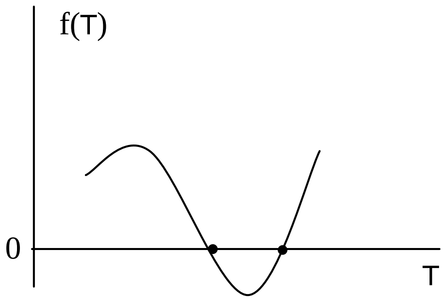
10.34, Numerical Methods Applied to Chemical Engineering
Professor William H. Green
Lecture #6: Modern Methods for Solving Nonlinear Equations.

1D-Problem

unknown: T of reactor

$f(x) = 0$ $\qquad Q_{rxn}\exp(^{-Ea}/_{RT}) + h(T - T_a) + c(T^4 - T_a^4) = 0$

heat of reaction (+) Gain heat — convection (-) Lose heat — radiation (-) Lose heat

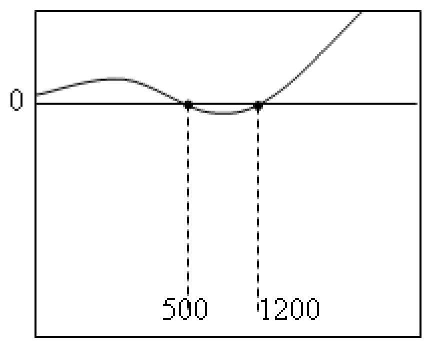


2 steady state temperatures
Make a plot with MATLAB

Figure 1. 1D problem

```
      *netheat.m*
function qdot = netheat(T)
% computes the net heating rate of a reactor
% qdot = 0 at the steady state
qdot = Q.*exp(-Ea/(R.*T)) + h.*(T-Ta) + c.*(T.^4-Ta.^4);
```

```
Q = -2e-5;
Ea = 5000;
R = 1.987;
h = 3;
Ta = 300;
c = 1e-8;
```




```
Tvec = linspace(300,3000)
qdot = netheat(Tvec)
plot(Tvec,qdot)
```

Figure 2. Professor Green modified variables Q and c until the plot looked like the one above. Increased Q and decreased c.

To solve for steady state zeros

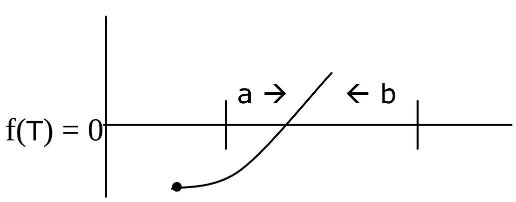


Figure 3. Have computer bracket in and find small range where plot goes from negative to positive.

Bisection

start a,b
such that f(a)<0 and f(b) < 0

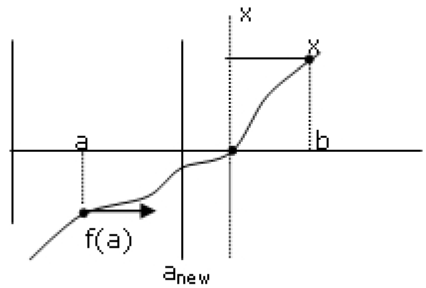
$$x = \frac{a+b}{2}$$

if f(x) · f(a) > 0
  a = x
else
  b = x

This is a problem of TOLERANCE

if((b-a) < tol) stop

Figure 4. Function must be continuous.



Types of tolerance

Absolute tolerance
atol: has units
if |f(x)| < $\underbrace{\text{atol}\cdot\text{f}}_{\text{has to be BIG number}}$

Relative tolerance

rtol: if(b-a) < rtol*|a|

```
while abs(b-a) > atolx
        x = (a+b)/2
if f(x)·f(a) > 0
        a = x
else
        b = x
end
```

<u>In MATLAB</u>

```
    *bisect.m*
function x = bisect(f,a,b,atolx,rtolx,
                                atolf)
%solves f(x) = 0
while abs(b-a) > atolx
      x = 0.5*(b+a);
      if((feval(f,x)*feval(f,a))>0)
            a=x;
      else
            b=x;
      end
end
```

<u>Command Window</u>

```
x = bisect(@netheat,300,2000,0.1,0,0)
x = 1.2373e+003
```

CHECK: netheat(1237) = -1.0474 ← close
Keep in mind: never get actual solution, but can come close

We can change tolerances to improve results.

