STIMATH
User Guide

Software Techniques Inc.

773069 Hwy. 10 (RR#2)

Proton Station, ON. N0C 1L0

Phone: (519) 923-5111

Technical Support: support@stiwww.com

Introduction

STIMATH is a high-level tool for mathematics, linear algebra, plotting and expression evaluation that can be used from any programming environment or even from a command line interface. It is an executable that takes a single argument, the name of a text file containing a series of MATLAB style expressions with virtually the full range of MATLAB expressions being supported. The output from processing the expressions can be written files or used to generate plots.

Key Features

- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving differential equations and boundary value problems
- Matrix-based notation
- Extensive graphics for visualizing data and tools for creating custom plots
- Variables of any data type can be dynamically defined and used in expressions

Categories of Functions and Commands

- Elementary Math
- Trigonometry
- Exponents and Logarithms
- Complex Numbers
- Discrete Math
- Polynomials
- Special Functions such as Bessel, Error, Gamma and many more
- Cartesian Coordinate System Conversion
- Constants and Test Matrices
- Matrix Operations such as dot, cross and transpose
- Linear Equations
- Matrix Decomposition
- Eigenvalues and Singular Values
- Matrix Functions and Analysis
- Descriptive Statistics
- Random Number Generation
- 1D and Multidimensional Interpolation
- Optimization
- Numerical Integration and Ordinary Differential Equations
- Boundary Value Problems
- Delay Differential Equations
- Partial Differential Equations
- Numerical Integration and Differentiation
- Fourier Analysis and Filtering
- Sparse Matrix Creation and Manipulation Including Graphing and Tree Algorithms
- Computational Geometry
- 2-D and 3-D Plots
- Volume Visualization
- Array Creation and Concatenation
- Indexing
- Sorting and Reshaping Arrays
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Set Operators
- String Creation, Concatenation and Parsing
- Time Series
- Data Type Conversion and Identification
- Functions for Processing Dates and Time
- Data Import and Export

Software Installation

STIMATH uses the MATLAB Compiler Runtime (MCR) and therefore MCR must be installed in order to use the STIMATH executable. STIMATH uses version 8.1 (R2013a) of MCR. Download the Windows 64-bit version of the MCR for R2013a from the MathWorks website

http://www.mathworks.com/products/compiler/mcr/index.html

Follow instructions for installation of MCR. Next install and run the STIMATH licensing application. On first time use, the licensing application will detect that the application has not been registered in the Windows registry and will attempt to validate the license key, which a user will have received by email following payment confirmation. Enter the license key in textbox (shown below) and click the Validate button. The validation and registration is done on our server and therefore requires an Internet connection. Once installed, the STIMATH executable can reside in any folder a user wishes to use.

How To Use STIMATH

STIMATH is an executable that takes a single argument, which is the name of a text file containing one or more expressions to be evaluated. STIMATH supports standard MATLAB expression syntax.

For example, if we want to plot y(x)=sin(x) in the interval $-2\pi \le x \le 2\pi$ at point spacing of 0.01, we would save the following expression as a text file.

```
x=-2*pi:0.01:2*pi;
y=sin(x);
plot(x,y)
```

then enter

STIMATH *filename*

where *filename* is the name of the text file containing the expressions. The semicolon is used to suppress display of any output from that command or to separate commands entered on the same line. As with MATLAB syntax, some commands such as plot do not require use of the semicolon.

If we want to add more detail to the plot such as axis labels and a title, we would write

```
x=-2*pi:0.01:2*pi;
y=sin(x);
plot(x,y)
```

```
xlabel('x')
ylabel('sin(x)')
title('Plot of sin(x)')
```

Details from any error resulting from the evaluation of an expression are logged to a file called ErrorLog.txt, in the same folder as the STIMATH executable. If the file is not present, it means that the expression entered was executed without any errors.

The following series of examples offer a tiny range of the calculations that can be performed with STIMATH.

Sample projects demonstrate how STIMATH can be used from VB.NET and C# in Visual Studio; however STIMATH can be used from any programming environment such as C++, FORTRAN or even from the command line environment of DOS.

Example 1

Description

Define a vector variable x that spans a range of values for example between 0 and pi in steps of 0.001. Define another vector variable y that is a function of x such as sin(x). Concatenate arrays x and y to form a 2 column array and write data to text file called "myY.dat" in comma-separated value (csv) format with numerical precision specified in C format as "%10.5f".

Script

```
x=0:0.001:pi;
y=sin(x);
dlmwrite('C:\STIMATH\Output\MyY.dat',[x' y'],'delimiter',',','precision','%10.5f')
```

Output

An array composed of concatenating the transpose of vectors x and y are written to a comma-delimited file MyY.dat with precision specified by "10.5f".

