Problem 1: See attached code

Note: B&F 2.4.2 b has a typo in function $f$.

It should be:

$$f(x) = x^6 + 6x^5 + 9x^4 - 2x^3 - 6x^2 + 1$$

Problem 2

(a) If iteration converges, it converges to a fixed point of the function $F(x)$:

$$x_* = \frac{24x_* + 23/x_*^2}{25}$$

$$\Rightarrow \frac{x_*}{25} = \frac{1}{x_*^2} \Rightarrow x_*^3 = 25$$

The only real root is $x_* = \sqrt[3]{25}$.

(b) We need to verify:

- $|F'(x)| \leq C < 1$ for $x \in [2,3]$\]

$$F'(x) = \frac{24}{25} - \frac{1}{2x^3} = \text{monotonically increasing on } [2,3]$$

$$F'(2) = \frac{24}{25} - \frac{1}{2 \times 8} \approx 0.90$$

$$F'(3) = \frac{24}{25} - \frac{1}{2 \times 27} \approx 0.94$$

$\Rightarrow F$ is a contraction on $[2,3]$.\]
F maps $[2, 3]$ to $[2, 3]$:

Since $F'(x) > 0$ for $x \in [2, 3]$, $F$ must be monotonically increasing on $[2, 3]$ i.e.:

$$[2, 3] \xrightarrow{F} [F(2), F(3)] = [2.17, 2.99] \subseteq [2, 3]$$

Thus by the contractive mapping theorem, $F$ admits a unique fixed point in $[2, 3]$ and fixed point iteration converges for any $x_0 \in [2, 3]$.

(c) Steffensen's method converges faster. See attached code & output.

---

**Problem 3**

$$p(\beta) = 3\beta^5 - 7\beta^4 - 5\beta^3 + 3\beta^2 - 8\beta + 2$$

(a) 

<table>
<thead>
<tr>
<th>3</th>
<th>-7</th>
<th>-5</th>
<th>1</th>
<th>-8</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>60</td>
<td>244</td>
<td>944</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>15</td>
<td>61</td>
<td>236</td>
<td>946</td>
</tr>
</tbody>
</table>

$\Rightarrow \quad p(4) = 946$
(6) Applying Horner's algorithm successively we get:

\[
\begin{array}{cccccc}
3 & -7 & -5 & 1 & -8 & 2 \\
4 & & & & & \\
5 & 3 & 12 & 20 & 60 & 244 \\
4 & & & & & \\
5 & 3 & 12 & 68 & 332 & 1572 \\
4 & & & & & \\
5 & 3 & 12 & 116 & 796 & 1808 \\
4 & & & & & \\
5 & 3 & 12 & 199 & 363 & 1189 \\
4 & & & & & \\
5 & 3 & 12 & 363 & & \\
\end{array}
\]

Thus:
\[
P(x) = 946 + 1808(x - 4) + 1189(x - 4)^2 \\
+ 363(x - 4)^3 + 53(x - 4)^4 + 3(x - 4)^5
\]

(c) \( \delta_1 = 30 - \frac{P(30)}{P'(30)} \) (Newton update)

From table above: \( P(30) = 946, P'(30) = 1808 \)

Thus:
\[
\delta_1 = 4 - \frac{946}{1808} = \frac{4 \times 904 - 473}{904} = \frac{3143}{904} \approx 3.477
\]
Problem 4: See attached code & output.

Problem 5

(a) If \( z_\star \) is a root of multiplicity \( m \) of \( p(z) \) then:

\[
p(z) = (z - z_\star)^m q(z), \text{ with } q(z_\star) \neq 0.
\]

Thus

\[
p^{(n)}(z) = \sum_{k=0}^{m} \binom{n}{k} (z - z_\star)^{m-k} q^{(n-k)}(z)
\]

Now note that:

\[
((z - z_\star)^m)^{(k)} = \begin{cases} 
\frac{m!}{(m-k)!} (z - z_\star)^{m-k} & \text{if } 0 \leq k < m \\
\frac{m!}{m!} & \text{if } k = m \\
0 & \text{if } k > m
\end{cases}
\]

Thus:

\[
p^{(n)}(z_\star) = 0 \quad \text{if } 0 \leq n \leq m-1
\]

but

\[
p^{(m)}(z_\star) = m! q(z_\star), \text{ w/ } q(z_\star) \neq 0.
\]

\( \square \)
(b) Assume \( p \) is a poly for which:

\[
P(z^*) = p'(z^*) = \cdots = p^{(m-1)}(z^*) = 0
\]

but \( p^{(m)}(z^*) \neq 0 \).

Thus its Taylor expansion about \( z = z^* \) is:

\[
p(z) = \sum_{k=0}^{\infty} \frac{p^{(k)}(z^*)}{k!} (z-z^*)^k
\]

where \( n = \) degree of poly \( p \).

