(Multivariate) rootfinding via eigenvalues

Yuji Nakatsukasa
University of Oxford

Joint work with Vanni Noferini (Aalto) and Alex Townsend (Cornell)
and Maria Quintana Ponce (Aalto) and Nick Trefethen (Harvard)

Back to the roots seminar, KU Leuven

Bivariate rootfinding (zerofinding)

$f(x, y), g(x, y)$: bivariate functions $[-1, 1]^2 \to \mathbb{R}$

Problem: find pairs $(x_*, y_*) \in [-1, 1]^2$ s.t.

$$\begin{pmatrix} f(x_*, y_*) \\ g(x_*, y_*) \end{pmatrix} = 0$$

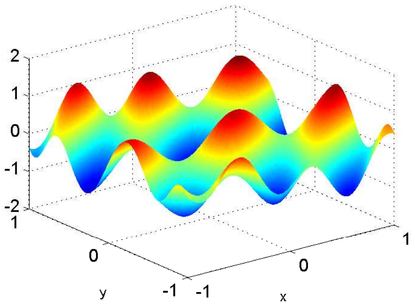
This talk (Part I): Chebfun2's `roots` [N.-Noferini-Townsend (15)]

- Speedup $O(n^6) \to\approx O(n^4)$, where n is polynomial degree
  - feasible degree: previously $n \approx 30$, now $n \approx 1000$
- Numerically stable solutions employing
  - Chebyshev interpolation via FFT
  - conditioning analysis $\Rightarrow$ local refinement for accuracy
  - stable eigensolver for polynomial eigenproblems

Bivariate rootfinding: applications

- Solving $f_x(x_*, y_*) = f_y(x_*, y_*) = 0$ gives critical points of f
  - Computing $\|f\|_\infty = \max_{x,y\in[-1,1]^2} |f(x,y)|$
  - Finding maximum/minimum values of f
- KKT conditions in optimization problems
  - e.g. ellipsoid distance, non-convex problems

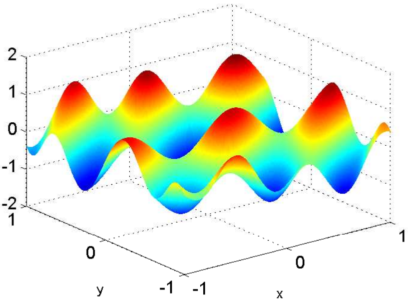
$f(x, y)$



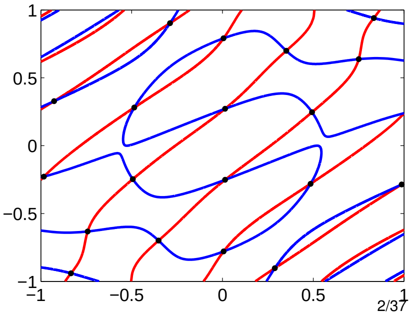
Bivariate rootfinding: applications

- Solving $f_x(x_*, y_*) = f_y(x_*, y_*) = 0$ gives critical points of f
  - Computing $\|f\|_\infty = \max_{x,y\in[-1,1]^2} |f(x,y)|$
  - Finding maximum/minimum values of f
- KKT conditions in optimization problems
  - e.g. ellipsoid distance, non-convex problems

$f(x,y)$



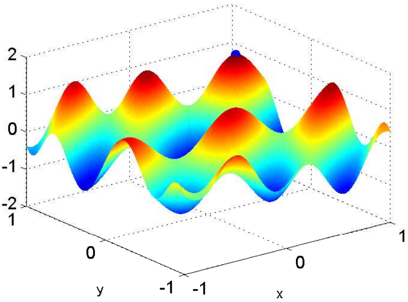
Curves $f_x(x,y) = 0, f_y(x,y) = 0$



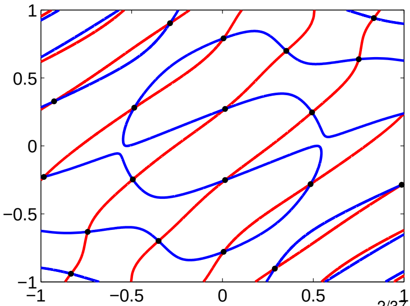
Bivariate rootfinding: applications

- Solving $f_x(x_*, y_*) = f_y(x_*, y_*) = 0$ gives critical points of f
  - Computing $\|f\|_\infty = \max_{x,y\in[-1,1]^2} |f(x,y)|$
  - Finding maximum/minimum values of f
- KKT conditions in optimization problems
  - e.g. ellipsoid distance, non-convex problems

$f(x,y)$



Curves $f_x(x,y) = 0$, $f_y(x,y) = 0$



Rootfinding $f(x_*) = 0$ on $[-1, 1]$: principles

1. Approximate $f(x)$ with a polynomial $p(x)$: –[J. Boyd (02)]

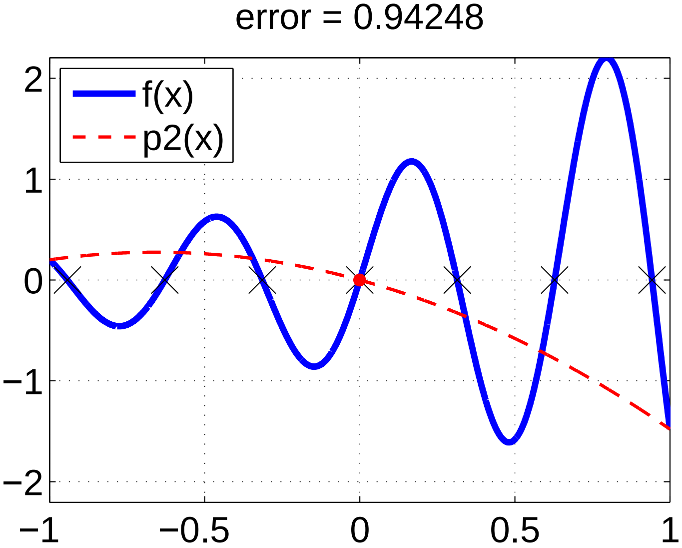
$$\|f(x) - p(x)\|_\infty = O(\epsilon)\|f(x)\|_\infty \quad \text{on} \quad [-1, 1]$$

2. Find roots of the polynomial $p(x)$ via generalized eigenvalues
– e.g. companion linearization for $p(x) = \sum_{i=0}^{n} a_i x^i$: $Yv = \lambda Xv$, where

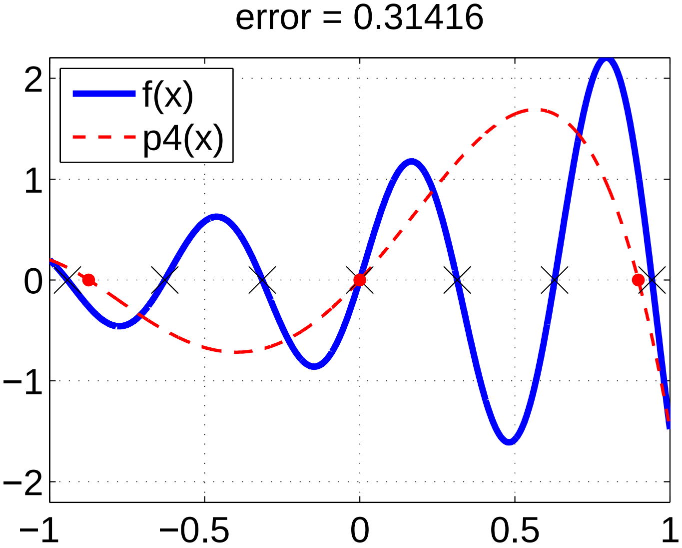
$$(X, Y) = \left(\begin{bmatrix} a_n & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix} \right)$$

- The computed solutions $\widehat{x_*}$ are backward stable: $f(\widehat{x_*}) = O(\epsilon) \Leftrightarrow \tilde{f}(\widehat{x_*}) = 0$ for $\|f - \tilde{f}\|_\infty = O(\epsilon)$
- Applications: local extrema, comuting $|f(x)|$, sign$(f(x))$, $\|f\|_1 = \int_{-1}^{1} |f(x)| dx$, ...

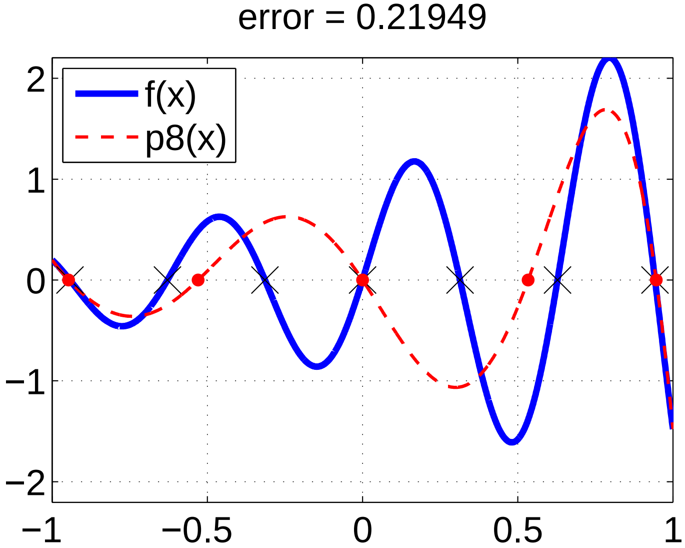
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



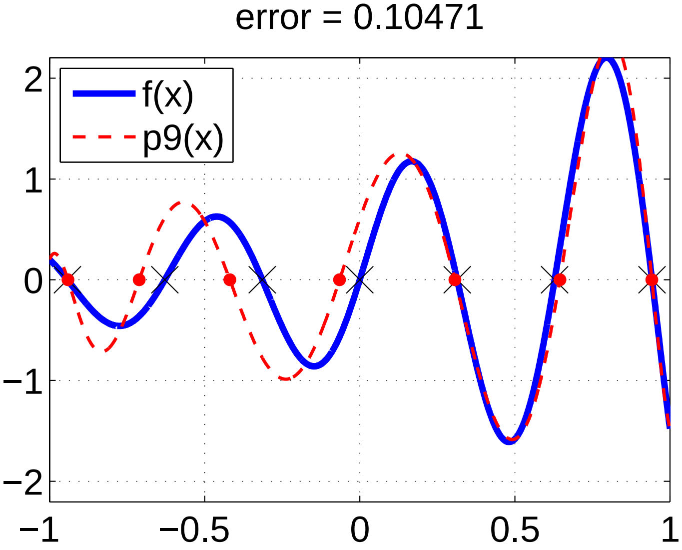
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



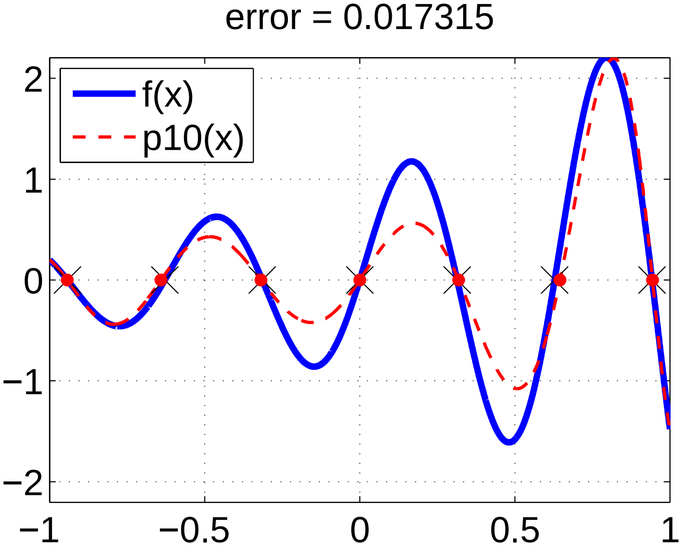
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



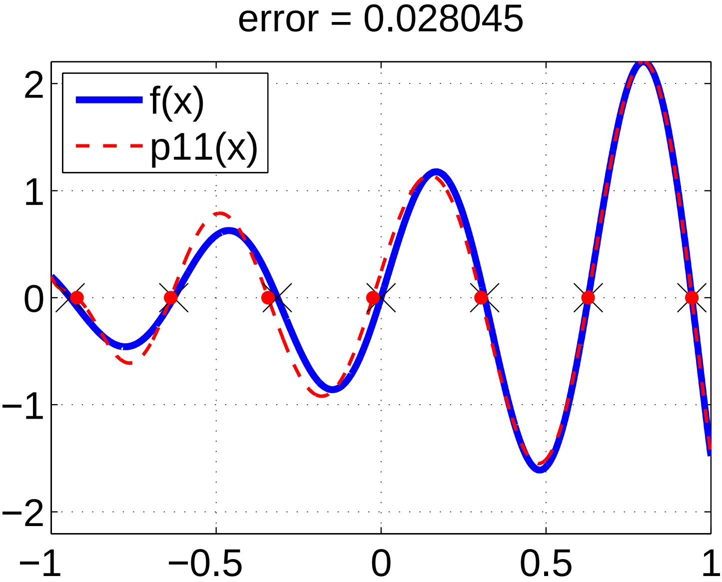
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



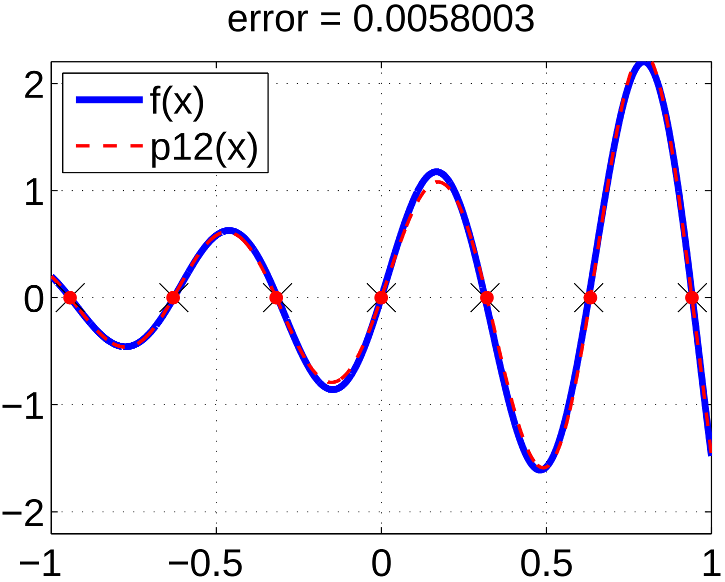
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



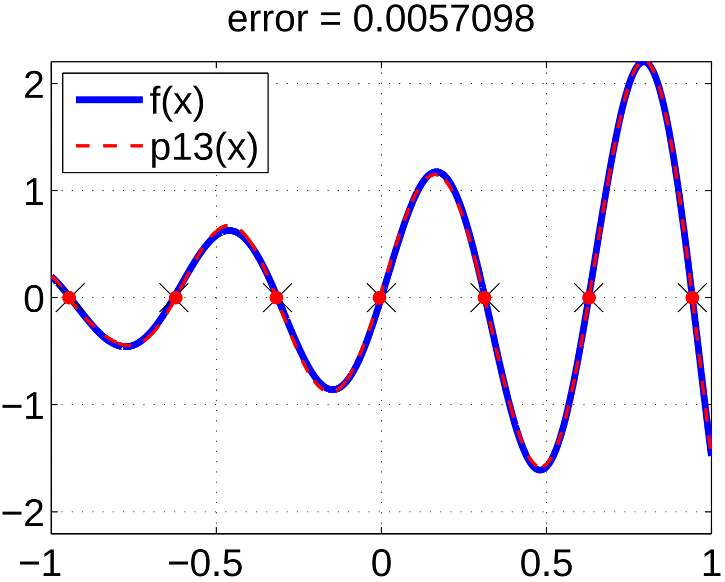
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



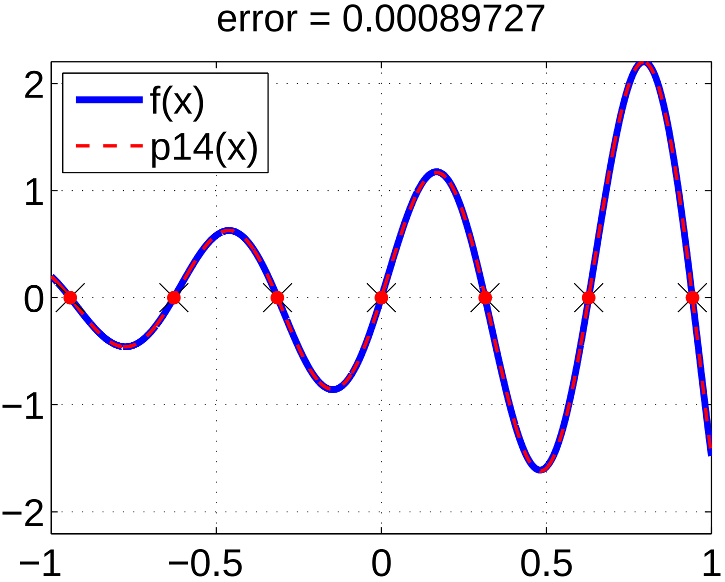
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



