# UPAS MATLAB - IV

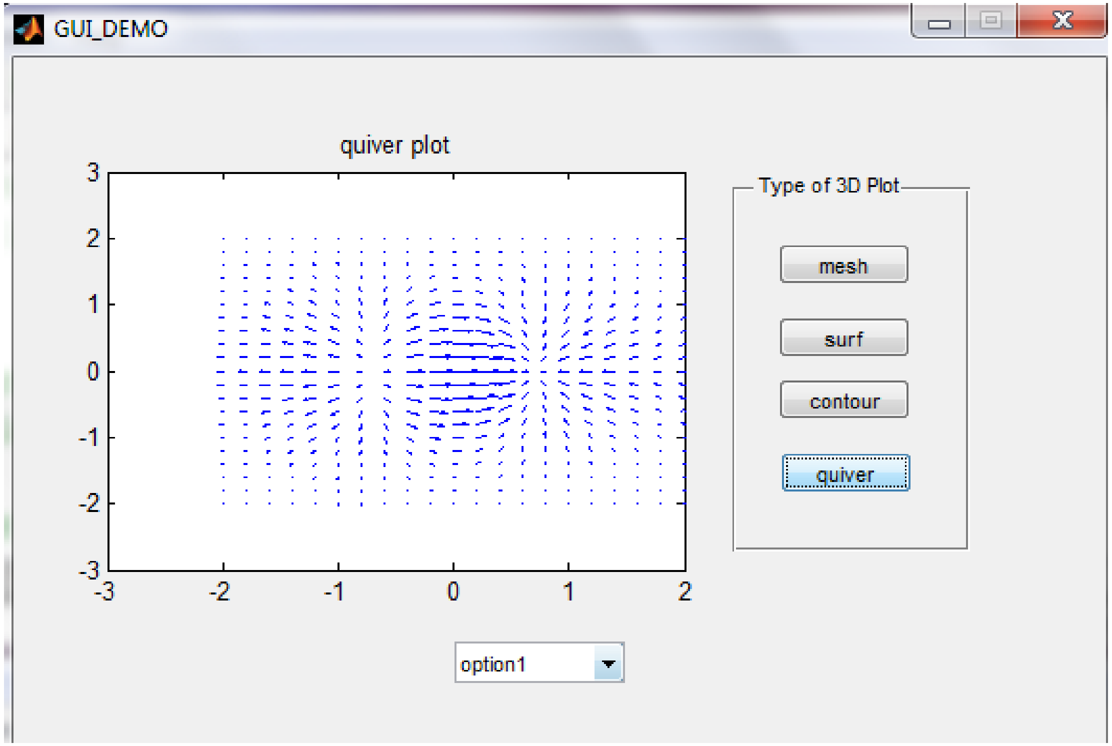
- **Reprise some topics – repetition is good. Readunderstand the scripts rather than just executing them.**

- **Grid based solutions of pde's.**

- **A few new topics**

# GUI_DEMO

- Look at script. Run through a palatte. This has only pushbuttons



# Construct a GUI

- **Suggest a script to be wrapped .....**
- **Project?**

# Callbacks

- **e.g. pushbutton**
- **Look at other GUI for menu, executable text, pb, sliders, etc.**

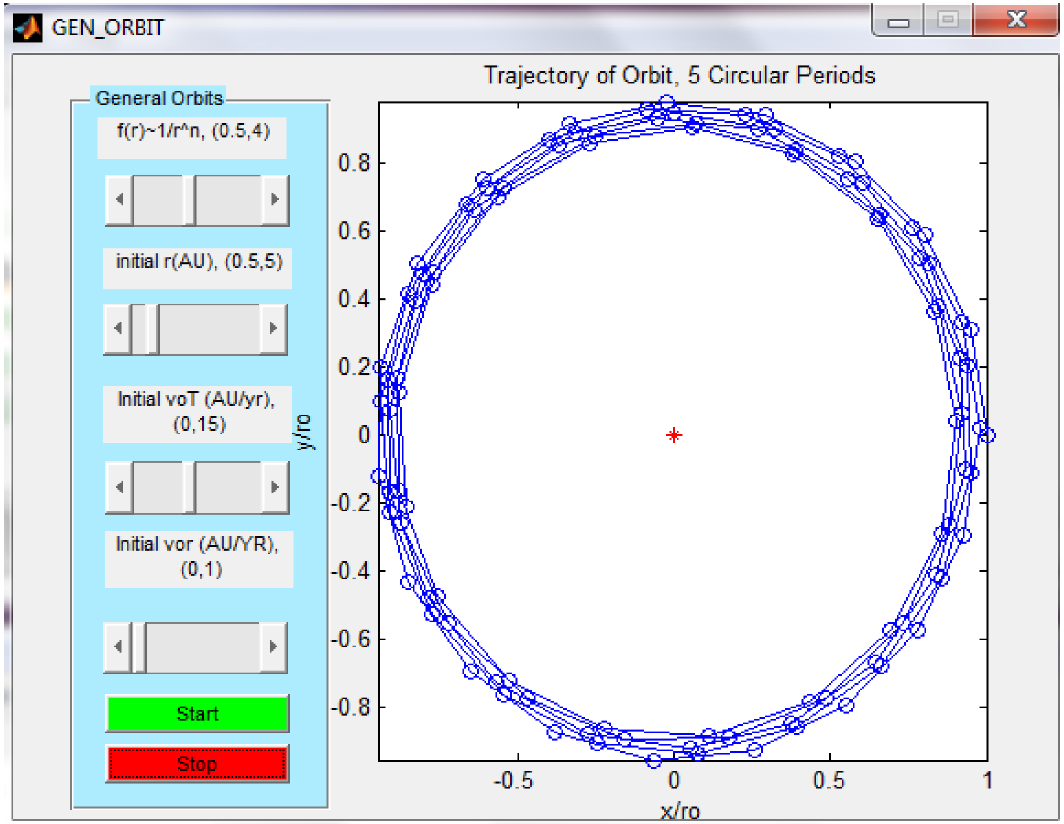
```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
[X,Y] = meshgrid(-2:0.2:2);
Z = X .*exp(-X .^2 - Y .^2);
mesh(Z)
title('mesh plot')
%
```

