150 Jahre culture of excellence

Sparse Grids and their Impact on HPC & Big Data

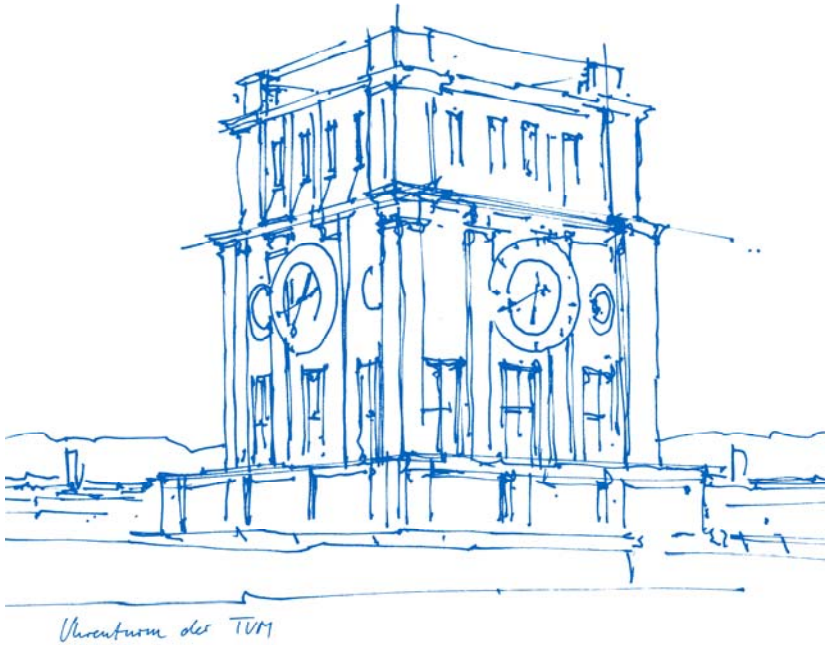
Hans-Joachim Bungartz

Technical University of Munich
Department of Informatics
Chair of Scientific Computing

IPAM Long Program “Science at Extreme Scales”

WS1: Big Data Meets Large-Scale Computing

IPAM, UCLA, September 25, 2018



Contents

Contents

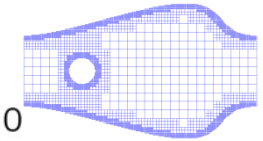
HPC – Core Enabler for Computational Science & Engineering: The Classical Simulation & Optimization Cycle

Mathematical model

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \frac{1}{\rho}\nabla p - \nu\,\Delta\mathbf{u} = 0$$
$$\nabla\cdot\mathbf{u} = 0$$

Discretization & solver

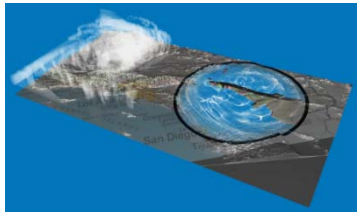
$$M\mathbf{u}_h = 0$$
$$A\dot{\mathbf{u}}_h + D\mathbf{u}_h + C(\mathbf{u}_h)\mathbf{u}_h - M^T p_h/\rho = 0$$



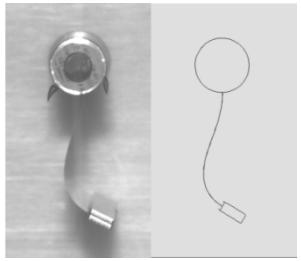
Parallel implementation, HPC

Software

Exploration



Validation, data integration



Insight, design

Today #1: no cycle any more, interplay of all phases
Toady #2: less space for single-field experts
Today #3: outer loops (opt., coupled problems, control, UQ, …)
Today #4: in-situ integration of data, rather than mere pre-/post-processing

A Brief History of “Computational”

1st generation: qualitative (forward) simulation

- ➔ *HPC: one large run*
- ➔ *BD: only through post-processing*

2nd generation: optimization, inverse problems

- ➔ *HPC: appearance of outer loops*
- ➔ *BD: parameter identification & reduction*

3rd generation: quantitative through data integration

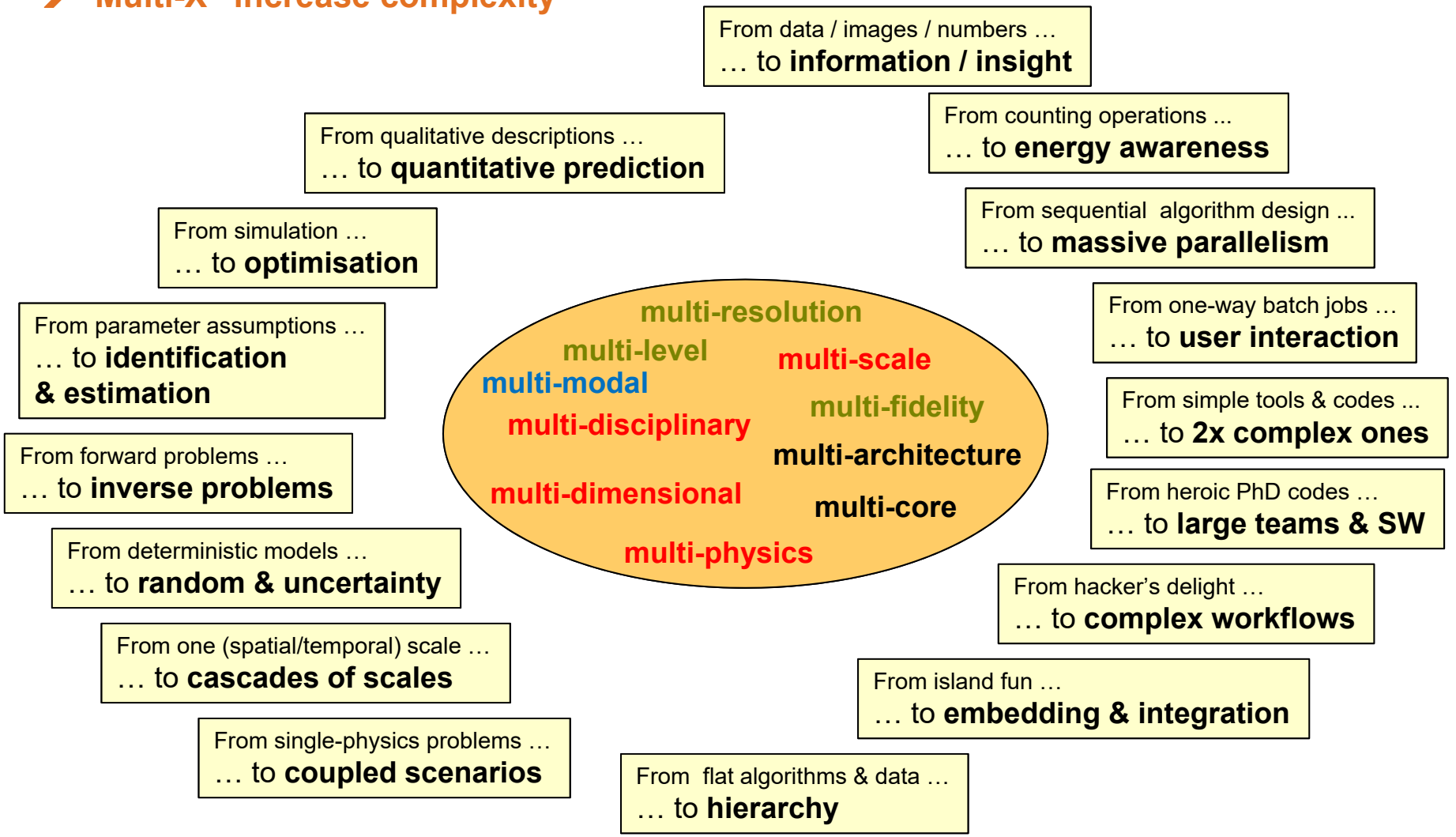
- ➔ *HPC: more outer loops (UQ)*
- ➔ *BD: integration of simulation & experimental data*

Ultimate goal: “predictive” science

In that sense: still exploring the potential of computational technology to overcome the gap between theory & experiment

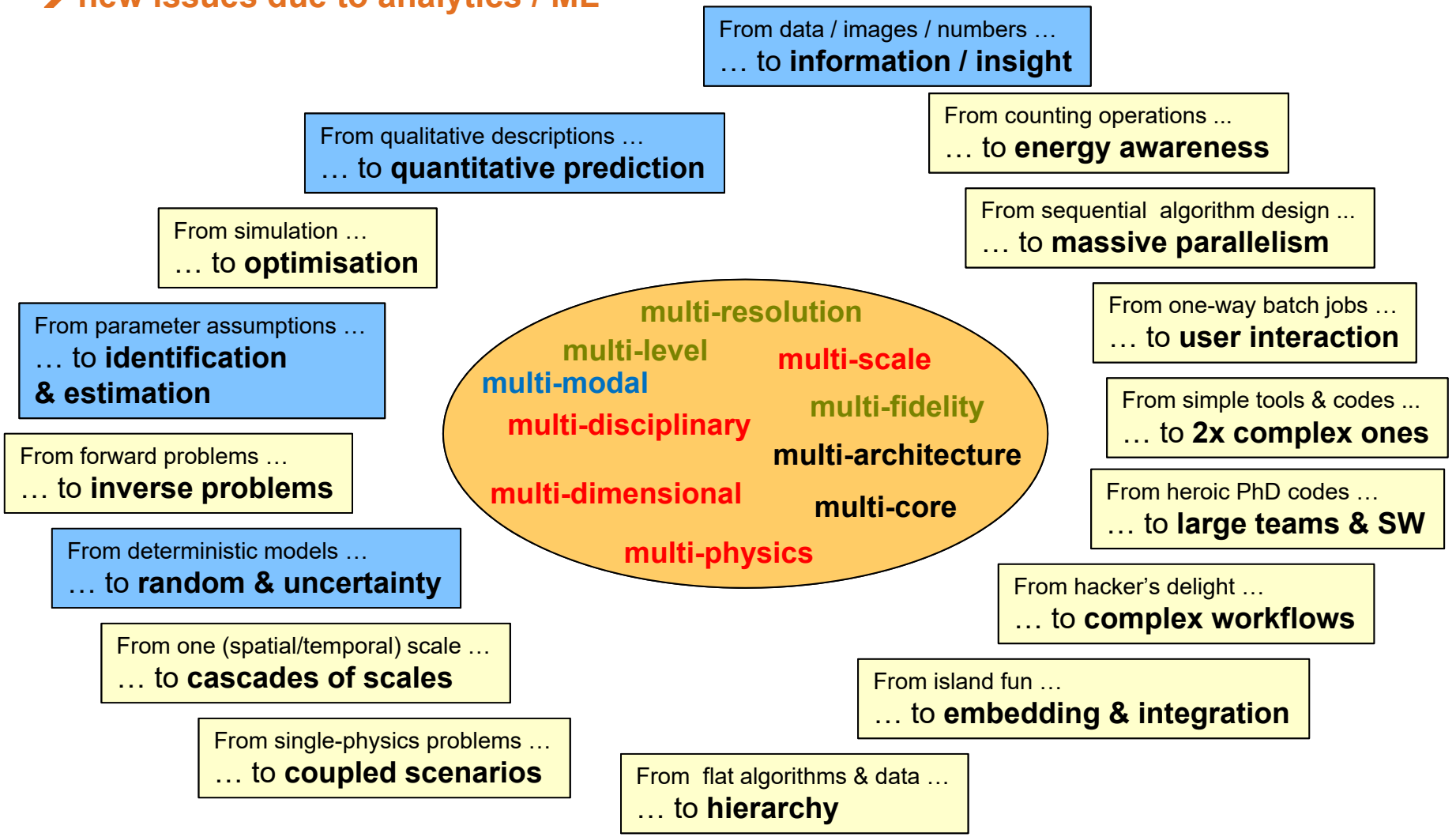
CSE&HPC Challenges

➔ “Multi-X” increase complexity



CSE&HPC Challenges

➔ new issues due to analytics / ML



Multi-Disciplinarity – not really the easiest of the Multi's

There are, at least, six levels of multi-disciplinarity ☺ :

