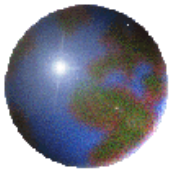


The LINGO Algebraic Modeling System

What is LINGO?

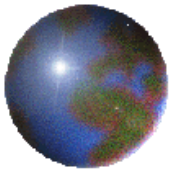
Language & system for developing optimization based planning models.

Available from: www.lindo.com

Used since the 1990's in all major industries:

- Petroleum : blending, exploration,
- Finance: portfolio design,
- Marketing: Ad campaign design,
- Supply Chain Management,
- . . .

Keywords: Getting started, Introduction, Matrix,Tutorial.



What are the Unique Features of LINGO ?

Ease of Use: Download, install, and
solve your first model in 5 minutes from www.lindo.com.

Tech support is fast, with no finger pointing because
it is an integrated system from one supplier.

Library of "template" models, 500+, well documented, at *www.lindo.com*
to get you started.

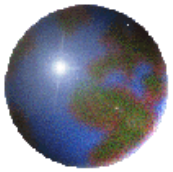
LINGO text downloadable from *www.lindo.com* to get you started modeling.

Automatic problem type recognition. Do not have to figure out
which solver will solve my problem.

Automatic Linearization of many nonlinear functions,
e.g., max(), min(), abs( ), IF( ) => solve faster

General: Same GUI on Windows, Linux, Apple Mac.

Robust, well tested, fast solvers, ALL problem types,
linear, integer, quadratic, conic, semi-definite, nonlinear, global, stochastic, etc.



What are the Unique Features of LINGO ?

It is Feature-rich:

Matrix commands:

Eigenvalue, Q-R factorization, Inverse, Transpose, Matrix multiply, Sort.

Graphics/Charts for displaying results:

Histograms, Networks, Space-Time, Gantt, Tornado, (20+)

Probability distributions: Normal, etc. (20+), cdf, pdf, and inverse cdf for each.

Built-in programming language e.g. for managing systems with multiple model solves.

Data management functions built-in, multi-level sorts, time/calendar arithmetic.

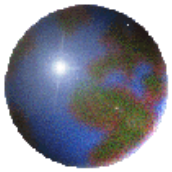
Interface with Excel, SQL, text file: simple “one line” fashion, e.g., demand = @OLE( );

Common constraint types built-in:

@AllDiff (All Different values), @POSD (Positive Definite), SOS2, etc

DEBUG command applies to ALL model types. Find why my model is infeasible.

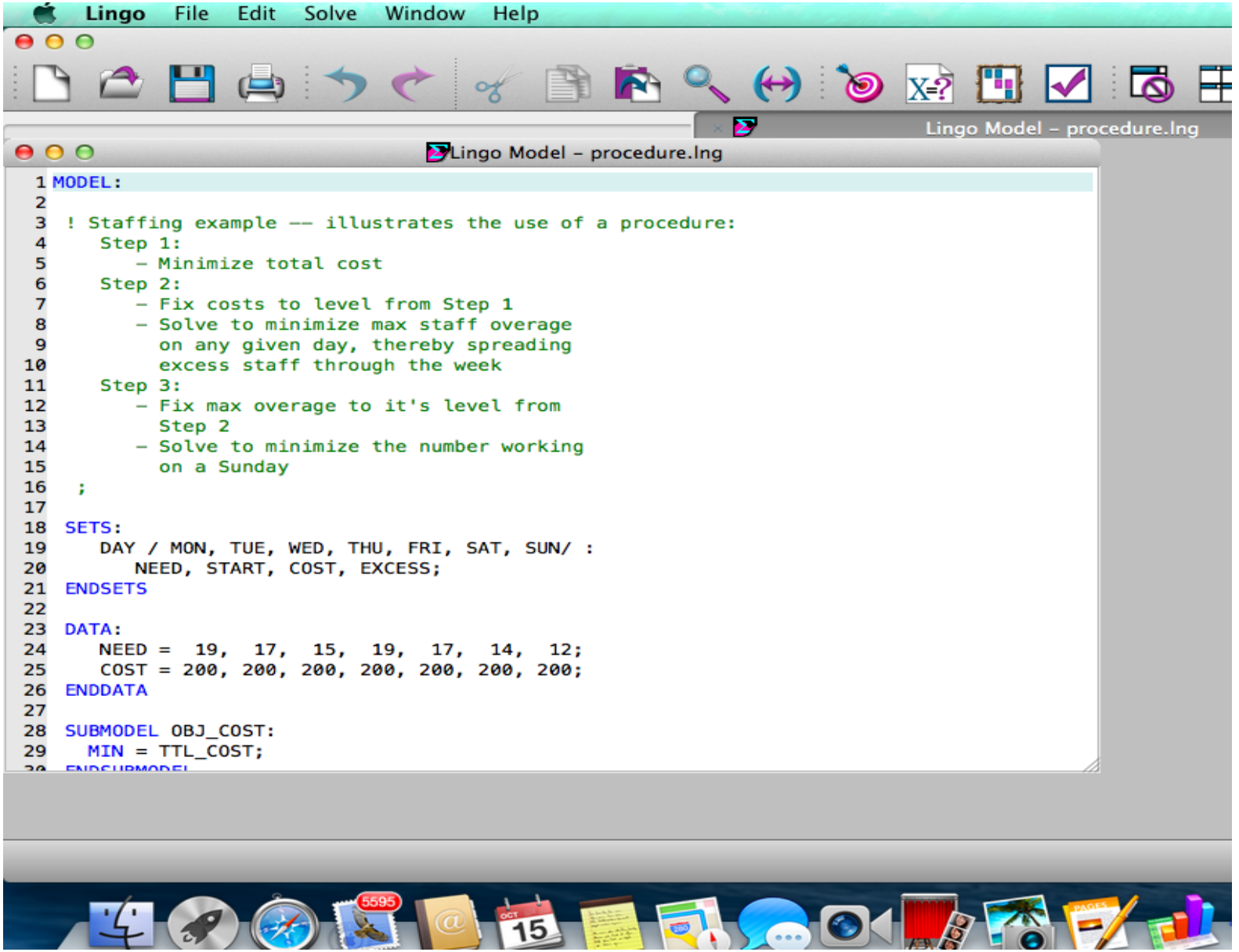
Programming interfaces to R, Python, C/C++, Java/JNI, Fortran, etc.



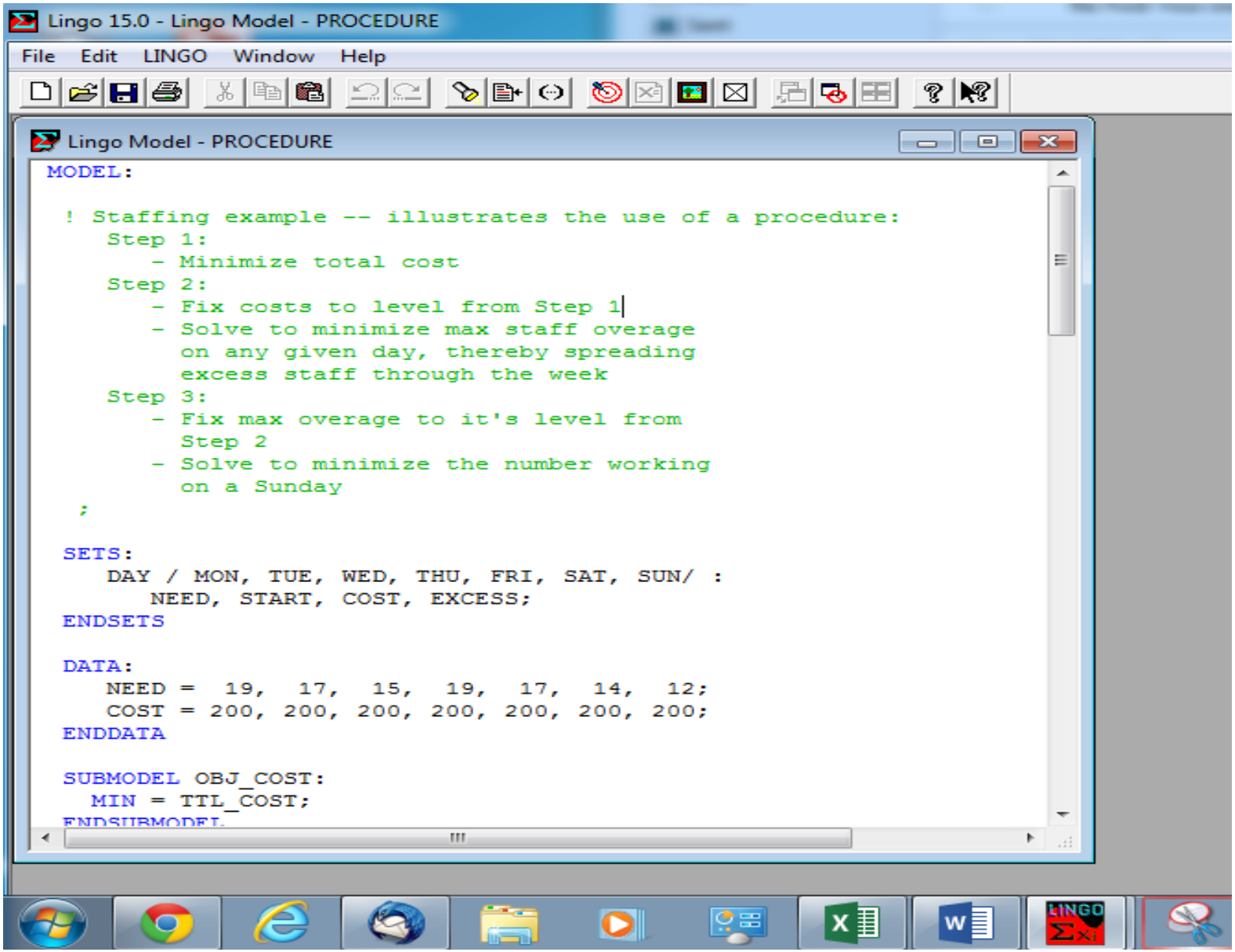
Let's Look at Specifics…

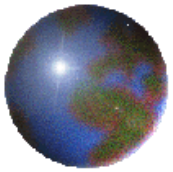
In the following slides we look at the details of the features mentioned in the previous slide.

