مقدمه ای بر برنامه نویسی با MATLAB

سعید سهیلی
استادیار گروه مکانیک
دانشگاه آزاد اسلامی مشهد


بسم الله الرحمن الرحیم

فصل چهارم

حلقه ها

حلقه ها

- ساختارهایی هستند که اجازه تکرار فرامین را می دهند.
- دو نوع ساختار حلقه وجود دارد:
- حلقه های while: کدهای درون این حلقه به تعداد نامشخص تا رسیدن به شرایط مورد نظر تکرار می شوند.

- حلقه های for: کدهای درون این حلقه ها به تعداد مشخص تکرار می شوند.

حلقه های while

- کدهای درون این حلقه به تعداد نامشخص تا رسیدن به شرایط مورد نظر تکرار می شوند.

```
while conditional expression
    ........
    ........        A group of
    ........        MATLAB commands.
end
```

- تا زمانی که *مقدار شرط* درست باشد، حلقه تکرار می شود.

long as its value is equal to or smaller than 15.

```
x=1
while x<=15
    x=2*x
end
```

Initial value of x is 1.

The next command is executed only if x <= 15.

In each pass x doubles.

```
x =
     1
x =
     2
x =
     4
x =
     8
x =
    16
```

Initial value of x.

In each pass x doubles.

When x = 16, the conditional expression in the `while` command is false and the looping stops.

مثال) آنالیز آماری

- برنامه ای بنویسید که مجموعه ای از اندازه گیری ها را بخواند و مقدار متوسط و انحراف معیار مجموعه داده های ورودی را محاسبه کند.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1)}}$$

- هنگامی که تمامی داده ها تمام شد باید روشی را بیابیم که به برنامه اطلاع اتمام داده ها داده شود. مثلا ورود یک داده منفی

1. **بیان مساله:** محاسبه مقدار متوسط و انحراف معیار مجموعه ای از اعداد مثبت یا صفر. از آنجایی که تعداد اعداد مشخص نیست از ورودی منفی به عنوان پایان ورود داده ها استفاده شود.
2. **تعریف ورودی ها و خروجی ها:** ورودی ها تعداد نامشخص از اعداد مثبت یا صفر و خروجی ها مقدار متوسط، انحراف معیار و تعداد داده های ورودی هستند.
3. **طراحی الگوریتم:** ابتدا باید داده ها وارد و جمع اعداد و جمع مربعات اعداد محاسبه شود.

```
Initialize n, sum_x, and sum_x2 to 0
Prompt user for first number
Read in first x
while x >= 0
   n ← n + 1
   sum_x ← sum_x + x
   sum_x2 ← sum_x2 + x^2
   Prompt user for next number
   Read in next x
end
```

- در قدم بعدی مقدار متوسط و انحراف معیار محاسبه می شود.

```
x_bar ← sum_x / n
std_dev ← sqrt((n*sum_x2 - sum_x^2) / (n*(n-1)))
```

- در مرحله آخر نتایج چاپ می شوند

```
Write out the mean value x_bar
Write out the standard deviation std_dev
Write out the number of input data points n
```

4. تبدیل الگوریتم به دستورات Matlab

5. امتحان برنامه: برای سه مقدار ۳و۴و۵ محاسبات را به صورت دستی انجام می دهیم و با نتایج برنامه مقایسه می کنیم.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{3}(12) = 4$$

