Secant Method Dataflow Computing

Abdulwahab Alkharashi
2016 Indiana University, Bloomington

Outline

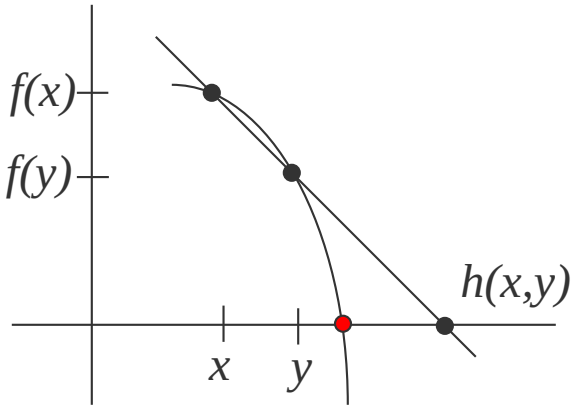
- Secant Method
- Example
- Applications
- Data Interpretation

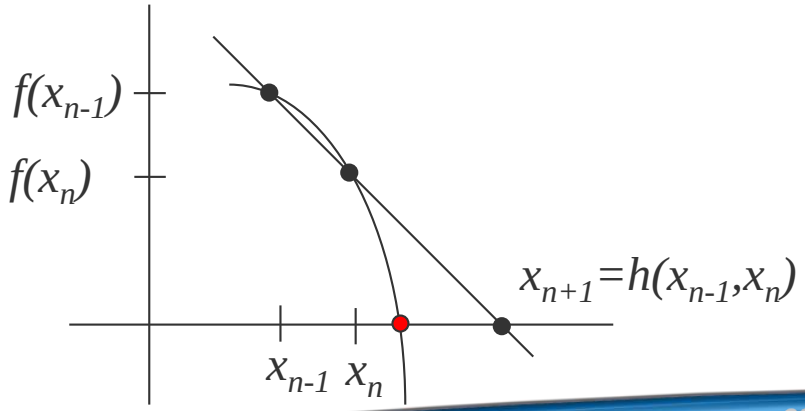
Secant Method

The secant method is a recursive method used to find the solution to an equation like Newton's Method. The idea for it is to follow the secant line to its x-intercept and use that as an approximation for the root. This is like Newton's Method (which follows the tangent line) but it requires two initial guesses for the root.

The big advantage of the secant method over Newton's Method is that it does not require the given function $f(x)$ to be a differential function or for the algorithm to have to compute a derivative. This can be a big deal in other languages since many derivatives can only be estimated.

The recursive function $h(x,y)$ depends on two parameters x and y the x-coordinates of two points on the function.





To get x_{n+1} from x_n and x_{n-1} we write out the equation of the secant line and using the points x_n and x_{n-1}. We then plug in the point $(x_{n+1},0)$ and solve for x_{n+1}.

$$y - f(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x - x_n)$$ equation of secant

$$0 - f(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x_{n+1} - x_n)$$ substitute $(x_{n+1},0)$

$$\frac{-f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} = x_{n+1} - x_n$$ solve for x_{n+1}

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$ $x_{n+1}=h(x_{n-1},x_n)$

The equation above gives the recursively defined sequence for x_n. This is what is used for the Secant Method. The halting condition is usually given by the Standard Cauchy Error.

CPU Code View

```
float rtsec(float (*func)(float), float x1, float x2, float xacc)
{
```

Using the secant method, find the root of a function func thought to lie between x1 and x2. The root, returned as rtsec, is refined until its accuracy is ±xacc

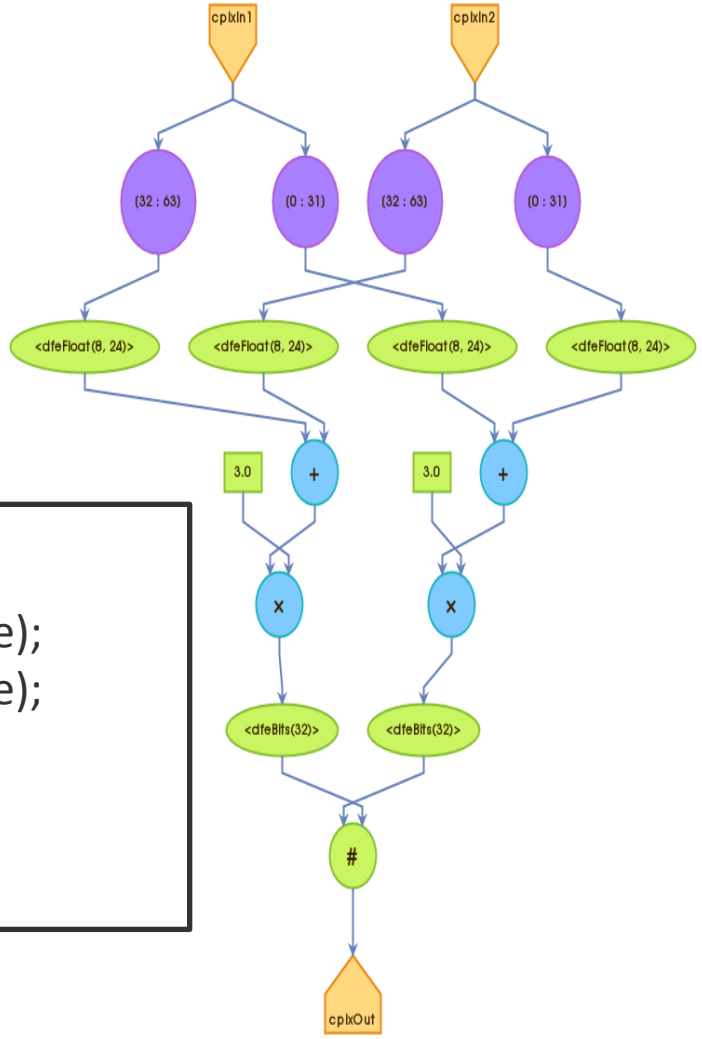
Secant loop

```
for (j=1;j<=MAXIT;j++) {
dx=(xl-rts)*f/(f-fl);
Increment with respect to latest value.
xl=rts; fl=f; rts += dx; f=(*func)(rts);
if (fabs(dx) < xacc || f == 0.0) return rts;
}
}
```

Increment with respect to latest value.

Convergence.

Kernel View




```
DFEComplex cplxIn1 = io.input("cplxIn1", cplxType);
DFEComplex cplxIn2 = io.input("cplxIn2", cplxType);
    DFEComplex result = (cplxIn1 + cplxIn2) * 3;
      io.output("cplxOut", result, cplxType);   }
```

Discussion

The Secant method uses about 25.5 test cases for Mutation Adequate and about 4.4 test cases for the Data Flow Adequate