LINGO Appears the Same on Apple Mac, Windows, Linux



LINGO on MS Windows





Which Solver/Problem Types Supported?

Comprehensive and large scale solver engine. Solve

Linear programs with millions of variables and constraints, using primal, dual, and/or barrier algorithms.

Quadratic objective and quadratic constraints, all forms, not restricted to only convex/positive definite. Does not quit with obscure message "stopping: matrix not positive definite."

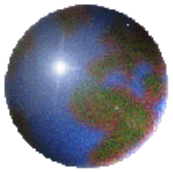
Semi-Definite Programs/ Conic programs, e.g., can solve "Value at Risk" portfolio models, or find closest valid correlation matrix.

Nonlinear models, including Multi-Start for highly nonlinear models.

Global solver, solve to guaranteed global optimality even if functions not convex, not continuous.

SP (Stochastic Programming) optimization under uncertainty,

Integer variables allowed anywhere.



How Do I Learn to Use LINGO?

Documentation is Built-in / On-line.

To see what functions are available, and to paste them into your model, click on:

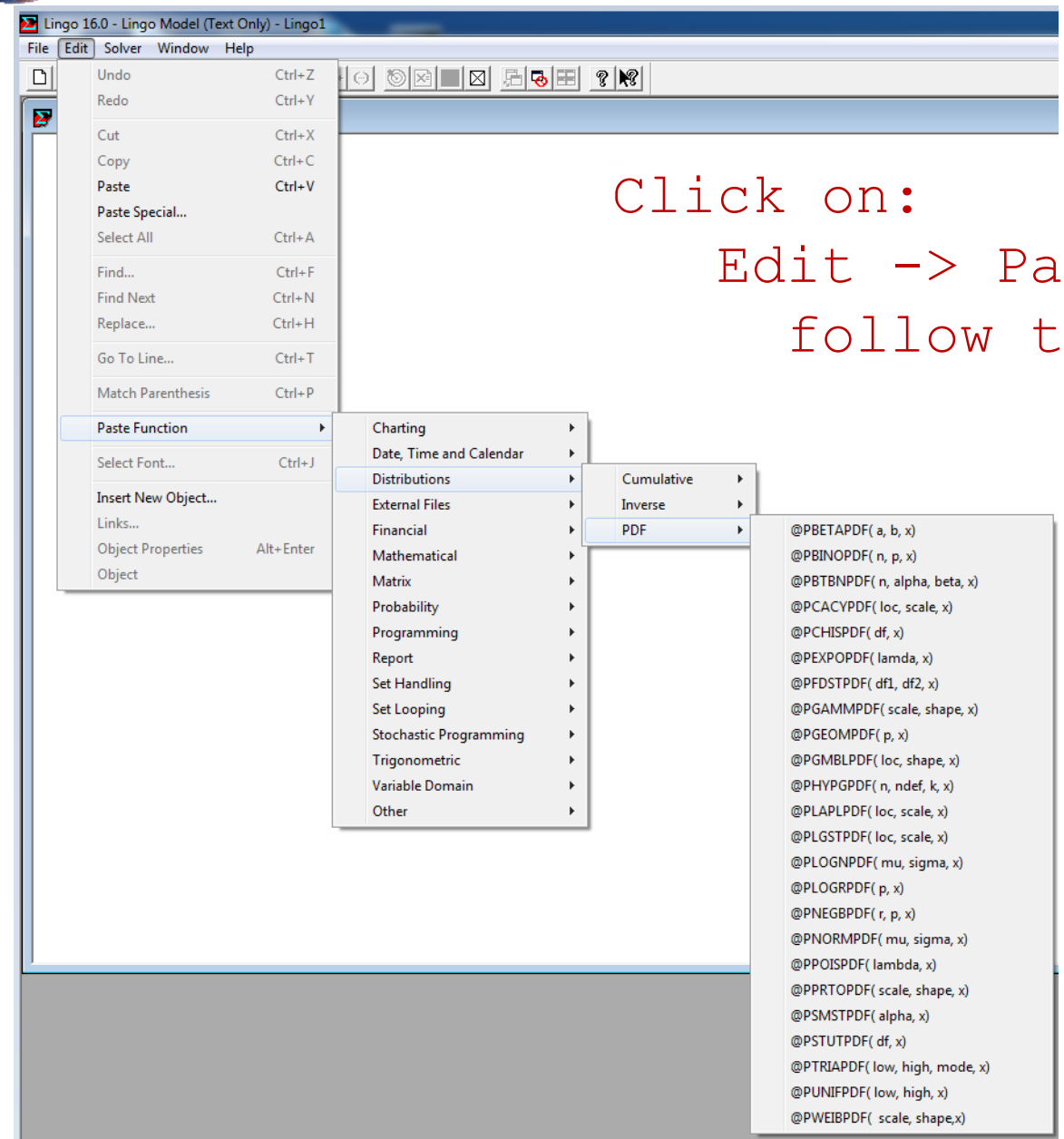
`Edit -> Paste Function`

To set various usage parameters, click on

`Solve -> Options`

Editor has syntax coloring, points to line with error.

How Do I Find the Available Functions in LINGO?



Click on:

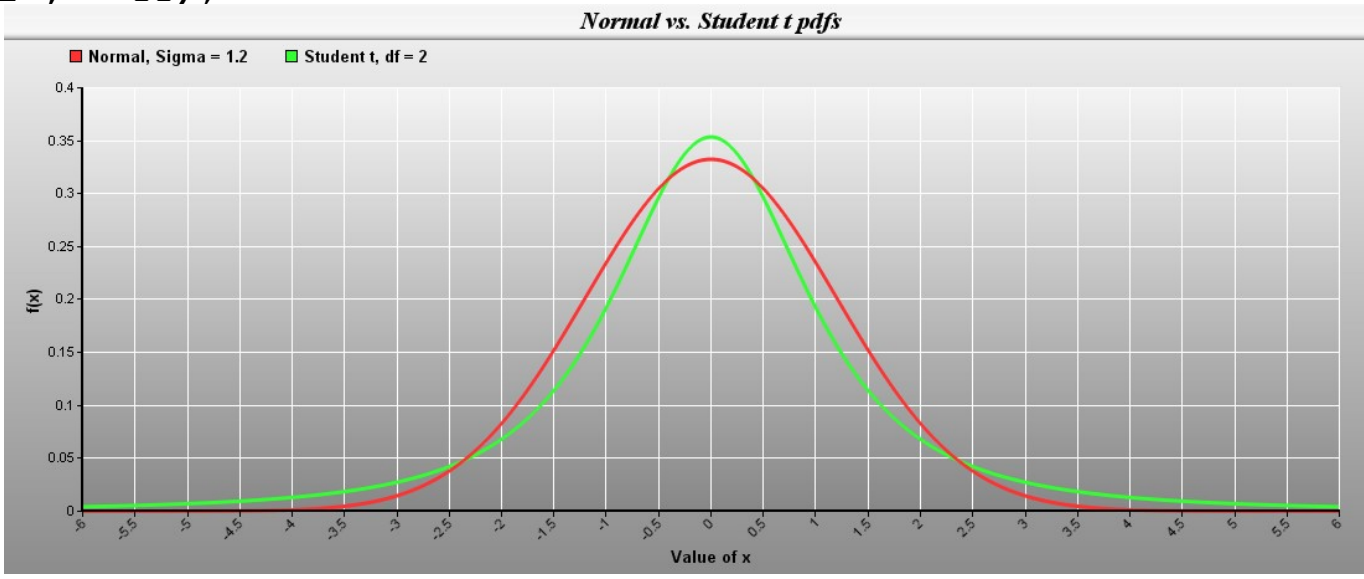
Edit -> Paste Function…

follow the links.

LINDO SYSTEMS INC.

Graphs and Procedures in LINGO

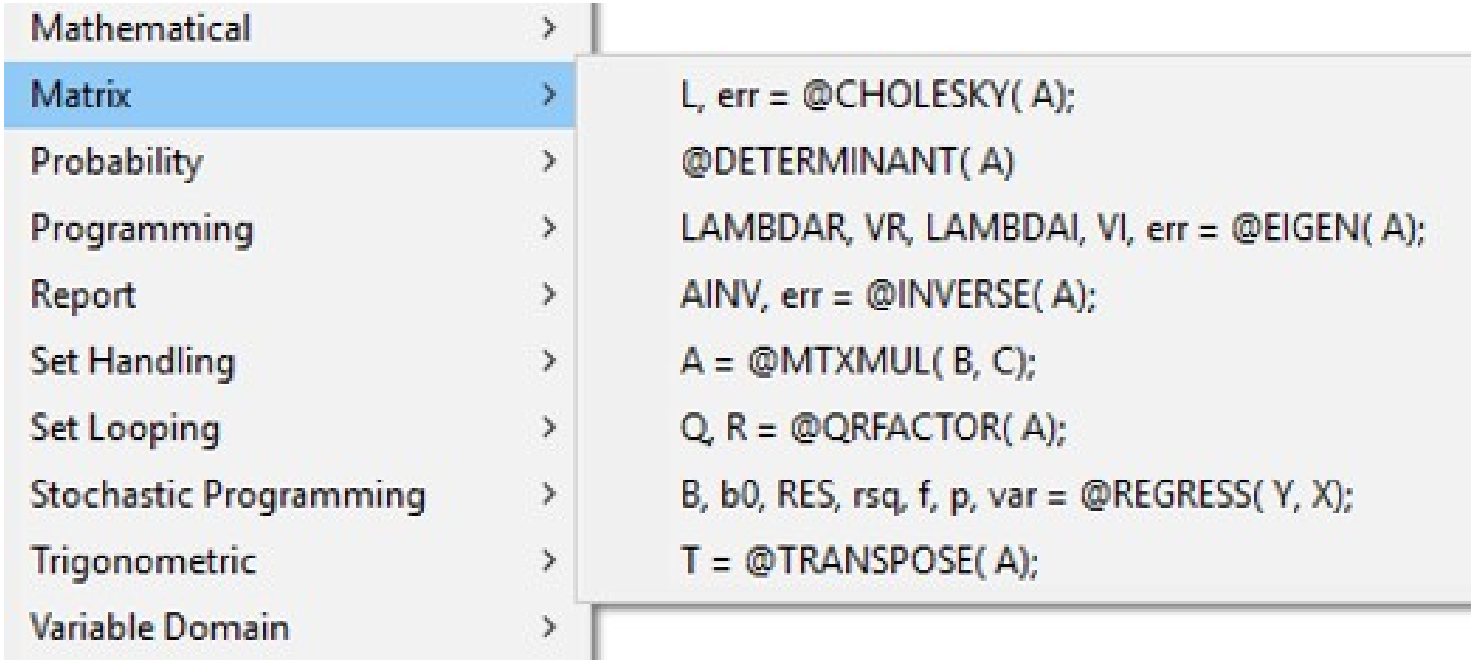
```
! Plotting using Procedures,
  Normal vs. Student t Distributions;
PROCEDURE CALCPDF:
  YN = @PNORMPDF( MU, SIGMA, x1);
  YT = @PSTUTPDF( 2, x1);
ENDPROCEDURE

CALC:
  @SET( 'TERSEO',2); ! Output level (0:verb, 1:terse, 2:only errors, 3:none);
  MU = 0;
  SIGMA = 1.2;
  @CHARTPCURVE('Normal vs. Student t pdfs','Value of x','f(x)',
     CALCPDF, X1, -5*SIGMA, 5*SIGMA,
    'Normal, Sigma = 1.2', YN,
    'Student t, df = 2',   YT);
ENDCALC
```



