Parallel Computing Notes
Topic: "Wave" Generator Using Matrix-Matrix Multiplication of Airy Disk Function

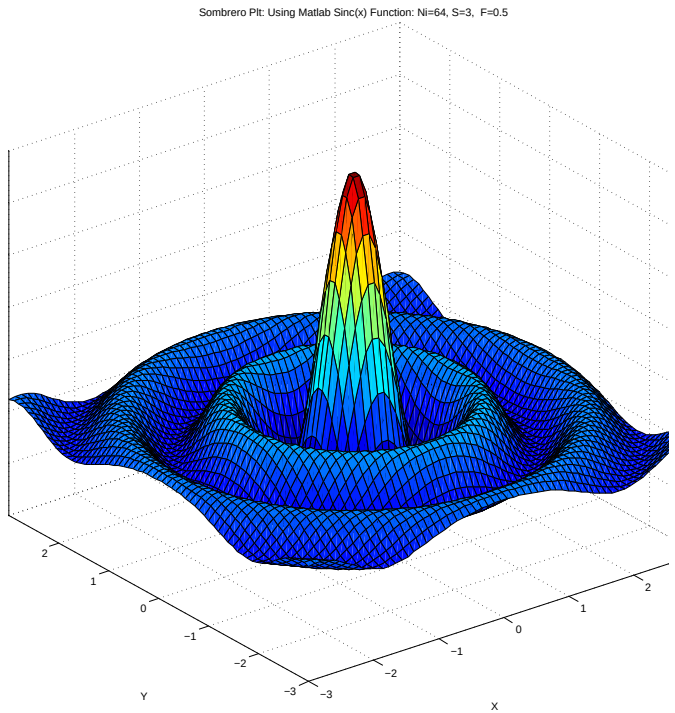
Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Last Update: 11/01/15

Table of Contents

Calculating Bessel Functions Using Matrix-Matrix Multiplication

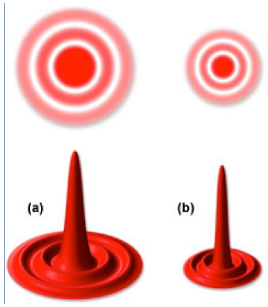


Plot of a 2D, rotationally symmetric Bessel function.

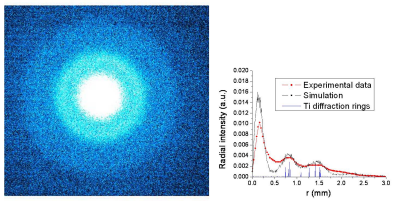
In these notes, we use a visual example to develop a parallel matrix matrix multiplication application and visualize the correct results.

Radially symmetric phenomena in Nature

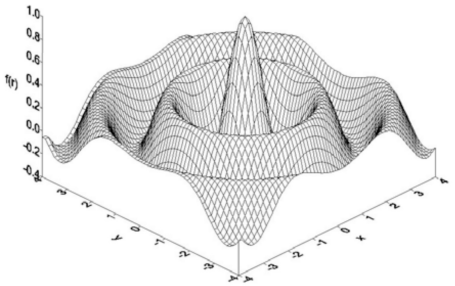
- Diffraction of light through a small, circular aperture (top image).
- Particle diffraction: at small scales, particles behave like light photons and can be scattered by nuclei (bottom).
- Signal Processing: A square wave is composed of all frequencies and a cylinder is a 2D rotation of a square wave: the Fourier transform of a cylinder is the *Circ* function (bottom image).
- Other phenomena that are wave like:
  - water waves when a rock is dropped
  - sounds waves on a snare drum
  - mechanical vibrations



Airy disk pattern made by light passing through a small aperture.



Electron Diffraction from Titanium foil).



2D rotationally symmetric *Circ* function.

