

```

1 Imports Microsoft.VisualBasic
2 Imports System.Data
3
4 Imports System.Text
5 Imports System.Drawing
6 Imports System.Drawing.Imaging
7 Imports System.Drawing.Text
8 Imports System.Drawing.Drawing2D
9 Imports System.Math
10
11 Public Class MatrixOperations
12     Dim y2() As Single
13     Dim yout As Single
14
15     Private Sub spline(ByVal x() As Single, ByVal y() As Single, ByVal n As Integer, ByVal yp1 As Single,
16     ByVal ypn As Single)
17         Dim i, k As Integer
18         Dim p, qn, sig, un, u() As Single
19
20         ReDim y2(n)
21         ReDim u(n)
22
23         If yp1 > 9.9E+29 Then
24             y2(0) = u(0) = 0.0
25         Else
26             y2(0) = -0.5
27             u(0) = (3.0 / (x(1) - x(0))) * ((y(1) - y(0)) / (x(1) - x(0)) - yp1)
28         End If
29
30         For i = 1 To n - 1
31             sig = (x(i) - x(i - 1)) / (x(i + 1) - x(i - 1))
32             p = sig * y2(i - 1) + 2
33             y2(i) = (sig - 1) / p
34             u(i) = (y(i + 1) - y(i)) / (x(i + 1) - x(i)) - (y(i) - y(i - 1)) / (x(i) - x(i - 1))
35             u(i) = (6 * u(i) / (x(i + 1) - x(i - 1)) - sig * u(i - 1)) / p
36         Next i
37
38         If ypn > 9.9E+29 Then
39             qn = un = 0.0
40         Else
41             qn = 0.5
42             un = (3.0 / (x(n) - x(n - 1))) * (ypn - (y(n) - y(n - 1)) / (x(n) - x(n - 1)))
43         End If
44
45         y2(n) = (un - qn * u(n - 1)) / (qn * y2(n - 1) + 1)
46         For k = n - 1 To 1 Step -1
47             y2(k) = y2(k) * y2(k + 1) + u(k)
48         Next
49
50     End Sub
51
52     Private Function splint(ByVal xa() As Single, ByVal ya() As Single, ByVal y2a() As Single,
53     ByVal n As Integer, ByVal x As Single)
54         Dim klo, khi, k As Integer
55         Dim h, b, a As Single
56
57         klo = 0
58         khi = n
59         Do While khi - klo > 1
60             k = khi + klo >> 1
61
62             If xa(k) > x Then
63                 khi = k
64             Else
65                 klo = k
66             End If
67         Loop
68         h = xa(khi) - xa(klo)
69         a = (xa(khi) - x) / h
70         b = (x - xa(klo)) / h
71         yout = a * ya(klo) + b * ya(khi) + ((a ^ 3 - a) * y2a(klo) + (b ^ 3 - b) * y2a(khi)) * (h ^ 2) / 6
72
73         Return yout

```

```

74     End Function
75
76 #Region "Imagery"
77     ''' <summary>With 2-D array, returns an gaussian-filtered array</summary>
78     ''' <param name="Mat"> 2-D array </param>
79     ''' <param name="sigma"> Dispersion determines FWHM </param>
80     ''' <returns></returns>
81     Function GaussianBlur(ByVal Mat(,) As Double, ByVal sigma As Double)
82         Dim x0 As Integer = Mat.GetUpperBound(0)
83         Dim x1 As Integer = Mat.GetUpperBound(1)
84
85         Dim x As Integer = Round(Sqrt(2 * sigma ^ 2 * Math.Log(5 * sigma)), 0)
86
87         Dim ks As Integer = 2 * x + 1
88         Dim c As Integer = CInt(ks / 2 - 0.5)
89
90         Dim kernel(ks, ks) As Double
91         For i As Integer = 0 To ks - 1
92             For j As Integer = 0 To ks - 1
93                 kernel(i, j) = (1 / (2 * Math.PI * sigma ^ 2)) * Math.Exp(-((i - c) ^ 2 + (j - c) ^ 2) / (2 * sigma ^ 2))
94             Next
95         Next
96
97         Dim O(x0 + 1 - ks + 1, x1 + 1 - ks + 1) As Double
98         For i As Integer = 0 To x0 - ks + 1
99             For j As Integer = 0 To x1 - ks + 1
100                 For k As Integer = 0 To ks - 1
101                     For l As Integer = 0 To ks - 1
102                         O(i, j) += Mat(i + k, j + l) * kernel(k, l)
103                     Next
104                 Next
105             Next
106         Next
107
108         Return O
109     End Function
110
111     ''' <summary>With Matrix(n,m), saves a Red Temp color-contour bitmap of intensities</summary>
112     ''' <param name="Mat"> 2-D Array </param>
113     ''' <param name="FullPath"> Path to Save Bitmap </param>
114     Sub CreateRedImage(ByVal Mat(,) As Double, ByVal FullPath As String, ByVal format As System.
115         Drawing.Imaging.ImageFormat)
116         Dim x() As Single = {0, 16, 32, 48, 64, 80, 100}
117         Dim yr() As Single = {28, 102, 155, 208, 255, 255, 255}
118         Dim yg() As Single = {0, 0, 0, 45, 113, 187, 249}
119         Dim yb() As Single = {0, 0, 0, 0, 0, 113, 243}
120
121         Dim m As Integer = Mat.GetLength(1)
122         Dim n As Integer = Mat.GetLength(0)
123
124         Dim width As Integer = m
125         Dim height As Integer = n
126         Dim max As Single = 0
127         Dim min As Single = 10 ^ 5
128
129         For i As Integer = 0 To n - 1
130             For j As Integer = 0 To m - 1
131                 If Mat(i, j) > max Then
132                     max = Mat(i, j)
133                 End If
134                 If Mat(i, j) < min Then
135                     min = Mat(i, j)
136                 End If
137             Next
138         Next
139
140         Dim scale As Single = 0
141
142         If max > 0 Then
143             scale = 100 / max
144         End If
145
146         Dim objBitmap As Bitmap
147         Dim objGraphics As Graphics

