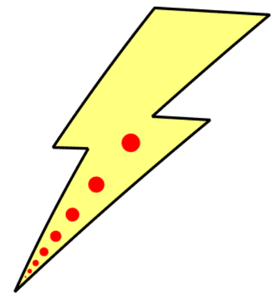
An Exploration of a Lightning-Fast Laplace Solver

by Jim Vargas
under direction of Dr. Jeff Ovall
Portland State University

This talk is based on...



Solving Laplace Problems with Corner Singularities via Rational Functions

- ...A paper written by Gopal and Trefethen, published in SIAM Journal on Numerical Analysis September 2019
- The Lightning Laplace code, based on the paper, yields accurate approximations quickly (on nice problems)
- https://epubs.siam.org/doi/pdf/10.1137/19M125947X
- https://people.maths.ox.ac.uk/trefethen/lightning.html

Here's the problem

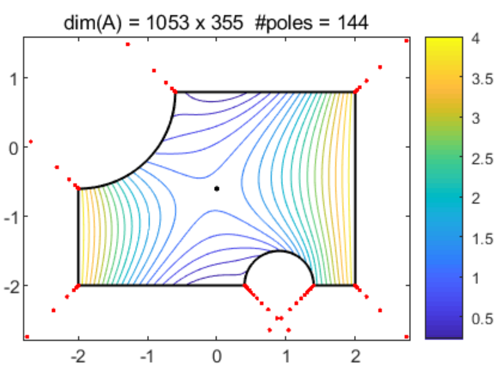
We wish to find a (real) function u over a domain $\Omega \subset \mathbb{C}$ which satisfies

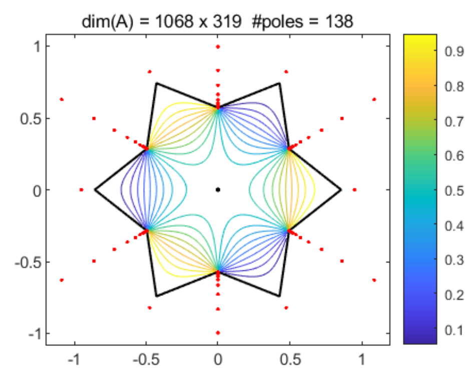
$$\Delta u(z) = 0, \quad z \in \Omega \qquad\qquad u(z) = h(z), \quad z \in \Gamma.$$

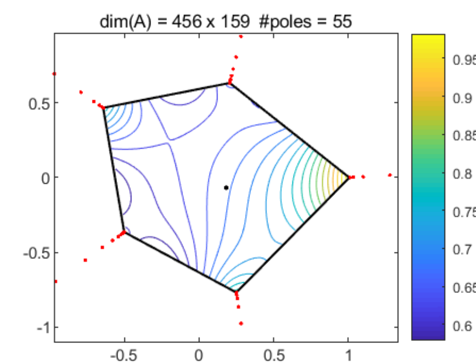
In particular, we want to be able to handle a domain with sharp corners, curves etc.

We will find r, and approximation of u ($u \approx \text{Re}[r]$).

$$r(z) = \sum_{j=1}^{N_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{N_2} b_j (z - z_*)^j$$







Why this problem?

- Problems involving the Laplace operator $\Delta = \nabla^2$ frequently appear in physical equations:
  - Heat Equation $\alpha \nabla^2 u = \partial_t u$
  - Schrodinger Equation $\left[\frac{-\hbar^2}{2m}\nabla^2 + V\right]\Psi = i\hbar\,\partial_t \Psi$
  - Wave Equation $c^2 \nabla^2 u = \partial_t^2 u$
  - And more...
- Functions which satisfy Laplace's Equation have very nice properties, and are called harmonic.

Some nice properties of functions of interest

- The real and imaginary parts of a holomorphic (and thus also an analytic) function $f = u + iv$ are harmonic;
- f is also smooth (infinitely differentiable); by extension this applies to u and v as well.
- Maximum Principle: a harmonic function on a compact domain attains a max. (and min.) on the boundary.

Some nice properties of functions of interest

- The real and imaginary parts of a holomorphic (and thus also an analytic) function $f = u + iv$ are harmonic;
- f is also smooth (infinitely differentiable); by extension this applies to u and v as well.
- Maximum Principle: a harmonic function on a compact domain attains a max. (and min.) on the boundary.

On a simply connected domain we can construct a holomorphic function from a harmonic one: given u, define $g = u - iu$. The theory will work with holomorphic functions, which will trickle down to our problem.

If r approximates f, having real part u, the worst we'll do over the whole domain in approximating u is $||\, u(z) - \mathrm{Re}[r(z)]\, ||_\infty$, $z \in \Gamma$.

Back to the problem

$$r(z) = \underbrace{\sum_{j=1}^{N_1} \frac{a_j}{z - z_j}}_{\text{"Newman"}} + \underbrace{\sum_{j=0}^{N_2} b_j (z - z_*)^j}_{\text{"Runge"}}$$

- Using the scheme in the paper, we can have root exponentially good approximations for u. The task at hand is finding the coefficients a_j, b_j.

$$||f - r_n||_\Omega = O(e^{-C\sqrt{n}})$$

- The theorems in the paper are based on interpolation, showing existence.
- In the code, the problem is solved via a least squares approach using QR factorization. Code is written in MATLAB.

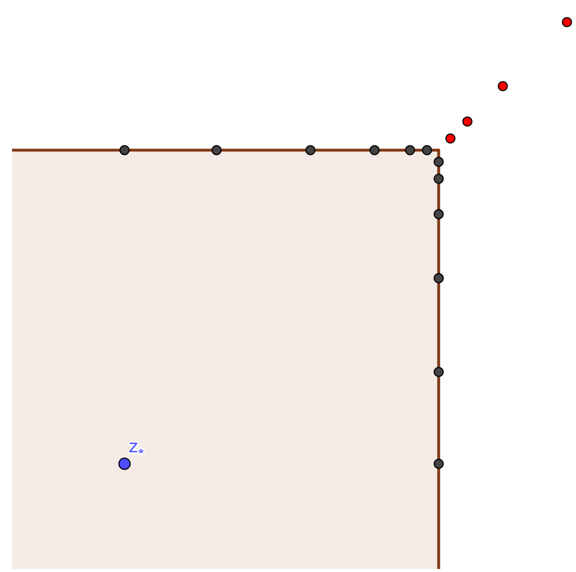
$$\min_{\substack{\{a_1 \ldots, a_{N_1}\} \\ \{b_1 \ldots, b_{N_2}\}}} \sum_{j=0}^{M} |r(y_j) - h(y_j)|^2$$

Describing r

$$r(z) = \sum_{j=1}^{N_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{N_2} b_j (z - z_*)^j$$

Newman Part: built to handle corners.

- The terms z_j are poles, exponentially clustered near a corner on the exterior of Ω (works for spacing scaled at least $O(n^{-1/2})$).
- "Rational functions are more powerful than polynomials for approximating functions near singularities..."[a]



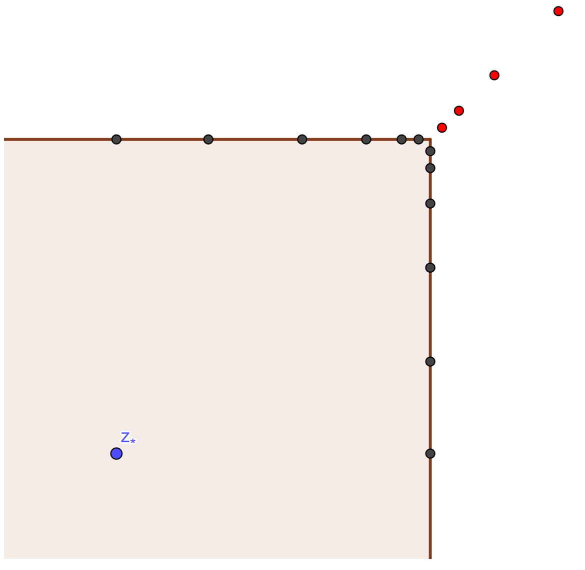
[a] Lloyd N. Trefethen. 2013. *Approximation theory and approximation practice*, Society for Industrial and Applied Mathematics.

Describing r

$$r(z) = \sum_{j=1}^{N_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{N_2} b_j (z - z_*)^j$$

The Runge part: built to handle the interior.

- The term z_* is an expansion point, near the middle of Ω.
- Polynomials can approximate root exponentially well on a nice domain (going back to Runge).



The function r is harmonic

$$r(z) = \sum_{j=1}^{N_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{N_2} b_j (z - z_*)^j$$

To prove r is harmonic, consider $f(z) = 1/z$ and $g(z) = z^k$. The function f can be decomposed as $f = u + iv$, where

$$u(x, y) = \frac{x}{x^2 + y^2} \qquad v(x, y) = \frac{-y}{x^2 + y^2}.$$

Taking derivatives will show that u and v satisfy the Cauchy-Riemann equations, $\partial_x u = \partial_y v$, $\partial_y u = -\partial_x v$, meaning f is (holomorphic, and thus) harmonic.

The function r is harmonic

$$r(z) = \sum_{j=1}^{N_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{N_2} b_j (z - z_*)^j$$

To prove r is harmonic, consider $f(z) = 1/z$ and $g(z) = z^k$. The function f can be decomposed as $f = u + iv$, where

$$u(x, y) = \frac{x}{x^2 + y^2} \qquad v(x, y) = \frac{-y}{x^2 + y^2}.$$

Taking derivatives will show that u and v satisfy the Cauchy-Riemann equations, $\partial_x u = \partial_y v$, $\partial_y u = -\partial_x v$, meaning f is (holomorphic, and thus) harmonic.

Writing g in polar form, then in terms of sines and cosines is enough to see g is harmonic:

$$g(z) = \rho e^{ik\theta} = \rho[\cos(k\theta) + i \sin(k\theta)].$$

Adding these templates, applying translations and scaling as necessary give us our result.

An important lemma

Hermite integral formula for rational interpolation.

Let Ω be a simply connected domain in $\mathbb{C}$ bounded by a closed curve Γ, and let f be analytic in that domain and extend continuously to the boundary. Let interpolation points $\alpha_0, \ldots, \alpha_{n-1} \in \Omega$ and poles $\beta_0, \ldots, \beta_{n-1}$ anywhere in the complex plane be given. Let r be the unique type $(n-1, n)$ rational function with simple poles at $\{\beta_j\}$ that interpolate f at $\{\alpha_j\}$. Then for any $z \in \Omega$,

$$f(z) - r(z) = \frac{1}{2\pi i} \int_\Gamma \frac{\phi(z)}{\phi(t)} \frac{f(t)}{t-z} dt,$$

$$\phi(z) = \prod_{j=0}^{n-1} (z - \alpha_j) \Big/ \prod_{j=0}^{n-1} (z - \beta_j).$$

