Applied Engineering Problem Solving

Lesson #5: Root Finding & Polynomial Manipulations

Prof. John R. White
Chemical and Nuclear Engineering
UMass-Lowell, Lowell MA

Lesson #5 Goals

Root Finding: What is the real value of x such that f(x) = 0?

Find the real roots of algebraic and transcendental equations

Find all the roots, both real and complex, of a polynomial equation

Use the methods within several illustrative applications

Introduce a sequence of Matlab commands for performing a variety of polynomial operations and manipulations

Gilat:
Chapters 8 and 9

Chapra:
Chapters 5 and 6
(browse Chapter 7 on Optimization)

Lesson # 5 Lecture Notes and Illustrative Examples

Root Finding: Find the roots of $a_1x^n + a_2x^{n-1} + \cdots a_nx + a_{n+1} = 0$

Motivation Problems

Problem 1: Volume of Liquid in a Horizontal Cylindrical Tank

The volume of liquid in a horizontal cylindrical tank is given by

$$V = \left[R^2 \cos^{-1}\left(\frac{R-h}{R}\right) - (R-h)\sqrt{2Rh-h^2} \right] L$$

where R is the tank inside radius, L is the length, and h is the height of liquid.

Case 1: Consider a particular tank with R = 2.5 m and L = 5 m. If the fluid height is 3.0 m, what is the fluid volume in the tank?

explicit
V = f(h,R,L)

Case 2: If another 1.5 m^3 of liquid is added to the tank in Part a, what will be the new value of fluid height?

implicit
F(h) = V - f(h,R,L) = 0

Motivation Problems (cont.)

Problem 2: van der Waal's Equation of State

van der Waal's equation of state is given as

$$\left(P + \frac{a}{\upsilon^2}\right)(\upsilon - b) = RT$$

where a and b are constants for the particular gas of interest.

Provide a plot for the molar volume of ammonia versus temperature for several pressures of interest.

Pressure: 1, 3, and 5 atms

Temperature: 250 < T < 400 K

a = 4.19 atm-(liters/gmole)^2
b = 0.0372 liters/gmole

This is an implicit problem

loop over all P_i and T_j

What is υ_{ij} such that $f(\upsilon_{ij}) = 0$?

$$f(\upsilon_{ij}) = \left(P_i + \frac{a}{\upsilon_{ij}^2}\right)(\upsilon_{ij} - b) - RT_j = 0$$

Motivation Problems (cont.)

Problem 3: Implicit Solutions to IVPs

The implicit solution to the IVP defined by

$$y' = \frac{-\left(2xy^3 + y^4\right)}{xy^3 - 2} \quad \text{with} \quad y(0) = 1$$

can be written as

$$u(x,y) = x^2 + xy + y^{-2} - 1 = 0$$

Plot the solution, y(x), over the domain $0 \le x \le 2$.

This is another implicit problem

loop over all x_i

Given x_i, what is y_i such that $f(y_i) = 0$ where

$$f(y_i) = x_i^2 + x_i y_i + y_i^{-2} - 1 = 0$$

Motivation Problems (cont.)

Problem 4: Roots of the Characteristic Equation

Find the general solution to the following 4th order ODE:

$$y^{(4)} + 3y' - 4y = 0$$

This is a homogeneous linear constant coefficient system with a solution of the form $y \to e^{rx}$.

Upon substitution, we get the characteristic equation,

$$r^4 + 3r - 4 = 0$$

This is a 4th order polynomial that has 4 roots, r_1, r_2, r_3, and r_4, and, assuming that the roots are distinct, the general solution can be written as

$$y(x) = c_1 e^{r_1 x} + c_2 e^{r_2 x} + c_3 e^{r_3 x} + c_4 e^{r_4 x}$$

both real and complex roots

The goal here is to find all four roots of the characteristic equation.

Motivation → Need

These motivation problems clearly suggest a need for numerical tools to solve problems of this type…

Case 1: Find the real roots of algebraic and transcendental equations

fzero

Case 2: Find all the roots, both real and complex, of a polynomial equation

roots

Our job will be to overview some basic techniques and some Matlab tools for treating these classes of problems…

Root Finding: Real Roots

Two approaches: Bracketing Methods and Open Methods

Advantages and Disadvantages

Bracketing Methods

Require two initial guesses that are on either side of a root.

Bracketing is maintained as the solution algorithm continually reduces the size of the bracket until convergence is reached.

Bracketing methods are always convergent, but the rate of convergence is usually relatively slow.

