In this problem you will extend the script to quadratic Lagrange finite elements...

To extend the script to quadratic elements, the following basis functions were used:

$$\begin{aligned}
\phi_1(\alpha,\beta) &= (1-\alpha-\beta)(1-2\alpha-2\beta)\\
\phi_2(\alpha,\beta) &= \alpha(2\alpha-1)\\
\phi_3(\alpha,\beta) &= \beta(2\beta-1)\\
\phi_4(\alpha,\beta) &= 4\alpha(1-\alpha-\beta)\\
\phi_5(\alpha,\beta) &= 4\alpha\beta\\
\phi_6(\alpha,\beta) &= 4\beta(1-\alpha-\beta)
\end{aligned}$$

associated with the interpolation points $x_i = (0,0),(1,0),(1,0),(0.5,0),(0.5,0.5),(0,0.5)$.

The associated gradient functions were calculated for these basis functions. The basis functions and gradient of the basis functions were used with a 3-point 2nd order quadrature to compute the bilinear form and the LHS of the weak-form PDE. The additional midpoint nodes were created for each element and were assigned a global numbering.

Coincident midpoint nodes belonging to two adjacent elements were found from the element adjacency matrix with the first occurrence of the node. This reduced the cost of checking for coincident nodes; instead of comparing a given node with all other midpoint nodes, it only need be compared to the 9 midpoint nodes from the adjacent elements.

Figure 1 shows an example computed solution using quadratic elements for the given BCs and forcing function from fesimple_detail.ipynb.

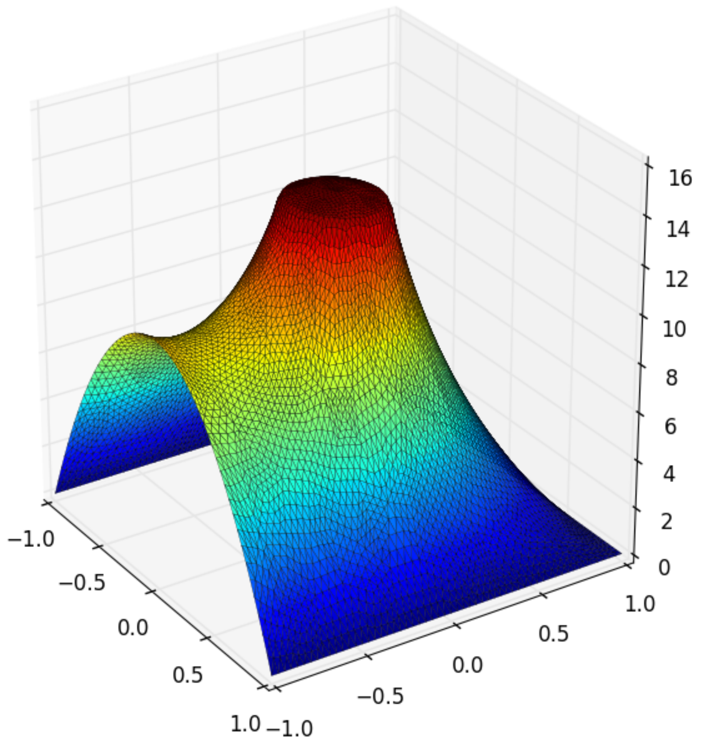

Figure 1: Computed solution for given κ, f and boundary conditions. Quadratic elements, 2752 elements.

Is the accuracy improved?

In order to determine the order of convergence an analytical test case was used to compare the computed solution with the exact solution:

$$-\nabla \cdot \kappa \nabla u = -sin(x)\, sin(y),\ \kappa = 1,\ u = 0 \in d\Omega$$

this permits the analytical solution:

$$u(x, y) = -\frac{1}{2\pi^2}\, sin(x)\, sin(y)$$

The discrete $l2$ norm of the error, $u - u^h$, was computed as:

$$\sum_{\Omega} \int_{\tau} |J|\, (u^h - u)^2 d\mathbf{x} \approx \sum_{\Omega} |J| \sum_{i} \left(w_i u(\mathbf{x}_i) - u_i^h \right)$$

where u_i^h is the value of the computed solution at the interpolation points x_i and w_i the associated quadrature weights.

