New data structure for univariate polynomial approximation and applications to root isolation

Guillaume Moroz

January 21, 2022

Problems

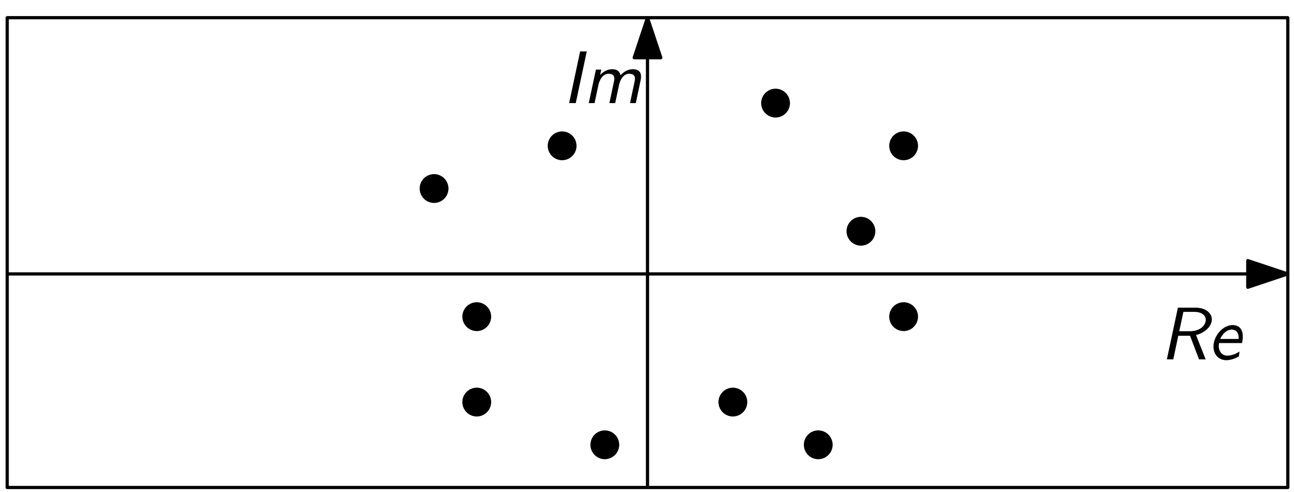
$$f(z) = a_0 + \cdots + a_d z^d \qquad a_k \in \mathbb{C}$$

Multipoint evaluation

Given d complex numbers z_k, evaluate all the $f(z_k)$.

Root finding

Find all the complex solutions ζ_k of $f(z) = 0$.



$\mathbb{C} \simeq \mathbb{R}^2$

Problems

Evaluation output

- Arbitrary precision
- Finite precision

Light-year: $9\,460\,730\,472\,580\,800$ m

$9.460 \cdot 10^{15}$ m

- Complexity for numbers of bit size m

  1 arithmetic operation costs $\widetilde{O}(m)$ bit operations

Root finding output

- Initial point and program for convergence

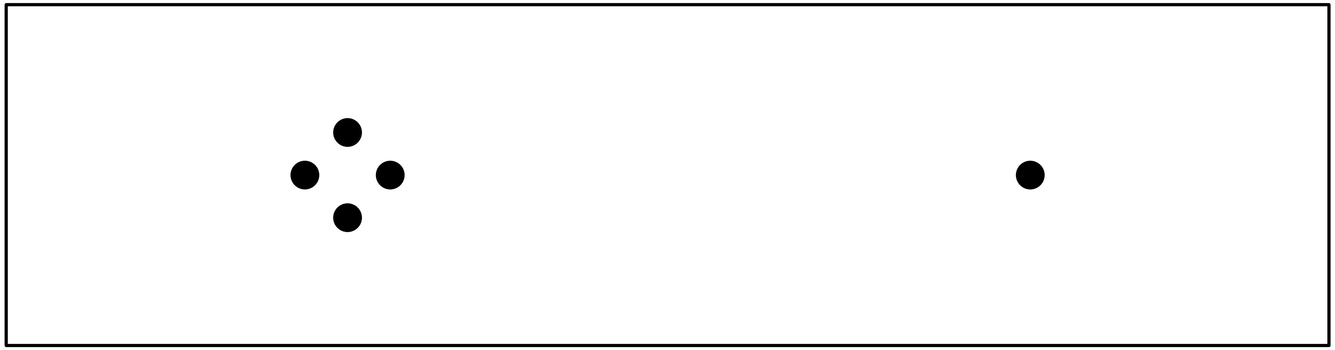
Newton: $x_0 = z$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

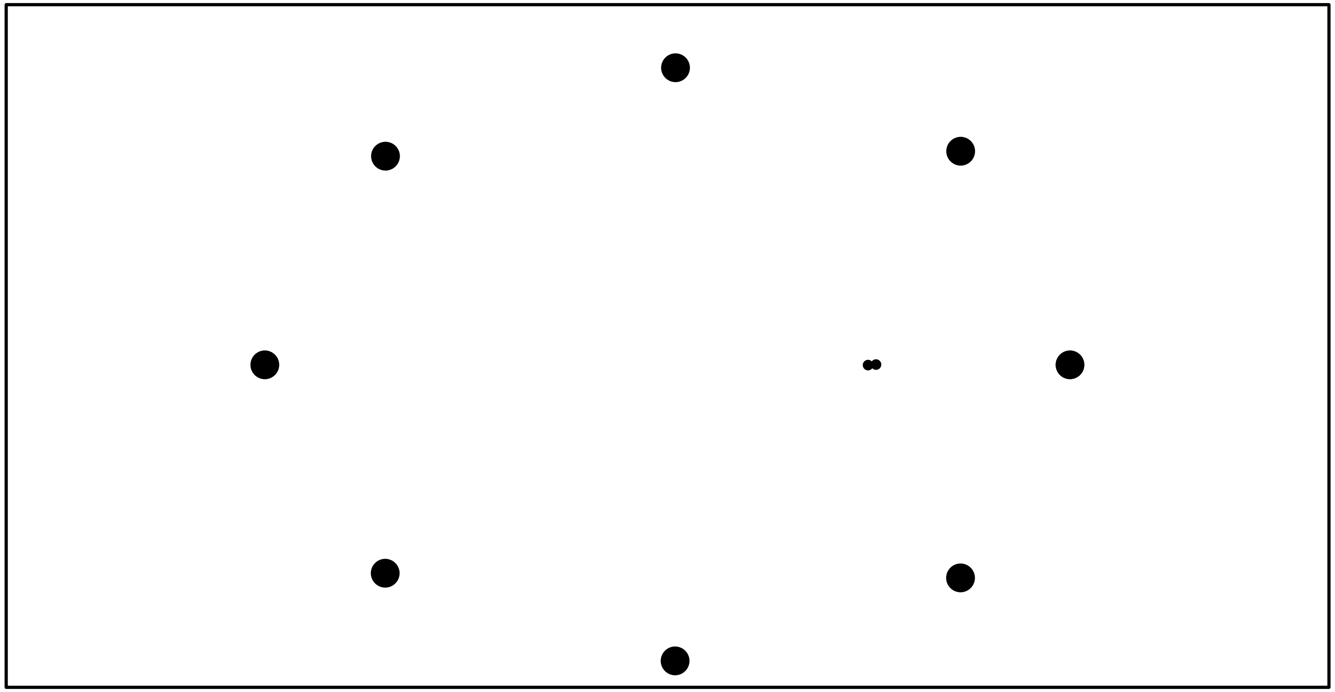
- Isolating disk

Numerically ill-conditioned root finding

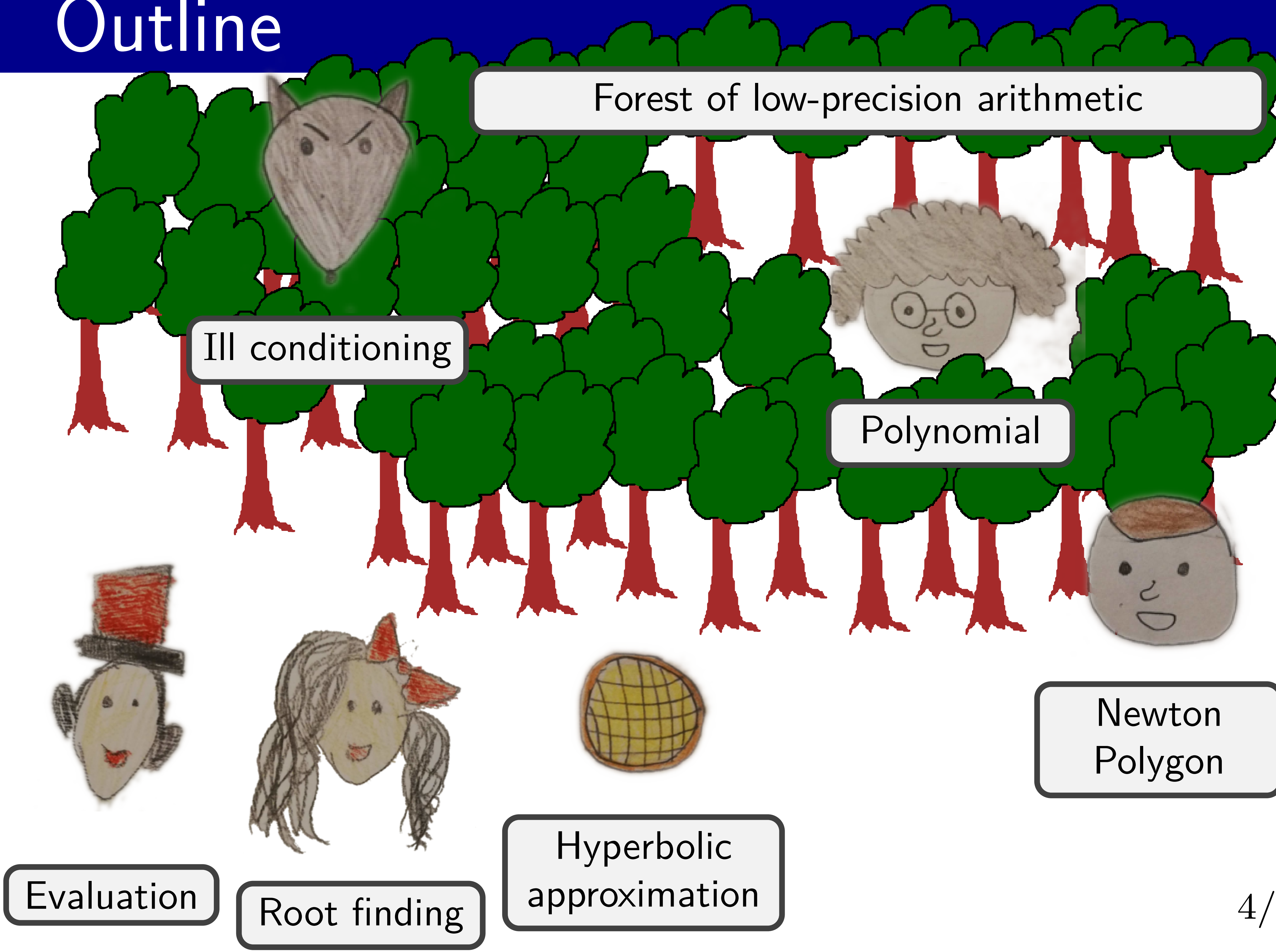
$$(z^4 - \epsilon)(z - 1) = 0$$



$$z^{10} - 2(2z^2 - 1)^2 = 0$$ [Mignotte 82]



Outline


Forest of low-precision arithmetic
Ill conditioning
Polynomial
Newton Polygon
Evaluation
Root finding
Hyperbolic approximation

Conditioning of root finding

$$f(z) = a_0 + \cdots + a_d z^d \qquad a_k \in \mathbb{C}$$

ζ simple root of f

Condition number [Bürgisser 2013]

$$\kappa_\zeta = \lim_{\|\Delta f\| \to 0} \frac{|\Delta\zeta|}{\|\Delta f\|} = \frac{\max(1, |\zeta|^d)}{|f'(\zeta)|}$$

Conditioning of root finding

$$f(z) = a_0 + \cdots + a_d z^d \qquad a_k \in \mathbb{C}$$

$$h(z) = h_0 + \cdots + h_d z^d \qquad \sum_k |h_k| \le \varepsilon$$

$\psi(a_0 + h_0, \ldots, a_d + h_d) =$ unique root of $f + h$ in U

ζ simple root of f

$\zeta \in U \subset \mathbb{C}$
neighborhood of ζ

Condition number [Bürgisser 2013]

$$\kappa_\zeta = \lim_{\varepsilon \to 0} \max_{\|h\| \le \varepsilon} \frac{|\psi(f+h) - \psi(f)|}{\varepsilon} = \frac{\max(1, |\zeta|^d)}{|f'(\zeta)|}$$

Proof: $\quad 0 = (f+h)(\psi(f+h)) - f(\psi(f))$

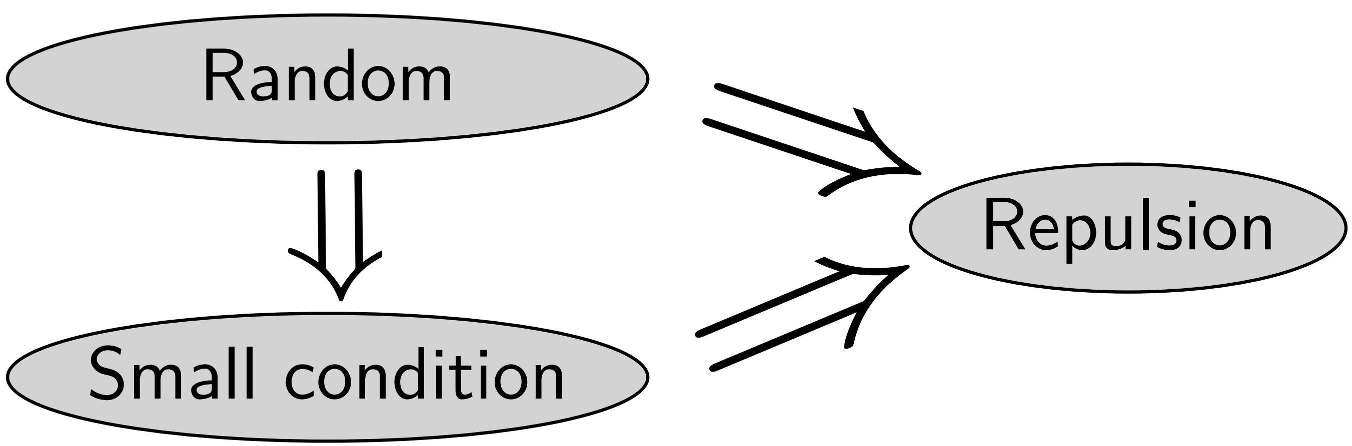
$$\approx h(\zeta) + f'(\zeta)(\psi(f+h) - \psi(f))$$

