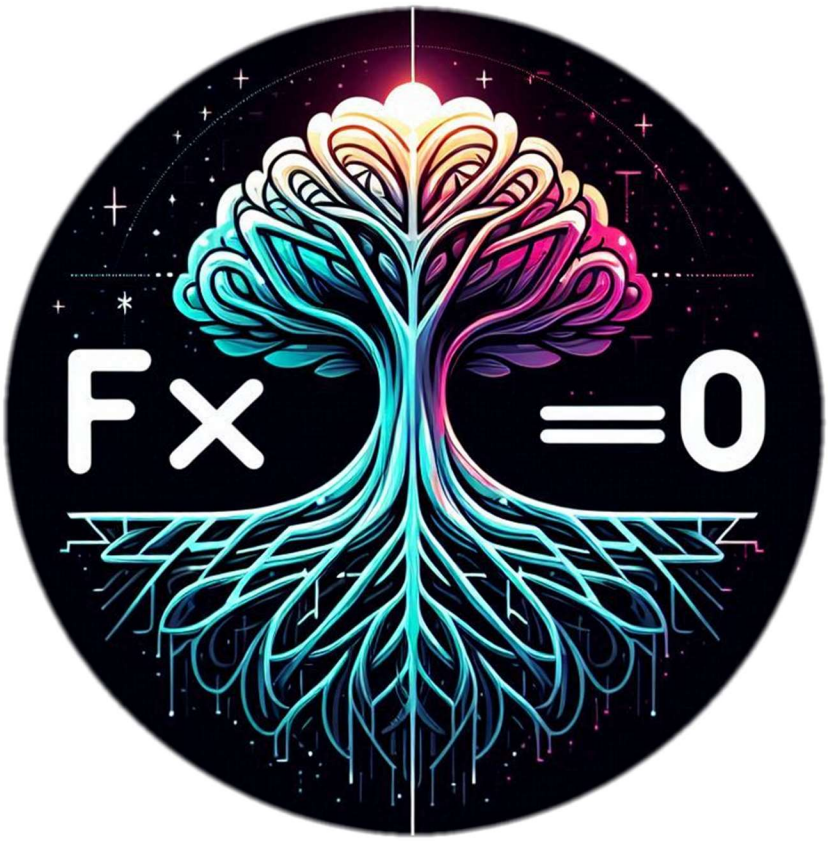

برنامه ریشه‌یابی

آنائیس گل‌بوداغیانس

دانشجوی مهندسی برق دانشگاه خواجه نصیرالدین طوسی

کد دانشجویی: ۴۰۱۲۲۱۱۳

استاد: دکتر ذاکری

به نام خدا

این برنامه درباره‌ی ریشه‌یابی است. هدف این بوده که کاربر حق انتخاب میان پنج روش ریشه‌یابی داشته باشد:

1. دوبخشی یا تنصیف
2. نابه‌جایی یا خط قاطع
3. تکرار ساده یا نقطه ثابت
4. نیوتون-رافسون
5. وتری

کاربر می‌تواند معیار توقف را نیز انتخاب کند. این معیارها عبارت‌اند از:

1. تعداد تکرار
2. $|f(x_n)| < \varepsilon$
3. $|x_n - x_{n-1}| < \varepsilon$
4. ترکیب معیار اول و دوم؛ یعنی علاوه‌بر این‌که شرط دوم را اقناع کرد، تا تکرار nام ریشه را پیدا کند.
5. ترکیب معیار اول و سوم.

علاوه‌بر این، برنامه قادر است سریع‌ترین روش ریشه‌یابی را تشخیص دهد و با آن ریشه‌یابی انجام دهد.

۸ عدد فایل متلب تهیه شده که app.m کد اصلی و سایر آن‌ها تابع‌هایی‌ست که در کد اصلی استفاده شده‌اند.