Example 2

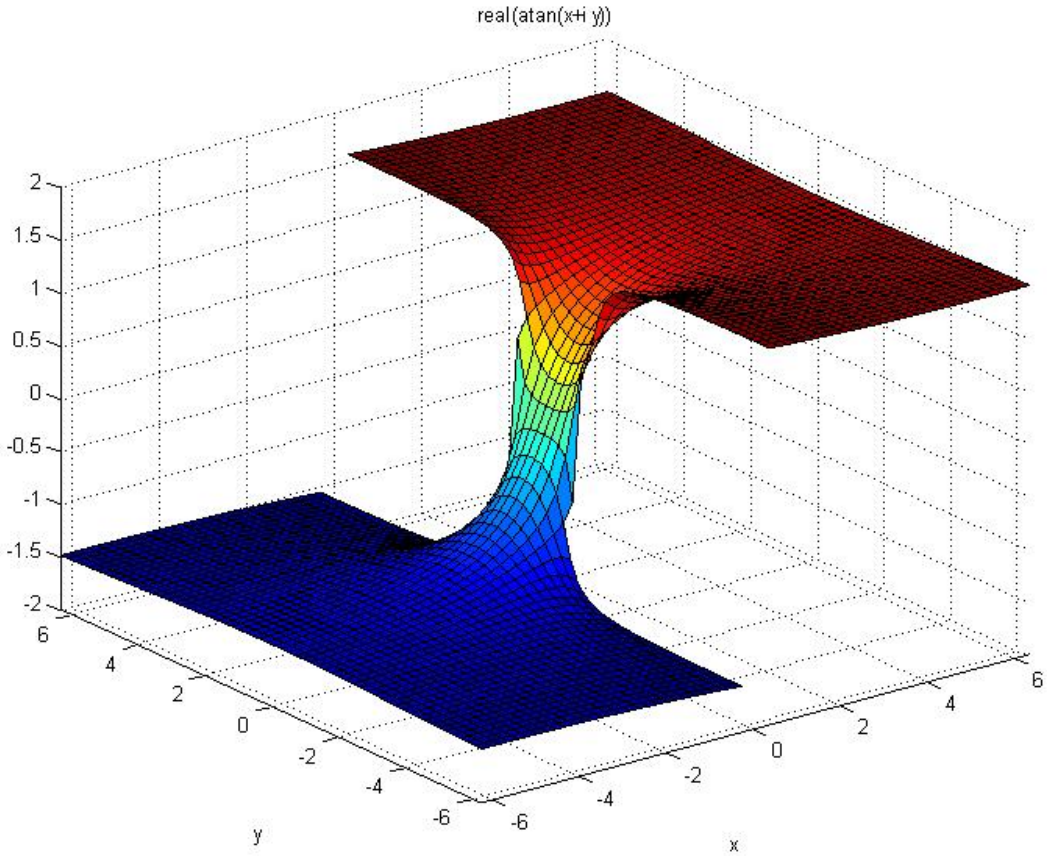
Description

Graph the function f(x,y) =real($\tan^{-1}$(x+i*y)) over the default domain $-2\pi < x < 2\pi$, $-2\pi < y < 2\pi$ using the ezsurf function and save the plot to file as a jpeg to a file called MyPlot1.jpg. Do not show the plot figure window.

Script

```
h=figure('Visible','Off');
ax=axes;
ezsurf('real(atan(x+i*y))');
print(h,'-djpeg','-r100','C:\STIMATH\Output\MyPlot1')
close(h)
```

Output



Example 3

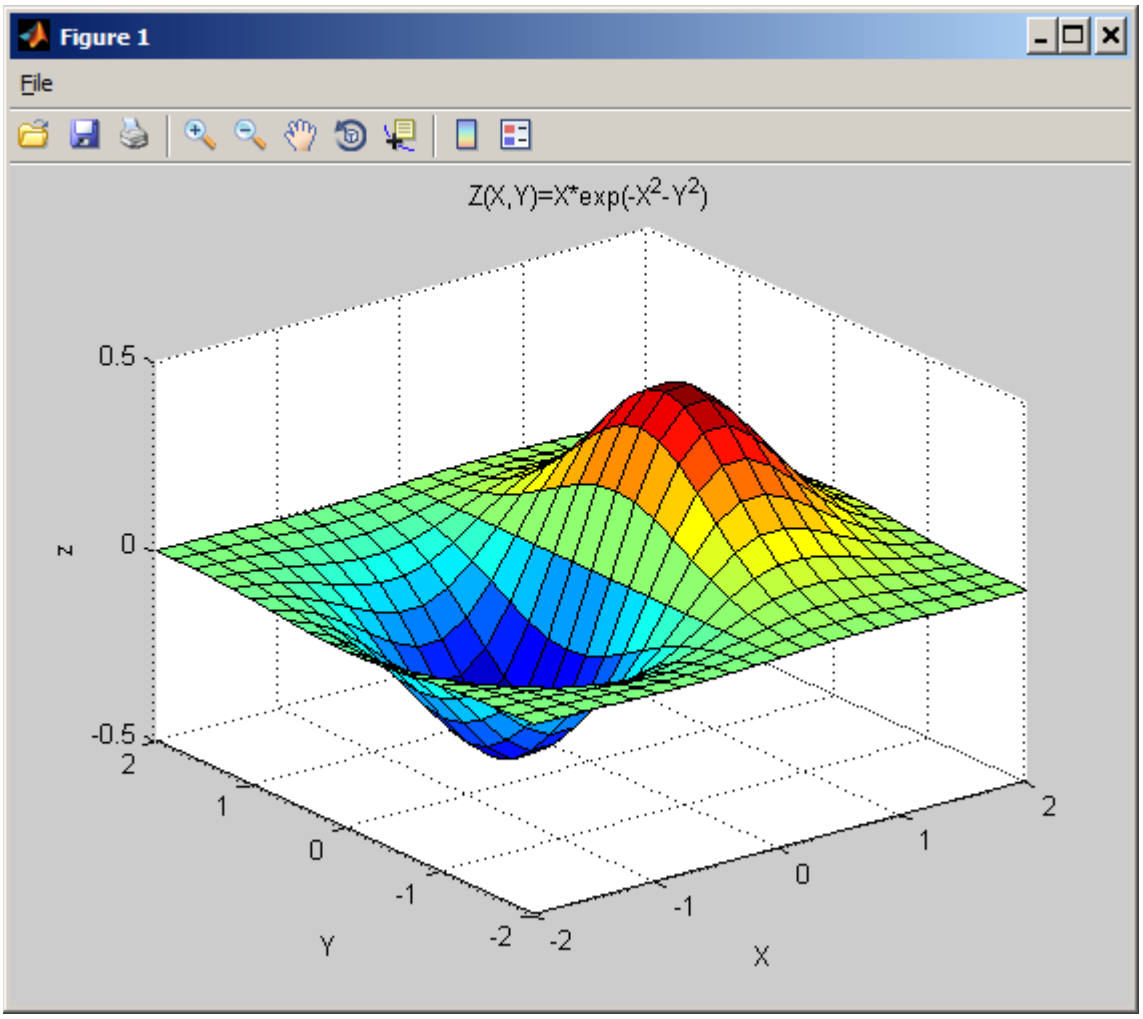
Description

Graph the function $Z = e^{(-X^2-Y^2)}$ where X and Y vary between -2 and 2 in steps of 0.2 using the surf function and display the plot. Use the function meshgrid to create a rectangular array for X and Y.

