Jong Park

Jan 30, 2018

Math 351 – Numerical Analysis

Prof. Higdon

HW 2

**Compute x = ln b = logeb, b is positive.**

1. Equation in form f(x) = 0 and b is root. Newton’s method to derive an algorithm for computing approximate solution. May use exponential function, but not log.
2. For what choices of initial guess x0 does the iteration coverage. Include diagram of iteration (graph of f and some tangent lines).

Note: Disregard overflow, underflow in finite-precision floats.

1. Ill conditioning of roots of polynomials.

For any positive integer n, let

for all real x. Roots = 1, 2, …, n.

**Perturbed polynomial:**

1. Execute the line in pert.m

format long

n=2; % only one with real root.

n=5; % Try these values as well.

n=10;

n=20;

n=21; % The computed roots aren't even real.

p = poly(1:n); % p Contains coefficients (in descending order of a polynomial whose roots are components of the vector 1:n

r = roots(p); % r contains the computed roots of this polynomial.

>> n=2; p=poly(1:n); r=roots(p)

r =

2

1

>> n=5; p=poly(1:n); r=roots(p)

r =

5.000000000000138

3.999999999999776

3.000000000000124

1.999999999999966

1.000000000000004

>> n=10; p=poly(1:n); r=roots(p)

r =

10.000000000340687

8.999999998493760

8.000000002751277

6.999999997320477

6.000000001506791

4.999999999494381

4.000000000107472

2.999999999983562

2.000000000001650

0.999999999999950

>> n=20; p=poly(1:n); r=roots(p)

r =

19.999874055724192

19.001295393676987

17.993671562737585

17.018541647321989

15.959717574548915

15.059326234074415

13.930186454760916

13.062663652011070

11.958873995343460

11.022464271003383

9.991190949230132

9.002712743189727

7.999394310958664

7.000096952230211

5.999989523351082

5.000000705531480

3.999999973862455

3.000000000444877

1.999999999998383

0.999999999999949

>> n=21; p=poly(1:n); r=roots(p)

r =

20.998100413694775 + 0.000000000000000i

20.017033950662430 + 0.000000000000000i

18.918164261947613 + 0.000000000000000i

18.171333577200958 + 0.000000000000000i

16.515480336345934 + 0.131337608147565i

16.515480336345934 - 0.131337608147565i

14.714057397598715 + 0.000000000000000i

14.190464878645530 + 0.000000000000000i

12.953491191832381 + 0.000000000000000i

12.003238491516015 + 0.000000000000000i

11.005259635037602 + 0.000000000000000i

9.997370845609455 + 0.000000000000000i

9.000560720953006 + 0.000000000000000i

7.999976310600354 + 0.000000000000000i

6.999984161455235 + 0.000000000000000i

6.000003849858241 + 0.000000000000000i

4.999999627121142 + 0.000000000000000i

4.000000013554953 + 0.000000000000000i

3.000000000022612 + 0.000000000000000i

1.999999999997278 + 0.000000000000000i

0.999999999999971 + 0.000000000000000i

1. Given n =7, perturb the coefficient of x6 by subtracting 0.002 of that coefficient. (Example 3.5.4, p 112-113).

Use roots function to compute the roots of the unperturbed polynomial () and perturbed () polynomial.

Display the roots of these polynomials side-by side in adjacent columns of an array.

Use format long.

%% 2. part b

epsilon = -0.002;

n=7;

v = poly(1:n); % Creates vector of polynomials from 1 to n

vpert = v;

vpert(2) = vpert(2) + epsilon; % Subtract eps from coefficient of x^6

r = roots(v); % Roots of normal polynomial

rpert = roots(vpert); % Roots of perturbed polynomial

format long

pertable = [sort(r(:)) sort(rpert(:))]

pertable =

1 1.000000000000020 + 0.000000000000000i 1.000002777842993 + 0.000000000000000i

2 1.999999999999781 + 0.000000000000000i 1.998938173110114 + 0.000000000000000i

3 3.000000000000553 + 0.000000000000000i 3.033125347258262 + 0.000000000000000i

4 4.000000000000223 + 0.000000000000000i 3.819569248146413 + 0.000000000000000i

5 4.999999999997638 + 0.000000000000000i 5.458675826856370 - 0.540125780968025i

6 6.000000000002801 + 0.000000000000000i 5.458675826856370 + 0.540125780968025i

7 6.999999999999008 + 0.000000000000000i 7.233012799929453 + 0.000000000000000i

1. Explain the difference between the unperturbed polynomial () and perturbed () polynomial using graphs. (2 of the roots of isn’t real).

