MATH2221
Mathematics Laboratory II

Lecture 11:
Calculus and Optimization Using MATLAB

Gary Choi

April 1, 2025

Recall: Reading/writing files and path/file management

- Reading/writing files with specific data format
  - fopen, fprintf, fscanf, fclose
  - Easily specify number of spaces/ decimal places
    - Example:

      fprintf(datafile,'%11i',i);

      fprintf(datafile,'%11.5f\n',(i-32)*5/9);

- Path/file management
  - Change current directory:
    - cd('newfolderpath') , cd('..'), ...
  - Show all files in the current directory
    - dir, dir *.txt, ...
  - Include/exclude folders from your path:
    - addpath, rmpath

Fahrenheit	Celsius
0	-17.77778
1	-17.22222
2	-16.66667
3	-16.11111
4	-15.55556
5	-15.00000
6	-14.44444
7	-13.88889
8	-13.33333
9	-12.77778
10	-12.22222
11	-11.66667
12	-11.11111

Recall: Image processing in MATLAB

- Reading an image: imread
- Writing an image: imwrite
- Displaying an image:
  - imshow
  - imshowpair

- Image type conversion:
  - rgb2gray
  - imbinarize

- Image editing:
  - Imresize, imcrop, imadjust, imrotate, imwarp

Recall: Video processing in MATLAB

- Reading a video:
  v = VideoReader(filename)

- Reading a frame:
  - read(v,i)
  - hasFrame, readFrame

- Writing a video:
  - v = VideoWriter(filename,profile)
    open(v)
    writeVideo(v,...)
    close(v)

Calculus using MATLAB

- MATLAB can be used (and is indeed very powerful) for Calculus!

- **Differentiation**
  - Approximating derivatives and gradients

- **Integration**
  - Integrating numeric data
  - Integrating functional expressions

- **Differential equations**
  - Solving differential equations
  - Solving systems of differential equations

Differentiation using MATLAB

- **Difference operator**: Y = diff(X)
  - Calculates differences between adjacent elements of X
  - i.e. $Y = [X(2) - X(1),\ \ X(3) - X(2),\ \ \ldots,\ \ X(m) - X(m-1)]$
  - If length(X) = m, then length(Y) = $m - 1$

Example:

```
>> X = [0,1,4,9,16];
>> Y = diff(X)
Y =
    1    3    5    7
```

- Computing the n-th difference: Y = diff(X,n)
  - i.e. diff(X,2) = diff(diff(X)) and so on

Example:

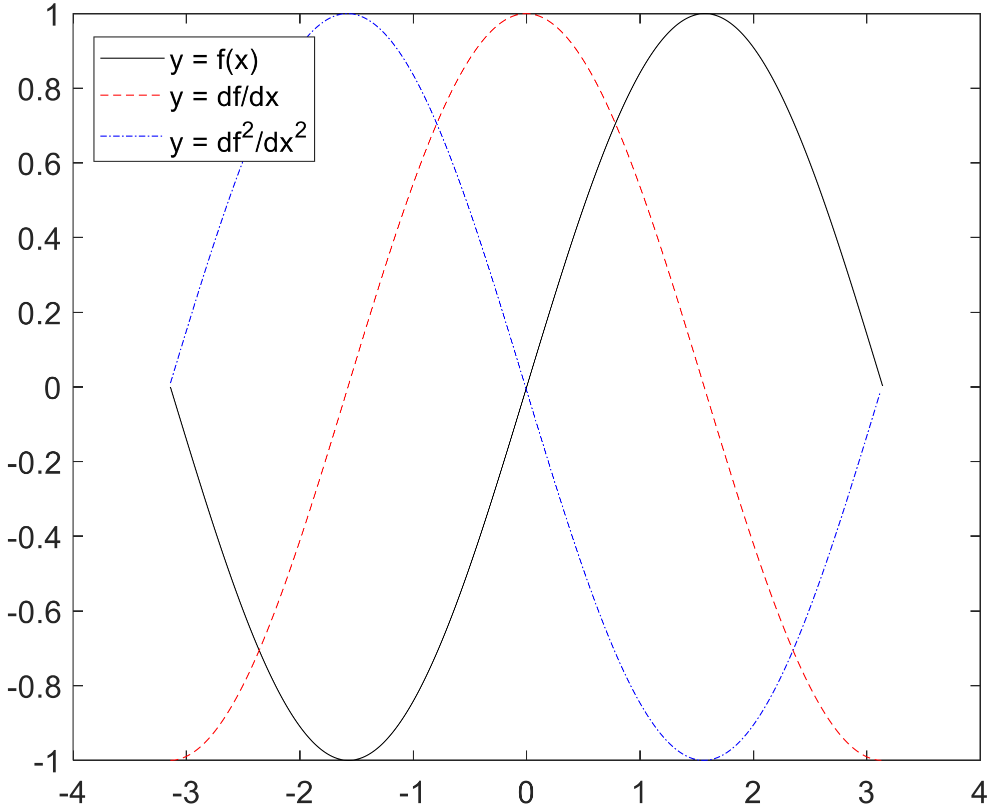
```
>> X = [0,1,4,9,16];
>> Y = diff(X,2)
Y =
    2    2    2
```

