Chapter 5
Loops & Vectorization
Loops

- **Loop** construct permits user to execute a block of instructions more than once.
- Base on how the repetition is controlled, there is two basic forms of loops:
  - **While** loops
    - repeat an indefinite number of times until a user-specified condition is satisfied
  - **For** loops
    - repeat a pre-determined number of times
Vectorization

- **Vectorization** is an alternative way to perform repeated instructions, and it is faster than using loops.
  - Basically, it is using array and matrix operators to replace sequences of iterations.
while Loop

- There are two types of while loop
  - while loop (or called repeat until loop)
  - do-while loop (MATLAB didn’t support this)
while Loop

- The general form of a **while** loop

```
while expression
  statement_block
end
```

- If the controlling expression returns **true**, the inner statements will be executed.
- The process will be repeated until the expression becomes **false**.
Example 5.1: Statistical Analysis

- Statistical Analysis
  - 1. State the problem:
    - The average or arithmetic mean of a set of numbers is defined as
    $$\mu = \frac{1}{N} \sum_{i=0}^{N-1} x_i$$
    - The standard deviation $\sigma$ is
    $$\sigma^2 = \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i - \mu)^2$$
    $$\sigma^2 = \frac{1}{N-1} \left[ \sum_{i=0}^{N-1} x_i^2 - \frac{1}{N} \left( \sum_{i=0}^{N-1} x_i \right)^2 \right]$$
Example 5.1: Statistical Analysis

2. Define the inputs and outputs:
   - Reads in a set of numbers and computes the mean and the standard deviation.

3. Define the algorithm:
   - Use a while loop to read a set of numbers
   - Compute the mean
   - Compute the variance and the standard deviation is its square-root
Example: Statistical Analysis

```matlab
% stdtest

% input data
done=false;
xin=[];
disp('keyin an array & 0 for quit')

while ~done
    keys=input('keyin: ');
    if keys==0
        done=true;
    else
        xin=[xin, keys];
    end
end

N=length(xin);

% computer mean & std
mean=sum(xin)/N;
std2=(sum(xin.^2)-sum(xin).^2/N)/(N-1);
std=sqrt(std2);

fprintf('N=%d, mean=%.3f, std=%.3f
',... 
    N,mean,std);
```

```
>> clear
>> edit stdtest
>> stdtest
keyin an array & 0 for quit
keyin: [11 5 43 12 56]
keyin: [28 13 25 44]
keyin: 0
```

```
N=9, mean=26.333, std=17.847
```
for Loop

- The general form of a **for** loop

```
for index=expr
  statement_block
end
```

- Variable `index` is the iteration counter and the output of `expr` should be an array.
- No. of iterations matches the size of columns.
- For the nth iteration, the index was assigned the nth column of the array.
for Loop\(^2\)

- The index can be set as:
  - output from a colon operation: (e.g. \([3:2:17]\))
  - output from a function: (e.g. \(\text{linspace}(2,3,5)\))
  - a set of numbers, (e.g. \([5\ 9\ 7]\))
  - a matrix (e.g. \([1\ 2\ 3;\ 4\ 5\ 6]\))
for Loop

- If the index array was modified during the execution of loop, the number of iteration will not be affected. However, the loop outcomes may be affected.
Example 5.2: Factorial Function

- **Statistical Analysis**
  1. **State the problem:**
     - The factorial function is defined as
     \[
     n! = \prod_{k=1}^{n} k \quad \forall n \in \mathbb{N}
     \]
  2. **Define the algorithm:**
     - With \( 0! = 1 \).
     \[
     n! = \begin{cases} 
     1 & \text{if } n \leq 1 \\
     n(n-1)! & \text{if } n > 1 
     \end{cases} \quad \forall n \in \mathbb{N}
     \]
Example 5.2: Factorial Function

```matlab
% factest.m

% input n
n=input('input n: '); 

% set result to 0!
res=1;

% compute n-factorial
fprintf('%d\!', n);
for i=n:-1:2
    fprintf('%dx', i);
    res=res*i;
end
fprintf('1=%d\n', res);
```

