Computational mathematics – challenges and opportunities

Markus Hegland, ANU, IAS/TUM Hans Fischer Fellow

October 2013

Table of contents

abstract

Computational mathematics is a branch of applied mathematics which utilises mathematical concepts and theorems to understand and develop computational techniques. These techniques are used in applications from weather forecast to machine learning. Equally broad is the scope of mathematics used – from algebraic and differential geometry over linear and multilinear algebra to functional and harmonic analysis.
The challenges originate from the applications considered and the computers used. They include the curse of dimensionality in statistics and uncertainty quantification, ill-posedness of inverse problems, parallelism, asynchronicity and faults in high performance computing and generally computational costs like energy and time.
In my talk I will discuss some computational techniques including sparse grids, low rank tensors and local iterated function systems. With these examples I intend to demonstrate how pure and applied mathematics can work with computer science and the application sciences to address our current computational challenges.

prelude: the most popular computer

smart phones – computers in our pockets?

image from Wikipedia

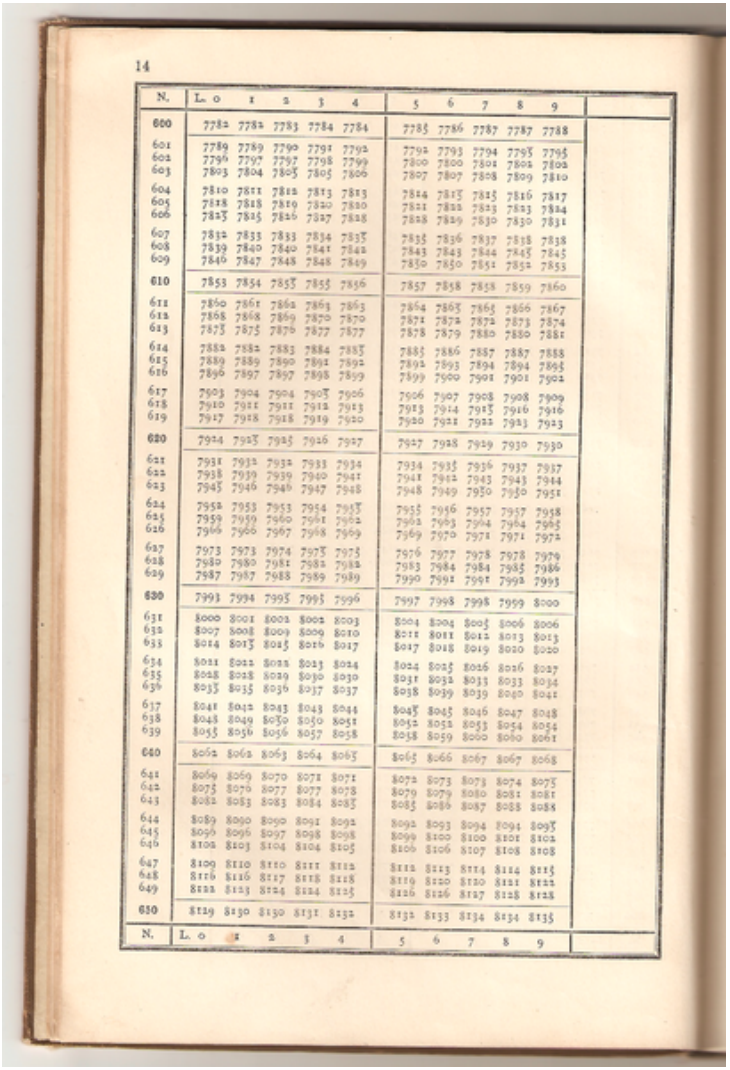
- smart phones are computers which can be used to do phone calls
- Linpack benchmark at 100 MFlop/s – about 1/2 of Cray 1, prominent supercomputer in 1980s
- while there is more to computing than dense linear systems, computational power of the multiple processors in phones may be used do computational science
- decisions based on complex computational procedures may be done computing a large number of scenarious off-line, storing the results in the cloud and then interpolating and displaying the results on the phone (or tablett) on-line

proof of concept: Raspberry Pi 2012

- full computer based on ARM processor used in mobile phone
- running Linux operating system
- simple simulations with Python SciPy or Octave
- uses 3.5 W

image from Wikipedia

the online / offline paradigm is very old



- logarithm table 17th until 20th century
- values computed in advance (offline) by several human computers and interpolated (online) by human computer when needed
- based on mathematical and computational advances
- mathematical tables even older . . .

reduced basis methods

example: solution of elliptic PDEs with parameter μ, weak form

$$a_\mu(u_\mu, v) = L(v) \quad \text{for } \mu \text{ from compact set}$$

approach

- determine $u_{\mu_1}, \ldots, u_{\mu_K}$ offline on supercomputer
- online (phone): new $u_\mu \in \text{span}(u_{\mu_1}, \ldots, u_{\mu_K})$ – Galerkin

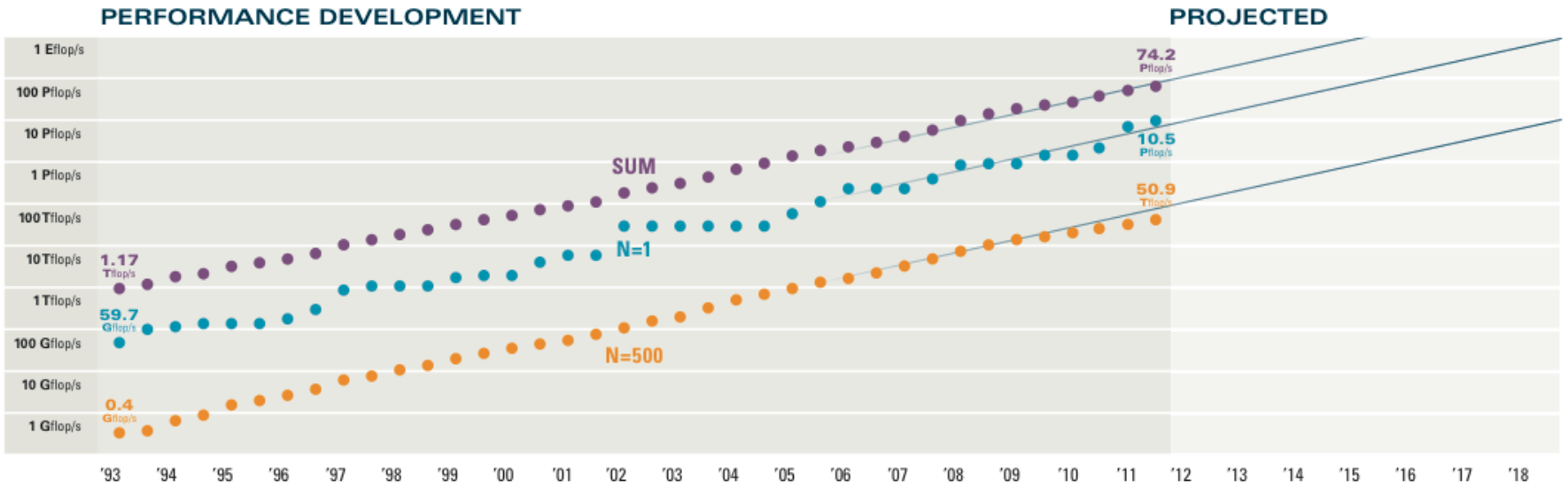
questions and comments

- how to choose original set of μ_j?
- if features $f(u_\mu)$ only – approximate I/O map
- model reduction in controller design (Anderson/Moore)
- requires mathematical error analysis – approximation theory
- high dimensional problem and high performance computing
- density estimation – statistical inverse problem

Binev et al 2011, H./Griebel 2010, Stuart 2010

high performance computers

Computational Performance Over Time



Source: Top-500

"I think there is a world market for maybe five computers." – Thomas Watson, chairman of IBM, 1943

Clusters

Fujitsu K-Computer, JP

LRZ SuperMUC (IBM System x), DE

NCI Raijin (Fujitsu Primergy), AU

- earlier computers resembled factory assembly lines
- today's computers are like complex networks of interacting workers

