Chapter 5

Mesh Plots
Scientific Visualization & Information Architecture

5.1 Mesh Plot

Definition 12 (Mesh Plot) *A plot which depicts a function $q(x, y)$ by plotting a fishnet-shaped lattice of horizontal and vertical lines in three-dimensional space as if the net had been draped over the surface of q.*

Alternative Names: FISHNET PLOT, WIREFRAME DIAGRAM

Variants: MESH-with-SKIRT, MESH-OVER-CONTOUR, WATERFALL PLOT, TRUNCATED-MESH, CUTAWAY MESH, VARIABLE-MESH

Merits:

- Good in black-and-white;
  color helpful but unnecessary.
- More natural than contour or pseudocolor graphs;
  relief is SEEN in a mesh plot rather than
  INFERRED from color or isolines.

Flaws:

- Less precise than contour or pseudocolor plots
  in the sense that it is difficult to accurately judge shapes and sizes
  in perspective than from contour lines
- Obstructed-view:
  low-lying features at the back hidden between tall foreground features
- More difficult to effectively overlay arrows ("quiver plot") to show the magnitude and direction of a two-dimensional vector field for a mesh plot than for a contour plot or pseudocolor graph.

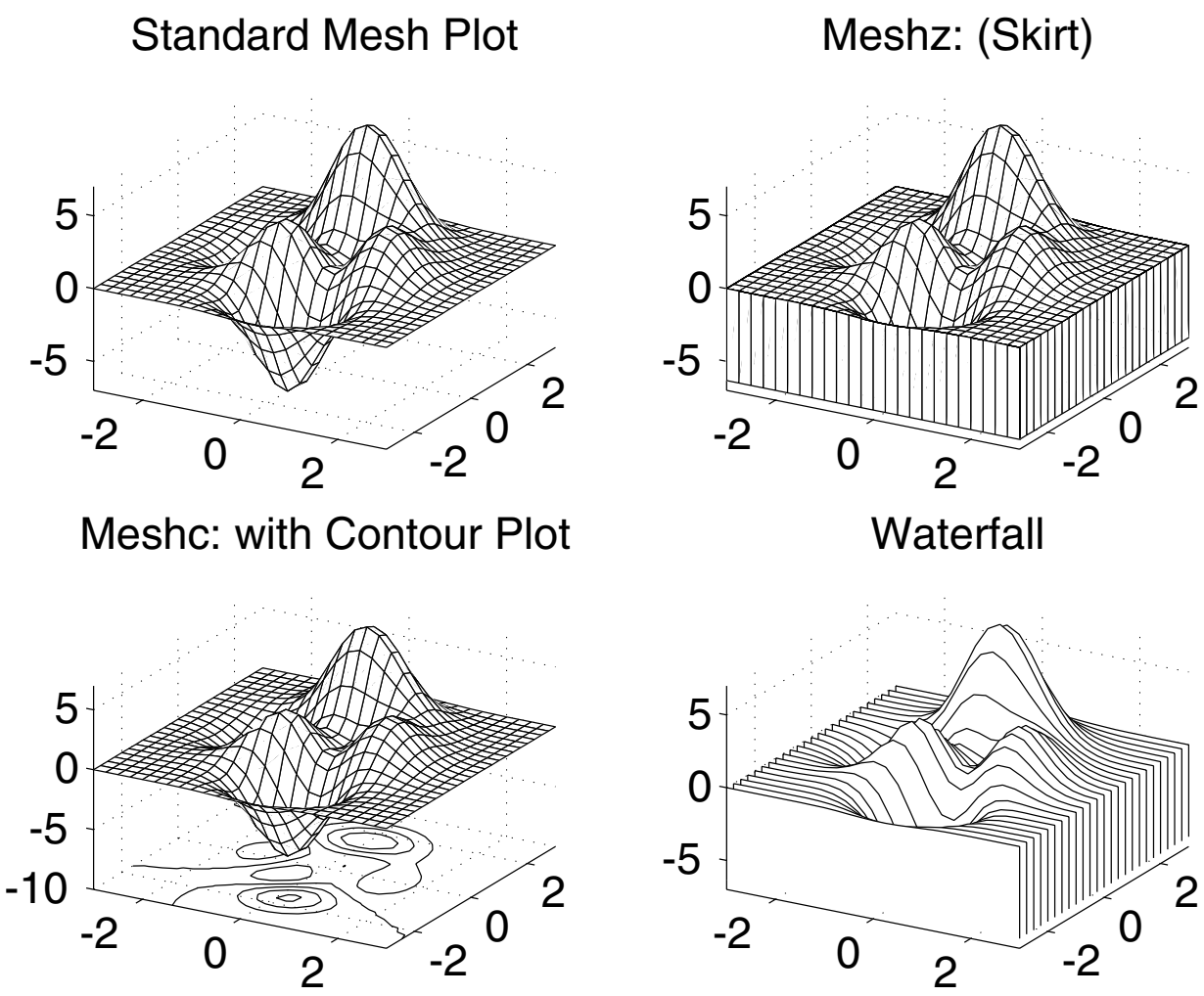

Figure 5.1: An ordinary mesh plot and three of its most popular variants. Other species of mesh plots are illustrated later.

5.2 Mesh Plot with a Curtain or Skirt

A "curtain" or "skirt" is a set of vertical lines that run from the edges of the surface to the base of a rectangle whose height is at the deepest valley of the surface.

The disadvantage of "skirting" is that additional lines are added which do not directly present data. In Tufte's lexicon, "skirting" lowers the data-ink ratio, which is usually bad.

Nevertheless, "skirting" is a built-in option in most fishnet subroutines. The reason is that a skirt provides additional visual cues that are quite helpful in interpreting a three-dimensional perspective image that is printed on a flat, two-dimensional surface.

The best advice is to plot the mesh first without the skirt. If the result is confusing, and altering the viewing angle does not eliminate the confusion, then plot the mesh with a skirt, and see if this helps. There is no hard-and-fast rule about "skirting"; it is a matter of experimentation to see what looks the clearest.

5.3 Mesh-Over-Contour Plot

One deficiency of a mesh plot is that is not very exact; it gives good qualitative insights into the shape of a surface, but it is hard to judge a set of peaks and valleys and perceive the maxima and minima with any precision. (Perhaps a mountain climber or a military engineer might disagree, and with reason because of extensive personal experience in interpreting this species of diagram). The contour plot underneath allows one to have the best of both the mesh plot and the contour plot.

The first technical issue is that in the mesh-over-contour figure, the contour plot is viewed *in perspective*. The sideways, slanting viewing angle makes the contours more difficult to interpret accurately. This is compensated for by the visual association between the contours and the mesh diagram just above them. However, in some cases, it may be preferable to use a two-panel diagram in which an ordinary mesh plot is the upper diagram and a standard contour plot is the lower panel.

