Low-Rank Approximation

Lecture 3 – Chebfun 2

Leonardo Robol, University of Pisa, Italy

Cagliari, 23–27 Sep 2019

Preliminaries

Until now, we have played with low-rank matrices; a theoretical mathematician might say that a matrix A is nothing else than a bivariate function:

$$A : \{1, \dots, m\} \times \{1, \dots, n\} \to \mathbb{R} \text{ (or } \mathbb{C}),$$

where we agree on the notation $A_{ij} := A(i, j)$.

Even if we like applied math, this suggests a natural questions: what if we have a bivariate function on a continuous domain — for instance $f(x, y) : [-1, 1]^2 \to \mathbb{R}$ — instead of a discrete one? Is it a "matrix", in an appropriate sense?

And obviously, the important questions: is it "low-rank"? Well, sometimes it is, and this is a quite powerful idea.

Scalar products

To jump into the continuous setting, we need some tools. Denote by:

$$\langle v, w \rangle : V \times V \to \mathbb{R}$$

any scalar product on some vector (Hilbert) space. As we all know, they induce a norm, and we have many examples of them:

- The canonical scalar product on $\mathbb{R}$: $\langle v, w \rangle := v^T w$, or $v^H w$ if we are on $\mathbb{C}^n$.
- On matrices, a scalar product is $\langle A, B \rangle := \mathrm{tr}(A^T B)$: this induces the Frobenius norm, and is equivalent to the canonical by identifying $\mathbb{R}^{m \times n}$ with $\mathbb{R}^{mn}$.
- The usual scalar product on $L^2([a, b])$, the functions whose square is integrable:

$$\langle f, g \rangle := \int_a^b f(x) g(x) \, dx.$$

- The previous scalar product, with some weight $w(x) \geq 0$:

$$\langle f, g \rangle := \int_a^b f(x) g(x) w(x) \, dx.$$

Orthogonal polynomials

We will concentrate on the interval $[-1, 1]$; given a scalar product, we may define orthogonal functions in an appropriate space:

$$f \perp g \iff \langle f, g \rangle = 0 \iff \int_{-1}^{1} f(x) g(x) w(x)\, dx = 0.$$

- Any choice of $w(x)$ gives rise to a family of orthogonal polynomials
- The family is generated by starting with $p_0(x) = 1$.
- ... and then we construct $p_{j+1}(x)$ as:

$$p_{j+1}(x) := x^{j+1} - \langle x^{j+1}, p_j \rangle \frac{p_j(x)}{\|p_j(x)\|^2} - \ldots - \langle x^{j+1}, p_0 \rangle \frac{p_0(x)}{\|p_0(x)\|^2}$$

- Clearly, $\{p_0, \ldots, p_j\}$ spans the vector space of polynomials of degree (at most) j.
- Different normalizations might be considered.

Not all weights are born equal

In practice, we will consider a special weight:

$$w(x) = \frac{1}{\sqrt{1 - x^2}}$$

- This choice generates the Chebyshev polynomials of the first kind.
- Incredibly relevant for approximation theory – and for numerical analysis in particular.
- As we will see, they are very well suited for quasi-optimal uniform approximation of functions.
- Obviously, up to a change of variable, we can always remap them from $[-1, 1]$ to any interval $[a, b]$ that is of interest.

Recurrence relation

- All the orthogonal polynomials satisfy a recurrence relation.
- By the definition, we know that the following procedure can be used to compute p_{j+1}:
  1. Compute $q_{j+1} := xp_j(x)$.
  2. Orthogonalize q_{j+1} against $p_0, \ldots, p_j$.
- In principle, this would give a formula for $p_j(x)$ of the form:

$$p_{j+1}(x) = xp_j(x) + \sum_{i=0}^{j} t_i p_i(x), \qquad t_i = -\frac{1}{\|p_i\|^2} \int_{-1}^{1} xp_j(x)p_i(x)w(x)\, dx$$

- However, most of the above terms are zeros (why?), and we can write:

$$p_{j+1}(x) = (\alpha_j x + \beta_j)p_j(x) - \gamma_j p_{j-1}(x).$$

- This is known as the three-term recurrence relation.

Recurrence relation for Chebyshev polynomials

For Chebyshev polynomials of the first kind, the recurrence relation is given by:

$$\begin{cases} T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x) \\ T_1(x) = x \\ T_0(x) = 1 \end{cases} .$$

As you might guess, there are also Chebyshev polynomials of the second kind, given by the recurrence relation:

$$\begin{cases} U_{j+1}(x) = 2xU_j(x) - U_{j-1}(x) \\ U_1(x) = 2x \\ U_0(x) = 1 \end{cases} .$$

It is a practical way if one wanted to compute the coefficients of the polynomials (almost never necessary!).

