Finite Element Method: Eigenvalue Problems

Aleksandar Ivanov

April 21, 2021

1 Problem Statements

Problem 1

Using finite element methods find the first few eigenvalues of a semicircular membrane. Extrapolating the solution to high density meshes determine the accuracy of this method. Compare this result with the well known fact that the aforementioned eigenvalues are simply the squares of the zeros of the Bessel J_m functions.

Problem 2

Estimate the eigenvalues again, now using the Galerkin method with the approximate basis set

$$\left\{r^{m+n}(1-r)\sin(m\phi)\right\}_{m=1,\cdots,M}^{n=0,\cdots,N}. \tag{1}$$

Problem 3

Find the eigenmodes of a membrane with the shape shown in fig. 1

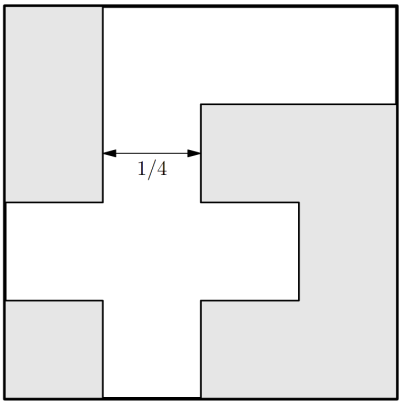

Figure 1: Shape of the membrane on which we're solving Problem 3.

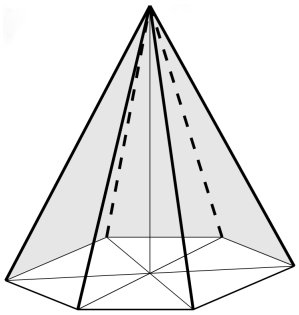

Figure 2: Trial functions for FEM.

2 Mathematical Setup

As in the previous task with FEM, we will be using a mesh on the domain of interest, triangulating it and on top of that expanding each function of interest in the basis of simplest possible functions — the functions ϕ_i, which are centered at the point (x_i, y_i) and are best described by fig. 2.

Defining the inner product

$$\langle f, g\rangle = \int_D fg\,\mathrm{d}S, \tag{2}$$

we rewrite the eigenvalue problem $\nabla^2 u = -\lambda u$ as

$$\langle \nabla u, \nabla u\rangle - \lambda\langle u, u\rangle = 0. \tag{3}$$

Expanding u as

$$u = \sum_{i=0}^{N_N} c_i\phi_i, \tag{4}$$

where N_N is the number of nodes we get the generalized matrix eigenvalue equation

$$A\mathbf{c} = \lambda B\mathbf{c}, \tag{5}$$

where $A, B \in \mathbb{R}^{N_N \times N_N}$ and

$$A_{ij} = \langle \nabla\phi_i, \nabla\phi_j \rangle , \qquad B_{ij} = \langle \phi_i, \phi_j \rangle . \tag{6}$$

We will again use a triangle based organization of the problem. Each node will be given a global index $i \in [1, N_N]$, which fixes a permutation of the rows/columns of the matrices A and B. Furthermore, we will index the triangles with an index $t \in [1, N_T]$. Within each triangle the vertices will also have a local index $m \in [1, 2, 3]$, so that each triangle is defined by a relation between the local and global indices of the vertices that make it up.

To build the matrices we will iterate through the triangles and slowly add in all the interactions. For two points with local indices m and n in a given triangle t, the integration from the inner product gives the contribution

$$A^t_{mn} = \frac{1}{4S_t}\left[(y_{m+1} - y_{m+2})(y_{n+1} - y_{n+2}) + (x_{m+2} - x_{m+1})(x_{n+2} - x_{n+1})\right], \tag{7}$$

where S_t is the area of the triangle t and the index addition is taken cyclically i.e. modulo 3. The full matrix A is then constructed as

$$A_{ij} = \sum_{t,m,n} A^t_{mn}, \tag{8}$$

where the sum goes over those combinations of nodes and triangles that can give rise to the global index on the left, i.e. such t and m that give i and such t and n that give j.