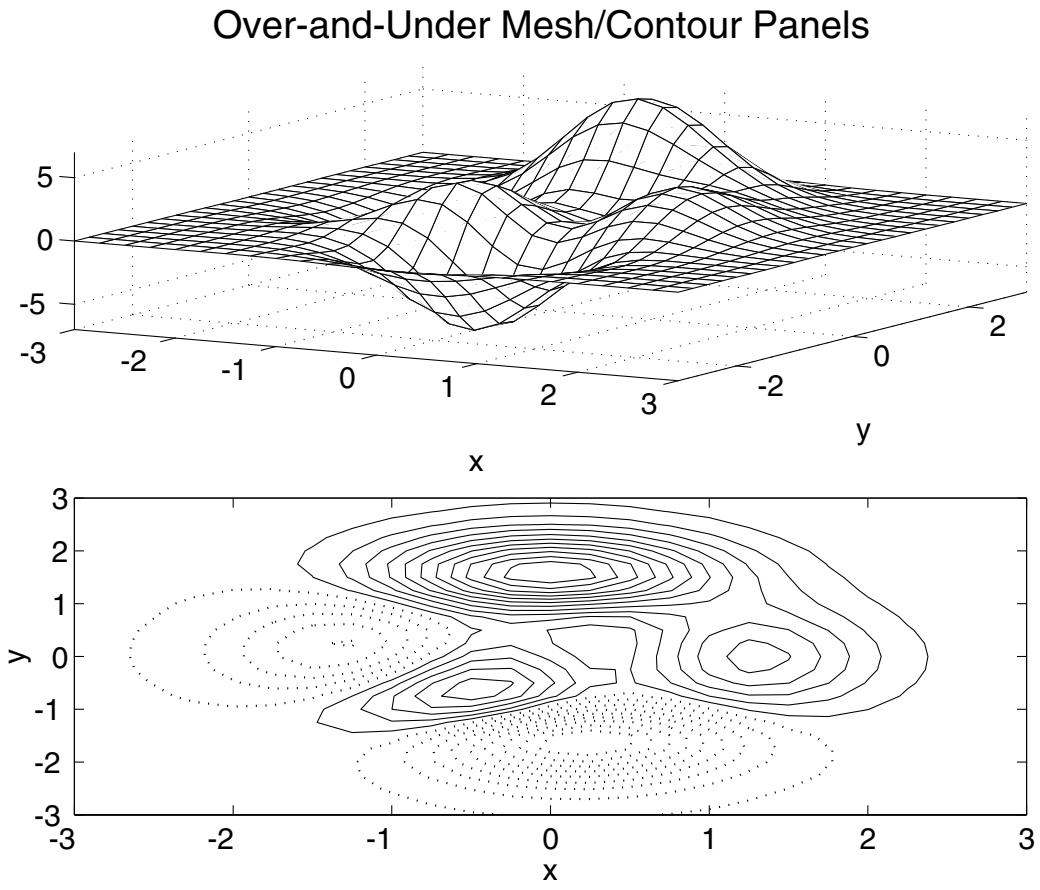

Figure 5.2: An alternative to **meshc** — mesh and contour plot overlaid in a single frame — is to use a two-panel graph with a mesh plot and a contour plot of the same function $q(x, y)$ in the top and bottom panels respectively.

The second issue is that the mesh plot often *obstructs* the contour plot beneath it. Unfortunately, the standard **meshc** command in Matlab does not allow to control the variable q_{con} is the vertical level at which the contour plot will be located; this is always set equal to the lower limit of the default vertical axis. In an appendix, we list a modified routine which allows one to specify the placement of the contour plot as a parameter. This makes it easy to create enough separation between contour plot and fishnet grid so that the isolines are all clearly visble.

Fig. 5.3 shows that increasing the vertical separation between mesh and contour plot eliminates obstruction of isolines. However, the vertical scale of the mesh plot is inevitably compressed by an increase in mesh/contour plot separation, which makes the fishnet diagram harder to interpret.

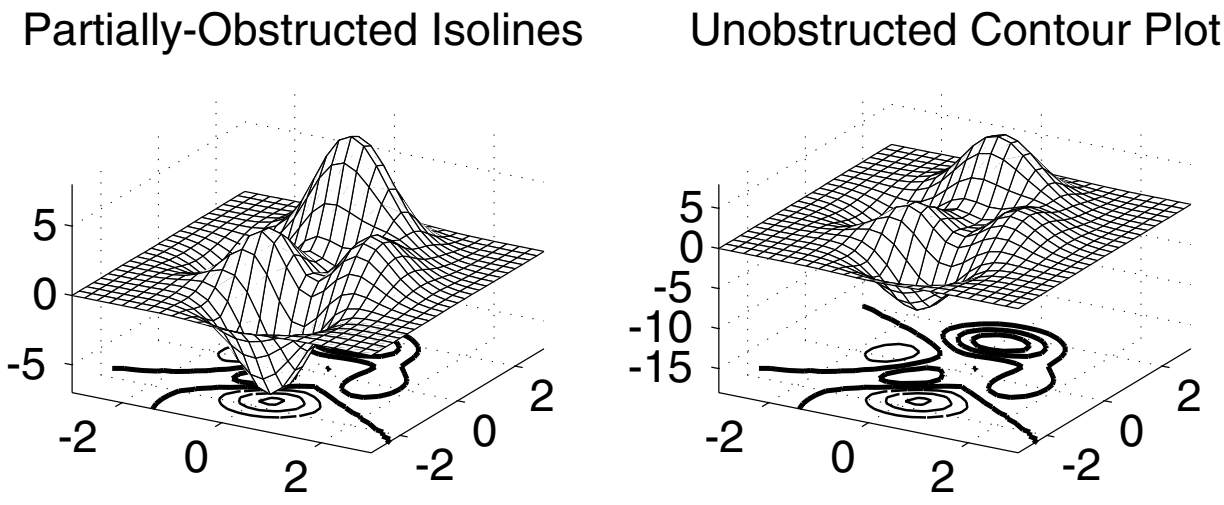

Figure 5.3: Left: Standard mesh-over-contour plot (Matlab **meshc**); the contours of the tallest peaks are completely obscured by the fishnet. Right: same except that the vertical separation has been increased so that isolines are now all visible.

A third issue is that sometimes what one really wants is to have the contours and fishnet SUPERIMPOSED. The easiest way to do this is to call **mesh**, then **hold** followed by **contour3**. A sample is shown as Superimposing all contours may be too much, but it is easy to add just one or two contours, and make them distinctive through color, linestyle or linewidth as illustrated in Fig. 5.4.

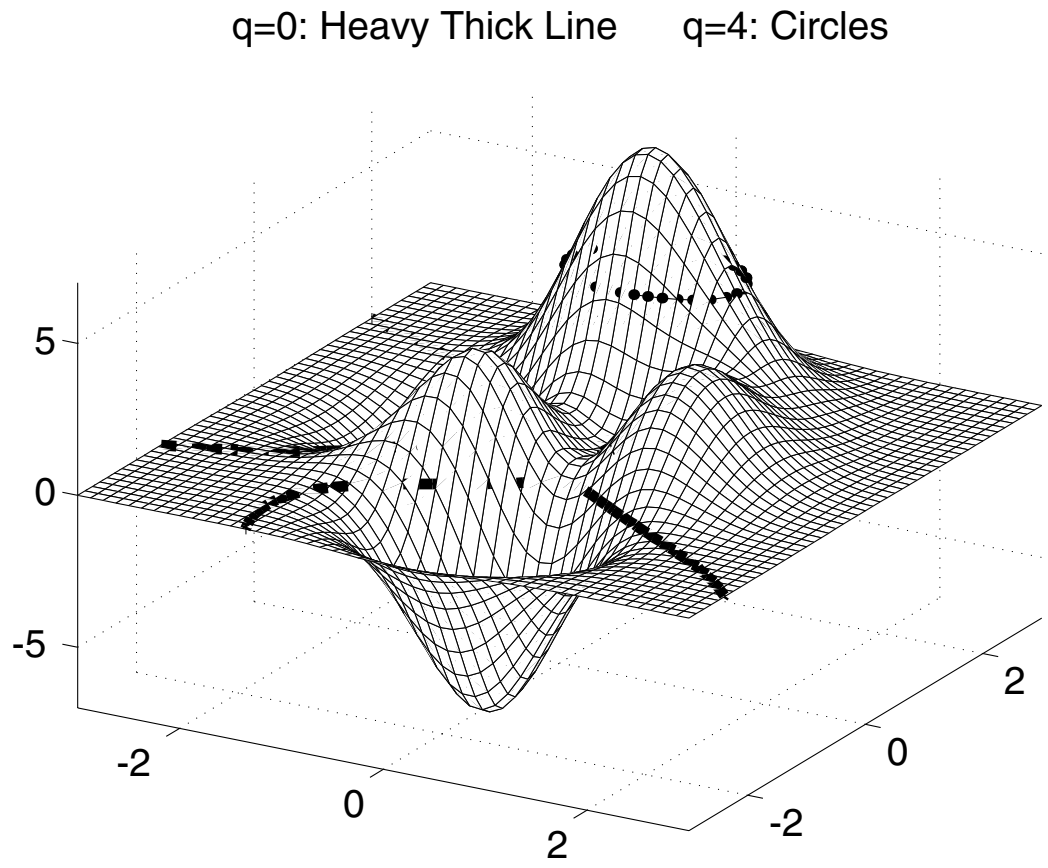

Figure 5.4: A mesh plot with two superimposed, three-dimensional contours. The thick $q = 0$ isoline was added to the mesh plot through **hold on; [cc,hh] = contour3(X,Y,Z,[0 0],'k'); set(hh,'LineStyle','-','LineWidth',5,'Color','k')** and similarly for other contours.