Orthogonal projection for polynomials

- Since we have a scalar product, it is natural to consider the orthogonal projection.
- For any polynomial $q(x)$, we can consider:

$$q_j := \frac{1}{\| T_j \|^2} \int_{-1}^{1} q(x) T_j(x) w(x) \, dx.$$

- Then, $q(x)$ can be written as the (finite!) sum

$$q(x) = \sum_{j=0}^{\infty} q_j T_j(x).$$

- We have $q_j = 0$ for any $j > \deg q(x)$.

Orthogonal projection for functions

- In fact, the orthogonal projector would be defined for all the functions in $L^\infty([-1,1])$:
$$f_j := \frac{1}{\|T_j\|^2} \int_{-1}^{1} T_j(x) f(x) w(x)\, dx.$$
- Natural question: under which conditions we have the following equality?
$$f(x) = \sum_{j=0}^{\infty} f_j T_j(x)$$
- We note that this summation is never finite — unless $f(x)$ is a polynomial.
- Even more importantly (for an applied mathematician), we may consider the truncated series $f_k(x)$:
$$f_k(x) := \sum_{j=0}^{k} f_j T_j(x)$$
Is it true that $f_k(x) \approx f(x)$?

Approximation properties

The answer to the previous questions is: mostly yes, the Chebyshev polynomials have very good approximation properties, essentially the same of the Fourier series.

There are several results in the literature — before stating one of interest for us, we will test this in practice on the following examples:

$$f(x) = \sin(x^2 - x + 1)$$

See: `example_chebfun1.m`

Speed of convergence

- The convergence of Chebyshev series is intimately connected with the regularity of the approximated function.
- Consider the following examples:

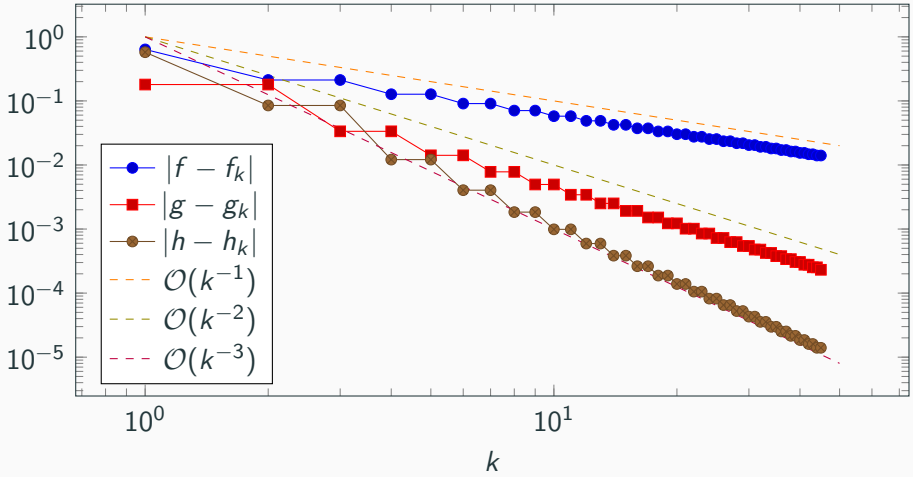
$$f(x) = |x| \qquad g(x) = |x| \cdot x \qquad h(x) = |x|^3$$

  - $f(x)$ is continuous — but its derivative has a jump at 0.
  - $g(x)$ is differentiable, but its second derivative has a jump.
  - And finally, $h(x)$ has a third derivative with a jump.
- We look at the maximum of $|f_k(x) - f(x)|$ over $[-1, 1]$, and analogously for $g(x)$ and $h(x)$.