Figure 2 shows an example computed solution of this test case. Figure 3 plots the order of convergence of the two elements. It can be seen that not only is the accuracy of the quadratic elements better for the same number of elements, but the order of convergence is better as well. As expected the order of convergence of the linear elements was 1.00, while the quadratic element convergence rate was 1.86, almost in line with the expected value of 2.

How does the timing of the linear assembly scale? Does this change for quadratics?

The assembly times for both linear and quadratic elements was measured for three refinement levels. Table 1 displays the timing in seconds. For both element types the assembly time was found to be proportional to the number of elements. Quadratic element assembly time was measured to take about 25% longer than linear elements.

Type/Num Elements	688	2752	11008
Linear	0.14 secs	0.57	2.20
Quadratic	0.18	0.70	2.77

Table 1: Scaling of assembly time for **A** for linear and quadratic elements.

Compare (an estimate of) the condition number. What can you conclude?

Table 2 charts the scaling of the condition number given by *numpy.linalg.cond(A.todense())*. It was found the condition number estimates given by *scipy.sparse.linalg.lsmr()* were not accurate. Due to computational constraints the condition number couldn't be computed for the third refinement level.

Regardless, the scaling of the condition number is consistent; the condition number of quadratic elements is $\sim$ 5.8x that of linear elements, the condition number for both element types is proportional to $\sim$ 1.1x the number of elements. While quadratic elements converge more quickly then linear elements, the resulting linear system will likely require more iterations of a linear solver to reach the solution due to the higher condition number.

Type/Num Elements	688	2752
Linear	1452	6549
Quadratic	8533	38083

Table 2: Scaling of condition number of **A** dependent on element type and number of elements.

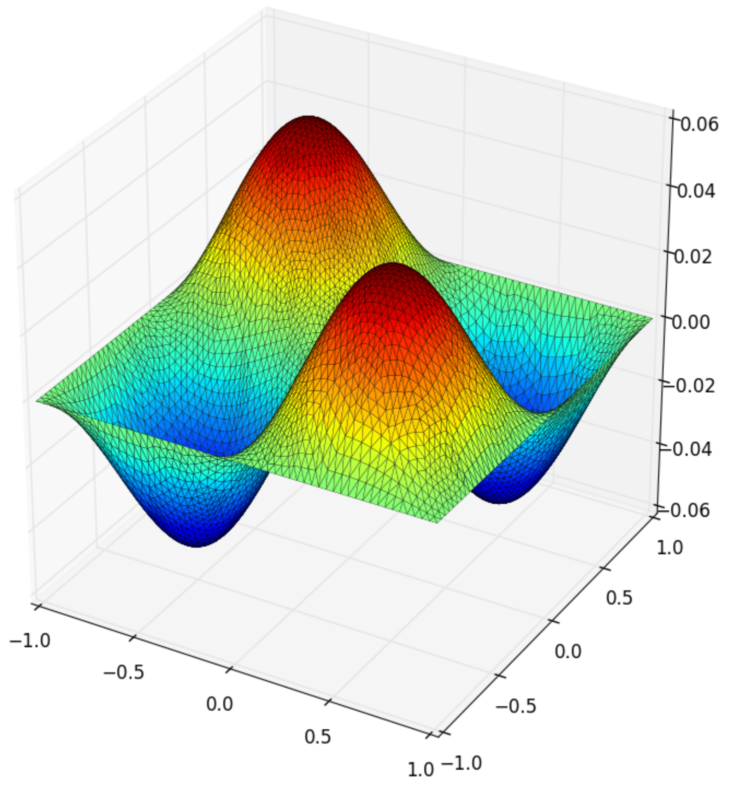

Figure 2: Example solution to the analytical test case, 2752 quadratic elements.

