General method:

out implementation of general method is based on dividing a specific area (defined in matlab code as it was stated in assignment that general method mustn’t take any input if it was acceptable the solution will be adding this inputs in gui or as file input something like configuration file) into small area and try to find the solution between this area using Brent method that will find the root then another function will take that root and find its multiplicity by differentiation.

Brent Method:

Brent's method is a [root-finding algorithm](https://en.wikipedia.org/wiki/Root-finding_algorithm) combining the [bisection method](https://en.wikipedia.org/wiki/Bisection_method), the [secant method](https://en.wikipedia.org/wiki/Secant_method) and [inverse quadratic interpolation](https://en.wikipedia.org/wiki/Inverse_quadratic_interpolation). It has the reliability of bisection but it can be as quick as some of the less-reliable methods. The algorithm tries to use the potentially fast-converging secant method or inverse quadratic interpolation if possible, but it falls back to the more robust bisection method if necessary.

Advantages:

* Finding all roots with this specific area(all roots except even multiplicity roots)
* Detect near root using a specific parameter eps.

Pitfalls:

* Even multiplicity root.
* Time in large intervals.

Why do we use Brent Method:

* Bracketing method which guarantee convergence if the bracketing condition was true.
* Contains of three methods which may be the fastest possible bracketing method.
* Suitable for our idea of divide range to small areas and search for roots.

Data structure used

* Matrix

Algorithm of Brent method:

**input** *a*, *b*, and (a pointer to) a function for *f*

calculate *f*(*a*)

calculate *f*(*b*)

**if** *f*(*a*)*f*(*b*) ≥ 0 **then** exit function because the root is not bracketed.

**if** |*f*(*a*)| < |*f*(*b*)| **then** swap (*a*,*b*) **end if**

*c* := *a*

**set** mflag

**repeat until** *f*(*b or s*) = 0 **or** |*b* − *a*| is small enough *(convergence)*

**if** *f*(*a*) ≠ *f*(*c*) **and** *f*(*b*) ≠ *f*(*c*) **then**

{\displaystyle s:={\frac {af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))}}+{\frac {bf(a)f(c)}{(f(b)-f(a))(f(b)-f(c))}}+{\frac {cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}}} *(*[*inverse quadratic interpolation*](https://en.wikipedia.org/wiki/Inverse_quadratic_interpolation)*)*

**else**

{\displaystyle s:=b-f(b){\frac {b-a}{f(b)-f(a)}}} *(*[*secant method*](https://en.wikipedia.org/wiki/Secant_method)*)*

**end if**

**if** *(condition 1)* *s* is not between {\displaystyle {\frac {3a+b}{4}}} and *b* **or**

*(condition 2)* (mflag is set **and** |*s*−*b*| ≥ |*b*−*c*|/2) **or**

*(condition 3)* (mflag is cleared **and** |*s*−*b*| ≥ |*c*−*d*|/2) **or**

*(condition 4)* (mflag is set **and** |*b*−*c*| < |*δ*|) **or**

*(condition 5)* (mflag is cleared **and** |*c*−*d*| < |*δ*|)

**then**

{\displaystyle s:={\frac {a+b}{2}}} *(*[*bisection method*](https://en.wikipedia.org/wiki/Bisection_method)*)*

**set** mflag

**else**

**clear** mflag

**end if**

calculate *f*(*s*)

*d* := *c* *(d is assigned for the first time here; it won't be used above on the first iteration because mflag is set)*

*c* := *b*

**if** *f*(*a*)*f*(*s*) < 0 **then** *b* := *s* **else** *a* := *s* **end if**

**if** |*f*(*a*)| < |*f*(*b*)| **then** swap (*a*,*b*) **end if**

**end repeat**

**output** *b* *or s (return the root)*

Algorithm of General method:

­ **input** *f,intervalLen,eps,startInterval,EndInterval*

Calculate pointsOfIntervals

Calculate f(pointsOfIntervals)