Script

```
[X,Y]=meshgrid(-2:.2:2, -2:.2:2);
Z=X .* exp(-X.^2 - Y.^2);surf(X,Y,Z);
xlabel('X')
ylabel('Y')
zlabel('z')
title('Z(X,Y)=X*exp(-X^2-Y^2)')
```

Output



Example 4

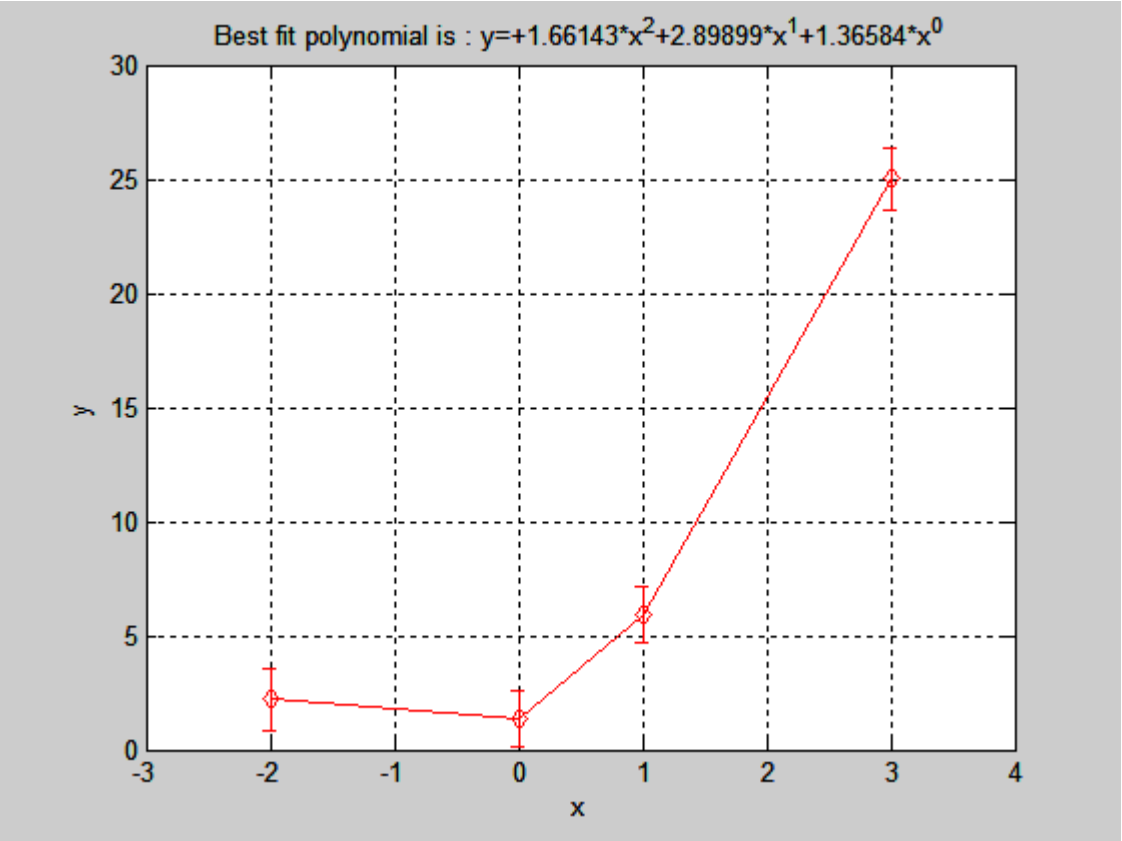
Description

Find the best-fit 2nd order polynomial to data points and plot error bars corresponding to fit. Plot data along with error bars.

Script

```
x=[-2,0,1,3];
y=[2.0782,2.0427,5.2494,25.1510];
PolynomialOrder=2;
[P,S]=polyfit(x,y,PolynomialOrder);
[Y,DELTA]=polyval(P,x,S);
errorbar(x,Y,DELTA,'rd-')
BestFitString=[];
for ii=1:PolynomialOrder+1
 BestFitString=[BestFitString num2str(P(ii),'%+g') '*x^' num2str(PolynomialOrder-ii+1)];
end
title(['Best fit polynomial is : y=' BestFitString])
xlabel('x')
ylabel('y')
grid on
```

