

## 1 Introduction

When integrating functions of one or more variables, analytic solutions are desired because they offer a complete description of solution behavior. For most situations of engineering interest, however, the governing equations defy all attempts to obtain a closed solution. In such instances, numerical integration provides a tool for exploring the solution space.

We solve the following problems to better understand the tools of numerical integration. As will be described in the methods section, our tools include the trapezoidal method, Simpson's method, and Richardson extrapolation.

### 1.1 Problem A

We wish to show that the following equality holds for  $a = 1/2$ :

$$\int_0^{2\pi} \frac{dx}{1 + a \cos x} = \frac{2\pi}{\sqrt{1 - a^2}}, \quad a^2 < 1. \quad (1)$$

This is a relatively simple finite integral of a continuous function with no singularities.

### 1.2 Problem B

We wish to confirm the equalities

$$(i) \quad \int_0^\infty \frac{\sin^4 x}{x^4} dx = \frac{\pi}{3}, \quad \text{and} \quad (ii) \quad \int_0^\infty \frac{\sin x}{\sqrt{x}} dx = \sqrt{\frac{\pi}{2}}. \quad (2)$$

Note that both integrands are indeterminate at  $x = 0$  and otherwise periodic, and that both integrals extend indefinitely. Particular care will need to be paid to singularities and periodicity.

### 1.3 Problem C

Moving to functions of two variables, we wish to estimate

$$I = \int_1^3 \int_1^2 f(x, y) dx dy \quad (3)$$

for the integrands

$$(i) \quad f(x, y) = xy(1 + x), \quad \text{and} \quad (ii) \quad f(x, y) = x^2 y^3(1 + x). \quad (4)$$

Like the first problem, the expressions are fairly simple, involving well-behaved integrands and finite bounds. Application of our chosen numerical method over a two-dimensional domain will be the crux of this problem.

## 2 Methodology

## 3 Results

## 4 Discussion

## 5 References

### Appendix: MATLAB Code

The following code listings generate all figures presented in this homework assignment.

**Listing 1: Homework\_1\_Driver.m**

```
1 function [] = Homework_1_Driver()
2 %%%
3 % HOMEWORK 1 DRIVER
4 %
5 % Run this file to calculate output for Homework 1.
6 %%%
7
8 Set_Default_Plot_Properties();
9
10 Blanks();
11
12 Problem('A');
13 Problem_A();
14 Blanks();
15
16 Problem('B');
17 Problem_B();
18 Blanks();
19
20 Problem('C');
21 Problem_C();
22 Blanks();
23
24 end
25
26 function [] = Problem( name )
27     fprintf('*****\n');
28     fprintf('*****>> Problem %s <<*****\n', name);
29     fprintf('*****\n');
30 end
31
32 function [] = Blanks()
33     fprintf('\n\n');
34 end
```

**Listing 2: RelErr.m**

```
1 function [ re ] = RelErr( fe, fc )
2     re = abs(fe - fc) ./ fe;
3 end
```

**Listing 3: Problem\_A.m**

```
1 function [] = Problem_A()
2
3     % Define the integrand with a function handle.
4     a = 1/2;
5     f = @(x) 1 / (1 + a*cos(x));
6
7     % Calculate pi, but name it pii since MATLAB has already defined the variable pi.
```

```

8      pii = 4 * atan(1);
9      fprintf('pi:  %28.20e\n', pii);
10
11     Dashes();
12
13     % Calculate the RHS.
14     rhs = 2 * pii / sqrt(1 - a^2);
15     fprintf('RHS: %28.20e\n', rhs);
16
17     Dashes();
18
19     % Calculate the LHS with n = {10, 100};
20
21     v10 = Simpsons(f, [0,2*pii], 10);
22     v100 = Simpsons(f, [0,2*pii], 100);
23
24     fprintf('LHS: %28.20e with n=10\n', v10);
25     fprintf('LHS: %28.20e with n=100\n', v100);
26
27     re10 = RelErr(v10, rhs);
28     re100 = RelErr(v100, rhs);
29     fprintf('re:  %28.20e with n=10\n', re10);
30     fprintf('re:  %28.20e with n=100\n', re100);
31
32     Dashes();
33
34     % Double the grid points, and re-calculate the LHS.
35
36     v20 = Simpsons(f, [0,2*pii], 20);
37     v200 = Simpsons(f, [0,2*pii], 200);
38
39     fprintf('LHS: %28.20e with n=20\n', v20);
40     fprintf('LHS: %28.20e with n=200\n', v200);
41
42     re20 = RelErr(v20, rhs);
43     re200 = RelErr(v200, rhs);
44     fprintf('re:  %28.20e with n=20\n', re20);
45     fprintf('re:  %28.20e with n=200\n', re200);
46
47     Dashes();
48
49     % Use Richardson extrapolation on each set.
50
51     ve1 = v20 + (v20 - v10) / 15;
52     ve2 = v200 + (v200 - v100) / 15;
53     fprintf('LHS: %28.20e with n={10, 20} (Richardson)\n', ve1);
54     fprintf('LHS: %28.20e with n={100,200} (Richardson)\n', ve2);
55
56     reR1 = RelErr(ve1, rhs);
57     reR2 = RelErr(ve2, rhs);
58     fprintf('re:  %28.20e with n={10, 20} (Richardson)\n', reR1);
59     fprintf('re:  %28.20e with n={100,200} (Richardson)\n', reR2);
60
61 end