# Perihelion Advance

- Use to explore callbacks for sliders; 2.1,1.0,1.33,0
- 1/r^n  is attractive

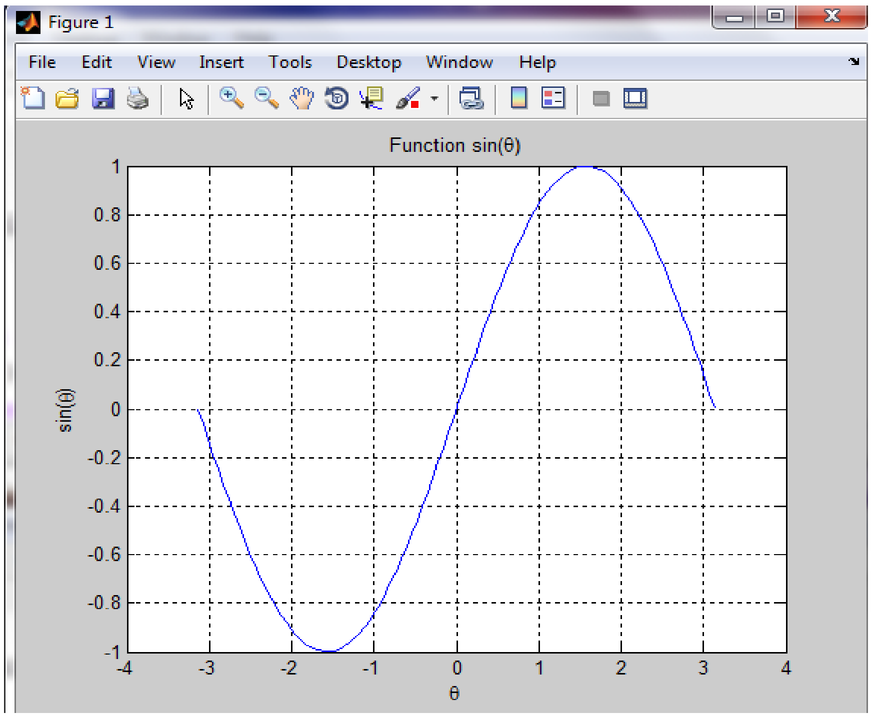
  But L^2/r^3

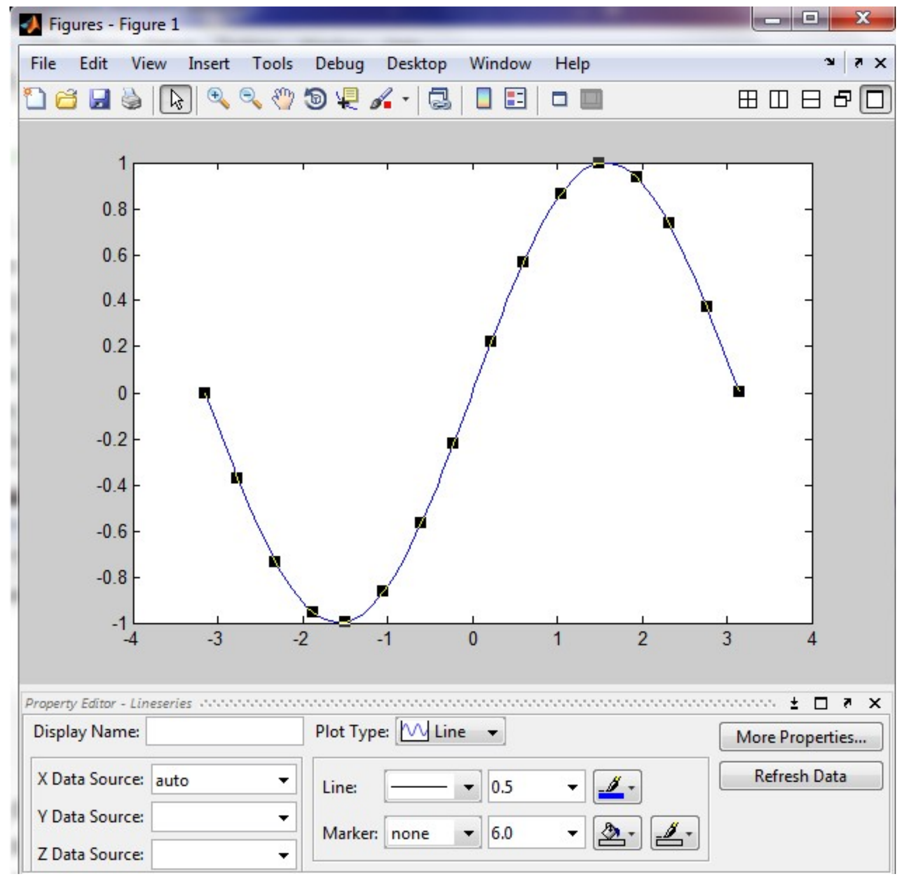
  Repulsive - minimum?
- Stable orbits?
- re-entrant?
- Generalized cm_kepl3



# plot_demo

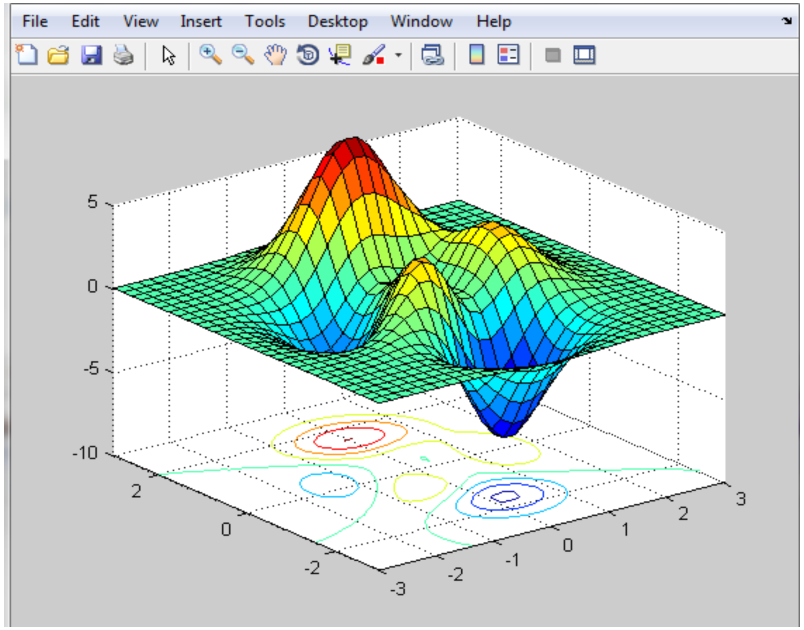
- **tools/edit plot/ - insert; x,y, title, etc.**

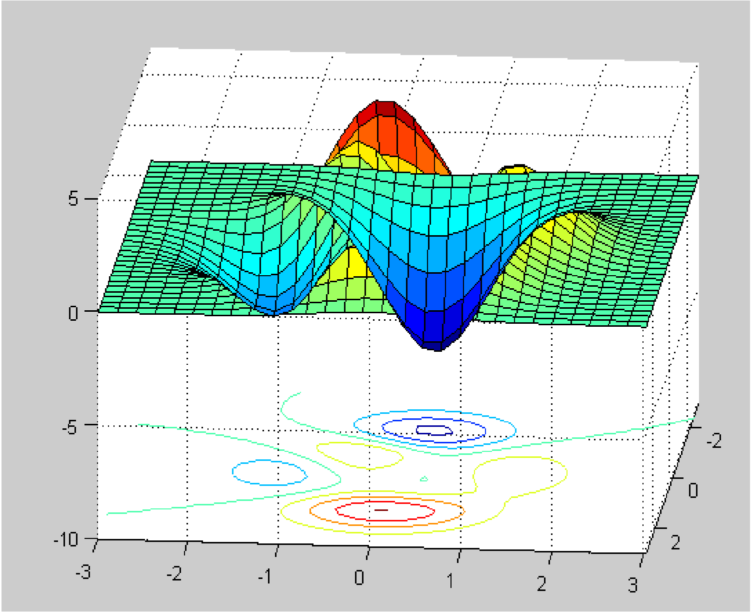


# demo_3d

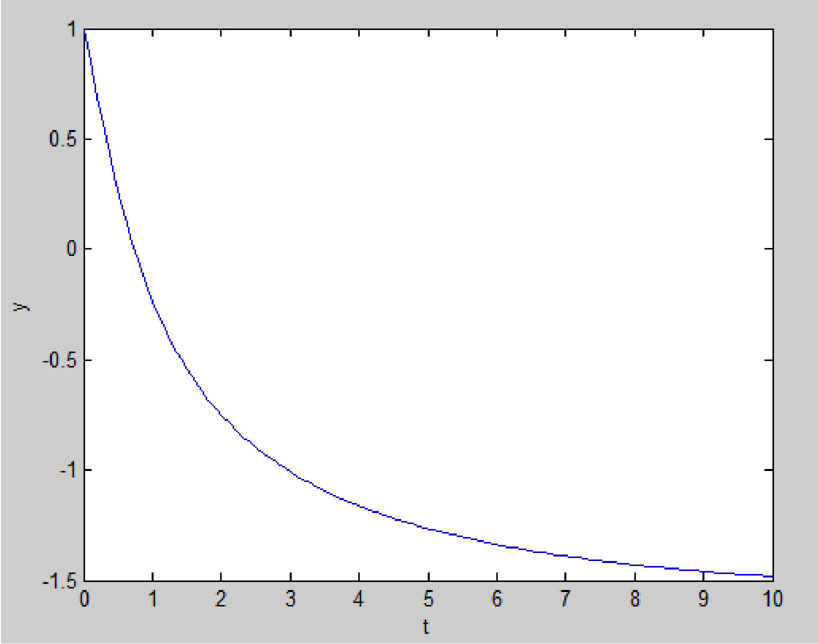
- View – dropdown menu
- Tools – rotate 3d
- Edit plot dynamically using tools for figures



