4. Hafta - 2

Döngüler ile program akış denetimi

`for` döngüleri
`while` döngüleri
`break - continue`

Örnekler:
Seri hesapları, karekök algoritması

Program Akış Kontrolü

Program akışı aşağıdaki kontrol yapıları ile kontrol edilir:

1. for ... end `% döngüler`
2. while ... end
3. break, continue
4. if ... end `% koşullu`
5. if ... else ... end
6. if ... elseif ... else ... end
7. switch ... case ... otherwise ... end
8. return

for-döngüleri ve koşullu if'ler en çok kullanılan kontrol yapılarıdır.

for - döngüleri

```
for variable = expression
     statements ...
end
```

satır vektör veya matris

```
for k = [1,2,3,4,5]
   x = 3.0 + 0.1*k;
end
x =
   3.1000
x =
   3.2000
x =
   3.3000
x =
   3.4000
x =
   3.5000
```

```
for k = 1:5
   x = 3.0 + 0.1*k;
end
x =
   3.1000
x =
   3.2000
x =
   3.3000
x =
   3.4000
x =
   3.5000
```

`for` döngülerinin bilinen tipleri

sınırlar tamsayı
a<=k<=b

```
for k = a:b
    ...
end
```

artırımlı

```
for k = a:s:b
    ...
end
```

`val` = herhangi bir satır vektör

```
for k = val
    ...
end
```

`val` = herhangi bir matris de olabilir

```
for k = val
    ...
end
```

```
k1 = [1; 0; -2];
k2 = [0; 3; 1];

for k = k1
    x = 3.0 + 0.1*k;
end

x =
   3.1000
   3.0000
   2.8000
```

```
[k1, k2]

ans =
     1   0
     0   3
    -2   1
```

```
k1 = [1; 0; -2];
k2 = [0; 3; 1];

for k = [k1, k2]
    x = 3.0 + 0.1*k;
end

x =
   3.1000
   3.0000
   2.8000

x =
   3.0000
   3.3000
   3.1000
```

matris

`for` döngüleri `if` durumlarını içerebilir

```
g = [92, 45, 90, 80, 94, 75];
count = 0;

for k = 1:length(g)
    if g(k) >=90                    % ya da daha basiti,
        count = count + 1;          % if-end durumu yerine
    end                             % count = count + (g(k) >= 90);
end

disp(count)
    3
```

```
count = sum(g >= 90); % vektörleştirilmiş versiyon

disp(count)
    3
```

for döngüleri veya while döngüleri ile toplam hesabı

$$S = \sum_{k=1}^{N} \frac{1}{k^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{N^2}$$

```
N = 1000; S = 0;
for k=1:N
    S = S + 1/k^2;
end

S

S =
   1.6439
```

```
k=1:N; S = sum(1./k.^2);  % vektörleştirilmiş versiyon
S =
   1.6439
```

`if` durumları `for` döngülerini içerebilir

```
type = 'odd';
N = 1000; S = 0;

if strcmp(type,'even')
    for k = 2:2:N
        S = S + 1/k^2;            % çiftlerin toplamı
    end
elseif strcmp(type,'odd')
    for k = 1:2:N
        S = S + 1/k^2;            % teklerin toplamı
    end
else
    disp('type must be ''odd'' or ''even''');
end

S =
   1.2332
```

İçiçe `for` döngüleri

```
% double döngü örneği

N = 4; M = 3;

for i = 1:N
    for j = 1:M
        A(i,j) = i+j;
    end
end

A =
  2    3    4
  3    4    5
  4    5    6
  5    6    7
```

İçiçe `for` döngüleri

```
% kısmi vektörleştirilmiş
% satır bazlı versiyon

N = 4; M = 3; j = 1:M;

for i = 1:N
    A(i,:) = i+j;
end
```

```
% kısmi vektörleştirilmiş
% sütun bazlı versiyon

N = 4; M = 3; i = 1:N;

for j = 1:M
    A(:,j) = i+j;
end
```

```
% tamamen vektörleştirilmiş
% meshgrid kullanarak

N = 4; M = 3;
i = 1:N; j = 1:M;

[J,I] = meshgrid(j,i);
A = I+J;
```

