

第一章作业

Anonymity 3220100000*

* College of Control Science and Engineering, Robotics Engineering

July 8, 2024

1 余弦函数展开

1.1 问题描述

余弦函数的无穷级数展开为 $\cos x = \sum_{i=0}^{\infty} \frac{(-1)^i x^{2i}}{(2i)!}$

(1) 分别以单精度和双精度数据类型, 计算 $x = 1$ 时的近似值, 要求计算结果具有 4 位有效数字;

(2) 如果采用单精度数据类型要求计算结果达到机器精度, 此时结果如何? (测试机器精度: 满足 $1 + \varepsilon > 1$ 的最小浮点数)

1.2 思路分析

通过不断累加级数的项, 直到达到指定的精度要求。

对于计算结果达到机器精度, 采取 PPT 中伪代码思路:

机器精度: 满足 $1 + \varepsilon > 1$ 的最小浮点数

- 测试机器精度的伪代码

- Epsilon=1

- Do

- If (Epsilon+1<=1) exit

- Epsilon=Epsilon/2

End Do

Epsilon=Epsilon*2

1.3 运行结果与分析

单精度和双精度分别计算 \cos 值, 在题目要求保留 4 位有效数字的情况下结果相同:



```
命令行窗口
disp(['双精度近似值: ', num2str(cos(1))])
单精度近似值: 0.5403
双精度近似值: 0.5403
fx >>
```

进一步分析在本次计算过程中， $x=1$ 时的单双精度区别：

```

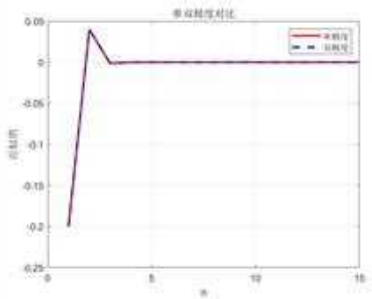
命令行窗口
hold off;
单精度近似值: 0.540277779102325
双精度近似值: 0.540277777777778
真实值: 0.540302305868140
单精度误差: 2.452676562825218e-05
双精度误差: 2.452809036201931e-05

```

可以看到，单精度与双精度对于本次计算的精度相近，原因是在 $x = 1$ 的情况下，截断误差和舍入误差可能会对单精度和双精度计算的结果产生相似的影响。

对于 $x = 1$ ，某些项的阶乘和幂运算较大，导致在单精度和双精度计算中都有相似的数值范围，从而使得舍入误差在两者中产生的影响相近。因此尽管单精度和双精度浮点数在表示范围和精度上存在差异，但对于某些计算，特别是在局部区域（例如 $x = 1$ 附近）的小范围内，它们的误差可能相对较接近。在其他情况下，可能会看到更大的差异，特别是在大范围的计算中，例如 x 的绝对值较大的情况下。

绘制单双精度计算差别随 n 的变化图像：



测试结果达到机器精度运行结果：证明了计算的准确性。

```

命令行窗口
term = (-1)^i * x^(2*i) / factorial(2*i);
end

disp('单精度计算结果: ', num2str(result_single,
机器精度: 1.192093e-07
单精度计算结果: 5.403023e-01
fx >>

```

2 递推公式计算积分

2.1 问题描述

请采用递推公式计算积分 $E_n = \int_0^1 x^n e^{-x} dx$, ($n = 1, 2, \dots$) 要求每项的绝对误差小于 0.01。

2.2 思路分析

首先，我们有积分 E_n 的表达式：

$$E_n = \int_0^1 x^n e^{-x} dx$$

接下来，我们使用分部积分法，其中选择 u 和 dv 如下：

$$u = x^n \quad \text{和} \quad dv = e^{-x} dx$$

计算 du 和 v ：

$$du = nx^{n-1} dx \quad \text{和} \quad v = -e^{-x}$$

分部积分的公式为：

$$\int u dv = uv - \int v du$$

将上述值代入分部积分公式，得到：

$$\begin{aligned} E_n &= \int x^n e^{-x} dx \\ &= -x^n e^{-x} \Big|_0^1 + \int_0^1 nx^{n-1} e^{-x} dx \\ &= -e^{-1} + n \int_0^1 x^{n-1} e^{-x} dx \end{aligned}$$