Convergence speed

Here, f_k, g_k, h_k are the truncated Chebyshev series of the functions

$$f(x) = |x|, \qquad g(x) = |x| \cdot x, \qquad h(x) = |x|^3.$$



Coefficients decay

The convergence of the series can be easily related with the decay of the coefficients in the expansions. Indeed, let:

$$f(x) = \sum_{j=0}^{\infty} \alpha_j T_j(x), \qquad \alpha_j := \frac{1}{\|T_j\|^2} \int_{-1}^{1} f(x) T_j(x) w(x)\, dx.$$

Then, we can use that the Chebyshev polynomials are bounded between -1 and 1 on $[-1, 1]$, so:

$$|f(x) - f_k(x)| \leq \left| \sum_{j=k+1}^{\infty} \alpha_j T_j(x) \right| \leq \sum_{j=k+1}^{\infty} |\alpha_j|.$$

For instance, if we have $|\alpha_j| \leq C j^{-\nu}$, with $\nu > 1$, then:

$$|f(x) - f_k(x)| \leq \sum_{j=k+1}^{\infty} |\alpha_j| \leq C \sum_{j=k+1}^{\infty} j^{-\nu} \leq C \int_{k}^{\infty} \frac{1}{x^{\nu}}\, dx = C k^{-\nu+1}.$$

If the approximant converges as $k^{-\nu}$, we would like the coefficients to decay as $k^{-\nu-1}$.

Algebraic decay for C^ν functions

If a function has ν derivatives, and $f^{(\nu)}$ is nice enough – we have indeed convergence of the coefficients as $k^{-(\nu+1)}$.

Theorem
Assume $f(x)$ has $\nu - 1$ continuous derivatives, and $f^{(\nu)}$ has bounded variation. Then, for every $j > \nu$,

$$|\alpha_j| \leq C(j-\nu)^{-(\nu+1)}.$$

- Combined with the previous result, this gives $\mathcal{O}(k^{-\nu})$ convergence for $f_k \to f$, in the infinite norm, assuming $f^{(\nu)}$ of bounded variation.
- Bounded variation includes continuous functions, but allows for a few jumps: we just restrict that the sum of the height of all jumps is finite, i.e., $f \in BV([-1,1])$ if and only if

  $$\|f'(x)\|_1 < \infty,$$

  where the above is taken in a distributional sense, if necessary.

Analytic functions

- What can we say if $f(x)$ is analytic — and not only C^ν?
- Based on the previous result, we have that $|\alpha_j| \to 0$ faster than any polynomial.
- To understand this — we need to look at how the functions behave on the complex plane.

By using analytic continuation we might be able to extend the definition of our function $f(z)$ on a larger set. The larger set, the faster the convergence.

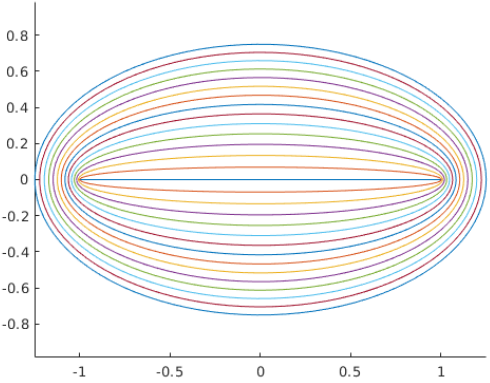
For Chebyshev series, the most natural sets to consider are Bernstein ellipses. In order to understand them, we need the Joukowski map:

$$J(z) := \frac{z + z^{-1}}{2}.$$

Note that $J(S^1) = [-1, 1]$.

The Joukowski map

- The Joukoski map maps S^1 (the unit circle) into $[-1, 1]$.
- z and z^{-1} are sent into the same spot.
- We call E_ρ the set $J(B(0, \rho))$.
- Circles are sent into ellipses! a few examples:



Bernstein ellipses

It is relatively easy to compute the maximum ρ that does not include the points $\pm i\beta$:

$$\frac{z+z^{-1}}{2} = \pm i\beta \iff \frac{i\rho+(i\rho)^{-1}}{2} = \pm i\beta \iff \frac{\rho^2}{2} - \beta\rho - \frac{1}{2} = 0.$$

Solving the quadratic equation yields

$$\rho = \beta + \sqrt{1+\beta^2}.$$

Theorem
If $f(z)$ is analytic on the Bernstein ellipse E_ρ, then

$$|\alpha_j| \le 2M\rho^{-j}, \qquad M := \max_{z\in E_\rho} |f(z)|.$$

In particular, $|f - f_k| \le 2M \sum_{j=k+1}^{\infty} \rho^{-j} = \frac{2M\rho^{k+1}}{1-\rho}$.

Let's check it in practice! See: `example_analytic_chebfun.m`

Converse result

Interestingly, the result also holds in the other direction!

Lemma
Let $f(z)$ a function whose Chebyshev series satisfies

$$|\alpha_j| \leq C\rho^{-j}$$

for some constant ρ. Then, $f(z)$ can be extended by analytic continuation on E_ρ.