A fourth issue is that sometimes what one really wants is a three-dimensional contour plot, viewed at a sideways angle, stacked above a similar contour plot, not a fishnet diagram. Unfortunately, this is not a standard subroutine in most software libaries, including Matlab's, so one is forced to hack one's own (given in an appendix). An example of a two-layer, three-dimensional contour plot is Fig. 5.5.

In meteorology and physical oceangraphy, so-called "two layer" and closely related "two level" models are widely used in research. A graph with two contour plots is then a more natural representation than a mesh-over-contour figure.

Other types of diagrams can be overlaid on a mesh-over-contour or contour-over-contour diagram. For example, in fluids, it is convenient to overlay three-dimensional quiver plots, i. e., arrows representing the magnitude and direction of the horizontal velocity. The isolines are usually those of pressure in this context.

In two-layer, three-dimensional contour plots, the variables graphed in the two isoline plots are almost always distinct, such as temperature for the lower layer of fluid and temperature for the upper layer of fluid. In a mesh-over-contour diagram, the two graphs usually give *different representations* of the *same* $q(x, y)$. However, it is worth noting that, just like a two-layer contour plot, a mesh-over-contour diagram can show two different unknowns, too, such as pressure in the mesh diagram and temperature in the velocity diagram. The spatial superposition is then helpful in perceiving the spatial correlation between the two unknowns.

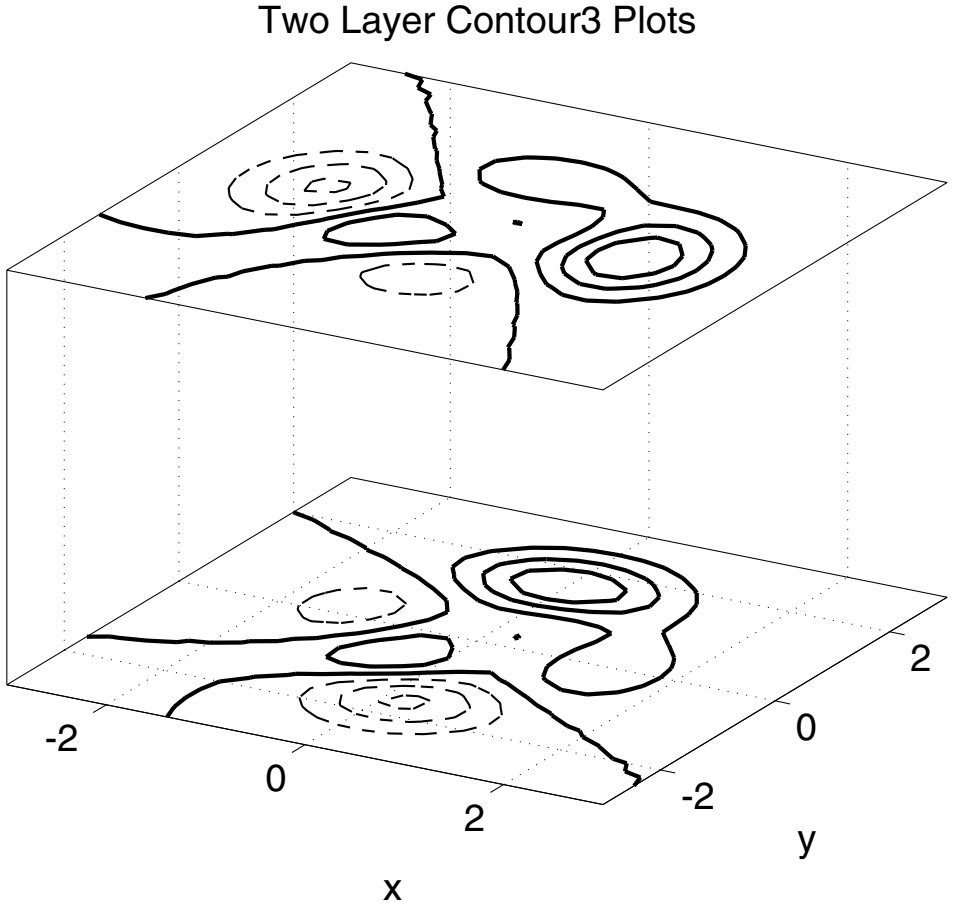

Figure 5.5: Two-layer, three-dimensional contour plot.

5.4 Waterfall Graph or Mesh-Lines-Parallel-To-One-Axis Only

A waterfall graph is identical to an ordinary mesh plot except that the surface is not depicted through a lattice of intersecting lines, but through lines parallel to the x-axis only.

Merits:

1. The plot is less cluttered and dark
2. Each horizontal line is just a line graph of the function at a particular time, which makes it easier to interpret.

Disadvantages:

1. Occasionally, the absence of vertical lines can make it harder to identify the peaks and valleys, but this is usually compensated for by the lack of clutter.

In Matlab, such plots are generated not by using any of the subroutines of the **mesh** family, but rather by using **waterfall**. This takes exactly the same arguments as **mesh**.

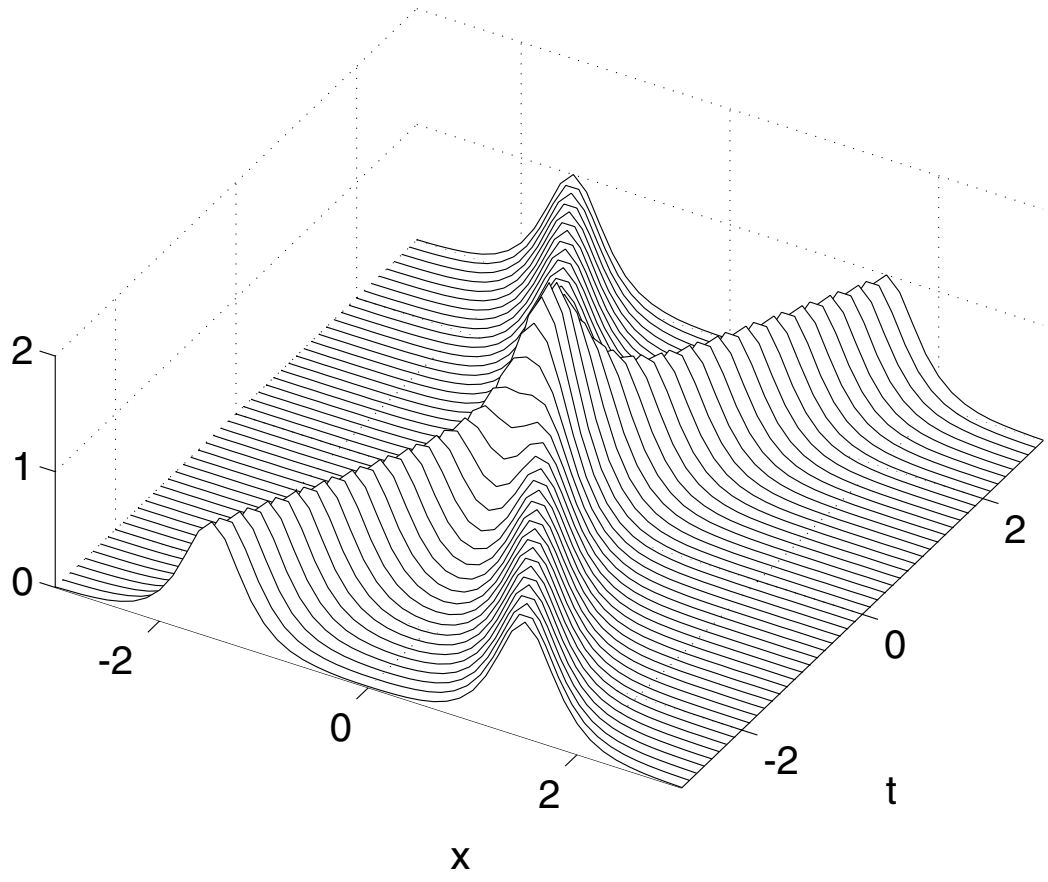

Figure 5.6: Schematic of a soliton-soliton collision, $h(x,t)$ where h is wave height. Waterfall plots are especially useful when one horizontal coordinate is time and the other is space.