The case of B is a bit simpler. To it, each triangle contributes

$$B^t_{mn} = \begin{cases} S_t/6 & m = n \\ S_t/12 & m \neq n \end{cases}, \tag{9}$$

which we then sum in the same way as before

$$B_{ij} = \sum_{t,m,n} B^t_{mn}. \tag{10}$$

The system we got can be solved as a generalized eigenvalue problem, or it can be converted into a normal eigenvalue problem by a Cholesky decomposition of the matrix B since it is positive definite. This gives

$$B = LL^T, \quad \tilde{A} = L^{-1}A\left(L^{-1}\right)^T, \quad \tilde{\mathbf{c}} = L^T\mathbf{c}, \\ \Rightarrow \tilde{A}\tilde{\mathbf{c}} = \lambda\tilde{\mathbf{c}}, \tag{11}$$

where instead of ever calculating inverses we, of course, solve upper and lower triangular systems.

The basis of the Galerkin method is the same as FEM but here the choice of a basis of functions is independent of the mesh on which we're plotting and generally based on symmetry considerations and intuition about the system. This has the advantage of being able to attain enormous precision with even a couple of basis functions, but its drawback is that a choice of basis functions for a generic domain is usually hard to come up with.

The analytical solutions for the semi-circular membrane can also be calculated and are given by Bessel functions as

$$u_{mn} = J_m(\xi_{mn} r)\sin(m\phi), \\ \lambda_{mn} = \xi^2_{mn}, \tag{12}$$

where ξ_{mn} is the n-th zero of the m-th Bessel function. Furthermore, m is limited to values in $\{1, 2, 3, \cdots\}$.

3 Numerical Setup

To solve the derived matrix eigenvalue problems, both generalized and not, we will use the built-in function `scipy.linalg.eigh`, which finds the eigenvalues and eigenvectors of symmetric matrices.

Generating triangulations for FEM will again be done with the built-in Delaunay triangulation under `scipy.spatial.Delaunay`.

4 Results

We begin by testing the speed of the algorithms. We will test both the Cholesky decomposition option and the built-in generalized solver without the decomposition. The results are shown in fig. 3. The Cholesky decomposition and the built-in method for generalized eigenvalue equations do about as well as each other; there is no perceptible trend between them. The fit in the figure is a power law fit, and it shows that we can't really claim too much precision in calculating the exponent. This is because the graph as a whole and the last 35 points tell a slightly different story about its value. The genera fit gives a value of $\sim$1.4 while the final few

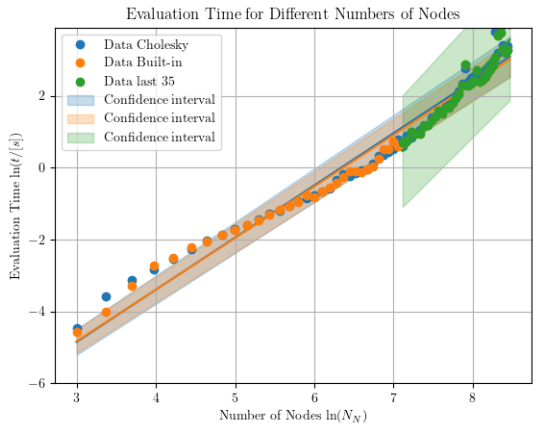

Figure 3: Power law fit for the evaluation times for both versions of the algorithm. In green is the fit of the last 35 points, which predicts a higher value of the exponent.

