EEE 208 – Programming for EEE

Assist. Prof. Dr. Engin Mendi

Polyfit in Electronics: Modeling a Diode

- Example: A forward-biased diode has the following corresponding voltage and current. Use MATLAB to determine the reverse saturation current, *Is* and diode parameter *n*.

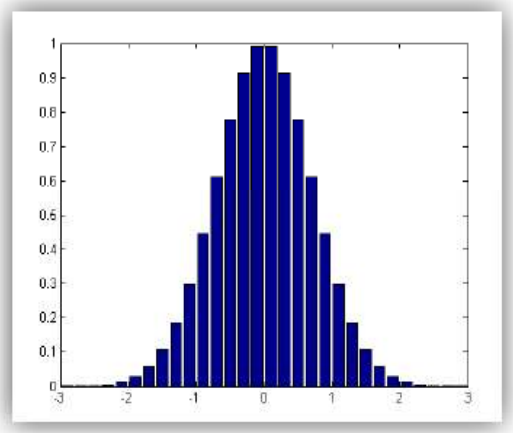
$$\ln(i) = \frac{v}{nV_T} + \ln(I_S)$$

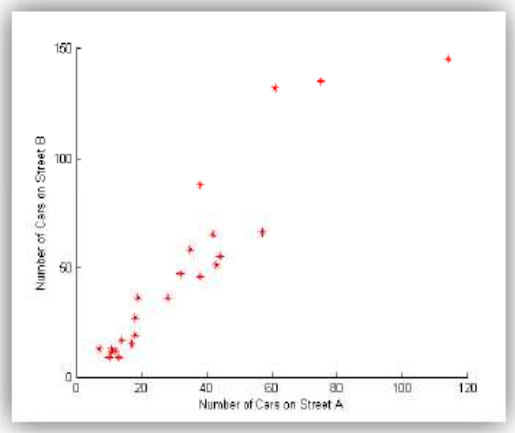
Forward Voltage, V	Forward Current, A
0.1	0.133e-12
0.2	1.79e-12
0.3	24.02e-12
0.4	0.321e-9
0.5	4.31e-9
0.6	57.69e-9
0.7	7.726e-7

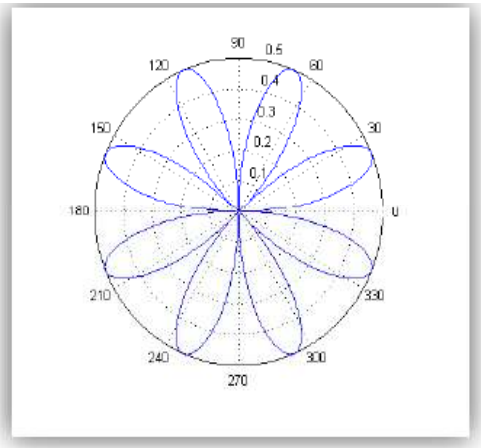
```
% Diode parameters
vt = 25.67e-3;
v = [0.1 0.2 0.3 0.4 0.5 0.6 0.7];
i = [0.133e-12 1.79e-12 24.02e-12 321.66e-
772.58e-9];
lni = log(i); % Natural log of current

% Coefficients of Best fit linear model is
p_fit = polyfit(v,lni,1);
% linear equation is y = m*x + b
b = p_fit(2);
m = p_fit(1);
ifit = m*v + b;
% Calculate Is and n
Is = exp(b)
n = 1/(m*vt)
% Plot v versus ln(i), and best fit linear model
plot(v,ifit,'k', v, lni,'oc')
axis([0,0.8,-35,-10]); xlabel('Voltage (V)');  ylabel('ln(i)'); title('Best
fit linear model')
```