First Theorem

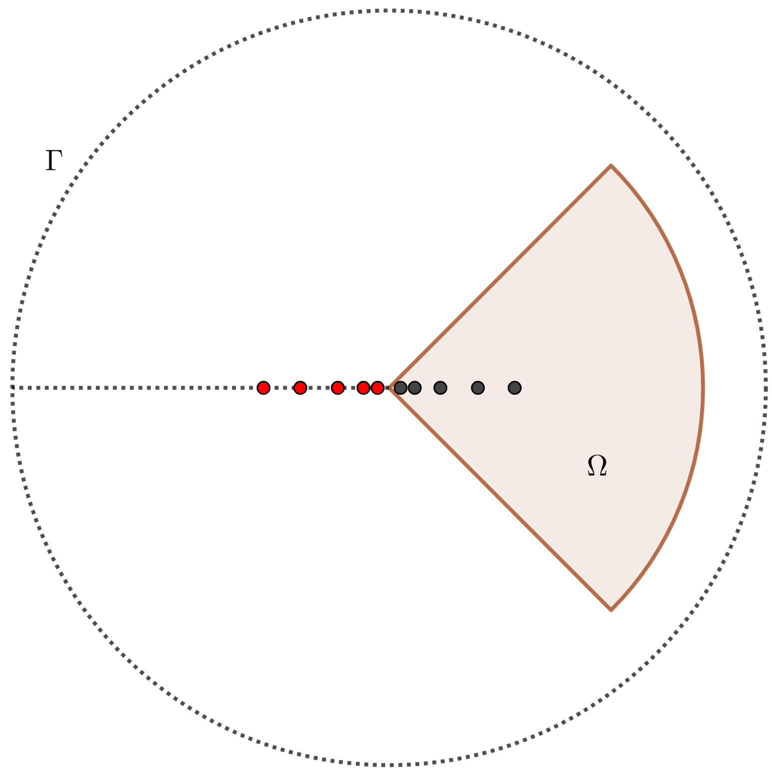
Let f be a bounded analytic function in the slit disk A_π that satisfies $f(z) = O(|z|^\delta)$ as $z \to 0$ for some $\delta > 0$, and let $\theta \in (0, \pi/2)$ be fixed. Then for some $0 < \rho < 1$ depending on θ but not on f, there exist type $(n-1, n)$ rational functions $\{r_n\}$, $1 \leq n < \infty$, such that

$$||f - r_n||_\Omega = O(e^{-C\sqrt{n}})$$

as $n \to \infty$ for some $C > 0$, where $\Omega = \rho A_\theta$. Moreover, each r_n can be taken to have simple poles only at

$$\beta_j = -e^{-\sigma j/\sqrt{n}}, \ \ 0 \leq j \leq n-1,$$

where $\sigma > 0$ is arbitrary.



$$A_\theta = \{z \in \mathbb{C} : |z| < 1, \;\; |\mathrm{arg}(z)| < \theta\} \qquad\qquad \Omega = \rho A_\theta$$

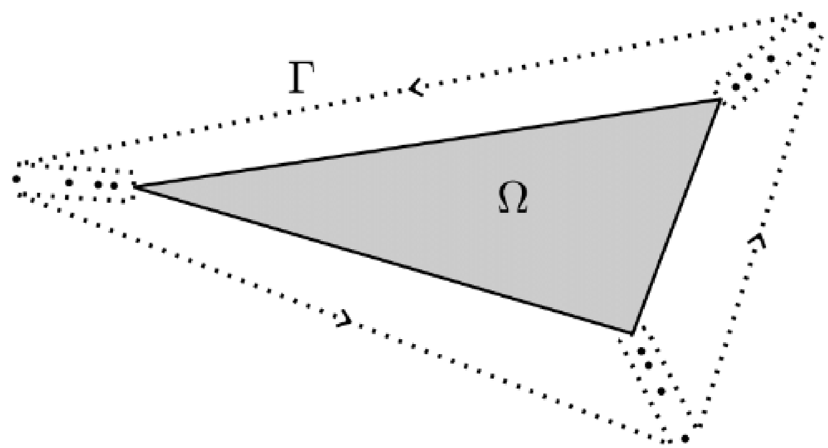
Second Theorem

Let Ω be a convex polygon with corners $w_1, \ldots, w_m$, and let f be an analytic function in Ω that is analytic on the interior of each side segment and can be analytically continued to a disk near each w_k with a slit along the exterior bisector there. Assume f satisfies $f(z) - f(w_k) = O(|z - w_k|^\delta)$ as $z \to w_k$ for each k for some $\delta > 0$. There exist degree n rational functions $\{r_n\}$, $1 \le n < \infty$ such that

$$||f - r_n||_\Omega = O(e^{-C\sqrt{n}})$$

as $n \to \infty$ for some $C > 0$. Moreover, each r_n can be taken to have finite poles only at points exponentially clustered along the exterior bisectors at the corners, with arbitrary clustering parameter σ, as long as the number of poles near each w_k grows at least in proportion to n as $n \to \infty$.

Second Theorem: the idea[a]



Split f into $2m$ terms, a "Newman" part and a "Runge" part:

$$f = \sum_{k=1}^{m} f_k + \sum_{k=1}^{m} g_k.$$

The Runge part can be handled by previously established results, and the Newman part can be handled by applying the first theorem to each corner.

[a]Image from Gopal, A., & Trefethen, L. N. (2019). *Solving Laplace Problems with Corner Singularities via Rational Functions*. SIAM Journal on Numerical Analysis.

Some extensions

Numerical experiments show that:

- We can get root exponentially good approximations on non-convex domains;
- We're not limited to sectors and convex polygons, we can have curvy edges.

These theorems apply to a holomorphic function f, but our problem involves a harmonic u.

If we assume u satisfies the corner behavior needed and Ω is simply connected, then so will a v, where we can have an $f = u + iv$.

The Algorithm

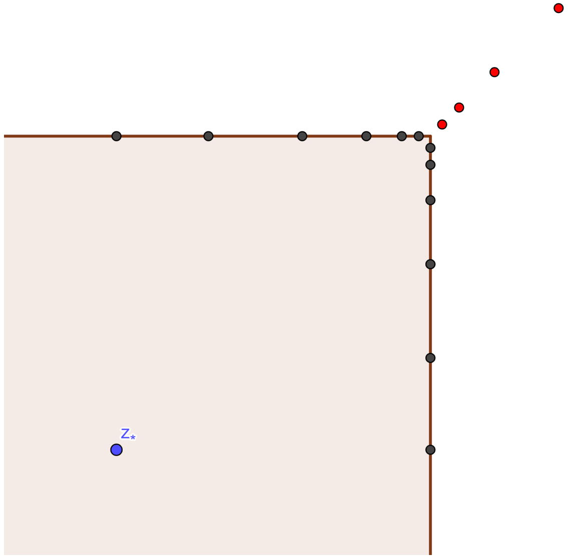
1. Define boundary Γ, corners $w_1, \ldots, w_m$, boundary function h, tolerance ε.
2. For increasing values of n with $\sqrt{n}$ approximately evenly spaced;
   - 2a. fix $N_1 = O(mn)$ poles $1/(z - z_k)$ clustered outside the corners;
   - 2b. fix $N_2 + 1 = O(n)$ monomials $1, (z - z_*), \ldots, (z - z_*)^{N_2}$ and set $N = N_1 + N_2 + 1$;
   - 2c. choose $M \approx 3N$ sample points on a boundary, also clustered near corners;
   - 2d. evaluate at sample points to obtain an $M \times N$ matrix A and M-vector b;
   - 2e. solve the least-squares problem $Ax \approx b$ for the coefficient vector x;
   - 2f. exit loop if $||Ax - b||_\infty < \varepsilon$ or if N is too large or the error is growing.
3. Confirm accuracy by checking the error on a finer boundary mesh.
4. Construct a function to evaluate $r(z)$ based on computed coefficients x.

The code

Code is branded "Lightning Laplace." We enter:

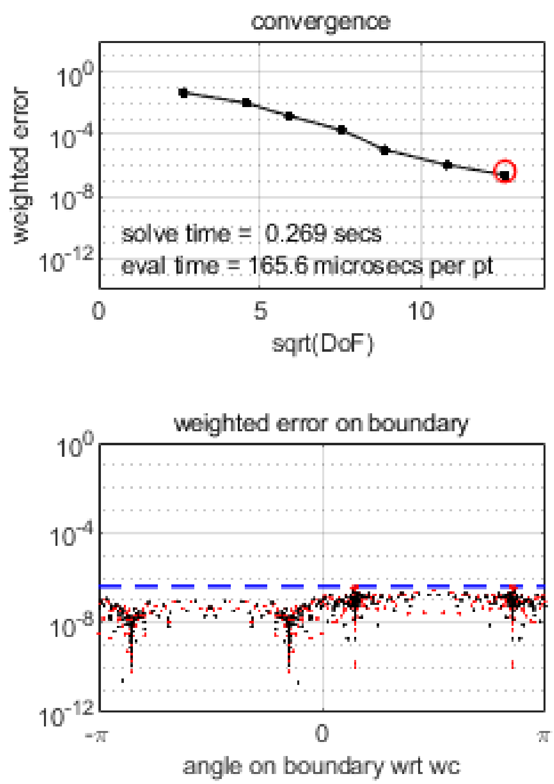
- Corners of a polygonal-ish/curvy domain in $\mathbb{C}$;
- boundary data in the form of a(n) real function handle(s), or scalar values, corresponding to the edges.

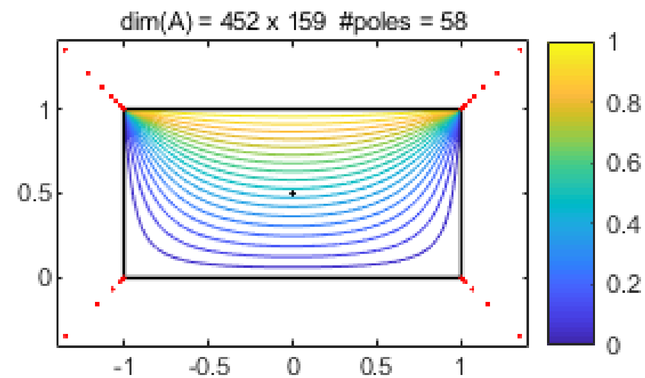
Errors are computed by comparing the procedure with a finer sampling (so not a true error).

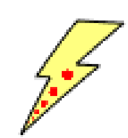


```
disp('rectangle with piecewise constant BCs')
P = [-1 1 1+1i -1+1i];
h = [0 0 1 0];
laplace(P,h,'plots','rel');
```