**repeat until** *size of f*

**if** *f(i)* **and** *f(i+1) have the opposite signs* **then**

**BrentMethod**

**findMultiplicty**

**addRoot in roots**

end if{\displaystyle s:={\frac {af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))}}+{\frac {bf(a)f(c)}{(f(b)-f(a))(f(b)-f(c))}}+{\frac {cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}}}ende

**end repeat**

**output** *roots(roots with multiplicity and intervals)*

Algorithm of multiplicity method:

­ **input** *f,root*

*multiplicity*:= 1

**repeat until** *f(root) is not equal zero*

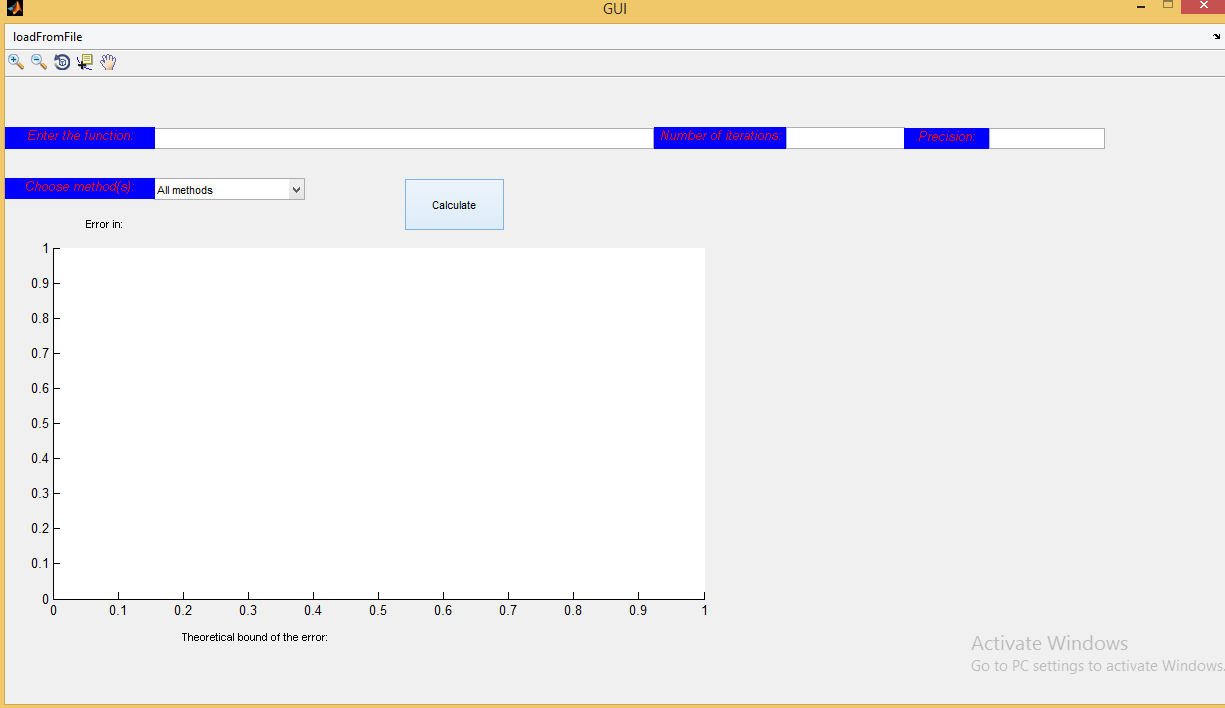
*multiplicity*++

f = differentiate(f)

**end repeat**

**output** *multiplicity(roots with multiplicity and intervals)*

User Guide:



* User should specify method and its required inputs which will be visible after selecting the method.
* Function should be in format of matlab string input functions like[ (x-3\*x-1)]
* Max number of iteration is a positive integer by (default 50)
* Precision is a positive integer by (default .00001)
* If input was through file it should be in the following format
  1. Line by line input.
  2. First line should contain method name typically like (case sensitive).
     + General
     + All methods
     + Bisection
     + False-position
     + Fixed point
     + Newton-Raphson
     + Secant
     + Bierge-Vieta
  3. Required input for each method.
  4. Max number of iteration (optional).
  5. Tolerance (optional).