Level 0 – "we don't need anything, we are intrinsically interdisciplinary"

… often the physics point of view

Level 1 – "let's buy and read a book from another discipline"

… the fallback position if level 0 does not apply at all

Level 2 – "let's hire a student assistant from another discipline"

… only if level 1 does not work at all

Level 3 – "let's hire a research assistant from another discipline"

… it starts to get painful & costly

Level 4 – "let's ask a colleague from another discipline"

… only in case of complete despair

Level 5 – Insight: "we are unable to solve the problem on our own"

… hardly ever with a professor

What's your level?

Where HPC Meets Big Data

Data analytics uses computational algorithms & needs HPC resources

- Statistics, stochastics (esp. high-dimensional), numerics
- Examples: eigenvalues, SVD, low-rank approximation, kernel-based methods
- Increasing share of analytics applications on HPC systems; need for (math) education

Modern HPC applications are data-related or data-driven

- Parameter identification: search in high-dim parameter spaces
- Model-order reduction: reduce computational demand by smart model design
- Multi-fidelity: combine models/simulations/data at different scales/resolution
- Uncertainty quantification: increase reliability of simulation results

Machine Learning in an HPC context

- Learn from experiments & simulations for more predictive & efficient runs
- Learn from application behaviour for a better dynamic job scheduling

Performance & efficiency as key requirements for both

For Performance, Algorithms are the Key

Broad range of computational algorithm functionalities:

- Discretization schemes, numerical solvers
- Parallelization, communication, load balancing
- Compression, dimension reduction
- Combinatorial, graph-based
- Statistic, stochastic
- Analytical (machine learning, …)

Algorithm life cycle: design – analysis – implementation – tuning

- **Algorithm Engineering: design – analysis** – implementation – tuning
- **Performance Engineering:** design – analysis – **implementation – tuning**
- **Efficiency comprises both: implement the "best" algorithms in the best way!**

Notions of efficiency – what is "best"?

- It's been a long way from the classical Landau *order-of-N-to-the-something* to today's multi-faceted performance landscape …

Notions of Efficiency

- **counting operations, $O(N^k)$ – the classical complexity-related one**
- **#ops – #flops – % of peak performance – the speed-oriented one**
  - both at node level and at system level – weak / strong scalability
- **#bytes (or #bytes/flop) – the communication-related one**
  - communication-avoiding / memory-efficient / compute-bound as good guidelines
- **#Watt (or #Watt/flops or #flops/Watt) – the energy-related one**
  - "cool" solvers – are they really different from classical "fast" ones? May slower be cooler?
- **time-to-solution – the pragmatic one**
  - reasonable at first glance – how long do I have to wait?
  - Q: what to include into consideration, i.e. "time"? Is runtime-only sufficient?
- **digits per flop – the accuracy-related one**
  - brings the model back to stage: a better model ➔ less computation ➔ more digits / flop
- **science (results? pubs?) per flop – the insight-related one**

Efficiency by Economics: Cost-Benefit Ratio

General principle:

- For a continuous problem **P** with solution ***u*** (the quantities of concern), provide discrete approximations $\mathbf{P}_n$ and $\boldsymbol{u}_n$ such that some error ε gets small.

Cost-benefit consideration (cf. notion of ε-complexity):

- **benefit**: obtained error/accuracy ε
- **cost** (I): the classical ones – (1) $N(\varepsilon)$ d.o.f. to obtain ε, (2) $g(N)$ ops. to solve $\mathbf{P}_n$
- **cost** (II): runtime $T(g(N))$, energy consumption $E(g(N))$ for building & solving $\mathbf{P}_n$
- **cost** (III): more metrics – memory/cache efficiency, communication avoiding, …

Classical approach (PDE-based models, *d* dimensions, regularity *p*):

- exponential growth of ***N*** in *d*: $N = O(\varepsilon^{-d/p})$ (curse of dimension)
- polynomial growth of *g* in ***N***: $g = O(N^k)$

Starting points for improving the cost-benefit ratio:

- fast solvers for $\mathbf{P}_n$ (multilevel): ***g* optimal** (linear in ***N***, for example)
- efficient discretizations: **improve *N*** (small, no or only weak *d*-dependence)
  - approaches of higher order, AMR (*problem-dependent, a-posteriori*)
  - **general grid structure**, *problem-independent, a-priori* ➔ cf. Sparse Grids

Contents

Huge Data in HPC: High-dimensional numerics

High: not 2, not 3, not 3 plus time, but everything beyond (10s, 100s, …)

*note: in literature, often "*high*" means more than "*higher*"* ☺

Where?

- quantum mechanics
- finance, statistics & stochastics
- parameter identification, optimisation, inverse problems, UQ
- data mining, classification, machine learning

Why a problem?

- intellectual complexity: think of 11-dimensional hyper-tetrahedra and their AMR … ☺
- computational complexity – the **curse of dimension**:
  - the cheapest 1-D discretisation … … and in 100-D?

➔ $1^{100} = 1$ ➔ cost ☺, benefit ☹

  - the second cheapest 1-D discretisation … … and in 100-D?

➔ 2^{100} approx. 10^{30} ➔ benefit ☹, cost ☹☹

Huge Data in HPC: High-dimensional numerics

High: not 2, not 3, not 3 plus time, but everything beyond (10s, 100s, …)

*note: in literature, often "*high*" means more than "*higher*"* ☺

Where?

- quantum mechanics
- finance, statistics & stochastics
- parameter identification, optimisation, inverse problems, UQ
- data mining, classification, machine learning

Why a problem?

- intellectual complexity: think of 11-dimensional hyper-tetrahedra and their AMR … ☺
- computational complexity – the **curse of dimension**:
  - the cheapest 1-D discretisation … … and in 100-D?

➔ $1^{100} = 1$ ➔ cost ☺, benefit ☹

  - the second cheapest 1-D discretisation … … and in 100-D?

➔ **$2^{100} \sim 10^{30}$; exa ~ 10^{18}**

Contents

A Priori Structure? Cf. Numerical Quadrature!

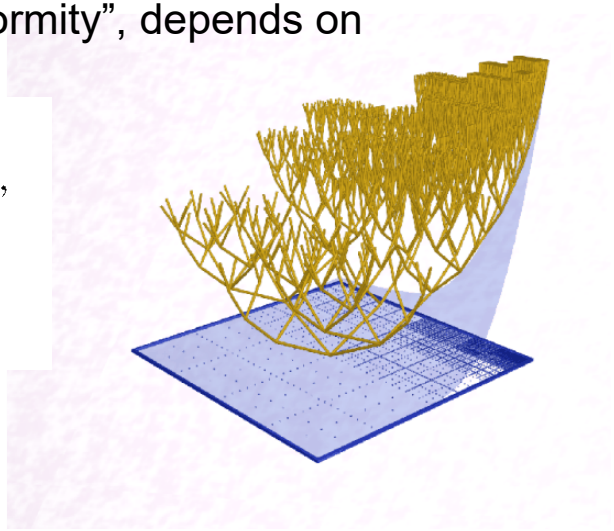
- **Gauß** quadrature: higher degree of accuracy via a non-equidistant choice of nodes (exact up to degree *2n-1*) ➔ fix pattern, a priori chosen
- multivariate quadrature formulas: **inequality of Koksma & Hlawka**

where $$\left|\frac{1}{n}\sum_{i=1}^{n} f(x_i) - \int_{[0,1]^d} f(x)dx\right| \leq V(f) \cdot D_n^*(x_1, \ldots, x_n),$$

  - *V*: **variation** (indicates the integrand's smoothness, depends on problem only)
  - *D*: **discrepancy** (indicates the grid points' "level of non-uniformity", depends on method only ➔ minimum-discrepancy methods)

$$D_n^*(x_1, \ldots, x_n) := \sup_{E \in \mathcal{M}} \left|\frac{\sum_{i=1}^{n} \chi_E(x_i)}{n} - \int_{[0,1]^d} \chi_E(x)dx\right|,$$

$$\mathcal{M} := \left\{[0, a_1) \times \ldots \times [0, a_d) \subseteq [0, 1]^d\right\}$$

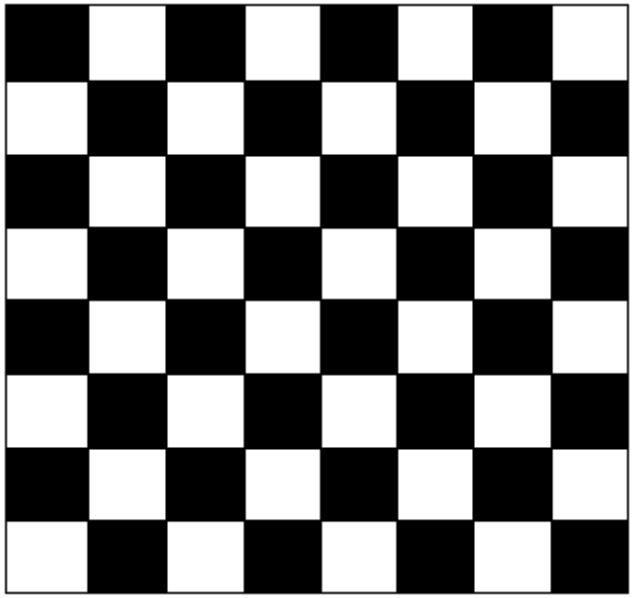


- **Smolyak** quadrature, **Babenko's** hyperbolic crosses
- **Archimedes** quadrature (divide&conquer), Cavalieri for *d>1*

A Priori Structure? Cf. “Battleship”!

