Mixing Numerical Integration Rules

By
Namir Shammas

Introduction

Numerical integration for single integral comes in two general flavors. You either work with an array of data points or with analytical functions. This article focuses on the latter case. Let me start by quickly going over some of the basic numerical integration algorithms that I want to include. They are:

- The Trapezoidal rule
- The Simpson's rule
- The Simpson's 3/8 rule.
- Boole's rule.

Trapezoidal Rule

This is the simplest algorithm and the least accurate. The method divides the area under the curves into small trapezoids, calculates the area of each trapezoid, and then adds these areas.

Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_1} f(x)dx = \frac{\mathrm{h}}{2}(f(x_0) + f(x_1)) \qquad (1)$$

Where h is basically the Δx value used to sample the values for independent variable x.

The following equation shows the extended or composite version:

$$\int_{x_0}^{x_1} f(x)dx = \frac{\mathrm{h}}{2}(f(x_0) + 2f(x_0 + \mathrm{h}) + 2f(x_0 + 2\mathrm{h}) + \cdots + 2f(x_0 + nh) + f(x_1)) \qquad (1b)$$

Simpson's Rule

Simpson's rule is an algorithm that is based on quadratic interpolation used to enhance the calculations of the integral.

Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_2} f(x)dx = \frac{\mathrm{h}}{3}(f(x_0) + 2f(x_1) + f(x_2))$$
(2)

The following equation shows the extended or composite version:

$$\int_{x_0}^{x_n} f(x)dx = \frac{\mathrm{h}}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 4f(x_{n-2}) + 2f(x_{n-1}) + f(x_n))$$
(2b)

Simpson's 3/8 Rule

Simpson's rule is an algorithm that is based on cubic interpolation used to enhance the calculations of the integral.

Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_3} f(x)dx = \frac{3\mathrm{h}}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3))$$
(3)

The following equation shows the extended or composite version:

$$\int_{x_0}^{x_n} f(x)dx = \frac{3\mathrm{h}}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + 2f(x_1) + \cdots + 3f(x_{n-2}) + 3f(x_{n-1}) + f(x_n))$$
(3b)

Boole's Rule

Boole's rule (also known as Bode's rule due to a historical typo) is an algorithm that is based on fifth-order polynomial interpolation used to enhance the calculations of the integral.

Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_4} f(x)dx = \frac{\mathrm{h}}{45}(14f(x_0) + 64f(x_1) + 24f(x_2) + 64f(x_3) + 14f(x_4))$$
(4)

A Second Look at the Trapezoidal Rule

I took a second look at the Trapezoidal rule. Equation 1 shows that each area calculates the mean of the function at two points that define the small integration

intervals. How can we enhance that average for the function value in hope to make the calculations more accurate? The approach I used is as follow:

1. Add one or more points inside each integration interval and study the effect of that change on the result.
2. Assign weight values to the points that I add inside the integration intervals. By altering these weights I hope to find an optimum value.

Adding One Additional Point

In the case of adding one additional point inside each integration interval, I calculate the average of the function at three points instead of two. Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}(f(x_0) + f(x_0 + h/2) + f(x_1)) \qquad (5)$$

Here is the version of the equation that uses a weight for the internal point:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}(f(x_0) + w\, f(x_0 + h/2) + f(x_1)) \qquad (5b)$$

You can regard equation (5) as a special case of equation (5b) where the weight w is 1.

Adding Two Additional Points

In the case of adding two additional points inside each integration interval, I calculate the average of the function at four points instead of two. Here the equation that represents the basic algorithm:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}(f(x_0) + f(x_0 + h/3) + f(x_0 + 2h/3) + f(x_1)) \qquad (6)$$

Here is the version of the equation that uses a weight for the internal point:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}(f(x_0) + w\, f(x_0 + h/3) + w\, f(x_0 + 2h/3) + f(x_1)) \qquad (6b)$$

You can regard equation (6) as a special case of equation (6b) where the weight w is 1.