# ode45

- Look at ode_demo
- odedemofun
- Simplest first order RK – gets started
- Many script example can be used (e.g. cm_kepl3)



# Symbolic ode wrapper – use it

```
>> SM_ODE3
  Program to symbolically solve ODE


Enter Single Differential Eq to Solve y(t); e.g., D2y+a*y=0
: D2y+a*y=0
Symbolic Solution, y(t) and v(t)

y =

C2*exp((-a)^(1/2)*t) + C3*exp(-(-a)^(1/2)*t)


v =

C2*(-a)^(1/2)*exp((-a)^(1/2)*t) - C3*(-a)^(1/2)*exp(-(-a)^(1/2)*t)
```

**Suggest an ODE to solve completely general**

**Project?**

# Thin Lense Doublet

```
function[fpttpl,fpltpt,fpttpt,x1,x2,x3,x4,x5,y1,y2,y3,y4,y5] = ....
    Thin_Lense(L,l,Lo,itype)
%
% thin lense values for  D and F focal length
% x is DF, y is FD,f(1) is first quad focal length, f(2) is second
%
fpttpl(1) = L .*sqrt(l ./(L + l)) ;
fpttpl(2) = (l .*L) ./fpttpl(1);
fpltpt(1) = sqrt(l .*(l + Lo));
fpltpt(2) = (l .*Lo) ./fpltpt(1);
c = L + l + Lo;
fpttpt(1) = L .*sqrt((l .*(l+Lo)) ./((l+L) .*c));
fpttpt(2) = (l .*L .*Lo) ./(c .*fpttpt(1));
%
if itype == 1
    f1 = fpttpl(1);
    f2 = fpttpl(2);
    xo = [0.;1.];
end;
if itype == 2
    f1 = fpltpt(1);
    f2 = fpltpt(2);
    xo = [1.;0.];
end;
if itype == 3
    f1 = fpttpt(1);
    f2 = fpttpt(2);
    xo = [0;1];
end;
```

**Focal lengths for the 3 Conditions. Try to prove one. Project?**

# Doublet - II

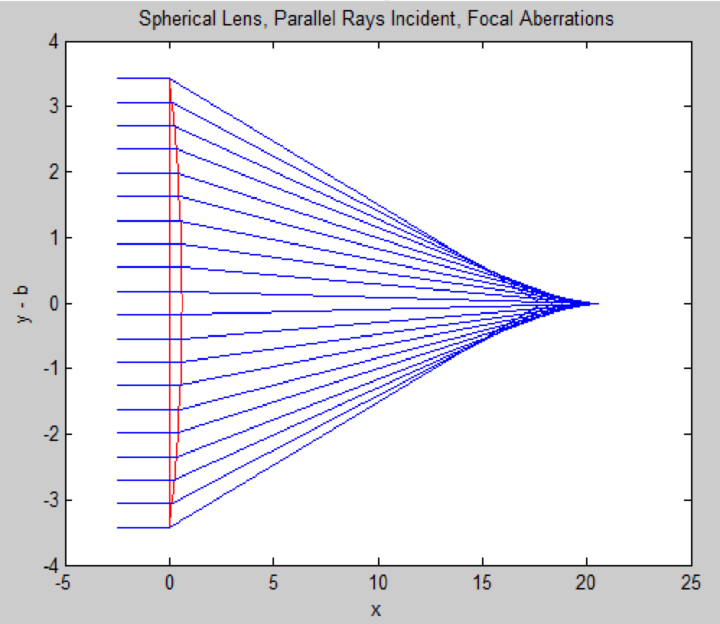
```
%
% x position and angle matrices
%
m1 = [1., L; 0., 1.];
m2 = [1., L;1.0 ./f1  ,1.0+L ./f1];
m3 = [1.0 + l ./f1 , L + l + (L .*l) ./f1;1.0 ./f1, 1.0 + L ./f1];
m4(1,1) = m3(1,1);
m4(1,2) = m3(1,2);
m4(2,1) = -l ./(f1 .*f2) + 1.0 ./f1 - 1.0 ./f2;
m4(2,2) = 1.0 + L ./f1 - (L + l) ./f2 - (L .*l) ./(f1 .*f2);
m5(2,1) = m4(2,1);
m5(2,2) = m4(2,2);
m5(1,1) = 1.0 + l ./f1 - Lo ./f2 -(Lo .*l) ./(f1 .*f2) + Lo ./f1;
m5(1,2)= l+L+Lo +(L .*(Lo+l)) ./f1 - (l .*L .*Lo) ./(f1 .*f2)-(Lo .*(L + l)) ./f2;
%
x1 = m1 * xo;
x2 = m2 * xo;
x3 = m3 * xo;
x4 = m4 * xo;
x5 = m5 * xo;
%
```

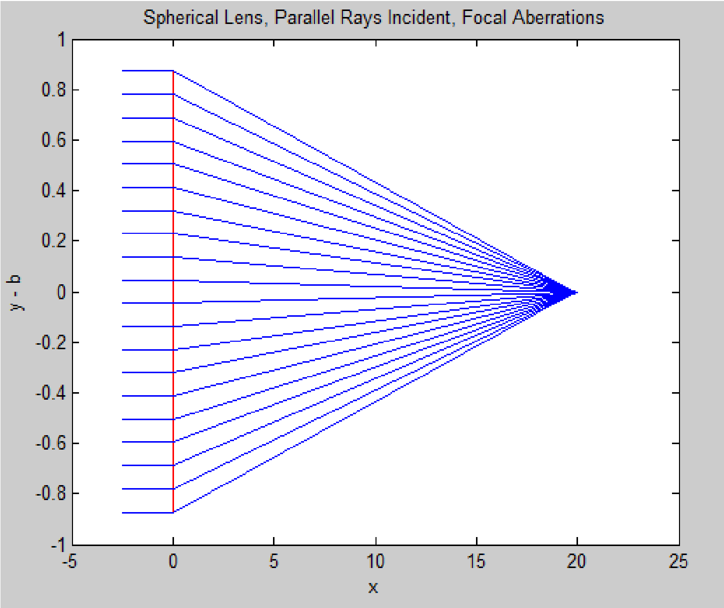
**Solutions arise from imposing focal constraints on the M5 matrix elements. Two Unknowns, f1 and f2. Two conditions on The matrix element in the x and y planes**

# Spherical_Lens2

- **Aperture abberations – 40 and 10 degrees – fill the spherical lense**
- **Useful aperture!**

```
>> Spherical_Lens2
 Ray tracing for a thick shperical lense


Lens Radius = 10, Index of Refraction = 1.5
 Lens Makers Equation - 1/f = (n-1)/R
 Enter The Angular Size of the Lense in Degrees, < 60: 20
Lense Thickness = 0.151922 and 1/2 Height = 1.73648
```