Output



Example 5

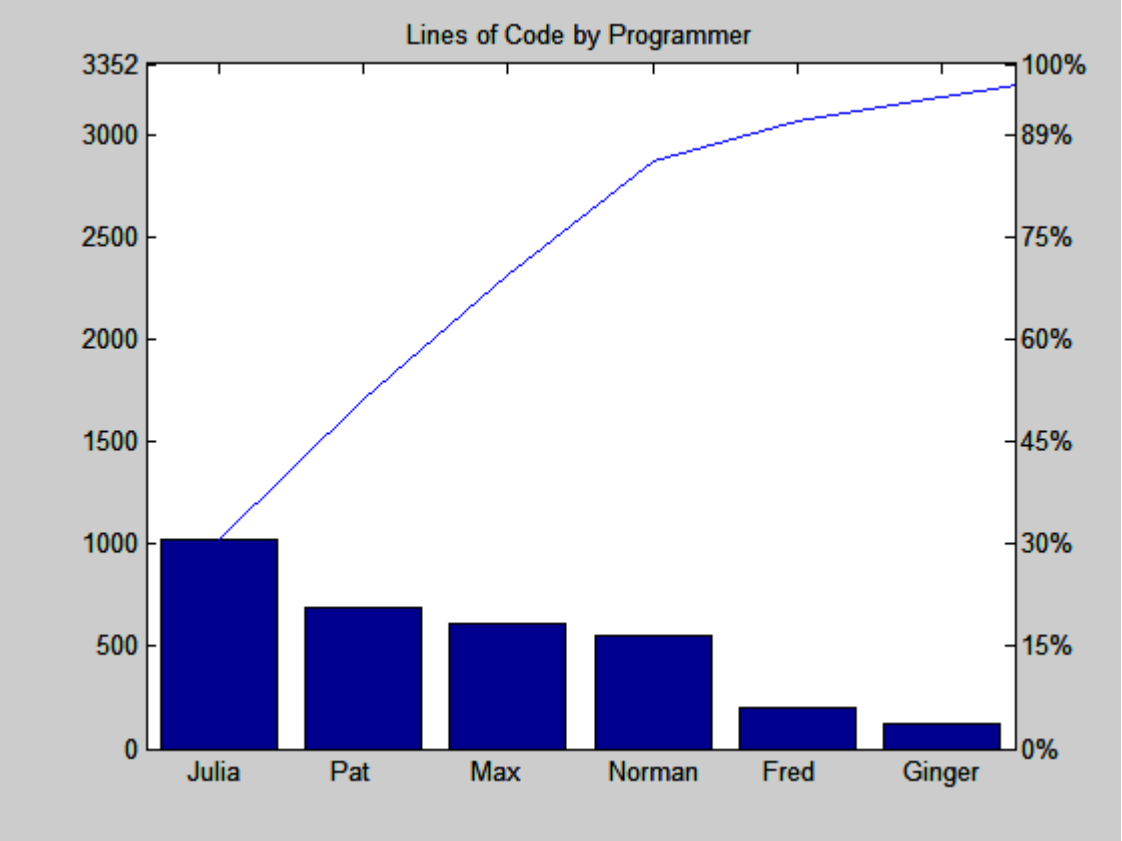
Description

Produce a Pareto chart.

Script

```
codelines = [200 120 555 608 1024 101 57 687];
coders = {'Fred','Ginger','Norman','Max','Julia','Wally','Heidi','Pat'};
pareto(codelines, coders)
title('Lines of Code by Programmer')
```

Output



Example 6

Description

Evaluation of vector dot, cross products and matrix multiplication. Load the files MyRandA.dat and MyRandB.dat, which contain data for variables a and b respectively to be used in the evaluation of vector cross, dot products and matrix multiplication; c= a × p, d = a • b and e = a * b.

Script

```
a = load('C:\STIMATH\Data Files\MyRandA.dat', '-ascii');
b = load('C:\STIMATH\Data Files\MyRandB.dat', '-ascii');
c=cross(a,b);d=dot(a,b);e=a*b';
save('C:\STIMATH\Output\CrossC.dat','c','-ascii')
save('C:\STIMATH\Output\DotD.dat','d','-ascii')
save('C:\STIMATH\Output\MultpE.dat','e','-ascii')
```

Output

Results of vector and matrix operations are output to text files CrossC.dat, DotD.dat and MultpE.dat

Example 7

Description

Compute the Kronecker tensor product of X and Y. The result is a large array formed by taking all possible products between the elements of X and those of Y. If X is m-by-n and Y is p-by-q, then kron(X,Y) is m*p-by-n*q.

Script

```
n=3;I=speye(n,n);
E=sparse(2:n,1:n-1,1,n,n);
D=E+E'-2*I;
A=kron(D,I)+kron(I,D);
spy(A)
Q=full(A);
save('C:\STIMATH\Output\KronDelta.dat','Q','-ascii')
```

Output

Result of Kronecker tensor product is output to text file KronDelta.dat

Example 8

Description

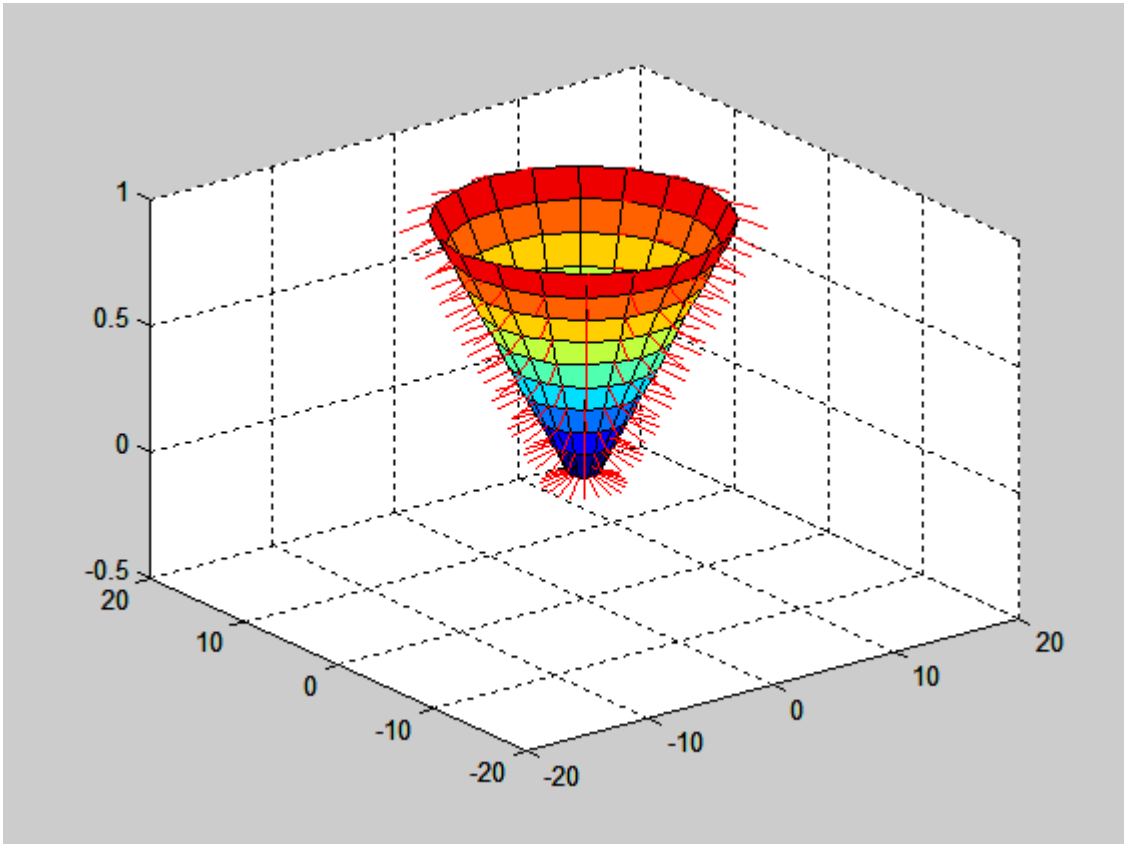
Compute and display 3-Dsurface normals for an arbitrary shape. Example uses surfnorm(X,Y,Z), which plots a surface and its surface normals from the vectors or matrices X, Y and matrix Z. X, Y and Z must be the same size. Z is a 2–D array of real numbers representing a surface. X is 2–D array of real numbers that defines the x component of the surface grid. Y is 2–D array of real numbers that defines the y component of the surface grid.

