A fast direct solver for boundary integral equations in two dimensions

P.G. Martinsson and V. Rokhlin

Yale University

Thanks to: M. Tygert

Mini-review of fast algorithms for boundary integral equations

We consider the integral equation

$$(1) \qquad u(x) + \int_\Gamma K(x,y)u(y)\,ds(y) = f(x), \qquad x \in \Gamma.$$

Upon discretization, equation (1) turns into a discrete equation

$$(2) \qquad (I + A)u = f$$

where A is a (typically dense) $n \times n$ matrix.

- **FMM** — Can multiply A by a vector in $O(n)$ time.
- **Iterative solver** — Solves (2) using $\sqrt{\kappa}$ matrix-vector multiplies, where κ is the condition number of $(I + A)$.
- **Total complexity** — $O(\sqrt{\kappa}\,n)$.

Some definitions:

A **fast** method solves a problem using $O(n \log^q n)$ arithmetic operations. $(q = 0, 1, 2)$.

A **direct** solver computes a representation for $(I + A)^{-1}$.

Direct solvers tend to outperform iterative solvers for problems involving:

- **ill-conditioned matrices**,
- multiple right-hand sides,
- up-dating a known solution to find the solution of another problem that is “close”,
- constructing the SVD and other factorizations of the matrix.

The method to be presented can be viewed as a generalization of previous work by E. Michielssen, A. Boag and W.C. Chew (1996).

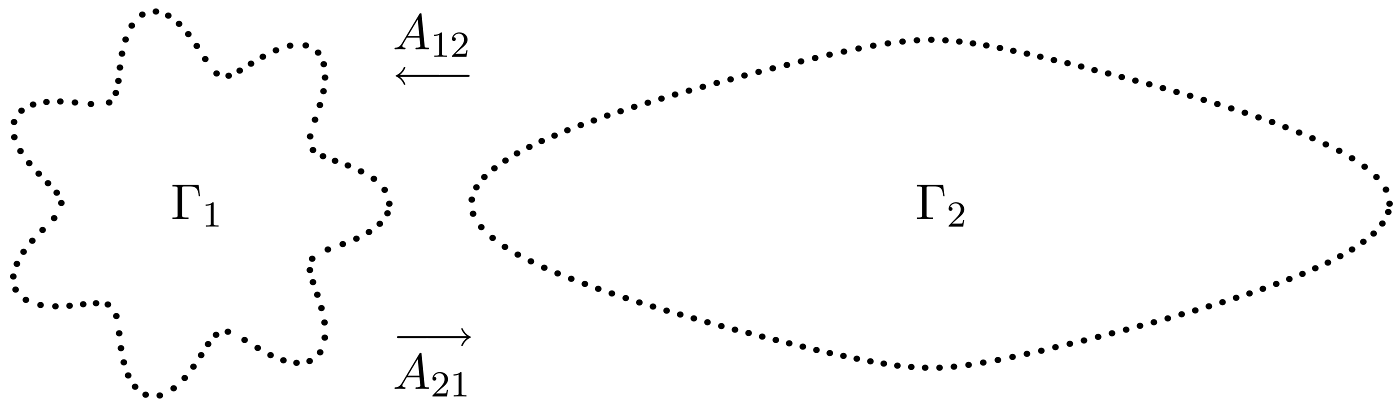
Related work:

- G. Beylkin and N. Coult (1998),
- $\mathcal{H}$-matrix methods (ca. 1998), W. Hackbusch, S. Börm, *et c.*
- Y. Chen (2002).

$$u(x) + \int_\Gamma K(x,y)u(y)\,ds(y) = f(x), \qquad x \in \Gamma. \tag{3}$$

We will present a fast direct solver for (3) in the following environment:

- The manifold Γ is one-dimensional.
  (We will also assume that $\Gamma \subset \mathbb{R}^2$ but this is not essential.)
- The kernel K is the single- or double-layer kernel associated with
  - Laplace's equation
  - Stokes' equation
  - Elasticity
  - Helmholtz (at low or moderate frequencies)
  - Maxwell (at low or moderate frequencies)
  - *etc*
- First kind equations can also be handled.

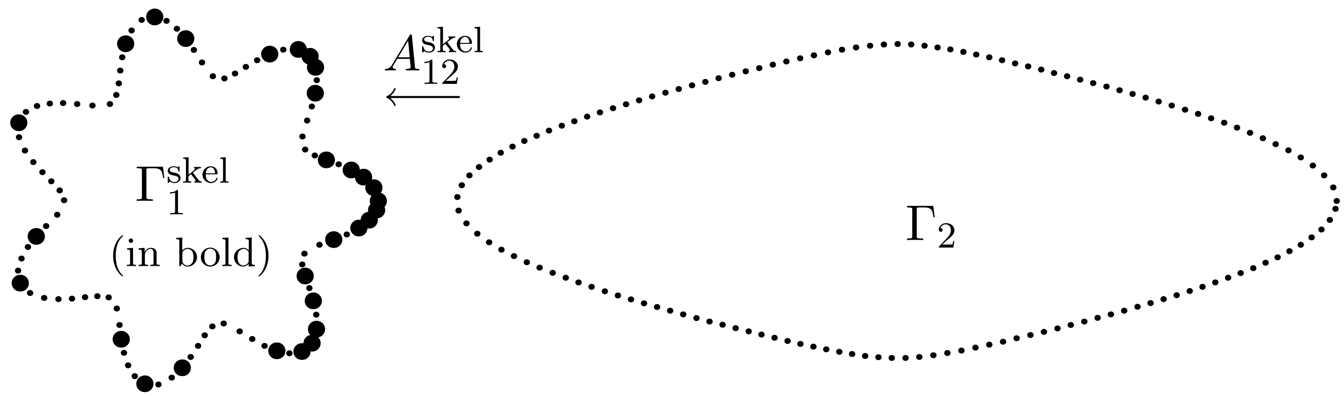


We consider the interaction between the two contours Γ_1 and Γ_2:

$$\text{Charges on } \Gamma_2 \xrightarrow{A_{12}} \text{Pot. on } \Gamma_1 \xrightarrow{A_{11}^{-1}} \text{Charges on } \Gamma_1 \xrightarrow{A_{21}} \text{Pot. on } \Gamma_2$$

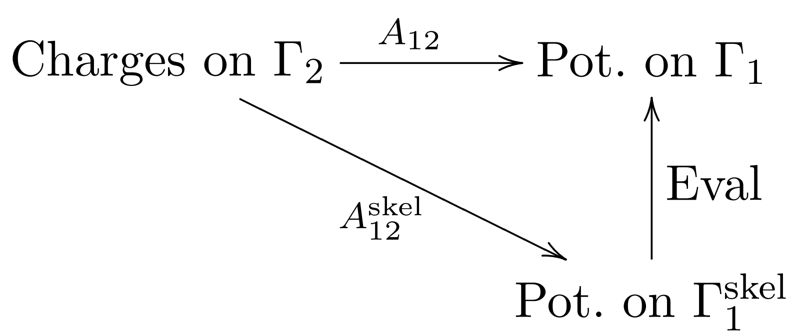
The maps A_{12} and A_{21} are typically rank-deficient (to finite precision).

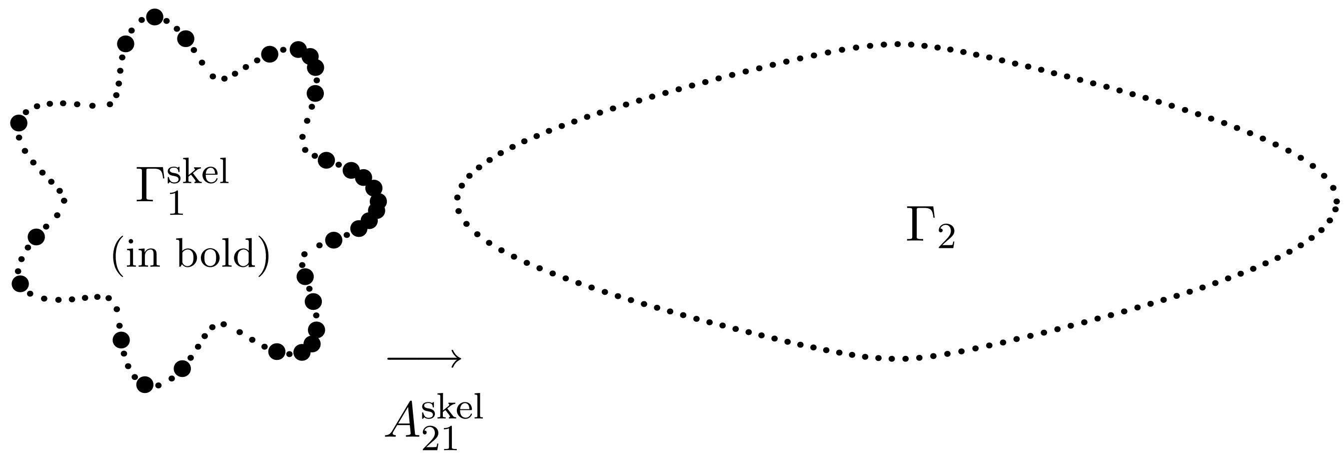
Example: Laplace double layer kernel: to accuracy 10^{-10}, the rank is 30.



Let k denote the rank of A_{12}.

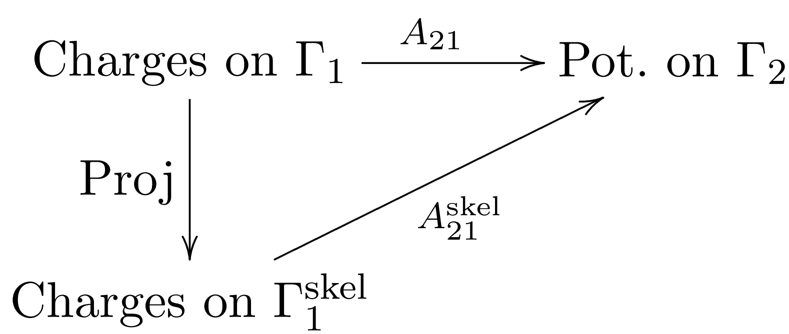
There exist a set $\Gamma_1^{\mathrm{skel}} \subset \Gamma_1$ with k points and a map Eval such that the following diagram commutes.

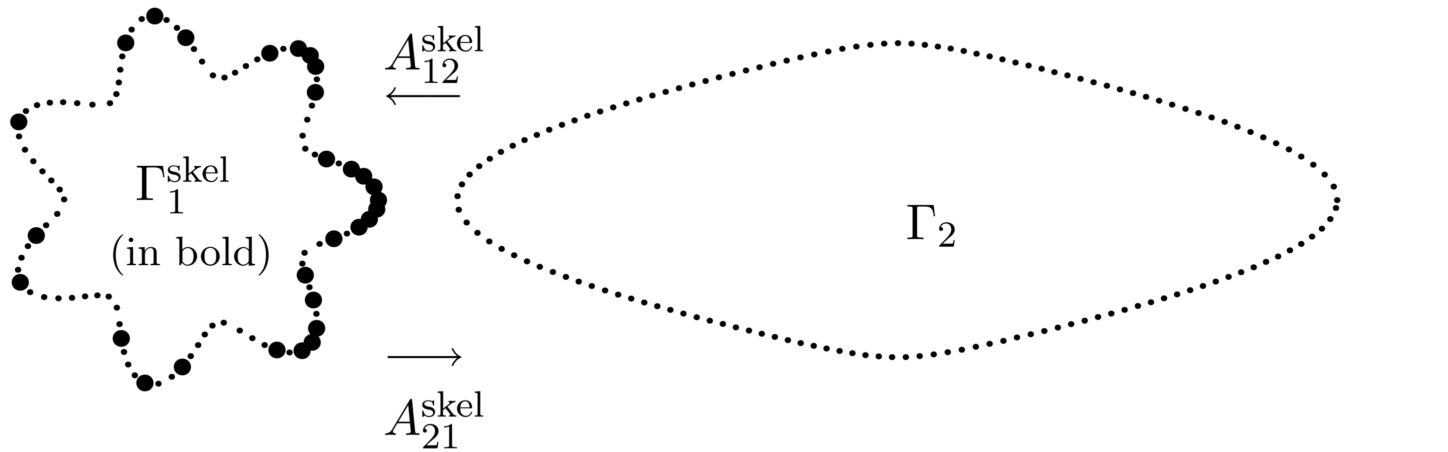




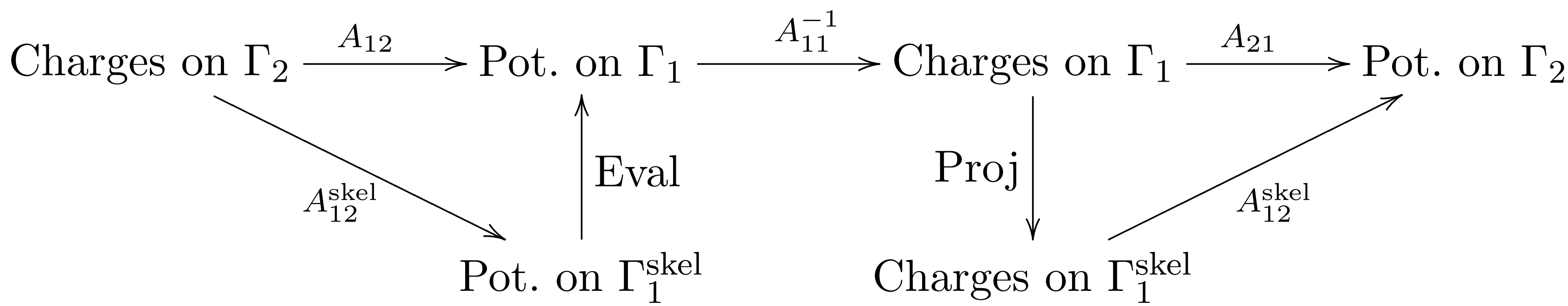
Analogously, we can compress A_{21}:

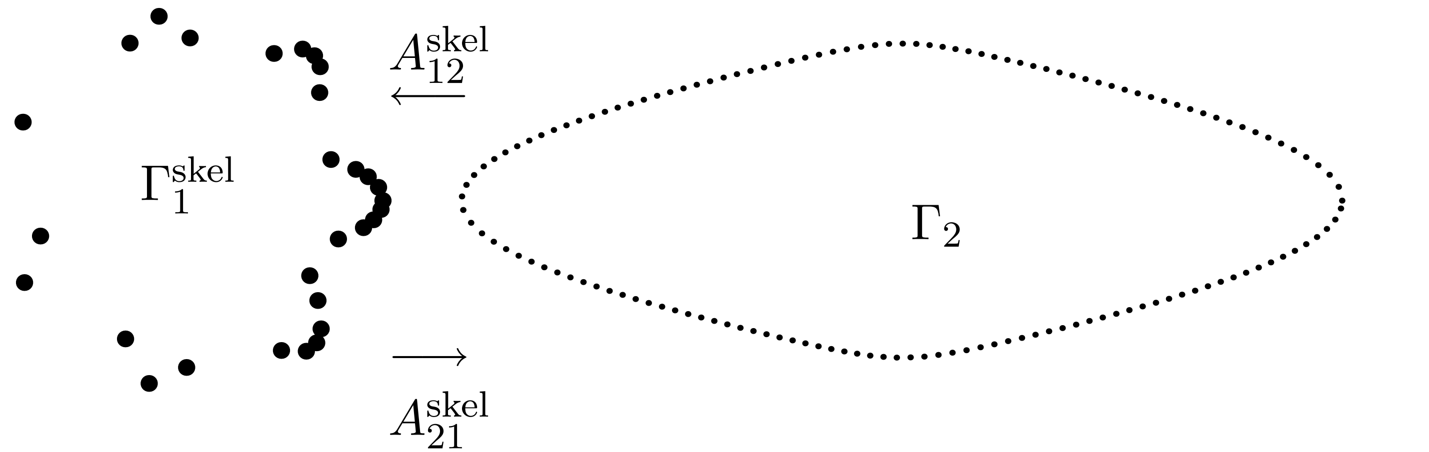
There exist a set $\Gamma_1^{\text{skel}} \subset \Gamma_1$ with k points and a map Proj such that the following diagram commutes.



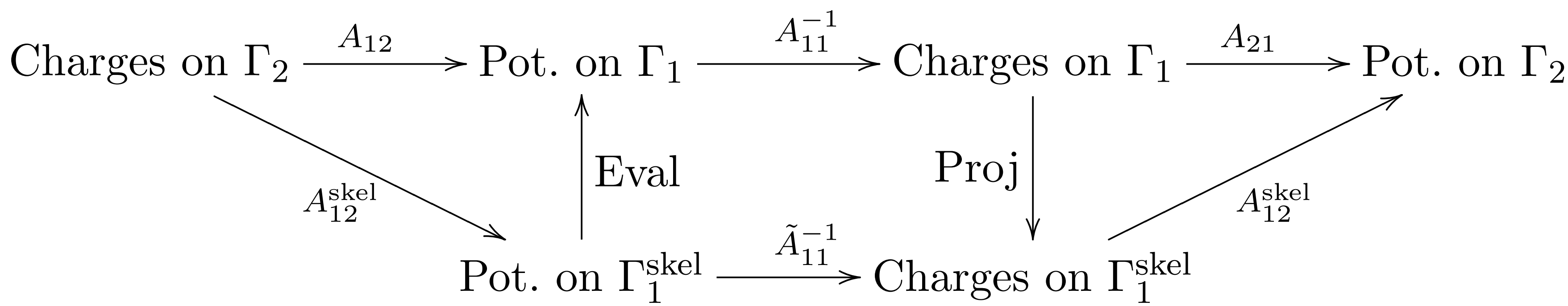


Now we can compress the entire interaction. . .



