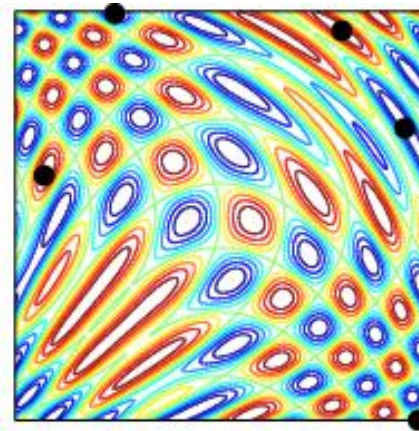


# From iterative Gaussian elimination to Chebfun2

Nick Trefethen and Alex Townsend  
Oxford



# Conjugate gradients for $Ax=b$ ( $n \times n$ )

Introduced in 1952 by Hestenes & Stiefel.

In theory, CG computes  $x$  exactly in  $n$  steps,  $O(n^3)$  operations.

This aspect of CG as a **direct method** was emphasized in the early years. But for the most part CG could not compete with other methods.

Wilkinson's 1965 magnum opus cites 148 references, H&S not among them.

Yet as H&S knew, CG also has an iterative aspect: it constructs successive approximations to  $x$  in a sequence of nested Krylov subspaces.

John Reid, 1971: "The method of conjugate gradients has several very pleasant features when regarded as an iterative method."

(Tractable  $n$  was getting larger by 1971, tipping the balance.)

In the years following Reid's paper, preconditioned iterations transformed scientific computing.

CG is now seen as the archetypical **iterative method**.

# Gaussian elimination for $Ax=b$

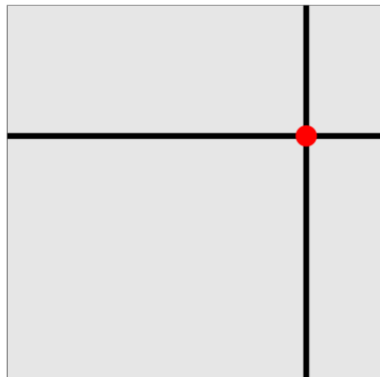
Roots in Chinese antiquity.

A **direct method**, finishing its work in  $n$  steps,  $O(n^3)$  operations.  
To this day, the standard algorithm for dense matrices.

Usual description: subtract rows from other rows to introduce zeros in  $A$ .

Alternative description: approximate  $A$  by  $A_1, A_2, \dots, A_n$  of rank  $1, 2, \dots, n$ .

Specifically, we consider GE with **complete pivoting**: at each step, first find the biggest entry  $a_{ij}$  of the remaining matrix, the "pivot".



## GE WITH COMPLETE PIVOTING

For  $k=1$  to  $n$ : Find biggest entry  $a_{ij}$  of  $A$

Set  $A = A - \text{col}_j \times \text{row}_i / a_{ij}$

After  $n$  steps,  $A$  is reduced to zero.

# Numerical example

17	5	17	19	8
1	9	5	21	7
3	12	10	14	4
8	20	15	9	13
3	6	13	11	8

0	0	0	0	0
0	0	0	0	0
0	0	2.30	0	-4.28
0	0	0	0	0
0	0	7.76	0	3.15

16.09	-3.14	12.47	0	1.67
0	0	0	0	0
2.33	6.00	6.67	0	-0.67
7.57	16.14	12.85	0	10.00
2.47	1.28	10.38	0	4.33

0	0	0	0	0
0	0	0	0	0
0	0	0	0	-5.22
0	0	0	0	0
0	0	0	0	0

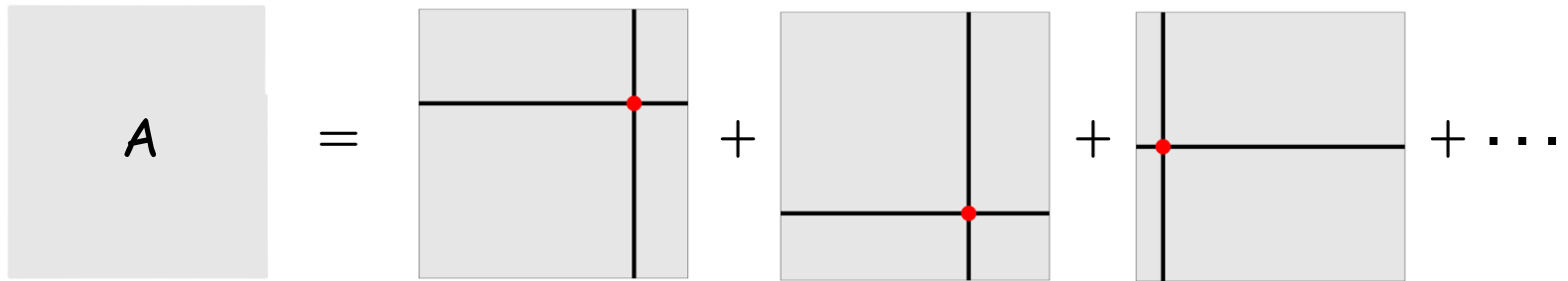
17.56	0	14.98	0	3.61
0	0	0	0	0
-0.48	0	1.88	0	-4.38
0	0	0	0	0
1.87	0	9.35	0	3.53