Properties of polynomials

- Small condition number $\Rightarrow$ large isolating disks
  [Kantorovitch 1948]

- Random coefficients $\Rightarrow$ small condition number
  [Cucker, Krick, Malajovich, Wschebor 2012]

- Random coefficients $\Rightarrow$ large isolating disks
  [Hough, Krishnapour, Peres, Virág 2009]



State of the art: multipoint evaluation

Evaluate $f(z)$ on d points with error in 2^{-m} $\quad |a_k| < 2^m$

Hörner

$$a_0 + z(a_1 + z(\cdots + z(a_{d-1} + z a_d)\cdots))$$

$\rightarrow$ multipoint evaluation in $\widetilde{O}(d^2 m)$ bit operations

Divide and conquer

$$f(z) \mod \prod_{k=1}^{d} (z - z_k)$$

- $\widetilde{O}(d)$ arithmetic operations [Fiduccia 1972]
- $\widetilde{O}(d(d+m))$ bit operations [van der Hoeven 2008]

Piecewise low-degree polynomial approximation

$\rightarrow$ multipoint evaluation in:

- $\widetilde{O}(d^{3/2} m^{3/2})$ bit operations [van der Hoeven 2008]

State of the art: multipoint evaluation

Evaluate $f(z)$ on d points with error in 2^{-m} $\quad |a_k| < 2^m$

Hörner

$$a_0 + z(a_1 + z(\cdots + z(a_{d-1} + za_d)\cdots))$$

$\rightarrow$ multipoint evaluation in $\widetilde{O}(d^2 m)$ bit operations

Divide and conquer

$f(z) \mod \prod_{k=1}^{d/2}(z - z_k)$ $\quad$ $f(z) \mod \prod_{k=d/2}^{d}(z - z_k)$

- $\widetilde{O}(d)$ arithmetic operations [Fiduccia 1972]
- $\widetilde{O}(d(d+m))$ bit operations [van der Hoeven 2008]

Piecewise low-degree polynomial approximation

$\rightarrow$ multipoint evaluation in:

- $\widetilde{O}(d^{3/2}m^{3/2})$ bit operations [van der Hoeven 2008]

State of the art: multipoint evaluation

Evaluate $f(z)$ on d points with error in 2^{-m} $\quad |a_k| < 2^m$

Hörner

$$a_0 + z(a_1 + z(\cdots + z(a_{d-1} + za_d)\cdots))$$

$\to$ multipoint evaluation in $\widetilde{O}(d^2 m)$ bit operations

Divide and conquer

$f(z) \mod (z - z_1)$ $\quad\cdots\quad$ $f(z) \mod (z - z_k)$ $\quad\cdots\quad$ $f(z) \mod (z - z_d)$

- $\widetilde{O}(d)$ arithmetic operations [Fiduccia 1972]
- $\widetilde{O}(d(d+m))$ bit operations [van der Hoeven 2008]

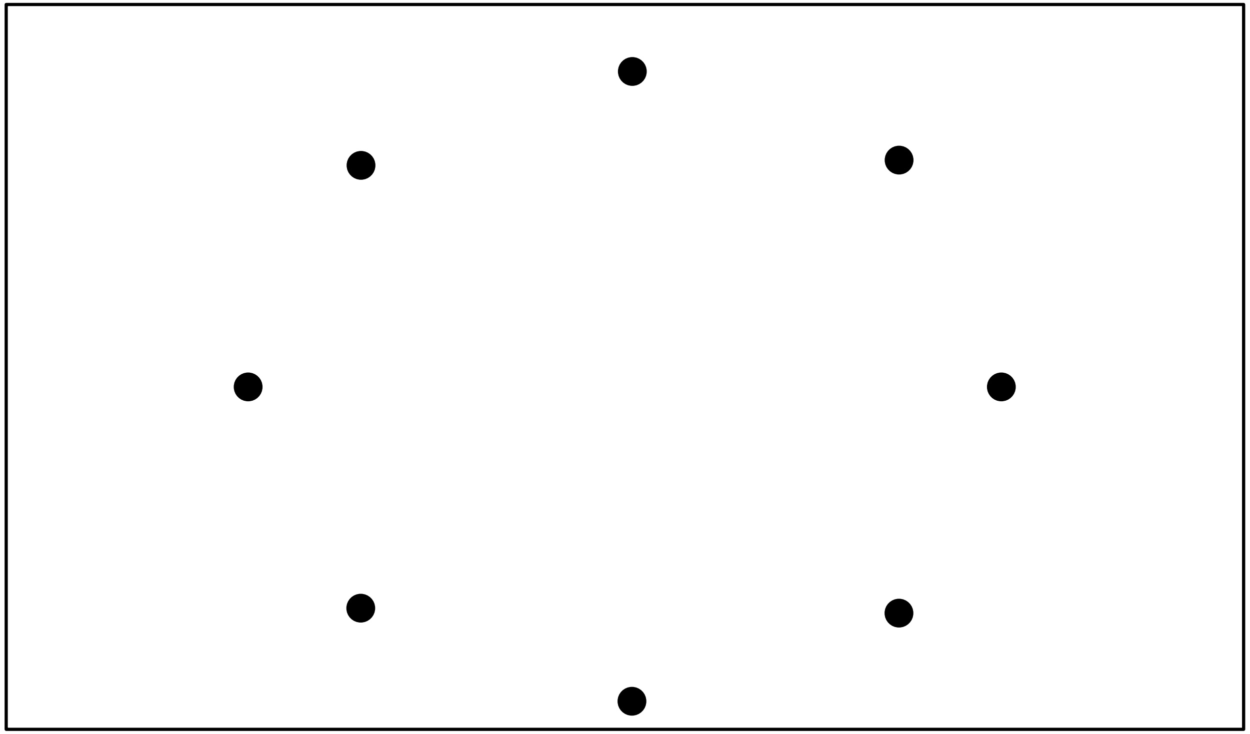
Piecewise low-degree polynomial approximation

$\to$ multipoint evaluation in:

- $\widetilde{O}(d^{3/2} m^{3/2})$ bit operations [van der Hoeven 2008]

State of the art: multipoint evaluation

Evaluation on the roots of unity $w_k = e^{i\pi k/d}$

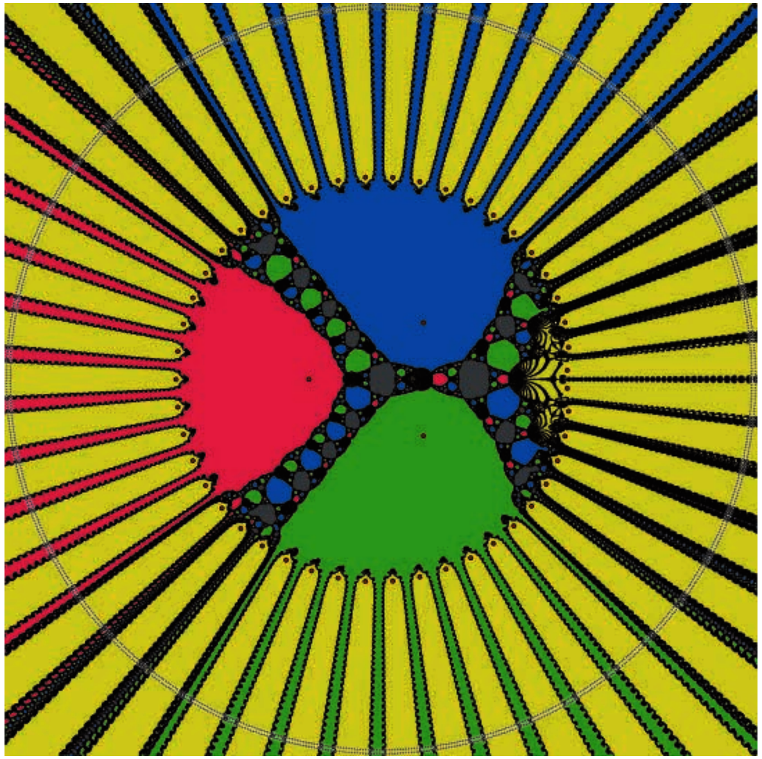


$\rightarrow$ evaluation on w_k in $\widetilde{O}(dm)$ using Fast Fourier Transform [Gauss 1805, Cooley, Tukey 1965, Schönhage 1982]

$\rightarrow$ interpolation from $f(w_k)$ in $\widetilde{O}(dm)$

State of the art: root finding

[Hubbard, Schleicher, Sutherland 2001]



Newton

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Aberth-Ehrlich variant (1967)

$$F(z) = \frac{f(z)}{(z-z_2)\cdots(z-z_d)}$$

Approximate factorization

$$\|\prod(z - z_k) - f(z)\| \leq 2^{-m}\|f\|$$

$\rightarrow$ approximation in $\widetilde{O}(d(d+m))$ bit operations

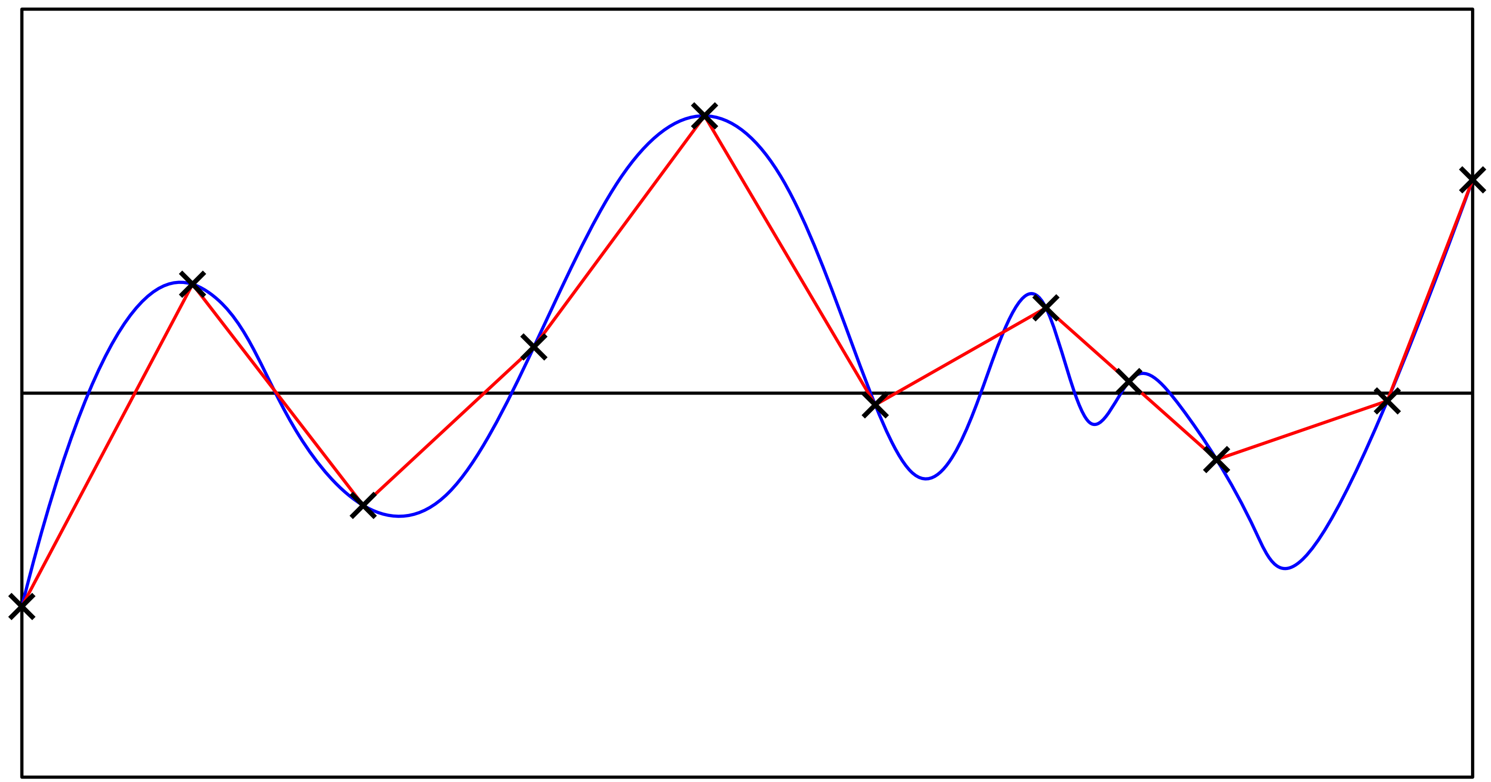
[Schönhage 1982, Pan 2002]

Other methods

Subdivision, Weierstrass, eigenvalue of companion matrix, . . .