REFs: `http://en.wikipedia.org/wiki/Sombrero_function`
`http://www.cis.rit.edu/class/simg716/handouts/06_notes_2Dfunctions.pdf`

Radially symmetric functions used to represent nature

- Geometry & Trigonometry: e.g. Newtons rings (light wave reflection pattern).
- Wave theory: e.g. the Point spread function and Xernike polynomials.
- Huyen-Fresnel Principle: e.g. Diffraction of light through a small, circular aperture
- Bessel Functions: solutions y(x) of Bessel's differential equation.
  - Fourier transform of a cylinder $\rightarrow$ *Circ* function
  - Using Gaussian profile to approximate the Airy Disk
  - Electromagnetic waves in a waveguide
  - many other applications
- $sinc(x) = sin(x)/x$ or "Sombrero" function – approximates some Bessel functions

http://www.citycollegiate.com/newtons_rings.htm
http://en.wikipedia.org/wiki/Point_spread_function
http://en.wikipedia.org/wiki/Zernike_polynomials
http://en.wikipedia.org/wiki/Huygens-Fresnel_principle
http://en.wikipedia.org/wiki/Bessel_function
http://www.cis.rit.edu/class/simg716/handouts/06_notes_2Dfunctions.pdf
http://en.wikipedia.org/wiki/Airy_disk

"Sombrero" Shaped Function: $sinc(x) = sin(x)/x$

Very easy to do in Matlab! More challenging in other languages.

- Matlab has a special function for $sinc(x)$:
- Works for 1D, 2D, 3D function.
- $sinc(x) = sin(x)/x$: has special internals to handle singularities at x=0
- Can be used to create a sombrero like plot.
- $sinc(x)$ is difficult to code in C because of divide by zero issues: need to avoid $(x, y) = (0, 0)$.

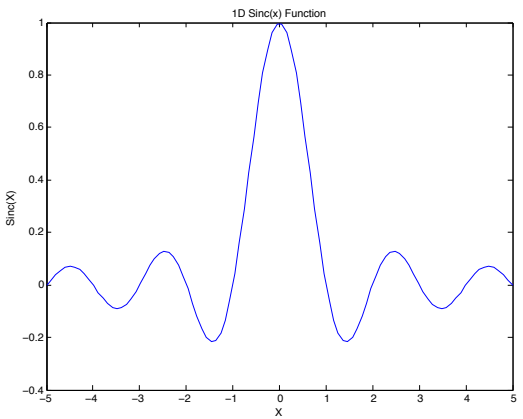
`http://www.mathworks.com/matlabcentral/cody/problems/1305-creation-of-2d-sinc-surface`

`http://www.mathworks.com/help/signal/ref/sinc.html`

Matlab code to used generate 1D $sinc(x)$ function

- $sinc(x) = sin(x)/x$ works in the example below for y=0: sinc(0)=sin(0)/0= NaN, but plotting program will ignore this.

```
t = linspace(-5,5);
y = sinc(t);
plot(t,y);
title(['1D Sinc(x) Function']);
xlabel('X');
ylabel('Sinc(X)');
```