توضیح کد:

```
clc;clear;close all
```

در خط اول ابتدا این سه دستور را می‌نویسیم تا command window پاک شود؛ تمامی متغیرها از workspace پاک شوند و تمام اشکال و figureها بسته شوند.

```
image_path = 'Designer.png';
Im = imread(image_path);

p = figure('Position',[100 100 1300 600]);

imshow(Im);
title('Find The Root', 'FontSize', 14, 'Color', 'black');
set(p, 'Name', 'Find The Root');
pause(2.5);
close all;
```

در اینجا عکسی که ابتدای گزارشکار بود فراخوانده می‌شود، موقعیت پنجره‌ای که در آن عکس را می‌بینیم تنظیم می‌شود و عنوانی بالای عکس و روی کادر پنجره قرار داده می‌شود. سپس برنامه به مدت 2.5 ثانیه مکث می‌کند و سپس پنجره بسته می‌شود. (فیلمی از اجرای کد، ضمیمه‌ی این فایل‌ها شده است.)

```
disp('By Anaies Golboudaghians');
pause(1);
disp('EE student at KNTU');
pause(1);
disp('Supervisor: Dr Zakeri');
pause(2);
clc;
```

در این خطوط، نام من، دانشگاه، رشته و استاد در command window نمایش داده می‌شود و در نهایت پیش از شروع برنامه ریشه‌یابی، همه‌ی این‌ها از command window با استفاده از دستور clc پاک می‌شوند.

```
syms x;
f(x) = input('Enter a function e.g 3.*x - exp(-x):');
n = 1;
```

در این خطوط تابع از کاربر دریافت می‌شود و تعداد تکرار در ابتدا یک فرض می‌شود. به n مقدار می‌دهیم تا بعداً موقع تغییر مقدار آن، خطایی در اجرای برنامه پیش نیاید.

```
disp('1.Bisection');
disp('2.False Position');
disp('3.Fixed Point');
disp('4.Newton');
disp('5.Secant');
disp('6.The fastest method');
method = input('What method do you want to use? Type the
number:');
disp('1.n times repeat');
disp('2. x(n+1) - x(n)<e');
disp('3. f(x)<e');
disp('4. 1&2');
disp('5. 1&3');
stop = input('What is your stopping criteria?');
```

در این بخش انواع روش‌های ریشه‌یابی و معیار توقف‌ها نمایش داده می‌شود و کاربر می‌تواند روش یا معیار مورد نظرش را وارد کند. مورد ۶ سریع‌ترین روش را تشخیص می‌دهد.

```
if stop == 1
    N = input('Enter the n:');
    N = N + 1;
    E =0;
```

```
elseif (stop ==2) || (stop==3)
    E = input('Enter the e - the tolerance:');
    N =0;
elseif (stop ==4)||(stop==5)
    N = input('Enter the n:');
    N = N + 1;
    E = input('Enter the e - the tolerance:');
end
```

اینجا، بسته به انتخاب کاربر، مقدار تکرار و اپسیلون از کاربر دریافت می‌شود. مقدار اپسیلون برابر با متغیر E است.

```
acc = input('How many decimal places do you want to be
accurate?');
```

اکنون تعداد رقم اعشاری که کاربر می‌خواهد ریشه را بیابد از کاربر گرفته می‌شود.

در ادامه به یک سری دستورات شرطی می‌پردازیم که به انتخاب روش کاربر بستگی دارد.

```
if method == 1
    disp('Do you want me to guess the range (a,b)');
    gss = input('1.Yes (Any key).No:');
    if gss==1
        [a, b] = rfinder(f);
    else
        a = input('Enter the a: ');
        b = input('Enter the b: ');
        while (f(a)*f(b)) > 0
            disp('There is no root in that range');
             a = input('Enter the a: ');
             b = input('Enter the b: ');
        end
    end
   [n, x, f] = bisection(a, b, f, stop, N, E);
```

اگر کاربر روش اول، یعنی دو بخشی را انتخاب کند، از کاربر پرسیده می‌شود که «آیا می‌خواهی من بازه‌ای که در آن ریشه هست حدس بزنم؟» اگر جواب کاربر مثبت باشد عدد یک را وارد می‌کند؛ در غیر این صورت هر کلیدی به‌جز ۱ فشار دهد جواب منفی محسوب می‌شود. اگر خود برنامه حدس بزند، تابع rfinder را فراخوانی می‌کند که کارش پیدا کردن بازه‌ای است که ریشه در آن وجود دارد.