Solve time: 0.269s, Epsilon: 1e-6, dim(A)=452 x 159, #poles: 58

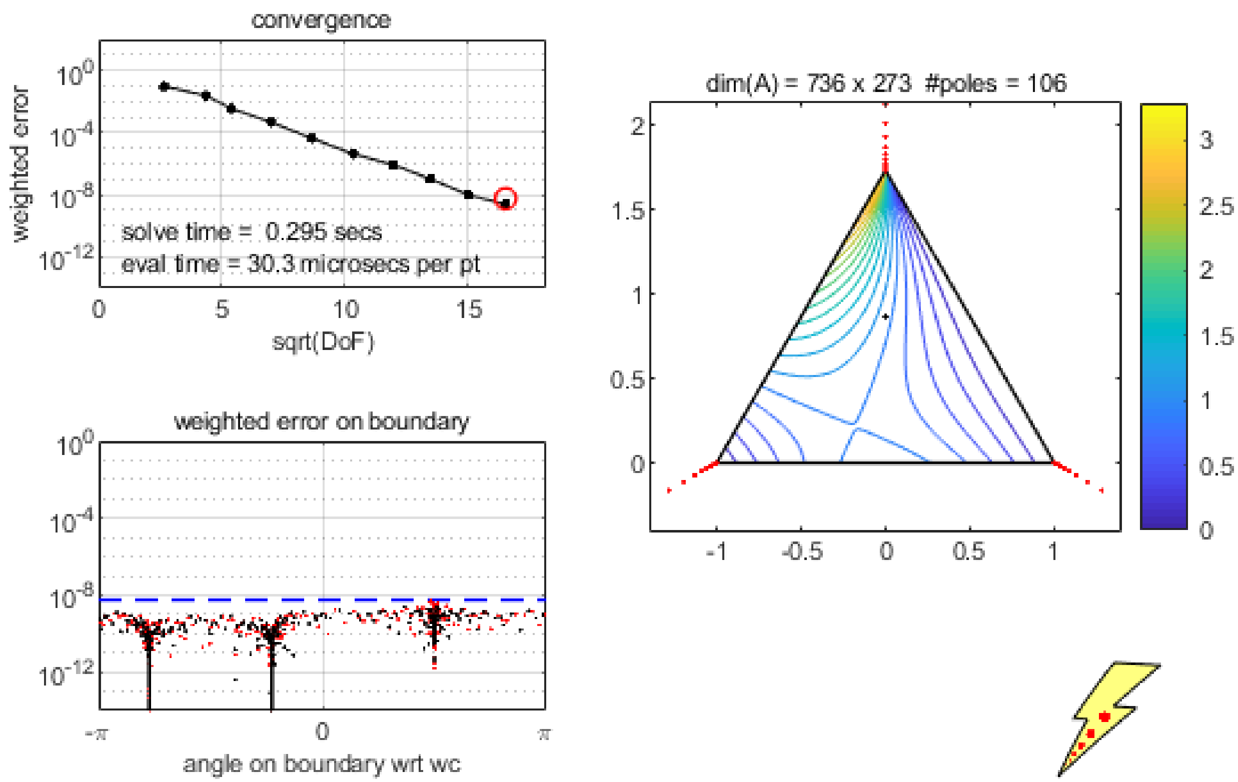


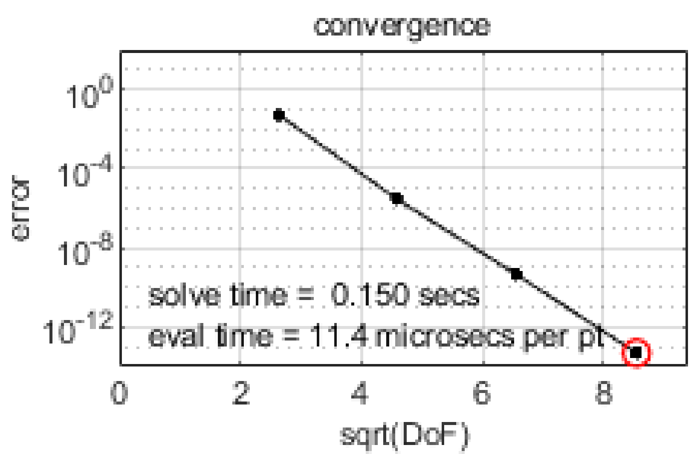


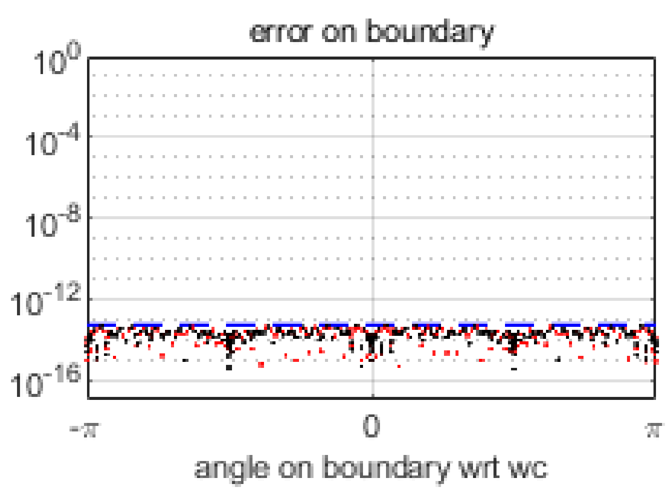


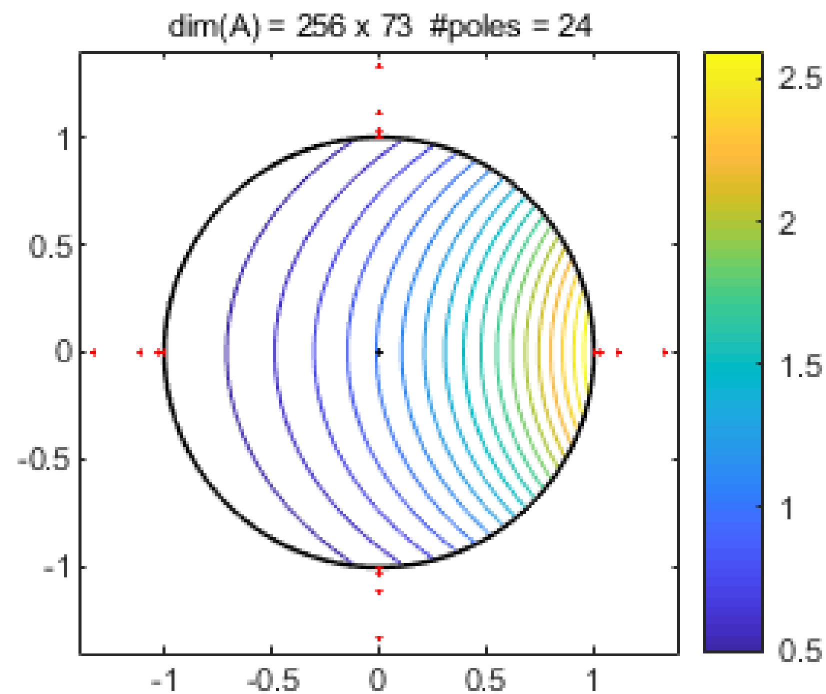
```
disp('equilateral triangle with one non-constant BC')
h = {@(x) cos(pi*x/2), @(z) 0*z, @(z) 2*imag(z)};
P = [-1 1 1i*sqrt(3)];
laplace(P,h,'plots','tol',1e-8);
```

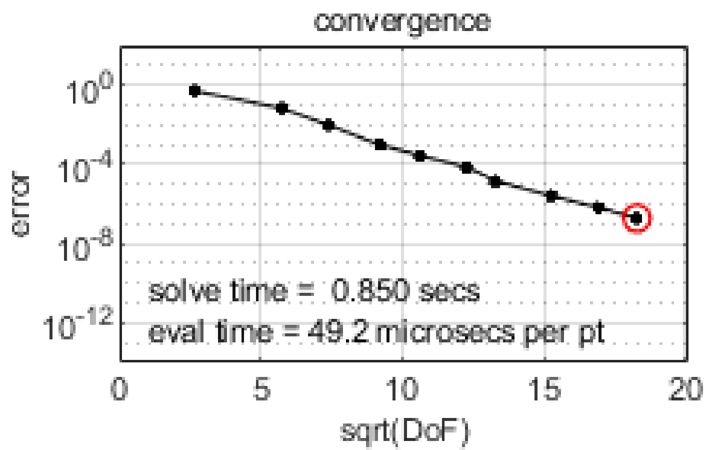
Solve time: 0.295s. Epsilon: 1e-8, dim(A)=736 × 273, #poles: 106

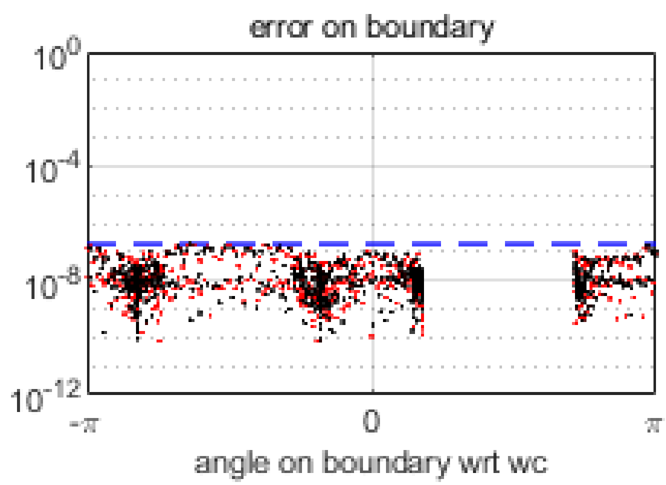


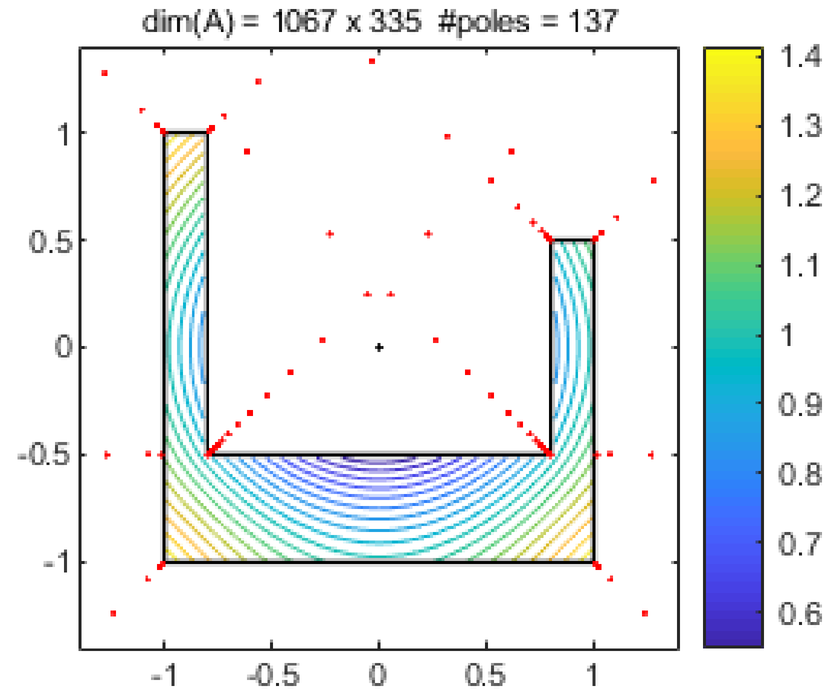

convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 0.150 secs
eval time = 11.4 microsecs per pt
0
2
4
6
8
sqrt(DoF)

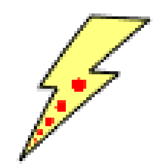

error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
10^{-16}
$-\pi$
0
π
angle on boundary wrt wc

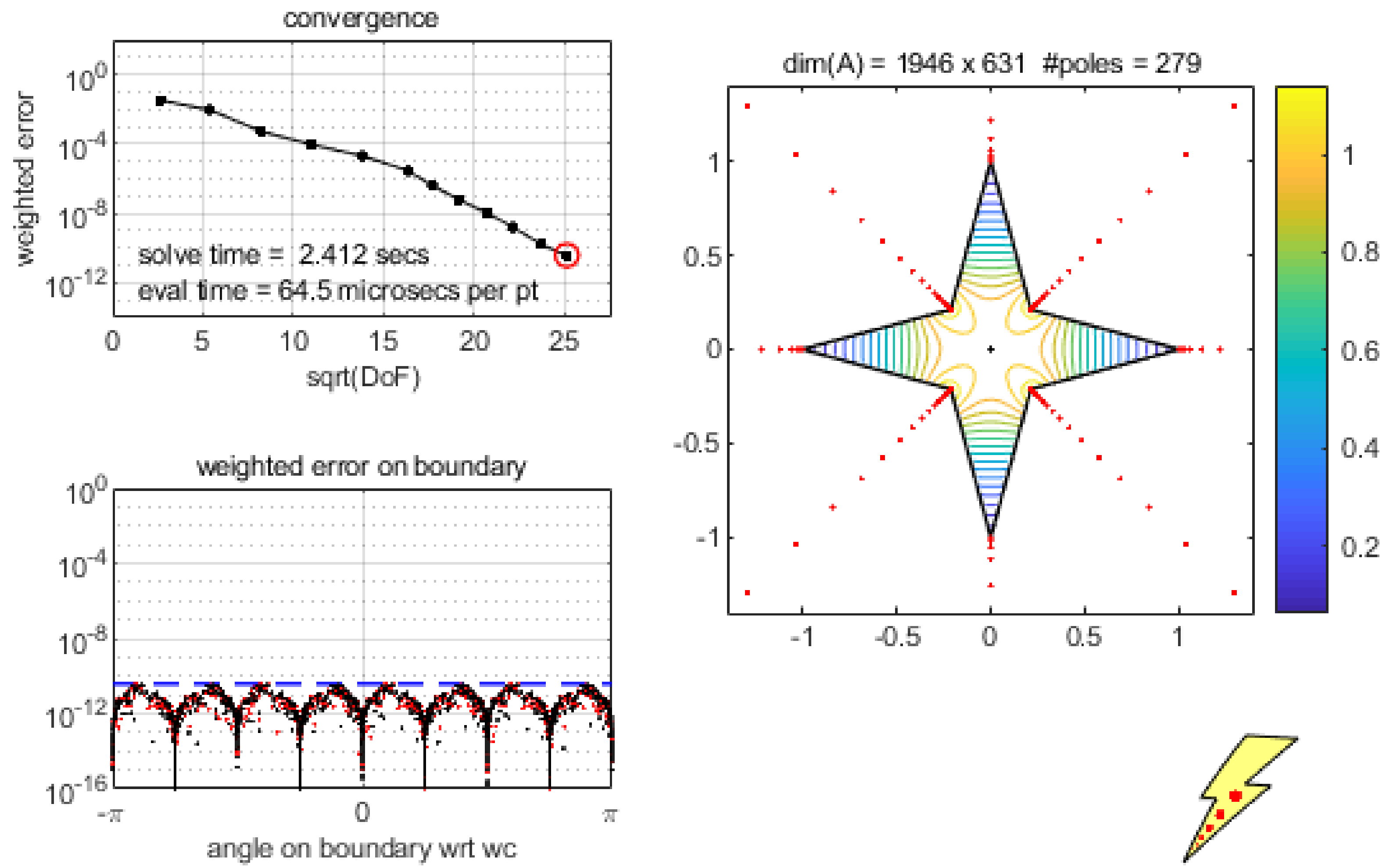

dim(A) = 256 x 73 #poles = 24
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
2.5
2
1.5
1
0.5