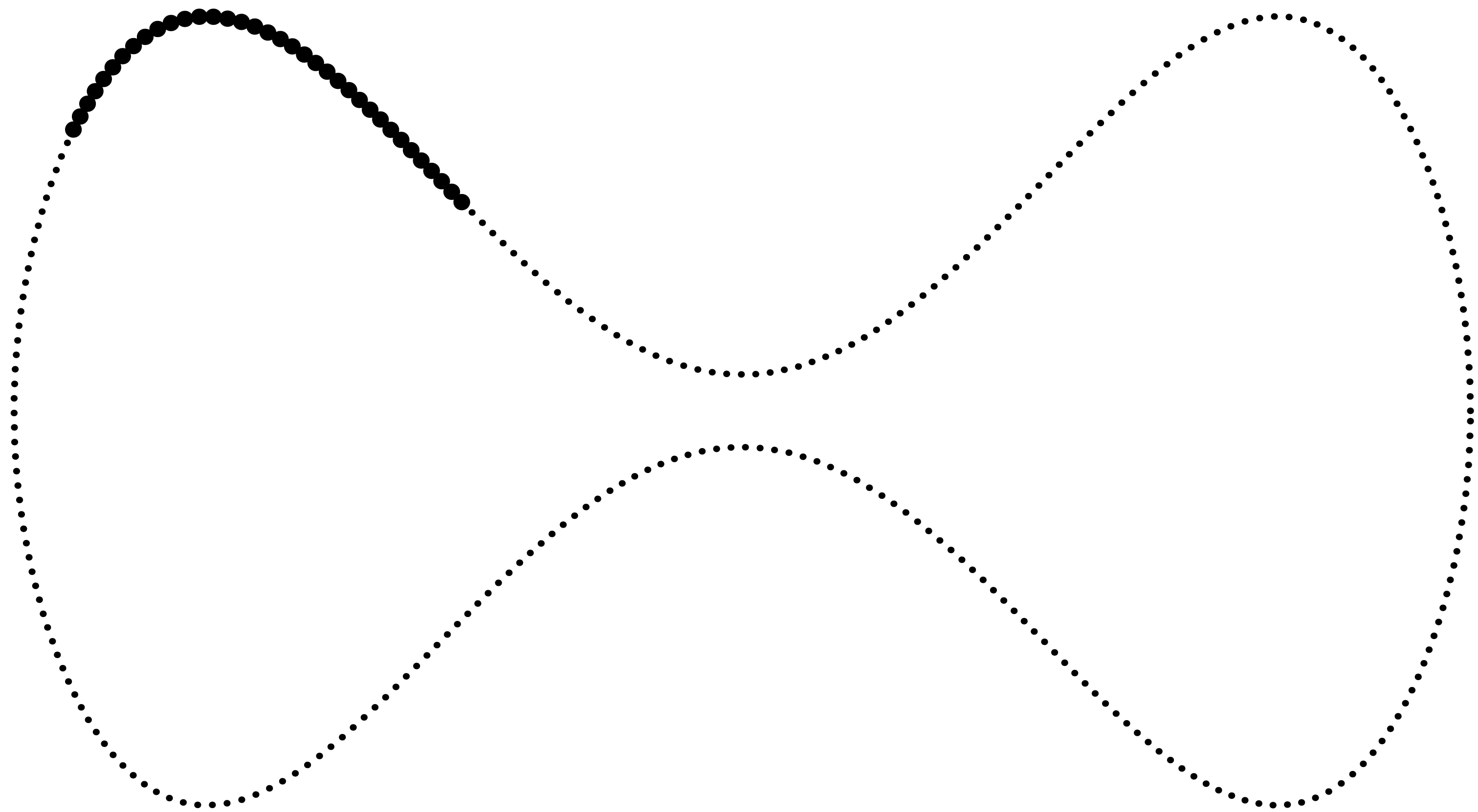


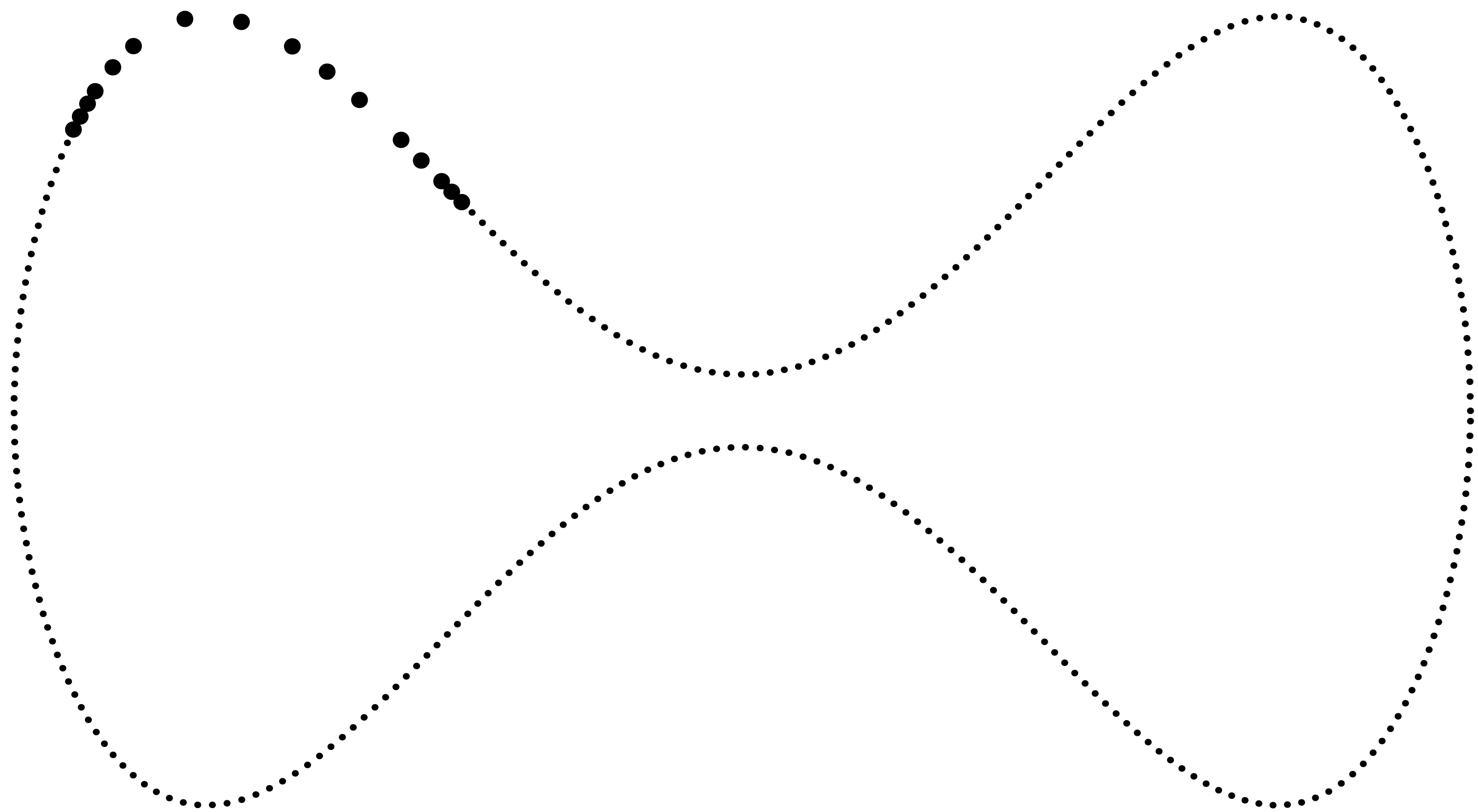
...and completely forget about the original points!

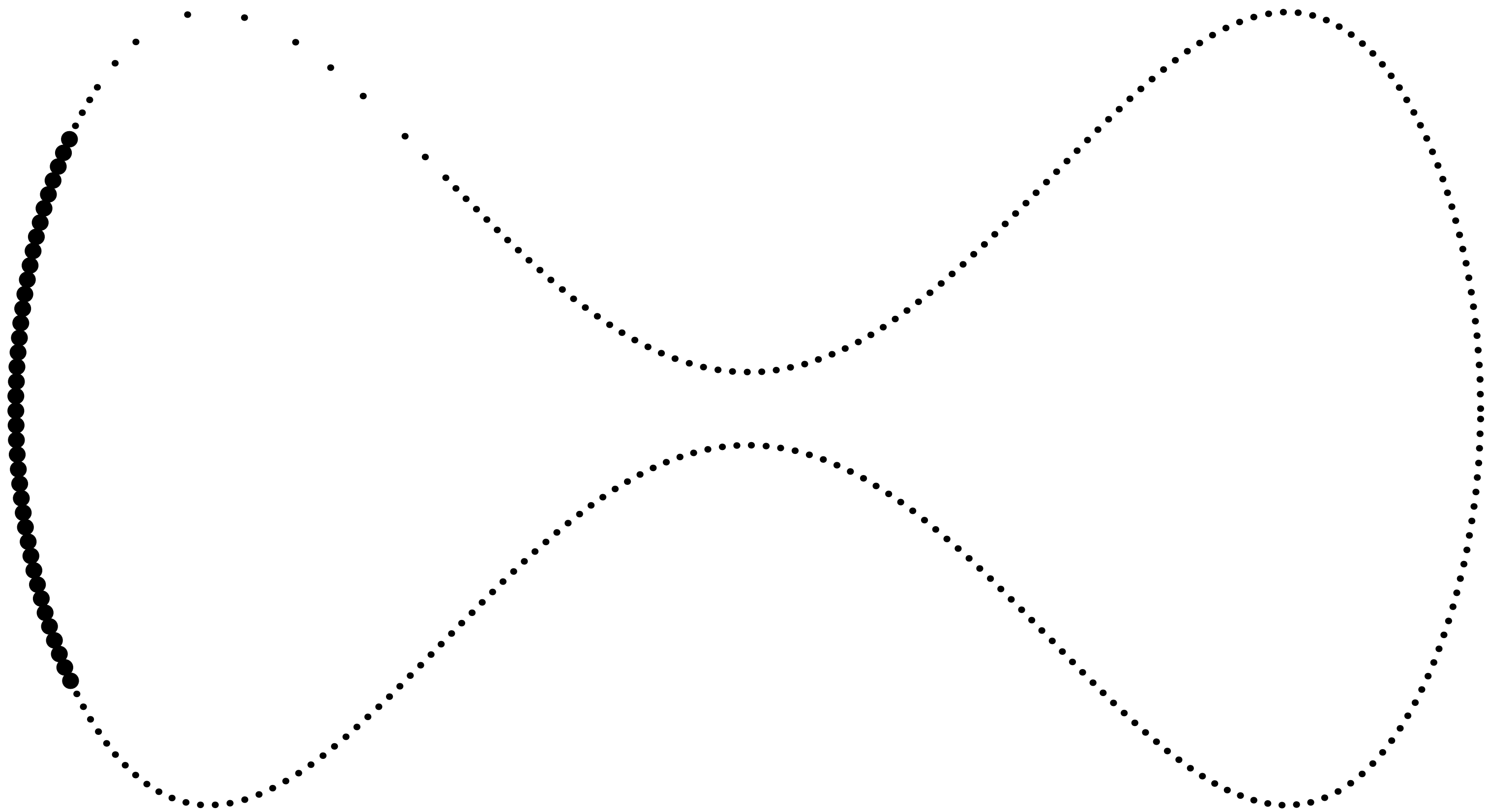


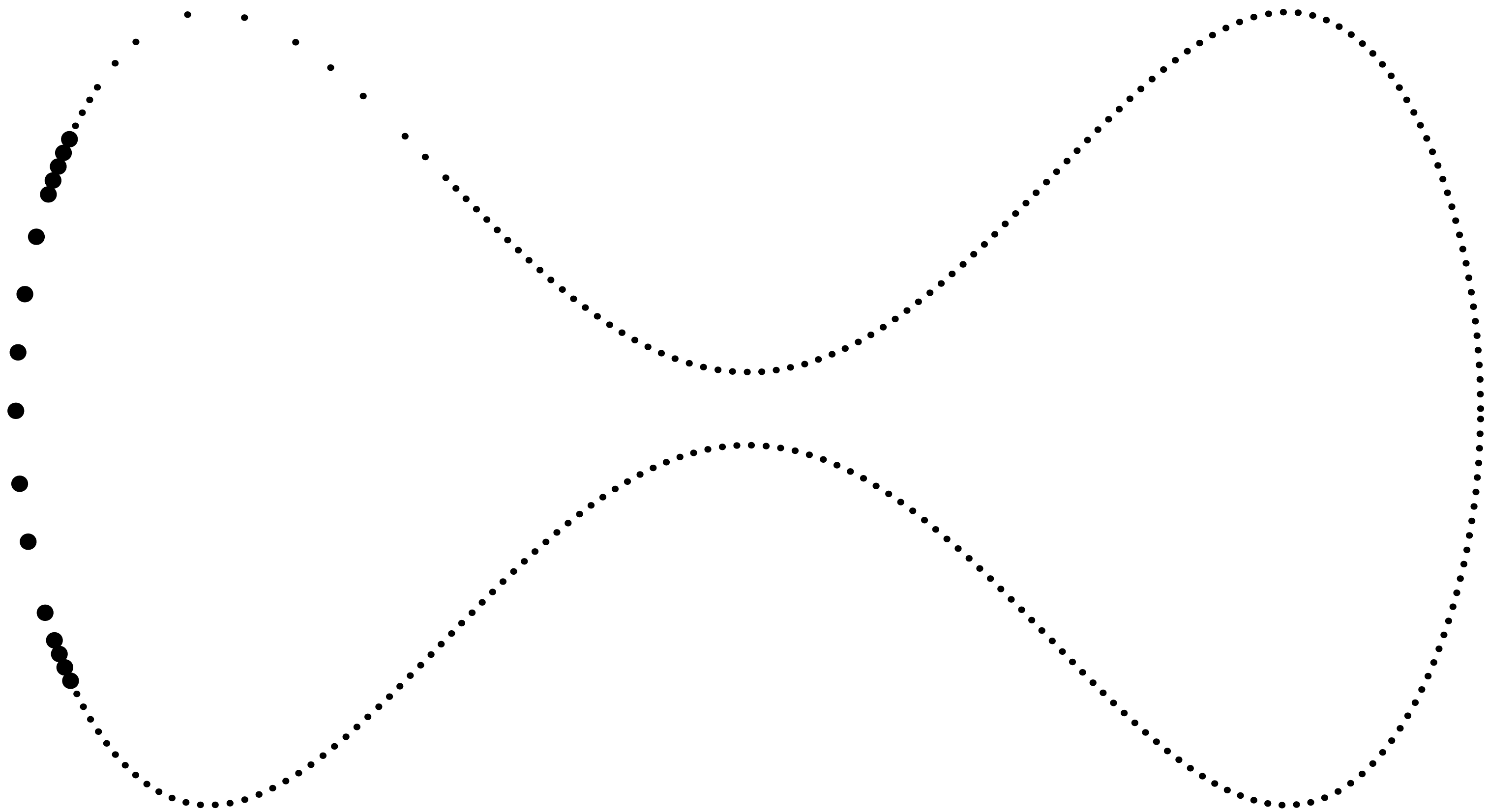
Notes:

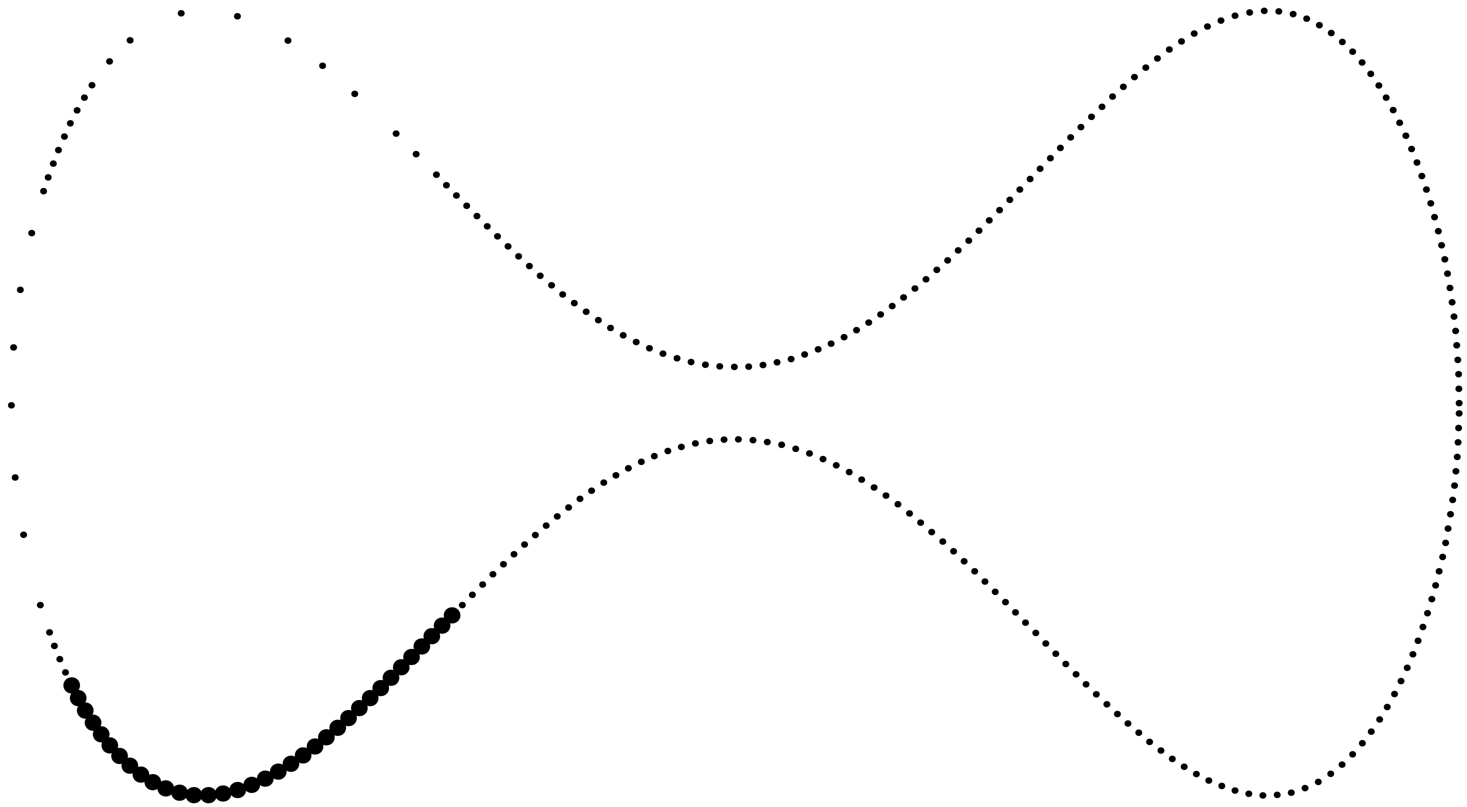
- A_{12}^{skel} consists of k of the rows of A_{12}.
- A_{21}^{skel} consists of k of the columns of A_{21}.
- The process consists of **pure linear algebra**.
- Proven to be accurate and well-conditioned.
  - Gu and Eisenstat (1996)
  - Cheng, Gimbutas, Martinsson, Rokhlin (2003)
  - Martinsson and Rokhlin (2003)