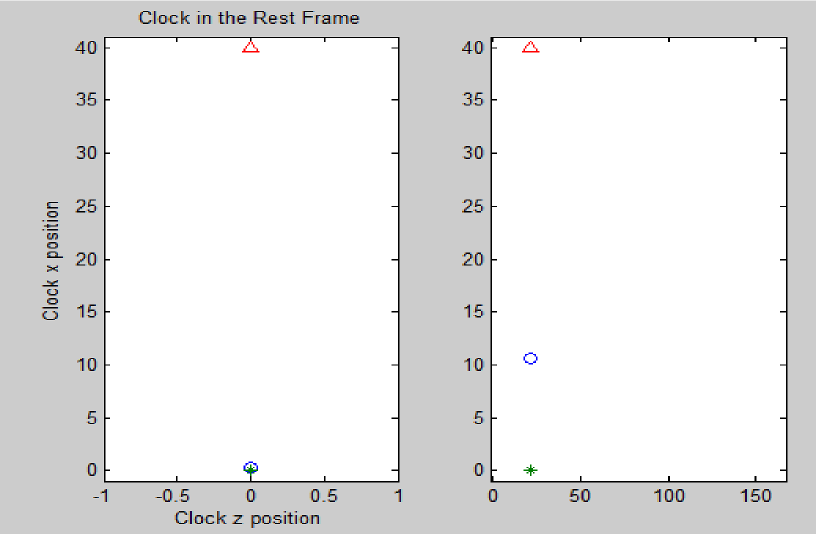


# SR_Time_Dilate

- Demo – proper time and interval – subplot = see also Damped_Driven_SHO
- 4 subplots there

```
>> SR_Time_Dilate
  Program to Illustrate Time Dilation With a Gedanken Clock


 Clock is a Light Source and a Mirror
Input the Velocity of the Clock w.r.t. c: 0.9

 Clock Ticks in Rest Frame = 20, in Moving Frame =  45
```




**Geometrically prove That t is dilated by gamma**

# Solving pde

- MATLAB has pde solver for 1 x and 1 d dimensions. Use in 1-d quantum mech.
- Other methods include representing the pde on a grid and solving numerically.
- 1-d grid – Gen_Eigen2 – use MATLAB eig
- Laplace eq. solution using complex variables
- 2-d grid – Laplace using BC and grid
- 2-d grid – Poisson using grid, FFT

# Eigenfunctions

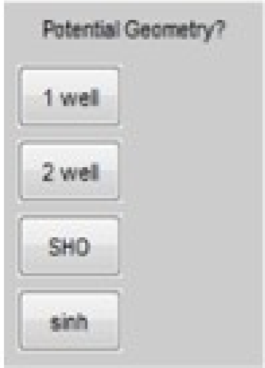
## • 1-d grid

The time independent Schrödinger equation can be solved numerically for any potential configuration. Examples are shown in the script "Gen_Eigen2" for a single well, two wells, the simple harmonic oscillator and a hyperbolic sin confining potential.

The Schrödinger equation on a numerical grid with x labeled by index j becomes:

$$d^2\psi / dx^2 = [(V(x) - E)2m / \hbar^2]\psi \qquad 6.12$$

$$\psi_{j+1} - 2\psi_j + \psi_{j-1} = [(V_j - E)2m / \hbar^2 \Delta x^2]\psi_j$$




```
>> Gen_Eigen2
  solve general time independent Schroedinger Eq
  for eigenvalues and eigenvectors - numerical solutions


x limits (0,50) A
Number of V x Samples = 1000:
Enter SHO k (~ 0.01): 0.01
Energy Eigenvalues = 0.990148 , 0.594093 , 0.198032
```

# Hamiltonian Matrix

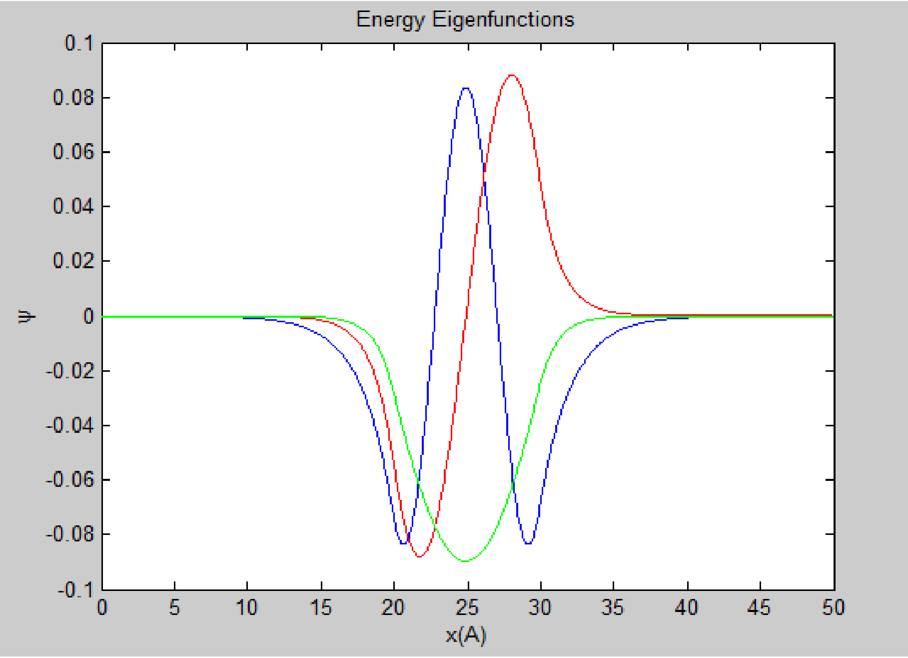
- Eigenvalue problem for the Hamiltonian matrix – 1 d here.
- Use MATLAB matrix utilities, diag, ones, zeros, eigs
- Matrix is very sparse – only derivative is off diagonal

$$H \sim \begin{pmatrix} V-2T & T & 0 \\ T & V-2T & T \\ 0 & T & V-2T \end{pmatrix}$$

$$T = \hbar^2 / 2mL^2$$

# QM - Gen_Eigen2

- **Use MATLAB to find eigenvalues and eigenfunctions (time independent eq) for a very large (1000,1000) but sparse matrix. Grid for second derivative in x**

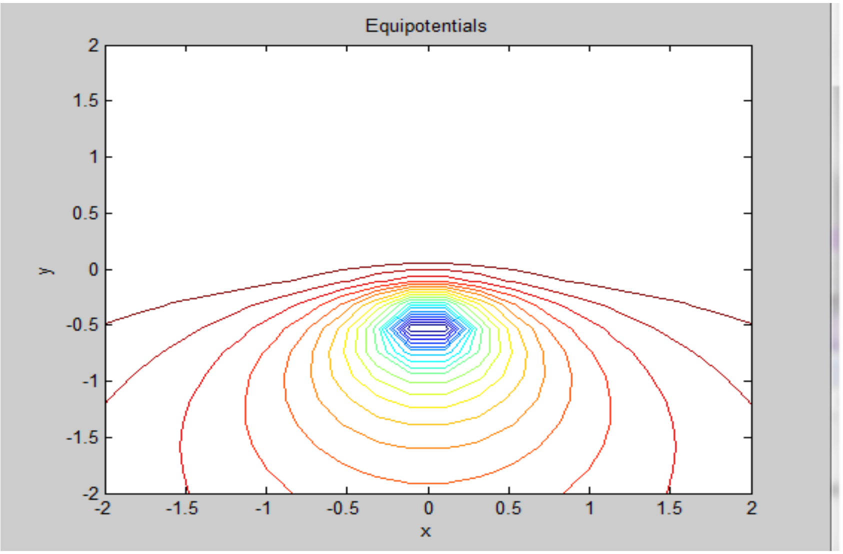



```
x limits (0,50) A
Number of V x Samples = 1000:
Enter Well Depth (~ 2 eV): 3
Energy Eigenvalues = 2.13239 , 0.991361 , 0.252515
```

