POLAR3D: A 3-Dimensional Polar Plot Function in Matlab®

J M De Freitas
QinetiQ Ltd, Winfrith Technology Centre
Winfrith, Dorchester
Dorset DT2 8XJ
United Kingdom

2 June 2005

This technical note describes POLAR3D, a Matlab® function that allows the user to plot 3-Dimensional polar data. Occasionally one is required to plot intensity or magnitude information obtained along an arc or radial direction. Typical examples are surface profiles obtained at several radii, or radial information obtained between given angles. POLAR3D is similar to the polar plot except that it attempts to plot the intensity/magnitude information for given (ρ,θ).

Terms and Conditions of Use

1. This function is made available to Matlab® users under the terms and conditions set out in the Matlab Exchange by The Mathworks, Inc.
2. Where the use of POLAR3D is to be cited, the following is recommended:
   J M De Freitas. 'POLAR3D: A 3-Dimensional Polar Plot Function in Matlab®'. QinetiQ Ltd, Winfrith Technology Centre, Winfrith, Dorchester DT2 8XJ. UK. 2 June 2005.
3. No offer of warranty is made or implied by the author and use of this work implies that user has agreed to take full responsibility for its use.

POLAR3D

POLAR3D is a Matlab® function that allows the user to plot 3-Dimensional polar data. The function was designed to plot 3D intensity data over a disc or on a sector of a disc between an inner and outer radius. Moreover, POLAR3D was designed to be simple, and one of the best ways of doing this is to make the plot data available as an output, to cater for a wide range of users with different plot needs.

The syntax of the function is as follows:

```
polar3d(Zin, theta_min, theta_max, Rho_min, Rho_max, meshscale)
polar3d(Zin,theta_min,theta_max, Rho_min, Rho_max, meshscale,plotspec)
polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max, meshscale, interpspec)
polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale, plotspec, interpspec)
[Xout,Yout,Zout] = polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,...)
```

Polar3D(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale) produces a mesh plot of the input data between theta_min and theta_max, radii Rho_min and Rho_max.

Note that when this call is made, the default interpolation over the input data is 'linear' and the default plot type is 'mesh'.

Polar3D(Zin,...,*plotspec*) plots the data as 'surf', 'surfc', 'mesh', 'meshc', 'contour', or 'contourf' as set by *plotspec*.

Note that if *plotspec* = 'off', the plot function is disengaged.

Polar3D(Zin,..., *interpspec*) plots the data as mesh plot with 'linear', 'cubic', 'nearest', or 'spline' 2D interpolation over the input data as set by *interpspec*. Refer to Matlab® documentation for further details on interpolation types.

[x,y,z] = Polar3D(Zin,...,'off') disengages the plot function and returns the (x,y,z) plot data. Note that x, y, and z are all square matrices whose dimensions depend on *meshscale* (see below).

[x,y,z] = Polar3D(Zin,..., *plotspec*, *interpspec*) carries out 2D interpolation over the input data as set by *interpspec*, plots the data according to *plotspec*, and returns the (x,y,z) plot data. Note that x, y, and z are all square matrices whose dimensions depend on *meshscale* (see below).

Polar3D(Zin,..., *meshscale, plotspec*, *interpspec*) carries out 2D interpolation over the input data as set by *interpspec*, plots the data according to *plotspec*, and changes the size of the squares on the mesh or surf plots as set by *meshscale*.

If *meshscale* is less than 1, the size of the squares on the plots is reduced, whereas if it is greater than 1, the size of the squares is enlarged. Thus for example, if *meshscale* = 0.2, then the size of the squares is reduced to 1/5th of its original size; if *meshscale* = 3.2, then it is magnified by a factor of 3.2.

Note that the dimensions of Zout will be reduced (relative to largest dimension of Zin) by the same scaling as *meshscale* if the latter is greater than 1, or scaled up if it is less than 1.

Polar3D as an interpolation tool

Polar3D can be used purely as an interpolation tool over the Cartesian space i.e. without reference to the polar format. Refer to the **Examples** section of this document for further details.