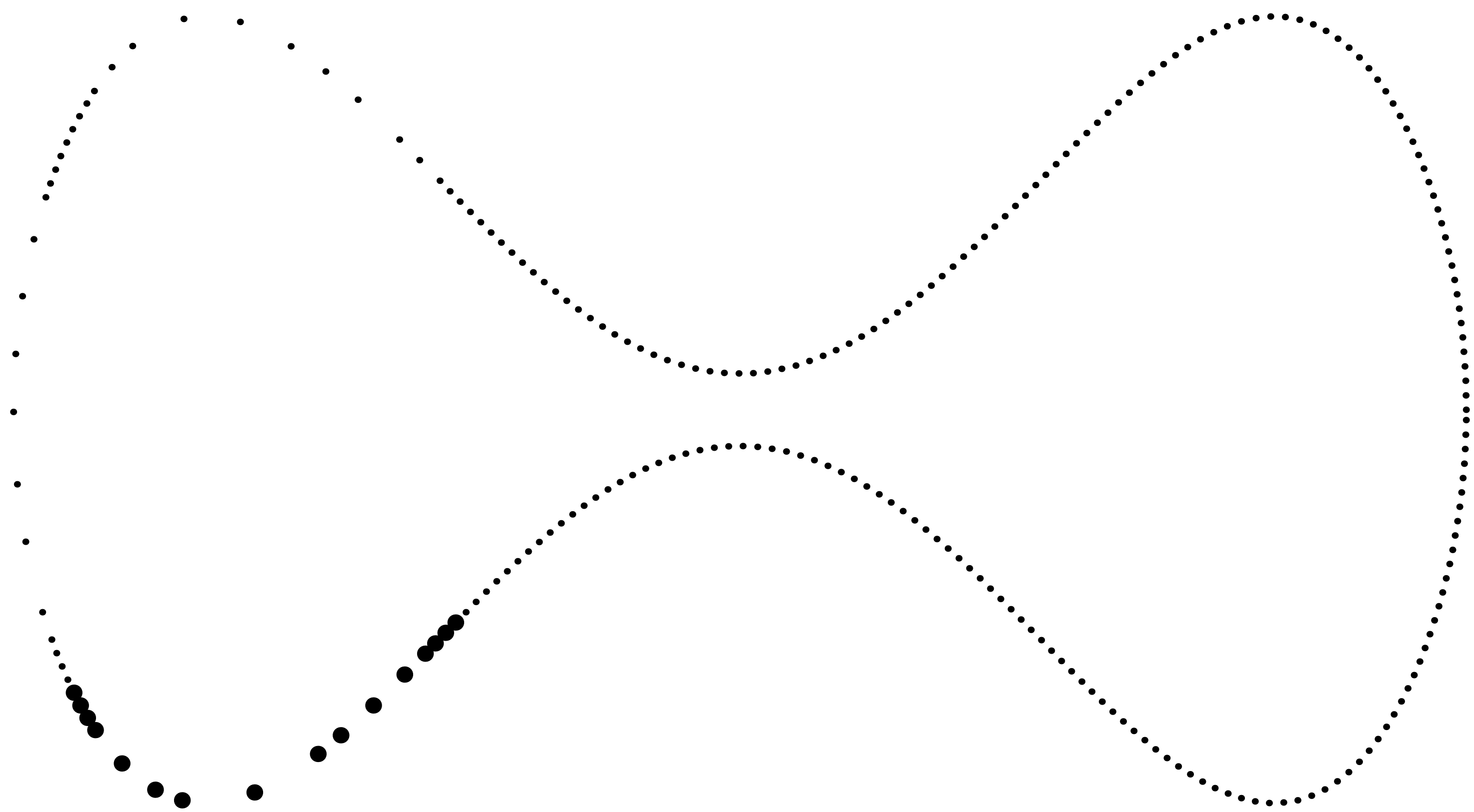


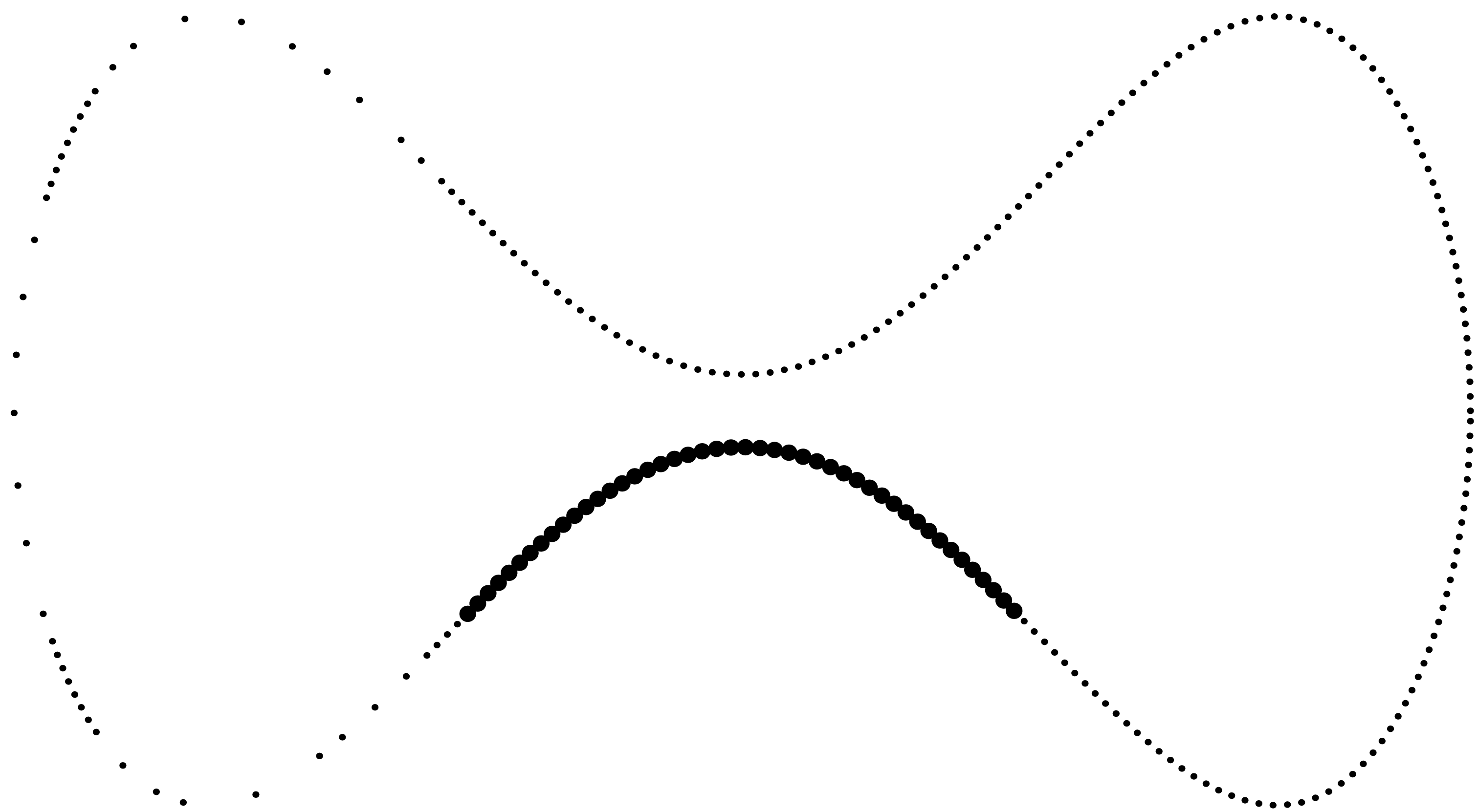


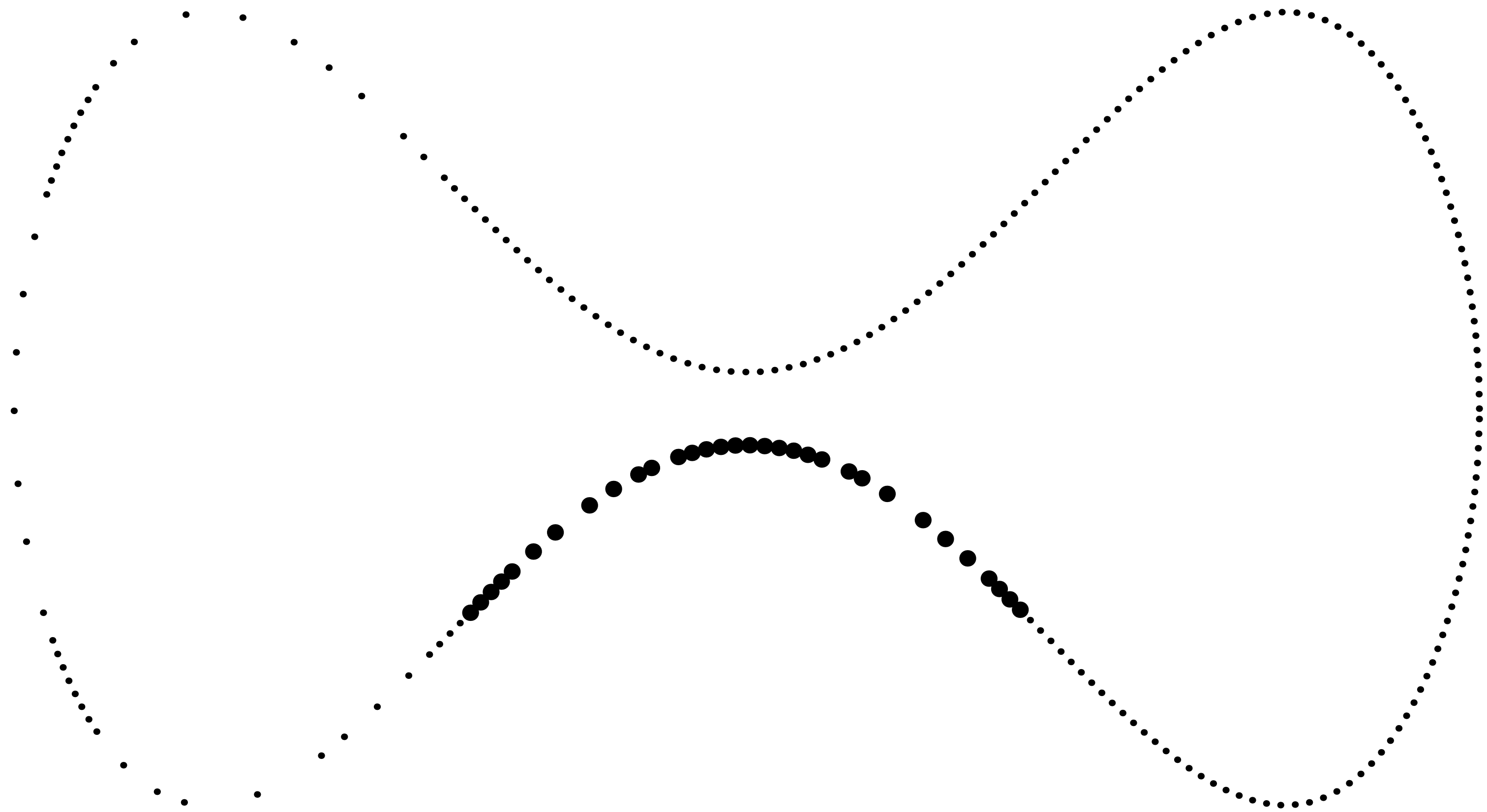


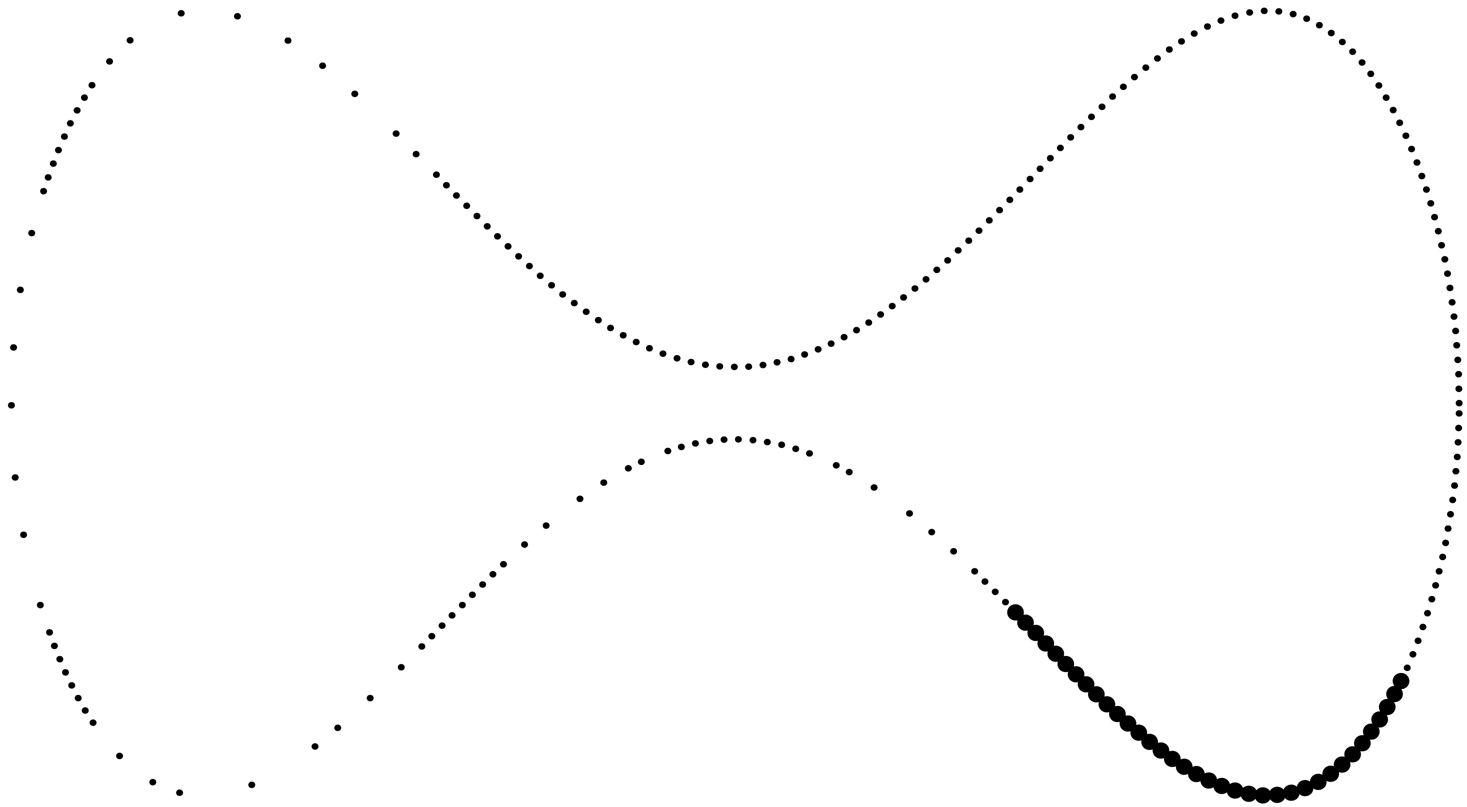


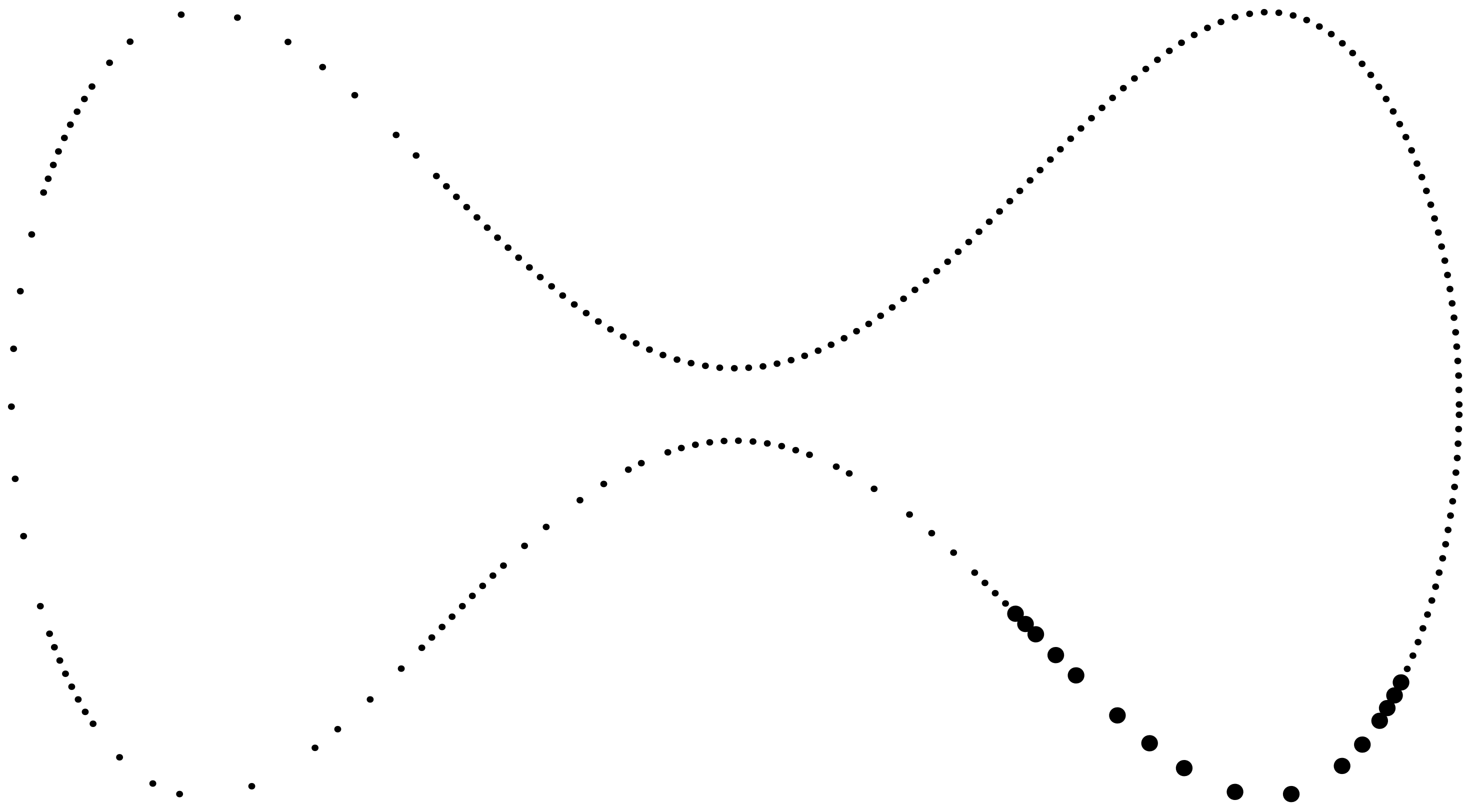


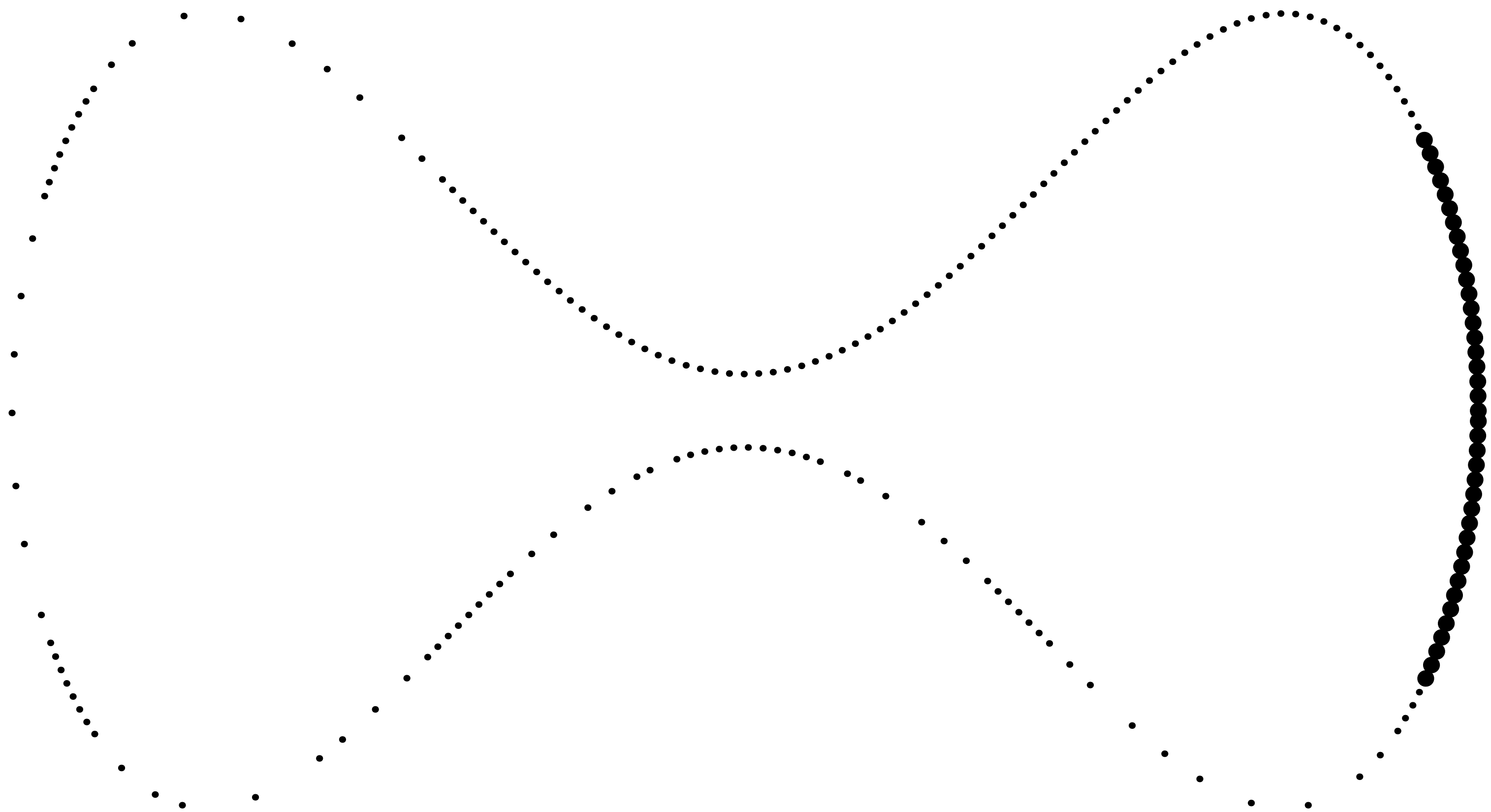


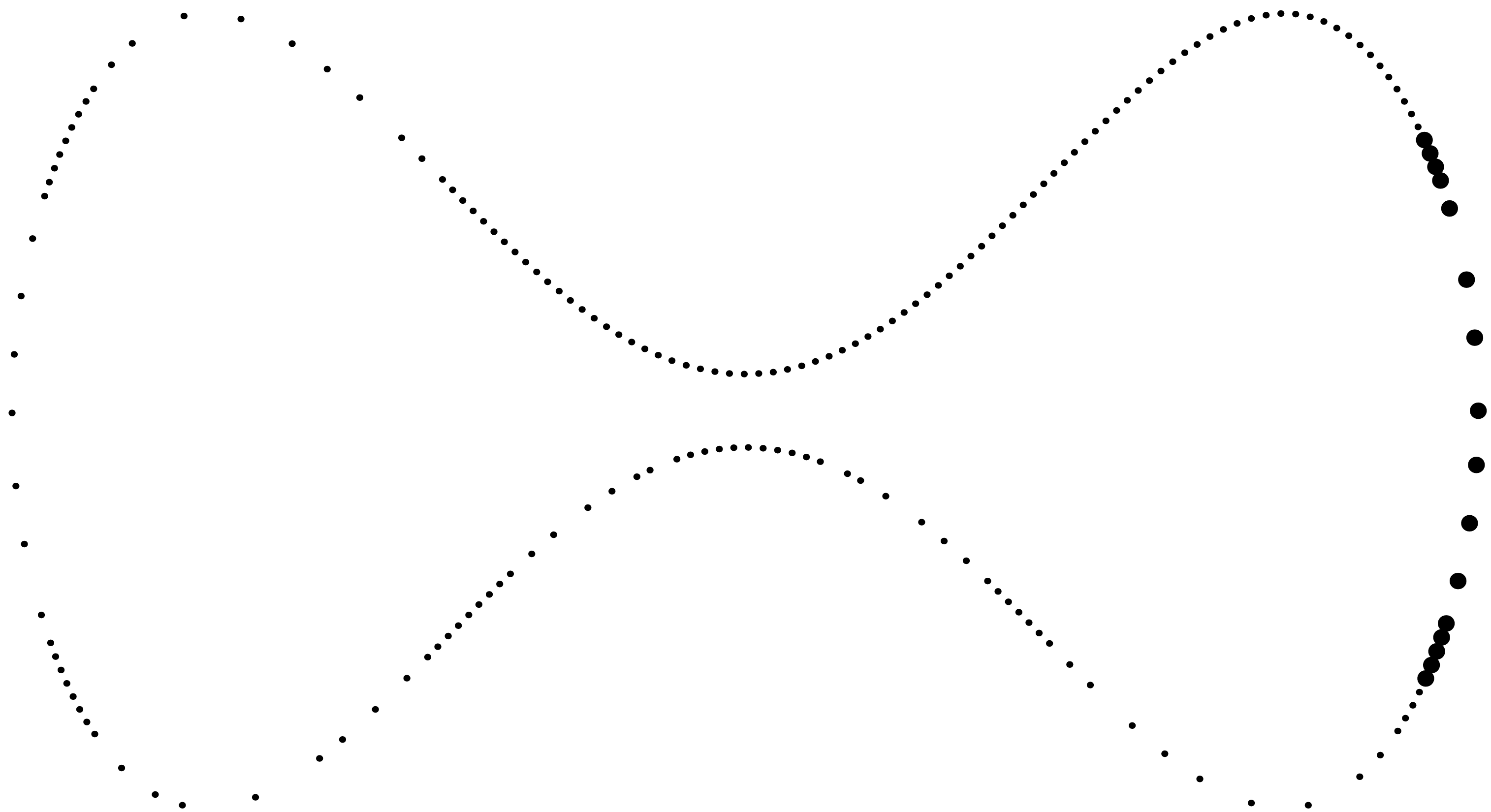


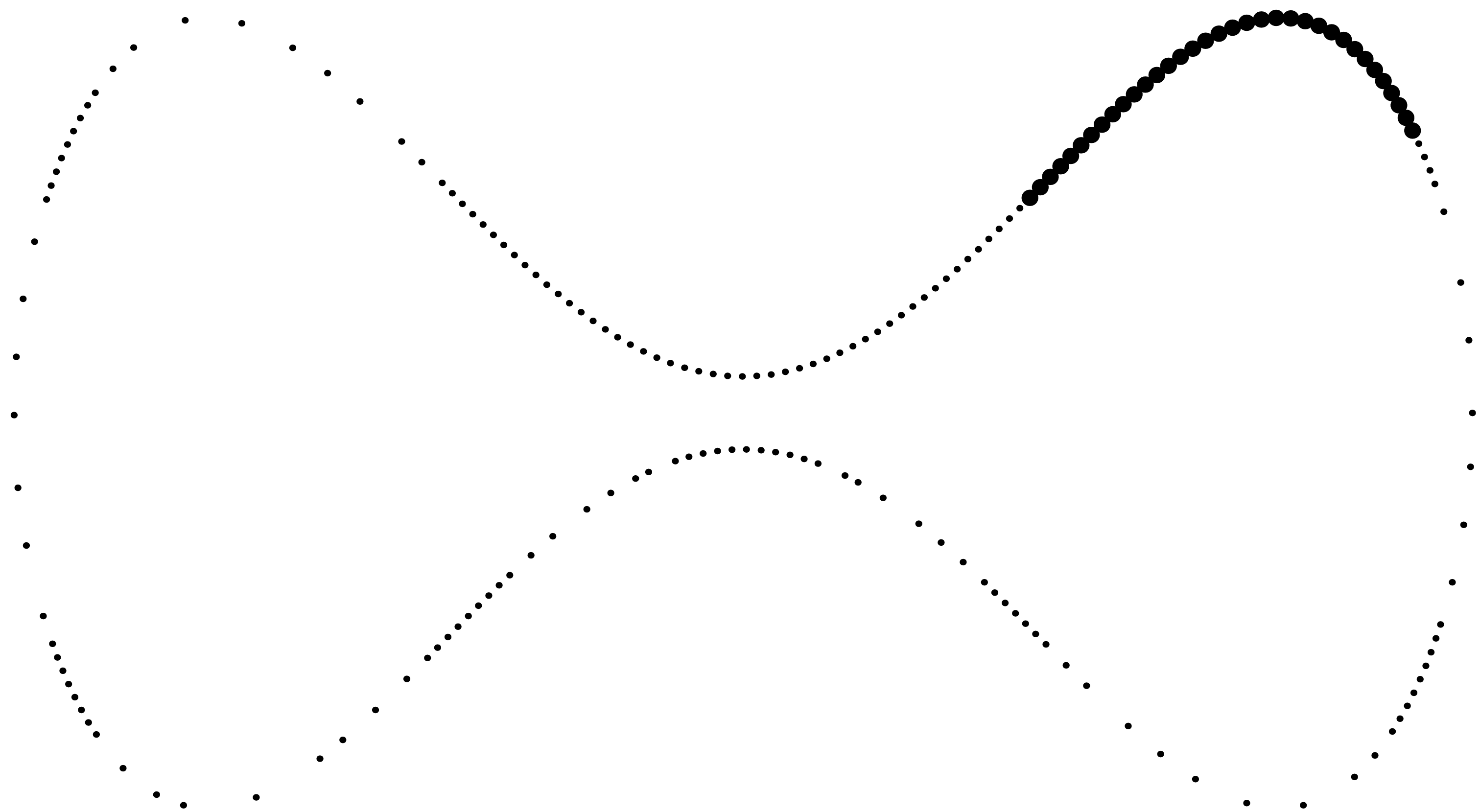


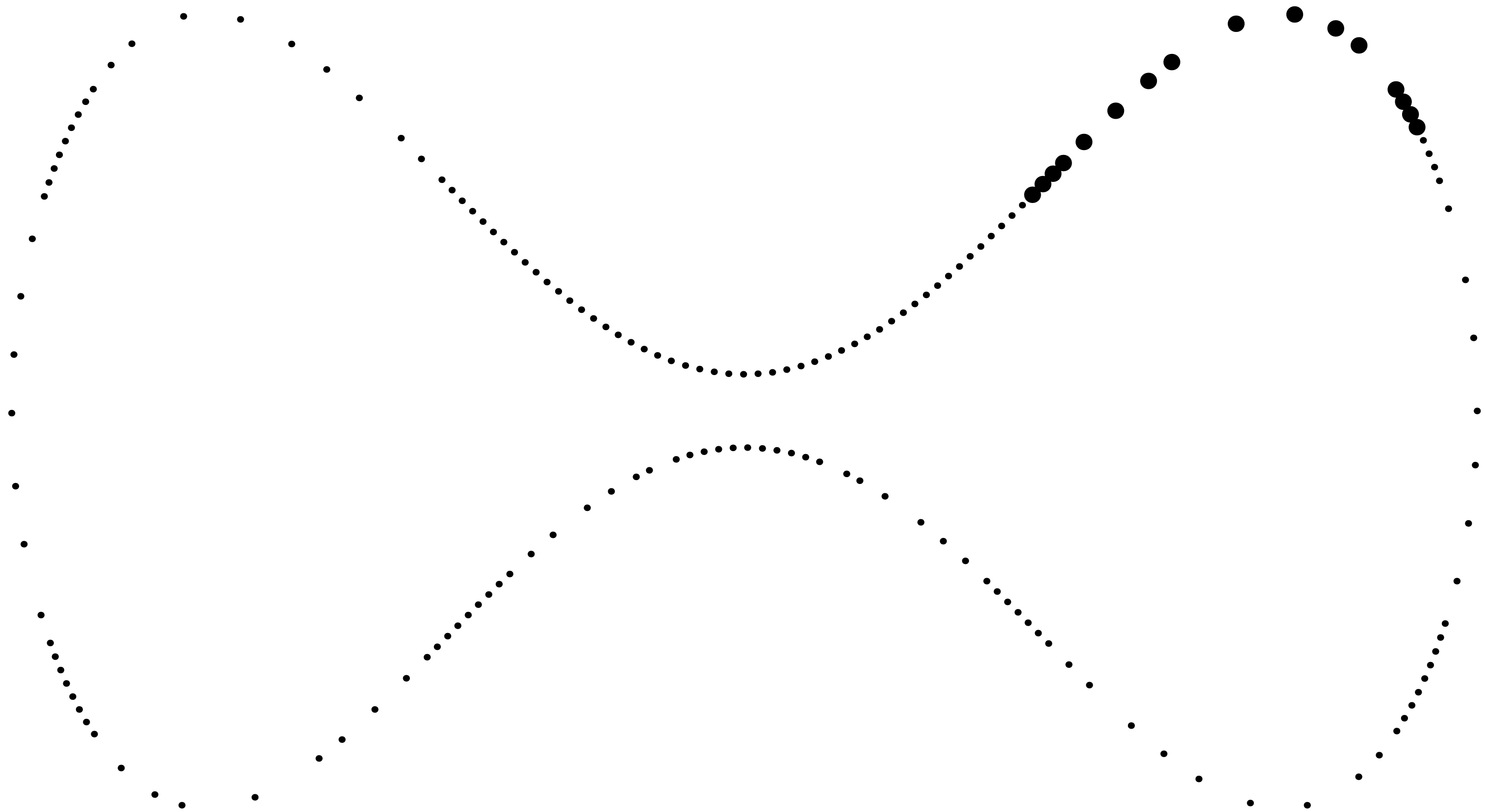


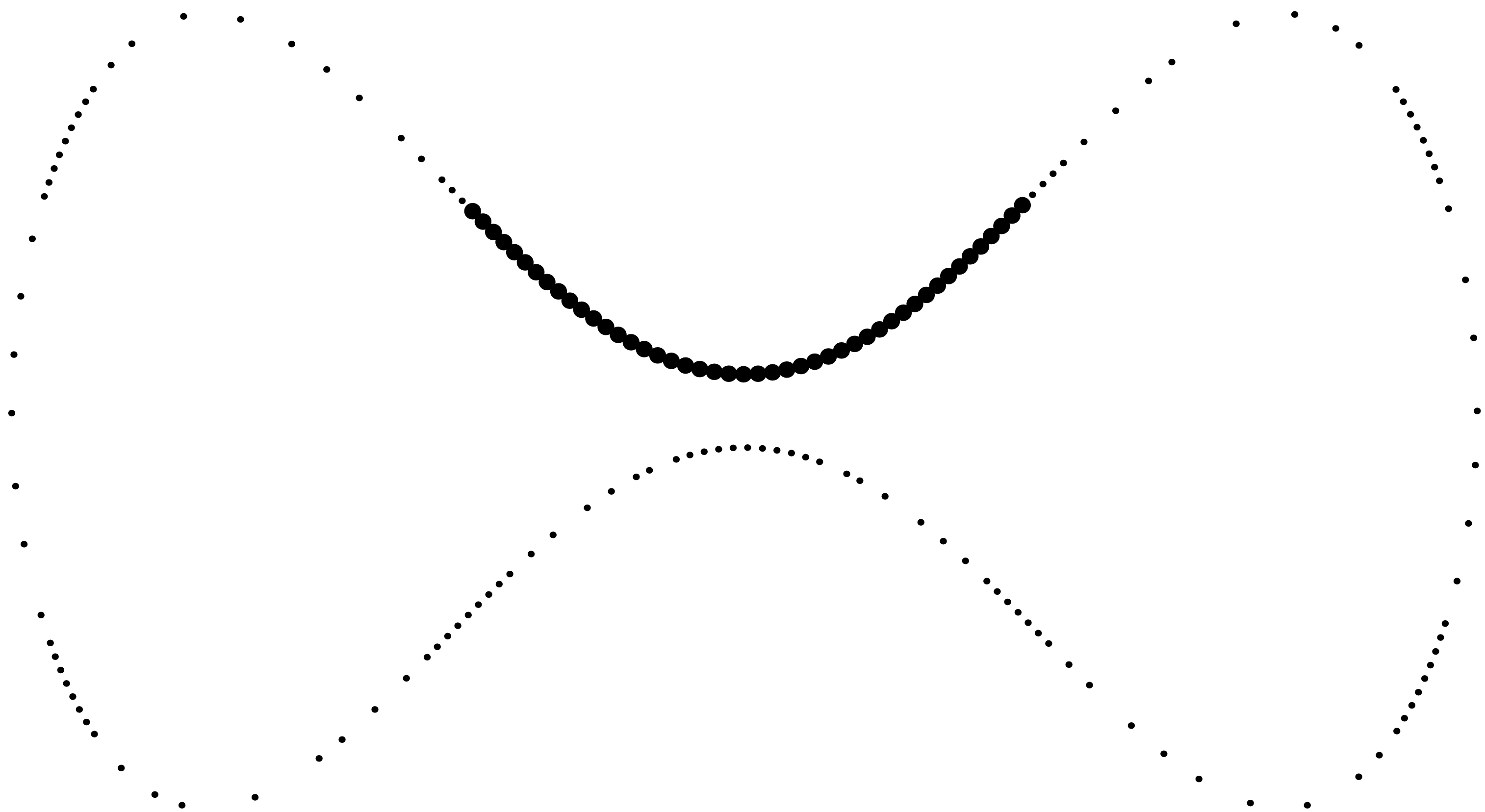


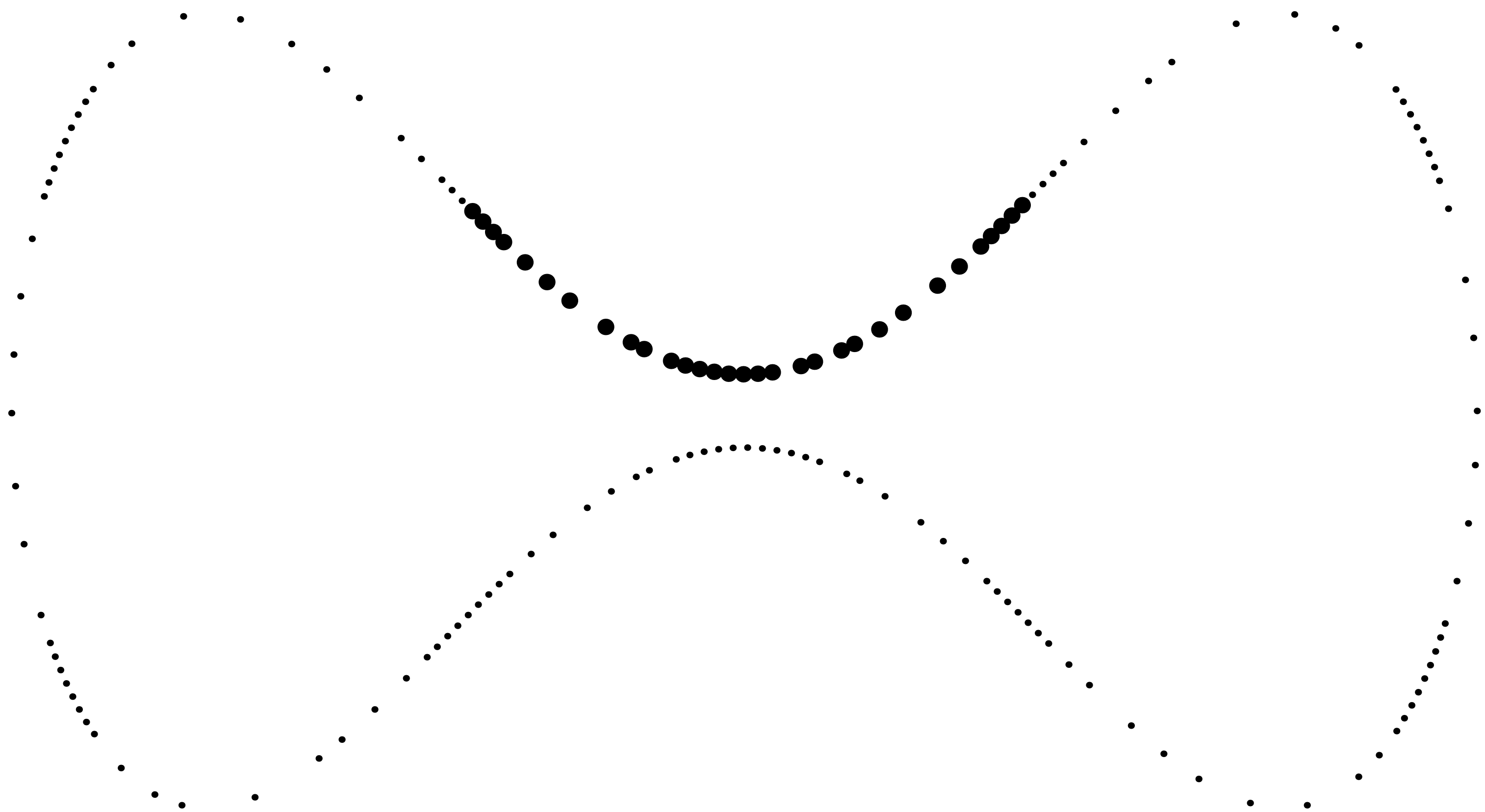


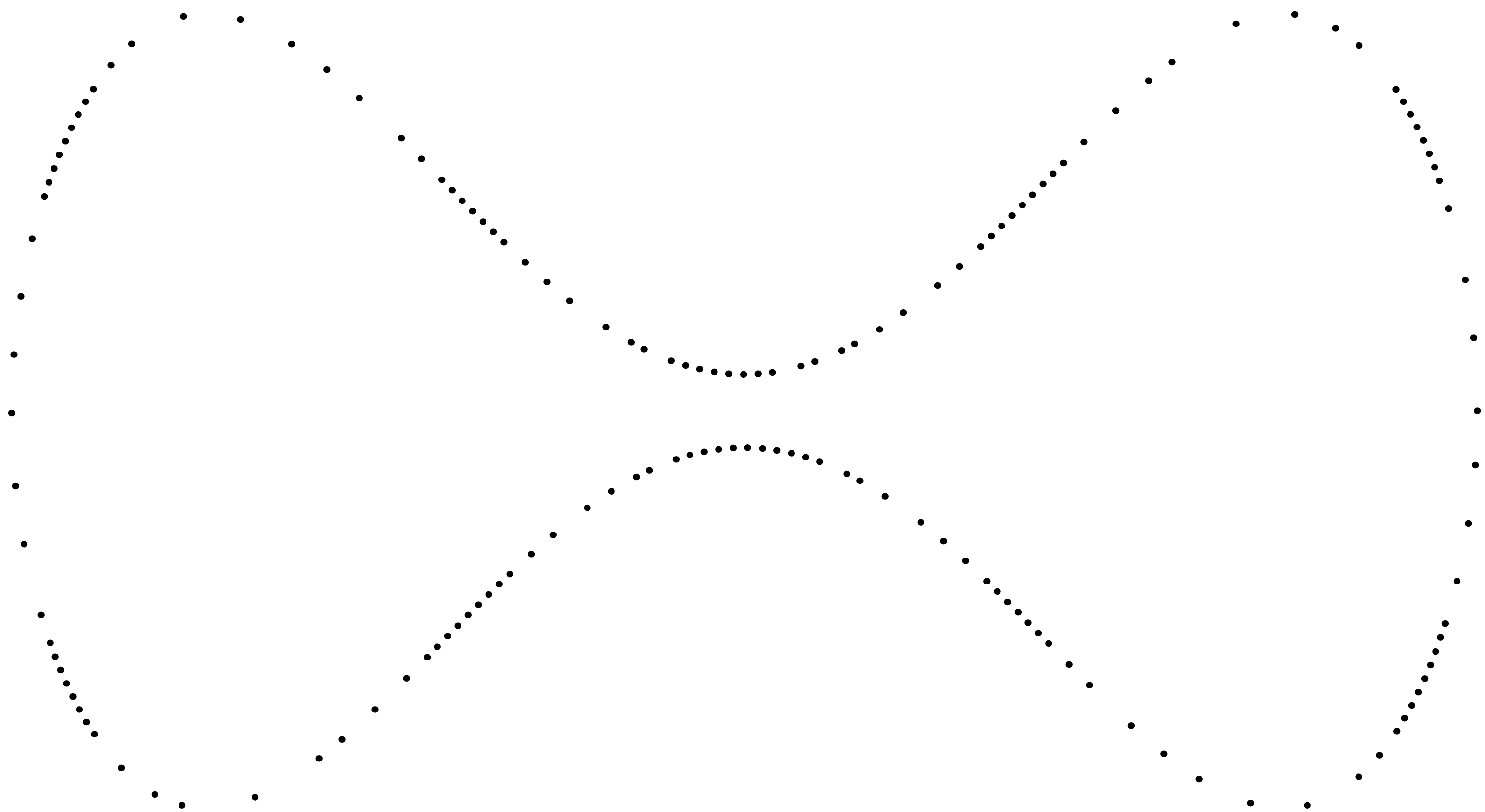