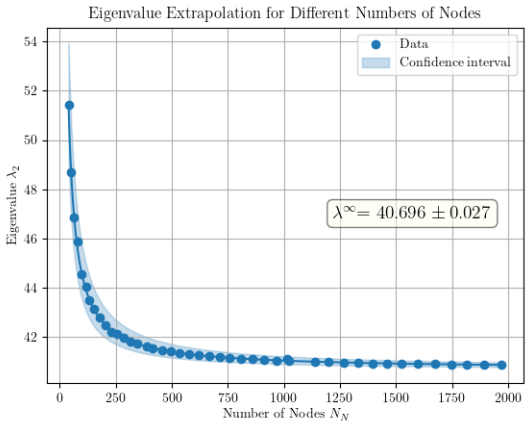

Figure 4: Extrapolation fit for the third eigenvalue.

points shown in green converge more towards ~ 2. This is slower than what we had last time for solving a system, but that is to be expected.

First we tackle the problem of calculating the eigenvalues for the semicircular membrane using FEM. As alluded to in the problem statement, we will do this using extrapolation of the results for $N_N \to \infty$.

From last time we know that the error in the values we're calculating doesn't quite decrease exponentially fast. So to extrapolate, we will use a power law dependence for the error. There is no way to make the power law fit that we want linear, so we will have to do a non-linear fit and take care to not take into account nonsensical results. Specifically, we will use the fit function

$$f(x) = \beta_0 + \beta_1 (x + \beta_2)^{-\beta_3}, \tag{13}$$

where β_0 is the value we're aiming for from this fit. For the case of $\beta_3 > 0$, which we're interested in, not including a β_2 fit coefficient would introduce a fixed singularity at the origin making the whole fit less stable and amplifying errors. Other fit models were tried, but this one worked best.

The number of nodes we will use vary in the interval $[\sim 30, \sim 2000]$. An example of this kind of fit is shown in fig. 4. We see that the calculated eigenvalue is always an overestimate of the real one or in other words $\beta_1 > 0$. This seems to be the most probable behavior but for some combinations of eigenvalue and mesh the reverse was also seen. Repeating the procedure for other eigenvalues in the first few we generate table 1. It shows the first 10 eigenvalues, their corresponding k and analytically calculated equivalent $\sqrt{\xi}$. δk denotes the statistical error of the extrapolation, while Δk is a measure of the accuracy, namely the difference between the calculated and analytical values. The relative error associated with the accuracy only goes

n	λ	k	δk	$\sqrt{\xi}$	Δk
0	14.69	3.833	0.001	3.832	0.001
1	26.415	5.14	0.001	5.136	0.004
2	40.696	6.379	0.002	6.38	0.001
3	49.441	7.031	0.05	7.016	0.016
4	57.771	7.601	0.005	7.588	0.012
5	71.011	8.427	0.014	8.417	0.01
6	76.047	8.72	0.012	8.772	0.051
7	92.704	9.628	0.04	9.761	0.133
8	97.749	9.887	0.022	9.936	0.049
9	102.369	10.118	0.024	10.174	0.056

Table 1: First 10 eigenvalues for the semicircular membrane. δk measures the statistical error from extrapolation, while Δk measures the difference from the analytical solution. Thus, the method can determine the correct eigenvalues to better than $\sim 1\%$.

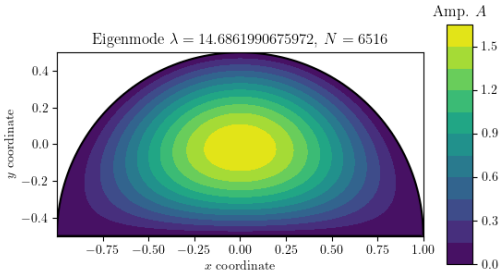

Figure 5: Profile of the first eigenmode calculated using FEM.

beyond 1% once and is usually much lower than that.

Of course, we can also calculate the eigenmodes themselves, a selection of which is shown in figs. 5 to 9. Qualitatively, they look like the analytical prediction, but we can say much more by normalizing and comparing. Here we will normalize to the maximum. The difference between the analytical solution and the calculated one is shown in figs. 10 and 11 for one lower end solution and one higher end solution. We can see that all around FEM does a good job of calculating the eigenmodes too.