Additional Notes

Zin is the input magnitude profiles where each column in Zin is assumed to represent the radial information in the plot i.e. each column represents the information along a radial line defined by theta, where theta varies between theta_min and theta_max.

Zin is a (M x N) matrix, where M and N are not necessarily equal. If M is not equal to N then the data are interpolated to make them equal. The final size is determined by the larger value of (M,N).

The N columns of Zin are assumed to be equally spaced measurements of the radial components, where Zin(1,1) corresponds to (theta_min,Rho_max), Zin(M,1) corresponds to (theta_min,Rho_min), and so on. Zin(1,N) corresponds to (theta_max,Rho_max) and Zin(M,N) corresponds to (theta_max,Rho_min). Theta increases in the anticlockwise direction on the plot. The relationship between Zin, Rho and theta is shown in Table 1. Note that no one dimension of Zin should be less than 3, or if Zin is square it should not be smaller than a 5 x 5 matrix.

The plot function when invoked in Polar3D will cause Zout to be plotted in an anticlockwise direction as shown in Figure 1.

Table 1. Relationship between ρ, θ and Zin

Increasing θ

Increasing ρ

	θ_{min}	θ_{min+1}			θ_{max}
ρ_{max}	`Zin(1,1)`	`Zin(1,2)`	...	...	`Zin(1,N)`
ρ_{max-1}	`Zin(2,1)`	`Zin(2,2)`			`Zin(2,N)`
	•	•			•
	•	•	...	...	•
	•	•			•
ρ_{min+1}	`Zin(M-1,1)`	`Zin(M-1,2)`			`Zin(M-1,N)`
ρ_{min}	`Zin(M,1)`	`Zin(M,2)`			`Zin(M,N)`

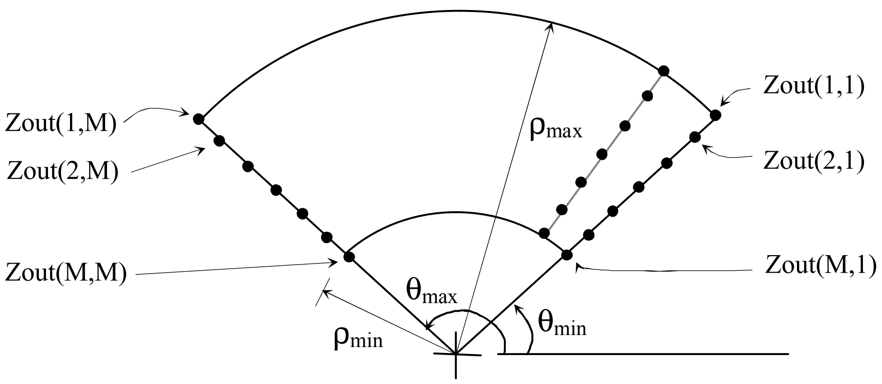
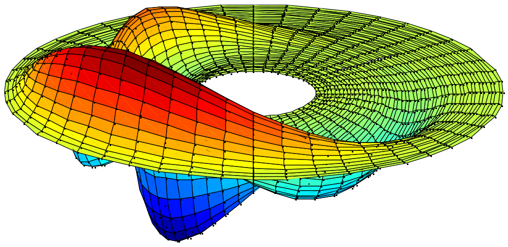
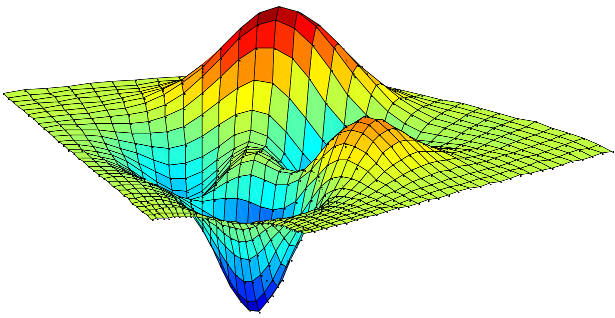


