The automatic solution of PDEs using a global spectral method

2014 CBMS-NSF Conference, 29th June 2014

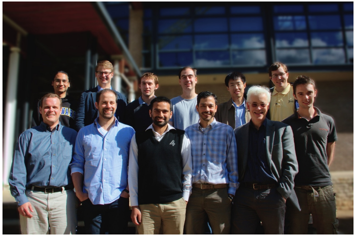
Alex Townsend
PhD student
University of Oxford

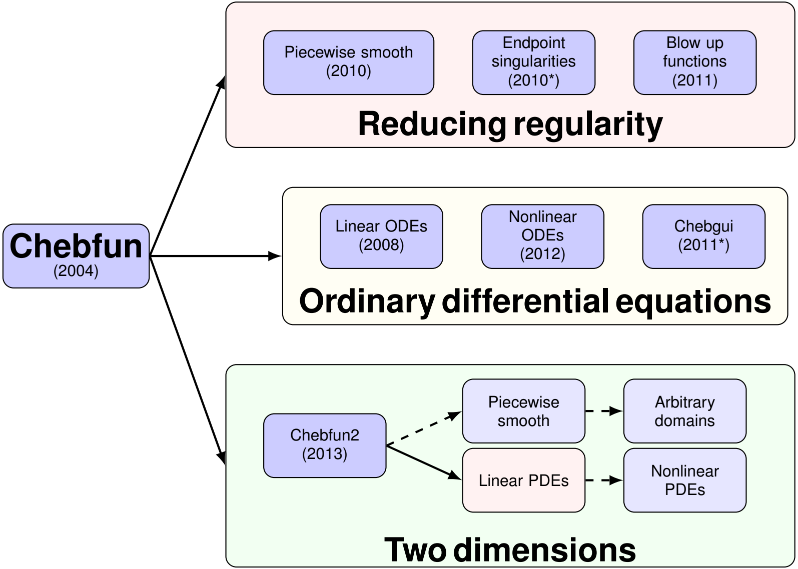
(with Sheehan Olver)

Supervised by Nick Trefethen, University of Oxford.

Supported by EPSRC grant EP/P505666/1.

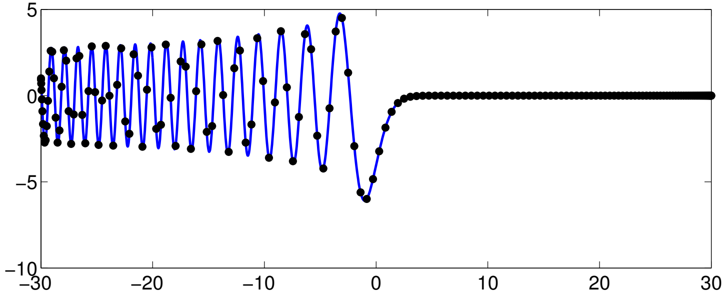
Introduction
Chebfun





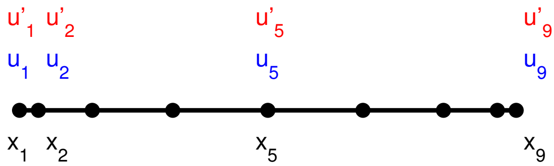
In 2008: Overload the MATLAB backslash command \ for operators [Driscoll, Bornemann, & Trefethen 2008].

```
L = chebop(@(x,u) diff(u,2)-x.*u,[-30 30]); % Airy equation
L.lbc = 1; L.rbc = 0;                        % Set boundary conditions
u = L \ 0; plot(u)                           % Solve and plot
```



Introduction
Spectral collocation basics

Given values on a grid, what are the values of the derivative on that same grid?:



$$D_n \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} u_1' \\ \vdots \\ u_n' \end{pmatrix}, \qquad D_n = \texttt{diffmat(n)}.$$

For example, $u'(x) + cos(x)u(x)$ is represented as

$$L_n = D_n + \text{diag}\left(cos(x_1), \ldots, cos(x_n)\right) \in \mathbb{R}^{n \times n}.$$

Introduction
Why do spectral methods get a bad press?