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

# GE as an iterative algorithm

What if we stop before step  $n$ ?

GE becomes an **iterative method** for constructing low-rank approxs of  $A$ .



- GE is thus a cheaper alternative to the SVD (singular value decomposition).

# Iterative GE in the literature

Variants of this idea, though without the name GE, have been appearing:

Bebendorf: **adaptive cross approximation**

Hackbusch: ***H*-matrices**

Tyrtysnikov: **skeletons, pseudoskeletons**

Martinsson & Rokhlin: **interpolative decompositions**

Mahoney & Drineas: **CUR decompositions**

Most of these are  
sometimes used  
hierarchically

Other contributors include Gesenhues, Goreinov, Grasedyck, Halko, Khoromskij, Liberty, Oseledets, Savostyanov, Tropp, Tytgert, Woolfe, Zamarashkin. The details vary widely (e.g., pivoting strategies).

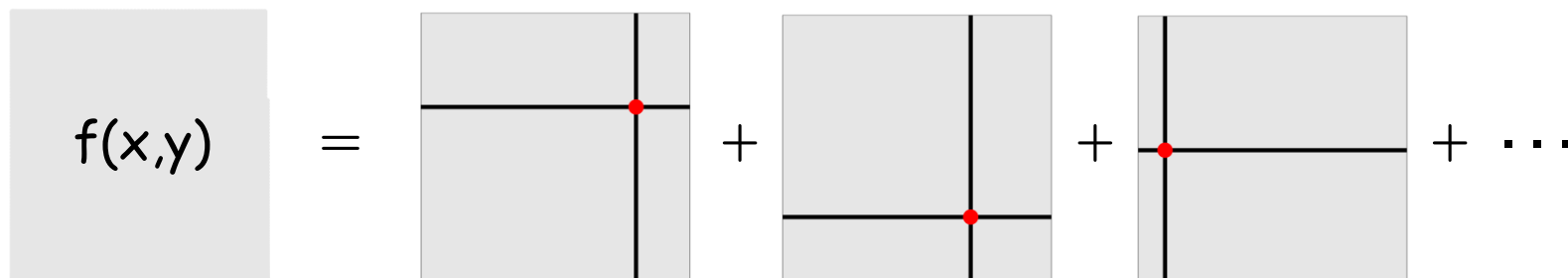
There are also links to rank-revealing factorization, model reduction, randomized principal component analysis, compressed sensing, Fast Multipole Method, matrix completion, the Netflix Prize,....

# GE for functions rather than matrices

Now suppose  $f(x,y)$  is a smooth function on a rectangle in the  $x$ - $y$  plane.

Motivated by Chebfun, we want efficient ways to manipulate such functions to 16-digit precision.

Idea: use GE in continuous mode to compress  $f$  to low rank.



Maple co-inventor Keith Geddes and his students have done such things for quadrature (cubature), calling the method **Geddes-Newton series**.

We like this method because:

- The approximations converge surprisingly fast (theory being developed);
- It leverages our well-developed 1D Chebfun technology

# Digression concerning

Chebfun is an open-source software system based on Matlab.  
Idea: overload Matlab's vectors & matrices to functions & operators.  
Computations are based on piecewise Chebyshev interpolants.  
Freely available. [Google chebfun](#).

V1 2004 Zachary Battles

V2 2008 Ricardo Pachón, Rodrigo Platte, Toby Driscoll

V3 2009 Rodrigo Platte, Toby Driscoll, Nick Hale,  
Ásgeir Birkisson, Mark Richardson

V4 2011 Nick Hale, Ásgeir Birkisson, Toby Driscoll, ...