**Need 3 grid points ~ d2/dx2**

# Laplace_z - Complex

```
>> Laplace_z
  Laplace_z - Laplace Electrostatics - BV, Laplace Eq. Potential, E Field
  Use Complex Variables


Complex Potential  - Modulus of psi is the Potential V, grad(V) = Electric Field
Enter a Number for an Electrostatic Example- 1 to 5: 2
Two Charges - Image Charge for Conducting Plane
```



# Iterative Grid

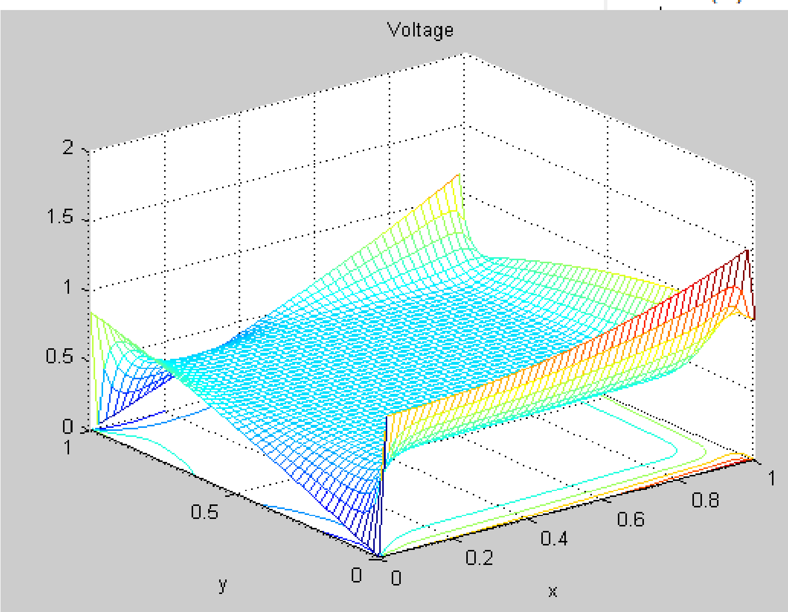
- Use the approximate grid for Poisson and Laplace Eq. To solve by iteration.
- For (x,y) space – not implemented in MATLAB.

$$u_{j,l}^{n+1} = 1/4\left[u_{j+1,l}^{n} + u_{j-1,l}^{n} + u_{j,l+1}^{n} + u_{j,l-1}^{n}\right] - \Delta^{2}\rho_{j,l}$$

# Laplace Eq in 2d - grid

- **2-d PDE**

```
>> EM_Laplace_Test2
  Solve static Laplace Eq. using Gauss Seidel, Cartesian, Boundary Voltages


Solve Finite Difference Eq: 4Vi,j = Vi,j+1 + Vi,j-1 + Vi+1,j + Vi-1,j
Input the Square Grid Number of Points, 0<x<1,0<y<1: 50
Input the Voltage Function on the Left Boundary,f(y)
: sin(y)
Input the Voltage Function on the Right Boundary, f(y)
: cos(y)
Input the Voltage Function on the Top Boundary, f(x)
: sinh(x)
Input the Voltage Function on the Bottom Boundary, f(x)
: cosh(x)
```




**Fix BC arbitrary**
**Solve interior as**
**No sources – second order**
**Partials in x and y = 0.**
**Need 5 grid points**

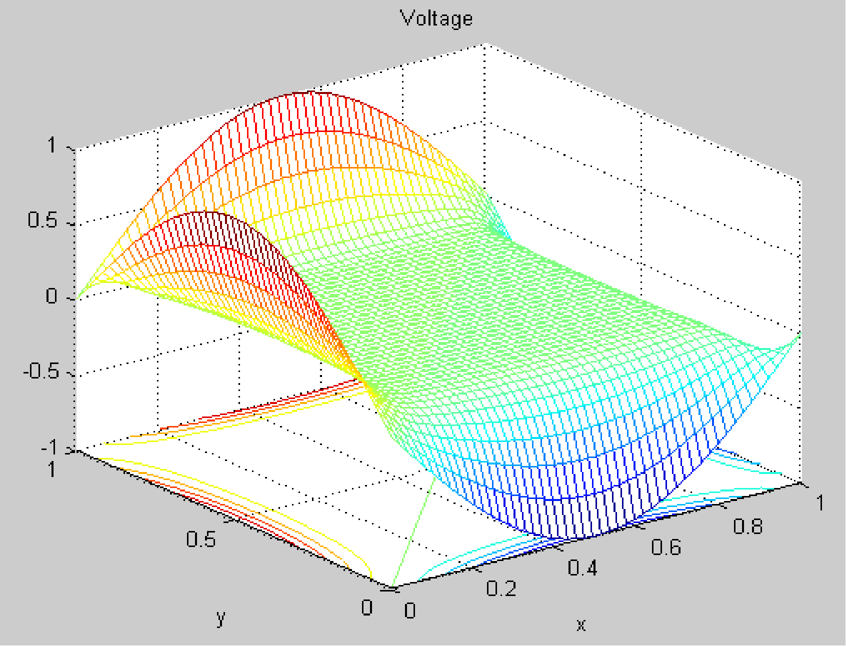
# Example

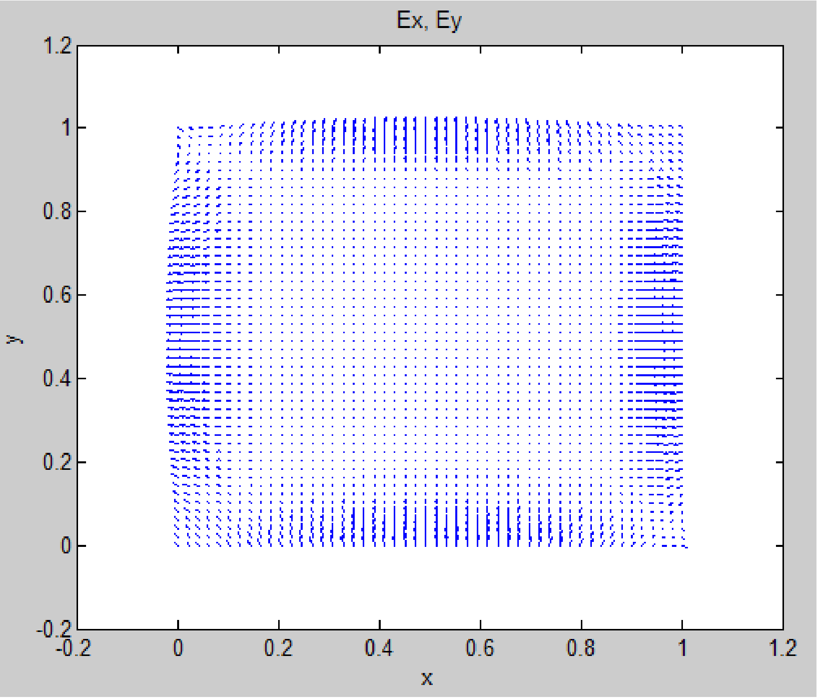
- Use EM_Laplace_Test2
- Specify BC on a rectangular boundary
- Flexible boundary values - symbolic

```
>> EM_Laplace_Test2
  Solve static Laplace Eq. using Gauss Seidel, Cartesian, Boundary Voltages


Solve Finite Difference Eq: 4Vi,j = Vi,j+1 + Vi,j-1 + Vi+1,j + Vi-1,j
Input the Square Grid Number of Points, 0<x<1,0<y<1: 50
Input the Voltage Function on the Left Boundary,f(y)
: sin(pi*y)
Input the Voltage Function on the Right Boundary, f(y)
: -sin(pi*y)
Input the Voltage Function on the Top Boundary, f(x)
: sin(pi*x)
Input the Voltage Function on the Bottom Boundary, f(x)
: -sin(pi*x)
```