State of the art: root finding

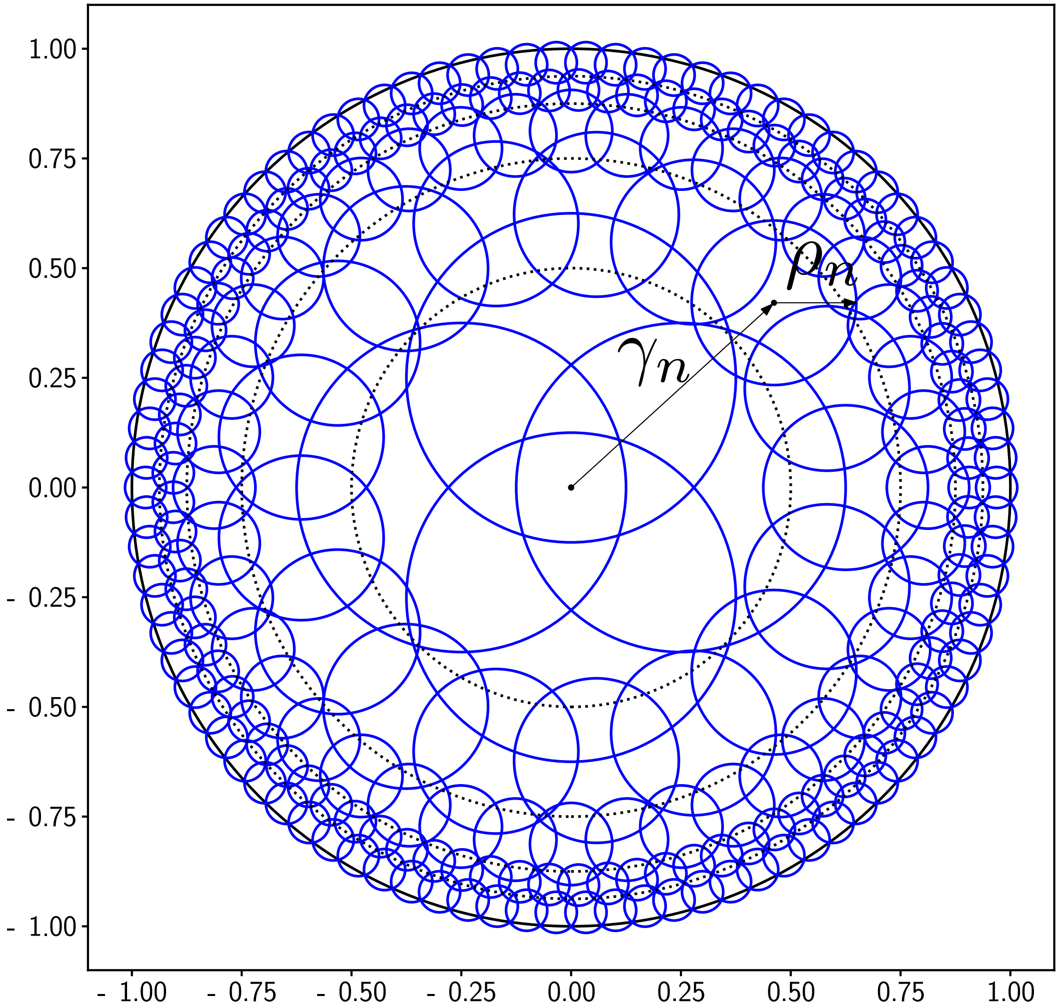
Piecewise linear approximation



→ piecewise low-degree polynomial approximations

[Boyd 2006, etc.]

Hyperbolic approximation



$$0 \le n < N - 1 = O\left(\log \frac{d}{m}\right)$$

$$\begin{cases} \gamma_n = 1 - \dfrac{3}{4}\dfrac{1}{2^n} \\ \rho_n = \dfrac{3}{8}\dfrac{1}{2^n} \end{cases}$$

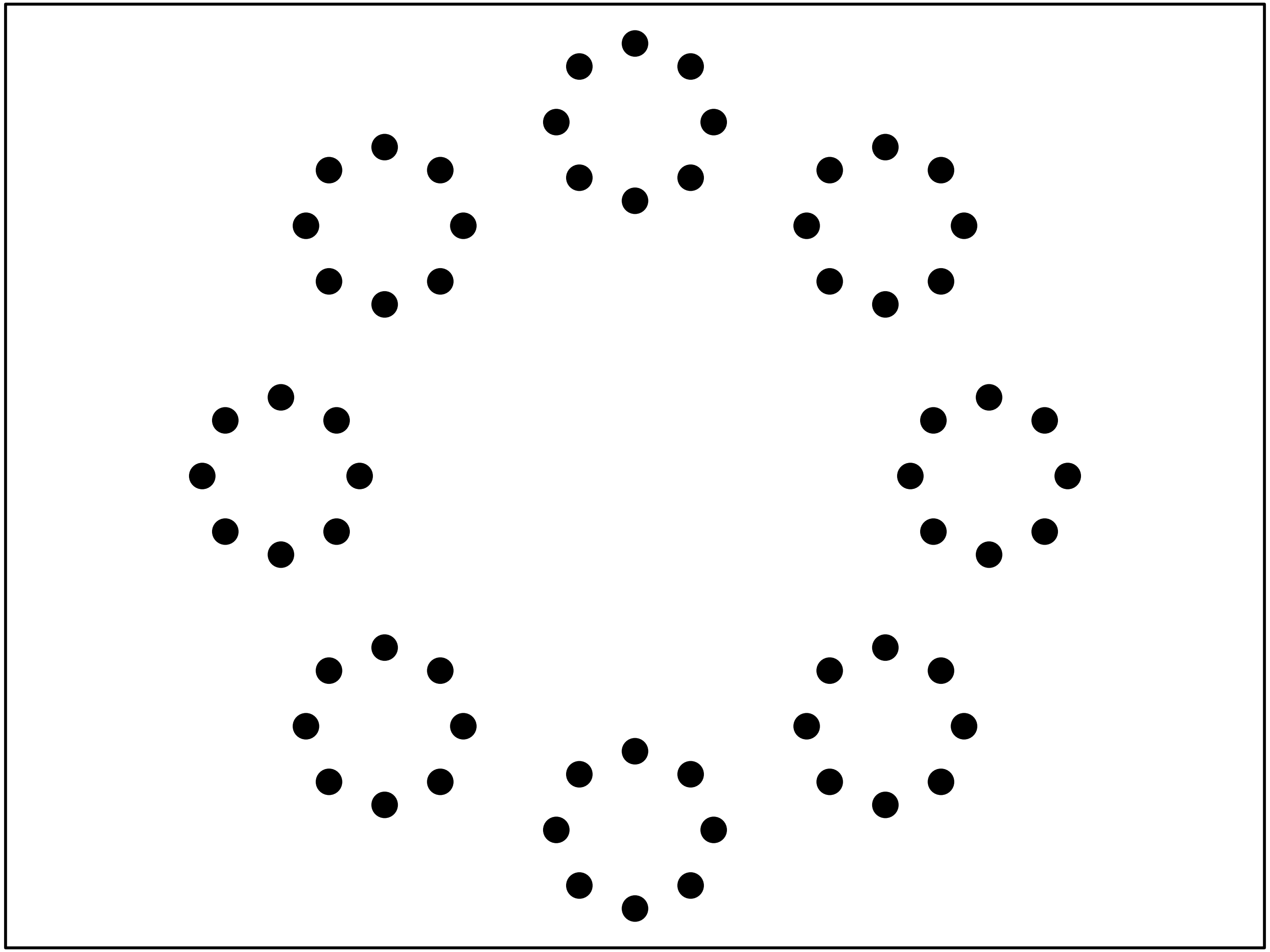
$$g(z) = f(\gamma + \rho z) \mod z^m$$

$\widetilde{O}(\frac{d}{m})$ polynomials of degree m

Approximation bound [New]