Other Algorithmic Modifications

I also tried different schemes, such as:

- Selecting random multiple points inside each integration interval.
- Selecting ten points inside each integration interval.

Neither approach gave worthy results.

Testing the Changes

I tested equations 5b and 6b against equations 1b and 2b for the following cases:

1. Integral of $\ln(x)$ for x_0 = 1 and x_n = 10.
2. Integral of $1/x$ for x_0 = 1 and x_n = 10.
3. Integral of x^3 for x_0 = 1 and x_n = 2.
4. Integral of $\sin(x)$ for x_0 = 0 and x_n = 0.5.
5. Integral of $x \ln(x)$ for x_0 = 1 and x_n = 5.

I calculated the areas using equation 1b, 2b, 5b, and 6b. I also obtained the exact integral and then calculated the error for each numerical method using 50 divisions for each integral.

Table 1 shows the ratio of the errors using equation 1b divided by the errors using 5b, for different weight values.

Weight	*Error Ratio*
1	2
2	4
3	10
4	High and varies
5	-14

Table 1. Results for comparing the error ratios between equations 2b and 5b.

Table 1 shows the ratio of the errors using equation 1b divided by the errors using 6b, for different weight values.

Weight	*Error Ratio*
1	3
2	9
3	High and varies
4	-15

Table 2. Results for comparing the error ratios between equations 2b and 6b.

Table 1 shows reduction is error as the weight increases from 1 to 4. The value of 4 seems optimum and allows equation 5b to produce errors that are even less than Simpson's rule in equation 2b. In fact, when I double the number of division used in Simpson's rule, I get errors that are very close to what equation 5b yields. When the weight is not 4, the results of Simpson's rule are much more accurate by one or more orders of magnitudes.

Table 2 shows reduction is error as the weight increases from 1 to 3. The value of 3 seems optimum and allows equation 6b to produce errors that are less than Simpson's rule in equation 3b—Simpson's 3/8 rule. When I double the number of intervals used in Simpson's rule, I get a relative error of about 2.2. When the weight is not 3, the results of Simpson's rule are much more accurate by one or more orders of magnitudes.

The values in tables 1 and 2 will show some moderate variations when changing (especially reducing) the number of integration intervals. You can think of the tabulated ratios as the values of limits in calculus.

The conclusion we can draw here that using the optimum weights of 4 and 3 in equations 5b and 6b, respectively, transforms the variant of the Trapezoidal algorithm into a Simpson's rule and Simpson's 3/8 rule! Thus, when the weight in equation 5b is 4, that equation matches the accuracy of equation 2b (with double the number of intervals). Therefore, we can use the Trapezoidal rule as a basic frame for the numeric integration. Within the integration intervals we can then use additional points with weights that correspond to weights in equations 2b, and 3b.

When I apply Boole's rule in equation 4 to my approach, I use three additional internal points and assign the array of weights of 14, 64, 24, 64, and 14 (leaving out the divider 45) that appear in equation 4 to the various function values within

each integration interval. The results are the most accurate, among the group of methods considered, in calculating the integrals.

There is a very interesting consequence of using the Trapezoidal method as an umbrella method for other integration methods. This combination allows you select virtually any number of intervals without worrying if that number is odd, even, or if it needs to comply with a specific numeric sequence. The intervals of the Trapezoidal rule act as an outer shell. Within that shell you can divide each interval as though it is the only one you are using! This approach makes it very simple to implement more efficient integration rules, since the points used by these rules are compartmentalized. Figure 1 depicts the scheme I am discussing and shows and example of using Boole's rule inside a single interval.

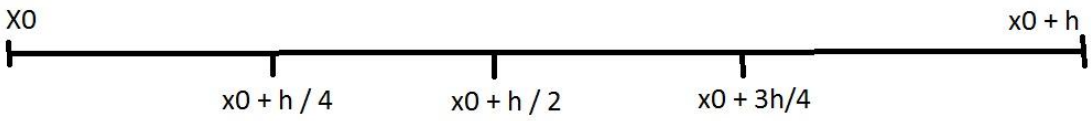

Figure 1. Using Bode's rule inside a single interval.

