Michael Cowan

Computational Physics

Problem Set 10

1. Buffon’s needle problem

In this problem, we write a program to calculate the value of π by the method of Buffon’s needle problem. In the problem, a needle of length L is dropped onto a surface containing parallel lines that are a distance *L* apart. The probability of the needle crossing one of the lines is known to be 2/π. This process is simulated by choosing two random numbers, corresponding to the angle *θ* and the spatial position *x* of the dropped needle. If , the needle will be crossing a line and a successful trial is recorded. π is then calculated as 2\*number of total trials/number of successful trials.

Code:

1. clear all;
2. L=1; %set the needle and line spacing length
3. s=0; %initialize the number of successes;
4. n=input('How many times should the needle be dropped? '); %number of trials
5. for i=1:n
6. %generate values of theta and x
7. theta=rand\*pi;
8. x=rand;
9. %success condition
10. if (L-x)<=L\*sin(theta)
11. s=s+1;
12. endif
13. endfor
14. p=2\*n/s;
15. fprintf('The calculated value of pi is: %g\n',p);

Results:

|  |  |  |
| --- | --- | --- |
| Number of drops | Calculated value of π | Variance from actual value [(x-π)2] |
| 1X103 | 3.1348 | 4.614X10-5 |
| 1X105 | 3.14485 | 1.061X10-5 |
| 1X106 | 3.14111 | 2.330X10-7 |
| 1X107 | 3.14137 | 4.957X10-8 |

The method appears to work very well. As expected, the variance decreases as the number of simulated drops increases.

2. Garcia Chapter 10 Problem 13

Here we use Romberg integration to numerically calculate the values of six different integrals. These numerical integration results are then compared to the values attained by integrating analytically.

Part a:

Code:

1. clear all;
2. %set values
3. a=0;
4. b=1;
5. N=10;
6. param=0;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'exp',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(e^(1)-1));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y=exp(x,param)
20. y=e^(x);
21. return;

Results:

1.85914 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

1.75393 1.71886 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

1.72722 1.71832 1.71828 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

1.72052 1.71828 1.71828 1.71828 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

1.71884 1.71828 1.71828 1.71828 1.71828 0.00000 0.00000 0.00000 0.00000 0.00000

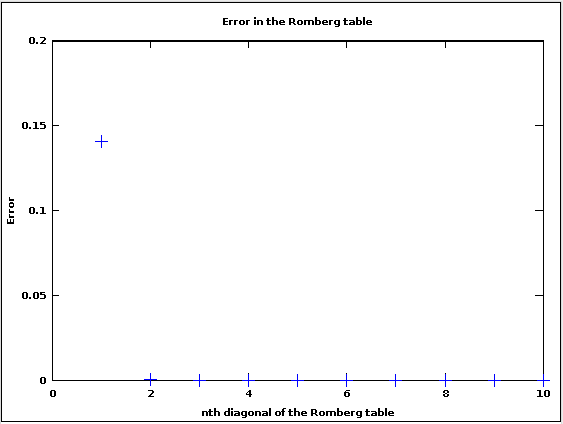
1.71842 1.71828 1.71828 1.71828 1.71828 1.71828 0.00000 0.00000 0.00000 0.00000

1.71832 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 0.00000 0.00000 0.00000

1.71829 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 0.00000 0.00000

1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 0.00000

1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828 1.71828



Part b:

Code:

1. clear all;
2. %set values
3. a=0;
4. b=2\*pi;
5. N=10;
6. param=0;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'sin4',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(3\*pi/4));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y=sin4(x,param)
20. y=(sin(8\*x))^4;
21. return;

Results:

0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

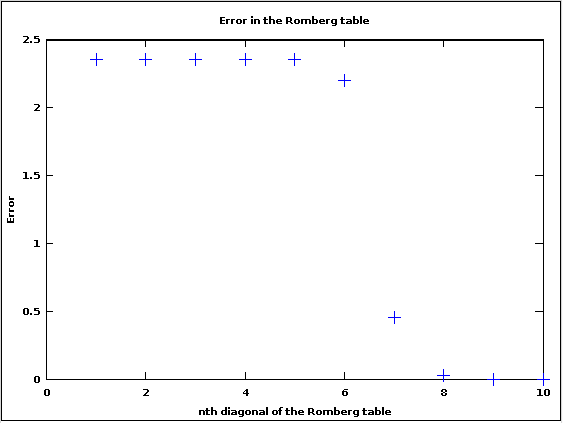
3.14159 4.18879 4.46804 4.53896 4.55676 4.56122 0.00000 0.00000 0.00000 0.00000

2.35619 2.09440 1.95477 1.91488 1.90458 1.90199 1.90134 0.00000 0.00000 0.00000

2.35619 2.35619 2.37365 2.38030 2.38212 2.38259 2.38271 2.38274 0.00000 0.00000

2.35619 2.35619 2.35619 2.35592 2.35582 2.35580 2.35579 2.35579 2.35579 0.00000

2.35619 2.35619 2.35619 2.35619 2.35620 2.35620 2.35620 2.35620 2.35620 2.35620



Part c:

Code:

1. clear all;
2. %set values
3. a=0;
4. b=1;
5. N=10;
6. param=0;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'root',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(2/3));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y=root(x,param)
20. y=sqrt(x);
21. return;

Result:

0.50000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.60355 0.63807 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.64328 0.65653 0.65776 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.65813 0.66308 0.66352 0.66361 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.66358 0.66540 0.66555 0.66559 0.66559 0.00000 0.00000 0.00000 0.00000 0.00000

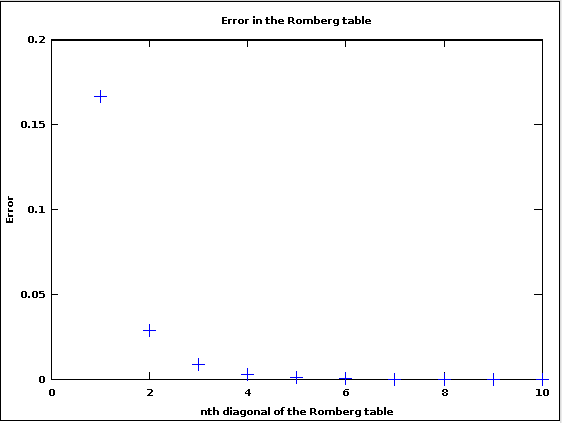
0.66556 0.66622 0.66627 0.66628 0.66629 0.66629 0.00000 0.00000 0.00000 0.00000

0.66627 0.66651 0.66653 0.66653 0.66653 0.66653 0.66653 0.00000 0.00000 0.00000

0.66653 0.66661 0.66662 0.66662 0.66662 0.66662 0.66662 0.66662 0.00000 0.00000

0.66662 0.66665 0.66665 0.66665 0.66665 0.66665 0.66665 0.66665 0.66665 0.00000

0.66665 0.66666 0.66666 0.66666 0.66666 0.66666 0.66666 0.66666 0.66666 0.66666



Part d:

Code:

1. clear all;
2. %set values
3. a=0;
4. b=1;
5. N=10;
6. param=0;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'root1',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(pi/4));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y=root1(x,param)
20. y=sqrt(1-x^2);
21. return;

Results:

0.50000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.68301 0.74402 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.74893 0.77090 0.77269 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.77245 0.78030 0.78092 0.78105 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.78081 0.78360 0.78382 0.78387 0.78388 0.00000 0.00000 0.00000 0.00000 0.00000

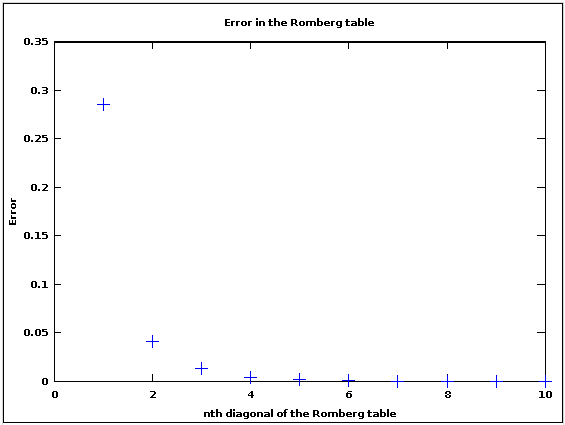
0.78378 0.78476 0.78484 0.78486 0.78486 0.78486 0.00000 0.00000 0.00000 0.00000

0.78482 0.78517 0.78520 0.78521 0.78521 0.78521 0.78521 0.00000 0.00000 0.00000

0.78520 0.78532 0.78533 0.78533 0.78533 0.78533 0.78533 0.78533 0.00000 0.00000

0.78533 0.78537 0.78537 0.78537 0.78537 0.78537 0.78537 0.78537 0.78537 0.00000

0.78537 0.78539 0.78539 0.78539 0.78539 0.78539 0.78539 0.78539 0.78539 0.78539



Part e:

1. clear all;
2. %set values
3. a=-1;
4. b=1;
5. N=10;
6. param=10;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'legndr',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(0));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y = legndr2(x,n)
20. % Legendre polynomials function
21. % Inputs
22. % n = Highest order polynomial returned
23. % x = Value at which polynomial is evaluated
24. % Output
25. % p = Vector containing P(x) for order 0,1,...,n
26. %\* Perform upward recursion
27. p(1)=1; % P(x) for n=0
28. if(n == 0) return; end
29. p(2)=x; % P(x) for n=1
30. for i=3:n+1 % Use upward recursion to obtain other n's
31. p(i) = ((2\*i-3)\*x\*p(i-1) - (i-2)\*p(i-2))/(i-1);
32. end
33. y=(p(n+1))^2;
34. return;

Results:

2.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.75391 0.33854 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.18872 0.00033 -0.02222 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.33715 0.38662 0.41238 0.41928 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.12648 0.05626 0.03423 0.02823 0.02670 0.00000 0.00000 0.00000 0.00000 0.00000

