A Hybrid Method for Solving $f(x) = 0$

Adapted from: *Afternotes on Numerical Analysis* by G.W. Stewart

Based on algorithms by Brent and Wilkinson

The hybrid algorithm below is a combination of bisection and the secant method. The method uses three points a, b, and c. The points a and b are the next points x_k and x_{k-1} in the secant method approximation. The points b and c form a sign change interval (proper bracket) for the root. The idea behind the method is that if the secant method produces an undesirable approximation, we take the midpoint of the sign change interval (next bisection iterate) as our next approximation.

Let $fa = f(a)$, $fb = f(b)$ and $fc = f(c)$ which must satisfy

Conditions:

1. $fa, fb, fc \neq 0$,
2. sign(fb) $\neq$ sign(fc) (sign change interval)
3. $|fb| \leq |fc|$.

The algorithm must be started with points b and $c = a$ which satisfy 1. and 2. above. In addition, the users must provide a tolerance ε so that when the condition $|c - b| \leq \varepsilon$, the algorithm is terminated. The iterations take place in a `while` loop. The program must take care of condition 3. Hence, if $|fb| > |fc|$, we must interchange b and c. In this case, a and b may no longer be a pair of secant iterates, and we must set $a = c$. Below is pseudo code to illustrate the discussion.

```
while{
       if (abs(fc) < abs(fb))
       {
          t=c; c=b; b=t;
          t=fc; fc=fb; fb=t;
          a=c; fa=fc;
       }
```

The convergence criterion is

```
if (abs(b-c) <= epsilon)
   break;
```

The iteration proceeds by first computing the secant approximation s using points a and b and the midpoint (bisection approximation) m using points b and c. We will choose one of these as our next iterate. Since $|fb| \leq |fc|$, we expect that the root r is closer to b then c and should lie in the sign change interval. Thus, if s

lies between b and m then we choose s as the next iterate;
otherwise we choose m.
We must be careful in computing the next iterate since we do not know the relative locations of the points b and c. We must test to find the locations and it is easiest to use the distances $ds = s - b$ and $dm = m - b$. The code below does what is needed and gives the value dd which leads to the next iterate $b + dd$.

```
dm = (c-b)/2;
df=(fa-fb);
if (df==0)
   ds=dm;
else
   ds=-fb*(a-b)/df;
if(sign(ds)!=sign(dm) || abs(ds)>abs(dm))
   dd = dm;
else
   dd = ds;
```

We can also include an adjustment to dd to deal with the situation when the iterates all lie on one side of the root (interval in secant method fails to approach 0) which is

```
if (abs(dd) < epsilon)
   dd = 0.5*sign(dm)*epsilon;
```

We now form the new iterate d

```
d = b + dd;
fd = f(d);
```

Finally, we must rename our variables so that conditions 1.-3. are satisfied and they have the proper interpretations. We check to see if $fd = 0$ (Condition 1)and make temporary assignments.

```
if (fd == 0){
   b=c=d;
   fb=fc=fd;
   break;
}
a=b;  b=d;
fa=fb;  fb=fd;
```

Condition 2 says that b and c form a sign change interval about the root r. If this is not so, simply replace c by the *old* value of b, which is contained in a.

```
if (sign(fb) == sign(fc)){
c=a;   fc=fa;
}
```

The third condition is handled at the beginning of the loop so now we are done.

```
}  % end while loop
```