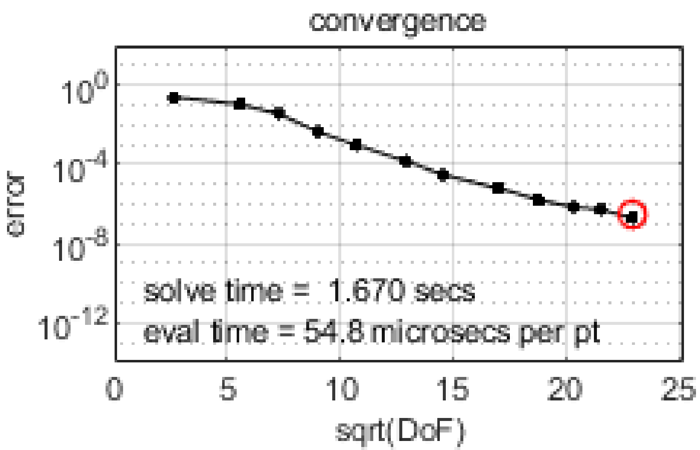

convergence
error
10^0
10^{-4}
10^{-8}
10^{-12}
solve time = 0.850 secs
eval time = 49.2 microsecs per pt
0
5
10
15
20
sqrt(DoF)

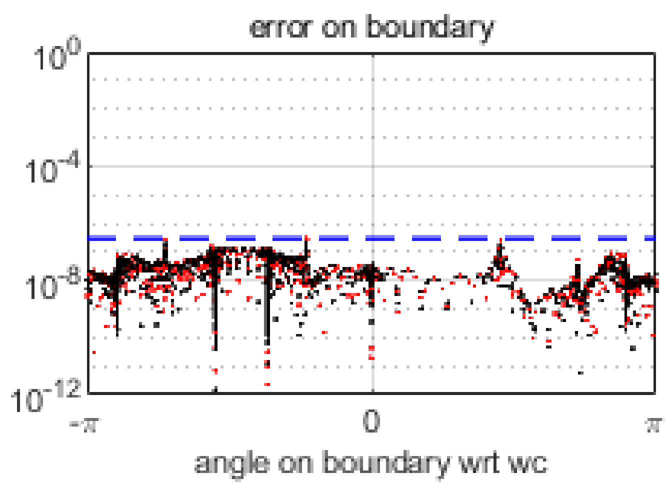

error on boundary
10^0
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

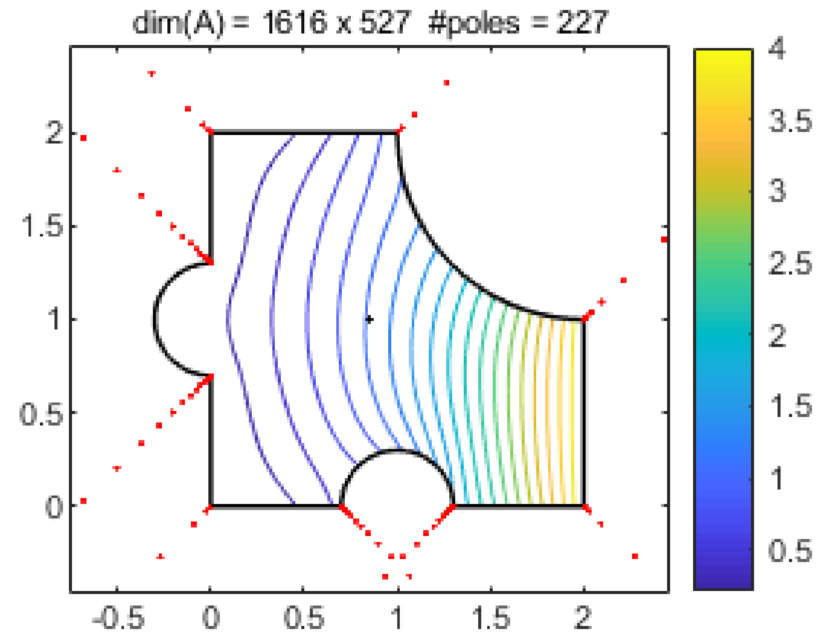

dim(A) = 1067 x 335 #poles = 137
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6

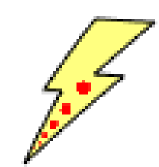


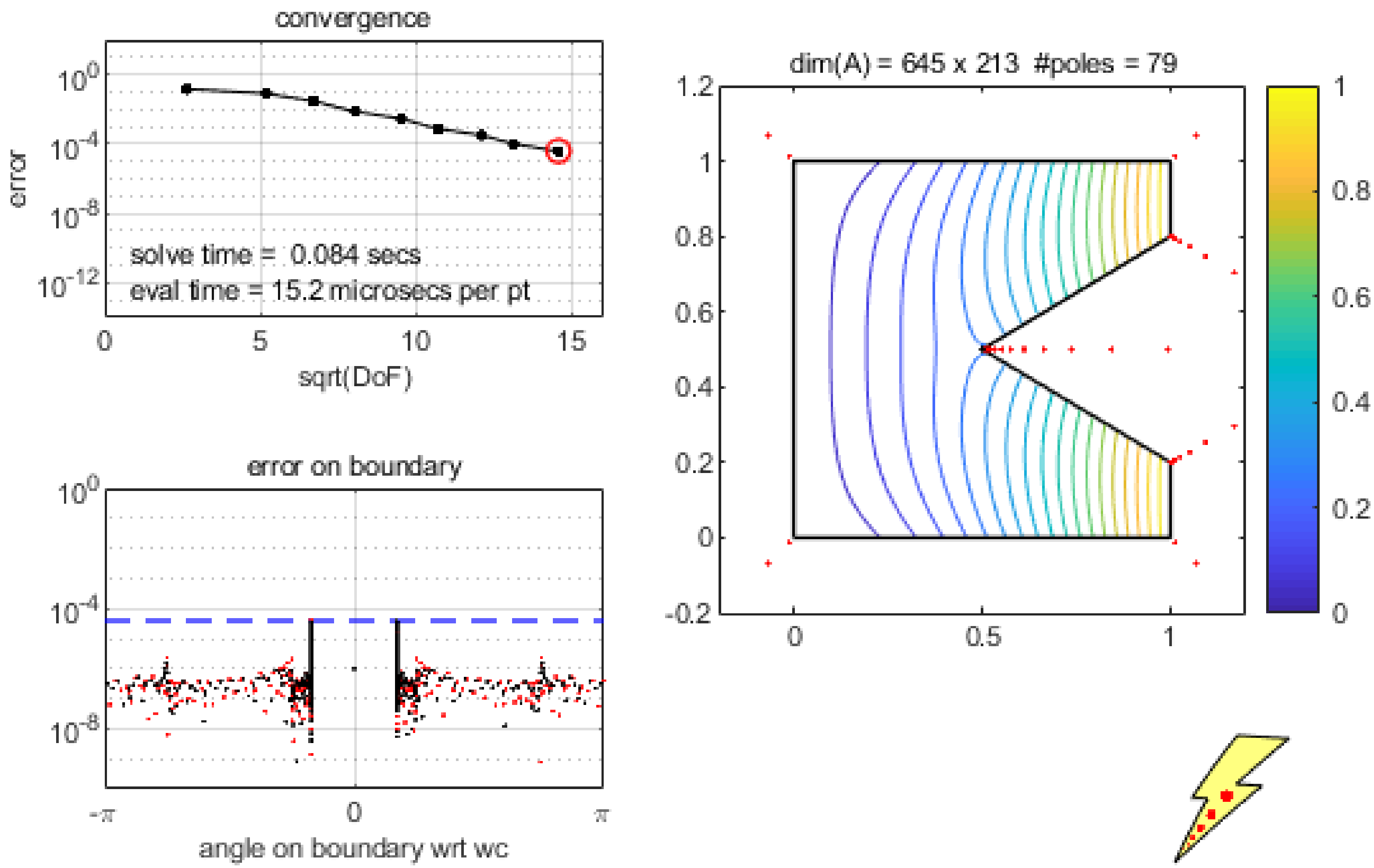

convergence
weighted error
sqrt(DoF)
solve time = 2.412 secs
eval time = 64.5 microsecs per pt
weighted error on boundary
angle on boundary wrt wc
dim(A) = 1946 x 631 #poles = 279

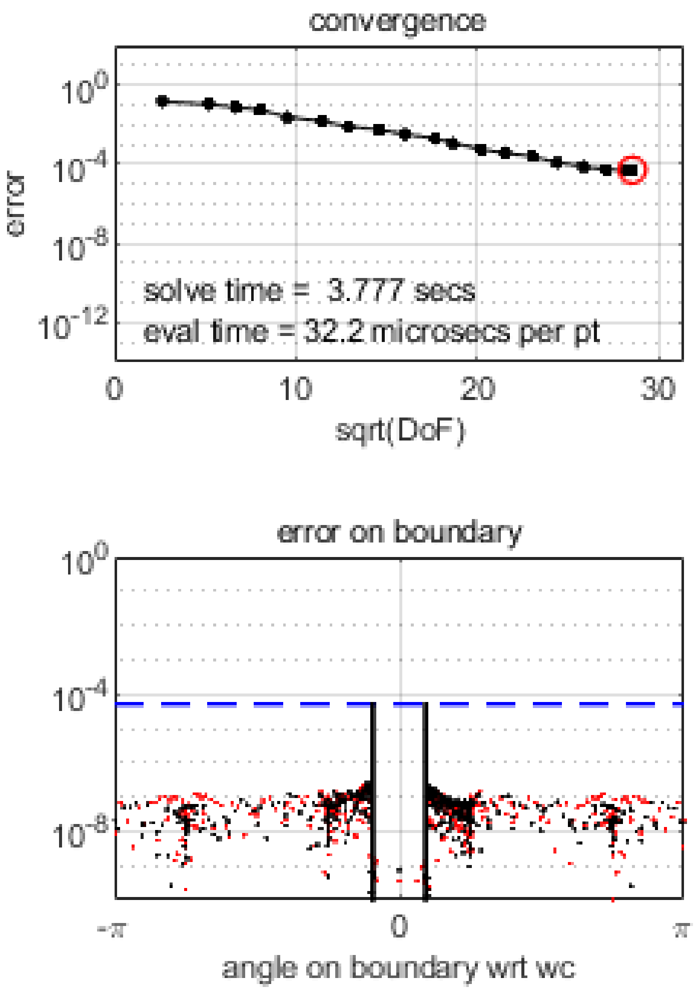

convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 1.670 secs
eval time = 54.8 microsecs per pt
0
5
10
15
20
25
sqrt(DoF)