$$s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1)}} = 1$$

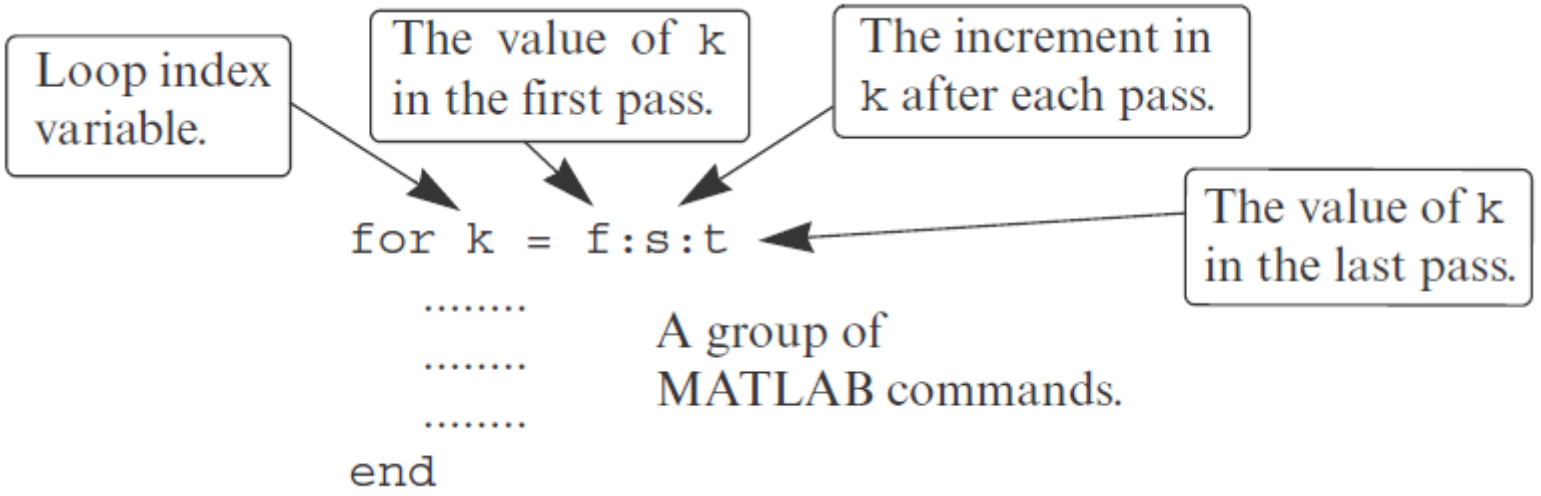
```
» stats_1
Enter first value: 3
Enter next value: 4
Enter next value: 5
Enter next value: -1
The mean of this data set is: 4.000000
The standard deviation is:    1.000000
The number of data points is: 3.000000
```

حلقه های for

- دسته ای از عبارات را به تعداد مشخصی اجرا می کند.

```
for index = expr
    ...
    ...
    ...
end
```

Body



```
for ii = 1:10
   Statement 1
   ...
   Statement n
end
```

- در عبارت فوق اندیس کنترلی یک آرایه 1*10 تولید می کند و عبارات داخل حلقه ۱۰ بار انجام می شوند.

```
for ii = 1:2:10
   Statement 1
   ...
   Statement n
end
```

- در عبارت فوق اندیس کنترلی یک آرایه 1*5 تولید می کند و عبارات داخل حلقه ۵ بار انجام می شوند.

```
for ii = [5 9 7]
   Statement 1
   ...
   Statement n
end
```

- در اینجا عبارت کنترلی یک آرایه 3*1 است و حلقه ۳ بار اجرا می شود.
- ii در بار اول ۵، بار دوم ۹ و در بار سوم ۷ است.

```
for ii = [1 2 3;4 5 6]
   Statement 1
   ...
   Statement n
end
```

- عبارت کنترلی یک آرایه 3*2 است و عبارت داخل حلقه ۳ بار اجرا می شوند.
- ii در بار اول بردار ستونی $\begin{bmatrix}1\\4\end{bmatrix}$ بار دوم $\begin{bmatrix}2\\5\end{bmatrix}$ و در بار سوم $\begin{bmatrix}3\\6\end{bmatrix}$ است.

```
for k=1:3:10
     x = k^2
end
```

```
>> x =
     1

x =
    16

x =
    49

x =
   100
```

مثال) تابع فاکتوریل

$$n! = \begin{cases} 1 & n = 0 \\ n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 & n > 0 \end{cases}$$

```
n_factorial = 1
for ii = 1:n
n_factorial = n_factorial * ii;
end
```

مثال) آنالیز آماری

- تعداد مشخصی از اعداد خوانده شود و مقدار متوسط و انحراف معیار محاسبه شود.

مثال) مجموع یک سری

1. **بیان مساله:** مجموع سری زیر را با گرفتن تعداد جملات مورد نیاز محاسبه کنید.

$$\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$$

2. **ورودی ها:** تعداد جملات

   خروجی ها: مجموع سری

3. **الگوریتم برنامه:**

دریافت تعداد جملات از کاربر ← n

```
for  k = 1:n
      S ← S+(-1)^k*k/2^k;
end
```

نمایش حاصل

4. تبدیل به کد matlab

```
n=input('Enter the number of terms ' );
S=0;
for k=1:n
    S=S+(-1)^k*k/2^k;
end
fprintf('The sum of the series is: %f',S)
```

Setting the sum to zero.

for-end loop.