# Example - II

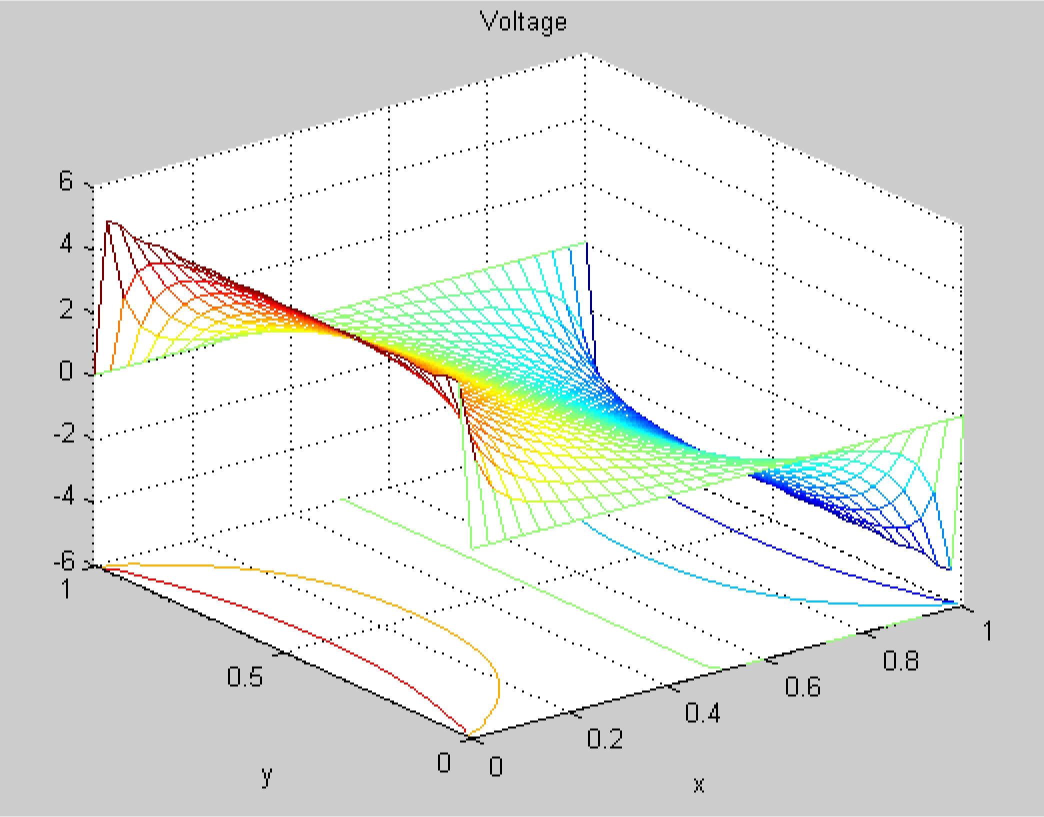
- 10 Iterations – start from mean voltage.
- Fairly slow - Gauss-Seidel



# Laplace_Series

- Do effectively a Fourier series to match BC,



# EM_Poisson_Test

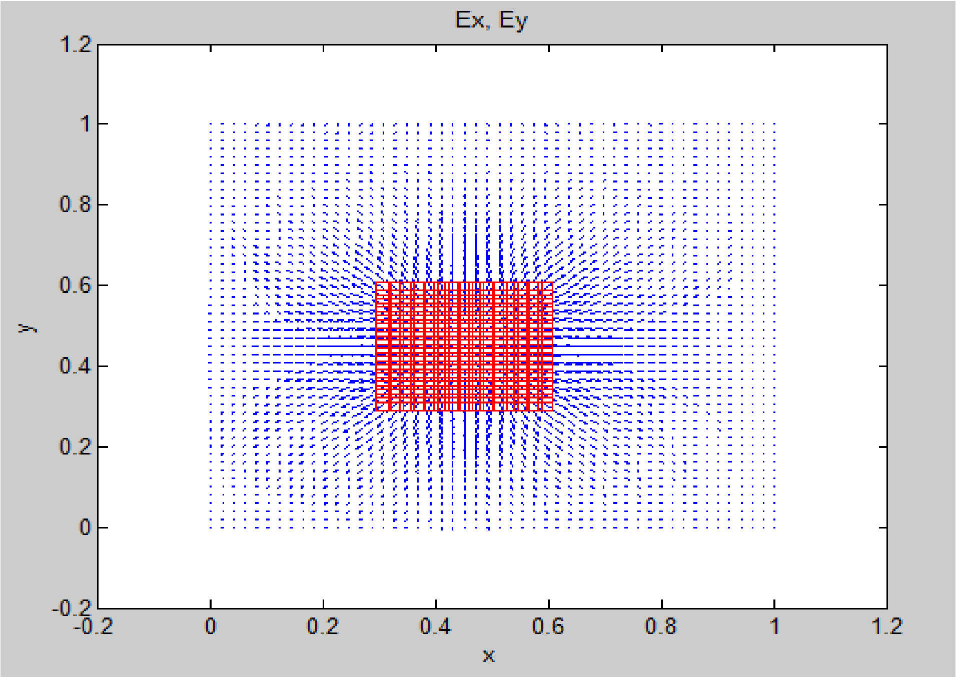
- **Solve for point and rectangular charges (e.g. parallel plate capacitor). Grid solution. Boundaries = 0 (note CMS magnet !)**
- **Uses MATLAB package for FFT; round, zeros,fft2, ifft2**

```
>> EM_Poisson_Test
  Solve static Poisson Eq. using FFT for Periodic BC, Cartesian - MATLAB


Solve Finite Difference Eq: 4Vi,j = Vi,j+1 + Vi,j-1 + Vi+1,j + Vi-1,j - rhoi,j*del*del
Input the Square Grid Number of Points, 0<x<1,0<y<1: 50
Input Number of Point Charges: 0
Input Number of Charged Rectangles: 1

 For Rectangle 1
Enter Top Right [x,y] Position: [0.6 0.6]
Enter Bottom Left [x,y] Position: [0.3 0.3]
Input Voltage on Rectangular Charge: 100
 FT of Diff eq for Vij is rhoij*del*del/[2(cos(2*pi*i/N) + cos(2*pi*j/N)-2)]
```