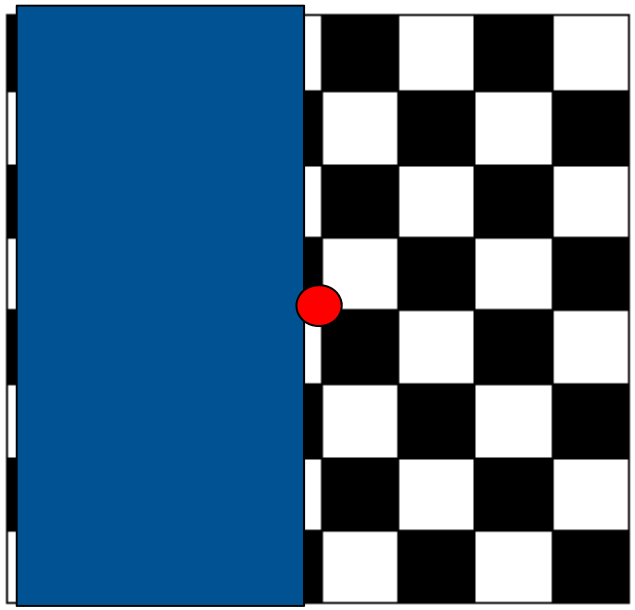
(not really a good analogy, I know – but nice & intuitive nevertheless … ☺)

- *Battleship* game: place shots such that the biggest “ships” (rectangles) that can be hidden get minimal!
- Minimizing the discrepancy means minimizing the maximum “empty” rectangles occurring



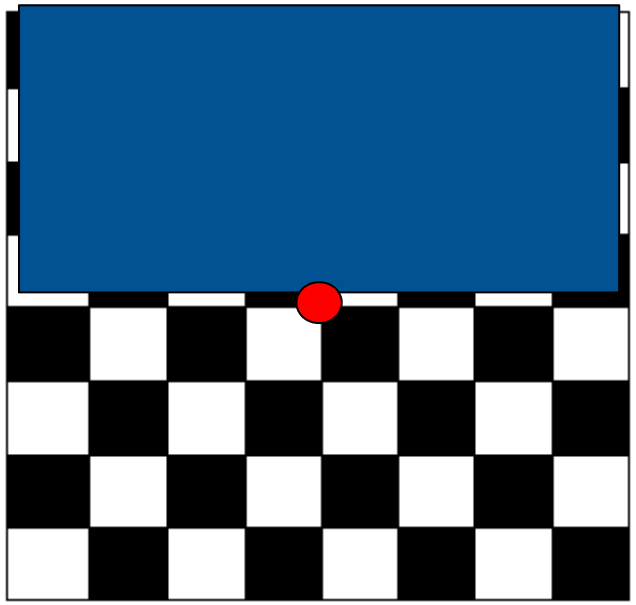
Who shoots
A1
A2
A3
A4
…
?????

Optimum A-priori Grid Structure



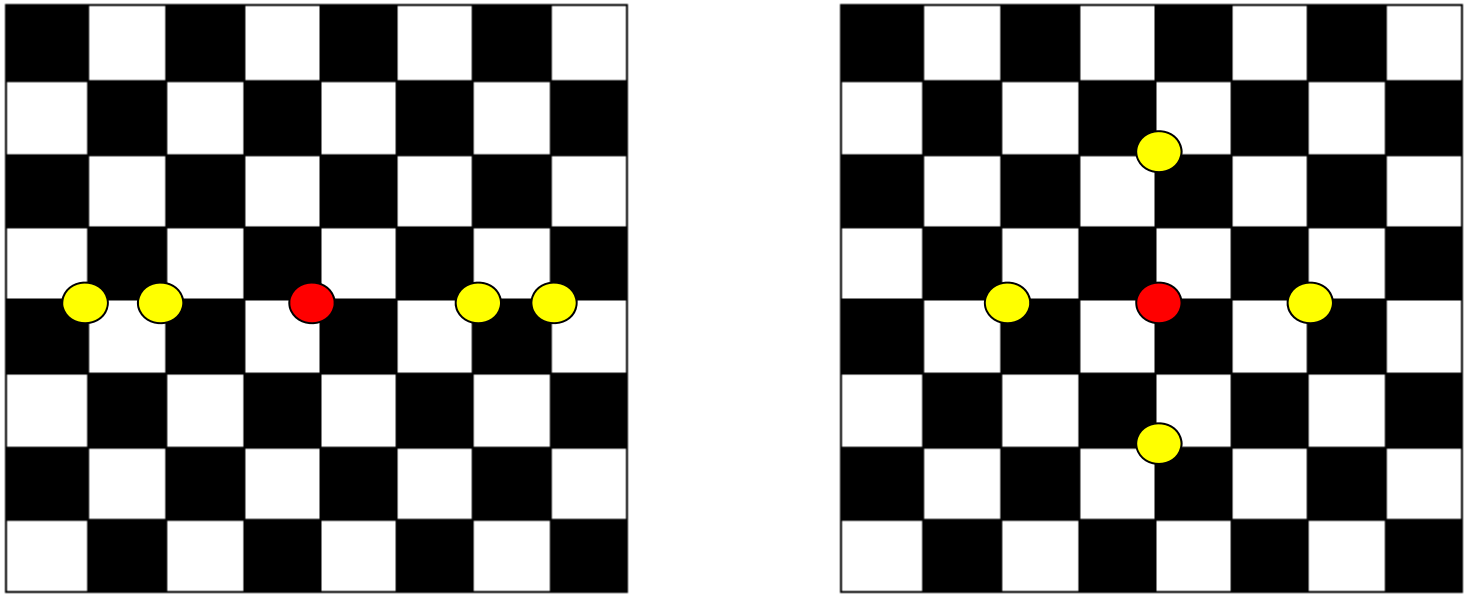
The biggest (abstract ☺) ship that can be hidden?

Optimum A-priori Grid Structure



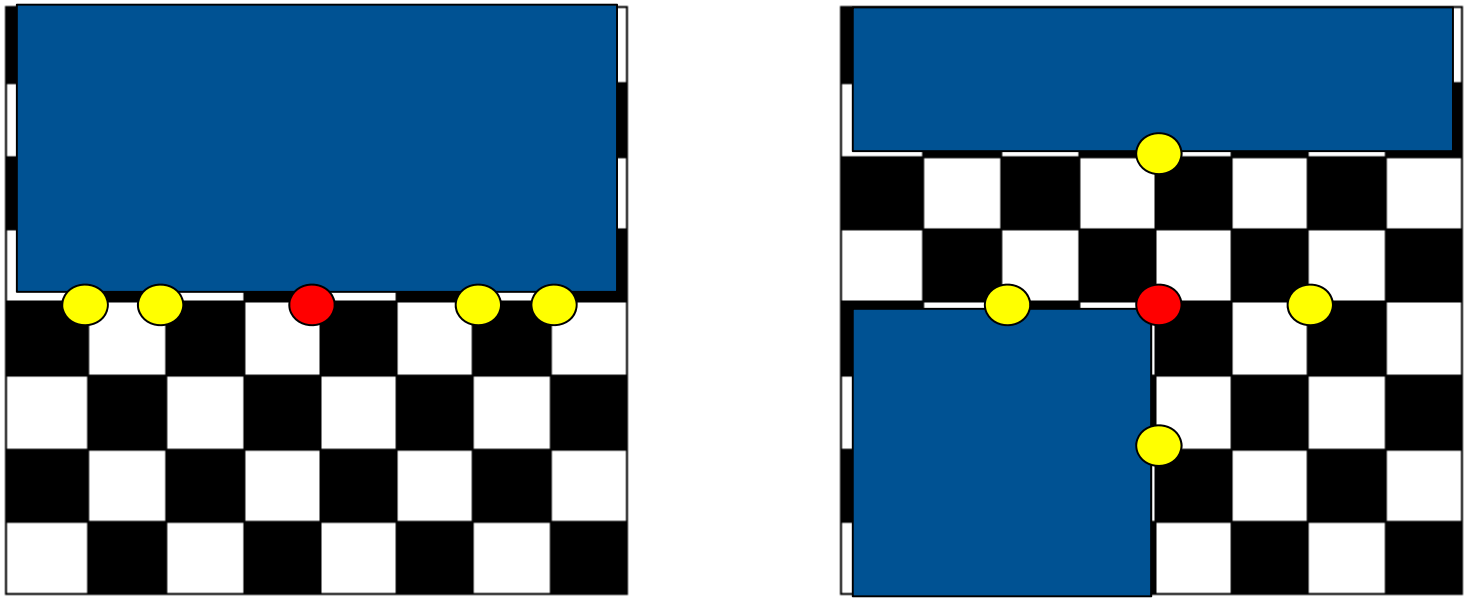
The biggest (abstract ☺) ship that can be hidden?

A-priori Structure – 4 More Points



The biggest (abstract ☺) ship that can be hidden?

A-priori Structure – 4 More Points



The biggest (abstract ☺) ship that can be hidden?

Sparse Grids in a Nutshell

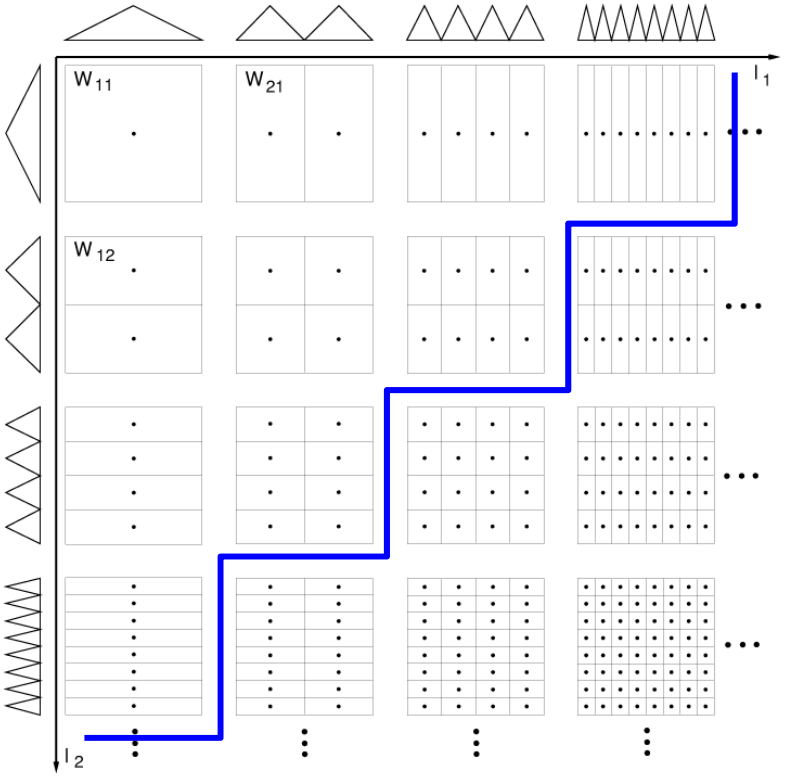
Starting point: finite element thinking!
How to choose grids, elements, basis functions?

