# Slope-Oriented Numerical Integration Methods

by

Namir Shammas

## Introduction

Numerical integration methods calculate the area under the curve of a function at various values of the independent variable X. Method like Simpson's rule calculate the area at regular intervals of X. By contrast, Gaussian quadrature methods calculate the area at roots of orthogonal polynomials. The general accuracy of the calculated area increases with increasing the number of sampled X values. In the case of methods like Simpson's rule, the smaller increments in X yield higher accuracy in calculating the area. In the case of Gauss quadrature, using higher polynomial orders yields more roots and consequently more accurate results.

This paper presents a new and adaptive approach for numerical integration. I will call this new method the Slope-Oriented Numerical Integration (or SONI for short). The new algorithm focuses on estimating the increment in X such that the change in the value of Y (=f(X)) does not exceed a preselected limit and remains approximately constant. This scheme allows the adaptive method to move fast in small absolute slope sections of f(X) and conversely move slow through high absolute slope values.

## The Basic Algorithm

The basic algorithm is very simple. Here is the pseudo-code for the basic algorithm:

```
Given Y=f(X), we need to integrate from X1 to Xmax, with YMaxDelta maximum
change in Y and max change in X of XMaxDelta:
  • Y1=f(X1)
  • Area = 0
  • Repeat
      o Deriv = f'(X1)
      o Adjust Deriv to be a small value if it is zero or near zero
      o h = 2 * |YMaxDelta / Deriv|
      o Repeat
          ▪ h = h / 2
          ▪ If h > XMaxDelta then h = XMaxDelta
```

```
            ▪ X2 = X1 + h
      o Until |Y2 - Y1| <= YMaxDelta
      o If X2 > Xmax then
            ▪ h = Xmax- X1
            ▪ X2 = Xmax
      o End
      o Area = Area + area of f(X) between X1 and X2 using an algorithm
        of your choice.
      o X1 = X2
      o Y1 = Y2
• Until X2 >= Xmax
```

The above pseudo-code shows that the step that updates the value of the area is a general one. This means that you can use any algorithm to calculate the (partial) area between X1 and X2. You can use the Trapezoidal rule, Simpson's rule, Simpson's $3/8^{th}$ rule, Gaussian Quadrature, and Romberg's method, t name a few. Using more accurate methods yields better final results.