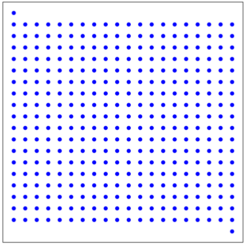
1. Dense matrices.
2. Ill-conditioned matrices.
3. When has it converged? Tricky.

See, for example: [Canuto et al. 07], , [Fornberg 98], [Trefethen 00].

Introduction

Why do spectral methods get a bad press?

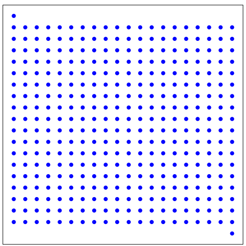
1. Dense matrices.
2. Ill-conditioned matrices.
3. When has it converged? Tricky.

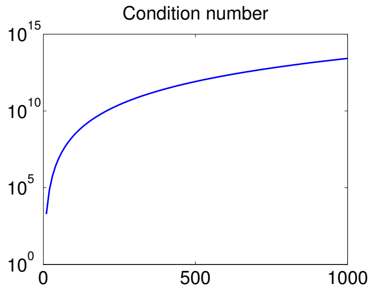


See, for example: [Canuto et al. 07], , [Fornberg 98], [Trefethen 00].

Why do spectral methods get a bad press?

1. Dense matrices.
2. Ill-conditioned matrices.
3. When has it converged? Tricky.



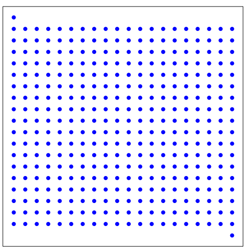


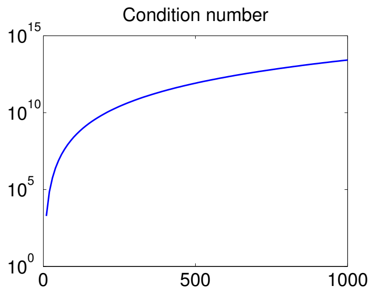
See, for example: [Canuto et al. 07], , [Fornberg 98], [Trefethen 00].

Introduction

Why do spectral methods get a bad press?

1. Dense matrices.
2. Ill-conditioned matrices.
3. When has it converged? Tricky.





See, for example: [Canuto et al. 07], , [Fornberg 98], [Trefethen 00].

A fast and well-conditioned spectral method
Differentiation operator

Work with coefficients: Spectral methods do not have to result in dense, ill-conditioned matrices. (Just don't discretize the differentiation operator faithfully.)

The idea is to use simple relations between Chebyshev polynomials:

$$\frac{dT_k}{dx} = \begin{cases} kU_{k-1}, & k \geq 1, \\ 0, & k = 0, \end{cases} \qquad T_k = \begin{cases} \frac{1}{2}\left(U_k - U_{k-2}\right), & k \geq 2, \\ \frac{1}{2}U_1, & k = 1, \\ U_0, & k = 0. \end{cases}$$

$$\mathcal{D} = \begin{pmatrix} 0 & 1 & & & \\ & & 2 & & \\ & & & 3 & \\ & & & & \ddots \end{pmatrix}, \qquad \mathcal{S} = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ & & \frac{1}{2} & 0 & -\frac{1}{2} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Olver & T., A fast and well-conditioned spectral method, SIAM Review, 2013.

A fast and well-conditioned spectral method
Multiplication operator

$$T_j T_k = \frac{1}{2} T_{|j-k|} + \frac{1}{2} T_{j+k}$$

$$\mathcal{M}[\boldsymbol{a}] = \frac{1}{2} \underbrace{\begin{pmatrix} 2a_0 & a_1 & a_2 & a_3 & \cdots \\ a_1 & 2a_0 & a_1 & a_2 & \ddots \\ a_2 & a_1 & 2a_0 & a_1 & \ddots \\ a_3 & a_2 & a_1 & 2a_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}}_{\text{Toeplitz}} + \frac{1}{2} \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ a_1 & a_2 & a_3 & a_4 & \iddots \\ a_2 & a_3 & a_4 & a_5 & \iddots \\ a_3 & a_4 & a_5 & a_6 & \iddots \\ \vdots & \iddots & \iddots & \iddots & \ddots \end{pmatrix}}_{\text{Hankel + rank-1}}$$

Multiplication is not a dense operator in finite precision. It is **m-banded**:

$$a(x) = \sum_{k=0}^{\infty} a_k T_k(x) = \sum_{k=0}^{m} \tilde{a}_k T_k(x) + \mathcal{O}(\epsilon),$$

A fast and well-conditioned spectral method

What about this new spectral method?