The energy challenge

Watts consumed in context

all of Google	260 MW
747	140 MW
exascale computer with current technology	200 MW
planned (2022) exascale computer	20 MW
K-computer	12.7 MW
Sequoia (LLNL)	8.6 MW
Car	100 KW
Brain	20 W

comments

- energy: 1 MW for 1 year = US \$ 10^6
- hardware cost: US \$ 200 M
- energy = time * watts
- challenge: large problems unaffordable

what is exascale

K	10^3	M	10^6
G	10^9	T	10^{12}
P	10^{15}	E	10^{18}

data from Jack Dongarra

energy saving in computational science

- save execution time:
  - continue with what we do today: reduce communication, increase cache usage, efficient algorithms
  - avoid synchronisation
  - recompute instead of compute-store-recall
  - replace double precision computations with single precision where possible
  - use autotuning – compilers can save as well
- hardware manufacturers support savings by
  - providing excess of computational resources (GPUs)
  - improving data access by sharing resources (multicore)
- hardware savings come at cost that parallel algorithms now need to be strongly scalable

high performance computing and mathematics

consequences of high level parallelism and energy savings

- component failures more frequent
- unpredictable completion times
- high and complex communication

requirements for the new algorithms and maths

- resilient against faults
- asynchronous, independent of order of processing
- decide what and how computed when and where ,,on the fly"
- we need new (stochastic) mathematical models and analysis to understand computational complexity and accuracy of the new algorithms

errors

causes

- faults
  - very infrequent in current clusters – use checkpoint-restart
  - this can bring exascale computers to a halt
- coarse scale approximation
  - used in sparse grids, reduces complexity
- single-precision arithmetic
  - to speed-up data movement

approaches

- approximation theory
  - extrapolation by combining different grids
  - robust combination overcomes instability
  - error analysis $\Rightarrow$ fault fingerprint
- optimisation
  - improved sparse grid approximation
  - recovery from faults

redundancy

- replicate all computations
  - for fault tolerance: unlikely that both processes will fault

    M.Bougeret, H.Casanova, Y.Robert, F.Vivien, D.Zaidouni, 2012

- recompute instead of store/recall
  - computation is more time and energy efficient

- linear algebra uses checksums for error-correction P.Du, A.Bouteiller, G.Bosilca, T.Herault, J.Dongarra, 2012

- checkpoint/restart – on failure

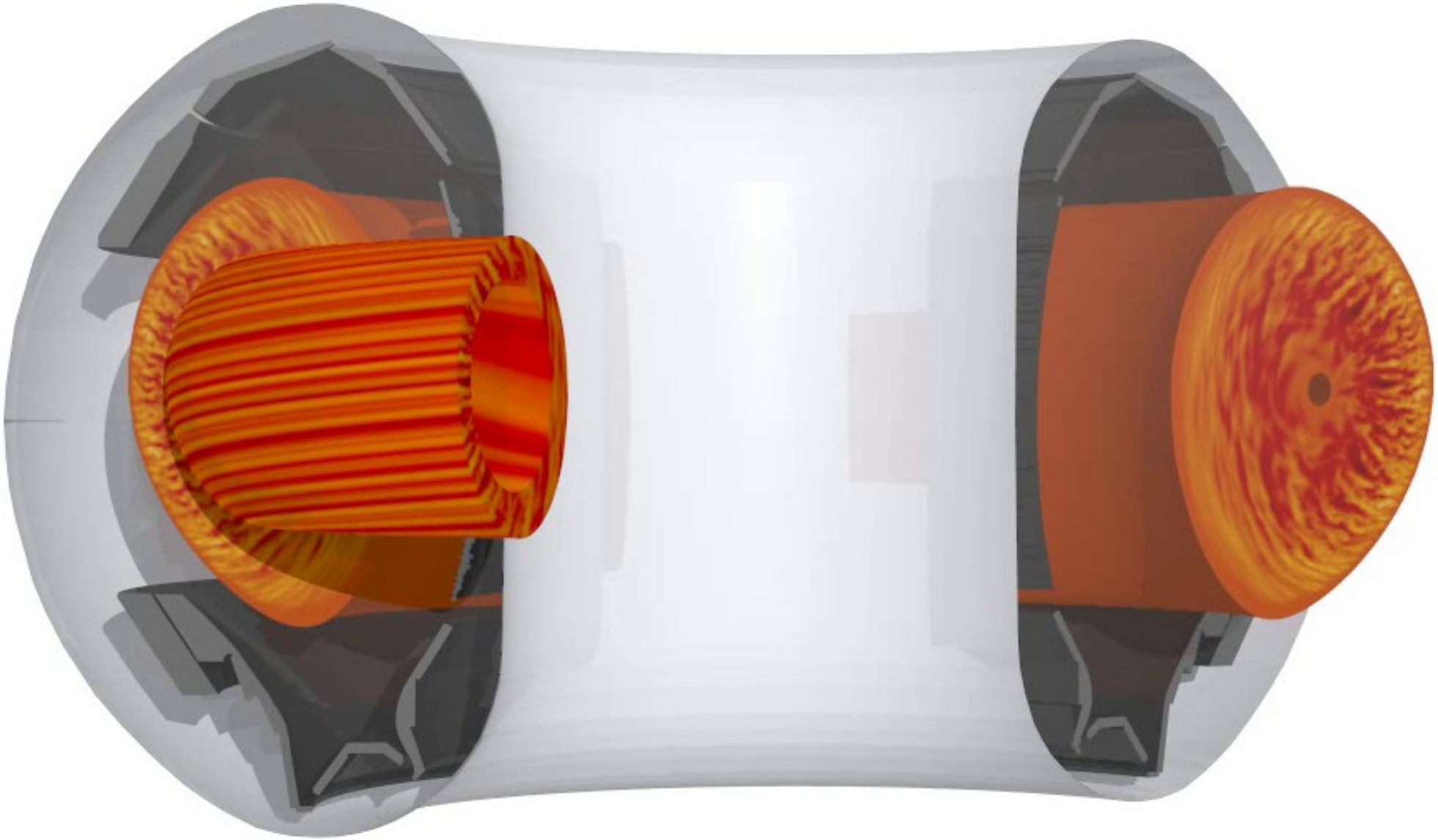
- natural redundancy of the sparse grid combination technique
  - different grids contain essentially same lower scale information
  - use this to get fault tolerance
  - reduce communication?

Synchronisation

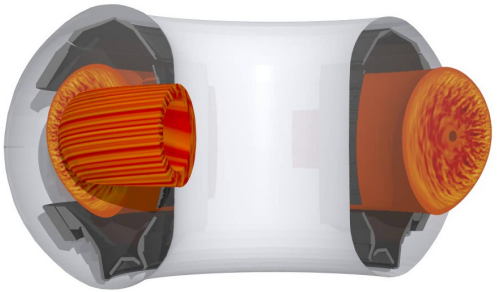
- standard iterative methods like conjugate gradients require synchronisation at every step
- in exascale computers one requires asynchronous algorithms
  - any synchronisation can lead to large performance degradation
  - control convergence locally
  - synchronize at the end only
  - asynchronous methods incurr additional error

a formidable challenge: the curse of dimensionality

Simulation of hot magnetized plasmas



Gyrokinetic Simulations



- FORTRAN90/95 code developed at IPP in Garching (group of Prof. Frank Jenko)
- sophisticated simulation code, which implements the gyrokinetic equations
- highly parallelized
- 5D problem

Vlasov–Maxwell Equations

Vlasov-Equation

$$\frac{\partial f_s}{\partial t} + \vec{v}\frac{\partial f_s}{\partial \vec{x}} + \frac{q_s}{m_s}(\vec{E} + \vec{v} \times \vec{B}) \cdot \frac{\partial f_s}{\partial \vec{v}} = 0$$