%% 2. part c

dx = 0.05;

x = 0 : dx : n+1;

y = polyval(v, x);

ypert = polyval(vpert, x);

perturbation = epsilon \* x.^(n-1);

n1 = 1+(1/dx); n2 = 1+(n/dx); % Indicies for x=1 and x=n respectively.

maxy = max(abs(y(n1:n2)));

maxypert = max(max(ypert(n1:n2)));

maxperturb = max(max(perturbation(n1:n2)));

maxy = 1.1 \* max([maxy, maxypert, maxperturb]);

plot(x, y, x, ypert, '--', x, perturbation, ':', ...

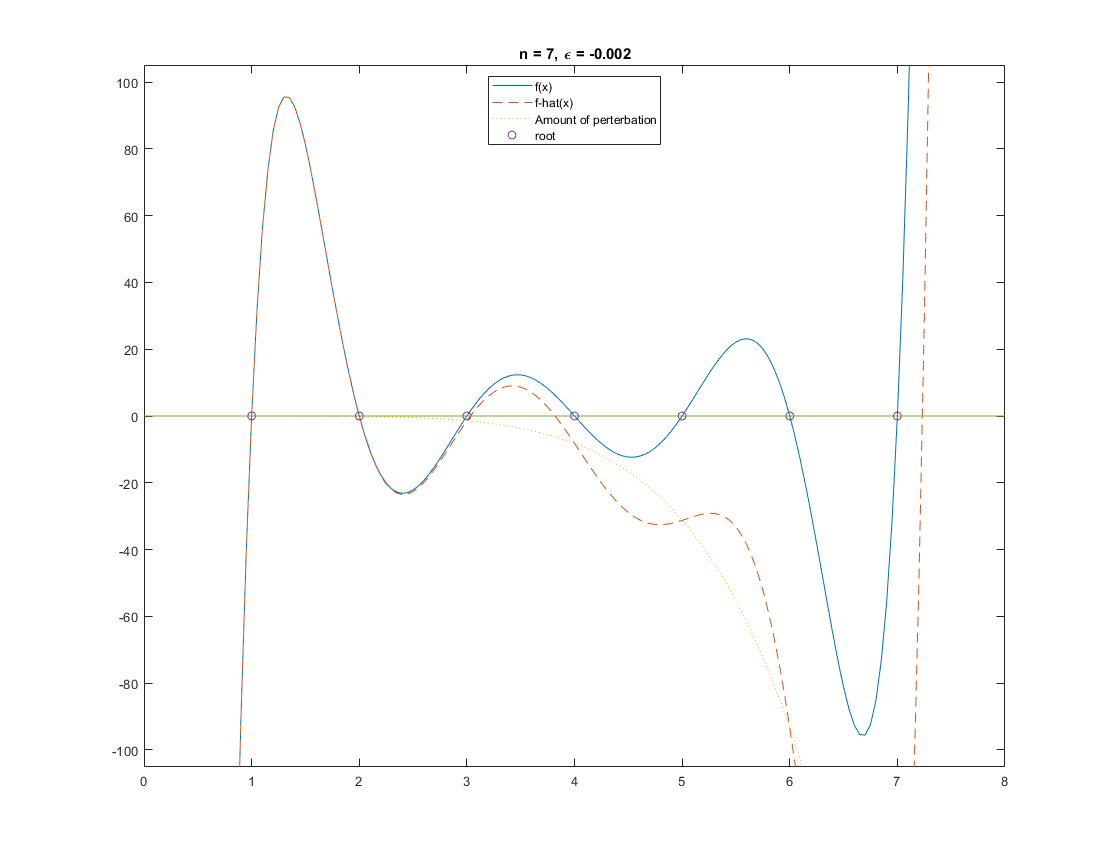
1:n, zeros(1,n), 'o', x, zeros(size(x)), '-')

axis([0 n+1 -maxy maxy])

s = ['n = ', num2str(n), ', \epsilon = ', num2str(epsilon)];

title(s)

legend({'f(x)','f-hat(x)','Amount of perterbation','root'},'Location','north')

You can see from the graph that both and has similar roots from n=1 to n=4. However, on n=5 and n=6, the (dotted line) does not cross the x-axis because at n=5 and n=6, it has imaginary root for . On n=7, the crosses the x-axis again because it has real roots.