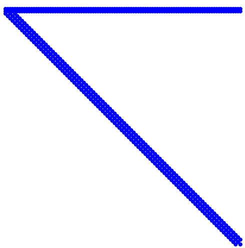
1. Almost banded matrices.
2. Well-conditioned matrices.
3. When has it converged? Trivial.

Other approaches: [Clenshaw 57], [Greengard 91], [Shen 03].

A fast and well-conditioned spectral method

What about this new spectral method?

1. Almost banded matrices.
2. Well-conditioned matrices.
3. When has it converged? Trivial.

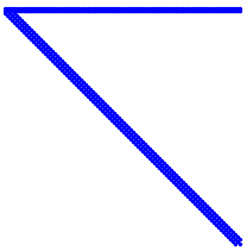


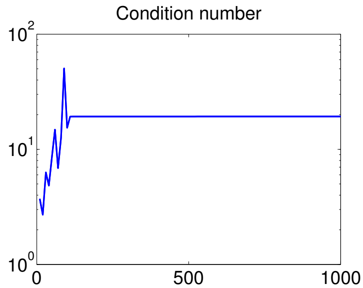
Other approaches: [Clenshaw 57], [Greengard 91], [Shen 03].

A fast and well-conditioned spectral method

What about this new spectral method?

1. Almost banded matrices.
2. Well-conditioned matrices.
3. When has it converged? Trivial.



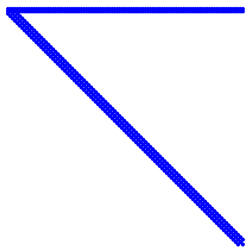


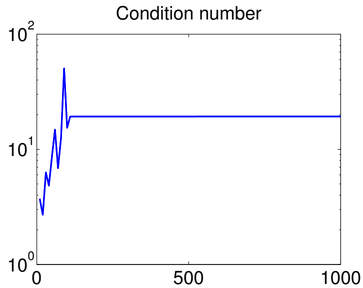
Other approaches: [Clenshaw 57], [Greengard 91], [Shen 03].

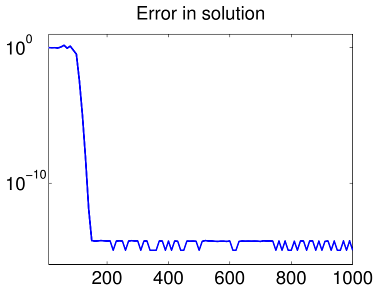
A fast and well-conditioned spectral method

What about this new spectral method?

1. Almost banded matrices.
2. Well-conditioned matrices.
3. When has it converged? Trivial.







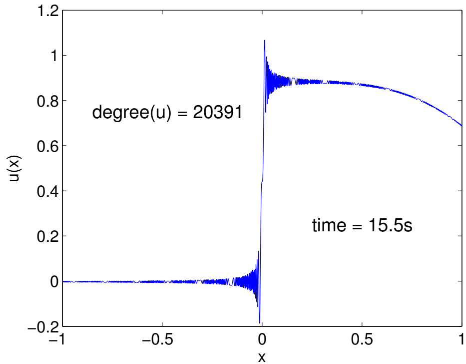
Other approaches: [Clenshaw 57], [Greengard 91], [Shen 03].

A fast and well-conditioned spectral method
First example

$$u'(x) + x^3 u(x) = 100\sin(20{,}000x^2), \qquad u(-1) = 0.$$

The exact solution is

$$u(x) = e^{-\frac{x^4}{4}}\left(\int_{-1}^{x} 100e^{\frac{t^4}{4}}\sin(20{,}000t^2)dt\right).$$



- `N = chebop(@(x,u) ···); N.lbc = 0; u = N \ f;`
- Adaptively selects the discretisation size.
- Forms a chebfun object [Chebfun V4.2].
- $\|\tilde{u} - u\|_\infty = 1.5 \times 10^{-15}$.