`(i,j)` yerine `(j,i)` Neden?

while - döngüleri

```
while condition
    statements ...
end
```

döngüye devam koşulu

```
N = 1000; S = 0; k = 1;

while k<=N,
    S = S + 1/k^2;
    k = k+1;
end

S =
   1.6439

k =
   1001
```

Elle birkaç iterasyon

$$S = 0, k = 1$$

$$S = S + 1/k^2 = 0 + 1/1^2 = 1$$

$$k = k + 1 = 1 + 1 = 2$$

$$S = S + 1/k^2 = 1 + 1/2^2$$

$$k = k + 1 = 2 + 1 = 3$$

$$S = S + 1/k^2 = 1 + 1/2^2 + 1/3^2$$

$$k = k + 1 = 3 + 1 = 4$$

$$S = \sum_{k=1}^{N} \frac{1}{k^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{N^2}$$

sonsuz `while` - döngüleri

```
while 1
    statements ...
    if  condition
        break;
    end
    statements ...
end
```

Döngüden ayrılma koşulu

```
N = 1000; S = 0; k = 1;

while 1,
    S = S + 1/k^2;
    if k>N
        break;
    end
    k=k+1;
end

S =                k =
   1.6439             1000
```

Not: Klasik döngünün devam koşulu ile eşdeğer while döngüsünün ayrılma koşulu birbirlerinin mantıksal olarak tamamlayıcısıdırlar.

break
continue

break

Bir döngünün çalışmasını sonlandırır ve döngünün sonundan (end) sonra devam eder. Yalnızca içiçe bir döngünün sonlanmasından sonra devam eder.

continue

Mevcut geçişi bir döngüden geçirir, ancak sonraki geçişe devam eder.

Örnek: Seri hesaplamaları

$$\pi = 2\sqrt{3}\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)3^k} = 2\sqrt{3}\lim_{n\to\infty}\sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)3^k}$$

$$S_n = \sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)3^k} = \sum_{k=0}^{n-1}\frac{(-1)^k}{(2k+1)3^k} + \frac{(-1)^n}{(2n+1)3^n}$$

$$S_n = S_{n-1} + \frac{(-1)^n}{(2n+1)3^n},\ n \geq 1,\ S_0 = 1$$

$$S_n = S_{n-1} + \frac{(-1)^n}{(2n+1)3^n},\ n \geq 1,\ S_0 = 1$$

$$\boxed{T_n = \frac{(-1)^n}{(2n+1)3^n}}$$

$$\boxed{S_n = S_{n-1} + T_n,\ n \geq 1,\ S_0 = 1}$$

Tekrarlama bir `for` döngüsü veya `while` döngüsü ile gerçekleştirilebilir.

$$\boxed{\textbf{bağıl hata} = r = \frac{|S_n - S_{n-1}|}{|S_{n-1}|} = \frac{|T_n|}{|S_{n-1}|}}$$

Örnek: Vektörleştirilmiş Taylor serisi hesaplamaları

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \lim_{n \to \infty} \sum_{k=0}^{n} \frac{x^k}{k!}$$

$$S_n = \sum_{k=0}^{n} \frac{x^k}{k!} = \sum_{k=0}^{n-1} \frac{x^k}{k!} + \frac{x^n}{n!}$$

$$T_n = \frac{x^n}{n!} = \frac{x x^{n-1}}{n(n-1)!} = \frac{x}{n} T_{n-1}, \; n \geq 1$$

$$\boxed{\begin{array}{l} S_n = S_{n-1} + T_n, \; n \geq 1 \\ S_0 = 1, \; T_0 = 1 \end{array}}$$

```matlab
% versiyon 1 - for döngüsünün kullanımı

x = [1 3 0 -4 10]';          % sütun vektör
S = ones(size(x));           % x'in boyutunu alır
T = 1;
N=1000;                      % maksimum iterasyon
tol = 1e-12;                 % hata toleransı

for n = 1:N
    T = T.*x/n;              % n. terim
    if max(abs(T))<tol       % |T|<tol olduğunda bırak
        break;               % Neden max(abs(T))?
    end
    S = S+T;                 % toplamı güncelle
end
```