→
```
while(abs(b-a)>atolx)&&(abs(b-a)>(rtolx*abs(a)))
x = 0.5*(b+a);
if(abs(feval(f,x))<atolf)
      return            %if value becomes low enough, return value
```
AND: must satisfy both conditions

```
x = bisect(@netheat,300,2000,0.1,1e-2,0.5)
x = 1.2363e+003
```
looser tolerance gives less accurate answer

Bisection cuts interval by 2 each time

Every time we cut 3 times, we lose a sig fig

In bisection, time grows linearly with the number of significant figures.

$a < x^{true} < b$
$x^{true} = x^{soln} \pm {}^{b-a}/_2$

Newton's Method (1-D)

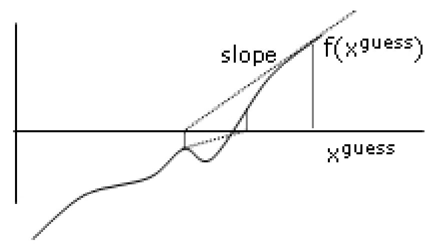


Figure 5. Newton's Method.

evaluates slope of f(x)
next guess is the x_{new} that satisfies $f(x_{new})=0$ for a line from $f(x_{guess})$ with the slope at $f(x_{guess})$

$f(x) = f(x_0)+f'(x_0)*(x-x_0)+O(\Delta x^2)$

$0 = f(x^{guess})+f'(x^{guess})*(x-x^{guess})$

$x^{new} = x^{guess} - f(x^{guess})/f'(x^{guess})$

For a good guess Newton's method doubles the number of significant figures after every iteration; however, we lose robustness if guess is poor

If $f'(x^{guess}) \approx 0$ -- doesn't work

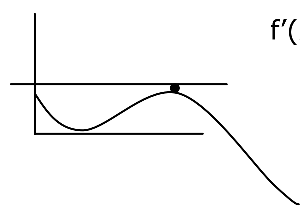


Figure 6. NO intersection

Another drawback is one needs a derivative of the function.

Secant Method

same as Newton's, but uses f'(x) approximate

$$f'^{approx}(x) = \frac{f(x^{[k]}) - f(x^{[k-1]})}{x^{[k]} - x^{[k-1]}}$$

Bisection method works only for 1D problems, but *Newton/Secant* can be used for problems with greater dimension

Broyden's Method (Multi-dimensional)

$\underline{F}(\underline{x}) = \underline{F}(\underline{x}_0) + \underbrace{\underline{\underline{J}}(\underline{x}_0)\cdot(\underline{x}-\underline{x}_0)}_{\sum_j \left.\left(\frac{\partial f_i}{\partial x_j}\right)\right|_{x_o} (x_j - x_{o,j})}$

Method breaks down when $\underline{\underline{J}}$ is singular

f(x) = 0

approx $\underline{\underline{J}} = \underline{\underline{B}}$

$$\mathbf{B}^{[k+1]} = \mathbf{B}^{[k]} + \frac{\overbrace{F(\mathbf{x}^{[k+1]}) * (\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]})^{\mathrm{T}}}^{\text{outer product is opposite of dot product}}}{\| \Delta\mathbf{x} \|^2}$$

Outer Product: $\begin{pmatrix} F_1\Delta x_1 & F_1\Delta x_2 & F_1\Delta x_3 & \cdots \\ F_2\Delta x_1 & F_2\Delta x_2 & F_2\Delta x_3 & \ldots \end{pmatrix}$

Newton's Method (Multi-dimensional)

$\underline{O} = \underline{F}(\underline{x}_0)+\underline{\underline{J}}(\underline{x}_0)\cdot(\underline{x}-\underline{x}_0)$

$\underline{\underline{J}}*\Delta\underline{x} = -\underline{F}(\underline{x}_0)$
LU

LU → $\underline{\underline{B}}^{[k]}\Delta\underline{x} = -\underline{F}$

LU → $LU^{[k+1]}$ without redoing factorization

Done in detail in homework problem.