Math128aHw3

Trustin Nguyen

September 25, 2024

Exercise Set 2.5

Exercise 2: Consider the function $f(x) = e^{6x} + 3(\ln 2)^2 e^{2x} - (\ln 8)e^{4x} - (\ln 2)^3$. Use Newton's method with $p_0 = 0$ to approximate a zero of f. Generate terms until $|p_{n+1} - p_n| < 0.0002$. Construct the sequence $|\hat{p}_n|$. Is the convergence improved?

Answer. * The code is below Using newton's method, I got $x = -0.1534$. The convergence is improved, and the calculated zero is at:

$$x = -0.1861550957$$

When we evaluate the function with the linearly convergent sequence, we get:

$$f(x) = 0.000077450866$$

but with Aitken's method we get:

$$f(x) = -0.00000006432390177$$

Here is the code:

```
function out = newton(f, p0, metric, tol, x_true)
    if ~exist('x_true')
        x_true = 0;
    end
    digits(10)
    x = p0;
    out.x = [vpa(p0)];
    out.p_hat = [];
    out.y = [vpa(f(p0))];
    out.y_hat = [];
    syms c;
    df = matlabFunction(diff(f(c), c));
    i = 1;
    while 1
        if metric(x, x_true, f) < tol
            break;
        end
        x = x - f(x) / df(x);
        out.x = [out.x; vpa(x)];
        i = i + 1;
        if i >= 3
            p_n = vpa(Aitken(out.x(end-2:end)));
            out.p_hat = [out.p_hat; p_n];
            out.y_hat = [out.y_hat; f(p_n)];
```

```
        end
        out.y = [out.y; vpa(f(x))];
    end

end

function p_hat = Aitken(x)
    delx = x(2:end) - x(1:end-1);
    del2x = delx(end:end) - delx(1:1);
    p_hat = x(1:1) - ((delx(1:1))^2 ./ del2x);

end


function y = myfunc1(x)
    y = exp(6*x) + 3*log(2)^(2)*exp(2*x) - (log(8))*exp(4*x) - log(2)^(3);
end

out = newton(@myfunc1, 2.5, @out_error, 0.0002)
x = out.x
y = out.y
p_hat = out.p_hat
y_hat = out.y_hat
iter = size(x, 1)
```

Exercise 4: Let $g(x) = 1 + (\sin x)^2$ and $p_0^{(0)} = 1$. Use Steffensen's method to find $p_0^{(1)}$ and $p_0^{(2)}$.

Answer. I got $p_0^1 = 2.1529$ and $p_0^{(2)} = 1.8735$. And also:

$$g(p_0^1) = 1.6977 \qquad g(p_0^2) = 1.9112$$

Here is the code:

```
function out = steffensen(f, p, tol)
    out.x = [p];
    out.y = [f(p)];
    while 1
        p0 = f(p);
        p1 = f(p0);
        aitk = Aitken([p, p0, p1]);
        out.x = [out.x; vpa(aitk)];
        out.y = [out.y; vpa(f(aitk))];
        if abs(p - aitk) < tol
            break
        end
        p = aitk;
    end

end

function p_hat = Aitken(x)
    delx = x(2:end) - x(1:end-1);
    del2x = delx(end:end) - delx(1:1);
    p_hat = x(1:1) - ((delx(1:1))^2 ./ del2x);

end
```

```
function y = myfunc2(x)
    y = 1 + (sin(x))^2;
end

out = steffensen(@myfunc2, 1, 1e-5);
x = out.x
y = out.y
```

Exercise 7: Use Steffensen's method to find, to an accuracy of 10^{-4}, the root of $x^3 - x - 1 = 0$ that lies in $[1, 2]$ and compare this to the results of Exercise 8 of Section 2.2.