Script

```
[x,y,z] = cylinder(1:10);
[nx, ny, nz]=surfnorm(x,y,z);
surfnorm(x,y,z)
save('C:\STIMATH\Output\nx.dat','nx','-ascii')
save('C:\STIMATH\Output\ny.dat','ny','-ascii')
save('C:\STIMATH\Output\nz.dat','nz','-ascii')
```

Output

Files nx.dat, ny.dat and nz.dat contain the 3D surface normals to object shown below.



Example 9

Description

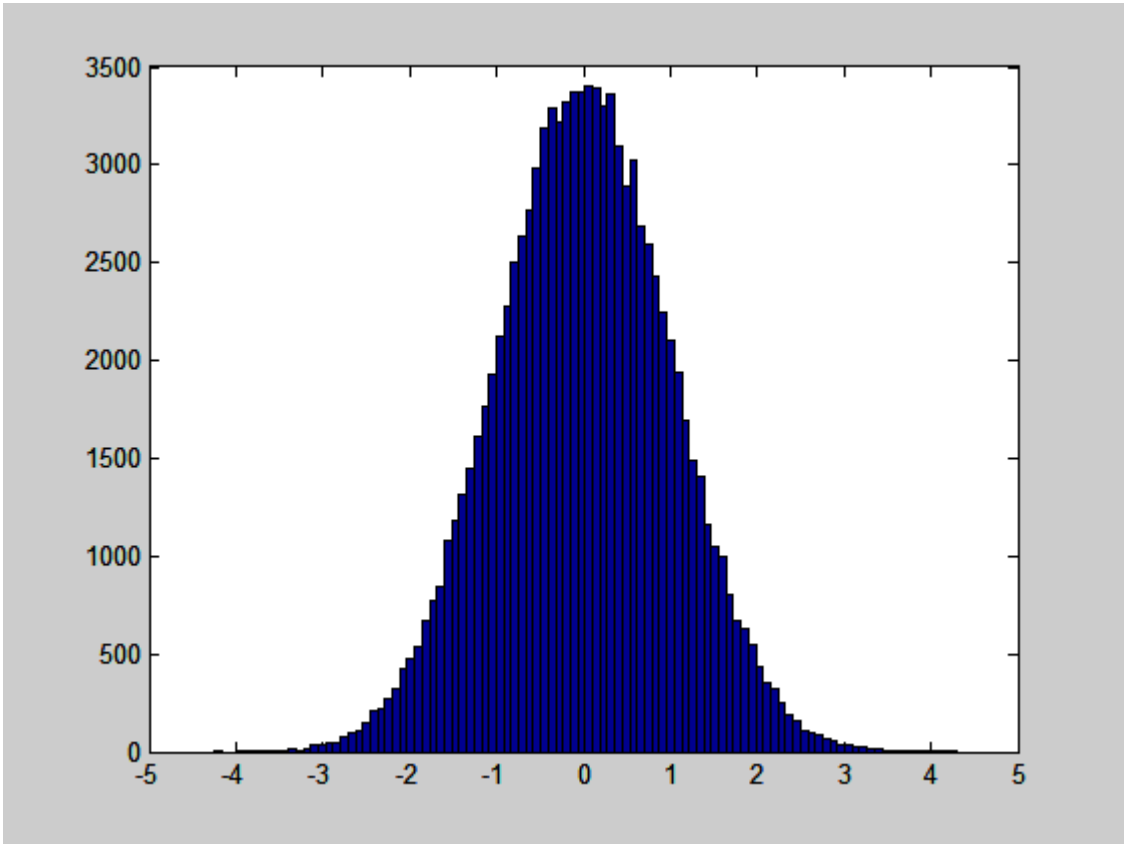
Sample 100,000 points from a normally distributed random variable and plot a histogram with 100 equally spaced bins. Save the number of elements in each bin and bin center to a text file.

Script

```
RandNSamps=randn(1,1e5)';
[nelements,xcenters]=hist(RandNSamps,100);
hist(RandNSamps,100);
dlmwrite('C:\STIMATH\Output\RandHist.dat',[nelements'
xcenters'],'delimiter',',','precision','%10.5f')
save('C:\STIMATH\Output\RandNVar.dat','RandNSamps','-ascii')
```

Output

Files RandHist.dat contains the number of elements at centers specified by xcenters. File RandVar.dat contains 100,000 randomly sampled elements from a normal distribution.



