Chapter 4: Plotting and Analyzing Engineering Functions and Data

4.1: Plotting Functions and Expressions

Try It! (p. 79)

Replot the function *v* described in Example 4-1 with a vertical range that begins at 0.

Hint

Consult the on-line help for plot for the necessary optional arguments, if needed.

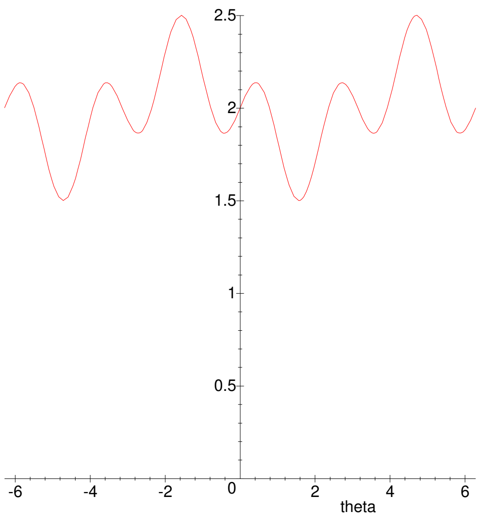
Solution

```
> restart;
> v := 2 + sin(theta) * cos(2*theta)/2;
```

$$v := 2 + \frac{1}{2}\sin(\theta)\cos(2\,\theta)$$

There are a variety of possible solutions, including

```
> plot( v, theta = -2*Pi .. 2*Pi, 0..2.5 );
```




```
> plot( v, theta = -2*Pi .. 2*Pi, view=[ -2*Pi..2*Pi, 0..2.5 ] );
> plot( v, theta = -2*Pi .. 2*Pi, 'v'=0..2.5 );
```

Note

The single quotes ('') are needed around the v in the third argument to prevent Maple from expanding the name v to the value that has been assigned to it.

```
>
```

Try It! (p. 81)

Repeat Example 4-2 with the first argument specified as a set of functions.

Solution

```
> restart;
> v1 := sin( theta ) ;
> v2 := theta -> cos(2*theta) / 2 ;
```

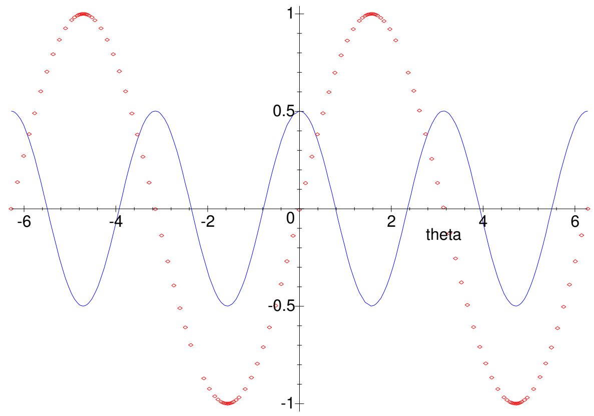
$$v1 := \sin(\theta)$$

$$v2 := \theta \rightarrow \frac{1}{2}\cos(2\,\theta)$$

Here is the plot from the text, i.e., when the first argument is a list

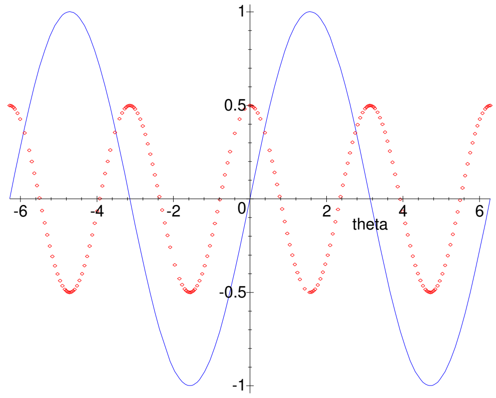
```
> plot( [ v1, v2(theta) ], theta = -2*Pi .. 2*Pi,
```

```
>       color=[RED,BLUE], style=[POINT,LINE] );
```



and the plot obtained when the first argument is a set

```
> plot( { v1, v2(theta) }, theta = -2*Pi .. 2*Pi,
>       color=[RED,BLUE], style=[POINT,LINE] );
```



The point of this exercise is that the color and style of the functions in the second plot may not be determined by the order of appearance of the functions in the set. The elements of a set are not ordered (there is no "first" element of a set). Sometimes the plot will appear one way, in others the color and style will be reversed. Try this in a lab setting and let the students see this firsthand.

```
>
```

Try It! (p. 84)

Use `display` and the `insequence=true` option to create an animation that shows the graphs of v, v_1, and v_2 as a sequence of three frames.

Solution

```
> restart; with( plots ):
> v := 2 + sin(theta) * cos(2*theta)/2;
```

$$v := 2 + \frac{1}{2}\sin(\theta)\cos(2\,\theta)$$

```
> v1 := sin( theta ) ;
```

$$v1 := \sin(\theta)$$

```
> v2 := theta -> cos(2*theta) / 2 ;
```

$$v2 := \theta \rightarrow \frac{1}{2}\cos(2\,\theta)$$

The three plots can be created as in Example 4-3 (p. 81)

```
> P1 := plot( v, theta = -2*Pi .. 2*Pi ):
> P2 := plot( v1, theta = 0 .. 2*Pi, color=GREEN, style=POINT ):
> P3 := plot( v2(theta), theta = -Pi .. 0, color=BLUE, linestyle=2 ):
```

The animation is created using

```
> display( [ P1, P2, P3 ], insequence=true );
```

Follow the directions in the text (p. 83) to navigate the animation.

```
>
```

What If? (p. 92)

What if the project constraints call for a more discriminating optical filter than the one with $F=20$? This means that the filter must have a narrower bandwidth. What does this imply regarding the bands of frequencies that are transmitted? Would the finesse of the new filter be larger or smaller than $F=20$? What is the relative change in the reflectivity corresponding to a doubling of the finesse?

Solution

From the information in the Fundamentals section of the Application, it is stated that a narrower bandwidth of frequencies is transmitted when the filter's finesse increases. In particular, the new filter would be expected to have a finesse larger than 20.

```
> restart;
```

The relative change in the reflectivity when the finesse doubles can be determined both for $F=20$ and in general.

```
> finesse := F=Pi*sqrt(R)/(1-R);
```

$$finesse := F = \frac{\pi\sqrt{R}}{1-R}$$

Suppose the original finesse is $F=F_0$. There are two values of R which satisfy the finesse equation

```
> Rorig := solve( subs( F=F0, finesse ), R );
```

$$Rorig := -\frac{-F0 + \frac{1}{2}\frac{\pi(-\pi+\sqrt{\pi^2+4\,F0^2})}{F0}}{F0}, -\frac{-F0 + \frac{1}{2}\frac{\pi(-\pi-\sqrt{\pi^2+4\,F0^2})}{F0}}{F0}$$

Before both of these solutions are accepted, recall that the reflectivity is between 0 and 1. Upon closer inspection it is seen that only one of these solutions satisfies this constraint.

```
> subs( F0=20, [Rorig] );
> evalf(");
```

$$\left[1-\frac{1}{800}\pi(-\pi+\sqrt{\pi^2+1600}), 1-\frac{1}{800}\pi(-\pi-\sqrt{\pi^2+1600})\right]$$

$$[.8547736446, 1.169900366]$$

Thus, the reflectivity corresponding to the original finesse is

```
> Rorig := normal( Rorig[1] );
```

$$Rorig := \frac{1}{2}\frac{2\,F0^2+\pi^2-\pi\sqrt{\pi^2+4\,F0^2}}{F0^2}$$

```
>
```

When the finesse is doubled, $F=2\,F_0$, the corresponding reflectivity can be found exactly as before:

```
> Rdoub := solve( subs( F=2*F0, finesse ), R );
```

$$Rdoub := -\frac{1}{2}\frac{-2\,F0 + \frac{1}{4}\frac{\pi(-\pi+\sqrt{\pi^2+16\,F0^2})}{F0}}{F0}, -\frac{1}{2}\frac{-2\,F0 + \frac{1}{4}\frac{\pi(-\pi-\sqrt{\pi^2+16\,F0^2})}{F0}}{F0}$$

```
> subs( F0=20, [Rdoub] );
> evalf(");
```

$$\left[1-\frac{1}{3200}\pi(-\pi+\sqrt{\pi^2+6400}),\ 1-\frac{1}{3200}\pi(-\pi-\sqrt{\pi^2+6400})\right]$$

$$[.9244838992,\ 1.081684604]$$

```
> Rdoub := normal( Rdoub[1] );
```

$$Rdoub := \frac{1}{8}\,\frac{8\,F0^2+\pi^2-\pi\sqrt{\pi^2+16\,F0^2}}{F0^2}$$

```
>
```

The ratio of the reflectivities is, for a general finesse,

```
> ratio := Rdoub/Rorig;
```

$$ratio := \frac{1}{4}\,\frac{8\,F0^2+\pi^2-\pi\sqrt{\pi^2+16\,F0^2}}{2\,F0^2+\pi^2-\pi\sqrt{\pi^2+4\,F0^2}}$$

and, in particular, when $F=20$,

```
> subs( F0=20, ratio );
> evalf( " );
```

$$\frac{1}{4}\,\frac{3200+\pi^2-\pi\sqrt{\pi^2+6400}}{800+\pi^2-\pi\sqrt{\pi^2+1600}}$$

$$1.081554052$$

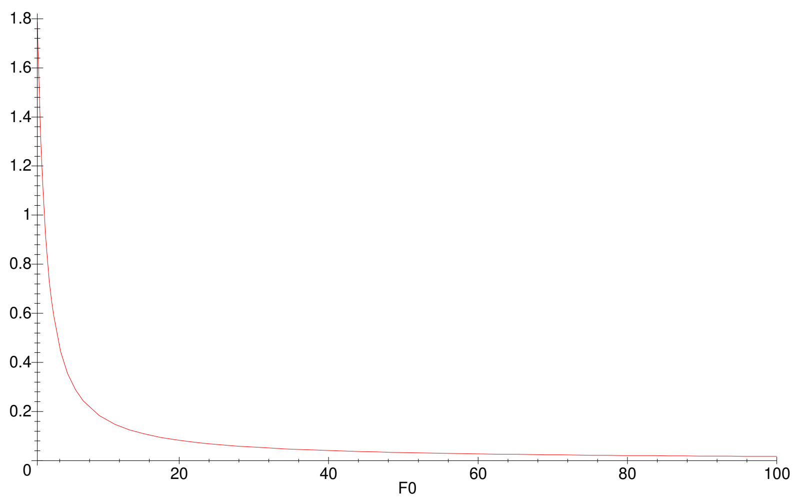
Thus, when the finesse doubles from F=20 to F=40, the reflectivity increases by slightly more than 8%.

```
>
```

The general expression for the change in reflectivity is rather complicated. From a graph, it is seen that the relative change in the reflectivity is a decreasing function of the finesse.

```
> plot( ratio-1, F0=1..100,
>      title=`Relative Change in Reflectivity when Finesse is Doubled` );
```

Relative Change in Reflectivity when Finesse is Doubled




```
>
```

4.2 More Plotting Commands

Try It! (p. 93)

Modify the plot of the ellipse to include the major and minor axes as dashed lines with different colors.

Hint

You may wish to use implicitplot to draw one or both of the line segments.)