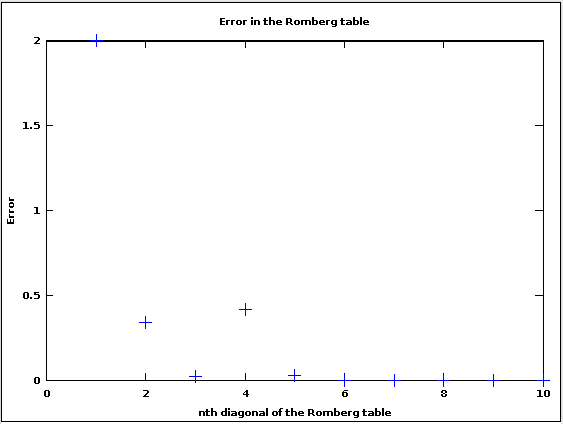
0.03473 0.00414 0.00067 0.00014 0.00003 0.00000 0.00000 0.00000 0.00000 0.00000

0.00888 0.00027 0.00001 0.00000 0.00000 -0.00000 -0.00000 0.00000 0.00000 0.00000

0.00223 0.00002 0.00000 0.00000 0.00000 -0.00000 -0.00000 -0.00000 0.00000 0.00000

0.00056 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00014 0.00000 0.00000 0.00000 -0.00000 -0.00000 -0.00000 -0.00000 -0.00000 -0.00000



Part f:

Code:

1. clear all;
2. %set values
3. a=-1;
4. b=1;
5. N=20;
6. param=10;
7. %calculate the Romberg table
8. R=rombf(a,b,N,'legndr2',param);
9. disp(R);
10. %calculate and display the error of the diagonals of the Romberg table
11. x=(1:N);
12. for i=1:N
13. delta(i)=abs(R(i,i)-(2/21));
14. endfor
15. plot(x,delta,'+');
16. xlabel('nth diagonal of the Romberg table');
17. ylabel('Error');
18. title('Error in the Romberg table');
19. function y = legndr2(x,n)
20. % Legendre polynomials function
21. % Inputs
22. % n = Highest order polynomial returned
23. % x = Value at which polynomial is evaluated
24. % Output
25. % p = Vector containing P(x) for order 0,1,...,n
26. %\* Perform upward recursion
27. p(1)=1; % P(x) for n=0
28. if(n == 0) return; end
29. p(2)=x; % P(x) for n=1
30. for i=3:n+1 % Use upward recursion to obtain other n's
31. p(i) = ((2\*i-3)\*x\*p(i-1) - (i-2)\*p(i-2))/(i-1);
32. end
33. y=(p(n+1))^2;
34. return;

Results:

2.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

1.06056 0.74742 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.56571 0.40076 0.37765 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.34227 0.26779 0.25892 0.25704 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.18891 0.13780 0.12913 0.12707 0.12656 0.00000 0.00000 0.00000 0.00000 0.00000

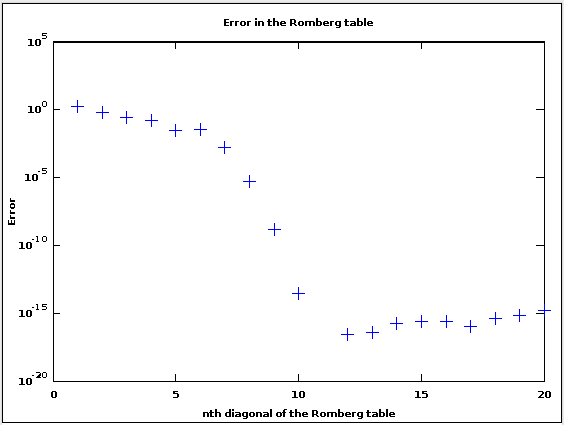
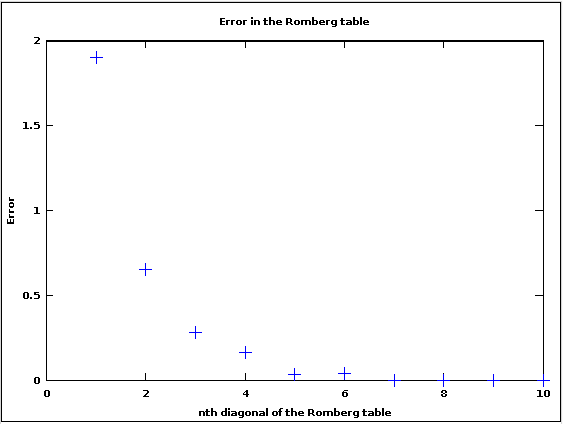
0.14734 0.13348 0.13319 0.13326 0.13328 0.13329 0.00000 0.00000 0.00000 0.00000

0.11176 0.09991 0.09767 0.09710 0.09696 0.09693 0.09692 0.00000 0.00000 0.00000

0.09963 0.09558 0.09529 0.09525 0.09525 0.09524 0.09524 0.09524 0.00000 0.00000

0.09635 0.09526 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524 0.00000

0.09552 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524 0.09524



Due to round-off errors, the error in the calculation can increase as the size of the table increases. This increase in error is small, but can be seen using a logarithmic plot, as show above on the right.