Command Window

```matlab
>> clear
>> edit factest
>> factest
input n: 5
5! = 5x4x3x2x1 = 120
>> factest
input n: 8
8! = 8x7x6x5x4x3x2x1 = 40320
9!
```
Example 5.3: Date of the Year

- **Date of the Year**
  1. **State the problem:**
     - Write a MATLAB script that accept a day and a year, then calculate the month, day, and weekday corresponding to the given date.
  2. **Define the algorithm:**
     - We need a reference, say Sat. 12/31/1904.
     - 1904 was a leap year and any day after that day starts at the first day of the week, a Sunday.
     - Find the total day from the reference to the target day, and then the reminder of 7 days is the weekday.
Example 5.3: Day of the Year

```matlab
% datetest.m

% initialization
wkdsstr={'Sun','Mon','Tue','Wed','Thr','Fri','Sat'};
monstr={'Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'};
mdays=[31,29,31,30,31,30,31,31,30,31,30,31]; % leap
31,28,31,30,31,30,31,31,30,31,30,31];
days=[366,365];

% day of the year for the last day of each month
ydays=mdays;
for i=2:12
    ydays(1,i)=ydays(1,i)+ydays(1,i-1);
ydays(2,i)=ydays(2,i)+ydays(2,i-1);
end

% get specified data from user
date_in=input('input [day_of_the_year, year]: '); 
the_day=date_in(1); the_year=date_in(2);

% determine whether is a leap year
yr_idx=mod(the_year,4)+1;
if yr_idx>1;
yr_idx = 2;
else
    fprintf('year %d is a leap year\n', the_year);
end
```
Example 5.3: Day of the Year

```matlab
% find the month and day of the month
done=false; the_mon=1;
while ~done
    if the_day<=ydays(yr_idx,the_mon)
        done=true;
    else
        the_mon=the_mon+1;
    end
end
if the_mon>1
    day_of_mon=the_day-ydays(yr_idx, the_mon-1);
else
    day_of_mon=the_day;
end

% find total days after Sat. 12/31/1900
ref_wkd=6; ref_yr=1905;
n_yr=the_year-ref_yr;
n_4yr=floor(n_yr/4);
n_days=n_yr*days(2)+n_4yr+the_day;
the_wkd=mod(ref_wkd+n_days,7)+1;

% output
fprintf('%d day of %d is %s %s %d\n',
    the_day, the_year, wkdstr{the_wkd},
    monstr{the_mon}, day_of_mon);
```
Practical Tips

- Indent the bodies of loops. It makes the code much more readable.
- Pre-allocating arrays. Even though it is easy to extend an existing array:
  - If `a=1:4` and then `a(8)=3` will result in `a=[1 2 3 4 0 0 0 3]`.
  - Internally, MATLAB did some extra work:
    - Create a new array `tmp`;
    - Copy the contents of `a` to `tmp`;
    - Add the new value to the last element in `tmp`;
    - Delete `a` and rename `tmp` as `a`. 
Vectorization

- Operations performed over vectors are faster than those over loops
- Comparing loops and vectorization:
  - To initialize a $1 \times n$ vector and then computer squares of this vector for $m$ times.
    - Use loop without initializing the array;
    - Use loop with pre-allocated array;
    - Use vectorization.
  - Use **tic** function to set a build-in elapsed time counter, and **toc** function returns the elapsed time in seconds since the last call to function **tic**.
Vectorization

```matlab
% benchmark.m

% decide how many cycles of 10*10000
n=input('number of cycles: ');
maxcnt=10;
dataSize=10000;

% loop w/o pre-allocation
tic;
for k=1:n*maxcnt
    clear data
    for i=1:dataSize
        data(i) = i^2;
    end
end
tUsage1=(toc);
fprintf(’%-21s: %7.4f sec
’,...
'loop w/uninitialized', tUsage1);
```
Vectorization³

```
% loop w/ pre-allocation
tic;
for k=1:n*maxcnt
    clear data
    data=zeros(1,dataSize);
    for i=1:dataSize
        data(i) = i^2;
    end
end
tUsage2=(toc);
fprintf('%-21s: %7.4f sec\n',...
    'loop w/initializezation', tUsage2);
%
% vectorize operations
 tic;
for k=1:n*maxcnt
    clear data
    i=1:dataSize;
    data=i.^2;
end
tUsage3=(toc);
fprintf('%-21s: %7.4f sec\n',...
    'vectorization', tUsage3);
```
Break and Continue