将积分 E_n 表示为积分 E_{n-1} 的形式：

$$E_n = -e^{-1} + nE_{n-1}$$

依此推导从而得到 n 取相应值时积分的值。

再通过 matlab 中自带的 `integral` 函数得到对应 n 的值，以此设为实际值，再与计算值进行对比。

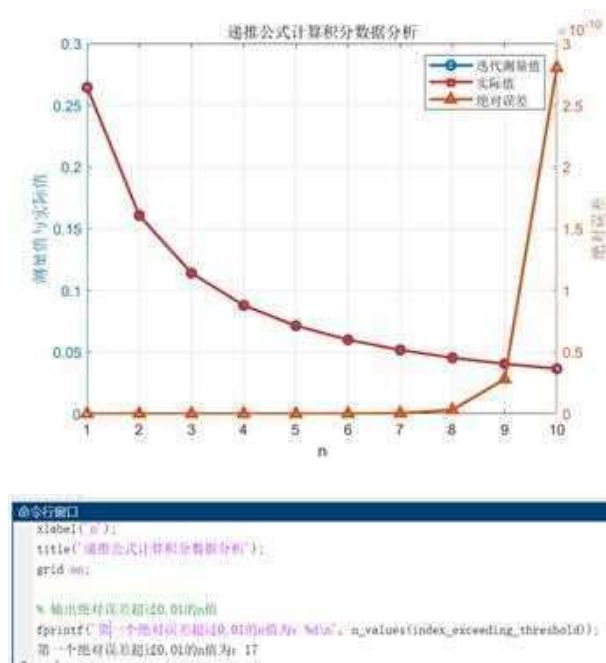
2.3 运行结果与分析

首先让 n 的取值从 1 到 20；绘制出测量值、实际值与绝对误差与 n 相关的输出结果截图如下：



由于美观考虑，将 n 的取值从 1 到 10 可视化绘制为折线图：

从上述结果不难发现，采用递推方式求值时误差逐渐累积，为单调递增函数，因此参照题目中要求，设置绝对误差阈值 0.01，输出得到“第一个绝对误差超过 0.01 的 n 值为：17”



3 实验心得

本次实验我进一步熟悉了 matlab 的使用，初步认识了数值计算的误差等方面，采用计算机编程方法尝试解决一些曾经较难解决的问题。同时，本次的作业应用了许多大一在微积分这门课程上学到的知识，真正实现了学以致用。

4 源代码

4.1 余弦函数展开

4.1.1 单双精度计算

```

1 % 单精度计算
2 x = single(1);
3 result_single = single(0);
4 tolerance = single(0.5*exp(-4));
5 term = single(1);
6 i = 0;
7
8 while abs(term) >= tolerance
9     result_single = result_single + term;
10    i = i + 1;
11    term = (-1)^i * x^(2*i) / factorial(2*i);
12 end
13
14 disp(['单精度近似值: ', num2str(result_single, '%.4f')]);

```

```

15
16 % 双精度计算
17 x = double(1);
18 result_double = double(0);
19 tolerance = 1e-4;
20 term = 1;
21 i = 0;
22
23 while abs(term) >= tolerance
24     result_double = result_double + term;
25     i = i + 1;
26     term = (-1)^i * x^(2*i) / factorial(2*i);
27 end
28
29 disp(['双精度近似值: ', num2str(result_double, '%.4f')]);

```

4.1.2 单双精度误差分析

```

1 % 单精度计算
2 x_single = single(1);
3 result_single = single(0);
4 tolerance_single = single(0.5 * 1e-4);
5 term_single = single(1);
6 i_single = 0;
7
8 while abs(term_single) >= tolerance_single
9     result_single = result_single + term_single;
10    i_single = i_single + 1;
11    term_single = (-1)^i_single * x_single^(2*i_single) / factorial(2*
        i_single);
12 end
13
14 disp(['单精度近似值: ', num2str(result_single, '%.15f')]);
15
16 % 双精度计算
17 x_double = double(1);
18 result_double = double(0);
19 tolerance_double = 1e-4;
20 term_double = 1;
21 i_double = 0;
22