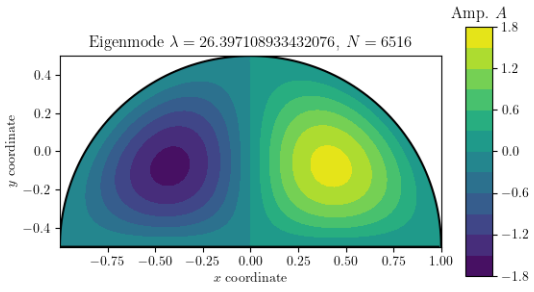

Figure 6: Profile of the second eigenmode calculated using FEM.

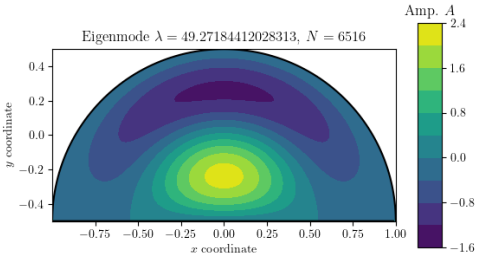

Figure 7: Profile of the fourth eigenmode calculated using FEM.

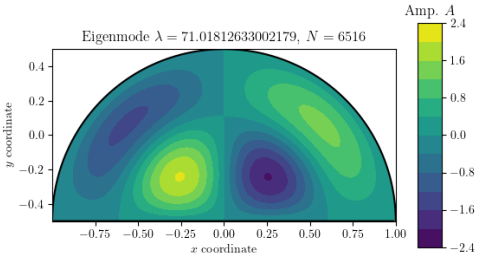

Figure 8: Profile of the sixth eigenmode calculated using FEM.

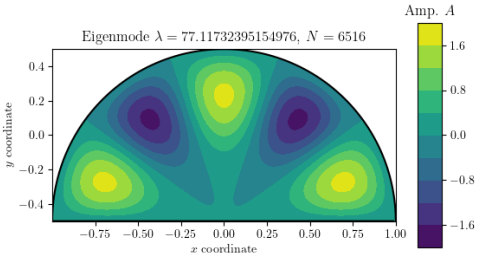

Figure 9: Profile of the seventh eigenmode calculated using FEM.

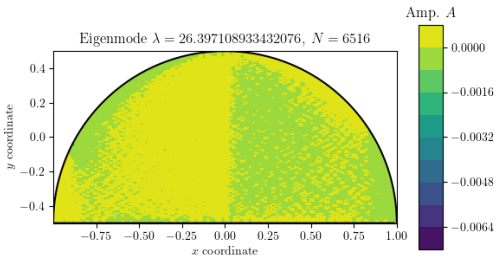

Figure 10: Difference from analytical for the second eigenmode calculated using FEM.

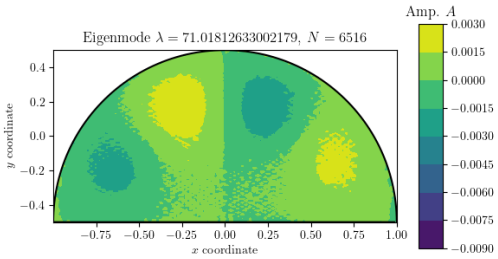

Figure 11: Difference from analytical for the sixth eigenmode calculated using FEM.

Continuing on with Galerkin, we need to calculate the elements of our matrices. The chosen basis has the nice property that the angular sin part is orthogonal for different basis functions so that contributes a $\delta_{mm'}$. This eventually means that the matrices will both be block-diagonal with each block representing one value of m. The radial part is not orthogonal, so it contributes a non-trivial factor to the matrices. After integration the formulas are

$$\delta_{mm'}\frac{\pi}{2}\left\{\frac{1}{2m+2+n+n'}-\frac{2}{2m+3+n+n'}+\frac{1}{2m+4+n+n'}\right\}, \tag{14}$$

for $B_{(mn)(m'n')}$ and

$$\delta_{mm'}\frac{\pi}{2}\left\{\frac{nn'}{2m+n+n'}-\frac{2nn'+n+n'}{2m+1+n+n'}+\frac{(n+1)(n'+1)}{2m+2+n+n'}\right\}, \tag{15}$$

for $A_{(mn)(m'n')}$.