Solution

```
> restart; with(plots):
```

The ellipse in question is

```
> ELLIPSE := (x-1)^2/4^2 + (y+2)^2/2^2 = 1;
```

$$ELLIPSE := \frac{1}{16}(x-1)^2 + \frac{1}{4}(y+2)^2 = 1$$

The plot of the ellipse is obtained as shown in the text - except that the plot is assigned to a name.

```
> ELL := implicitplot( ELLIPSE, x = -3 .. 5, y = -4 .. 0 ):
```

Note

Note the use of : instead of ; when assigning a plot to a name.

```
>
```

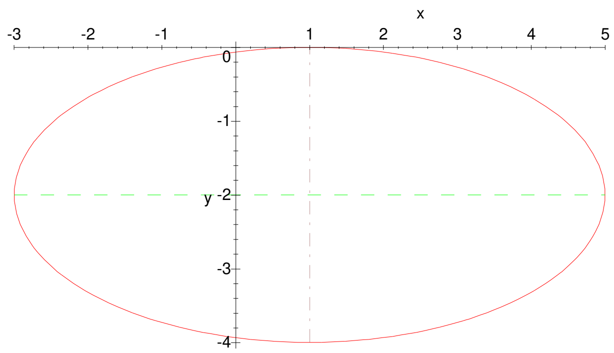
The major axis is a horizontal line, which is easily plotted

```
> MAJ := plot( -2, x=-3..5, color= green, linestyle=3 ):
```

The minor axis is a vertical line, which is not so easily plotted. There are a number of ways to plot a vertical line. Following the hint, you are likely to come up with something like

```
> MIN := implicitplot( x=1, x=-3..5, y=-4..0, color=pink, linestyle=4 ):
> display( { ELL, MAJ, MIN },
>          title=`Ellipse: center (1,-2), axes: 4 (hor) and 2 (ver)`,
>          scaling=CONSTRAINED );
```

Ellipse: center (1,-2), axes: 4 (hor) and 2 (ver)




```
>
```

Alternate Solution

An alternate solution is to create a line plot between the two endpoints of the minor axis. This approach could look like

```
> MIN2 := plot( [[1,-4], [1,0]], color=pink ):
> display( { ELL, MAJ, MIN2 },
>          title=`Ellipse: center (1,-2), axes: 4 (hor) and 2 (ver)`,
>          scaling=CONSTRAINED );
>
```

Try It! (p. 94)

To solve this problem, refer to Application 4. The transmission function, T, gives the ratio of light which passes through a filter to that which enters the filter. When x is proportional to the frequency of the light and F is the finesse, the tranmsission function is $T = \dfrac{1}{1 + \left(\dfrac{2\,F\sin(\pi\,x)}{\pi}\right)^2}$. Use implicitplot to plot the points (x, F) where $\mathrm{T}(x, F) = \dfrac{1}{2}$. Use this plot to compute the FWHM for $F = 20$ and to confirm that a higher finesse corresponds to more discriminating filters.

Solution

```
> restart; with(plots):
```

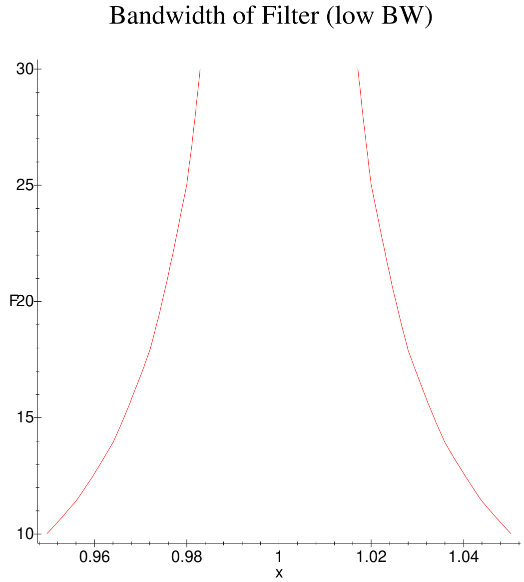
The transmission function is

```
> T := 1/(1+(2*F*sin(Pi*x)/Pi)^2);
```

$$T := \frac{1}{1 + 4\,\dfrac{F^2\sin(\pi\,x)^2}{\pi^2}}$$

The points (x, F) where exactly half of the incident light is transmitted can be plotted using implicitplot as follows:

```
> implicitplot( T=1/2, x=0.9..1.1, F=10..30,
>               title=`Bandwidth of Filter (low BW)` );
```



For a given finesse, F, the FWHM is the (horizontal) distance between the two points on the preceding graph. Since this distance decreases as the finesse increases, the bandwidth of a filter decreases as the finesse increases.

The points (0.9747,19.82) and (1.025,19.91) are good approximations to points on the graph with a finesse of 20. Thus, when $F = 20$, the FWHM is approximately 1.025 - 0.9747 = 0.053.

```
>
```

Try It! (p. 95)

It is natural to assume that the problem with the plot in Example 4-7 can be resolved by plotting more points. Investigate this by using the numpoints= option to determine the minimum number of points necessary to have the curve plotted as a single piece. Is any other detail lost when this occurs?

Hints

1.

It might be helpful to look at the plot of the individual points. This can be done either by specifying style=POINT as an optional argument (see ?plot,options) or interactively via the icons in the context bar.

2.

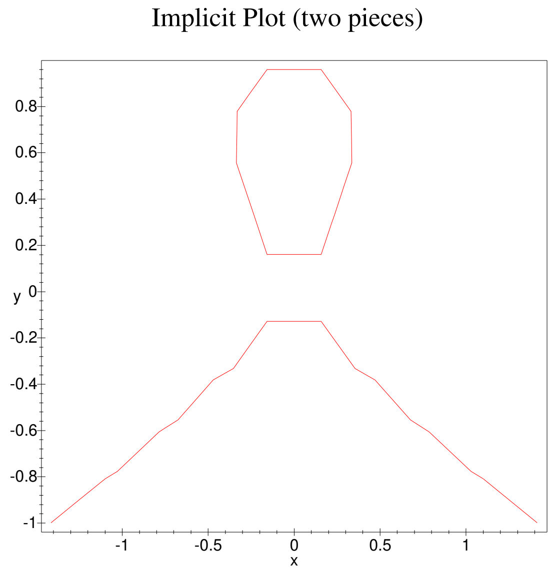
The default number of points in the plot is 49.

Solution

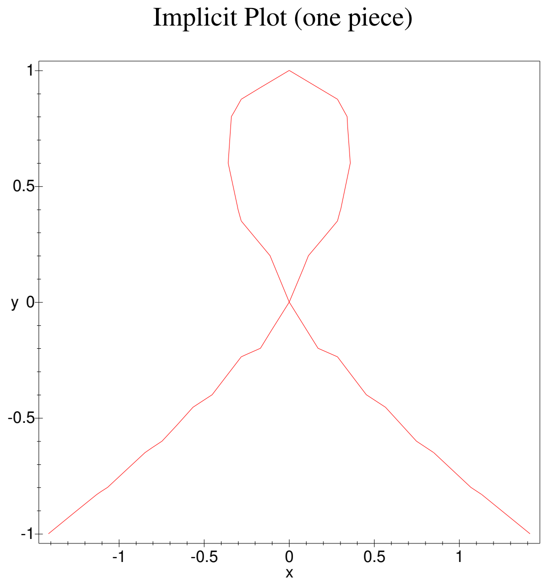
```
> restart; with(plots):
```

The following plots indicate that the two pieces become one when at least 87 points are plotted.

```
> implicitplot( x^2 = y^2 * ( 1 - y ), x = -sqrt(2) .. sqrt(2), y=-1 .. 1,
>               axes=BOXED, numpoints=86, title=`Implicit Plot (two pieces)` );
```




```
> implicitplot( x^2 = y^2 * ( 1 - y ), x = -sqrt(2) .. sqrt(2), y=-1 .. 1,
>               axes=BOXED, numpoints=87, title=`Implicit Plot (one piece)` );
```




```
>
```

Try It! (p. 98)

Create a 3 x 3 array of plots that confirms the earlier observation that the bandwidth of a filter decreases as the finesse increases. Be sure that each plot is labeled and that the same scaling is used in each plot.

Solution

```
> restart; with(plots):
```

The transmission function is defined as in Try It! (p. 94):

```
> T := 1/(1+(2*F*sin(Pi*x)/Pi)^2);
```

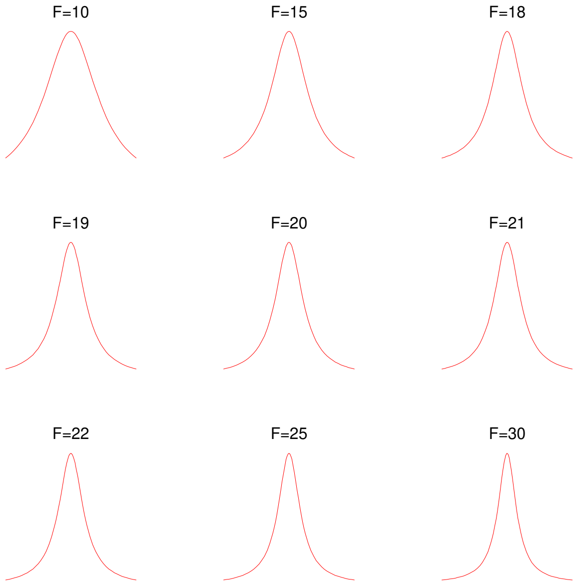
$$T := \frac{1}{1 + 4\,\frac{F^2 \sin(\pi\,x)^2}{\pi^2}}$$

Nine plots for nine different finesse values can be created by

```
> P11 := plot( subs( F=10, T ), x=0.9..1.1, 0..1, title=`F=10`, axes=NONE ):
> P12 := plot( subs( F=15, T ), x=0.9..1.1, 0..1, title=`F=15`, axes=NONE ):
> P13 := plot( subs( F=18, T ), x=0.9..1.1, 0..1, title=`F=18`, axes=NONE ):
> P21 := plot( subs( F=19, T ), x=0.9..1.1, 0..1, title=`F=19`, axes=NONE ):
> P22 := plot( subs( F=20, T ), x=0.9..1.1, 0..1, title=`F=20`, axes=NONE ):
> P23 := plot( subs( F=21, T ), x=0.9..1.1, 0..1, title=`F=21`, axes=NONE ):
> P31 := plot( subs( F=22, T ), x=0.9..1.1, 0..1, title=`F=22`, axes=NONE ):
> P32 := plot( subs( F=25, T ), x=0.9..1.1, 0..1, title=`F=25`, axes=NONE ):
> P33 := plot( subs( F=30, T ), x=0.9..1.1, 0..1, title=`F=30`, axes=NONE ):
```

The composite display can be created with

```
> display( array(1..3,1..3, [[P11,P12,P13],[P21,P22,P23],[P31,P32,P33]] ) );
```



The (expected) narrowing of the bandwidth at half-maximum is apparent from these plots.

```
>
```

4.3 Three-Dimensional Plots

Try It! (p. 100)

Interesting views related to this example include the cross-sections with temperature vs. time (orientation=[0,90]) and temperature vs. position (orientation=[90,90]) and the contour lines in the time vs. position (orientation=[90,0], style=CONTOUR) cross-section. Create each of these three plots. Use display and array to display the three plots side-by-side.

Solution

```
> restart; with(plots):
```

The temperature in the bar is given by

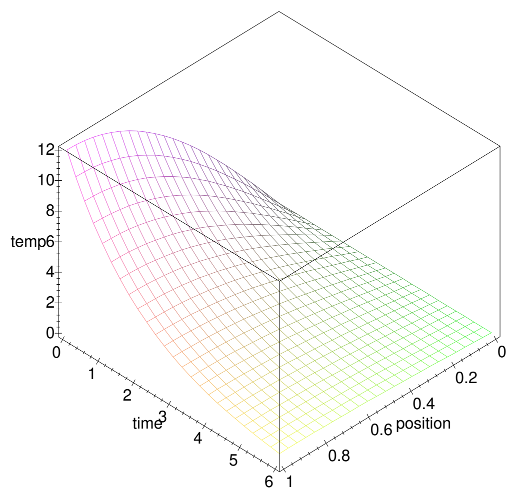
```
> Theta := 12*sin(Pi*x/2)*exp(-t/2);
```

$$\Theta := 12 \sin\left(\frac{1}{2}\pi x\right) e^{(-1/2\, t)}$$

The plot in the book shows the default view

```
> plot3d( Theta, x=0..1, t=0..6, axes=BOXED,
>          labels=[`position`,`time`,`temp`],
>          title=`Heat in a Rod` );
```