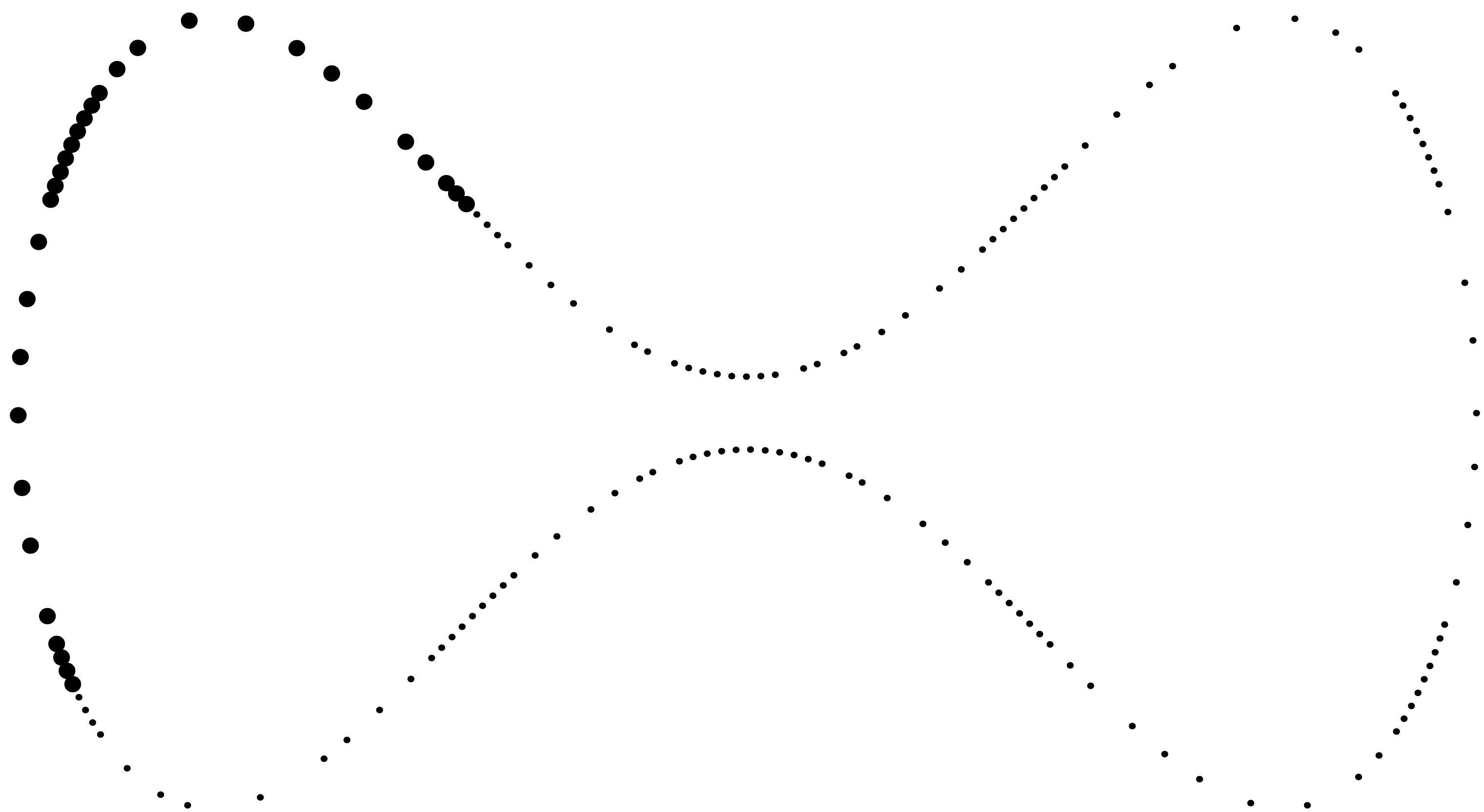


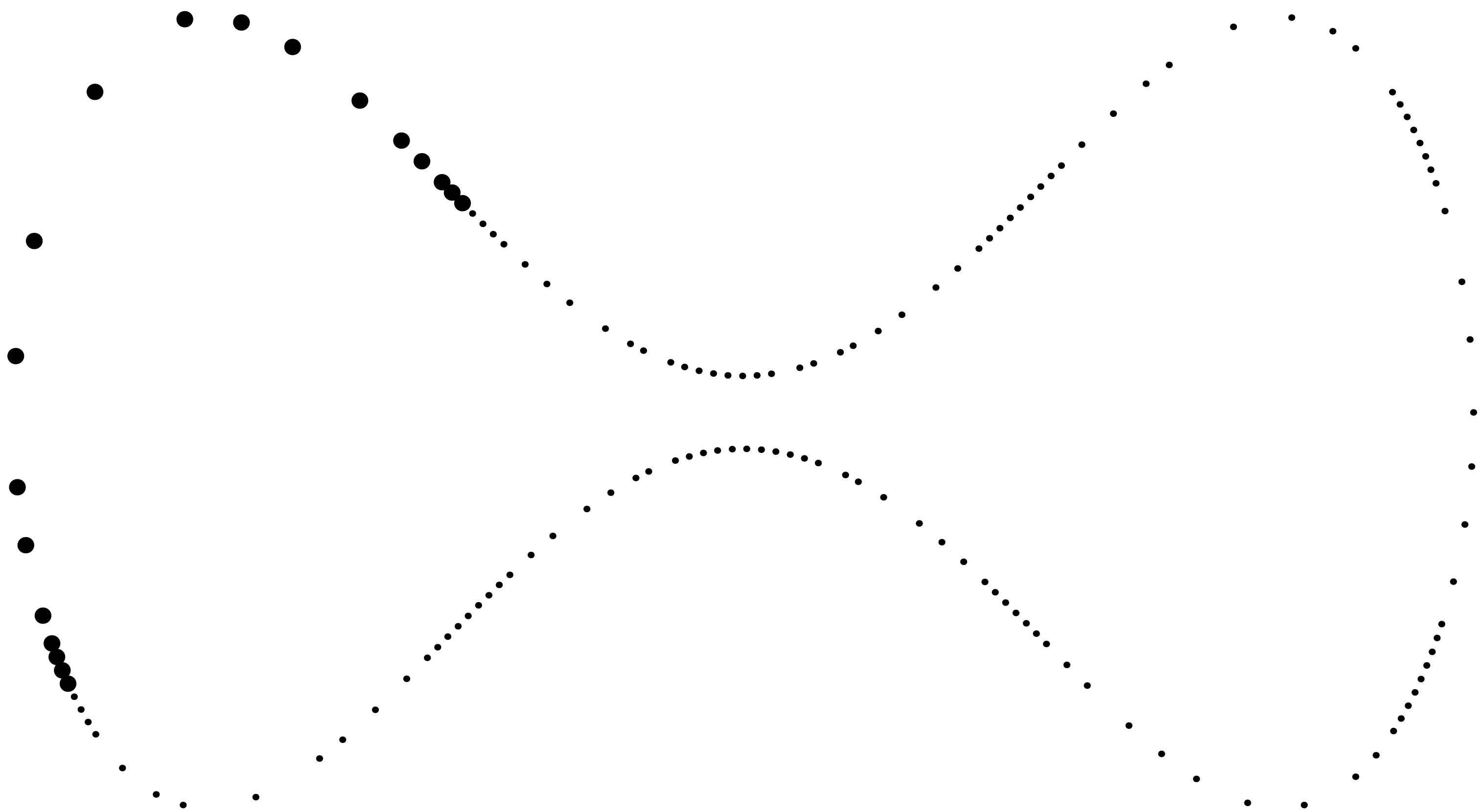


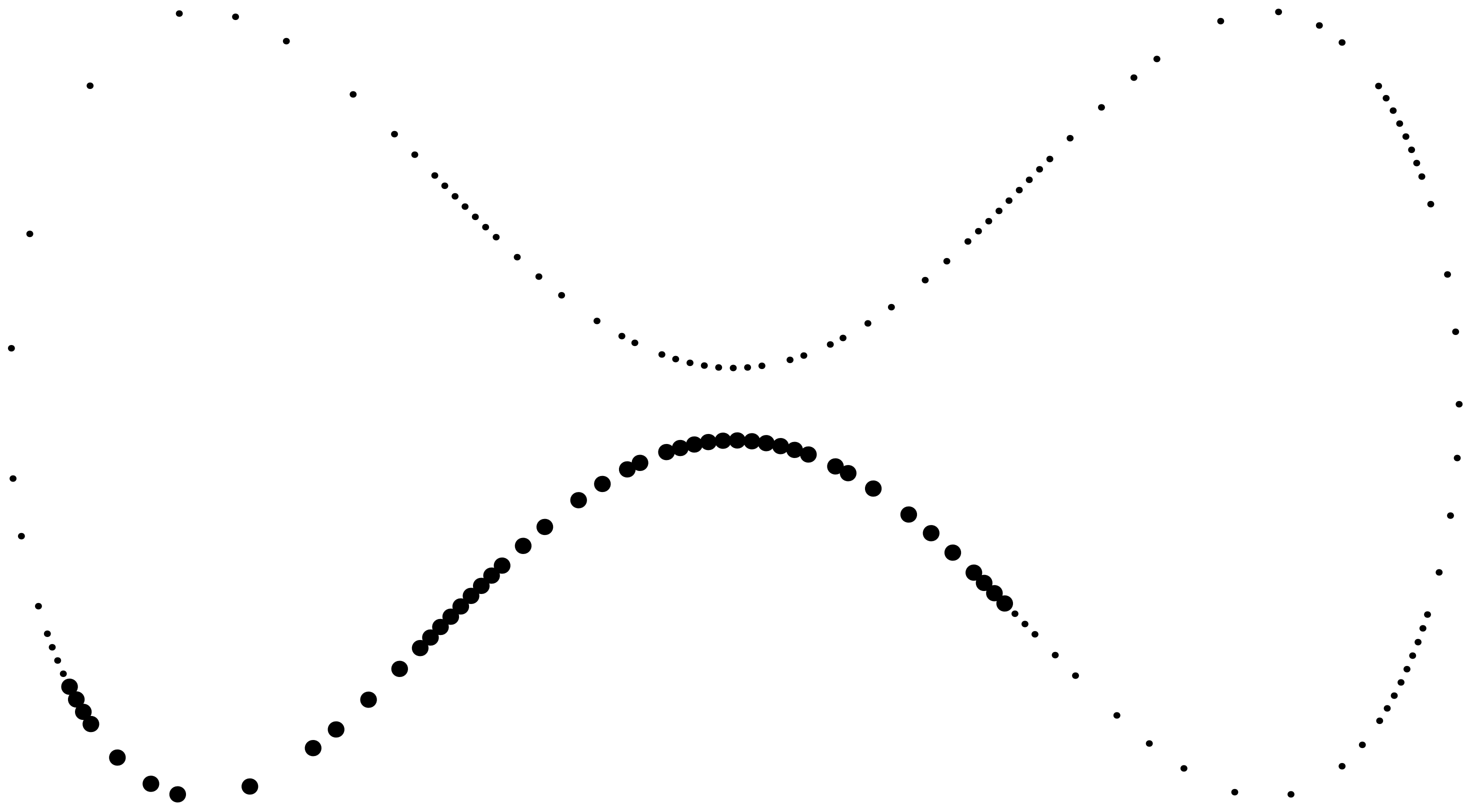


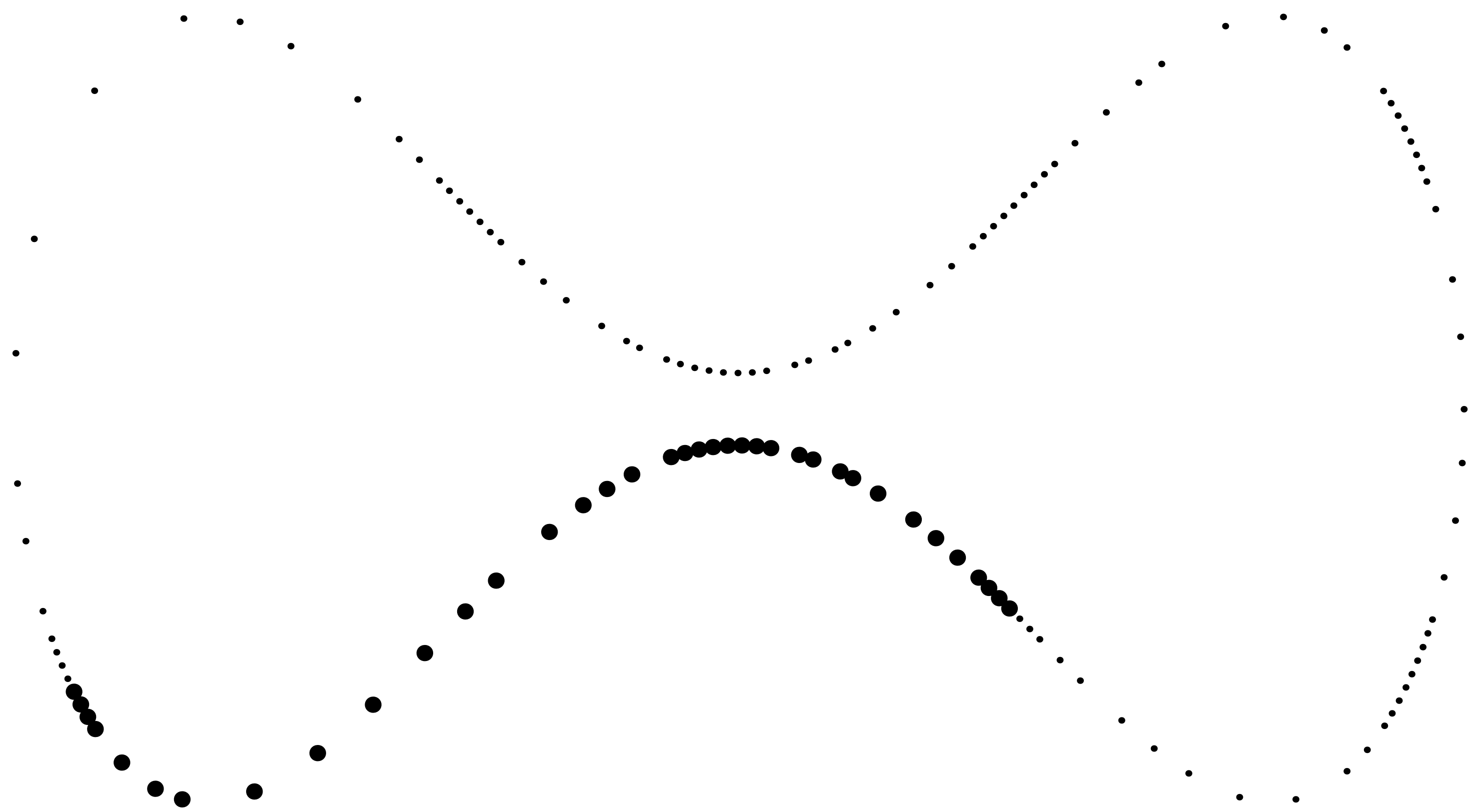


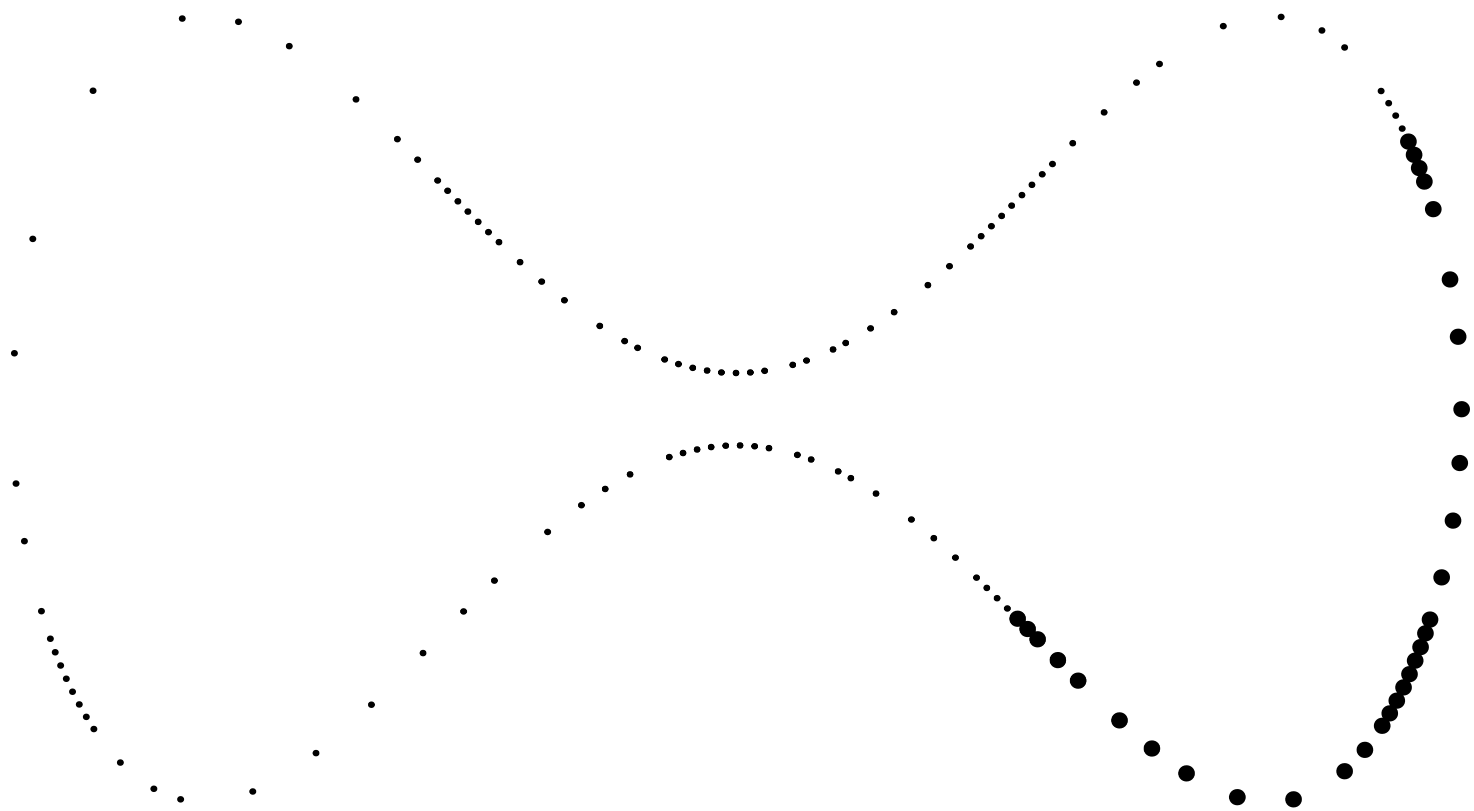


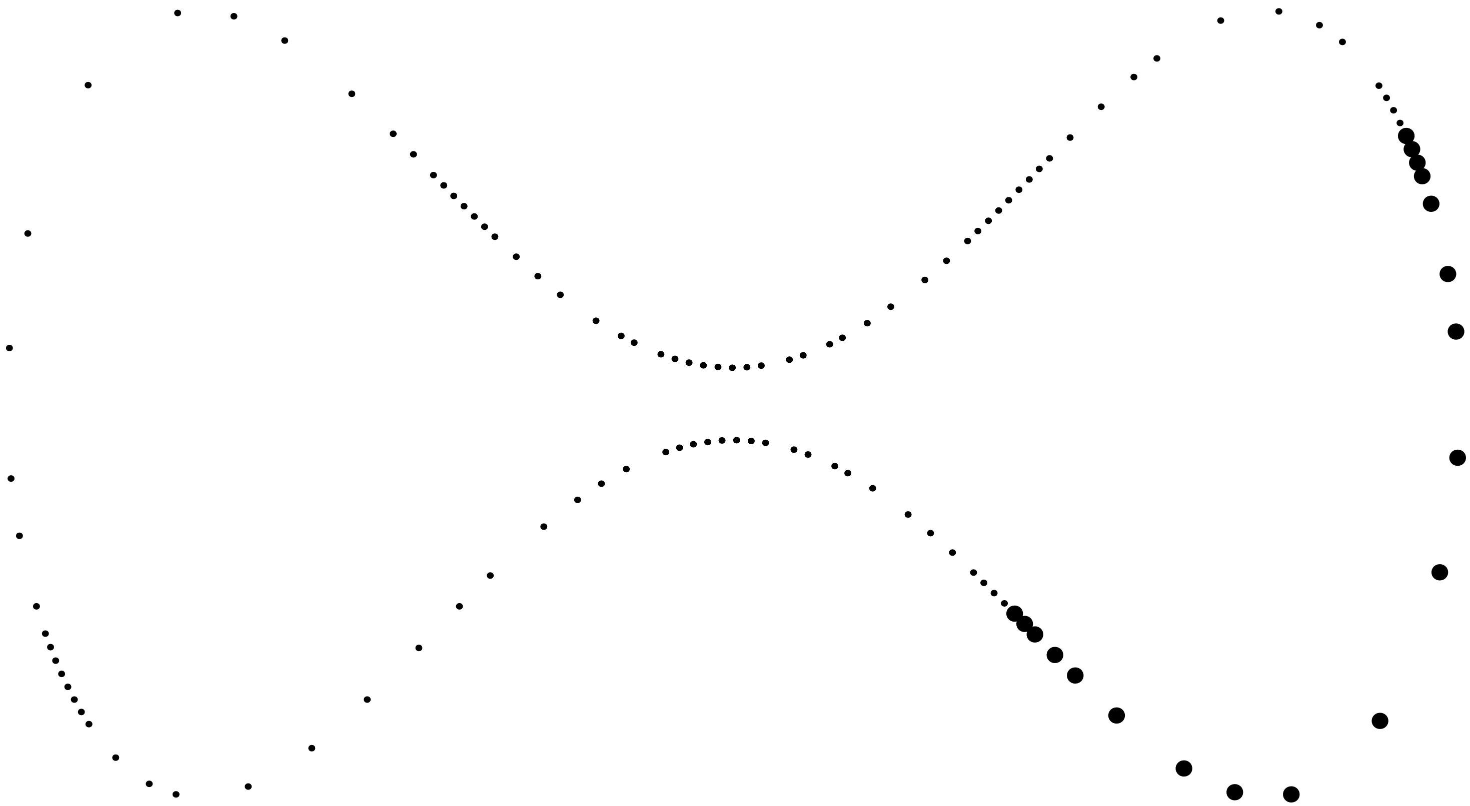


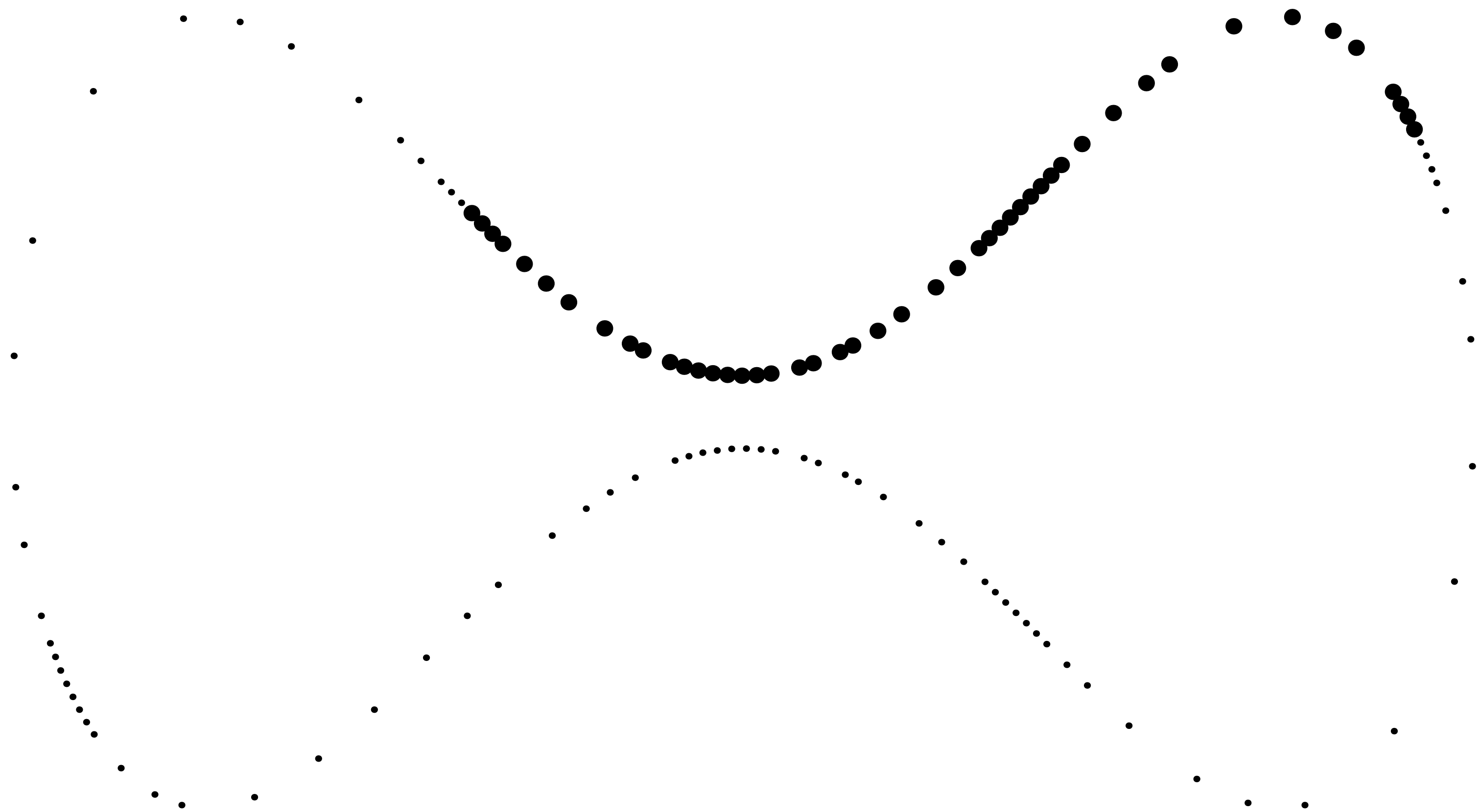


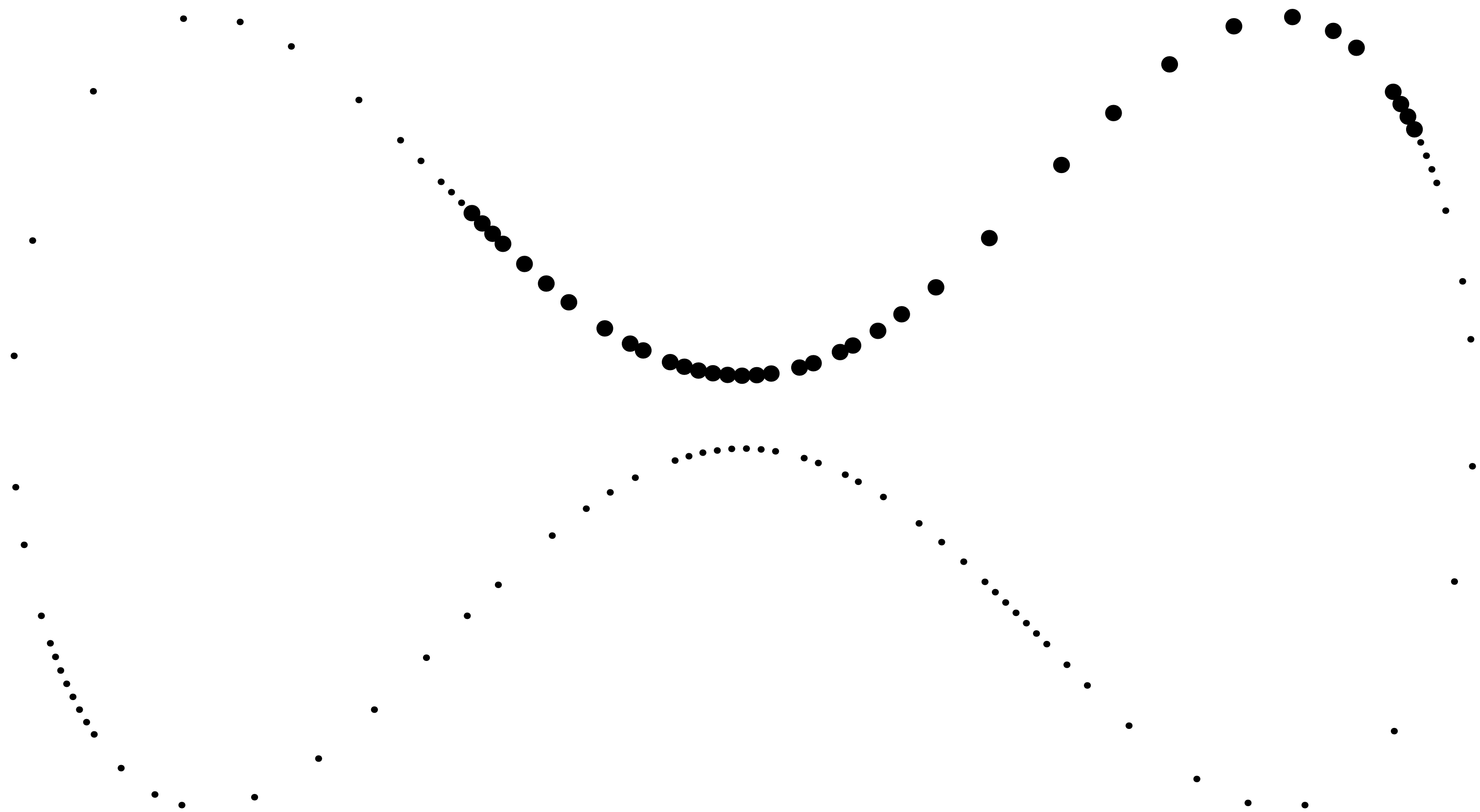


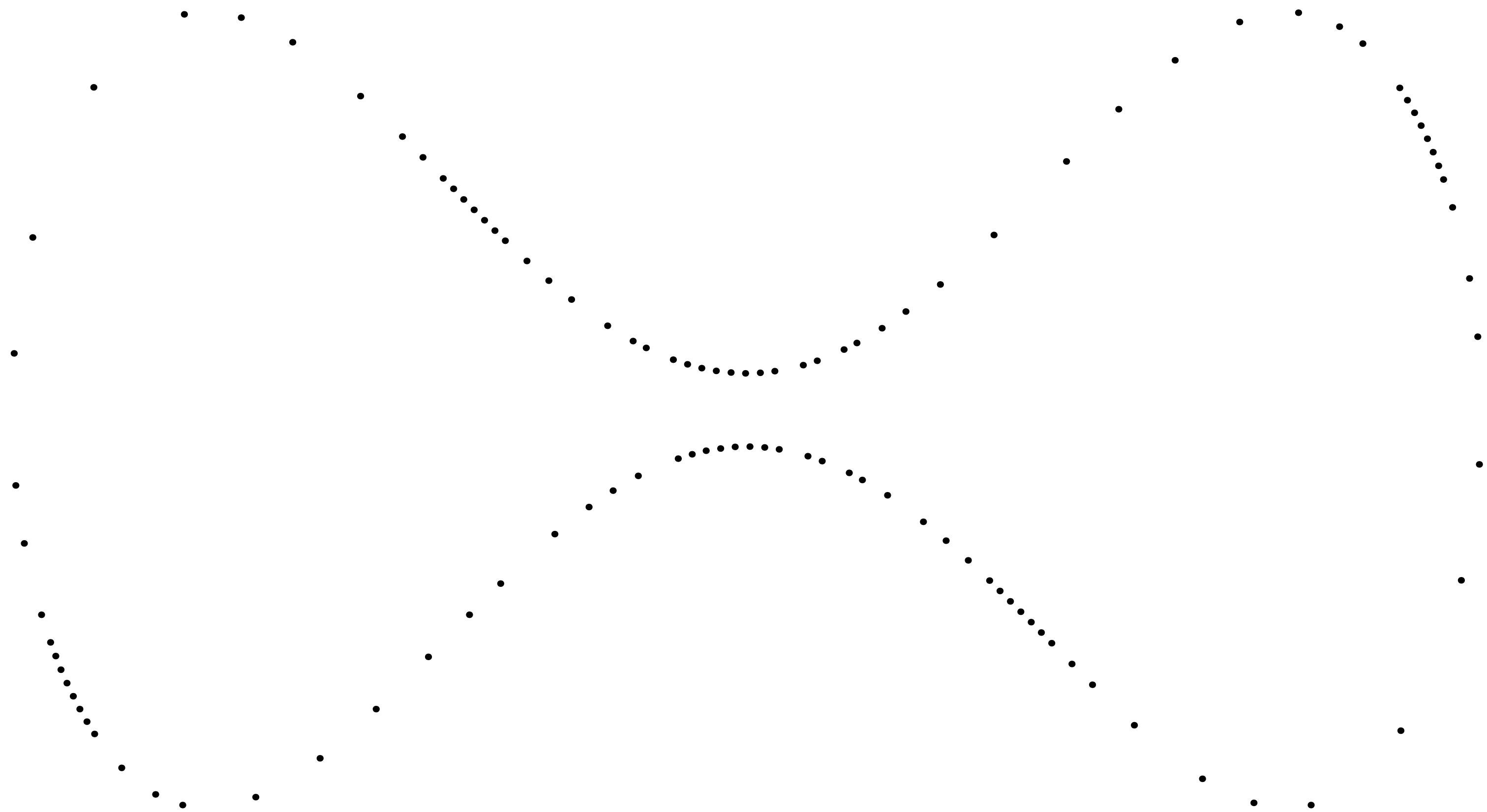












To be precise, each “compression” corresponds to a block-diagonal transformation.

In the end, we obtain a telescoped factorization