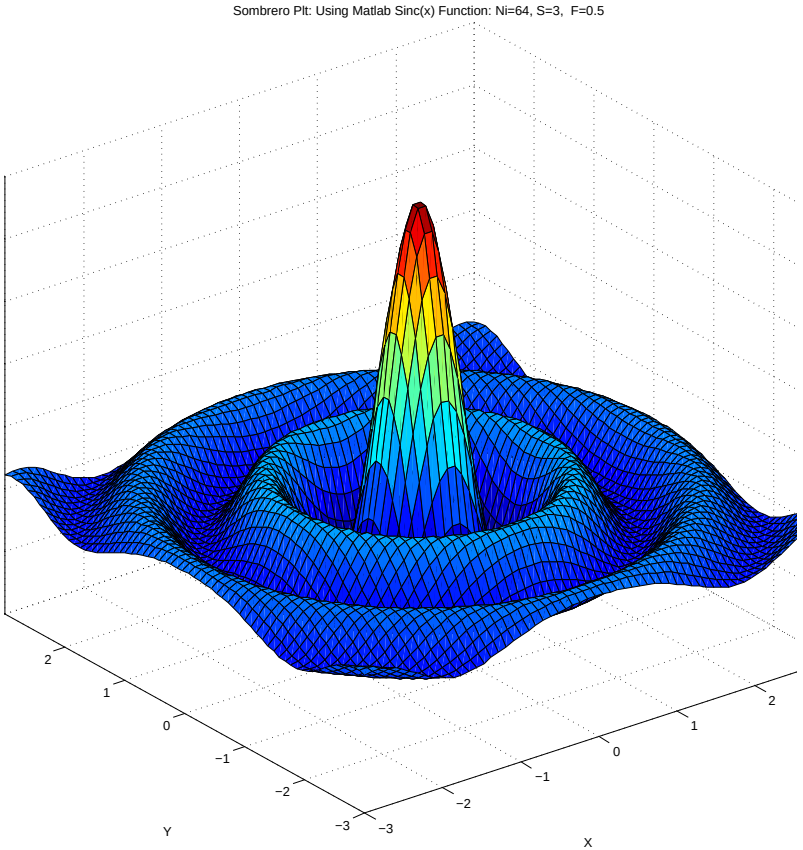
Plot of Matlab 1D $sinc(x)$ function.

REF: http://www.mathworks.com/matlabcentral/cody/problems/1305-creation-of-2d-sinc-surface

Matlab code to generate 2D $sinc(x)$ Sombrero

```
%
% Sombrero Test Case: Matlab code
%   Calculate sinc(x) using cartesian coordinates
%
%   By Mary Thomas
%       updated:  March, 2015
%       Created: March, 2014)
%
%  note: ni does have to equal nj
%  scaling factors affect max amplitude
%  number of wavelengths
%
%  Sinc Function:
%   http://en.wikipedia.org/wiki/Sombrero_function
%
clear all;
fprintf('Begin calc cartesian jinc(x) sombrero\n');

%%
% simple plot
%t = linspace(-5,5);


%%
% cartesian representation
ni=51;
s=8;
f=0.25;
g= .25;
%s=2;
x = linspace(-s , s ,ni);
y = linspace(-s , s ,ni);

%% 1D Sinc(x)
sinc1D = sinc(x);

%%  2D Sinc(x,y)
for i=1:ni
    for j=1:ni
        r=sqrt(x(i)^2 + y(j)^2);
        F(i,j) = sin(r)/r;

        xpts(i,j) = x(i); % these are for plotting
        ypts(i,j) = y(j);
    end
end
fprintf('Calc of f(x,y) completed. \n');

%%
figure;
plot(x,sinc1D);
title(['1D Sinc(x) Function']);
xlabel('X');   ylabel('Sinc(X)');
break;
%%
figure;
surf(xpts,ypts,F);
title(['Sombrero Peaks  --  Cartesian Sinc Function F: Ni=',num2s
xlabel('X');
ylabel('Y');
grid on;
fprintf('Plot of f(x,y) completed. \n');
```

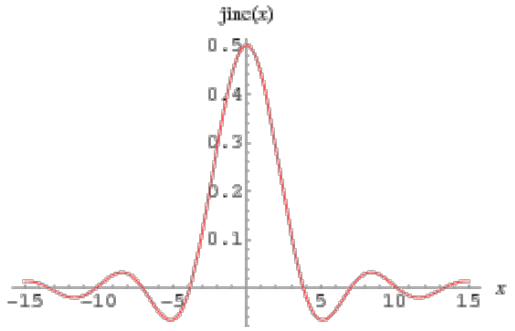
2D Sombrero Function Using Matlab sinc(x)