Differentiation using MATLAB

- **Approximate derivatives**: Y = diff(f)/h
  - f: some function values evaluated over some domain X
  - h: step size

- Example: For $f(x) = \sin x$, compute $f'(x), f''(\mathrm{x})$

```
h = 0.01;        % step size
X = -pi:h:pi;    % domain
f = sin(X);      % the function values
Y = diff(f)/h;   % first derivative
Z = diff(Y)/h;   % second derivative
figure;
plot(X,f,'k-');hold on;
plot(X(:,1:length(Y)),Y,'r--');
plot(X(:,1:length(Z)),Z,'b-.');
legend('y = f(x)','y = df/dx','y = df^2/dx^2');
```



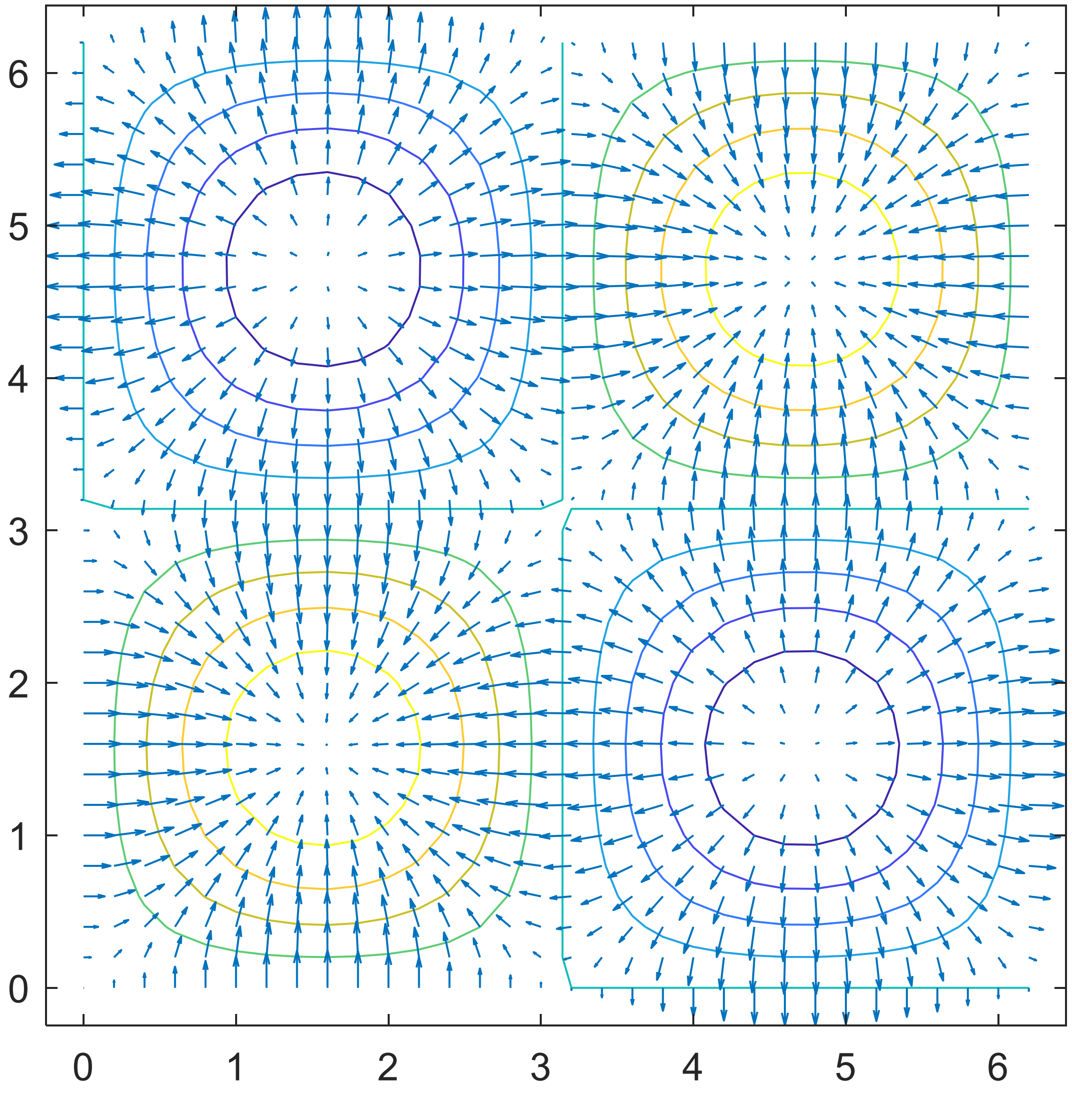
Differentiation using MATLAB

- **Gradient operator**: gradient
  - [FX,FY] = gradient(F)
    - F: a matrix
    - FX, FY: the partial derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$

- Example:

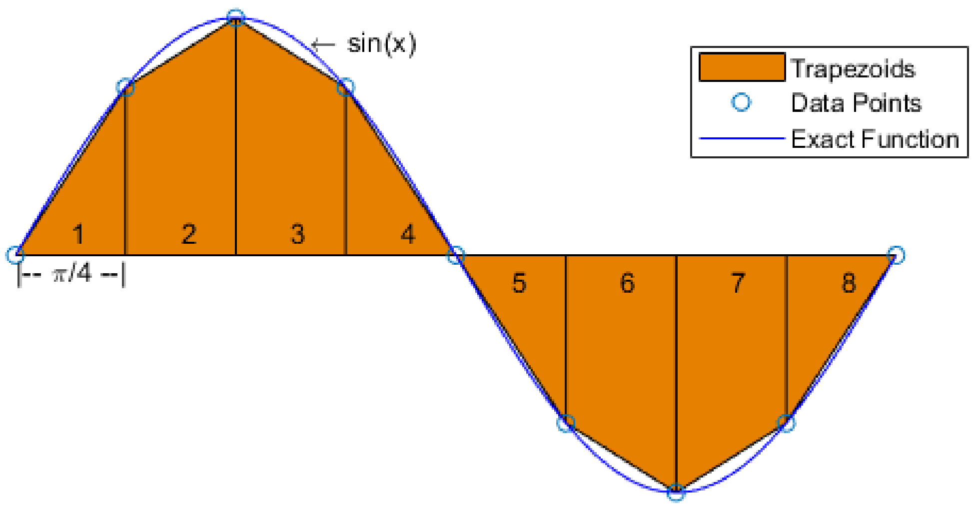
```
[X,Y] = meshgrid(0:0.2:2*pi, 0:0.2:2*pi);
Z = sin(X).*sin(Y);
[FX,FY] = gradient(Z);

figure;
contour(X,Y,Z);
hold on;
quiver(X,Y,FX,FY);
axis equal
```