$$\|f(\gamma + \rho z) - g(z)\| \le 2^{-m}\|f\|$$

Hyperbolic approximation computation

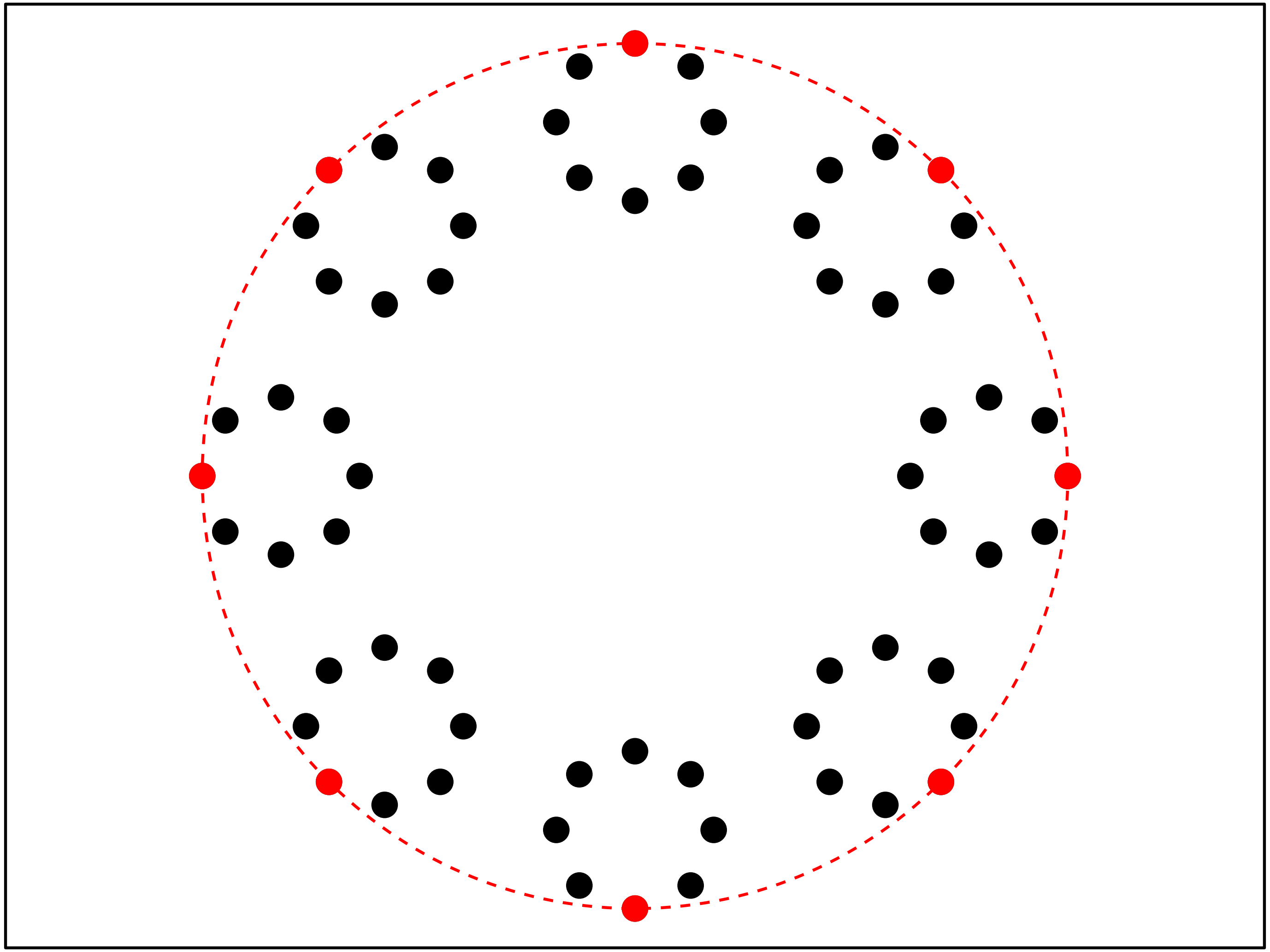


Do m times

SCALING d coefficients

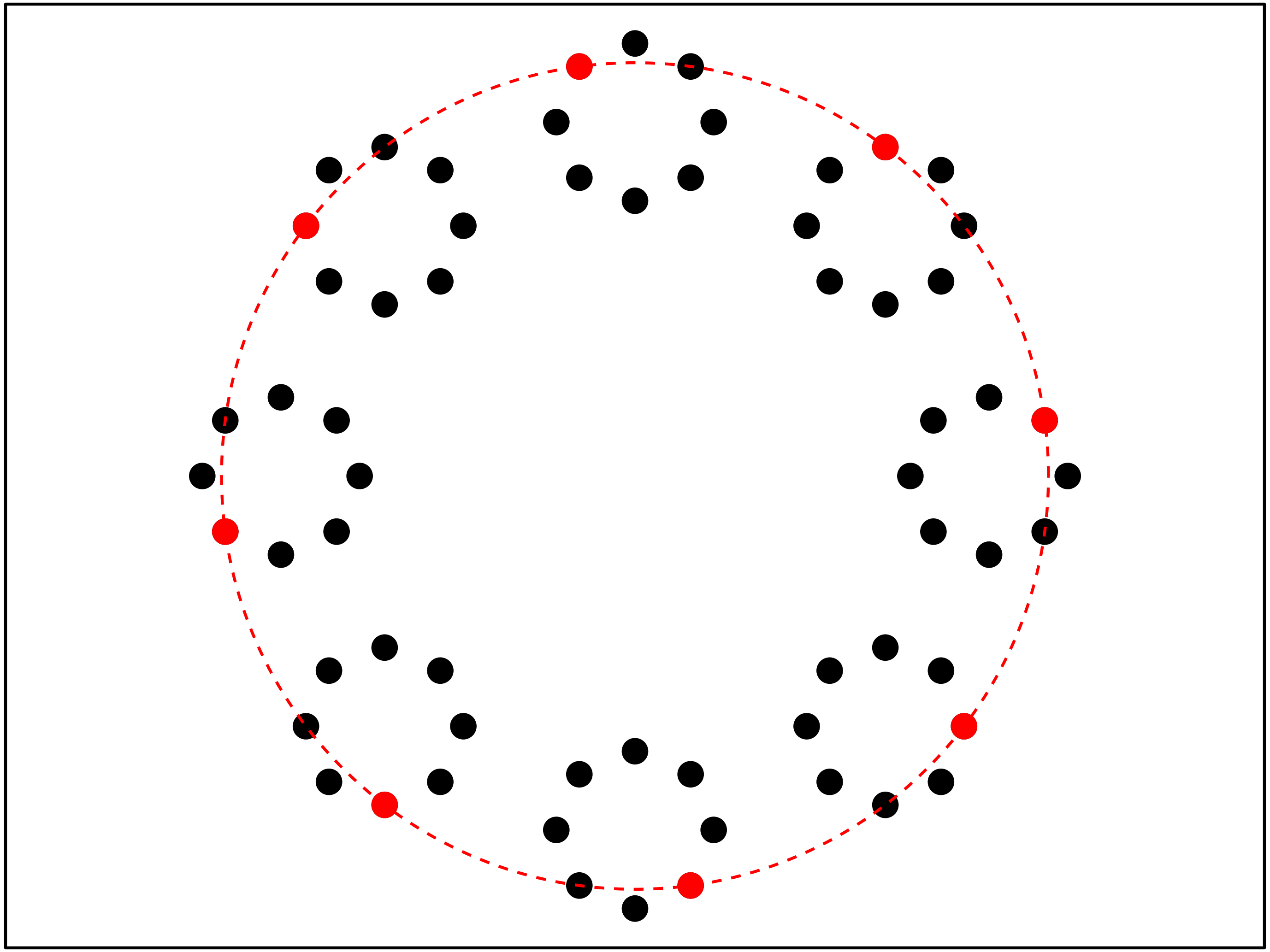
FFT on d/m roots of unity

Hyperbolic approximation computation



Do m times
  SCALING d coefficients
  FFT on d/m roots of unity

Hyperbolic approximation computation

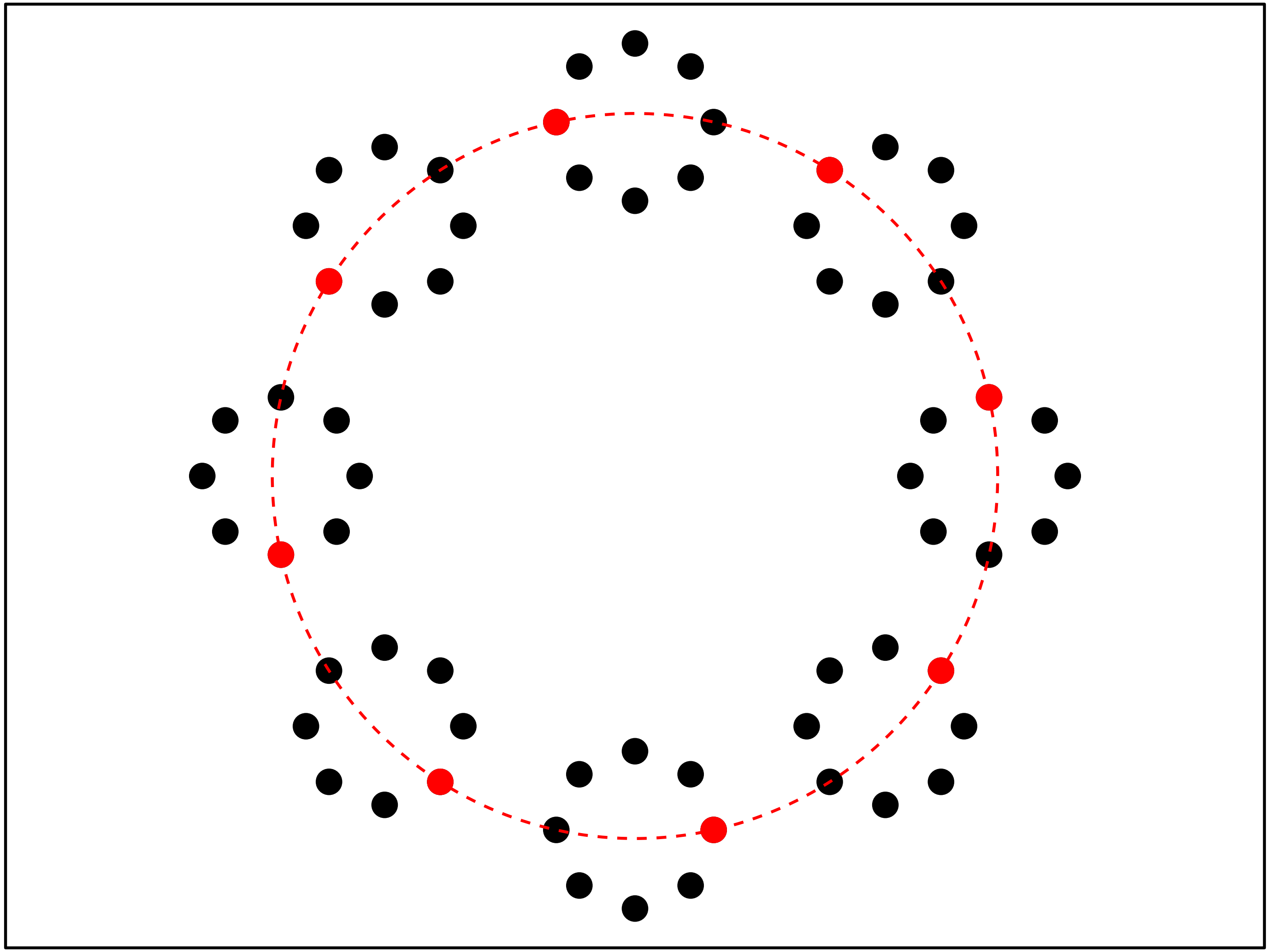


Do m times

SCALING d coefficients

FFT on d/m roots of unity

Hyperbolic approximation computation

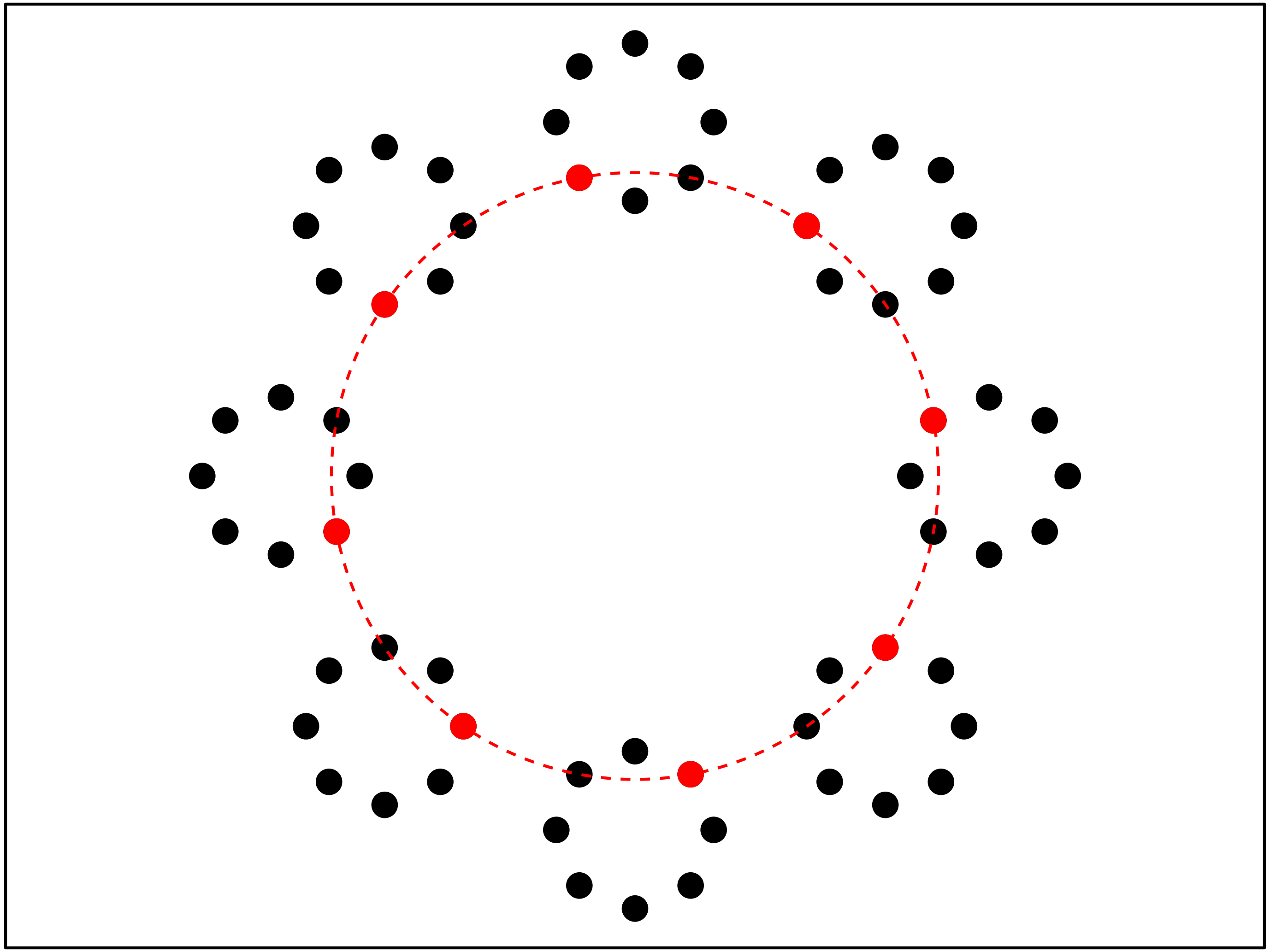


Do m times

$\quad$ SCALING d coefficients

$\quad$ FFT on d/m roots of unity