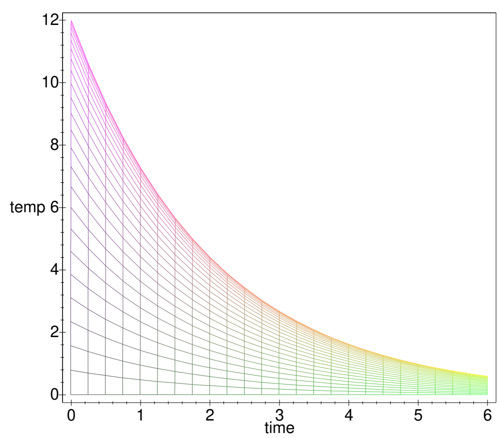
Heat in a Rod



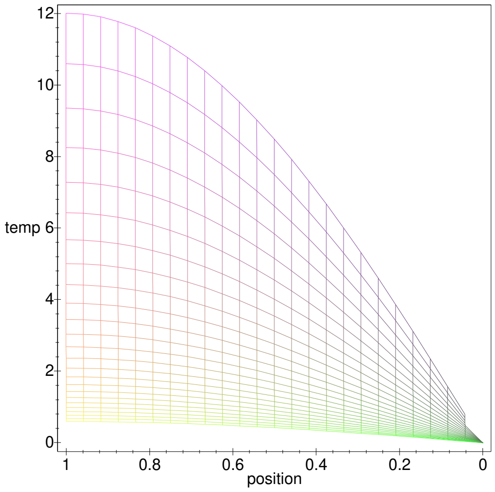

```
>
```

The three specified views of this function are created and displayed individually

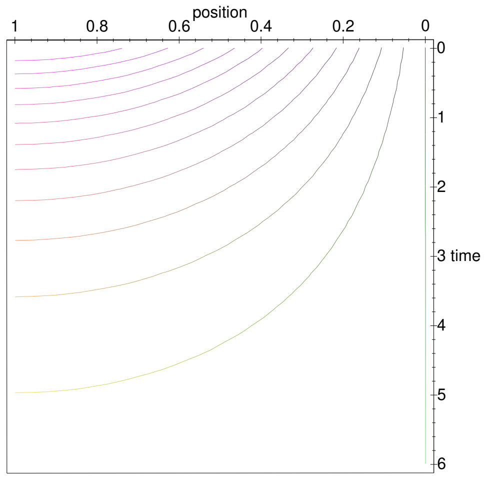
```
> P1 := plot3d( Theta, x=0..1, t=0..6, axes=BOXED,
>                labels=[`position`,`time`,`temp`],
>                orientation=[ 0,90] ):
> P1;
```




```
> P2 := plot3d( Theta, x=0..1, t=0..6, axes=BOXED,
  labels=[`position`,`time`,`temp`],
>                orientation=[90,90] ):
> P2;
```



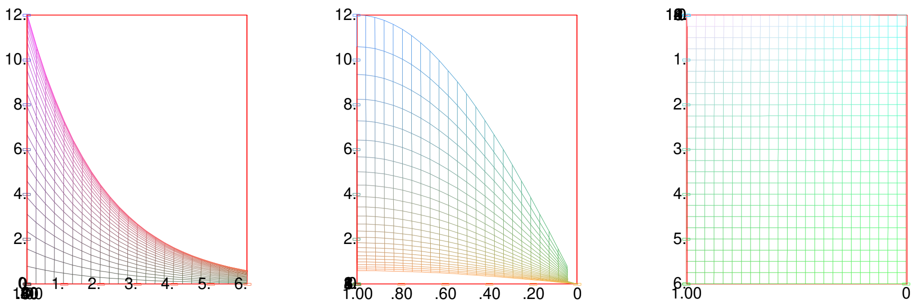

```
> P3 := plot3d( Theta, x=0..1, t=0..6, axes=BOXED,
  labels=[`position`,`time`,`temp`],
>                orientation=[90,0], style=CONTOUR ):
> P3;
```



Note - minor improvement/correction

The time axes in the second and third graphs are reversed from standard conventions. To reverse this, change θ from 90 to -90.

```
> display( array(1..3,[P1,P2,P3]) );
```



Note - bug in side-by-side display of 3D plots

It appears that there is a bug in the Maple code for displaying 3D plots. The options and style settings used to create the contour plot are not respected when the three plots are displayed side-by-side. This problem appears to still exist in Release 5.

```
>
```

Try It! (p. 101)

Find the on-line help for contourplot3d; then use contourplot3d to produce a second contour plot of the function u defined above. Determine where u has its largest and smallest values.

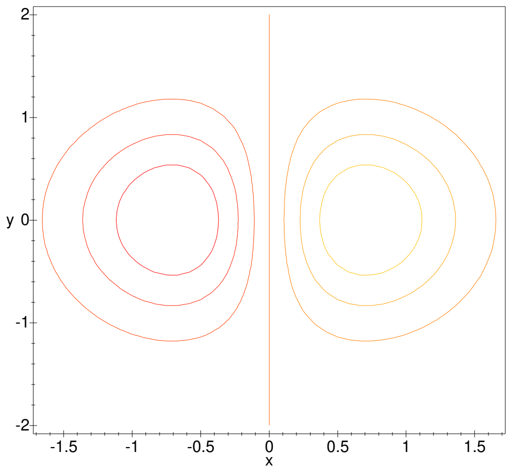
Solution

```
> restart; with(plots):
```

The function and the contour plot displayed in the book are

```
> u := 10 * x * exp(-x^2-y^2):
> contourplot( u, x = -2 .. 2, y=-2..2, grid = [49,49], axes=BOXED,
>                title=`Contour Plot` );
```

Contour Plot

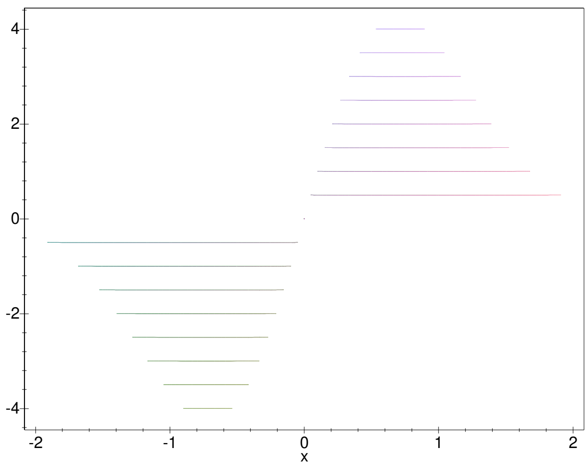



```
>
```

While it is possible to simply change contourplot to contourplot3d, it is more appropriate to include the specific orientation from which the extrema can be identified.

```
> contourplot3d( u, x = -2 .. 2, y = -2 .. 2, grid = [49,49], axes=BOXED,
>                title=`Contour Plot`, orientation=[-90,90] );
```

Contour Plot



From this plot it is apparent that the maximum occurs near $x = .75$ and the minimum occurs near $x = -.75$. The corresponding y coordinates can be estimated from the original 2D plot or by creating a second 3D plot with, e.g., orientation=[0,90]. In either case, the y-coordinate for both extrema is 0.

>

Try It! (p. 104)

Reconsider the transmission function T introduced in the second Try It! in Section 4-2. Originally, animation was used to display this function using two-dimensional plots. Use plot3d (and, possibly, display) to create a three-dimensional plot of $z = \mathrm{T}(x, F)$ and of the plane $z = \frac{1}{2}$ in the same plot. How does this picture relate to the individual frames of the animation? Does the animation contain information that is not (directly) available in the three-dimensional plot, or vice versa?

Solution

Correction

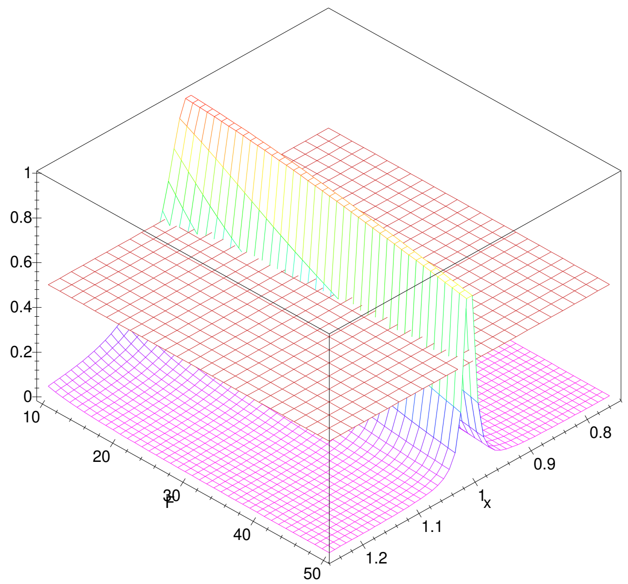
While the transmission function is introduced in Try It! (p. 94), animation is not used until Problem 4 (p. 114).

```
> restart; with(plots):
```

The requested plot could be created using a single plot3d command, but separate commands are used to specify the specific options for the different surfaces

```
> P1 := plot3d( 1/(1+(2*F/Pi)^2*sin(Pi*x)^2), x=0.75..1.25, F=10..50,
>               grid=[50,40], shading=ZHUE ):
> P2 := plot3d( 1/2, x=0.75..1.25, F=10..50, color=orange ):
> display( { P1, P2 }, title=`3D view of bandwidth`, axes=BOXED );
```

3D view of bandwidth



While the 3D view of these functions prevents us from extracting specific data values from the plot, it does give us a better view of the overall situation. For instance, in addition to a decrease in the FWHM, the maximum transmission decreases as the finesse increases.

>

4.4 Working with Discrete Data

Try It! (p. 105)

The five points in Example 4-9 are the vertices of a pentagon. Modify the solution to Example 4-9 to create a plot of this pentagon.

Solution

```
> restart; with(plots):
```

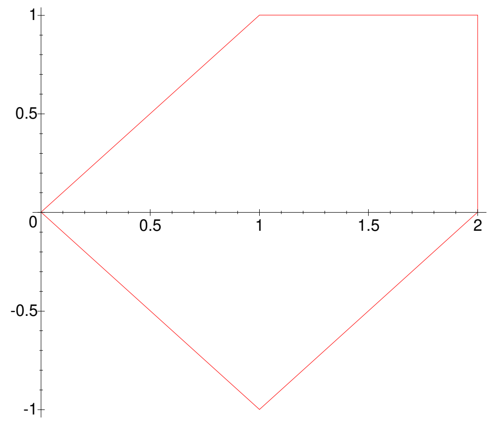
The collection of points is

```
> PTS := [0,0], [1,1], [2,1], [2,0], [1,-1];
```

$$PTS := [0, 0], [1, 1], [2, 1], [2, 0], [1, -1]$$

To draw the pentagon whose vertices are these five points, it is necessary to i) change from a point plot to a line plot and ii) connect the last and first points. Since line plots are Maple's default, this is acheived by simply omitting the style= option from the command in the text. To connect the last and first points, it suffices to append the first point as a sixth entry in the list of points.

```
> plot( [ PTS, PTS[1] ], title=`Five Points, Connected` );
```




```
>
```

Try It! (p. 108)

Repeat the previous steps to find the best quadratic fit to the same set of data. Plot the data points, the best linear fit, and the best quadratic fit all on the same set of axes.

Solution

```
> restart; with(plots): with(stats):
```

To begin, recall the data points and linear fit found in Example 4-10.

```
> X := [ 10, 15, 17, 19 ];
```

$$X := [10, 15, 17, 19]$$

```
> Y := [ 3, 4, 5, 6 ];
```

$$Y := [3, 4, 5, 6]$$

```
> PTS := zip( (x,y)->[x,y], X, Y );
```

$$PTS := [[10, 3], [15, 4], [17, 5], [19, 6]]$$

```
> P := plot( PTS, x=0..20, y=0..8, style=POINT, color=BLUE ):
> SL := fit[leastsquare[ {x,y}, y=m*x+b, {m,b} ] ] ( [ X, Y ] );
```

$$SL := y = \frac{58}{179}x - \frac{79}{179}$$

```
> P1 := plot( rhs(SL), x=0 .. 20 ):
```

The quadratic fit and its plot are found in the same manner - once the second-order term and the corresponding coefficient are added to the arguments of the fit command.

```
> QU := fit[leastsquare[ [x,y], y=a+b*x+c*x^2, {a,b,c} ] ] ([ X, Y ]);
```

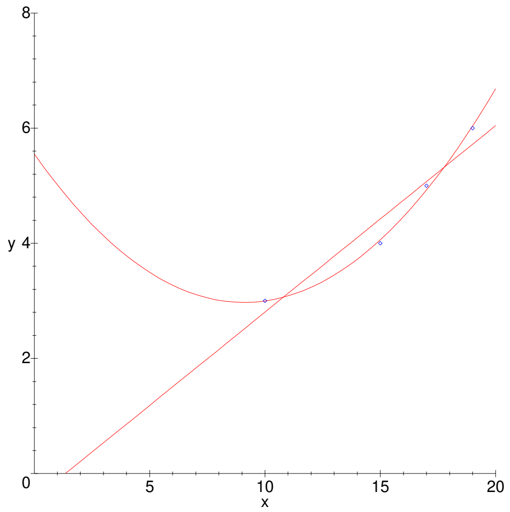
$$QU := y = \frac{37054}{6679} - \frac{7583}{13358}x + \frac{417}{13358}x^2$$

```
> P2 := plot( rhs(QU), x=0..20 ):
```

To conclude, here is the plot containing the data points and best linear and quadratic fits. (In this case it is easy to distinguish the two curves; in other cases it will be advisable to use different colors and/or line styles.)

```
> display( { P, P1, P2 },
>          title=`Data Points and Best Linear and Quadratic Fits` );
```

Data Points and Best Linear and Quadratic Fits




```
>
```

Try It! (p. 112)

Repeat the data-fitting computations looking for fits of the form $B=\frac{a}{F}+b$ and $B=\frac{a}{F^2}$ and $B=\frac{a}{F}+b\,F+c$. How do these compare with the best reciprocal fit?