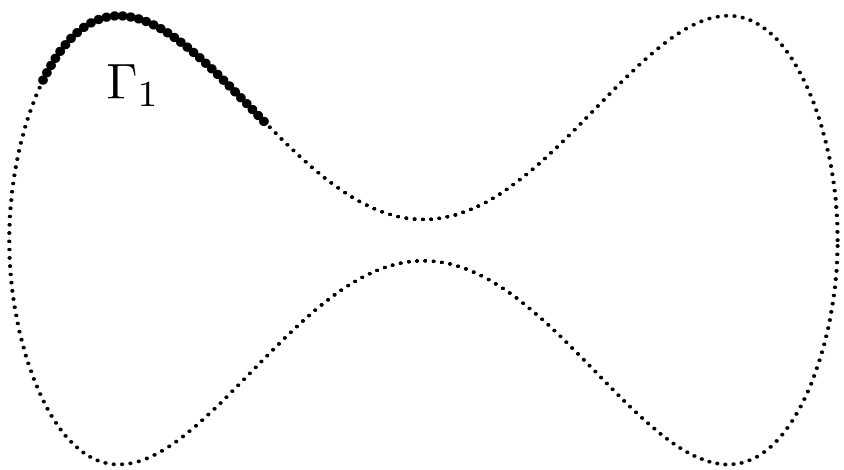
$$A^{-1} = B^{(1)} \left(B^{(2)} \tilde{A}^{-1} C^{(2)} + D^{(2)} \right) C^{(1)} + D^{(1)}.$$

A is the original matrix,
$\tilde{A}$ is the compressed matrix,
$B^{(j)}$, $C^{(j)}$, $D^{(j)}$ are block-diagonal, well-conditioned matrices.

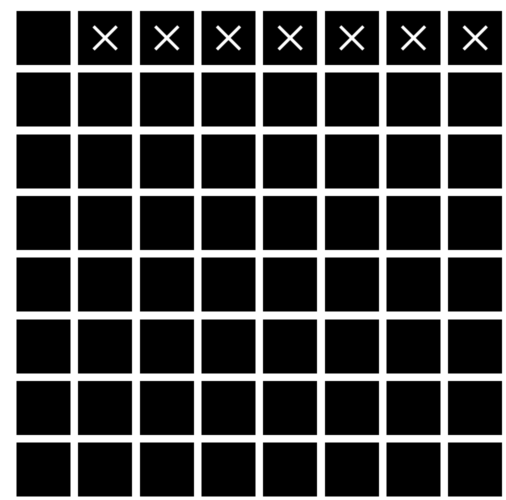
The final step is to invert $\tilde{A}$ by brute force.

So far, the algorithm is at best $O(n^2)$.

The bottle-neck is the formation and compression of the off-diagonal blocks.

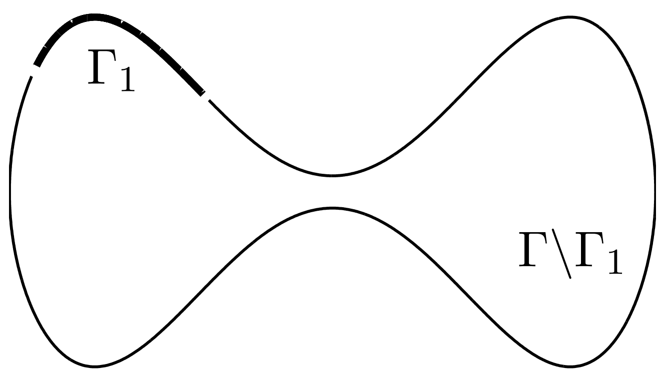