Proof.
Apply the inverse of the Joukoski map into the inside of the circle, and use the Taylor expansion there (the condition on α_j automatically becomes a condition on the decay of Taylor coefficients). □

Interpolation: practical computation of the approximants

Computing the approximants $f_k(x)$ by the integral formula

$$\alpha_j = \frac{1}{\|T_j\|^2} \int_{-1}^{1} f(x) T_j(x) w(x)\ dx$$

is not practical. Instead, we typically prefer to compute the interpolant polynomial $\tilde{f}_k(x)$ at the points :

$$x_j := \cos\left(\frac{\pi j}{k+1}\right), \qquad j = 0, \ldots, k+1.$$

- These points are called Chebsyhev points of the second kind.
- They are the roots of $U_k(x)$ — with the addition of ± 1.
- They are also the image of the roots of the unity through the Joukoski map.

Fast interpolation

How much does it cost to interpolate the polynomial at those points?

- In a general context, interpolation requires the solution of a Vandermonde system — typicall $\mathcal{O}(k^3)$ and potentially very ill-conditioned.
- Instead, the fact that the points are image of the roots of the unity through the Joukoski map allows to design a fast trasform — based on the FFT — this is known as the discrete cosine transform, or DCT.
- The interpolant is computed in $\mathcal{O}(k \log)$ flops, with a normwise stable procedure.
- This is the main reason why Chebyshev polynomials are used in Chebfun, in place of other families of orthogonal polynomials (together with the quasi-optimal approximation properties).

Rootfinding

In order to understand the potential features that we investigate for 2D functions — we consider the following problem: given a polynomial $p(x)$ expressed in the Chebyshev basis, how do we compute its roots?

We want to solve $p(x) = 0$, for a polynomial of degree k.

- This will allow to compute zeros of functions too — if $|f - f_k| \leq \epsilon$ than the roots of $f_k(x)$ inside $[-1, 1]$ are good approximations of the ones of $f(x)$!
- In addition, this can be used to compute the roots of $f'(x)$ as well (since the latter is computed easily by the polynomial expansion), with applications to global optimization; more on this later!

Reminder: for Chebyshev polynomials, we have the recurrence relation

$$\begin{cases} T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x) \\ T_1(x) = x \\ T_0(x) = 1 \end{cases} .$$

The colleague matrix

Consider the following $(k-1) \times k$ matrix pencil, multiplied by a polynomial vector:

$$\mathcal{L}(x)v(x) = \begin{bmatrix} 1 & -2x & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2x & 1 \\ & & & 1 & -x \end{bmatrix} \begin{bmatrix} T_{k-1}(x) \\ \vdots \\ T_1(x) \\ T_0(x) \end{bmatrix} = 0.$$

- Related to the concept of dual minimal bases (not for today).
- Essentially encodes the recurrence relation; if we choose $T_0(x) = 1$ then imposing the duality between $\mathcal{L}(x)$ and $v(x)$ automatically determines the basis.
- Can be used as a building block to transform the rootfinding problem into an eigenvalue problem.

The colleague matrix

Consider the following square matrix pencil, obtained by adding a top row:

$$\begin{bmatrix} \alpha_{k-1} + 2x\alpha_k & \alpha_{k-2} - \alpha_k & \alpha_{k-3} & \dots & \alpha_0 \\ 1 & -2x & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2x & 1 \\ & & & 1 & -x \end{bmatrix} \begin{bmatrix} T_{k-1}(x) \\ T_{k-2} \\ \vdots \\ T_1(x) \\ T_0(x) \end{bmatrix} = \begin{bmatrix} s(x) \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix},$$

where $s(x) = \alpha_k(2xT_{k-1}(x) - T_{k-2}(x)) + \sum_{j=0}^{k-1} \alpha_j T_j(x)$.

- Since $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, we have $s(x) = \sum_{j=0}^{k} \alpha_j T_j(x)$.
- Clearly, the polynomial matrix is singular whenever $s(x)$ has a root at x.
- If we write the polynomial matrix as $A - xB$, the roots of $s(x)$ are exactly the eigenvalues of AB^{-1}, which can be computed in $\mathcal{O}(k^3)$ by the QR method.

The cost of eigenvalues

- If the approximant has a large degree, then k^3 could be a large cost.
- Quadratic methods are available for the task.
- As an even more efficient alternative, chebfun decomposes the domain into small parts until the degree is sufficiently small. This brings the cost down to $\approx k \log k$.

