

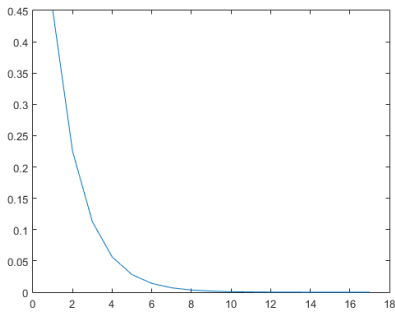
# MA207 Numerical Methods - Assignment

14CO255 - Mohammed Khursheed Ali Khan

April 2018

## 1 Problem 1

```
1 % defining the function
2 fs = '-32.17/(2*x^2)*((exp(x)-exp(-x))/2)-sin(x)-1.7';
3
4 % defining the error, TOL
5 a = -1;
6 b = -0.1;
7 TOL = 1e-5;
8 error = inf;
9
10 % applying bisection
11 f = inline(fs);
12 A = f(a);
13 iter = 1;
14 while ( error > TOL )
15     c = (a+b)/2;
16     C = f(c);
17     error(iter) = (b-a)/2;
18
19     if( A*C > 0 )
20         a = c;
21         A = C;
22     else
23         b = c;
24     end
25
26     iter = iter+1;
27 end
```



**Solution:** -0.3171

## 2 Problem 2

```

1 % initialize x, y as syms
2 syms x y;
3
4 % system of equations to solve
5 f1(x, y) = x + y - 20;
6 f2(x, y) = (x + x^0.5)*(y + y^0.5) - 155.55;
7
8 % derivative of f1, f2 w.r.t x and y
9 f1x(x, y) = diff(f1, x);
10 f1y(x, y) = diff(f1, y);
11
12 f2x(x, y) = diff(f2, x);
13 f2y(x, y) = diff(f2, y);
14
15 % Solving Ax = b
16 x_i = 4;
17 y_i = 10;
18 count = 0;
19 TOL = 1e-4;
20 x_j = x_i;
21 y_j = y_i;
22 while true
23     A = [subs(f1x, [x, y], [x_j, y_j]) subs(f1y, [x, y], [x_j, y_j]); ...
24         subs(f2x, [x, y], [x_j, y_j]) subs(f2y, [x, y], [x_j, y_j])];
25     A = eval(A);

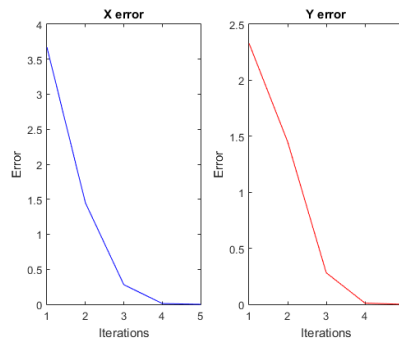
```

```

26     b = [-subs(f1, [x, y], [x_j, y_j]); -subs(f2, [x, y], [x_j, y_j])];
27     b = eval(b);
28     h = A\b;
29     x_i = x_j;
30     y_i = y_j;
31     x_j = x_i + h(1);
32     y_j = y_i + h(2);
33     count = count + 1;
34     errorx(count) = abs(x_j - x_i);
35     errory(count) = abs(y_j - y_i);
36     if abs(x_j - x_i) < TOL && abs(y_j - y_i) < TOL, break ; end
37 end
38
39 % plotting charts
40 figure
41 subplot(1, 2, 1)
42 plot(errorx, 'color', 'b');
43 xlabel('Iterations');
44 ylabel('Error');
45 title('X_error')
46
47 subplot(1, 2, 2);
48 plot(errory, 'color', 'r');
49 xlabel('Iterations');
50 ylabel('Error');
51 title('Y_error');

```

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



### 3 Problem 3

```

1 % initialize x and y as symbolic variables
2 syms y ;
3
4 % function to integrate , we are integrating the first quadrant only
5 f(y) = ((36 - 4*y^2)/9)^0.5;
6 l(y) = (1 + diff(f)^2)^0.5;
7
8 % calculate n and h
9 b = 3
10 a = 0
11 h = sqrt((12 * 1e-6) / ((b-a) * subs(diff(l, 2), 0)));
12 h = eval(h)
13 n = (b-a) / h;
14 n = ceil(n);
15
16 I = 0;
17
18 % calculating the points
19 for i = 1:n+1
20     x(i) = a + (i-1)*h;
21 end
22
23 % calculating the values
24 for i = 1:n+1
25     y(i) = vpa(subs(l, x(i)));
26 end
27
28 % length using trapezoidal
29 for i = 1:n+1
30     if ( i == 1 || i == n+1)
31         I = I + y(i)./2;
32     else
33         I = I + y(i);
34     end
35 end
36
37 lengtht = vpa(I * h);
38
39 % length of the whole ellipse is 4 * length
40 lengtht = 4 * lengtht
41
42 % length using simpson 1/3
43 I = 0