- *d=1*: **hierarchical bases** (here linear)
- *d>1*: **tensor product** approach
- **hierarchical subspaces**: collect all basis functions with support of same aspect ratio
- **regularity**: spaces $X(\Omega)$ of bounded mixed derivatives
- discretization / approximation as an **optimisation problem**: find optimum choice of subspaces (cf. **knapsack problem**)

$$\max_{u \in X(\Omega):\ |u|=1} \|u - u_{V(\text{opt})}\|$$

$$= \min_{U:\ |U|=N} \max_{u \in X(\Omega):\ |u|=1} \|u - u_U\|$$

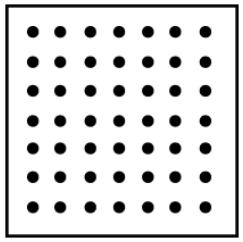
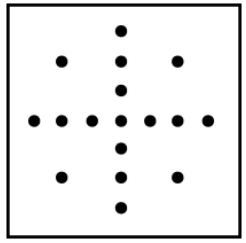
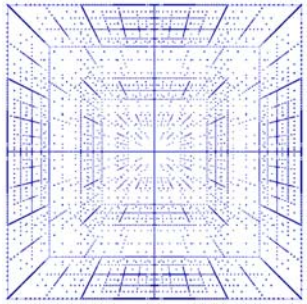
- result: triangular/simplizoidal cut – **sparse grids** [Zenger et al. 1990]



Sparse Grids in a Nutshell

Doing that mathematically in a finite element sense leads to Sparse Grids

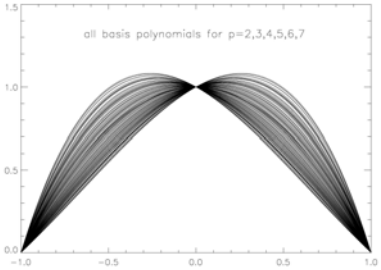
Appearance:

Cost (number of grid points) vs. **benefit** (contribution to interpolant):

(finest mesh width $h_n=2^{-n}$, hierarchical bases of piecewise degree p)

	sparse	full
# grid points	$O(h_n^{-1}\, n^{d-1})$	$O(h_n^{-d})$
error (max, L_2)	$O(h_n^{p+1}\, n^{d-1})$	$O(h_n^{p+1})$
error (energy)	$O(h_n^{p})$	$O(h_n^{p})$

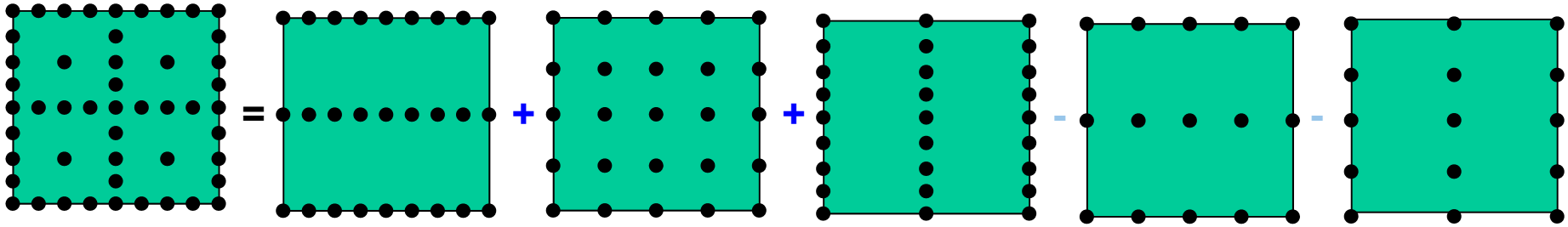


Extensions: straightforward access to **adaptive refinement**, generalisation to **piecewise polynomial** hierarchical bases, energy-optimal sparse grids

Simpler Access to Sparse Grids: Combination Technique

Indirect access via combination technique (a "multi-grid" view):

- extrapolation-type approach – variants: classical, dimension-adaptive, opti-com, ...
- superposition (combination) of several, but much smaller, full grids
- analogous combination of solutions: use standard codes on standard (full) grids



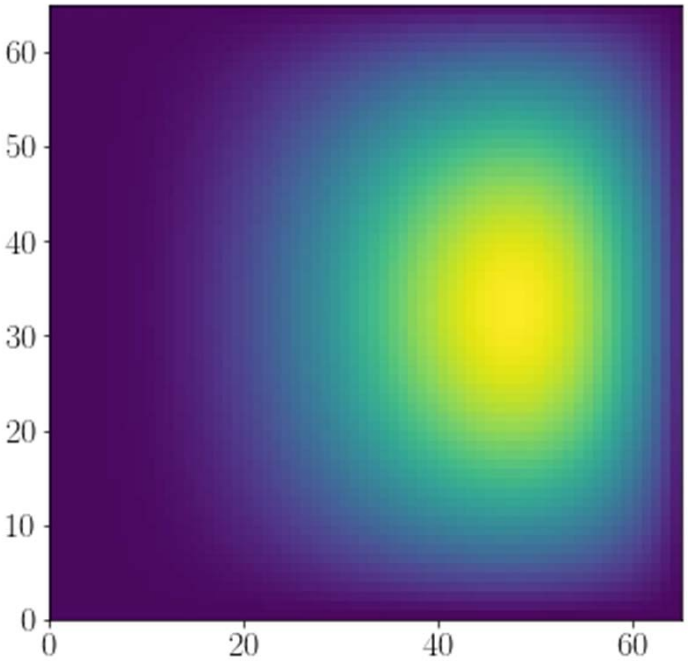
Combination Technique …

- … minimizes communication and synchronization, allows for efficient load balancing
- … provides an additional, asynchronous level of parallelism: more accurate/efficient than Monte Carlo, more decoupled than Domain Decomposition

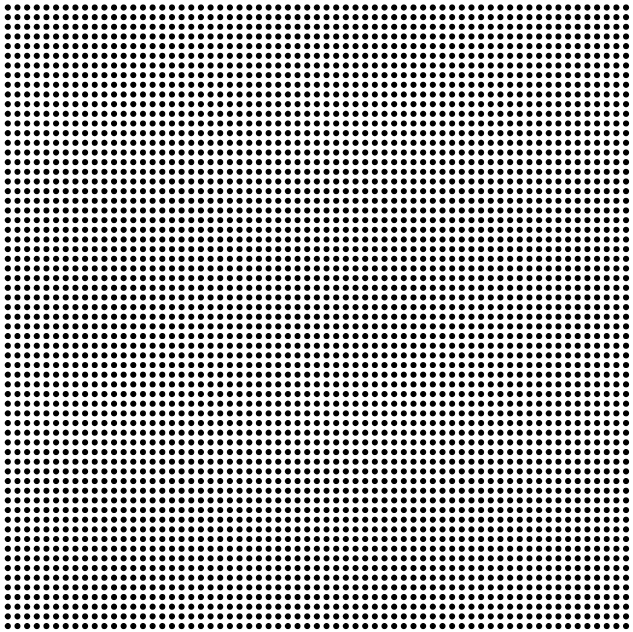
Advantages:

- #1: scalability (communication-light level)
- #2: resilience (error compensation without C/R *PLUS* error and outlier detection)

Solving PDEs on full grids …

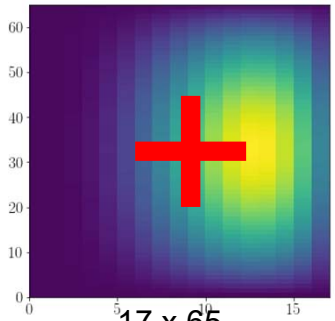


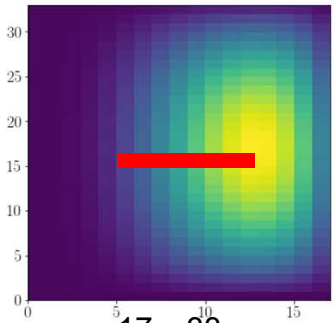
Expensive solution of a 2D PDE

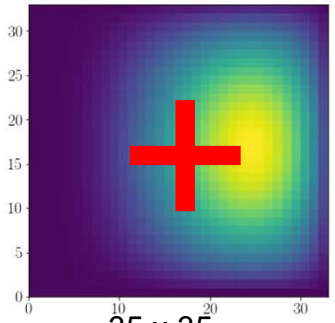


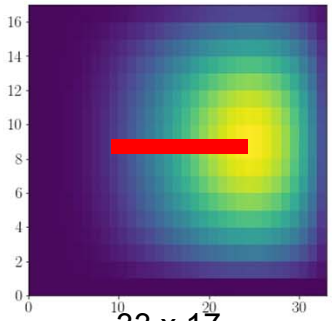
Underlying Cartesian grid (65 x 65)

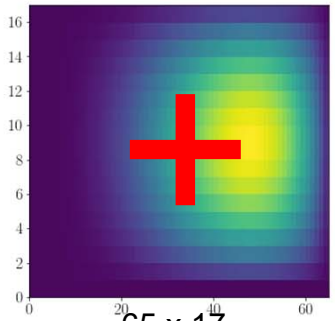
… or with the Combination Technique

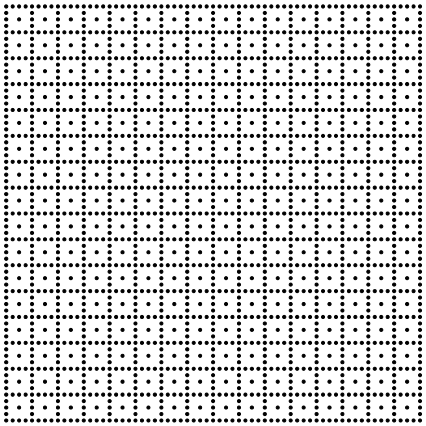












$O(d \log(N)^{d-1})$
grids of size $O(N)$

Does this Break the Curse of Dimensionality?