A segmented contour . . .

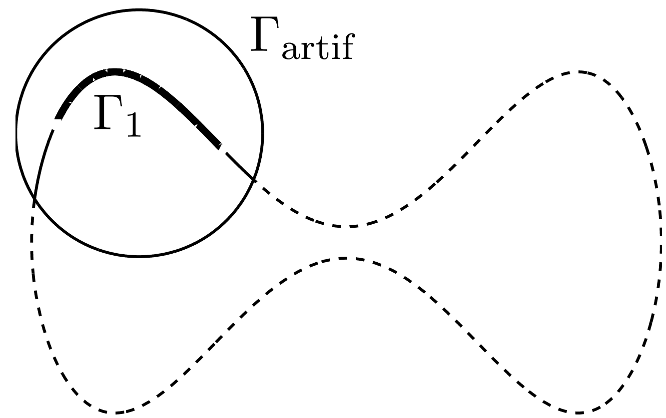


. . . and the corresponding matrix.

Localization using Green's identity:



Instead of compressing the large matrix representing the interaction between Γ_1 and all of the rest of the contour...



... it is sufficient to compress only the interaction between Γ_1 and the artificial contour Γ_{artif}.

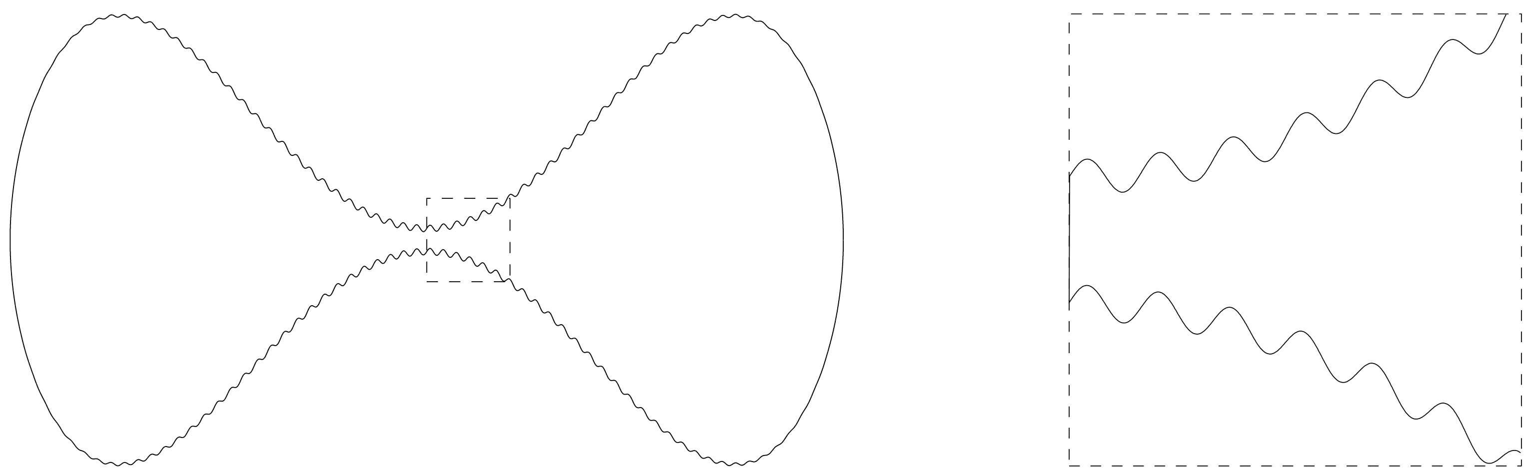
For non-oscillatory problems on one-dimensional contours, this technique brings the computational cost down to (at most) $O(n \log^2 n)$.