$S=1,\quad T=1$ **(başlangıç)**

Her zaman bazı iterasyonları elle kontrol edin

$n=1$

$T=T\cdot x/n=1\cdot x/1=x$

$S=S+T=1+x$

$n=2$

$T=T\cdot x/n=x\cdot x/2=x^2/2=x^2/2!$

$S=S+T=(1+x+x^2/2!)+x^3/3!=1+x+x^2/2!$

$n=3$

$T=T\cdot x/n=(x^2/2!)\cdot x/3=(x^2\cdot x)/(2\cdot 3)=x^3/3!$

$S=S+T=(1+x+x^2/2!)+x^3/3!=1+x+x^2/2!+x^3/3!$

```
fprintf('     x          exp(x)               S\n');
fprintf('----------------------------------------\n');
fprintf('% 7.2f    %12.6f    %12.6f\n', [x,exp(x),S]');
fprintf('----------------------------------------\n');
fprintf(['iterasyon n = ', int2str(n), '\n']);
```

```
     x          exp(x)               S
----------------------------------------
   1.00        2.718282        2.718282
   3.00       20.085537       20.085537
   0.00        1.000000        1.000000
  -4.00        0.018316        0.018316
  10.00    22026.465795    22026.465795
----------------------------------------
iterasyon n = 47
```

```
% norm(S-exp(x))    % 7.2760e-012 'e eşittir
```

```matlab
% versiyon 2 - while döngüsünün kullanımı

x = [1 3 0 -4 10]';          % sütun vektör

S = ones(size(x));           % x'in boyutunu alır
T = 1; n = 1;                % başlangıç
tol = 1e-12;                 % hata toleransı

while max(abs(T)) > tol
    T = T.*x/n;              % norm(T) > tol da kullanılabilir
    S = S+T;
    n = n+1;
end
```

Her zaman bazı iterasyonları elle kontrol edin

$S = 1, \quad T = 1, \quad n = 1,$ **(başlangıç)**

$T = T \cdot x / n = 1 \cdot x / 1 = x$

$S = S + T = 1 + x$

$n = 2$

$T = T \cdot x / n = x \cdot x / 2 = x^2 / 2 = x^2 / 2!$

$S = S + T = (1 + x + x^2 / 2!) + x^3 / 3! = 1 + x + x^2 / 2!$

$n = 3$

$T = T \cdot x / n = (x^2 / 2!) \cdot x / 3 = (x^2 \cdot x) / (2 \cdot 3) = x^3 / 3!$

$S = S + T = (1 + x + x^2 / 2!) + x^3 / 3! = 1 + x + x^2 / 2! + x^3 / 3!$

$n = 4$

```
fprintf('     x         exp(x)                  S\n');
fprintf('----------------------------------------\n');
fprintf('% 7.2f   %12.6f    %12.6f\n', [x,exp(x),S]');
fprintf('----------------------------------------\n');
fprintf(['iterasyon n = ', int2str(n-1), '\n']);
```

Neden n-1 ?

```
     x          exp(x)              S
----------------------------------------
   1.00       2.718282          2.718282
   3.00      20.085537         20.085537
   0.00       1.000000          1.000000
  -4.00       0.018316          0.018316
  10.00   22026.465795      22026.465795
----------------------------------------
iterasyon n = 47
```

```matlab
% versiyon 3 - sonsuz while döngüsünün kullanımı

x = [1 3 0 -4 10]';         % sütun vektör

S = ones(size(x));          % x'in boyutunu alır
T = 1; n = 1;               % başlangıç
tol = 1e-12;                % hata toleransı

while 1                     %sonsuz döngü
    T = T.*x/n;
    if max(abs(T)) < tol
       break;
    end
    S = S+T;
    n = n+1;
end
```

```
fprintf('     x            exp(x)                 S\n');
fprintf('----------------------------------------\n');
fprintf('% 7.2f   %12.6f    %12.6f\n', [x,exp(x),S]');
fprintf('----------------------------------------\n');
fprintf(['iterasyon n = ', int2str(n), '\n']);
```

```
     x            exp(x)                 S
----------------------------------------
   1.00        2.718282         2.718282
   3.00       20.085537        20.085537
   0.00        1.000000         1.000000
  -4.00        0.018316         0.018316
  10.00    22026.465795     22026.465795
----------------------------------------
iterasyon n = 47
```