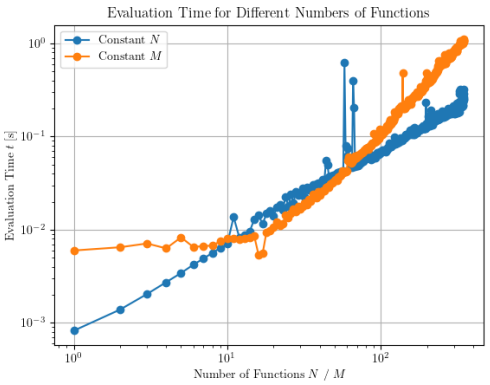

Figure 12: Evaluation time for the Galerkin method with our particular choice of basis functions.

With this setup, and the simplification for the angular part, we expect Galerkin to be $\mathcal{O}(MN^3)$, where M is the number of m values, and N is the number of n values. This is exactly what we see in practice as shown in fig. 12, with M constant we get a $\sim N^3$ dependence and with N constant we get a $\sim M$ dependence asymptotically.

Continuing on to the calculation of the eigenvalues we see the power of this method, table 2 shows the calculated and analytical values for the first 10 eigenvalues calculated with $N = 10$ and $M = 7$. The deviation from the analytical value is basically imperceptible. Accordingly, the asymptotic time dependence doesn't even play a role because we never care about values of N and M that are large.

n	λ	k	$\sqrt{\xi}$
0	14.68197064	3.83170597	3.83170597
1	26.37461643	5.135622302	5.135622302
2	40.70646582	6.380161896	6.380161896
3	49.21845632	7.01558667	7.01558667
4	57.5829409	7.588342435	7.588342435
5	70.84999892	8.417244141	8.41724414
6	76.93892833	8.771483816	8.771483816
7	95.27757281	9.761023144	9.76102313
8	98.72627248	9.936109524	9.936109524
9	103.4994653	10.1734687	10.17346814

Table 2: First 10 eigenvalues for the semicircular membrane calculated using the Galerkin guess with $N = 10$ and $M = 7$. There is basically no perceptible difference between the calculated and analytical values.

As the table would suggest, the eigenfunctions are similarly very precise. Here they are not plotted since qualitatively they look the same as the ones calculated with FEM.

The weakness of Galerkin, as discussed previously, is in the fact that it is very specialized to domains that have some symmetric ansatz to their eigenfunctions. Coming to the domain pictured in fig. 1 we can't even set up a family of basis functions that would satisfy the boundary conditions easily. The fact that's especially limiting is that one need to come up with a whole family of functions since for asymmetric domains we don't expect to cover even the first few eigenmodes with just a couple of functions.

This expectation will be confirmed when we calculate the actual eigenmodes of this domain.

We continue with FEM. Calculating with $N_N = 4232$ point we generate table 3. It shows the first 10 eigenvalues for this domain. It isn't highly enlightening except for the fact that the eigenvalues clearly start higher than for the semicircular membrane for example. This is an indication of the narrowness of the domain. A selection of eigenmodes is shown in figs. 13 to 18. The region outside of the domain is just a consequence of the triangulation and does not affect the inside in any way. The eighth mode particularly exhibits some symmetry the domain has, namely that it is built up out of eight $1/4 \times 1/4$ squares.

n	λ
0	106.33704621
1	150.33599144
2	193.40596707
3	218.94011083
4	221.42076162
5	251.33266070
6	286.48779241
7	317.24667339
8	375.01932035
9	414.31315590

Table 3: First 10 eigenvalues for the F-shaped membrane calculated using FEM with $N_N = 4232$ points.