Solution

```
> restart; with(plots): with(stats):
```

The basic setup is that introduced in Examples 4-11 -- 4-13.

```
> Flist := [ 5, 10.1, 15.01, 20.19, 30.01, 40.11, 49.93 ]:
> Blist := [ 1.096-0.9016, 1.049-0.9502, 1.033-0.966, 1.023-0.9745,
  1.016-0.9818, 1.012-0.9867, 1.01-0.9891 ]:
> PTS := zip( (x,y) -> [x,y], Flist, Blist );
```

$$PTS := [[5, .1944], [10.1, .0988], [15.01, .067], [20.19, .0485], [30.01, .0342], [40.11, .0253], [49.93, .0209]]$$

```
> P := plot( PTS, style=POINT, labels=[`Finesse`, `FWHM`], color=RED ):
> recipfit := stats[fit,leastsquare[ {F,B}, B=a/F], {a} ] ( [Flist,Blist] );
```

$$recipfit := B = \frac{.9811193933}{F}$$

```
> P1 := plot( rhs(recipfit), F=5..50 ):
>
```

The three least-square fits are obtained with separate calls to the fit command.

```
> fit1 := stats[fit,leastsquare[ [F,B], B=a/F+b], {a,b} ] ([Flist,Blist]);
```

$$fit1 := B = \frac{.9665390551}{F} + .001738250480$$

```
> fit2 := stats[fit,leastsquare[ [F,B], B=a/F^2], {a} ] ([Flist,Blist]);
```

$$fit2 := B = \frac{5.351005872}{F^2}$$

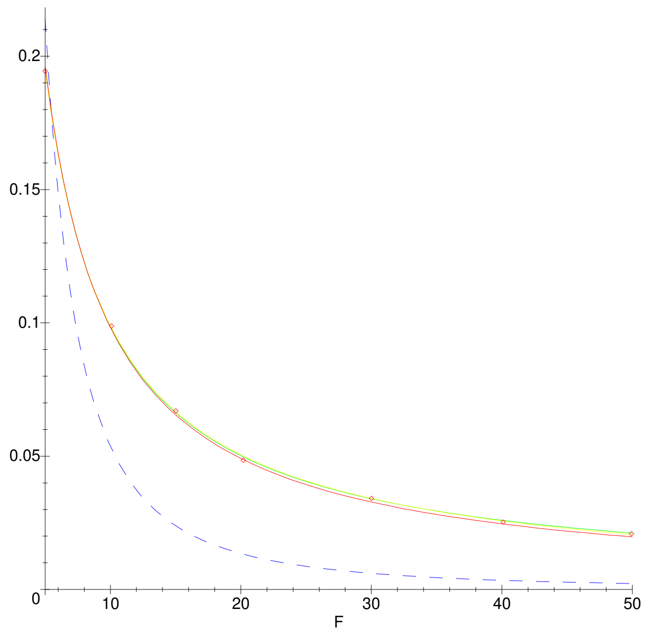
```
> fit3 := stats[fit,leastsquare[ [F,B], B=a/F+b*F+c], {a,b,c} ] ([Flist,Blist]);
```

$$fit3 := B = \frac{.9583832987}{F} - .00003915437855\,F + .003266015047$$

Note that the first and third fits are good approximations to the best reciprocal fit found in Example 4-13. It is somewhat more difficult to compare the second fit. For this we turn to graphical evidence:

```
> P2 := plot( [rhs(fit1), rhs(fit2), rhs(fit3)], F=5..50,
>             color=[GREEN,BLUE,YELLOW], linestyle=[1,3,5]):
> display( { P, P1, P2 }, title=`Bandwidth vs. Finesse - Three Fits` );
```

Bandwidth vs. Finesse - Three Fits



This plot shows that the inverse square approximation is not a good approximation to the FWHM data. On the other hand, it is quite difficult to distinguish between the other three least-square fits to this data. To determine which - if any - of these fits is appropriate requires additional symbolic analysis.

```
>
```

Problems (pp. 114 - 116)

Problem 1

Plot the following functions on the specified domains. Select optional arguments so that the final plot clearly illustrates the interesting features of the function. Be certain to include labels and a title.

(a)

$f(x) = e^{(-(x-2)^2)} \sin(\pi x)$ and $g(x) = e^{(-(x-2)^2)} \sin(\pi x)^2$ for the first two periods of the trigonometric terms

(b)

Plot $u(x, y) = x \sin(y) - y \cos(x)$ for 0 <= x, y <= 4π

(c)

Plot $F(u, v) = \dfrac{x^2 - y^2}{x^2 + y^2}$ for -1 <= x, y <= 1

Note

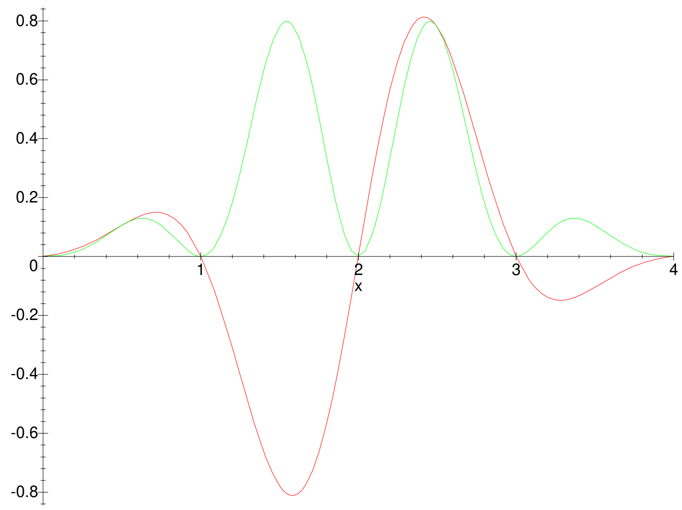
Look at the contour lines. What is the interesting point for this function?

(d)

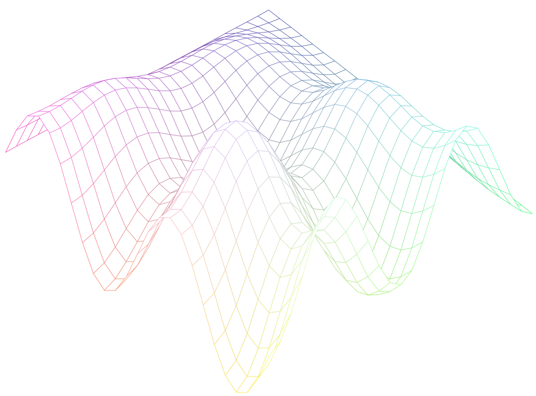
Plot $v_1(\theta) = \dfrac{\sin(\theta)}{\theta}$ and $v_2(\theta) = \dfrac{1 - \cos(\theta)}{\theta}$ for $|\theta| \le 5\pi$.

Solution

```
> restart;
```
(a) The trigonometric part of each function has period 2.
```
> plot( [ exp(-(x-2)^2) * sin(Pi*x), exp(-(x-2)^2) * sin(Pi*x)^2 ],
>       x=0..4, title=`Problem 1a` );
```
Problem 1a

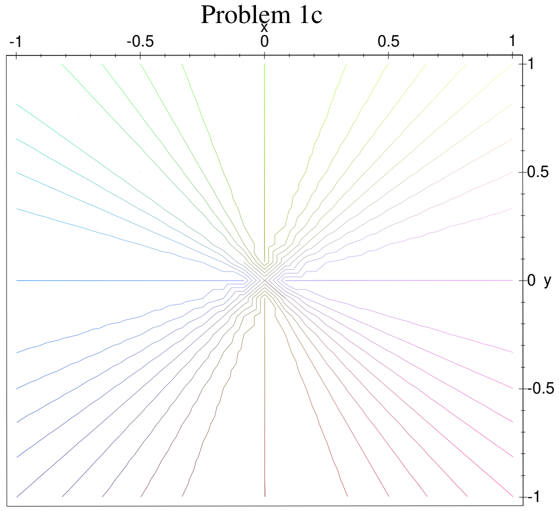



```
>
```
(b) This is a straight-forward 3D plot
```
> plot3d( x*sin(y)- y*cos(x), x=0..4*Pi, y=0..4*Pi,
>         title=`Problem 1b` );
```
Problem 1b



```
>
```
(c) The contour lines can be added via the GUI, or explicitly as an argument to the plot3d command.
```
> plot3d( (x^2-y^2)/(x^2+y^2), x=-1..1, y=-1..1, style=CONTOUR,
>         orientation=[-90,0], axes=BOXED, title=`Problem 1c` );
```

Problem 1c



Note that all contour lines are straight lines through the origin. Well, none of the contour lines actually includes the point (0,0). In general, contour lines for different level curves cannot cross. (In this case, the origin is not in the domain of the function. The fact that different limit are obtained along different lines means that this function is discontinuous at the origin and that this discontinuity cannot be "removed" by giving the special value at the origin.)

```
>
```

(d) There is nothing fancy about this one.

```
> plot( [ sin(x)/x, (1-cos(x))/x ], x=-5*Pi..5*Pi,
>        title=`Problem 1d` );
>
```

Problem 2

Describe the closed curve defined implicitly by

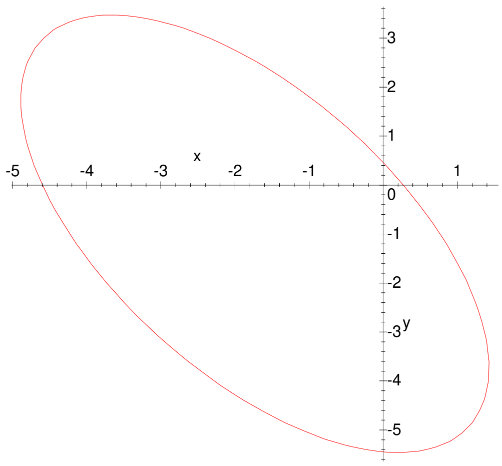
$$4\,x^2+2\sqrt{3}\,x\,y+2\,y^2+10\sqrt{3}\,x+10\,y=5.$$

Is the origin (0,0) inside the region bounded by this curve?

Solution

```
> restart; with(plots):
> implicitplot(4*x^2 + 2*sqrt(3)*x*y + 2*y^2 + 10*sqrt(3)*x + 10*y =5,
>              x=-5..2, y=-6..4, title=`Problem 2` );
```

Problem 2



This curve is an ellipse whose interior includes the origin. The foci and radii can be determined via standard calculus techniques (details omitted).

\>

Problem 3

Determine, both graphically and analytically, the percent error in the reflectivity that is needed to ensure that the finesse is controlled to within (plus or minus) 3% of 50.