```

```

44 for i = 1:n+1
45     if ( i == 1 || i == n+1)
46         I = I + y(i)./3;
47     elseif (mod(i, 2) == 0)
48         I = I + y(i).*(2/3);
49     else
50         I = I + y(i).*(4/3);
51     end
52 end
53
54 lengths = vpa(I * h);
55
56 % length of the whole ellipse is 4 * length
57 lengths = 4 * lengths

```

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

## 4 Problem 4

```

1 % system of equations to evaluate, ensure diagonall eements are non zero
2 A = [-1 0 0 sqrt(2)/2 1 0 0 0; ...
3      0 -1 0 sqrt(2)/2 0 0 0 0; ...
4      0 0 -1 0 0 0 1/2 0; ...
5      0 0 0 -sqrt(2)/2 0 -1 -1/2 0; ...
6      0 0 0 0 -1 0 0 1; ...
7      0 0 0 0 0 1 0 0; ...
8      0 0 0 -sqrt(2)/2 0 0 sqrt(3)/2 0; ...
9      0 0 0 0 0 0 -sqrt(3)/2 -1] ;
10
11 b = [0;
12      0;
13      0;
14      0;
15      0;
16      10000;
17      0;
18      0];
19
20 U = triu(A, 1) .* -1;

```

```

21 L = tril(A, -1) .* -1;
22 D = diag(diag(A));
23 TOL = 1e-3;
24
25 % Jacobi
26 G = D \ (L + U);
27 c = D \ b;
28 error = inf;
29 x_i = b;
30 x = x_i;
31 iter = 1;
32 while error > TOL
33     x_i = x;
34     x = G * x_i + c;
35     error(iter) = abs(norm(x, inf) - norm(x_i, inf));
36     iter = iter + 1;
37 end
38
39 % Gauss - Siedel
40 G = (D - L) \ U;
41 c = (D - L) \ b;
42 error = inf;
43 x_i = ones(8, 1);
44 x = x_i;
45 iter = 1;
46 while error > TOL
47     x_i = x;
48     x = G * x_i + c;
49     error(iter) = abs(norm(x, inf) - norm(x_i, inf));
50     iter = iter + 1;
51 end
52
53 % SOR
54 w = 1.25;
55 G = (D - w*L) \ ((1 - w)*D + w*U);
56 c = w * ((D - w*L) \ b);
57 error = inf;
58 x_i = ones(8, 1);
59 x = x_i;
60 iter = 1;
61 while error > TOL
62     x_i = x;
63     x = G * x_i + c;
64     error(iter) = abs(norm(x, inf) - norm(x_i, inf));
65     iter = iter + 1;
66 end

```

**Solution:**

Jacobi

$1.0e+04 * [-1.0000 \ -1.0000 \ 0 \ -1.4142 \ 0 \ 1.0000 \ -1.1547 \ 0]$

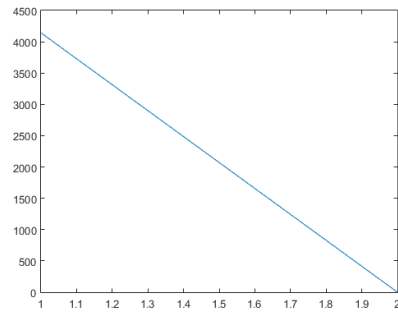


Figure 1: Convergence of Jacobi

Gauss Siedel

$1.0e+04 * [0.1924 \ -0.5635 \ -0.3254 \ -0.9541 \ 0.5635 \ 1.0000 \ -0.7790 \ 0.6746]$

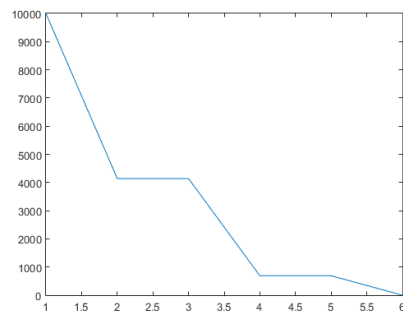


Figure 2: Convergence of Gauss-Siedel

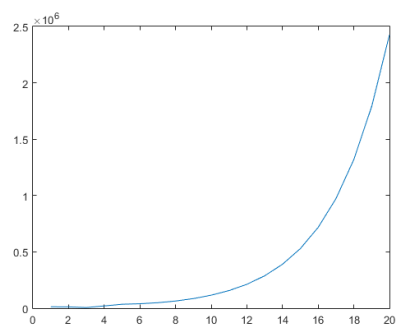


Figure 3: Divergence of SOR