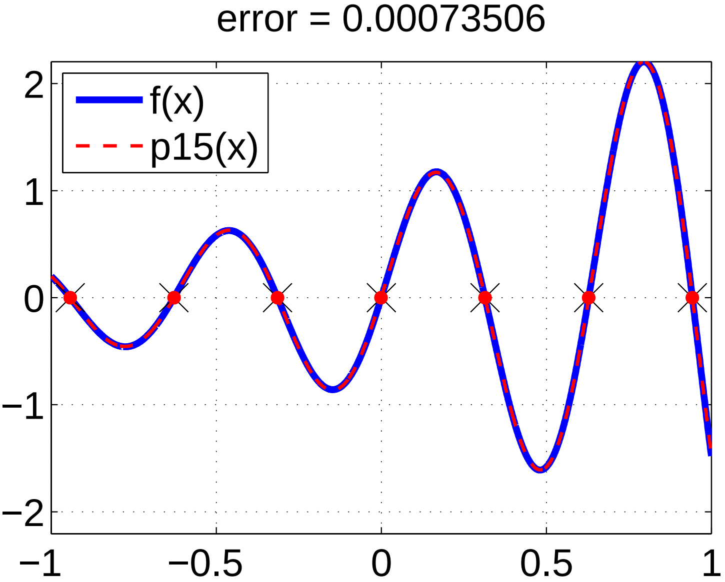
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



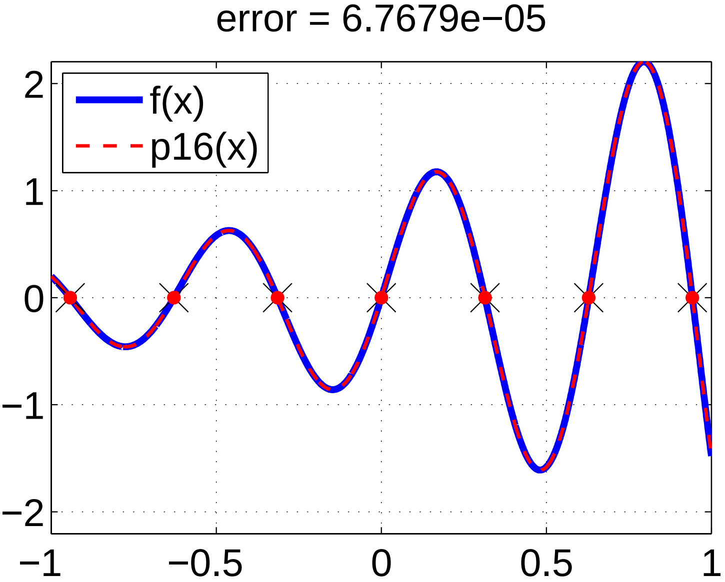
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



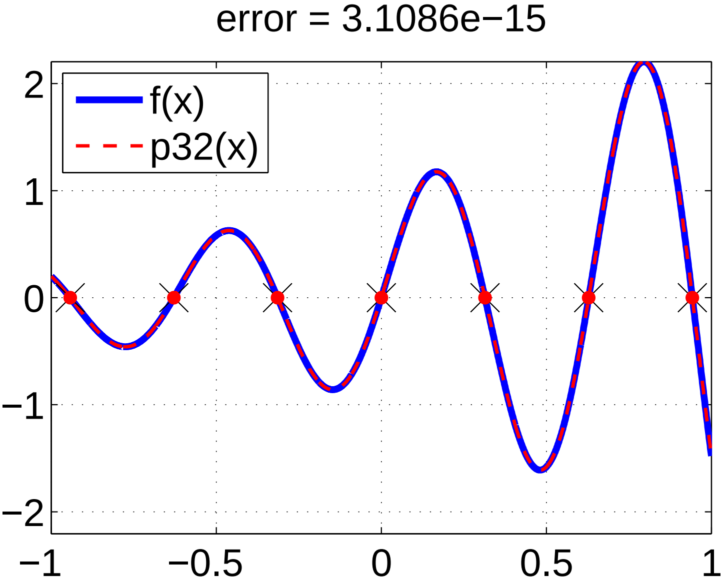
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



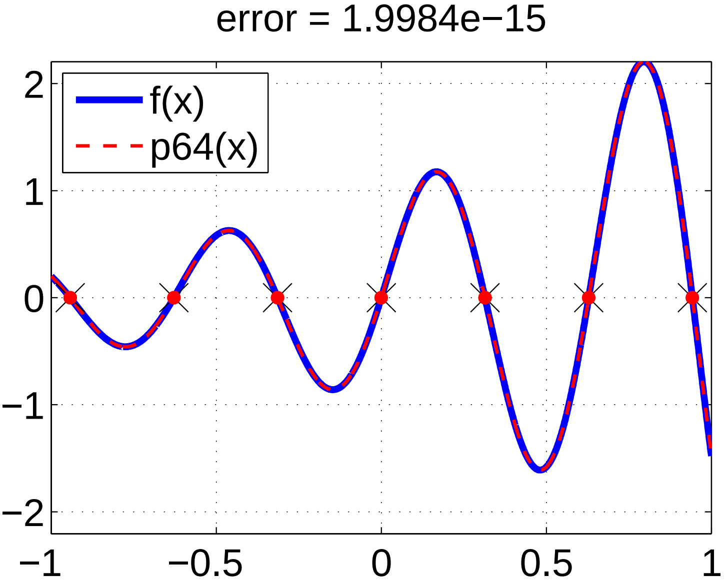
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



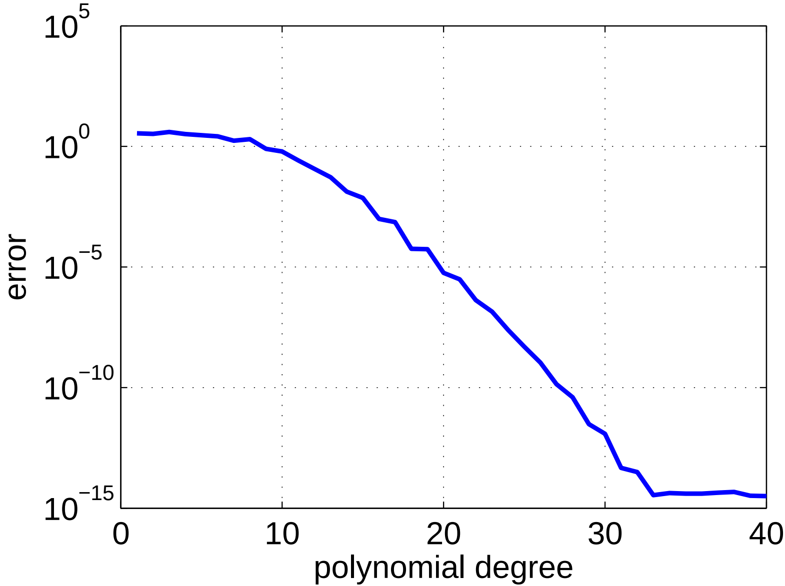
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



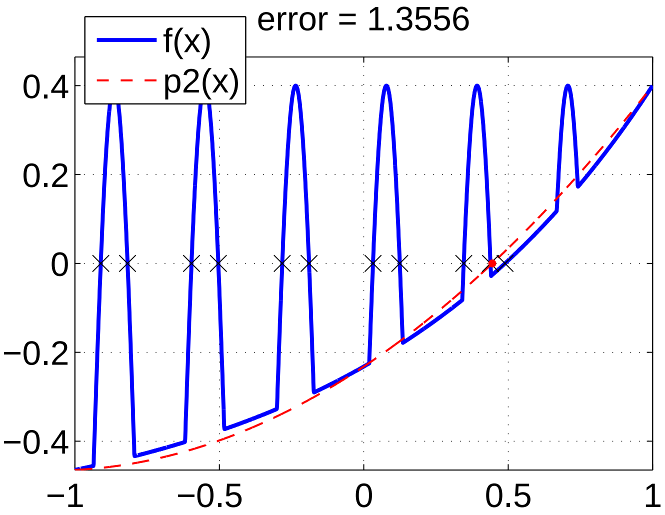
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



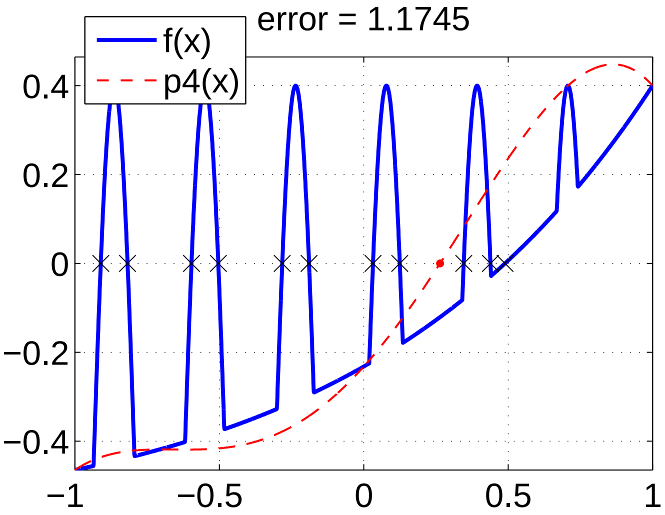
Rootfinding $f(x) = 0$ on $[-1, 1]$: example



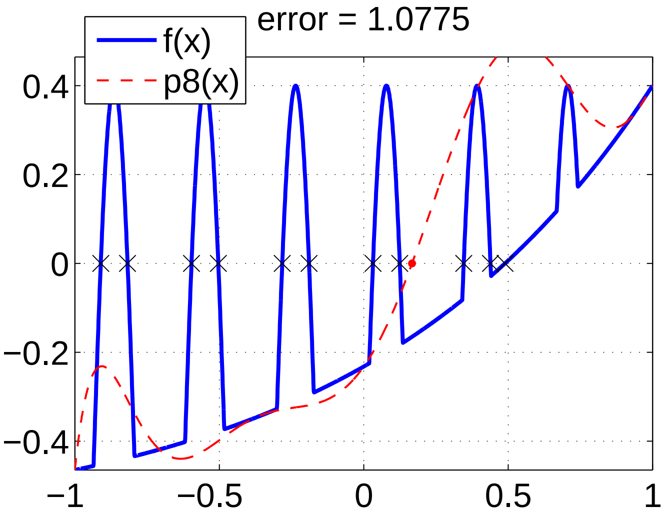
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



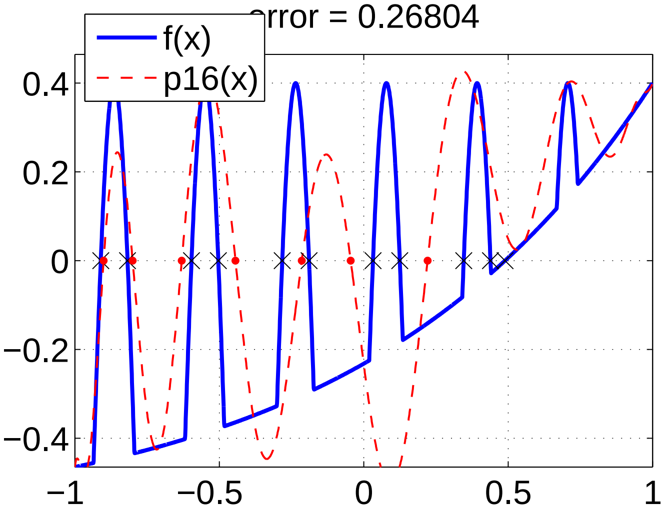
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



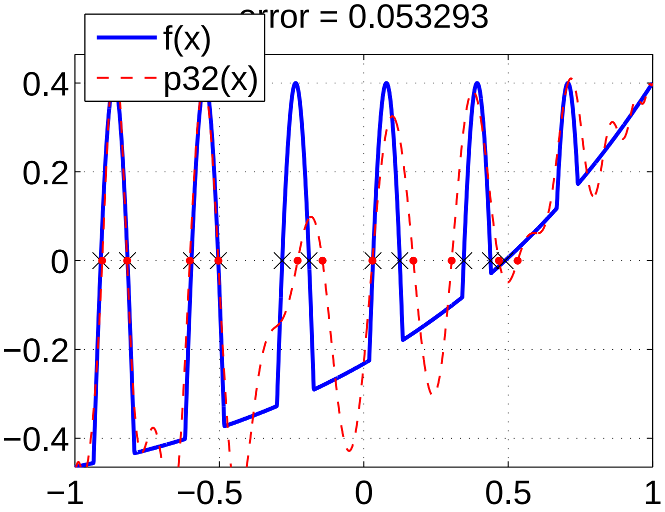
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



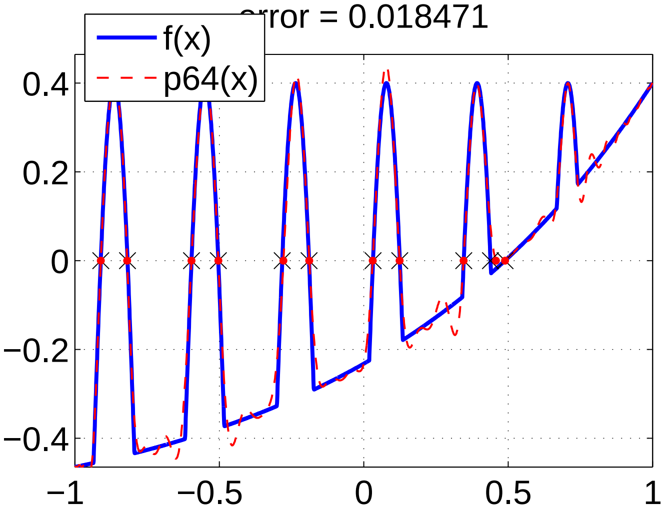
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



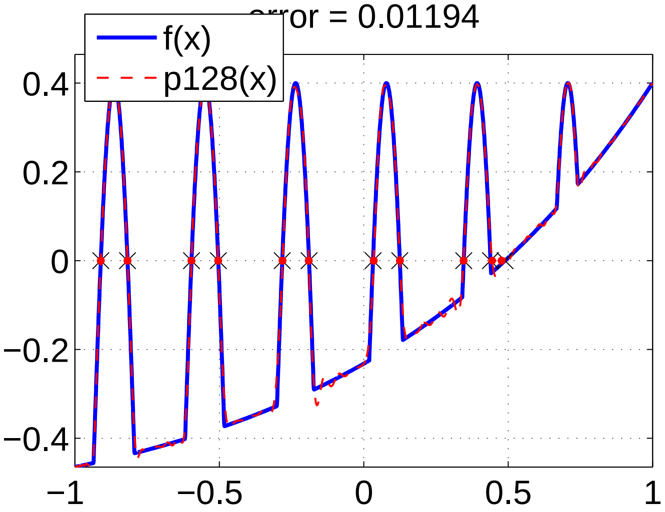
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



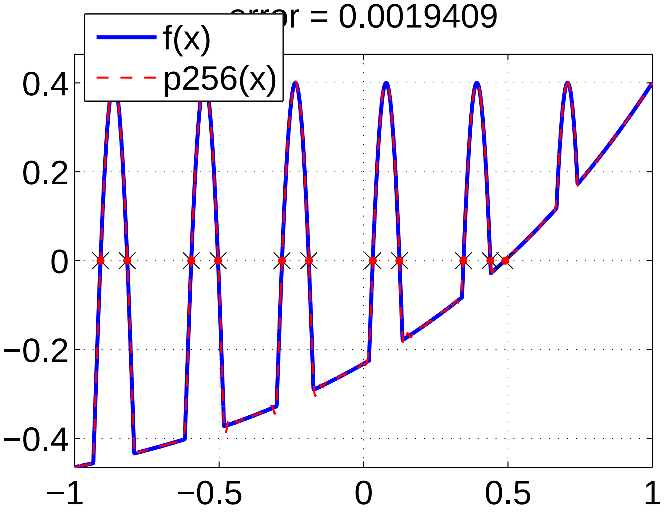
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