Moments of the Distribution Function f

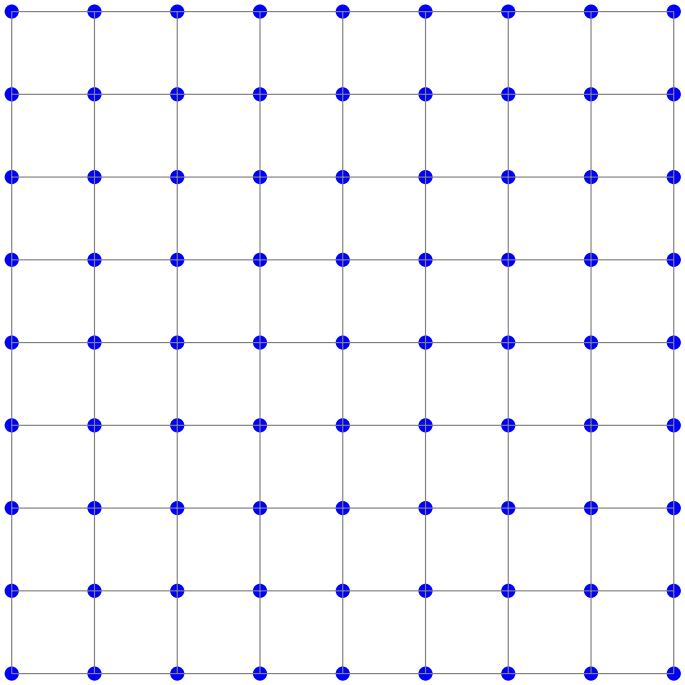
$$\rho(\vec{x}, t) = \sum_s q_s \int f_s(\vec{x}, \vec{v}, t) \mathrm{d}\vec{v} \qquad \vec{j}(\vec{x}, t) = \sum_s q_s \int f_s(\vec{x}, \vec{v}, t) \vec{v} \mathrm{d}\vec{v}$$

Maxwell Equations

$$-\frac{1}{c^2}\frac{\partial \vec{E}}{\partial t} + \nabla \times \vec{B} = \mu_0 \vec{j} \qquad \nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0}$$

$$\frac{\partial \vec{B}}{\partial t} + \nabla \times \vec{E} = 0 \qquad \nabla \cdot \vec{B} = 0$$

regular isotropic grid



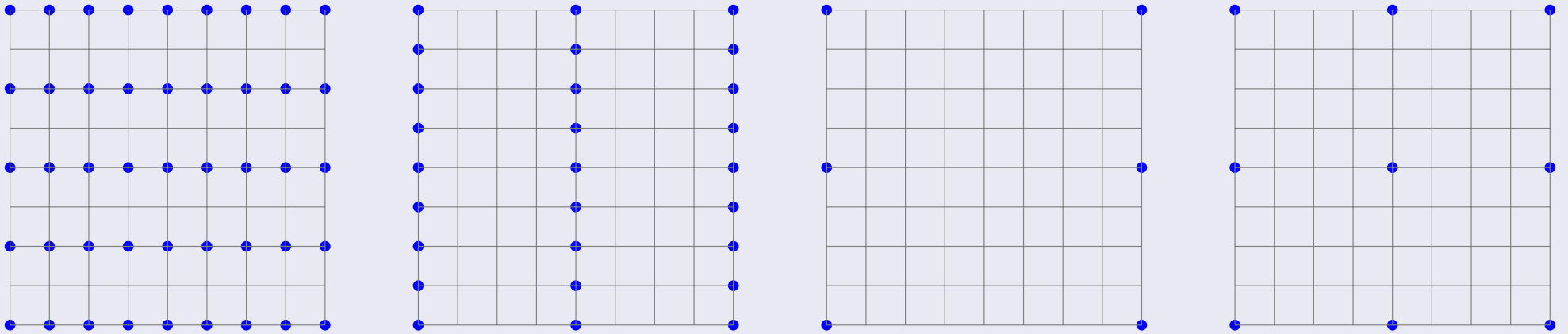
- approximate unknown function $u(x, y)$
- compute only values $u(x_i, y_j)$ on discrete grid points
- interpolate values $u(x, y)$ for other points (x, y)
- regular isotropic grid: $x_i = ih$ and $y_j = jh$

the challenge: curse of dimension

In two dimensions $1/h^2$ grid points, in d dimensions $1/h^d$ grid points but accuracy proportional to h^2

regular anisotropic grids

more general regular grids

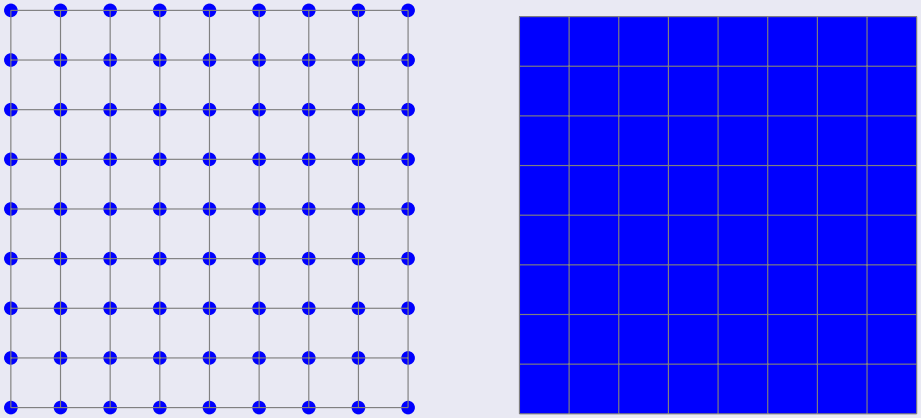


- choose fine grid when $u(x, y)$ has large gradients
- choose coarse grid when $u(x, y)$ is smooth
- gradients may be different in different directions
- choose anisotropic grid when $u(x, y)$ varies differently in different directions

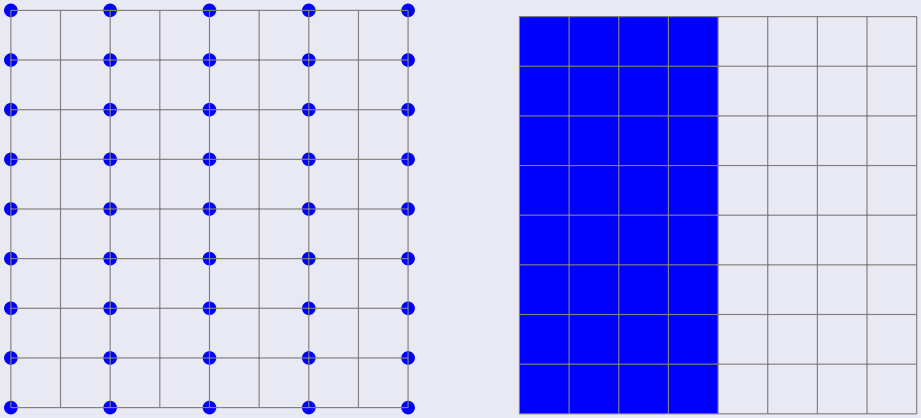
with anisotropic grids one can approximate multi-dimensional $u(x_1, \ldots, x_d)$ if u very smooth in most x_k