Example 10

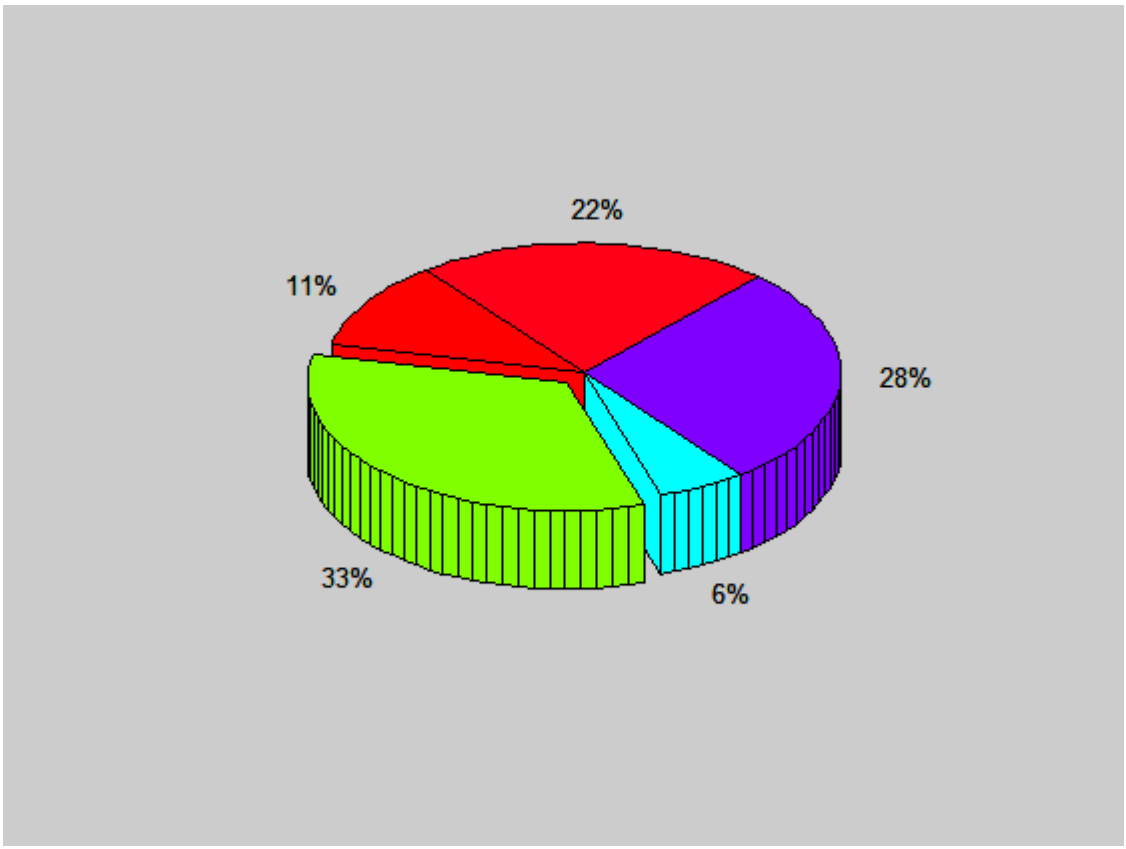
Description

Produce a 3D pie chart.

Script

```
x = [1 3 0.5 2.5 2];
explode = [0 1 0 0 0];
pie3(x,explode)
colormap hsv
```

Output



Example 11

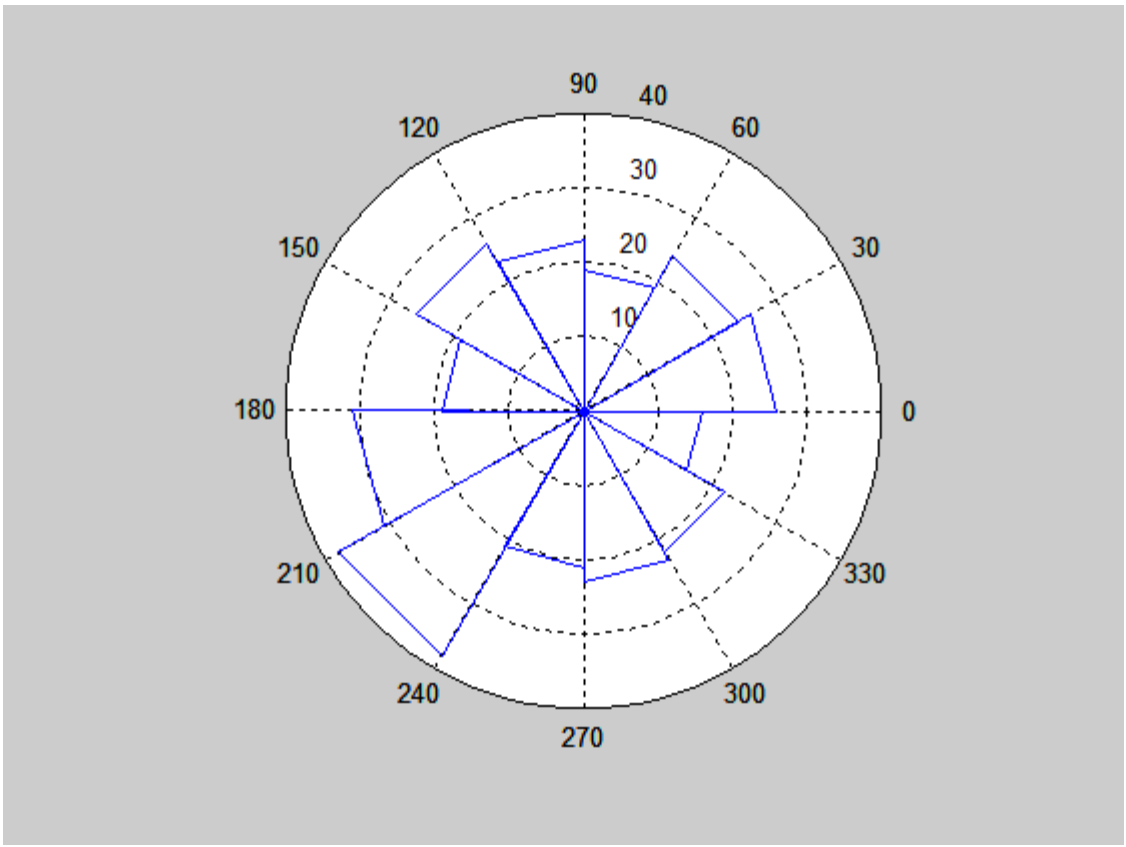
Description

Produce an angle histogram (Rose) plot.

Script

```
sunspot=load('C:\STIMATH\Data Files\sunspot.dat','-ascii');
rose(sunspot(:,2),12)
```