V5 2013 (we hope)

... also Anthony Austin, Folkmar Bornemann, Pedro Gonnet, Stefan Güttel,  
Mohsin Javed, Sheehan Olver, Alex Townsend, Joris Van Deun,  
Kuan Xu, ...

demo



# Chebfun2 — moving Chebfun to 2D

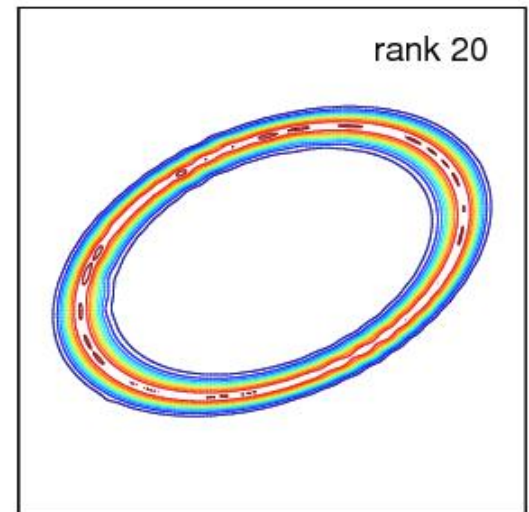
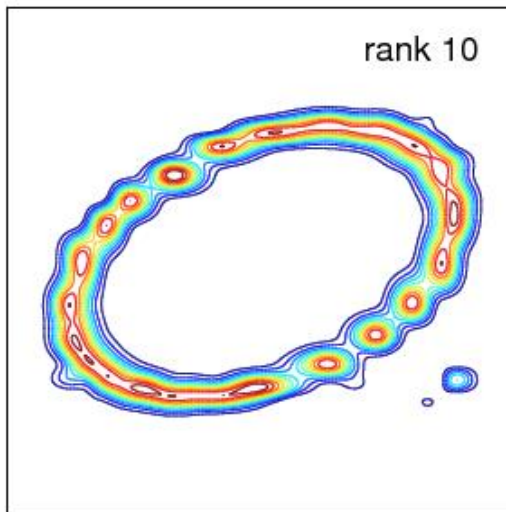
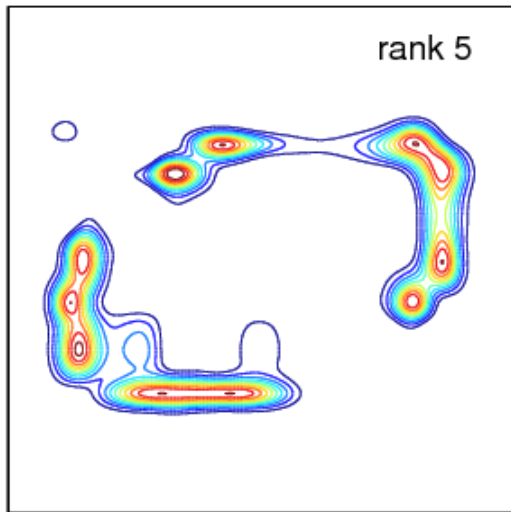
A project begun last year after much discussion, building on related work by the whole Chebfun team, especially Nick Hale.

DESIGN PRINCIPLES (in analogy to floating point arithmetic):

- Represent functions on rectangles to 16 digits by low-rank approxs
- After each operation like  $\exp(f)$  or  $f+g$ , trim the rank as far as possible
- Build on Chebfun's powerful 1D capabilities

# Example of low-rank approximation

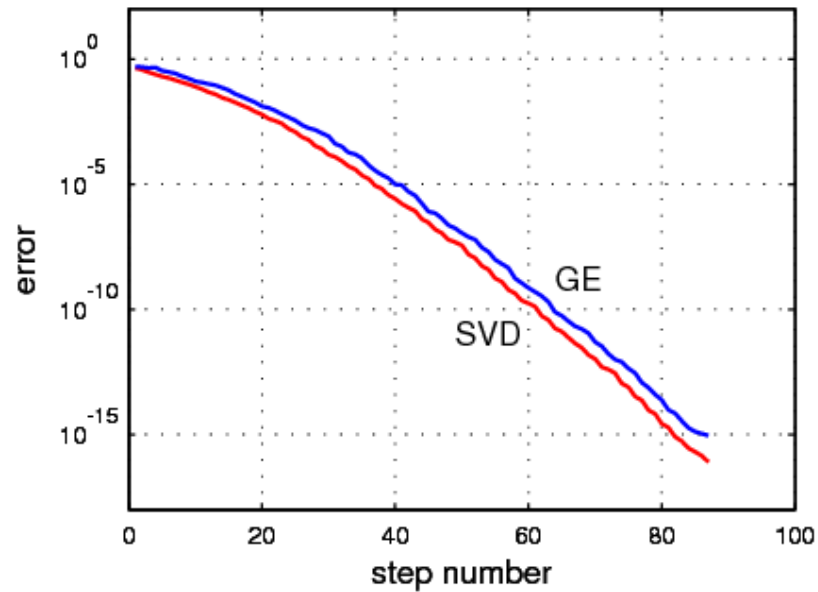
$$f(x,y) = \exp(-100(x^2 - xy + 2y^2 - \frac{1}{2})^2) \text{ on } [-1,1] \times [-1,1]$$