- More general command: [FX,FY,FZ,...,FN] = gradient(F,hx,hy,...,hN)

Integration using MATLAB



- Integrating **numeric data**: trapz
  - Trapezoidal numerical integration

$$\int_a^b f(x) \approx \frac{1}{2}\sum_{n=1}^{N} \Delta x_n \big(f(x_n) + f(x_{n+1})\big)$$

where $a = x_1 < x_2 < \cdots < x_{N+1} = b$ and $\Delta x_n = x_{n+1} - x_n$

- MATLAB commands:
  - Q = trapz(Y): computes the approximate integral of Y with unit spacing
    - i.e. $Y = [f(x_1), f(x_2), \dots, f(x_{N+1})]$ and $\Delta x_n = 1$ for all n
  - Q = trapz(X,Y): integrates Y with respect to the spacing specified by X
    - i.e. $Y = [f(x_1), f(x_2), \dots, f(x_{N+1})]$ and $X = [x_1, x_2, \dots, x_{N+1}]$

Integration using MATLAB

- Example: Integrate $f(x) = x^2$ in the domain $[0,5]$.

  ```
  >> Y = [0, 1, 4, 9, 16, 25];
  >> Q = trapz(Y)
  Q =
       42.5000
  ```

  (Actual: $\left[\frac{x^3}{3}\right]_0^5 \approx 41.6667$)

- Example: Integrate $f(x) = \sin x$ from 0 to π

  ```
  >> X = 0:pi/100:pi;
  >> Y = sin(X);
  >> Q = trapz(X,Y)
  Q =
     1.9998
  ```

  (Actual: $[-\cos x]_0^{\pi} = 2$)

Integration using MATLAB

- We can also integrate **functional expressions**
- MATLAB command: q = integral(fun,xmin,xmax)
  - fun: a functional expression defined using a *function handle (@(x) …)* or a *function file (.m)*
  - xmin: lower limit
  - xmax: upper limit
  - can also handle improper integral
- Example: $\int_0^1 \sin(\cos(\tan x))\,dx$

```
>> f = @(x) sin(cos(tan(x)));
>> q = integral(f,0,1)
q =
   0.6596
```

- Example: $\int_0^\infty e^{-x^4}(\ln x)^3 dx$

```
>> f = @(x) exp(-x.^4).*log(x).^3;
>> q = integral(f,0,Inf)
q =
   -5.9905
```

Integration using MATLAB

- **Double integral**: q = integral2(fun,xmin,xmax,ymin,ymax)
  - Similar to the 1D case
  - xmin, xmax must be scalar
  - ymin, ymax can be scalar or function handle of x

- Example: Integrate $f(x,y) = \frac{\sin x + \cos y}{\sqrt{x^2+y^2+1}}$ over the triangular region bounded by $0 \le x \le 1$ and $0 \le y \le 1 - x$.

```
>> f = @(x,y) (sin(x) + cos(y))./sqrt(x.^2+y.^2+1);
>> ymax = @(x) 1-x;
>> q = integral2(f,0,1,0,ymax)
q =
   0.5388
```

- **Triple integral**: q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax)
  - Similar to the 2D case, where ymin, ymax can be function handles (*of x*) and zmin, zmax can be function handles (*of x and y*)

Solving differential equations using MATLAB

- Basic idea:
  - Replace derivatives with **finite difference**
    - $y'(x_n) \approx \frac{y_{n+1} - y_n}{h}$ (forward difference)
    - $y'(x_n) \approx \frac{y_n - y_{n-1}}{h}$ (backward difference)
    - $y'(x_n) \approx \frac{y_{n+1} - y_{n-1}}{2h}$ (central difference)
  - Then solve the differential equation using linear algebra methods and/or iterative schemes

- Example: Solve $\frac{dy}{dx} = f(x, y) = 2x - 3xe^y + 1$ with $y(0) = 0$
  - Using forward difference, we have