## Testing with Excel VBA Code

I tested the new algorithms using Excel taking advantage of the application's worksheet for easy input and the display of intermediate calculations. The following listing shows the Excel VBA code used for testing. It implements the SONI algorithm use a variety of basic numerical integration methods:

```
Option Explicit

' Version 2.1. Handles slopes that are zero or close to zero1
Const EPSILON = 0.00000001

Function MyFx(ByVal sFx As String, ByVal X As Double) As Double
  sFx = UCase(sFx)
  sFx = Replace(sFx, " ", "")
  sFx = Replace(sFx, "EXP(", "!!")
  sFx = Replace(sFx, "X", "(" & X & ")")
  sFx = Replace(sFx, "!!", "EXP(")
  MyFx = Evaluate(sFx)
End Function

Function MyDx(ByVal sFx As String, ByVal X As Double) As Double
  Dim h As Double, Res As Double, Fxp As Double

  h = 0.001 * (Abs(X) + 1)
  Fxp = MyFx(sFx, X + h)
  Res = (Fxp - MyFx(sFx, X - h)) / 2 / h
  If Abs(Res) < EPSILON Then
    Res = EPSILON * Sgn(Res)
    If Res = 0 Then Res = EPSILON
  End If
  MyDx = Res
End Function
```

```
Function GaussQuad2(ByVal sFx As String, ByVal A As Double, ByVal B As
Double) As Double
  Const ORDER = 2
  Dim Z(ORDER) As Double, Wt(ORDER) As Double, Sum As Double
  Dim I As Integer
  Dim K1 As Double, K2 As Double

  K1 = (B - A) / 2
  K2 = (A + B) / 2
  Wt(1) = 1
  Z(1) = 1 / Sqr(3)
  Wt(2) = 1
  Z(2) = -Z(1)

  Sum = 0
  For I = 1 To ORDER
    Sum = Sum + Wt(I) * MyFx(sFx, K1 * Z(I) + K2)
  Next I
  GaussQuad2 = K1 * Sum
End Function

Function GaussQuad3(ByVal sFx As String, ByVal A As Double, ByVal B As
Double) As Double
  Const ORDER = 3
  Dim Z(ORDER) As Double, Wt(ORDER) As Double, Sum As Double
  Dim I As Integer
  Dim K1 As Double, K2 As Double

  K1 = (B - A) / 2
  K2 = (A + B) / 2
  Z(1) = 0
  Wt(1) = 8 / 9
  Z(2) = Sqr(3 / 5)
  Wt(2) = 5 / 9
  Z(3) = -Z(2)
  Wt(3) = Wt(2)

  Sum = 0
  For I = 1 To ORDER
    Sum = Sum + Wt(I) * MyFx(sFx, K1 * Z(I) + K2)
  Next I
  GaussQuad3 = K1 * Sum
End Function

Function GaussQuad4(ByVal sFx As String, ByVal A As Double, ByVal B As
Double) As Double
  Const ORDER = 4
  Dim Z(ORDER) As Double, Wt(ORDER) As Double, Sum As Double
  Dim I As Integer
  Dim K1 As Double, K2 As Double

  K1 = (B - A) / 2
  K2 = (A + B) / 2
  Z(1) = Sqr((3 - 2 * Sqr(6 / 5)) / 7)
  Wt(1) = (18 + Sqr(30)) / 36
  Z(2) = Sqr((3 + 2 * Sqr(6 / 5)) / 7)
```

```
  Wt(2) = (18 - Sqr(30)) / 36
  Z(3) = -Z(1)
  Wt(3) = Wt(1)
  Z(4) = -Z(2)
  Wt(4) = Wt(2)

  Sum = 0
  For I = 1 To ORDER
    Sum = Sum + Wt(I) * MyFx(sFx, K1 * Z(I) + K2)
  Next I
  GaussQuad4 = K1 * Sum
End Function

Function GaussQuad5(ByVal sFx As String, ByVal A As Double, ByVal B As
Double) As Double
  Const ORDER = 5
  Dim Z(ORDER) As Double, Wt(ORDER) As Double, Sum As Double
  Dim I As Integer
  Dim K1 As Double, K2 As Double

  K1 = (B - A) / 2
  K2 = (A + B) / 2
  Z(1) = 0
  Z(2) = Sqr(5 - 2 * Sqr(10 / 7)) / 3
  Z(3) = Sqr(5 + 2 * Sqr(10 / 7)) / 3
  Z(4) = -Z(2)
  Z(5) = -Z(3)

  Wt(1) = 128 / 225
  Wt(2) = (322 + 13 * Sqr(70)) / 900
  Wt(3) = (322 - 13 * Sqr(70)) / 900
  Wt(4) = Wt(2)
  Wt(5) = Wt(3)

  Sum = 0
  For I = 1 To ORDER
    Sum = Sum + Wt(I) * MyFx(sFx, K1 * Z(I) + K2)
  Next I
  GaussQuad5 = K1 * Sum
End Function

Sub UseTrapezoid()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
```

```
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + h / 2 * (Y1 + Y2)
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D2].Value = "Trapezoid"
  [E2].Value = Sum
End Sub

Sub UseSimpson()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + (X2 - X1) / 6 * (MyFx(sFx, X1) + 4 * MyFx(sFx, (X1 + X2) / 2)
+ MyFx(sFx, X2))
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D3].Value = "Simpson"
  [E3].Value = Sum
End Sub

Sub UseSimpson38()
```

```
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + (X2 - X1) / 8 * (MyFx(sFx, X1) + 3 * MyFx(sFx, (2 * X1 + X2)
/ 3) + 3 * MyFx(sFx, (X1 + 2 * X2) / 3) + MyFx(sFx, X2))
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D4].Value = "Simpson 3/8"
  [E4].Value = Sum
End Sub


Sub UseGauss2()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
```

```
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + GaussQuad2(sFx, X1, X2)
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D5].Value = "Gauss 2"
  [E5].Value = Sum
End Sub

Sub UseGauss3()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + GaussQuad3(sFx, X1, X2)
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D6].Value = "Gauss 3"
  [E6].Value = Sum
End Sub

Sub UseGauss4()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
```

```
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + GaussQuad4(sFx, X1, X2)
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D7].Value = "Gauss 4"
  [E7].Value = Sum
End Sub

Sub UseGauss5()
  Dim Sum As Double
  Dim X1 As Double, Y1 As Double, Xmax As Double
  Dim X2 As Double, Y2 As Double
  Dim h As Double, YMaxDelta As Double, XMaxDelta As Double
  Dim sFx As String

  X1 = [B1].Value
  Xmax = [B2].Value
  YMaxDelta = [B3].Value
  XMaxDelta = [B4].Value
  sFx = [B5].Value
  Y1 = MyFx(sFx, X1)
  Sum = 0
  Do
    DoEvents
    h = 2 * Abs(YMaxDelta / MyDx(sFx, X1))
    Do
      h = h / 2
      If h > XMaxDelta Then h = XMaxDelta
      X2 = X1 + h
      Y2 = MyFx(sFx, X2)
    Loop Until Abs(Y2 - Y1) <= YMaxDelta

    If X2 > Xmax Then
      X2 = Xmax
```

```
      Y2 = MyFx(sFx, X2)
    End If
    Sum = Sum + GaussQuad5(sFx, X1, X2)
    X1 = X2
    Y1 = Y2
  Loop Until X1 >= Xmax
  [D8].Value = "Gauss 5"
  [E8].Value = Sum
End Sub

Sub DoAll()
  [D1].Value = "Method"
  [E1].Value = "Integral"

  UseTrapezoid
  UseSimpson
  UseSimpson38
  UseGauss2
  UseGauss3
  UseGauss4
  UseGauss5
End Sub
```

