第二次作业 报告

1400012141
邵智轩
2017年11月

1 Runge 效应

这一部分代码提交在文件“Runge_Phenomenon.py”中。

1.1 多项式内插

由于插值多项式的存在唯一性，Newton插值、Lagrange插值、Neville 插值的效果应该是一样的。在这三种方法中，Neville 算法更适合于外推到某一个固定的点，而不适合求插值多项式本身。我在代码中分别试验了前两种插值，由于结果是差不多的，下面仅列出Lagrange插值得到的41个点的数值。（由于$f(x)$是一个偶函数，只列出了21个点，运行代码得到的数据可以检验：拟合出的多项式也是偶函数（仅有很小的误差），这一点也能从图像上看出。）

Table 1: Lagrange多项式插值

	1	2	3	4	5
x	-1	-0.95	-0.9	-0.85	-0.8
$f(x)$	0.03846154	0.04244032	0.04705882	0.05245902	0.05882353
$L_{20}(x)$	0.03846154	-39.95244903	0.04705882	3.45495780	0.05882353
$\|L_{20}(x) - f(x)\|$	0.0	39.99488935	0.0	3.40249878	0.0
	6	7	8	9	10
x	-0.75	-0.7	-0.65	-0.6	-0.55
$f(x)$	0.06639004	0.07547170	0.08648649	0.1	0.11678832
$L_{20}(x)$	-0.44705196	0.07547170	0.20242262	0.1	0.08065999
$\|L_{20}(x) - f(x)\|$	0.11593613	0.0	0.03612833	0.0	0.01481418
	11	12	13	14	15
x	-0.5	-0.45	-0.4	-0.35	-0.3
$f(x)$	0.13793103	0.16494845	0.2	0.24615384	0.30769231
$L_{20}(x)$	0.13793103	0.17976263	0.2	0.23844593	0.30769231
$\|L_{20}(x) - f(x)\|$	0.0	0.00770791	0.0	0.00484915	0.0
	16	17	18	19	20
x	-0.25	-0.2	-0.15	-0.1	-0.05
$f(x)$	0.39024390	0.5	0.64	0.8	0.94117647
$L_{20}(x)$	0.39509305	0.5	0.63675534	0.8	0.94249038
$\|L_{20}(x) - f(x)\|$	0.00324466	0.0	0.00131391	0.0	0.00131391
	21	...			
x	0.0	...			
$f(x)$	1.0	...			
$L_{20}(x)$	1.0	...			
$\|L_{20}(x) - f(x)\|$	0.0	...			

由此可见，内插多项式虽然保证穿过每一个插值节点，但在拟合Runge函数时，在-1和+1两端附近，非插值节点的函数值$L_{20}(x)$与原函数$f(x)$相差非常大。随着内插阶数的升高，Lagrange多项式体现出越来越严重的震荡特性，与原函数的最大偏离的绝对值会发散。从原函数和插值多项式的图像可以更清晰地看出这一点。