Figure 1

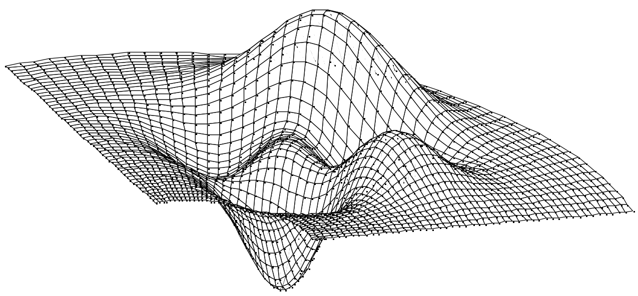
Examples



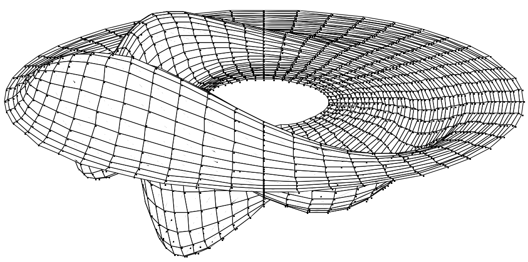
```
P = peaks(49);
Polar3d(-P,0,2*pi,100,400,1,'surf');        %change sign of P for
                                             %demonstration purposes
```



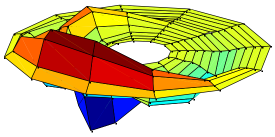
```
P = peaks(25);
Polar3d(-P,0,75*pi/180,100,400,1.54,'surf'); %change sign of P for
                                              %demonstration
                                              %purposes
```



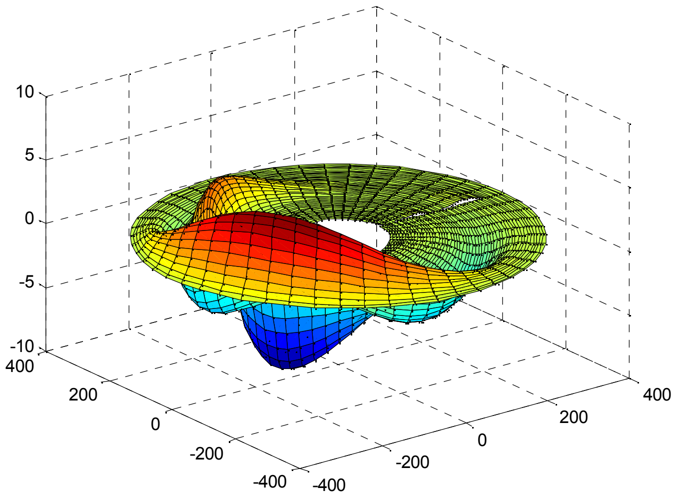
```
P = peaks(49);
Polar3d(-P,0,100*pi/180,100,400,1);          %change sign of P for
                                             %demonstration purposes
```



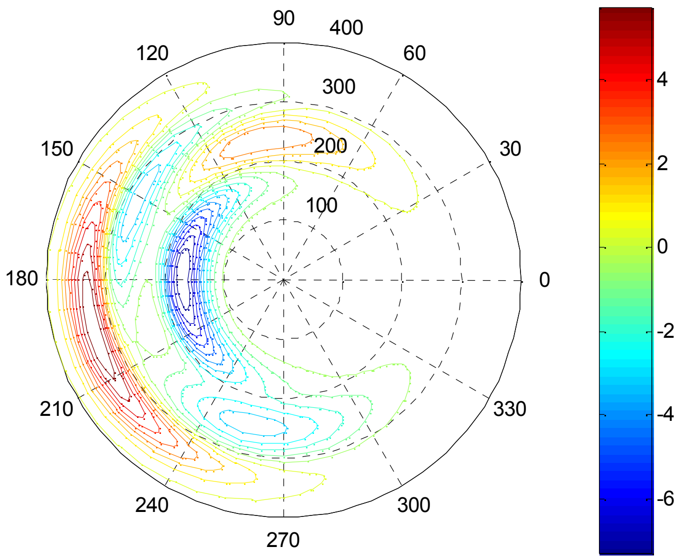
```
P = peaks(49);
Polar3d(-P,0,2*pi,100,400,1);                %change sign of P for
                                             %demonstration purposes
```



peaks(13)