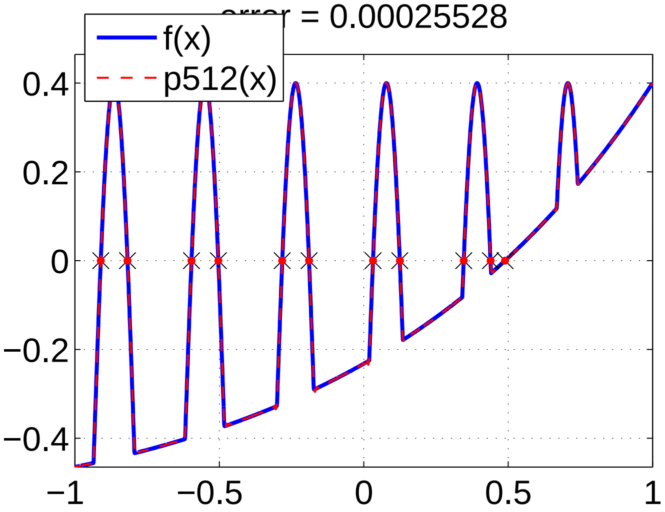
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



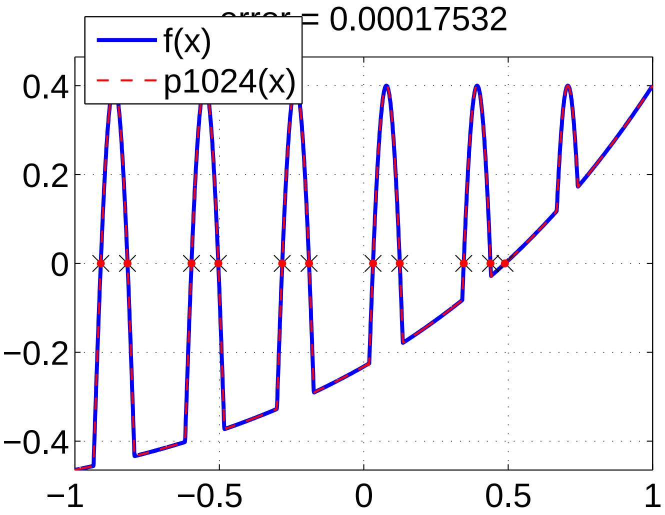
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



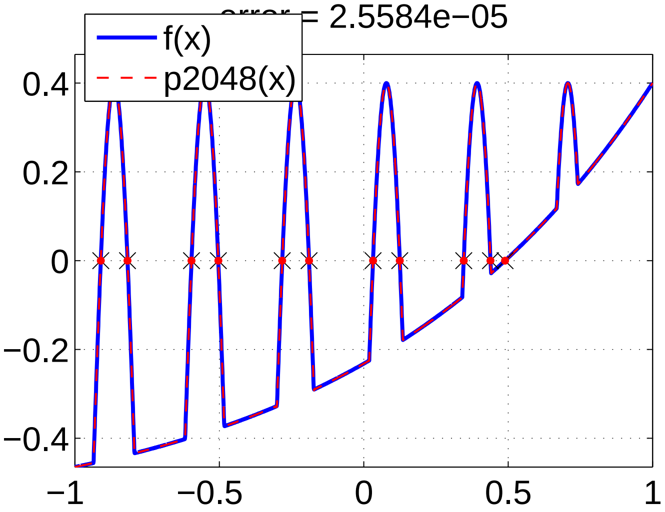
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



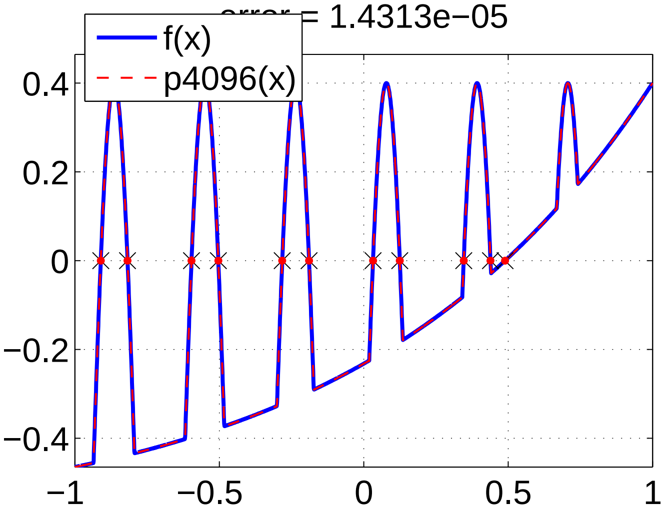
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



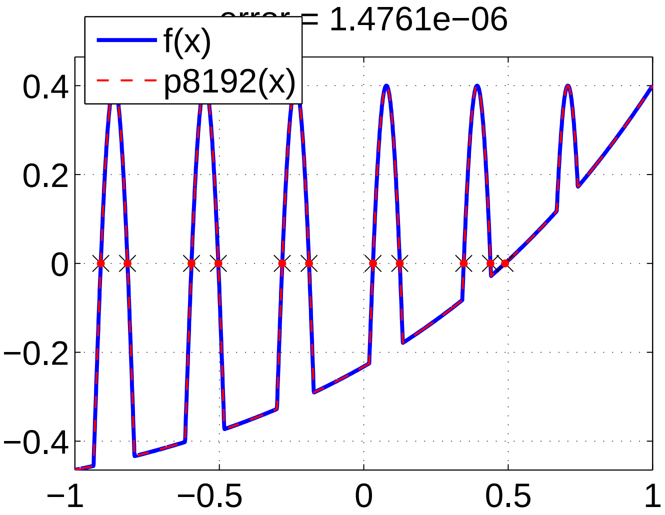
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



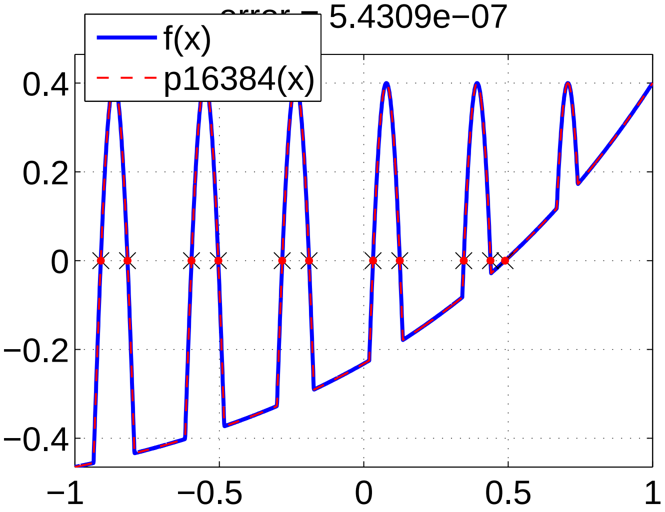
Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



Rootfinding $f(x) = 0$ on $[-1, 1]$: zigzag example



- High-degree polynomial approximation is accurate and stable

Recap: Rootfinding $f(x) = 0$ on $[-1, 1]$: principles

1. Approximate $f(x)$ with a polynomial $p(x)$:

$$\|f(x) - p(x)\|_\infty = O(\epsilon)\|f(x)\|_\infty \quad \text{on} \quad [-1, 1]$$

2. Find roots of $p(x)$ via generalized eigenvalues
   – e.g. companion linearization for $p(x) = \sum_{i=0}^{n} a_i x^i$: $Yv = \lambda Xv$,

$$(X, Y) = \left(\begin{bmatrix} a_n & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix} \right)$$

Recap: Rootfinding $f(x) = 0$ on $[-1, 1]$: principles

1. Approximate $f(x)$ with a polynomial $p(x)$:

$$\|f(x) - p(x)\|_\infty = O(\epsilon)\|f(x)\|_\infty \quad \text{on} \quad [-1, 1]$$

2. Find roots of $p(x)$ via generalized eigenvalues
   – e.g. companion linearization for $p(x) = \sum_{i=0}^{n} a_i x^i$: $Yv = \lambda Xv$,

$$(X, Y) = \left(\begin{bmatrix} a_n & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix} \right)$$

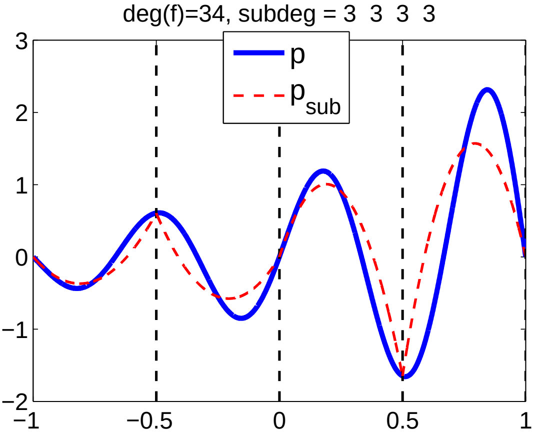
   – for p expressed in the Chebyshev basis $p(x) = \sum_{i=0}^{n} a_i T_i(x)$:

–[Good (61)]

$$(X, Y) = \left(\begin{bmatrix} a_n & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \frac{1}{2} \begin{bmatrix} -a_{n-1} & 1 - a_{n-2} & -a_{n-3} & \cdots & -a_0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 2 & 0 \end{bmatrix} \right)$$

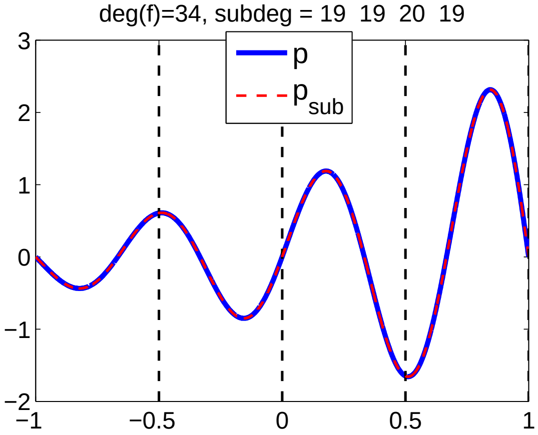
Solving $p_n(x) = 0$: subdivision for efficiency

- Finding $p_n(x) = 0$ via companion eigenvalues requires $O(n^3)$: dominant cost (recall interpolation is $O(n \log n)$)
- Remedy-Domain subdivision: Approximate $p_n(x)$ on smaller intervals with lower degree polynomials



Solving $p_n(x) = 0$: subdivision for efficiency

- Finding $p_n(x) = 0$ via companion eigenvalues requires $O(n^3)$: dominant cost (recall interpolation is $O(n \log n)$)
- Remedy-Domain subdivision: Approximate $p_n(x)$ on smaller intervals with lower degree polynomials



Bivariate rootfinding $f(x_*, y_*) = g(x_*, y_*) = 0$ on $[-1, 1]^2$

1. Approximate $f(x, y), g(x, y)$ with a bivariate polynomials $p(x, y), q(x, y)$ via e.g. 2D Chebyshev interpolation:

$$\|f(x, y) - p(x, y)\|_\infty = O(\epsilon)\|f(x, y)\|_\infty,$$
$$\|g(x, y) - q(x, y)\|_\infty = O(\epsilon)\|g(x, y)\|_\infty.$$

or often better: Chebfun2 [Townsend-Trefethen 13]

2. Find values of y_* via a polynomial eigenvalue problem using Bézout resultants

–[E. Bézout (1779)]

3. Find common roots of $p(x, y_*)$ and $q(x, y_*)$ via univariate rootfinding

Goal: Bivariate rootfinding of high degree

Problem: find (x_*, y_*) s.t.

$$\begin{pmatrix} p(x_*, y_*) \\ q(x_*, y_*) \end{pmatrix} = 0$$

where p, q are of degree n or less:

$$p(x,y) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_{i,j} T_i(x) T_j(y), \ \ q(x,y) = \sum_{i=0}^{n} \sum_{j=0}^{n} Q_{i,j} T_i(x) T_j(y)$$

Conventionally: complexity $O(n^6)$ or worse + erroneous solutions

- reduce to $O(n^4)$ in practice
- maximum degree estimate

	max(n)	solution
Mathematica, Maple	20	$\mathbb{C}$
Our	100 ~ 2000	$\mathbb{R}$

Bézoutian of two univariate polynomials $p(x), q(x)$

$\mathcal{B}(p,q)$ is a bivariate polynomial

$$\mathcal{B}_{p,q}(s,t) = \frac{p(s)q(t) - p(t)q(s)}{s - t}$$

Express the coefficients using a symmetric matrix $B = (b_{ij})$:

$$\mathcal{B}_{p,q}(s,t) = \sum_{i,j=0}^{N-1} b_{ij} T_i(s) T_j(t).$$

- $\det B$ is the resultant of p, q:

$$\det B = \Pi_{i,j}(roots_{p_i} - roots_{q_j})$$

Theorem 1

p, q share a root $p(x_*) = q(x_*)$
$\Leftrightarrow$ B is singular with null space $[T_0(x_*), T_1(x_*), \ldots, T_{N-1}(x_*)]^T$
$\Leftrightarrow$ $\mathcal{B}_{p,q}(x_*, t) = \mathcal{B}_{p,q}(s, x_*) = 0$

Using $\mathcal{B}_{p,q}(s,t)$ for solving $p(x_*,y_*) = q(x_*,y_*) = 0$

1. Regard y as fixed in $p(x,y), q(x,y)$. $\mathcal{B}_{p,q}(s,t)$ is a bivariate polynomial

$$\mathcal{B}_{p,q}(s,t) = \frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s-t} = \sum_{i,j=0}^{N-1} b_{ij} T_i(s) T_j(t).$$

2. $B(y) = (b_{ij})$ is a matrix polynomial in y. Since $\det B(y) = 0$ iff $p_y(x), q_y(x)$ share a root, we can find y_* by solving

$$\det B(y) = 0,$$

resulting in a polynomial eigenvalue problem.

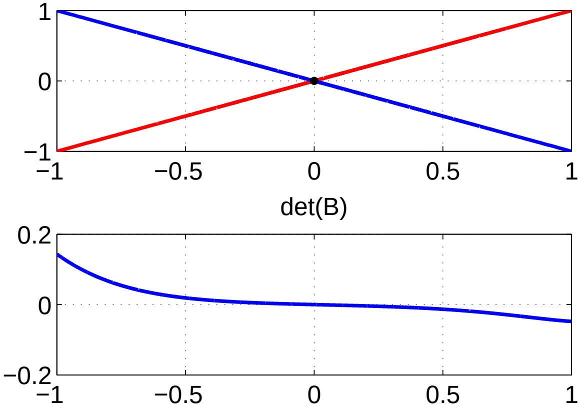
When $\deg(p_x, p_y, q_x, q_y)$ are all n, $B(y)$ is $n \times n$, degree $2n$

- Linearization is size $O(n^2) \Rightarrow \mathcal{O}(n^6)$ cost, infeasible for $n \gtrsim 50$
- Susceptible to numerical errors

$\mathcal{B}(p,q)$ for solving $p(x_*, y_*) = q(x_*, y_*) = 0$: examples

$$\mathcal{B}(p,q) = \frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s-t} = \sum_{i,j=0}^{N-1} b_{ij} T_i(s) T_j(t).$$

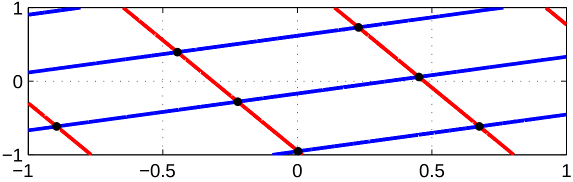
Let $B(y) = (b_{ij})$. We find y_* by solving $\det B(y) = 0$

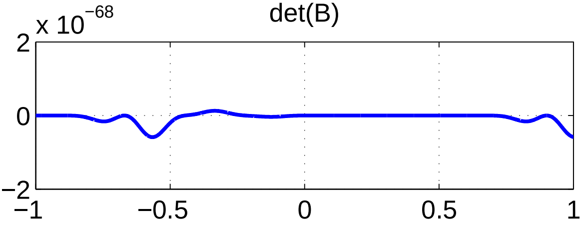


$\mathcal{B}(p,q)$ for solving $p(x_*,y_*) = q(x_*,y_*) = 0$: examples

$$\mathcal{B}(p,q) = \frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s-t} = \sum_{i,j=0}^{N-1} b_{ij}T_i(s)T_j(t).$$

Let $B(y) = (b_{ij})$. We find y_* by solving $\det B(y) = 0$





Companion-like matrix in Chebyshev basis

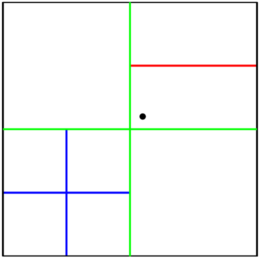
The colleague matrix pencil for a matrix polynomial $P(\lambda) = \sum_{i=0}^{M} A_i T_i(\lambda)$, $A_i \in \mathbb{R}^{N\times N}$ is

$$\lambda X + Y = \lambda \begin{bmatrix} A_M & & & \\ & I_N & & \\ & & \ddots & \\ & & & I_N \end{bmatrix} - \frac{1}{2} \begin{bmatrix} -A_{M-1} & I_N - A_{M-2} & -A_{M-3} & \cdots & -A_0 \\ I_N & 0 & I_N & & \\ & \ddots & \ddots & \ddots & \\ & & I_N & 0 & I_N \\ & & & 2I_N & 0 \end{bmatrix}.$$

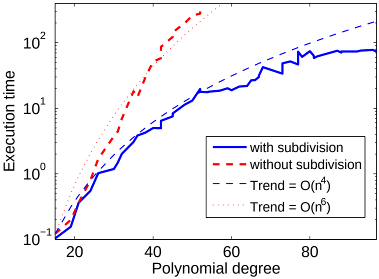
- Eigenvalues λ_* s.t. $\det P(\lambda_*) = 0$ satisfy $\det(\lambda_* X + Y) = 0$: generalized eigenvalue problem
- Other (infinitely many) linearizations available

Domain subdivision for efficiency (+stability)

- subdivide in x or/and y until the polynomial interpolants have degree 16 or less
- complexity typically $O(n^6) \to O(n^4)$



Subdivision for
$f = \sin((x - 1/10)y)\cos(1/(x + (y - 9/10) + 5)) = 0,$
$g = (y - 1/10)\cos((x + (y + 9/10)^2/4)) = 0.$



$\sin(\omega(x + y)) = \cos(\omega(x - y)) = 0$, where $\omega = 1, \ldots, 50$ (up to 20 without subdivision).

