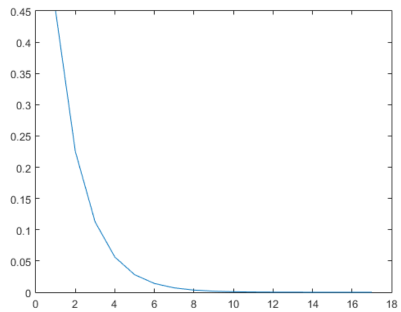
# MA207 Numerical Methods - Assignment

14CO255 - Mohammed Khursheed Ali Khan

April 2018

## 1 Problem 1

```
% defining the function
fs = '-32.17/(2*x^2)*((exp(x)-exp(-x))/2 -sin(x))-1.7';

% defining the error, TOL
a = -1;
b = -0.1;
TOL = 1e-5;
error = inf;

% applying bisection
f = inline(fs);
A = f(a);
iter = 1;
while ( error > TOL )
    c = (a+b)/2;
    C = f(c);
    error(iter) = (b-a)/2;

    if( A*C > 0 )
        a = c;
        A = C;
    else
        b = c;
    end

    iter = iter+1;
end
```



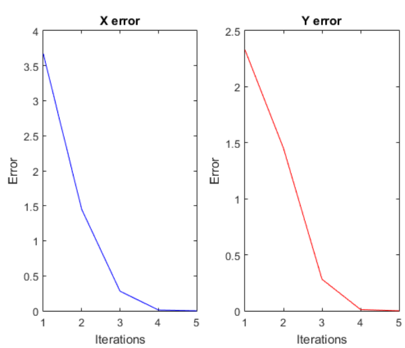

**Solution**: -0.3171

# 2 Problem 2

```
% initiallize x, y as syms
syms x y;

% system of equations to solve
f1(x, y) = x + y - 20;
f2(x, y) = (x + x^0.5)*(y + y^0.5) - 155.55;

% derivative of f1, f2 w.r.t x and y
f1x(x, y) = diff(f1, x);
f1y(x, y) = diff(f1, y);

f2x(x, y) = diff(f2, x);
f2y(x, y) = diff(f2, y);

% Solving Ax = b
x_i = 4;
y_i = 10;
count = 0;
TOL = 1e-4;
x_j = x_i;
y_j = y_i;
while true
    A = [subs(f1x, [x, y], [x_j, y_j]) subs(f1y, [x, y], [x_j, y_j]); ...
         subs(f2x, [x, y], [x_j, y_j]) subs(f2y, [x, y], [x_j, y_j])];
    A = eval(A);
```

```
    b = [-subs(f1, [x, y], [x_j, y_j]); -subs(f2, [x, y], [x_j, y_j])];
    b = eval(b);
    h = A\b;
    x_i = x_j;
    y_i = y_j;
    x_j = x_i + h(1);
    y_j = y_i + h(2);
    count = count + 1;
    errorx(count) = abs(x_j - x_i);
    errory(count) = abs(y_j - y_i);
    if abs(x_j - x_i) < TOL && abs(y_j - y_i) < TOL, break ; end
end

% plotting charts
figure
subplot(1, 2, 1)
plot(errorx, 'color','b');
xlabel('Iterations');
ylabel('Error');
title('X error')

subplot(1, 2, 2);
plot(errory, 'color', 'r');
xlabel('Iterations');
ylabel('Error');
title('Y error');
```

**Solution**: x = 6.5128 and y = 13.4872



# 3 Problem 3

```
% initialize x and y as symbolic variables
syms y ;

% function to integrate, we are integrating the first quadrant only
f(y) = ((36 - 4*y^2)/9)^0.5;
l(y) = (1 + diff(f)^2)^0.5;

% calculate n and h
b = 3
a = 0
h = sqrt((12 * 1e-6) / ((b-a) * subs(diff(l, 2), 0)));
h = eval(h)
n = (b-a) / h;
n = ceil(n);

I = 0;

% calculating the points
for i = 1:n+1
    x(i) = a + (i-1)*h;
end

% calculating the values
for i = 1:n+1
    y(i) = vpa(subs(l, x(i)));
end

% length using trapezoidal
for i = 1:n+1
    if ( i == 1 || i == n+1)
        I = I + y(i)./2;
    else
        I = I + y(i);
    end
end

lengtht = vpa(I * h);

% length of the whole ellipse is 4 * length
lengtht = 4 * lengtht

% length using simpson 1/3
I = 0
```

```
for i = 1:n+1
    if ( i == 1 || i == n+1)
        I = I + y(i)./3;
    elseif (mod(i, 2) == 0)
        I = I + y(i).*(2/3);
    else
        I = I + y(i).*(4/3);
    end
end

lengths = vpa(I * h);

% length of the whole ellipse is 4 * length
lengths = 4 * lengths
```