Hyperbolic approximation computation

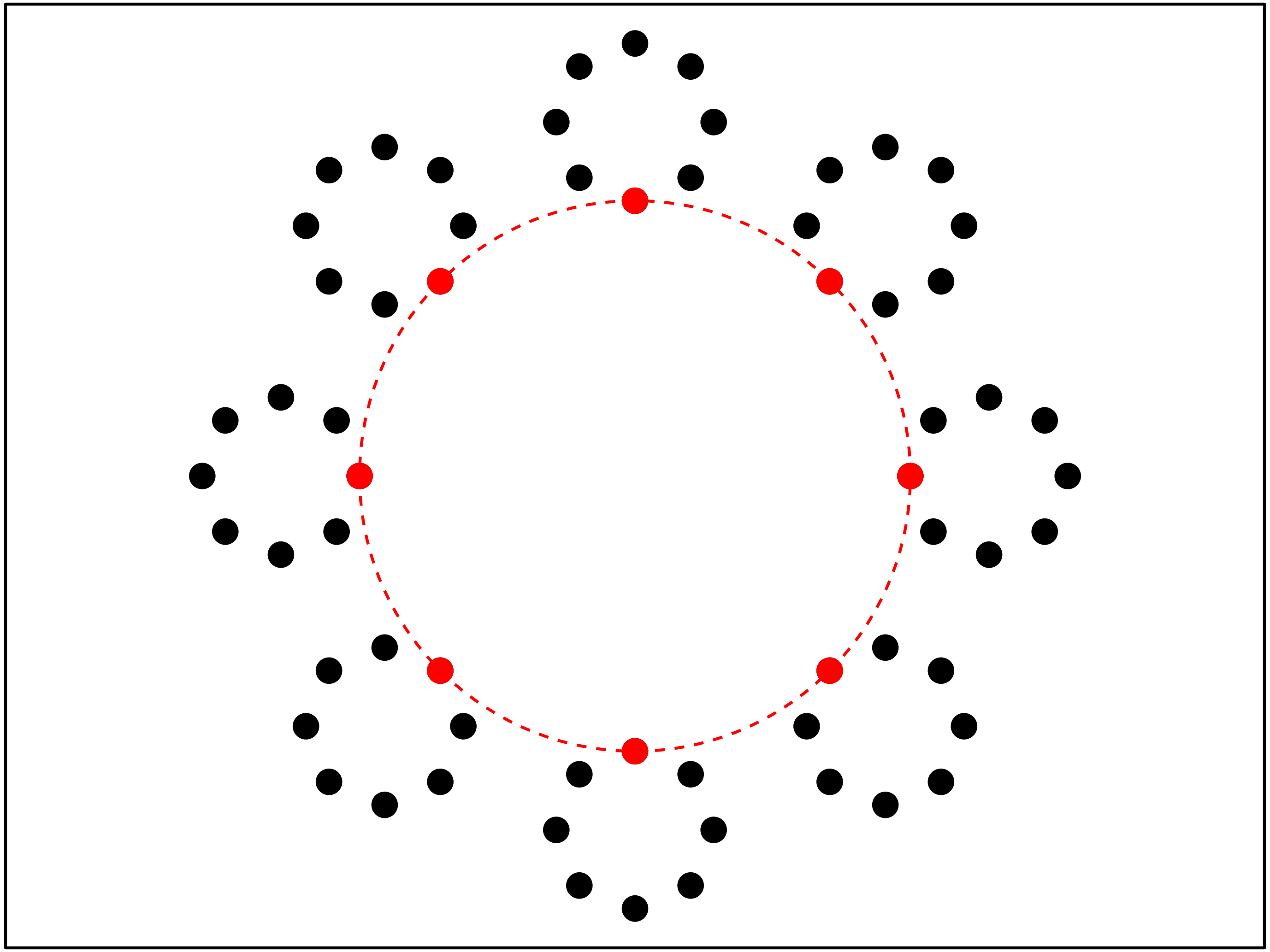


Do m times

SCALING d coefficients

FFT on d/m roots of unity

Hyperbolic approximation computation

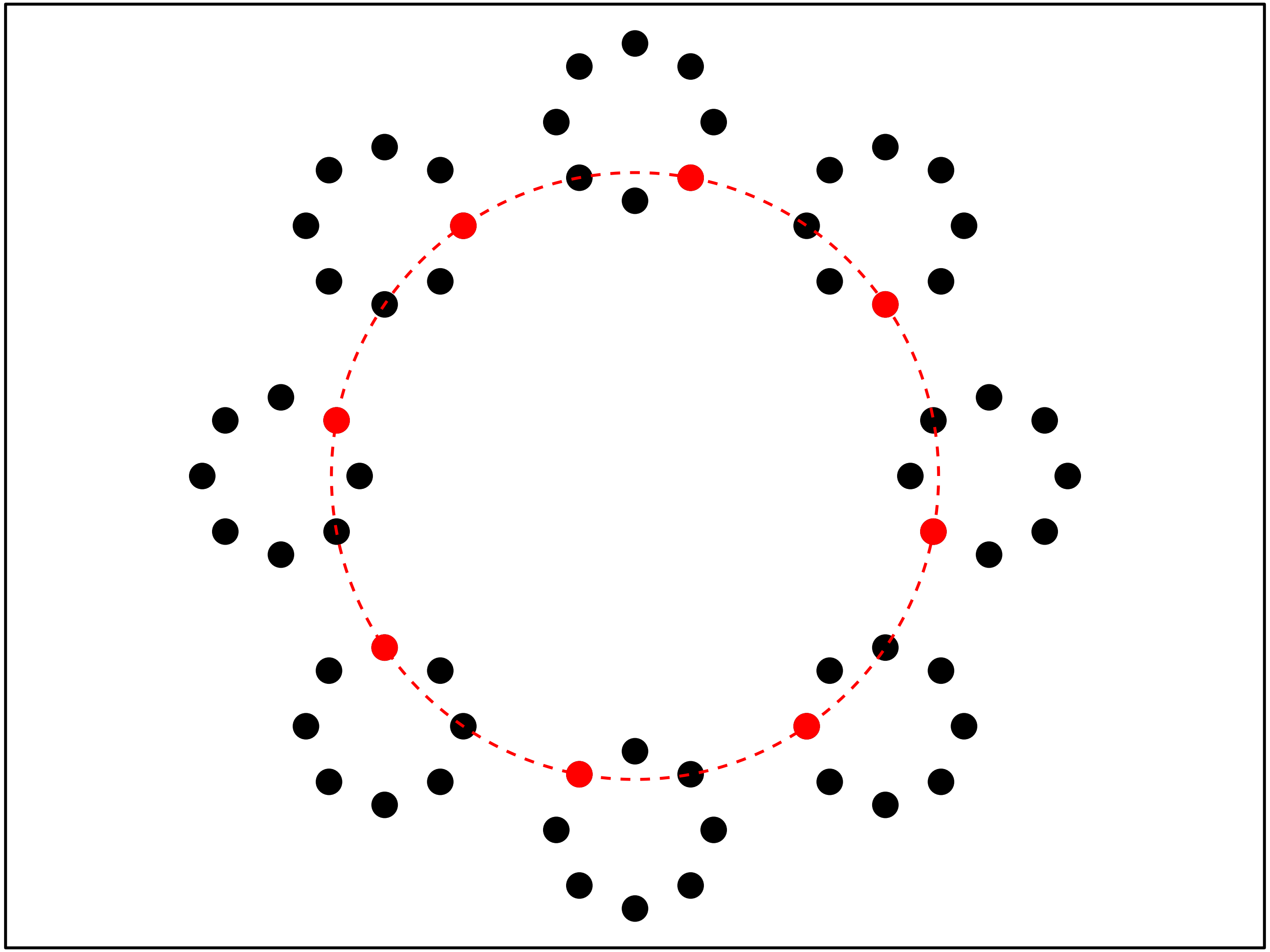


Do m times

SCALING d coefficients

FFT on d/m roots of unity

Hyperbolic approximation computation

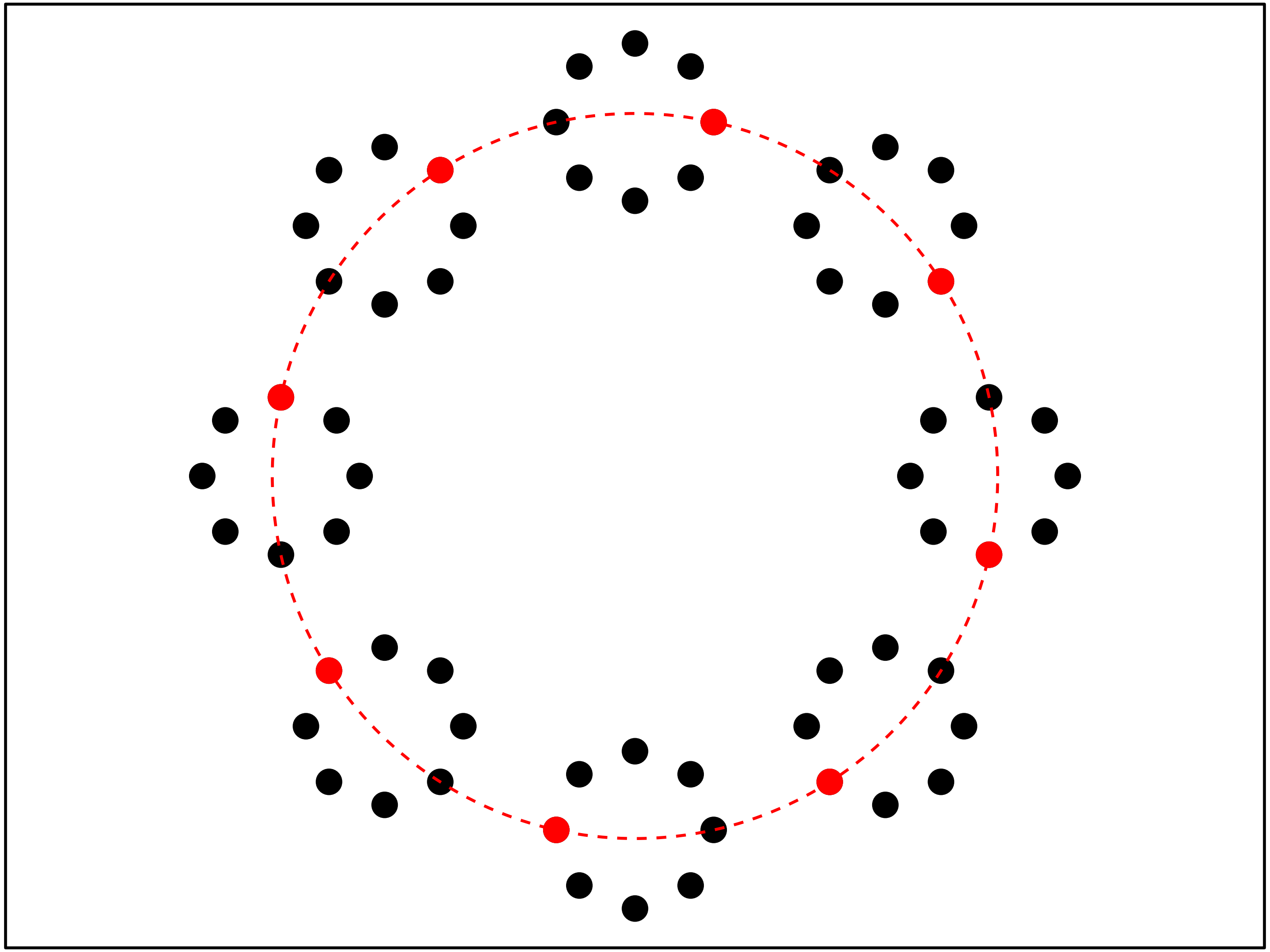


Do m times

SCALING d coefficients

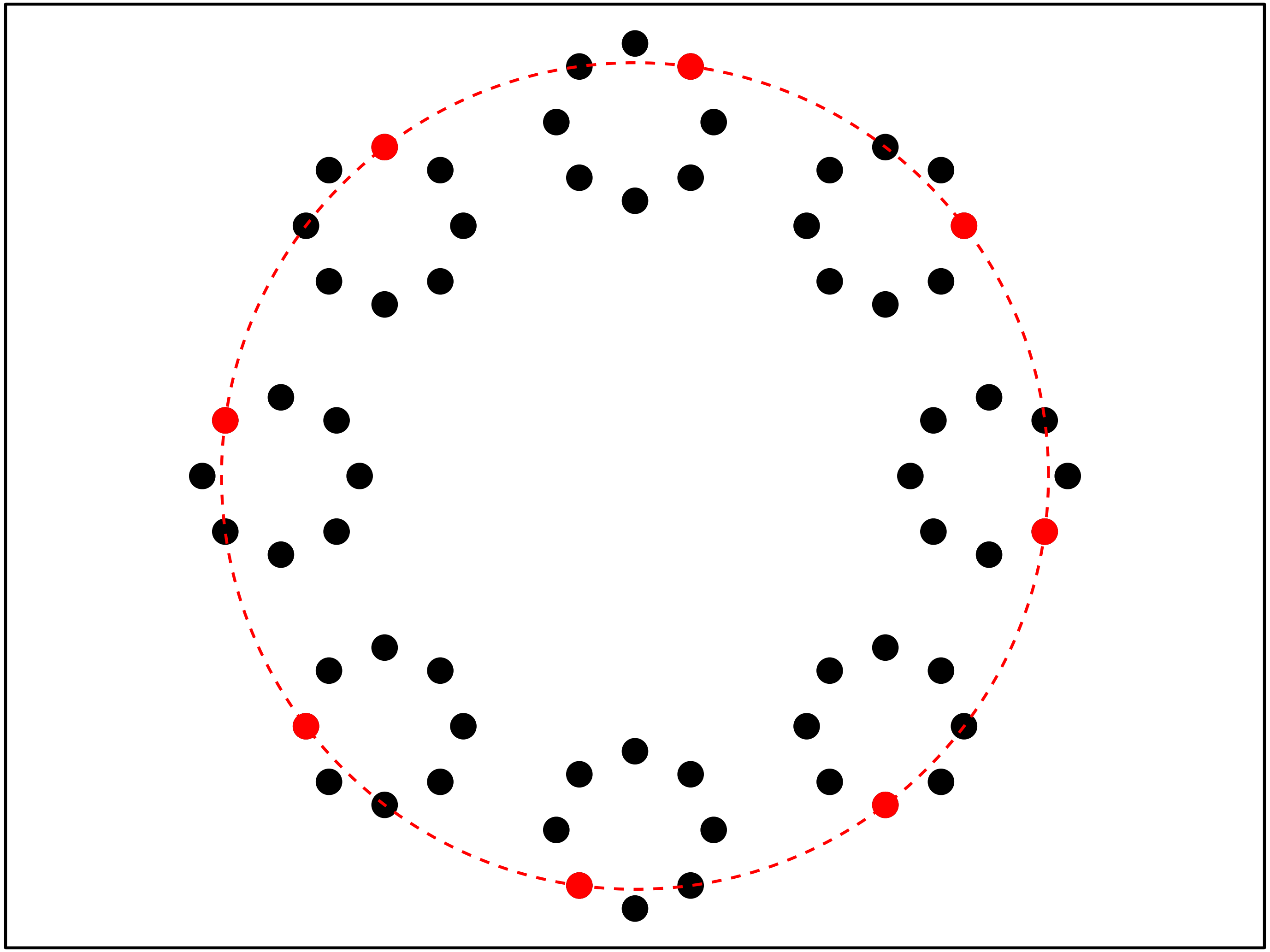
FFT on d/m roots of unity

Hyperbolic approximation computation



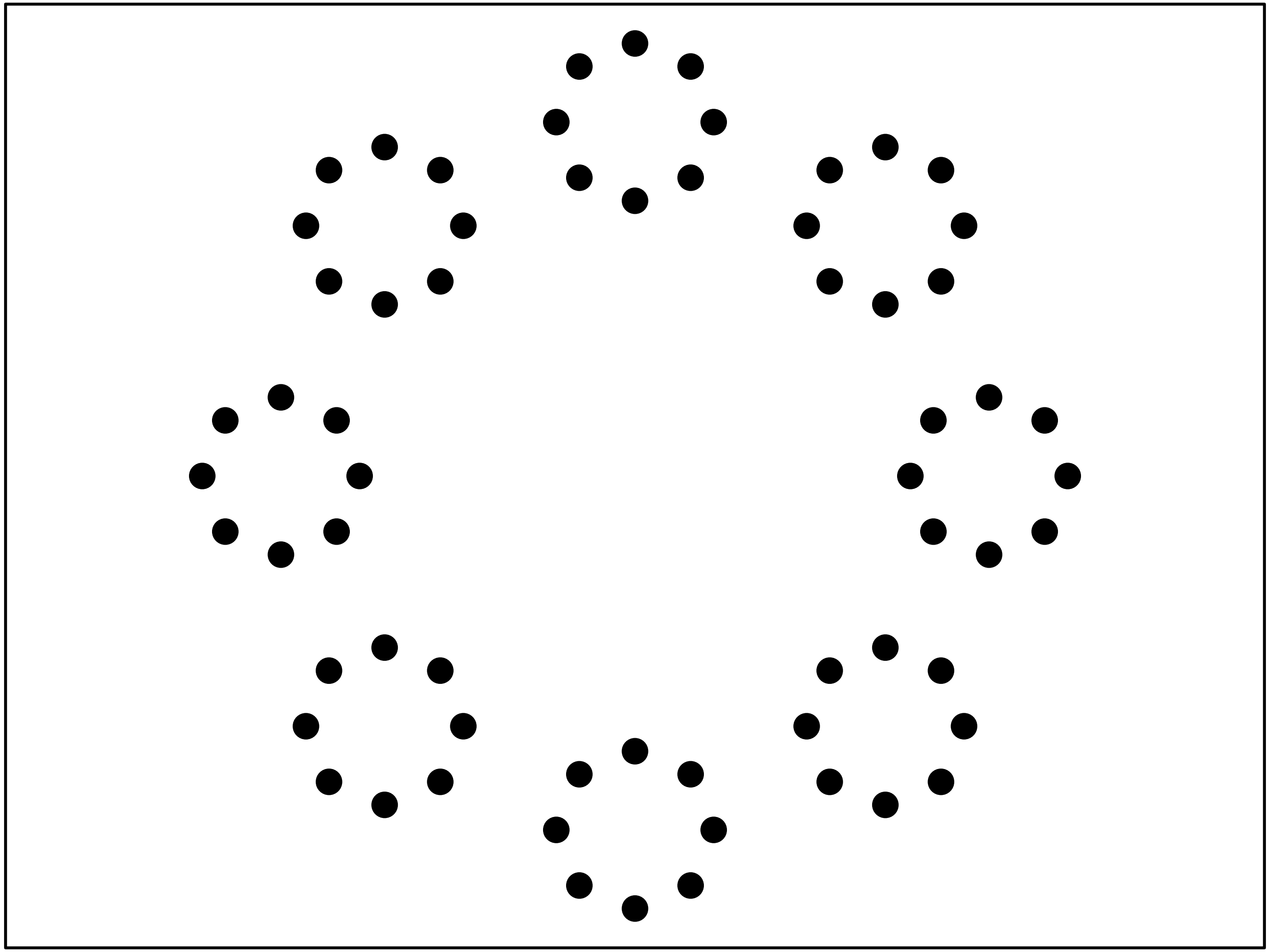
Do m times
SCALING d coefficients
FFT on d/m roots of unity