Of course using more points inside each integration interval adds more computing effort. You can compensate for that extra effort by moderately reducing the number of integration intervals without losing much accuracy.

Sample Code

In this section I present functions coded in Excel VBA. Most of these VBA functions calculate the numerical integral using methods discussed in this article. I have coded the functions to emphasize clarity of code and not implementation efficiency. When you study the code, notice how the VBA functions *ThreePoints*, *FourPoints*, and *FivePoints* implement equations 4b, 5b, and 6b, respectively. When the optimum arguments for the weight parameters are supplied to these functions, they work *as though* they are implementing Simpson's rule, Simpson's 3/8 rule, and Boole's rule, respectively, and they do so in a very subtle way.

The VBA functions appear next:

```
Function Fx(ByVal sFx As String, ByVal X As Double) As Double
' Evaluates string expression to contain variable X.
' The variable X must appear as $X
  sFx = UCase(Replace(sFx, " ", ""))
  If X >= 0 Then
    sFx = Replace(sFx, "$X", CStr(X))
```

```
  Else
    sFx = Replace(sFx, "$X", "(" & CStr(X) & ")")
  End If
  Fx = Evaluate(sFx)
End Function

Function Trapezoidal(ByVal sFx As String, ByVal A As Double, _
                     ByVal B As Double, ByVal N As Integer) As Double
' Calculates the area under the curve using the Trapezoidal rule.
'
' Parameters:
'
' sFx - String expression that has the function to be
' integrated. The variable X as to appear as $X.
' An example is: $X*LOG(X$) which is an expression for function
' f(x)=x*ln(x)
' A, B - Lower and Upper limit for the integral.
' N - The number of integration intervals.
'
  Dim Sum As Double, DeltaX As Double, X As Double
  Dim I As Integer

  Sum = 0
  DeltaX = (B - A) / N
  X = A
  For I = 1 To N
    Sum = Sum + (Fx(sFx, X) + Fx(sFx, X + DeltaX)) / 2
    X = X + DeltaX
  Next I
  Sum = DeltaX * Sum
  Trapezoidal = Sum
End Function

Function Simpson(ByVal sFx As String, ByVal A As Double, ByVal B As Double, _
                 ByVal N As Integer) As Double
' Calculates the area under the curve using Simpson's rule.
'
' Parameters:
'
' sFx - String expression that has the function to be
' integrated. The variable X as to appear as $X.
' An example is: $X*LOG(X$) which is an expression for function
' f(x)=x*ln(x)
' A, B - Lower and Upper limit for the integral.
' N - The number of integration intervals.
'
  Dim SumOdd As Double, SumEven As Double, DeltaX As Double, X As Double
  Dim Sum As Double
  Dim I As Integer

  ' make sure N is  even
  If (N Mod 2) = 1 Then N = N + 1
  DeltaX = (B - A) / N
  Sum = 0
  X = A
  For I = 1 To N Step 2
    Sum = Sum + Fx(sFx, X) + _
```

```
            4 * Fx(sFx, X + DeltaX) + _
                Fx(sFx, X + 2 * DeltaX)
    X = X + 2 * DeltaX
  Next I
  Simpson = DeltaX / 3 * Sum
End Function


Function ThreePoints(ByVal sFx As String, ByVal A As Double, ByVal B As
Double, _
                     ByVal N As Integer, ByVal Wt As Integer) As Double
' Calculates the area under the curve using the Trapezoidal rule
' with one additional point inside the integration interval.
'
' Parameters:
'
' sFx - String expression that has the function to be
' integrated. The variable X as to appear as $X.
' An example is: $X*LOG(X$) which is an expression for function
' f(x)=x*ln(x)
' A, B - Lower and Upper limit for the integral.
' N - The number of integration intervals.
' Wt - The weight for the function value calculated for the
' additional point inside the integration interval.
'