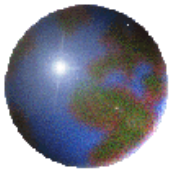
Matrix Commands in LINGO

Standard matrix operations are available: Matrix Multiply, Inverse, Transpose, Eigenvalues, Cholesky factorization, QR factorization, Determinant, Regression;

Click on Edit | Paste Function | . . .

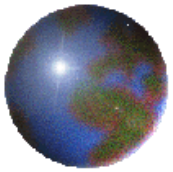


Matrix related: Multi-level sort,
Positive-Definite constraint,
AllDiff constraint.



Sorting of Data, One or More Submodels

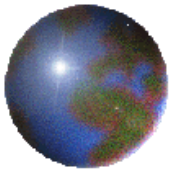
```
! A Product Mix Model with a
  Sorted Output Report;
SETS:
 PROD: PC, USE1, USE2, RCOST, UB, X, SORTORD;
ENDSETS
DATA:
 PROD= P1 P2 P3 P4 P5 P6 P7;
 PC =  26 30 47 14 46 28 42; ! Profit Contribution;
 UB =  60 50 20 40 23 45 27; ! Upper Bound on Production;
USE1=   1  0  1  1  2  1  3; ! Usage rate in Dept 1;
USE2=   1  2  2  0  2  1  1; ! Usage rate in Dept 2;
CAP1 = 160; ! Capacity in Dept 1;
CAP2 = 170; ! Capacity in Dept 2;
ENDDATA

SUBMODEL PRODMIX:
MAX= @SUM( PROD(j): PC(j)*X(j));
     @SUM( PROD(j): USE1(j)*X(j)) <= CAP1;
     @SUM( PROD(j): USE2(j)*X(j)) <= CAP2;
@FOR( PROD( j):
   X( j) <= UB(j);
    );
ENDSUBMODEL
```



Sorting of Data (Mult-level, +/-, Order Preserving)

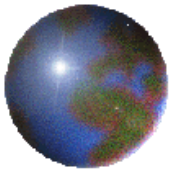
```
CALC:
 @SOLVE( PRODMIX);
!Get Reduced Cost;
 @FOR( PROD(j):
   RCOST(j) = @DUAL( X(j));
     );

! Sort decreasing in X, increasing in RCOST,
    decreasing in profit contribution;
   SORTORD = @SORT( -X, RCOST, -PC);

  @WRITE(' Product  Volume ReducedCost  PC', @NEWLINE(1));
  @FOR( PROD( j):
     js = SORTORD(j);
     @WRITE('   ', PROD(js),' ', @FORMAT(X(js),'10.2f'),' ',
       @FORMAT( RCOST(js),'8.2f'),'     ', PC(js), @NEWLINE(1));
      );
ENDCALC
```

Product	Volume	ReducedCost	PC
P1	55.00	0.00	26
P6	45.00	0.00	28
P4	40.00	0.00	14
P3	20.00	0.00	47
P2	15.00	0.00	30
P5	0.00	6.00	46
P7	0.00	6.00	42



Transpose of a Matrix

```
! Illustrating Transpose of a matrix.
 Example: We wrote our model assuming
  distance matrix is in From->To form,
  but now we have a user who has his
  matrix in To->From form;
SETS:
  PROD;
  PXP( PROD, PROD): DIST, TOFDIST;
ENDSETS
DATA:
  PROD =
     VANILLA BUTTERP,  STRAWB, CHOCO;
 ! A changeover time matrix.
From runs vertically, To runs horizontally;
    DIST=
      0       1         1       1  ! Vanilla;
      2       0         1       1  ! ButterP;
      3       2         0       1  ! StrawB;
      5       4         2       0; ! Choco;
ENDDATA

CALC:
  @SET( 'TERSEO',2);   !Output level (0:verb, 1:terse, 2:only errors, 3:none);
  @TABLE( DIST);
  @WRITE( @NEWLINE(1),' The transpose( To->From form)',@NEWLINE(1));
  TOFDIST = @TRANSPOSE( DIST);
  @TABLE( TOFDIST);
ENDCALC
```



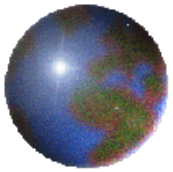
Transpose of a Matrix

Changing from a row to a column

	VANILLA	BUTTERP	STRAWB	CHOCO
VANILLA	0.000000	1.000000	1.000000	1.000000
BUTTERP	2.000000	0.000000	1.000000	1.000000
STRAWB	3.000000	2.000000	0.000000	1.000000
CHOCO	5.000000	4.000000	2.000000	0.000000

The transpose(To->From form)

	VANILLA	BUTTERP	STRAWB	CHOCO
VANILLA	0.000000	2.000000	3.000000	5.000000
BUTTERP	1.000000	0.000000	2.000000	4.000000
STRAWB	1.000000	1.000000	0.000000	2.000000
CHOCO	1.000000	1.000000	1.000000	0.000000



Inverse of a Matrix

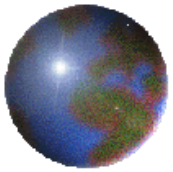
```
! Illustration of the @INVERSE() function.
If we need the solution to the matrix equation A*X = RHS
for several different RHS, then it is
efficient to first compute AINV and then
use the result that X = AINV*RHS;
SETS:
 DIM;
 DXD( DIM, DIM): A, AINV, RESULT1;
 CASE;
 DXC( DIM, CASE): RHS, RESULT2;
ENDSETS
DATA:
 DIM = 1..4;

!Example 1;
 A=
  5  7  3 -1
  1 -2  3  4
  1  2  4  5
  9  3 -4  7;

!Example 2;
! A permutation matrix.
 A(i,j) = 1 means move element in position i to position j;
! A=
  0  1  0  0
  0  0  1  0
  1  0  0  0
  0  0  0  1;

  CASE = 1..3; ! Different RHS cases;
 RHS =  14  24   7
         6  22  29
        12  37  36
        15  31  56;
ENDDATA
```

LINDO SYSTEMS INC.

Inverse of a Matrix

```
CALC:
 @SET( 'TERSEO',2);    ! Output level (0:verb, 1:terse, 2:only errors, 3:none);
 AINV, err = @INVERSE( A);
@WRITE(' The error code is: ', err, @NEWLINE(1));
@WRITE(' The inverse is: ', @NEWLINE(1));
@TABLE( AINV);

@WRITE( @NEWLINE(1),' AINV * A =',@NEWLINE(1));
MMULT1;  !Matrix multiply;
@TABLE( RESULT1);
@WRITE( @NEWLINE(1));

@WRITE( @NEWLINE(1),' AINV * RHS =',@NEWLINE(1));
MMULT2; !Matrix multiply;
@TABLE( RESULT2);
@WRITE( @NEWLINE(1));
ENDCALC
```

```
The error code is: 0
 The inverse is:
                     1                2                3                4
   1       0.1205575       0.2501742      -0.2355401   0.4250871E-01
   2   0.1393728E-02      -0.2745645       0.2111498   0.6271777E-02
   3   0.9547038E-01       0.1923345  -0.3623693E-01  -0.7038328E-01
   4      -0.1010453  -0.9407666E-01       0.1916376   0.4529617E-01

 AINV * A =
              1         2         3         4
   1  1.000000  0.000000  0.000000  0.000000
   2  0.000000  1.000000  0.000000  0.000000
   3  0.000000  0.000000  1.000000  0.000000
   4  0.000000  0.000000  0.000000  1.000000

 AINV * RHS =
              1         2         3
   1  1.000000  1.000000  2.000000
   2  1.000000  2.000000  0.000000
   3  1.000000  3.000000  1.000000
   4  1.000000  4.000000  6.000000
```

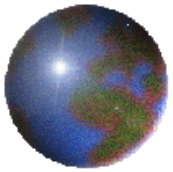


Cholesky Factorization: Generate Correlated R.V.s

```
! Using Cholesky Factorization to Generate    (CholeskyNormal.lng)
 Multi-variate Normal Random Variables with a
 specified covariance matrix.
   Using matrix notation, if
  XSN = a row vector of independent random variables
        with mean 0 and variance 1,
  and E[ ] is the expectation operator, and
  XSN' is the transpose of XSN, then its covariance matrix,
  E[ XSN'*XSN] = I, i.e., the identity matrix with
  1's on the diagonal and 0's everywhere else.
 Further, if we apply a linear transformation L,
   E[(L*XSN)'*(L*XSN)]
   = L'*L*E[XSN'*XSN] = L'*L.
 Thus, if L'*L = Q, then the
 L*XSN will have covariance matrix Q.
```

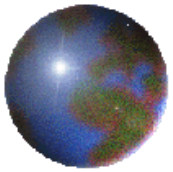
Cholesky factorization is a way of finding a matrix L, so that **L'*L = Q**. So we can think of L as the matrix square root of Q.