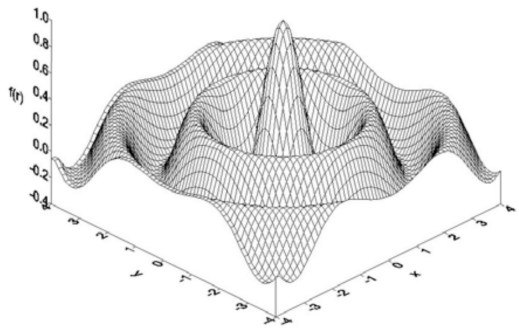
Sombrero Plt: Using Matlab Sinc(x) Function: Ni=64, S=3, Freq=0.5

Generating *sinc*(x) using "Jinc" Bessel Function

- Bessel Functions can be used to approximate *sinc*(x)
- Often this function is called the *bsinc* or *jsinc*
- Bessel Function $Jinc(x) = J_1(x)/x$
- `http://mathworld.wolfram.com/JincFunction.html`



Plot of a 1D *Jinc*(x) function.



2D rotationally symmetric *Circ* function.

`http://en.wikipedia.org/wiki/Bessel_function`
`http://www.cis.rit.edu/class/simg716/handouts/06_notes_2Dfunctions.pdf`

Fraunhoffer Diffraction: the Airy Disk Function

Fraunhofer diffraction pattern of a circular aperture is given by the squared modulus of the Fourier transform (Bessel function) of the circular aperture:

$$I(\theta) = I_0 \left(\frac{2J_1(ka\,sin\theta)}{ka\,sin\theta} \right)^2 = I_0 J_1 \left(\frac{2J_1(x)}{x} \right)^2$$

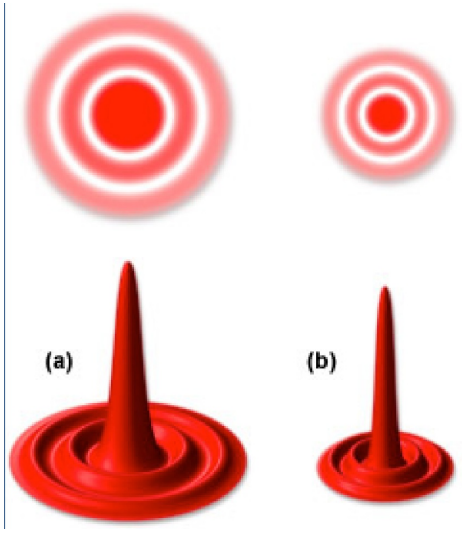
I_0	is the maximum intensity of the pattern at the Airy disc center
J_1	*Bessel* function of the first kind of order one
k	wavenumber $= 2\pi/\lambda$
a	is the radius of the aperture
θ	the angle of observation
R	is the focal length

The Airy pattern falls slowly to zero with increasing distance from the center, which allows us to use a Gaussian profile approximation for the intensity distribution:

$$I(r) \approx I_0' exp\left(\frac{-r^2}{\lambda N}\right) \approx \left(\frac{P_0 A}{\lambda^2 R^2}\right) exp\left(\frac{-r^2}{\lambda N}\right)$$

$$\approx \left(\frac{A}{\lambda^2 R} cos^2(r)\right) exp\left(\frac{-r^2}{\lambda N}\right)$$

http://en.wikipedia.org/wiki/Airy_disk



Airy disk pattern made by light passing through a small aperture.