Conditioning of original problem

Suppose

- $p(x_*, y_*) = q(x_*, y_*) = 0$, $\|p\|_\infty = \|q\|_\infty = 1$
- for perturbed $\hat{p} = p + \delta p$, $\hat{q} = q + \delta q$, $\hat{p}(\hat{x}, \hat{y}) = \hat{q}(\hat{x}, \hat{y}) = 0$ for $(\hat{x}, \hat{y}) = (x_* + \delta x, y_* + \delta y)$

Then to first order in $\delta p, \delta q$,

$$0 = \begin{bmatrix} \hat{p}(\hat{x}, \hat{y}) \\ \hat{q}(\hat{x}, \hat{y}) \end{bmatrix} = \begin{bmatrix} \partial_x p(x_*, y_*) & \partial_y p(x_*, y_*) \\ \partial_x q(x_*, y_*) & \partial_y q(x_*, y_*) \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} + \begin{bmatrix} \delta p(\hat{x}, \hat{y}) \\ \delta q(\hat{x}, \hat{y}) \end{bmatrix}.$$

– A stable solution has error of size $O(\kappa_* u)$, where κ_* is the absolute condition number

$$\kappa_* = \lim_{\epsilon \to 0^+} \sup \left\{ \frac{1}{\epsilon} \min \left\| \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \right\|_2 : \hat{p}(\hat{x}, \hat{y}) = \hat{q}(\hat{x}, \hat{y}) = 0, \|\hat{p} - p\|_\infty \le \epsilon, \|\hat{q} - q\|_\infty \le \epsilon \right\},$$

It follows that $\kappa_* = \left\| \begin{bmatrix} \partial_x p(x_*, y_*) & \partial_y p(x_*, y_*) \\ \partial_x q(x_*, y_*) & \partial_y q(x_*, y_*) \end{bmatrix}^{-1} \right\|_2 := \|J^{-1}\|_2$, where J is the Jacobian matrix

Conditioning of our formulation: Bézout eigenproblem

The error in the computed eigenvalues of $B(y)$ is bounded by (conditioning)·(backward error), where the conditioning is

$$\kappa(y_*, B) = \lim_{\epsilon \to 0^+} \sup \left\{ \frac{1}{\epsilon} \min |\hat{y} - y_*| : \det \hat{B}(\hat{y}) = 0, \|B - \hat{B}\| \leq \epsilon \right\},$$

Theorem 2

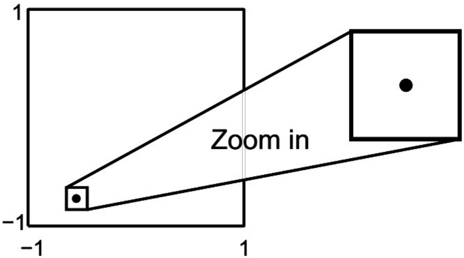
$$\frac{\kappa_*}{\left\| \begin{bmatrix} \partial_y q & -\partial_y p \\ -\partial_x q & \partial_x p \end{bmatrix} \right\|_2} \leq \kappa(y_*, B) \leq \frac{\kappa_* n}{\left\| \begin{bmatrix} \partial_y q & -\partial_y p \\ -\partial_x q & \partial_x p \end{bmatrix} \right\|_2}$$

- $\kappa_* \gg k_*$ (unstable) if $\mathrm{adj}(J) = \begin{bmatrix} \partial_y q & -\partial_y p \\ -\partial_x q & \partial_x p \end{bmatrix}$ is small

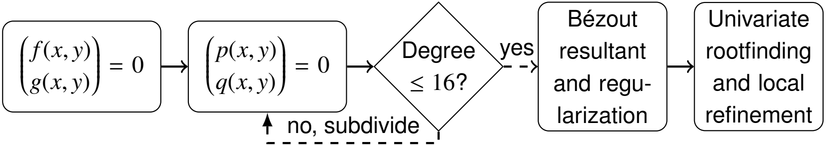
Local refinement for accuracy

$$\frac{\kappa_*}{\left\|\begin{bmatrix} \partial_y q & -\partial_y p \\ -\partial_x q & \partial_x p \end{bmatrix}\right\|_2} \leq \kappa(y_*, B) \leq \frac{\kappa_* n}{\left\|\begin{bmatrix} \partial_y q & -\partial_y p \\ -\partial_x q & \partial_x p \end{bmatrix}\right\|_2}$$

- (Bézout conditioning) $\gg$ (original conditioning) if derivatives are small: $|\partial p| \ll \|p\|_\infty$ and/or $|\partial q| \ll \|q\|_\infty$
- Remedy- Local refinement: very small region in which $\|p\|_\infty = O(|\partial p|), \|q\|_\infty = O(|\partial q|)$ (after scaling to $[-1, 1]^2$)



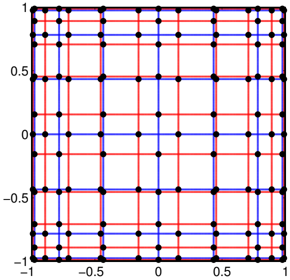
Flowchart



- dominant cost: Bézout polynomial eigenproblem
- choose first variable (hidden in Bézoutian) x or y from the size of the generalized eigenproblem (colleague linearization)
- multiple eigenvalues (e.g. solutions at (x_1, y_0), (x_2, y_0)) do not cause instability
  - Defective eigenvalues are indeed ill-conditioned
  - Non-defective eigenvalues are as well-conditioned as simple eigenvalues

Experiments

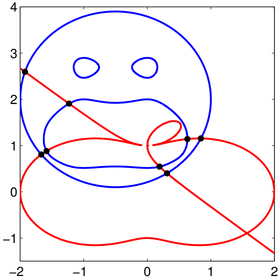
Coordinate alignment



$$(m_p, n_p, m_q, n_q) = (20, 20, 24, 30)$$

$$\begin{pmatrix} T_7(x)T_7(y)\cos(xy) \\ T_{10}(x)T_{10}(y)\cos(x^2y) \end{pmatrix} = 0$$

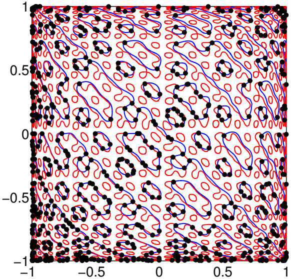
Face and apple



$$(m_p, n_p, m_q, n_q) = (10, 18, 8, 8)$$

Experiments-2

Hadamard

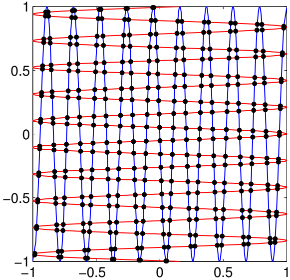


$$(m_p, n_p, m_q, n_q) = (31, 31, 63, 63)$$

$$f(x_i, x_j) = H_{32}(i, j),$$

$$g(x_i, x_j) = H_{64}(i, j)$$

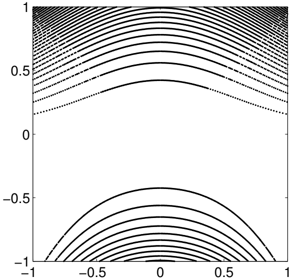
Travelling waves



$$(m_p, n_p, m_q, n_q) = (7, 63, 62, 6)$$

Experiments-3

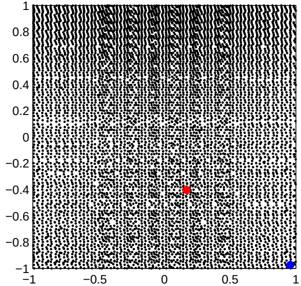
Airy



$$(m_p, n_p, m_q, n_q) = (171, 120, 569, 568)$$

$$\begin{pmatrix} \mathrm{Ai}(-13(x^2y + y^2)))) \\ J_0(500x)y + xJ_1(500y) \end{pmatrix} = 0$$

SIAM 100-Digit Challenge problem



$$(m_p, n_p, m_q, n_q) = (1781, 1204, 1781, 1204)$$

$$f(x, y) = \left(\frac{x^2}{4} + e^{\sin(100x)} + \sin(140\sin(x))\right) + \left(\frac{y^2}{4} + \sin(120e^y) + \sin(\sin(160y))\right)$$

Part I (bivariate rootfinding) Summary and future work

Summary:

- Algorithm for common zeros of two bivariate functions via Bézout resultant
- Improvements to solve polynomial interpolants of high degree up to 2000:
  - Domain subdivision for speed
  - Local refinement for resolving conditioning + removing spurious solutions

Future work:

- Trivariate (multivariate) zerofinding (many unsuccessful attempts)
- Non-analytic functions (singularities, poles,...)
- Finding appropriate initial domain in applications

Part II-a: The AAA algorithm for rational approximation

[N.-Sète-Trefethen (18)] Given $f(Z)$, $Z = \{Z_i\}_{i=1}^{M}$, find rational function r s.t.

$$\boxed{f(Z) \approx r(Z)}$$

- rationals outperform polynomials when f nonsmooth

Key ideas in AAA:

- Barycentric representation of rational functions

$$f(z) \approx r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z - t_j} \bigg/ \sum_{j=0}^{n} \frac{\beta_j}{z - t_j}$$

- Adaptive selection of support points, hence basis functions
- Least-squares fitting

Part II-a: The AAA algorithm for rational approximation

[N.-Sète-Trefethen (18)] Given $(f(Z), Z = \{Z_i\}_{i=1}^{M}) \leftarrow f$, find rational function r s.t.

$$\boxed{(f(Z) \approx r(Z)) \leftarrow f \approx r}$$

- rationals outperform polynomials when f nonsmooth

Key ideas in AAA:

- Barycentric representation of rational functions
  $f(z) \approx r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z - t_j} \Big/ \sum_{j=0}^{n} \frac{\beta_j}{z - t_j}$
- Adaptive selection of support points, hence basis functions
- Least-squares fitting

The (standard) AAA algorithm

Given sample points $\{Z_i\}_{i=1}^{M}$,

$$f(z) \approx r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z - t_j} \Bigg/ \sum_{j=0}^{n} \frac{\beta_j}{z - t_j}$$

---

1. $n \leftarrow n + 1$, add support point $t_n \in Z$,
2. Solve via SVD $\min_{\|\beta\|_2=1} \left\| \begin{bmatrix} \frac{f(Z_1^{(n)})-f_0}{Z_1^{(n)}-t_0} & \cdots & \frac{f(Z_1^{(n)})-f_n}{Z_1^{(n)}-t_n} \\ \vdots & \ddots & \vdots \\ \frac{f(Z_M^{(n)})-f_0}{Z_M^{(n)}-t_0} & \cdots & \frac{f(Z_M^{(n)})-f_n}{Z_M^{(n)}-t_n} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_n \end{bmatrix} \right\|$

Key idea: choice of support points $\{t_j\}$

Desiderata:

- Approximation error $\|f - r\|$ decreases
- “Basis matrix” $C_{ij} = \frac{1}{Z_i^n - t_j}$ well conditioned

Choice of support points $\{t_j\}$

$$f(z) \approx r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z - t_j} \Bigg/ \sum_{j=0}^{n} \frac{\beta_j}{z - t_j}, \qquad z \in Z^{(n)}$$

- Error after n steps: $e_n(z) := f(z) - r(z)$
- Greedy choice: take $t_{n+1} :=$ argmax$_{z \in Z}|e(z)|$
  - then $e_{n+1}(t_{n+1}) = 0$: interpolation property
- Basis $\{\frac{1}{z-t_j}\}$ chosen adaptively, depending on f

Desiderata:

- Approximation error $\|f - r\|$ decreases
- "Basis matrix" $C_{ij} = \frac{1}{Z_i^n - t_j}$ well conditioned

Choice of support points $\{t_j\}$

$$f(z) \approx r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z - t_j} \Bigg/ \sum_{j=0}^{n} \frac{\beta_j}{z - t_j}, \qquad z \in Z^{(n)}$$

- Error after n steps: $e_n(z) := f(z) - r(z)$
- Greedy choice: take $t_{n+1} :=$ argmax$_{z \in Z}|e(z)|$
  - then $e_{n+1}(t_{n+1}) = 0$: interpolation property
- Basis $\{\frac{1}{z - t_j}\}$ chosen adaptively, depending on f

Desiderata:

- Approximation error $\|f - r\|$ decreases
  - $\surd$ largest error position$\mapsto 0$
- "Basis matrix" $C_{ij} = \frac{1}{Z_i^n - t_j}$ well conditioned
  - $\surd$ 'localization'

Sample points in AAA

In standard AAA, the sample points Z are given in advance.

- Often, equispaced or randomly drawn points in domain
- Often $|Z| \gg n$, e.g. $|Z| = 10^4$ where degree $n \leq 100$
  (overkill? Issue when f is expensive to sample)
- When f has singularities (e.g. $f(x) = |x|$), need to cluster sample points exponentially near them (but their location is often **unknown**)

Sample points in AAA

In standard AAA, the sample points Z are given in advance.

- Often, equispaced or randomly drawn points in domain
- Often $|Z| \gg n$, e.g. $|Z| = 10^4$ where degree $n \leq 100$ (overkill? Issue when f is expensive to sample)
- When f has singularities (e.g. $f(x) = |x|$), need to cluster sample points exponentially near them (but their location is often **unknown**)

Question: Can we automate the choice of sample points?

Sample points in AAA

In standard AAA, the sample points Z are given in advance.

- Often, equispaced or randomly drawn points in domain
- Often $|Z| \gg n$, e.g. $|Z| = 10^4$ where degree $n \leq 100$ (overkill? Issue when f is expensive to sample)
- When f has singularities (e.g. $f(x) = |x|$), need to cluster sample points exponentially near them (but their location is often **unknown**)

Question: Can we automate the choice of sample points?