در غیر این صورت، کاربر می‌تواند خودش بازه را وارد کند و اگر اشتباهی کرد، برنامه می‌گوید: «در بازه‌ای که دادی ریشه وجود ندارد» و مجدداً بازه از کاربر گرفته می‌شود.

در پایان تابع bisection فراخوانی می‌شود و ریشه، تعداد تکرار، مقدار نهایی تابع با ریشه پیدا شده را به کاربر می‌دهد. (پس از پایان کد اصلی، تابع‌ها تشریح خواهند شد.)

```
elseif method == 2
    disp('Do you want me to guess the range (a,b)');
    gss = input('1.Yes (Any key).No:');
    if gss==1
        [a, b] = rfinder(f);
    else
        a = input('Enter the a: ');
        b = input('Enter the b: ');
        while (f(a)*f(b)) > 0
            disp('There is no root in that range');
             a = input('Enter the a: ');
             b = input('Enter the b: ');
        end
    end
   [n, x, f] = false_pos(a, b, f, stop, N, E);
```

اکنون به روش نابه‌جایی رسیدیم. مانند روش قبلی، همه‌چیز اجرا می‌شود.

```
elseif method ==3
    g(x) = input('Enter your g(x):');
    disp('Do you want me to guess the range (a,b)');
    gss = input('1.Yes (Any key).No:');
    if gss==1
        [a, b] = rfinder(f);
    else
        a = input('Enter the a: ');
        b = input('Enter the b: ');
        while (f(a)*f(b)) > 0
            disp('There is no root in that range');
             a = input('Enter the a: ');
             b = input('Enter the b: ');
        end
    end
    valid = gvalid(g, a, b);

    while valid<=0
        if valid == -3
            disp('g(x) is invalid in both conditions');
            g(x) = input('enter g(x):');
            valid = gvalid(g, a, b);
        elseif valid == 0
```

```
            disp("|g'(x)|>1");
            g(x) = input('enter g(x):');
            valid = gvalid(g, a, b);
        elseif valid == -1
            disp("range of g(x) is not in (a,b)");
            g(x) = input('enter g(x):');
            valid = gvalid(g, a, b);
        end
    end
    if valid == 2
        disp('g(x) is valid');
    end

    x0 = input('Enter a the point of start:');
    [n, x, fval] = fixed_point(f, g, x0, stop, N, E);
```

در روش نقطه ثابت یا تکرار ساده، ابتدا تابع g(x)؛ سپس بازه‌ای که ایکس در آن وجود دارد حدس زده یا از کاربر گرفته می‌شود. صحت بازه ابتدا چک می‌شود و بعد، نوبت صحت تابع g(x) است. برد تابع باید در بازه صدق کند و مشتق تابع نباید از مقدار یک به بالا تجاوز کند. تمام این کارها را تابع gvalid انجام می‌دهد. این تابع، g و بازه را دریافت می‌کند و مقداری به عنوان خروجی می‌دهد. اگر این مقدار دو باشد، صحت تابع تایید می‌شود. اگر مقدار ۳- را دهد یعنی تابع در دو شرط صدق نمی‌کند. صفر به معنای بزرگ بودن مشتق تابع از یک است و ۱- یعنی در برد a و b صادق نیست.

پس از سنجش صحت، نقطه شروع تکرار از کاربر گرفته می‌شود و تابع نقطه ثابت فراخوانی می‌شود.

```
elseif method ==4
    x0 = input('Enter a the point of start:');
    [n, x, fval] = newton(f, x0, stop, N, E);
```

اگر روش نیوتون انتخاب شده باشد، باز نقطه شروع گرفته شده و ریشه یابی انجام می‌شود.

```
elseif method ==5
    x0 = input('Enter the point of start:');
    x1 = input('Enter second point of start:');
    [n, x, fval] = secant(f, x0, x1, stop, N, E);
```