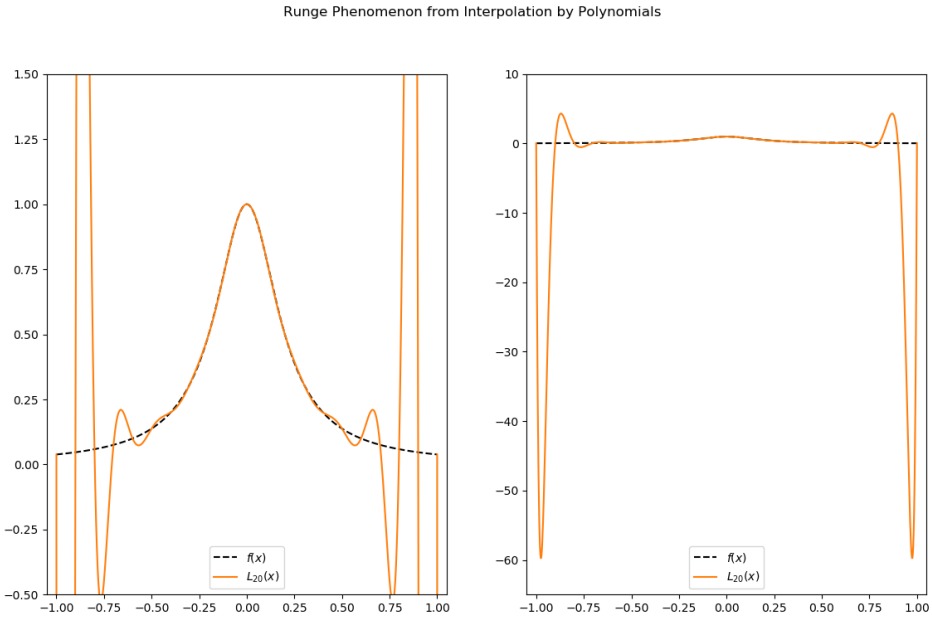


Figure 1: Lagrange多项式插值

1.2 Chebyshev 近似

同样地，Chebyshev近似函数是对称的，仅列出前21个点。

Table 2: ChebyShev近似

	1	2	3	4	5
x	-1	-0.95	-0.9	-0.85	-0.8
$f(x)$	0.03846154	0.04244032	0.04705882	0.05245902	0.05882353
$C(x)$	0.03701576	0.04084866	0.04868520	0.05226088	0.05671349
$\|C(x) - f(x)\|$	0.00144578	0.00159166	0.00162638	0.00019813	0.00211004
	6	7	8	9	10
x	-0.75	-0.7	-0.65	-0.6	-0.55
$f(x)$	0.06639004	0.07547170	0.08648649	0.1	0.11678832
$C(x)$	0.06716920	0.07825215	0.08653370	0.09641294	0.11412551
$\|C(x) - f(x)\|$	0.00077916	0.00278045	0.00004721	0.00358706	0.00266281
	11	12	13	14	15
x	-0.5	-0.45	-0.4	-0.35	-0.3
$f(x)$	0.13793103	0.16494845	0.2	0.24615384	0.30769231
$C(x)$	0.14052347	0.17112412	0.20276314	0.24017480	0.29633274
$\|C(x) - f(x)\|$	0.00259244	0.00617567	0.00276314	0.00597904	0.0113595
	16	17	18	19	20
x	-0.25	-0.2	-0.15	-0.1	-0.05
$f(x)$	0.39024390	0.5	0.64	0.8	0.94117647
$C(x)$	0.38533503	0.51189455	0.66385410	0.81260597	0.92207346
$\|C(x) - f(x)\|$	0.00490887	0.01189455	0.02385410	0.01260597	0.01910301
	21	...			
x	0.0	...			
$f(x)$	1.0	...			
$C(x)$	0.96240967	...			
$\|C(x) - f(x)\|$	0.03759033	...			

Chebyshev近似很好地规避了Runge 效应。与Lagrange多项式不同，Chebyshev 拟合曲线反倒是两端偏离小，中间偏离大。这是因为插值节点是Chebyshev多项式的零点：