One particularly appropriate use of waterfall plots is when the two horizontal coordinates have a different character — for example, when one is a SPACE coordinate and the other is TIME. Standard mesh plots (lines in parallel to both horizontal axes) are more natural when BOTH coordinates are SPATIAL.

Waterfall graphs have been widely used to illustrate the collisions of solitary waves in $x - t$ plane. One could use a set of ordinary line graphs in x to depict the wave height at different times, labelling each curve with t. However, it is impractical to superimpose more than four or five curves in a line graph. A waterfall plot allows one to superimpose fifty or a hundred snapshots of the height. Fig. 5.6 illustrates the collision of two solitary waves, shown as a waterfall.

Tactical note: with a waterfall plot, one can usually employ a higher viewing angle, i. e., look more from above at the surface, than with an ordinary mesh plot. The perceptual reasons for this are unclear, but with a waterfall graph, one should not be shy about experimenting with higher view angles (i.e., second argument of the **view** command nearer 90 than 0).

If one wishes the mesh lines to run parallel to the y-axis, rather than the x-axis, it is only necessary to call the waterfall plot with transposed arguments: **waterfall(X', Y', Z')**. The viewing angle can be rotated by 180 degrees in the horizontal plane, and the horizontal axis labels switched.

5.5 Truncated Mesh Plots

For clarity, it is often necessary to modify a mesh plot by cutting away some of its structure. One easily implemented form of cutaway is *truncation*. This can be performed by adding an if statement to test each height value. Values which are above or below user-chosen limits are then replaced by these limits. Thus, mountain-tops are flattened. [1]

Fig. 5.7 illustrates such a plot. Because functions in the complex plane often have regions of exponential growth, truncated mesh plots are very widely used in the ancient but still useful handbook of mathematical functions by Janke and Emde. In the more modern NBS Handbook, truncated mesh plots can be found on

However, the strategy of truncating a mesh plot is very general, and in no way limited to plots where the horizontal coordinates are the real and imaginary parts of a complex variable.

[1] In oceanography, submerged mountains often have flat tops; these are called "guyots". Truncating a mesh or surface plot might be called the "Make-the-Mountain-a-Guyot" strategy.

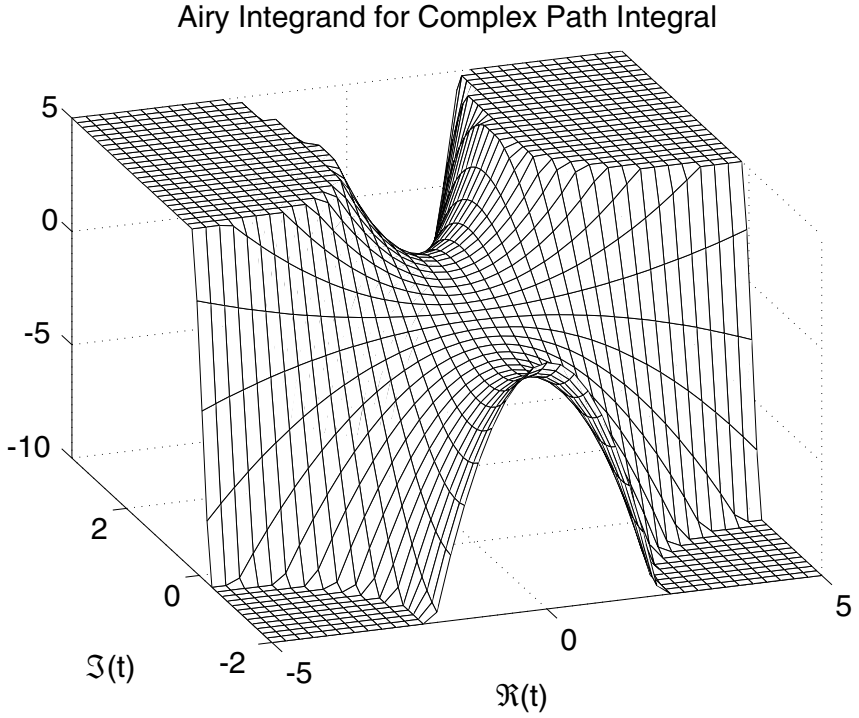

Figure 5.7: Example of a TRUNCATED MESH PLOT. The function is the integrand of a complex path integral representation for the Airy function Ai(z). Because the integrand blows up in magnitude at an exponential rate, it is difficult to plot the function in the complex t-plane so that the saddle point or "col", which is crucial to the steepest descent method for asymptotic approximation of the integrand, to be seen; without truncation, the vertical variations around the saddle point are too small to be seen. Truncating the mountain and valley allows the saddle point and the overall topology of the function to be clearly visible.

In Matlab, the code to do truncate an array Q is as follows:

indextoobig=find(($Q > qmax$)); indextoosmall=find(($Q < qmax$));
Q(indextoobig)=qmax; Q(indextoosmall)=qmin;

The Matlab **find** function returns a vector of indices into the matrix Q which satisfies the logical condition specified in the argument of **find**. Thus, the statement **indextoobig=find(($Q > qmax$));** finds all elements of the array $Q(ii, j)$ which are larger than $qmax$, which the user can specify arbitrarily, and then *indextoobig* contains the indices of all these too-large elements. The statement **Q(indextoobig)=qmax** modifies all these elements to set them equal to the upper truncation $qmax$ while leaving all the other elements — all those $Q(ii, j)$ such that $Q(ii, j) < qmax$ unchanged. The ability of Matlab matrices to be called with an argument which is a vector of indices, and have vector operations performed only on elements with those indices and no others, is one of the most powerful features of the language.

5.6 Cutaway Mesh Plots

Another important strategy is to delete part of the domain in the $x - y$ plane, which in the absence of such deletions is always a rectangle. Fig. 5.8 shows two different mesh plots, one standard and one cutaway, of a function that mimics the waves of the two-dimensional wave equation spreading away from the origin. To use geographical language, the function has a ring-shaped ridge at large radius from the origin and a shallower, ring-shaped valley at smaller radius. The valley is COMPLETELY INVISIBLE in the standard mesh plot because our view of this depression is blocked by the tall ridge in front of it. Rotating a plot will sometimes solve such obstructed-view difficulties, but this function is radially symmetric, so it is invariant under rotations in the horizontal plane. With a very high or low viewing angle, we would glimpse a bit of the depression, at least enough to know that there was a depression, but it is difficult to accurately estimate the size and extent of such features from extreme viewing angles.

When one quadrant of the horizontal domain is chopped away, however, all becomes clearer. We can see *into* the region enclosed by the ring-ridge to see the depression.

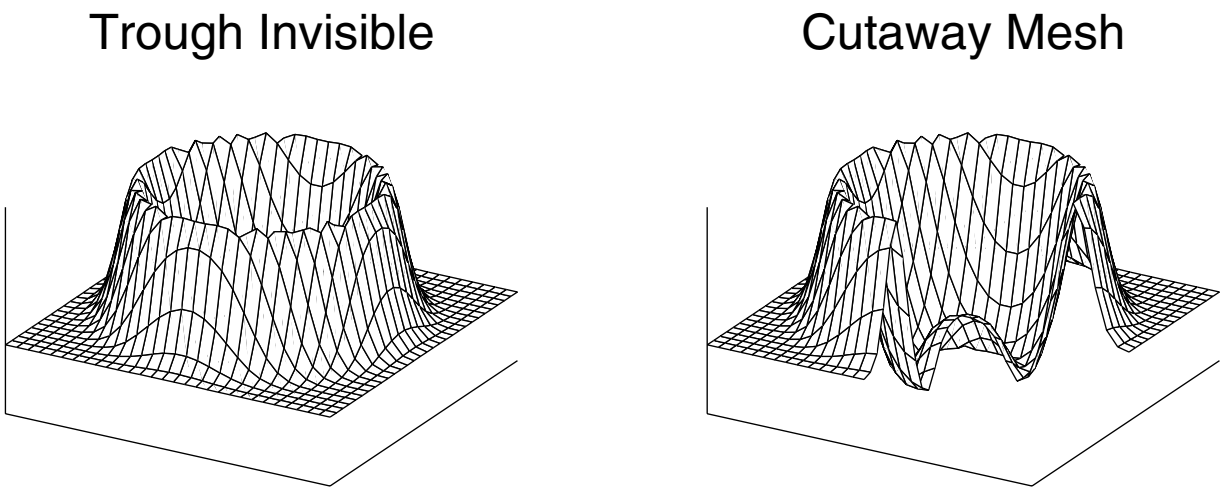

Figure 5.8: The function plotted is radially symmetric with a maximum of one at $r = 2$ and a minimum of -0.35 at $r = 1$ where r is distance from the origin in the $x - y$ plane. In a standard mesh plot, the minimum is INVISIBLE. When a quarter of the mesh is cut away, both the ridge and the trough are clearly visible. Inspired by Fig. 7.11 in Morse and Feshbach (1954).

