数值分析第六次上机作业

宋歌 2015080086 数 52

12/30/2018

试用不同数值积分方法计算 $I(f)=\int_1^3 f(x)dx$ 的近似值，其中 $f(x)=\frac{1}{x^2}\sin\frac{2\pi}{x}$.

1 实验原理

1.1 Gauss-Legendre 求积公式

若求积公式 $I_n(f)=\sum_{k=0}^{n} A_k f(x_k)$ 的代数精度达到 $2n+1$，我们称其为高斯型求积公式。已知高斯型求积公式需让求积节点 $x_0, x_1, \ldots, x_n$ 是 $[a,b]$ 上权函数为 ρ 的 $n+1$ 次正交多项式 $\varphi_{n+1}(x)$ 的零点，且有

$$A_j=-\frac{a_{n+2}}{a_{n+1}}\frac{\sigma_{n+1}}{\varphi_{n+2}(x_j)\varphi'_{n+1}(x_j)},\quad j=0,\ldots,n,\quad \sigma_k=(\varphi_k,\varphi_k)$$

现取 $[a,b]=[-1,1]$ 上权 $\rho=1$ 的 Legendre 正交多项式：

$$P_0(x)=1,\quad P_n(x)=\frac{1}{n!2^n}\frac{d^n}{dx^n}[(x^2-1)^n],\quad n\geqslant 1$$

代入上述公式可得到相应的

$$A_k=\frac{2}{n+2}\frac{1}{P_n(x_k)P'_{n+1}(x_k)}$$

可以证明，$\forall f\in C[a,b]$，对于 Gauss 型求积公式有 $\lim_{n\to\infty} I_n(f)=I(f)$.

1.2 Romberg 求积算法

我们有 Richardson 外推定理：若 $\varphi(h)$ 在 $h\to 0$ 时收敛到 $\varphi(0)=\varphi^*$，余项写作 $\varphi^*-\varphi(h)=\sum_{k=1}^{+\infty} a_k h^{p_k}, 0<p_1<p_2<\ldots$. 其中 p_k, a_k 与 h 无关，$a_k\neq 0$. 取 $q\in(0,1)$，定义新序列：

$$\varphi_1(h)=\varphi(h),\varphi_{m+1}(h)=\frac{\varphi_m(qh)-q^{p_m}\varphi_m(h)}{1-q^{p_m}},\quad m=1,2,\ldots$$

则 $\{\varphi_m(h)\}$ 以更快的速度收敛到 φ^*.

将上述 Richardson 外推思想与等距网格减半加密技术结合起来就得到了以下的 Romberg 求积算法：

- 重复利用梯形公式：$h_j=\frac{b-a}{2^j}$，即减半加密网格：

$$\begin{aligned}
T_1^{(0)}&=T_{2^0}=\frac{b-a}{2}[f(a)+f(b)],\\
T_1^{(1)}&=T_{2^1}=\frac{1}{2}[T_1^{(0)}+h_0 f(\frac{a+b}{2})],\\
&\vdots\\
T_1^{(k)}&=T_{2^k}=\frac{1}{2}[T_1^{(k-1)}+h_{k-1}H_{k-1}].
\end{aligned}$$

其中

$$H_j = \sum_{l=1}^{2^j} f\left(a + (l - \frac{1}{2})h_j\right)$$

- 利用 Richardson 思想加速：

$$T_{j+1}^{(k-1)} = \frac{4^j T_j^{(k)} - T_j^{(k-1)}}{4^j - 1}, \quad j = 1, 2, \ldots, \quad k = 1, 2, \ldots$$

- 当迭代到 $|T_j^{(0)} - T_{j+1}^{(0)}| < \varepsilon$ 时终止加密迭代，输出 $T_{j+1}^{(0)}$ 作为 $I(f)$ 的近似。

2 实验设计

2.1 Gauss-Legendre 求积公式

先构造求直到 n 阶的 Legendre 正交多项式 $\{P_k(x)\}_{k=1}^{n}$ 的函数 Legendre(n). 如图 1，先写出易求得的

$$P_0(x) = 1, P_1(x) = x, P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$$

然后利用递推公式 $(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$ 求得直到 n 阶的 Legendre 多项式。最终返回的函数值储存了 1 至 n 阶的形式上的 Legendre 多项式。

```
function [P] = Legendre(n)
p0 = 1;
p{1} = poly(0);
p{2} = [3/2 0 -1/2];

for k = 3:n
    p{k} = sym2poly(poly2sym(((2*k-1)/k) .* conv(poly(0),p{k-1})) ...
        - poly2sym(((k-1)/k) .* p{k-2}));
end

for j = 1:n
    P(j) = poly2sym(p{j});
end
end
```

Figure 1: 求出直到 n 阶 Legendre 正交多项式的 MATLAB 函数

接下来构造用 $n+1$ 阶 Legendre 多项式求 f 在 $[a,b]$ 上积分的函数 GLint(f,a,b,n). 由于 Legendre 多项式是 $[-1,1]$ 上的正交多项式，故对于积分区间为 $[a.b]$ 的积分需要先作变量替换 $x = \frac{a+b}{2} + \frac{b-a}{2}t$:

$$\int_a^b f(x)dx = \int_{-1}^{1} f(\frac{a+b}{2} + \frac{b-a}{2}t)\frac{b-a}{2}dt = \frac{b-a}{2}\int_{-1}^{1} g(t)dt$$