- Statements to control the flow of loops:
  - `break` causes the loop to terminate prematurely.
  - `continue` causes the loop to skip a pass, but continue on the rest of the loop.
  - Both are used in conjunction with an `if` statement.
  - Note that if the `break` or `continue` is used in a nested loops then it terminates or skips the current loop only.
Example: `continue`
Example: **break**

```matlab
% testbreak
% a modified n-factorial

% input n,m
n=input('input n: ');
m=input('break at factor of m: ');

% set result to 0!
res=1;

% compute n-factorial
fprintf('%d!',n);
for i=n:-1:2
    if (mod(i,m) == 0)
        break;
    end
    fprintf('%dx', i);
    res=res*i;
end
fprintf('1=%d\n', res);
```

```
>> clear
>> edit testbreak
>> testbreak
input n: 10
break at factor of m: 5
10!=1=1
>> testbreak
input n: 10
break at factor of m: 4
10!=10x9x8x7x1=90
>> testbreak
input n: 10
break at factor of m: 6
10!=10x9x8x7x1x6=5040
```
Nested Loops

- If one loop is completely inside another one, the two loops are called nested loops.
- Each loop in nested loops should have a distinctive loop index variable.
- When MATLAB encounters an end or a break statement, it associates the statement with the innermost currently executing for loop construct.
Example: Multiple Table

```matlab
function multitable
% printing misc
fprintf('  
for i=1:9
  fprintf('%2d ',i)
end
fprintf('
 * 
for i=1:9
  fprintf('---')
end
fprintf('\n')

% outer loop
for i=1:9
  fprintf('%d|',i)
  % inner loop
  for j=1:9
    res=i*j;
    fprintf('%2d ', res)
  end
end
fprintf('
')
end
```
Logical Arrays

- Logical data has one of two possible values: **true** (1) or **false** (0).

- Scalars and arrays of logical data are created as the output of relational and logic operators.

```
>> clear
>> a=[1:4;5:8]
a =
7
5 2 3 4
3
6 7 8
>> b=mod(a,2)==0
b =
1
0 1 0 1
0 1 0 1
>> whos
Name      Size         Bytes  Class
a         2x4          64  double
b         2x4          8  logical
```
Logical Mask

- Logical arrays can serve as a **mask** for arithmetic operations.
- A mask is an array that selects the elements of another array for use in an operation.
- For example:
  - Take the square of all the elements that are odd numbers.

![Image showing workspace and command window examples](image-url)
Logical Mask & Vectorization

- For example:
  - Take the square root of all the elements that are greater than 5. Two approaches: by using
    - either `if` construct in a `for` loop construct
    - or a logical mask.
Logical Mask & Vectorization

```matlab
% tstlogic

% use if construct
b=a;
tic;
for i=1:size(a,1)
    for j=1:size(a,2)
        if a(i,j)>5
            a(i,j)=sqrt(a(i,j))
        end
    end
end
fprintf('%9.6f sec\n',(toc));

% use logical mask
a=b;
tic;
mask=a>5;
a(mask)=sqrt(a(mask));
fprintf('%9.6f sec\n',(toc));
```

Command Window

```matlab
>> edit tstlogic
>> a=round(rand(5,8)*10)
a =
6 9 2 2 3 10 6 1
4 9 3 2 9 4 3 3
8 6 5 2 4 1 6 3
5 6 2 2 2 3 7 4
4 6 8 4 9 4 2 5
>> tstlogic
0.000071 sec
0.000031 sec
>> a=round(rand(5,8)*10)
a =
1 7 10 6 0 3 1 9
3 5 5 7 9 3 7 9
8 6 5 4 9 7 5 3
0 2 2 4 8 1 8 7
9 5 5 10 1 7 7 2
>> tstlogic
0.000166 sec
0.000063 sec
```
Homework Assignment #10