The VBA function **MyFX** calculates the function value based on a string that contains the function's expression. This expression must use **X** as the variable name. The function **MyDx** calculates the slope of a function.

The subroutine **DoAll** tests the various versions of SONI. These versions tap into the following basic integration methods:

- Trapezoidal rule.
- Simpson's rule.
- Simpson's 3/8th rule.
- Gaussian quadrature of order 2, 3, 4, and 5.

The subroutine DoAll calls other subroutines to test different versions of the SONI methods. These subroutines include **UseTrapezoid**, **UseSimpson**, **UseSimpson38**, **UseGauss2**, **UseGauss3**, **UseGauss4**, and **UseGauss5**. The last four subroutines call functions **GaussQuad2**, **GaussQuad3**, **GaussQuad4**, and **GaussQuad5**, respectively.

Figure 1 shows a sample Excel sheet that contains the input and output data

| | | | | | |
|---|---|---|---|---|---|
| X0 | 1 | | Method | Integral | Error |
| Xmax | 2 | | Trapezoid | 0.699360657 | 0.006213477 |
| Max Delta y | 0.01 | | Simpson | 0.693147181 | 2.8028E-10 |
| Max Delta X | 0.1 | | Simpson 3/8 | 0.693147181 | 1.24572E-10 |
| f(X) | 1/X | | Gauss 2 | 0.69314718 | -1.8685E-10 |
| Exact Integral | 0.693147 | | Gauss 3 | 0.693147181 | -3.3307E-15 |
| | | | Gauss 4 | 0.693147181 | 0 |
| | | | Gauss 5 | 0.693147181 | 0 |

*Figure 1. The Excel spreadsheet used to compare the different versions of the SONI algorithm.*

### The Input Cells

The VBA code relies on the following cells to obtain data:

- Cells B1 and B2 supply the range for integration.
- Cell B3 contains the maximum value in f(X).
- Cell B4 contains the maximum step in X.
- Cell B5 contains the expression for f(X). Notice that the expression in cell B4 use **X** as the variable name. The expression is case insensitive.
- Cell B6 contains the value of the exact integral.

### Output

The output appears in columns D, E, and F. The integrals appear in column E and their associated errors appear in column F.

## The Results

I tested the various versions of the SONI method on different functions, integrals, and maximum increment in f(X) values.