Output



Example 12

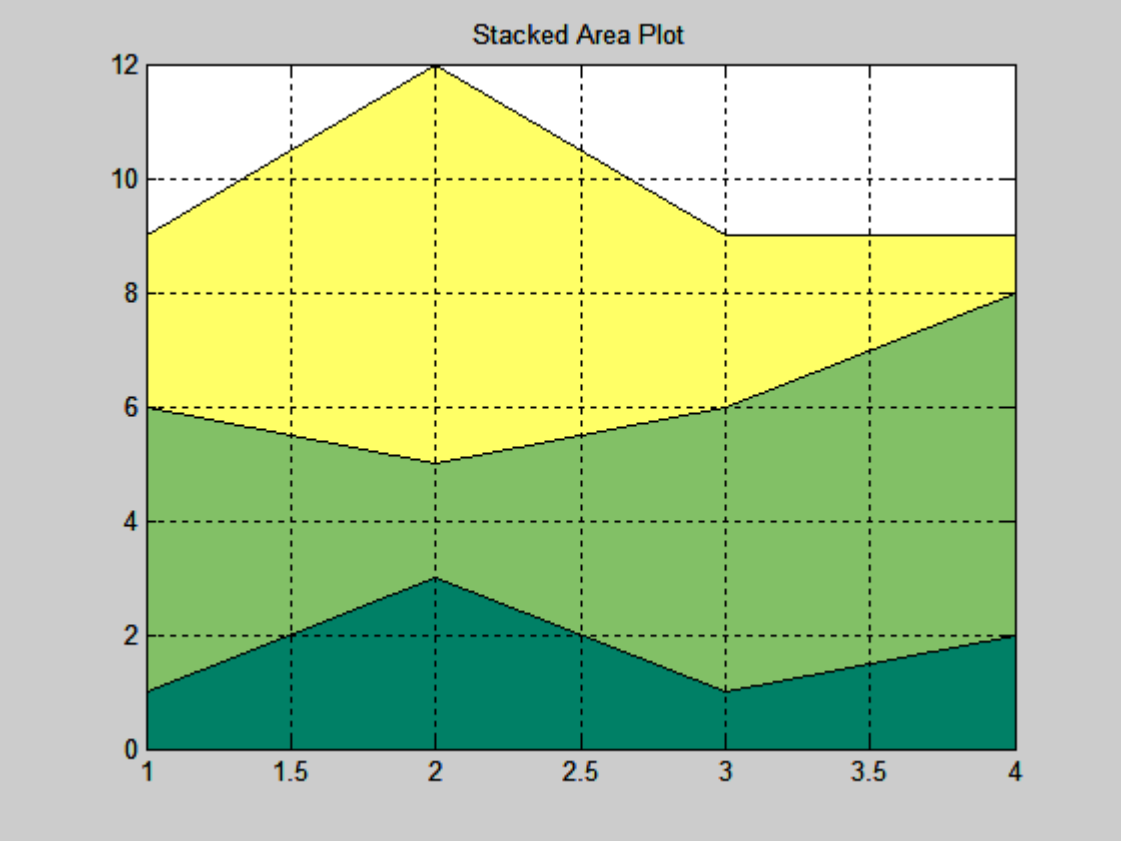
Description

Produce a stacked area plot.

Script

```
Y = [1, 5, 3;3, 2, 7;1, 5, 3;2, 6, 1];
area(Y)
grid on
colormap summer
set(gca,'Layer','top')
title('Stacked Area Plot')
```

Output



Example 13

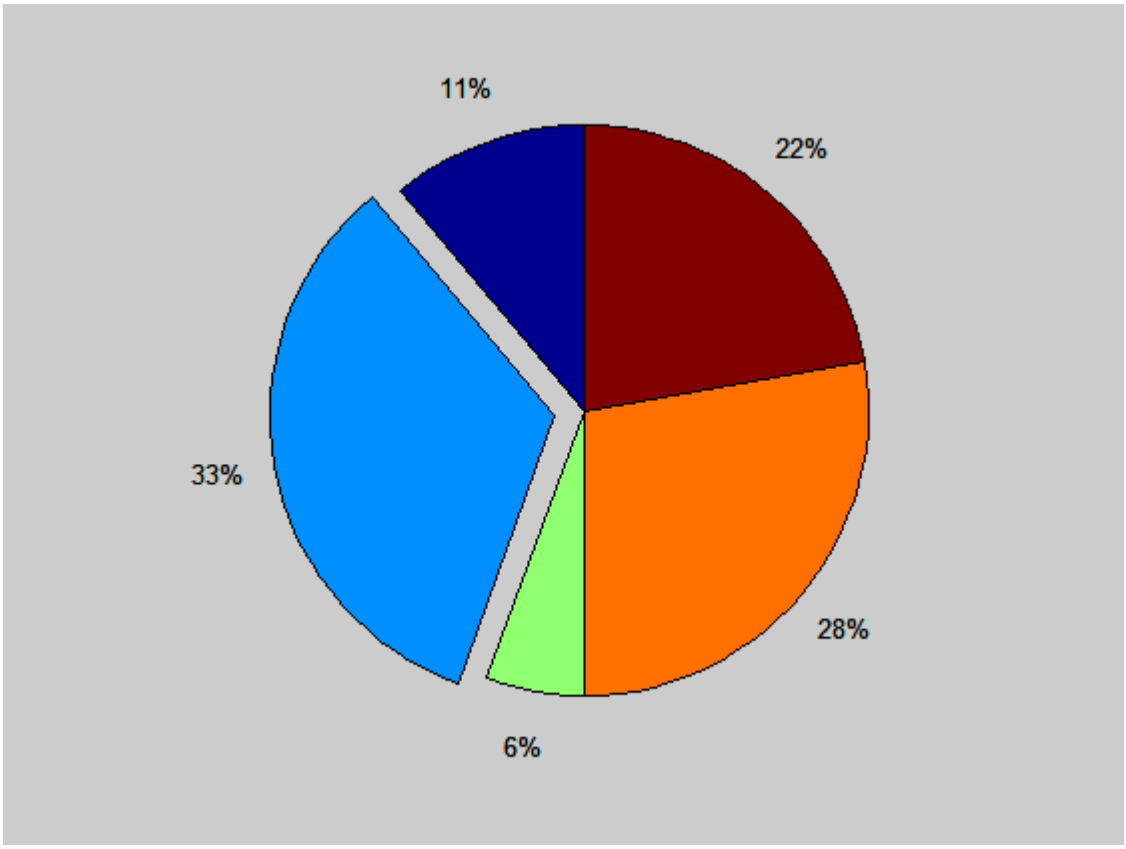
Description

Produce a pie chart.