در روش وتری دو نقطه شروع از کاربر دریافت می‌شود.

```
elseif method ==6
    disp('Do you want me to guess the range (a,b)');
    gss = input('1.Yes (Any key).No:');
    if gss==1
        [a, b] = rfinder(f);
    else
```

```
    a = input('Enter the a: ');
    b = input('Enter the b: ');
    while (f(a)*f(b)) > 0
        disp('There is no root in that range');
         a = input('Enter the a: ');
         b = input('Enter the b: ');
    end
end
[n1, x_1, f] = bisection(a, b, f, stop, N, E);
[n2, x_2, f] = false_pos(a, b, f, stop, N, E);
g(x) = input('Enter your g(x):');
valid = gvalid(g, a, b);

while valid<=0
    if valid == -3
        disp('g(x) is invalid in both conditions');
        g(x) = input('enter g(x):');
        valid = gvalid(g, a, b);
    elseif valid == 0
        disp("|g'(x)|>1");
        g(x) = input('enter g(x):');
        valid = gvalid(g, a, b);
    elseif valid == -1
        disp("range of g(x) is not in (a,b)");
        g(x) = input('enter g(x):');
        valid = gvalid(g, a, b);
    end
end
if valid == 2
    disp('g(x) is valid');
end

x0 = input('Enter a the point of start:');
[n3, x_3, fval] = fixed_point(f, g, x0, stop, N, E);
[n4, x_4, fval] = newton(f, x0, stop, N, E);
x1 = input('Enter second point of start:');
[n5, x_5, fval] = secant(f, x0, x1, stop, N, E);
ns = [n1, n2, n3, n4, n5];
n = min(ns);
    if n == n1
        disp('Method no.1_bisection is the fastest');
        x = x_1;
    end
    if n == n2
```

```
            disp('Method no.2_false position is the
fastest');
            x = x_2;
        end
        if n == n3
            disp('Method no.3_fixed point is the fastest');
            x = x_3;
        end
        if n == n4
            disp('Method no.4_newton is the fastest');
            x = x_4;
        end
        if n == n5
            disp('Method no.5_secant is the fastest');
            x = x_5;
        end
end
```

در این بخش تمامی روش‌ها اجرا می‌شوند و مقدار تکرار آن‌ها با توجه به شماره روش، به n1، n2 ... n5 نامیده می‌شوند و در آرایه ns قرار می‌گیرند. با دستور min، کوچیکترین مقدار آرایه پیدا می‌شود و n نام‌گذاری می‌شود. در ادامه با دستورات شرطی، بررسی می‌شود که n برابر کدام عضوی از آرایه ns است. اگر برابر باشد، برای کاربر چاپ می‌شود که این روش، سریع‌ترین است. از دستور elseif استفاده نشده تا اگر دوتا روش سریع‌ترین شدند، هردو نشان داده شوند.

```
disp(n-1);
```

در این خط تعداد تکرار نمایش داده می‌شود. n منهای یک می‌شود زیرا در کامپیوتر تکرار صفرم تکرار اول قرار داده شده است و در متلب آرایه‌ی صفرم نداریم.

```
acc = round(abs(acc));
formatSpecx = ['%.', num2str(acc), 'f'];
fprintf(formatSpecx, x(n));
disp('    ');
y = matlabFunction(f);
formatSpecy = ['%.', num2str(acc+1), 'f'];
fprintf(formatSpecy,y(x(n)));
```

در این خطوط، مقدار اعداد اعشار به فرم صحیح غیر منفی درمی‌آیند. از num2str برای تبدیل عدد به رشته استفاده شده است تا میزان رقم اعشار دلخواه کاربر جای داده شود و چاپ شود. مقدار تابع در آن ریشه یک رقم اعشار بیشتری دارد.

از matlabFunction استفاده شده تا فرم تابع از syms یا symbolic به فرم handle function دربیاید تا با قرار دادن هر مقداری داخل آن، مقدار تابع را محاسبه کند. اگر از این دستور استفاده نمی‌شد، خروجی تابع مقدار نبود بلکه ایکسی بود که مقدار تابع قرار داده شده بود. یعنی:

مثال: x=1، y=3*(1) – exp(-1)

مقدار y محاسبه نمی‌شد!

```
table = zeros(n, 3);
for i=[1:n]
    table(i,1) = i-1;
    table(i,2) = x(i);
    table(i,3) = y(x(i));
end

disp('   ');
disp(table);
```

در پایان همه‌ی داده‌ها در جدول یا ماتریسی مرتب می‌شود و به کاربر نمایش داده می‌شود.