5.7 Variable Grid Spacing to Code Positive and Negative Heights

It is highly desirable to distinguish positive-valued regions from negative-valued regions through some additional visual cue beyond the spatial shapes of the meshlines themselves. Fig. 5.9 shows an interesting and effective device for doing this in black-and-white graphics. Parts of the surface above the plane $z = 0$ are shown with a fine mesh; the valleys are show with a coarse mesh.

Unfortunately, most mesh routines do not offer this as a standard option. However, work-arounds are possible. In Matlab, for example, one can plot the mesh twice. Negative values are replaced by Nan ("Not-a-Number"), and are therefore not shown in the first plot, which uses a fine mesh. The NaN's are reversed, and replaced positive values, for the second plot, which uses a coarse mesh. When the two plots are superimposed using the **hold** command, the result will imitate this graph from Wolf.

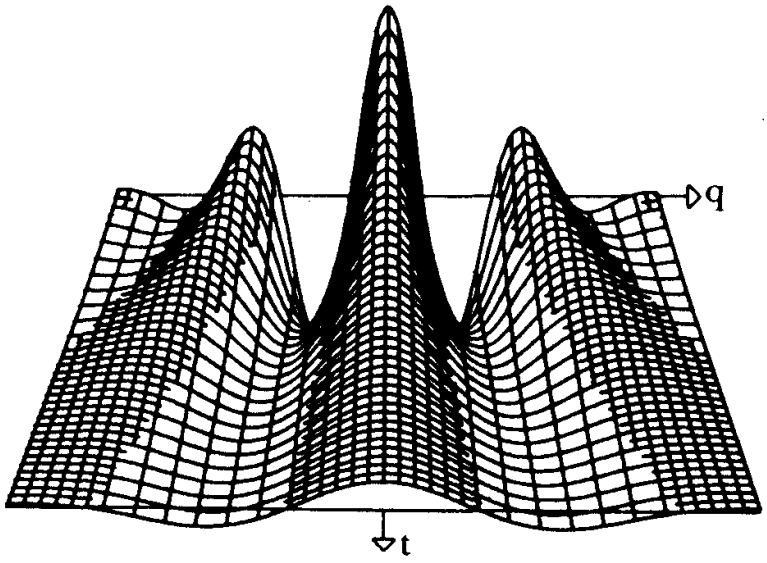

Figure 5.9: A mesh plot in which positive-value parts of the surface are depicted with a fine mesh whereas a coarse mesh is used for parts of the surface below the plane $z = 0$. From *Integral Transforms in Science and Engineering* by Kurt Bernardo Wolf, Plenum Press, New York, pg. 427, (1979).

5.8 Compromise in Grid Density

One tricky issue with mesh plots is that if the grid is coarse, important but small-scale features will be invisible and visible cues for depth perception will be so limited that the eye struggles to perceive the three-dimensionality of the plot (Fig. 5.10). However, if the lines are too dense, the graph will become an unreadable mess of black.

A dense plot is especially tricky when the plot is reduced for publishing, either for the Web or by a commercial printer. The full-size original may look okay, but the published miniaturized version has the lines so close together as to be unrecognizable.

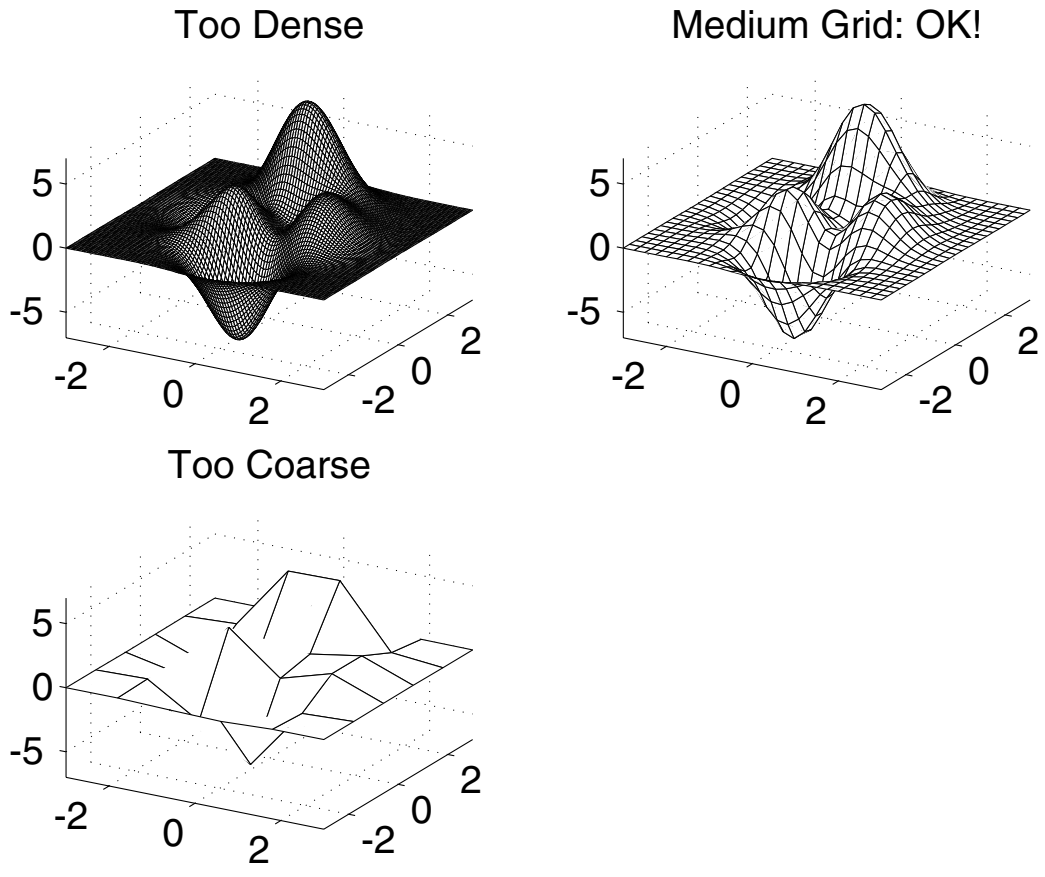

Figure 5.10: The 100 x 100 grid (upper left) is too black when printed; the 6 x 6 (lower left) is so coarse that the mesh curves are jagged with large slope discontinuities and minor features are invisible. The 25 x 25 grid (upper right) is a reasonable compromise.

5.9 Importance of Viewpoint

A mesh plot is not completely specified until the *viewing angle* has been chosen: two angles that specify the observer's point of view in a spherical coordinate system system centered on the origin in the $x - y - q$ space. The default angles are often satisfactory, but more often than not, different viewpoint provides a clearer perception of the peaks and valleys of a given figure. In Matlab, one can vary the viewing angle by **view(horizontal angle, vertical angle)** where the angles are specified in *degrees*.

Fig. 5.11 compares four different views of the Matlab "peaks" function. It is necessary to specify $q(x, y)$ because the optimum viewing angle is generally different for each different function.

The first argument of **view** spins the mesh about the vertical axis. It is important to clearly label the x and y axis so that the reader understands the orientation of the surface.

The second argument of **view** specifies the vertical angle; a choice of ninety is to look down on the surface of $q(x, y)$ from directly above. When the vertical angle is small, valleys and low hills are easily obscured or even completely hidden by taller features in the foreground. (Therefore, one usually should choose the horizontal angle so that the taller features are in the background, that is, farthest from the viewer's eye.) When the vertical angle is tall, the effects of vertical relief are muted so that it becomes much harder to judge heights and depths than when looking from the side. It is necessary to experiment and choose a suitable compromise for each plotted function.