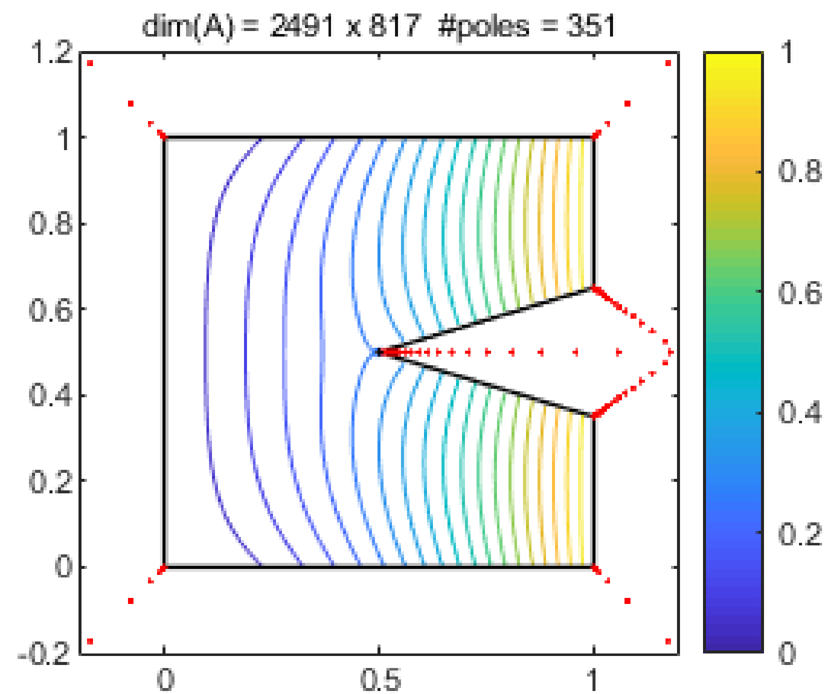

error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

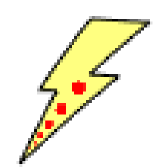

dim(A) = 1616 x 527 #poles = 227
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
4
3.5
3
2.5
2
1.5
1
0.5

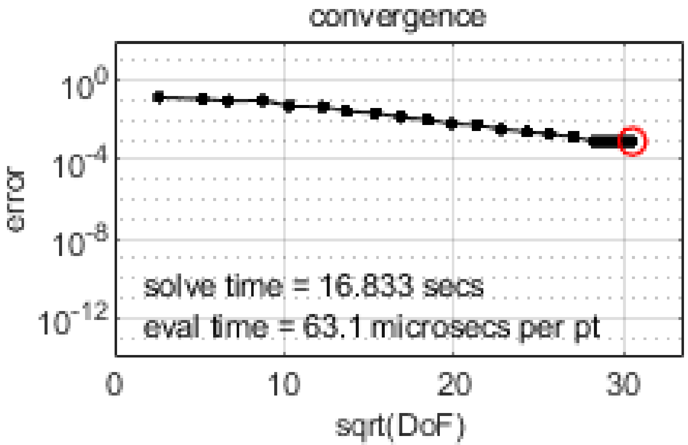


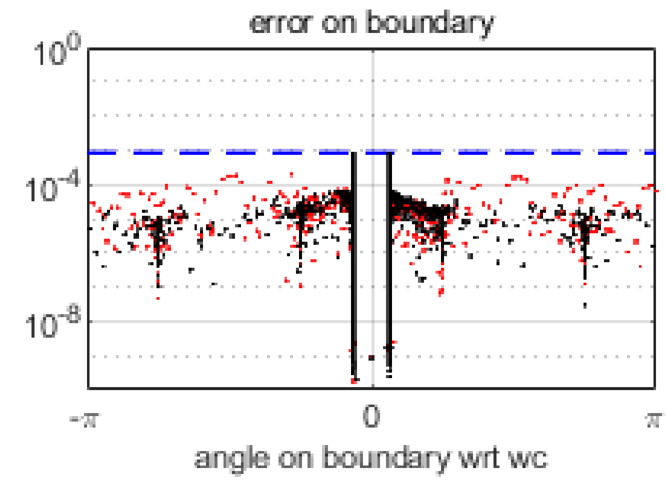

convergence
error
sqrt(DoF)
solve time = 0.084 secs
eval time = 15.2 microsecs per pt
error on boundary
angle on boundary wrt wc
dim(A) = 645 x 213 #poles = 79

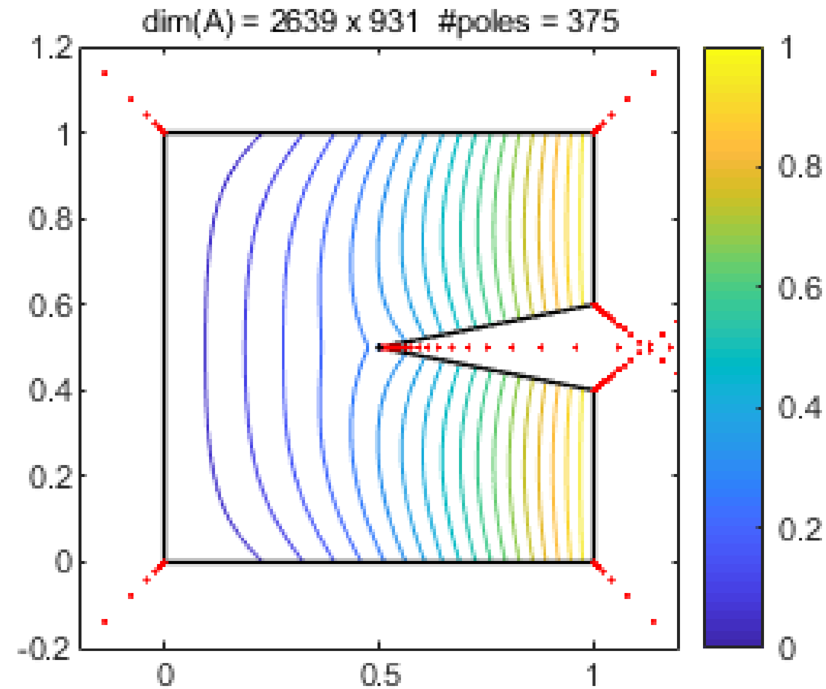

convergence
error
10^0
10^{-4}
10^{-8}
10^{-12}
solve time = 3.777 secs
eval time = 32.2 microsecs per pt
0
10
20
30
sqrt(DoF)
error on boundary
10^0
10^{-4}
10^{-8}
$-\pi$
0
π
angle on boundary wrt wc

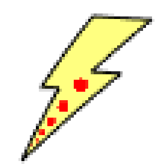

dim(A) = 2491 x 817 #poles = 351
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.8
0.6
0.4
0.2
0

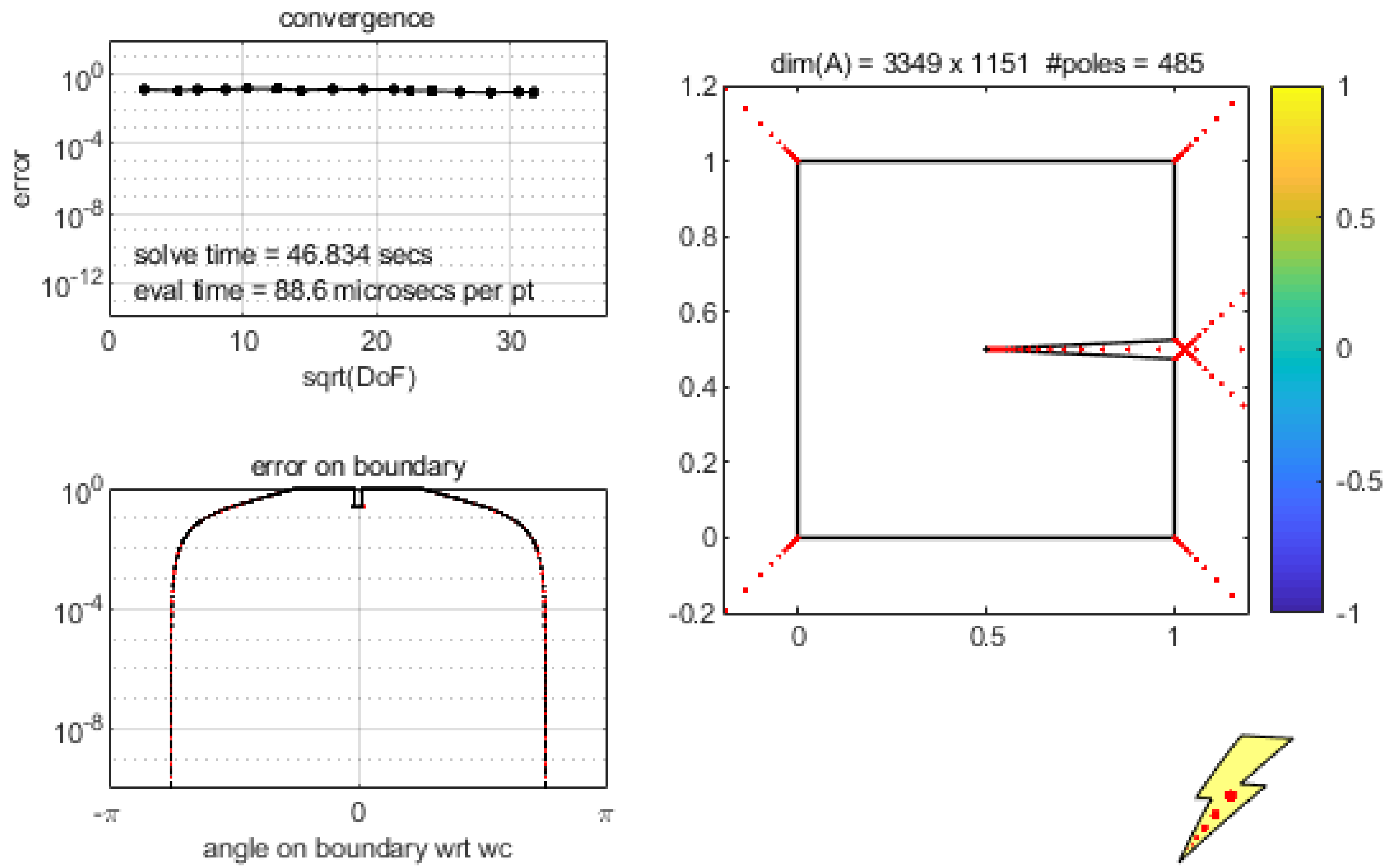


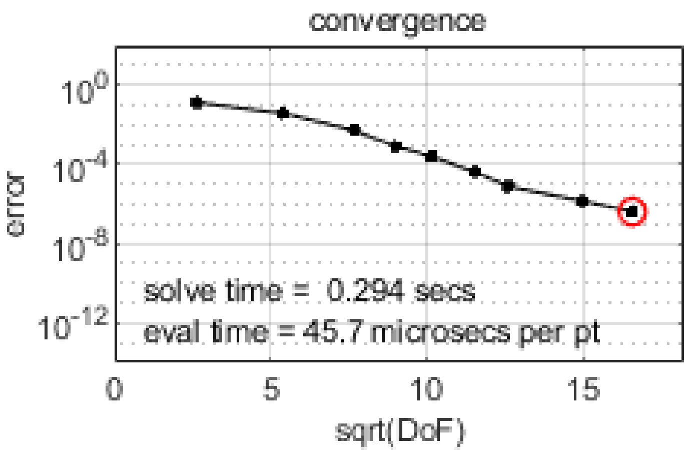

convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 16.833 secs
eval time = 63.1 microsecs per pt
0
10
20
30
sqrt(DoF)