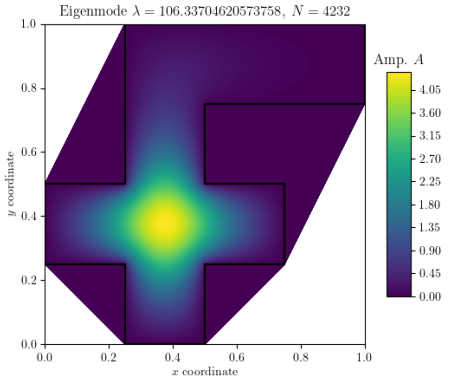

Figure 13: First eigenmode for F-shaped membrane calculated using FEM.

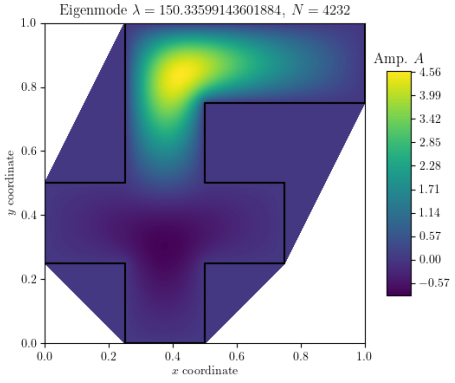

Figure 14: Second eigenmode for F-shaped membrane calculated using FEM.

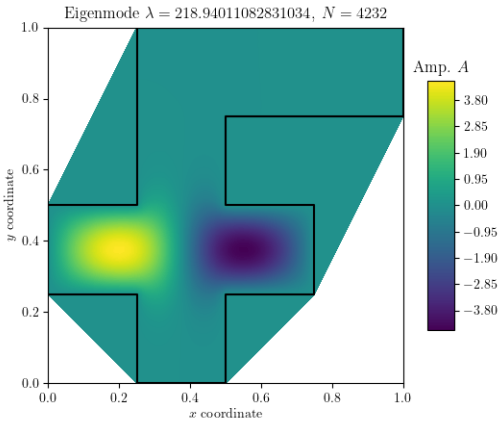

Figure 15: Fourth eigenmode for F-shaped membrane calculated using FEM.

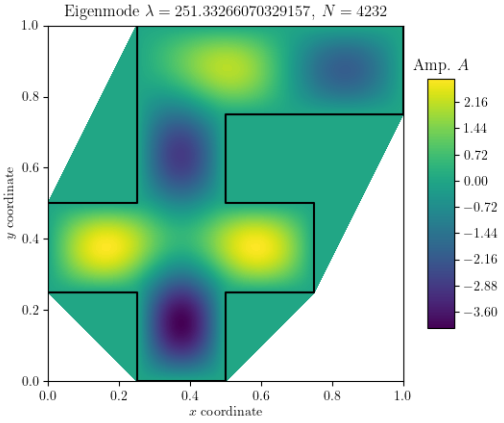

Figure 16: Sixth eigenmode for F-shaped membrane calculated using FEM.

Seeing the calculated shapes of the modes we partly confirm our expectation that the Galerkin method would need more basis functions because the eigenmodes are localized to parts of the F, and generically we would need more functions to cover this.

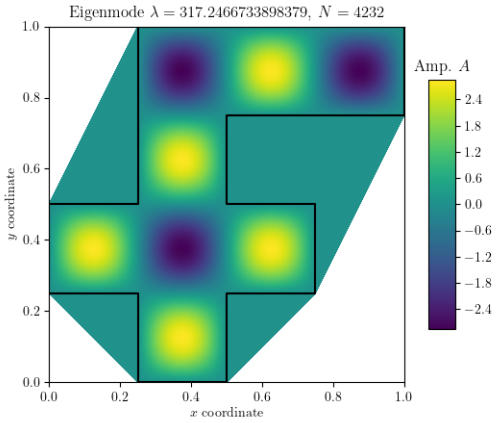

Figure 17: Eighth eigenmode for F-shaped membrane calculated using FEM.