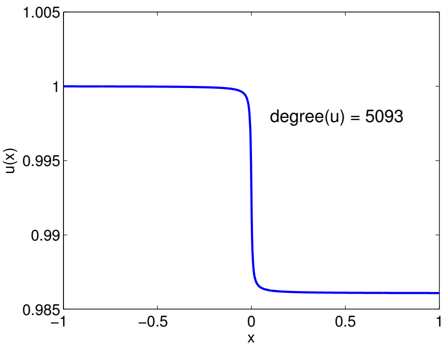
A fast and well-conditioned spectral method

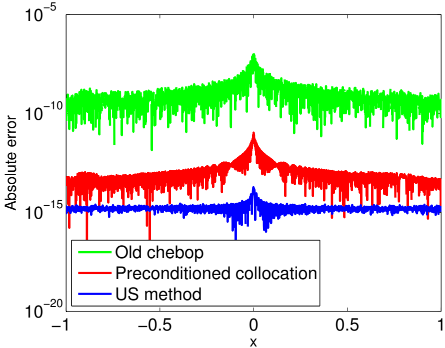
Another example

$$u'(x) + \frac{1}{1 + 50{,}000x^2} u(x) = 0, \qquad u(-1) = 1.$$

The exact solution with $a = 50{,}000$ is

$$u(x) = \exp\left(-\frac{\tan^{-1}(\sqrt{a}x) + \tan^{-1}(\sqrt{a})}{\sqrt{a}}\right).$$



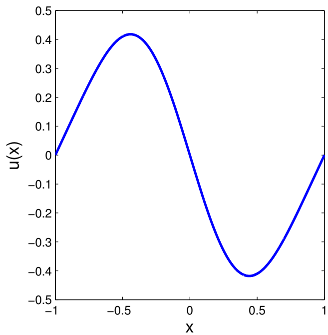


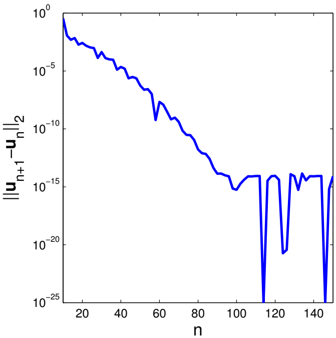
A fast and well-conditioned spectral method

A high-order example

$$u^{(10)}(x) + \cosh(x)u^{(8)}(x) + \cos(x)u^{(2)}(x) + x^2u(x) = 0$$

$$u(\pm 1) = 0,\ u'(\pm 1) = 1,\ u^{(k)}(\pm 1) = 0,\ k = 2, 3, 4.$$



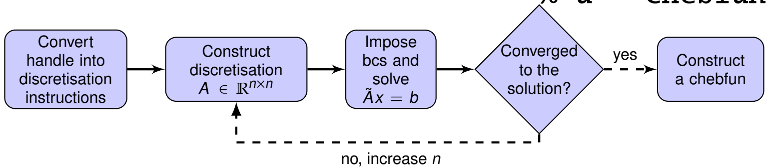


$$\left(\int_{-1}^{1} (\tilde{u}(x) + \tilde{u}(-x))^2\right)^{\frac{1}{2}} = 1.3 \times 10^{-14}.$$

Chebop and Chebop2

Convenience for the user

```
L = chebop(@(x,u) diff(u,2)-x.*u,[-30 30]); % Airy equation
L.lbc = 1; L.rbc = 0;                       % Set boundary conditions
u = L \ 0;                                  % u = chebfun
```

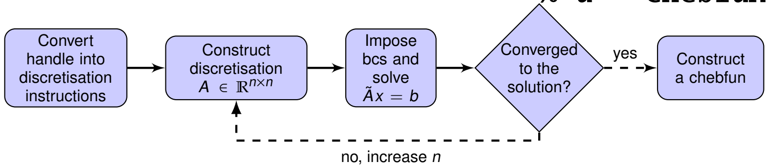



```
L = chebop2(@(x,y,u) laplacian(u)+(1000+y)*u);% Helmholtz with gravity
L.lbc = 1; L.rbc = 1; L.ubc = 1; L.dbc = 1;% Set boundary conditions
u = L \ 0;                                  % u = chebfun2
```

