数值分析实验报告 - Code 0

Chase Young

2024 年 2 月 27 日

1　实验目的

本实验要求计算级数

$$\varphi(x) = \sum_{k=1}^{\infty} \frac{1}{k(k+x)} \tag{1}$$

分别在 $x = 0, 0.1, 0.2, \ldots, 1.0$ 和 $10, 20, \ldots, 300$ 处的函数值，并要求误差小于 10^{-6}，同时给出得到上述误差时相应的 k 的最小值。

要求分 3 列输出每一组 $x, \varphi(x)$ 和对应的最小的 k。

2　实验方法

为了计算 x 点处，级数(1)在前 k 项的截断误差，我们首先需要计算出此级数的在 x 处的精确值。为此，我们使用 Mathematica 计算函数 $\varphi(x)$ 的值，并保留 7 位有效数字[1]。相关 Mathematica 代码见文件 exactValue.nb。计算结果保存在文件 exactValue.txt 中，每条数据用换行符隔开。

得到精确解后，我们使用北太天元来具体计算 $\varphi(x)$ 和对应最小的 k。我们先定义用于计算 $\varphi(x)$ 的函数：

- **function** [value, k] = hamming(x, exactValue)

其中 x 是自变量的取值，exactValue 是该点处的精确值；函数通过 while 循环计算得到满足误差要求的 k 和此时的函数值 value，作为返回值。此函数实现见脚本 hamming.m。

在脚本 code_0.m 中，使用 for 循环，对要求的每一个 x 逐一计算满足要求的 k 和对应的 $\varphi(x)$ 的值，并按照要求打印。

3　实验结果

实验得到的 $x, k, \varphi(x)$ 如表 1, 2, 3, 4所示。

[1] 由于计算要求误差小于 10^{-6}，而在要求计算的点 x 处，$\varphi(x)$ 都小于 10，因此保留 7 位有效数字足够精度要求。

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
k	103428	957026	960111	101339	999210	102274	100389	959200	102802	982463	100000
$\varphi(x)$	1.644933	1.534606	1.440878	1.360082	1.289577	1.227410	1.172104	1.122518	1.077758	1.037110	0.999999

表 1: 实验结果：$x, k, \varphi(x)$

x	10	20	30	40	50	60	70	80	90	100
k	975228	101743	963658	102459	100322	999755	100488	998975	100326	100479
$\varphi(x)$	0.292896	0.179886	0.133165	0.106963	0.089983	0.077997	0.069040	0.062067	0.056472	0.051873

表 2: 实验结果 (续)

x	110	120	130	140	150	160	170	180	190	200
k	995437	100088	998874	995208	100267	100478	100053	998128	998290	995183
$\varphi(x)$	0.048019	0.044740	0.041911	0.039445	0.037274	0.035346	0.033622	0.032071	0.030667	0.029389

表 3: 实验结果 (续)

x	210	220	230	240	250	260	270	280	290	300
k	998192	100072	998533	100459	998879	100087	100346	100301	995616	996925
$\varphi(x)$	0.028221	0.027150	0.026162	0.025249	0.024402	0.023614	0.022879	0.022191	0.021547	0.020941

表 4: 实验结果 (续)

观察上述实验结果，可以发现，在当前精度要求下，求出的最小的 k 值几乎都在 10^6 和 10^5 附近。而求出的 $\varphi(x)$ 值随着 x 的增大而减小，这也符合表达式 (1)。

4 后续讨论

在实际运行代码的过程中，发现在北太天元数值软件中使用 for 循环的效率极低。虽然北太天元软件官方人员建议尽量少用 for 循环，而是使用向量化的加速技巧，和同是以向量化加速闻名的 Matlab 软件相比，Matlab 的 for 循环效率仍然明显比北太天元高。

A 代码

本部分包含了主要用到的代码。

hamming.m

```
function [value, k] = hamming(x, exactValue)
    % compute Hamming function, return value and min k
    value = 0;
    k = 1;
    while abs(value-exactValue) >= 1e-6
        value = value + 1/(k*(k+x));
        k = k + 1;
        if k > 1e7
            break;
        end
    end
end
```

code_0.m

```
tmp1 = 0.0:0.1:1.0;
tmp2 = 10:10:300;
```

```
xList = [tmp1, tmp2];
kList = 1:41;
valueList = 1:41;

exactValue = readmatrix("exactValue.txt");
for i = 1:length(exactValue)
    [valueList(i), kList(i)] = hamming(xList(i), exactValue(i));
end

for i = 1:length(exactValue)
    fprintf("%.1f  %.6f  %d\n", xList(i), valueList(i), kList(i));
end

for i = 1:length(exactValue)
    fprintf("%.6f\t", valueList(i));
end
```