351 Project: Muller's Method

Composed by Jenny Lee and Athalia Santoso

1. Background of Muller's Method

Muller's Method is a general version of the Secant method,but Muller's does not necessitate the derivative of functions. It is iterating method that takes three starting points $(p_0, f(p_0))$, $(p_1, f(p_1))$, $(p_2, f(p_2))$. With quadratic formula, we find the next approximation on root that passes through the three points in a parabola-finding the intersection of parabola with the x-axis(x_3). It has advantages that Muller's method converges faster than the secant method, and as fast as Newton's method. Also, the method can be used to find real or complex zeros of a function and can compute complex arithmetic that it can find imaginery roots. However, Muller's method is recursive method that it's difficult to compute by hand, and extraeous roots can be deduced.

2. Muller's Method.

```
Muller[f0_, p0_, p1_, p2_, max_] :=
  Module[{a, b, c, det, count, err},
   x0 = p0; y0 = N[f[p0]];
   x1 = p1; y1 = N[f[p1]];
   x2 = p2; y2 = N[f[p2]];
   count = 2;
   Print["k= 0", "\t P0= ", N[x0], ",\t f[p0]  = ", y0];
   Print["k= 1", "\t P1= ", N[x1], ",\t f[p1]  = ", y1];
   Print["k= 2", "\t P2= ", N[x2], ",\t f[p02]  = ", y2];
   While[count < max,
    (*If[err<10^-8,Break[]]; *)
    h0 = x0 - x2; h1 = x1 - x2; c = y2; d0 = y0 - c; d1 = y1 - c;
    det = h0 * h1 (h0 - h1);
    If[c == 0, Break[]];
    (*If[det==0,Break[]];*)
    a = ((d0 * h1) - (h0 * d1))/det; b = ((d1 * h0 * h0) - (d0 * h1 * h1))/det;
    If[b^2 > 4 a * c, res = Sqrt[b^2 - 4 a * c], res = 0];
    If[b < 0, res = -res];
    err = -(2 c)/(b + res);
    x3 = x2 + err;
```

```
  If[Abs[x3 - x1] < Abs[x3 - x0],
   s = x1; x1 = x0; x0 = s; t = y1; y1 = y0; y0 = t;];
  If[Abs[x3 - x2] < Abs[x3 - x1],
   s = x2; x2 = x1; x1 = s; t = y2; y2 = y1; y1 = t;];
  (*For next recursion,*)
  x2 = x3;
  y2 = f[x2];
  Print["k= ", count + 1, "\t p"count+1, " = ", N[x2],
   ",\t f[", "p"count+1, "]= ", y2, "\t |dp|<= ", N[Abs[err]]];
  (*Print[err];*)
  count = count + 1;
 ];

 (*Result Print:*)
 Print[""]
  Print["\t \t RESULTS:"];
 Print["The function is f[x] = ", f[x]];
 Print["The root: "];
 Print["   p = ", PaddedForm[x2, {16, 16}]];
 Print[" f[p] = ", N[y2]];
 root = FindRoot[f[x] == 0, {x, 1.5}][[1, 2]];
 Print["Root computed by built-in: ", PaddedForm[root, {16, 16}]];
 Print["error: ", Abs[root - x2]];
];
```

3. Comparison with Secant and Newton

Secant Method

```
Secant[f0_, x0_, x1_, {delta_, max_}] :=
Module[{k, small, cond},
 p0 = x0; Y0 = N[f[p0]]; p1 = x1; Y1 = N[f[p1]];
 k = 0;  cond = 0;
 Print["p = ", N[p0]];
 Print["p = ", N[p1]];
 While[cond == 0 && k < max,
   Df = (Y1 - Y0) / (p1 - p0);
  If[Df == 0, cond = 1; Dp = p1 - p0; p2 = p1, Dp = Y1/Df; p2 = p1 - Dp];
   Y2 = N[f[p2]];
   Print["p = ", N[p2], "\t  |dp| <= ", Abs[N[p1 - p0]]];
  If[N[Abs[p1 - p0]] < delta, cond = 1];
   p0 = p1; Y0 = Y1; p1 = p2; Y1 = Y2;
   k = k + 1; ];
    root = FindRoot[f[x] == 0, {x, 1.5}][[1, 2]];
    Print["Root computed by built-in: ", PaddedForm[root, {16, 16}]];
    Print["error: ", Abs[root - p2]];];
```

Newton's Method

```
NewtonIte[f0_, x0_, {delta_, max_}] :=
 Module[{k, small, cond},
 p0 = x0; Y0 = N[f[p0]]; p1 = p0 + 1; k = 0;
 cond = 0;
 Print["p = ", N[p0]];
 While[cond == 0 && k < max,
  Df = N[f'[p0]];
   If[Df == 0, cond = 1; Dp = p1 - p0;
      p1 = p0, Dp = Y0/Df; p1 = p0 - Dp];
  Y1 = N[f[p1]];
  Print["p = ", N[p1], "\t  |dp| <= ", Abs[N[Dp]]];
   If[N[Abs[p1 - p0]] < delta, cond = 1];
  p0 = p1; Y0 = Y1; k = k + 1; ];
    root = FindRoot[f[x] == 0, {x, 1.5}][[1, 2]];
    Print["Root computed by built-in: ", PaddedForm[root, {16, 16}]];
    Print["error: ", Abs[root - p1]];];
```