توضیح توابع:

```
function [n, x, f] = bisection(a, b, f, stop, N, E)
    n=1;
    x(n) = (a+b)./2;

    if stop == 1
        while n<N
            if f(a)*f(x(n))<0
                b = x(n);
            else
                a = x(n);
            end
            n = n+1;
            x(n)= (a+b)./2;

        end
    elseif stop == 2
        if f(a)*f(x(n))<0
            b = x(n);
        else
```

```
        a = x(n);
    end
    n =2;
    x(2) = (a+b)./2;
    while abs(x(n)-x(n-1))>=E
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n)= (a+b)./2;

    end

elseif stop == 3
    while abs(f(x(n)))>=E
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n)= (a+b)./2;

    end
elseif stop == 4
    if f(a)*f(x(n))<0
        b = x(n);
    else
        a = x(n);
    end
    n =2;
    x(2) = (a+b)./2;
    while (abs(x(n)-x(n-1))>=E)
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n)= (a+b)./2;
    end
    for i = [n:N]
```

```
            if f(a)*f(x(i))<0
                b = x(i);
            else
                a = x(i);
            end
            x(i)= (a+b)./2;
        end
        n = N;
    elseif stop == 5
        while (abs(f(x(n)))>=E)
            if f(a)*f(x(n))<0
                b = x(n);
            else
                a = x(n);
            end
            n=n+1;
            x(n) = (a+b)./2;
        end
        for i = [n:N]
            if f(a)*f(x(i))<0
                b = x(i);
            else
                a = x(i);
            end
            x(i)= (a+b)./2;
        end
        n = N;
    end

end
```

کد بالا تابع ریشه‌یابی دو بخشی است. با توجه به انتخاب کاربر، معیار توقف در این تابع بررسی می‌شود و ایکس و مقدار تابع را می‌دهد.

ایکس ابتدا میانگین بازه می‌باشد و اگر با توجه به قضیه مقدار میانی، میان ایکس جدید و a ریشه‌ای نباشد (مقدار ضرب تابع آن‌ها منفی شود)، بازه عوض می‌شود و ایکس جدید ساخته می‌شود و این حلقه تا برقرار شدن معیارهای توقف ادامه می‌یابد.

از abs برای اجرای قدر مطلق استفاده شده است.

```
function [n, x, f] = false_pos(a, b, f, stop, N, E)
    n=1;
```

```
x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));


if stop == 1
    while n<N
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n = n+1;
        x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));

    end
elseif stop == 2
    if f(a)*f(x(n))<0
        b = x(n);
    else
        a = x(n);
    end
    n =2;
    x(2) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    while abs(x(n)-x(n-1))>=E
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));

    end

elseif stop == 3
    while abs(f(x(n)))>=E
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));

    end
```

```
elseif stop == 4
    if f(a)*f(x(n))<0
        b = x(n);
    else
        a = x(n);
    end
    n =2;
    x(2) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    while (abs(x(n)-x(n-1))>=E)
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    end
    for i = [n:N]
        if f(a)*f(x(i))<0
            b = x(i);
        else
            a = x(i);
        end
        x(i) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    end
    n = N;
elseif stop == 5
    while (abs(f(x(n)))>=E)
        if f(a)*f(x(n))<0
            b = x(n);
        else
            a = x(n);
        end
        n=n+1;
        x(n) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    end
    for i = [n:N]
        if f(a)*f(x(i))<0
            b = x(i);
        else
            a = x(i);
        end
        x(i) = (a.*f(b) - b.*f(a))./(f(b)-f(a));
    end
```

```
        n = N;
    end
end
```