```

#### Listing 4: Simpsons.m

```

1  function [ F ] = Simpsons( f, bounds, n_slabs )
2
3      n_pts = 1 + 2 * n_slabs;
4      x = linspace(bounds(1), bounds(2), n_pts);
5      h = x(2) - x(1);
6
7      y = zeros(1,n_pts);
8      for i = 1:n_pts
9          y(i) = feval(f, x(i));
10     end
11

```

```

12     F = 0;
13     for i = 1:n_slabs
14         % Determine indices involved in Simpson's rule for this slab.
15         ii = 1 + 2*(i-1)*ones(1,3) + [0,1,2];
16         F = F + y(ii(1)) + 4 * y(ii(2)) + y(ii(3));
17     end
18     F = F * h/3;
19
20 end

```

### Listing 5: Problem\_B.m

```

1 function [] = Problem_B()
2
3     % Calculate pi, but name it pii since MATLAB has already defined the variable pi.
4     pii = 4 * atan(1);
5
6     %%%%%%%%%%%
7     % Part (i) %
8     %%%%%%%%%%%
9
10    % Calculate the RHS.
11    rhs = pii / 3;
12    fprintf('RHS: %28.20e\n', rhs);
13
14    % Define the integrand with a function handle.
15    f = @(x) sin(x)^4 / x^4;
16
17    figure();
18    x = linspace(0,25);
19    y = zeros(0,length(x));
20    for i = 1:length(x)
21        y(i) = f(x(i));
22    end
23    plot(x,y);
24    xlabel('x');
25    ylabel('y');
26    legend('sin(x)^4 / x^4');
27
28    min_x = 0;
29    slabspp = 4;
30    y0 = 1;
31
32    pers = (1:9)' * 10.^(0:3);
33    pers = reshape(pers, 1, numel(pers));
34    F1 = zeros(1, length(pers));
35    F2 = zeros(1, length(pers));
36    Fe = zeros(1, length(pers));
37
38    for i = 1:length(pers)
39
40        per = pers(i);
41
42        F1(i) = Trapezoidal_Inf(f, min_x, slabspp, 2*pi, per, y0);
43        F2(i) = Trapezoidal_Inf(f, min_x, slabspp*2, 2*pi, per, y0);
44        Fe(i) = F2(i) + (F2(i) - F1(i)) / 3;
45
46    end
47
48    re1 = RelErr(F1, rhs);
49    re2 = RelErr(F2, rhs);
50    ree = RelErr(Fe, rhs);
51
52    figure();
53    loglog(pers, re1, pers, re2, pers, ree);
54    xlim([min(pers),max(pers)]);
55    xlabel('Periods Integrated');
56    ylabel('Relative Error');