- Idea: Use support points to guide where more samples are needed
- Roughly: Sample three additional points between support points

Please see [Driscoll-N.-Trefethen (SISC, to appear)] for details (or chat later)

Part II-b: roots of rational functions via eigenvalues

$$r(x) = p(x) + \frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} + \cdots \frac{p_n(x)}{q_n(x)}$$

- barycentric form $r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z-t_j} \Big/ \sum_{j=0}^{n} \frac{\beta_j}{z-t_j}$, partial fraction, continued fraction

Goal: compute the roots x_0 of r s.t. $r(x_0) = 0$

- Bisection, Newton etc...
  - unclear how to verify all roots are computed
- "polynomialization": compute roots of polynomial $r(x) \prod_{i=1}^{n} q_i(x)$ via linearization
  - taking product can cause numerical issues

Part II-b: roots of rational functions via eigenvalues

$$r(x) = p(x) + \frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} + \cdots \frac{p_n(x)}{q_n(x)}$$

- barycentric form $r(z) = \frac{N(z)}{D(z)} = \sum_{j=0}^{n} \frac{\beta_j f_j}{z-t_j} \Big/ \sum_{j=0}^{n} \frac{\beta_j}{z-t_j}$, partial fraction, continued fraction

Goal: compute the roots x_0 of r s.t. $r(x_0) = 0$

- Bisection, Newton etc...
  - unclear how to verify all roots are computed
- "polynomialization": compute roots of polynomial $r(x)\prod_{i=1}^{n} q_i(x)$ via linearization
  - taking product can cause numerical issues
- This work: linearization without polynomializing
  - avoids numerical issues + easier to construct (Frobenius-style)

Companion linearization: monomial basis x^i

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

The companion linearization is

$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

- $\text{eig}(C) = \text{roots}(p)$

Colleague linearization: Chebyshev basis $T_i(x)$

$$p(x) = T_n(x) + a_{n-1}T_{n-1}(x) + \cdots + a_1T_1(x) + a_0T_0(x)$$

$T_i(x)$: Chebyshev polynomial

$$C = \frac{1}{2}\begin{bmatrix} -a_{n-1} & 1-a_{n-2} & -a_{n-3} & \cdots & -a_0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 2 & 0 \end{bmatrix}$$

- eig(C) = roots(p)

Comrade linearization: orthogonal polynomial basis $\phi_i(x)$

$$p(x) = \phi_n(x) + a_{n-1}\phi_{n-1}(x) + \cdots + a_1\phi_1(x) + a_0\phi_0(x)$$

$\phi_i(x)$: orthogonal polynomial with recurrence

$$x\phi_i(x) = \alpha_i\phi_{i+1}(x) + \beta_i\phi_i(x) + \gamma_i\phi_{i-1}(x)$$

$$C = \begin{bmatrix} \beta_{n-1} - a_{n-1} & \gamma_{n-1} - a_{n-2} & -a_{n-3} & \cdots & -a_0 \\ \alpha_{n-2} & \beta_{n-2} & \gamma_{n-2} & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_1 & \beta_1 & \gamma_1 \\ & & & \alpha_0 & \beta_0 \end{bmatrix}$$

- eig(C) = roots(p)
- Other extensions known: confederate (degree graded), congenial (degree bounded)..., trigonometric polynomials $c + \sum_{i=1}^{k}(a_k \sin kx + b_k \cos kx)$ [Boyd 2013]

Companion-like linearizations: how?

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

For $p(\lambda) = 0$,

$$\begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \lambda^{n-1} \\ \lambda^{n-2} \\ \vdots \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} \lambda^{n-1} \\ \lambda^{n-2} \\ \vdots \\ 1 \end{bmatrix}$$

- Bottom $n-1$ rows: $\lambda^i = \lambda^{i-1} \cdot \lambda$
- First row: $p(x) = 0$

Companion-like linearizations: how?

$$p(x) = T_n(x) + a_{n-1}T_{n-1}(x) + \cdots + a_1T_1(x) + a_0T_0(x)$$

For $p(\lambda) = 0$,

$$\frac{1}{2}\begin{bmatrix} -a_{n-1} & 1 - a_{n-2} & -a_{n-3} & \cdots & -a_0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 2 & 0 \end{bmatrix} \begin{bmatrix} T_{n-1}(\lambda) \\ T_{n-2}(\lambda) \\ \vdots \\ T_0(\lambda) \end{bmatrix} = \lambda \begin{bmatrix} T_{n-1}(\lambda) \\ T_{n-2}(\lambda) \\ \vdots \\ T_0(\lambda) \end{bmatrix}$$

- Bottom $n-1$ rows: $\frac{1}{2}(T_i(\lambda) + T_{i-2}(\lambda)) = T_{i-1}(\lambda) \cdot \lambda$
- First row: $p(\lambda) = 0$

Companion-like linearizations: how?

$$p(x) = \phi_n(x) + a_{n-1}\phi_{n-1}(x) + \cdots + a_1\phi_1(x) + a_0\phi_0(x)$$

$\phi_i(x)$: orthogonal polynomial with recurrence

$$x\phi_i(x) = \alpha_i\phi_{i+1}(x) + \beta_i\phi_i(x) + \gamma_i\phi_{i-1}(x)$$

For $p(\lambda) = 0$,

$$\begin{bmatrix} \beta_{n-1} - a_{n-1} & \gamma_{n-1} - a_{n-2} & -a_{n-3} & \cdots & -a_0 \\ \alpha_{n-2} & \beta_{n-2} & \gamma_{n-2} & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_1 & \beta_1 & \gamma_1 \\ & & & \alpha_0 & \beta_0 \end{bmatrix} \begin{bmatrix} \phi_{n-1}(\lambda) \\ \phi_{n-2}(\lambda) \\ \vdots \\ \phi_0(\lambda) \end{bmatrix} = \lambda \begin{bmatrix} \phi_{n-1}(\lambda) \\ \phi_{n-2}(\lambda) \\ \vdots \\ \phi_0(\lambda) \end{bmatrix}$$

- Bottom $n-1$ rows: $\lambda\phi_i(\lambda) = \alpha_i\phi_{i+1}(\lambda) + \beta_i\phi_i(\lambda) + \gamma_i\phi_{i-1}(\lambda)$
- First row: $p(\lambda) = 0$

Linearization for rational function in partial fraction form

$$r(x) = p(x) + \sum_{i=1}^{n} \frac{r_i}{(x - \gamma_i)}$$

$p(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$: polynomial

Note: includes root/polefinding for barycentric $r(z) = \sum_{j=0}^{n} \frac{\beta_j f_j}{z-t_j} \Big/ \sum_{j=0}^{n} \frac{\beta_j}{z-t_j}$,

$$(Cy =) \begin{bmatrix} -a_{d-1} & -a_{d-2} & \ldots & -a_0 & -r_1 & -r_2 & \ldots & r_n \\ 1 & & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & 1 & \gamma_1 & & & \\ & & & 1 & & \gamma_2 & & \\ & & & \vdots & & & \ddots & \\ & & & 1 & & & & \gamma_n \end{bmatrix} y = \lambda y, \qquad y = \begin{bmatrix} \lambda^{d-1} \\ \vdots \\ 1 \\ \frac{1}{\lambda-\gamma_1} \\ \frac{1}{\lambda-\gamma_2} \\ \vdots \\ \frac{1}{\lambda-\gamma_n} \end{bmatrix}$$

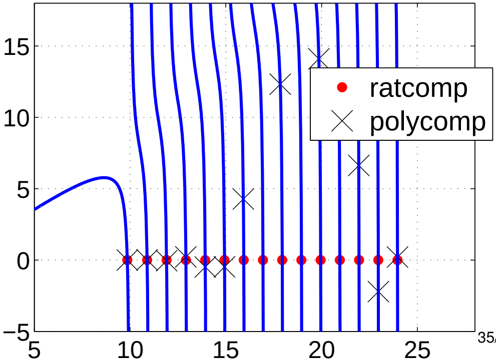
- eig(C) = roots(r) [Saad, El-Guide, Miedlar (19)]
- bottom rows: $1 + \frac{\gamma_i}{\lambda-\gamma_i} = \frac{1}{\lambda-\gamma_i} \cdot \lambda$

Example: 'Perfidious rational function'

$$r(x) = x + \sum_{i=1}^{15} \frac{1}{x-9-i}$$

- Polynomialize: $p(x) = r(x)\prod_{i=1}^{15}(x-9-i)$ has coefficients $\mathcal{O}(10^{18})$
- Rational companion: $\|C\|_2 \approx 20$
- `eig`: computes eigenvalues of $C + \epsilon\|C\|$

	$\max_i \lvert r(\hat{x}_i)\rvert$
C for $r(x)$	4e-11
C for $p(x)$	6.8e+3



Linearization for continued fractions

$$r(x) = p(x) + \cfrac{b_1}{x + a_1 + \cfrac{b_2}{x+a_2+\frac{b_3}{x+a_3+\cdots}}}$$

$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$: polynomial (below shown as $*$)

$$C_0 + \lambda C_1 = \begin{bmatrix} * & b_1 & & & \\ 1 & -a_1 & -b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -a_{k-1} & -b_{k-1} \\ & & & 1 & -a_k \end{bmatrix} + \lambda \begin{bmatrix} a_0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

with eigenvector $(C_0 + \lambda C_1)x = 0, \quad x = \begin{bmatrix} \text{(poly)} \\ \vdots \\ \frac{\frac{1}{\lambda+a_k}}{\lambda+a_{k-1}+\frac{b_k}{\lambda+a_k}} \\ \frac{1}{\lambda+a_{k-1}+\frac{b_k}{\lambda+a_k}} \end{bmatrix}.$

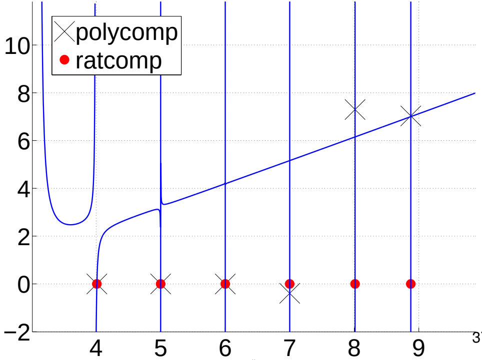
- eig(C) = roots(r) [Salazer 23]
- bottom rows: $g(x) := \frac{\frac{1}{x+a_2}}{x+a_1+\frac{b_1}{x+a_2}}$, $xg(x) = \frac{x}{x+a_{k-1}+\frac{b_k}{x+a_k}} = 1 + \frac{-a_{k-1}-\frac{b_k}{x+a_k}}{x+a_{k-1}+\frac{b_k}{x+a_k}}$

'Perfidious rational function', continued fraction

$$r(x) = x - 2 + \cfrac{1}{x - 1 + \cfrac{-1}{x+2+\cfrac{1}{x+3-\cdots+1/(x+11)}}}$$

- Polynomialize: $p(x) = r(x)\prod_{i=1}^{15}(x-9-i)$ has coefficients $\mathcal{O}(10^8)$
- Rational companion: $\|C\|_2 \approx 10$
- `eig`: computes eigenvalues of $C + \epsilon\|C\|$

	$\max_i \lvert r(\hat{x}_i)\rvert$
C for $r(x)$	4e-4
C for $p(x)$	7.2e0



Linearization for rational functions, more general form

$$r(x) = p(x) + \sum_{i=1}^{n} \left(\sum_{j=1}^{m_i} \frac{r_{ij}}{(x - \gamma_i)^j} \right), \tag{1}$$

$$C = \begin{bmatrix}
-a_d & -a_{d-1} & \dots & -a_0 & -r_{11} & \dots & -r_{1m_1} & -r_{21} & \dots & -r_{nm_n} \\
1 & & & & & & & & & \\
 & 1 & & & & & & & & \\
 & & \ddots & & & & & & & \\
 & & & 1 & \gamma_1 & & & & & \\
 & & & & \ddots & \ddots & & & & \\
 & & & & & 1 & \gamma_1 & & & \\
 & & & 1 & & & & \gamma_2 & & \\
 & & & & & & & 1 & \ddots & \\
 & & & 1 & & & & & & \gamma_n
\end{bmatrix}$$

- $\mathrm{eig}(C) = \mathrm{roots}(p)$

Conclusion and discussion

Summary

- Linearize rational function without polynomializing
  - typically reduced matrix norm, improved stability

Other things possible

- DL-type linearization
- Matrix rational functions (as opposed to scalars [Su,Bai 2011])

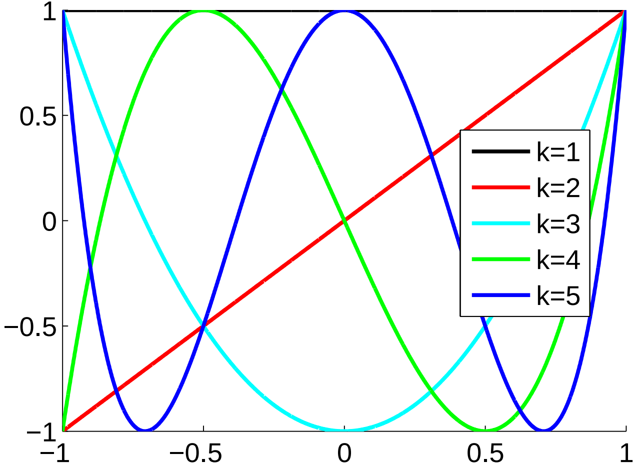
To be examined

- Conditioning
- Exploit structure (low-rank etc) in matrix case
- Trigonometric + polynomial (or rational)

Coming: more systematic framework using Schur complements (with Vanni Noferini and Maria Quintana Ponce),

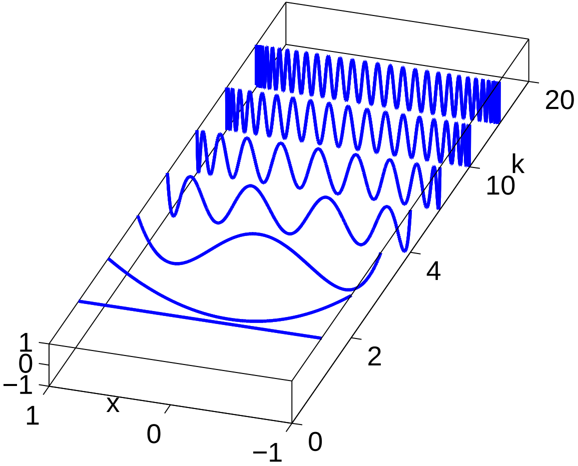
Chebyshev polynomials $T_k(x)$

$$T_k(\cos\theta) = \cos k\theta$$



Chebyshev polynomials $T_k(x)$

$$T_k(\cos\theta) = \cos k\theta$$



Polynomial bases and Chebyshev coefficients

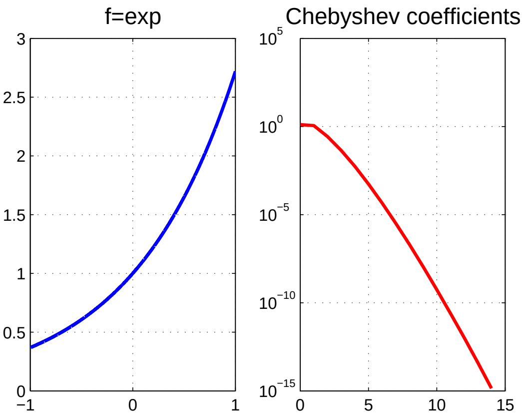
$p(x)$ expressed in

- monomial basis: $p(x) = \sum_{i=0}^{n} c_i x^i$
- Chebyshev basis: $p(x) = \sum_{i=0}^{n} c_i T_i(x)$
  - For smooth $f \approx p$, $|c_n| \to 0$ as $n \to \infty$, often like $|c_n| = e^{-cn}$

Polynomial bases and Chebyshev coefficients

$p(x)$ expressed in

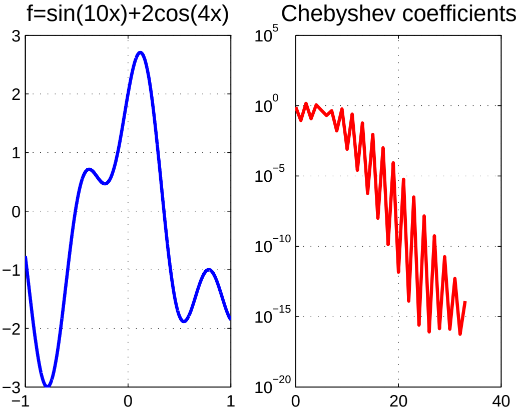
- monomial basis: $p(x) = \sum_{i=0}^{n} c_i x^i$
- Chebyshev basis: $p(x) = \sum_{i=0}^{n} c_i T_i(x)$
  - For smooth $f \approx p$, $|c_n| \to 0$ as $n \to \infty$, often like $|c_n| = e^{-cn}$



Polynomial bases and Chebyshev coefficients

$p(x)$ expressed in

- monomial basis: $p(x) = \sum_{i=0}^{n} c_i x^i$
- Chebyshev basis: $p(x) = \sum_{i=0}^{n} c_i T_i(x)$
  - For smooth $f \approx p$, $|c_n| \to 0$ as $n \to \infty$, often like $|c_n| = e^{-cn}$

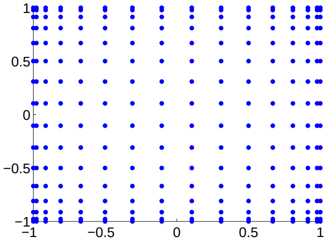


2D polynomial approximation by interpolation

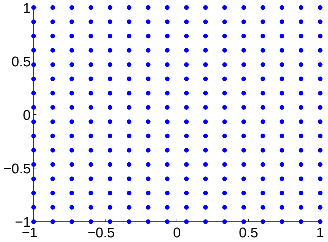
Interpolation: find $p(x,y)$ s.t.