Solution

```
> restart;
```

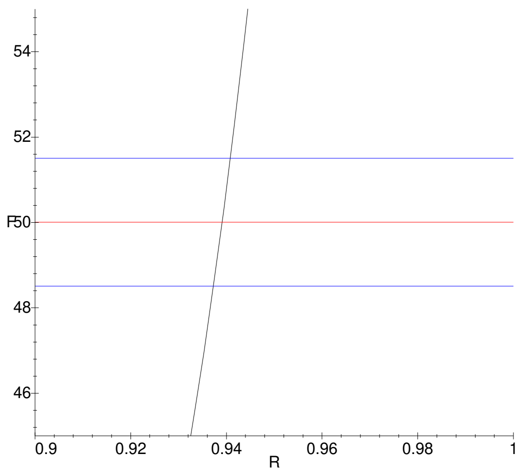
Recall (p. 88) that the finesse is defined to be

```
> F := Pi*sqrt(R)/(1-R);
```

$$F := \frac{\pi\sqrt{R}}{1-R}$$

To obtain a graphical answer to this problem, a plot (which is not shown here) can be used to determine that the reflectivity will be between 0.9 and 1. Then, to facilitate the error analysis, draw the target level ($F=50$) together with "error bars" at 3% ($F=48.5$ and $F=51.5$).

```
> plot( [F,48.5,50,51.5], R=0.9..1, 'F'=45..55,
>       color=[BLACK,BLUE,RED,BLUE],
>       title=`Finesse vs. Reflectivity ( F=50 +- 3% )` );
```



From this graph, the finesse is 50 when the reflectivity is approximately 0.9392, F=48.5 when R=0.9374, and F=51.5 when R=0.9409.

\>

The numerical solution is based on the use of fsolve to find approximate solutions to the same three equations:

```
> R50 := fsolve( F=50, R );
```

$$R50 := .9391110692$$

```
> R485 := fsolve( F=48.5, R );
```

$$R485 := .9372888368$$

```
> R515 := fsolve( F=51.5, R );
```

$$R515 := .9408304421$$

Note that the numerical and graphical data agree to three decimal digits.

The conversion of these results into percent can be done just as you would on a calculator

```
> 100*abs( R515 - R50 )/R50;
```

$$.1830851490$$

```
> 100*abs( R485 - R50 )/R50;
```

$$.1940380068$$

In summary, the reflectivity must be controlled to within 0.2% to achieve a 3% error in the finesse (when $F = 50$).

>

Problem 4

The ratio of light which passes through a filter to that which enters the filter is $T = \dfrac{1}{1 + \dfrac{4 F^2 \sin(\pi x)^2}{\pi^2}}$ where x is proportional to the frequency of the light and F is the finesse.

(a)

Plot the transmitted light for a filter with a finesse $F = 20$ and for 0.9 <= x <= 1.1.

Note that the transmission is greatest when $x = 1$ in this plot. More generally, $T = 1$ whenever x is an integer. The bandwidth of a filter is typically determined by its full width at half maximum (FWHM). That is, the maximum transmission coefficient is 1 (100%) when $x = 1$ and half of the signal is transmitted ($T = \frac{1}{2}$) when x=0.974 and again when x=1.03; the difference between these "frequencies" is the FWHM for $F = 20$.

(b)

Use the animate command to determine whether larger or smaller values of F are needed to produce a filter with a narrower bandwidth. To simplify the identification of the frequencies used to compute the FWHM, also plot the horizontal lines $T = \frac{1}{2}$ in each frame of the animation.

Solution

```
> restart; with(plots):
```

(a) We are told that the transmission coefficient (T) depends on the frequency (x) and finesse (F) in the following way

```
> T := 1/(1+4*F^2/Pi^2*sin(Pi*x)^2);
```

$$T := \frac{1}{1 + 4 \dfrac{F^2 \sin(\pi x)^2}{\pi^2}}$$

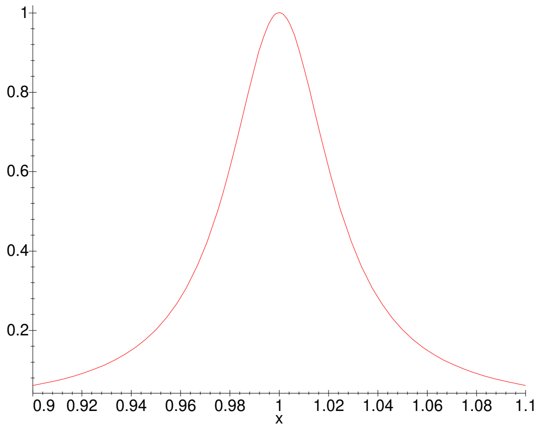
When the finesse is $F = 20$, the transmission coefficient is

```
> T20 := subs( F=20, T );
```

$$T20 := \frac{1}{1 + 1600 \dfrac{\sin(\pi x)^2}{\pi^2}}$$

```
> plot( T20, x=0.9 .. 1.1, title=`Transmitted Light (F=20)` );
```

Transmitted Light (F=20)

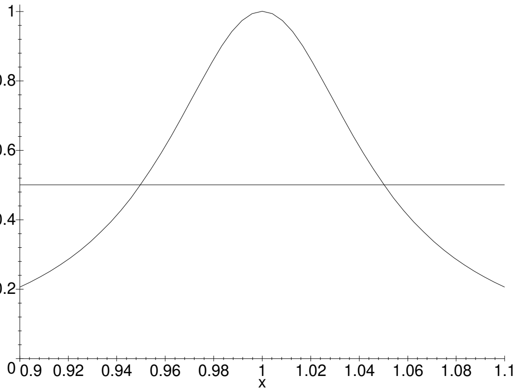



```
>
```

(b)

```
> animate( { T, 1/2 }, x = 0.9 .. 1.1, F = 10 .. 50,
>           title=`Transmitted Light (10 <= F <= 50)` );
```

Transmitted Light (10 <= F <= 50)



Note

The animate command will not accept a list of functions to be plotted. When more than one function is to be included in the display, they must be specified as a set.

This clearly illustrates that the FWHM decreases as the finesse increases.

```
>
```

Problem 5

(a)

Beginning with the optical filter transmission function $T(x, F)$ given in problem 4, use Maple to symbolically solve for the bandwidth (B) in terms of finesse (F). Does B change from peak to peak in this example?

(b)

Repeat (a) for the high-finesse case by replacing $\sin(\pi x)$ by πx in $T(x, F)$. What advantages are obtained from making this approximation? How small does x need to be to ensure the errors arising from this approximation are not too large.

Hint

Use a plot.

(c)

Plot B vs. F on the same graph using your results from (a) and (b). Over what range of F does the approximation incorporated in (b) have an error of less than 1%?

(d)

Plot T vs. x for $0<x<3$ for $0.5<F<50$ using the animate function. Do you see a problem with the bandwidth definition for low F?

Hint

might be more useful to look at the animation with decreasing values of F.

Solution

```
> restart; with(plots):
```

(a) As in Problem 4, the transmission coefficient is

```
> Tex := 1/(1+4*F^2/Pi^2*sin(Pi*x)^2);
```

$$Tex := \frac{1}{1+4\frac{F^2\sin(\pi x)^2}{\pi^2}}$$

The solve command can be used to find the frequencies when the transmission coefficient is half of its maximum value

```
> HMex := solve( Tex=1/2,x );
```

$$HMex := \frac{\arcsin\left(\frac{1}{2}\frac{\pi}{F}\right)}{\pi}, -\frac{\arcsin\left(\frac{1}{2}\frac{\pi}{F}\right)}{\pi}$$

As expected, Maple finds two solutions. The difference between these solutions is the FWHM.

```
> Bex := abs( HMex[1]-HMex[2] );
```

$$Bex := 2\frac{\left|\arcsin\left(\frac{1}{2}\frac{\pi}{F}\right)\right|}{\pi}$$

```
>
```

(b) When the transmission coefficient is approximated by replacing $\sin(\pi x)$ by πx,

```
> Tap := subs( sin(Pi*x)=Pi*x, Tex );
```

$$Tap := \frac{1}{1+4F^2x^2}$$

new expressions for the frequencies at which half of the signal is transmitted are obtained

```
> HMap := solve( Tap=1/2, x );
```

$$HMap := \frac{1}{2}\frac{1}{F}, -\frac{1}{2}\frac{1}{F}$$

In this case the FWHM is

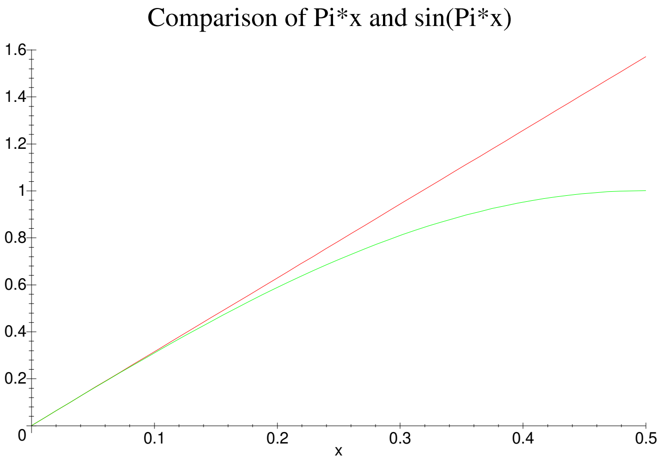
```
> Bap := abs( HMap[1]-HMap[2] );
```

$$Bap := \frac{1}{|F|}$$

This result is confirmation of the results found in Examples 4-11, 4-12, and 4-13 and Try It! (p. 112) in which the best least squares fit to the FWHM is found to be very close to $\frac{1}{F}$. However, remember that this result is obtained using an approximation to the transmission coefficient. The "exact" FWHM is the expression found in part (a) of this problem.

The investigation of the approximation introduced in this part of the problem can be conducted in any number of ways. However, to see exactly how the frequency (x) affects this approximation it makes most sense to look at the functions πx and $\sin(\pi x)$.

```
> plot( [ Pi*x, sin(Pi*x) ], x=0..0.5,
>       title=`Comparison of Pi*x and sin(Pi*x)` );
```

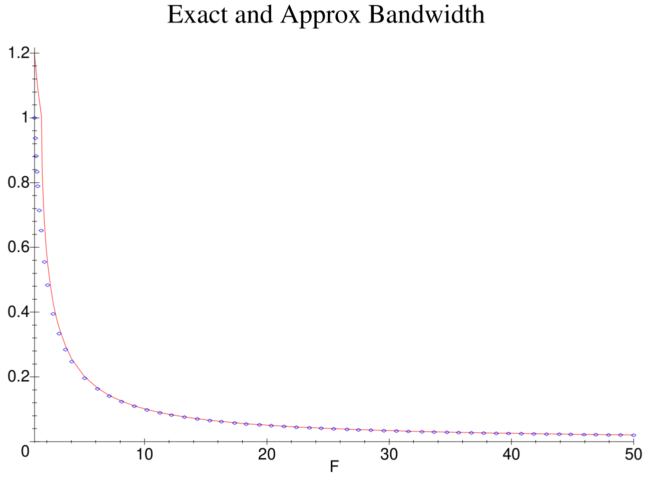


While additional information would need to be provided before saying exactly when this approximation is "good", it is reasonable to say that the functions are esssentially indistinguishable for $0 <= x <= 0.1$ and show significant differences for $x > 0.25$.

```
>
```

(c) The plot of the two expressions for the full-width at half maximum bandwidth is

```
> plot( [ Bex, Bap ], F = 1 .. 50, color=[RED,BLUE],
>        style=[LINE,POINT], title=`Exact and Approx Bandwidth` );
```



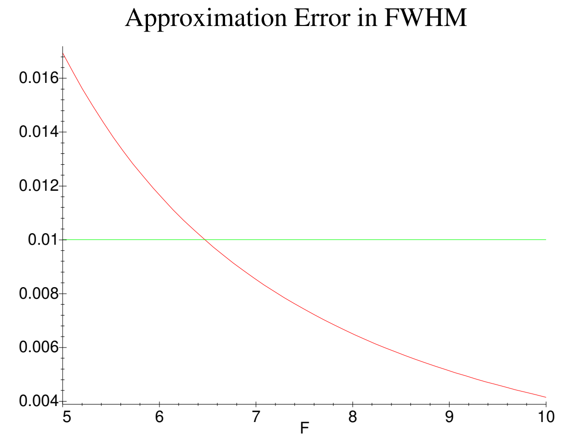
The relative error between the exact and approximate bandwidth is

```
> err := abs( (Bap - Bex) / Bex );
```

$$err := \frac{1}{2}\pi\left|\frac{\frac{1}{|F|} - 2\frac{\left|\arcsin\left(\frac{1}{2}\frac{\pi}{F}\right)\right|}{\pi}}{\left|\arcsin\left(\frac{1}{2}\frac{\pi}{F}\right)\right|}\right|$$

While this expression is rather complicated, it is not difficult to plot. The 1% tolerance is included to facilitate the interpretation of the graph.

```
> plot( [ err, 0.01 ], F = 5 .. 10,
>        title=`Approximation Error in FWHM` );
```

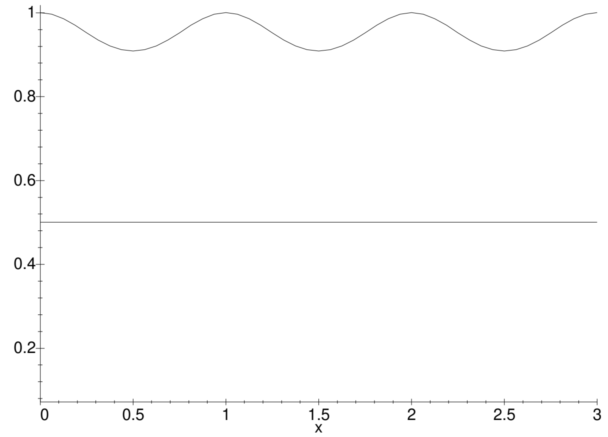


From this plot, it is seen that the two expressions for the bandwidth agree to within 1% when F > 6.44.

```
>
```

(d) The requested animation is

```
> animate( {Tex,1/2}, x=0..3, F=1/2..5 );
```



For low values of the finesse the transmission coefficient always exceeds half of the maximum. In these situations the full-width at half maximum is not defined.
(The specific cutoff for the finesse is not easily determined from the animiation. How could you determine the smallest finesse for which the FWHM bandwidth is defined?)

```
>
```

Problem 6

In Example 4-11 we (correctly) guessed that there is a reciprocal relationship between the finesse and bandwidth of a filter. A log-log plot can be used to obtain similar information about a set of data. The basic idea is that if $F=B^{\alpha}$, for some constant α, then $\log(F)=\alpha\log(B)$. That is, the graph of $\log(B)$ vs. $\log(F)$ will be a straight line with slope α.