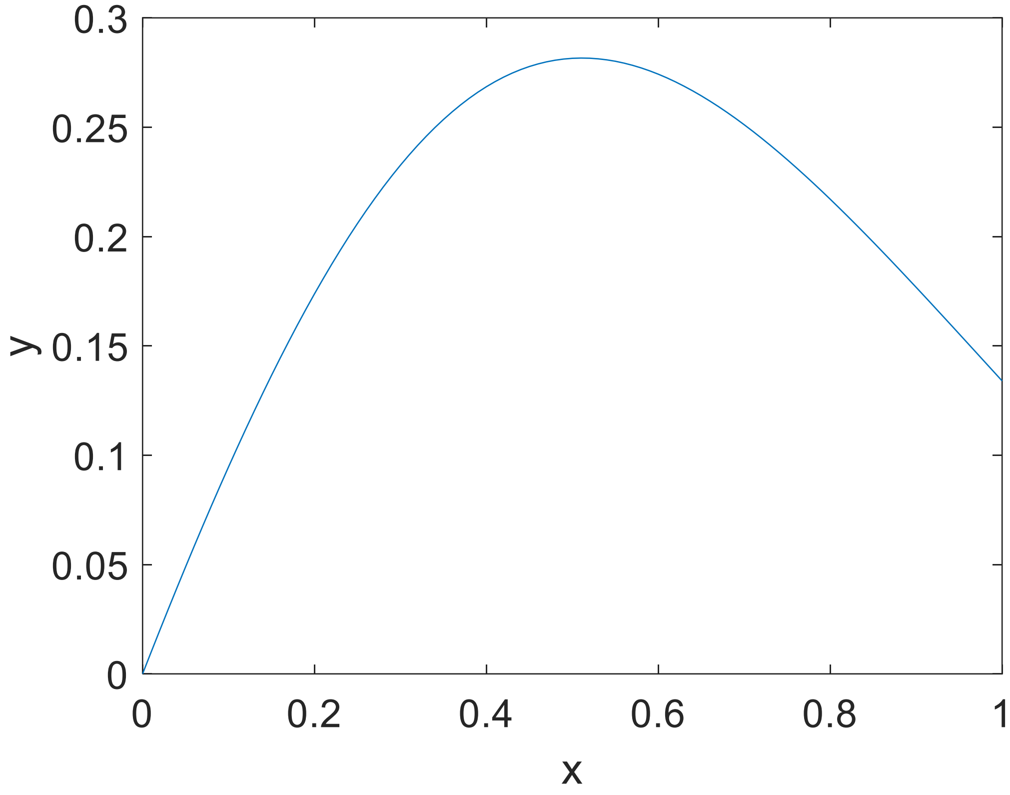
$$\frac{y_{n+1} - y_n}{h} = 2x_n - 3x_n e^{y_n} + 1 \Longrightarrow y_{n+1} = y_n + h(2x_n - 3x_n e^{y_n} + 1)$$

  which can be computed using a for-loop

Solving differential equations using MATLAB

- Example:

```
y = zeros(1,101);
x = linspace(0,1,101);
h = 0.01;
for n = 1:100
    y(n+1) = y(n)+h*(2*x(n) - 3*x(n)*exp(y(n))+1);
end
figure;
plot(x,y);
xlabel('x')
ylabel('y')
```