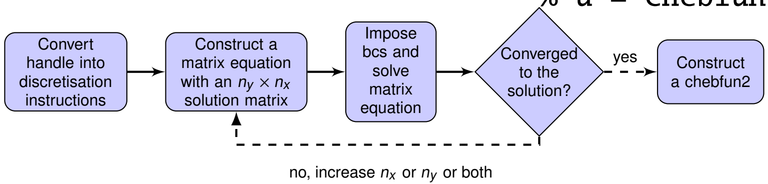
Chebop and Chebop2

Convenience for the user

```
L = chebop(@(x,u) diff(u,2)-x.*u,[-30 30]); % Airy equation
L.lbc = 1; L.rbc = 0;                       % Set boundary conditions
u = L \ 0;                                  % u = chebfun
```




```
L = chebop2(@(x,y,u) laplacian(u)+(1000+y)*u);% Helmholtz with gravity
L.lbc = 1; L.rbc = 1; L.ubc = 1; L.dbc = 1;% Set boundary conditions
u = L \ 0;                                  % u = chebfun2
```

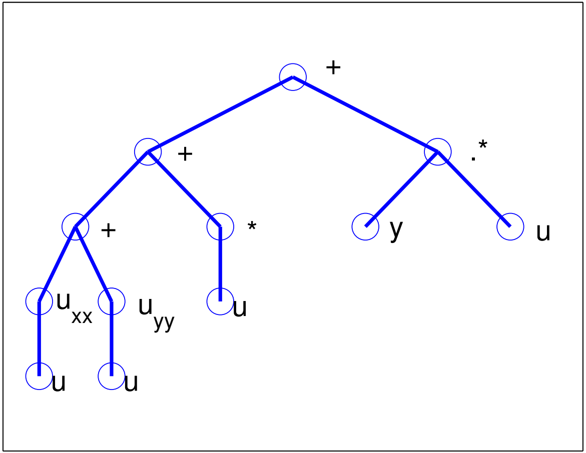


Interpreting user-defined input
Automatic differentiation

- Implemented by forward-mode operator overloading
- Interpret anonymous function as a sequence of elementary operations
- Can also calculate Fréchet derivatives

Key people:
Ásgeir Birkisson and Toby Driscoll

$u_{xx} + u_{yy} + 50u + yu$



Low rank approximation
Numerical rank

For $A \in \mathbb{C}^{m\times n}$, SVD gives best rank k wrt 2-norm [Eckart & Young 1936]

$$A = \sum_{j=1}^{\min(m,n)} \sigma_j u_j v_j^* \approx \sum_{j=1}^{k} \sigma_j u_j v_j^*, \qquad \sigma_{k+1} < \text{tol}.$$

For Lipschitz smooth bivariate functions [Schmidt 1909, Smithies 1937]

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y) v_j(x) \approx \sum_{j=1}^{k} \sigma_j u_j(y) v_j(x).$$

For compact linear operators acting on functions of two variables,

$$\mathcal{L} \stackrel{!}{=} \sum_{j=1}^{\infty} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x \approx \sum_{j=1}^{k} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x.$$

Low rank approximation
Numerical rank

For $A \in \mathbb{C}^{m \times n}$, SVD gives best rank k wrt 2-norm [Eckart & Young 1936]

$$A = \sum_{j=1}^{\min(m,n)} \sigma_j u_j v_j^* \approx \sum_{j=1}^{k} \sigma_j u_j v_j^*, \qquad \sigma_{k+1} < \text{tol}.$$

For Lipschitz smooth bivariate functions [Schmidt 1909, Smithies 1937]

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y) v_j(x) \approx \sum_{j=1}^{k} \sigma_j u_j(y) v_j(x).$$

For compact linear operators acting on functions of two variables,

$$\mathcal{L} \stackrel{!}{=} \sum_{j=1}^{\infty} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x \approx \sum_{j=1}^{k} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x.$$

Low rank approximation
Numerical rank

For $A \in \mathbb{C}^{m\times n}$, SVD gives best rank k wrt 2-norm [Eckart & Young 1936]

$$A = \sum_{j=1}^{\min(m,n)} \sigma_j u_j v_j^* \approx \sum_{j=1}^{k} \sigma_j u_j v_j^*, \qquad \sigma_{k+1} < \text{tol}.$$