Confirm the results of Example 4-13 by creating a log-log plot of the data points used in that example.

Solution

Correction

The reciprocal fit is first mentioned in Example 4-12.

```
> restart; with(plots):
```

Here is the data introduced in Example 4-11.

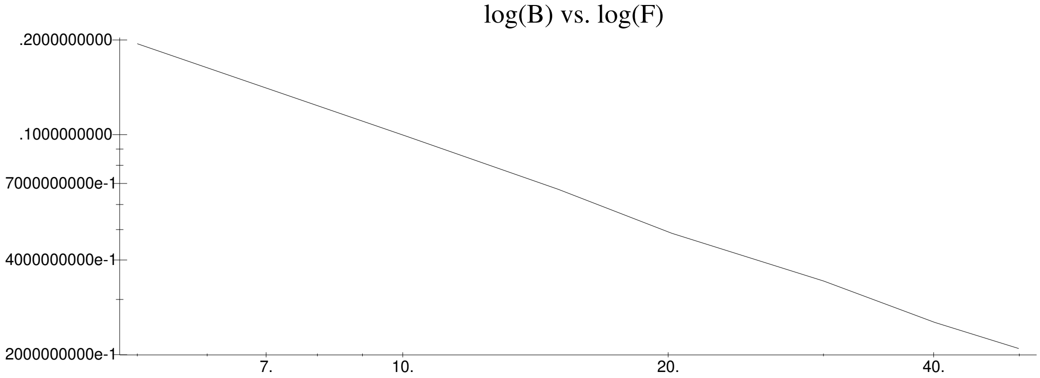
```
> Flist := [ 5, 10.1, 15.01, 20.19, 30.01, 40.11, 49.93 ]:
> Blist := [ 1.096-0.9016, 1.049-0.9502, 1.033-0.966, 1.023-0.9745,
  1.016-0.9818, 1.012-0.9867, 1.01-0.9891 ]:
> PTS := zip( (f,b) -> [f,b], Flist, Blist );
```

$PTS := [[5, .1944], [10.1, .0988], [15.01, .067], [20.19, .0485], [30.01, .0342], [40.11, .0253], [49.93, .0209]]$

```
>
```

A log-log plot is created using the loglogplot command from the plots package. The basic usage is

```
> loglogplot( PTS, title=`log(B) vs. log(F)` );
```



Before we say that this line appears to have slope close to -1, are you sure the same scaling is used for both axes? Three ways in which this can be enforced are i) to click on the 1:1 icon in the context bar, ii) to add the optional argument scaling=CONSTRAINED to the loglogplot command, and iii) to specify a view in which the range for each axis displays the same number of magnitudes of values. The last two options can be implemented using the following commands (plots omitted)

```
> loglogplot( PTS, axes=BOXED, scaling=CONSTRAINED,
>             title=`log(B) vs. log(F)` );
> loglogplot( PTS, axes=BOXED, view=[1..100,0.01..1],
>             title=`log(B) vs. log(F)` );
```

Regardless of the method used, it is quite apparent that the slope of the log-log plot is close to -1. That is, that the FWHM bandwidth and finesse are inversely proportional.

```
>
```

Problem 7

Perform a least-squares fit of the wind tunnel and CFD data in Application 3 (Chapter 3). Graph the data points, the lift-to-drag function found in Step 4, and the best-fit solution. How do these fits compare with the estimate found in Step 4 of the five-step solution? What are the drag coefficients and thrust requirements when C_L=0.5060?

Solution

Note: Problems 7 and 12

This problem is related to Problem 12. In fact, it is advisable to solve Problem 12 before solving this problem.

```
> restart; with(stats):
```

The wind tunnel and CFD data for Application 3 are found on p. 61.

```
> CLwt := [ 0.01, 0.13, 0.21, 0.40, 0.65 ];
> CDwt := [ 0.0155, 0.0165, 0.0181, 0.0249, 0.0380 ];
```

$CLwt := [.01, .13, .21, .40, .65]$

$CDwt := [.0155, .0165, .0181, .0249, .0380]$

```
> CLcfd := [ 0.00, 0.11, 0.25, 0.38, 0.51, 0.66, 0.80 ];
> CDcfd := [ 0.0157, 0.0163, 0.0187, 0.0228, 0.0293, 0.0389, 0.0501 ];
```

$CLcfd := [0, .11, .25, .38, .51, .66, .80]$

$CDcfd := [.0157, .0163, .0187, .0228, .0293, .0389, .0501]$

The specific lift-to-drag function found in Step 4 of that application was found, on p. 65, in the following manner

```
> liftdrag := CD = CD0 + alpha*CL^2;
```

$$liftdrag := CD = CD0 + \alpha\, CL^2$$

```
> eq1 := subs( [ CL=0.40, CD=0.0249 ], liftdrag );
> eq2 := subs( [ CL=0.65, CD=0.0380 ], liftdrag );
```

$$eq1 := .0249 = CD0 + .1600\,\alpha$$

$$eq2 := .0380 = CD0 + .4225\,\alpha$$

```
> LDcoef := solve( { eq1, eq2 }, { CD0, alpha } );
```

$$LDcoef := \{ \alpha = .04990476190, CD0 = .01691523810 \}$$

```
> LtoD := evalf( subs( LDcoef, liftdrag ), 3 );
```

$$LtoD := CD = .0169 + .0499\, CL^2$$

```
>
```

The least-squares fits look for functions of the form $C_D = C_{D_0} + \alpha\, C_L^{\ 2}$. For the wind tunnel data, the resulting relationship is

```
> LtoDwt := fit[leastsquare[ {CD,CL}, liftdrag, {CD0,alpha} ] ]
>              ( [ CDwt, CLwt ] );
```

$$LtoDwt := CD = .01574954267 + .05321983634\, CL^2$$

The corresponding best fit for the CFD data is

```
> LtoDcfd := fit[leastsquare[ {CD,CL}, liftdrag, {CD0,alpha} ] ]
>               ( [ CDcfd, CLcfd ] );
```

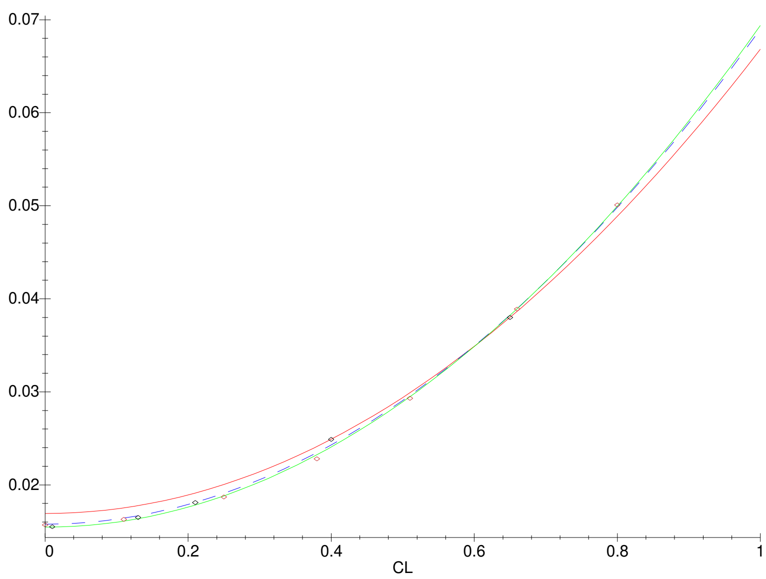
$$LtoDcfd := CD = .01542412816 + .05392107985\, CL^2$$

```
>
```

Observe that all three approximate lift-to-drag functions are quite similar. To compare these fits with the data, a composite plot containing the data and the three functions is created. (See also Problem 12.)

```
> PTSwt := zip( (x,y)->[x,y], CLwt, CDwt ):
> PTScfd := zip( (x,y)->[x,y], CLcfd, CDcfd ):
> plot( [ PTSwt, PTScfd, rhs(LtoD), rhs(LtoDwt), rhs(LtoDcfd) ], CL = 0 .. 1,
>       color=[BLACK,ORANGE,RED,BLUE,GREEN],
>       style=[POINT,POINT,LINE,LINE,LINE],
>       linestyle=[0,0,1,3,5],
>       title=`Lift-to-Drag Data and Fitted Functions` );
```

Lift-to-Drag Data and Fitted Functions



This plot shows that each of the approximations does a good job of fitting the data but that the two least-squares fits do a better job of approximating the data over the entire range of C_L values.

>

The specific coefficients of drag produced when $C_L = .5060$ are

```
> subs( CL=0.5060, LtoD );
```

$$CD = .02967619640$$

```
> subs( CL=0.5060, LtoDwt );
```

$$CD = .02937573669$$

```
> subs( CL=0.5060, LtoDcfd );
```

$$CD = .02922986576$$

Once again, all three results are fairly close.

>

To determine the thrust for each approximation requires some additional parameter definitions from Application 3 (pp. 63 - 65).

```
> PARAM := evalf( [ w = 500000, b = 200, AR = 10, M = 0.84, gamma1 = 1.4, p0 =
  14.696*12^2, delta = 0.2360, rho0 = 0.002377, sigma = 0.3106 ], 3 ):
> VARS := [ weight=w, V = M*a, S = b^2/AR, rho=sigma*rho0 ]:
> Vsound := subs( [p=delta*p0, rho=sigma*rho0], a=sqrt(p/rho*gamma1) ):
> VARS := subs( Vsound, VARS ):
> drag := rho*V^2*S*CD/2:
> balance2 := thrust=drag;
```

$$balance2 := thrust = \frac{1}{2} \rho V^2 S\, CD$$

Now, at long last, the thrust estimates for the original approximation and the least-square approximation based on the wind tunnel and CFD data are

```
> subs( VARS, PARAM, LtoD, CL=0.5060, balance2 );
```

$$thrust = 29334.09567$$

```
> subs( VARS, PARAM, LtoDwt, CL=0.5060, balance2 );
```

$$thrust = 29037.09959$$

```
> subs( VARS, PARAM, LtoDcfd, CL=0.5060, balance2 );
```

$$thrust = 28892.91023$$

The agreement between these results provides confidence in our original estimate.

>

Problem 8

Use contourplot (or contourplot3d) to plot level curves of the following two equations

$$\left(u - \frac{r}{1+r}\right)^2 + v^2 = \frac{1}{(1+r)^2}$$

$$(u-1)^2 + \left(v - \frac{1}{x}\right)^2 = \frac{1}{x^2}$$

Plot lines having constant r=0, 0.5, 1, 2, 5, 10 in the $u-v$ plane. Do the same for lines having constant x=0, +0.5, -0.5, +1, -1, +2, -2, +5, -5, +10, and -10. This type of plot, called a Smith chart, is used by engineers to describe more complicated relationships between various quantities in microwave engineering. For example, they may describe a transformation between reflection coefficient and normalized impedance in a coaxial cable being used as a transmission line by electrical engineers analyzing a communication channel.