Answer. I got:

$$x = 1.324717994 \qquad f(x) = 0.0000008245189529$$

Here is the code:

```
function out = steffensen(f, p, tol)
    out.x = [p];
    out.y = [f(p)];
    while 1
        p0 = f(p);
        p1 = f(p0);
        aitk = Aitken([p, p0, p1]);
        out.x = [out.x; vpa(aitk)];
        out.y = [out.y; vpa(f(aitk))];
        if abs(p - aitk) < tol
            break
        end
        p = aitk;
    end

end

function p_hat = Aitken(x)
    delx = x(2:end) - x(1:end-1);
    del2x = delx(end:end) - delx(1:1);
    p_hat = x(1:1) - ((delx(1:1))^2 ./ del2x);

end


function y = myfunc3(x)
    y = x^3 - 1;
end

out = steffensen(@myfunc3, 1.5, 1e-4);
x = out.x
y = out.y
x_res = y(end:end);
y_res = x_res^3 - x_res - 1
```

Exercise 14: A sequence $\{p_n\}$ is said to be **superlinearly convergent** to p if

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} = 0$$

a. Show that if $p_n \to p$ of order α for $\alpha > 1$, then $\{p_n\}$ is superlinearly convergent to p

Proof. Since $p_n \to p$ of order α, we know that

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda$$

for some $\lambda > 0$ constant. Since our sequence $s_n = \frac{|p_{n+1}-p|}{|p_n-p|^{\alpha}}$ is nonzero for sufficiently high $n > N$, we can let another sequence $t_n = \{1\}_{n\geqslant 0}$. Then we have:

$$\lim_{n\to\infty} \frac{t_n}{s_n} = \frac{\lim_{n\to\infty} 1}{\lim_{n\to\infty} s_n} = \frac{1}{\lambda}$$

Now we can take the product of our sequence t_n/s_n with $s'_n = \frac{|p_{n+1}-p|}{|p_n-p|}$ to get:

$$\lim_{n\to\infty} \frac{t_n}{s_n} \cdot s'_n = \lim_{n\to\infty} \frac{|p_n - p|^{\alpha}}{|p_{n+1} - p|} \cdot \frac{|p_{n+1} - p|}{|p_n - p|} = \lim_{n\to\infty} |p_n - p|^{\alpha-1} = 0 = \lim_{n\to\infty} \frac{t_n}{s_n} \cdot \lim_{n\to\infty} s'_n$$

So either $\lim_{n\to\infty} t_n/s_n = 0$ or $\lim_{n\to\infty} s'_n = 0$. But $\lim_{n\to\infty} \frac{t_n}{s_n} = \frac{1}{\lambda} \neq 0$. So $\lim_{n\to\infty} s'_n = 0$ which completes the proof. ☐

b. Show that $p_n = \frac{1}{n^n}$ is superlinearly convergent to 0 but does not converge to 0 of order α for any $\alpha > 1$.

Proof. Part(I) We see that $\lim_{n\to\infty} n^n = \infty$ so $\lim_{n\to\infty} \frac{1}{n^n} = 0$. Now we calculate:

$$\lim_{n\to\infty} \frac{\left|\frac{1}{(n+1)^{n+1}}\right|}{\left|\frac{1}{n^n}\right|} = \lim_{n\to\infty} \frac{n^n}{(n+1)^{n+1}} = \lim_{n\to\infty} \frac{n^n}{(n+1)^n} \cdot \lim_{n\to\infty} \frac{1}{n+1} = \frac{1}{e} \cdot 0 = 0$$

which shows that it is superlinearly convergent.

