Algorithm 4: Romberg Integration

|  |
| --- |
| Method introduction: |
| Using    the method can be inductively defined by    or    where n>=m, m>=1.  The zeroeth extrapolation, R(n, 0), is equivalent to the trapezoidal rule with 2n + 1 points; the first extrapolation, R(n, 1), is equivalent to Simpson's rule with 2n + 1 points. The second extrapolation, R(n, 2), is equivalent to Boole's rule with 2n + 1 points. Further extrapolations differ from Newton Cotes formulas. In particular further Romberg extrapolations expand on Boole's rule in very slight ways, modifying weights into ratios similar as in Boole's rule. In contrast, further Newton Cotes methods produce increasingly differing weights, eventually leading to large positive and negative weights. This is indicative of how large degree interpolating polynomial Newton Cotes methods fail to converge for many integrals, while Romberg integration is more stable. |
| Algorithm Design |
| To estimate the area under a curve the trapezoid rule is applied first to one-piece, then two, then four, and so on. After trapezoid rule estimates are obtained, Richardson extrapolation is applied.   * For the first iteration the two piece and one piece estimates are used in the formula (4 × (more accurate) − (less accurate))/3 The same formula is then used to compare the four piece and the two piece estimate, and likewise for the higher estimates * For the second iteration the values of the first iteration are used in the formula (16(more accurate) − less accurate)/15 * The third iteration uses the next power of 4: (64 (more accurate) − less accurate)/63 on the values derived by the second iteration. * The pattern is continued until there is one estimate. |
| Matlab code |
| function I = RombergInterg(fun, a, b, npanel, tol, flag)  % RombergInterg 用Romberg方法求积分  %  % Synopsis: I = RombergInteg(fun,a,b,n,tol)  %  % Input: fun = (string) 被积函数的函数名  % a, b = 积分下限和积分上限  % npanel = (optional) 将积分区间平分的段数，默认为25  % tol = (optional) 计算误差上限，默认为5e-9  % flag = (optional) 是否显示Romberg-T数表，默认为0为不显示  %  % Output: I = 通过Romberg方法求积分的近似值  if nargin < 4  npanel = 25;  end  if nargin < 5  tol = 5e-9;  end    if nargin < 6  flag = 0;  end    T(1,1) = TrapezoidInteg(fun, a, b, npanel); %T0(h) = T(h)  err = 1; %初始化误差值  m = 2; %初始化行计算的行序号    while err >= tol  T(m,1) = TrapezoidInteg(fun, a, b, 2^m\*npanel); %计算第m行T0(h/2^m)  T(m,m) = 0; %将矩阵T变成m\*m  for n = 2:m  T(m,n) = ( 4^(n-1)\*T(m,n-1) - T(m-1,n-1)) / (4^(n-1) - 1); %Tm(h/2^k)与Tm-1(h/2^(k+1))和Tm-1(h/2^k)的递推关系  end  err = abs( T(m,m) - T(m-1,m-1) ); %计算误差值  m = m + 1; %计算下一行  end    I = T(m-1,m-1);    if flag ~= 0  disp(T);  end  function I = TrapezoidInteg(fun, a, b, npanel)  % TrapezoidInteg 用复化梯形公式求积分  %  % Synopsis: I = TrapezoidInteg(fun,a,b,n)  %  % Input: fun = (string) 被积函数的函数名  % a, b = 积分下限和积分上限  % npanel = (optional) 将积分区间平分的段数，默认为25  %  % Output: I = 通过复化梯形公式求积分的近似值  if nargin < 4  npanel = 25;  end  nnode = npanel + 1; %节点数 = 段数 + 1  h = (b-a)/(nnode-1); %步长  x = a:h:b; %将积分区间分段  f = feval(fun,x); %求节点处被积函数的值  I = h \* ( 0.5\*f(1) + sum(f(2:nnode-1)) + 0.5\*f(nnode) ); |
| Examples and Result |
| Use Romberg method to calculate based on, Compare with exact solution .   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | n | Tn | Sn | Cn | Rn | On | Bn | | 2 | 3.115048238642856 | 0 | 0 | 0 | 0 |  | | 4 | 3.138268511098500 | 3.146008601917048 | 0 | 0 | 0 | 0 | | 8 | 3.140417031779047 | 3.141133205339229 | 3.140808178900707 | 0 | 0 | 0 | | 16 | 3.141176944839529 | 3.141430249193023 | 3.141450052116610 | 3.141460240580354 | 3.141460240580354 | 0 | | 32 | 3.141445667092197 | 3.141535241176419 | 3.141542240641979 | 3.141543703951906 | 3.141544031259245 | 3.141445667092197 | | 64 | 3.141540684024472 | 3.141572356335230 | 3.141574830679151 | 3.141575347981329 | 3.141575472075562 | 3.141540684024472 |   Remarks |
| 此处写该方法程序设计的一些注意事项，也可以空白 |
|  |