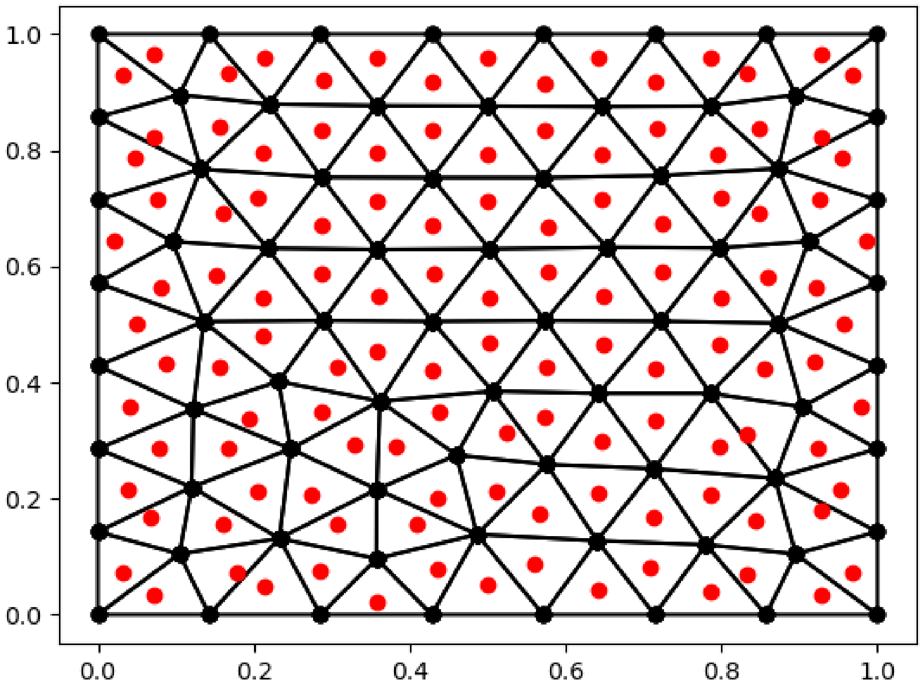
MA3H0 Assignment 2

Joshua Rydell

February 12, 2024

1. We make a mesh a mesh using points from the pygmsh library. We then apply the Bowyer-Watson algorithm to make a triangulation from these points. The red dot is my choice of, x_K, which is the centroid of the each triangle :



2. My choice of f was $f(x, y) = y(1 - y)x^3$. I decided to implement my estimate via
$$\frac{1}{m(K)} \int_K f(x)dx \approx f(x_k)$$
this is implemented when we solve the for the system to find the matrix

```python
# Calculating A and B

A = np.zeros((len(triangulation),len(triangulation)))
B = np.zeros(len(triangulation))


for i in range(len(triangulation)):
    B[i] = triangulation[i].m*f(triangulation[i].centroid)
    for edge in triangulation[i].Faces :
        A[i,i] += edge.Tau
        if( edge.triangleIndex[1] != -1) :
            A[i, edge.triangleIndex[1]] = edge.Tau
U = np.linalg.solve(A,B)
```

$A:$

3. I implement the Tau in the Face class:

```python
def calcTau(self, triangleMesh) : #calculate the value of d
    if(self.triangleIndex[1] == -1) : #this case is for when the face is on the boundar
        xK = triangleMesh[self.triangleIndex[0]].centroid
        #d should be the shortest distance from xK to the line that is this face, ysigma should be the point on the line that fuf
                        (self.points[0][0] - xK[0])*(self.points[1][1] - self.points[0][1]))/dist(self.points[0],self.points[1])
        self.Tau= self.m/self.d
    else : #do this if face is an interior
        xK = triangleMesh[self.triangleIndex[0]].centroid
        xL = triangleMesh[self.triangleIndex[1]].centroid
        self.d = dist(xK,xL)
        self.Tau= self.m/self.d
```

4.

5. Let us enumerate the triangles. We write $A = a_{ij}$ and $B = b_i$. We have $b_i = m(T_i)f_{T_i}$, where $i = 0, \dots, \#\mathcal{T}$. Moving on to calculating A, we see that each Row of A will have three or four entries. Consider a triangle T_i. Suppose T_i is an internal triangle and that T_i shares a face with triangles $T_{k_1}, T_{k_2}, T_{k_3}$ then $a_{ik_\ell} = \tau_{T_i|T_{k_\ell}}$ where $\ell = 1, 2, 3$ and $a_{ii} = \sum_{\ell=1}^{3} -\tau_{T_i|T_{k_\ell}}$.

   If T_i is on the border then suppose it still shares a border with T_{k_1}, T_{k_2}, then we still have $a_{ik_\ell} = \tau_{T_i|T_{k_\ell}}$ where $\ell = 1, 2$, but $a_{ii} = \sum_{\ell=1}^{2} \tau_{T_i|T_{k_\ell}} + \tau_{i,\sigma}$. Where σ is the border edge.

   These matrices give us the linearised form of problem (2), so we can write $AU = B$.