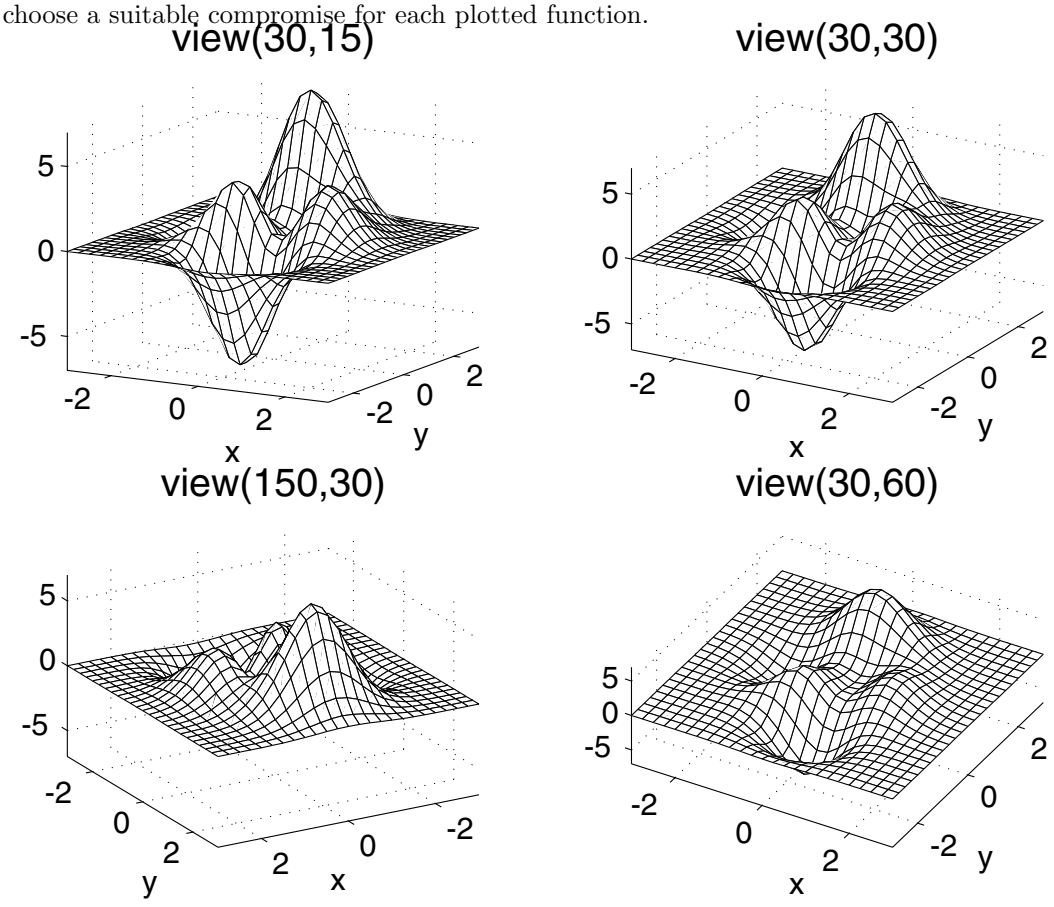

Figure 5.11: The same figure as viewed from 4 different viewing angles; the titles give the arguments of the Matlab **view** command.

5.10 Spinning the Mesh Plot or Three-Dimensionality Through Animation

The Matlab lines below allow animate the rotation of the mesh diagram with an option to convert the movie to a Quicktime video. This may be helpful in experimenting with the choice of horizontal and vertical viewing angles. (Horizontal only in the code fragment, but it is trivial to spin the mesh in the vertical if one wishes.)

The greatest usefulness of video-of-spinning-mesh is that sometimes there is NO single viewpoint that is optimum for viewing all features. One option is to make a composite graph which shows the same surface from two or more viewing angles. In a static medium like print, this may be the best that one can do.

On the computer screen, however, or at a conference with a suitable projection system, one may be able to do much better by using the animation as the endproduct. Our visual systems are very good at using motion cues. Indeed, psychologists have shown that for some lower lifeforms such as forms, a food source which is static may be invisible to the animal, even when right in front of the creature's eyes. Only when the flies are moving are they recognized as food-insects by the frog's cognitive system.

Human beings have much better capabilities for unmoving visual fields; still, the greatest threats to humans and our primate ancestors were threats that *moved*. When we view a rotating image, we may notice features or details that were missed when looking at a set of stills from the same sequence.

Animations-of-rotating-mesh-figures do have a couple of disadvantages. First, it is a lot more trouble to present them at conferences, or in an electronic journal, that it is to make a transparency of a non-rotating mesh diagram. Second, an animation requires STUDY; a quick glance at a printed mesh diagram may suffice to identify the major mountains and valleys, but a rotating image requires some careful attention. It may be very difficult to achieve such study and attention in the confusion of a conference presentation or seminar where the session chair is frowning at his watch, and half the audience is fidgeting about hours and hours of listening to earlier presentations.

The result is that rotating-mesh-animations are probably most useful for *exploratory* graphics. The interesting details that are noticed at leisure can then be condensed into a handful of static mesh plots.

Matlab Code to Change Views and Make a Quicktime Movie

```
[X,Y,Z] = peaks(30);
mesh(X,Y,Z);
axis([-3 3 -3 3 -10 10])    % fix axes so scaling does not change
axis off        % erase axes because they jump around
shading interp;          colormap(copper);
% SPIN and ANIMATE
nframes=30;
MovieMatrix=moviein(nframes);

for ii=1:nframes
view(-37.5+(360/nframes)*(ii-1),30)
MovieMatrix(:,ii)=getframe;
end

movie(MovieMatrix)   % play themovie
qtwrite(MovieMatrix,colormap,'SpinPeaksDemo30.qt')
```

5.11 Mesh Plots on Non-Rectangular Domains

Just as for contour plots, it is usually possible to make a mesh plot for a non-rectangular domain using standard software so long as the grid is *logically rectangular*. By this we mean that the function $q(x, y)$ is represented by a matrix $Q(i, j)$ whose indices fill a rectangle in the $i - j$ plane. The corresponding grid point values in Cartesian coordinates, $X(i, j)$ and $Y(i, j)$ need not specify a rectangle in the $x - y$ plane.

Figs. 5.12 and 5.13 are examples. Both illustrate a frequent artifice with complicated geometries: a second mesh surface, flattened by replacing the array $Q(i, j)$ by a constant independent of (i, j), has been placed under the surface to illustrate the underlying grid. With a contour plot, the same effect is best achieved by using the *hold* command and superposing a linegraph that shows the boundary.