## Integral of 1/X

I calculated the integral of 1/X for different ranges. Figure 2 shows results for the integral from 1 to 2 (equal to ln(2)) and for the maximum change in function value equal to 0.01:

| | | | | | |
|---|---|---|---|---|---|
| X0 | 1 | | Method | Integral | Error |
| Xmax | 2 | | Trapezoid | 0.699360657 | 0.006213477 |
| Max Delta y | 0.01 | | Simpson | 0.693147181 | 2.8028E-10 |
| Max Delta X | 0.1 | | Simpson 3/8 | 0.693147181 | 1.24572E-10 |
| f(X) | 1/X | | Gauss 2 | 0.69314718 | -1.8685E-10 |
| Exact Integral | 0.693147 | | Gauss 3 | 0.693147181 | -3.3307E-15 |
| | | | Gauss 4 | 0.693147181 | 0 |
| | | | Gauss 5 | 0.693147181 | 0 |

*Figure 2. The result for integrating 1/X from 1 to 2.*

As expected, the more advanced functions generated less errors. The Gaussian quadrature of order 4 ad 5 gave exact values in Excel.

For the same integral between 1 and 10 (equal to ln(10)) and a maximum change in function value equal to 0.0001. Figure 3 shows these results.

| | | | | | |
|---|---|---|---|---|---|
| X0 | 1 | | Method | Integral | Error |
| Xmax | 10 | | Trapezoid | 2.3095793 | 0.006994192 |
| Max Delta y | 0.001 | | Simpson | 2.3025851 | 1.99467E-11 |
| Max Delta X | 0.1 | | Simpson 3/ | 2.3025851 | 8.86313E-12 |
| f(X) | 1/X | | Gauss 2 | 2.3025851 | -1.32969E-11 |
| Exact Integral | 2.302585 | | Gauss 3 | 2.3025851 | 0 |
| | | | Gauss 4 | 2.3025851 | 0 |
| | | | Gauss 5 | 2.3025851 | 0 |

*Figure 3. The result for integrating 1/X from 1 to 10.*

Again, the more advanced methods yield better (and accurate) results. In the case of the integral between 1 and 100, Figure 4 shows the results:

| | | | | | |
|---|---|---|---|---|---|
| X0 | 1 | | Method | Integral | Error |
| Xmax | 100 | | Trapezoid | 4.64859732 | 0.043427134 |
| Max Delta y | 0.001 | | Simpson | 4.605170186 | 4.10045E-11 |
| Max Delta X | 0.1 | | Simpson 3/ | 4.605170186 | 1.8229E-11 |
| f(X) | 1/X | | Gauss 2 | 4.605170186 | -2.7329E-11 |
| Exact Integral | 4.60517019 | | Gauss 3 | 4.605170186 | 0 |
| | | | Gauss 4 | 4.605170186 | 0 |
| | | | Gauss 5 | 4.605170186 | 0 |

*Figure 4. The result for integrating 1/X from 1 to 100.*

On the Gaussian quadrature of orders 3, 4, and 5 yield accurate results.

## The $e^{-x}$ sin(x) Integral

Figure 5 shows the results of the integrating the damped oscillating function $e^{-x}$sin(x) between 0 and 5, for a maximum function change value of 0.01:

| | | | | | |
|---|---|---|---|---|---|
| X0 | 0 | | Method | Integral | Error |
| Xmax | 5 | | Trapezoid | 0.501177985 | -0.001096955 |
| Max Delta y | 0.01 | | Simpson | 0.502274925 | -1.54401E-08 |
| Max Delta X | 0.1 | | Simpson 3/8 | 0.502274933 | -6.86215E-09 |
| f(X) | EXP(-X)*SIN(X) | | Gauss 2 | 0.50227495 | 1.02935E-08 |
| Exact Integral | 0.50227494 | | Gauss 3 | 0.50227494 | -1.32561E-13 |
| | | | Gauss 4 | 0.50227494 | 0 |
| | | | Gauss 5 | 0.50227494 | 0 |