Hyperbolic approximation computation



Do m times
  $\textsc{Scaling}$ d coefficients
  FFT on d/m roots of unity

Hyperbolic approximation computation

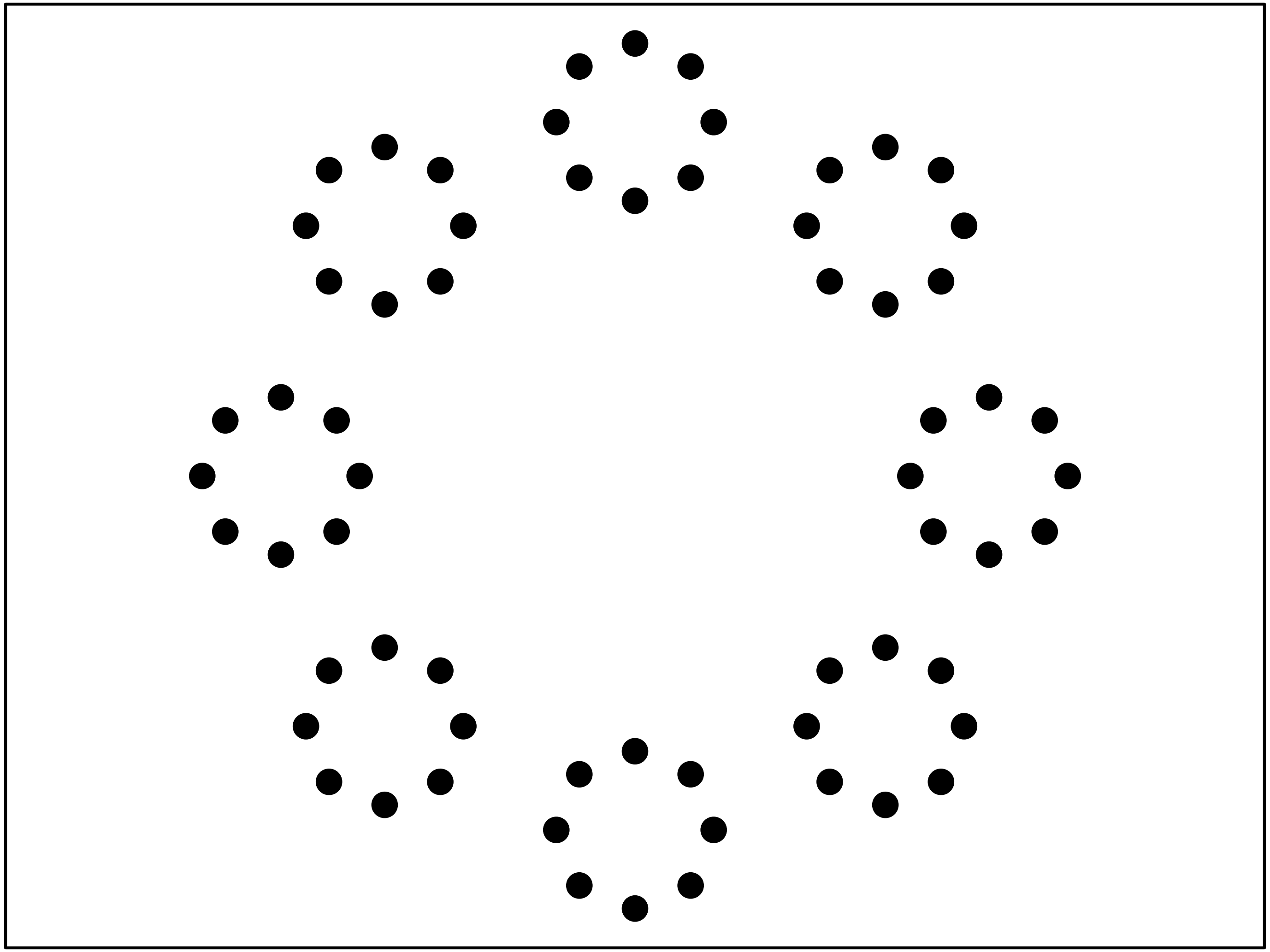


Do m times

$\quad$ SCALING d coefficients $\qquad \widetilde{O}(dm)$

$\quad$ FFT on d/m roots of unity $\qquad \widetilde{O}(d)$

Hyperbolic approximation computation

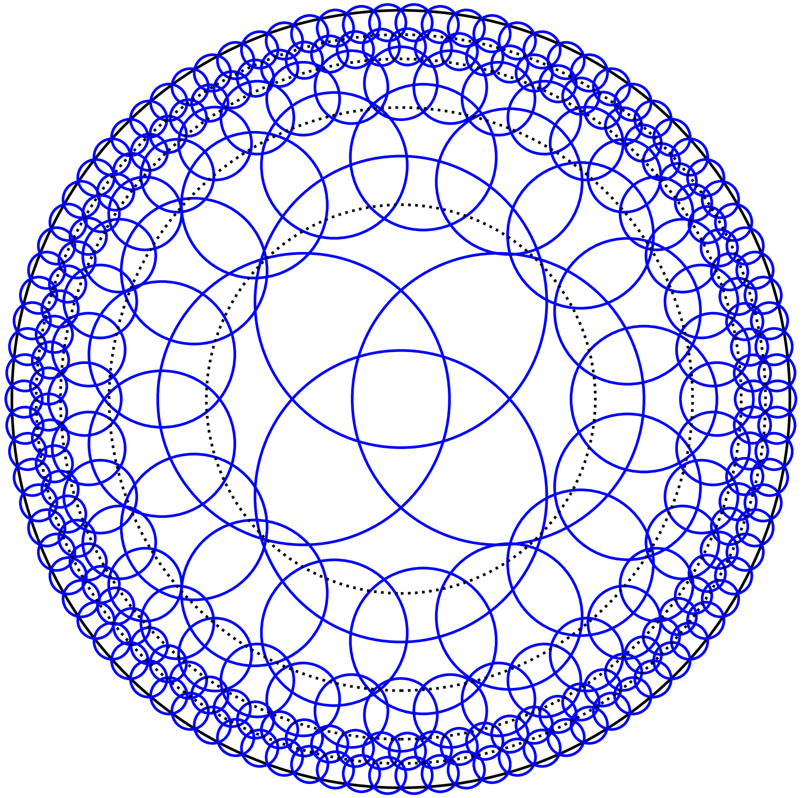


Do m times

  Scaling d coefficients $\widetilde{O}(d)$

  FFT on d/m roots of unity $\widetilde{O}(d)$

Multipoint evaluation in $\widetilde{O}(dm)$ [New]



INPUT:

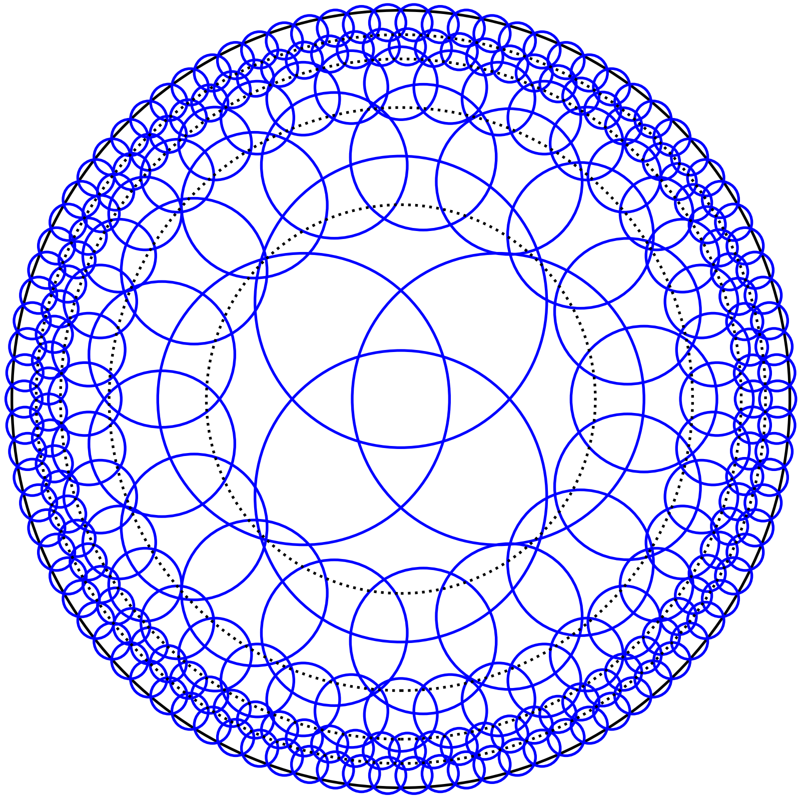
- f polynomial of degree d
- d points z_k
- precision m

OUTPUT:

- y_k such that $|y_k - f(z_k)| < 2^{-m}\|f\|$

Compute m-hyperbolic approximation of f	$\widetilde{O}(dm)$
For each pair of disk D and polynomial g:	$O(d/m)$
QUERY the n_k points in D	$\widetilde{O}(n_k)$
EVALUATE g on the n_k points	$\widetilde{O}(m(n_k + m))$

Root finding in $\widetilde{O}(d \log(\|f\|\kappa))$ [New]



INPUT:

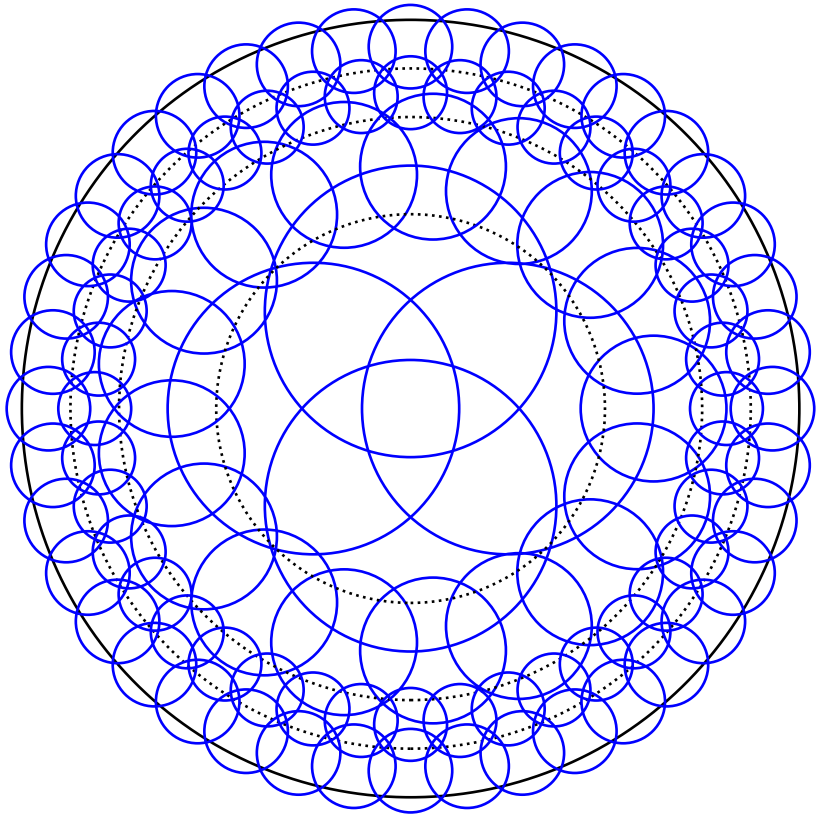
- f squarefree polynomial

OUTPUT:

- d root-isolating disks

Compute 1-hyperbolic approximation of f	$\widetilde{O}(d)$
For each polynomial g:	$O(d)$
APPROXIMATE roots of g	$\widetilde{O}(1)$
COMPUTE enclosing disks	$\widetilde{O}(1)$
Check if we have d isolating disks	$\widetilde{O}(d)$

Root finding in $\widetilde{O}(d\log(\|f\|\kappa))$ [New]



INPUT:

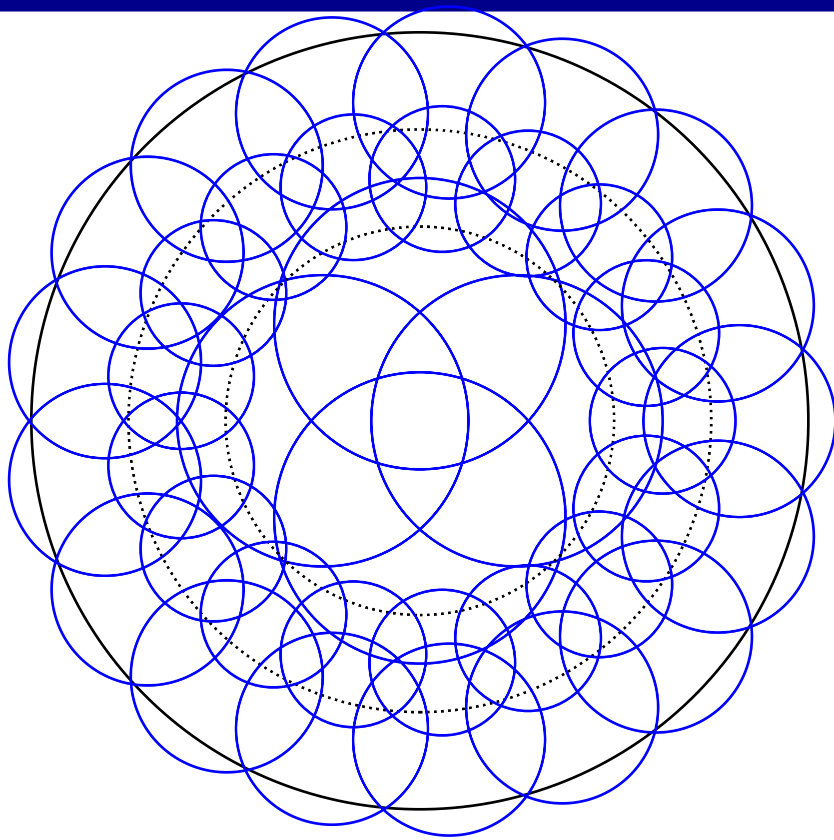
- f squarefree polynomial

OUTPUT:

- d root-isolating disks

Compute 2-hyperbolic approximation of f	$\widetilde{O}(d)$
For each polynomial g:	$O(d)$
APPROXIMATE roots of g	$\widetilde{O}(1)$
COMPUTE enclosing disks	$\widetilde{O}(1)$
Check if we have d isolating disks	$\widetilde{O}(d)$