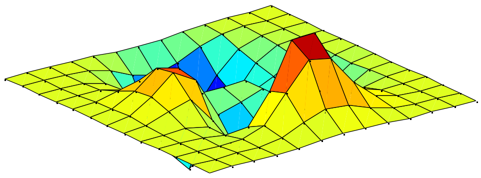



```
P = peaks(13);
[x,y,z] = Polar3d(-P,0,2*pi,100,400,0.25,'off','spline');
surf(x,y,z)
```

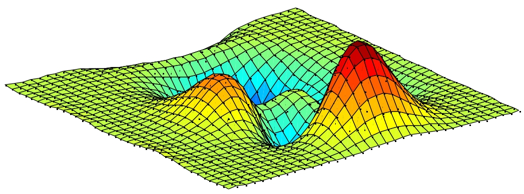



```
P = peaks(49);
Polar3d(-P,0,2*pi,100,400,1,'contour');
```

Polar3d as an interpolation tool



```
P = peaks(13);
surf(-P); axis off; grid off;
```



```
P = peaks(13);
[x,y,z] = Polar3d(-P,0,2*pi,100,400,0.36,'off','spline');
surf(fliplr(z)');
axis off; grid off;
```

Polar3D.m

```
function [Xout, Yout, Zout] = polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale,varargin)
%
% POLAR3D   Plots a 3D polar surface.
%
%           POLAR3D(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale) plots
%           the profiles as a mesh plot for Zin between radii Rho_min and
%           Rho_max for polar angles equally spaced between theta_min and theta_max.
%
%           POLAR3D(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale,plotspec)
%           plots the profiles Zin between radii Rho_min and Rho_max for
%           polar angles between theta_min and theta_max with a plot type
%           specification. If plotspec = 'surf' a standard Matlab surface
%           is plotted,whereas 'mesh', 'surfc' or 'meshc' will plot mesh,
%           surface with countour, or mesh with contour, respectively.
%           The size of the squares on the mesh or surf plots is determined
%           by meshscale. The default plot is a mesh plot.
%
%
%           [Xout,Yout,Zout] = POLAR3D(Zin,theta_min,theta_max,Rho_min,
%           Rho_max, meshscale)returns Zout values corresponding to the
%           Cartesian positions (Xout,Yout).
%
% SYNTAX    polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale)
%           polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale,plotspec)
%           polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale,interpspec)
%           polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,meshscale, plotspec,interpspec)
%           [Xout,Yout,Zout] = polar3d(Zin,theta_min,theta_max,Rho_min,Rho_max,...)
%
% INPUT     Zin         input magnitude profiles where each column in Zin is
%                       assumed to represent the radial information in the
%                       plot i.e. each column represents the information
%                       along a radial line defined by theta, where theta
%                       varies between theta_min and theta_max.
%
%                       Zin is a (M x N) matrix, where M and N are not
%                       necessarily equal. If M is not equal to N then the
%                       data are interpolated to make them equal. The final
%                       size is determined by the larger value of (M,N).
%
%                       The N columns of Zin are assumed to be equally
%                       spaced measurements of the radial components, where
%                       Zin(1,1) corresponds to (theta_min,Rho_max) and
%                       Zin(M,1) corresponds to (theta_min,Rho_min), and so on.
%                       Zin(1,N) corresponds to (theta_max,Rho_max) and
%                       Zin(M,N) corresponds to (theta_max,Rho_min). Theta
%                       increases in the anticlockwise direction.
%
%           theta_min   the lower value in radians of the angular range
%                       over which the data is defined. Theta_min is a
%                       scalar quantity.
%
%           theta_max   the upper value in radians of the angular range