$$x_k = \cos\left(\frac{\pi(k+1/2)}{20}\right), \quad k = 0, 1, 2, ...$$

在$[-1, 1]$之间不是均匀分布的。在这些节点处$C(x)$严格等于$f(x)$，从图上的红点也可以看出这一点。

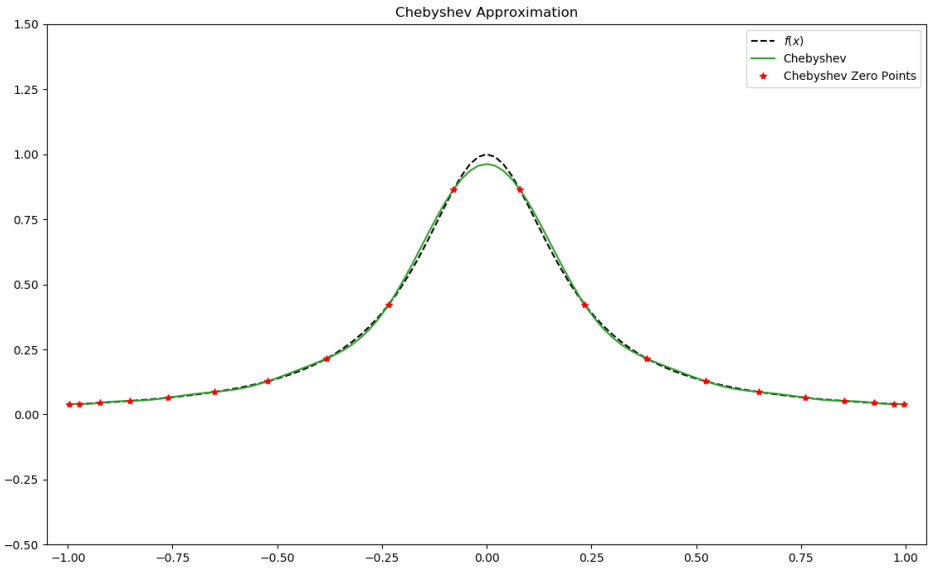


Figure 2: Chebyshev近似

1.3 三次样条插值

Table 3: 三次样条插值

	1	2	3	4	5
x	-1	-0.95	-0.9	-0.85	-0.8
$f(x)$	0.03846154	0.04244032	0.04705882	0.05245902	0.05882353
$S(x)$	0.03846154	0.04253422	0.04705882	0.05243129	0.05882353
$\|S(x) - f(x)\|$	0.00000000	0.00009390	0.00000000	0.00002773	0.00000000
	6	7	8	9	10
x	-0.75	-0.7	-0.65	-0.6	-0.55
$f(x)$	0.06639004	0.07547170	0.08648649	0.1	0.11678832
$S(x)$	0.06639405	0.07547170	0.08647363	0.10000000	0.11678687
$\|S(x) - f(x)\|$	0.00000401	0.00000000	0.00001286	0.00000000	0.00000145
	11	12	13	14	15
x	-0.5	-0.45	-0.4	-0.35	-0.3
$f(x)$	0.13793103	0.16494845	0.2	0.24615384	0.30769231
$S(x)$	0.13793103	0.16486456	0.20000000	0.24626816	0.30769231
$\|S(x) - f(x)\|$	0.00000000	0.00008390	0.00000000	0.00011431	0.00000000
	16	17	18	19	20
x	-0.25	-0.2	-0.15	-0.1	-0.05
$f(x)$	0.39024390	0.5	0.64	0.8	0.94117647
$S(x)$	0.38941957	0.50000000	0.64316894	0.80000000	0.93886621
$\|S(x) - f(x)\|$	0.00082433	0.00000000	0.00316894	0.00000000	0.00231026
	21	...			
x	0.0	...			
$f(x)$	1.0	...			
$S(x)$	1.0000000	...			
$\|S(x) - f(x)\|$	0.0000000	...			