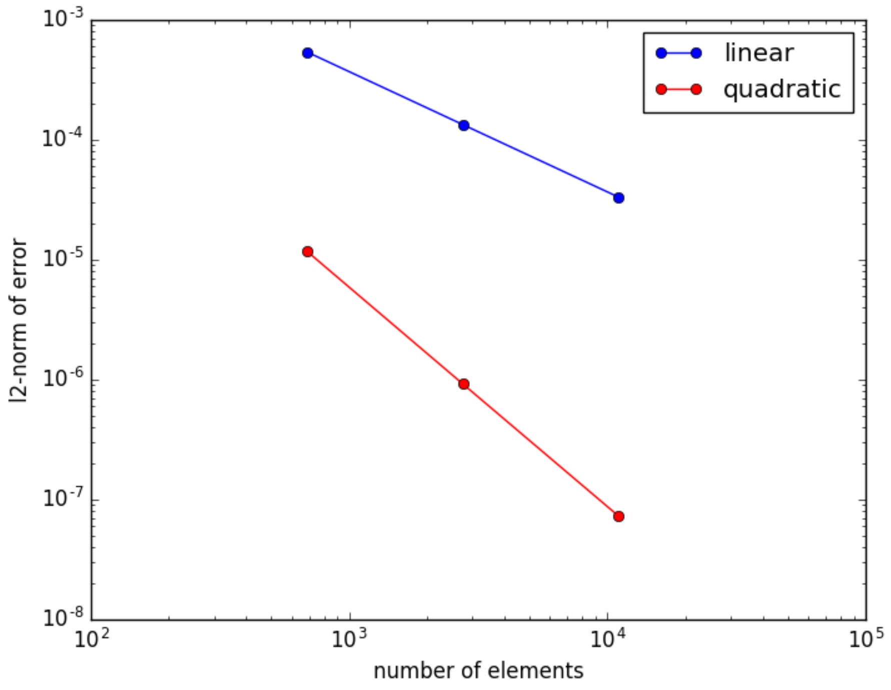

Figure 3: Convergence plots for linear and quadratic elements, 688,2752,11008 elements. Observed order of convergence for linear elements: 1.00, quadratic elements: 1.86

Compare the sparsity of the linear system for the linears versus quadratics.

Figures 4 and 5 compare the sparsity of **A** for linear and quadratic elements. At first glance Figure 4 would indicate that the quadratic element is less sparse. However counting the number of non-zeros relative to matrix size, the linear element matrix is nearly 2.3x as dense.

The confusing figure can partially be blamed on the random numbering of the element vertices, particularly the triangle midpoints which were numbered without any care for optimality. If a Reverse Cuthill-Mckee ordering is computed to minimize the bandwidth a more understandable sparsity pattern emerges, as shown in Figure 5. It is evident from the plots that indeed the quadratic element matrix is more sparse, although it has nearly 4 times the bandwidth of the linear element matrix. The greater bandwidth shouldn't be surprising given the additional basis functions (and therefore the size of the element stencil) required for the quadratic element.