%                       over which the data is defined. Theta_max is a
%                       scalar quantity.
%
%           Rho_min     the lower value of the radial component for which
%                       the data is defined. Rho_min is a scalar quantity.
%
%           Rho_max     the upper value of the radial component for which
%                       the data is defined. Rho_max is a scalar quantity.
%
%           meshscale   a scalar that determines the size of the squares
%                       on the mesh or surf plots. If meshscale is 1, the
%                       mesh remains unchanged relative to the input grid;
%                       if meshscale is 2, the size of the squares is doubled,
%                       if 3.2, say, it is scaled accordingly. Moreover, if
%                       meshscale is less than 1, e.g. 0.2, the size of the
%                       squares is reduced into a finer mesh. The dimensions
%                       of Xout, Yout and Zout are reduced or enlarged by
%                       meshscale.
%
%           plotspec    = 'surf'  produces a surface plot.
%                       = 'surfc'  produces a surface plot with contour.
%                       = 'mesh'  produces a mesh plot.
%                       = 'meshc'  produces a mesh plot with countour.
%                       = 'contour'  produces a 2D contour plot.
%                       = 'contourf'  produces a 2D filled contour plot.
%                       = 'off' disengages plot function.
%
%           interpspec  = 'linear'  bilinear interpolation on Zin.
%                       = 'spline'  spline interpolation on Zin.
%                       = 'nearest'  nearest neighbour interpolation on Zin.
%                       = 'cubic'  bicubic interpolation on Zin.
%
```

```
% OUTPUT    Zout        output magnitude profiles defined by Zin at
%                       positions (Xout,Yout).
%
%                       Zout is square with dimensions determined by the
%                       maximum dimension of the input matrix Zin. The
%                       dimensions of Zout are reduced or enlarged by meshscale.
%
%           Xout        output X-positions corresponding to polar positions
%                       (rho,theta). Xout is square with dimensions
%                       determined by the maximum dimension of the input
%                       matrix Zin. The dimensions of Xout are reduced or
%                       enlarged by meshscale.
%
%           Yout        output Y-positions corresponding to polar positions
%                       (rho,theta). Yout is square with dimensions
%                       determined by the maximum dimension of the input
%                       matrix Zin. The dimensions of Yout are reduced or
%                       enlarged by meshscale.
%

%           Written by JM DeFreitas, QinetiQ Ltd, Winfrith Technology
%           Centre, Dorchester DT2 8XJ, UK. jdefreitas@qinetiq.com.
%
%           Revision 1.0 4 April 2005.
%           Released 5 April 2005. (Beta Release).
%
%           Revision 1.1 17 April 2005
%           1. Inroduced new check on Zin to cover case of dimensions (M x 2)
%              or (2 x N) matrix. These are not allowed.
%           2. Changed L2 so that when meshscale > 1 and theta ranges over
%              360 degrees the data wraps around without spaces.
%           3. Removed Xout1, Yout1 and Zout1.
%           4. Changed 'theta(j,:) = ones([(L2/n) 1])*angl(j);' to
%              'theta(j,:) = ones([1 (L2/n)])*angl(j)'; so that it is