# 1 rectangle

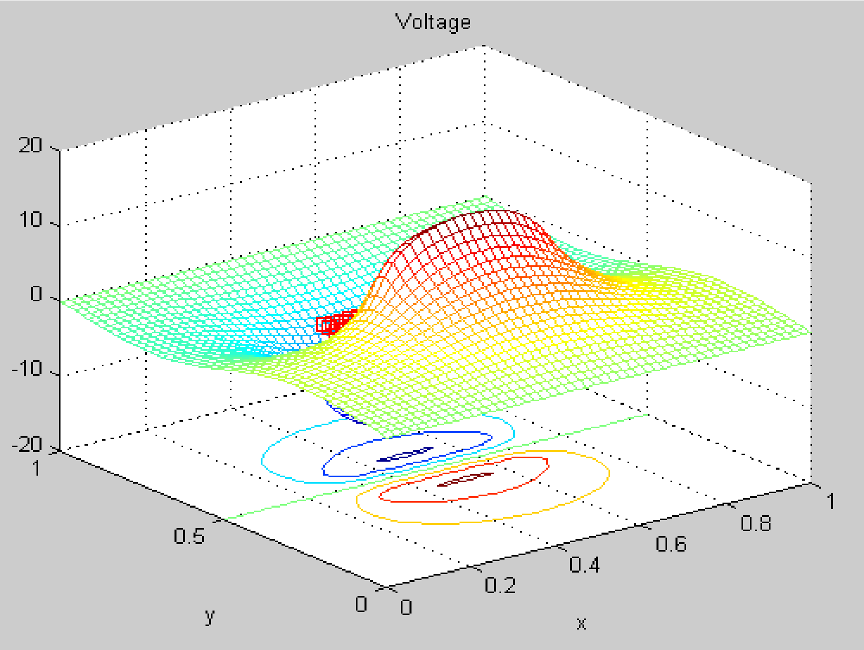
- **E fields**

$$u_{j,l}^{n+1} = 1/4[u_{j+1,l}^{n} + u_{j-1,l}^{n} + u_{j,l+1}^{n} + u_{j,l-1}^{n}] - \Delta^2 \rho_{j,l}$$



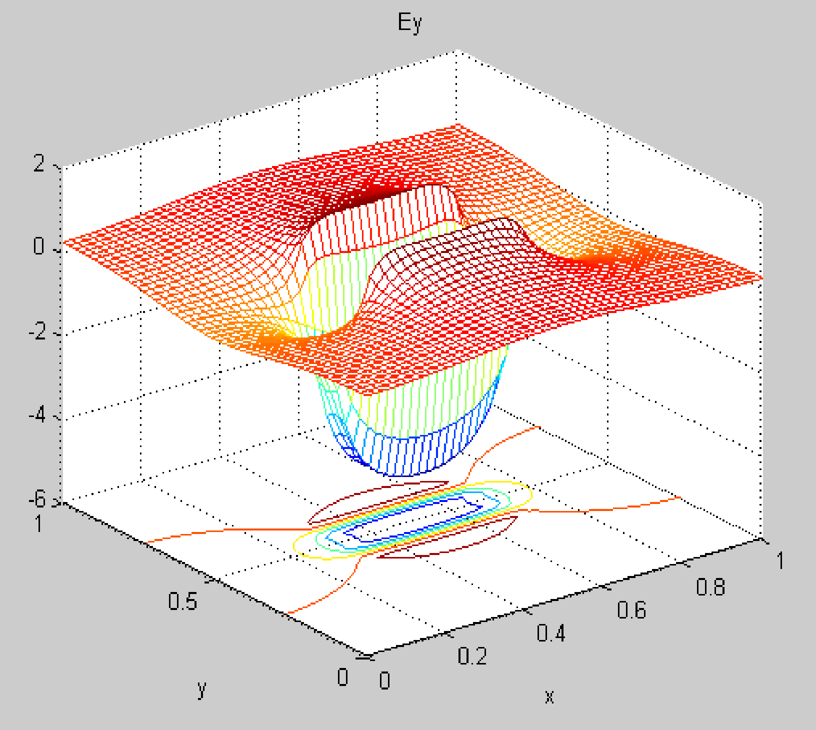
# Poisson - II

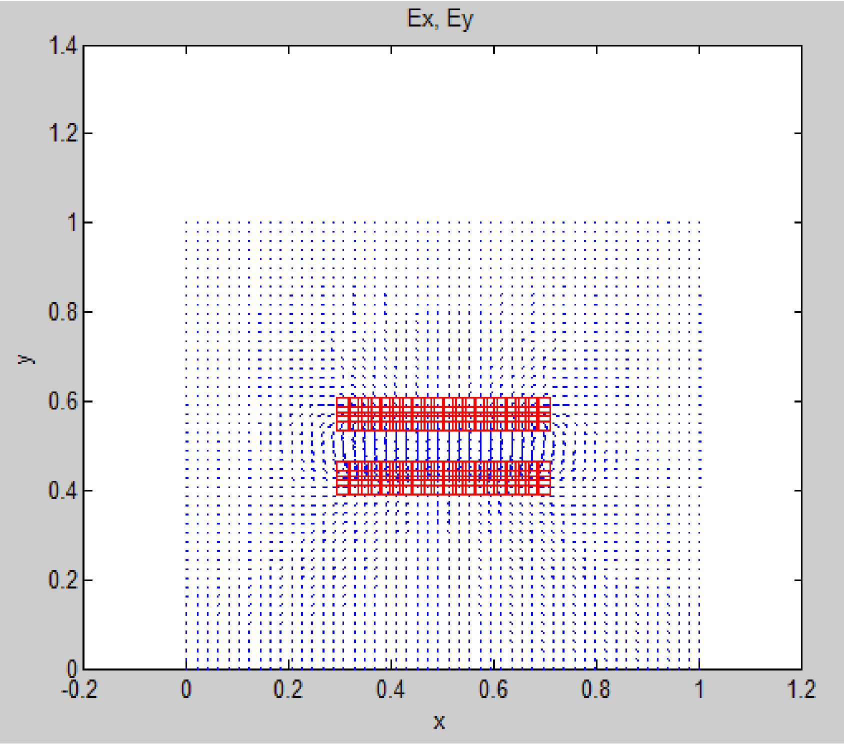
- **Make FT of sources. Then solve for FT of potential analytically. Use inverse FT, ifft2 to put potential back into (x,y). MATLAB tools.**



# Poisson - III

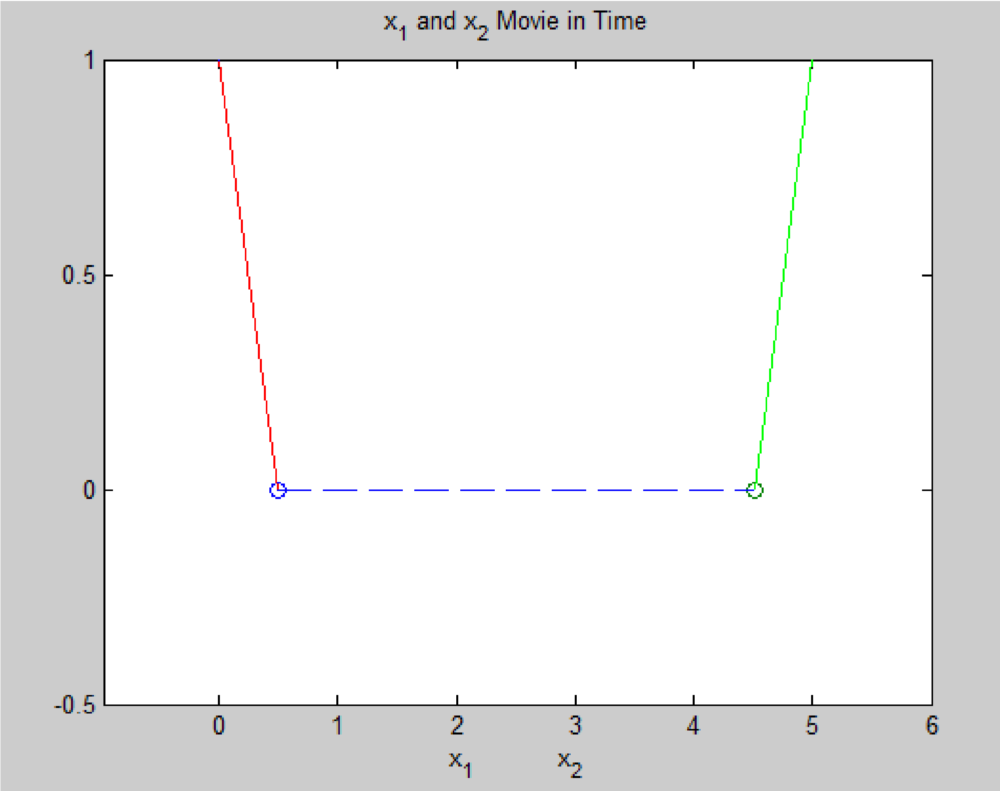
- Ey and quiver – example for parallel plate capacitor model.