Numerical examples

The algorithm was implemented in Matlab (using mex-programs for the skeletonization).

The experiments were run on a Pentium IV with a 2.8Ghz processor and 512 Mb of RAM.

Example 1 - an exterior Laplace Dirichlet problem



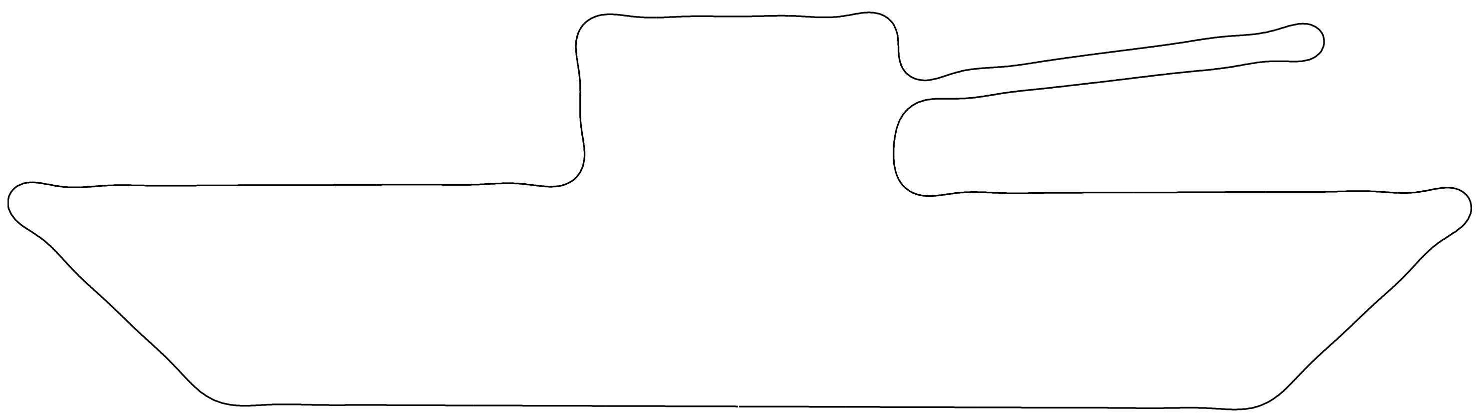
(a) (b)

(a) A rippled contour. (b) A close-up of the area marked by a dashed rectangle in (a). The number of ripples change between the different experiments to keep a constant ratio of 80 discretization nodes per wavelength.