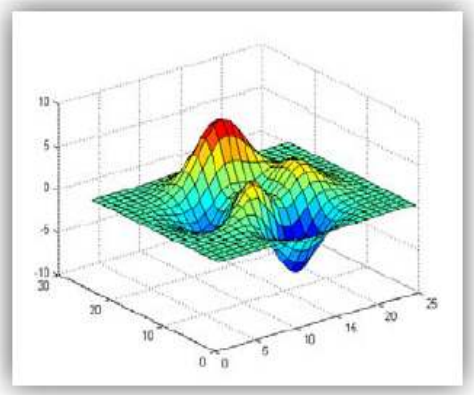
Visualization: Plotting Variables, Part 2

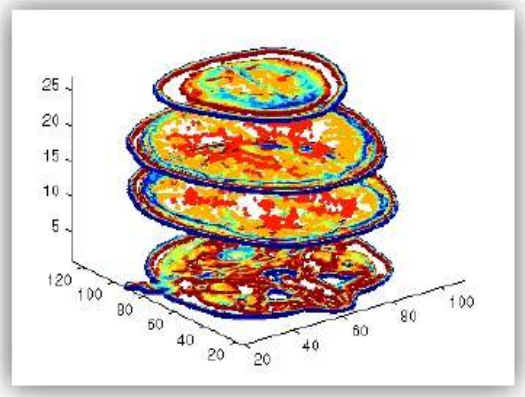
- 2D plots (plot, loglog, semilogx, semilogy, scatter, bar, polar, stem, fill, area, etc)
- 3D plots (plot3, cylinder, contour, meshgrid, mesh, meshc, meshz,surf, surfc, surfl, etc)

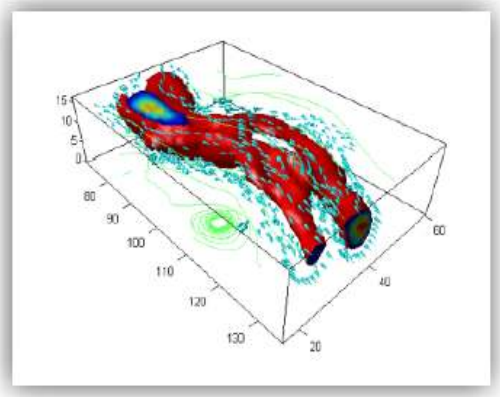












More on 2D Plots: fill, area, stem

example_2D3D_plots.m

```
%Ex 1
t= linspace(0,2*pi,200);
r= sqrt(abs(5*cos(3*t)));
x= r.*cos(t);
y= r.*sin(t);
figure(1); fill(x,y, 'k'); axis('square')

%Ex 2
x= linspace(-5*pi,5*pi,100);
y= cos(x)./x;
figure(2); area(x,y);
xlabel('x (rad)'); ylabel('cos(x)/x');

%Ex 3
t= linspace(0,2*pi,200);
f= exp(-0.6*t).*sin(t);
figure(3); stem(t,f)
```

3D Plots: Functions 1

- [X,Y] = meshgrid(x,y) transforms the domain specified by vectors x and y into arrays X and Y that can be used for the evaluation of functions of two variables and 3-D surface plots. The rows of the output array X are copies of the vector x and the columns of the output array Y are copies of the vector y.

- mesh(X,Y,Z,C) plots the colored parametric mesh defined by four matrix arguments. The view point is specified by VIEW. The axis labels are determined by the range of X, Y and Z, or by the current setting of AXIS. The color scaling is determined by the range of C, or by the current setting of CAXIS. The scaled color values are used as indices into the current COLORMAP. mesh(X,Y,Z) uses C = Z, so color is proportional to mesh height.

- meshc gives the combination mesh/contour plot. meshc(...) is the same as mesh(...) except that a contour plot is drawn

3D Plots: Functions 2

- contour(Z) is a contour plot of matrix Z treating the values in Z as heights above a plane. A contour plot are the level curves of Z for some values V. The values V are chosen automatically. In contour(X,Y,Z), X and Y specify the (x,y) coordinates of the surface as for SURF. contour is really a 2D plot

- surf(X,Y,Z,C) plots the colored parametric surface defined by four matrix arguments. surf(X,Y,Z) uses C = Z, so color is proportional to surface height. surfl uses light effects

- surfc gives the combination surf/contour plot. surfc(...) is the same as surf(...) except that a contour plot is drawn beneath the surface.