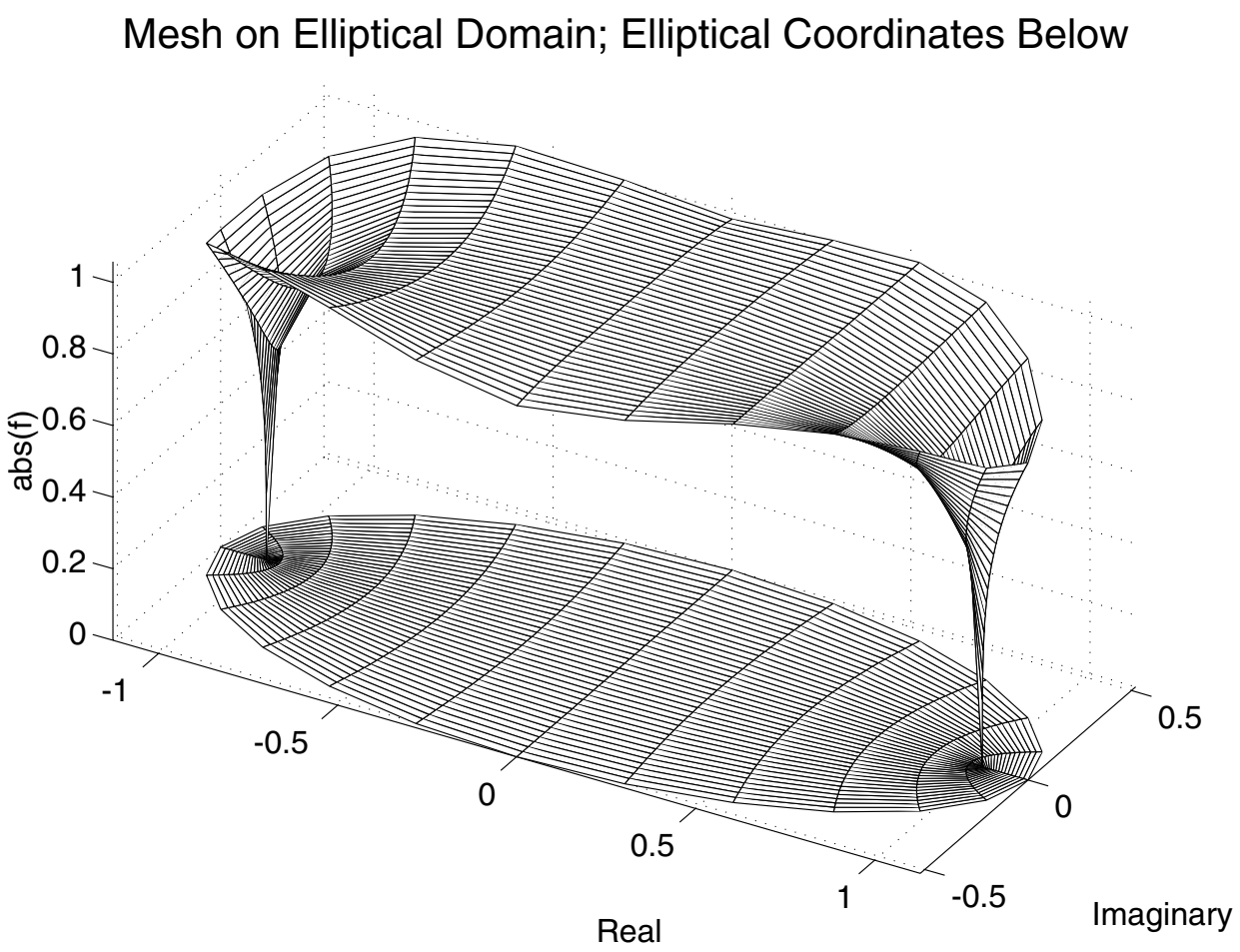

Figure 5.12: Mesh plot on an elliptical domain, obtained through Matlab's standard command **mesh(X,Y,Q)**. The arrays X, Y which store the grid point locations in the Cartesian coordinates x and y are the images of a rectangular grid in the elliptic coordinates (μ, θ) under the usual mapping from the elliptic to Cartesian coordinates: $X(i, j) = cosh(\mu_i)\cos(\theta_j), Y(i, j) = \sinh(\mu_i)\sin(\theta_j)$.

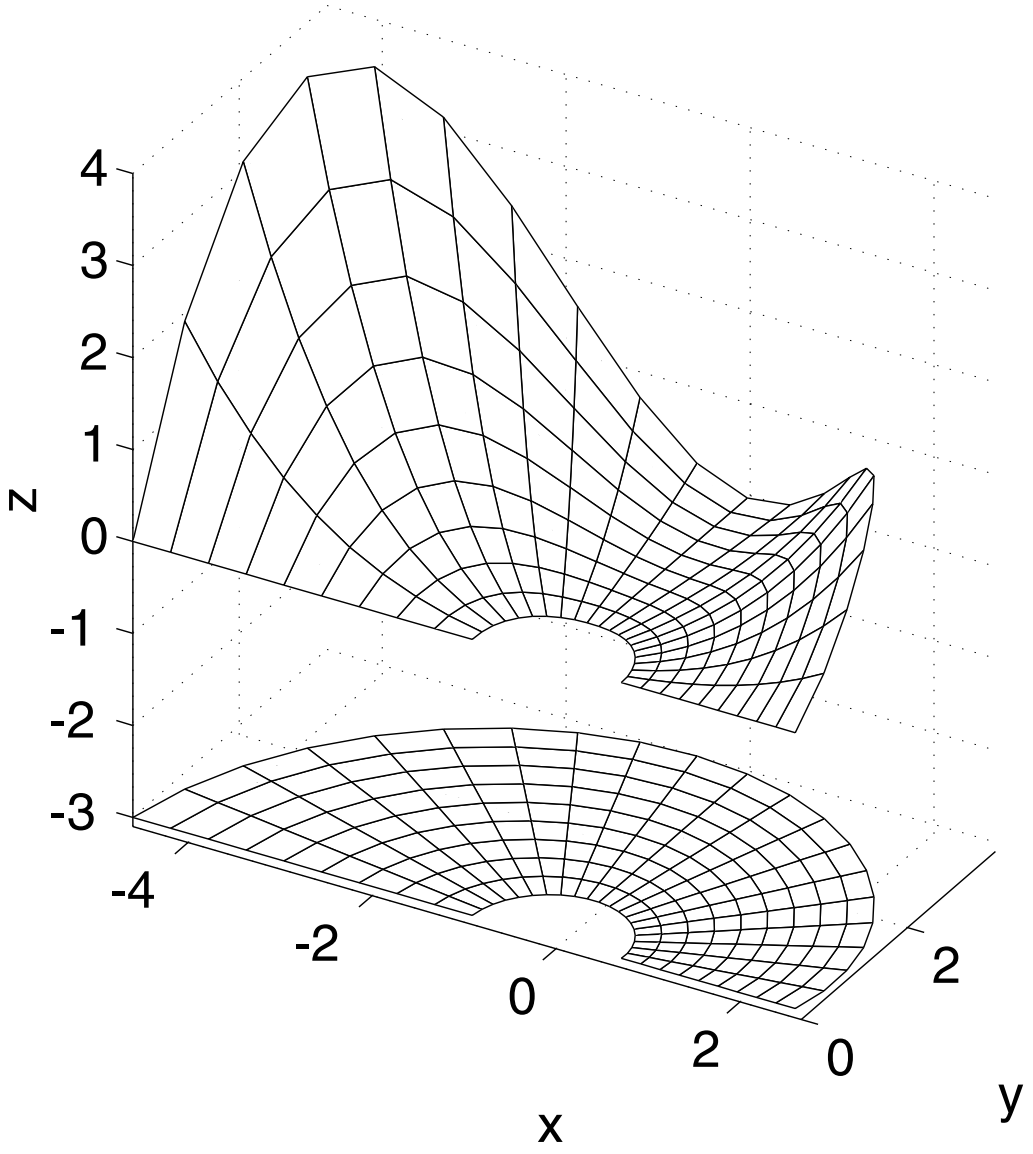

Figure 5.13: Mesh plot on a grid taken from the listing for Fig. 2.27 of *Numerical Analysis and Graphic Visualization with Matlab* by Shoichiro Nakamura, Prentice-Hall (1996). The lower mesh plot, made by multiplying replacing the heights by a constant, shows the grid. The upper mesh is of $q \equiv x^2 y$, which is different from Nakamura's example on the same grid.

For very irregular geometries, Matlab 5.x provides a function (**delaunay**) to compute a Delauny triangulation of an irregular set of points. Matlab also supplies subroutines **trimesh** and **trisurf** for computing mesh and surface plots on the triangular grid created by **delaunay**. Fig. 5.14 illustrates the Delaunay triangulation of a complicated region, followed by application of the Matlab program **trimesh**.

Unfortunately, there is no Matlab routine for contouring on triangles. Nakamura(1996) provides a contour plot routine for an irregular triangular mesh. His routines are available as freeware on the Mathworks, Inc., Web site as user-contributed software.