N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	σ_{min}	M
400	160	2.4e-01	4.6e-03	2.3e-09	2.0e-09	1.2e-09	4.0e-02	954
800	214	4.7e-01	8.9e-03	2.3e-09	2.5e-09	2.8e-10	3.1e-02	2110
1600	286	7.5e+00	2.6e-02	1.9e-09	2.1e-09	9.8e-11	2.2e-02	4710
3200	361	1.1e+01	3.7e-02	—	1.4e-09	1.8e-10	1.8e-02	9781
6400	437	1.5e+01	7.2e-02	—	2.0e-09	1.3e-10	1.5e-02	20484
12800	508	2.1e+01	1.5e-01	—	1.6e-09	9.2e-11	1.4e-02	42307
25600	559	3.7e+01	2.9e-01	—	2.0e-09	1.3e-10	1.3e-02	86481
51200	599	8.0e+01	6.1e-01	—	1.8e-09	2.8e-10	—	177442
102400	634	1.9e+02	1.2e+00	—	1.4e-09	—	—	365495

Computational results for the double layer potential associated with an exterior Laplace Dirichlet problem on the rippled contour.

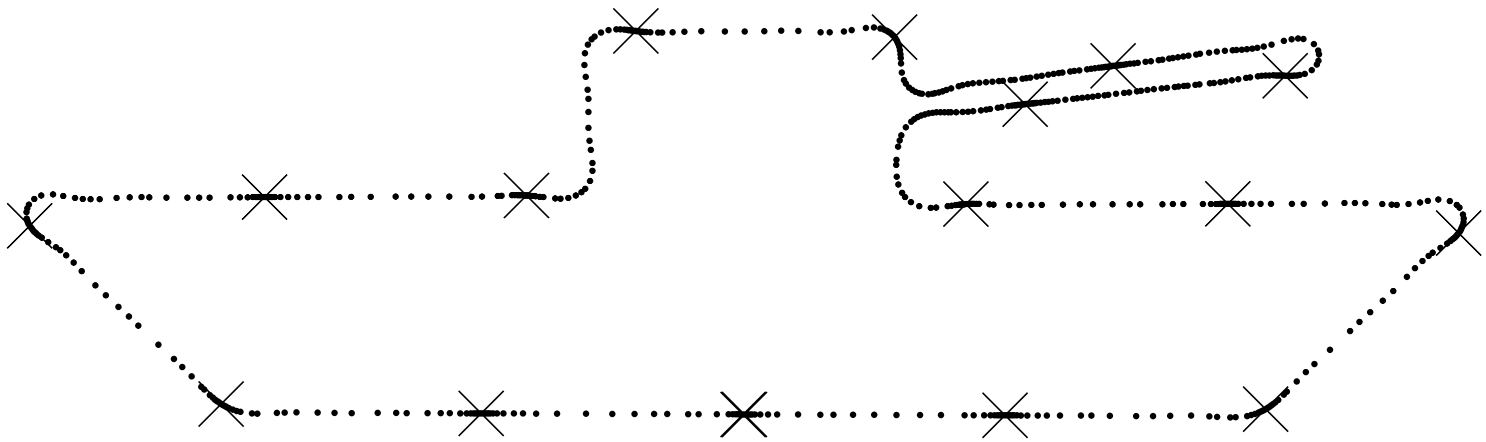
Example 2 - An exterior Helmholtz Dirichlet problem



A smooth contour. Its length is roughly 15 and its horizontal width is 2.

k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{res}	E_{pot}	σ_{min}	M
21	800	435	1.5e+01	3.3e-02	9.7e-08	7.1e-07	6.5e-01	12758
40	1600	550	3.0e+01	6.7e-02	6.2e-08	4.0e-08	8.0e-01	25372
79	3200	683	5.3e+01	1.2e-01	5.3e-08	3.8e-08	3.4e-01	44993
158	6400	870	9.2e+01	2.0e-01	3.9e-08	2.9e-08	3.4e-01	81679
316	12800	1179	1.8e+02	3.9e-01	2.3e-08	2.0e-08	3.4e-01	160493
632	25600	1753	4.3e+02	7.5e+00	1.7e-08	1.4e-08	3.3e-01	350984

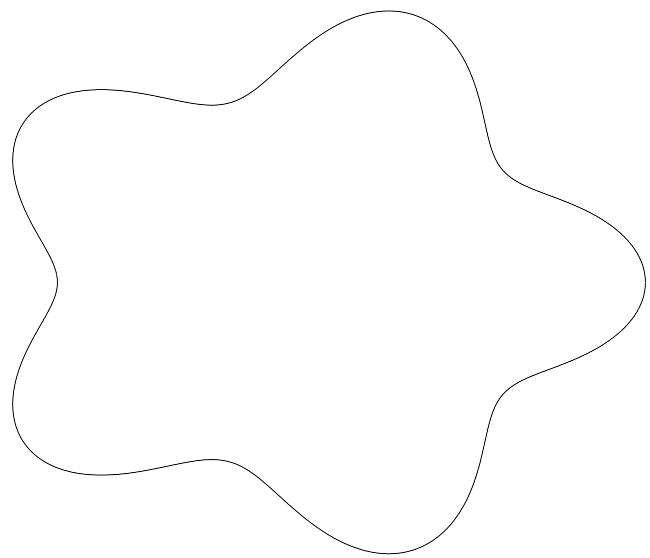
Computational results for an exterior Helmholtz Dirichlet problem discretized with 10^{th} order accurate quadrature. The Helmholtz parameter was chosen to keep the number of discretization points per wavelength constant at roughly 45 points per wavelength (resulting in a quadrature error about 10^{-12}).



The points left after two rounds of compression.
The crosses mark the boundary points between adjacent clusters.
(The figure actually shows the results of a Laplace problem.)

Example 3 - An interior Helmholtz Dirichlet problem

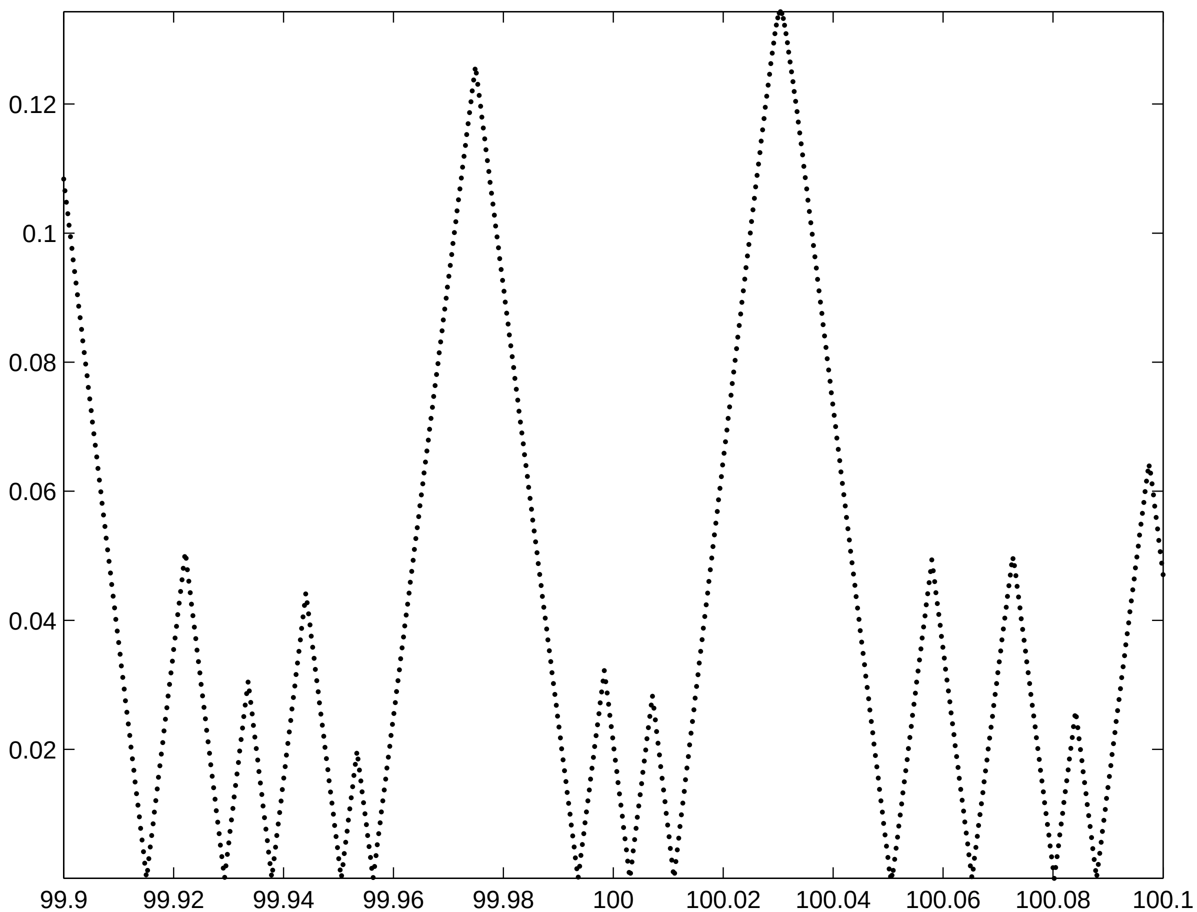
Close to a resonance.



A smooth pentagram. Its diameter is 2.5 and its length is roughly 8.3.

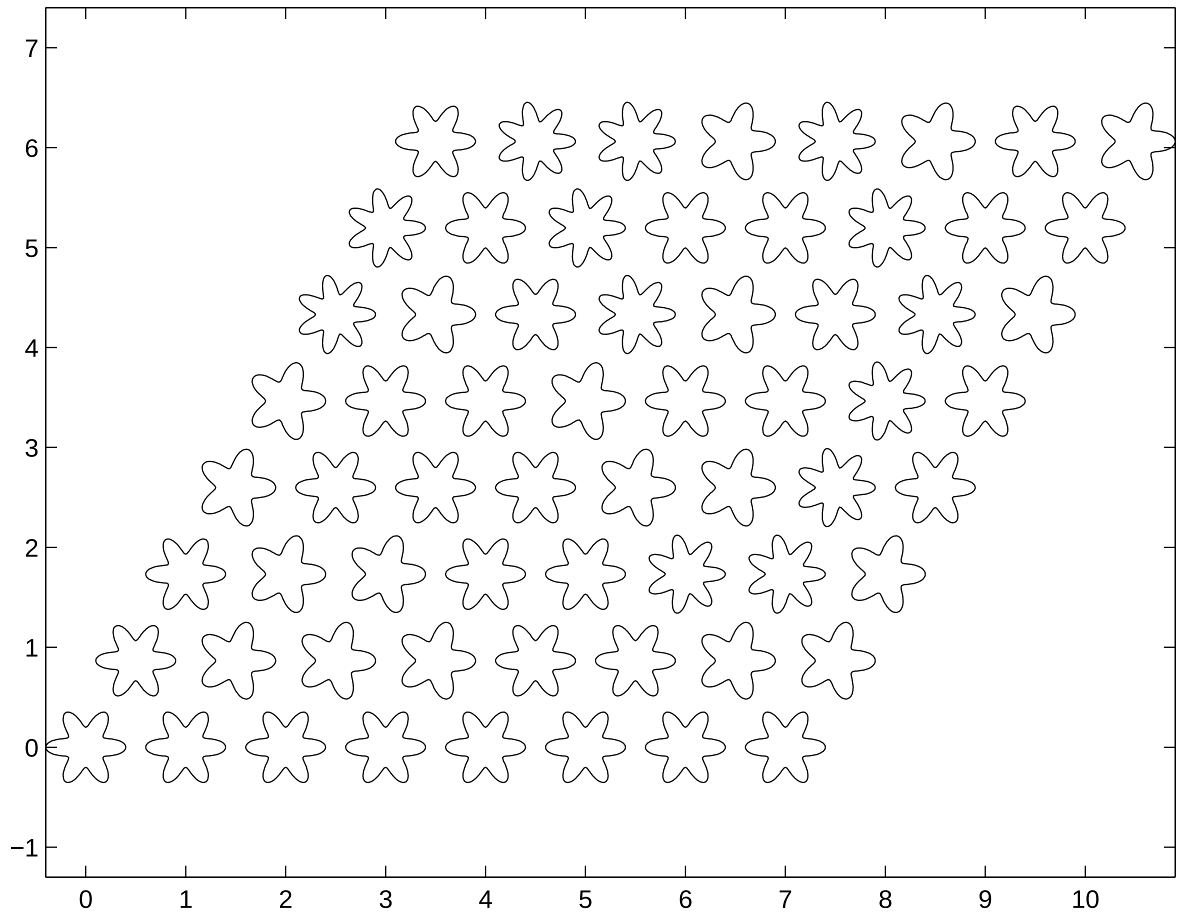
j	p_j	n_j	γ_j	t_j	$\|\|C^{(j)}\|\|_\infty$	$\|\|B^{(j)}\|\|_\infty$	$\|\|D^{(j)}\|\|_\infty$
1	128	50.00	0.76	15.50	1.12e+00	1.12e+00	4.20e-02
2	64	76.00	0.59	14.32	3.27e+01	3.27e+01	1.75e+00
3	32	89.72	0.60	8.94	1.63e+01	1.62e+01	9.28e-01
4	16	107.00	0.64	6.27	9.09e+00	9.17e+00	2.41e+00
5	8	138.00	0.72	5.97	7.32e+00	7.31e+00	3.64e+00
6	4	199.50	0.80	7.76	3.22e+00	3.23e+00	3.86e+00

Interior Helmholtz Dirichlet problem on a smooth pentagram for the case $N = 6\,400$, $k = 100.011027569\cdots$ and $\sigma_{\text{min}} = 0.00001366\cdots$. For each level j, the table shows the number of clusters p_j on that level, the average size of a cluster n_j, the compression ratio γ_j, the time required for the factorization t_j and the size of the matrices $B^{(j)}$, $C^{(j)}$ and $D^{(j)}$ in the maximum norm. For this computation, $E_{\text{res}} = 2.8 \cdot 10^{-10}$ and $E_{\text{pot}} = 3.3 \cdot 10^{-5}$.



Plot of $\sigma_{\min}$ versus k for an interior Helmholtz problem on the smooth pentagram. The values shown were computed using a matrix of size $N = 6400$. Each point in the graph required about $60s$ of CPU time.

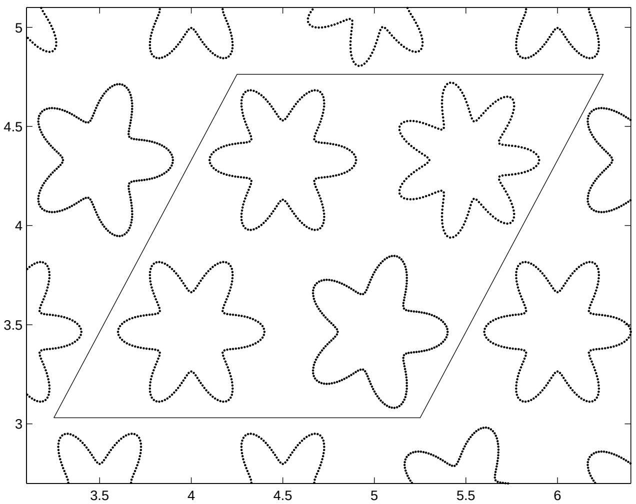
Example 4



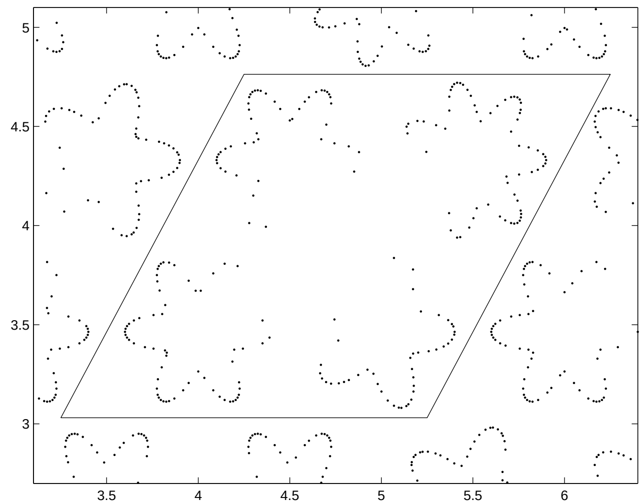
Contour:	t_{tot}	N_{start}	N_{final}	M
Rippled dumb-bell	37s	25 600	559	86Mb
Star-fish lattice	172s	25 600	1202	210Mb

Test results for two experiments concerning the matrix obtained by discretizing a double layer Laplace Dirichlet problem.

For the lattice problem, the computational complexity turns out to be $O(N^{3/2})$.



(a)



(b)

Fig. (a) shows a close-up of the star-fish lattice. Fig. (b) shows the nodes remaining after the interaction between the cluster formed by the points inside the parallelogram and the remainder of the contour has been compressed.

Summary

We have presented an $O(n \log^2 n)$ direct solver for contour integral equations with non-oscillatory (or moderately oscillatory) kernels.

Work in progress:

- Applications of the scheme.
- Computing standard factorizations (SVD) of a dense matrix.
- Integral equations defined on surfaces rather than curves.
- Highly oscillatory problems.

Tech reports describing these techniques are available on the web (off the Yale math department home page) or by request.