Solution

```
> restart; with(plots):
```

The two equations of interest in this problem are

```
> E1 := (u-r/(1+r))^2 + v^2 = 1/(1+r)^2;
> E2 := (u-1)^2+(v-1/x)^2 = 1/x^2;
```

$$E1 := \left(u - \frac{r}{1+r}\right)^2 + v^2 = \frac{1}{(1+r)^2}$$

$$E2 := (u-1)^2 + \left(v - \frac{1}{x}\right)^2 = \frac{1}{x^2}$$

We aren't really interested in "contour lines" of the above functions. Rather, we need to plot the implicitly defined functions that are obtained when the specified values of r and x are substituted into these equations. This could be achieved with the use of implicitplot, but contourplot seems the more reasonable choice.

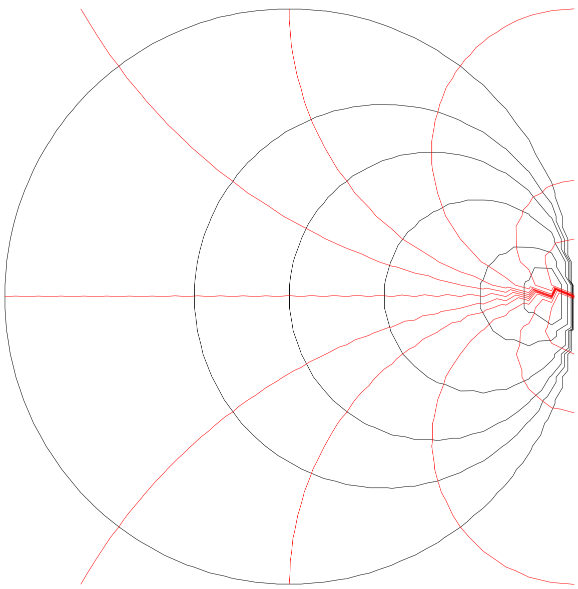
```
> R := solve( E1, r );
> X := solve( E2, x );
```

$$R := -\frac{u^2 - 1 + v^2}{-2\,u + u^2 + v^2 + 1}$$

$$X := 2\,\frac{v}{-2\,u + u^2 + v^2 + 1}$$

```
> C1 := contourplot( R, u=-1..1, v=-1..1, color=BLACK,
>                    contours=[0,1/2,1,2,5,10] ):
> C2 := contourplot( X, u=-1..1, v=-1..1, color=RED,
>                    contours=[0,1/2,-1/2,1,-1,2,-2,5,-5,10,-10] ):
> display( {C1,C2}, title=`Smith Chart (Problem 7)`, axes=NONE );
```

Smith Chart (Problem 7)



The quality of this plot can be improved by including the optional argument grid=[50,50] in each contourplot command.

```
>
```

Problem 9

The textplot (and textplot3d) commands (from the plots package) can be used to insert labels in a two- or three-dimensional plot. Find the online help for these commands, then use them to identify the two curves plotted in Example 4-2.

Solution

```
> restart; with(plots):
```

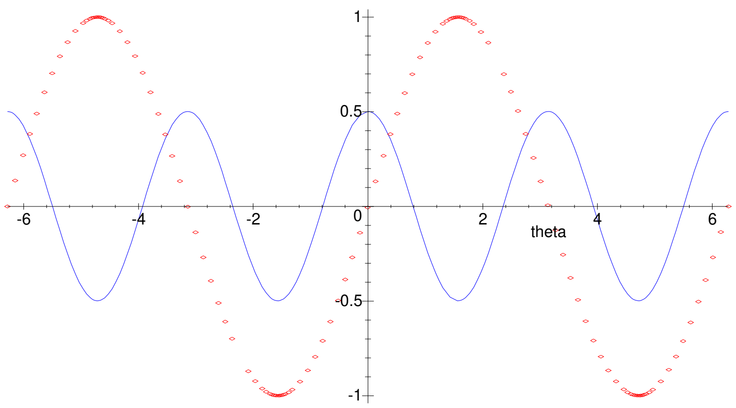
The two functions and the original plot from Example 4-2 (p. 80) are

```
> v1 := sin( theta ) ;
> v2 := theta -> cos(2*theta) / 2 ;
```

$$v1 := \sin(\theta)$$

$$v2 := \theta \rightarrow \frac{1}{2}\cos(2\,\theta)$$

```
> P := plot( [ v1, v2(theta) ], theta = -2*Pi .. 2*Pi,
>             color=[RED,BLUE], style=[POINT,LINE] ):
> P;
```

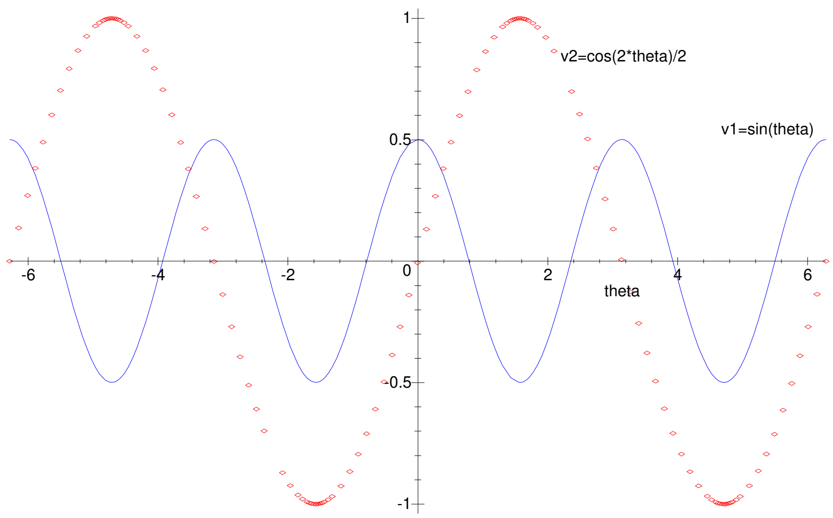



```
>
```

From the online help for textplot, it is learned that the following commands can be used to label the individual functions in the plot. (The coordinates of the messages were determined using the GUI.)

```
> T1 := textplot( [6.1,0.5,`v1=sin(theta)`], align={ABOVE,LEFT} ):
> T2 := textplot( [2.2,0.8,`v2=cos(2*theta)/2`], align={ABOVE,RIGHT} ):
> display( {P,T1,T2}, title=`Example 4-2 - with text labels` );
```

Example 4-2 - with text labels




```
>
```

Problem 10

Consider any periodic function of time f with period T, returning once each cycle to any selected time reference. In other words, $f(t+T)=f(t)$, where T is called the period of the periodic function. This period is the minimum time it takes the function to duplicate itself. Such a function may be represented by a group of purely sinusoidal functions, consisting of a fundamental frequency and its harmonics. Fourier analysis allows us to find the ''weighted'' coefficients of each of the sinusoidal terms in such a way that, after adding them together, they approximate the original periodic function. For example, let $\theta = \frac{2\pi t}{T}$, where t is the time and T is the period; θ is in radians. Define

$$f_{2n-1}(\theta) = \frac{1}{2} + \frac{2\left(\sum_{k=1}^{n} \frac{(-1)^k \cos((2k-1)\theta)}{2k-1}\right)}{\pi} \quad \text{for n=1,2,....}$$

Graph, in one plot, the functions f_1, f_3, and f_5 for at least two periods. Then plot f_{25} on a separate plot. What periodic function is being represented by this group of sinusoidal functions?

Solution

```
> restart; with(plots):
```

The sequence of functions can be defined in a number of different ways. Be careful about how the index is implemented. In the following definition the upper limit of the summation is modified so that f_m can be obtained via, e.g., subs(n=m, f) when m is an odd integer.

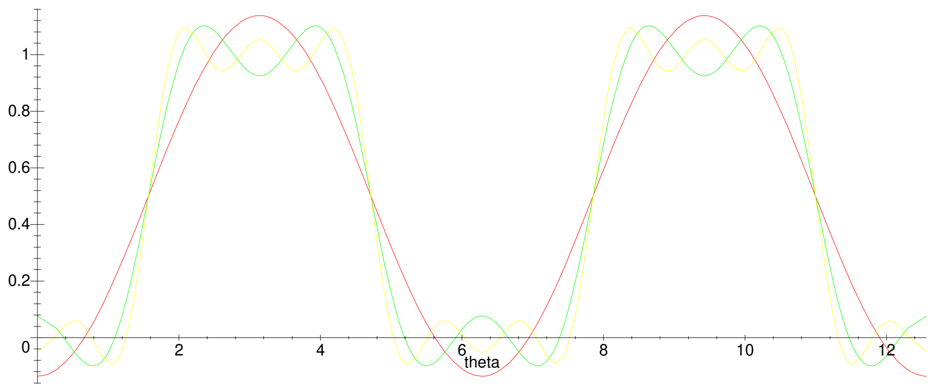
```
> f := 1/2 + 2/Pi*Sum((-1)^k*cos((2*k-1)*theta)/(2*k-1),k=1..(n+1)/2);
```

$$f := \frac{1}{2} + 2\,\frac{\sum_{k=1}^{1/2\,n+1/2} \frac{(-1)^k \cos((2k-1)\theta)}{2k-1}}{\pi}$$

Thus, the functions corresponding to f_1, f_3, and f_5 can be plotted with the single command

```
> plot( [ seq( f, n=[1,3,5] ) ], theta=0..4*Pi,
>       title=`Plot of f[1], f[3], and f[5]` );
```

Plot of f[1], f[3], and f[5]



Note that the explicit use of subs is more cumbersome - and becomes more so as the number of functions to be plotted increases.

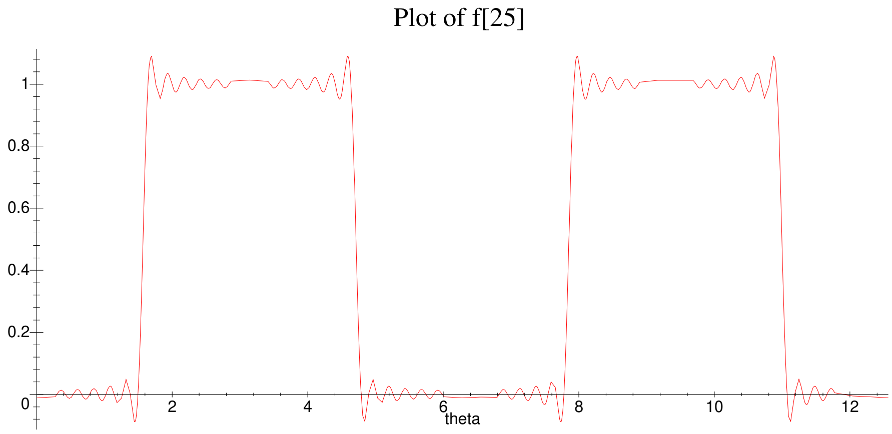
```
>
```

The plot of the single function f_{25} is simpler, but is useful in our attempt to understand the approximation properties of this sequence of functions.

```
> f25 := subs( n=25, f );
```

$$f25 := \frac{1}{2} + 2\,\frac{\sum_{k=1}^{13} \frac{(-1)^k \cos((2k-1)\theta)}{2k-1}}{\pi}$$

```
> plot( f25, theta=0..4*Pi, title=`Plot of f[25]` );
```



