1 Code Listings

Listing 1: a7.m.

```
function_names = ["Rosenbrock", "f(x) (n = 10)", "f(x) (n = 100)"];
functions = {f_rosenbrock() a7_function(10) a7_function(100)};
x0s = {[-1.2; 1] a7_function_x0(10) a7_function_x0(100)};

epsilon = 1e-6;

for i = 1:1
    f = functions{i};

    x0 = x0s{i};
    [x_k, k] = newton_inexact(f, x0, epsilon);

    disp(function_names(i));
    disp("Solution x:");
    disp(x_k);
    disp("Solution f(x):");
    disp("f(x) = " + f(x_k));
    disp("Number of iterations:");
    disp(k);
    disp("-----------------");
end
```

Listing 2: f_rosenbrock.m.

```
function f_rosenbrock = f_rosenbrock()
f_rosenbrock = @(x) 100 * (x(2) - x(1)^2)^2 + (1 - x(1))^2;
end
```

Listing 3: a7_function.m.

```
function f = a7_function(n)
f = @(x) 0;
for i = 1:n
    F_i = @(x) x(i) - 1;
    f = @(x) f(x) + F_i(x) ^ 2;
end
F_n1 = @(x) 0;
for j = 1:n
    F_n1 = @(x) F_n1(x) + j * (x(j) - 1);
end

f = @(x) f(x) + F_n1(x) ^ 2;
F_n2 = @(x) F_n1(x)^2;
f = @(x) f(x) + F_n2(x)^ 2;

end
```

Listing 4: a7_function_x0.m.

```
function x0 = a7_function_x0(n)
x0 = zeros(n, 1);
for i = 1:n
    x0(i) = 1 - i / n;
end
end
```

Listing 5: newton_inexact.m.

```
function [x, k] = newton_inexact(f, x0, epsilon)
beta = 0.5;
sigma = 1e-4;
rho = 1e-8;
p = 2.1;
c1 = 1e-2;
c2 = 1;

k = 0;
x = x0;
grad_val = gradest(f, x).';
n = norm(grad_val);
while n > epsilon
    eta = min([c1 / (k + 1), c2 * n]);

    % CG method (to solve inexact Newton equation)
    d = cg(hessian(f, x), -grad_val, -grad_val, eta * n);

    if grad_val' * d > - rho * (norm(d)^p)
        d = -grad_val;
    end

    % Armijo update
    t = armijo(f, x, sigma, grad_val, d, beta);

    x = x + t*d;
    k = k + 1;
    grad_val = gradest(f, x).';
    n = norm(grad_val);
end
end
```

Listing 6: armijo.m.

```
function t = armijo(f, x_k, sigma, grad_val, d, beta)
l = 0;
t = 1;
f_val = f(x_k);
rhs = sigma * grad_val.' * d;
while f(x_k + t*d) > f_val + t*rhs
    l = l + 1;
    t = beta ^ l;
end
end
```

Listing 7: cg.m.

```
function x = cg(A, x0, b, epsilon)
x = x0;
g = A * x - b;
d_cg = -g;
while norm(g) > epsilon
    g_norm_2 = norm(g)^2;
    t_cg = g_norm_2 / (d_cg' * A * d_cg);

    x = x + t_cg * d_cg;
    g = g + t_cg * A * d_cg;
    beta_cg = norm(g)^2 / g_norm_2;
    d_cg = -g + beta_cg * d_cg;
end
end
```