Bracketing methods always work but are relatively slow

Root Finding: Real Roots (cont.)

Two approaches: Bracketing Methods and Open Methods

Advantages and Disadvantages

Open Methods

Initial guess or guesses do not need to bracket a root.

Information about the function and its derivative at the root estimate are used to efficiently extrapolate to a new root.

However, open methods are not guaranteed to converge, and the success of the method may depend on the goodness of the initial guess.

Open methods may diverge, but they usually converge quite rapidly when they work

Root Finding: Real Roots (cont.)

Two approaches: Bracketing Methods and Open Methods

Advantages and Disadvantages

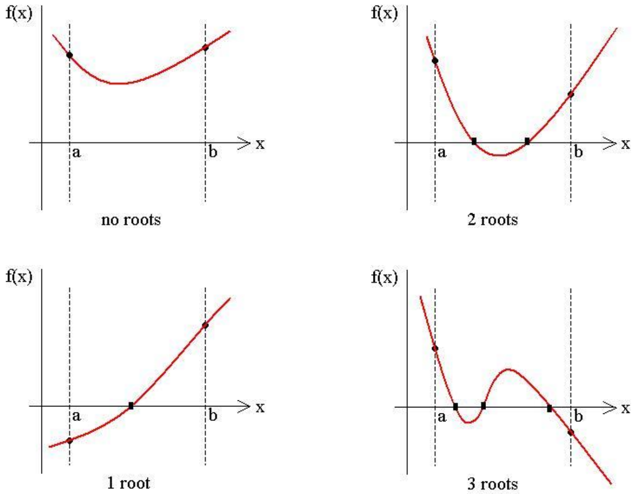
Hybrid Methods

At the expense of some additional programming detail, it is possible to combine a bracketing method and an open technique to give a Hybrid Method that keeps the best qualities of both methods -- producing a fast method which always converges.

In particular, the *fzero* routine in Matlab uses a hybrid approach to give a practical tool that is both robust and efficient.

Matlab's *fzero* routine is robust and efficient and represents a good choice for general purpose applications

Bracketing Methods



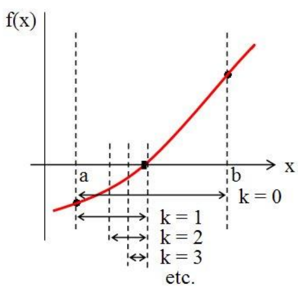
All bracketing methods are based on the fact that, if f(x) is continuous, there is at least one zero crossing within the interval $a \leq x \leq b$ if $f(a)f(b) < 0$

If f(a) and f(b) have different signs, then there must be at least one zero crossing within the interval [a,b]

Bisection Method

Algorithm:

1. **Choose $b > a$ such that $f(a)f(b) < 0$ which guarantees at least one root within the interval [a,b]. Also set the convergence criterion, tol, and the maximum number of iterations, M.**
2. **Evaluate $x_r = (a+b)/2$ and compute $f(x_r)$.**
3. **If $f(x_r)f(a) < 0$, set $b = x_r$ (root is in the first half interval). If not, set $a = x_r$.**
4. **Increment iteration counter, $k = k+1$.**
5. **If $k \leq M$ and $|f(x_r)| > tol$, go to Step 2.**
6. **At this point, if $k\text{-}1 \leq M$, x_r is the desired estimate of the root -- use it as needed.**



see bisection.m

CHEN.3170 Applied Engineering Problem Solving
Lesson #5: Root Finding and Polynomial Manipulations (Nov. 2017)

Open Methods

The basic idea behind all open methods is quite straightforward.

Instead of looking for a zero crossing as done for bracketing methods, open methods actually search for a value of x where two functions are equal.

To see this, let's start with the original statement that f(x) = 0, where our goal is to find a real value of x that satisfies this condition.

Let's write f(x) as the difference between two functions, or

$$f(x) = h(x) - g(x) = 0$$

which leads to $h(x) = g(x)$

In most cases, we simply let h(x) = x, giving $x = g(x)$

This suggests an iterative process, or $\boxed{x_{k+1} = g(x_k)}$

One-Point Iteration

One-Point Iteration – An Example

Problem: Find the root of f(x) = x – e^{-x}.