Fraunhoffer Diffraction: the Airy Disk Function

- The intensity pattern can be written using cartesian coordinates as:
  $$F(x,y) = f * cos^2(r) * e^{(-gr^2)}$$
  where (x, y) are cartesian coordinates, $r = \sqrt{x^2 + y^2}$, and f and g are constants that control the shape of the waves.
- This particular form allows us to factor the function into a matrix multiplication equation of the form:
  $$F(x,y) = A * B$$
  where:
  $$A = f * cos^2\left(\sqrt{x^2 + y^2}\right) \quad \text{and} \quad B = e^{-g\left(x^2+y^2\right)}$$
- The function is calculated using Hadamard (*element-wise*) matrix multiplication: Ref:
  `http://en.wikipedia.org/wiki/Hadamard_product_(matrices)`

Hadamard (element-wise) Matrix Multiplication

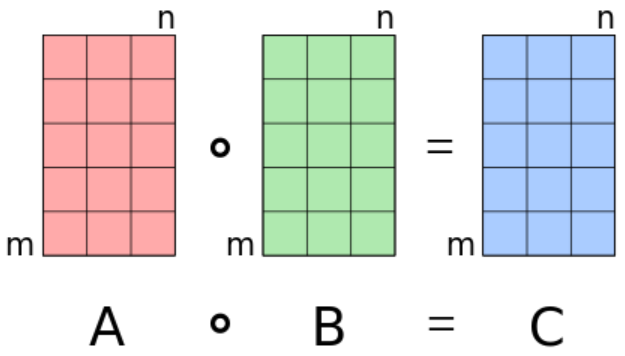
The Hadamard (or Schur) product is a binary operator that operates on 2 identically-shaped matrices and produces a third matrix of the same dimensions.

Definition: If $A = [a_{ij}]$ and $B = [b_{ij}]$ are $m \times n$ matrices, then the Hadamard product of A and B is defined to be:

$$(A \circ B)_{ij} = (A)_{ij} \cdot (B)_{ij}$$

is an $m \times n$ matrix $C = [c_{ij}]$ such that

$$c_{ij} = a_{ij} * b_{ij}$$



The Matlab *times* operation ($C = A.*B$) is a Hadamard product.

Refs: http://en.wikipedia.org/wiki/Hadamard_product_(matrices)
http://www.mathworks.com/help/matlab/ref/times.html