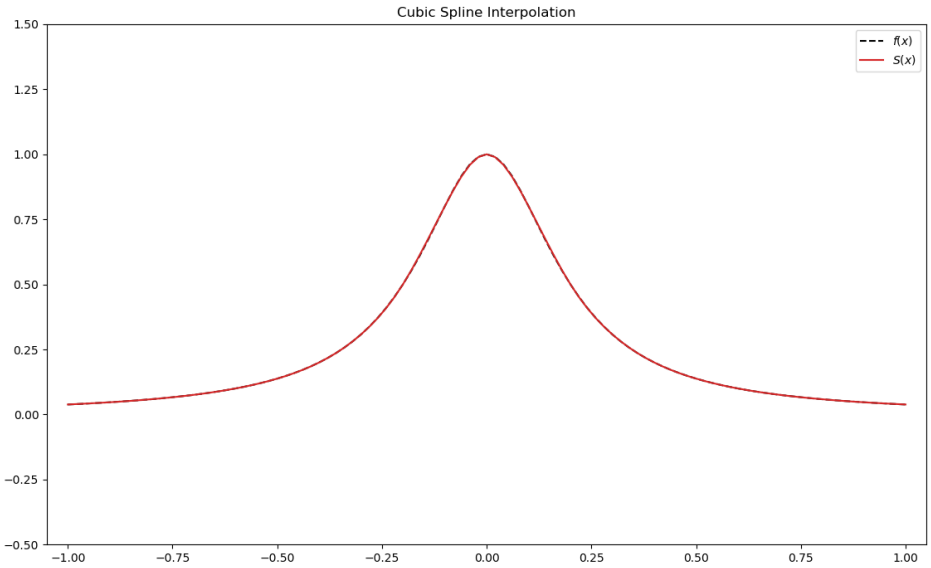


Figure 3: 三次样条插值

三次样条函数内插同样也避免了Runge现象，光滑性好，而且与原函数拟合得非常贴近。

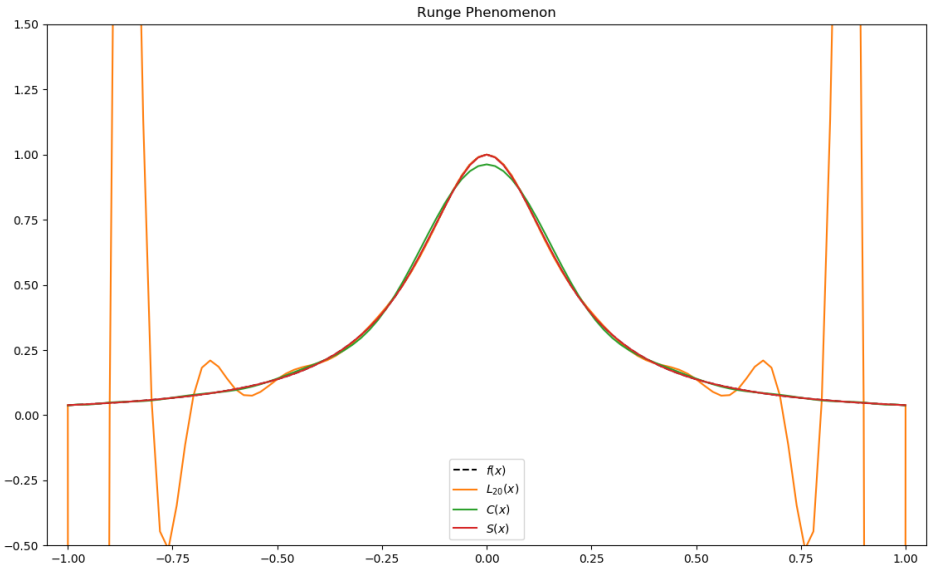


Figure 4: 三种拟合方法的对比

2 样条函数在计算机绘图中的应用

这部分代码提交在文件“CubicSpline_Plot.py”中。

2.1 求出(x_t, y_t)

Table 4: 插值点的坐标

t	0	1	2	3	4	5	6	7	8
x_t	0.0000	0.2071	0.0000	-1.2071	-2.0000	-1.2071	-0.0000	0.2071	0.0000
y_t	0.0000	0.2071	1.0000	1.2071	0.0000	-1.2071	-1.0000	-0.2071	-0.0000