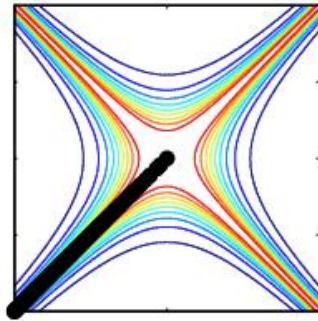
The final 16-digit approximation is of rank 88.

# Comparison of iterative GE and SVD

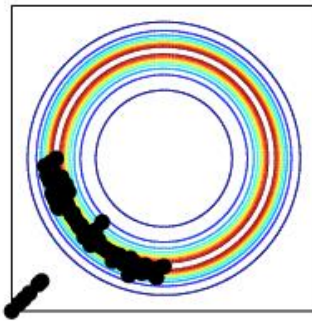
$f(x,y) = \exp(-100(x^2 - xy + 2y^2 - \frac{1}{2})^2)$  again



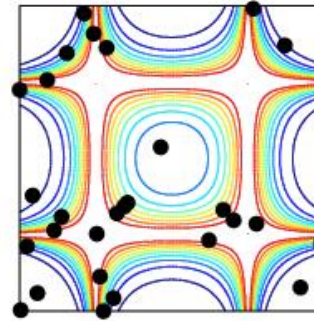
# More functions, showing pivot points



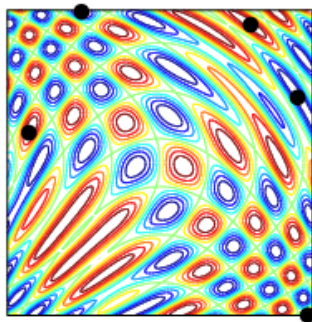
rank 125



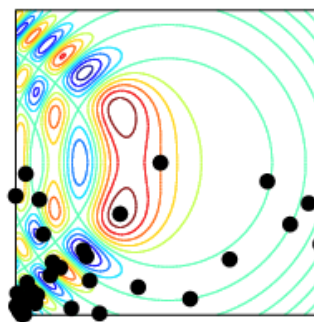
rank 65



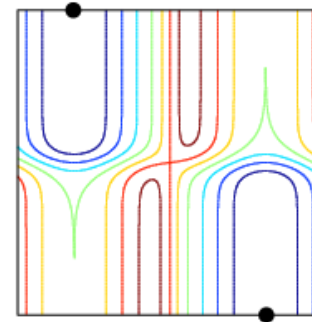
rank 28



rank 5



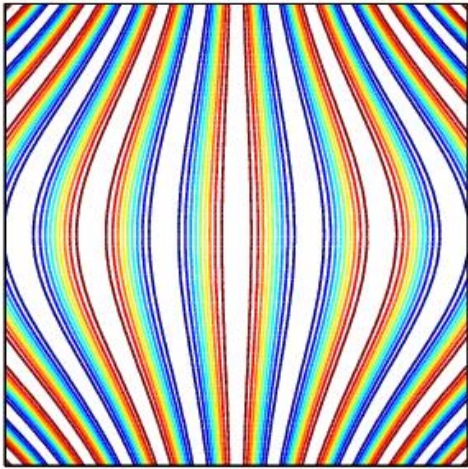
rank 33



rank 2

For efficiency, Chebfun2 doesn't actually do GE in purely "continuous mode" but uses an approximation starting from Chebyshev grids of size  $9 \times 9$ ,  $17 \times 17$ ,  $33 \times 33$ , ....