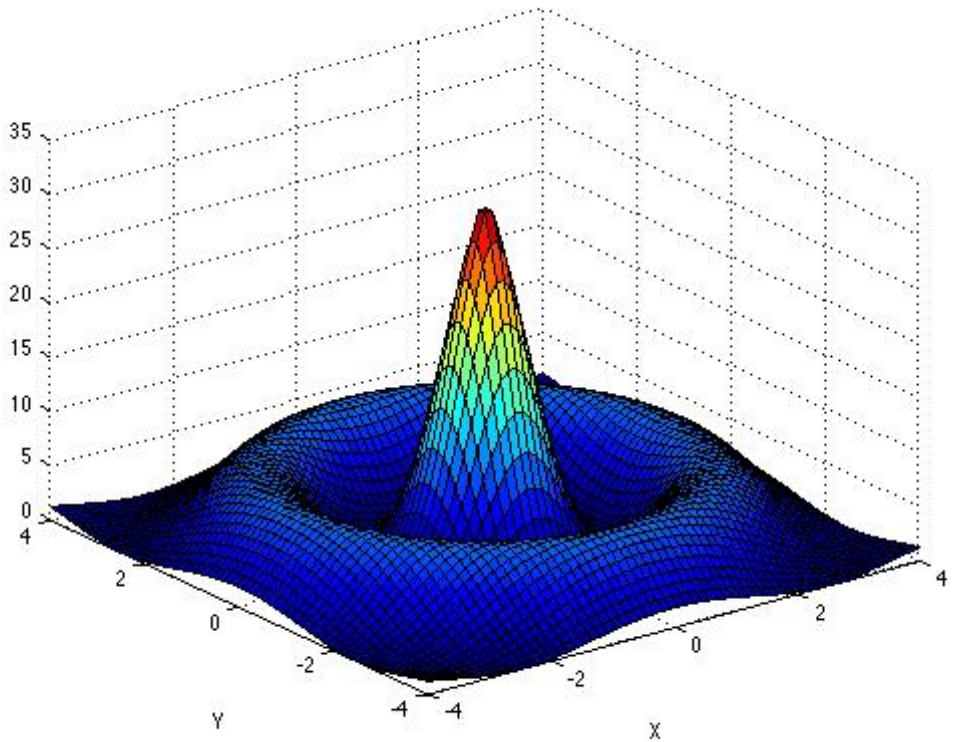
Matlab code to generate Sombrero using Hadamard Multiplication on Modified Circle Function

```
% File: sombrero_using_bessel_formulation_hadamard.m
% Airy disc/Diffraction pattern: Matlab code
%
%   By Mary Thomas
%       updated:  March, 2015
%       Created: March, 2014
%
%  note: ni does have to equal nj, but it is for this problem
%  scaling factors affect max amplitude
%  number of wavelengths
%
clear all;
ni=64;
size=ni*ni;
fprintf('begin calc bessel sombrero\n');
%% Tweak control parameters to adjust amplitude,
%% number of peaks, and spacing/depth of peaks
% f: max amplitude
f=1;
%f=0.2;
%f= 0.25 ;
%f=0.3,;
%f= 0.4 ;
%f=0.5 ;
%f=0.75 ;
%f=0.9 ;
%f=1.0];
f=famp(5);

% g: inverse wavelength
g=.1;
%g=0.25;
%g=.5; %g=0.6;
%g=.75; %g=.1;

% s:  number of wavelengths in hat ~ s+1
%s=2;
s=3; %s=4;  %s=5;
%s=8;
x = linspace(-s , s ,ni);
y = linspace(-s , s ,ni);

fprintf('init x,y complete.\n');
%% serial matrix-matrix multiplication
%
C=zeros(ni,ni);
for i=1:ni
    for j=1:ni
        r(i,j)=sqrt(x(i)^2 + y(j)^2);  %*f*pi; %convert to radian
        xpts(i,j) = x(i); % these are for plotting
        ypts(i,j) = y(j);
    end
end

for i=1:ni
    for j=1:ni
            A(i,j) = f*(cos(r(i,j)))^2;
            B(i,j) = exp(-g*(r(i,j)^2));
      end
end

%calculate the Hadamard Product
for i=1:ni
    for j=1:ni
            C(i,j)  =  A(i,j) * B(i,j);
      end
end

fprintf('calc C complete.\n');
```

Matlab code to generate Sombrero using Hadamard Multiplication on Modified Circle Function

```
%% create 3D surface node mesh %surf(c);
figure;
surf(xpts,ypts,C);
title(['Sombrero Peaks  --  C=A*B: Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');
ylabel('Y');
grid on;


%%
%% create 3D surface node mesh %surf(c);
D=A.*B;
figure;
surf(xpts,ypts,D);
title(['Sombrero Peaks  --  D=A*B: Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');
ylabel('Y');
grid on;
```

Sombrero Function Using Hadamard Multipilcation of Modified Circle Function



Sombrero Plot Using Bessel Function: Ni=64, S=3, Freq=0.5

"Wave" Generator Using Matrix-Matrix Multiplication of Airy Disk Function

- For this assignment you will use *matrix-matrix* multiplication (not the Hadamard product, see above) to calculate a matrix product using the matrices defined by the Airy disc/Fraunhaufer Diffraction pattern described above:

  $$F(x,y) = f * cos^2(r) * e^{(-gr^2)}$$

  where (x,y) are cartesian coordinates, $r = \sqrt{x^2+y^2}$, and f and g are constants that control the shape of the waves.
- This particular form allows us to factor the function into a matrix multiplication equation of the form:

  $$F(x,y) = A * B$$

  where:

  $$A = f * cos^2\left(\sqrt{x^2+y^2}\right) \qquad \text{and} \qquad B = e^{-g\left(x^2+y^2\right)}$$
- Notes:
  1. (x,y) are the grid cartesian coordinates, and f and g are constants that control the shape of the waves.
  2. The results will differ from those obtained using Hadamard matrix multiplication.

