```vb
Imports Microsoft.VisualBasic
Imports System.Data

Imports System.Text
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Drawing.Text
Imports System.Drawing.Drawing2D
Imports System.Math

Public Class MatrixOperations
    Dim y2() As Single
    Dim yout As Single

    Private Sub spline(ByVal x() As Single, ByVal y() As Single, ByVal n As Integer, ByVal yp1 As Single, ByVal ypn As Single)
        Dim i, k As Integer
        Dim p, qn, sig, un, u() As Single

        ReDim y2(n)
        ReDim u(n)

        If yp1 > 9.9E+29 Then
            y2(0) = u(0) = 0.0
        Else
            y2(0) = -0.5
            u(0) = (3.0 / (x(1) - x(0))) * ((y(1) - y(0)) / (x(1) - x(0)) - yp1)
        End If

        For i = 1 To n - 1
            sig = (x(i) - x(i - 1)) / (x(i + 1) - x(i - 1))
            p = sig * y2(i - 1) + 2
            y2(i) = (sig - 1) / p
            u(i) = (y(i + 1) - y(i)) / (x(i + 1) - x(i)) - (y(i) - y(i - 1)) / (x(i) - x(i - 1))
            u(i) = (6 * u(i) / (x(i + 1) - x(i - 1)) - sig * u(i - 1)) / p
        Next i

        If ypn > 9.9E+29 Then
            qn = un = 0.0
        Else
            qn = 0.5
            un = (3.0 / (x(n) - x(n - 1))) * (ypn - (y(n) - y(n - 1)) / (x(n) - x(n - 1)))
        End If

        y2(n) = (un - qn * u(n - 1)) / (qn * y2(n - 1) + 1)
        For k = n - 1 To k >= 1 Step -1
            y2(k) = y2(k) * y2(k + 1) + u(k)

        Next


    End Sub

    Private Function splint(ByVal xa() As Single, ByVal ya() As Single, ByVal y2a() As Single, ByVal n As Integer, ByVal x As Single)
        Dim klo, khi, k As Integer
        Dim h, b, a As Single

        klo = 0
        khi = n
        Do While khi - klo > 1
            k = khi + klo >> 1

            If xa(k) > x Then
                khi = k
            Else
                klo = k
            End If
        Loop
        h = xa(khi) - xa(klo)
        a = (xa(khi) - x) / h
        b = (x - xa(klo)) / h
        yout = a * ya(klo) + b * ya(khi) + ((a ^ 3 - a) * y2a(klo) + (b ^ 3 - b) * y2a(khi)) * (h ^ 2) / 6

        Return yout
```

```
    End Function

#Region "Imagery"
    ''' <summary>With 2-D array, returns an gaussian-filtered array</summary>
    ''' <param name="Mat"> 2-D array </param>
    ''' <param name="sigma"> Dispersion determines FWHM </param>
    ''' <returns></returns>
    Function GaussianBlur(ByVal Mat(,) As Double, ByVal sigma As Double)
        Dim x0 As Integer = Mat.GetUpperBound(0)
        Dim x1 As Integer = Mat.GetUpperBound(1)

        Dim x As Integer = Round(Sqrt(2 * sigma ^ 2 * Math.Log(5 * sigma)), 0)

        Dim ks As Integer = 2 * x + 1
        Dim c As Integer = CInt(ks / 2 - 0.5)

        Dim kernel(ks, ks) As Double
        For i As Integer = 0 To ks - 1
            For j As Integer = 0 To ks - 1
                kernel(i, j) = (1 / (2 * Math.PI * sigma ^ 2)) * Math.Exp(-((i - c) ^ 2 + (j - c) ^ 2) / (2 * sigma ^ 2))
            Next
        Next

        Dim O(x0 + 1 - ks + 1, x1 + 1 - ks + 1) As Double
        For i As Integer = 0 To x0 - ks + 1
            For j As Integer = 0 To x1 - ks + 1
                For k As Integer = 0 To ks - 1
                    For l As Integer = 0 To ks - 1
                        O(i, j) += Mat(i + k, j + l) * kernel(k, l)
                    Next
                Next
            Next
        Next

        Return O
    End Function

    ''' <summary>With Matrix(n,m), saves a Red Temp color-contour bitmap of intensities</summary>
    ''' <param name="Mat"> 2-D Array </param>
    ''' <param name="FullPath"> Path to Save Bitmap </param>
    Sub CreateRedImage(ByVal Mat(,) As Double, ByVal FullPath As String, ByVal format As System.Drawing.Imaging.ImageFormat)
        Dim x() As Single = {0, 16, 32, 48, 64, 80, 100}
        Dim yr() As Single = {28, 102, 155, 208, 255, 255, 255}
        Dim yg() As Single = {0, 0, 0, 45, 113, 187, 249}
        Dim yb() As Single = {0, 0, 0, 0, 0, 113, 243}

        Dim m As Integer = Mat.GetLength(1)
        Dim n As Integer = Mat.GetLength(0)

        Dim width As Integer = m