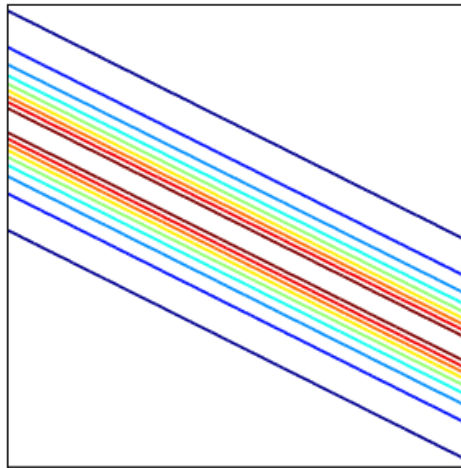
# Example of algebraic operations in Chebfun2



$$f(x,y) = \cos(10x(1+y^2))$$

rank 18

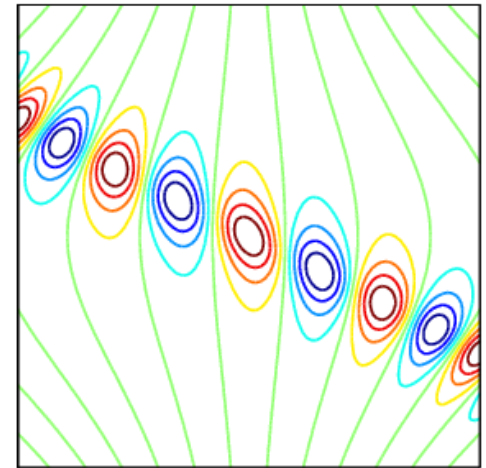
x



$$g(x,y) = (1+10(x+2y)^2)^{-1}$$

rank 85

=



$$h(x,y) = f(x,y)g(x,y)$$

rank 98  
(not 1530)

# Chebfun2 methods

```
>> methods chebfun2
```

```
Methods for class chebfun2:
```

abs	del2	integral2	min	quad2d	surfc
cheb2poly	diag	isempty	min2	rdivide	surfl
cheb2polyplot	diff	isequal	minandmax2	real	svd
chebfun2	display	isreal	minus	restrict	tan
chebpolyplot	eig	laplacian	mldivide	roots	tand
chol	exp	length	movie	schur	tanh
complex	feval	log	mrdivide	sign	times
conj	flipdim	log10	mtimes	sin	trace
contour	fliplr	lu	norm	sinh	transpose
contourf	flipud	max	pivotplot	sqrt	uminus
cos	fred	max2	pivots	squeeze	uplus
cosh	get	mean	plot	std2	vertcat
ctranspose	gradient	mean2	plot3	subsref	volt
cumprod	gsvd	median	plus	sum	waterfall
cumsum	horzcat	mesh	power	surf	wronskian
cur	imag	meshc	prod	surface	
dblquad	integral	meshz	qr	surfacearea	

# Vector functions ( $\rightarrow$ chebfun2v objects)

$f(x,y)$ : scalar field       $F(x,y)$ : vector field

Some operations:

$H = f * G$                       [scalar-vector product]

$h = F .* G$                       [dot product]

$h = F * G$                       [cross product]

$g = \text{div}(F)$

$G = \text{grad}(f)$

$g = \text{div}(\text{grad}(f)) = \text{laplacian}(f)$

$g = \text{curl}(F)$

$\text{plot}(f)$ ,  $\text{contour}(f)$

$\text{quiver}(F)$

# A summary of some Chebfun2 algorithms

Key theme always: exploit the 1D rows and columns and the "Chebyshev technology" they inherit from Chebfun

DIFFERENTIATION — from Chebyshev series for 1D interpolants  
`diff(f,k,1), diff(f,k,2)` [kth derivative wrt y, x]

INTEGRATION — Clenshaw-Curtis quadrature  
`sum(f,1), sum(f,2); sum2(f)` [integral wrt y, x; global integral]

MINIMA/MAXIMA — initial guess via `convhulln`, then Newton iteration  
`max(f,[],1), max(f,[],2), max2(f)` [maxima along y, x slices; global maximum]

SCALAR ZEROFINDING — initial guess via `contourc`, then Newton iteration  
`roots(f)` [curves]

VECTOR ZEROFINDING — 2D subdivision, regularization, & resultants  
`roots(F)` [points]

demo



# Ahead

- Theory of convergence of low-rank approximations
- Partial differential equations
- Numerical linear algebra of operators (what is `chol(f)`?)
- PhD for Alex



# Maybe later

- Singularities
- Non-rectangular domains
- 3D

Thank you!

## Our style

Chebfun grew slowly, aiming to do numerical calculation with 1D functions reliably to 16-digit precision.

ODES — a big success — came later.

Chebfun2 too has a general-purpose, high-precision aim: numerical computation with functions in 2D. This is not a CFD package.