In each pass one element of the series is calculated and is added to the sum of the elements from the previous passes.

5. تست برنامه

```
Enter the number of terms 4
The sum of the series is: -0.125000
```

```
Enter the number of terms 20
The sum of the series is: -0.222216
```

برخی ریزه کاری ها در کار با حلقه ها

- Index یک حلقه را درجایی درون حلقه نباید تغییر داد.

```
for ii = 1:10
   ...
   ii = 5;    % Error!
   ...
   a(ii) = <calculation>
end
```

- قبل از اجرای حلقه تمامی آرایه های استفاده شده در حلقه را از پیش بسازید. با این کار سرعت افزایش می یابد.

```
square = zeros(1,100);
for ii = 1:100
   square(ii) = ii^2;
end
```

برداری کردن: روشی سریعتر بجای حلقه

- در بسیاری از مواقع می توان محاسبات حلقه را به وسیله بردار انجام داد.
- کد زیر مربعات، جذر و ریشه سوم تمامی اعداد بین ۱و ۱۰۰ را محاسبه می کند.

```
tic
for ii = 1:100
    square(ii) = ii^2;
    square_root(ii) = ii^(1/2);
    cube_root(ii) = ii^(1/3);
end
toc
```

- همان محاسبات را به کمک بردارها می توان انجام داد.

```
tic
ii = 1:100;
square = ii.^2;
square_root = ii.^(1/2);
cube_root = ii.^(1/3);
toc
```

- محاسبات با حلقه حدود ۱۵ بار کندتر است.

عبارت break و continue

- break اجرای حلقه را متوقف و به ادامه برنامه را بعد از حلقه انجام می دهد.

```
for ii = 1:5
   if ii == 3;
      break;
   end
   fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);
```

```
» test_break
ii = 1
ii = 2
End of loop!
```

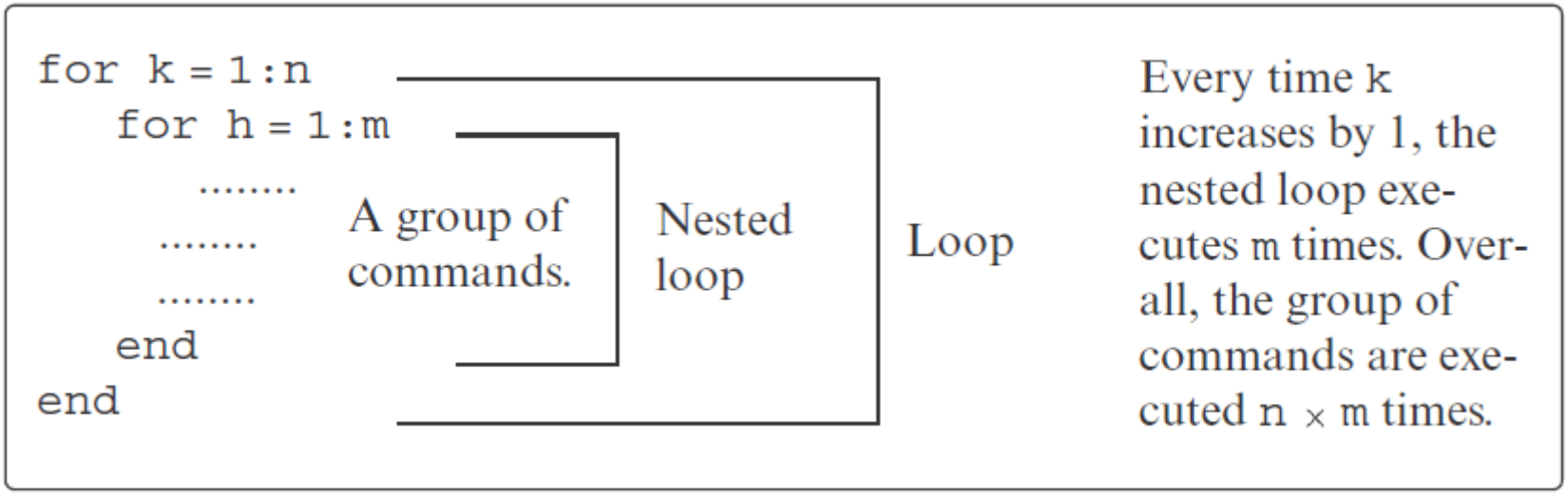
- continue اجرای حلقه را متوقف اما به ابتدای حلقه باز می گردد و حلقه را مجددا اجرا می کند.

```
for ii = 1:5
   if ii == 3;
      continue;
   end
   fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);
```

```
» test_continue
ii = 1
ii = 2
ii = 4
ii = 5
End of loop!
```