```

```

148 objBitmap = New Bitmap(width, height)
149 objGraphics = Graphics.FromImage(objBitmap)
150
151 'Create Contour Map
152 For i As Integer = 0 To n - 1
153     For j As Integer = 0 To m - 1
154         Dim val As Integer = scale * Mat(i, j)
155
156         spline(x, yb, 6, 0, 0)
157         Dim blue As Integer = splint(x, yb, y2, 6, val)
158         spline(x, yr, 6, 0, 0)
159         Dim red As Integer = splint(x, yr, y2, 6, val)
160         spline(x, yg, 6, 0, 0)
161         Dim green As Integer = splint(x, yg, y2, 6, val)
162
163         If green < 0 Then green = 0
164         If red < 0 Then red = 0
165         If blue < 0 Then blue = 0
166
167         If green > 255 Then green = 255
168         If blue > 255 Then blue = 255
169         If red > 255 Then red = 255
170
171         objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
172     Next
173 Next
174
175 objBitmap.Save(FullPath, format)
176
177 objBitmap.Dispose()
178 objGraphics.Dispose()
179 End Sub
180
181 Sub CreateLogImage(ByVal Mat(,) As Double, ByVal FullPath As String)
182     Dim m As Integer = Mat.GetLength(1)
183     Dim n As Integer = Mat.GetLength(0)
184
185     Dim width As Integer = m
186     Dim height As Integer = n
187     Dim max As Single = 0
188
189     For i As Integer = 0 To n - 1
190         For j As Integer = 0 To m - 1
191             If Mat(i, j) > max Then
192                 max = Mat(i, j)
193             End If
194         Next
195     Next
196
197     Dim scale As Double = 0
198
199     If max > 0 Then
200         scale = 255 / Math.Log(max)
201     End If
202
203     Dim objBitmap As Bitmap
204     Dim objGraphics As Graphics
205
206     objBitmap = New Bitmap(width, height)
207     objGraphics = Graphics.FromImage(objBitmap)
208
209     For i As Integer = 0 To n - 1
210         For j As Integer = 0 To m - 1
211             Dim val As Integer
212             If Mat(i, j) <= 0 Then
213                 val = 0
214             Else
215                 val = Math.Round(scale * Math.Log(Mat(i, j)), 0)
216             End If
217
218             If val < 0 Then val = 0
219             If val > 255 Then val = 255
220
221             objBitmap.SetPixel(j, i, Color.FromArgb(255, val, val, val))
222         Next
223     Next

```

```

224         objBitmap.Save(FullPath)
225
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