If XSN is a vector of standard Normal random variables with mean 0 and variance 1, then L*XSN has a multivariate Normal distribution with covariance matrix L'*L;



Cholesky Factorization: Generate Correlated R.V.s

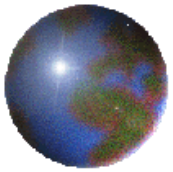
```
DATA:
 ! Number of scenarios or observations to generate;
  SCENE = 1..8000;
  MU = 50 60 80;  ! The means;
 ! The covariance matrix;
  Q = 16  4  -1
       4 15   3
      -1  3  17;
  SEED = 26185; ! A random number seed;
! Generate some quasi-random uniform variables in interval (0, 1);
  XU = @QRAND( SEED);
ENDDATA
CALC:
 ! Compute the "square root" of the covariance matrix;
  L, err  = @CHOLESKY( Q);

  @SET( 'TERSEO',3);    ! Output level (0:verb,1:terse,2: only errors,3:none);
  @WRITE(' Error=', err,' (0 means valid matrix.', @NEWLINE(1));
  @WRITE(' The L matrix is:', @NEWLINE(1));

  @TABLE( L); ! Display it;

  ! Generate the Normal random variables;
  @FOR( SCENE( s):
  !  Generate a set of independent Standard ( mean= 0, SD= 1)
     Normal random variables;
    @FOR( DIM( j):
       XSN(j) =  @PNORMINV( 0, 1, XU(s,j));
      );

  ! Now give them the appropriate means and covariances.
   The matrix equivalent of XN = MU + L*XSN;
    @FOR( DIM( j):
      XN(s,j) = MU(j) + @SUM( DIM(k): L(j,k)*XSN(k));
      );
    );
```



Cholesky Factorization: Generate Correlated R.V.s

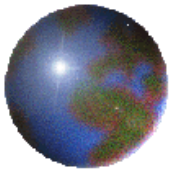
```
 The Q matrix is:
              1           2           3
  1    16.00000    4.000000   -1.000000
  2    4.000000    15.00000    3.000000
  3   -1.000000    3.000000    17.00000

Error=0 (0 means valid matrix found).
The L matrix is:
              1           2           3
  1    4.000000    0.000000    0.000000
  2    1.000000    3.741657    0.000000
  3   -0.2500000   0.8685990    4.022814

The empirical means:
      50.00002  59.99982  79.99992
Empirical covariance matrix:
              1           2           3
  1    15.99428    4.006238  -0.9971722
  2    4.006238    15.00404    2.972699
  3   -0.9971722   2.972699    16.98160
```

LINDO SYSTEMS INC.



Eigenvalues in LINGO

```
!   Compute the eigenvalues/vectors of a covariance matrix.
  Alternatively, do Principal Components Analysis.
  If there is a single large eigenvalue for the covariance
  matrix, then this suggests that there is a single factor,
  e.g., "the market" that explains all the variability;
! Some things to note,
   1)In general, given a square matrix A, then we try to find
  an eigenvalue, lambda, and its associated eigenvector X,
  to satisfy the matrix equation:
    A*X = lambda*X.
   2) the sum of the eigenvalues = sum of the terms on the diagonal
      of the original matrix, the variances if it is a covariance
      matrix.
   3) the product of the eigenvalues = determinant of the original matrix.
   4) A positive definite matrix has all positive eigenvalues;
! Keywords: Eigenvalue, PCA, Principal Component Analysis,
  Singular value decomposition, Covariance;
SETS:
   ASSET: lambda;
   CMAT( ASSET, ASSET) : X, COVAR;
ENDSETS
```



Eigenvalues in LINGO, II

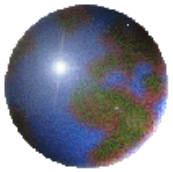
```
DATA:
! The investments available(Vanguard funds);
  ASSET =   VG040         VG102          VG058          VG079          VG072          VG533;
    ! Covariance matrix, based on June 2004 to Dec 2005;
 COVAR=
    !       VG040          VG102          VG058          VG079          VG072          VG533;
 !VG040;  0.6576337E-02  0.7255873E-02 -0.7277427E-03  0.6160186E-02  0.5015276E-02  0.1082129E-01
 !VG102;  0.7255873E-02  0.8300280E-02 -0.1067516E-02  0.7007361E-02  0.5651304E-02  0.1208505E-01
 !VG058; -0.7277427E-03 -0.1067516E-02  0.1366187E-02 -0.1281051E-02 -0.1113421E-02 -0.1179400E-02
 !VG079;  0.6160186E-02  0.7007361E-02 -0.1281051E-02  0.1020367E-01  0.8663464E-02  0.1477201E-01
 !VG072;  0.5015276E-02  0.5651304E-02 -0.1113421E-02  0.8663464E-02  0.1568187E-01  0.1510857E-01
 !VG533;  0.1082129E-01  0.1208505E-01 -0.1179400E-02  0.1477201E-01  0.1510857E-01  0.3002400E-01;

 ENDDATA
 CALC:
  @SET( 'TERSEO', 2);   ! Turn off default output;
  @SET( 'LINLEN', 100); ! Terminal page width (0:none);
! Compute the eigenvalues and eigenvectors;
   LAMBDA, X = @EIGEN( COVAR);
! Get the sum of the variances;
  TOTVAR = @SUM( ASSET(j): COVAR(j,j));
  CUMEVAL= 0;
  @WRITE(' Eigen-  Frac-        Associated Eigenvector', @NEWLINE(1),
         ' value   tion  ');
  @FOR( ASSET(j):
     @WRITE( @FORMAT( ASSET(j), '%7s'));
      );
  @WRITE( @NEWLINE(1));
  @FOR( ASSET(j):
    CUMEVAL = CUMEVAL + LAMBDA(j);
    @WRITE( @FORMAT(lambda(j),'6.3f'), @FORMAT( CUMEVAL/TOTVAR,'7.3f'),': ');
    @FOR( ASSET(i):
      @WRITE(' ',@FORMAT(X(i,j),'6.3f'));    );
    @WRITE( @NEWLINE(1));
       );
 @WRITE(' Sum of variances  = ', @SUM(ASSET(j): COVAR(j,j)), @NEWLINE(1));
 @WRITE(' Sum of eigenvalues= ', @SUM(ASSET(j): LAMBDA(j)), @NEWLINE(1));
 ENDCALC
```

Eigenvalues in LINGO, III

Eigen-value	Frac-tion	Associated Eigenvector VG040	VG102	VG058	VG079	VG072	VG533
0.057	0.791:	-0.285	-0.321	0.042	-0.385	-0.416	-0.702
0.008	0.900:	-0.374	-0.419	0.000	-0.042	0.818	-0.118
0.004	0.954:	0.393	0.477	-0.219	0.132	0.336	-0.663
0.000	0.956:	-0.733	0.665	0.145	0.001	0.006	-0.001
0.002	0.986:	-0.263	-0.221	-0.310	0.848	-0.211	-0.150
0.001	1.000:	-0.135	0.051	-0.913	-0.338	-0.027	0.178

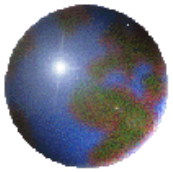
Sum of variances = 0.072152344

Sum of eigenvalues= 0.072152344



Regression, e.g., Forecasts for Production Planning

```
! Names of the variables;
  VARS =
    EMPLD PRICE GNP__  JOBLS MILIT   POPLN  YEAR_;
! The dataset;
  DATATAB =
    60323   83  234289  2356  1590  107608  1947
    61122  88.5 259426  2325  1456  108632  1948
    60171  88.2 258054  3682  1616  109773  1949
    61187  89.5 284599  3351  1650  110929  1950
    63221  96.2 328975  2099  3099  112075  1951
    63639  98.1 346999  1932  3594  113270  1952
    64989  99   365385  1870  3547  115094  1953
    63761 100   363112  3578  3350  116219  1954
    66019 101.2 397469  2904  3048  117388  1955
    67857 104.6 419180  2822  2857  118734  1956
    68169 108.4 442769  2936  2798  120445  1957
    66513 110.8 444546  4681  2637  121950  1958
    68655 112.6 482704  3813  2552  123366  1959
    69564 114.2 502601  3931  2514  125368  1960
    69331 115.7 518173  4806  2572  127852  1961
    70551 116.9 554894  4007  2827  130081  1962;

! Dependent variable. There can be only 1;
  DEPVAR = EMPLD;

! Explanatory variables. Should not include DEPVAR;
  EXPVAR = PRICE GNP__ JOBLS MILIT POPLN YEAR_;
ENDDATA
```