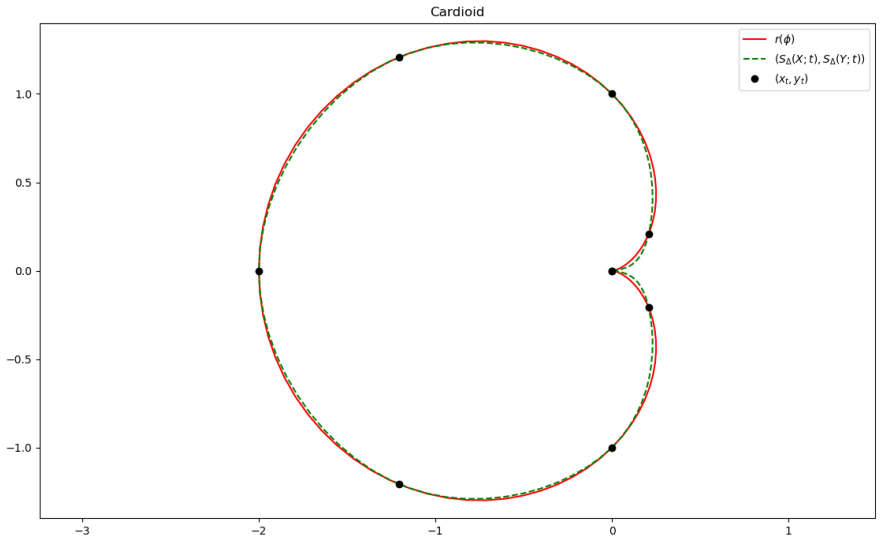
2.2 给出过这8个点的三次样条插值函数

$$S_\Delta(X;t) = \begin{cases} (0.00000)(1-t)^3 + (-0.03850)(t-0)^3 + (0.00000)(1-t) + (0.24560)(t-0) & t \in [0,1] \\ (-0.03850)(2-t)^3 + (-0.26023)(t-1)^3 + (0.24560)(2-t) + (0.26023)(t-1) & t \in [1,2] \\ (-0.26023)(3-t)^3 + (0.07943)(t-2)^3 + (0.26023)(3-t) + (-1.28654)(t-2) & t \in [2,3] \\ (0.07943)(4-t)^3 + (0.35673)(t-3)^3 + (-1.28654)(4-t) + (-2.35673)(t-3) & t \in [3,4] \\ (0.35673)(5-t)^3 + (0.07943)(t-4)^3 + (-2.35673)(5-t) + (-1.28654)(t-4) & t \in [4,5] \\ (0.07943)(6-t)^3 + (-0.26023)(t-5)^3 + (-1.28654)(6-t) + (0.26023)(t-5) & t \in [5,6] \\ (-0.26023)(7-t)^3 + (-0.03850)(t-6)^3 + (0.26023)(7-t) + (0.24560)(t-6) & t \in [6,7] \\ (-0.03850)(8-t)^3 + (0.00000)(t-7)^3 + (0.24560)(8-t) + (0.00000)(t-7) & t \in [7,8] \end{cases}$$

$$S_\Delta(Y;t) = \begin{cases} (0.00000)(1-t)^3 + (0.17350)(t-0)^3 + (0.00000)(1-t) + (0.03361)(t-0) & t \in [0,1] \\ (0.17350)(2-t)^3 + (-0.10819)(t-1)^3 + (0.03361)(2-t) + (1.10819)(t-1) & t \in [1,2] \\ (-0.10819)(3-t)^3 + (-0.32650)(t-2)^3 + (1.10819)(3-t) + (1.53361)(t-2) & t \in [2,3] \\ (-0.32650)(4-t)^3 + (-0.00000)(t-3)^3 + (1.53361)(4-t) + (0.00000)(t-3) & t \in [3,4] \\ (-0.00000)(5-t)^3 + (0.32650)(t-4)^3 + (0.00000)(5-t) + (-1.53361)(t-4) & t \in [4,5] \\ (0.32650)(6-t)^3 + (0.10819)(t-5)^3 + (-1.53361)(6-t) + (-1.10819)(t-5) & t \in [5,6] \\ (0.10819)(7-t)^3 + (-0.17350)(t-6)^3 + (-1.10819)(7-t) + (-0.03361)(t-6) & t \in [6,7] \\ (-0.17350)(8-t)^3 + (0.00000)(t-7)^3 + (-0.03361)(8-t) + (-0.00000)(t-7) & t \in [7,8] \end{cases}$$