$$f(x_i,y_j)=p(x_i,y_j),\quad i=0,1,\dots,n,\quad j=0,1,\dots,m.$$

Chebyshev points (good)



Equispaced points (bad)

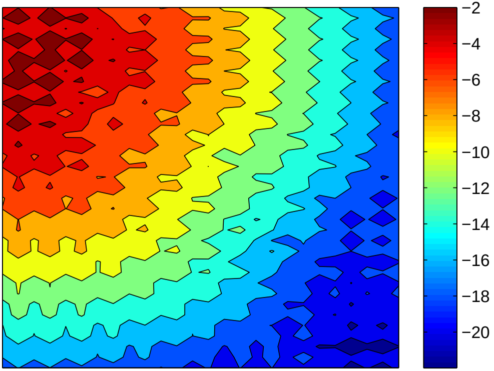


- For sufficiently smooth $f(x,y)$, with Chebyshev interpolation

$$\|f-p_{n,m}\|\le Ce^{-c\min(n,m)}$$

Bivariate polynomial approximation: $f(x,y) \approx p(x,y)$

$$p(x,y) = \sum_{i=0}^{m_p} \sum_{j=0}^{n_p} P_{ij} T_i(y) T_j(x)$$

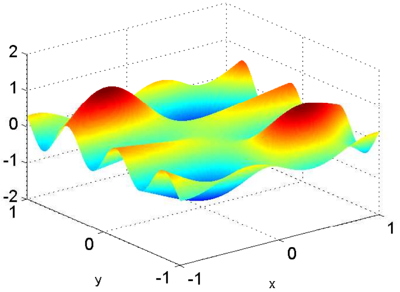


$\log_{10} |P_{ij}|$ contour plot

Bivariate Chebyshev interpolation: convergence

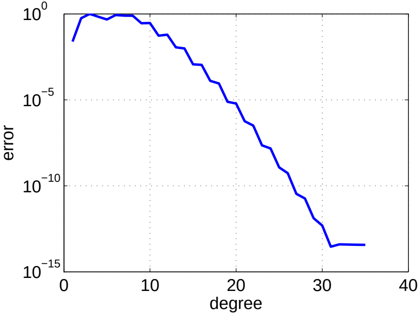
$$f(x,y) = \sin(5xy)\cos(5y)\exp((x+y)/10)$$

function



convergence:

$\|f(x,y) - p(x,y)\|_\infty$



Regularization for improved numerical stability

– $B(y)$ is nearly singular: $B(y)$ is ill-conditioned for every y

- decaying Chebyshev coefficients $\Rightarrow$ small bottom-right corner

Remedy-Regularization: partition $B(y)$ as

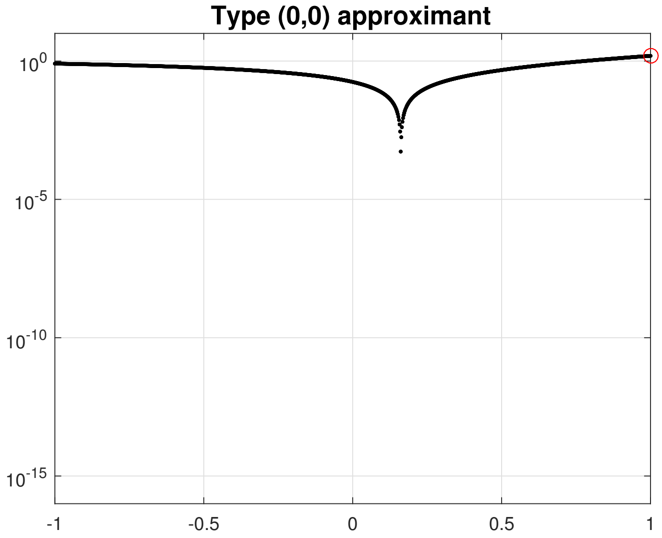
$$B(y) = \begin{bmatrix} B_1(y) & E(y)^T \\ E(y) & B_0(y) \end{bmatrix},$$

$B_1(y)$ is the largest numerically nonsingular part, i.e., for any $y_0 \in [-1, 1]$

- $||B_0(y_0)||_2 = \mathcal{O}(u)$
- $||E(y_0)||_2 = \mathcal{O}(u^{1/2})$

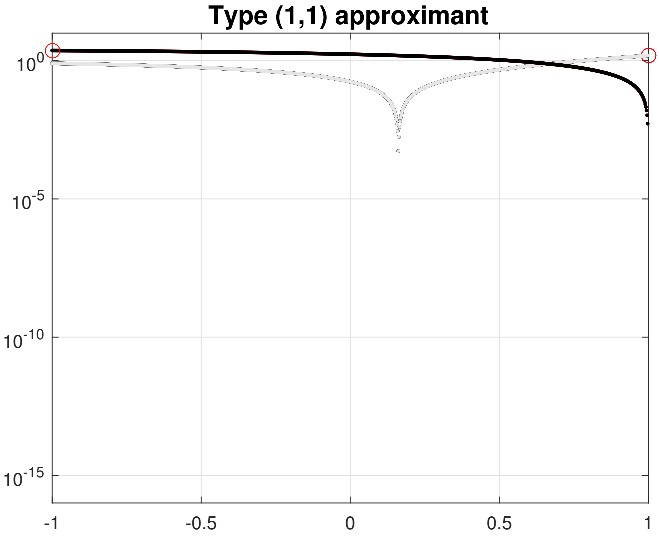
We prove eig$(B_1(y))$ are within $O(\epsilon)$ of the desired eig$(B(y))$
$\Rightarrow$ work with $B_1(y)$, “more regular” (accurate) + efficient

Example $f(x) = \mathrm{e}^x$ on $[-1, 1]$



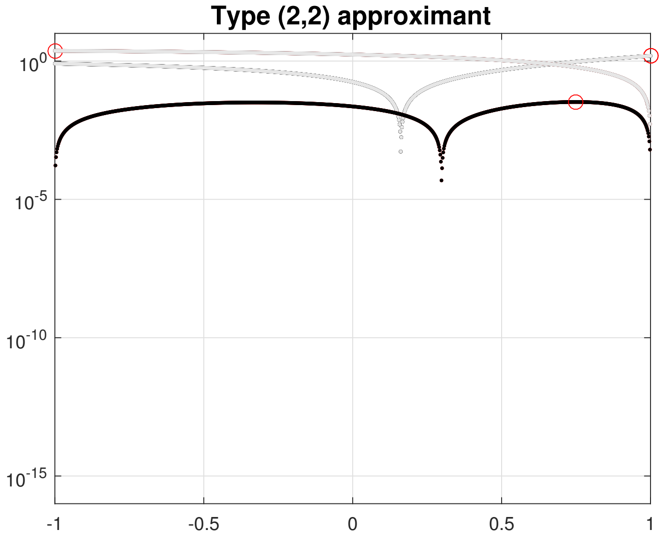
Red circle: support points

Example $f(x) = e^x$ on $[-1, 1]$



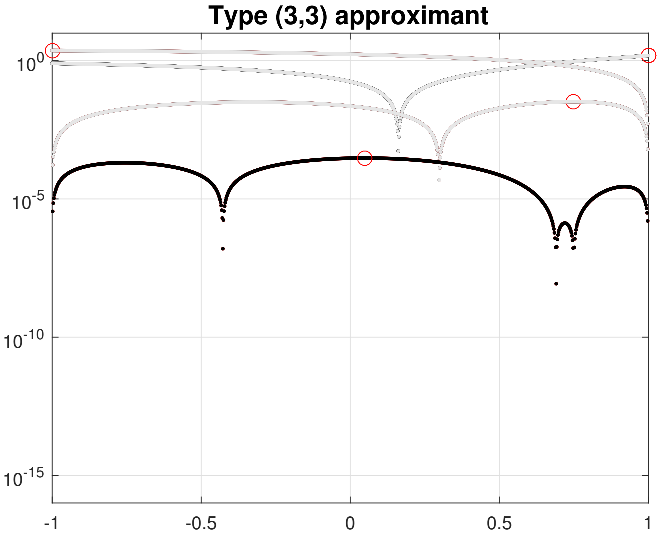
Red circle: support points

Example $f(x) = e^x$ on $[-1, 1]$



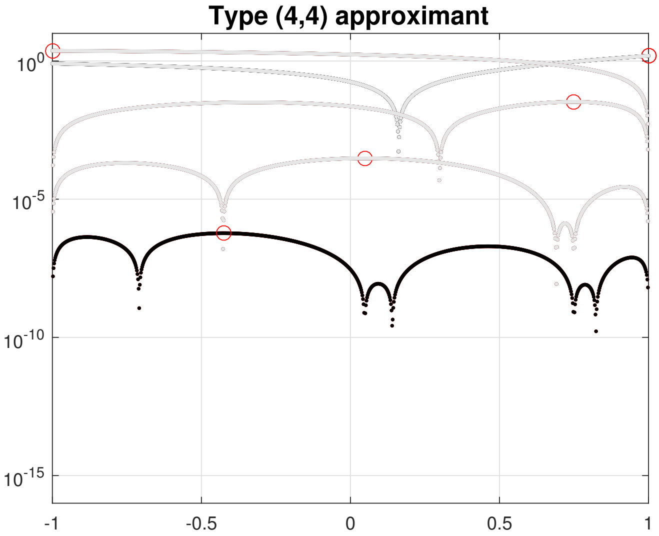
Red circle: support points

Example $f(x) = \mathrm{e}^x$ on $[-1, 1]$



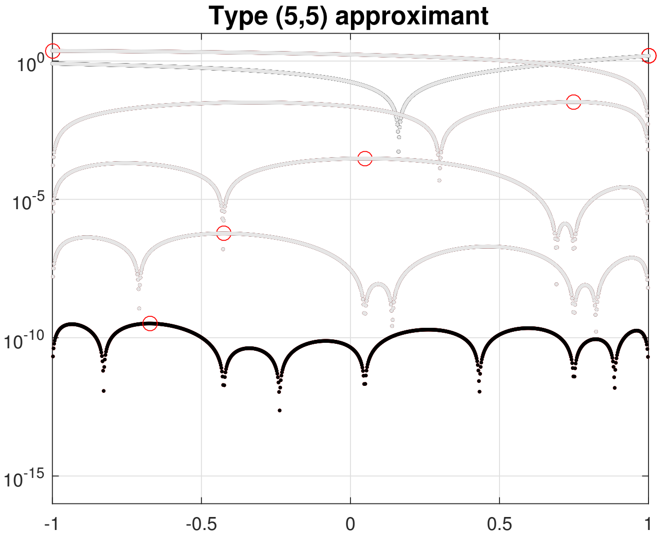
Red circle: support points

Example $f(x) = \mathrm{e}^x$ on $[-1, 1]$



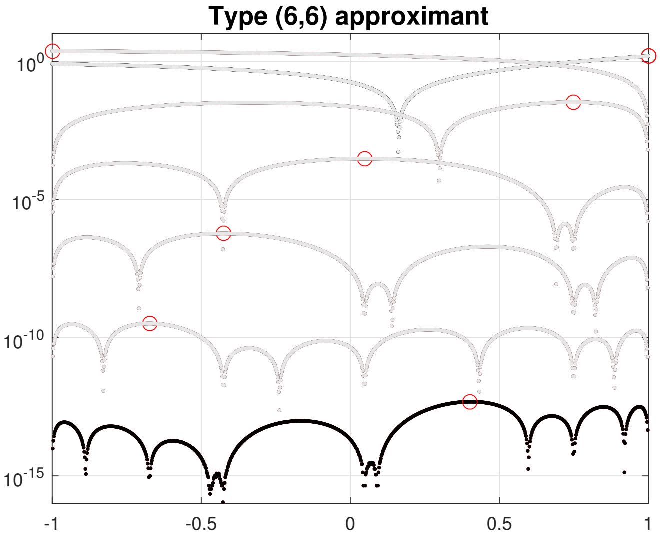
Red circle: support points

Example $f(x) = e^x$ on $[-1, 1]$



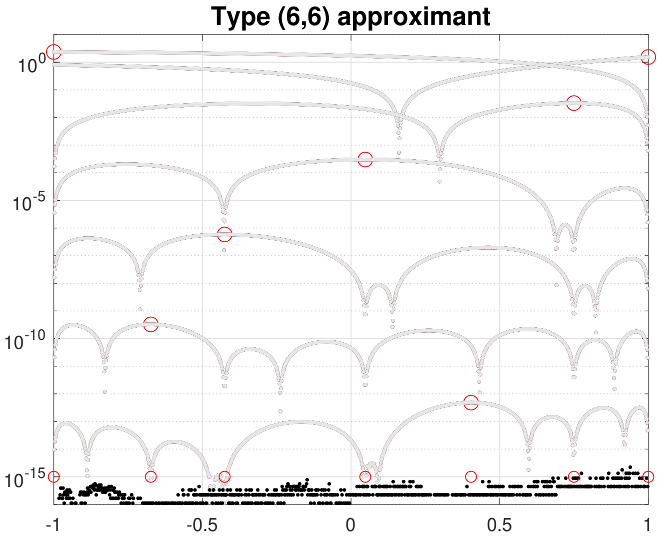
Red circle: support points

Example $f(x) = e^x$ on $[-1, 1]$



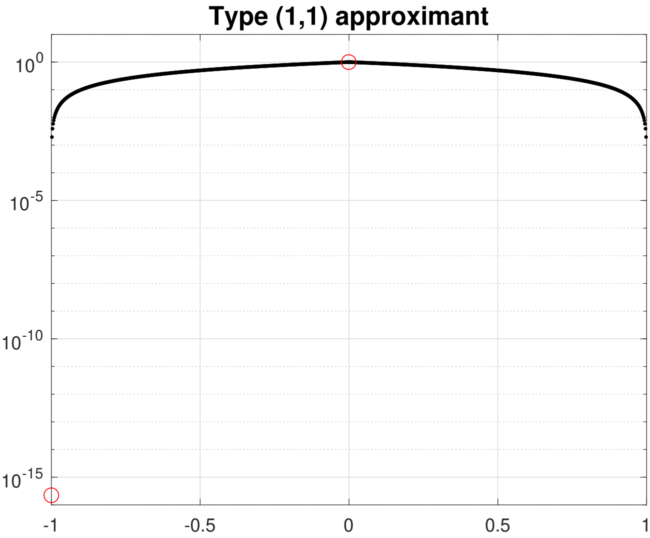
Red circle: support points

Example $f(x) = e^x$ on $[-1, 1]$



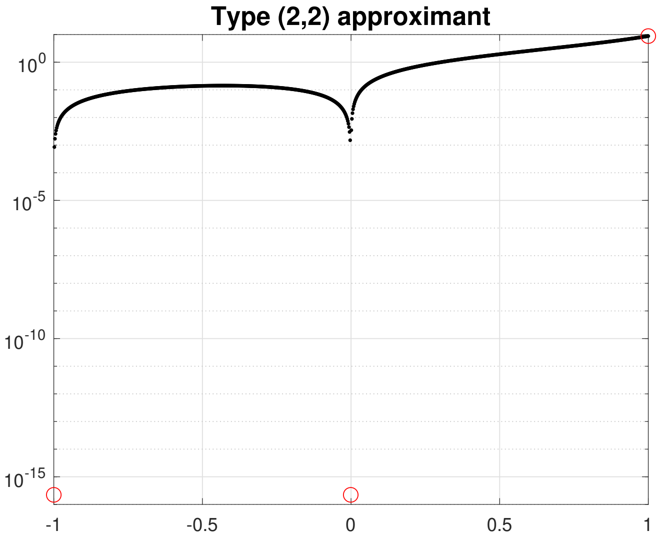
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



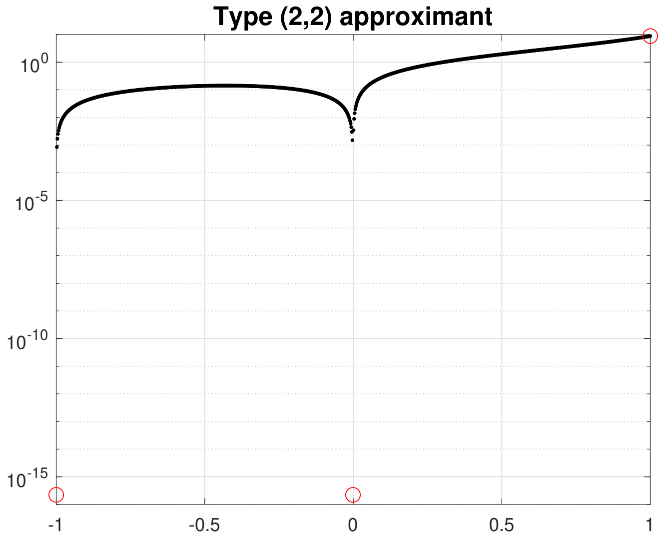
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