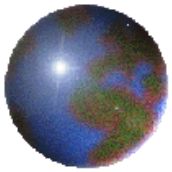


Regression, e.g., Forecasts for Production Planning

```
CALC:
  @SET( 'TERSEO',2); ! Output level (0:verb, 1:terse, 2:only errors, 3:none);
  !Set up data for the @REGRESS function;
  @FOR( OBS( I):
     Y( I) = DATATAB( I, @INDEX( EMPLD));
    @FOR( OXE( I, J): X( I, J) = DATATAB( I, J))
    );

   ! Do the regression;
   B, B0, RES, rsq, f, p, var = @REGRESS( Y, X);

  NOBS = @SIZE( OBS);
  NEXP = @SIZE( EXPVAR);
  @WRITE( '   Explained error/R-Square: ', @FORMAT( RSQ, '16.8g'), @NEWLINE(1));
  @WRITE( '   Adjusted R-Square:        ',
       @FORMAT( 1 - ( 1 - RSQ) * ( NOBS - 1)/( NOBS - NEXP - 1), '18.8g'),@NEWLINE( 1));
  @WRITE( '   Mean square residual:     ', @FORMAT(var,'16.8g'), @NEWLINE(1));
  @WRITE( '   F statistic:              ', @FORMAT( F,'18.12f'), @NEWLINE(1));
  @WRITE( '   p statistic:              ', @FORMAT( p,'18.12f'), @NEWLINE(2));

  @WRITE( '   B( 0):     ', @FORMAT( B0, '16.8g'), @NEWLINE( 1));
  @FOR( EXPVAR( I):
     @WRITE( '   B( ', EXPVAR( I), '): ', @FORMAT( B( I), '16.8g'), @NEWLINE( 1));
    );
ENDCALC
```

Regression, e.g., Forecasts for Production Planning

```
Explained error/R-Square:          0.995479
Adjusted R-Square:                 0.99246501
Mean square residual:             92936.006
F statistic:                    330.285339234521
p statistic:                      0.000000000498

B( 0):             -3482258.6
B( PRICE):          15.061872
B( GNP__):       -0.035819179
B( JOBLS):         -2.0202298
B( MILIT):         -1.0332269
B( POPLN):       -0.051104106
B( YEAR_):          1829.1515
```



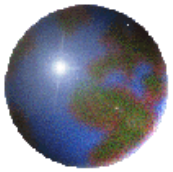
AllDiff Constraint (LINGO 17 & above)

You can specify that a certain set of variables must taken on different integer values.
Example: Sudoku puzzle: each digit appears once in

a) each column,
b) each row,
c) each of the nine 3x3 subsquares,
d) (optionally) the main diagonal,
e) (optionally) in the reflected diagonal;

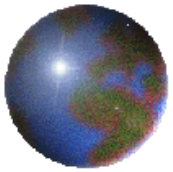
Sudoku Puzzle Solution:

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.   5   9   8   .   1   3   6   .   7   2   4   .
.   1   7   2   .   9   4   8   .   3   5   6   .
.   6   4   3   .   5   7   2   .   8   9   1   .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.   4   1   5   .   8   2   7   .   6   3   9   .
.   2   3   9   .   6   1   5   .   4   8   7   .
.   8   6   7   .   3   9   4   .   2   1   5   .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.   7   5   6   .   2   8   1   .   9   4   3   .
.   3   2   4   .   7   5   9   .   1   6   8   .
.   9   8   1   .   4   6   3   .   5   7   2   .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```



New Features: AllDiff Constraint

```
! For each row i, the entries must be all different integers;
 @for( side( i):
   @for( side(j):
      @alldiff( 'row'+side(i), y(i,j));
       );
     );

! For each col j, the entries must be all different;
 @for( side( j):
   @for( side(i):
      @alldiff( 'col'+side(j), y(i,j));
       );
     );

 ! Each entry is in the interval [1, 9];
@for( sxs(i,j):
  @bnd( 1, y(i,j), 9);
    );
```



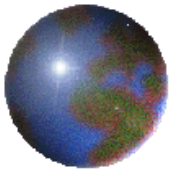
New Features: Enforcing Convexity–Positive Definiteness

Planning under Uncertainty Using Covariance Matrices.

```
! Make minimal adjustments to an
 initial guessed correlation matrix to
 make it a valid, Positive Semi-Definite matrix.

  Application:  If we have an incomplete data set,
e.g., not every variable appears in every observation,
we may be able to get
estimates of every correlation term individually,
however, taken together,
the initial matrix may not be Positive Definite, a feature
that should be true for any correlation or covariance matrix.
So for a "guessed" correlation matrix, we would like to make
minimal adjustments (move towards 0) of the off-diagonal
terms to make the matrix POSD;

   You need POSD for a Local Optimum to be a Global Optimum.
Most Quadratic Programming Solvers require that the
Q matrix be POSD.
```

J14 =WBPOSD(D14:G17)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Using the Positive Semi Definite feature, via the WBPOSD() function.**													
2														
3				A Guess at a covariance matrix										
4				that happens not to be Positive Semi-definite.										
5				IBM	GE	AAPL	XOM					Can you detect		
6			IBM	0.011	-0.013	0.012	-0.021			Inputs		that D6:G9 is not		
7			GE	-0.013	0.061	0.07	-0.094					positive definite?		
8			AAPL	0.012	0.07	0.06	0.057							
9			XOM	-0.021	-0.094	0.057	0.084							
10														
11				Select the adjustable values that is Positive Semi Definite										
12				and close to the original guessed matrix						Use WBPOSD() function to constrain range				
13				IBM	GE	AAPL	XOM			D14:G17 to be positive semi-definite.				
14			IBM	0.0169	0.0008	0.0002	-0.0088			**WBPOSD**				
15			GE	0.0008	0.0935	0.0423	-0.0653							
16			AAPL	0.0002	0.0423	0.0836	0.0325			Make sure: Quadratic Recognition set to On,				
17			XOM	-0.0088	-0.0653	0.0325	0.1094			and Adjustable cells are set to free type.				
18														
19														
20				The matrix of differences between the two matrices										
21				IBM	GE	AAPL	XOM							
22			IBM	-0.00587	-0.0138	0.01176	-0.01221							
23			GE	-0.0138	-0.03247	0.02766	-0.02871							
24			AAPL	0.01176	0.02766	-0.02357	0.02447							
25			XOM	-0.01221	-0.02871	0.02447	-0.02539							
26														
27				**Minimize the sum of squared differences**										
28				0.00762										



Multi-Period Optimization Models, Steady State, Eigenvalues

A linear model would be represented by a matrix A describing how the different species interact, so in vector notation:

$$P(t) = A*P(t\text{-}1);$$

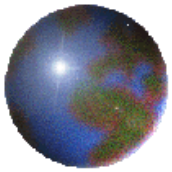
If we are interested in steady (exponential) growth or decay, we might ask, is there a scalar constant growth rate, λ, so that:

$$P(t) = A*P(t\text{-}1) = \lambda*P(t\text{-}1);$$

i.e., each species grows or decays by the same factor λ each period, or more simply, is there a "steady state growth" solution to:

$$A*P = \lambda*P;$$

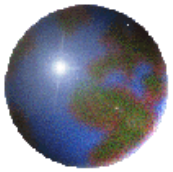
The is the eigenvalue equation.This is easy to solve in LINGO with its matrix commands.



Eigenvalues in LINGO & Steady State Solutions

```
! Compute eigenvalues and eigenvectors; !(EigenExample.lng);
! Keywords: Eigenvalues, PCA, Population growth;
sets:
  item: evalr, evali;
  ixi( item, item): a, evecr, eveci;
endsets

data:
  item = 1..3; ! The number of items;
! The matrix of interest;
  a =
! A population change matrix, P(t+1) = A*P(t);
   ! Bunnies Clover  Foxes;
     0.612   2.291  -1.200
    -0.295   2.321  -0.560
     0.500  -0.050   0.110 ;
 ! Notice that Bunnies have a negative effect (-0.295 ) on clover.
   Clover has a very positive effect (2.291) on Bunnies.
   Foxes are not very good (-1.2) for Bunnies;
enddata

calc:
 ! Get the
    eigenvalues(real part), eigenvectors(real part),
    eigenvalues(imaginary part),and eigenvectors(imaginary part);

  evalr, evecr, evali, eveci = @eigen(a);
endcalc
```

LINDO SYSTEMS INC.



Eigenvalues in LINGO

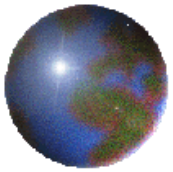
```
! There are in fact three stable growth configurations;
 eigenvalues, real part:
   1    0.7066731
   2                 1.011007
   3                                  1.325320

 eigenvectors(columns), real part:
                  1             2             3
   1    0.7350684    0.8206666    0.8397024
   2    0.3381721    0.3706966    0.4330663
   3    0.5876343    0.4348452    0.3276485
```

Eigenvalues in LINGO

For example, if we started with 82067 Bunnies, 37070 units of Clover, and 43485 Foxes, and then the populations would remain essentially unchanged, (eigenvalue = 1.011).

If we start with a different configuration, the populations will grow.

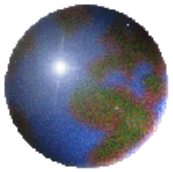
	BUNNIES	Growth Rate	CLOVER	Growth Rate	FOXES	Growth Rate
	83970.0		43307.0		32765.0	
1	111288.0	1.325	57396.0	1.325	43423.8	1.325
2	147493.9	1.325	76068.8	1.325	57550.8	1.325
3	195479.0	1.325	100816.6	1.325	76274.1	1.325
4	259075.0	1.325	133615.5	1.325	101088.8	1.325
5	343360.5	1.325	177084.7	1.325	133976.5	1.325
6	455065.9	1.325	234695.5	1.325	177563.4	1.325
7	603111.6	1.325	311048.3	1.325	235330.2	1.325
8	799319.6	1.325	412240.2	1.325	311889.7	1.325
9	1059358.2	1.325	546351.9	1.325	413355.7	1.325