Örnek: Karekök algoritması

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \quad n = 0,1,2,\ldots$$

$$x_n \to \sqrt{a}$$

```
a = 20;                          % sqrt(a) = 4.472135954999580
N = 10;
x(1) = 8;                        % keyfi başlangıç deðeri

for n=1:N-1
    x(n+1) = (x(n) + a/x(n))/2;
end
```

```
fprintf('  n                  x                 \n');
fprintf('------------------------------\n');
fprintf('%3.0f         %17.15f\n', [1:N; x]);
```

```
  n                   x
------------------------------
  1          8.000000000000000
  2          5.250000000000000
  3          4.529761904761905
  4          4.472502502972279
  5          4.472135970019965
  6          4.472135954999580
  7          4.472135954999580
  8          4.472135954999580
  9          4.472135954999580
 10          4.472135954999580
```

6 iterasyonda yakınsıyor

```matlab
a = 20; N = 10; x(1) = 8;                     % başlangýç
tol = 1e-12;                                  % bizim seçimimiz

fprintf('  n            x(n)            \n');
fprintf('------------------------------\n');

for n = 1:N-1
    fprintf('%2.0f        %17.15f\n', n, x(n));
    if abs(x(n)^2 - a) <= tol
        break;
    end
    x(n+1) = (x(n)+a/x(n))/2;
end
```

Hata toleransına (tol) yakınsadığı zaman döngüyü sonlandırır

n	x(n)
1	8.00000000000000
2	5.25000000000000
3	4.52976190476190
4	4.47250250297228
5	4.47213597001997
6	4.47213595499958

6 iterasyonda yakınsıyor

Bir sonraki slaytta while döngüsü kullanarak aynı sonuç elde edildi.

```
a = 20; n = 1; x = 8; tol = 1e-12;

fprintf('   n             x              \n');
fprintf('------------------------------\n');

while abs(x^2 - a) > tol
    x = (x + a/x)/2;
    n = n+1;
    E = abs(x^2-a);
    fprintf(' %1d        %17.15f       %8.2e\n', n, x, E);
end
```

klasik while döngüsü

```
   n              x                   hata
 ------------------------------------------
   2       5.250000000000000        7.56e+00
   3       4.529761904761905        5.19e-01
   4       4.472502502972279        3.28e-03
   5       4.472135970019965        1.34e-07
   6       4.472135954999580        3.55e-15
```

Son hata değeri $E = |x^2-a|$ tol değerinden daha küçük

```
a = 20; n = 1; x = 8; tol = 1e-12;
fprintf('  n            x                    hata\n');
fprintf('------------------------------------------\n');

while 1
    if abs(x^2 - a) <= tol break; end
    x = (x + a/x)/2;
    n = n+1;
    E = abs(x^2-a);
    fprintf(' %1d        %17.15f       %8.2e\n', n, x, E);
end
```

sonsuz while döngüsü

```
  n              x                    hata
 ------------------------------------------
  2        5.250000000000000       7.56e+00
  3        4.529761904761905       5.19e-01
  4        4.472502502972279       3.28e-03
  5        4.472135970019965       1.34e-07
  6        4.472135954999580       3.55e-15
```