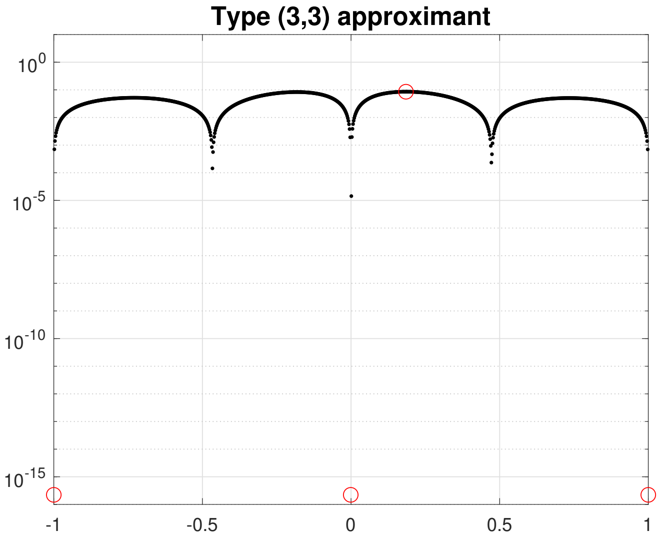
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



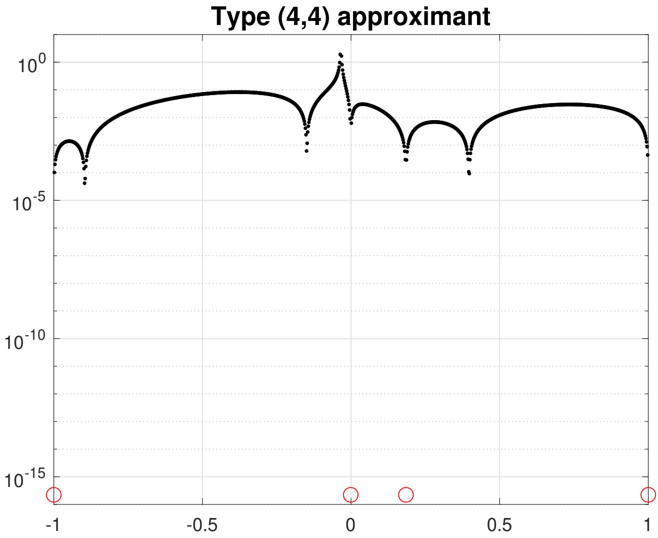
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



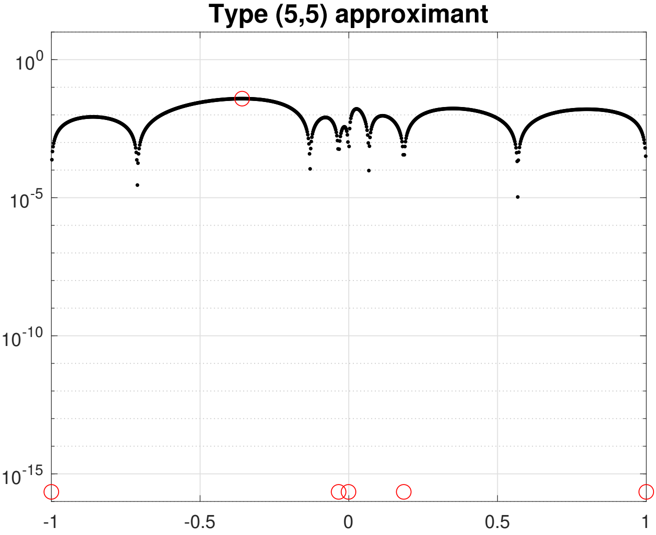
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



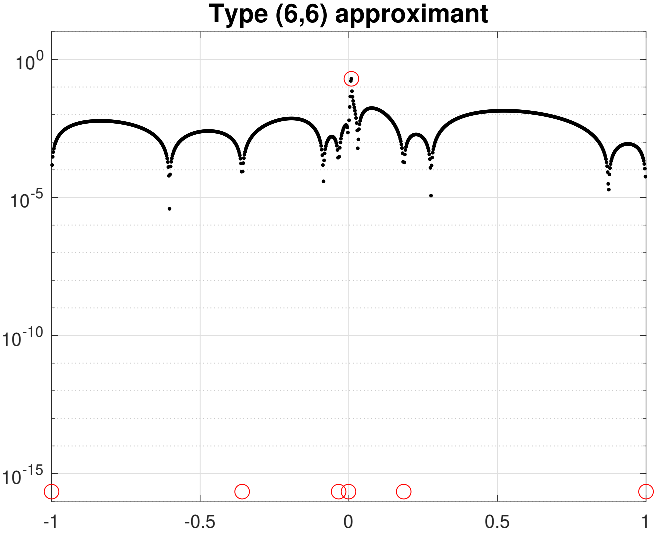
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



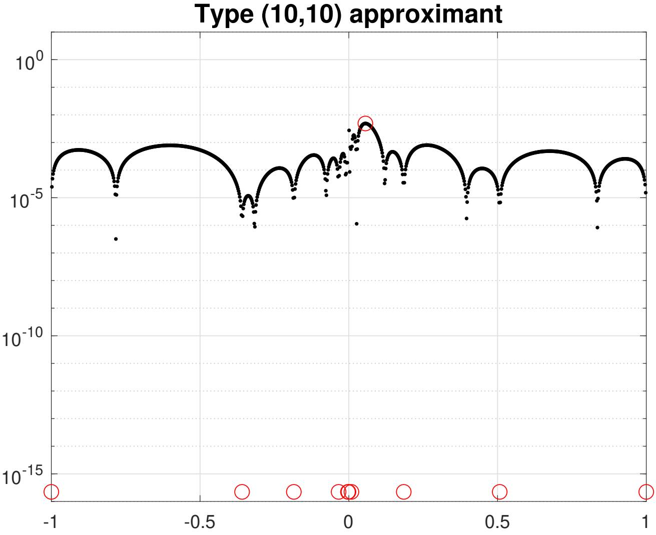
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



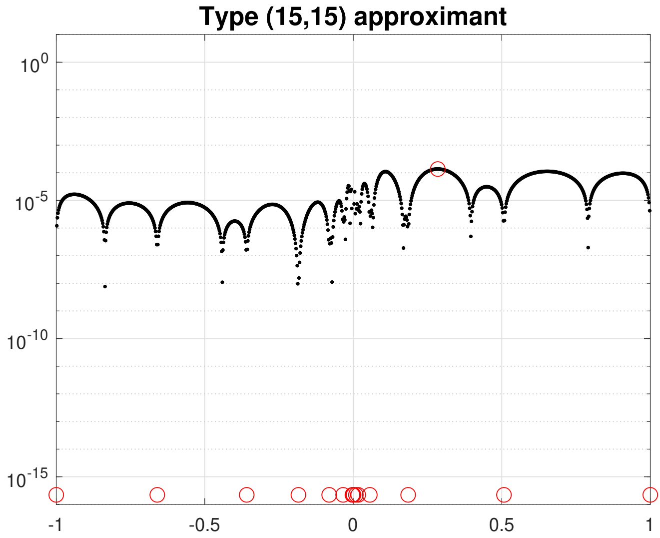
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



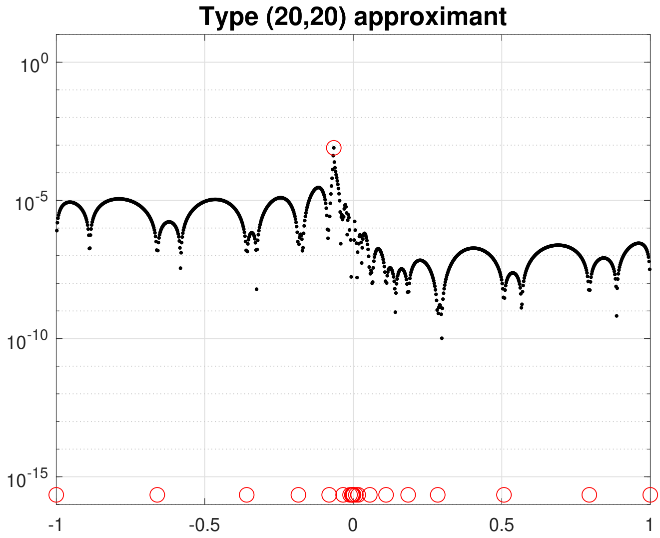
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



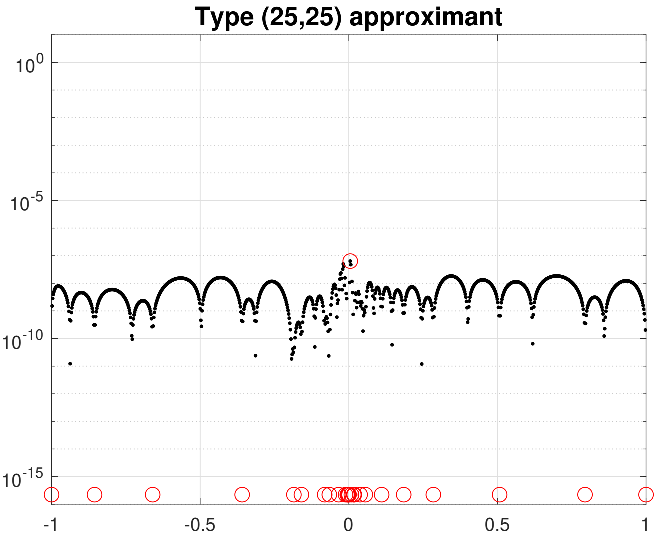
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



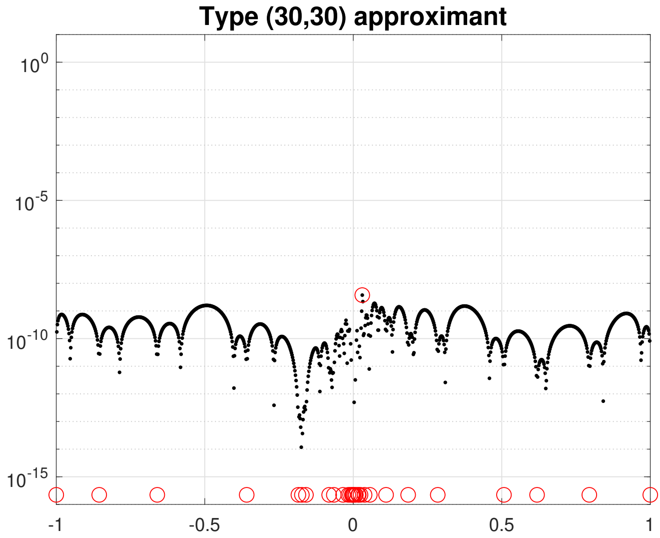
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



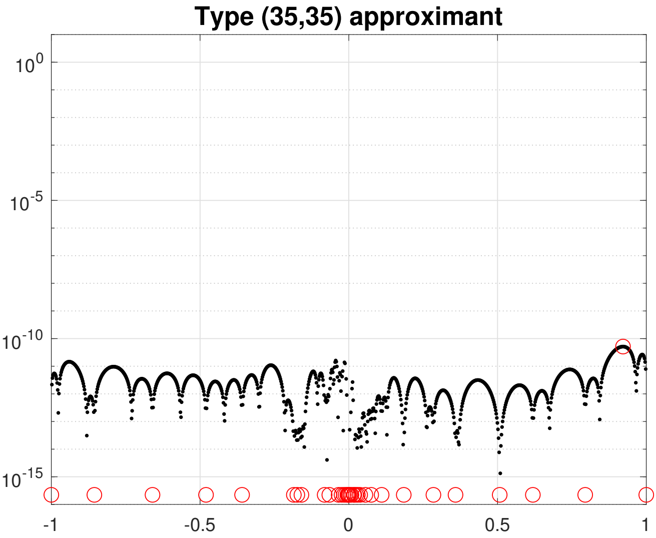
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



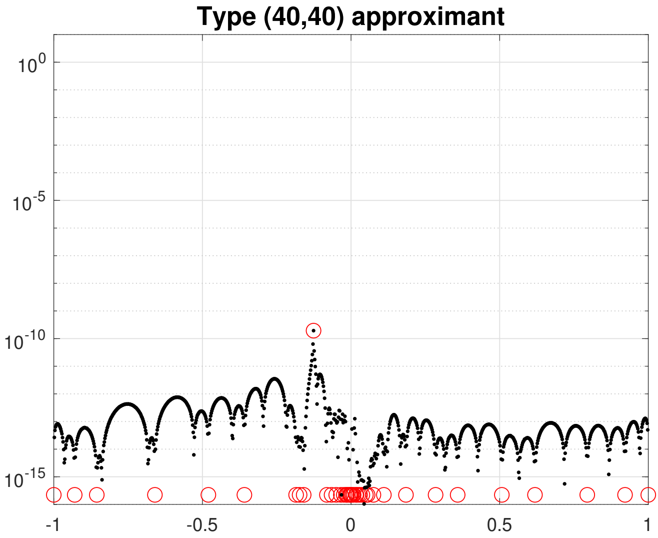
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



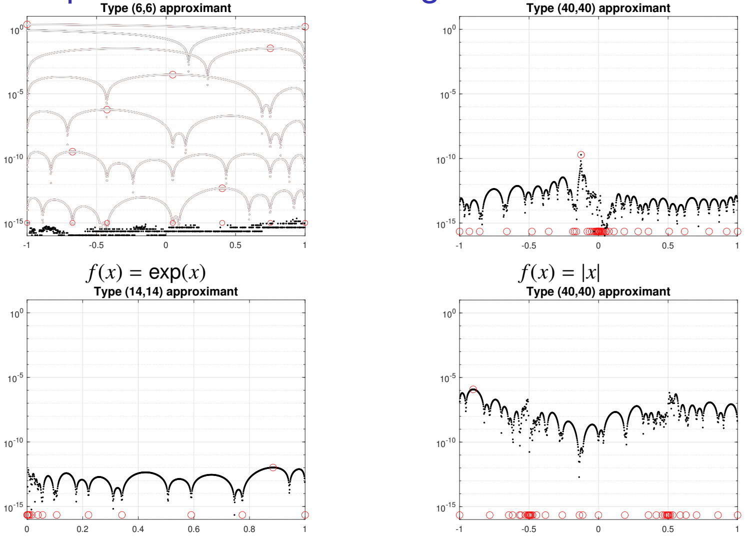
Red circle: support points

Example $f(x) = |x|$ on $[-1, 1]$



Red circle: support points

Support points cluster near singularities

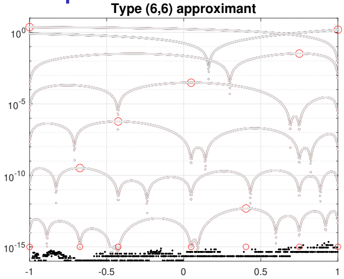


$f(x) = \exp(x)$ $\quad$ $f(x) = |x|$

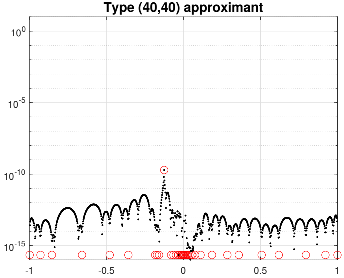
$f(x) = \sqrt{x}$ $\quad$ $f(x) = |x - 0.5| + |x + 0.5|$

- For functions with singularities, support points cluster near them

Support points cluster near singularities



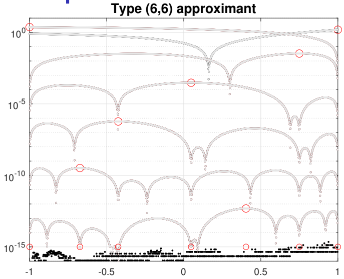
$f(x) = \exp(x)$



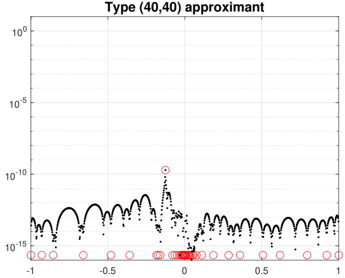
$f(x) = |x|$

- Support points cluster near singularities
- Informally: because hard to get error small there