In general, computing the roots is quite efficient with Chebfun. Note that we only get the roots inside $[-1, 1]$! We do not get roots out of that interval, or in the complex plane.

Test: we can try with $\sin(5\pi x)$, which has roots at $\pm\frac{j}{5}$ for $j = 0, \ldots, 5$.

Global optimization

We can use the method for computing roots for global optimization:

- Given a chebfun f, we compute its derivative $f'(x)$ — also a chebfun.
- We compute the roots of $f'(x)$, which are the stationary points.
- We inspect them one by one, and find maxima and minima (plus we check the endpoints).

As before, we can try with the simple function $f(x) = \sin(5\pi x)$.

Going bi-dimensional

Assume we are given a function in two variables:

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

Then, we may expand it in a bivariate Chebyshev series:

$$f(x, y) = \sum_{i,j} \alpha_{ij} T_i(x) T_j(y).$$

- Theoretically, there is little change with respect to the previous approach.
- If we assume the regularity in the two variables, we have the decay in the α_{ij} for $i \to \infty$, and for $j \to \infty$ as well.
- In particular, only the coefficients for small enough i, j are relevant.

Function factorization

Given the truncated expansion

$$f_{mn}(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_{ij}\, T_i(x)\, T_j(y),$$

we can refactor it in "matrix form":

$$f_{mn}(x,y) = \begin{bmatrix} T_0(x) & \dots & T_m(x) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \dots & \alpha_{1n} \\ \vdots & & \vdots \\ \alpha_{m1} & \dots & \alpha_{mn} \end{bmatrix} \begin{bmatrix} T_0(y) & \dots & T_n(y) \end{bmatrix}^T .$$

- Somehow, this resembles the SVD: we have outer factors which are orthogonal (with respect to a different scalar product than the usual one), and a central core.
- However, the bases do not depend on $f(x,y)$, so this representation is probably redundant. Indeed, often the central matrix has low numerical rank. Why?

SVD for bivariate functions

Let $f(x, y) : [-1, 1]^2 \to \mathbb{R}$ be a function in $L^2([-1, 1])$. Then, we may define the integral operator:

$$T_f(g(x)) := \int_{-1}^{1} f(x, y) g(y)\, dy.$$

- The operator is linear, and maps L^2 into L^2.
- It is bounded, since $\| T_f \|_2 \leq \| f \|_2$.
- It is a compact operator: therefore, $T_f^T T_f$ has a discrete spectrum (possibly accumulating at 0), and can therefore be decomposed with an SVD.

Rank one operators

Since we are working in L^2, we can make use of the Riesz representation theorem to represent operators in the dual space $(L^2)^* \sim L^2$ as

$$\varphi_g(h) := \int_{-1}^{1} g(x)h(x)dx.$$

Therefore, rank 1 operators have the form $f(x)\varphi_g$ where $f, g \in L^2$; since the SVD is obtained as a sum of rank 1 terms of the form $\sigma_j f_j \varphi_{g_j}$ where $\|f_j\|_2 = \|g_j\|_2 = 1$, we have the decomposition:

$$T_f = \sigma_1 f_1 \varphi_{g_1} + \sigma_2 f_2 \varphi_{g_2} + \dots$$

Putting all the pieces together yields:

$$\begin{aligned} T_f(h) &= \int_{-1}^{1} f(x,y)h(y)\ dy = \sum_{j=0}^{\infty} \int_{-1}^{1} \sigma_j f_j(x) g_j(y) h(y)\ dy \\ &= \int_{-1}^{1} \sum_{j=1}^{\infty} \left(\sigma_j f_j(x) g_j(y)\right) h(y)\ dy \end{aligned}$$

Using the SVD to separate variables

The equality can be derived also in the L_2 sense, so we can write

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j f_j(x) g_j(y).$$

- As usual, this is the same in the complex case by adding appropriate conjugations.
- As it is, we only have convergence in L^2, and we know that $\sigma_j \to 0$ but not the speed.

Lemma
If $f(x,y)$ is also Lipschitz, then the above series convergeces uniformly, and therefore also pointwise. In addition, the factors $f_j(x)$ and $g_j(y)$ can be chosen continuous.

Practical approximation

How do we compute such expansion? We can try to follow our rules from yesterday:

- Is the problem low-dimensional enough that we can perform an SVD directly?