其中

$$g(t) = f(\frac{a+b}{2} + \frac{b-a}{2}t) = f(x)$$

如图 2，先利用之前构造的函数 Legendre(n) 求出直至 $n+1$ 阶的 Legendre 多项式 P，取出需要用到的 P_n 与 P'_{n+1}，将多项式 P_{n+1} 的根 $t = [t_0, t_1, \ldots, t_n]$ 作为积分节点。由公式

$$A_k = \frac{2}{n+2} \frac{1}{P_n(t_k) P'_{n+1}(t_k)}$$

求出求积公式中的系数，并求出与 $t(k)$ 相应的变量替换后的 $x(k)$. 从而可以得到数值积分

$$I_n(f) = \frac{b-a}{2} \int_{-1}^{1} g(t)dt = \frac{b-a}{2} \sum_{k=0}^{n} A_k g(t_k) = \frac{b-a}{2} \sum_{k=0}^{n} A_k f(x_k)$$

```
function [I] = GLint(f,a,b,n)
h = 2/n; t0 = -1; x0 = a;
P = Legendre(n+1);
Q1 = sym2poly(P(n));
Q2 = sym2poly(diff(P(n+1)));
A0 = 2/((n+1) * polyval(Q1,t0) * polyval(Q2,t0));
t = roots(sym2poly(P(n+1)));t = t(2:n+1);
for k = 1:n
    x(k) = (a+b)/2 + ((b-a)*t(k))/2;
    A(k) = 2/((n+1) * polyval(Q1,t(k)) * polyval(Q2,t(k)));
end
I = A0 * f(x0);
for k = 1:n
    I = I + A(k)*f(x(k));
end
I = I * (b-a) / 2;
end
```

Figure 2: 用 Gauss-Legendre 方法对函数进行数值积分的 MATLAB 函数

如此，只需输入被积函数 f，积分区间 $[a,b]$ 及将该区间分成的份数 n 便可输出相应的用 Gauss-Legendre 方法得到的数值积分值。

2.2 Romberg 求积算法

如图 3，在找到使得 $|T_k^{(0)} - T_{k-1}^{(0)}| < \text{tol}$ 的 $T_k^{(0)}$ 之前，我们需要不断地求出新的 $T_k^{(0)}$. 已知

$$T_k^{(0)} = \frac{4^{k-1} T_{k-1}^{(1)} - T_{k-1}^{(0)}}{4^{k-1} - 1}$$

其中 $T_{k-1}^{(0)}$ 是上一步已知的，即需要求出 $T_{k-1}^{(1)}$，从而在第 k 步需要新计算的项为 $T_j^{(k-j)}, j=1,2,\ldots,k-1$. 其中 $T_1^{(k-1)}$ 可以由以下递推公式求得

$$T_1^{(k-1)} = T_{2^{k-1}} = \frac{1}{2}[T_1^{(k-2)} + h_{k-2}H_{k-2}]$$

其余项可由以下公式求得

$$T_j^{(m)} = \frac{4^{j-1}T_{j-1}^{(m+1)} - T_{j-1}^{(m)}}{4^{j-1}-1}$$

```
function [I] = Romberg(f,a,b,tol)
k = 1; err = 1;
T(1,1) = ((b-a)/2) * (f(a)+f(b));
while err >= tol
    k = k + 1; n = 2^(k-1); h = (b-a)/n; H = f((a+b)/2);
    for i = 1:(n/2)
        H = H + f(a + (2*i - 1) * h);
    end
    T(k,1) = (T(k-1,1) + 2*h*H) / 2;
    for j = 2:k
        T(k+1-j,j) = (4^(j-1)*T(k+2-j,j-1) - T(k+1-j,j-1)) / (4^(j-1) - 1);
    end
    err = abs(T(k,1) - T(k-1,1));
end
I = T(k,1);
end
```

Figure 3: 用 Romberg 求积算法对函数进行数值积分的 MATLAB 函数

3 结果与分析

按照题目要求令函数 $f(x) = \frac{1}{x^2}\sin\frac{2\pi}{x}$，积分区间 $[a,b]=[1,3]$。Gauss-Legendre 方法中 $n=4$，Romberg 求积算法中 $\mathrm{tol}=10^{-7}$. 即运行如下代码。

```
f = @(x)sin((2*pi)./x) ./ x.^2;
a = 1; b = 3; n = 4; tol = 1e-7;
I_GL = GLint(f,a,b,n);
I_R = Romberg(f,a,b,tol);
```

所得结果 $I_{GL} = -0.2376, I_R = -0.2387$. 与给出的精确值 $I(f) = -0.238732414\ldots$ 相比，Gauss-Legendre 方法的相对误差约为 0.47%，Romberg 求积算法的相对误差约为 0.013%. 可见 Romberg 求积算法比 Legendre 方法精度高很多，即 Richardson 外推大大提高了序列收敛速度。