        Dim height As Integer = n
        Dim max As Single = 0
        Dim min As Single = 10 ^ 5

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                If Mat(i, j) > max Then
                    max = Mat(i, j)
                End If
                If Mat(i, j) < min Then
                    min = Mat(i, j)
                End If
            Next
        Next

        Dim scale As Single = 0

        If max > 0 Then
            scale = 100 / max
        End If

        Dim objBitmap As Bitmap
        Dim objGraphics As Graphics

```

```
        objBitmap = New Bitmap(width, height)
        objGraphics = Graphics.FromImage(objBitmap)

        'Create Contour Map
        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                Dim val As Integer = scale * Mat(i, j)

                spline(x, yb, 6, 0, 0)
                Dim blue As Integer = splint(x, yb, y2, 6, val)
                spline(x, yr, 6, 0, 0)
                Dim red As Integer = splint(x, yr, y2, 6, val)
                spline(x, yg, 6, 0, 0)
                Dim green As Integer = splint(x, yg, y2, 6, val)

                If green < 0 Then green = 0
                If red < 0 Then red = 0
                If blue < 0 Then blue = 0

                If green > 255 Then green = 255
                If blue > 255 Then blue = 255
                If red > 255 Then red = 255

                objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
            Next
        Next

        objBitmap.Save(FullPath, format)

        objBitmap.Dispose()
        objGraphics.Dispose()
    End Sub

    Sub CreateLogImage(ByVal Mat(,) As Double, ByVal FullPath As String)
        Dim m As Integer = Mat.GetLength(1)
        Dim n As Integer = Mat.GetLength(0)

        Dim width As Integer = m
        Dim height As Integer = n
        Dim max As Single = 0

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                If Mat(i, j) > max Then
                    max = Mat(i, j)
                End If
            Next
        Next

        Dim scale As Double = 0

        If max > 0 Then
            scale = 255 / Math.Log(max)
        End If

        Dim objBitmap As Bitmap
        Dim objGraphics As Graphics

        objBitmap = New Bitmap(width, height)
        objGraphics = Graphics.FromImage(objBitmap)

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                Dim val As Integer
                If Mat(i, j) <= 0 Then
                    val = 0
                Else
                    val = Math.Round(scale * Math.Log(Mat(i, j)), 0)
                End If

                If val < 0 Then val = 0
                If val > 255 Then val = 255

                objBitmap.SetPixel(j, i, Color.FromArgb(255, val, val, val))
            Next
        Next
```

```
224
225         objBitmap.Save(FullPath)
226
227         objBitmap.Dispose()
228         objGraphics.Dispose()
229
230     End Sub
231
232
233     Sub CreateColorLogImage(ByVal Mat(,) As Double, ByVal FullPath As String)
234         Dim x(7) As Single
235         Dim yb(7) As Single
236         Dim yr(7) As Single
237         Dim yg(7) As Single
238
239         x(0) = 0
240         x(1) = 16
241         x(2) = 32
242         x(3) = 48
243         x(4) = 64
244         x(5) = 80
245         x(6) = 100
246
247         yr(0) = 52
248         yr(1) = 75
249         yr(2) = 0
250         yr(3) = 0
251         yr(4) = 255
252         yr(5) = 255
253         yr(6) = 255
254         yg(0) = 0
255         yg(1) = 0
256         yg(2) = 0
257         yg(3) = 128
258         yg(4) = 255
259         yg(5) = 165
260         yg(6) = 0
261         yb(0) = 52
262         yb(1) = 130
263         yb(2) = 255
264         yb(3) = 0
265         yb(4) = 0
266         yb(5) = 0
267         yb(6) = 0
268
269         Dim m As Integer = Mat.GetLength(1)
270         Dim n As Integer = Mat.GetLength(0)
271
272         Dim width As Integer = m
273         Dim height As Integer = n
274         Dim max As Single = 0
275
276         For i As Integer = 0 To n - 1