**Solution**: trapezoidal length = 15.69073687 and simpson length = 15.83011200

# 4 Problem 4

```
% system of equations to evaluate, ensure diagonall eements are non zero
A = [-1 0 0 sqrt(2)/2 1 0 0 0; ...
      0 -1 0 sqrt(2)/2 0 0 0 0; ...
      0 0 -1 0 0 0 1/2 0; ...
      0 0 0 -sqrt(2)/2 0 -1 -1/2 0; ...
      0 0 0 0 -1 0 0 1; ...
      0 0 0 0 0 1 0 0; ...
      0 0 0 -sqrt(2)/2 0 0 sqrt(3)/2 0; ...
      0 0 0 0 0 0 -sqrt(3)/2 -1] ;

b = [0;
    0;
    0;
    0;
    0;
    10000;
    0;
    0];

U = triu(A, 1) .* -1;
```

```
L = tril(A, -1) .* -1;
D = diag(diag(A));
TOL = 1e-3;

% Jacobi
G = D\(L + U);
c = D\b;
error = inf;
x_i = b;
x = x_i;
iter = 1;
while error > TOL
    x_i = x;
    x = G * x_i + c;
    error(iter) = abs(norm(x, inf) - norm(x_i, inf));
    iter = iter + 1;
end

% Gaauss - Siedel
G = (D - L)\U;
c = (D - L)\b;
error = inf;
x_i = ones(8, 1);
x = x_i;
iter = 1;
while error > TOL
    x_i = x;
    x = G * x_i + c;
    error(iter) = abs(norm(x, inf) - norm(x_i, inf));
    iter = iter + 1;
end

% SOR
w = 1.25;
G = (D - w*L)\((1 - w)*D + w*U);
c = w * ((D - w*L)\b);
error = inf;
x_i = ones(8, 1);
x = x_i;
iter = 1
while error > TOL
    x_i = x;
    x = G * x_i + c;
    error(iter) = abs(norm(x, inf) - norm(x_i, inf));
    iter = iter + 1;
end
```

**Solution**:
Jacobi
1.0e+04 * [-1.0000 -1.0000 0 -1.4142 0 1.0000 -1.1547 0]

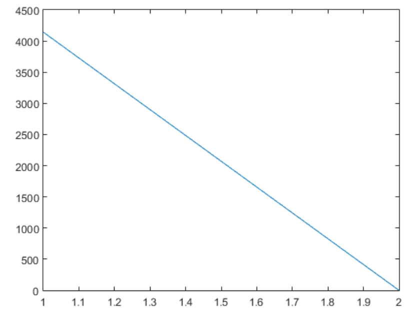



Figure 1: Convergence of Jacobi

Gauss Siedel
1.0e+04 * [0.1924 -0.5635 -0.3254 -0.9541 0.5635 1.0000 -0.7790 0.6746]

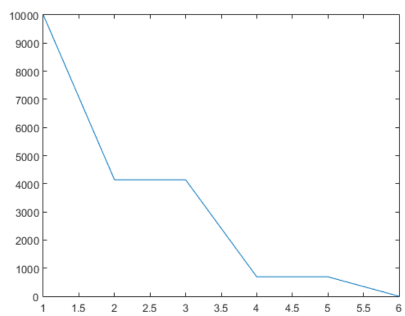



Figure 2: Convergence of Gauss-Siedel

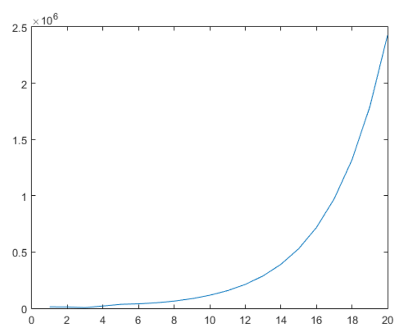



Figure 3: Divergence of SOR