sampling and scale space

full grid captures all scales



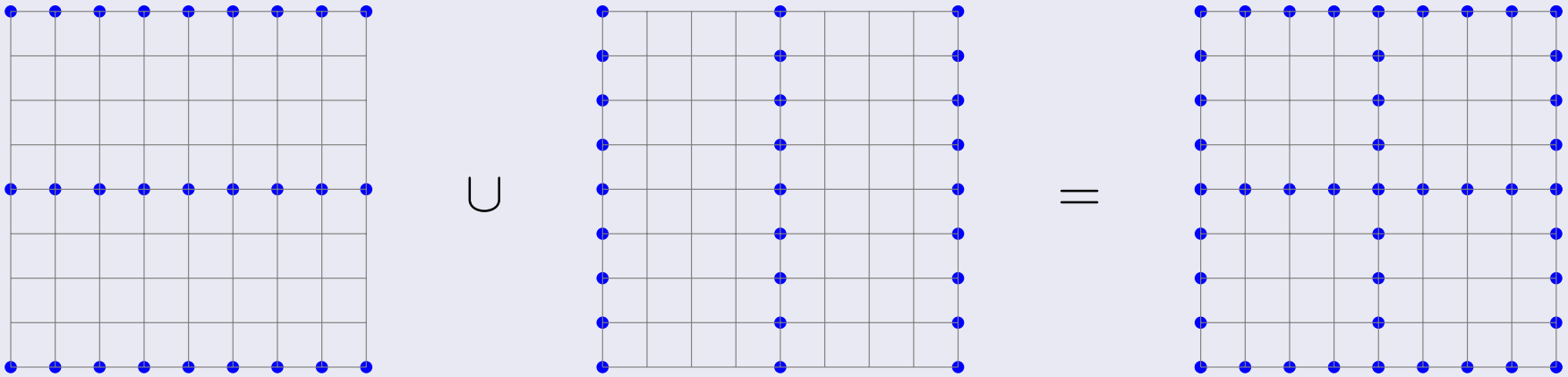
subgrid captures less scales



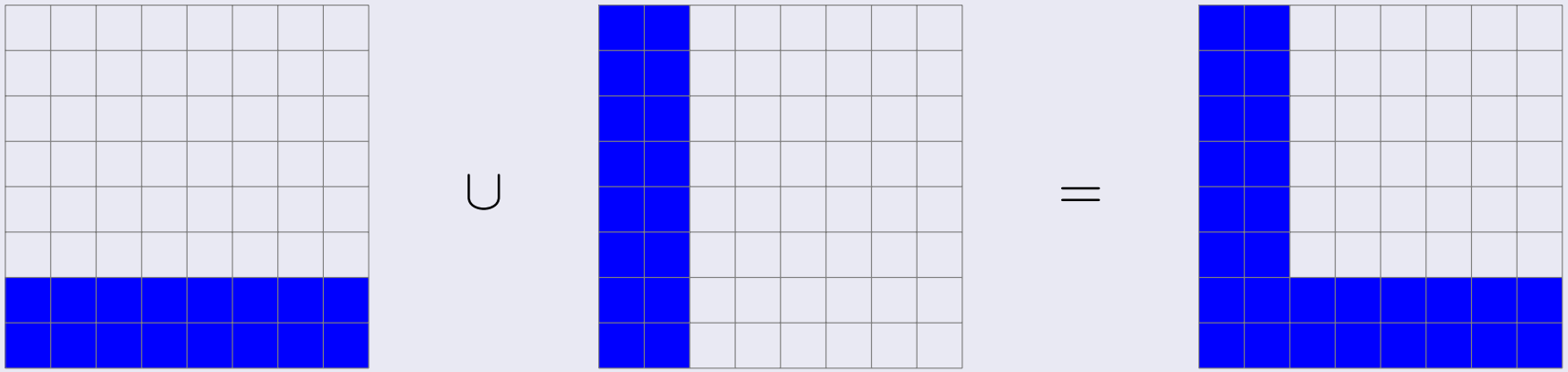
- evaluation of $u(x, y)$ on the grid corresponds to sampling u on the grid points
- sampling on a fine grid captures high frequencies – small scale fluctuations (Nyqvist/Shannon)
- with anisotropic grids one can capture small scales in one dimension and different scales in another

geometric definition of sparse grid

a simple sparse grid



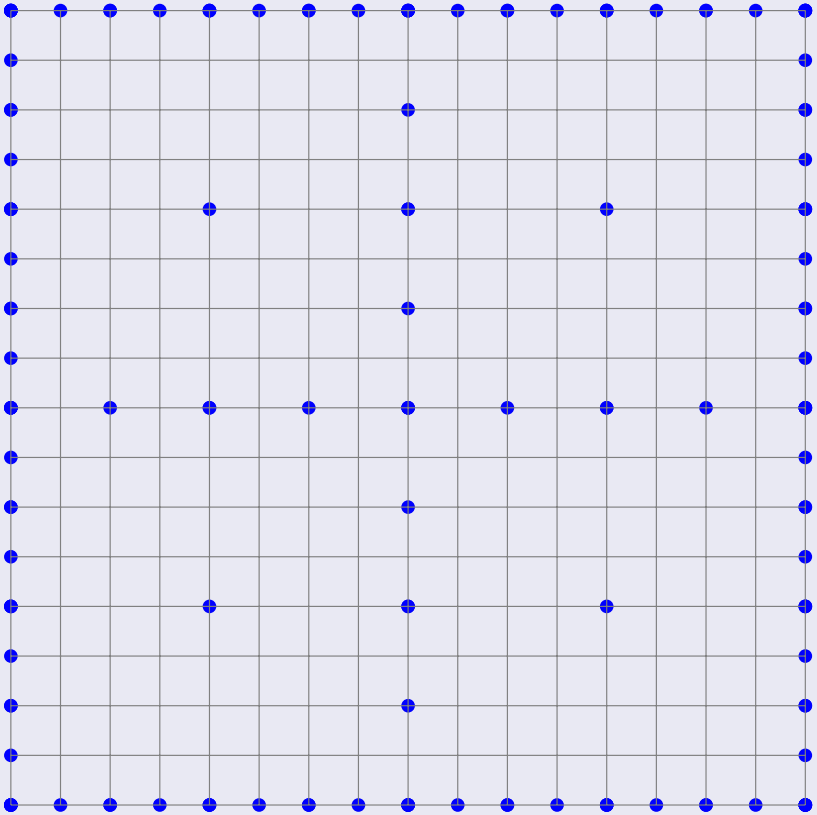
sparse grid in frequency / scale space



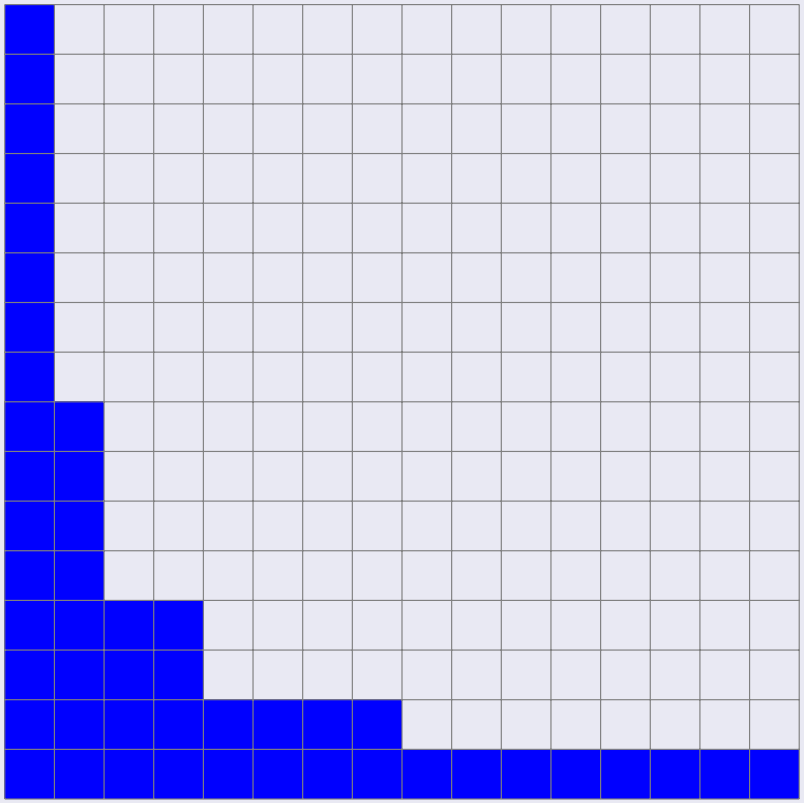
captures fine scales in both dimensions but not joint fine scales

hyperbolic cross

sparse grid points



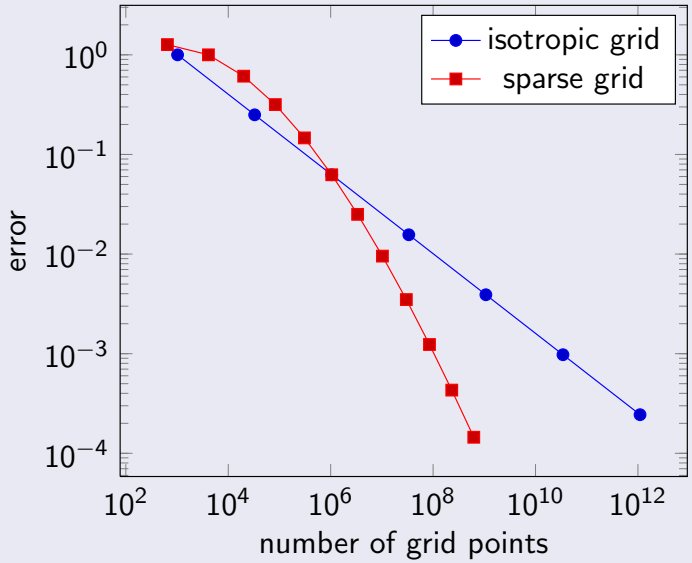
sparse grid scale diagram



the scale diagram displays (a quarater of) a hyperbolic cross

asymptotic error rates

five dimensional case



- only asymptotic error rates given here
- constants and preasymptotics also depend on dimension
- practical experience: with sparse grids up to 10 dimensions
- Zenger 1991