2D Matrix-Matrix Multiplication (Mat-Mat-Mult)

```
/* Serial_matrix_mult */
for (i = 0; i < n; i++)
   for (j = 0; j < n; j++) {
      C[i][j] = 0.0;
      for (k = 0; k < n; k++)
         C[i][j] = C[i][j] + A[i][k]*B[k][j];
   printf(... )
}
```

Dimensions [ROWS x COLS]

$A = m \times k$

$B = k \times n$

$C = m \times n$

$$\begin{bmatrix} a_{11} \dots a_{1j} \dots a_{1k} \\ \dots \\ a_{i1} \dots a_{ij} \dots a_{ik} \\ \dots \\ a_{m1} \dots a_{mj} \dots a_{mk} \end{bmatrix} * \begin{bmatrix} b_{11} \dots b_{1j} \dots b_{1n} \\ \dots \\ b_{i1} \dots b_{ij} \dots b_{in} \\ \dots \\ b_{k1} \dots b_{kj} \dots b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} \dots c_{1j} \dots c_{1k} \\ \dots \\ c_{i1} \dots c_{ij} \dots c_{ik} \\ \dots \\ c_{m1} \dots c_{mj} \dots c_{mk} \end{bmatrix}$$

2D Matrix-Matrix Multiplication (Mat-Mat-Mult)

Definition: Let A be an $[m \, x \, k]$ matrix, and B be a be an $[k \, x \, n]$, then C will be a matrix with the dimensions $[m \, x \, n]$.

Then

$$AB = \lfloor c_{ij} \rfloor \; c_{ij} = \sum_{t=1}^{k} a_{it} b_{tj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{k1} b_{kj}$$

$$= \begin{bmatrix} a_{00} & \cdots & a_{0j} & \cdots & a_{0,k-1} \\ & & \cdots & & \\ a_{i0} & \cdots & a_{ij} & \cdots & a_{i,k-1} \\ & & \cdots & & \\ a_{m-1,0} & \cdots & a_{m-1,j} & \cdots & a_{m-1,k-1} \end{bmatrix} \bullet \begin{bmatrix} b_{00} & \cdots & b_{0j} & \cdots & b_{0,n-1} \\ & & \cdots & & \\ b_{i0} & \cdots & b_{ij} & \cdots & b_{i,n-1} \\ & & \cdots & & \\ b_{k-1,1} & \cdots & b_{kj} & \cdots & b_{n-1,p-1} \end{bmatrix}$$

$$= \begin{bmatrix} c_{00} & \cdots & c_{1j} & \cdots & c_{1,n-1} \\ & & \cdots & & \\ c_{i0} & \cdots & c_{ij} & \cdots & c_{i,n-1} \\ & & \cdots & & \\ c_{m-1,0} & \cdots & c_{mj} & \cdots & c_{m-1,n-1} \end{bmatrix}$$

