eq2,eq3,eq4,eq5];
```

$$EqList2 := \left[ x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 = \frac{1}{6}, \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 = \frac{1}{7}, \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 = \frac{1}{8}, \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 = \frac{1}{9}, x_1 + 2x_2 + 3x_3 + 4x_4 = 5 \right]$$

```
> AM2:=GenerateMatrix(EqList2,Vars,augmented=true);
```

$$AM2 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{7} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{9} \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

We first solve the linear systems using Maple's [solve](#) command.

```
> soln:=solve({eq1,eq2,eq3,eq4},{x[1],x[2],x[3],x[4]});
```

$$\text{soln} := \left\{ x_1 = -\frac{2}{63}, x_2 = \frac{25}{42}, x_3 = -\frac{50}{21}, x_4 = \frac{25}{9} \right\}$$

```
> soln2:=solve({eq1,eq2,eq3,eq4,eq5},{x[1],x[2],x[3],x[4]});
soln2 :=
```

We see the second system has no solution. In what follows, for the first system, the A series matrices use exact values, while the B series matrices use decimal approximations with **Digits=8** (implemented by **evalf[8]**). Saving the matrix AM for later use, we use the [Matrix](#) command to enter the matrices for the system. Note that the numbers 4 and 5, giving the dimensions, are optional.

```
> A:=Matrix(4,5,[[1,1/2,1/3,1/4,1/6],[1/2,1/3,1/4,1/5,1/7],[1/3,1/4,1/5,1/6,1/8],[1/4,1/5,1/6,1/7,1/9]]);
```

$$A := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{7} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{9} \end{bmatrix}$$

```
> B:=evalf[8](Matrix([[1,1/2,1/3,1/4,1/6],[1/2,1/3,1/4,1/5,1/7],[1/3,1/4,1/5,1/6,1/8],[1/4,1/5,1/6,1/7,1/9]]));
```

$$B := \begin{bmatrix} 1. & 0.5000000 & 0.3333333 & 0.2500000 & 0.1666667 \\ 0.5000000 & 0.3333333 & 0.2500000 & 0.2000000 & 0.1428571 \\ 0.3333333 & 0.2500000 & 0.2000000 & 0.1666667 & 0.1250000 \\ 0.2500000 & 0.2000000 & 0.1666667 & 0.1428571 & 0.1111111 \end{bmatrix}$$

We do step-by-step Gaussian elimination on both A and B. [RowOperation](#) is the command we use. Here, the row named by the second argument is replaced by that row plus the fourth argument times the row named by the third argument.

```
> A1:=RowOperation(A,[2,1],-1/2);
```

$$A1 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} & \frac{5}{84} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{9} \end{bmatrix}$$