By hypothesis:

\[
p(z) = \frac{p^{(m)}(z^*)}{m!} (z-z^*)^m + \sum_{k=m+1}^{\infty} \frac{p^{(k)}(z^*)}{k!} (z-z^*)^k
\]

\[
= (z-z^*)^m \left[ \frac{p^{(m)}(z^*)}{m!} + \sum_{k=1}^{n-m} \frac{p^{(k+m)}(z^*)}{(k+m)!} (z-z^*)^k \right]
\]

\[
= q(z)
\]

\( q(z) \) is a poly which does not vanish at \( z = z^* \) since:

\[
q(z^*) = \frac{p^{(m)}(z^*)}{m!}
\]

\( \therefore \)
>> prob1
Part a: NM = 0.73907868 in 17 iter MNM = 0.73908513 in 6 iter
Part b: NM = -2.87939396 in 15 iter MNM = -2.87938523 in 5 iter

>> prob2
fixed point Steffensens
root 2.92403530 2.92401774
iter 14 13
g evals 13 8
true value: (25)^{1/3} = 2.92401774

>> prob3
--- part a ---
found root r1 = 4.123105626 in 5 iterations
found root r2 = -4.123105626 in 8 iterations
deflated polynomial is:
1.000000000000000   4.999999999999526   7.999999999999584
with roots:
-2.499999999999763 + 1.322875655532586i
-2.499999999999763 - 1.322875655532586i
the roots according to Matlab are:
-2.499999999999999 - 1.322875655532298i
-2.499999999999999 + 1.322875655532298i
-4.123105625617663
 4.123105625617665

--- part h ---
found root r1 = 0.5857864376 in 5 iterations
found root r2 = 3.414213562 in 6 iterations
found root r3 = 3 in 5 iterations
there are no imaginary roots
the roots according to Matlab are:
 0.585786437626905
 2.999999999999985
 3.414213562373112
%% HW 3 Problem 1
%% B&F 2.4.2 and 2.4.4
tol = 1e-5; maxit = 20;

%% Part A
x0 = 0; % initial guess
f = @(x) 1 - 4*x*cos(x) + 2*x^2 + cos(2*x);
fp = @(x) 4*x - 2*sin(2*x) - 4*cos(x) + 4*x*sin(x);
fpfp = @(x) 8*sin(2*x) - 4*cos(2*x) + 4*x*cos(x) + 4;
xs1 = newton(f,fp,x0,tol,maxit);
xs2 = newton_mult(f,fp,fpfp,x0,tol,maxit);
fprintf('Part a: NM=%13.8f in %4d iter MNM=%13.8f in %4d iter
','
xs1(end), length(xs1), xs2(end), length(xs2));

%% Part B
x0 = -3; % initial guess
% NOTE: the function below has a typo in the book
% the first term in the sum should be x^6 and
% not x^2
f = @(x) x^6 + 6*x^5 + 9*x^4 - 2*x^3 - 6*x^2 + 1;
fp = @(x) 6*x^5 + 30*x^4 + 36*x^3 - 6*x^2 - 12*x;
fpfp = @(x) 30*x^4 + 120*x^3 + 108*x^2 - 12*x - 12;
xs1 = newton(f,fp,x0,tol,maxit);
xs2 = newton_mult(f,fp,fpfp,x0,tol,maxit);
fprintf('Part b: NM=%13.8f in %4d iter MNM=%13.8f in %4d iter
','
xs1(end), length(xs1), xs2(end), length(xs2));

%% HW 3 Problem 2

$\text{cube root of 25 (or } 25^{(1/3)}) \text{ can be approximated by the following fixed point}$
$\text{iteration}$
g = @(x) (x + 25/x^2)/2;

x0 = 5; % initial guess
maxit = 50; % max number of iterations
tol = 1e-4; % tolerance

xs1 = fixedpoint(g,x0,maxit,tol);
xs2 = steffensen(g,x0,maxit,tol);
fprintf('%10s %13s %13s
',' ','fixed point','Steffensens');
fprintf('%10s %13.8f %13.8f
','root',xs1(end),xs2(end));
fprintf('%10s %13d %13d
','iter', length(xs1), length(xs2));
fprintf('%10s %13d %13d
','g evals', length(xs1) - 1, 2*(length(xs2) - 1)/3);
% for Steffensen's 2 out of 3 iterations are function evaluations.
fprintf('true value: (25)^{(1/3)} = %13.8f
',(25)^(1/3));
% HW 3 Problem 3

% try starting at z0 = 5
z0=5;
[r1,k]=polynm(p,z0,tol,maxit);
fprintf('found root r1 = %13.10g in %d iterations
',r1,k);

% try starting at z0 = -1
z0=-1;
[r2,k]=polynm(p,z0,tol,maxit);
fprintf('found root r2 = %13.10g in %d iterations
',r2,k);  