کد بالا تابع ریشه‌یابی روش نابه‌جایی است.

```
function [n, x, fval] = fixed_point(f, g, x0, stop, N, E)

    n = 1;
    x(1) = x0;
    f_num = matlabFunction(f);
    g_num = matlabFunction(g);


    if stop ==1
        while n<N
            x(n+1) = g_num(x(n));
            n=n+1;
        end
    elseif stop ==2
        x(n+1) = g_num(x(n));
        n=n+1;
        while abs(x(n)-x(n-1))>=E
            x(n+1) = g_num(x(n));
            n=n+1;
        end
    elseif stop ==3
        while abs(f_num(x(n))) >= E
            x(n+1) = g_num(x(n));
            n=n+1;
        end
    elseif stop ==4
        x(n+1) = g_num(x(n));
        n=n+1;
        while abs(x(n)-x(n-1))>=E
            x(n+1) = g_num(x(n));
            n=n+1;
        end
        for i = [n:N]
            x(i+1) = g_num(x(i));
        end
        n = N;
    elseif stop==5
        while abs(f_num(x(n))) >= E
            x(n+1) = g_num(x(n));
```

```
            n=n+1;
        end
        for i = [n:N]
            x(i+1) = g_num(x(i));
        end
        n = N;
    end
    fval = f_num(x(n));
end
```

کد بالا تابع روش تکرار ساده است. ایکس صفر را برابر درایه اول آرایه ایکس قرار می‌دهید. ایکس‌های جدید برابر مقدار تابع جی ایکس قبلی است. این حلقه‌ها تا زمانی ادامه پیدا می‌کنند تا به میزان معیار توقف برسد.

```
function valid = gvalid(g, a, b)
    g_num = matlabFunction(g);
    dg = diff(g);
    dg_num = matlabFunction(dg);

    r1 = 0;
    for i=[1:1:10000000]
        r1 = -500;
        r2 = 200.*i;
        x = [r1:0.0001:r2];
        max_d(i) = abs(max(dg_num(x)));
        min_d(i) = abs(min(dg_num(x)));
        r1 = r2;
        if (max_d(i)>1)|| (min_d(i)>1)
            valid = -1;
            break
        else
            valid = 1;
            disp("g(x) is considered valid in |g'(x)|<1");
            apr = input('Do you want to check more? 1.Yes,
(any key).No');
            if apr == 1
                continue
            else
                break
            end
        end
    end
```

```
    x = [a:0.0001:b];
    max_s = max(g_num(x));
    min_s = min(g_num(x));
    if max_s>b || min_s<a
        valid = valid - 2;
    else
        valid = valid + 1;
    end

end
```

تابع بالا صحت تابع جی ایکس را می‌سنجد. ابتدا مشتق تابع g تشکیل می‌شود. حلقه‌ای نوشته شده تا قابلیت انبساط داشته باشد، به این معناست که اگر کاربر بخواهد ببیند تابع در بازه بزرگتری، بردش بیشتر از یک می‌شود، بازه تکرار حلقه انبساط پیدا می‌کند. این کار به سرعت اجرای نرم‌افزار کمک می‌کند.

در گام بعدی برد بازه چک می‌شود. به گونه‌ای اگر ماکسیمم یا مینیمم تابع بیشتر از بازه تعیین شده شود، صحت جی ایکس باطل می‌شود.

```
function [n, x, fval] = newton(f, x0, stop, N, E)

    n = 1;
    x(1) = x0;

    df = diff(f);
    f_num = matlabFunction(f);
    df_num = matlabFunction(df);
    if stop ==1
        while n<N
            x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
            n = n + 1;
        end
    elseif stop ==2
        x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
        n = n + 1;
        while abs(x(n)-x(n-1))>=E
            x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
            n = n + 1;
        end
    elseif stop ==3
        while abs(f_num(x(n))) >= E
            x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
            n = n + 1;
        end
```

```
    elseif stop ==4
        x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
        n = n + 1;
        while abs(x(n)-x(n-1))>=E
            x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
            n = n + 1;
        end
        for i = [n:N]
            x(i+1) = x(i) - f_num(x(i)) / df_num(x(i));
        end
        n = N;
    elseif stop==5
        while abs(f_num(x(n))) >= E
            x(n+1) = x(n) - f_num(x(n)) / df_num(x(n));
            n = n + 1;
        end
        for i = [n:N]
            x(i+1) = x(i) - f_num(x(i)) / df_num(x(i));
        end
        n = N;
    end
    fval = f_num(x(n));
end
```