Example: N = 1024, d = 5

Full grid	**Sparse grid**	**Combination technique**
Total number of points: $1024^5 > 10^{15} \approx$ 1 Billion Mio	Total number of points: 25,416,705 ≈ 26 Mio	Total number of points: 1,876 * 82,000 ≈ 153 Mio

→ Delays the curse of dimensionality, helps a lot – esp. for moderate d

Contents

Sparse Grids – Applications

Num/HPC

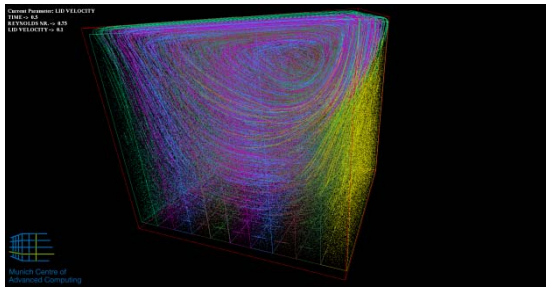
- **Numerical Quadrature**
- **PDE – Continuum Mechanics**
- **PDE – Finance**

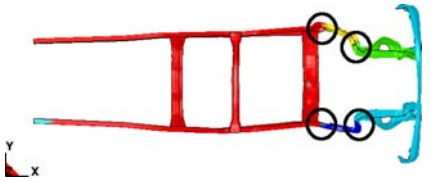
Analytics

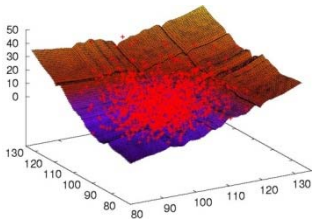
- **Classification & Regression in Data Mining**
- **Clustering**

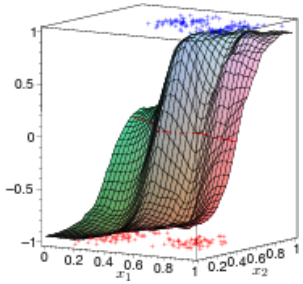
Both

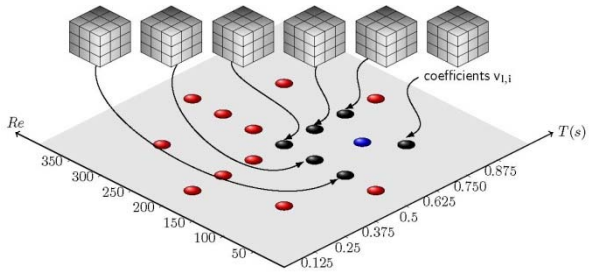
- **Computational Steering**
- **Statistics & Stochastics**

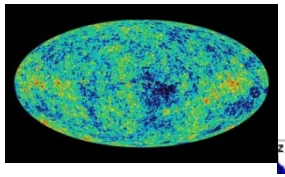


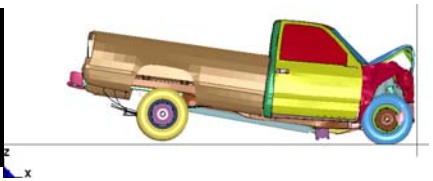












Application #1: Numerical Quadrature

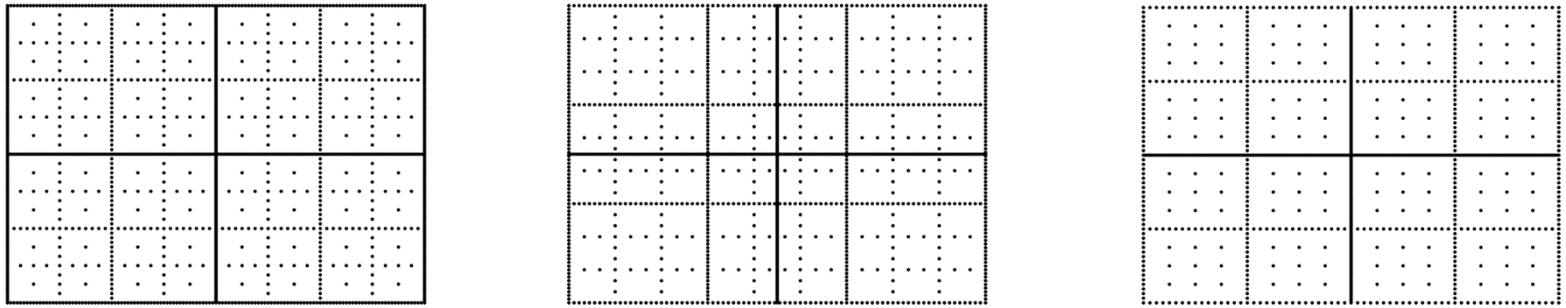
Roots of S.G.: approximation, quadrature

- hyperbolic crosses:

$$\Gamma(n) := \left\{ \mathbf{k} \in^d : \prod_{j=1}^{d} \max\{|k_j|, 1\} \leq n \right\}$$

- Smolyak quadrature:

$$Q_n^{(d)} f := \left(\sum_{i=0}^{n} \left(Q_i^{(1)} - Q_{i-1}^{(1)} \right) \otimes Q_{n-i}^{(d-1)} \right) f$$



Variants:

- Smolyak with trapezoidal rule or Clenshaw-Curtis
- explicit sparse grids (piecewise linear & hierarchical polynomial bases)
- Smolyak with incremental schemes such as Gauß-Patterson
- lattice sampling

Quadrature (moments) with CT so far the probably most widespread SG application

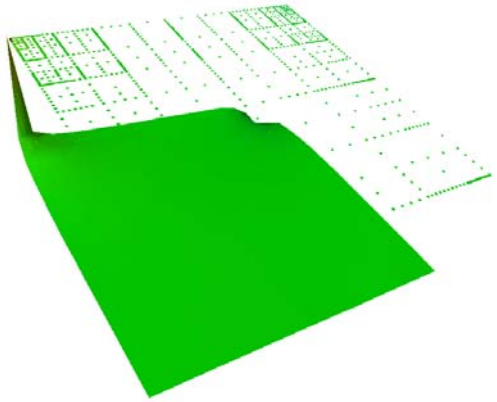
Application #2: PDE – Flows & More

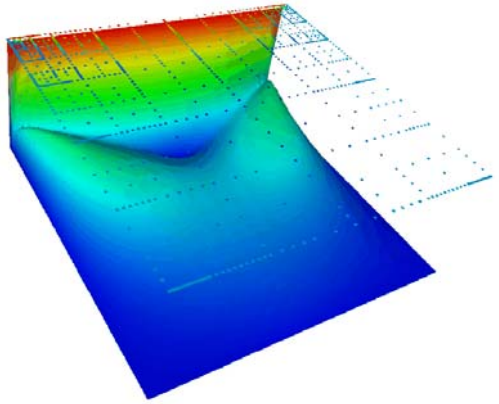
Most work of various groups world-wide done via combination technique:

- Fluid dynamics: Navier-Stokes
- Finance: Black-Scholes
- Quantum mechanics: Schrödinger

Direct discretization:

- finite differences / elements / volumes
- WENO
- Spectral elements



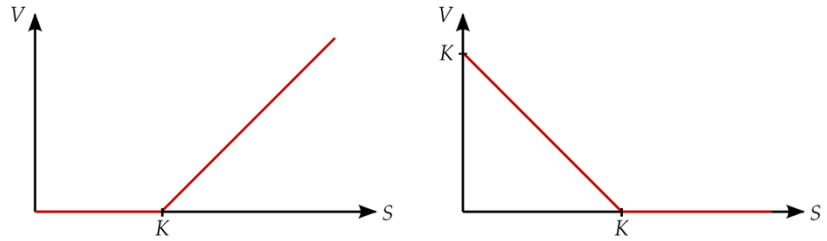


Application #3: PDE – Finance

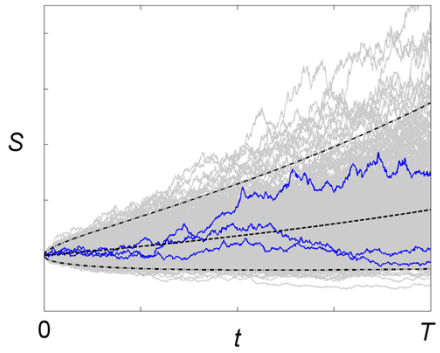
Definition: An option gives the holder the right to buy (Call) or sell (Put) an underlying asset at a certain time T in the future (expiration time) or before for a certain price K (strike price)

Examples: European, American, Asian

Problem 1: inherent non-smoothness



Problem 2: stochastic modeling of stock price development



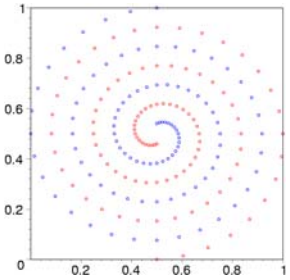
Problem 3: interest in whole baskets, not single assets ➔ high-dim

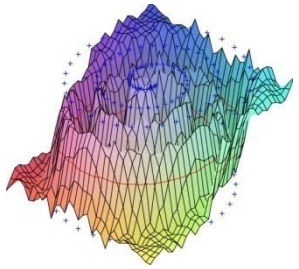
Application #4: Classification & Regression in Mining

- **Problem**: machine learning of a 2-class problem
- Given a **pre-classified data set**

$$S = \{(\mathbf{x}_i, y_i) \in [0,1]^d \times \{-1,1\}\}_{i=1}^{M}$$

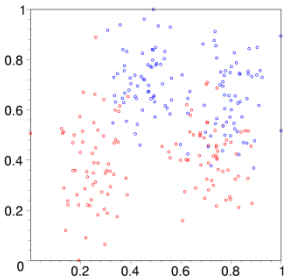
of normalized data points $\mathbf{x}_i$ with class labels y_i
(regression: real numbers instead of discrete labels)

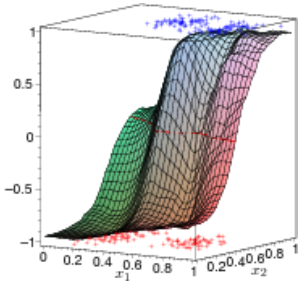




- **Typically: presence of noise**

[sampling ➔ noise ➔ no mere interpolation]





- **Computational task:**
  - construct classifier/ machine learner (ML)

$$f : [0,1]^d \to \{-1,1\}$$

  - provides class predictions when applied to new data points

Regularization Network Approach

- Classification as scattered data approximation problem *plus* additional regularization terms (ill-posed problem, noise):

$$\text{minimize } H[f] = \frac{1}{M}\sum_{i=1}^{M} \mathcal{V}(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2$$