Practical approximation

How do we compute such expansion? We can try to follow our rules from yesterday:

- Is the problem low-dimensional enough that we can perform an SVD directly? No — not a viable solution (unless ...)
- Is it easy to perform a matrix vector multiplication by T_f or T_f^T? In this case this would mean computing

$$\int_{-1}^{1} f(x,y)g(y)\ dy, \qquad \text{or} \qquad \int_{-1}^{1} f(y,x)g(y)\ dy.$$

Practical approximation

How do we compute such expansion? We can try to follow our rules from yesterday:

- Is the problem low-dimensional enough that we can perform an SVD directly? No — not a viable solution (unless . . .)
- Is it easy to perform a matrix vector multiplication by T_f or T_f^T? In this case this would mean computing

$$\int_{-1}^{1} f(x,y)g(y)\ dy, \qquad \text{or} \qquad \int_{-1}^{1} f(y,x)g(y)\ dy.$$

  This might be problem dependent, but in general it seems quite difficult: no Lanczos or randomized sampling.
- Can we evaluate the matrix point-wise? In this case, this would be the equivalent of computing $A(i,j) = e_i^T A e_j = \langle e_i, Ae_j \rangle$, which means

$$\left\langle \delta(x - x_0), \int_{-1}^{1} f(x,y)\delta(y - y_0)\ dy \right\rangle,$$

  where δ is the Dirac delta: this is nothing else than $f(x_0, y_0)$! We can use ACA.

ACA in a functional setting

Recalling how ACA works, it can be easily adapted to work in this functional setting[1]

- We choose a pivot point (x_0, y_0) where $f(x, y)$ is maximum in modulus.
- We consider the rank 1 approximant

$$f_1(x, y) = f(x, y_0) f(x_0, y_0)^{-1} f(x_0, y).$$

- We subtract $f_1(x, y)$ from $f(x, y)$ and we continue the iteration, until the pivot point becomes too small.

Notice that this uses complete pivoting, so it would require to find the maximum of the function at each pivot choice.

[1] Indeed, ACA was first designed for the approximation of bivariate matrix functions, and the case of matrices came as an afterthought.

ACA in Chebfun

The actual algorithm implemented in Chebfun uses some heuristic:

- At the beginning, the function $f(x, y)$ is interpolated on a 9×9 grid, where $9 = 2^j + 1$ with $j = 3$.
- The rank of this approximant is determined by recompressing using an economy SVD. If it is larger than $2^{j-2} + 1$, j is increased to $j + 1$, and the function is interpolated on a larger grid going back to point 1.
- The first rough approximation will be used to choose the first pivot points $(x_1, y_1), \ldots, (x_k, y_k)$.
- The Gaussian elimination is started, choosing the pivot on the grid. The function in the various directions are resolved down to machine precision.
- The process is repeated on the residual function $f(x, y) - f_k(x, y)$ until convergence.

A few tests: `example_chebfun2.m`

Arithmetic operations

- Chebfun allows to perform arithmetic operations between chebfun and chebfun2 objects.
- For instance, one can sum, multiply, take square roots, ...
- Many operations are implemented by resampling the function.
- In some cases, it is possible to give a representation directly (example: $f + g$).
- In the case of chebfun2 there might be a need for recompression:

$$f \approx UV^T, \qquad g \approx WZ^T \implies f + g \approx [U\ V]\ [W\ Z]^T.$$

- The recompression algorithm is exactly the same as in the matrix case.

2D rootfinding

Rootfinding in 2D follows the same ideas of the 1D case, but is more delicate. We could have two meanings for "2D rootfinding":

- We determine $f(x, y) = 0$, which in general will be a level curve – and can be represented using a (piecewise) chebfun.
- Given two chebfuns, we look for solution to the system

$$\begin{cases} f_1(x, y) = 0 \\ f_2(x, y) = 0 \end{cases}.$$

  Generally, this set is finite, and we want to compute the points.

In the latter case, if f_1, f_2 are polynomials of degree m, n, by Bezout theorem we know that the system has mn solutions.

See: `example_rootfinding.m`

Optimization in 2D

As in the 1D case, one of the most interesting applications is global optimization of 2D functions. We may consider a function

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

and then compute the solution of the system of equations:

$$\begin{cases} \frac{\partial f}{\partial x}(x, y) = 0 \\ \frac{\partial f}{\partial y}(x, y) = 0 \end{cases}$$

- The partial derivatives can be computed with the command `diff`, or directly using `gradient`.
- They are also Chebfun — indeed, it is the same process of computing derivatives in 1D.

See: `example_optimization.m`