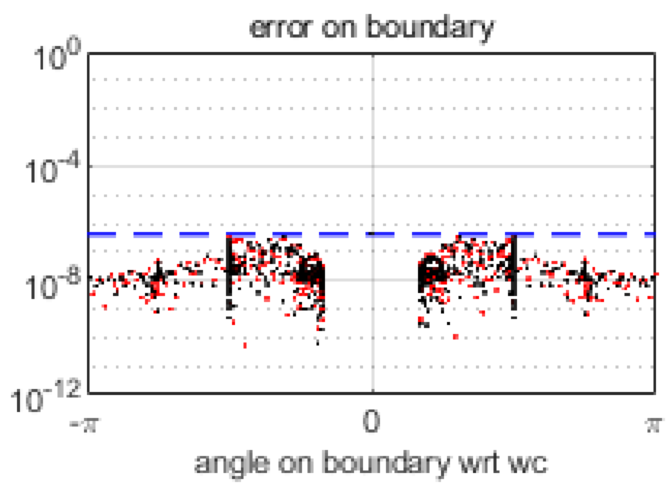

error on boundary
10^{0}
10^{-4}
10^{-8}
$-\pi$
0
π
angle on boundary wrt wc

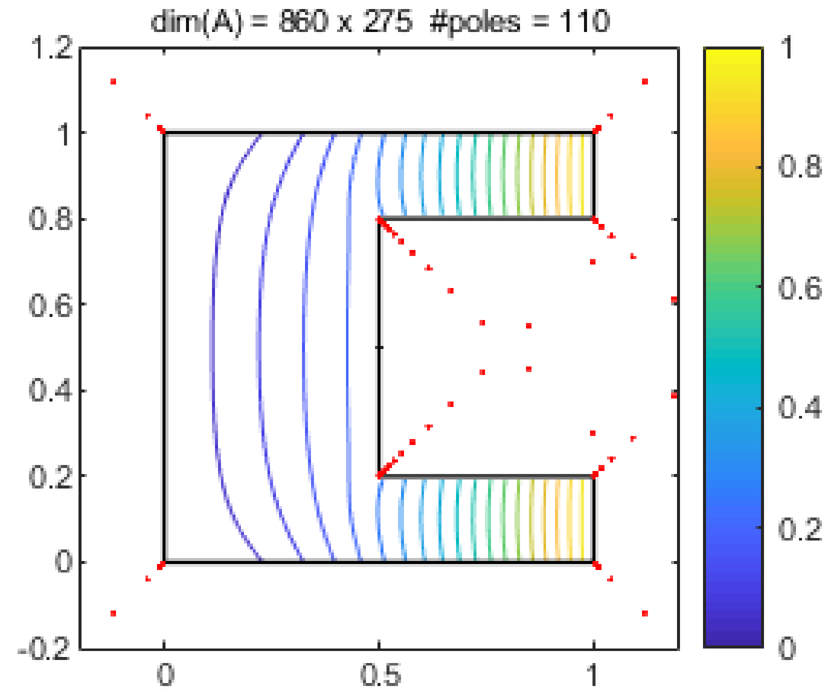

dim(A) = 2639 x 931 #poles = 375
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.8
0.6
0.4
0.2
0

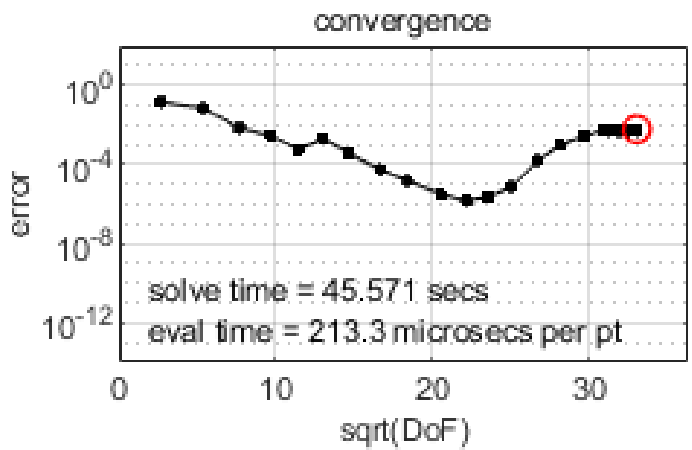



convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 46.834 secs
eval time = 88.6 microsecs per pt
0
10
20
30
sqrt(DoF)
error on boundary
10^{0}
10^{-4}
10^{-8}
$-\pi$
0
π
angle on boundary wrt wc
dim(A) = 3349 x 1151 #poles = 485
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.5
0
-0.5
-1

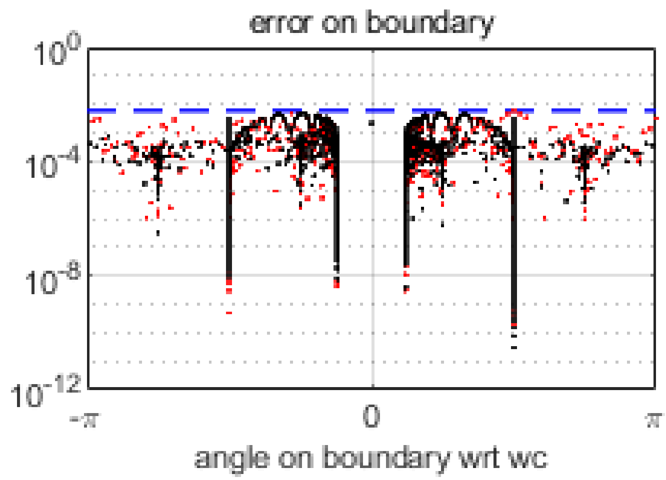

convergence
error
10^0
10^{-4}
10^{-8}
10^{-12}
solve time = 0.294 secs
eval time = 45.7 microsecs per pt
0
5
10
15
sqrt(DoF)


error on boundary
10^0
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

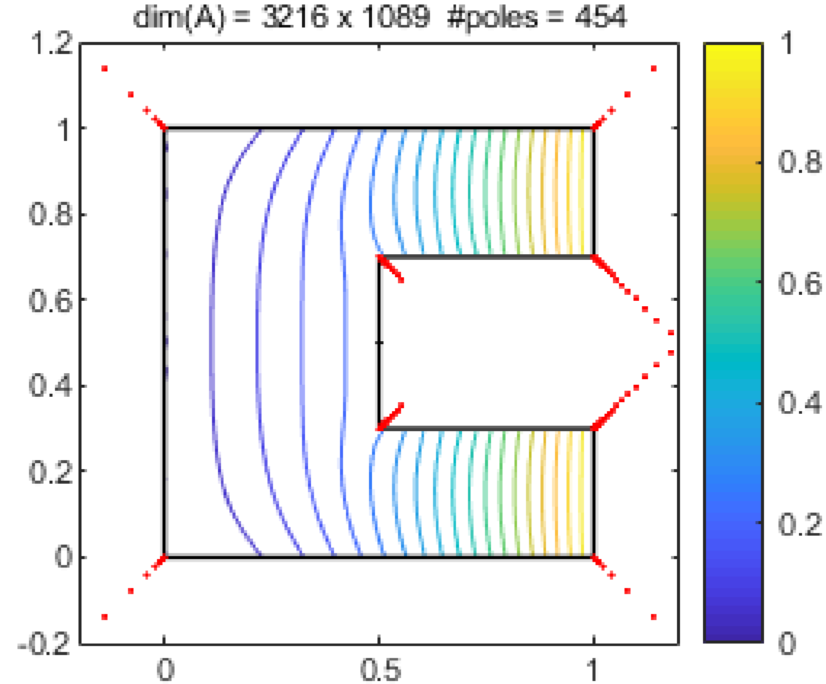

dim(A) = 860 x 275 #poles = 110
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.8
0.6
0.4
0.2
0



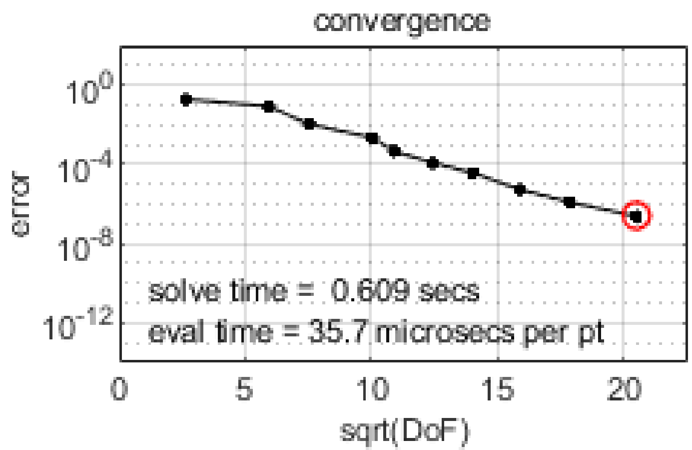
convergence
error
10^0
10^{-4}
10^{-8}
10^{-12}
solve time = 45.571 secs
eval time = 213.3 microsecs per pt
0
10
20
30
sqrt(DoF)

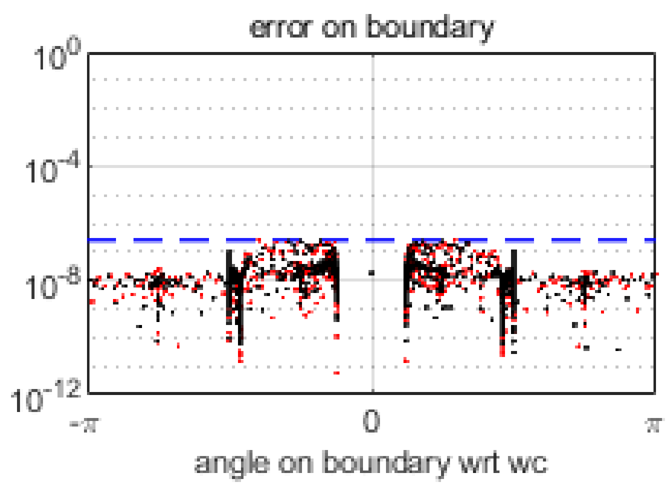


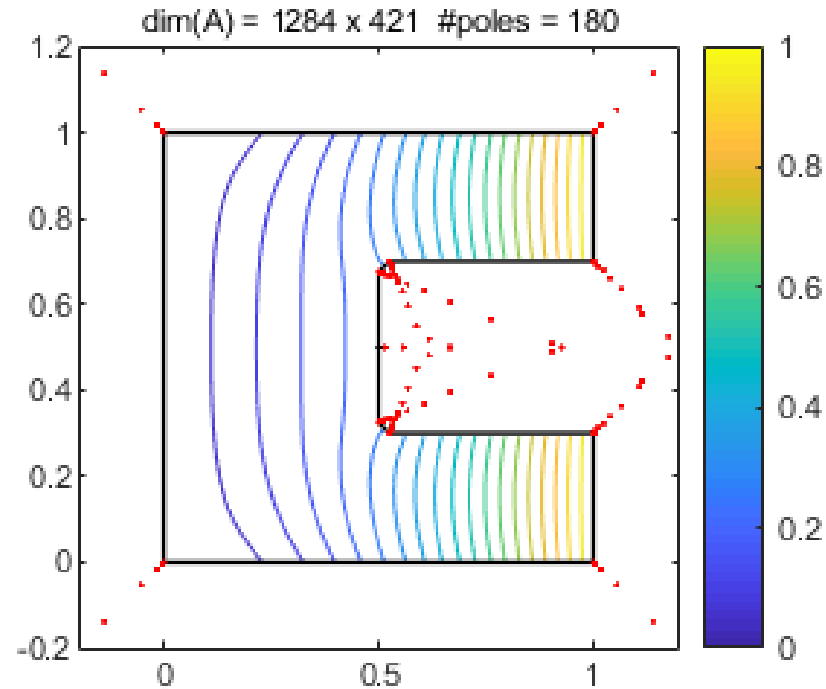
error on boundary
10^0
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

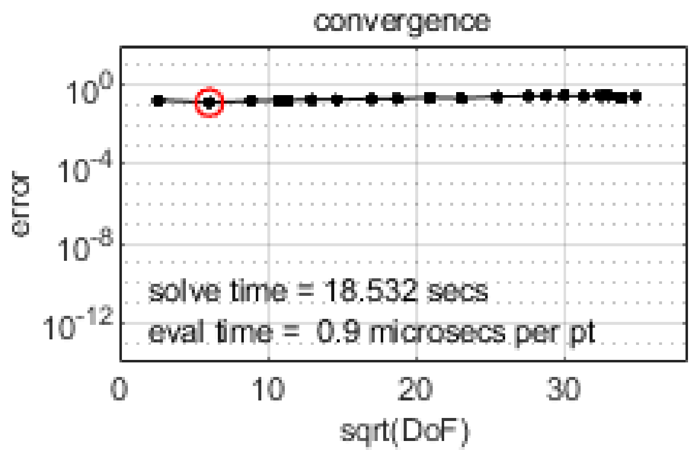


dim(A) = 3216 x 1089 #poles = 454
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.8
0.6
0.4
0.2
0