277             For j As Integer = 0 To m - 1
278                 If Mat(i, j) > max Then
279                     max = Mat(i, j)
280                 End If
281             Next
282         Next
283
284         Dim scale As Double = 0
285
286         If max > 0 Then
287             scale = 100 / Math.Log(max)
288         End If
289
290         Dim objBitmap As Bitmap
291         Dim objGraphics As Graphics
292
293         objBitmap = New Bitmap(width, height)
294         objGraphics = Graphics.FromImage(objBitmap)
295
296         For i As Integer = 0 To n - 1
297             For j As Integer = 0 To m - 1
298                 Dim val As Integer
299                 If Mat(i, j) <= 1 Then
```

```
                    val = 0
                Else
                    val = Math.Round(scale * Math.Log(Mat(i, j)), 0)
                End If


                spline(x, yb, 6, 0, 0)
                Dim blue As Integer = splint(x, yb, y2, 6, val)
                spline(x, yr, 6, 0, 0)
                Dim red As Integer = splint(x, yr, y2, 6, val)
                spline(x, yg, 6, 0, 0)
                Dim green As Integer = splint(x, yg, y2, 6, val)

                If green < 0 Then green = 0
                If red < 0 Then red = 0
                If blue < 0 Then blue = 0

                If green > 255 Then green = 255
                If blue > 255 Then blue = 255
                If red > 255 Then red = 255

                objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
            Next
        Next

        objBitmap.Save(FullPath)

        objBitmap.Dispose()
        objGraphics.Dispose()

    End Sub

    Sub CreateColorImage(ByVal Mat(,) As Double, ByVal FullPath As String)
        Dim x(7) As Single
        Dim yb(7) As Single
        Dim yr(7) As Single
        Dim yg(7) As Single

        x(0) = 0
        x(1) = 16
        x(2) = 32
        x(3) = 48
        x(4) = 64
        x(5) = 80
        x(6) = 100

        yr(0) = 52
        yr(1) = 75
        yr(2) = 0
        yr(3) = 0
        yr(4) = 255
        yr(5) = 255
        yr(6) = 255
        yg(0) = 0
        yg(1) = 0
        yg(2) = 0
        yg(3) = 128
        yg(4) = 255
        yg(5) = 165
        yg(6) = 0
        yb(0) = 52
        yb(1) = 130
        yb(2) = 255
        yb(3) = 0
        yb(4) = 0
        yb(5) = 0
        yb(6) = 0

        Dim m As Integer = Mat.GetLength(1)
        Dim n As Integer = Mat.GetLength(0)

        Dim width As Integer = m
        Dim height As Integer = n
        Dim max As Single = 0

        For i As Integer = 0 To n - 1
```

```
            For j As Integer = 0 To m - 1
                If Mat(i, j) > max Then
                    max = Mat(i, j)
                End If
            Next
        Next

        Dim scale As Double = 0

        If max > 0 Then
            scale = 100 / max
        End If

        Dim objBitmap As Bitmap
        Dim objGraphics As Graphics

        objBitmap = New Bitmap(width, height)
        objGraphics = Graphics.FromImage(objBitmap)

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                Dim val As Integer
                If Mat(i, j) = 0 Then
                    val = 0
                Else
                    val = Round(scale * Mat(i, j), 0)
                End If


                spline(x, yb, 6, 0, 0)
                Dim blue As Integer = splint(x, yb, y2, 6, val)
                spline(x, yr, 6, 0, 0)
                Dim red As Integer = splint(x, yr, y2, 6, val)
                spline(x, yg, 6, 0, 0)
                Dim green As Integer = splint(x, yg, y2, 6, val)

                If green < 0 Then green = 0
                If red < 0 Then red = 0
                If blue < 0 Then blue = 0

                If green > 255 Then green = 255
                If blue > 255 Then blue = 255
                If red > 255 Then red = 255

                objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
            Next
        Next

        objBitmap.Save(FullPath)

        objBitmap.Dispose()