```

```

57     legend('4 Slabs / Period', '8 Slabs / Period', 'Richardson Extrapolation');
58
59     %%%%%%%%%%%%%%
60     % Part (ii) %
61     %%%%%%%%%%%%%%
62
63     % Calculate the RHS.
64     rhs = sqrt(pii / 2);
65     fprintf('RHS: %28.20e\n', rhs);
66
67     % Define the integrand with a function handle.
68     f = @(x) sin(x) / sqrt(x);
69
70     figure();
71     x = linspace(0,500,5000);
72     y = zeros(0,length(x));
73     for i = 1:length(x)
74         y(i) = f(x(i));
75     end
76     plot(x,y);
77     xlabel('x');
78     ylabel('y');
79     legend('sin(x) / sqrt(x)');
80
81     min_x = 0;
82     y0 = 1;
83
84     % Do slabs/period = {4, 8}.
85
86     slabspp = 4;
87
88     pers = (1:9)' * 10.^(0:2);
89     pers = reshape(pers, 1, numel(pers));
90     F1 = zeros(1, length(pers));
91     F2 = zeros(1, length(pers));
92     Fe = zeros(1, length(pers));
93
94     for i = 1:length(pers)
95
96         per = pers(i);
97
98         F1(i) = Trapezoidal_Inf(f, min_x, slabspp, 2*pi, per, y0);
99         F2(i) = Trapezoidal_Inf(f, min_x, slabspp*2, 2*pi, per, y0);
100        Fe(i) = F2(i) + (F2(i) - F1(i)) / 3;
101
102    end
103
104    re1 = RelErr(F1, rhs);
105    re2 = RelErr(F2, rhs);
106    ree = RelErr(Fe, rhs);
107
108    figure();
109    loglog(pers, re1, pers, re2, pers, ree);
110    xlim([min(pers),max(pers)]);
111    xlabel('Periods Integrated');
112    ylabel('Relative Error');
113    legend('4 Slabs / Period', '8 Slabs / Period', 'Richardson Extrapolation');
114
115    % Do slabs/period = {22, 44}.
116
117    slabspp = 22;
118
119    pers = (1:9)' * 10.^(0:2);
120    pers = reshape(pers, 1, numel(pers));
121    pers = 1:150;
122    F1 = zeros(1, length(pers));
123    F2 = zeros(1, length(pers));
124    Fe = zeros(1, length(pers));
125

```

```

126     for i = 1:length(pers)
127
128         per = pers(i);
129
130         F1(i) = Trapezoidal_Inf(f, min_x, slabspp, 2*pii, per, y0);
131         F2(i) = Trapezoidal_Inf(f, min_x, slabspp*2, 2*pii, per, y0);
132         Fe(i) = F2(i) + (F2(i) - F1(i)) / 3;
133
134     end
135
136     re1 = RelErr(F1, rhs);
137     re2 = RelErr(F2, rhs);
138     ree = RelErr(Fe, rhs);
139
140     figure();
141     loglog(pers, re1, pers, re2, pers, ree);
142     xlim([min(pers),max(pers)]);
143     xlabel('Periods Integrated');
144     ylabel('Relative Error');
145     legend('28 Slabs / Period', '56 Slabs / Period', 'Richardson Extrapolation');
146
147 end

```

**Listing 6:** Trapezoidal\_Inf.m

```

1  function [ F ] = Trapezoidal_Inf( f, min_x, n_slabpp, T, n_periods, y0 )
2
3  %%%
4  %
5  % Trapezoidal integration of a function handle f, assuming ...
6  %   - indefinite right bound, and
7  %   - function has some period T, and
8  %   - zero-value is undefined
9  %
10 % Note: n_slabpp is number of slabs per period.
11 %
12 % If the most recently calculated period's integral F_T divided by the total integral
13 %   F + F_T, is below tolerance, integration stops.
14 %
15 % If zero_value is specified, y(x==0) will be set to zero_value analytically. This is
16 %   intended to circumvent divide-by-zero errors at x==0 by passing the analytic value of
17 %   the limit to this function.
18 %
19 %%%
20
21 % Number of points per period for the trapezoidal rule.
22 n_ppp = n_slabpp + 1;
23
24 % Index of the period we are evaluating.
25 period_i = 0;
26
27 % Value of the integral.
28 F = 0;
29
30 while period_i < n_periods
31
32     % This period's contribution to the integral.
33     F_T = 0;
34
35     a = min_x + period_i * T;
36     b = min_x + (period_i + 1) * T;
37     x = linspace(a, b, n_ppp);
38
39     h = x(2) - x(1);
40
41     y = zeros(1, n_ppp);
42     for i = 1:n_ppp
43         if x(i) == 0