College Planning Under Uncertainty in LINGO

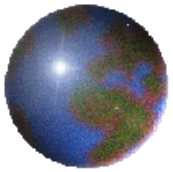
Lingo Model - SPCollegeDisc

```
MODEL:      ! (SPCollegeDisc.lg4);
! How to finance a college education, four stages
  from now, with uncertain stocks and bonds investments;

! Core Model ------------------------------------------+;
 SETS:
   TIME;      ! The time periods/stages;
   ASSETS:;   ! Assets available each stage;
   TXA( TIME, ASSETS): RETURN, INVEST;
 ENDSETS

 DATA:
   INITIAL = 55; !Initial wealth;
   GOAL = 80;    ! What we need later;
   PENALTY = 4;  !  for falling short of goal;
   TIME = T0..T3;! The stages;
   ASSETS = BONDS, STOCKS; ! Investmens available;
ENDDATA

! 1) Core model;
MIN = PENALTY * UNDER - OVER;

! Stage 0;
[R_INIT] @SUM( ASSETS( A): INVEST( 1, A)) = INITIAL;
         @FOR( ASSETS( A): RETURN( 1, A) = 0);
! Wealth this period = return*(wealth last period);
@FOR( TIME( T) | T #GT# 1:
  @SUM( ASSETS( A): RETURN( T, A) * INVEST( T - 1, A)) =
   @SUM( ASSETS( A): INVEST( T, A))
);

FINAL = @SUM( ASSETS( A): INVEST( @SIZE( TIME), A));
OVER - UNDER = FINAL - GOAL; ! Deviation from goal;
```



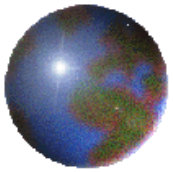
College Planning Under Uncertainty in LINGO - II

```
! SP Related Declarations ----------------------------+;
!  2) Staging information;
!    Declare INVEST(T,A) to be a
     decision variable of stage T-1;
@FOR( TXA( T, A):
   @SPSTGVAR( T-1, INVEST( T, A));
    );

! Declare  RRETURN(T,A) to be a
   random variable of stage T-1;
@FOR( TXA( T, A) | T #GT# 1:
   @SPSTGRNDV( T - 1, RETURN( T, A));
     );

! 3) Connect the random variables RETURN(T,A)
      to a scenario table;
! Construct the outcome table;
SETS:
   OUTCOMES;
   SXA( OUTCOMES, ASSETS): O_RETURN;
ENDSETS
DATA:
   OUTCOMES = GOOD, BAD;
   O_RETURN =
        1.14 1.25
        1.12 1.06 ;
ENDDATA

! Declare the parent discrete distribution;
@SPTABLESHAPE( 'D1', @SIZE( OUTCOMES), @SIZE( ASSETS));
```



Continuous Time, and How to Use it

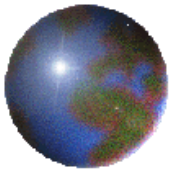
```
!Illustrate use of Calendar/time feature of LINGO;
! Keywords: Calendar, Date function, GMT, Time arithmetic;
SETS:
 MONAME; ! Month names;
 AMPM;   ! AM/PM label;
 DAYWK;  ! Day of week names;
ENDSETS
DATA:
 MONAME = JAN..DEC;
 AMPM = AM PM;
 DAYWK = SUN..SAT;
ENDDATA
CALC:
  @SET( 'TERSEO',2);  ! Output level (0:verb, 1:terse, 2:only errors, 3:none);
! GMT (Greenwich Mean Time) offset in hours ;
   GMTOFFSETLV = -8; ! for LasVegas ;
   GMTOFFSETHN = -10; ! for Honolulu;
 ! Day Light Saving time offset for Summer;
   DSTOFFSETLV = 1; ! LasVegas does observe DST;
   DSTOFFSETHN = 0; ! Honolulu does not;
   flgtm = 6.4; ! Flight time in hours;

! When our flight departs LasVegas;
   yr = 2017;   mo = 4;  da = 3;
   dwk = 2; hr = 9;   mn = 5;  sc = 0;
```



Time, and How to Use it - II

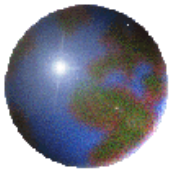
```
! Convert LasVegas time to scalar time in seconds;
   stimelv = @YMD2STM( yr, mo, da, hr, mn, sc);

@WRITE( 'Depart LasVegas: ',yr,' ',  MONAME(mo),' ',
   @FORMAT(da,'2.0F'),' ',DAYWK(dwk),' ', @FORMAT(HR,'2.0F'),
   'h ',@FORMAT(mn,'2.0F'),'m ',
     @FORMAT(sc, '2.0f'),'s', @NEWLINE(1));

! Adjust for DST and to GMT, LasVegas;
  stimelv = stimelv - 3600*( GMTOFFSETLV + DSTOFFSETLV);

! Arrival time, GMT, in Honolulu;
  stimehn = stimelv + 3600* flgtm;
@WRITE(' Flight time is ', flgtm,' hours.', @NEWLINE(1));

! Adjust for GMT and DST back to local time in Honolulu;
  stimehn = stimehn + 3600 *( GMTOFFSETHN + DSTOFFSETHN);
! Convert to YMDHMS;
   yr, mo, da, dwk, hr, mn, sc = @STM2YMDHMS( stimehn);
@WRITE( 'Arrive at  Honolulu: ',yr,' ',  MONAME(mo),' ',@FORMAT(da,'2.0F'),
   ' ',DAYWK(dwk),' ', @FORMAT(HR,'2.0F'),'h ',
   @FORMAT(mn,'2.0F'),'m ',
     @FORMAT(sc, '2.0f'),'s ', @NEWLINE(1));
```

```
Depart LasVegas: 2017 APR  3 MON  9h  5m  0s
Flight time is 6.4 hours.
Arrive at  Honolulu: 2017 APR  3 MON 12h 29m  0s
```



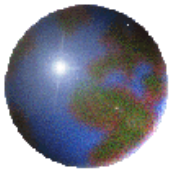
Jet Taxi Routing Problem, Time Functions in LINGO

```
! The (Jet) Taxi Routing Problem.
 Given a set of desired flights or trips to be covered,
 figure out how to route planes/vehicles to cover these flights.
 Repositioning/deadheading flights are allowed at a cost.
 This model illustrate several features:
  1) Calendar routines for coordinating the times of
       flights involving different locations and time zones,
  2) The @INSERT( ) function for creating a derived set
       of repositioning/deadhead legs according to fairly arbitrary rules.
       We do not know in advance which deadhead legs might be needed.
  3) Charting/Graphing routines for displaying a network;
! Define the data structures;
SETS:
 CITY: INITA, GMTOFF, LATI, LNGT;
 LEG ;
 CXC( CITY, CITY): TRVTIM;
 ! Loaded legs;
 LODPAIR( LEG, CITY, CITY): Year, Month, Day, Hour, Minute,
      DLTIME, Y, PLFLAG,
      DCITY, ACITY, ALTIME;
 RLEG;
! Reposition city pairs;
 RPAIR( RLEG, CITY, CITY): DRTIME, U;
 DOW /SUN..SAT/;  ! Days of week;
 CARCS: ORG, DST; ! List to be created of OD pairs;
! Set of repositioning legs actually used;
 RPAIRU(RLEG, CITY, CITY): DUCITY, AUCITY, DUTIME, AUTIME ;
ENDSETS
```



Jet Taxi Routing Problem

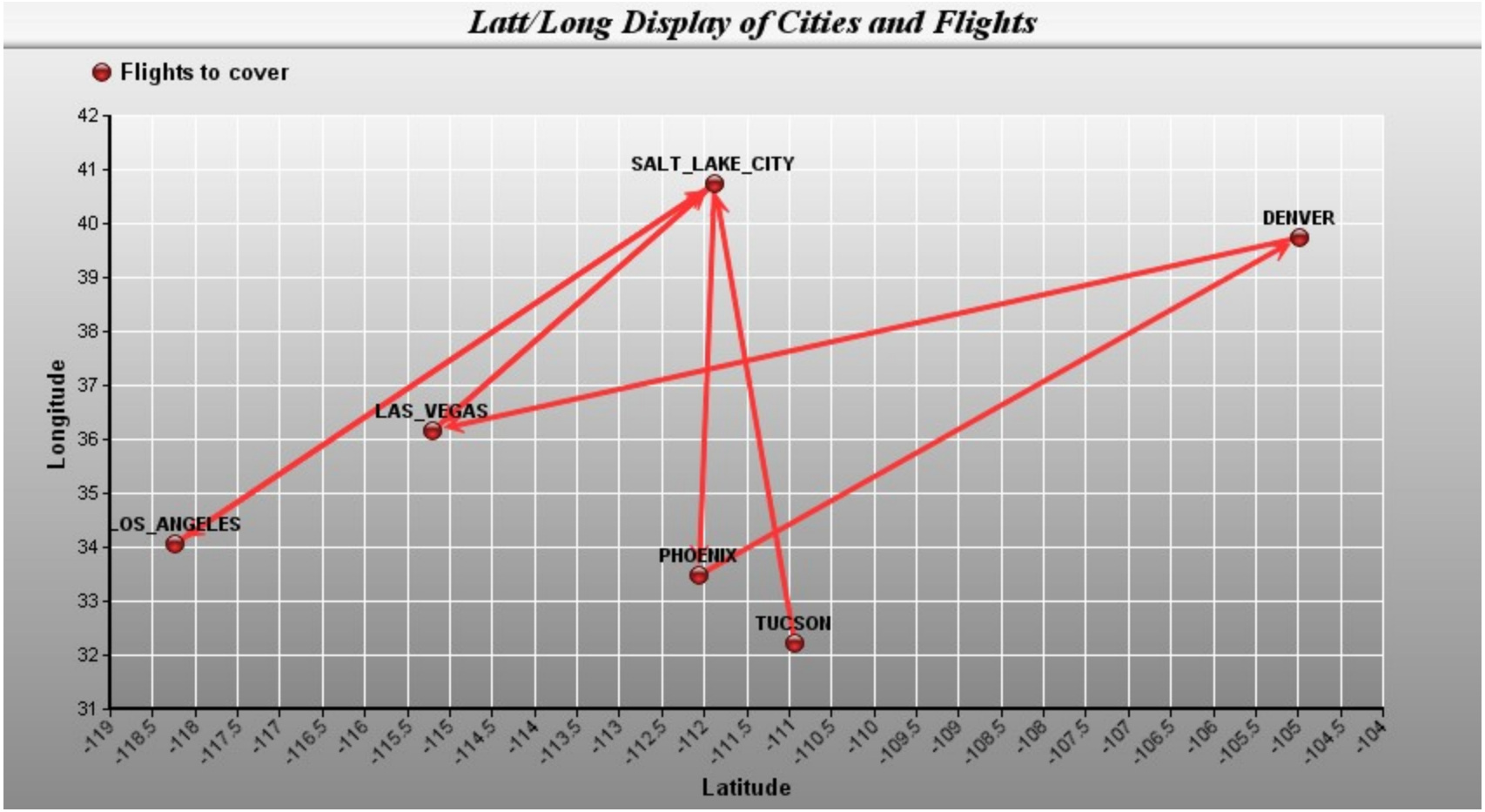
```
DATA:
! Scalar data;
  NRPLG = 1000; ! Max number reposition legs;
  NLLG = 7;     ! Number loaded legs;
  VL = 1;     ! Relative value of covering a loaded flight;
  RP = 0.35;  ! Relative cost of a repositioning flight;
  RA = 0.8;   ! Relative cost of an aircraft;
  LA = 2;     ! Limit on number of aircraft;

  RLEG = 1..NRPLG; ! Possible number of repositioning legs;
  LEG = 1..NLLG; ! Get data on each loaded candidate OD pair;

! Vector data;
!  The Cities, their offset in hours from GMT, Lat and Long;
      City,      GMTOFF,  LATI,    LNGT=
! 1; Denver          -6  39.7392 -104.9903 ! Denver is 6 hours ;
! 2; Salt_Lake_City  -6  40.7500 -111.8833 ! behind Greenwich Mean Time
! 3; Los_Angeles     -8  34.0522 -118.2428
! 4; Phoenix         -7  33.4833 -112.0667
! 5; Las_Vegas       -7  36.1667 -115.2000
! 6; Tucson          -7  32.2217 -110.9258;
```