% deflate polynomial
[q1,r] = horner(p,r1);
[q2,r] = horner(q1,r2);
fprintf('deflated polynomial is:
');
disp(q2)
fprintf('with roots:
');
disp([-q2(2)+sqrt(del))/2/q2(1);(-q2(2)-sqrt(del))/2/q2(1)])

% compare to the roots that Matlab finds:
fprintf('the roots according to Matlab are:
');
disp(sort(roots(p)));

% try starting at z0 = 0
z0 = 0;
[r1,k]=polynm(p,z0,tol,maxit);
fprintf('found root r1 = %13.10g in %d iterations
',r1,k);

% try starting at z0 = 4
z0 = 4;
[r2,k]=polynm(p,z0,tol,maxit);
fprintf('found root r2 = %13.10g in %d iterations
',r2,k);

% try starting at z0 = 2
z0 = 2.1;
[r3,k]=polynm(p,z0,tol,maxit);
fprintf('found root r3 = %13.10g in %d iterations
',r3,k);

fprintf('there are no imaginary roots
');

% compare to the roots that Matlab finds:
fprintf('the roots according to Matlab are:
');
disp(sort(roots(p)));
Newton method to find a root of a function $f$ starting at $x_0$.

**Inputs**
- $f$: function we want to find the root of
- $fp$: derivative of $f$
- $fpp$: second derivative of $f$
- $x_0$: initial iterate
- $tol$: desired tolerance or precision
- $maxit$: maximum number of iterations

**Outputs**
- $xs$: iteration history, last iterate is $xs(end)$

```matlab
function xs = newton(f,fp,fpp,x0,tol,maxit)
    xs = [x0]; % store iterates
    x = x0;
    for k=1:maxit,
        xnew = x - f(x)*fp(x)/((fp(x))^2 - f(x)*fpp(x));
        xs = [xs xnew]; % store iterates
        if abs(xnew-x)<tol break; end; % check convergence
        x = xnew; % prepare next iteration
    end;
end
```

Fixed point iteration

**Inputs**
- $g$: contraction (function taking a real argument and returning a real number)
- $x_0$: initial iterate
- $maxit$: maximum number of iterations
- $tol$: tolerance for two consecutive iterates

**Outputs**
- $xs$: vector containing the iterates

```matlab
function xs = fixedpoint(g,x0,maxit,tol)
    x = x0;
    xs = x;
    for k = 1:maxit,
        xnew = g(x);
        xs = [xs xnew]; % store iterate
        if abs(xnew-x)<tol break; end; % check convergence
        x = xnew; % prepare next iteration
    end;
end
```
Steffensen's method to accelerate convergence of certain fixed point iterations

Inputs:
- \( g \) contraction (function taking a real argument and returning a real number)
- \( x_0 \) initial iterate
- \( \text{maxit} \) maximum number of iterations
- \( \text{tol} \) tolerance for two consecutive iterates

Outputs:
- \( \text{xs} \) vector containing the iterates

```matlab
function \( \text{xs} = \text{steffesnse}(g,x_0,\text{maxit},\text{tol}) \)
x_0 = \text{xs}; % store iterate
k=1;
for k = 1:maxit,
x1 = g(x0);
x2 = g(x1);
x3 = x0 - (x1-x0)^2/(x2-2*x1+x0);
x = [x x1 x2 x3]; % store iterates
if abs(x3 -x0)<\text{tol} break; end; % check convergence
x0 = x3; % prepare next iteration
end;
```

Newton's method using Horner's algorithm to evaluate a polynomial and its derivative efficiently

Inputs:
- \( p \) Vector of polynomial coefficients, from highest to lowest degree
- \( z_0 \) initial guess
- \( \text{tol} \) tolerance
- \( \text{maxit} \) maximum number of iterations

Outputs:
- \( z \) last iterate
- \( k \) number of iterations

```matlab
function \( [z,k] = \text{polynm}(p,z_0,\text{tol},\text{maxit}) \)
for k=1:maxit,
[alpha,beta] = \text{hornermn}(p,z_0);
z = z_0 - alpha/beta;
if abs(z-z0)<\text{tol} break; end;
end;
```
% Computes value of a polynomial and its derivative at a point
% using Horner's algorithm twice.
% Inputs
% p  vector of polynomial coefficients, from highest to lowest degree
% z0  evaluation point
% Outputs
% alpha  p(z0)
% beta   p'(z0)
function [alpha, beta] = hornernm(p, z0)
    n = length(p);
    alpha = p(1);
    beta = 0;
    for k=1:n-1,
        beta = alpha + z0*beta;
        alpha = p(k+1)+z0*alpha;
    end
end

function [q, r] = horner(p, z0)
    n = length(p);
    q = zeros(size(p));
    q(1) = p(1);
    for k=1:n-1,
        q(k+1) = p(k+1) + z0 * q(k);
    end;
    r = q(n);
    q = q(1:n-1);