This can be written two ways:

$\boxed{x = e^{-x} \quad \rightarrow \quad x_{k+1} = e^{-x_k}}$ or as $\boxed{x = -\ln x \quad \rightarrow \quad x_{k+1} = -\ln x_k}$

With a first guess x_1 = 1.0, we obtain →

k		x_k	$g_1(x_k) = e^{-x_k}$		x_k	$g_2(x_k) = -\ln x_k$
1		1.0	0.36788		1.0	0.0
2		0.36788	0.69220		0.0	undefined
3		0.69220	0.50047		(we have divergence)	
4		0.50047	0.60625			
5		0.60625	0.54539			
6		0.54539	0.57962			
7		0.57962	0.56011			
8		0.56011	0.57115			
9		0.57115	0.56488			
10		0.56488	0.56843			

As apparent, one case works and the other fails miserably!

This is the primary disadvantage associated with all open methods -- sometimes they work and sometimes they don't…

Newton's Method (from Taylor Series)

The one-point iteration scheme outlined above is not usually implemented in this fashion because of the arbitrary nature for choosing the iteration function, g(x).

The most common one-point iteration algorithm used in practice, which has a specific representation for g(x), is the Newton-Raphson (NR) method (often simply referred to as Newton's method).

The iteration formula for this method is easily derived using a truncated Taylor series expansion,

$$f_{k+1} = f_k + f'_k\left(x_{k+1} - x_k\right) + O(\Delta x^2)$$

Dropping the error term and solving this expression for x_{k+1} gives

$$x_{k+1} - x_k = \frac{f_{k+1} - f_k}{f'_k} \quad \text{or} \quad \boxed{x_{k+1} = x_k + \frac{f_{k+1} - f_k}{f'_k}}$$

Newton's Method (Graphical Approach)

From the figure, we see that a forward approximation for the first derivative at x_k, is given by

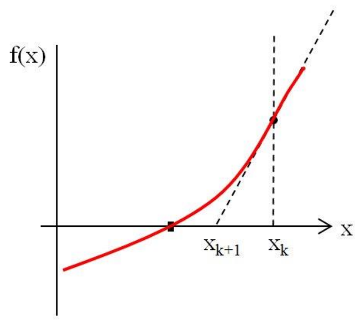
$$f'_k = \frac{f_{k+1} - f_k}{x_{k+1} - x_k}$$

Rearranging this gives,

$$x_{k+1} - x_k = \frac{f_{k+1} - f_k}{f'_k}$$

or

$$\boxed{x_{k+1} = x_k + \frac{f_{k+1} - f_k}{f'_k}}$$



This is the same result as obtained from the Taylor Series

Newton and Secant Methods

Now, the next guess for a root, x_{k+1}, will be chosen so that, by definition, $f(x_{k+1}) = f_{k+1} = 0$ (i.e. if x_{k+1} is the root, then $f_{k+1} = 0$).

Thus, we have

$$\boxed{x_{k+1} = x_k - \frac{f_k}{f'_k}}$$

Newton's Method

For general purpose use, it is often inconvenient to require the evaluation of both f(x) and f ′(x) at each root estimate, x_k.

The Secant method is similar to the NR method except that f'_k is approximated by a backward FD formula,

$$\boxed{f'_k = \frac{f_k - f_{k-1}}{x_k - x_{k-1}}}$$

and substitution gives

$$\boxed{x_{k+1} = x_k - f_k\left[\frac{x_k - x_{k-1}}{f_k - f_{k-1}}\right]}$$

Secant Method

One-Point Iteration (Secant Method)

Algorithm:

1. Make two guesses for the root, x_{k-1} and x_k, for k = 1. Also set the convergence criterion, tol, and the maximum number of iterations, M.
2. Evaluate the iteration formula and compute $f(x_{k+1}) = f_{k+1}$.

$$\boxed{x_{k+1} = x_k - f_k\left[\frac{x_k - x_{k-1}}{f_k - f_{k-1}}\right]}$$

see secant.m

3. Increment the iteration counter, k = k+1.
4. If $k \le M$ and $|f(x_{k+1})| > tol$, go to Step 2.
5. At this point, if $k-1 \le M$, x_{k+1} is the desired root estimate -- use it as desired.

Matlab's *fzero* Routine…

Matlab's *fzero* routine is robust and efficient and represents a good choice for general purpose applications

fzero **implements a hybrid combination of the robust bisection method and two open methods (secant method and an inverse quadratic interpolation method) for increased efficiency.**

see discussion of Brent's Method in Chapra

Basic Syntax:

```
fx = @(x) func_name(x, p1, p2, …)
```

see roots_1 demo

see optimset function

```
xr = fzero(fx, xg, options)
```

where `func_name` is the name of the function file containing the implicit function, f(x) = 0, and `fx` is a handle to the function.