（我采用自然边界条件，即样条函数在两个端点的二阶导数取为零。）

2.3 样条内插曲线与原曲线作图



2.4 为什么这个算法可以平滑地连接所有的点

该算法本身使得样条满足插值点约束，每一段都为三次多项式，且各段连接处的一阶导和二阶导都连续。由最小模定理，三次样条插值函数是使得势能（一种模）：

$$||f|| = \int_a^b |f''(x)|\mathrm{d}x$$

最小，在某种意义上是过插值节点的最光滑的函数。

另外，$S_\Delta(X;t)$与$S_\Delta(Y;t)$都是对t的C^2的函数，则$X \to Y$一般也是C^2的：

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = \frac{\mathrm{d}}{\mathrm{d}x}(\frac{\mathrm{d}y}{\mathrm{d}x}) = \frac{\mathrm{d}}{\mathrm{d}x}(\frac{\mathrm{d}y/\mathrm{d}t}{\mathrm{d}x/\mathrm{d}t}) = \frac{\mathrm{d}}{\mathrm{d}x}(\frac{y'_t}{x'_t}) = \frac{y''_t}{(x'_t)^2} - \frac{y'_t x''_t}{(x'_t)^3}$$

仍然是连续的。

3 含有zeta 函数的方程求解

这部分代码提交在文件“ZetaFunction.py”中。

3.1 分析，对于参量$q^2 \in (0,3)$，如果要求zeta 函数$Z_{00}(1;q^2)$ 的精度达到6位或12位有效数字，那么计算公式(6) 中的无穷求和分别至少应保留多少项？

当l和m都取0时，第二项为常数值$(-\pi)$，第三项亦为与$\mathbf{n}$无关的数，可用泰勒级数展开：

$$\frac{\pi}{2}\int_0^1 \mathrm{d}t t^{-3/2}(e^{tq^2}-1) = \frac{\pi}{2}\sum_{i=1}^{\infty}\frac{(q^2)^i}{(i-1/2)i!}$$

实际上只需要求和有限多项（当某一项绝对值小于ϵ时，舍弃余项）。和科学计算软件Mathematica比对可知这一项的计算是非常精确的。图中给出了第三项R_3随q^2的变化关系，R_3的数量级在$[0,20]$。

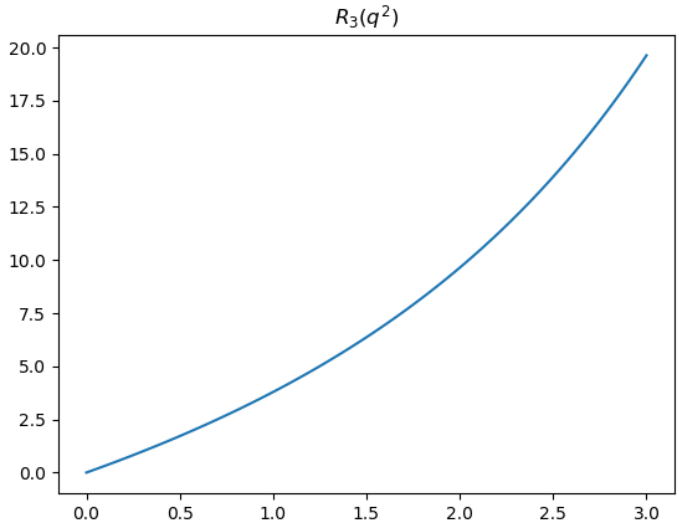


Figure 5: 第三项的值

包含无穷求和的只有第一项和第四项。先将$\mathbf{n}^2$从小到大排序（程序中的SortLattice函数）。先看第四项$R4$，采用Romberg外推法积分（其一，被积函数满足使用条件；其二，达到相同精度所需时间远远小于复合辛普森公式），得到了第四项R_4随q^2的变化关系，R_4的数量级在$[0,0.0005]$。