```

```

44         y(i) = y0;
45     else
46         y(i) = feval(f, x(i));
47     end
48 end
49
50 % Sum each slab's contribution to this period's integral.
51 for i = 1:n_slabpp
52     slab = (h/2) * ( y(i) + y(i+1) );
53     F_T = F_T + slab;
54 end
55
56 % Update total integral.
57 F = F + F_T;
58
59 % Move to the next period.
60 period_i = period_i + 1;
61
62 end
63
64 end

```

**Listing 7: Problem\_C.m**

```

1  function [] = Problem_C()
2
3      % Calculate pi, but name it pii since MATLAB has already defined the variable pi.
4      pii = 4 * atan(1);
5
6      %%%%%%%%%%%
7      % Part (i) %
8      %%%%%%%%%%%
9
10     rhs = 46/3;
11
12     f = @(x,y) x * y * (1+x);
13
14     F = Simpsons_2D(f, [1,2], [1,3], 1, 1);
15
16     re = RelErr(F, rhs);
17     fprintf('re: %28.20e (Part i)\n', re);
18
19     %%%%%%%%%%%
20     % Part (ii) %
21     %%%%%%%%%%%
22
23     rhs = 365/3;
24
25     f = @(x,y) x^2 * y^3 * (1+x);
26
27     F = Simpsons_2D(f, [1,2], [1,3], 1, 1);
28
29     re = RelErr(F, rhs);
30     fprintf('re: %28.20e (Part ii)\n', re);
31
32 end

```

**Listing 8: Simpsons\_2D.m**

```

1  function [ F ] = Simpsons_2D( f, bounds_x, bounds_y, n_slabs_x, n_slabs_y )
2
3      n_pts_x = 1 + 2 * n_slabs_x;
4      n_pts_y = 1 + 2 * n_slabs_y;
5      x = linspace(bounds_x(1), bounds_x(2), n_pts_x);
6      y = linspace(bounds_y(1), bounds_y(2), n_pts_y);
7      hx = x(2) - x(1);
8      hy = y(2) - y(1);

```

```

9
10 z = zeros(n_pts_x, n_pts_y);
11 for i = 1:n_pts_x
12     for j = 1:n_pts_y
13         z(i,j) = feval(f, x(i), y(j));
14     end
15 end
16
17 % Weights to use for each coordinate within the slab.
18 simpson_square = [1, 4, 1; ...
19                  4, 16, 4; ...
20                  1, 4, 1];
21
22 F = 0;
23 for i = 1:n_slabs_x
24     for j = 1:n_slabs_y
25         % Determine indices involved in Simpson's rule for this slab.
26         ii = 1 + 2*(i-1);
27         jj = 1 + 2*(j-1);
28         % Calculate volume of slab.
29         slab = sum(sum( simpson_square .* z(ii:ii+2, jj:jj+2) ));
30         F = F + slab;
31     end
32 end
33 F = F * (hx * hy / 9);
34
35 end

```

**Listing 9: Set\_Default\_Plot\_Properties.m**

```

1 function [] = Set_Default_Plot_Properties()
2
3     set(groot, 'DefaultTextInterpreter', 'none', ...
4             'DefaultLegendInterpreter', 'none', ...
5             'DefaultLineLineWidth', 1.5, ...
6             'DefaultAxesLineWidth', 1.5, ...
7             'DefaultAxesBox', 'on', ...
8             'DefaultAxesXGrid', 'on', ...
9             'DefaultAxesYGrid', 'on', ...
10            'DefaultAxesZGrid', 'on', ...
11            'DefaultAxesGridColor', [0.5,0.5,0.5], ...
12            'DefaultAxesFontName', 'Helvetica', ...
13            'DefaultAxesFontSize', 12, ...
14            'DefaultAxesTitleFontSizeMultiplier', 1.2, ...
15            'DefaultAxesLabelFontSizeMultiplier', 1.2 ...
16            );
17
18 end

```

**Listing 10: Dashes.m**

```

1 function [] = Dashes()
2     fprintf('-----\n');
3 end

```