Part(II) Now to show that it doesn't converge to any order α for $\alpha > 1$:

$$\lim_{n\to\infty} \frac{|n^{\alpha n}|}{|(n+1)^{n+1}|} = \lim_{n\to\infty} \left|\frac{n}{n+1}\right|^n \cdot \lim_{n\to\infty} \frac{n^{\alpha}}{n+1} = \frac{1}{e} \cdot \infty$$

so it does not converge, since we compare the highest power in the denominator and the numerator: $\alpha > 1$ ☐

Exercise 15: Suppose that $\{p_n\}$ is superlinearly convergent to p. Show that

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{|p_n - p|} = 1.$$

Proof. We see that by triangle inequality:

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{|p_n - p|} - \lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} \leqslant \lim_{n\to\infty} \frac{|p_n - p|}{|p_n - p|} = 1$$

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{|p_n - p|} \leqslant 1$$

Another operation:

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{|p_n - p|} \geqslant \lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} + \lim_{n\to\infty} \frac{|p - p_n|}{|p_n - p|}$$

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{p_n - p} \geqslant 1$$

By the two inequalities:

$$\lim_{n\to\infty} \frac{|p_{n+1} - p_n|}{|p_n - p|} = 1$$

☐

Exercise Set 2.6

Exercise 2: Find approximations to within 10^{-5} to all the zeros of each of the following polynomials by first finding the real zeros using Newton's method and then reducing the polynomials of lower degree to determine any complex zeros.

b. $f(x) = x^4 - 2x^3 - 12x^2 + 16x - 40$

Answer. Running Newton's method at $p0 = -5$ gave $x = -3.548232899$ and another iteration at $p0 = 5$ gave $x = 4.381113445$. Run the original polynomial through two divisions:

$$\frac{x^4 - 2x^3 - 12x^2 + 16x - 40}{x - 4.4} \approx x^3 + 2.4x^2 - 1.44x + 9.664$$
$$\frac{x^3 + 2.4x^2 - 1.44x + 9.664}{x + 3.55} \approx x^2 - 1.15x + 2.6425$$

Now solve using the quadratic formula:

$$\begin{aligned} x &= \frac{1.15 \pm \sqrt{(1.15)^2 - 4(2.6425)}}{2} \\ &= \frac{1.15 \pm \sqrt{1.3225 - 10.57}}{2} \\ &= \frac{1.15 \pm \sqrt{-9.2475}}{2} \\ &\approx .575 \pm 1.52i \end{aligned}$$

e. $f(x) = 16x^4 + 88x^3 + 159x^2 + 76x - 240$

Answer. Running Newton's method with $p0 = .5$, I got $x = 0.8467425876$, and for $p0 = -3$, I got $x = -3.358044481$. Then run the polynomial through two divisions:

$$\frac{16x^4 + 88x^3 + 159x^2 + 76x - 240}{x - 0.85} \approx 16x^3 + 101.6x^2 + 245.36x + 284.556$$
$$\frac{16x^3 + 101.6x^2 + 245.36x + 284.556}{x + 3.36} \approx 16x^2 + 47.84x + 84.6176$$

Finally, solve using the quadratic formula:

$$\begin{aligned} x &= \frac{-47.84 \pm \sqrt{(47.84)^2 - 4(16)(84.6176)}}{32} \\ &= \frac{-47.84 \pm \sqrt{2288.6656 - 5415.5264}}{32} \\ &= \frac{-47.84 \pm \sqrt{-3126.8608}}{32} \\ &= \frac{-47.84 \pm 55.9i}{32} \end{aligned}$$

Exercise 4: Repeat Exercise 2 using Muller's method.

b. $f(x) = x^4 - 2x^3 - 12x^2 + 16x - 40$

Answer. Using Muller's method, I got $x = 0.5836 \pm 1.4942i$ as a root for an initial approximation of $p0, p1, p2 = .5, .55, .6$. I received the same real roots compared to Newton's method as well.