asymptotic rates	number of points	L_2 error
regular isotropic grids	h^{-d}	h^2
sparse grids	$h^{-1} \lvert \log_2 h \rvert^{d-1}$	$h^2 \lvert \log_2 h \rvert^{d-1}$

combining solutions from multiple grids

regular grid approximation

- regular grid $G_{\mathbf{h}}$
- function space $V_{\mathbf{h}}$
- Galerkin equations for $u_{\mathbf{h}}$

$$a(u_{\mathbf{h}}, v_{\mathbf{h}}) = \langle f, v_{\mathbf{h}} \rangle$$

for all $v_{\mathbf{h}} \in V_{\mathbf{h}}$

sparse grid approximation

- sparse grid $G_{SG} = \bigcup_{\mathbf{h}} G_{\mathbf{h}}$
- function space $V_{SG} = \sum_{\mathbf{h}} V_{\mathbf{h}}$
- Galerkin equations for u_{SG}

$$a(u_{SG}, v_{SG}) = \langle f, v_{SG} \rangle$$

for all $v_{SG} \in V_{SG}$

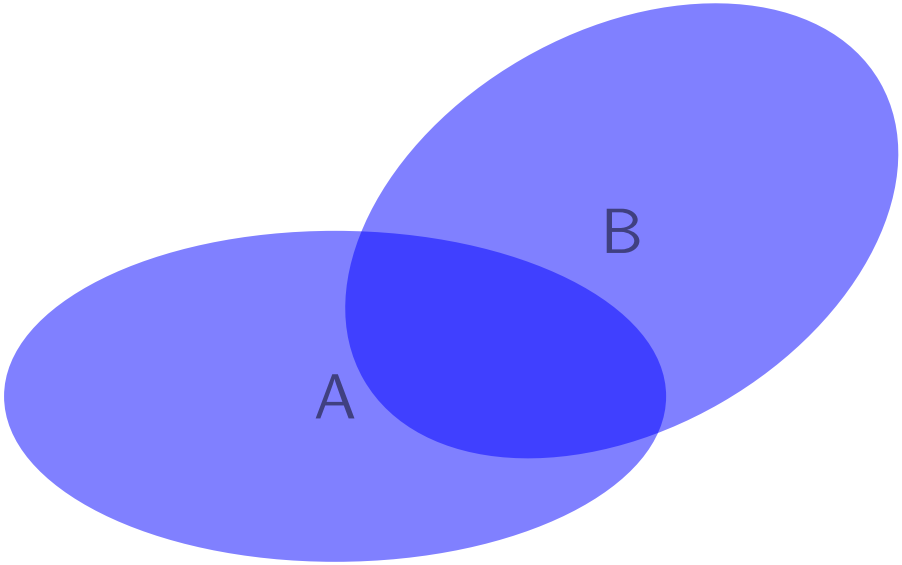
combination technique – where HPC comes in

compute all $u_{\mathbf{h}}$ in parallel and combine solutions using parallel reduction:

$$u_C = \sum_{\mathbf{h}} c_{\mathbf{h}} u_{\mathbf{h}}$$

Big question: when is $u_C \approx u_{SG}$?

Inclusion / exclusion principle in combinatorics



for the cardinality of sets

$$|A \cup B| = |A| + |B| - |A \cap B|$$

more general for additive α:

$$\alpha(A \cup B) = \alpha(A) + \alpha(B) - \alpha(A \cap B)$$

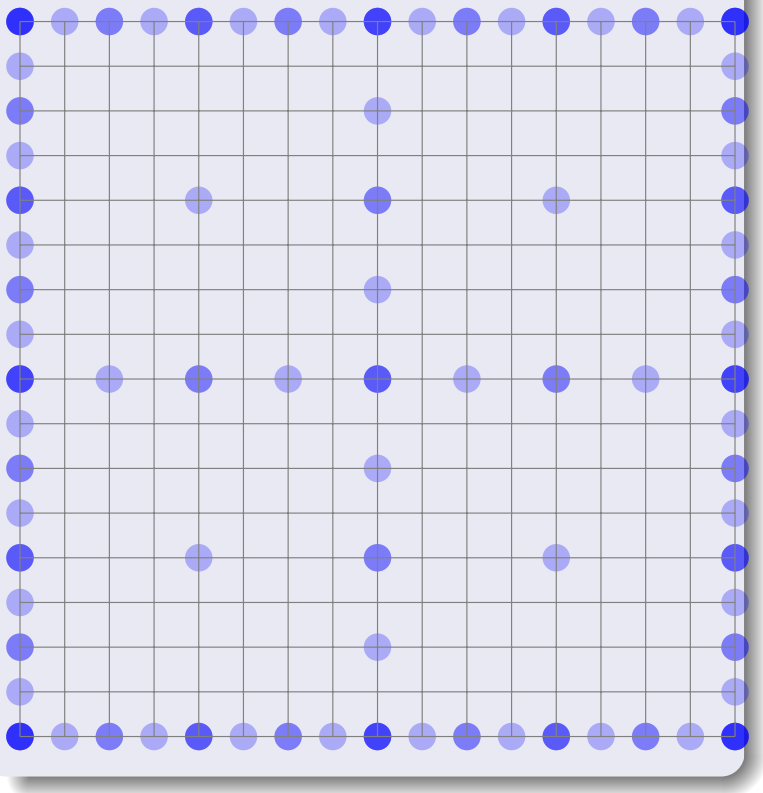
Theorem (de Moivre)

If $A_1, \ldots, A_m$ form intersection structure then

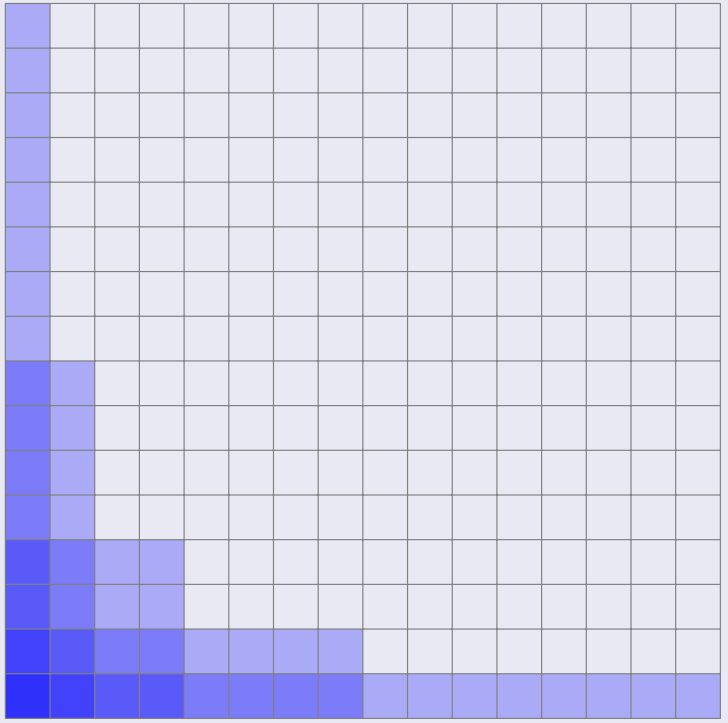
$$\alpha\left(\bigcup_{i=1}^{m} A_i\right) = \sum_{i=1}^{m} c_i\, \alpha(A_i), \quad \text{for some } c_i \in \mathbb{Z}$$