*Figure 5. The first set of results for integrating $e^{-X}$sin(X) from 0 to 5.*

The higher order Gaussian quadrature versions yield more accurate results. The improvement is a few order in magnitude. Figure 6 shows the same integrals for a maximum function change value of 0.1:

| | | | | | |
|---|---|---|---|---|---|
| X0 | 0 | | Method | Integral | Error |
| Xmax | 5 | | Trapezoid | 0.482978533 | -0.019296407 |
| Max Delta y | 0.1 | | Simpson | 0.50227487 | -6.98014E-08 |
| Max Delta X | 0.1 | | Simpson 3/8 | 0.502274909 | -3.10208E-08 |
| f(X) | EXP(-X)*SIN(X) | | Gauss 2 | 0.502274987 | 4.65361E-08 |
| Exact Integral | 0.50227494 | | Gauss 3 | 0.50227494 | -1.96576E-12 |
| | | | Gauss 4 | 0.50227494 | 0 |
| | | | Gauss 5 | 0.50227494 | 0 |

*Figure 6. The second set of results for integrating $e^{-X}$sin(X) from 0 to 5.*

As expected, the reduction in the maximum change in function value also increases the errors in the results by several order of magnitudes. Gaussian quadrature of orders 4 and 5 still maintain exact solutions.

## The $e^{-x}$ $\sin(x)^2$ Integral

Figure 7 shows the results of the integrating the damped oscillating function $e^{-x}\sin(x)^2$ between 0 and 5, for a maximum function change value of 0.001:

| | | | | | |
|---|---|---|---|---|---|
| X0 | 0 | | Method | Integral | Error |
| Xmax | 5 | | Trapezoid | 3.097892139 | 2.701093357 |
| Max Delta y | 0.001 | | Simpson | 0.396798786 | 4.81825E-09 |
| Max  Delta X | 0.1 | | Simpson 3/8 | 0.396798784 | 2.14134E-09 |
| f(X) | EXP(-X)*(SIN(X))^2 | | Gauss 2 | 0.396798778 | -3.21227E-09 |
| Exact Integral | 0.396798782 | | Gauss 3 | 0.396798782 | 1.06137E-13 |
| | | | Gauss 4 | 0.396798782 | 0 |
| | | | Gauss 5 | 0.396798782 | 4.996E-16 |

*Figure 7. The first set of results for integrating $e^{-X}sin(X)^2$ from 0 to 5.*

The Trapezoidal method generates a very large error! The 4th order of Gaussian quadrature generates an exact result. The other method still generate good results. Reducing the maximum function change value to 0.01 yields the results, in Figure 8, that show increased errors, except for the Gaussian quadrature of order 4 and 5.

| | | | | | |
|---|---|---|---|---|---|
| X0 | 0 | | Method | Integral | Error |
| Xmax | 5 | | Trapezoid | 30.97892139 | 30.58212261 |
| Max Delta y | 0.01 | | Simpson | 0.396798828 | 4.62901E-08 |
| Max  Delta X | 0.1 | | Simpson 3/8 | 0.396798802 | 2.05702E-08 |
| f(X) | EXP(-X)*(SIN(X))^2 | | Gauss 2 | 0.396798751 | -3.08629E-08 |
| Exact Integral | 0.396798782 | | Gauss 3 | 0.396798782 | 3.0726E-12 |
| | | | Gauss 4 | 0.396798782 | 0 |
| | | | Gauss 5 | 0.396798782 | 0 |

*Figure 8. The second set of results for integrating $e^{-X}sin(X)^2$ from 0 to 5.*

## Conclusion

The Slope-Oriented Numerical Integral method is a novice and flexible adaptive strategy for numerical integration. It allows you to accurately keep pace with a fux while using traditional numerical analysis to perform micro versions of numerical integration.

## References

1. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd edition, Cambridge University Press; 3rd edition, September 10, 2007.
2. Richard L. Burden, J. Douglas Faires, *Numerical Analysis*, Cengage Learning, 9th edition, August 9, 2010.

## Document Information

| *Version* | *Date* | *Comments* |
|---|---|---|
| 1.0.0 | 3/12/2014 | Initial release. |
| 1.1.0 | 3/15/2014 | Modified the Excel VBA to handle slopes of zero and near zero. |
| 1.2.0 | 3/26/2014 | Added a maximum limit for the change in X. This change increased, in general, the accuracy of the results. |