For Lipschitz smooth bivariate functions [Schmidt 1909, Smithies 1937]

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y) v_j(x) \approx \sum_{j=1}^{k} \sigma_j u_j(y) v_j(x).$$

For compact linear operators acting on functions of two variables,

$$\mathcal{L} \overset{!}{=} \sum_{j=1}^{\infty} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x \approx \sum_{j=1}^{k} \sigma_j \mathcal{L}_j^y \otimes \mathcal{L}_j^x.$$

Low rank approximation

Do the low rank stuff before discretization

Low rank-then-discretize: Instead of low rank techniques after discretization, do them before.

For example, Helmholtz is of rank 2

$$\nabla^2 u + K^2 u = \left(u_{xx} + \frac{K^2}{2}u\right) + \left(u_{yy} + \frac{K^2}{2}u\right) = \left(\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}\right) \otimes \mathcal{I} + \mathcal{I} \otimes \left(\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}\right).$$

Let A be your favourite ODE discretization of $\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}$, then (typically)

$$AXI + IXA^T.$$

In general, if $\mathcal{L}$ is of rank k we have

$$\sum_{j=1}^{k} A_j X B_j^T = F$$

Low rank approximation
Do the low rank stuff before discretization

Low rank-then-discretize: Instead of low rank techniques after discretization, do them before.

For example, Helmholtz is of rank 2

$$\nabla^2 u + K^2 u = (u_{xx} + \frac{K^2}{2}u) + (u_{yy} + \frac{K^2}{2}u) = (\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}) \otimes \mathcal{I} + \mathcal{I} \otimes (\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}).$$

Let A be your favourite ODE discretization of $\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}$, then (typically)

$$AXI + IXA^T.$$

In general, if $\mathcal{L}$ is of rank k we have

$$\sum_{j=1}^{k} A_j X B_j^T = F$$

Low rank approximation
Do the low rank stuff before discretization

Low rank-then-discretize: Instead of low rank techniques after discretization, do them before.

For example, Helmholtz is of rank 2

$$\nabla^2 u + K^2 u = (u_{xx} + \frac{K^2}{2}u) + (u_{yy} + \frac{K^2}{2}u) = (\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}) \otimes \mathcal{I} + \mathcal{I} \otimes (\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}).$$

Let A be your favourite ODE discretization of $\mathcal{D}^2 + \frac{K^2}{2}\mathcal{I}$, then (typically)

$$AXI + IXA^T.$$

In general, if $\mathcal{L}$ is of rank k we have

$$\sum_{j=1}^{k} A_j X B_j^T = F$$

Low rank approximation

Computing the rank of a partial differential operator

Recast differential operators as polynomials: Once you have polynomials computing the rank is easy.

The rank of

$$\mathcal{L} = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(x, y) \frac{\partial^i}{\partial y^i} \frac{\partial^j}{\partial x^j}$$

equals a TT-rank [Oseledets 2011] (between $\{x, s\}$ and $\{y, t\}$) of

$$h(x, s, y, t) = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(s, t) y^i x^j = \sum_{j=1}^{k} c_j(t, y) r_j(s, x).$$

Rank 1:
ODEs
Trivial PDEs

Rank 2:
Laplace, Helmholtz
Transport, Heat, Wave
Black-Scholes

Rank 3:
Biharmonic
Lots here.

Low rank approximation

Computing the rank of a partial differential operator

Recast differential operators as polynomials: Once you have polynomials computing the rank is easy.

The rank of

$$\mathcal{L} = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(x,y) \frac{\partial^i}{\partial y^i} \frac{\partial^j}{\partial x^j}$$

equals a TT-rank [Oseledets 2011] (between $\{x, s\}$ and $\{y, t\}$) of

$$h(x,s,y,t) = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(s,t) y^i x^j = \sum_{j=1}^{k} c_j(t,y) r_j(s,x).$$

Rank 1:
ODEs
Trivial PDEs

Rank 2:
Laplace, Helmholtz
Transport, Heat, Wave
Black-Scholes

Rank 3:
Biharmonic
Lots here.