Here is the code:

```
function out = muller(f, x, tol)
    out.x = x;
    out.y = arrayfun(f, x);

    while 1
        cf = quad_coeff(f, out.x(end-2:end));
        a = cf.a; b = cf.b; c = cf.c;
        p3 = 0;
        if sign(b) == 1
            p3 = out.x(end) - (2*c) / (b + sqrt(b^2 - 4*a*c));
        else
            p3 = out.x(end) - (2*c) / (b - sqrt(b^2 - 4*a*c));
        end
        out.x = [out.x; vpa(p3)];
        out.y = [out.y; f(p3)];
        if norm(out.y(end)) < tol
            break
        end
    end
end


function y = myfunc6(x)
    y = x^4 - 2*x^3 - 12*x^2 + 16*x - 40;
end


out = muller(@myfunc6, [.5; .55; .6], 1e-5);
x = out.x
y = out.y
out = muller(@myfunc6, [6; 5.5; 5], 1e-5);
x = out.x
y = out.y
out = muller(@myfunc6, [-6; -5; -4], 1e-5);
x = out.x
y = out.y
```

e. $f(x) = 16x^4 + 88x^3 + 159x^2 + 76x - 240$

Answer. Using initial values $p0, p1, p2 = -1.2, -1.3, -1.4$, I got an imaginary solution as $x = -1.4943 \pm 1.7442i$. The real solutions were the same from Newton's method.

Here is the code:

```
function out = muller(f, x, tol)
    out.x = x;
    out.y = arrayfun(f, x);

    while 1
        cf = quad_coeff(f, out.x(end-2:end));
        a = cf.a; b = cf.b; c = cf.c;
        p3 = 0;
        if sign(b) == 1
            p3 = out.x(end) - (2*c) / (b + sqrt(b^2 - 4*a*c));
        else
            p3 = out.x(end) - (2*c) / (b - sqrt(b^2 - 4*a*c));
        end
```

```
            out.x = [out.x; vpa(p3)];
            out.y = [out.y; f(p3)];
            if norm(out.y(end)) < tol
                break
            end
        end
    end

    function y = myfunc7(x)
        y = 16*x^4 + 88*x^3 + 159*x^2 + 76*x - 240;
    end

    out = muller(@myfunc7, [-1; -.5; 0], 1e-5);
    x = out.x
    y = out.y
    out = muller(@myfunc7, [-1; -2; -2.5], 1e-5);
    x = out.x
    y = out.y
    out = muller(@myfunc7, [-1.2; -1.3; -1.4], 1e-5);
    x = out.x
    y = out.y
```

Exercise 7: Use each of the following methods to find a solution in $[0.1, 1]$ accurate to within 10^{-4} for

$$600x^4 - 550x^3 + 200x^2 - 20x - 1 = 0$$

a. Bisection method

Answer. I got $x = 0.2323577881$ in 15 iterations.

b. Newton's method

Answer. I got $x = 0.2323578624$ in 4 iterations with an initial guess of .5.

c. Secant method

Answer. I got $x = 0.2323529651$ in 8 iterations with two initial guesses as $.5, .8$.

e. Muller's method

Answer. I got $x = 0.3600766973 + 0.2654917388i$ in 11 iterations with the three initial guesses as $.3, .4, .5$.