3D Plots: Functions 3

- plot3(x,y,z), where x, y and z are three vectors of the same length, plots a line in 3-space through the points whose coordinates are the elements of x, y and z.

- [X,Y,Z] = cylinder(R,N) forms the unit cylinder based on the generator curve in the vector R. Vector R contains the radius at equally spaced points along the unit height of the cylinder. The cylinder has N points around the circumference. surf(X,Y,Z) displays the cylinder. [X,Y,Z] = cylinder(R), and [X,Y,Z] = cylinder default to N = 20 and R = [1 1]. cylinder(R) causes the cylinder to be displayed with a surf command and no outputs to be returned.

- Check the rest of m-file example_2D3D_plots.m

3D Plots: Examples

- Ex 4
- `r= -7:0.2:7;`
- `[X,Y]= meshgrid(r,r);`
- `Z= -0.333*X.^2 + 2*X.*Y+Y.^2;`
- **`figure(4); cs= contour(X,Y,Z);`**
- Ex 5. Draw the surface plot of the curve described by these equations:

$$-4 \le x \le 4, -4 \le y \le 4, \qquad z = 2^{-1.5\sqrt{x^2+y^2}} sin(x) * cos(0.5 * y)$$

- `u=-4:0.25:4;  % it's better to decrease the points to see a more clear surface`
- `[x,y]= meshgrid(u,u);`
- `z= 2.^(-1.5*sqrt(x.^2+y.^2)).*sin(x).*cos(0.5*y);`
- **`figure(5); surf(x,y,z)`**
- `xlabel('x');ylabel('y');zlabel('z'); grid on;`

3D Plots: Examples, cont.

- Ex 6. Draw the 3D plot of the curve described by these equations:

$$|x| \le 7, |y| \le 7, \qquad z = cos(x) * cos(y) * e^{-\frac{\sqrt{x^2+y^2}}{5}}$$

- `u=-7:0.02:7;`
- `[x,y]= meshgrid(u,u);`
- `z= cos(x).*cos(y).*exp(-sqrt(x.^2+y.^2)/5);`
- **`figure(6); plot3(x,y,z);`** `grid on;`

- Ex 7. Plot the cylinder described by these equations:

$$0 \le z \le 1, 0 \le \theta \le 2\pi, \qquad r = sin(5\pi z) + 3$$

- `z=0:0.2:1;`
- `r=sin(5*pi*z)+3;`
- **`figure(7); cylinder(r)`**
- What happens when you change the step size?

3D Plots: Examples, cont.

- Ex8. Plot the 3D curve described by these parametric equations:

$$x = \sqrt{t}\,sin(3t), y = \sqrt{t}\,cos(3t), z = 0.8\,t, \qquad 0 \le t \le 6\pi$$

- Ex9. Draw the mesh, mesh and contour, surface, surface and contour plots

$$-4 \le x \le 4, -4 \le y \le 4, \qquad z = 2^{-1.5\sqrt{x^2+y^2}}\,sin(x) * cos(0.5 * y)$$