```
> B1:=evalf[8](RowOperation(B,[2,1],-1/2));
```

$$B1 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333333 & 0.08333334 & 0.07500000 & 0.059523805 \\ 0.33333333 & 0.25000000 & 0.20000000 & 0.16666667 & 0.12500000 \\ 0.25000000 & 0.20000000 & 0.16666667 & 0.14285714 & 0.11111111 \end{bmatrix}$$

> **A2:=RowOperation(A1,[3,1],-1/3);**

$$A2 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} & \frac{5}{84} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{9} \end{bmatrix}$$

> **B2:=evalf[8](RowOperation(B1,[3,1],-.333333333));**

$$B2 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333333 & 0.08333334 & 0.07500000 & 0.059523805 \\ 0. & 0.08333334 & 0.08888889 & 0.083333338 & 0.069444444 \\ 0.25000000 & 0.20000000 & 0.16666667 & 0.14285714 & 0.11111111 \end{bmatrix}$$

> **A3:=RowOperation(A2,[4,1],-1/4);**

$$A3 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} & \frac{5}{84} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & \frac{3}{40} & \frac{1}{12} & \frac{9}{112} & \frac{5}{72} \end{bmatrix}$$

> **B3:=evalf[8](RowOperation(B2,[4,1],-.25));**

$$B3 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333333 & 0.08333334 & 0.07500000 & 0.059523805 \\ 0. & 0.08333334 & 0.08888889 & 0.083333338 & 0.069444444 \\ 0. & 0.07500000 & 0.083333338 & 0.080357140 & 0.069444442 \end{bmatrix}$$

The following implementation of [RowOperation](#) swaps rows 2 and 3. I'm just doing this as an illustration here.

> **A4:=RowOperation(A3,[2,3]);**

$$A4 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} & \frac{5}{84} \\ 0 & \frac{3}{40} & \frac{1}{12} & \frac{9}{112} & \frac{5}{72} \end{bmatrix}$$

> **B4:=evalf[8](RowOperation(B3,[2,3]));**

$$B4 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333334 & 0.08888889 & 0.08333338 & 0.06944444 \\ 0. & 0.08333333 & 0.08333334 & 0.07500000 & 0.059523805 \\ 0. & 0.07500000 & 0.08333338 & 0.080357140 & 0.069444442 \end{bmatrix}$$

We continue with Gaussian elimination.

> **A5:=RowOperation(A4,[3,2],-1);**

$$A5 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & 0 & -\frac{1}{180} & -\frac{1}{120} & -\frac{5}{504} \\ 0 & \frac{3}{40} & \frac{1}{12} & \frac{9}{112} & \frac{5}{72} \end{bmatrix}$$

> **B5:=evalf[8](RowOperation(B4,[3,2],-.08333333/.08333334));**

$$B5 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333334 & 0.08888889 & 0.08333338 & 0.06944444 \\ 0. & 0. & -0.005555539 & -0.008333328 & -0.009920631 \\ 0. & 0.07500000 & 0.08333338 & 0.080357140 & 0.069444442 \end{bmatrix}$$

> **A6:=RowOperation(A5,[4,2],-9/10);**

$$A6 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & 0 & -\frac{1}{180} & -\frac{1}{120} & -\frac{5}{504} \\ 0 & 0 & \frac{1}{300} & \frac{3}{560} & \frac{1}{144} \end{bmatrix}$$

> **B6:=evalf[8](RowOperation(B5,[4,2],-.075/.08333334));**

$$B6 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333334 & 0.08888889 & 0.08333338 & 0.06944444 \\ 0. & 0. & -0.005555539 & -0.008333328 & -0.009920631 \\ 0. & 0. & 0.003333343 & 0.005357142 & 0.006944447 \end{bmatrix}$$

> **A7:=RowOperation(A6,[4,3],3/5);**

$$A7 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & 0 & -\frac{1}{180} & -\frac{1}{120} & -\frac{5}{504} \\ 0 & 0 & 0 & \frac{1}{2800} & \frac{1}{1008} \end{bmatrix}$$

> **B7:=evalf[8](RowOperation(B6,[4,3],.003333343/.005555539));**

$$B7 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333334 & 0.08888889 & 0.08333338 & 0.06944444 \\ 0. & 0. & -0.005555539 & -0.008333328 & -0.009920631 \\ 0. & 0. & 0. & 0.0003571158 & 0.0009920334 \end{bmatrix}$$

A third way in which [RowOperation](#) can be used is to multiply a row by a number. Here we multiply the fourth row.

> **A8:=RowOperation(A7,4,2800);**

$$A8 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{4}{45} & \frac{1}{12} & \frac{5}{72} \\ 0 & 0 & -\frac{1}{180} & -\frac{1}{120} & -\frac{5}{504} \\ 0 & 0 & 0 & 1 & \frac{25}{9} \end{bmatrix}$$

> **B8:=evalf[8](RowOperation(B7,4,1/.0003571158));**

$$B8 := \begin{bmatrix} 1. & 0.50000000 & 0.33333333 & 0.25000000 & 0.16666667 \\ 0. & 0.08333334 & 0.08888889 & 0.08333338 & 0.06944444 \\ 0. & 0. & -0.005555539 & -0.008333328 & -0.009920631 \\ 0. & 0. & 0. & 0.99999998 & 2.7779039 \end{bmatrix}$$

Now that the matrices are in row reduced form, we use **back substitution**, implemented by [BackwardSubstitute](#) to solve for the variables.