        objGraphics.Dispose()

    End Sub

#End Region


#Region "File Handling"
    Sub SaveCSV(ByVal Mat(,) As Double, ByVal FileName As String)
        Dim strm As New System.IO.StreamWriter(FileName, False)
        Dim n As Integer = Mat.GetLength(0) 'num of rows
        Dim m As Integer = Mat.GetLength(1) 'num of columns

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To m - 1
                strm.Write(Mat(i, j))
                If j < m - 1 Then
                    strm.Write(",")
                End If
            Next
            strm.Write(strm.NewLine)
        Next

        strm.Close()
    End Sub
```

```
452
453     Sub SaveSSV(ByVal Mat(,) As Double, ByVal FileName As String)
454         Dim strm As New System.IO.StreamWriter(FileName, False)
455         Dim n As Integer = Mat.GetLength(0) 'num of rows
456         Dim m As Integer = Mat.GetLength(1) 'num of columns
457 
458         For i As Integer = 0 To n - 1
459             For j As Integer = 0 To m - 1
460                 strm.Write(Mat(i, j))
461                 If j < m - 1 Then
462                     strm.Write(" ")
463                 End If
464             Next
465             strm.Write(strm.NewLine)
466         Next
467 
468         strm.Close()
469     End Sub
470 
471     ''' <summary>With FileName, loads a csv file into a 2-D array</summary>
472     ''' <param name="FileName"> Full Path to the csv file </param>
473     ''' <param name="HasHeaders"> Does the csv file have column names? </param>
474     Function OpenCSV(ByVal FileName As String, ByVal HasHeaders As Boolean)
475         Dim csv As New CSVData
476         csv.LoadCSV(FileName, HasHeaders)
477         Dim dt As DataTable = csv.CSVDataSet.Tables(0)
478         Dim n As Integer = dt.Rows.Count
479         Dim m As Integer = dt.Columns.Count
480         Dim Mat(n, m) As Double
481         For i As Integer = 0 To n - 1
482             For j As Integer = 0 To m - 1
483                 Mat(i, j) = dt.Rows(i).Item(j)
484             Next
485         Next
486         dt.Dispose()
487         csv.Dispose()
488         Return Mat
489     End Function
490 #End Region
491 
492 
493 #Region "Coords"
494     Function FindCenter(ByVal Mat(,) As Double)
495         Dim m As Integer = Mat.GetLength(1)
496         Dim n As Integer = Mat.GetLength(0)
497         Dim pnt As New Point(Round(m / 2, 0), Round(n / 2, 0))
498         m = pnt.X
499         n = pnt.Y
500 
501         Dim val As Double = 0
502         For i As Integer = n - 10 To n + 10
503             For j As Integer = m - 10 To m + 10
504                 If Mat(i, j) > val Then
505                     val = Mat(i, j)
506                     pnt.X = j
507                     pnt.Y = i
508                 End If
509             Next
510         Next
511 
512         Return pnt
513     End Function
514 
515     Function polar2xy(ByVal Mat(,) As Double)
516         'Dim L As Integer = Math.Truncate(dt.Rows.Count / Math.Sqrt(2))
517         Dim R As Integer = Mat.GetLength(0) - 1
518         Dim out(2 * R + 1, 2 * R + 1) As Double
519 
520         For y As Integer = -R To R
521 
522             For x As Integer = -R To R
523 
524                 If Sqrt(x ^ 2 + y ^ 2) < R Then
525                     Dim rcoord, thetacoord As Integer
526                     rcoord = Round(Sqrt(x ^ 2 + y ^ 2), 0)
527                     thetacoord = Round((Atan2(y, x)) * R / (2 * PI), 0)
```

```
                    If thetacoord < 0 Then thetacoord += R
                    out(y + R, x + R) = Mat(rcoord, thetacoord)
                End If
            Next
        Next

        Return out
    End Function

    Function xy2polar(ByVal Mat(,) As Double)
        Dim ro As Integer = Mat.GetUpperBound(0) / 2
        Dim out(ro, ro) As Double

        For r As Integer = 0 To ro
            For t As Integer = 0 To ro
                out(r, t) += Mat(r * Math.Cos(t * 2 * Math.PI / ro) + ro, r * Math.Sin(t * 2 * Math.PI / ro) + ro)
            Next
        Next
        Return out
    End Function
#End Region

End Class

```