- Quiz 5.1
  - Page 198: 6-10
- This homework is due by next week.
- Late submission will be penalized.
MATLAB Profile

- MATLAB profile can be used to identify the parts of a program that consume the most execution time.
- Profile can be started by selecting the Run and Time button from the Editor window.
- Or use ‘profile on’ in command window before running a program, then use ‘profile viewer’ to show the profile window.
% circles.m

% initialize primary data
a=0; b=0; dd=.15;
t=0:.05:2*pi;
dim=length(t);
r=1:dd:10;
colors = 'rygbcmkrygbcmkrygbcmkrygbcmkrygbcmkrygbcmk';

% draw first set of circles
subplot(1,2,1)
for i=1:length(r)
    x=r(i)*cos(t)+a;
    y=r(i)*sin(t)+b;
    x(dim+1)=x(1); y(dim+1)=y(1);
    plot(x,y,colors(i),'LineWidth',1.25)
    hold on
end
axis image
hold off

% draw second set of circles
subplot(1,2,2)
for i=1:length(r)
    x=r(i)*cos(t)+a;
    y=r(i)*sin(t)+b;
    x(dim+1)=x(1); y(dim+1)=y(1);
    plot(x,y,colors(i),'LineWidth',1.25)
    a = a + dx; b = b + dy;
end
axis image
hold off

>> edit circles
>> profile on
>> circles
>> profile viewer
### Profile Summary
Generated 18-Mar-2016 00:03:52 using cpu time.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Calls</th>
<th>Total Time</th>
<th>Self Time*</th>
<th>Total Time Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>circles</td>
<td>1</td>
<td>0.160 s</td>
<td>0.010 s</td>
<td></td>
</tr>
<tr>
<td>newplot</td>
<td>42</td>
<td>0.070 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>graphics\private\clc</td>
<td>2</td>
<td>0.070 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>cla</td>
<td>2</td>
<td>0.070 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>newplot&gt;ObserveAxesNextPlot</td>
<td>42</td>
<td>0.070 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>setdiff</td>
<td>4</td>
<td>0.060 s</td>
<td>0.010 s</td>
<td></td>
</tr>
<tr>
<td>setdiff&gt;setdifflegacy</td>
<td>4</td>
<td>0.050 s</td>
<td>0.020 s</td>
<td></td>
</tr>
<tr>
<td>hold</td>
<td>44</td>
<td>0.030 s</td>
<td>0.030 s</td>
<td></td>
</tr>
<tr>
<td>axis</td>
<td>2</td>
<td>0.030 s</td>
<td>0.010 s</td>
<td></td>
</tr>
<tr>
<td>axis&gt;LocSetTight1</td>
<td>2</td>
<td>0.020 s</td>
<td>0.010 s</td>
<td></td>
</tr>
<tr>
<td>unique</td>
<td>4</td>
<td>0.020 s</td>
<td>0.010 s</td>
<td></td>
</tr>
<tr>
<td>axis&gt;LocSetTight</td>
<td>2</td>
<td>0.020 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>axis&gt;LocSetImage</td>
<td>2</td>
<td>0.020 s</td>
<td>0.000 s</td>
<td></td>
</tr>
<tr>
<td>subplot</td>
<td>2</td>
<td>0.010 s</td>
<td>0.010 s</td>
<td></td>
</tr>
</tbody>
</table>
Noisy Measurements

- It is known fact that we can never make perfect measurements from a science experiment.
- It means that the measurements always include some noise.
- In order to estimate the theoretical relationship, a method of linear regression is used.
- Given a noisy set of measurements, we can find a line $y=mx+b$ to best fit the data.
Noisy Measurements

