

```

[> restart;
[>
[> romberg := proc(f::algebraic,a::numeric,b::numeric,n::nonnegint,
  tol::nonnegative,S::name)
  local F, OK, A, B, N, H, R, I, SUM, M, K, J, L, TOL;
[> F := unapply(f,x);
[> A := a;
[> B := b;
[> if A > B then
[> ERROR("The lower limit of integration must be less than the upper.
  ");
[> fi;
[> N:=n;
  if N=0 then N:=1000 fi;
  TOL:=tol;
STEP 1
[> H := B-A;
[> R[1,1] := evalf((F(A)+F(B))/2*H);
  printf(`\nR[1,1] = %20.15f\n`,R[1,1]);
STEP 2
[> OK:=0;
  I:=2;
STEP 3
[> while I<=N and OK<2 do
STEP 4
approximation from Trapezoidal method
[> SUM := 0;
[> M := 2^(I-2);
[> for K from 1 to M do
[> SUM := SUM+F(A+(K-0.5)*H);
[> od;
[> R[I,1] := evalf((R[I-1,1]+H*SUM)/2);
  printf(`R[%d,1] = %20.15f\n`,I,R[I,1]);
STEP 5
extrapolation
[> for J from 2 to I do
[> L := 2^(2*(J-1));
[> R[I,J] := evalf(R[I,J-1]+(R[I,J-1]-R[I-1,J-1])/(L-1));
  printf(`R[%d,%d] = %20.15f\n`,I,J,R[I,J]);
[> od;
STEP 6
[> H := H/2;
  if abs(R[I,I]-R[I-1,I-1])<TOL then
    OK:=OK+1
  else
    OK:=0
  fi;
  I:=I+1;
[> od;
  M:=I-1;
  L:=iquo(M,5) + 1;
  if irem(M,5) = 0 then L:=L-1 fi;
  printf(`\nRomberg Integration Table:\n\n`);
  for J from 1 to L do
    for I from 5*J-4 to M do
      for K from 5*J-4 to min(5*J,I) do
[> printf(` %14.8f`,R[I,K]);
[> od;
[> printf(`\n\n`);

```

```

od;
printf(`\n\n`);
od;
S:=R[M,M];
> end;
Warning, imaginary unit `I` used as a local variable in procedure romberg
romberg := proc(f::algebraic, a::numeric, b::numeric, n::nonnegint, tol::nonnegative, S::name)

```

```

    local F, OK, A, B, N, H, R, I, SUM, M, K, J, L, TOL;

```

```

    F := unapply(f, x);

```

```

    A := a;

```

```

    B := b;

```

```

    if B < A then

```

```

        ERROR("The lower limit of integration must be less than the upper.")

```

```

    end if;

```

```

    N := n;

```

```

    if N = 0 then

```

```

        N := 1000

```

```

    end if;

```

```

    TOL := tol;

```

```

    H := B - A;

```

```

    R[1, 1] := evalf(1/2 * (F(A) + F(B)) * H);

```

```

    printf(`

```

```

        R[1,1] = %20.15f

```

```

        `, R[1, 1]);

```

```

    OK := 0;

```

```

    I := 2;

```

```

    while I <= N and OK < 2 do

```

```

        SUM := 0;

```

```

        M := 2^(I - 2);

```

```

        for K to M do

```

```

            SUM := SUM + F(A + (K - 0.5) * H)

```

```

        end do;

```

```

        R[I, 1] := evalf(1/2 * R[I - 1, 1] + 1/2 * H * SUM);

```

```

        printf(`R[%d,1] = %20.15f

```

```

            `, I, R[I, 1]);

```

```

for J from 2 to I do
    L := 2^(2 * J - 2);
    R[I, J] := evalf(R[I, J - 1] + (R[I, J - 1] - R[I - 1, J - 1]) / (L - 1));
    printf(`R[%d,%d] = %20.15f`
           ` , I, J, R[I, J])
end do;
H := 1/2 * H;
if abs(R[I, I] - R[I - 1, I - 1]) < TOL then
    OK := OK + 1
else
    OK := 0
end if;
I := I + 1
end do;
M := I - 1;
L := iquo(M, 5) + 1;
if irem(M, 5) = 0 then
    L := L - 1
end if;
printf(`
    Romberg Integration Table:

`);
for J to L do
    for I from 5 * J - 4 to M do
        for K from 5 * J - 4 to min(5 * J, I) do
            printf(`%14.8f`, R[I, K])
        end do;
        printf(
            )
    end do;

```

```

    printf(
        )
end do;
S := R[M, M]
end proc
>
> romberg_dir:=proc()
printf(`romberg returns an approximation to a definite integral.
\n\n`);
printf(`The arguments for romberg are:\n`);
printf(`(1)the function being integrated\n`);
printf(`(2)the lower limit of integration\n`);
printf(`(3)the upper limit of integration\n`);
printf(`(4)number of rows (0 if stopping on a tolerance)\n`);
printf(`(5)tolerance (0 if choosing a number of rows) \n`);
printf(`(6)variable for returning the integral
approximation\n\n`);
printf(`If assigning the result to a variable, have the\n`);
printf(`variable and the 6th argument the same.\n\n`);
printf(`If S is the variable for returning the integral's
value\n`);
printf(`and has already been given a value,\n`);
printf(`the procedure should be preceded by the statement:\n`);
printf(`S:='S'`);
end;
romberg_dir := proc()

    printf(`romberg returns an approximation to a definite integral.

    `);

    printf(`The arguments for romberg are:

    `);

    printf(`(1)the function being integrated

    `);

    printf(`(2)the lower limit of integration

    `);

    printf(`(3)the upper limit of integration

    `);

    printf(`(4)number of rows (0 if stopping on a tolerance)

    `);

    printf(`(5)tolerance (0 if choosing a number of rows)

    `);

```

printf(`(6)variable for returning the integral approximation

`);

printf(`If assigning the result to a variable, have the

`);

printf(`variable and the 6th argument the same.

`);

printf(`If S is the variable for returning the integral's value

`);

printf(`and has already been given a value,

`);

printf(`the procedure should be preceded by the statement:

`);

printf(`S:='S`)

end proc

[>
[>