```

```

23 while abs(term_double) >= tolerance_double
24     result_double = result_double + term_double;
25     i_double = i_double + 1;
26     term_double = (-1)^i_double * x_double^(2*i_double) / factorial(2*
        i_double);
27 end
28
29 disp(['双精度近似值: ', num2str(result_double, '%.15f')]);
30
31 % 真实值及误差
32 true_value = cos(1);
33 disp(['真实值: ', num2str(true_value, '%.15f')]);
34 disp(['单精度误差: ', num2str(abs(result_single - true_value), '%.15e'
        )]);
35 disp(['双精度误差: ', num2str(abs(result_double - true_value), '%.15e'
        )]);
36
37 % 绘制图像
38 n_values = 1:15; % 修改此处以限制横坐标n的范围
39 single_values = zeros(size(n_values), 'single');
40 double_values = zeros(size(n_values));
41
42 for n = n_values
43     single_values(n) = sum((-1).^(0:n) .* x_single.^(2*(0:n)) /
        factorial(2*(0:n)));
44     double_values(n) = sum((-1).^(0:n) .* x_double.^(2*(0:n)) /
        factorial(2*(0:n)));
45 end
46
47 figure;
48 plot(n_values, single_values, 'r-', 'LineWidth', 2, 'DisplayName', '单
        精度');
49 hold on;
50 plot(n_values, double_values, 'b--', 'LineWidth', 2, 'DisplayName', '
        双精度');
51 xlabel('n');
52 ylabel('近似值');
53 title('单双精度对比');
54 legend('show');
55 grid on;
56 hold off;

```

4.1.3 测试机器精度

```
1 % 测试机器精度的伪代码实现
2 epsilon = single(1);
3
4 while (epsilon + 1 > 1)
5     epsilon = epsilon / 2;
6 end
7
8 epsilon = epsilon * 2;
9
10 disp(['机器精度: ', num2str(epsilon, '%e')]);
11
12 % 计算结果达到机器精度的情况
13 x = single(1);
14 result_single = single(0);
15 tolerance = epsilon; % 使用机器精度作为计算精度要求
16 term = single(1);
17 i = 0;
18
19 while abs(term) >= tolerance
20     result_single = result_single + term;
21     i = i + 1;
22     term = (-1)^i * x^(2*i) / factorial(2*i);
23 end
24
25 disp(['单精度计算结果: ', num2str(result_single, '%e')]);
```

4.2 递推公式计算积分

4.2.1 递归函数

```
1 function En = calculateEn(n)
2     if n == 1
3         En = 1 - 2/exp(1);
4     elseif n > 1
5         En_minus_1 = 1 - 2/exp(1); % 初始值 E_1
6         for k = 2:n
7             En = -exp(-1) + k * En_minus_1;
8             En_minus_1 = En;
9         end
10    else
```

```

11         error('Input must be a positive integer.');
```

```

12     end
```

```

13 end
```

4.3 数据分析代码

```

1 % 设置n的取值范围为1到10
2 n_values = 1:20;
3
4 % 初始化存储计算结果的数组
5 calculated_values = zeros(size(n_values));
6 actual_values = zeros(size(n_values));
7 absolute_errors = zeros(size(n_values));
8
9 % 计算不同n对应的积分值，实际值和绝对误差
10 for i = 1:length(n_values)
11     n = n_values(i);
12
13     % 使用calculateEn计算积分值
14     calculated_values(i) = calculateEn(n);
15
16     % 使用integral计算实际值
17     f = @(x) x.^n .* exp(-x);
18     result = vpa(integral(f, 0, 1), 15);
19     actual_values(i) = double(result);
20
21     % 计算绝对误差
22     absolute_errors(i) = abs(calculated_values(i) - actual_values(i));
23
24     % 输出测量值、实际值和绝对误差
25     fprintf('n = %d: 测量值 = %.10f, 实际值 = %.10f, 绝对误差 = %.15f\
        n', n, calculated_values(i), actual_values(i), absolute_errors
        (i));
26 end
27
28 % 寻找第一个绝对误差超过0.01的n值
29 threshold = 0.01;
30 index_exceeding_threshold = find(absolute_errors > threshold, 1, '
    first');
31
32 % 绘制折线图
```



```

33 figure;
34
35 yyaxis left;
36 plot(n_values, calculated_values, 'o-', 'LineWidth', 2, 'DisplayName',
    '迭代测量值');
37 hold on;
38 plot(n_values, actual_values, 's-', 'LineWidth', 2, 'DisplayName', '实
    际值', 'Color', [0.8 0.2 0.2]);
39 ylabel('测量值与实际值');
40 legend('show');
41
42 yyaxis right;
43 plot(n_values, absolute_errors, '^-', 'LineWidth', 2, 'DisplayName', '
    绝对误差');
44 ylabel('绝对误差');
45 xlabel('n');
46 title('递推公式计算积分数据分析');
47 grid on;
48
49 % 输出绝对误差超过0.01的n值
50 fprintf('第一个绝对误差超过0.01的n值为: %d\n', n_values(
    index_exceeding_threshold));

```