Support points cluster near singularities



$f(x) = \exp(x)$

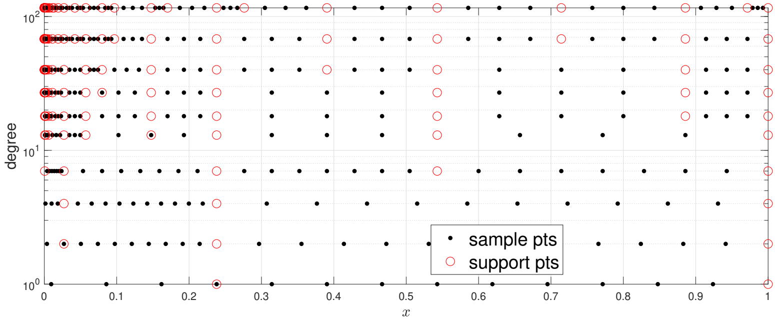


$f(x) = |x|$

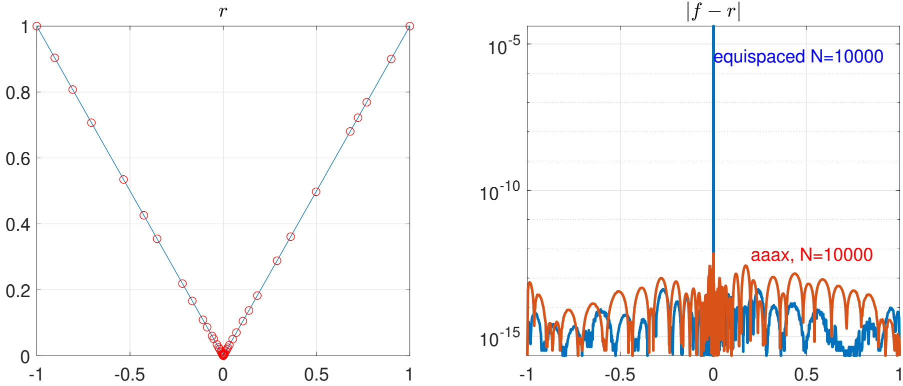
- Support points cluster near singularities
- Informally: because hard to get error small there
- Hence support points useful for clustering sample points on the fly
- We'll add three sample points between support points (i.e., six new points per step)

Continuum AAA: support+sample points

Sample points with aaax on $x > 0$, for $f(x) = |x|$:

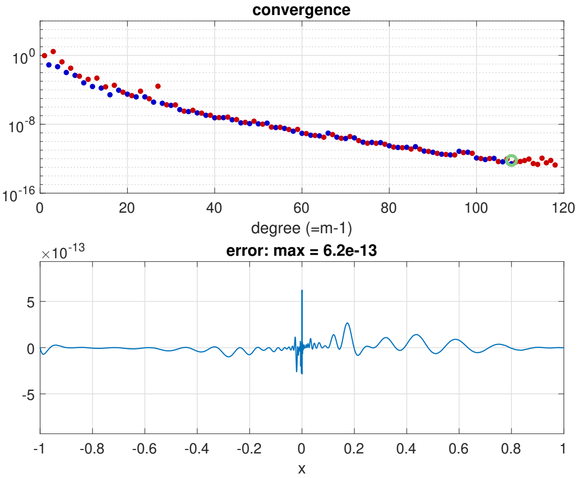


AAA vs. continuum AAA for $|x|$



- AAA with equispaced sample points yield poor error near singularity
- Continuum AAA yields good accuracy with fewer samples
- Monitor poles on $[-1, 1]$ ('bad poles'), ensure output is pole-free

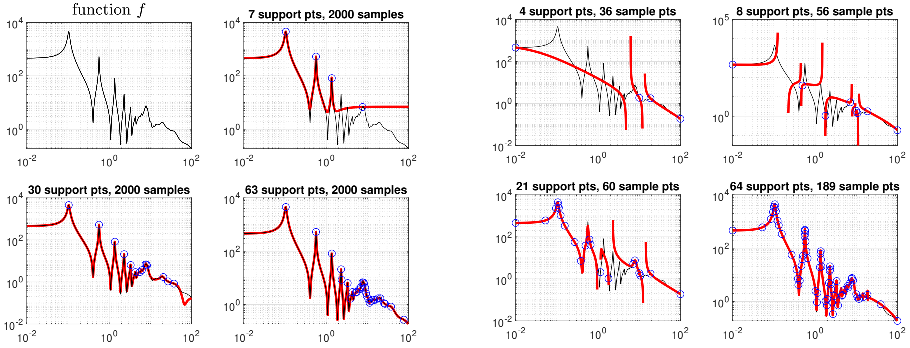
Example: `f = @(x) abs(x); r = aaax(f)`



- red: bad poles (in $[-1, 1]$) present
- green: output, pole-free on $[-1, 1]$

NICONET Beam example

$f(z) = c^T(zI - A)^{-1}b$, $A \in \mathbb{C}^{n\times n}$, $n = 348$; f expensive to evaluate



aaai succeeds with many fewer samples

Linearization and orthogonal polynomials

- Polynomial linearization entries are coefficients in the three-term recurrence, e.g. for Chebyshev

$$2xT_n(x) = T_{n+1}(x) + T_{n-1}(x)$$

- Partial fraction linearization: two-term recurrence $x\frac{1}{x-\gamma_i} = 1 + \frac{\gamma_i}{x-\gamma_i}$
- Continued fraction is historically connected to orthogonal polynomials (evaluation scheme via recursion), and the three-term recurrence gives linearization coefficients
- For non-orthogonal polynomial bases, C is dense (e.g. Hessenberg)

General rational function

$$r(x) = p(x) + \frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} + \cdots \frac{p_n(x)}{q_n(x)}$$

$$C = \begin{bmatrix} * & b_1 & & & & & \\ 1 & -a_1 & -b_2 & & & & \\ & & \ddots & \ddots & b_n & & \\ & & & 1 & -a_n & & \\ 1 & & & & & \gamma_1 & \\ 1 & & & & & & \gamma_n \end{bmatrix}$$

with eigenvector $Cx = \lambda x$, $\quad x = \begin{bmatrix} \text{(poly)} \\ \vdots \\ \frac{\frac{1}{\lambda+a_k}}{\lambda+a_{k-1}+\frac{b_k}{\lambda+a_k}} \\ \frac{1}{\lambda+a_{k-1}+\frac{b_k}{\lambda+a_k}} \\ \frac{1}{\lambda-\gamma_1} \\ \vdots \\ \frac{1}{\lambda-\gamma_n} \end{bmatrix}.$

- eig(C) = roots(r)

It is a strong linearization for polynomialization

$C(\lambda)$ is a linearization for polynomial $p(\lambda) \Leftrightarrow C = E(\lambda)\begin{bmatrix} p & \\ & I \end{bmatrix}F(\lambda)$ for unimodular E, F, *strong* linearization if $\mathrm{rev}(C) = \hat{E}(\lambda)\begin{bmatrix} \mathrm{rev}(p) & \\ & I \end{bmatrix}\hat{F}(\lambda)$

- The pencil belongs to $\mathbb{L}_1$ [Mackey, Mackey, Mehrmann, Mehl SIMAX 05], although in a nonstandard polynomial basis
- $\mathbb{L}_1$ pencil is a strong linearization iff regular [YN,VN,AT preprint]
- Since our pencils are regular, they are a strong linearization

C is a linearization for $r(x)\prod_{i=1}^{n} q_i(x)$ (i.e. mathematically equivalent to polynomializing, but numerically different)

Stability in matrix $\Rightarrow$ stability in polynomial?

The computed $\hat{x}_i$ are exact eigvals of a pertubed matrix pencil:

$$\{\hat{x}_i\} = \text{eig}(\lambda(X + \Delta X) + (Y + \Delta Y))$$

But stability in polynomial means $\hat{x}_i$ are exact roots of $p + \Delta p$:

$$p(x) + \Delta p(x) = \alpha \prod_{i=1}^{n} (x - \hat{x}_i), \quad ||\Delta p|| \le \epsilon ||p||$$

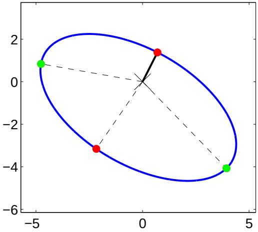
- For companion, Van Dooren and prove stability if QZ is used
- For comrade (Chebyshev + Jacobi polynomial bases), [YN and VN, Math. Comp.] proves stability, again if QZ is used (QR can be unstable)
- For rational linearization, stability is open problem

Applications: point-ellipsoid distance

Problem: find x on ellipsoid $(x-b)^T A^{-1}(x-b) = K^2$ closest to origin

- KKT conditions are

$$(x-b)^T A^{-1}(x-b) = K^2,$$
$$Ax = \lambda(x-b).$$



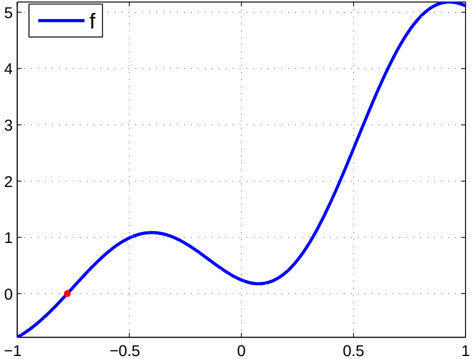
Writing $A = QDQ^T$ and $y := Q^T b$, this leads to solving for λ

$$K^2 = \sum_{i=1}^{n} \frac{d_i y_i^2}{(\lambda - d_i)^2}.$$

$$\Rightarrow \text{eigenvalues of } \begin{bmatrix} K^2 & 0 & -y_1 & 0 & -y_2 \\ 1 & d_1 & & & \\ & 1 & d_1 & & \\ 1 & & & d_2 & \\ & & & 1 & d_2 \end{bmatrix}, \text{ then } x = -(A - \lambda I)^{-1} b$$

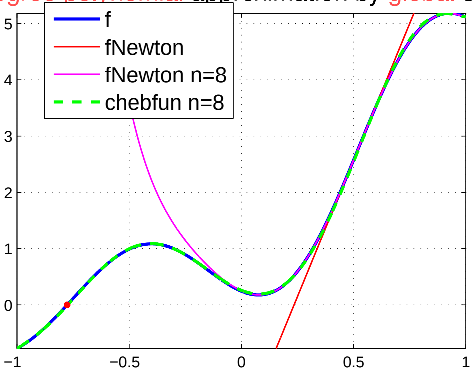
Take-home message

- Approximate with polynomials of high degree
  - Often considered inaccurate (Newton(quadratic) > Halley (cubic)) and unstable (Vandermonde matrix...)
  - If properly implemented: accurate (global approximation instead of local) and stable (Chebyshev basis + FFT)
- Replace algorithms based on linear approximation by local sampling with high-degree polynomial approximation by global sampling(?)



Take-home message

- Approximate with polynomials of high degree
  - Often considered inaccurate (Newton(quadratic) > Halley (cubic)) and unstable (Vandermonde matrix...)
  - If properly implemented: accurate (global approximation instead of local) and stable (Chebyshev basis + FFT)
- Replace algorithms based on linear approximation by local sampling with high-degree polynomial approximation by global sampling(?)



Explaining $\|f - p_n\| \leq O(\log n)\|f - p_{n,best}\|$: Lebesgue constants

- The Lebesgue constant Λ of $\{x_i\}_{i=0}^n$ is

$$\Lambda = \sup_f \frac{\|\mathcal{L}_n f\|_\infty}{\|f\|_\infty} \tag{2}$$

  Interpretation: Given data values on an $(n+1)$-point grid from sampling $\|f\|_\infty = 1$, Λ is the largest possible value of the interpolant p.

- Λ is an accurate measure of the interpolation points $\{x_i\}_{i=0}^n$:

$$\|f - p_n\|_\infty \leq (\Lambda + 1)\|f - p_{n,best}\|_\infty.$$

- Characterization using Lagrange polynomials $\ell_j(x)$ defined by $\ell(x) = \prod_{i=0}^n (x - x_i)$, $\ell_i(x) = \frac{\ell(x)}{\ell'(x_i)(x-x_i)}$:

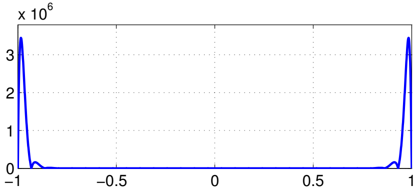
$$\Lambda = \sup_{x\in[-1,1]} \sum_{j=0}^n |\ell_j(x)|.$$

Explaining $\|f - p_n\| \le O(\log n)\|f - p_{n,best}\|$: Lebesgue function

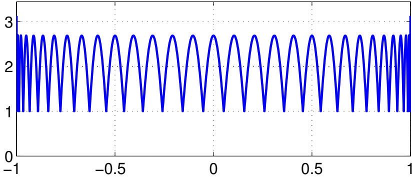
$$\Lambda = \sup_{x\in[-1,1]} \sum_{j=0}^{n} |\ell_j(x)|.$$

$\sum_{j=0}^{n} |\ell_j(x)|$ is called the Lebesgue function of $\{x_i\}_{i=0}^{n}$

$\{x_i\}_{i=0}^{n}$: Equispaced points



$\{x_i\}_{i=0}^{n}$: Chebyshev points



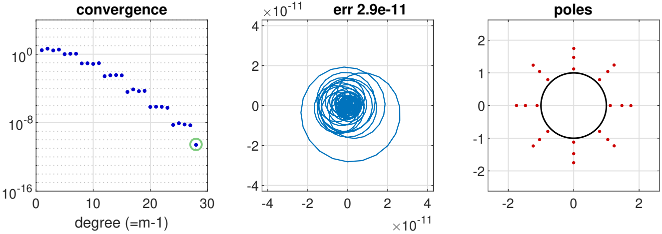
Matrix size

Resultant	Size of A_i	Degree	Size of $Cv = \lambda Ev$
Bézout (y first)	$\max(n_p, n_q)$	$m_p + m_q$	$\max(n_p, n_q)(m_p + m_q)$
Bézout (x first)	$\max(m_p, m_q)$	$n_p + n_q$	$\max(m_p, m_q)(n_p + n_q)$
Sylvester (y first)	$n_p + n_q$	$\max(m_p, m_q)$	$\max(m_p, m_q)(n_p + n_q)$
Sylvester (x first)	$m_p + m_q$	$\max(n_p, n_q)$	$\max(n_p, n_q)(m_p + m_q)$

Table: Sizes and degrees of matrix polynomials constructed from the Bézout and Sylvester resultant matrices. The product of the size of the A_i and degree is the size of the resulting generalized eigenvalue problem $Cv = \lambda Ev$, which depends on whether the x- or y-variable is solved for first. We use the Bézout resultant matrix and solve for the y-values first if $\max(n_p, n_q)(m_p + m_q) \leq \max(m_p, m_q)(n_p + n_q)$; the x-values first, otherwise.

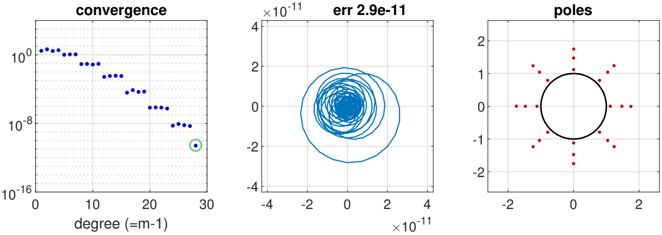
aaaz: on unit circle

Same idea, sample points on unit circle

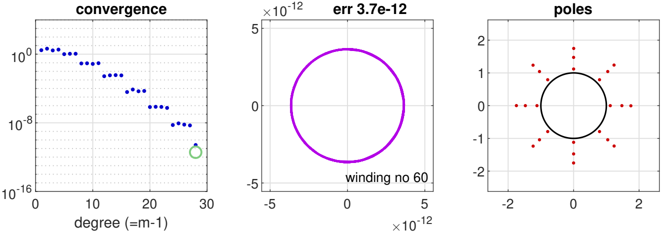


aaaz: on unit circle

Same idea, sample points on unit circle



- Poles allowed in $|z| < 1$ (mero=1) or disallowed (mero=0, here)
- with Lawson steps (find minimax approx, [N.-Trefethen 2020])



aaai: on imaginary axis

Use map

$$z = M\frac{1+w}{1-w}, \qquad w = \frac{z-M}{z+M}$$

- $z \in i\mathbb{R} \Leftrightarrow |w| = 1$
- Then apply aaaz to $f(w(z))$
- $M \in \mathbb{R}$ arbitrary, we set to 1.207