حلقه های تو در تو

- این امکان وجود دارد که یک حلقه کاملا داخل حلقه دیگر باشد



- حلقه بیرونی مقدار یک را میگیرد و سپس حلقه درونی سه بار اجرا می شود.

```
for ii = 1:3
   for jj = 1:3
      product = ii * jj;
      fprintf('%d * %d = %d\n',ii,jj,product);
   end
end
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

آرایه های منطقی و برداری کردن

- آرایه های منطقی درست بودن (1) یا غلط بودن (0) یک عبارت را مشخص می کنند.

```
a = [1 2 3; 4 5 6; 7 8 9];
b = a > 5;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```
» whos
 Name            Size            Bytes  Class
 a               3x3                72  double array
 b               3x3                 9  logical array
Grand total is 18 elements using 81 bytes
```

- از آرایه های منطقی می توان به عنوان یک ماسک استفاده نمود.
- عبارت زیر جذر کلیه عناصری که آرایه منطقی b غیرصفر است را محاسبه می کند.

```
» a(b) = sqrt(a(b))
a =
    1.0000    2.0000    3.0000
    4.0000    5.0000    2.4495
    2.6458    2.8284    3.0000
```

- عبارت زیر همین کار را با استفاده از حلقه انجام می دهد. اما نسبت به روش برداری کندتر است.

```
for ii = 1:size(a,1)
   for jj = 1:size(a,2)
      if a(ii,jj) > 5
         a(ii,jj) = sqrt(a(ii,jj));
      end
   end
end
```

- از آرایه های منطقی برای انجام یک سری عملیات روی عناصر منتخب یک آرایه استفاده می شود.

معادل سازی ساختار if/else با آرایه های منطقی

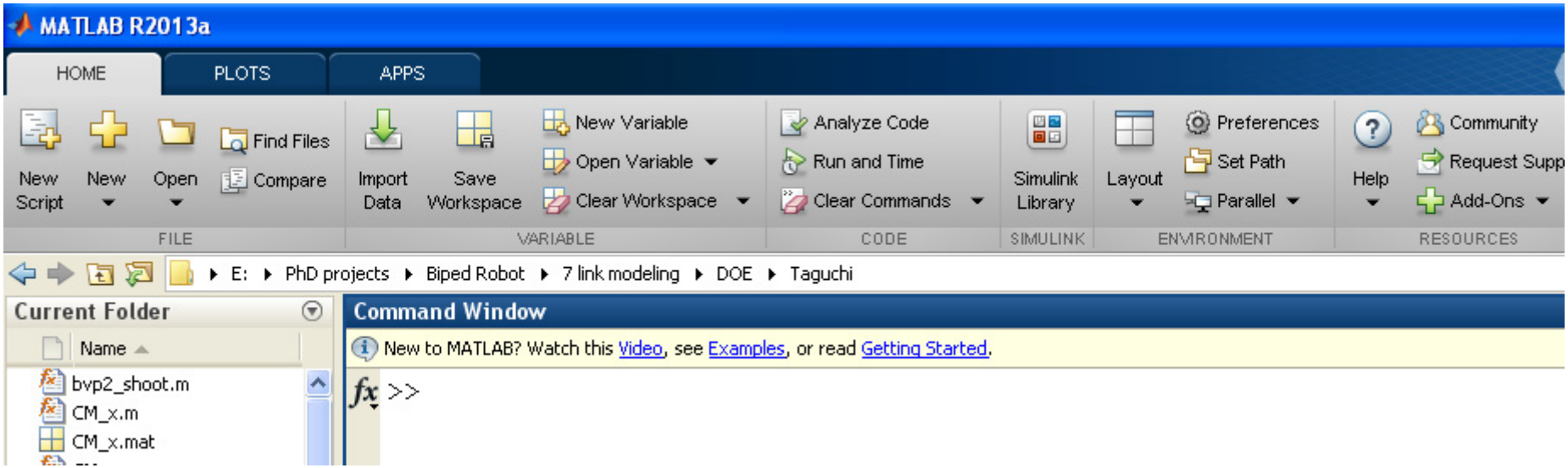
- با اضافه کردن عملگر not (~) می توان روی عناصر انتخاب نشده آرایه عملیاتی متفاوت انجام داد.
- به عنوان مثال) جذر تمام عناصر بزرگتر از ۵ را به دست آورده شود و سایر عناصر به توان دو بر سند.