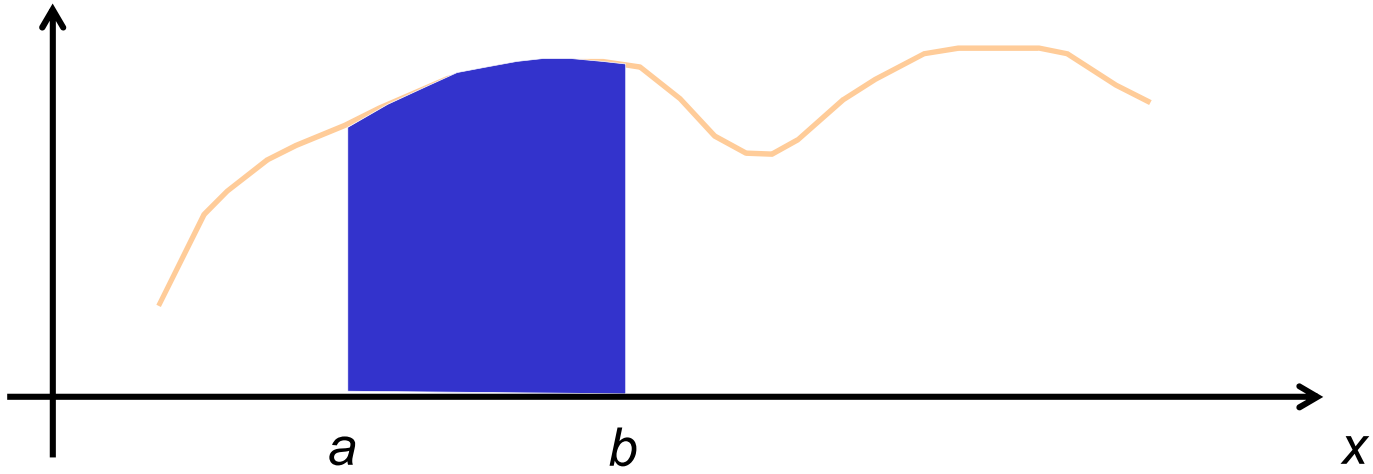
- ❖ Using the rotate 3D icon, you can change the view of these plots
- Ex10. Consider limits of x and y are not equal and repeat Ex9

$$-4 \le x \le 4, -5 \le y \le 5, \qquad z = 2^{-1.5\sqrt{x^2+y^2}}\,sin(x) * cos(0.5 * y)$$

- `x= -4:0.25:4; y= -5:0.25:5;`
- `[X,Y]= meshgrid(x,y);`
- `Z= 2.^(-1.5*sqrt(X.^2+Y.^2)).*sin(X).*cos(0.5*Y);`

Review of Integration, 1

- For a function f,



- The "integral of f from a to b" is the area under the graph of the function as shown in this figure.
- If f is continuous, then the area is well defined, as the common limit of upper and lower sums. The integral is shown by