%              compatible with Matlab6 R12.1.
%           5. Reorganised meshgrid so that interpolation now works with
%              meshscale < 1.
%           6. Changed error traps from '((p ~= 1)&(q ~= 1))' to
%              '((p ~= 1)|(q ~= 1))' where used.
%
%           Full Release 26 May 2005. All Rights Reserved.
%
%           Terms and Conditions of Use
%
%           1.  This function is made available to Matlab® users under the
%               terms and conditions set out in the Matlab Exchange by
%               The Mathworks, Inc.
%           2.  Where the use of POLAR3D is to be cited, the following is recommended:
%               J M De Freitas. 'POLAR3D: A 3-Dimensional Polar Plot Function
%               in Matlab®'. QinetiQ Ltd, Winfrith Technology Centre, Winfrith,
%               Dorchester DT2 8XJ. UK. 2 June 2005.
%           3.  No offer of warranty is made or implied by the author and
%               use of this work implies that the user has agreed to take full
%               responsibility for its use.
%

if (nargin < 6)
    disp('Polar3d Error: Too few input arguments.');
    return
elseif (nargin > 8)
    disp('Polar3d Error: Too many input arguments.');
    return
end

[p,q] = size(theta_min);
if (((p ~= 1)|(q ~= 1))|~isreal(theta_min))|ischar(theta_min)
    disp('Polar3d Error: theta_min must be scalar and real.');
    return
end
[p,q] = size(theta_max);
if (((p ~= 1)|(q ~= 1))|~isreal(theta_max))|ischar(theta_max)
    disp('Polar3d Error: theta_max must be scalar and real.');
    return
end
if theta_max <= theta_min
    disp('Polar3d Error: theta_max less than or equal theta_min.');
    return
end
if abs(theta_max - theta_min) > 2*pi
    disp('Polar3d Error: range of theta greater than 2pi.');
    return
end

[p,q] = size(Rho_max);
if (((p ~= 1)|(q ~= 1))|~isreal(Rho_max))|(ischar(Rho_max)|Rho_max < 0)
    disp('Polar3d Error: Rho_max must be scalar, positive and real.');
    return
end
```

```
[p,q] = size(Rho_min);
if (((p ~= 1)|(q ~= 1))|~isreal(Rho_min))|(ischar(Rho_min)|Rho_min < 0)
    disp('Polar3d Error: Rho_min must be scalar, positive and real.');
    return
end
if Rho_max <= Rho_min
    disp('Polar3d Error: Rho_max less than or equal Rho_min.');
    return
end

[p,q] = size(meshscale);
if (((p ~= 1)|(q ~= 1))|~isreal(meshscale))|ischar(meshscale)
    disp('Polar3d Warning: mesh scale must be scalar and real.');
    meshscale = 1;
end
if (meshscale <= 0)
    disp('Polar3d Warning: mesh scale must be scalar and positive.');
    meshscale = 1;
end

% Set up default plot and interpolation specifications.
str1 = 'mesh';
str2 = 'linear';
if length(varargin) == 2
% Sort out plot and interpolation specification if both strings given.
    str1 = [varargin{1}(:)]';
    str2 = [varargin{2}(:)]';
    g1 = (~isequal(str1,'mesh')&~isequal(str1,'surf'))&~isequal(str1,'off');
    g2 = (~isequal(str1,'meshc')&~isequal(str1,'surfc'));
    g5 = (~isequal(str1,'contour')&~isequal(str1,'contourf'));
    g3 = (~isequal(str2,'cubic')&~isequal(str2,'linear'));
    g4 = (~isequal(str2,'spline')&~isequal(str2,'nearest'));
    if (g1&g2&g5)
        disp('Polar3d Warning: Incorrect plot specification. Default to mesh plot.');
        str1 = 'mesh';
    end
    if (g3&g4)
        disp('Polar3d Warning: Incorrect interpolation specification.');
        disp('Default to linear interpolation.');
        str2 = 'linear';
    end
elseif length(varargin) == 1
% Sort out plot or interpolation specification from single string input.