کد بالا تابع ریشه‌یابی به روش نیوتون است.

```
function [n, x, fval] = secant(f, x0, x1, stop, N, E)

    n = 2;
    x(1) = x0;
    x(2) = x1;

    f_num = matlabFunction(f);

    if stop ==1
        while n<N
            x(n+1) = (x(n-1).*f_num(x(n)) -
x(n).*f_num(x(n-1))).\(f_num(x(n))-f_num(x(n-1)));
            n = n + 1;
        end
    elseif stop ==2
        x(n+1) = (x(n-1).*f_num(x(n)) - x(n).*f_num(x(n-
1))).\(f_num(x(n))-f_num(x(n-1)));
        n = n + 1;
        while abs(x(n)-x(n-1))>=E
```

```
            x(n+1) = (x(n-1).*f_num(x(n)) -
x(n).*f_num(x(n-1))./(f_num(x(n))-f_num(x(n-1)));
            n = n + 1;
        end
    elseif stop ==3
        while abs(f_num(x(n))) >= E
            x(n+1) = (x(n-1).*f_num(x(n)) -
x(n).*f_num(x(n-1))./(f_num(x(n))-f_num(x(n-1)));
            n = n + 1;
        end
    elseif stop ==4
        x(n+1) = (x(n-1).*f_num(x(n)) - x(n).*f_num(x(n-
1)))./(f_num(x(n))-f_num(x(n-1)));
        n = n + 1;
        while abs(x(n)-x(n-1))>=E
            x(n+1) = (x(n-1).*f_num(x(n)) -
x(n).*f_num(x(n-1))./(f_num(x(n))-f_num(x(n-1)));
            n = n + 1;
        end
        for i = [n:N]
            x(i+1) = (x(i-1).*f_num(x(i)) -
x(i).*f_num(x(i-1)))./(f_num(x(i))-f_num(x(i-1)));
        end
        n = N;
    elseif stop==5
        while abs(f_num(x(n))) >= E
            x(n+1) = (x(n-1).*f_num(x(n)) -
x(n).*f_num(x(n-1))./(f_num(x(n))-f_num(x(n-1)));
            n = n + 1;
        end
        for i = [n:N]
            x(i+1) = (x(i-1).*f_num(x(i)) -
x(i).*f_num(x(i-1)))./(f_num(x(i))-f_num(x(i-1)));
        end
        n = N;
    end
    fval = f_num(x(n));
end
```

کد بالا تابع روش وتری است.

```
function [a, b] = rfinder(f)
    for j=[1:1:100000]
        for i=[-10*j:1:10*j]
            if f(i)*f(i+1)<=0
                a = i;
                b = i+1;
                break
            end
        end
        if isempty(a)
            continue
        else
            break
        end
    end
end
```

کد بالا، تابع پیدا کردن بازه‌ای است که در آن ریشه وجود دارد. از قضیه مقدار میانی استفاده شده است و بازه تکرار حلقه در صورتی که بازه‌ی دارای ریشه را پیدا نکند، انبساط پیدا می‌کند تا آخر بازه مورد نظر را پیدا کند.

نمونه‌ای از خروجی نرم‌افزار (نمونه‌های بیشتر به‌صورت ویدیو ضمیمه فایل‌ها شده است.):

Enter a function e.g 3.*x - exp(-x):x-cos(x)

1.Bisection

2.False Position

3.Fixed Point

4.Newton

5.Secant

6.The fastest method

What method do you want to use? Type the number:6

1.n times repeat

2. x(n+1) - x(n)<e

3. f(x)<e

4. 1&2

5. 1&3

What is your stopping criteria?3

Enter the e - the tolerance:0.0001

How many decimal places do you want to be accurate?6

Do you want me to guess the range (a,b)

1.Yes (Any key).No:1

Enter your g(x):2*x

g(x) is invalid in both conditions

enter g(x):cos(x)

g(x) is considered valid in |g'(x)|<1

Do you want to check more? 1.Yes, (any key).No5

g(x) is valid

Enter a the point of start:1

Enter second point of start:1.5

Method no.4_newton is the fastest

2

0.739113

0.0000465

0	1.0000	0.4597
1.0000	0.7504	0.0189
2.0000	0.7391	0.0000