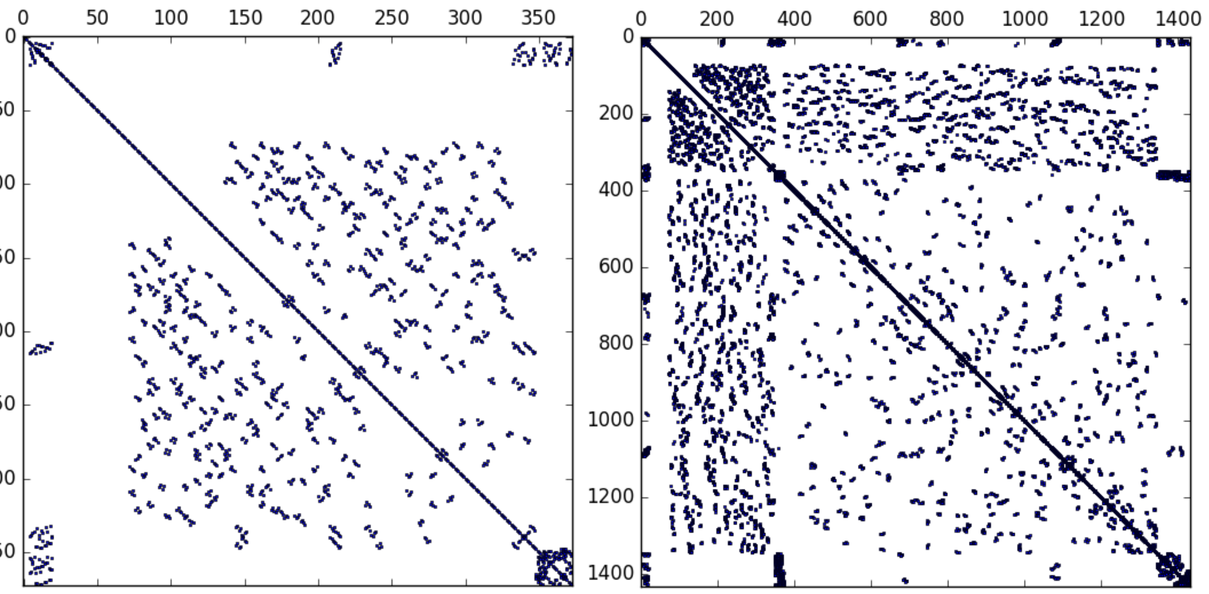

Figure 4: Sparsity pattern of **A** for linear(left) and quadratic(right) elements, 688 elements.

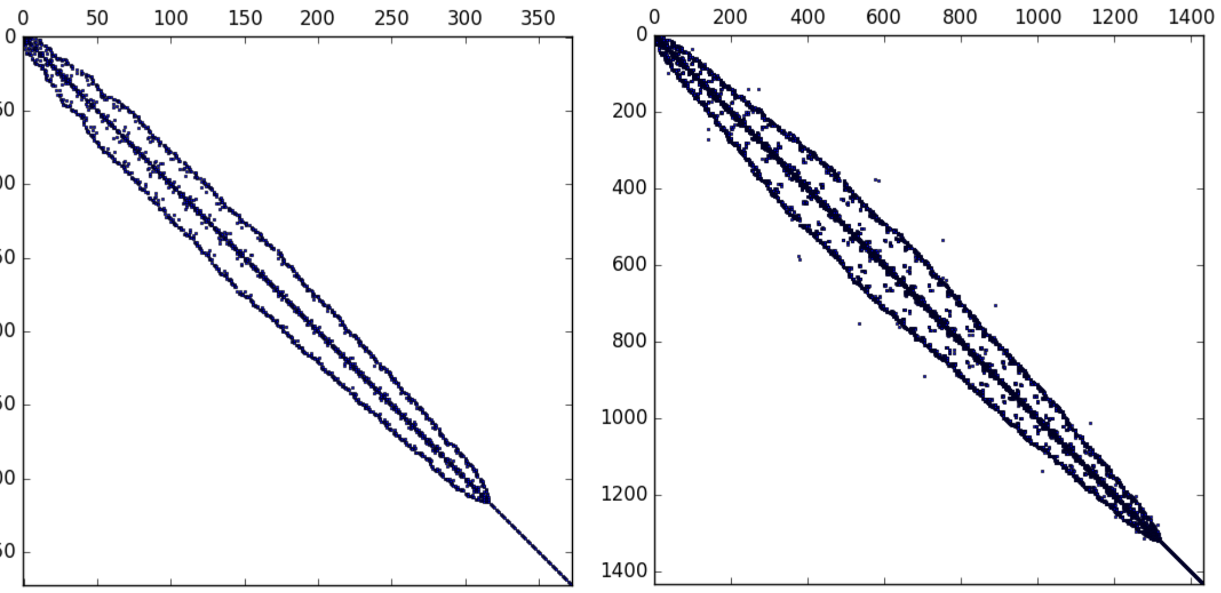

Figure 5: Same matrices as in Figure 4, but after Reverse Cuthill-Mckee ordering.