Dim Sum As Double, DeltaX As Double, X As Double
  Dim I As Integer

  Sum = 0
  DeltaX = (B - A) / N
  X = A
  For I = 1 To N
    Sum = Sum + (Fx(sFx, X) + Wt * Fx(sFx, X + DeltaX / 2) + _
                 Fx(sFx, X + DeltaX)) / (Wt + 2)
    X = X + DeltaX
  Next I
  ThreePoints = DeltaX * Sum
End Function


Function FourPoints(ByVal sFx As String, ByVal A As Double, _
         ByVal B As Double, ByVal N As Integer, _
         ByVal Wt As Integer) As Double
' Calculates the area under the curve using the Trapezoidal rule
' with two additional points inside the integration interval.
'
' Parameters:
'
' sFx - String expression that has the function to be
' integrated. The variable X as to appear as $X.
' An example is: $X*LOG(X$) which is an expression for function
' f(x)=x*ln(x)
' A, B - Lower and Upper limit for the integral.
' N - The number of integration intervals.
' Wt - The weight for the function values calculated for the
' additional points inside the integration interval.
'
```

```
  Dim Sum As Double, DeltaX As Double, X As Double, H As Double
  Dim I As Integer, SumWt As Integer

  Sum = 0
  DeltaX = (B - A) / N
  H = DeltaX / 3
  SumWt = 2 * (1 + Wt)
  X = A
  For I = 1 To N
    Sum = Sum + (Fx(sFx, X) + _
                 Wt * Fx(sFx, X + H) + _
                 Wt * Fx(sFx, X + 2 * H) + _
                 Fx(sFx, X + DeltaX)) / SumWt
    X = X + DeltaX
  Next I
  FourPoints = DeltaX * Sum
End Function

Function FivePoints(ByVal sFx As String, ByVal A As Double, _
                    ByVal B As Double, ByVal N As Integer, _
                    ByRef WtArr() As Integer) As Double
 ' Calculates the area under the curve using the Trapezoidal
' rule with three additional points inside the integration
' interval.
'
' Parameters:
'
' sFx - String expression that has the function to be
' integrated. The variable X as to appear as $X.
' An example is: $X*LOG(X$) which is an expression for function
' f(x)=x*ln(x)
' A, B - Lower and Upper limit for the integral.
' N - The number of integration intervals.
' WtArr - The array of weights for the function values
' calculated for the additional points inside the integration
' interval.
'
  Dim Sum As Double, DeltaX As Double, X As Double
  Dim I As Integer, H As Double
  Dim SumWt As Integer

  Sum = 0
  SumWt = 0
  DeltaX = (B - A) / N
  H = DeltaX / 4
  X = A
  For I = 1 To UBound(WtArr)
    SumWt = SumWt + WtArr(I)
  Next I

  For I = 1 To N
    Sum = Sum + (WtArr(1) * Fx(sFx, X) + _
                 WtArr(2) * Fx(sFx, X + H) + _
                 WtArr(3) * Fx(sFx, X + 2 * H) + _
                 WtArr(4) * Fx(sFx, X + 3 * H) + _
                 WtArr(5) * Fx(sFx, X + DeltaX)) / SumWt
    X = X + DeltaX
```

```
  Next I
  FivePoints = DeltaX * Sum
End Function
```

Summary

This article shows that you can use the Trapezoidal rule with an entirely new twist. Think of the new approach as constructing a multilevel building. You first build the skeleton framework of the building and then fill in each level. The Trapezoidal rule would parallel the skeleton framework of the building. The building levels parallel the integration intervals. Just like the layout of individual level can vary from one another, you apply different (non-extended versions) integration algorithms, such as Simpson's rule, Simpson's 3/8 rule, and Boole's rule to each interval. In fact you can even alternate between these algorithms, applying different ones for each integration interval, if you so desire. This variation is possible because the Trapezoidal rule compartmentalizes the other algorithms as they are applied to the integration intervals.