$$\int_a^b f(x)\,dx$$

Review of Integration, 2

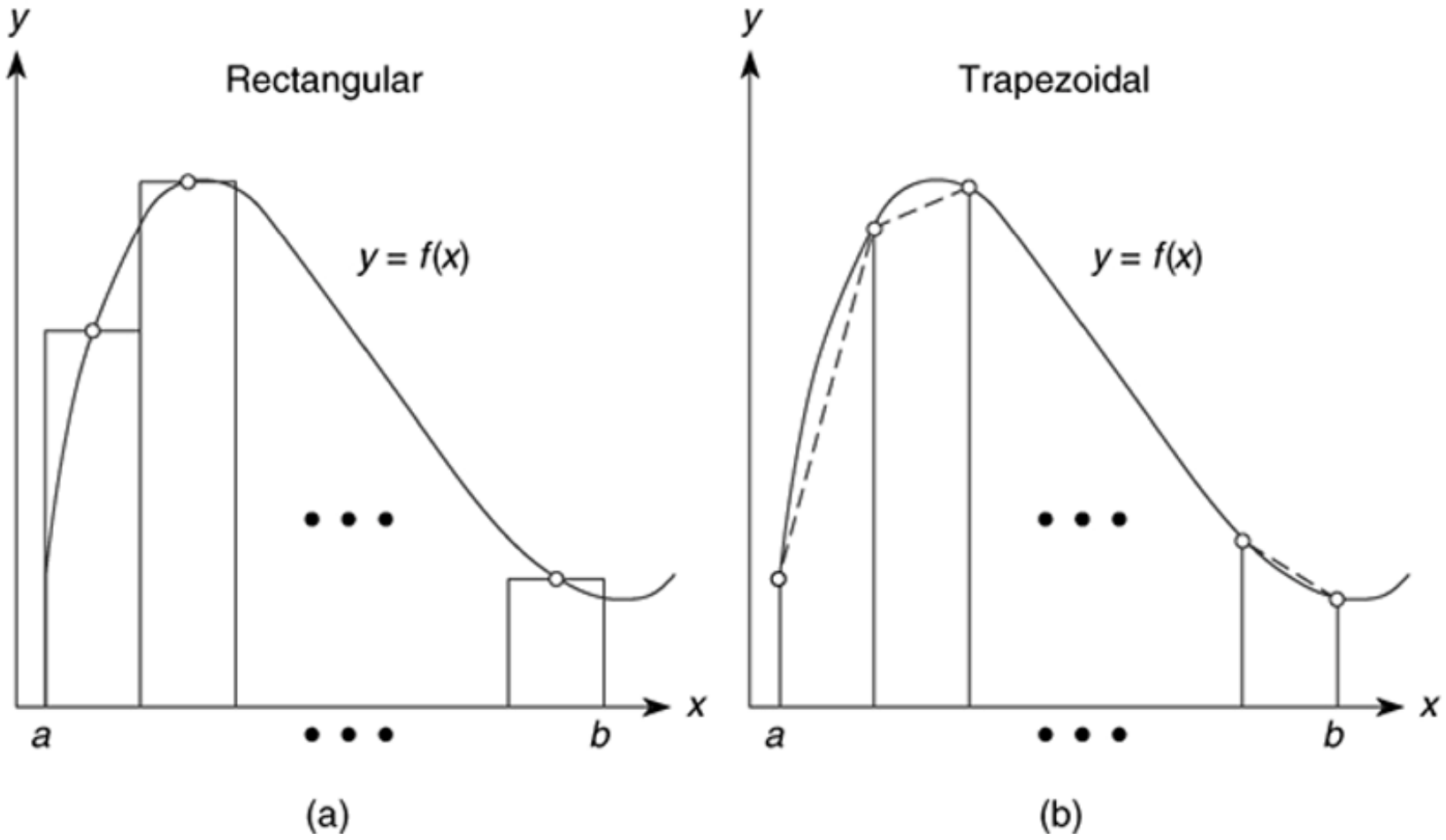
- If a function g is the antiderivative of f, namely $g'(x) = f(x)$

for all x, then the fundamental theorem of calculus says that the integral of f can be computed by finding the value of g at points a and b and subtracting them:

$$\int_a^b f(x)dx = g(b) - g(a)$$

- But, in so many cases in science, engineering or even economics, the functions do not have known antiderivatives, so we must use a finite number of functions to approximate the integral (or area).

Illustration of (a) rectangular and (b) trapezoidal numerical integration



Numerical Integration Method 1: Trapezoid Approximation

- Let $h=(b-a)/3$. The sum of all the approximations is

$$I = \frac{h}{2}(f(a)+f(a+h))+\frac{h}{2}(f(a+h)+f(a+2h))+\frac{h}{2}(f(a+2h)+f(b))$$
$$= \frac{h}{2}(f(a)+2f(a+h)+2f(a+2h)+f(b))$$

Numerical Integration Method 2: Simpson's Rule

- See the next slides

Derivation of Simpson's Rule: Suppose $f(x)=x^3$

$$\int_a^b x^3\,dx = \frac{1}{4}\left(b^4 - a^4\right)$$

$$= \frac{1}{4}(b-a)\left(b^3 + b^2a + ba^2 + a^3\right)$$

$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{1}{3}b^3 + b^2a + ba^2 + \frac{1}{3}a^3 + \frac{2}{3}a^3\right)$$

$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{1}{3}(b+a)^3 + \frac{2}{3}a^3\right)$$