- A standard method for finding the regression coefficients $m$ and $b$ is the method of least squares.
  - The slope of the least-squares line is given by
    \[ m = \frac{\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} = \frac{n \sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \]
  - The intercept of the least-squares line is given by
    \[ b = \bar{y} - m \bar{x} \]
Example 5.6

- Write a program to compute the least-squares slope $m$ and y-axis intercept $b$ for a given set of noisy measured points $(x, y)$. 
Example 4.6

```matlab
% lsqrfit.m
%
% input
clear x y; x(1:20)=0; y(1:20)=0;
done=false; n=0;
while ~done
    pin=input('input [x y], 0 to end: ');
    if pin==0
        done=true;
        continue;
    end
    n=n+1;
    x(n)=pin(1); y(n)=pin(2);
end
x(n+1:end)=[]; y(n+1:end)=[];

% calculate all the summations
sum_x=sum(x); sum_y=sum(y);
sumxy=sum(x.*y); sumx2=sum(x.^2);

% calculate the slope and intercept
num=(n*sumxy-sum_x*sum_y);
denom=(n*sumx2-sum_x^2);
m=num/denom;
b=sum_y/n-m*sum_x/n;
printf('y = %.3f x + %.3f', m, b);

% plot points and least squares fit line
plot(x, y, 'bo')
hold on
xmin=min(x); xmax=max(x);
ymin=m*xmin+b; ymax=m*xmax+b;
plot([xmin xmax], [ymin ymax], 'r--', 'Linewidth',2);
title('least squared fit', 'color','b')
grid on
hold off
```
Example 4.7

- Assume that a ball is thrown with an initial velocity $v_0$ at an angle of $\theta$ degree. Write a program to plot the trajectory of the ball.
  - Determine the horizontal distance travelled before it hits the ground.
  - Plot the trajectories of the ball for all angles from 5 to 85° in 10° steps.
  - Determine the angle that maximizes the range of the ball.
Example 4.7²

- Physics analysis
  - If $t$ is the time the ball will remain in the air, then
    \[
    x(t) = x_0 + v_0 \cos \theta \ t \\
    y(t) = y_0 + v_0 \sin \theta \ t + 1/2 \ g \ t^2
    \]
    with $y_0=0$, we have the ball at ground again at
    \[ t = 2 \ \frac{v_0 \sin \theta}{g} \]
  - With $x_0=0$,
    \[
    x(\theta) = 2 \ v_0 2 \cos \theta \sin \theta / g = K_0 \sin(2\theta) \\
    d/d\theta \ x = 2 \ K_0 \cos(2\theta).
    \]
    Theoretically, the max happens at $\theta=45^\circ$
% initial data
v0 = 20;  color = 'rkbmcrcmbcmr';
xmax = 0;  d_max=0;  k=1;
hold on
for d_i=5:10:85
    d_r = d_i*pi/180;
vsthetar = v0 * sin(d_r);
vcthetar = v0 * cos(d_r);
t = 2 * vsthetar / 9.8;
np=40;  i=2;
x=zeros(1,50);  y=zeros(1,50);
for tt=linspace(0, t, 50);
    x(i)=vcthetar*tt;
    y(i)=vsthetar*tt -.5*9.8*tt^2;
    i=i+1;
end
cst=['-o' color(k)];  k=k+1;
plot(x, y, cst, 'MarkerSize', 3);
if (xmax_i > xmax)
    xmax = xmax_i;
d_max = d_i;
end
hold off
grid on
fprintf('the max occurs at %.2f degree

x >> |
The `textread` function

- It is more convenience to read large amount of data from a file.
- The `textread` function reads ASCII files that are formatted into columns of data, where each column can be of a different type, and stores the contents of each column in a separate output array.
- The syntax: `[x,y,...]=textread(fname, fmt,[n])`
  - Format examples: ‘%s’ or ‘%f %d’
The `textread` function

- Example: to read the contents of a file named as `tst-data.txt`.

![Notepad and Command Window](image)
Homework Assignment #11

- 5.8 Exercises
  - Page 220: 5.1, 5.3, 5.7, 5.8, 5.9, 5.23
- This homework is due by the next week.
- Late submission will be penalized.