```
> restart
>
> with(LinearAlgebra)
```

```
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CharacteristicMatrix,        (1
    CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, ConditionNumber, ConstantMatrix,
    ConstantVector, Copy, CreatePermutation, CrossProduct, 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]
```

```
>
>
> logPCM := Matrix([[0 , a12 , a13, a14 ], [-a12, 0 , a23 , a24], [ -a13 , -a23, 0 , a34] , [-a14 , -a24, -a34 , 0 ] ]);
```

$$logPCM := \begin{bmatrix} 0 & a12 & a13 & a14 \\ -a12 & 0 & a23 & a24 \\ -a13 & -a23 & 0 & a34 \\ -a14 & -a24 & -a34 & 0 \end{bmatrix} \qquad (2$$

```
>
```

$$> \quad T := Matrix\left(\left[\begin{array}{cccccc} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right]\right) :$$

```
WeightVectors := Matrix(16, 4, [[0 , 0 , 0 , 0  ]]) :
 for tindex from 1 to 16 do
AdjMatT := Matrix( [[0 , 0 , 0 , 0  ], [0 , 0 , 0 , 0  ], [ 0 , 0 , 0 , 0 ], [0 , 0 , 0 , 0 ]]) :
DEGR := Matrix( [[0 , 0 , 0 , 0  ], [0 , 0 , 0 , 0  ], [ 0 , 0 , 0 , 0 ], [0 , 0 , 0 , 0 ]]) :
 t := T[tindex] :
 if t[1] = 1 then  AdjMatT[1, 2] := 1 : AdjMatT[2, 1] := 1 :   end if :
 if t[2] = 1 then  AdjMatT[1, 3] := 1 :   AdjMatT[3, 1] := 1 :   end if :
 if t[3] = 1 then  AdjMatT[1, 4] := 1 :   AdjMatT[4, 1] := 1 :   end if :
 if t[4] = 1 then  AdjMatT[2, 3] := 1 :   AdjMatT[3, 2] := 1 :   end if :
 if t[5] = 1 then  AdjMatT[2, 4] := 1 :   AdjMatT[4, 2] := 1 :   end if :
 if t[6] = 1 then  AdjMatT[3, 4] := 1 :   AdjMatT[4, 3] := 1 :   end if :

for indexi from 1 to 4 do
  s := 0 :
```

```
  for indexj from 1 to 4 do
  s := s + AdjMatT[indexi, indexj] :
  end do:
  DEGR[indexi, indexi] := s :
end do:
 L := DEGR − AdjMatT :
Lti := DeleteRow(DeleteColumn(L, 1), 1) :

logIPCM := logPCM·~AdjMatT :
 logRHS := Vector([0, 0, 0]) :
 for indexi from 2 to 4 do
  s := 0 :
  for indexj from 1 to 4 do
  s := s + logIPCM[indexi, indexj] :
  end do:
  logRHS[indexi − 1] := s :
end do:
 logwT := MatrixVectorMultiply(MatrixInverse(Lti), logRHS) :
 WeightVectors[tindex, 1 ..] := Transpose(Vector([0, logwT])) :
 end do:
 eval(WeightVectors);
interface(rtablesize = 100) :
WeightVectors
```

```
[      16 x 4 Matrix       ]
[  Data Type: anything     ]
[  Storage: rectangular    ]
[  Order: Fortran_order    ]
```

$$\begin{bmatrix}
0 & a24 - a14 & a34 - a14 & -a14 \\
0 & a23 + a34 - a14 & a34 - a14 & -a14 \\
0 & a24 - a14 & -a23 + a24 - a14 & -a14 \\
0 & a24 - a13 - a34 & -a13 & -a13 - a34 \\
0 & a23 - a13 & -a13 & -a13 - a34 \\
0 & a23 - a13 & -a13 & a23 - a24 - a13 \\
0 & a24 - a14 & -a13 & -a14 \\
0 & a23 - a13 & -a13 & -a14 \\
0 & -a12 & -a12 - a24 + a34 & -a12 - a24 \\
0 & -a12 & -a12 - a23 & -a12 - a23 - a34 \\
0 & -a12 & -a12 - a23 & -a12 - a24 \\
0 & -a12 & a34 - a14 & -a14 \\
0 & -a12 & -a12 - a23 & -a14 \\
0 & -a12 & -a13 & -a13 - a34 \\
0 & -a12 & -a13 & -a12 - a24 \\
0 & -a12 & -a13 & -a14
\end{bmatrix}$$

(3

```
>
>
```