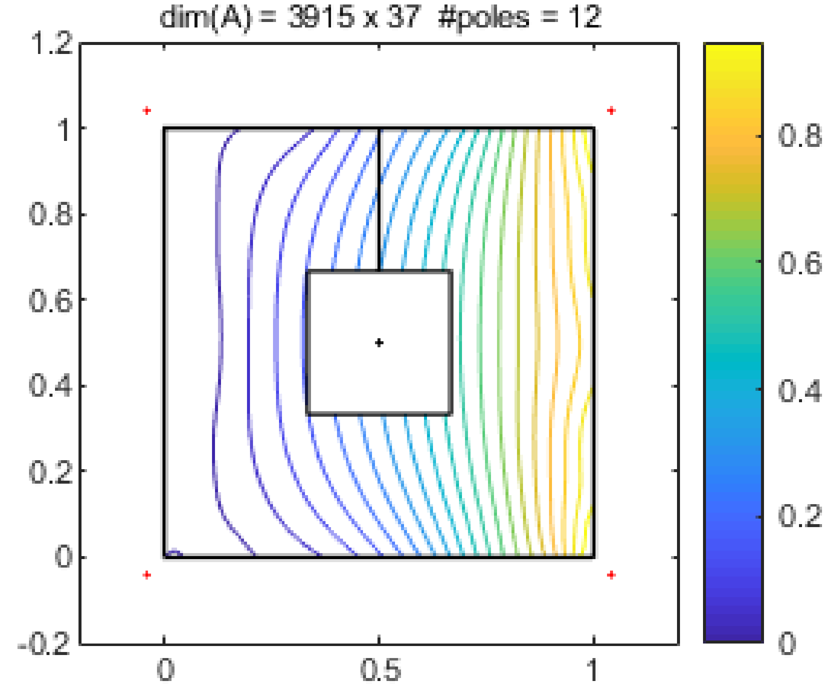

convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 0.609 secs
eval time = 35.7 microsecs per pt
0
5
10
15
20
sqrt(DoF)

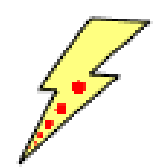

error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

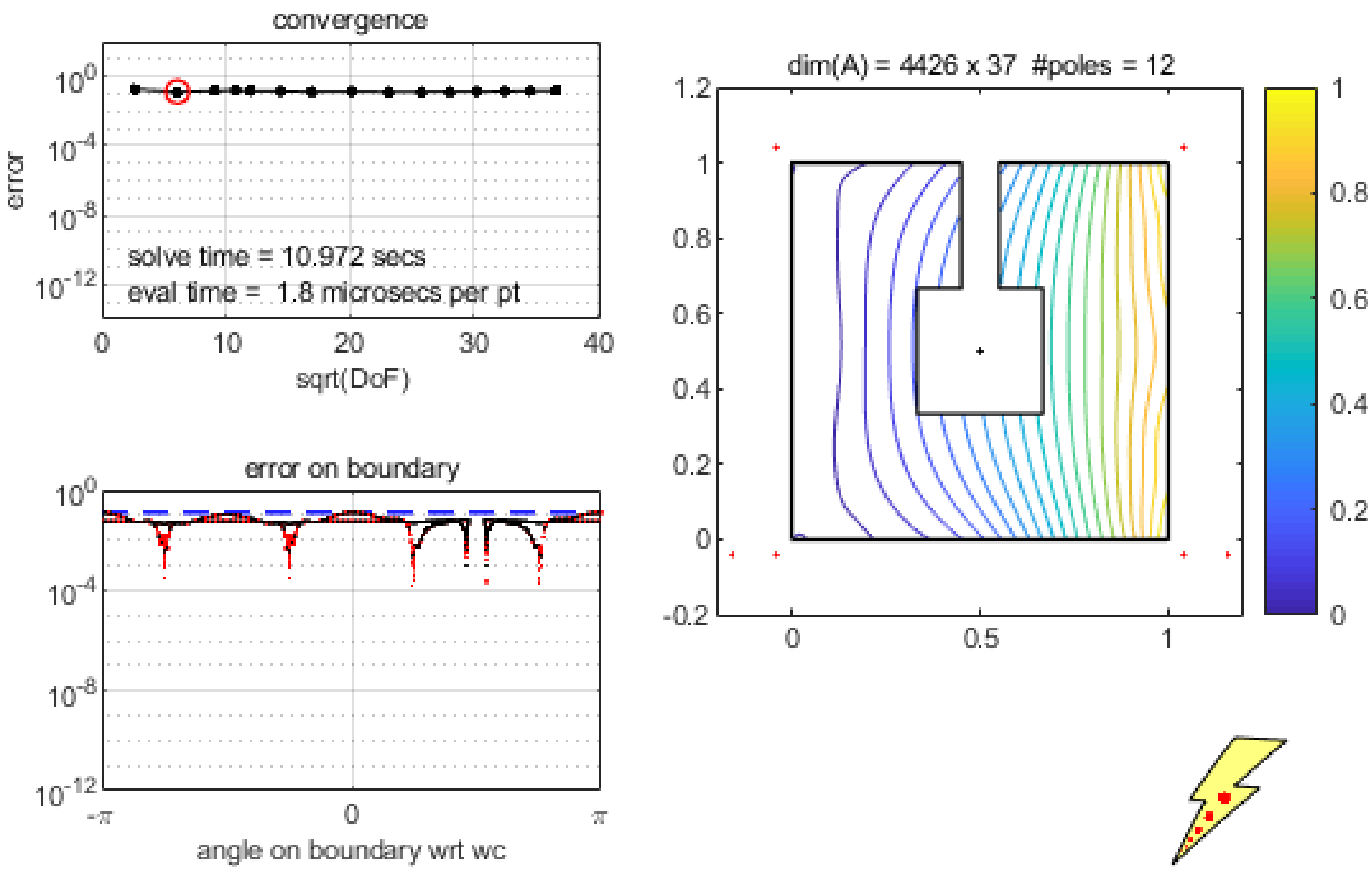

dim(A) = 1284 x 421 #poles = 180
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
0.8
0.6
0.4
0.2

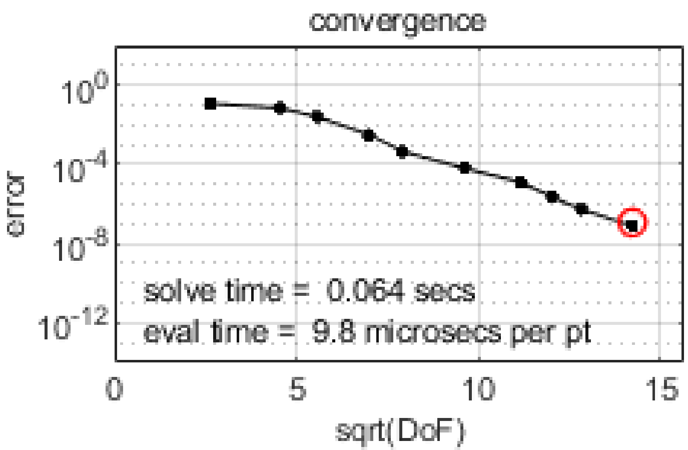

convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 18.532 secs
eval time = 0.9 microsecs per pt
0
10
20
30
sqrt(DoF)

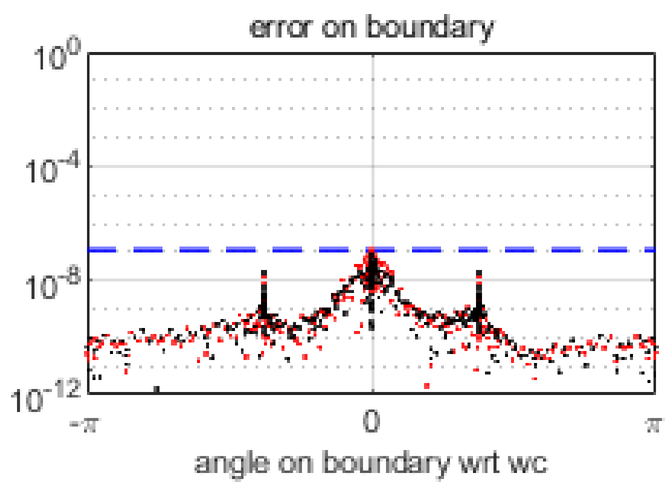

error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

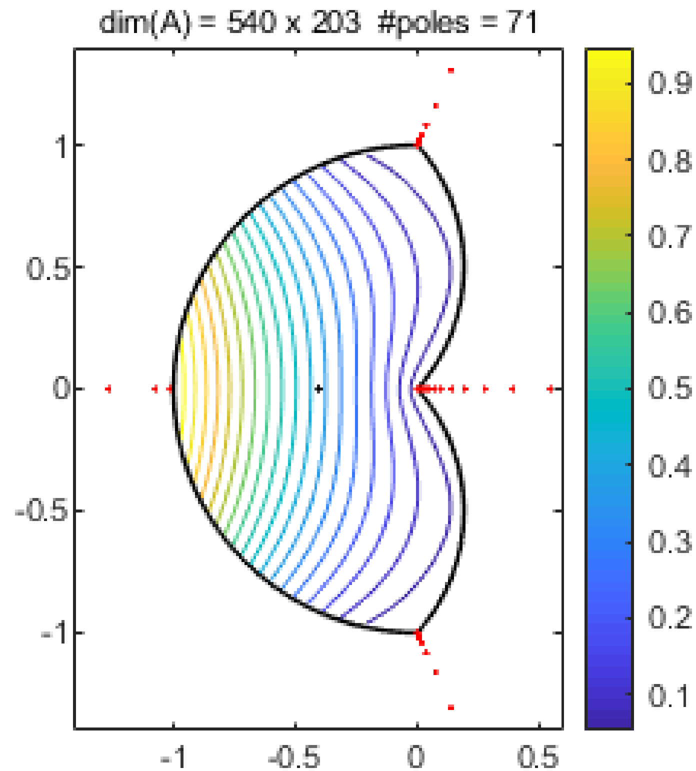

dim(A) = 3915 x 37 #poles = 12
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
0.8
0.6
0.4
0.2
0

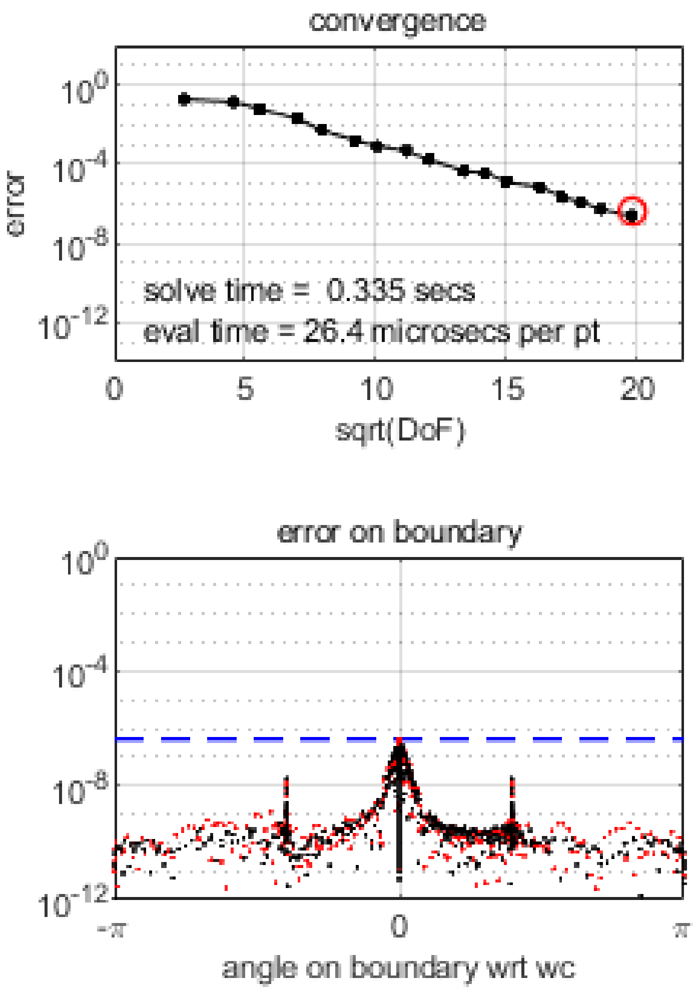



convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 10.972 secs
eval time = 1.8 microsecs per pt
0
10
20
30
40
sqrt(DoF)
error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc
dim(A) = 4426 x 37 #poles = 12
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1
0.8
0.6
0.4
0.2
0