overlap of grids and combination

sparse grid points



sparse grid scale diagram



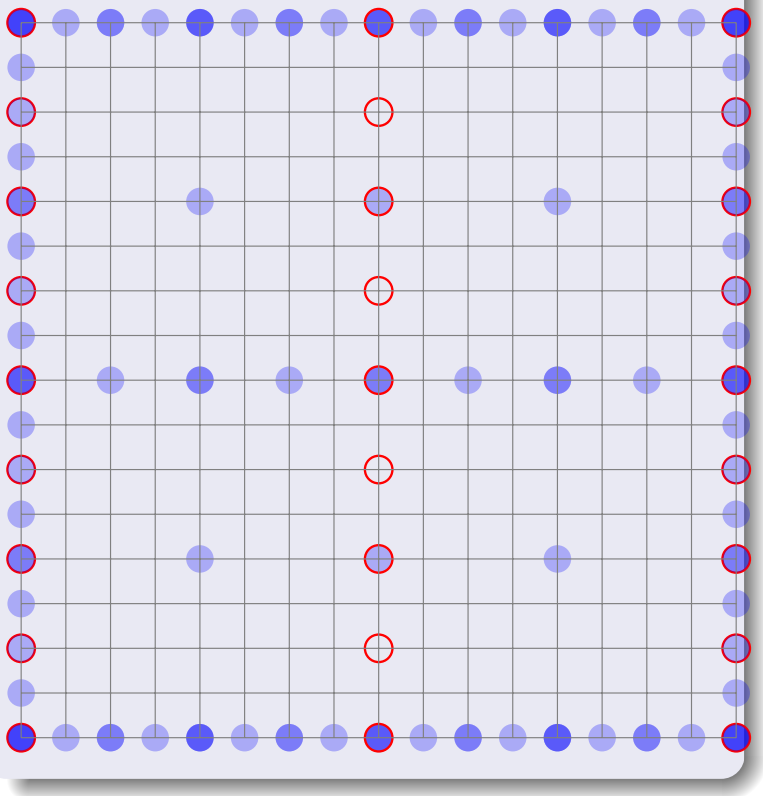
combination formula

$$u_C = u_{1,16} + u_{2,8} + u_{4,4} + u_{8,2} + u_{16,1} - u_{1,8} - u_{2,4} - u_{4,2} - u_{8,1}$$

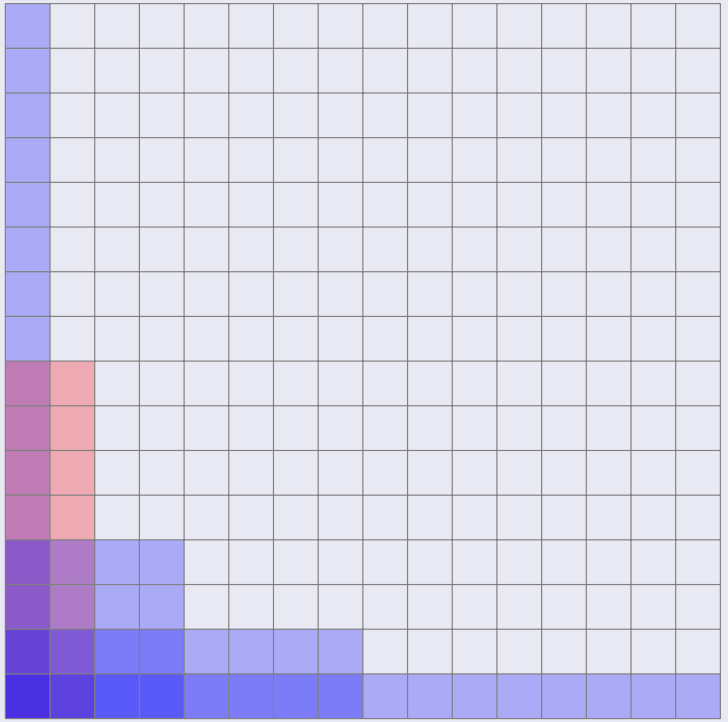
Griebel, Schneider, Zenger 1992

overlap = redundancy ⇒ (lossy) fault tolerance

sparse grid points



sparse grid scale diagram



revised combination formula

$$u_C = u_{1,16} + u_{4,4} + u_{8,2} + u_{16,1} - u_{4,2} - u_{8,1} - u_{1,4}$$

H. CTAC 2003, Harding 2012

tensors

tensors

- tensor = multidimensional array $u \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
- for $d > 4$ curse of dimension:
  - $O(n^d)$ storage not feasible
  - many computations not feasible

feasible tensors

- rank one: $u(x) = \prod_i u_i(x_i)$
- product of lower-dimensional tensors:
  $u(x) = \prod_{\alpha \in C} u_\alpha(x_\alpha)$
  where $C \subset 2^{\{1,\ldots,d\}}$ and
  $x_\alpha = (x_{\alpha_1}, \ldots, x_{\alpha_k})$

examples

- probabilities on discrete state space, data mining, ML and statistics
- chemical master equation
- quantum mechanics
- graphical models

Kolda and Bader: Tensor decompositions and applications, SIAM Review 2009

motivation: matrix decompositions $A = BC^T$

$$a_{ij} = \sum_{k=1}^{r} b_{ik}\, c_{jk}$$

- examples: LU, QR, SVD
- if rank r low use factors B and C in computations
- $Ax = B(C^T x)$ requires $O(rn)$ operations instead $O(n^2)$

key ingredients

- augmented (3D) feasible tensor T with elements

$$t_{ikj} = b_{ik} c_{jk}$$

- original matrix elements defined as sum

$$a_{ij} = \sum_{k} t_{ikj}$$

tensor decompositions

recover u from augmented tensor t

$$u(x) = \sum_{\alpha} t(x, \alpha) = \sum_{\alpha} \prod_{j} t_j(x, \alpha)$$

examples of augmented tensors

- CP: $t(x_1, x_2, x_3, \alpha) = u_1(x_1, \alpha)\, u_2(x_2, \alpha)\, u_3(x_3, \alpha)$
- Tucker: $t(x_1, x_2, x_3, \alpha_1, \alpha_2, \alpha_3) =$
  $u_1(x_1, \alpha_1)\, u_2(x_2, \alpha_2)\, u_3(x_3, \alpha_3)\, w(\alpha_1, \alpha_2, \alpha_3)$
- hierarchical: $t(x_1, x_2, x_3, \alpha_1, \alpha_2, \alpha_3) =$
  $u_1(x_1, \alpha_2)\, u_2(x_2, \alpha_2)\, u_3(x_3, \alpha_3)\, w_1(\alpha_1, \alpha_2)\, w_2(\alpha_1, \alpha_3)$
- tensor train:
  $t(x_1, x_2, x_3, \alpha_1, \alpha_2) = u_1(x_1, \alpha_1)\, u_2(x_2, \alpha_1, \alpha_2)\, u_3(x_3, \alpha_2)$

Tensor Trains – a stable low-rank approximation

$$A(i_1, \ldots, i_d) = G(i_1)G(i_2)\cdots G(i_d)$$

where $G(i_k)$ are $r_{k-1} \times r_k$ matrices ($r_0 = r_d = 1$)

properties

- get $G(i_k)$ with SVD of smaller matrices
- the ranks r_k are the equal to the ranks of the matrices A_k which are obtained from A by unfolding, i.e., A_k is a $n_1 \cdots n_k$ by $n_{k+1} \cdots n_d$ matrix with elements $A(i_1, \ldots, i_k; i_{k+1}, \ldots, i_d)$
- Choose low-rank approximations in the SVDs with errors ϵ_k, get error bound for approximation B in Frobenius norm:

$$\|A - B\|_F^2 \leq \sum_{i=1}^{d} \epsilon_k^2.$$

Oseledets, Tyrtyschnikov, 2009, Oseledets 2012, earlier work in quantum mechanics

computing with fractals

iterated function systems

- contractive functions $f_i : X \to X$ on metric space X
- Hutchinson operator:

$$F(A) = \bigcup_i f_i(A)$$

- fractal = fixpoint G of F:

$$G = F(G)$$

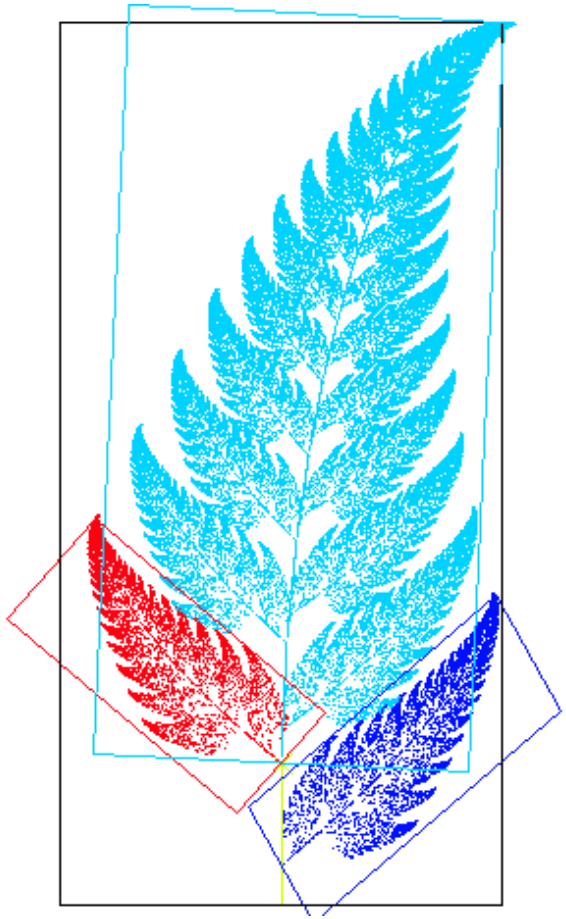
Hutchinson 1981

iterated function systems are everywhere

- in particular in computational mathematics

Barnsley, "Fractals everywhere", 1988

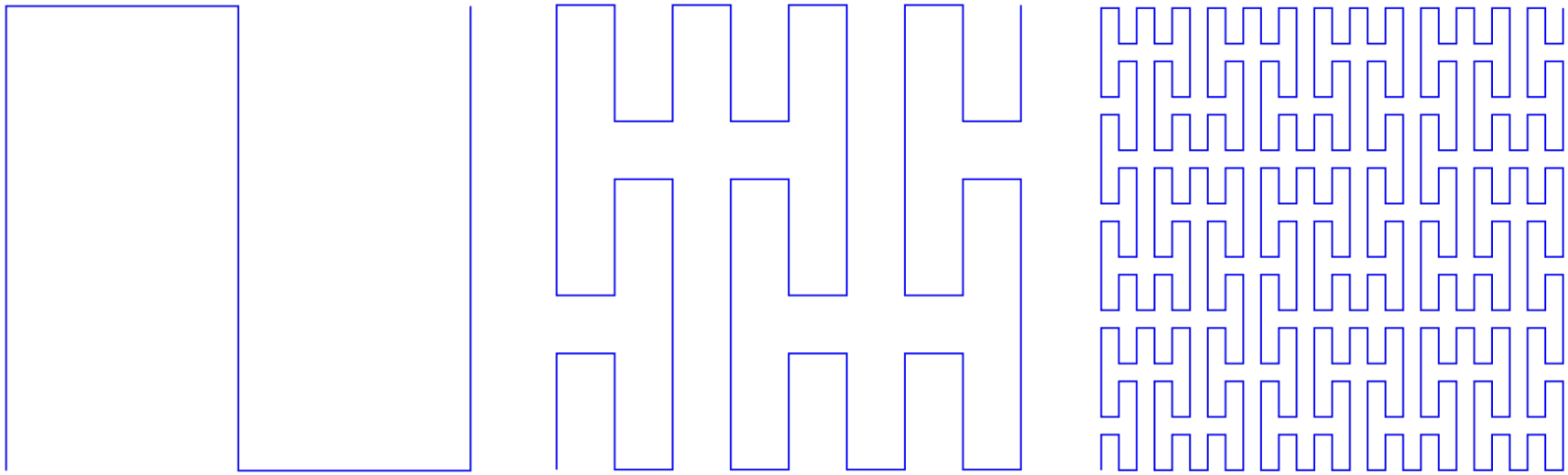
example 1: Barnsley's fern



- 3 functions f_1, f_2, f_3

from: Wikipedia, 2006, de Campos

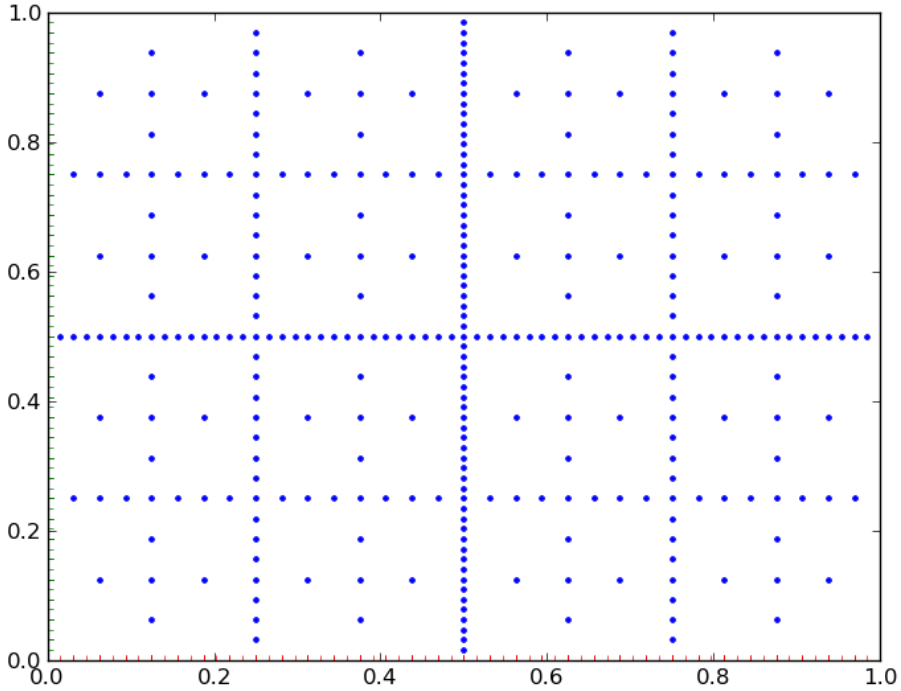
example 2: Peano curve



- 9 functions $f_1, \ldots, f_9$
- space filling curve, continuous map from one to two dimensions
- used to order grid points in high performance computing (Griebel, Zumbusch 2002)

from: Wikimedia Commons, 2007, de Campos

example 3: sparse grids



- use IFS to generate fine grid from coarse grid
- fix point = unit square
- grid = "incomplete fractal"
- sparse grid has IFS with $2d + 1$ elements, regular grid 2^d elements

fractals and computation

- local IFS $f_i : X_i \to X$

$$F(A) = \bigcup_i f_i(A \cap X_i)$$

gives piecewise polynomials and wavelets (refinement equations), used in computer graphics (subdivision)

- numerical grids obtained by truncating evaluation of IFS
- coding points on fractals

$$x = f_{i_1} \circ f_{i_2} \circ \cdots (x_0)$$

then represent x by its "digits" $(i_1, i_2, \ldots)$

- chaos game to generate random points on fractal (Barnsley)

$$x_{k+1} = f_R(x_k), \quad R \text{ chosen randomly}$$

more on numerics and fractals: Barnsley, Massopust, H. 2013

Collage Theorem (Barnsley 1988)

If

- $f_1, \ldots, f_N$ contractive IFS on complete metric space X
- M nonempty compact subset of X

and

$$d_H(M, \bigcup_{i=1}^{N} f_i(M)) < \epsilon$$

then Hausdorff distance between M and attractor A of IFS satisfies

$$d_H(M, A) < \frac{\epsilon}{1-s}$$

where $s = \max_i \operatorname{Lip} f_i$

collage fitting algorithm

- minimise Φ
- IFS defines operator $F(\cdot, \alpha)$
- choose initial u_0

iteration $k = 0, 1, 2, \ldots$

- $\alpha_{k+1} = \operatorname{argmin}_\alpha \Phi(F(u_k, \alpha))$
- $u_{k+1} = F(u_k, \alpha_{k+1})$

convergence

- if $F(\cdot, \alpha)$ provides a descent direction for Φ

Barnsley, H., Massopust 2013

epilogue: regularisation

Two applications of the Galerkin method

Dirichlet problem

$$-\Delta u = f \quad \text{in } \Omega$$
$$f(x) = 0 \quad x \in \partial\Omega$$

linear system of equations

$$Au = f$$

A symmetric positive definite

Finite elements

minimisation problem

$$u_h = \text{argmin}_{V_h} \frac{1}{2} \int_\Omega |\nabla v|^2 - \int_\Omega f\, v$$