Script

```
x = [1 3 0.5 2.5 2];
explode = [0 1 0 0 0];
pie(x,explode)
colormap jet
```

Output



Example 14

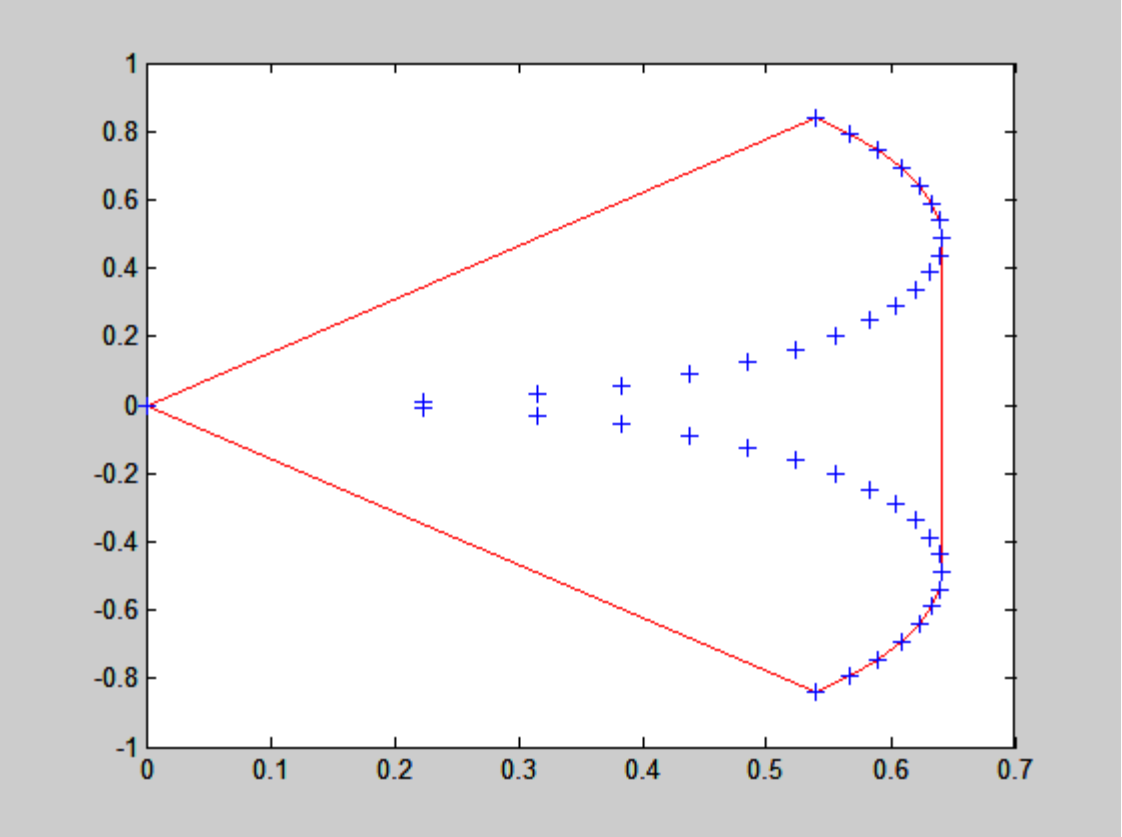
Description

Produce a plot of the 2-D convex hull of the points (X,Y),where X and Y are column vectors. The convex hull K is expressed in terms of a vector of point indices arranged in a counter clockwise cycle around the hull.

Script

```
xx = -1:.05:1;
yy = abs(sqrt(xx));
[x,y] = pol2cart(xx,yy);
k = convhull(x,y);
plot(x(k),y(k),'r-',x,y,'b+')
```

Output



Example 15

Description

x = A\B solves the system of linear equations A*x = B. The matrices A and B must have the same number of rows.

Script

```
A = rand(3);
B = [15; 15; 15];
x = A\B;
save('C:\STIMATH\Output\MatInvx.dat','x','-ascii')
```

Output

Result of A\B is saved to text file MatInvx.dat

Example 16

Description

Compute the matrix inverse of a square matrix x. Data for matrix to be inverted is contained in a text file.

Script

```
A=load('C:\STIMATH\Data Files\MYAInverseInputMatrix.dat','-ascii');
x=inv(A);
save('C:\STIMATH\Output\MatInversion.dat','x','-ascii')
```

Output

Result of matrix inverse is saved to text file MatInversion.dat

Example 17

Description

Compute the ordinary least squares solution to the linear system of equations X*b= y in the presence of known covariance V, where se_b is the estimated standard errors in b and mse is the mean squared error. Syntax is [b,se_b,mse]=lscov(X,y,V)

Script

```
x1=[.2 .5 .6 .8 1.0 1.1]';
x2=[.1 .3 .4 .9 1.1 1.4]';
X=[ones(size(x1)) x1 x2];
y=[.17 .26 .28 .23 .27 .34]';
V=.2*ones(length(x1)) + .8*diag(ones(size(x1)));
[b,se_b,mse]=lscov(X,y,V);
save('C:\STIMATH\Output\MyOut17.dat','b','se_b','mse','-ascii')
```

Output

Calculated values for b,se_b,mse are saved to text file MyOut17.dat.

Example 18

Description

Find the minimum of an unconstrained multivariable function using derivative-free method. In this case the Rosenbrock banana function, which is a function of 2 variables and is given by $f(x)=100(x_2-x_1^2)^2 + (1-x_1)^2$ is minimized.

Script

```
[x,fval] = fminsearch(@(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2,[-1.2, 1]);
save('C:\STIMATH\Output\MyOut18.dat','x','fval','-ascii')
```

Output

The location of the minimum and the minimum value are written to text file MyOut18.dat.

Example 19

Description

Find the minimum of a single-variable function on specified interval. In this example, the function given by $f(x) = x^3 - 2x - 5$ for $0 \leq x \leq 2$. Syntax used is [x,fval] =fminbnd(f(x),lower limit,upper limit). x is the location of the minimum value, fval is the minimum value.

Script

```
[x,fval] =fminbnd( @(x)x.^3-2*x-5,0,2);
save('C:\STIMATH\Output\MyOut19.dat','x','fval','-ascii')
```

Output

The location of the minimum, x, and the minimum value, fval, are written to text file MyOut18.dat.

Example 20

Description

Find eiginvalues (V) and eigenvectors (D) of the matrix A.

Script

```
A=[3 2 -2; -3 -1 3; 1 2 0];
[V,D]=eig(A);
DD=diag(D);
save('C:\STIMATH\Output\MyOut20.dat','V','DD','-ascii')
```

Output

Calculated eignevalues and eignevectors are saved to file MyOut20.dat.