Örnek: Çarpım hesaplamaları

$$\frac{\sin x}{x} = \prod_{k=1}^{\infty} \cos\left(\frac{x}{2^k}\right) = \cos\left(\frac{x}{2^1}\right)\cos\left(\frac{x}{2^2}\right)\cos\left(\frac{x}{2^3}\right)\ldots$$

$$\boxed{S_k = S_{k-1} \cdot \cos\left(\frac{x}{2^k}\right), \quad k = 1, 2, 3, \ldots, \quad S_0 = 1}$$

$$\textbf{bağıl hata} = r = \frac{\left|S_k - S_{k-1}\right|}{\left|S_{k-1}\right|}$$

```
x = [0.1, 0.2, 1, 4, 8]';                       % sütun vektör

S = ones(size(x));                              % x'in boyutunu alır
k = 1;
r = 1e-10;                                      % bağıl hata
while 1                                         % sonsuz döngü
    F = cos(x/2^k);                             % k. çarpan
    S1 = S.*F;
    if norm(S1-S) < r*norm(S)
        break;
    end
    S = S1;
    k = k+1;
end
```

S ile S1 arasındaki uzaklığı ölçmek için vektör normunu kullanır.

```
fprintf('   x                 sin(x)/x                 S\n');
fprintf('--------------------------------------\n');
fprintf('%6.2f    %12.6f    %12.6f\n', [x, sin(x)./x, S]');
fprintf('--------------------------------------\n');
fprintf(['iterasyon k = ', int2str(k), '\n']);
```

```
    x          sin(x)/x              S
--------------------------------------
  0.10         0.998334         0.998334
  0.20         0.993347         0.993347
  1.00         0.841471         0.841471
  4.00        -0.189201        -0.189201
  8.00         0.123670         0.123670
--------------------------------------
iterasyon k = 18
```

Örnek: Mevduat hesabı

$R =$ **Yıllık bileşik faiz**

$r = \dfrac{R}{1200} =$ **Aylık faiz**

$y_0 =$ **Başlangıç bakiyesi**

$x =$ **Aylık depozito**

Tekrarlı hesaplama:

$$y(1) = y_0$$

$$y(k) = (1+r)\,y(k-1) + x, \quad k \geq 2$$

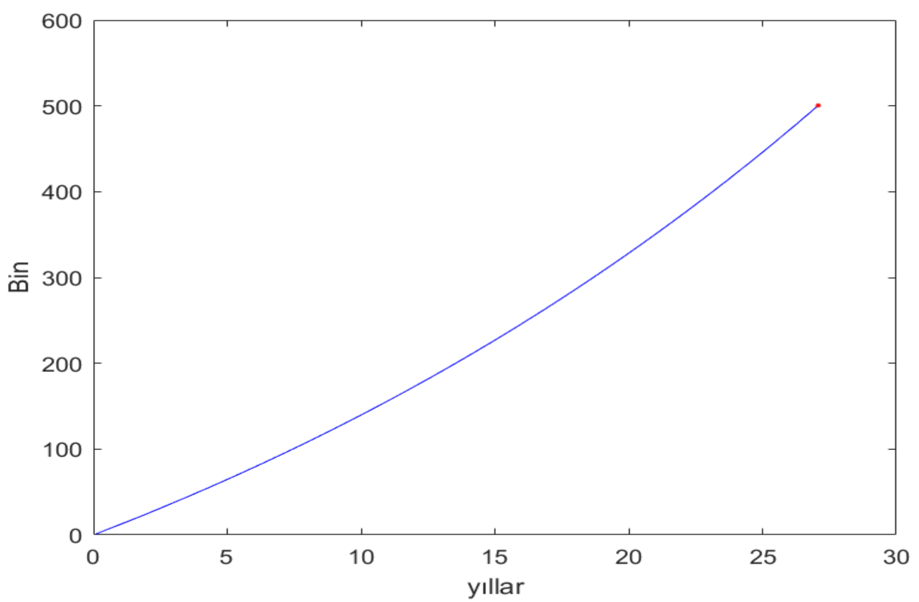
Problem: $y_k \geq y_{\max}$ **sağlayan minimum k'yi bulun.**

```
R = 3; r = R/1200; a = 1+r;
y0 = 1000; x = 1000;
ymax = 500000;
```

```
y(1) = y0; k = 1;

while y(k) < ymax
    k = k+1;
    y(k) = a*y(k-1) + x;
end
```