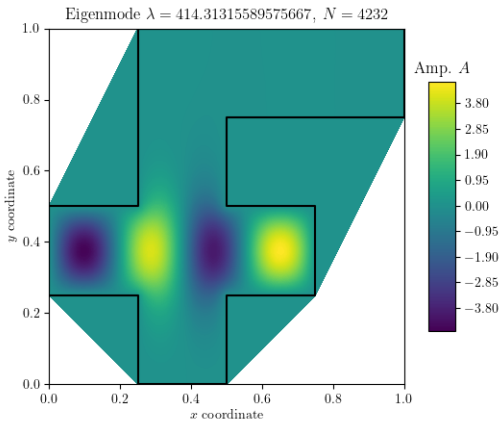

Figure 18: Tenth eigenmode for F-shaped membrane calculated using FEM.

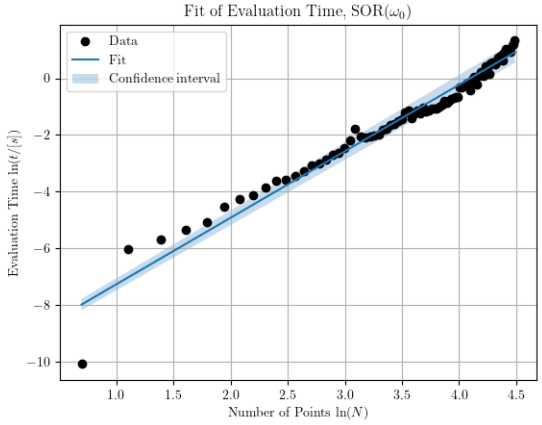

Figure 19: Evaluation time for SOR method at optimal parameter ω.

5 Comparing the methods

In regard to FEM and the Galerkin method, the previous sections have taught us the following. If we're solving a problem on some domain with enough symmetry, for which we can make an ansatz for a family of functions that satisfy the boundary conditions, that we're much better off using the Galerkin method. However, generically, when we have a domain for which we can't leverage symmetry for an ansatz, especially for multiple basis functions, then FEM presents a much better choice. In this respect it is the most robust of the methods we've examined. The strengths and weaknesses of these two methods are in a way complementary so that a direct comparison between the two would always be somewhat biased.

Along with these we also previously met SOR. It's most easily compared to FEM. The advantage of SOR is its ease of implementation. Regarding its evaluation time we previously found that it was $\sim (N^2)^{1.15}$, which is comparable to FEM's $\sim N_N^{1.07}$, which we found last time, so that in terms of speed they're asymptotically similar. However, FEM is slower in practice by around a factor of 10 due to the prefactor as can be seen between figs. 19 and 20. Where FEM does better, is in its handling of boundaries. Most other methods including SOR have a hard time with boundaries because it's not guaranteed that they'll have a point on the boundary itself to satisfy boundary conditions precisely. We saw this with SOR for the example of an astroid shape, where even after including corrections to take into account the boundary, we still had trouble with computing its Poiseuille coefficient. Even with the birdhouse-shaped region FEM outperformed SOR in the comparison from last time (Poiseuille coefficients: $C_{\text{FEM}} = 0.5892, C_{\text{SOR}} = 0.593.$, with FEM being consistent.).

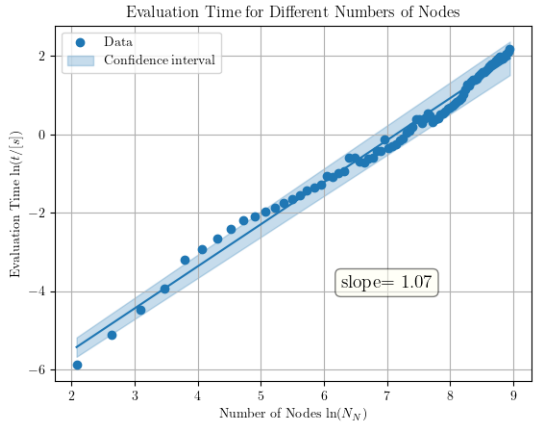

Figure 20: Evaluation time for FEM on the same problem as SOR.