```

```

300         val = 0
301     Else
302         val = Math.Round(scale * Math.Log(Mat(i, j)), 0)
303     End If
304
305
306     spline(x, yb, 6, 0, 0)
307     Dim blue As Integer = splint(x, yb, y2, 6, val)
308     spline(x, yr, 6, 0, 0)
309     Dim red As Integer = splint(x, yr, y2, 6, val)
310     spline(x, yg, 6, 0, 0)
311     Dim green As Integer = splint(x, yg, y2, 6, val)
312
313     If green < 0 Then green = 0
314     If red < 0 Then red = 0
315     If blue < 0 Then blue = 0
316
317     If green > 255 Then green = 255
318     If blue > 255 Then blue = 255
319     If red > 255 Then red = 255
320
321     objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
322 Next
323 Next
324
325 objBitmap.Save(FullPath)
326
327 objBitmap.Dispose()
328 objGraphics.Dispose()
329
330 End Sub
331
332 Sub CreateColorImage(ByVal Mat(,) As Double, ByVal FullPath As String)
333     Dim x(7) As Single
334     Dim yb(7) As Single
335     Dim yr(7) As Single
336     Dim yg(7) As Single
337
338     x(0) = 0
339     x(1) = 16
340     x(2) = 32
341     x(3) = 48
342     x(4) = 64
343     x(5) = 80
344     x(6) = 100
345
346     yr(0) = 52
347     yr(1) = 75
348     yr(2) = 0
349     yr(3) = 0
350     yr(4) = 255
351     yr(5) = 255
352     yr(6) = 255
353     yg(0) = 0
354     yg(1) = 0
355     yg(2) = 0
356     yg(3) = 128
357     yg(4) = 255
358     yg(5) = 165
359     yg(6) = 0
360     yb(0) = 52
361     yb(1) = 130
362     yb(2) = 255
363     yb(3) = 0
364     yb(4) = 0
365     yb(5) = 0
366     yb(6) = 0
367
368     Dim m As Integer = Mat.GetLength(1)
369     Dim n As Integer = Mat.GetLength(0)
370
371     Dim width As Integer = m
372     Dim height As Integer = n
373     Dim max As Single = 0
374
375     For i As Integer = 0 To n - 1

```

```

376         For j As Integer = 0 To m - 1
377             If Mat(i, j) > max Then
378                 max = Mat(i, j)
379             End If
380         Next
381     Next
382
383     Dim scale As Double = 0
384
385     If max > 0 Then
386         scale = 100 / max
387     End If
388
389     Dim objBitmap As Bitmap
390     Dim objGraphics As Graphics
391
392     objBitmap = New Bitmap(width, height)
393     objGraphics = Graphics.FromImage(objBitmap)
394
395     For i As Integer = 0 To n - 1
396         For j As Integer = 0 To m - 1
397             Dim val As Integer
398             If Mat(i, j) = 0 Then
399                 val = 0
400             Else
401                 val = Round(scale * Mat(i, j), 0)
402             End If
403
404             spline(x, yb, 6, 0, 0)
405             Dim blue As Integer = splint(x, yb, y2, 6, val)
406             spline(x, yr, 6, 0, 0)
407             Dim red As Integer = splint(x, yr, y2, 6, val)
408             spline(x, yg, 6, 0, 0)
409             Dim green As Integer = splint(x, yg, y2, 6, val)
410
411             If green < 0 Then green = 0
412             If red < 0 Then red = 0
413             If blue < 0 Then blue = 0
414
415             If green > 255 Then green = 255
416             If blue > 255 Then blue = 255
417             If red > 255 Then red = 255
418
419             objBitmap.SetPixel(j, i, Color.FromArgb(255, red, green, blue))
420         Next
421     Next
422
423     objBitmap.Save(FullPath)
424
425     objBitmap.Dispose()
426     objGraphics.Dispose()
427
428
429 End Sub
430
431 #End Region
432
433
434 #Region "File Handling"
435 Sub SaveCSV(ByVal Mat(,) As Double, ByVal FileName As String)
436     Dim strm As New System.IO.StreamWriter(FileName, False)
437     Dim n As Integer = Mat.GetLength(0) 'num of rows
438     Dim m As Integer = Mat.GetLength(1) 'num of columns
439
440     For i As Integer = 0 To n - 1
441         For j As Integer = 0 To m - 1
442             strm.Write(Mat(i, j))
443             If j < m - 1 Then
444                 strm.Write(",")
445             End If
446         Next
447         strm.Write(strm.NewLine)
448     Next
449
450     strm.Close()
451 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)

```

```

528         If thetacoord < 0 Then thetacoord += R
529         out(y + R, x + R) = Mat(rcoord, thetacoord)
530     End If
531 Next
532 Next
533
534     Return out
535 End Function
536
537 Function xy2polar(ByVal Mat(,) As Double)
538     Dim ro As Integer = Mat.GetUpperBound(0) / 2
539     Dim out(ro, ro) As Double
540
541     For r As Integer = 0 To ro
542         For t As Integer = 0 To ro
543             out(r, t) += Mat(r * Math.Cos(t * 2 * Math.PI / ro) + ro, r * Math.Sin(t * 2 * Math
544             .PI / ro) + ro)
545         Next
546     Next
547     Return out
548 End Function
549 #End Region
550 End Class
551

```