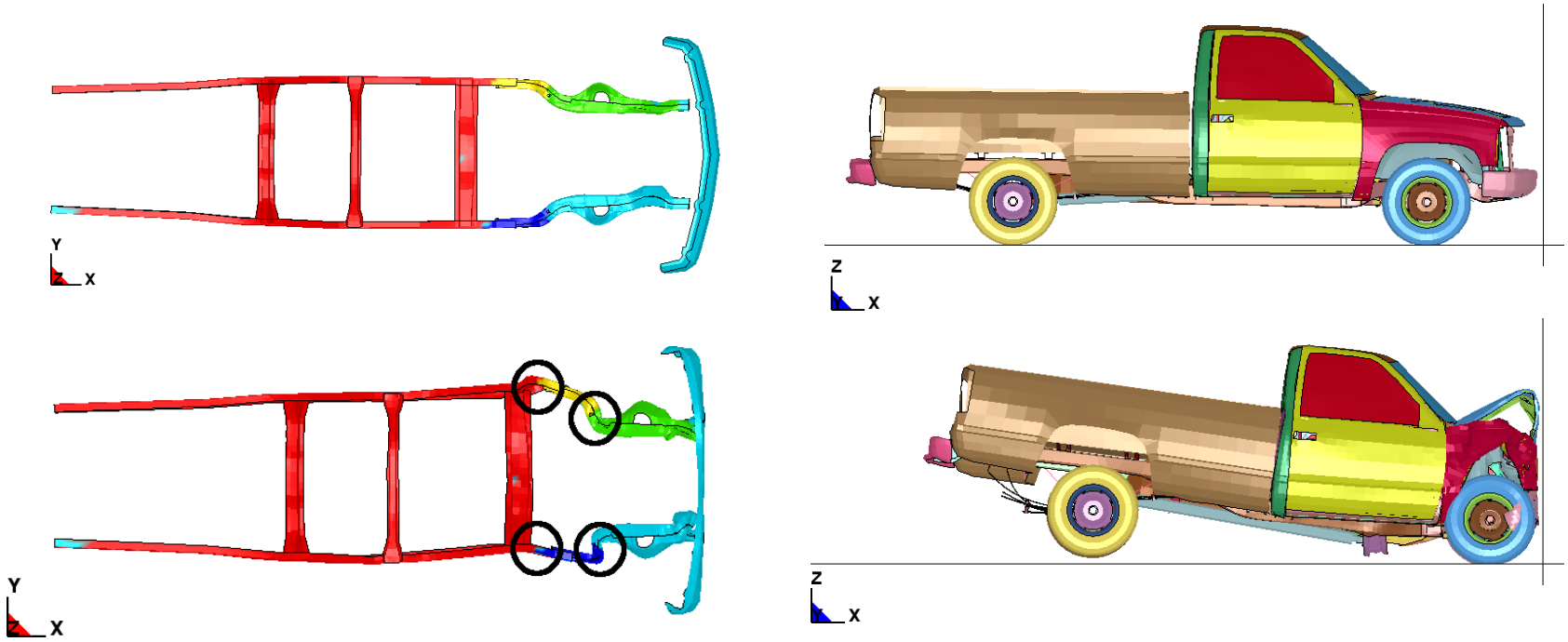
  - cost/error function, for example $\mathcal{V} := (y_i - f(\mathbf{x}_i))^2$
  - regularization operator/stabilizer, for example $\|f\|_K^2 := \|\nabla f\|_{L_2}^2$
  - simpler, but astonishingly useful

$$\text{minimize } H[f] = \frac{1}{M}\sum_{i=1}^{M}(y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{i=1}^{N} \alpha_i^2$$

  - regularization parameter λ, $f_N(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_i(\mathbf{x})$
- Minimize trade-off between cost and smoothness via λ
- Various approaches (neural networks, support-vector-machines) can be formulated as Regularization Network Approach
- **Common classification algorithms**:
  - discretization of feature space not feasible (curse of dimension, $O(N^d)$)
  - global ansatz functions associated to data points ($O(M^2)$, large training data?)
- **Idea: use sparse grids to discretize feature space – *linear in data size***

Example

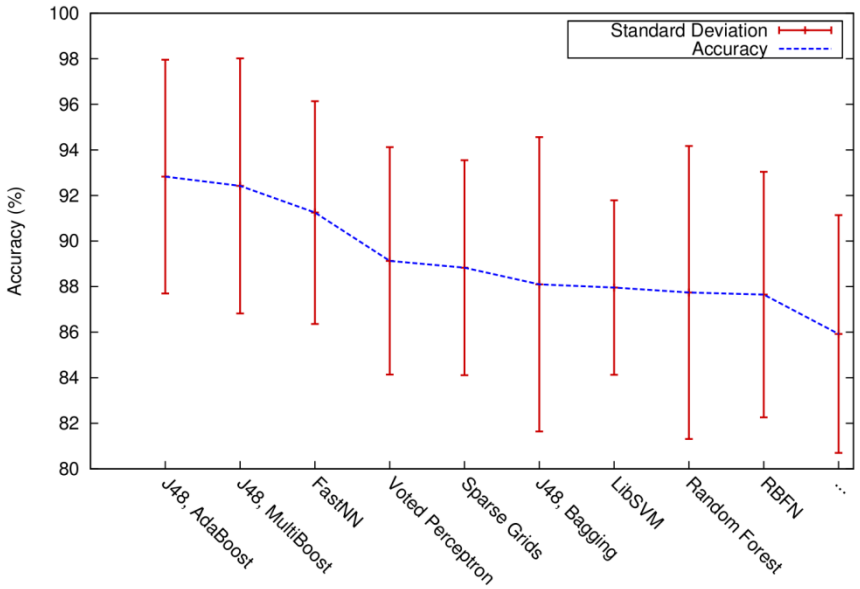
Learning from crash (simulation) data

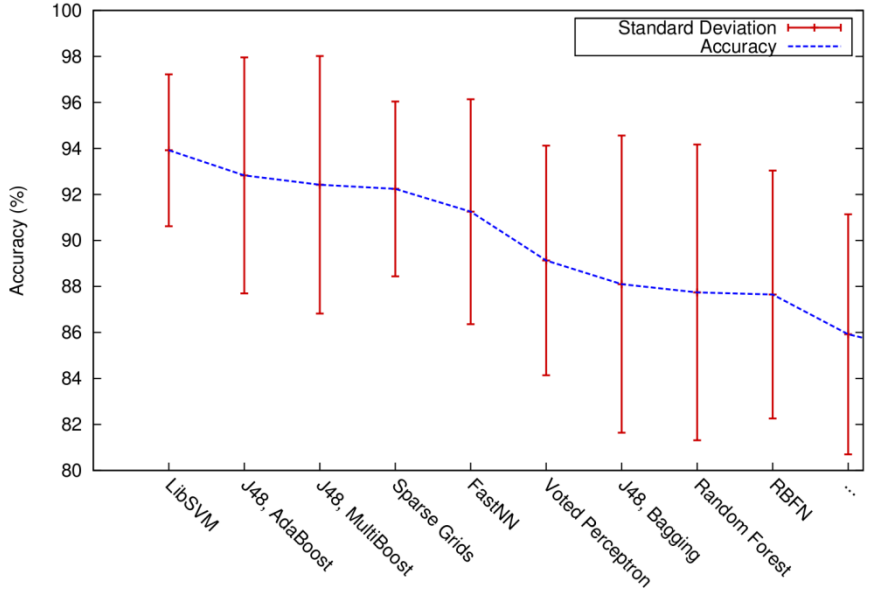


Towards Higher Dimensionalities

Musk data sets, *d=166*

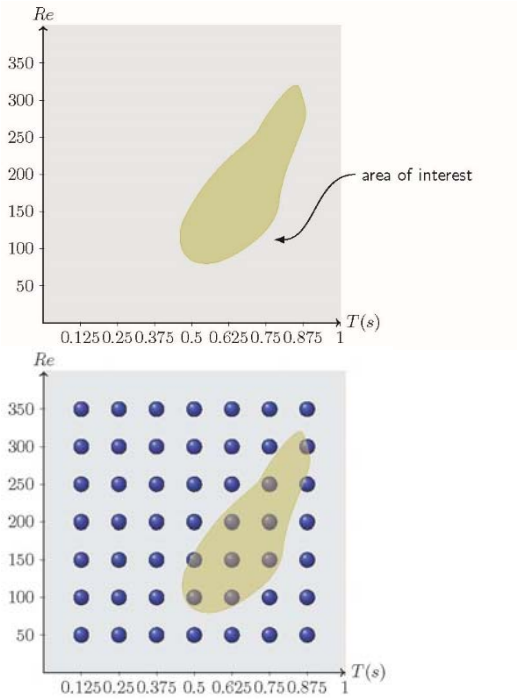
- separate molecules (166 attributes, mainly describing distance features of conformations)
- 10x 10-fold cross-validation
- M=476 (left) and smaller data set (right, with PCA leading to *d=35*)
- only two refinements before PCA, more possible after PCA
- benchmark study done 2008 to compare 45 classification algorithms; best 9 out of 38

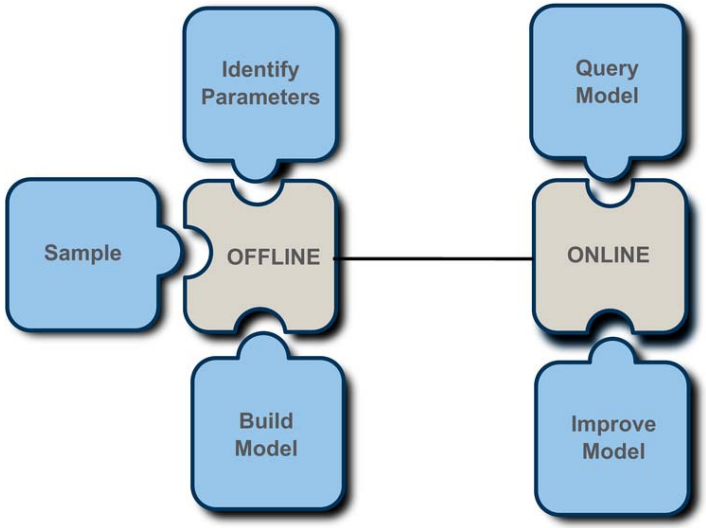




Example #5: Computational Steering

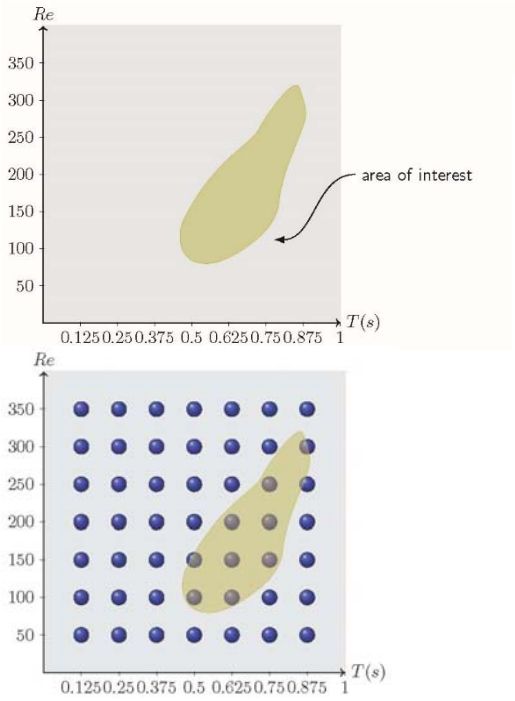
Goal: allow for interaction in high-dim parameter-dependent simulation scenarios
Idea: use a sparse grid of pre-computed simulation data sets (offline/online)
Related: model order reduction, surrogate models, reduced basis functions