Root finding in $\widetilde{O}(d\log(\|f\|\kappa))$ [New]



INPUT:

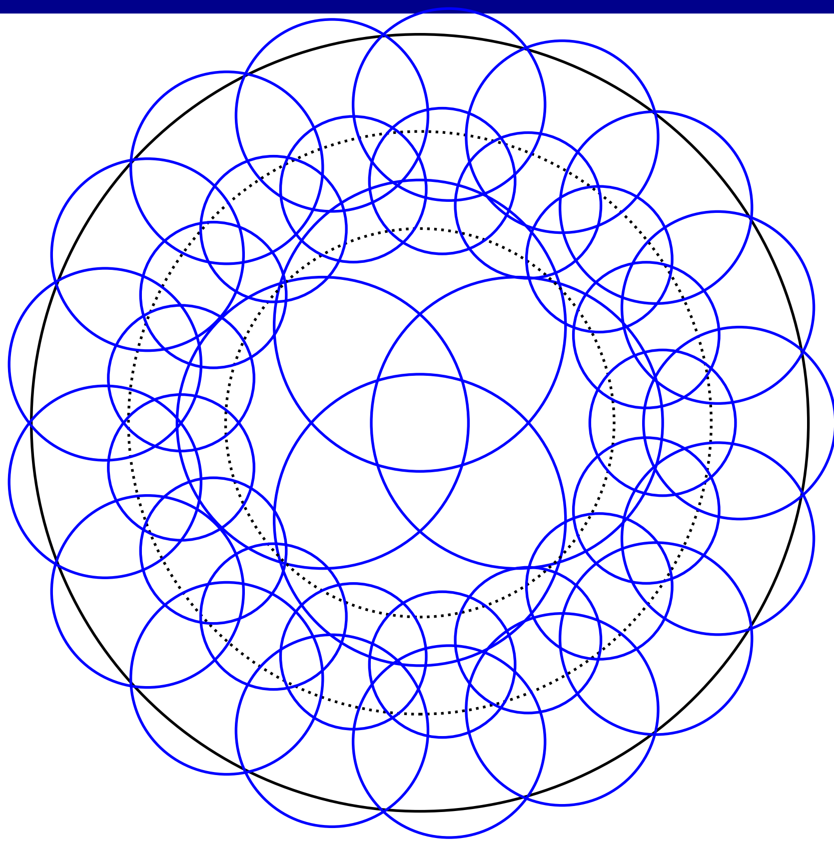
- f squarefree polynomial

OUTPUT:

- d root-isolating disks

Compute m-hyperbolic approximation of f	$\widetilde{O}(dm)$
For each polynomial g:	$O(d/m)$
APPROXIMATE roots of g	$\widetilde{O}(m^2)$
COMPUTE enclosing disks	$\widetilde{O}(m^2)$
Check if we have d isolating disks	$\widetilde{O}(dm)$

Root finding in $\widetilde{O}(d \log(\|f\|\kappa))$ [New]



INPUT:

- f squarefree polynomial

OUTPUT:

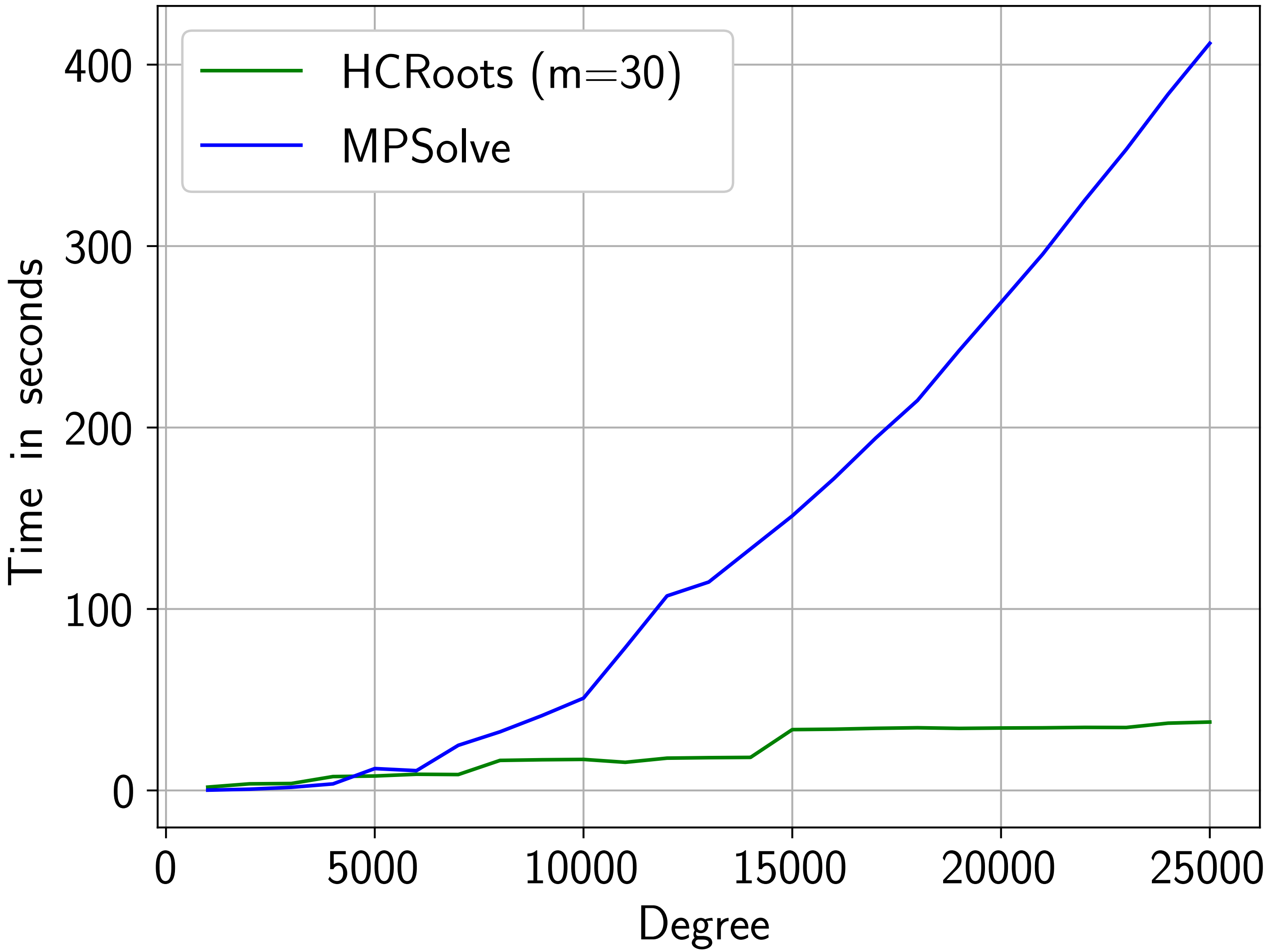
- d root-isolating disks

COMPLEXITY:

- $\widetilde{O}(dm)$ bit operations
- m in $\widetilde{O}\left(\log(\|f\|\kappa)\right)$

Compute m-hyperbolic approximation of f	$\widetilde{O}(dm)$
For each polynomial g:	$O(d/m)$
APPROXIMATE roots of g	$\widetilde{O}(m^2)$
COMPUTE enclosing disks	$\widetilde{O}(m^2)$
Check if we have d isolating disks	$\widetilde{O}(dm)$

Root finding in $\widetilde{O}(d \log(\|f\|\kappa))$ [New]

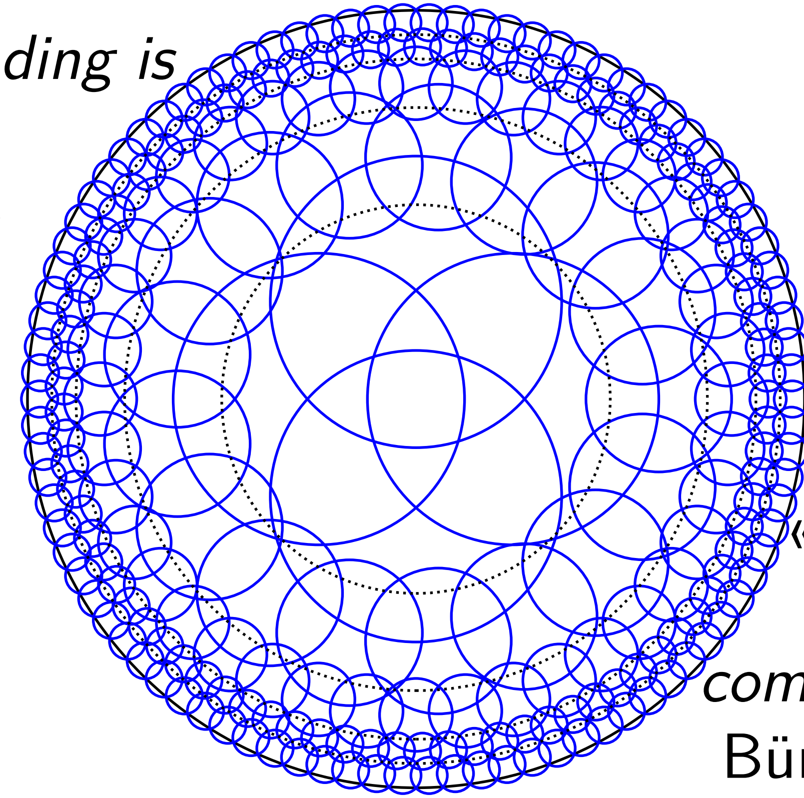


Root finding with small source code

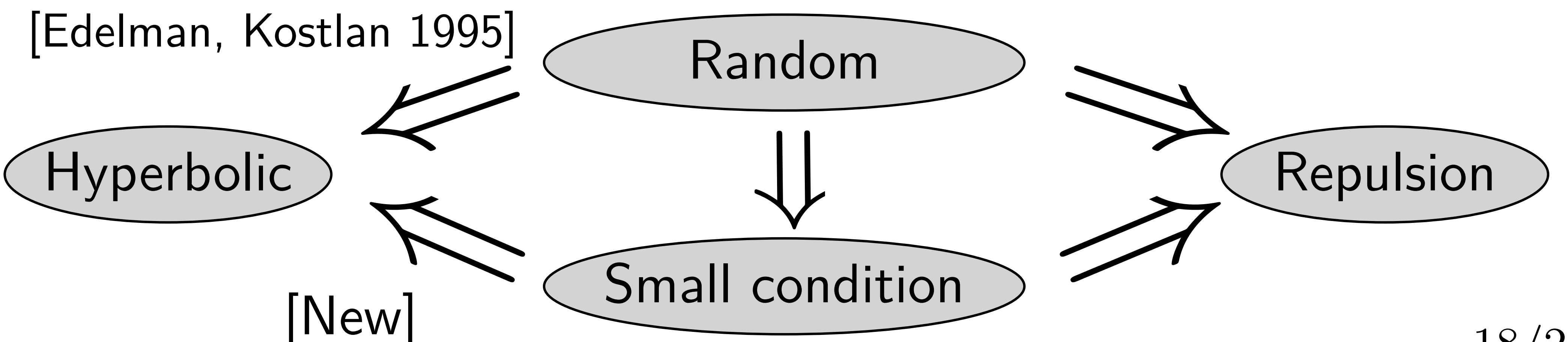
```
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 2 of the License, or
# (at your option) any later version.

import numpy as np

# Compute the disks of a hyperbolic covering
def disks(d, m):
    N = np.math.ceil(np.log2(3*np.e*d/min(m-1,d)))
    r = 1 - 1/2**(np.arange(N+1))
    r[-1] = 1
    gamma = 1/2*(r[1:] + r[:-1])
    rho = 3/4*(r[1:] - r[:-1])
    K = np.ceil(3*np.pi*r[1:]/(np.sqrt(5)*rho)).astype(int)
    K[0] = 4
    return gamma, rho, K

# Compute the m-hyperbolic approximation
def hyperbolic_approximation(coeffs, m=30):
    d = coeffs.shape[-1]
    shape = coeffs.shape[:-1]
    gamma, rho, K = disks(d, m)
    N = gamma.size
    Kmax = ((d-1)//K.max()+1)*K.max()
    r = rho/gamma
    D = np.arange(d)
    G = np.zeros(shape + (N, Kmax, m), dtype='complex128')
    P = gamma[:, np.newaxis]**D * coeffs[..., np.newaxis, :]
    G[...,0] = np.fft.fft(P, Kmax)
    for i in range(m-1):
        P *= (D-i)/(i+1) * r[:, np.newaxis]
        G[..., i+1] = np.fft.fft(P[...,i+1:], Kmax)
    return G, gamma, rho, K

# Solve polynomials of small degree
def solve_small(p, m=30, guarantee=True, e=0):
    result = [np.empty(0)]*p.shape[0]
    abs_p = np.abs(p)
    nosol = abs_p[:,0] > abs_p[:,1:].sum(axis=-1)
    unksol = ~nosol
    sols = list(map(np.polynomial.polynomial.polyroots, p[unksol]))
    for i,j in enumerate(np.flatnonzero(unksol)):
        result[j] = sols[i][np.abs(sols[i])<=1]
    if guarantee:
        validate(result, p, e)
    return result

# Guarantee that there is a unique solution nearby
def validate(sols, p, e):
    nonempty = [i for i,x in enumerate(sols) if x.size>0]
    p0 = p[nonempty]
    p1 = np.polynomial.polynomial.polyder(p0, axis=-1)
    p2 = np.polynomial.polynomial.polyder(p1, axis=-1)
    s = np.linalg.norm(p2, 1, axis=-1)
    for i, j in enumerate(nonempty):
        q = 10*s[i]*(np.abs(np.polynomial.polynomial.polyval(sols[j], p0[i]))+e)/\
                    (np.abs(np.polynomial.polynomial.polyval(sols[j], p1[i]))-e)**2
        sols[j] = sols[j][q <= 1]


# Solve using truncated polynomials
def solve_piecewise(G, gamma, rho, K, m=30, rtol=8, guarantee=True, e=0):
    result = np.array([],dtype='complex128')
    Kmax = G.shape[1]
    for p, g, r, Kn in zip(G,gamma,rho,K):
        step = (Kmax-1)//(Kn-1)  # step * (Kn-1) < Kmax
        w = np.exp(-2j*np.pi*np.arange(0,Kmax,step)/Kmax)
        sols = solve_small(p[::step], m, guarantee, e)
        for i in range((Kmax-1)//step + 1):
            sols[i] = g*w[i] + r*sols[i]
            result = np.append(result, sols[i])
    rounded = np.round(result, decimals=-int(np.log10(rtol)))
    _, ind = np.unique(rounded, return_index=True)
    return result[ind]

# truncate and solve a polynomial over the complex
def solve(p, m=30, rtol=None, guarantee=True):
    rtol =  max(3*2**(-m), 2**-35) if rtol is None else rtol
    dtype = p.dtype if hasattr(p, 'dtype') else 'complex128'
    p = np.trim_zeros(p, 'b')
    coeffs = np.zeros((2, len(p)), dtype=dtype)
    coeffs[0] = p
    coeffs[1] = coeffs[0,::-1]
    G, gamma, rho, K = hyperbolic_approximation(coeffs, m)
    e = 3*np.linalg.norm(coeffs[0], 1)*(m+2)/2**m
    sols = solve_piecewise(G[0], gamma, rho, K, m, rtol, guarantee, e)
    invsols = solve_piecewise(G[1], gamma, rho, K, m, rtol, guarantee, e)
    result = np.concatenate([sols , 1/invsols])
    rounded = np.round(result, decimals=-int(np.log10(rtol)))
    _, ind = np.unique(rounded, return_index=True)
    return result[ind]
```