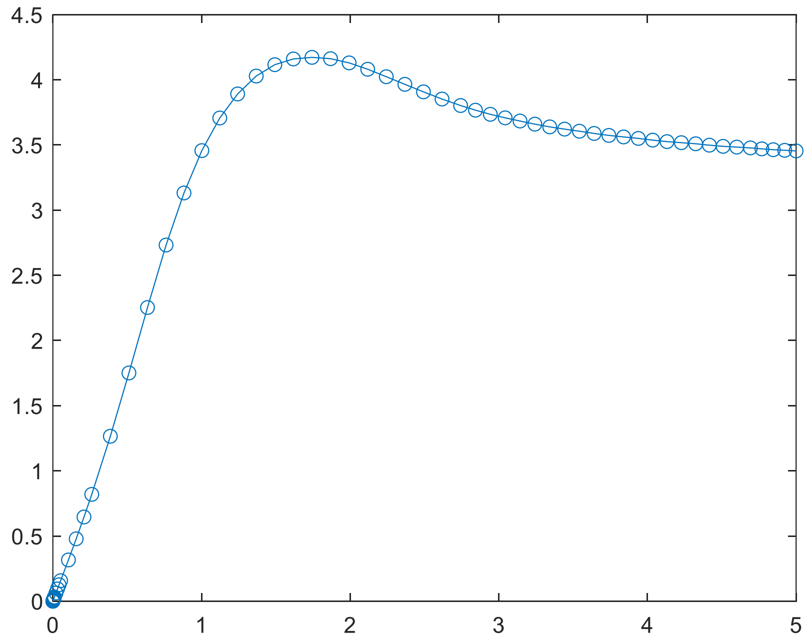
- Remark:
  - For backward difference, we have $\frac{y_n - y_{n-1}}{h} = f(x_n, y_n)$
  - May not be able to express y_n in terms of x_n, y_{n-1} explicitly
  - In this case, we may need to solve a matrix equation using \ or solve a nonlinear equation using fsolve or fzero

Solving differential equations using MATLAB

- A very powerful **ODE solver** in MATLAB: ode45
  - Solve $\frac{dy}{dt} = f(t, y)$ with $y(t_0) = y_0$
  - Based on a type of *Runge-Kutta* methods (see MATH3230 and 3240)

- MATLAB command: [t,y] = ode45(odefun,tspan,y0)
  - odefun: function handle or function file for $f(t, y)$
  - tspan: [t0, tf], where t0 is the starting point and tf is the ending point
  - y0: the initial condition (i.e. $y(t_0) = y_0$)

- Example: Solve $y' = 2t\sin y + 3$
  in the time interval [0,5] with $y_0 = 0$
  >> [t,y] = ode45(@(t,y) 2*t*sin(y)+3, [0,5], 0);
  >> figure;
  >> plot(t,y,'-o');



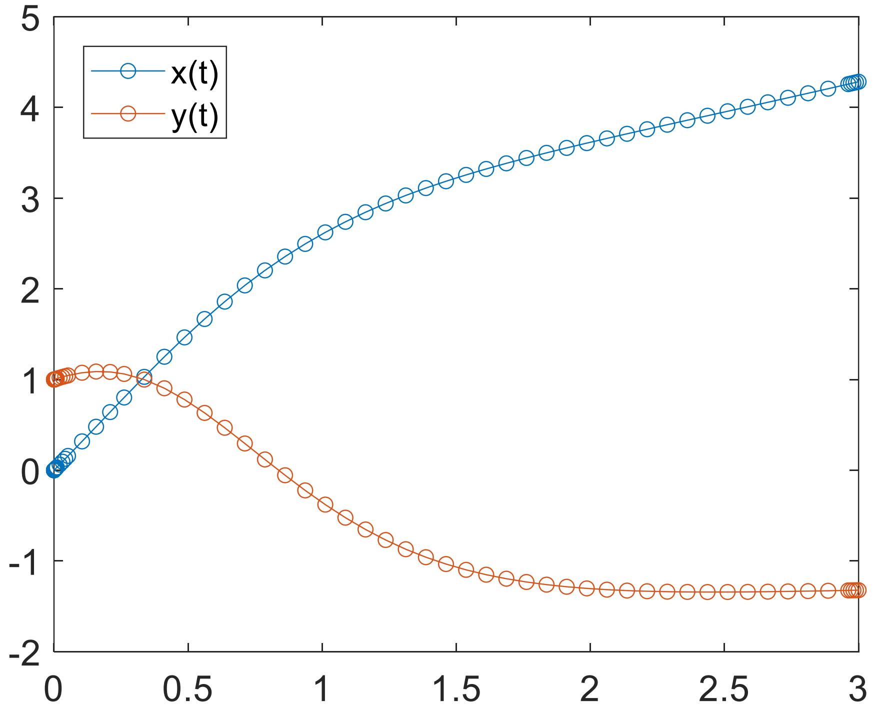
Solving differential equations using MATLAB

