MEK 4250 Elementmethod
Mandatory Assignment 3

Andreas Slyngstad

14. november 2016

We let $((x_k, y_k, z_k))_{k=1}^m$ be some available data points in $\mathbb{R}^3$, and we want to define a model based on a smaller set of coordinates $(u_j, v_j)_{j=1}^n$ in $\mathbb{R}^2$. As mentioned in the introduction (u_j, v_j) will be the vertices of the triangles yodelling the the set of data points. Further we define piecewise linear functions $N_j(x, y)_{j=1}^n$ such that $N_j(u_j, v_j) = 1$, $N_j(u_i, v_i) = 0$ if i $\neq$ j. These functions are much alike the "hat functionsfound in the Finite Element method. Our goal will be to find a least squares fit to approximate the given dataset with these functions.

Exercise 1

Visualisations of a piecewise linear function N_j
(Figure from MEK4250 course book)

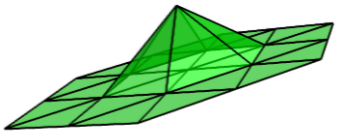

Figure 5.5: Basis functions (global) 2–D

Our set of linear functions $N_j(x, y)_{j=1}^n$ are linear independent if

$$\sum_{j=1}^{n} c_j N_j(x, y) = c_1 N_1 + c_2 N_2 .... c_n N_n = 0$$

for the trivial solution $c_1 = c_2 .. c_n = 0$.
Let us assume that at least two u_j, v_j are located at the same point, say $j = 1, 2$. Then by definition $N_1(u_1, v_1)$ and $N_2(u_2, v_2)$ will be defined over same area with the same properties as introduced. Hence our linear system can be written as $\sum_{j=1}^n c_j N_j(x, y) = c_1 N_1 + c_2 N_2 .... c_n N_n = (c_1 + c_2) N_2 + c_3 N_3 .... c_n N_n = 0$. Hence the system has a nontrivial solution, and the set of functions are linear dependent. We therefore must choose $(u_j, v_j)_{j=1}^n$ at different points to have a linear independent set of functions.

Exercise 2

For simplicity let $(u_j, v_j)_{j=1}^{n}$ for n < m, be our chosen coordinates from the dataset such that $(u_1, v_1) = (x_1, y_1), (u_2, v_2) = (x_2, y_2), .., (u_n, v_n) = (x_n, y_n)$ increasing index order. For a linear independent system we must have that that

$$\sum_{k=1}^{n} \Big(\sum_{j=1}^{n} c_j N_j(x_k, y_k) \Big) + \sum_{k=n+1}^{m} \Big(\sum_{j=1}^{n} c_j N_j(x_k, y_k) \Big) = 0$$

Since

$$\sum_{k=1}^{n} \Big(\sum_{j=1}^{n} c_j N_j(x_k, y_k) \Big) = \sum_{k=1}^{n} \Big(\sum_{j=1}^{n} c_j N_j(u_k, v_k) \Big) = 0$$

only has the trivial solution $c_1 + c_2 .... c_n = 0$, this holds for the whole system and B is linarly independent.

Exercise 3

Now for some k, the triangles sharing (u_k, v_k) as a vertex spans the function N_k. Clearly if these triangles doesn't contain any data point $(\mathrm{x}_j, y_j), then \sum_{j=1}^{n} N_k(x_j, y_j) = 0$. As a result we have a zero column of B, and $\sum_{j=1}^{n} c_j N_j(x, y) = 0$ has a non-trivial solution, since c_k can be choosen freely.

Exercise 4

This simple program writes out the barycentric coordinates of $\mathbf{p}$, relative to the triangle spanned by $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$. A simple run set by the coordinates $\mathbf{p}_1 = (0,0)$, $\mathbf{p}_2 = (1,1)$, $\mathbf{p}_3 = (-1,1)$ for $\mathbf{p} = (0.5,0)$ yeilds

```
Find the barycentric coordinates of (0, 0.5)
In the triangle spanned by

p1 = [0, 0]
p2 = [1, 1]
p3 = [-1, 1]

The barycentric coordinates of p relative to the spanned triangle:
b1 = 0.5,  b2 = 0.25,   b3 = 0.25
```

```
import numpy as np

def barycentric(p1, p2, p3, p):
    #barycentric takes 4 arguments.
    #p1, p2, p3 are the points spanning the triangle,
    #hence these must not lie on a line
    #p is the point within the triangle

    print "Find the barycentric coordinates of (%g, %g) \n" \
    "In the triangle spanned by \n" % (p[0], p[1])
    count = 1
    for i in [p1, p2, p3]:
        print "p%d = [%g, %g]" % (count, i[0], i[1])
        count += 1

    cond = [1 for i in range(3)]

    #Assemble the left hand side of the system
    A =  np.vstack([p1, p2, p3])
    A = np.append(np.transpose(A), [cond], axis=0)

    #Assemble the right hand side of the system
    L = np.insert(np.array(p), len(p), 1)

    #solve system
    b = np.linalg.solve(A, L)

    #Print result
    print "\nThe barycentric coordinates of p relative to the spanned triangle:"
    print "b1 = %g,  b2 = %g,   b3 = %g" % (b[0], b[1], b[2])

p1 = [0, 0]
p2 = [1, 1]
p3 = [-1, 1]

p = [0, 0.5]

barycentric(p1, p2, p3, p)
```