```
for ii = 1:size(a,1)
   for jj = 1:size(a,2)
      if a(ii,jj) > 5
         a(ii,jj) = sqrt(a(ii,jj));
      else
         a(ii,jj) = a(ii,jj)^2;
      end
   end
end
```

```
b = a > 5;
a(b) = sqrt(a(b));
a(~b) = a(~b).^2;
```

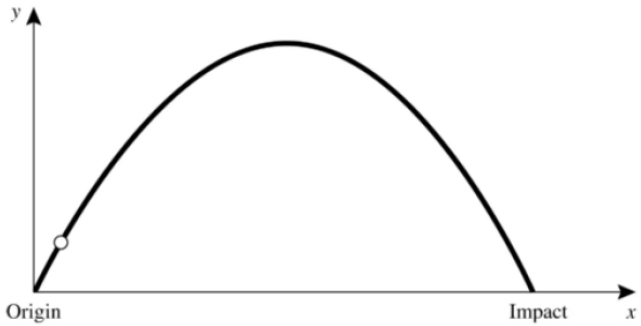
نمایه MATLAB (The MATLAB Profiler)

- با استفاده از نمایه matlab می توان آن قسمتی از کد را مشخص نمود که زمان اجرای بیشتری را مصرف می کنند.



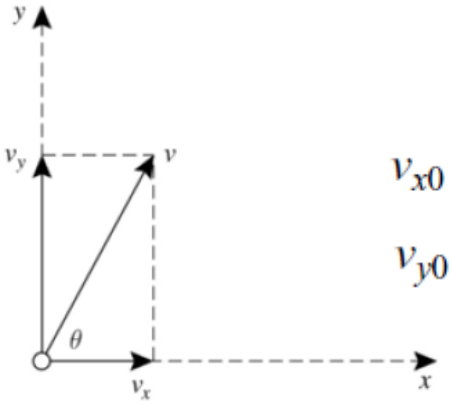
مثال)حرکت پرتابی

- در حرکت پرتابی با صرف نظر از اصطکاک هوا وانحنای زمین، مسیر حرکت سهموی است.



$$y(t) = y_0 + v_{y0}t + \frac{1}{2}gt^2$$

$$x(t) = x_0 + v_{x0}t$$



$$v_{x0} = v_0 \cos\theta$$

$$v_{y0} = v_0 \sin\theta$$

- با فرض سرعت اولیه 20m/s، برنامه ای بنویسید که مسیر طی شده را رسم و مسافت افقی طی شده را محاسبه کند. مسیر توپ برای تمام زوایای ۵ تا ۸۵ در گام های ۱۰ درجه رسم شود. مسافت افقی برای زوایای بین ۰ تا ۹۰ با گام ۱ درجه انجام شود. در نهایت زاویه ای که برای آن مسافت طی شده ماکزیمم می شود را تعیین کند.

- زمان برخورد به زمین به صورت زیر محاسبه می شود.

$$y(t) = y_0 + v_{y0}t + \frac{1}{2}gt^2$$

$$0 = 0 + v_{y0}t + \frac{1}{2}gt^2$$

$$0 = \left(v_{y0} + \frac{1}{2}gt\right)t \qquad t_2 = -\frac{2v_{y0}}{g}$$

1. **بیان مساله:** مسافتی که توپ بعد از پرتاب با سرعت اولیه ۲۰ متر بر ثانیه و زوایای اولیه θ می کند، محاسبه نمایید. این مسافت برای تمام زوایای بین ۰ تا ۹۰ با گام های ۱ درجه محاسبه نمایید. زاویه ای که منجر به بیشترین مسافت توپ می شود را به دست آورید. مسیر حرکت توپ را برای تمام زوایای بین ۵ تا ۸۵ با گام های ۱۰ درجه رسم کنید. مسیر حرکت بیشینه را به یک رنگ دیگر و یک خط دیگر مشخص کنید.

2. **تعریف ورودی ها و خروجی ها:** ورودی خاصی ندارد.

خروجی ها شامل جدولی از مسافت های طی شده برای زوایای مختلف و زاویه ای که منجر به مسافت ماکزیمم می شود. همچنین نمودارهای نمایش دهنده مسیر حرکت توپ.

- طراحی الگوریتم:

```
Calculate the range of the ball for θ between 0 and 90°
Write a table of ranges
Determine the maximum range and write it out
Plot the trajectories for θ between 5 and 85°
Plot the maximum-range trajectory
```

- برای محاسبه مسافت طی شده از فرمول های زیر استفاده می شود.

$$v_{x0} = v_0 \cos\theta$$

$$v_{y0} = v_0 \sin\theta$$

$$y(t) = y_0 + v_{y0}t + \frac{1}{2}gt^2$$

$$x(t) = x_0 + v_{x0}t$$

$$t_2 = -\frac{2v_{y0}}{g}$$

```
Create and initialize an array to hold ranges
for ii = 1:91
   theta ← ii - 1
   vxo ← vo * cos(theta*conv)
   vyo ← vo * sin(theta*conv)
   max_time ← -2 * vyo / g
   range(ii) ← vxo * max_time
end
```

- در مرحله بعد باید جدول مسافت های طی شده تشکیل شود.

```
Write heading
for ii = 1:91
   theta ← ii - 1
   print theta and range
end
```

- ماکزیمم مسافت طی شده توسط تابع max

```
[maxrange index] ← max(range)
Print out maximum range and angle (=index-1)
```

- از حلقه های for تو در تو برای محاسبه و رسم مسیر حرکت ها استفاده می شود.

```
for ii = 5:10:85

   % Get velocities and max time for this angle
   theta ← ii - 1
   vxo ← vo * cos(theta*conv)
   vyo ← vo * sin(theta*conv)
   max_time ← -2 * vyo / g

   Initialize x and y arrays
   for jj = 1:21
      time ← (jj-1) * max_time/20
      x(time) ← vxo * time
      y(time) ← vyo * time + 0.5 * g * time^2
   end
   plot(x,y) with thin green lines
   Set "hold on" after first plot
end
Add titles and axis labels
```

- نهایتا ماکزیمم مسیر حرکت با رنگ دیگر مشخص شود.

```
vxo ← vo * cos(max_angle*conv)
vyo ← vo * sin(max_angle*conv)
max_time ← -2 * vyo / g

Initialize x and y arrays
for jj = 1:21
   time ← (jj-1) * max_time/20
   x(jj) ← vxo * time
   y(jj) ← vyo * time + 0.5 * g * time^2
end
plot(x,y) with a thick red line
hold off
```

4. تبدیل الگوریتم به عبارت MATLAB

5. **آزمایش برنامه:** جواب ها برای چند زاویه به صورت دستی محاسبه می شود.

θ	$v_{x_0} = v_0 \cos\theta$	$v_{y_0} = v_0 \sin\theta$	$t_2 = -\frac{2v_{y_0}}{g}$	$x = v_{x_0} t_2$
0°	20 m/s	0 m/s	0 s	0 m
5°	19.92 m/s	1.74 m/s	0.355 s	7.08 m
40°	15.32 m/s	12.86 m/s	2.621 s	40.15 m
45°	14.14 m/s	14.14 m/s	2.883 s	40.77 m

نکات مفید برای برنامه نویسی

- حلقه while برای تکرار بخشی از برنامه که تعداد تکرار آن از قبل مشخص نیست مورد استفاده قرار می گیرد.
- حلقه for برای تکرار بخشی از برنامه که تعداد تکرار آن از قبل مشخص است مورد استفاده قرار می گیرد.
- هیچ گاه شمارنده حلقه for را درون برنامه تغییر ندهید.
- برای افزایش سرعت آرایه های استفاده شده در حلقه را قبل از شروع حلقه مقداردهی کنید.
- از روش برداری کردن بجای حلقه در صورت امکان استفاده شود.
- از آرایه های منطقی بجای ساختار if در صورت امکان استفاده کنید.