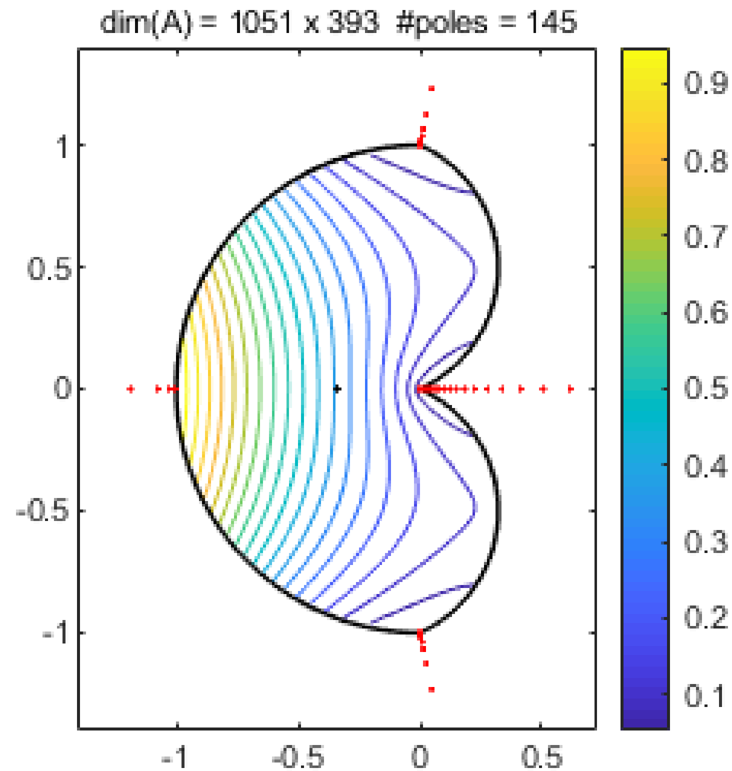

convergence
error
10^0
10^{-4}
10^{-8}
10^{-12}
solve time = 0.064 secs
eval time = 9.8 microsecs per pt
0
5
10
15
sqrt(DoF)


error on boundary
10^0
10^{-4}
10^{-8}
10^{-12}
$-\pi$
0
π
angle on boundary wrt wc

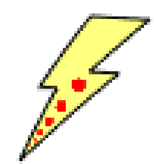

dim(A) = 540 x 203 #poles = 71
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

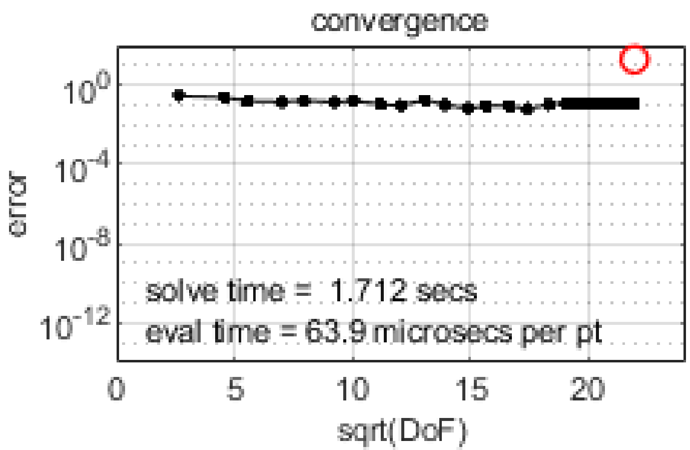


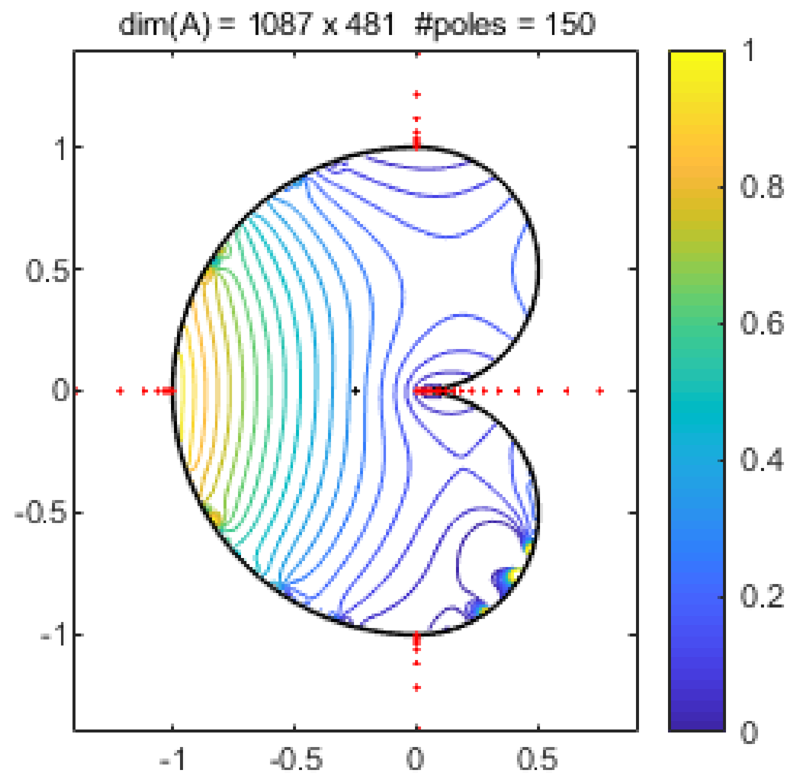
convergence
error
solve time = 0.335 secs
eval time = 26.4 microsecs per pt
sqrt(DoF)
error on boundary
angle on boundary wrt wc

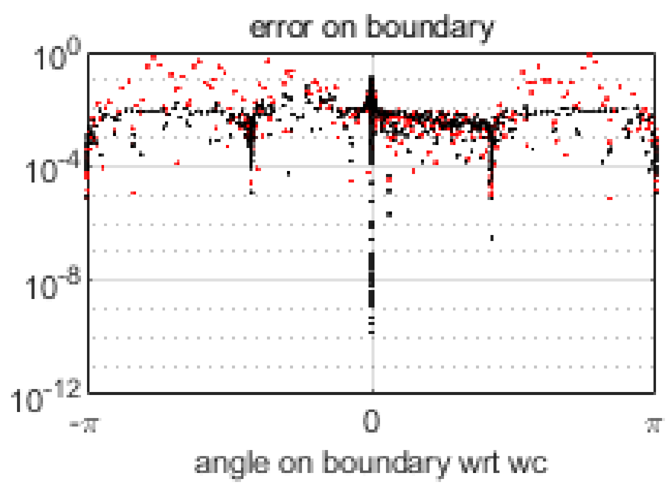


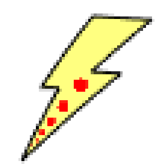
dim(A) = 1051 x 393 #poles = 145

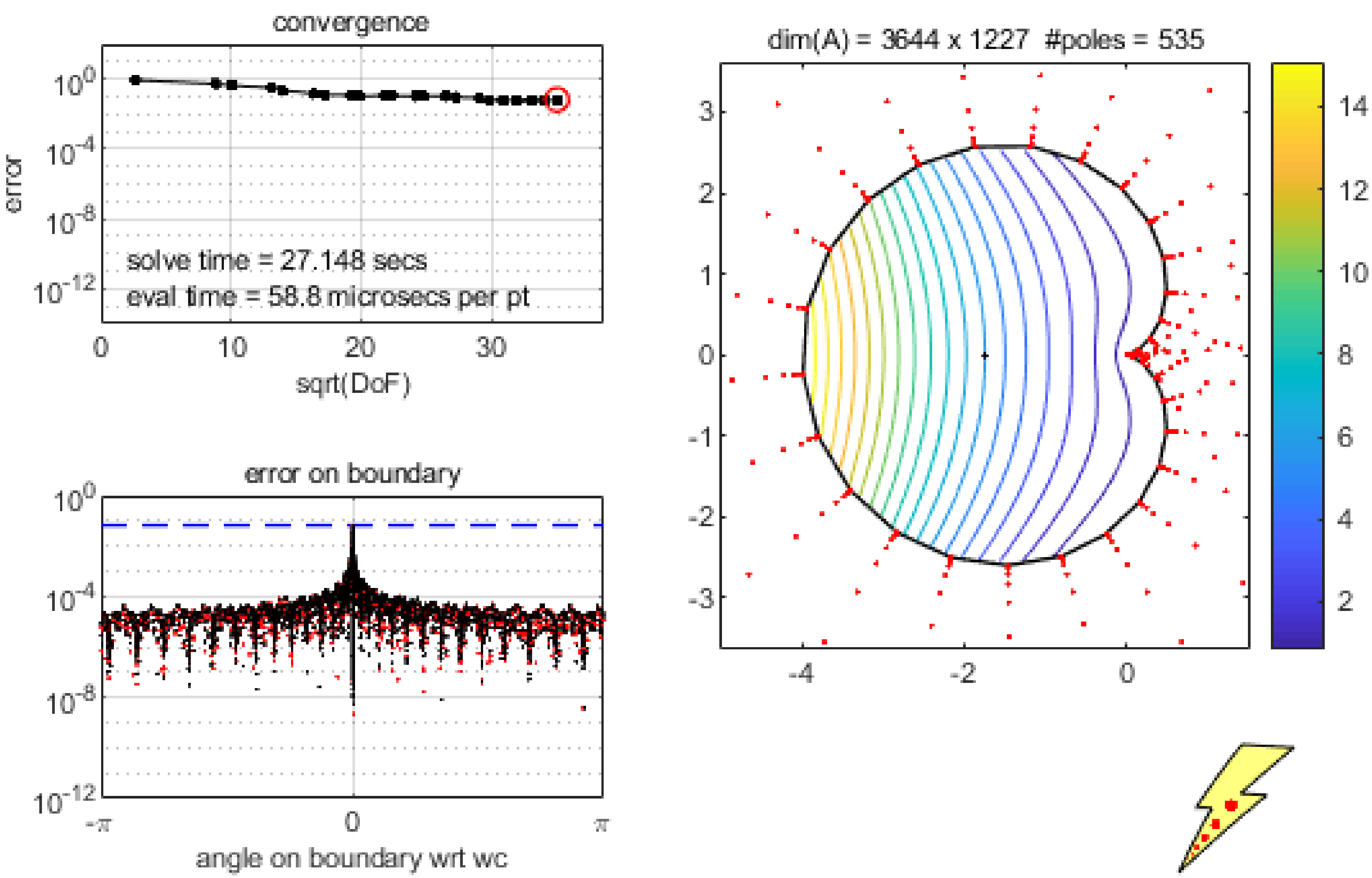



convergence
error
10^{0}
10^{-4}
10^{-8}
10^{-12}
solve time = 1.712 secs
eval time = 63.9 microsecs per pt
0 5 10 15 20
sqrt(DoF)


dim(A) = 1087 x 481 #poles = 150
1
0.5
0
-0.5
-1
-1 -0.5 0 0.5
1 0.8 0.6 0.4 0.2 0


error on boundary
10^{0}
10^{-4}
10^{-8}
10^{-12}
$-\pi$ 0 π
angle on boundary wrt wc





convergence
error
solve time = 27.148 secs
eval time = 58.8 microsecs per pt
sqrt(DoF)
error on boundary
angle on boundary wrt wc
dim(A) = 3644 x 1227 #poles = 535

What else...

Variants:

- Discontinuous boundary conditions: pretty much no problem, except supremum norm convergence to zero. Instead, use a supremum norm weighted by distance to the nearest corner.
- Multiply connected domains: very much a problem.
- Poisson equation $\Delta u = f$ with boundary conditions: find $\Delta v = f$ with arbitrary bd. conds. first, then do $w = u - v$.
- Faster than root exponential convergence: not possible.
- Domain size, matrix size: Runge part of r works less optimally far away from z_*; m corners means operation count $O(m^3|\log(\varepsilon)|^6)$.

What else...

- Authors say Finite Element Methods can't match Lightning Laplace's simplicity and performance.
- Boundary Integral Equations are good when applicable, and is "the most powerful tool currently available."
- Proofs for various more general domains
- Expanding code functionality and usability

The End