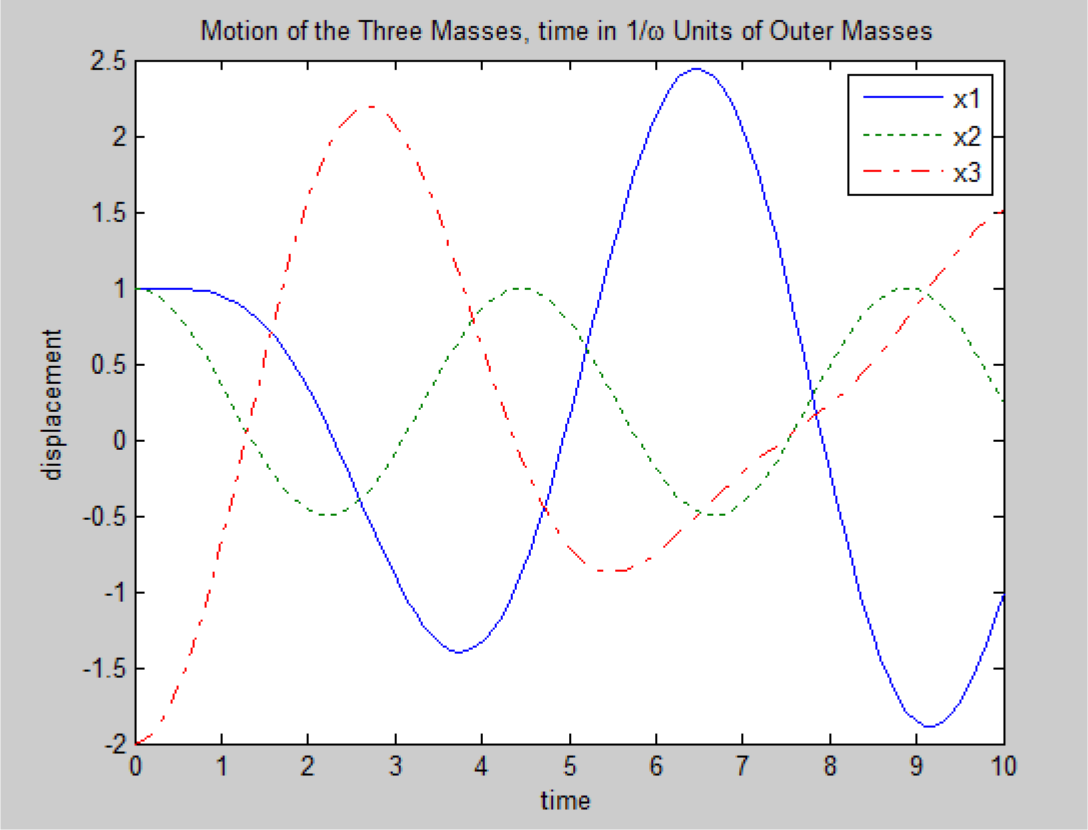
# Coupled SHO

- Coupled objects – normal modes and eigenvalues, "cm_2sho"



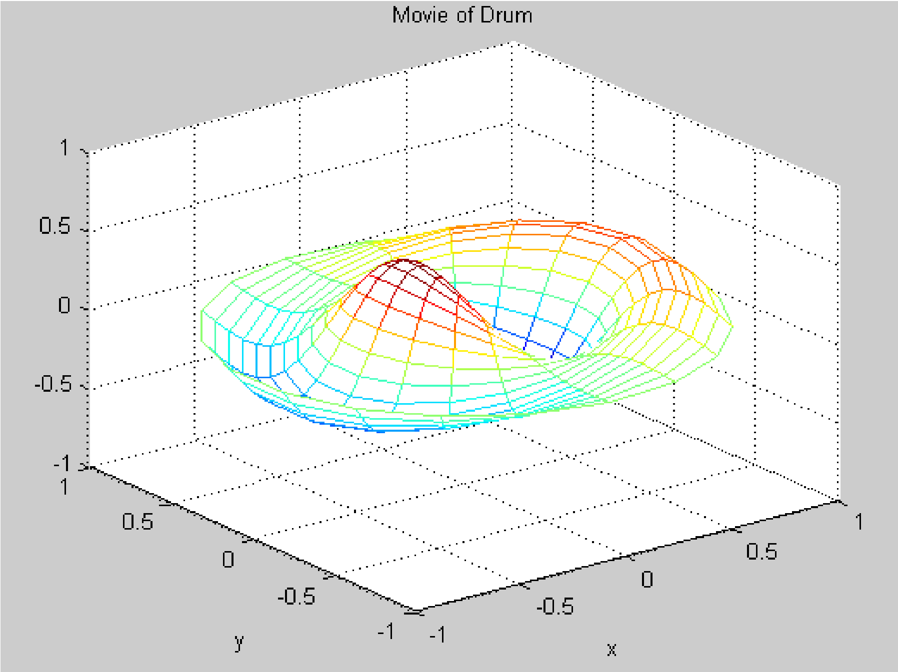
# cm_triatomic

- **Can setup eigenvectors as initial conditions**



# Drum Oscillations

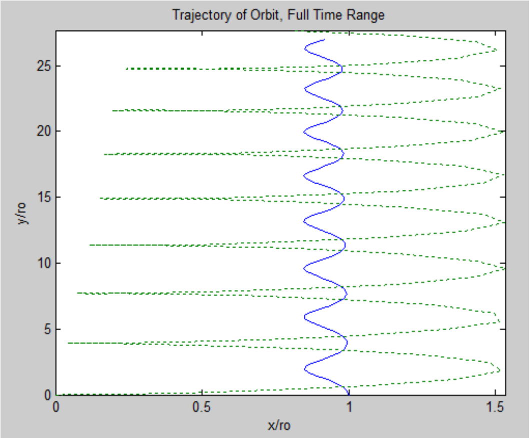
- Drum_Modes, series solutions vanish at r = 1
- Wave equation – 2-d + time, Bessel functions



# Binary System

```
>> Binary2
  Binary - Program to compute binary orbits - ode45, RK
  Planetary wobble to equal mass binaries


Enter Initial Distance Between the 2 Masses ro(AU): 1
Enter Mass of Body1 in Solar Units: 1
Enter Mass of Body2 in Solar Units: 0.1
Velocity of circular orbit, v = 29921.6 m/sec
For circular orbit, period = 3.12883e+07 sec
Enter initial tangential velocity Body1 (AU/yr), 2pi for Circle: 6
Enter initial radial velocity Body1 (AU/yr): 0
Enter initial tangential velocity Body2 (AU/yr), 2pi for Circle: 0
Enter initial radial velocity Body2 (AU/yr): 0
```

**RK in ode45. Look for wobble of sun for planet at 0.1 solar mass at 1 AU**



# That all folks

- **Thanks for being an attentive and active group**
- **This was a fun week**
- **We look forward to see your projects.**