$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{8}{3}\left(\frac{b+a}{2}\right)^3 + \frac{2}{3}a^3\right)$$

$$= \frac{1}{6}(b-a)\left(b^3 + 4\left(\frac{b+a}{2}\right)^3 + a^3\right)$$

$$= \frac{1}{6}(b-a)\left(f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right)$$

Generalization of Simpson's Rule

- For $f(x) = x,\ x^2$ etc, you can prove that

$$\int_a^b f(x)dx = \frac{1}{6}(b-a)\left(f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right)$$

- So, for any continuous function $f(x)$ that can be modeled like this, the Simpson's approximation to $\int_a^b f(x)dx$

is equal to:

$$\mathrm{I}_{\mathrm{Simpson}} = \frac{1}{6}(b-a)\left(f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right)$$

MATLAB Functions for Numerical Integration

Command	Description
`quad(fun,a,b)`	➢ Uses an adaptive Simpson's rule to compute the integral of the function whose handle is `fun`, with `a` as the lower integration limit and `b` as the upper limit. The function fun must accept a vector argument.
`quadl(fun,a,b)`	➢ Uses Lobatto quadrature to compute the integral of the function `fun`. The rest of the syntax is identical to `quad`.
`trapz(x,y)`	➢Uses trapezoidal integration to compute the integral of `y` with respect to `x`, where the array `y` contains the function values at the points contained in the array `x`.

Numerical Integration: Example

- Using the Adaptive Simpson's quadrature (the input must be a function)

```
q1 = quad(@(x) sin(x).*exp(x),0,pi)
q2 = quad('myfun3', 0,10)
```

- Using the Trapezoidal rule (the input must be a vector)

```
x = 0: 0.01: pi;
z1 = trapz(x,sin(x))
z2 = trapz(x,sqrt(exp(x))./x)
```

Class Exercise 1

1. Write an m-file that calculates the value of $\int_0^5 x\, e^{-x/3}\, dx$ using two different methods: a. Simpson's Rule and b. Trapezoidal rule.
2. Save the answers in different variables, and find the difference between your numerical answers and the analytical answer (which is equal to $-24\ e^{-5/3} + 9$).

Numerical Derivatives of Vectors, 1

- The function DIFF differences and approximates derivative.
- diff(X), for a vector X with n elements, is [X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)], with n-1 elements.
- diff(X), for a matrix X, is the matrix of row differences, [X(2:n,:) - X(1:n-1,:)].
- % Note: the length of diff's output is one element less than the original matrix/vector
- m-file example_diff_1
- `mat = [1 4 6; 9 3 5]    % a 2*3 matrix`
- `dm = diff(mat)    % first difference: a 1*3 matrix`
- `dm1_2 = diff(mat,1,2) % first difference along the second dimension (between columns) is a 2*2 matrix`
- `dm1_1 = diff(mat,1,1) % first difference along the first dimension (between rows) is a 1*3 matrix`
- `dm2_2 = diff(mat,2,2) % second difference along the second dimension (between columns) is a 2*1 matrix`
- `dm2_1 = diff(mat,2,1) % second difference along the second dimension (between rows) is an empty matrix`

Numerical Derivatives of Vectors, 2

B = diff(A)

resulting difference

array to difference

Example 1: A has both rows and columns

```
A = [1, 2, 6; 4, -7, 0];
B = diff(A(:));
C = diff(A);
```