Roots distribution

«*Polynomial rootfinding is an ill-conditioned problem in general*»
Trefethen and Bau



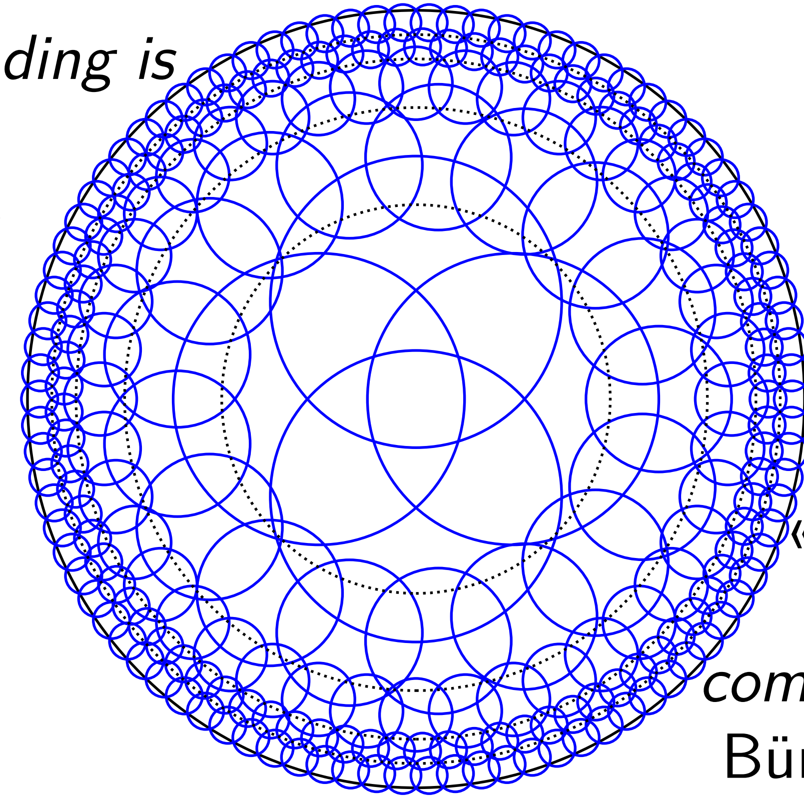
«*Typical polynomials are well-conditioned for the computation of their zeros*»
Bürgisser, Cucker, Cardozo



Roots distribution

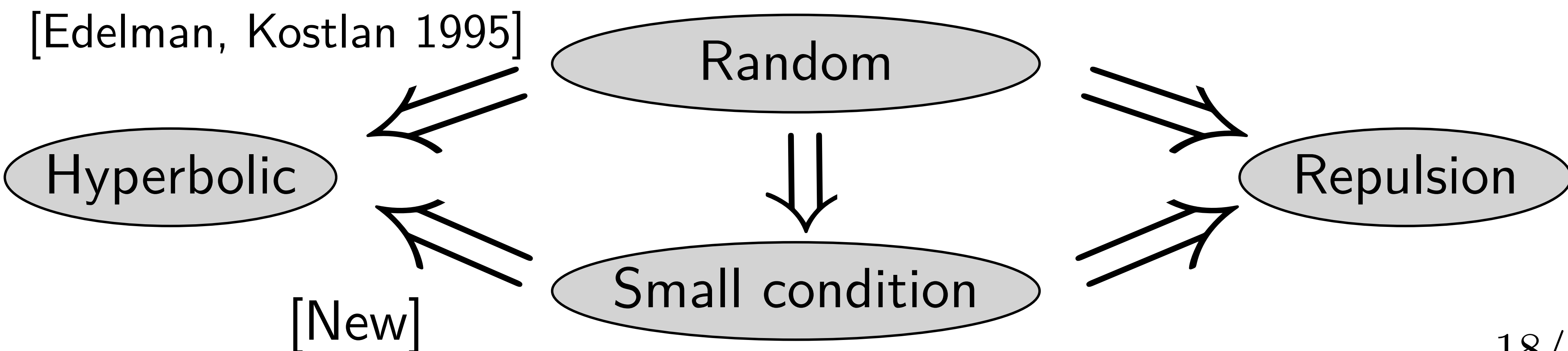
«*Polynomial rootfinding is an ill-conditioned problem in general*»
Trefethen and Bau

For uniform distribution of roots



For uniform distribution of coefficients

«*Typical polynomials are well-conditioned for the computation of their zeros*»
Bürgisser, Cucker, Cardozo



End of the story?

Ongoing work

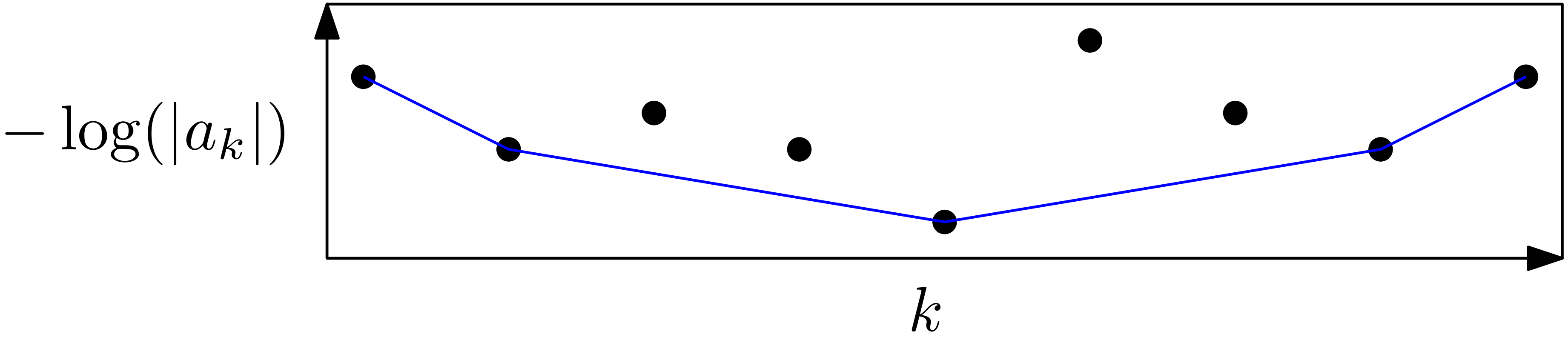
$$f^+(z) = |a_0| + \cdots + |a_d||z|^d$$

Adaptive approximation

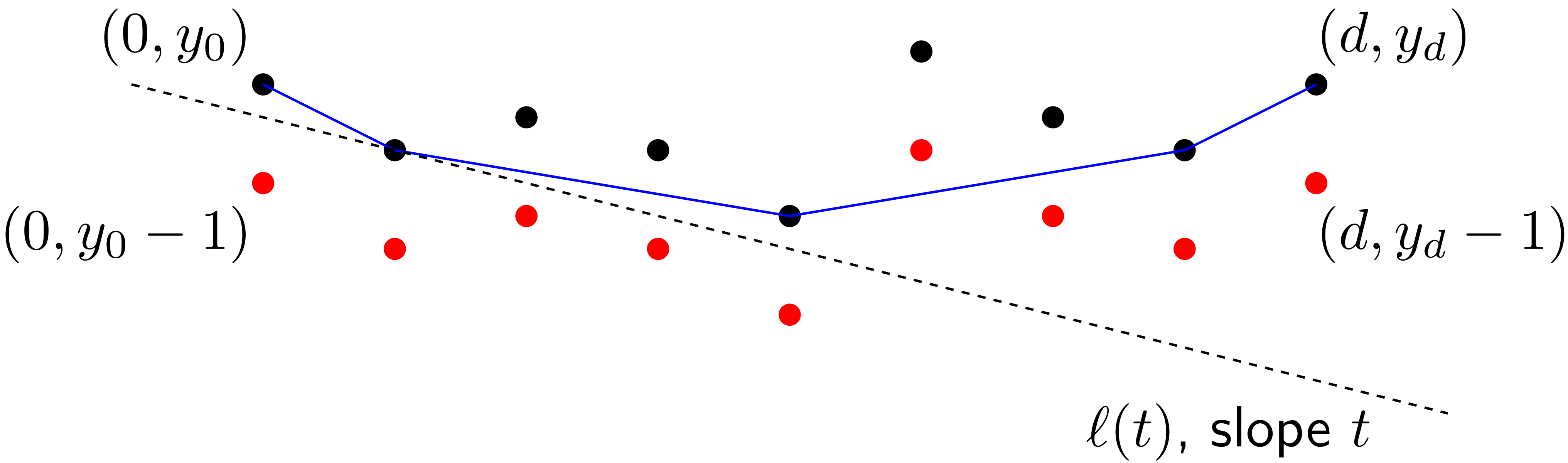
$$\|f(\gamma + \rho z) - g(z)\| \leq 2^{-m} f^+(z)$$

Based on Newton polygon

- Used in MPSolve for initial root module estimation
- Can be used for adaptive repartition of disks



Complexity: a geometric problem



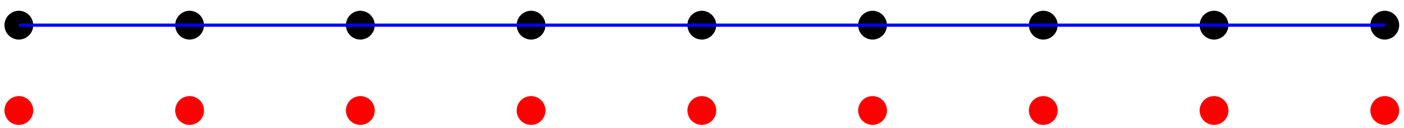
Open question

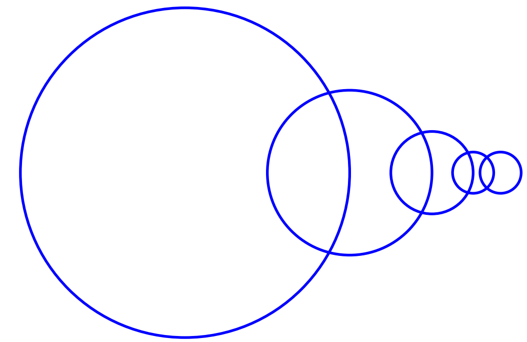
Let $n(t)$ be the number of red points below $\ell(t)$.

$$\int_{t=-1}^{1} n^2(t)\, dt = \widetilde{O}(d) \ ?$$

Complexity: a geometric problem

Case $|a_k| = 1$



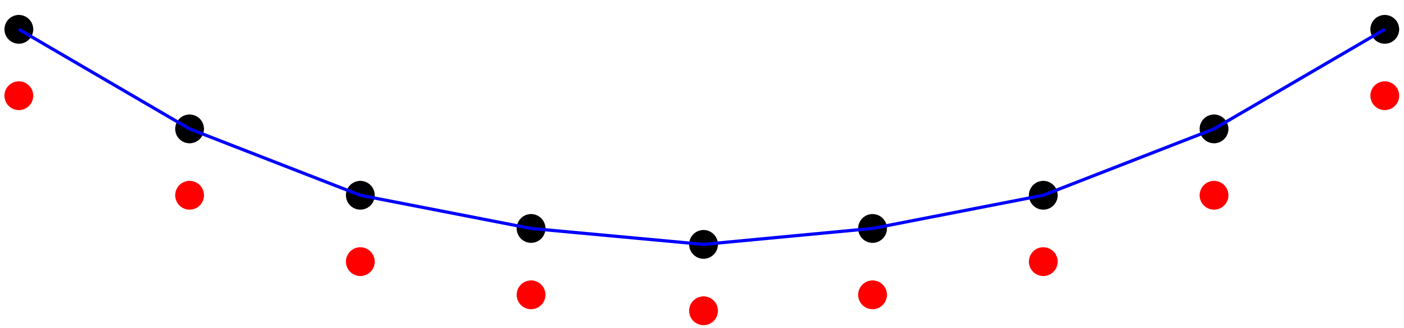


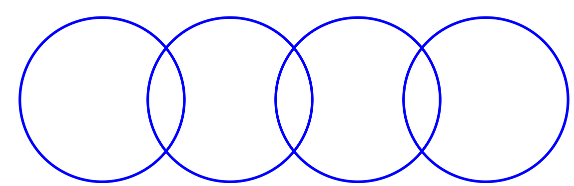
$$n(t) \leq \min(\tfrac{1}{t}, d) \Rightarrow \widetilde{O}(d)$$

artanh(γ) uniform

Case $|a_k| = \sqrt{\binom{d}{k}}$

[M. 2021]





$$n(t) = O(\sqrt{d}) \Rightarrow \widetilde{O}(d)$$

arctan(γ) uniform

Thank you!