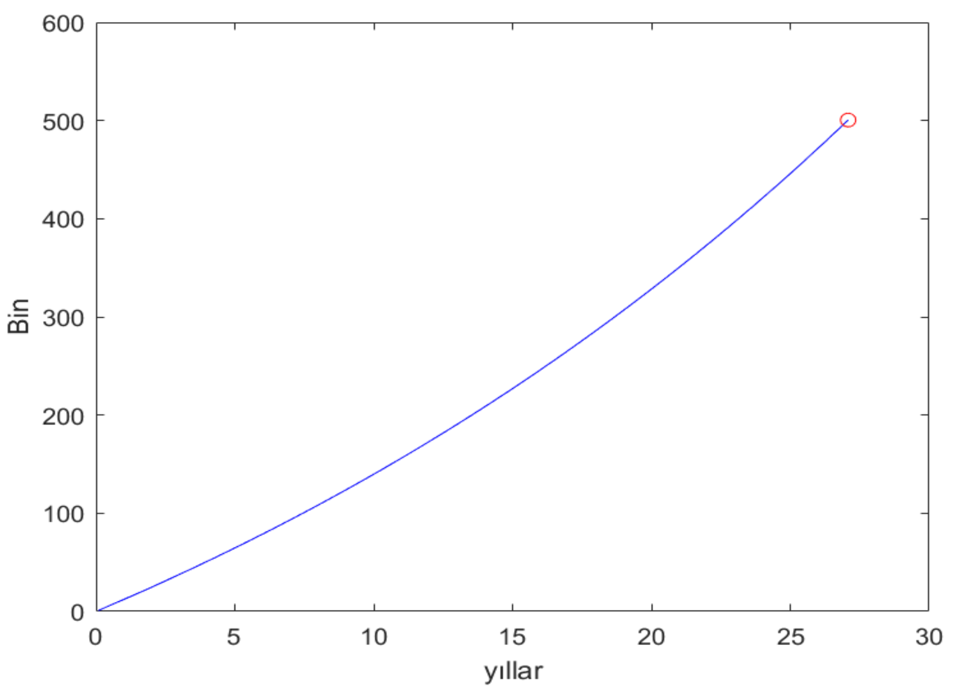
Klasik while döngüsü

```
% k = 325 = 27 yıl + 1 ay
% y(k) = $ 500500.40
% x*k = $ 325000

n = 1:k;
plot(n/12, y/1e3, 'b');
hold on
plot(k/12, y(k)/1e3, 'r.');
```



```
y(1) = y0; k = 1;

while 1
    k = k+1;
    y(k) = a*y(k-1) + x;
    if y(k) >= ymax
        break;
    end
end

% k = 325 = 27 yıl + 1 ay
% y(k) = $ 500500.40

n = 1:k;
plot(n/12, y/1e3, 'b');
hold on
plot(k/12, y(k)/1e3, 'ro');
```

Sonsuz while döngüsü



Örnek: Yakınsak sonsuz seriler ve çarpımlar

Toplamlar

Problem: Belirlenmiş doğrulukla sonsuz toplam hesabı

$$\sum_{k=1}^{\infty} T(k) = T(1) + T(2) + T(3) + \ldots = A = \textbf{limit değeri}$$

Tekrarlı hesaplama:

$$\boxed{\begin{aligned} &S_0 = 0 \\ &S_k = S_{k-1} + T(k), \quad k = 1, 2, 3, \ldots \end{aligned}}$$