Jet Taxi Routing Problem

```
!  The city pair trips we want to cover/service;
  LODPAIR, Year, Month, Day, Hour, Minute =
!           Origin         Destination                      Departure time
LEG         City                        City          Year    Month        Day Hour Minute ;
1     Los_Angeles       Salt_Lake_City  2016     4     22    10     0
2     Salt_Lake_City    Phoenix         2016     4     23    16    20
3     Salt_Lake_City    Los_Angeles     2016     4     25    16     0
4     Phoenix           Denver          2016     4     24    11    20
4     Salt_Lake_City    Las_Vegas       2016     4     26    16     0
5     Las_Vegas         Salt_Lake_City  2016     4     27    12     0
6     Tucson            Salt_Lake_City  2016     4     23    14     0
7     Denver            Las_Vegas       2016     4     24     8    30
;

! Matrix data;
!  Get travel time matrix in minutes;
! Den   SLC   LAX   Phn   Lvg   Tuc;
  TRVTIM =
    0    85   155   120   115   120  ! Denver;
   80     0   110   100    85   120  ! Salt_Lake_City;
  145   110     0   120   120   120  ! Los_Angeles;
  110   100   120     0    85    60  ! Phoenix;
  110    85   120    85     0    95  ! Las_Vegas;
  110   120   120    60    95     0 ;! Tucson;

ENDDATA
```

Jet Taxi Routing Problem, Charting/Graphs

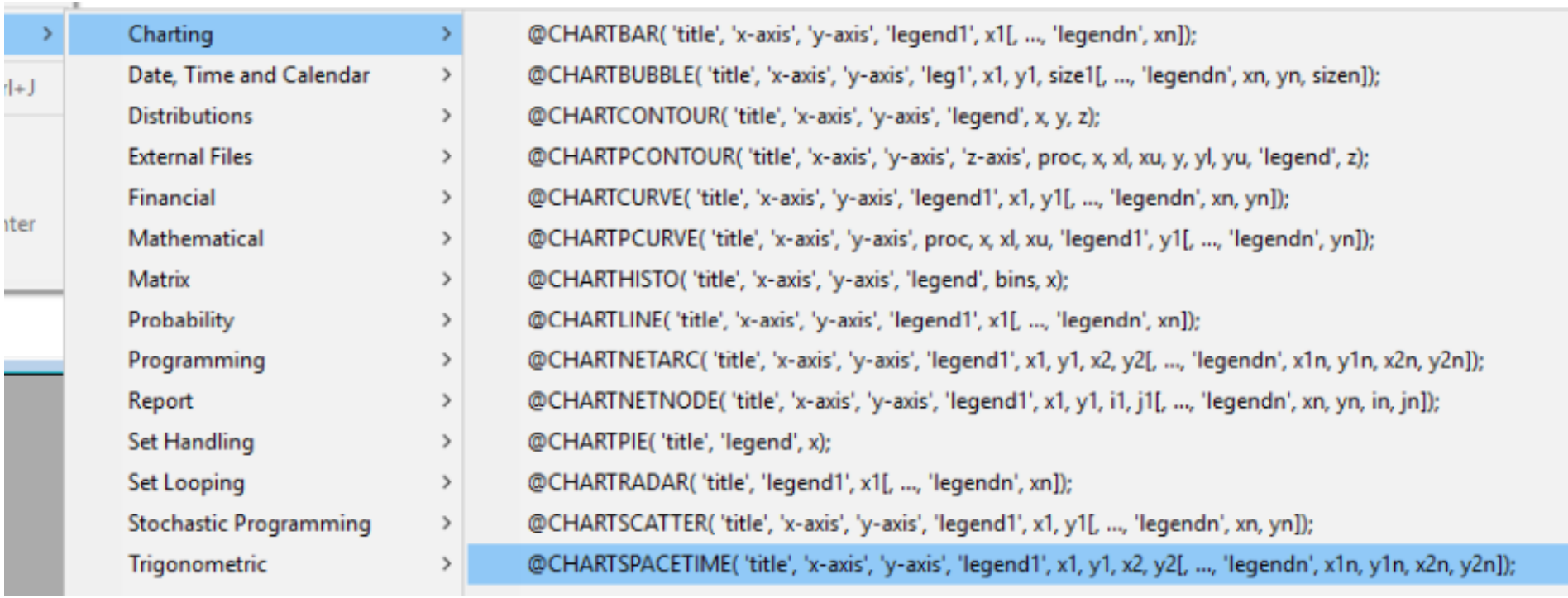


LINDO SYSTEMS INC.

Charts/Graphs: How to Find Types Available

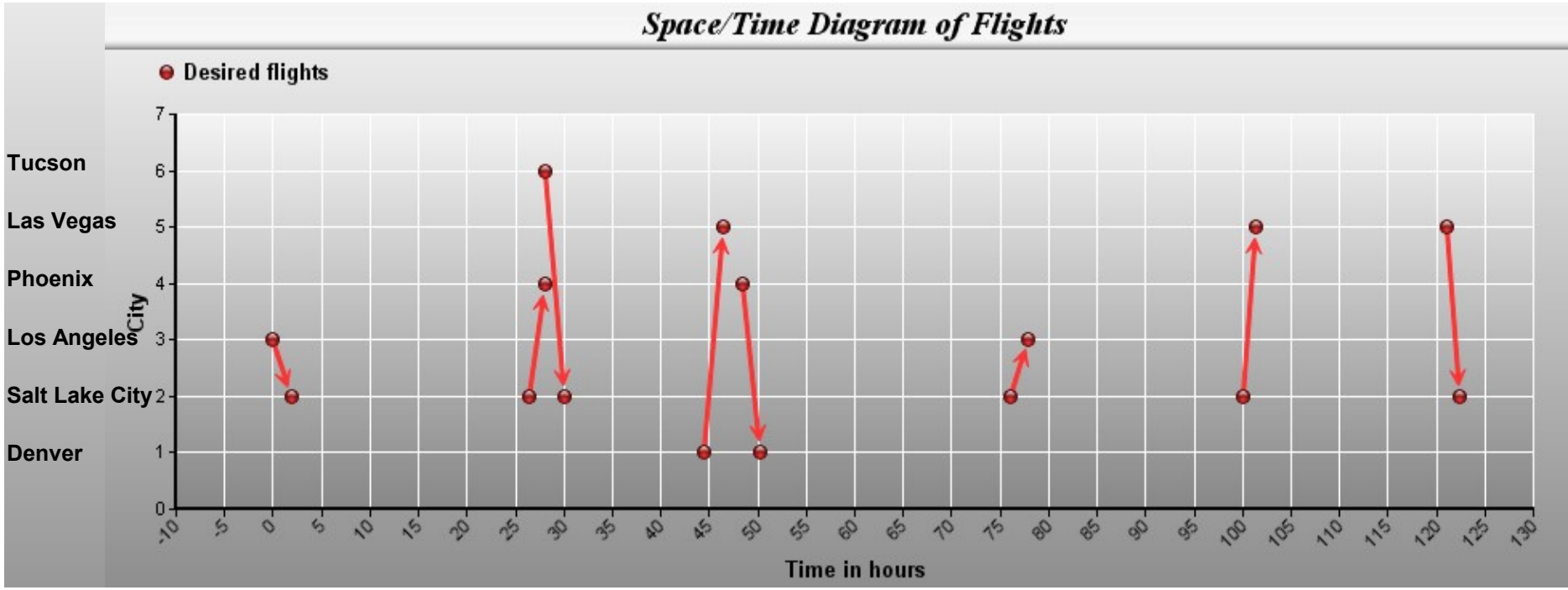
To insert a particular chart type, click on:

*Edit -> Paste function -> Charting -> @CHART****

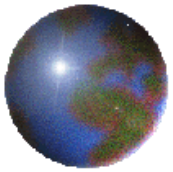


Jet Taxi Routing Problem

How many aircraft would you need to cover all these flights?

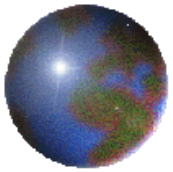


LINDO SYSTEMS INC.