```
> x:=BackwardSubstitute(A8);
```

$$x := \begin{bmatrix} -\frac{2}{63} \\ \frac{25}{42} \\ -\frac{50}{21} \\ \frac{25}{9} \end{bmatrix}$$

We note that this solution is a vector.

```
> whattype(x);
```

*Vector*<sub>column</sub>

```
> x[2];
```

$$\frac{25}{42}$$

We continue.

```
> y:=BackwardSubstitute(B8);
```

$$y := \begin{bmatrix} -0.0317532838012194 \\ 0.595319102193576 \\ -2.38114678236673 \\ 2.77790395555808 \end{bmatrix}$$

We check the differences between the exact and rounded-off solutions.

```
> x:=evalf(x);
```

$$x := \begin{bmatrix} -0.03174603175 \\ 0.5952380952 \\ -2.380952381 \\ 2.777777778 \end{bmatrix}$$

```
> difference:=x-y;
```

$$difference := \begin{bmatrix} 0.00000725205121938910 \\ -0.0000810069935756053 \\ 0.000194401366732322 \\ -0.000126177558079199 \end{bmatrix}$$

The following single statement [GaussianElimination](#) does the entirety of Gaussian elimination in a single step.

```
> AA:=GaussianElimination(A);
```

$$AA := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} & \frac{5}{84} \\ 0 & 0 & \frac{1}{180} & \frac{1}{120} & \frac{5}{504} \\ 0 & 0 & 0 & \frac{1}{2800} & \frac{1}{1008} \end{bmatrix}$$

> **x:=BackwardSubstitute(AA);**

$$x := \begin{bmatrix} -\frac{2}{63} \\ \frac{25}{42} \\ -\frac{50}{21} \\ \frac{25}{9} \end{bmatrix}$$

> **BB:=GaussianElimination(B);**

BB := [[1., 0.5000000000000000, 0.3333333300000000, 0.2500000000000000, 0.1666666700000000],  
 [0., 0.0833333350000000, 0.0888888911111111, 0.0833333375000000, 0.0694444438888889],  
 [0., 0., -0.0055555507777777, -0.0083333324999998, -0.0099206347222223],  
 [0., 0., 0., 0.0003571282999925, 0.000992051571418480]]

> **y:=BackwardSubstitute(BB);**

$$y := \begin{bmatrix} -0.0317503151028448 \\ 0.595287687781521 \\ -2.38107381930422 \\ 2.77785762550632 \end{bmatrix}$$

The term **rref** stands for [ReducedRowEchelonForm](#). The solutions can actually be read directly from the matrix in this case.

> **AAA:=ReducedRowEchelonForm(A);**

$$AAA := \begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{2}{63} \\ 0 & 1 & 0 & 0 & \frac{25}{42} \\ 0 & 0 & 1 & 0 & -\frac{50}{21} \\ 0 & 0 & 0 & 1 & \frac{25}{9} \end{bmatrix}$$

```
> BBB:=ReducedRowEchelonForm(B);
```

$$BBB := \begin{bmatrix} 1. & 0. & 0. & 0. & -0.0317503151028448 \\ 0. & 1. & 0. & 0. & 0.595287687781521 \\ -0. & -0. & 1. & -0. & -2.38107381930422 \\ 0. & 0. & 0. & 1. & 2.77785762550632 \end{bmatrix}$$

Treat the numbers with high negative exponents as 0's. After using the [SubMatrix](#) command to create the coefficient and right-hand side matrices, we use the [LinearSolve](#) command to solve the system.