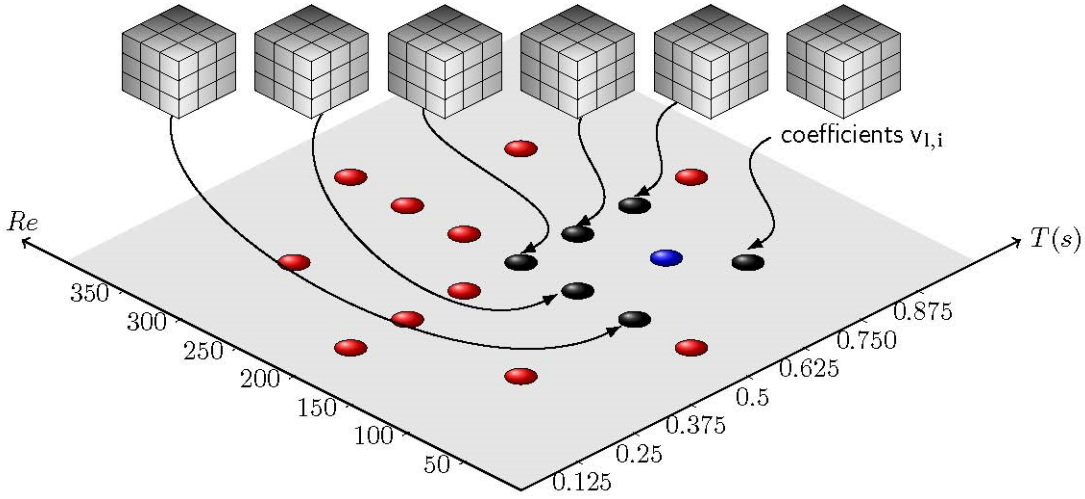




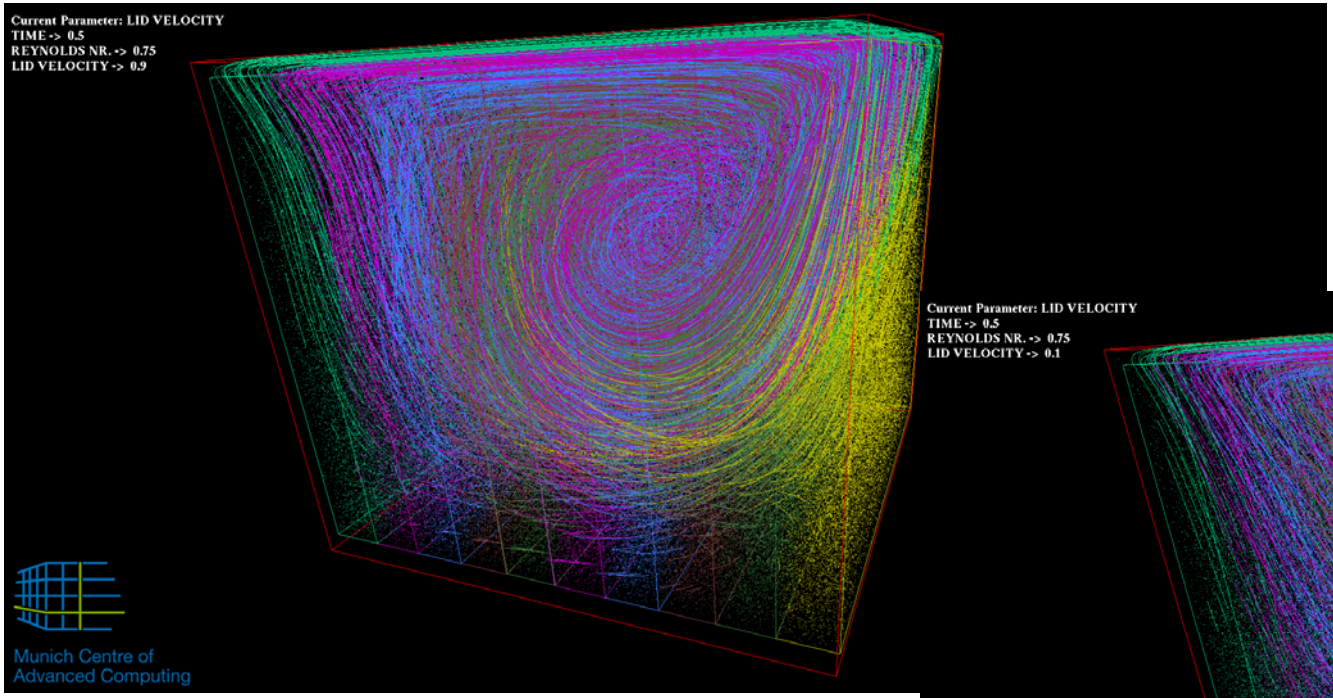
Example #5: Computational Steering

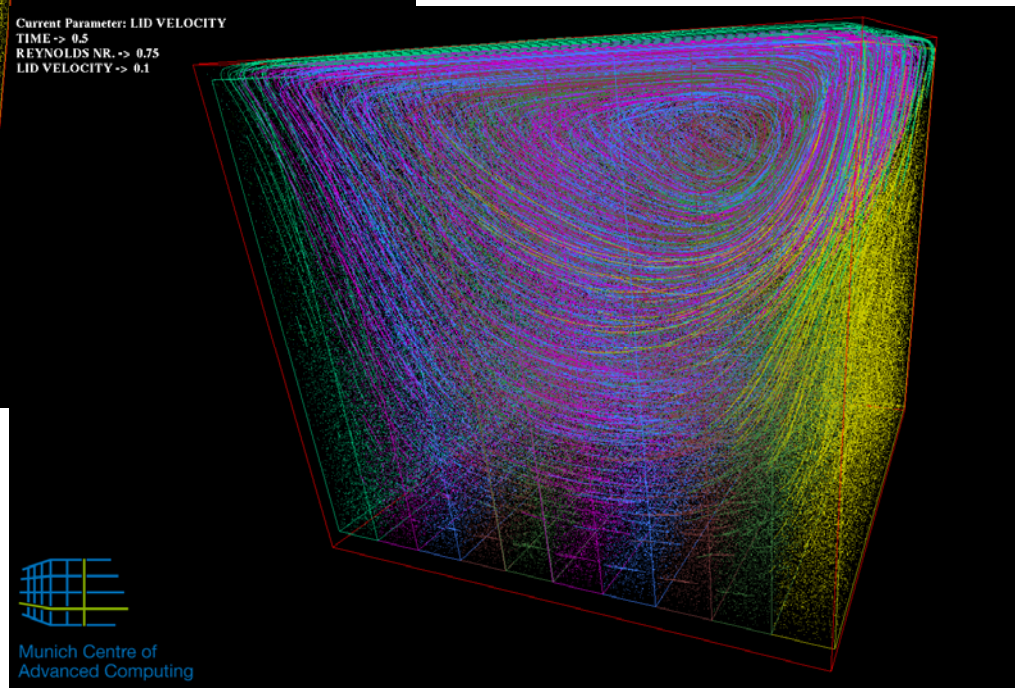
Goal: allow for interaction in high-dim parameter-dependent simulation scenarios
Idea: use a sparse grid of pre-computed simulation data sets (offline/online)
Related: model order reduction, surrogate models, reduced basis functions





Example: Lid-Driven Cavity

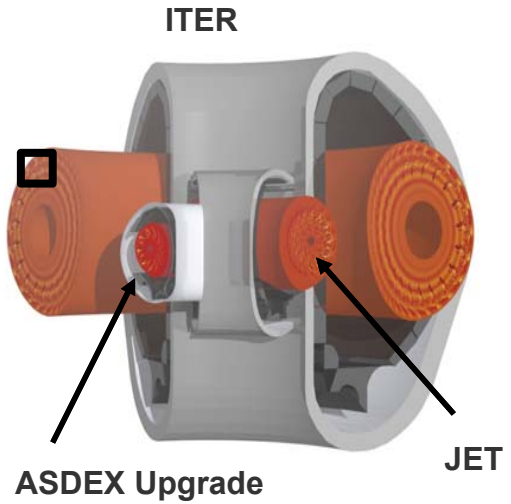




Application #6: Simulation of Plasmas … and More

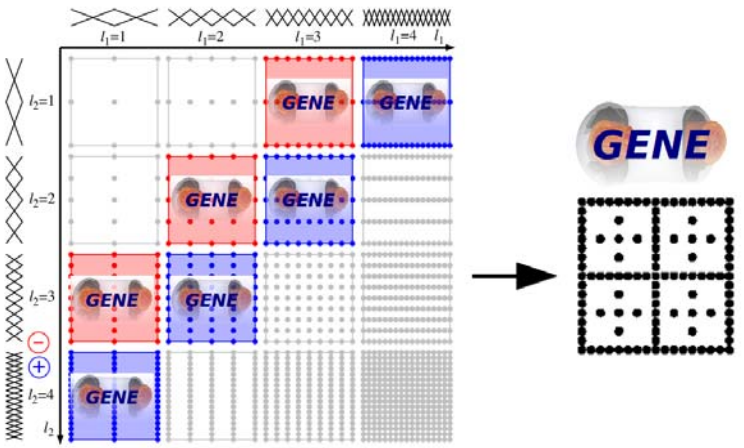
How?

- State-of-the-art plasma simulation code: ***GENE*** (tackling the **5D** gyrokinetic Vlasov equation)
- Gives invaluable insight into the turbulent flow of confined plasmas → a limiting factor for successful fusion reactions



State of the art:

- Excellent performance, up to > 260 k cores
- Yet, only a small section of a reactor can be simulated
- Ongoing algorithmic improvements (solver, adaptivity, …)
- Here: tackling the high dimensionality issue with the combination technique
- i.e.: run GENE on each subgrid, combine
- Allows for the simulation of scenarios that have been out of range so far