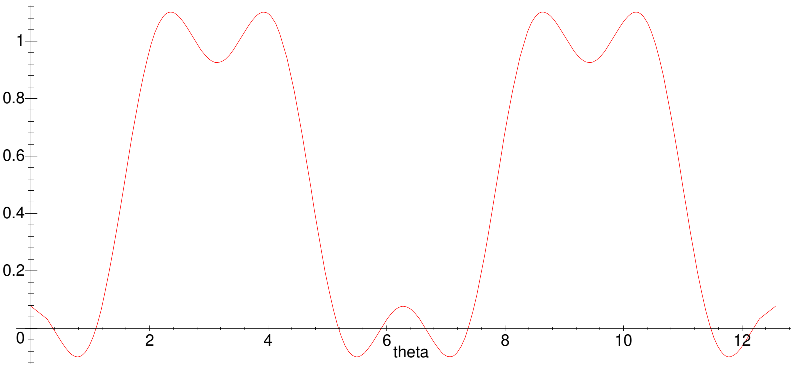

```
>
```

From this plot it appears that these functions are somehow related to the "unit step function". That is, the periodic function with period $T = 2\pi$ that is zero for $\theta < \frac{\pi}{2}$, one for $\frac{\pi}{2} < \theta < \frac{3\pi}{2}$, then zero again for $\frac{3\pi}{2} < \theta < 2\pi$. (Observe that defining the function for a single interval of length 2π defines the function for the entire real line.) A different view of this fact can be seen in an animated display of the functions as the index increases.

```
> index := [ seq( 2*n+1, n=1..10 ) ];
```

$$index := [3, 5, 7, 9, 11, 13, 15, 17, 19, 21]$$

```
> display( [ seq( plot( f, theta=0..4*Pi ), n=index ) ],
>           insequence=true );
```



Note

The oscillations that occur at the "jumps" in the function are due to the oscillatory nature of the Fourier series. This phenomenon is called the "Gibbs effect".

```
>
```

Problem 11

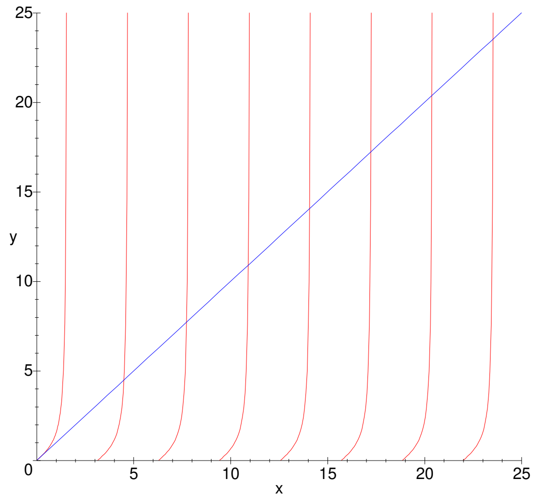
Plot the functions $y = \tan(x)$ and $y = x$ on a single graph. Be sure to choose the domain so that the graph contains the first five positive values of x for which $\tan(x) = x$. Use the interface to identify the first five positive values of x for which $\tan(x) = x$. Compare these results with the values found in Example 3-17.

Solution

```
> restart;
```

The graph can be obtained by the following command.

```
> plot( [ tan(x), x ], x=0..25, y=0..25,
>        color=[RED,BLUE], discont=true,
>        title=`Graphs of y=tan(x) and y=x` );
```



Note

Observe that, for all intents and purposes, this plot is identical to the one on p. 71 of the text.

```
>
```

The approximate location of the first five positive intersections of these two functions are: 4.55, 7.74, 10.93, 14.12, and 17.26.

Note

The specific value you obtain will be somewhat dependent on your specific system. However, the values all agree with the solutions found in Example 3-17 to at least two significant digits.

Problem 12

Create, in one plot, a graph of the CFD data from Application 3 (Chapter 3) and the quadratic function relating the coefficients of lift and drag that is given in Step 4.

Solution

Note: Problems 7 and 12

This problem is related to part of Problem 7. In fact, it is advisable to solve this problem before solving Problem 7.

```
> restart; with(plots):
```

The CFD data is obtained from p. 61.

```
> CLcfd := [ 0.00, 0.11, 0.25, 0.38, 0.51, 0.66, 0.80 ]:
> CDcfd := [ 0.0157, 0.0163, 0.0187, 0.0228, 0.0293, 0.0389, 0.0501 ]:
> PTScfd := zip( (x,y)->[x,y], CLcfd, CDcfd );
```

$$PTScfd := [[0, .0157], [.11, .0163], [.25, .0187], [.38, .0228], [.51, .0293], [.66, .0389], [.80, .0501]]$$

```
>
```

The original (two-point) quadratic fit is found on pp. 65 -- 66

```
> liftdrag := CD = CD0 + alpha*CL^2;
```

$$liftdrag := CD = CD0 + \alpha\, CL^2$$

```
> eq1 := subs( [ CL=0.40, CD=0.0249 ], liftdrag );
> eq2 := subs( [ CL=0.65, CD=0.0380 ], liftdrag );
```

$$eq1 := .0249 = CD0 + .1600\,\alpha$$

$$eq2 := .0380 = CD0 + .4225\,\alpha$$

```
> LDcoef := solve( { eq1, eq2 }, { CD0, alpha } );
```

$$LDcoef := \{ CD0 = .01691523810, \alpha = .04990476190 \}$$

```
> LtoD := evalf( subs( LDcoef, liftdrag ), 3 );
```

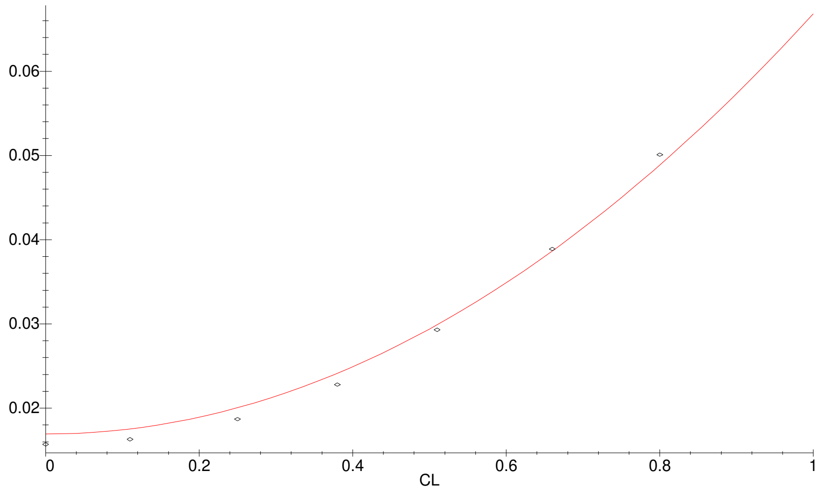
$$LtoD := CD = .0169 + .0499\, CL^2$$

```
>
```

These two pieces of information can be displayed in the following way

```
> plot( [ PTScfd, rhs(LtoD) ], CL = 0 .. 1,
>       style=[POINT,LINE],
>       color=[BLACK,RED],
>       title=`CFD Data and Quadratic Approximation to Lift-to-Drag Function` );
```

CFD Data and Quadratic Approximation to Lift-to-Drag Function




```
>
```

Problem 13

(a)

The curve $x^2 = y^2 (1-y)$ was plotted as an implicitly defined function and parametrically in Section 4-2. The points with $t = \sqrt{2}$ and $t = -\sqrt{2}$ correspond to the two endpoints of the curve. Find the value(s) of the parameter t when the curve passes through the origin.

(b)

Use animate to show how the curve is traced out as t increases from $-\sqrt{2}$ to $\sqrt{2}$.

Solution

```
> restart; with(plots):
```

(a) The parametric representation of this function introduced in Section 4-2 (p. 96) is

```
> EQS := [ y=1-t^2, x=t*(1-t^2) ];
```

$$EQS := [y = 1 - t^2, x = t\,(1 - t^2)]$$

To determine the parameter values when the curve passes through the origin, simply substitute $x = 0$ and $y = 0$ into the equations and solve for the parameter value. (Note that the syntax for solve requires that the equations be converted into a set.)

```
> solve( convert( subs(x=0,y=0,EQS), set ), t );
```

$$\{t = 1\}, \{t = -1\}$$

Two solutions are returned by solve. It is simple to verify that these are correct (and not much more difficult to see that these are the only possible values for which the curve passes through the origin).

```
>
```

(b) The "natural" solution would be something like

```
> PARAM := subs( EQS, [x,y,t=-sqrt(2) .. T ] );
```

$$PARAM := [t\,(1 - t^2), 1 - t^2, t = -\sqrt{2} .. T]$$

```
> animate( PARAM, T=-sqrt(2)..sqrt(2) );
```

However, this does not work. The problem is that all ranges must be real constants (and not parameters). This makes the solution of this problem much more complicated.

As a first attempt, use the signum function to "cut off" the functions when the curve parameter (t) exceeds the animation parameter (T).

```
> PARAM2 := [ PARAM[1]*signum(t<T),
>             PARAM[2]*signum(t<T),
>             t=-sqrt(2) .. sqrt(2) ];
```

$$PARAM2 := [t\,(1-t^2)\,\mathrm{signum}(t<T),\,(1-t^2)\,\mathrm{signum}(t<T),\,t=-\sqrt{2}\,..\,\sqrt{2}]$$

Note: signum and sign

For this usage, the sign function could be used in place of the signum function. In general, signum is for expressions and sign for polynomials.

Now, the natural animate command leads produces an animation.

```
> animate( PARAM2, T=-sqrt(2)..sqrt(2), color=RED,
>          title=`Animation with extra artifact` );
```

The only problem with this solution is that the last point on the curve is always connected back to the origin. This effect may be useful in some circumstances, but let's see if it can be eliminated.

```
>
```

The extra artifact produced by the previous approach can be avoided by defining the parametric curve so that it evaluates to FAIL whenever the animation parameter exceeds the curve parameter. This can be accomplished using the piecewise command.

```
> PARAM3 := [ piecewise( t<T, PARAM[1], FAIL),
>             piecewise( t<T,PARAM[2], FAIL),
>             t=-sqrt(2) .. sqrt(2) ];
```

$$PARAM3 := \left[\begin{cases} t\,(1-t^2) & t<T \\ FAIL & otherwise \end{cases},\ \begin{cases} 1-t^2 & t<T \\ FAIL & otherwise \end{cases},\ t=-\sqrt{2}\,..\,\sqrt{2}\right]$$

```
> animate( PARAM3, T=-sqrt(2)..sqrt(2), color=RED,
>          title=`Animation - Final!` );
>
```

Note

All plots are omitted from this presentation as they are animations for which the first frame is essentially empty.

Problem 14

Verify that the gradient and normal fields for $V=\sqrt{x^2+y^2+4}$ are orthogonal by superimposing the plots of the two vector fields on top of one another. (Use different colors to distinguish the two vector fields.)

Solution

```
> restart; with(plots):
```

The definitions of the vector field together with its gradient and normal fields can be defined in the text (pp. 96 -- 97).

```
> V := sqrt( x^2+y^2+4 );
```

$$V := \sqrt{x^2+y^2+4}$$

```
> gradV := [ diff(V,x), diff(V,y) ];
```

$$gradV := \left[\frac{x}{\sqrt{x^2+y^2+4}},\ \frac{y}{\sqrt{x^2+y^2+4}}\right]$$

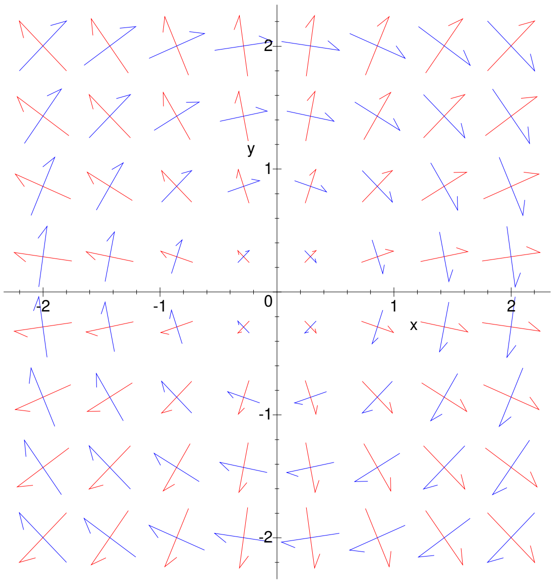
```
> normV := [ gradV[2], -gradV[1] ];
```

$$normV := \left[\frac{y}{\sqrt{x^2+y^2+4}},\ -\frac{x}{\sqrt{x^2+y^2+4}}\right]$$

Only a few minor changes have been made to the optional arguments, otherwise these are the same commands that were used in the discussion in the text.

```
> G := fieldplot( gradV, x=-2..2, y=-2..2, grid=[8,8], color=RED ):
> N := fieldplot( normV, x=-2..2, y=-2..2, grid=[8,8], color=BLUE ):
> display([ G, N ], title=`Gradient and Normal Fields are Orthogonal`);
```

Gradient and Normal Fields are Orthogonal



The use of the same grid in both plots makes the orthogonality of these vector fields particularly easy to see.

>