The first non-singleton dimension is 1

dim 1 ↓ dim 2 →

$$A = \begin{bmatrix} 1 & 2 & 6 \\ 4 & -7 & 0 \end{bmatrix}$$

$$B = \text{diff}(A(:)) = \begin{bmatrix} 3 \\ -2 \\ -9 \\ 13 \\ -6 \end{bmatrix}$$

$$C = \text{diff}(A) = \begin{bmatrix} 3 & -9 & -6 \end{bmatrix}$$

Example 2: A has just one row

```
A = [1, 2, 6];
B = diff(A);
```

The first non-singleton dimension is 2

$$A = \begin{bmatrix} 1 & 2 & 6 \end{bmatrix}$$

$$B = \text{diff}(A) = \begin{bmatrix} 1 & 4 \end{bmatrix}$$

Example 3: A has just one column

```
A = [1; 4];
B = diff(A);
```

The first non-singleton dimension is 1

$$A = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

B = diff(A) = 3

Numerical Derivatives of Vectors, 3

example_diff_2

```
x = 0: 0.01:2*pi;
y= sin(x);
dydx= diff(y) ./ diff(x);
plot(x,y,'r',x(1,2:end),dydx,'b')
axis tight;
grid on;
text(3*pi/4,sin(pi/4),'\leftarrow
y=sin(x)','HorizontalAlignment','left')
text(3*pi/4,cos(3*pi/4),'\leftarrow
dy/dx=cos(x)','HorizontalAlignment','left‘)
```

The Gradient

- The *gradient* of a function of two variables, *F(x,y)*, is defined as $\nabla F = \frac{\partial F}{\partial x}\hat{\imath} + \frac{\partial F}{\partial y}\hat{\jmath}$

It is a collection of vectors pointing in the direction of increasing values of *F*. In MATLAB, we can compute numerical gradients (differences) for functions with any number of variables. For a function of *N* variables, *F(x,y,z)*,

- `v = -2:0.2:2; [x,y] = meshgrid(v); z = x .* exp(-x.^2 - y.^2); [px,py] = gradient(z,.2,.2); contour(v,v,z), hold on, quiver(v,v,px,py), hold off`

- `F(:,:,1) = magic(3); F(:,:,2) = pascal(3);` `gradient(F)` takes dx = dy = dz = 1.

- check example_grad.m

The Laplacian

- If the matrix U is regarded as a function u(x,y) evaluated at the point on a square grid, then 4*del2(U) is a finite difference approximation of Laplace's differential operator applied to u, that is

$$If\ U = u(x,y), then\ l = \frac{\nabla^2 u}{4} = \frac{1}{4}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

- `x=-5:0.25:5;`
- `y=-5:0.25:5;`
- `[x,y]=meshgrid(x,y);`
- `U=x.^2+y.^2;`
- `lap=4*del2(U);`
- `figure(1); meshc(x,y,U)`

- check example_laplacian.m
- You may study more about these functions in your Numerical Methods class

The Symbolic Math ToolBox, 1

- MATLAB has a Symbolic Math Toolbox, in which you can define the functions just like the way you do in Calculus, and find their derivatives, integrals, Laplace transform, etc, or solve Differential Equations. The student version of Matlab installed in the computer lab does not have this toolbox, but you can still use it on your laptops. This part will be specially useful for the topics on control theory and electrical circuits.
- `1/2 + 1/3`
- `sym('1/2') + sym('1/3')`

- Before using any parameter, you have to define it as a Symbolic variable (Class: sym)
- `syms x y`
- `simplify((x^3 - y^3)/(x - y))`

- `solve('x^2 - 2*x - 4 = 0')`

- `ezplot('x^2 + x + 1', [-2 2]) % easy to use function plotter`

- `syms x y`
- `factor(x^4-y^4)`

The Symbolic Math ToolBox, 2

- `syms x y`
- `simplify(cos(x)^2 - sin(x)^2)`
- Suppose you want to define *f(t)*, a function of the independent parameter *t*, and find its Laplace transform.
- `syms t`
- `f = 7*t^3*cos(5*t+pi/3);`
- `laplace(f)`

- You can also find the integral of a function using this toolbox:
- `syms x1`
- `int(x1*log(1+x1),0,1)` %the answer is $\frac{1}{4}$
- Compare it with the numerical method:
- `z=linspace(0,1,100);`
- `z2 = trapz(z,z.*log(1+z))`
- the answer is 0.25