Check out the "Motivation Problems"

Problem 1: Volume of Liquid in a Horizontal Cylindrical Tank

$$V = \left[R^2 \cos^{-1}\left(\frac{R-h}{R}\right) - (R-h)\sqrt{2Rh - h^2}\right]L$$

see horizontal_cyl_1.m

Problem 2: van der Waal's Equation of State

$$\left(P + \frac{a}{\upsilon^2}\right)(\upsilon - b) = RT$$

see eqofst_1.m

Problem 3: Implicit Solutions to IVPs

Plot the solution, y(x), where $x^2 + xy + y^{-2} - 1 = 0$

see analytical_ivp1.m

Polynomial Manipulations

Algebraic manipulations with polynomials, such as **addition**, **subtraction**, **multiplication**, and **division**, as well as simple **polynomial evaluation**, are quite routine for many applications.

And, of course, we are also interested in obtaining **all the roots** of polynomial equations.

In **Matlab**, all these operations are relatively easy to do…

In particular, a **generic n^{th} order polynomial**

$$f(x) = a_1 x^n + a_2 x^{n-1} + \cdots a_n x + a_{n+1}$$

is represented by a **row vector of length n+1**

$$p = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n & a_{n+1} \end{bmatrix}$$

Polynomial Manipulations (cont.)

Consider the following three polynomials:

$f_1 = x^5 - 3x^4 - 10x^3 + 10x^2 + 44x + 48 \rightarrow$ p1 = [1 -3 -10 10 44 48]

$f_2 = x^2 + 2x + 2 \rightarrow$ p2 = [1 2 2]

$f_3 = x^2 + 2x + 1 \rightarrow$ p3 = [1 2 1]

Addition/Subtraction: $f_2 \pm f_3 \rightarrow$ p2 ± p3

$f_1 \pm f_2 \rightarrow$ p1 ± [0 0 0 p2]

same size

Multiplication: $f_1 * f_2 \rightarrow$ conv(p1,p2)

Division: $f_1 / f_2 \rightarrow$ [Q,R] = deconv(p1,p2)

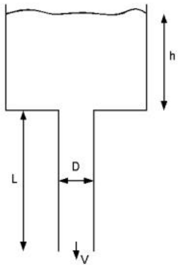
Polynomial Evaluation: $f_1(x) \rightarrow$ x = linspace(xo,xf,#)
f1 = polyval(p1,x)

Horner's Rule

Polynomial Roots: x values that satisfy $f_1(x) = 0 \rightarrow$ r1 = roots(p1)

Etc…

More Illustrative Examples



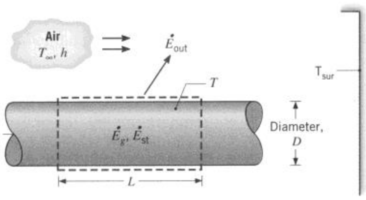
Friction Effects in Pipe Flow

Determine the exit velocity, V, vs. pipe length, L, for different pipe diameters, D. This example addresses how pipe friction affects the flow rate in the system…

see
pipe_friction_1.pdf

Energy Balance on a Conducting Rod

Determine the steady state temperature of a metal rod as a function of the current, I, in the rod. This heat transfer problem involves convection and radiation heat transfer from the surface to the surroundings…



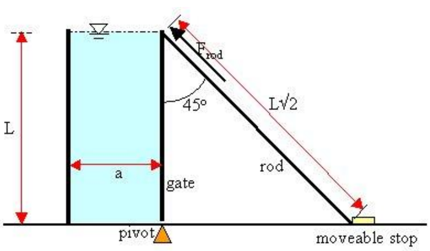
see
conducting_rod_1.pdf

More Illustrative Examples (cont.)

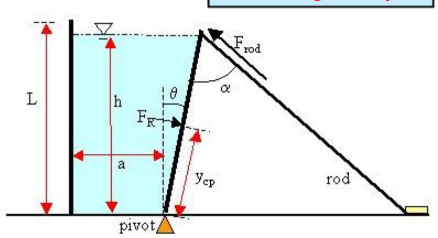
Comprehensive Analysis of a Slanted Gate

The goal of this particular example is to analyze this water reservoir system in detail and to answer several questions concerning the system when the gate is at various angles, θ.

see
slanted_gate_1.pdf



Initial configuration of gate



Gate at some angle θ