4. Examples:

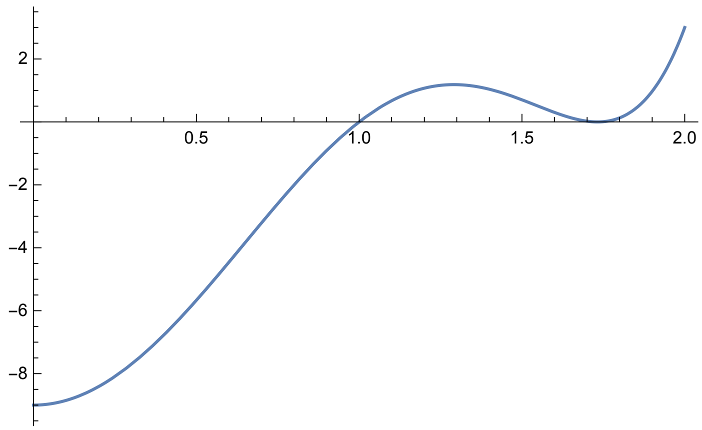
```
Clear[f]
Definition[f]
```

```
Null
```

Example 1)

```
f[x_] := x^6 - 7 x^4 + 15 x^2 - 9
```

```
Plot[f[x], {x, 0, 2.0}]
```




```
NewtonIte[f[x], 1.3, {10^-10, 15}];
```

```
p = 1.3
p = 6.26648      |dp| <= 4.96648
p = 5.28476      |dp| <= 0.981727
p = 4.47862      |dp| <= 0.806138
p = 3.82081      |dp| <= 0.657805
p = 3.28879      |dp| <= 0.532022
p = 2.86383      |dp| <= 0.424962
p = 2.53023      |dp| <= 0.333595
p = 2.27459      |dp| <= 0.255644
p = 2.08503      |dp| <= 0.189556
p = 1.95055      |dp| <= 0.134486
p = 1.86036      |dp| <= 0.0901919
p = 1.80372      |dp| <= 0.0566365
p = 1.77047      |dp| <= 0.0332528
p = 1.75205      |dp| <= 0.0184191
p = 1.74227      |dp| <= 0.00977685
Root computed by built-in:  1.7320507885711760
error: 0.0102201
```

```
Secant[f[x], 1.3, 1.5, {10^-10, 15}];
```

```
p = 1.3
p = 1.5
p = 1.79237      |dp| <= 0.2
p = 1.84084      |dp| <= 0.292369
p = 1.77379      |dp| <= 0.0484729
p = 1.76401      |dp| <= 0.0670505
p = 1.75085      |dp| <= 0.00978548
p = 1.74419      |dp| <= 0.0131542
p = 1.73955      |dp| <= 0.0066577
p = 1.73674      |dp| <= 0.00464384
p = 1.73495      |dp| <= 0.00281481
p = 1.73385      |dp| <= 0.00178243
p = 1.73316      |dp| <= 0.00110302
p = 1.73274      |dp| <= 0.000685893
p = 1.73248      |dp| <= 0.00042466
p = 1.73231      |dp| <= 0.000262948
p = 1.73221      |dp| <= 0.000162652
Root computed by built-in:  1.7320507885711760
error: 0.000162874
```

Muller[f[x], 1.3, 1.4, 1.5, 15];

k= 0 P_0= 1.3, f[p_0] = 1.18411

k= 1 P_1= 1.4, f[p_1] = 1.03834

k= 2 P_2= 1.5, f[p_{02}] = 0.703125

k= 3 p_3 = 1.62765, f[p_3]= 0.202916 |dp|<= 0.127647

k= 4 p_4 = 1.67427, f[p_4]= 0.0698556 |dp|<= 0.0466214

k= 5 p_5 = 1.70349, f[p_5]= 0.0183142 |dp|<= 0.0292192

k= 6 p_6 = 1.72005, f[p_6]= 0.00336299 |dp|<= 0.0165597

k= 7 p_7 = 1.7275, f[p_7]= 0.000491592 |dp|<= 0.00745388

k= 8 p_8 = 1.73063, f[p_8]= 0.0000482516 |dp|<= 0.00312946

k= 9 p_9 = 1.7317, f[p_9]= 2.87942×10^{-6} |dp|<= 0.00107373

k= 10 p_{10} = 1.73199, f[p_{10}]= 9.52692×10^{-8} |dp|<= 0.000283505

k= 11 p_{11} = 1.73204, f[p_{11}]= 1.46193×10^{-9} |dp|<= 0.0000552041

k= 12 p_{12} = 1.73205, f[p_{12}]= 8.63309×10^{-12} |dp|<= 7.20543×10^{-6}

k= 13 p_{13} = 1.73205, f[p_{13}]= 7.10543×10^{-15} |dp|<= 5.86406×10^{-7}

k= 14 p_{14} = 1.73205, f[p_{14}]= -7.10543×10^{-15} |dp|<= 2.22892×10^{-8}

k= 15 p_{15} = 1.73205, f[p_{15}]= 7.10543×10^{-15} |dp|<= 1.50852×10^{-8}

RESULTS:

The function is f[x] = $-9 + 15 x^2 - 7 x^4 + x^6$

The root:

p = 1.7320508017927160

f[p] = 7.10543×10^{-15}

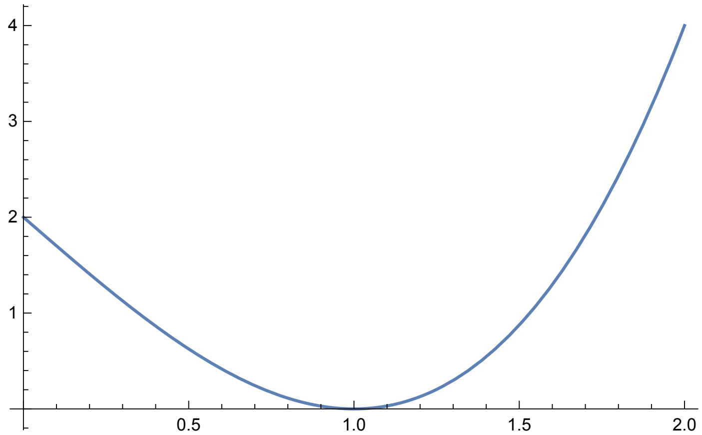
Root computed by built-in: 1.7320507885711760

error: 1.32215×10^{-8}

Example 2)

f[x_] := $x^3 - 3x + 2$

```
Plot[f[x], {x, 0, 2.0}]
```




```
Muller[f[x], 1.2, 1.3, 1.4, 15];
```

k= 0 P_0= 1.2, f[p_0] = 0.128

k= 1 P_1= 1.3, f[p_1] = 0.297

k= 2 P_2= 1.4, f[p_{02}] = 0.544

k= 3 p_3 = 1.01958, f[p_3]= 0.00115769 |dp|<= 0.38042

k= 4 p_4 = 0.985551, f[p_4]= 0.000623337 |dp|<= 0.0340298

k= 5 p_5 = 0.995913, f[p_5]= 0.0000500516 |dp|<= 0.010362

k= 6 p_6 = 1.00004, f[p_6]= 5.85861×10^{-9} |dp|<= 0.00413156

k= 7 p_7 = 0.999987, f[p_7]= 4.98522×10^{-10} |dp|<= 0.0000570818

k= 8 p_8 = 1., f[p_8]= 4.44089×10^{-15} |dp|<= 0.0000129298

k= 9 p_9 = 1., f[p_9]= 0. |dp|<= 3.26293×10^{-8}

RESULTS:

The function is f[x] = $2 - 3x + x^3$

The root:

p = 1.0000000062729790

f[p] = 0.

Root computed by built-in: 1.0000000182935770

error: 1.20206×10^{-8}

```
Secant[f[x], 1.2, 1.5, {10^-10, 15}];
```

```
p = 1.2
p = 1.5
p = 1.14859      |dp| <= 0.3
p = 1.11826      |dp| <= 0.351406
p = 1.06721      |dp| <= 0.0303303
p = 1.04344      |dp| <= 0.0510565
p = 1.02661      |dp| <= 0.023769
p = 1.01659      |dp| <= 0.0168278
p = 1.01025      |dp| <= 0.0100199
p = 1.00635      |dp| <= 0.00633702
p = 1.00393      |dp| <= 0.00390336
p = 1.00243      |dp| <= 0.0024237
p = 1.0015      |dp| <= 0.00149841
p = 1.00093      |dp| <= 0.000927163
p = 1.00057      |dp| <= 0.000573222
p = 1.00035      |dp| <= 0.0003544
p = 1.00022      |dp| <= 0.000219068
Root computed by built-in:  1.0000000182935770
error: 0.000219103
```

```
NewtonIte[f[x], 1.2, {10^-10, 15}];
```

```
p = 1.2
p = 1.10303      |dp| <= 0.0969697
p = 1.05236      |dp| <= 0.0506739
p = 1.0264      |dp| <= 0.0259556
p = 1.01326      |dp| <= 0.0131431
p = 1.00664      |dp| <= 0.00661432
p = 1.00333      |dp| <= 0.00331804
p = 1.00166      |dp| <= 0.00166177
p = 1.00083      |dp| <= 0.000831573
p = 1.00042      |dp| <= 0.000415959
p = 1.00021      |dp| <= 0.000208023
p = 1.0001      |dp| <= 0.000104022
p = 1.00005      |dp| <= 0.0000520138
p = 1.00003      |dp| <= 0.0000260076
p = 1.00001      |dp| <= 0.000013004
```

p = 1.00001 |dp| <= 6.50202×10^{-6}

Root computed by built-in: 1.0000000182935770

error: 6.48376×10^{-6}