```
> coeffmat:=SubMatrix(A,1..4,1..4);
```

$$coeffmat := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

```
> rhsmat:=SubMatrix(A,1..4,5);
```

$$rhsmat := \begin{bmatrix} \frac{1}{6} \\ \frac{1}{7} \\ \frac{1}{8} \\ \frac{1}{9} \end{bmatrix}$$

```
> x:=LinearSolve(coeffmat,rhsmat);
```

$$x := \begin{bmatrix} -\frac{2}{63} \\ \frac{25}{42} \\ -\frac{50}{21} \\ \frac{25}{9} \end{bmatrix}$$

Here x is a matrix.

```
> whattype(x);
```

*Matrix*

```
> x[3,1];
```

$$-\frac{50}{21}$$

Now we use [LinearSolve](#) with our augmented matrices AM and AM2 from the beginning of the worksheet.

> **x:=LinearSolve(AM);**

$$x := \begin{bmatrix} -\frac{2}{63} \\ \frac{25}{42} \\ -\frac{50}{21} \\ \frac{25}{9} \end{bmatrix}$$

Here x is again a vector.

> **whattype(x);**

*Vector*<sub>column</sub>

> **x[3];**

$$-\frac{50}{21}$$

> **x:=LinearSolve(AM2);**

**Error, (in LinearAlgebra:-LinearSolve) inconsistent system**

This means there is no solution. Now we consider a third, new system, and enter it by giving its augmented matrix.

> **AM3:=Matrix(4,7,[[10,-27,1,1,2,-11,1],[20,-62,29,20,11,-16,1],[ -1,-8,36,24,9,9,1],[-8,27,-19,-13,-6,5,-5]]);**

$$AM3 := \begin{bmatrix} 10 & -27 & 1 & 1 & 2 & -11 & 1 \\ 20 & -62 & 29 & 20 & 11 & -16 & 1 \\ -1 & -8 & 36 & 24 & 9 & 9 & 1 \\ -8 & 27 & -19 & -13 & -6 & 5 & -5 \end{bmatrix}$$

We can use the command [GenerateEquations](#) to generate the equations from the matrix.

> **Eq:=GenerateEquations(AM3,[z[1],z[2],z[3],z[4],z[5],z[6]]);**

$$Eq := [10z_1 - 27z_2 + z_3 + z_4 + 2z_5 - 11z_6 = 1, 20z_1 - 62z_2 + 29z_3 + 20z_4 + 11z_5 - 16z_6 = 1, -z_1 - 8z_2 + 36z_3 + 24z_4 + 9z_5 + 9z_6 = 1, -8z_1 + 27z_2 - 19z_3 - 13z_4 - 6z_5 + 5z_6 = -5]$$

We solve the system as above.

> **z:=LinearSolve(AM3);**

$$z := \begin{bmatrix} 215 + 19 \_t6_2 - 54 \_t6_3 - 39 \_t6_4 \\ \_t6_2 \\ \_t6_3 \\ \_t6_4 \\ -145 - 10 \_t6_2 + 33 \_t6_3 + 24 \_t6_4 \\ 169 + 13 \_t6_2 - 43 \_t6_3 - 31 \_t6_4 \end{bmatrix}$$

We see there are infinitely many solutions with **free parameters**  $\_t6_2$ ,  $\_t6_3$ , and  $\_t6_4$ . We can assign a name to the free parameters by using the **free** option.

> **z:=LinearSolve(AM3,free='s');**

$$z := \begin{bmatrix} 215 + 19 s_2 - 54 s_3 - 39 s_4 \\ s_2 \\ s_3 \\ s_4 \\ -145 - 10 s_2 + 33 s_3 + 24 s_4 \\ 169 + 13 s_2 - 43 s_3 - 31 s_4 \end{bmatrix}$$