approximation space

$$V_h = \text{pw polynomial functions}$$

Conjugate gradients

minimisation problem

$$u_h = \text{argmin}_{V_h} \frac{1}{2}(v, Av) - (f, v)$$

approximation space

$$V_h = \text{span}\{v_0, Av_0, \ldots, A^k v_0\}$$

Galerkin error

Solving minimisation problem is equivalent to minimising error in the energy norm

$$\|u_h - u\|_E = \int_\Omega |\nabla(u_h - u)|^2$$

error in function values (use inverse inequality)

$$\|u_h - u\|_\infty \leq C_h \|u_h - u\|_E$$

large C_h in high dimensions

$$\|u_h - u\|_A = (u_h - u, A(u_h - u))$$

error in components

$$\|u_h - u\|_\infty \leq C_A \|u_h - u\|_A$$

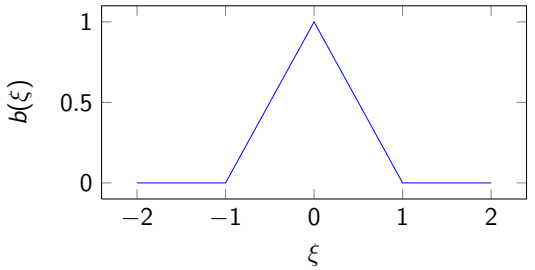
large C_A if A ill-conditioned

Things can go wrong when the constants C_h and C_A are too large $\Longrightarrow$ this is a first indicator of an underlying ill-posed problem

dimension-related ill-posedness

FEM basis functions

- hat function: $b(\xi) = (1 - |\xi|)_+$
- tent function: $t(x) = \prod_i b(x_i)$
- scale and shift: $u_{h,z}(x) := t((x - z)/h)$



properties of u for $z \in \Omega$ and h small

- $u_{h,z}(z) = 1$ and $u_{h,z}(x) = 0$ on boundary
- H_1 semi-norm $\|v\|_1 = \sqrt{\int |\nabla v|^2}$ is then

$$\|u_{h,z}\|_1^2 = 2\,d\,\left(\frac{2}{3}\right)^{d-1} h^{3d-4} \leq 2\,d\,\left(\frac{2}{3}\right)^{d-1}$$

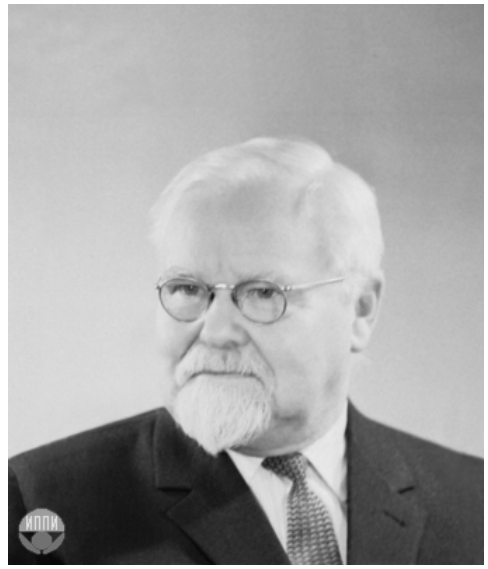
  - $d \geq 2$ and $h \to 0$: $\|u_{h,z}\|_1 \to 0$
  - $d \to \infty$: $\|u_{h,z}\|_1 \to 0$

ill-posed problems

J. Hadamard (1865-1963)

- Hadamard: a problem $Au = g$ is well-posed if a unique solution u exists which depends continuously on f, otherwise the problem is ill-posed
- popular method for solution of ill-posed problem: Tikhonov regularization

$$u_\delta = \text{argmin}_v \, \|Av - f_\delta\|^2 + \alpha \|v\|^2$$



A.N. Tikhonov (1906-1993)

Tikhonov regularisation simultaneously stabilises the norm of the solution and minimizes the residual

error theory for linear ill-posed problems in a nutshell

the classical theory

error bound for solution of $Au = f$ given f_δ

$$\|u_\delta - u\| \leq (\|Au_\delta - f_\delta\| + \|f_\delta - f\|)^{\frac{s}{s+1}} (\|u_\delta\|_s + \|u\|_s)^{\frac{1}{s+1}}$$

from Hölder inequality and triangle inequality

size of the four terms:

- consistency: $\|Au_\delta - f_\delta\|$ small
- data error: $\|f_\delta - f\| \leq \delta$
- numerical stability: $\|u_\delta\|_s = \|(A^*A)^{-\frac{s}{2}} u_\delta\|$ bounded
- source condition = regularity of solution: $u \in R((A^*A)^{\frac{s}{2}})$

error $= O(\epsilon^{\frac{s}{s+1}}) \Longrightarrow$ small for large s

Issues with the source condition $u \in R((A^*A)^{s/2})$

examples where classical theory fails

- derivatives of analytic functions
- statistical inverse problem
- Stokes enhancement in spectroscopy

relations between operator A and u and f

- operator A models either
  - observational procedure (tomography)
  - computational method (spectral sharpening)
- as $Au = f$ one has $f \in R(A)$ necessarily
- stronger conditions on f or u translate to conditions on the operator A for a fixed u and either
  - prescribe properties of measurement devices
  - prescribe properties of computational procedure

we need more flexible source conditions

Reconstruction R_α of $u = A^{-1}f$ from f_ϵ

- worst case error of R_α for data error ϵ and $u \in M$

$$e(R_\alpha, M, \epsilon) = \sup\{\|R_\alpha(f_\epsilon) - u\| \mid f_\epsilon \in U_\epsilon(Au),\ u \in M\}$$

- error bound for optimal choice of R_α

$$e(M, \epsilon) = \inf\{e(R_\alpha, M, \epsilon) \mid R_\alpha : Y \to X\}$$

A bound for the optimal error

$$\omega(M, \epsilon) \leq e(M, \epsilon) \leq \omega(M, 2\epsilon)$$

where modulus of continuity of A^{-1} on M is

$$\omega(M, \epsilon) = \sup\{\|x\| \mid x \in M,\ \|Ax\| \leq \epsilon\}$$

use variable Hilbert scales to get bounds

Ivanov/Korolyuk, 1969, Micchelli/Rivlin 1980, Anderssen/H./Hofmann 1992–2013

Bounding the modulus of continuity of A^{-1}

Mathe/Pereverzev'03 and Hofmann/Mathe/Schieck'08

$$\omega(M, \epsilon) \leq R\,\overline{\psi}\,(\Theta^{-1}(\epsilon/R))$$

- $\overline{\psi} \nearrow$ and $M = \overline{\psi}(A^*A)[B_R]$
- $\Theta(t) = \sqrt{t}\,\overline{\psi}(t)$ and $\overline{\psi}^2(\Theta^{-1}(\sqrt{t}))$ concave

H./Hofmann'10 – same bound but simpler criterion

$$\omega(M, \epsilon) \leq \epsilon\sqrt{\Psi(R^2/\epsilon^2)}$$

- $M = B_\chi(R)$ ball in X_χ
- $\chi(\lambda) \geq \Psi^{-1}(\lambda)/\lambda$ and Ψ concave

conclusions

- earlier computations focussed on low-dimensional, well-posed problems which were to be solved on simple computational techniques
- today many computational problems involve large data sets, are ill-posed and also high dimensional
- the new computational infrastructure is changing:
  - on one size there are ever increasing very low-powered devices
  - on the other side the large high performance engines in compute and data centres
  - major factor is energy, requires rethinking models and algorithms
- collaboration between mathematicians, computer scientists, engineers and scientists leads to new technology

thanks

- Matthias Wong, Brendan Harding
- Christoph Kowitz, Hans Bungartz, Dirk Pflueger
- Jochen Garcke and Michael Griebel, Bonn
- Bob Anderssen CSIRO and Bernd Hofmann, Chemnitz
- Peter Strazdins, Steve Roberts, Alistair Rendell, Jay Larson and Linda Stals
- Fujitsu Laboratories Europe, Ross Nobes, James Southern
- many others from Australia and overseas
- Technische Universität München Institute for Advanced Study, funded by the German Excellence Initiative (DFG)
- several ARC Discovery and Linkage grants, ARC CoE Bioinformatics, NICTA, APAC, ACSys CRC, DFG SFB 611 and others