- Note: ode45 can also solve **systems of ODEs**
  - In [t,y] = ode45(odefun,tspan,y0), use a vector for odefun and y0

- Example: Solve $\begin{cases} x'(t) = y(t) + 2 \\ y'(t) = \big(1 - x(t)\big)y(t) - x(t) \end{cases}$

  with $x(0) = 0, y(0) = 1$ in the time interval [0,3]

```
[t,y] = ode45(@(t,y) [y(2)+2; ...
                (1-y(1))*y(2)-y(1)], [0,3], [0;1]);
figure;
plot(t,y,'-o');
legend('x(t)','y(t)')
```



Solving differential equations using MATLAB

- What about high-order ODE/ODE systems?
  - Try to rewrite them as first-order ODE systems
  - Then use ode45

- Example: $y'' + p(t)y' + q(t)y = r(t)$
  - Let $y_1(t) = y$, $y_2(t) = y'$
  - Then we have $\begin{cases} y_1' = y_2 \\ y_2' = r(t) - p(t)y_2 - q(t)y_1 \end{cases}$
  - *a first-order ODE system*!

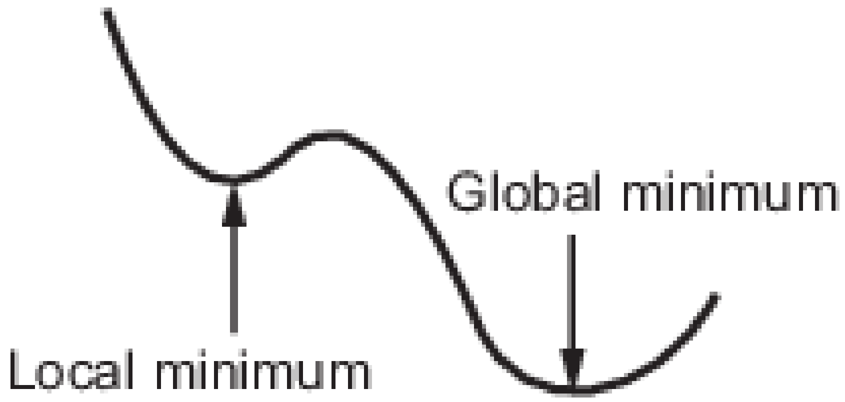
- More generally, every **n-th order linear ODE** can be rewritten as a **system of n first-order ODEs**

Optimization using MATLAB

- We can use MATLAB to solve various **optimization problems**!

  - **Unconstrained** optimization

  - **Constrained** optimization



- Remarks:
  - Similar to many numerical optimization schemes,
    MATLAB optimization solvers are typically based on gradient descent

  - They typically return a local minimum but not necessarily the global minimum

fminbnd: single-variable optimization problem on a fixed interval

- Find a (local) minimum of a single-variable function on a fixed interval

$$\min_{x} f(x) \quad \text{with} \quad x_1 < x < x_2$$

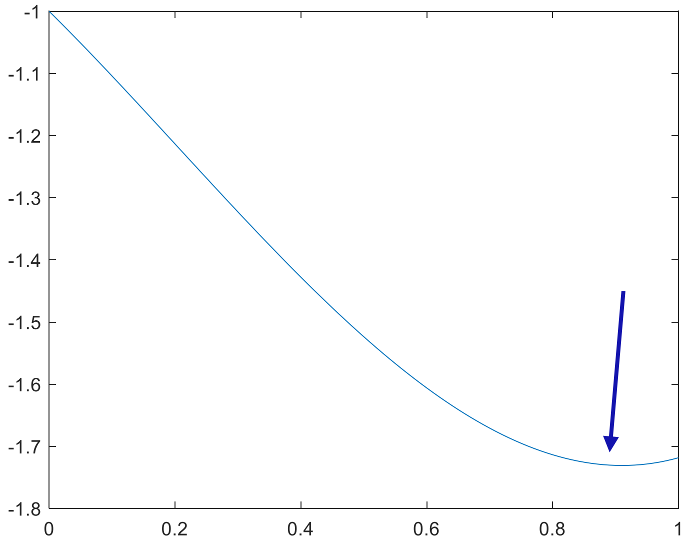
- MATLAB command: [x,fval] = fminbnd(fun,x1,x2)
  - fun: Function to minimize
  - x1: Lower bound
  - x2: Upper bound
  - x: the solution x
  - fval: the value of the objective function at x