Low rank approximation
Computing the rank of a partial differential operator

Recast differential operators as polynomials: Once you have polynomials computing the rank is easy.

The rank of

$$\mathcal{L} = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(x,y) \frac{\partial^i}{\partial y^i} \frac{\partial^j}{\partial x^j}$$

equals a TT-rank [Oseledets 2011] (between $\{x, s\}$ and $\{y, t\}$) of

$$h(x,s,y,t) = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} a_{ij}(s,t) y^i x^j = \sum_{j=1}^{k} c_j(t,y) r_j(s,x).$$

Rank 1:
ODEs
Trivial PDEs

Rank 2:
Laplace, Helmholtz
Transport, Heat, Wave
Black-Scholes

Rank 3:
Biharmonic
Lots here.

Low rank approximation

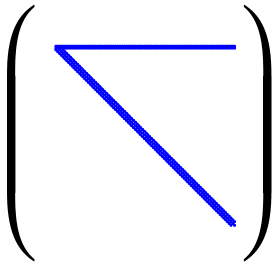
Construct a n_x by n_y generalised Sylvester matrix equation

If the PDE is $\mathcal{L}u = f$, where $\mathcal{L}$ is of rank-k then we solve for $X \in \mathbb{C}^{n_y \times n_x}$ in,

$$\sum_{j=1}^{k} \sigma_j A_j X B_j^T = F, \qquad A_j \in \mathbb{C}^{n_y \times n_y}, \quad B_j \in \mathbb{C}^{n_x \times n_x}.$$

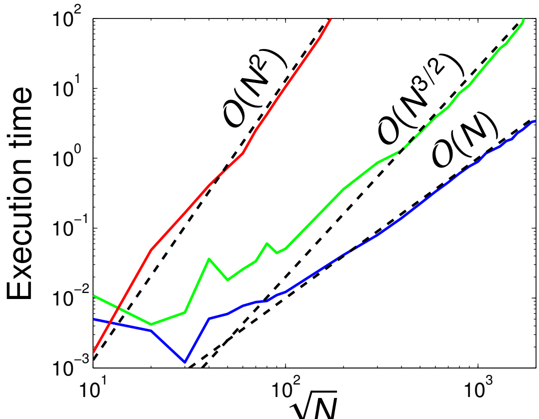
$X =$ solution's coefficients $\qquad$ A_j, B_j = 1D spectral discretization of $\mathcal{L}_j^y$, $\mathcal{L}_j^x$

$$A_j =$$



Low rank approximation
Matrix equation solvers

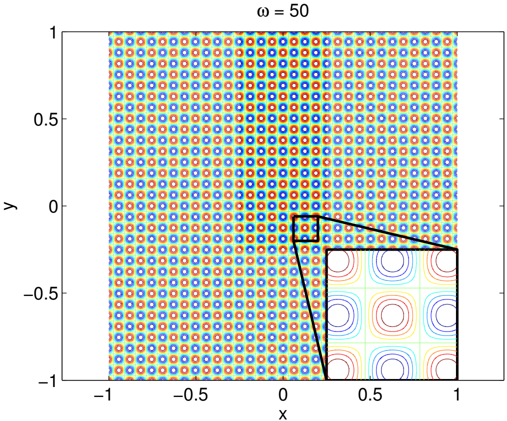
- **Rank 1:** $A_1XB_1^T = F$. Solve $A_1Y = F$, then $B_1X^T = Y^T$.
- **Rank 2:** $A_1XB_1^T + A_2XB_2^T = F$. Generalised Sylvester solver (RECSY) [Jonsson & Kågström, 2002].
- **Rank k, k ≥ 3:** Solve $N \times N$ system using almost banded structure.