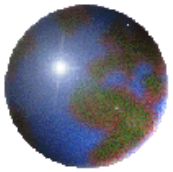
Jet Taxi Routing Problem

```
SUBMODEL ROUTEM:
 ! Variables:
      Y(n,d,a) = 1 if we do flight leg n,
                    from city d to city a,
      U(n,d,a) = 1 if we do the nth possible deadheading flight,
                    from city d to city a;

 ! Maximize number of requested flights flown
  - cost of repositioning flights
  - cost of aircraft;
  MAX = VL*@SUM( LODPAIR( n, d, a): Y(n,d,a)) ! Loaded flights;
      - RP*@SUM( RPAIR( n, d, a): U(n,d,a)) ! Repositions;
      - RA*@SUM( CITY(i): INITA(i)); ! Initial AC at city i;

 ! You either fly it or you do not;
  @FOR( LODPAIR( n, d, a): @BIN(Y(n,d,a)));
  @FOR( RPAIR( n, d, a): @BIN(U(n,d,a)));

 ! For every departing loaded flight from d to a at time DLTIME,
  the number of earlier arrivals - earlier departures must be >= Y(n,d,a);
 @FOR( LODPAIR( n, d, a):
  [LFLO] INITA(d)                ! Note, scalar time is in seconds, not minutes;
  + @SUM( LODPAIR( n1, d1, d) | DLTIME(n1,d1,d) + TRVTIM(d1,d)*60 #LE# DLTIME(n,d,a):
       Y(n1,d1,d))  ! loaded flights into d;
  + @SUM( RPAIR( n1, d1, d) | DRTIME(n1,d1,d) + TRVTIM(D1,d)*60 #LE# DLTIME(n,d,a):
       U(n1,d1,d))  ! Dead-head (unloaded) flights into d;
  - @SUM(LODPAIR(n1,d,a1) | DLTIME(n1,d,a1) #LT# DLTIME(n,d,a):
       Y(n1,d,a1))  ! Loaded flights out of d;
  - @SUM( RPAIR(n1,d,a1) | DRTIME(n1,d,a1) #LE# DLTIME(n,d,a):
       U(n1,d,a1))  ! Dead head flights out;
    >= Y(n,d,a);
     );
```



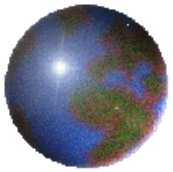
Jet Taxi Routing Problem

```
CALC:
  @SET( 'TERSEO', 2);  ! Set output to terse;
  ! Convert Year, Month, Day, Hour, Minute(n,d,a) to a scalar DLTIME(n,d,a)
    so we can do simple comparisons and arithmetic;
  ! Compute departure time in scalar GMT time for each loaded flight;
  @FOR( LODPAIR( n, d, a):
    DLTIME(n,d,a) = @YMD2STM( Year(n,d,a), Month(n,d,a), Day(n,d,a) , Hour(n,d,a), Minute(n,d,a), 0)
                    - 3600*GMTOFF(d); ! Take into account local time, convert hours to seconds;
      );

 ! Construct the set of candidate repositioning legs. For each departing
  flight from city j, we add a candidate repositioning leg from every
  other city i, j #NE# i, at TRVTIM(i,j) minutes earlier ;
  k = 0;
  @FOR( LODPAIR( n, j, a):
    @FOR( CITY(i) | i #NE# j:
      k = k+1;
  ! Insert into the RPAIR set, the triple (k, i, j);
      @INSERT( RPAIR, k, i, j);
  ! Now that the element has been added to the set, we can set its attributes;
      ! Note, travel times are in minutes, scalar TRVTIM(i,j) in seconds;
      DRTIME(k,i,j) = DLTIME(n,j,a) - TRVTIM(i,j)*60;
        );
      );
```



Jet Taxi Routing Problem

```
@WRITE(@NEWLINE(1),' Repositioning Flights:',@NEWLINE(1));
@WRITE('    Origin              Destination       yyyy  mm  dd hh  mm dwk',@NEWLINE(1));

@FOR( RPAIR( n, d, a) | U(n,d,a) #GT# 0.5:
  ! Convert DRTIME(n,d,a) back to year month, day, hour minute;
    CTIME = DRTIME(n,d,a)+ 3600*GMTOFF(d); ! Take into account local time, convert hrs to secs;
    IYR, IMON, IDAY, IWKD, IHR, IMIN, ISEC = @STM2YMDHMS( ctime);

    @WRITE( @FORMAT(CITY(d),'18s'),'   ',@FORMAT(CITY(a),'18s'),'   ',IYR,'  ',
    IMON,'  ',@FORMAT(IDAY,'2.0F'),' ',@FORMAT(IHR,'2.0F'),'  ',@FORMAT(IMIN,'2.0F'),'
',DOW(IWKD),@NEWLINE(1));
    );

! Build vectors to prepare to draw various networks;
   @FOR( LODPAIR(n,d,a):
     DCITY(n,d,a) = d;  ! Departure city of leg;
     ACITY(n,d,a) = a;  ! Arrival city of leg;
! Time that a flight arrives at destination city;
     ALTIME(n,d,a) = DLTIME(n,d,a) + TRVTIM(d,a)*60;
       );

! If we want to plot;
  PLOTIT;
! Write Flat file of input and output;
!  WriteFlatFile;
 ENDCALC
```



Jet Taxi Routing Problem

```
Value/trip covered= 1
Relative cost/repositioning= 0.35
Relative cost/aircraft= 0.8
Number aircraft allowed= 2

Number requested trips covered= 8 (of 8)
Number aircraft used= 2

Cities and their coordinates:
           DENVER     39.739   -104.9905     -6   0
    SALT_LAKE_CITY     40.750   -111.8835     -6   0
       LOS_ANGELES     34.052   -118.2435     -8   1
           PHOENIX     33.483   -112.0675     -7   0
         LAS_VEGAS     36.167   -115.2005     -7   0
            TUCSON     32.222   -110.9265     -7   1
```

Initial Aircraft Positions:

City	#Aircraft
LOS_ANGELES	1
TUCSON	1

Trips flown:

Origin	Destination	Depart at(local time) yyyy	mm	dd	hr	mm	dwk
LOS_ANGELES	SALT_LAKE_CITY	2016	4	22	10	0	FRI
SALT_LAKE_CITY	PHOENIX	2016	4	23	14	20	SAT
TUCSON	SALT_LAKE_CITY	2016	4	23	15	0	SAT
DENVER	LAS_VEGAS	2016	4	24	8	30	SUN
PHOENIX	DENVER	2016	4	24	11	20	SUN
SALT_LAKE_CITY	LOS_ANGELES	2016	4	25	16	0	MON
SALT_LAKE_CITY	LAS_VEGAS	2016	4	26	16	0	TUE
LAS_VEGAS	SALT_LAKE_CITY	2016	4	27	12	0	WED

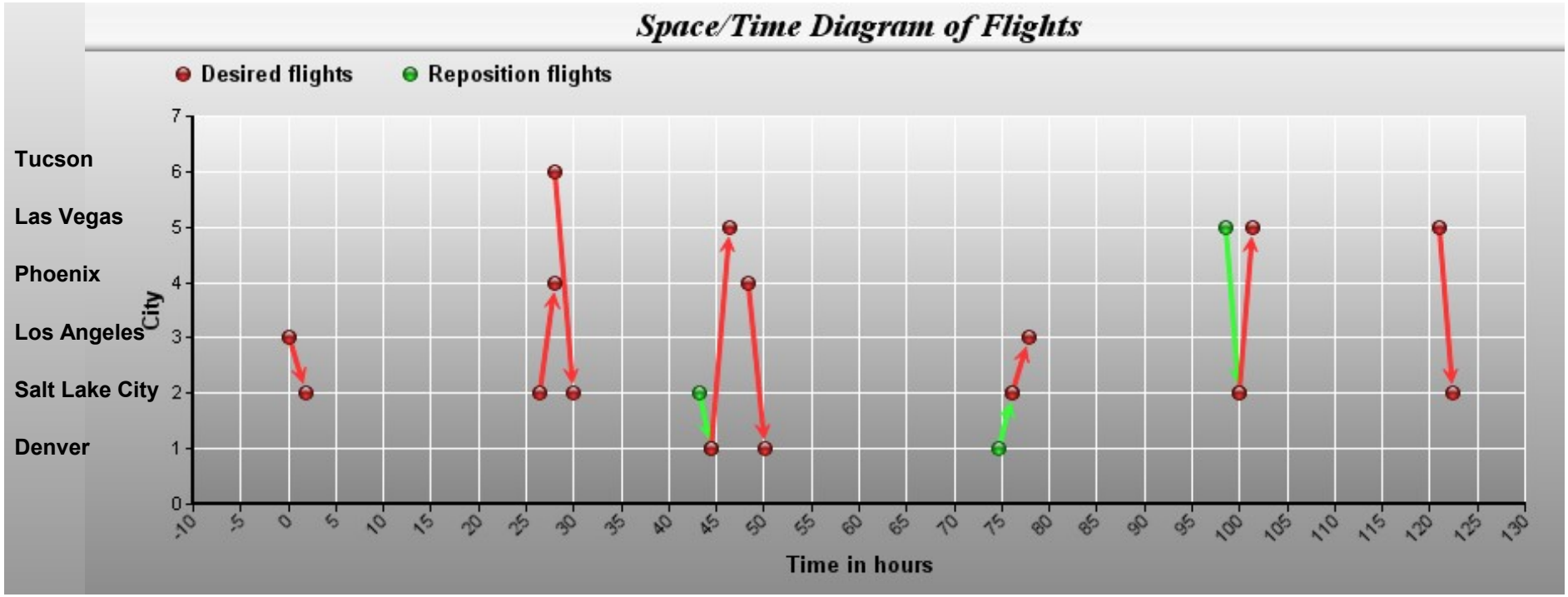
Repositioning Flights:

Origin	Destination	yyyy	mm	dd	hh	mm	dwk
DENVER	SALT_LAKE_CITY	2016	4	25	14	35	MON
LAS_VEGAS	SALT_LAKE_CITY	2016	4	26	13	35	TUE
SALT_LAKE_CITY	DENVER	2016	4	24	7	10	SUN

LINDO SYSTEMS INC.

Jet Taxi Routing Problem

Deadheading/Repositioning Flights Added



LINDO SYSTEMS INC.