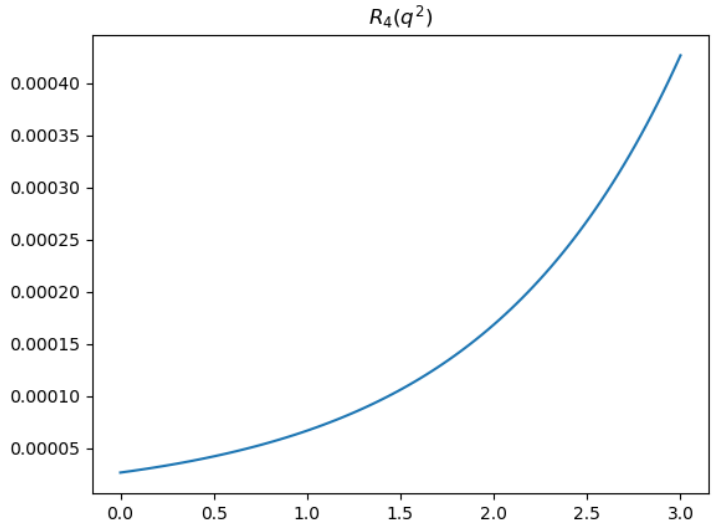


Figure 6: 第四项的值

实际上第四项随$\mathbf{n}^2$的增加收敛非常快，如果求和采用判停条件：

$$|\pi y_{00} \int_0^\infty \mathrm{d}t t^{-3/2} e^{tq^2-(\pi^2/t)\mathbf{n}^2}| < \epsilon$$

计算得到，若ϵ取10^{-6}，当$\mathbf{n}^2 \geq 2$时，余下的项可以忽略；若ϵ取10^{-12}，当$\mathbf{n}^2 \geq 3$时，余下的项可以忽略。

再看第一项求和。显然当q^2趋于整数点，如$\{0,1,2,3\}$时，求和中有一项分母趋于零，是发散的。图像给出R_1随q^2的变化关系。

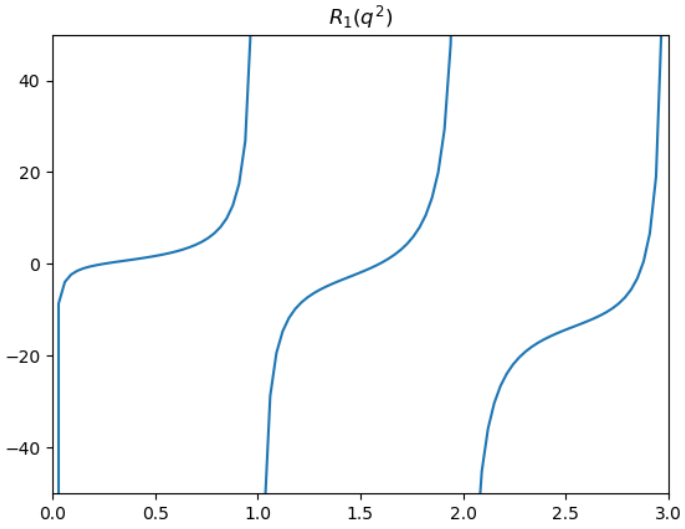


Figure 7: 第一项的值

采用判停条件：

$$|y_{00}\frac{e^{q^2-\mathbf{n}^{*2}}}{\mathbf{n}^{*2}-q^2}| < \max\left(1, \sum_{\mathbf{n}^2<\mathbf{n}^{*2}} y_{00}\frac{e^{q^2-\mathbf{n}^2}}{\mathbf{n}^2-q^2}\right)\cdot\epsilon$$

即当该项的绝对值或与求和式的相对值小于ϵ时忽略剩余的项。考虑到当求和式值很大时（如q^2接近整数点时），$\mathbf{n}^2$与q^2接近的这一项尤其大，其他项与之相比都可忽略，这时判停条件对应于相对值小于ϵ；而当q^2取某些值使得Zeta函数较小（甚至于趋于0）时，这时判停条件对应于绝对值小于ϵ。

计算得到，若ϵ取10^{-6}，对于大多数点当$\mathbf{n}^2 \geq 12$时余下的项可以忽略（相对值判据），对于接近零点的点当$\mathbf{n}^2 \geq 16$时余下的项可以忽略（绝对值判据）；若ϵ取10^{-12}，对于大多数点当$\mathbf{n}^2 \geq 25$时余下的项可以忽略（相对值判据），对于接近零点的点当$\mathbf{n}^2 \geq 30$时余下的项可以忽略（绝对值判据）。

3.2 求解方程

使用二分法求解方程。得到:

$$q^2 \approx 0.794516$$