    str1 = [varargin{1}(:)]';
    g1 = (~isequal(str1,'mesh')&~isequal(str1,'surf'))&~isequal(str1,'off');
    g2 = (~isequal(str1,'meshc')&~isequal(str1,'surfc'));
    g5 = (~isequal(str1,'contour')&~isequal(str1,'contourf'));
    g3 = (~isequal(str1,'cubic')&~isequal(str1,'linear'));
    g4 = (~isequal(str1,'spline')&~isequal(str1,'nearest'));
    if (g1&g2)&(g3&g4&g5)
        disp('Polar3d Error: Incorrect plot or interpolation specification.');
        return
    elseif isequal(str1,'cubic')
        str2 = str1;
        str1 = 'mesh';
    elseif isequal(str1,'linear')
        str2 = str1;
        str1 = 'mesh';
    elseif isequal(str1,'spline')
        str2 = str1;
        str1 = 'mesh';
    elseif isequal(str1,'nearest')
        str2 = str1;
        str1 = 'mesh';
    elseif isequal(str1,'off')
        str2 = 'linear';
    end
end;

% Check if dimensions of input data are acceptable.
[r,c] = size(Zin');
if (r < 5)&(c < 5)
    disp('Polar3d Error: Input matrix dimensions must be greater than (4 x 4).');
    return
end
% Check if input data has two rows or columns or less.
if (r < 3)|(c < 3)
    disp('Polar3d Error: One or more input matrix dimensions too small.');
    return
end

% Transpose and setup input magnitude matrix.
temp = Zin';
for j = 1:c
    P(:,j) = temp(:,c-j+1); % swap columns over
end
Zin = P;
[r,c] = size(Zin);
```

```
% Check if meshscale is compatible with dimensions of input data.
scalefactor = round(max(r,c)/meshscale);
if scalefactor < 3
    disp('Polar3d Error: mesh scale incompatible with dimensions of input data.');
    return
end

% Set up meshgrid corresponding to larger matrix dimension of Zin
% for interpolation if required.
n = meshscale;
if r > c
    L = r;
    L2 = fix(L/n)*n;
    step = r/(c-1);
    [X1,Y1] = meshgrid(0:step:r,1:r);
    if n < 1
        [X,Y] = meshgrid(0:n:(L2-n),0:n:(L2-n));
    else
        [X,Y] = meshgrid(1:n:L2,1:n:L2);
    end
    T = interp2(X1,Y1,Zin,X,Y,str2);
elseif c > r
    L = c;
    L2 = fix(L/n)*n;
    step = c/(r-1);
    [X1,Y1] = meshgrid(1:c,0:step:c);
    if n < 1
        [X,Y] = meshgrid(0:n:(L2-n),0:n:(L2-n));
    else
        [X,Y] = meshgrid(1:n:L2,1:n:L2);
    end
    T = interp2(X1,Y1,Zin,X,Y,str2);
else
    L = r;
    L2 = fix(L/n)*n;
    [X1,Y1] = meshgrid(1:r,1:r);
    if n < 1
        [X,Y] = meshgrid(0:n:(L2-n),0:n:(L2-n));
    else
        [X,Y] = meshgrid(1:n:L2,1:n:L2);
    end
    T = interp2(X1,Y1,Zin,X,Y,str2);
end

[p,q] = size(T);
L2 = max(p,q);

% Set up angles
angl = theta_min:abs(theta_max-theta_min)/(L2-1):theta_max;
for j = 1:L2
    theta(j,:) = ones([1 L2])*angl(j);
end

% Set up radial components
Rho = Rho_min:abs(Rho_max-Rho_min)/(L2-1):Rho_max;

% Convert to Cartesian coordinates
for j = 1:L2
    [Xout(j,:) Yout(j,:) Zout(j,:)] = pol2cart(theta(j,:),Rho,T(j,:));
end

% Plot Cartesian surface
switch str1;
case 'mesh'
    colormap([0 0 0]);
    mesh(Xout,Yout,Zout);
    axis off;
    grid off;
case 'meshc'
    colormap([0 0 0]);
    meshc(Xout,Yout,Zout);
    axis off;
    grid off;
case 'surf'
    surf(Xout,Yout,Zout);
    axis off;
    grid off;
case 'surfc'
    surfc(Xout,Yout,Zout);
    axis off;
    grid off;
    hold off
case 'contour'
    axis equal;
    h = polar([theta_min theta_max], [Rho_min Rho_max]);
    delete(h)
    hold on
```

```
    contour(Xout,Yout,Zout,20);
    hold off
    colorbar;
case 'contourf'
    axis equal;
    contourf(Xout,Yout,Zout,20);
    axis off;
    grid off;
    colorbar;
end

return
```