- Example: Minimize $x^3 - e^x$ in $(0,1)$

```
f = @(x) x^3 - exp(x);
[x,fval] = fminbnd(f,0,1)
```

```
x =
    0.9100

fval =
   -1.7308
```



- More generally, we have x = fminbnd(fun,x1,x2,options)
  - options: control the displayed information, maximum number of iterations, etc.
  - e.g. options = optimset('Display','iter','MaxIter',20);

fminunc: Unconstrained optimization of multivariable function

- Find the minimum of a multivariable function without constraints

$$\min_{x_1,x_2,\ldots,x_n} f(x_1, x_2, \ldots, x_n)$$

- MATLAB command: [x,fval] = fminunc(fun,x0)
  - fun: Function to minimize
  - x0: the initial guess (a vector with the size being the number of variables in f)
  - x: the solution x (a vector)
  - fval: the value of the objective function at x

- Example: Minimize $f(x_1, x_2) = 3x_1^2 + 2x_1x_2 + x_2^2 - 4x_1 + 5x_2$

```
f = @(x)3*x(1)^2 + 2*x(1)*x(2) + x(2)^2 - 4*x(1) + 5*x(2);
x0 = [1,1]; % initial guess
[x,fval] = fminunc(f,x0)
x =                                  fval =
    2.2500   -4.7500                   -16.3750
```

fminunc: Unconstrained optimization of multivariable function

- More generally, we have [x,fval] = fminunc(fun,x0,options)
  - As in fminbnd, we can use options to control the displayed information, maximum number of iterations, etc.
  - e.g. options = optimset('Display','iter','MaxIter',20);

- Additionally, we can **specify the gradient** to improve the optimization process
  - Use a function file (.m) to create fun with *both the function and the gradient*
  - Set options = optimoptions('fminunc', 'SpecifyObjectiveGradient',true)

- Example: $f(x_1, x_2) = x_1^2 + x_2 + 100 \sin x_1 x_2$

Step 1: create a function

```
function [f,g] = myfun(x)
% the objective function f
f = x(1)^2 + x(2)+ 100*sin(x(1)*x(2));
% the gradient g
g = [2*x(1) + 100*x(2)*cos(x(1)*x(2));
    1 + 100*x(1)*cos(x(1)*x(2))];
end
```

Step 2: run fminunc

```
>> x0 = [0,1];
>> fun = @myfun;
>> options = optimoptions('fminunc', …
'SpecifyObjectiveGradient', true);
>> [x,fval] = fminunc(fun,x0,options)
x =
   -0.9205    1.6947
fval =
  -97.4521
```

fmincon: Constrained optimization of multivariable function

- Find minimum of constrained nonlinear multivariable function

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

- MATLAB command: [x,fval] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
  - Can set the inputs as [ ] if they are not applicable

- Example: Minimize $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$ with $x_1 + 2x_2 \le 1$

```
fun = @(x)100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
x0 = [-1,2];
A = [1,2];
b = 1;
[x, fval] = fmincon(fun,x0,A,b)
```

```
x =
    0.5022    0.2489

fval =
    0.2489
```

Reminder: Lab 9 this week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17		

Lecture 1- Lecture 13

Lab 1 - Lab 10 (40%)

Test 1 (30%)
Test 2 (30%)

Thank you!

Next time:

- Symbolic computation using MATLAB