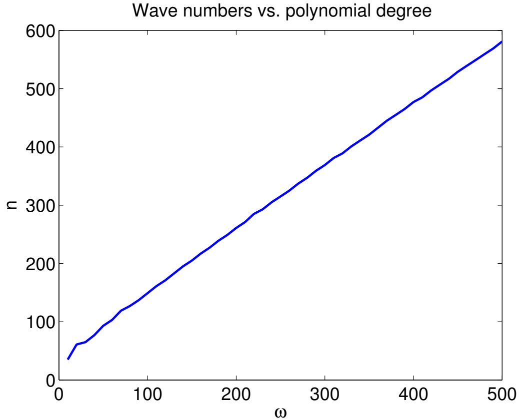


blue = rank 1
green = rank 2
red = rank 3

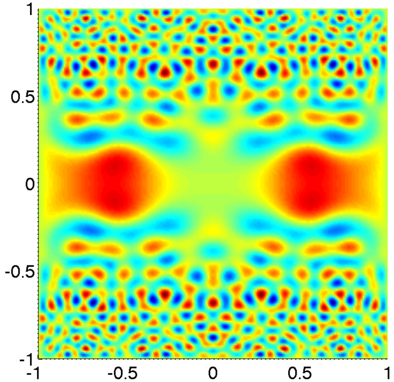
$$\nabla^2 u + 2\omega^2 u = 0, \qquad u(\pm 1, y) = f(\pm 1, y), \quad u(x, \pm 1) = f(x, \pm 1),$$

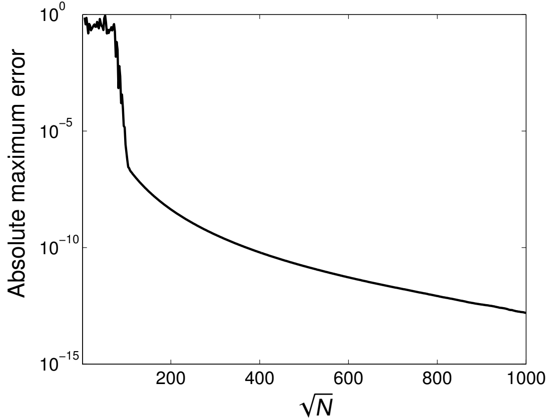
where $f(x, y) = \cos(\omega x)\cos(\omega y)$.





```
N = chebop2(@(x,y,u) laplacian(u) + 10000(1/2+sin(x)^2).*cos(y)^2.*u);
N.lbc = 1; N.rbc = 1; N.ubc = 1; N.dbc = 1;
u = N \ chebfun2(@(x,y) cos(x.*y));
```



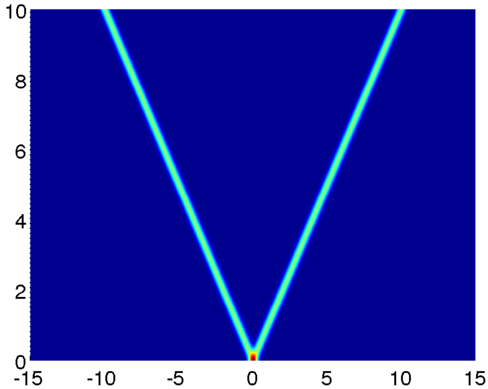


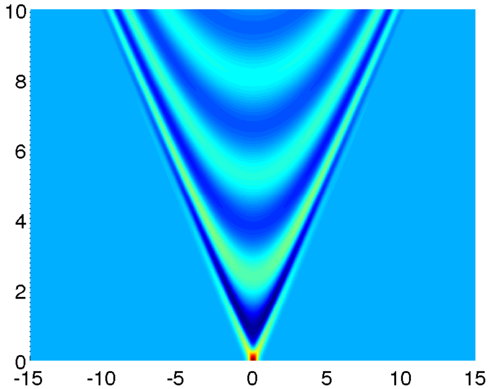
$N = 1{,}050{,}625, \quad \text{error} \approx 1.47 \times 10^{-13}, \quad \text{time} = 44.2s.$

Examples

Wave and Klein–Gordon equation

```
N = chebop2(@(u) diff(u,2,1) - diff(u,2,2) + 5*u); % u_tt - u_xx + 5u
N.dbc = @(x,u) [u-exp(-10*x) diff(u)]; N.lbc = 0; N.rbc = 0;
u = N \ 0;
```





Conclusion

- Spectral methods do not have to be ill-conditioned. (Don't discretize differentiation faithfully.)
- Spectral methods are extremely convenient and flexible.
- As of 2014, global spectral methods are heavily restricted to a few geometries.

Thank you for listening