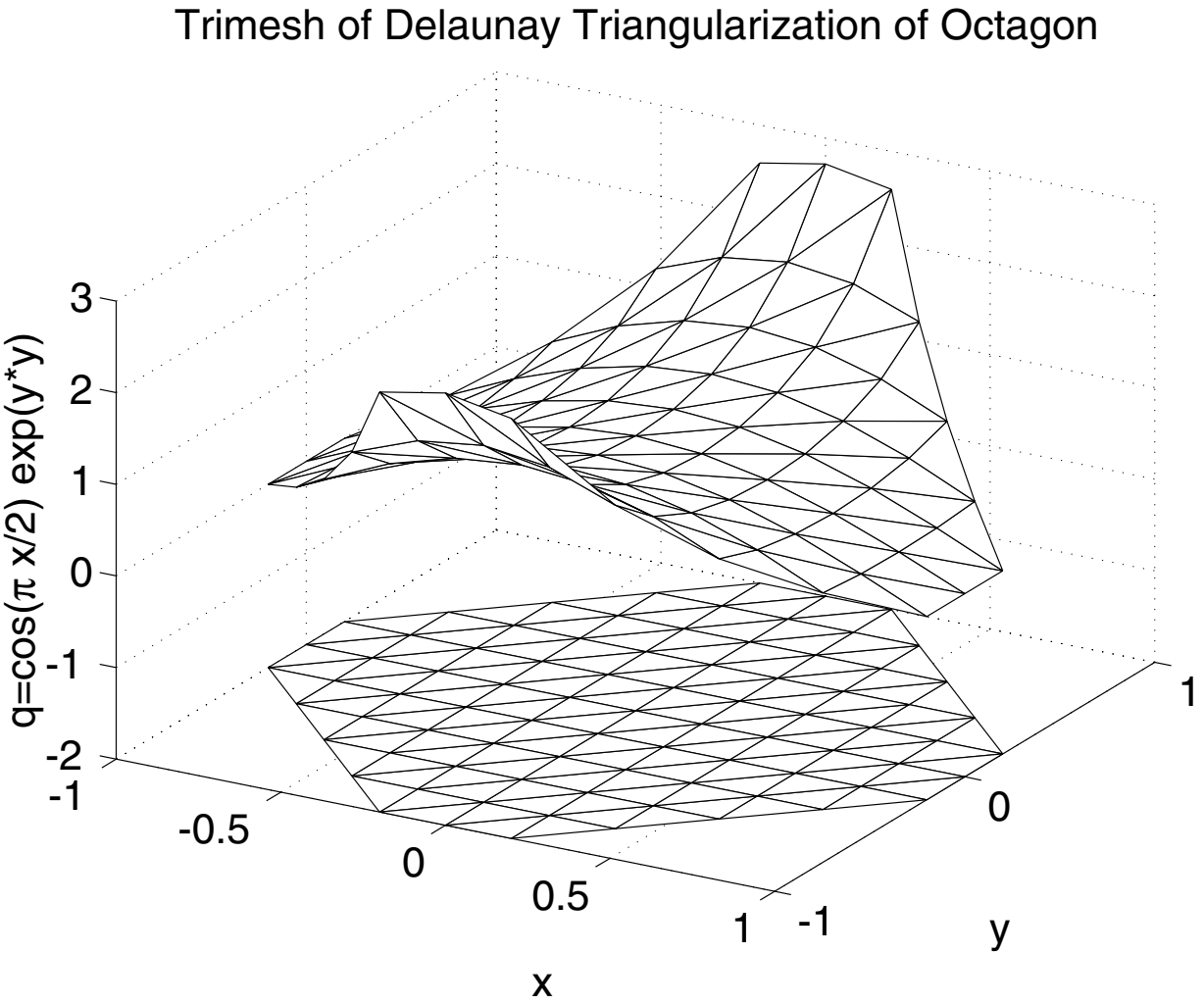

Figure 5.14: Surface of the function $q(x, y) = cos(\pi x/2)exp(y^2)$ on a triangular mesh

Appendix: Non-Standard meshc

```
function h=meshc(x,y,z,c,zpos)
%MESHC  Combination mesh/contour plot.
%   MESHC(...) is the same as MESH(...) except that a contour plot
%   is drawn beneath the mesh.
%
%   Because CONTOUR does not handle irregularly spaced data, this
%   routine only works for surfaces defined on a rectangular grid.
%   The matrices or vectors X and Y define the axis limits only.
%
%   See also MESH.

%         input: c is the colormap
%         zpos is the vertical position of the contour plot.

error(nargchk(1,5,nargin));

if nargin==1,  % Generate x,y matrices for surface z.
  z = x;
  [m,n] = size(z);
  [x,y] = meshgrid(1:n,1:m);

elseif nargin==2,
  z = x; c = y;
  [m,n] = size(z);
  [x,y] = meshgrid(1:n,1:m);

end

if min(size(z))==1,
  error('The surface Z must contain more than one row or column.');
end

% Determine state of system
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;

% Plot mesh.
if nargin==2 | nargin==4,
  hm=mesh(x,y,z,c);
else
  hm=mesh(x,y,z);
end

hold on;

a = get(gca,'zlim');

if nargin ~= 5
```

```
zpos = a(1); % Always put contour below the plot.
end % if

% Get D contour data
[cc,hh] = contour3(x,y,z);

%%% size zpos to match the data

for ii = 1:length(hh)
    zz = get(hh(ii),'Zdata');
if zz(1) < 0
set(hh(ii),'LineStyle','--','LineWidth',1);
else
set(hh(ii),'LineWidth',2)
end % if
    set(hh(ii),'Zdata',zpos*ones(size(zz)));
end

if ~hold_state, set(cax,'NextPlot',next); end
if nargout > 0
    h = [hm; hh(:)];
end
```

Appendix: Matlab code for Three-Dimensional, Two-Layer Contour Plot

```
function h=contour_twolayer(x,y,zbottom,ztop,zpos,labelflagtop,labelflagbottom)
% two layer plot ... the five arguments are mandatory
%         input: c is the colormap
%            Use colormap([0 0 0]); cmap = colormap;
%             to get black contour lines
%         zpos is the vertical separation in
%         between the two 3D contour plots

% both the sixth and seventh arguments must be both used or
% both omitted. labelflagtop=1 turns on inline contour labels.
% and similarly labelflagbottom=1 labels the lower layer.

% Convention: negative-valued contours are dashed and thinner than solid isolines.

error(nargchk(5,7,nargin));

if nargin==5,labelflagtop=0; labelflagbottom=0;  end % if

if min(size(zbottom))==1,
error('The surface Z must contain more than one row or column.'); end

% Determine state of system
cax = newplot; next = lower(get(cax,'NextPlot')); hold_state = ishold;
```

```
                    [ccbottom,hhbottom] = contour3(x,y,zbottom);
for ii=1:length(hhbottom)
    zz = get(hhbottom(ii),'Zdata');  % all elements in zz equal zz(1)
if zz(1) < 0,   % Set negative-value contours to dashed
set(hhbottom(ii),'LineStyle','--','LineWidth',1);
else
set(hhbottom(ii),'LineWidth',2),  % Thicken positive-valued isolines
end % if
    set(hhbottom(ii),'Zdata',0*ones(size(zz))); % Flatten 3D contours to z=0
end % ii

if labelflagbottom==1, textlabelhandles=clabel(ccbottom,hhbottom);
set(textlabelhandles,'FontSize',12);  end % if
                            hold on;
                [cc,hh] = contour3(x,y,ztop);

if labelflagtop==1, textlabelhandles=clabel(cc,hh);
set(textlabelhandles,'FontSize',12); end % if

for ii = 1:length(hh)
    zz = get(hh(ii),'Zdata');
if zz(1) < 0
set(hh(ii),'LineStyle','--','LineWidth',1);
else
set(hh(ii),'LineWidth',2)
end % if
    set(hh(ii),'Zdata',zpos*ones(size(zz))); % Flatten 3D contours to z=zpos
end

xmin=min(min(x));  xmax=max(max(x)); ymin=min(min(y));  ymax=max(max(y));
axis([xmin xmax ymin ymax   0 zpos]);
% put rectangular horizontal frame around each contour plot
xframe=[xmin xmax xmax xmin xmin ]; yframe=[ymin ymin ymax ymax ymin];
plot3(xframe,yframe,[0 0 0 0 0],'k',xframe,yframe,[zpos zpos zpos zpos zpos],'k')
set(gca,'ZTick',[]);  % Remove ticks from z axis, which have no meaning here

if ~hold_state, set(cax,'NextPlot',next); end
if nargout > 0,  h = [hhbottom; hh(:)]; end
```