Durdurma kriteri

1. $\left|S_k - S_{k-1}\right| < \varepsilon$ **(ardışık terimler yakın)**

2. $\left|S_k - A\right| < \varepsilon$ **(limite olan uzaklık küçük)**

3. $\left|S_k - S_{k-1}\right| < \varepsilon\left|S_k\right|$ **(hata yüzdesi)**

4. $\left|S_k - A\right| < \varepsilon\left|A\right|$ **(hata yüzdesi)**

ε **: kullanıcı tarafından seçilen hata toleransı, örneğin** $\varepsilon = 10^{-10}$

$$\sum_{k=1}^{\infty} k a^{k} = \frac{a}{(1-a)^{2}} = A, \ |a| < 1$$

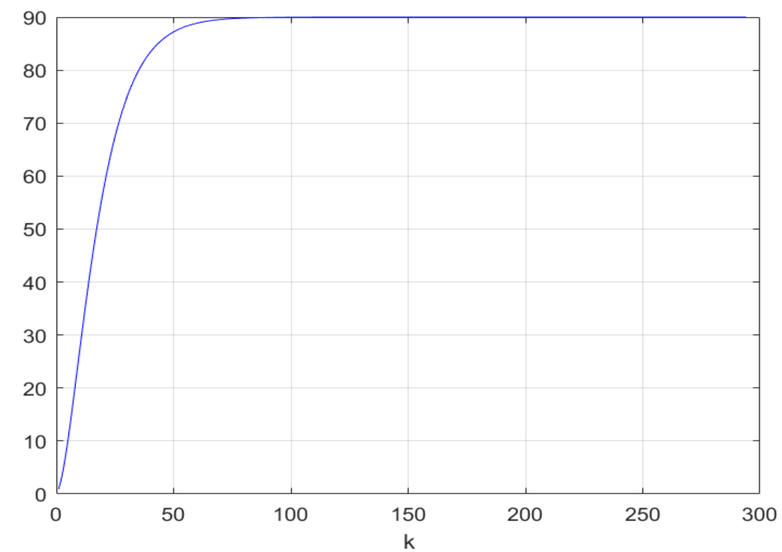
Geometrik seri eşitliği

$$\sum_{k=1}^{\infty} k (0.9)^{k} = \frac{0.9}{(1-0.9)^{2}} = 90 = A$$

```
a = 0.9; A = a/(1-a)^2;
tol = 1e-10;
S = 0; k = 1; T = k*a^k; S = S + T;     % başlangıç
while abs(S-A) > tol                    % limite olan uzaklık
    k = k+1;
    T = k*a^k;
    S = S + T;
end

% k = 294, |S-A| = 9.6350e-011
```

```
a = 0.9; A = a/(1-a)^2;
tol = 1e-10;

S = 0; k = 1; S = S + k*a^k;

while 1
    if abs(S-A) <= tol
        break;
    end
    k = k+1;
    S = S + k*a^k;
end

% k = 294
% |S-A| = 9.6350e-011

n = 1:k;
y = cumsum(n.*a.^n);
plot(n,y,'b');
```

Sonsuz while döngüsü



```
a = 0.9; A = a/(1-a)^2;
tol = 1e-10;

S = 0; k = 1; T = k*a^k; S = S + T;    % başlangıç

while abs(T) > tol                     % ardışık terimler
    k = k+1;
    T = k*a^k;
    S = S + T;
end

k, abs(T)

k =
   272
ans =
   9.7394e-011
```

Klasik while döngüsü

```
a = 0.9; A = a/(1-a)^2;
tol = 1e-10;

S = 0; k = 1; T = k*a^k; S = S + T;    % başlangıç

while 1
    if abs(T) <= tol
        break;
    end
    k = k+1;
    T = k*a^k;
    S = S + T;
end

k, abs(T)
k =
   272
ans =
   9.7394e-011
```

Sonsuz while döngüsü

Çarpımlar

$$\frac{\pi}{2} = \prod_{k=1}^{\infty} \left(\frac{4k^2}{4k^2 - 1} \right)$$

Wallis (1655)

```
A = pi/2; tol = 1e-5;

S = 1; k = 1; T = 4*k^2/(4*k.^2 - 1); S = S*T;

while abs(S-A) > tol
    k = k+1;
    T = 4*k.^2/(4*k^2 - 1);
    S = S*T;
end

k, abs(S-A)  % k = 39270,  ans = 9.9998e-006
```