Here is the code:

```
function out = bisection(f, a, b, t)
out.x = [];
while 1
    p = (a + b) / 2;
    out.x = [out.x; vpa(p)];
    if abs(f(p)) < t, break; end
    if f(a) * f(p) > 0
        a = p;
    else
        b = p;
    end
end
```

```
end

function out = newton(f, p0, metric, tol, x_true)
    if ~exist('x_true')
        x_true = 0;
    end
    digits(10)
    x = p0;
    out.x = [vpa(p0)];
    out.p_hat = [];
    out.y = [vpa(f(p0))];
    out.y_hat = [];
    syms c;
    df = matlabFunction(diff(f(c), c));
    i = 1;
    while 1
        if metric(x, x_true, f) < tol
            break;
        end
        x = x - f(x) / df(x);
        out.x = [out.x; vpa(x)];
        i = i + 1;
        if i >= 3
            p_n = vpa(Aitken(out.x(end-2:end)));
            out.p_hat = [out.p_hat; p_n];
            out.y_hat = [out.y_hat; f(p_n)];
        end
        out.y = [out.y; vpa(f(x))];
    end

end

function out = secant(f, p0, p1, tol)
    y0 = f(p0);
    y1 = f(p1);
    out.x = [vpa(p0); vpa(p1)];
    iter = 1;
    while 1
        if abs(y0 - y1) < tol
            break;
        end
        inv_df = (p1 - p0) / (y1 - y0);
        x = out.x(end) - y1 * inv_df;
        out.x = [out.x; vpa(x)];
        p0 = p1;
        p1 = x;
        y0 = y1;
        y1 = f(x);
        iter = iter + 1;
    end
end

function out = muller(f, x, tol)
    out.x = x;
    out.y = arrayfun(f, x);
```

```
    while 1
        cf = quad_coeff(f, out.x(end-2:end));
        a = cf.a; b = cf.b; c = cf.c;
        p3 = 0;
        if sign(b) == 1
            p3 = out.x(end) - (2*c) / (b + sqrt(b^2 - 4*a*c));
        else
            p3 = out.x(end) - (2*c) / (b - sqrt(b^2 - 4*a*c));
        end
        out.x = [out.x; vpa(p3)];
        out.y = [out.y; f(p3)];
        if norm(out.y(end)) < tol
            break
        end
    end
end


tol = 1e-4;
out = bisection(@myfunc8, 0.1, 1, tol);
x = out.x
out = newton(@myfunc8, .5, @out_error, tol);
x = out.x
out = secant(@myfunc8, .5, .8, tol);
x = out.x
out = muller(@myfunc8, [.3; .4; .5], tol);
x = out.x

function y = myfunc8(x)
    y = 600*x^4 - 550*x^3 + 200*x^2 -20*x - 1;
end
```

Exercise 9: A can in the shape of a right circular cylinder is to be constructed to contain 1000 cm^3. The circular top and bottom of the can must have a radius of 0.25 cm more than the radius of the can so that the excess can be used to form a seal with the side. The sheet of material being formed into the side of the can must also be 0.25cm longer than the circumference of the can so that a seal can be formed. Find, to within 10^{-4}, the minimal amount of material needed to construct the can.

Answer. We have an expression giving the amount of material needed:

$$\begin{aligned} \text{material for caps} + \text{material for side} &= \text{material for can} \\ 2\pi(r+0.25)^2 + h(.25 + 2\pi r) &= \text{material for can} \end{aligned}$$

There is also the restriction that

$$\pi r^2 h = 1000$$

or

$$h = \frac{1000}{\pi r^2}$$

Plugging this into the materials equation:

$$2\pi(r+0.25)^2 + \frac{1000}{\pi r^2}(.25 + 2\pi r) = f$$

and to find the minimum, we find the zeros of the derivative of f. I got a radius of $r = 5.363857879$ and the total material needed was $m = 573.6490$.

Here is the code:

```
function out = newton(f, p0, metric, tol, x_true)
    if ~exist('x_true')
        x_true = 0;
    end
    digits(10)
    x = p0;
    out.x = [vpa(p0)];
    out.p_hat = [];
    out.y = [vpa(f(p0))];
    out.y_hat = [];
    syms c;
    df = matlabFunction(diff(f(c), c));
    i = 1;
    while 1
        if metric(x, x_true, f) < tol
            break;
        end
        x = x - f(x) / df(x);
        out.x = [out.x; vpa(x)];
        i = i + 1;
        if i >= 3
            p_n = vpa(Aitken(out.x(end-2:end)));
            out.p_hat = [out.p_hat; p_n];
            out.y_hat = [out.y_hat; f(p_n)];
        end
        out.y = [out.y; vpa(f(x))];
    end

end

syms r;
f(r) = 2*pi*(r + 0.25)^2 + 1000*(.25 + 2*pi*r) / (pi*r^2);
df = matlabFunction(diff(f, r));

out = newton(df, 3, @out_error, 1e-4);
x = out.x(end)
y = double(subs(f, r, x))
```