Matlab code: Mat-Mult to generate Waves

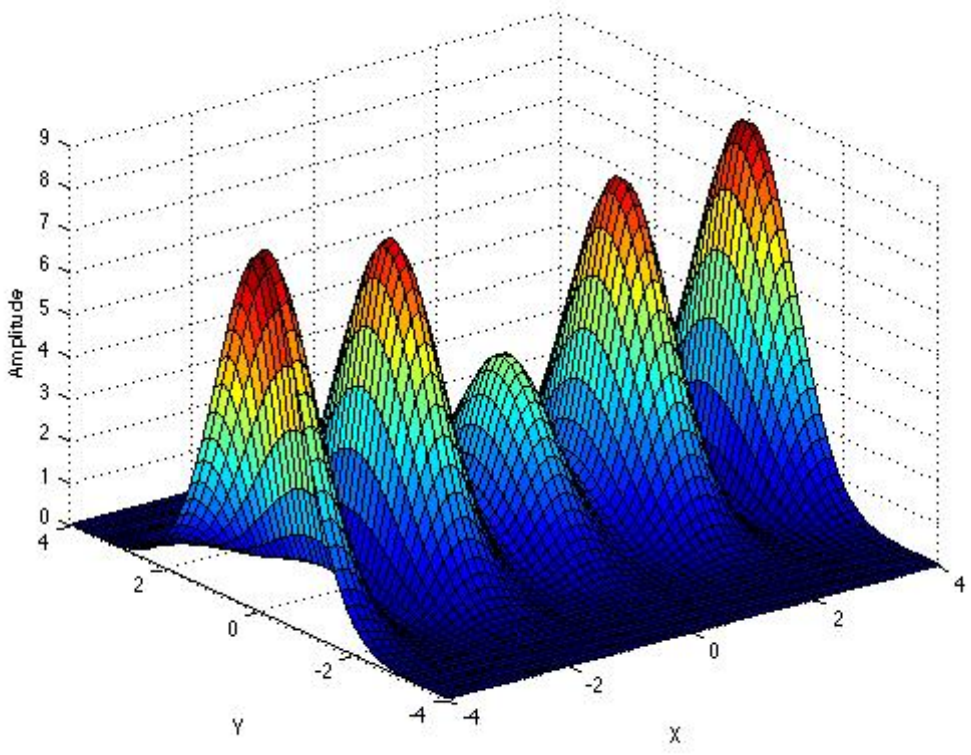
```
%
% Wave Function Test Case: Matlab code
%   Calculates modified bessel function
%       using mat-mat multiplication:  C=A.*B
%
%   By Mary Thomas (March, 2014)
%       updated: March, 2015
%       Created: March, 2014
%
%  scaling factors affect max amplitude
%  number of wavelengths
%
%
clear all;
ni=64;
fprintf('init wave function test case\n');
%%
% f: max amplitude
f=1.0;
f=0.5;
%f=.25;
%f=.1;

% g: inverse wavelength
g=.1;
g=0.25;
%g=.5;
%g=.75;
%g=1.;

% s:  number of wavelengths in hat ~ s+1
s=1;
%s=2;
%s=3;
s=4;
%s=6;
```

```
x = linspace(-s , s ,ni);
y = linspace(-s , s ,ni);

fprintf('init x,y\n');

%% serial matrix-matrix multiplication
%
C=zeros(ni,ni);
A=zeros(ni,ni);
B=zeros(ni,ni);

for i=1:ni
    for j=1:ni
        r(i,j)=sqrt(x(i)^2 + y(j)^2)*pi*f;  %convert to ra

        xpts(i,j) = x(i); % these are for plotting
        ypts(i,j) = y(j);

        A(i,j) = f*(cos(r(i,j)))^2;

        B(i,j) =  exp(-g*r(i,j)^2);
    end
end

fprintf('calc A,B ok\n');
%%
% using r(i,k) produces waves
for i=1:ni
    for j=1:ni
        for k=1:ni;
            C(i,j) = C(i,j) + A(i,k) * B(k,j);
        end
    end
end
fprintf('calc C ok\n');
```

Matlab code: Mat-Mult to generate Waves

```
%% plot 3D surface node mesh -- waves
figure;
surf(xpts,ypts,C);
title(['Bessel Waves -- Calc C(i,j)  Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');ylabel('Y');zlabel('Amplitude');
grid on;

%% plot 3D surface node mesh -- waves
D=A*B;
figure;
surf(xpts,ypts,D);
%title(['Bessel Waves -- Calc D=A*B,  Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');ylabel('Y');zlabel('Amplitude');
grid on;
```

"Wave" Generator Using Matrix-Matrix Multiplication of Airy Disk Function



"Wave" Generator Using Matrix-Matrix Multiplication of Airy Disk Function