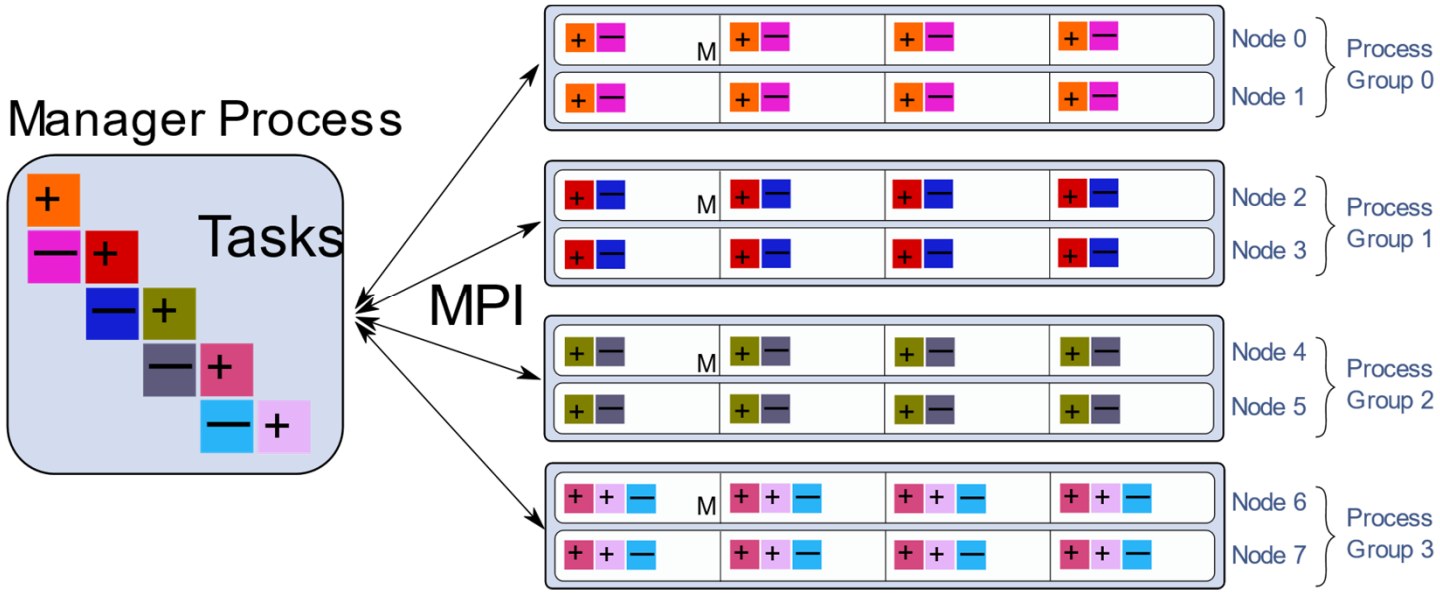


Combination Technique Revisited

- **Minimizes communication and synchronization**
- **Additional level of parallelism (sub-problems, cf. MC & DD)**
  - More accurate/efficient than Monte Carlo
  - More decoupled than Domain Decomposition
- **Many variants: OptiCom, iterative Opticom, …**
- **Advantage #1: scalability**
  - Communication-light level
- **Advantage #2: resilience**
  - Error compensation w/o checkpoint/restart *PLUS* error and outlier detection
- **Advantage #3: load balancing**
  - Job deployment adapts to topology
  - Coarse- and fine-grained scheduling
- **Topic of the SPPEXA project EXAHD (partners: T. Dannert, M. Griebel, M. Hegland, F. Jenko, D. Pflüger) – pilot: plasma physics**

A Highly Parallel and Fault-tolerant Framework

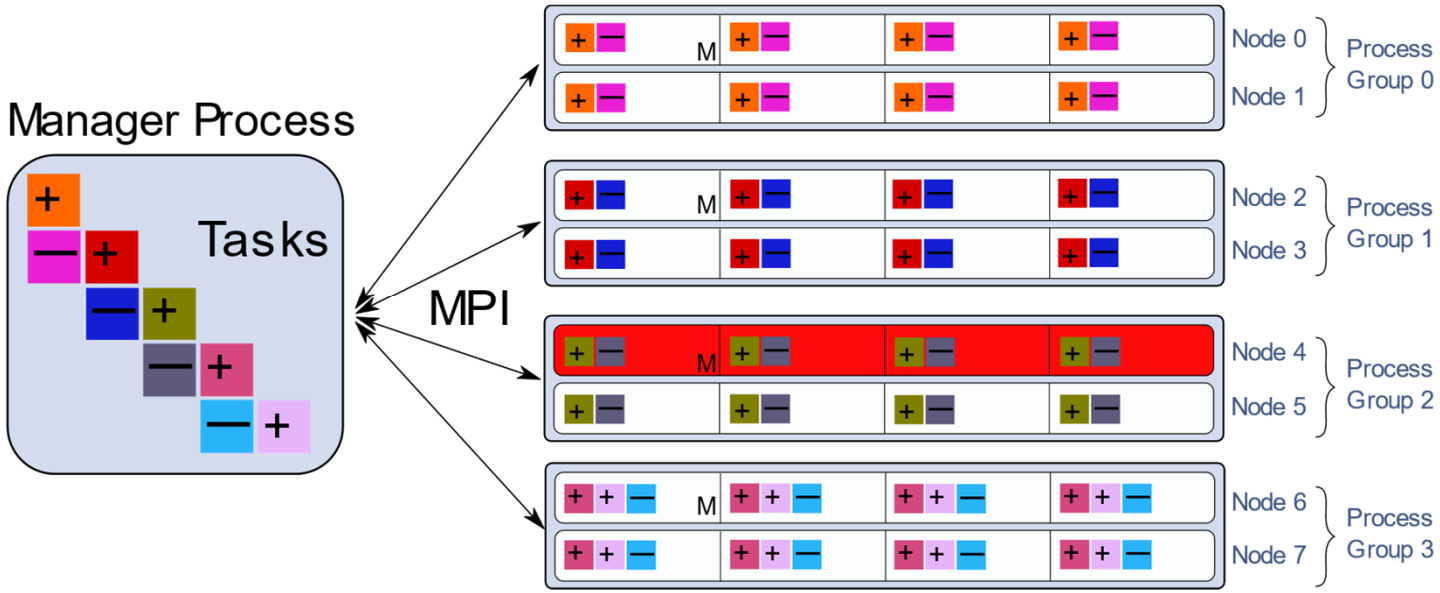
➔ Algorithm-Based Fault Tolerance



- Manager worker approach
- Tasks distributed to Process Groups
- 2 Layers of parallelism

A Highly Parallel and Fault-tolerant Framework

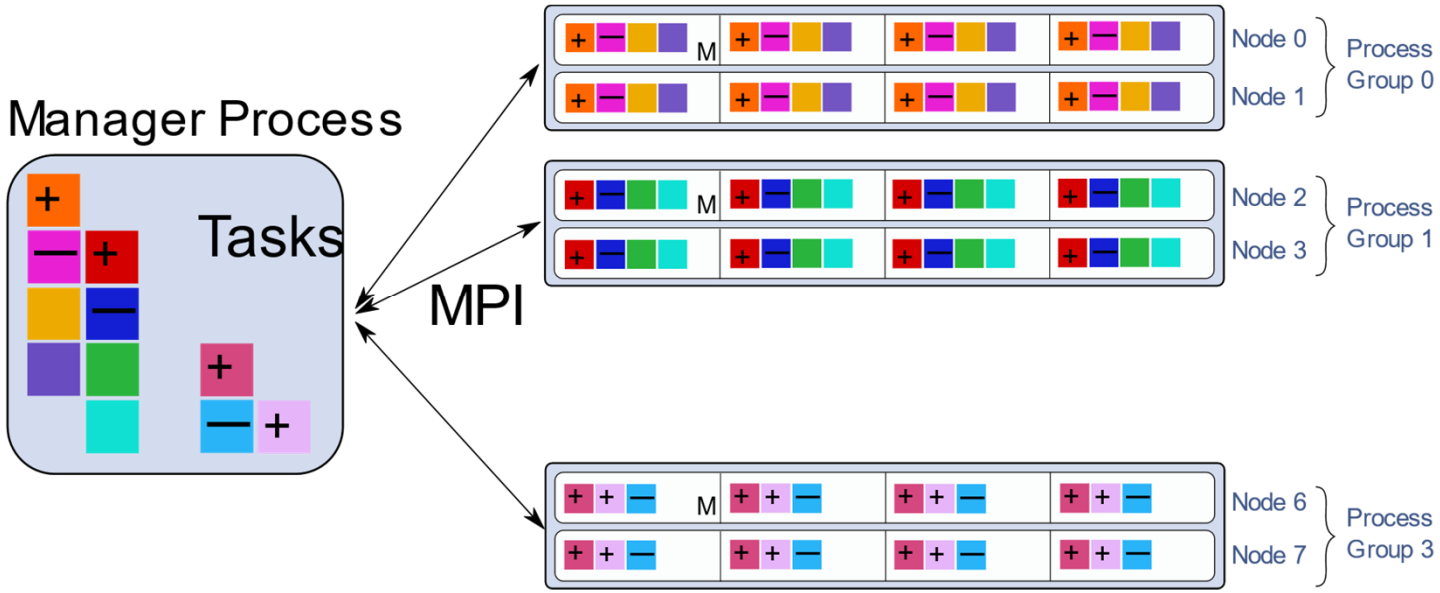
➔ Algorithm-Based Fault Tolerance



- Failing groups are detected

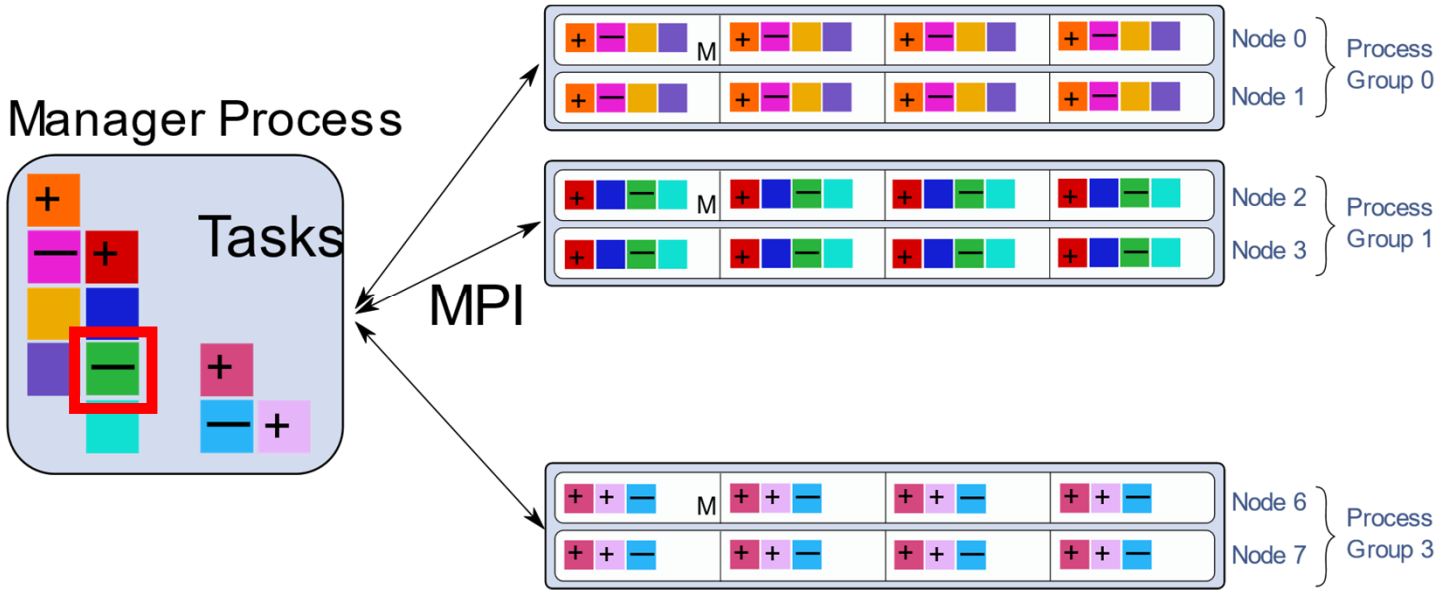
A Highly Parallel and Fault-tolerant Framework

➔ Algorithm-Based Fault Tolerance



- Failing groups are detected
- Failed grids are lost

A highly parallel and fault-tolerant framework



- Failing groups are detected
- Failed grids are lost
- Compute alternative combination scheme

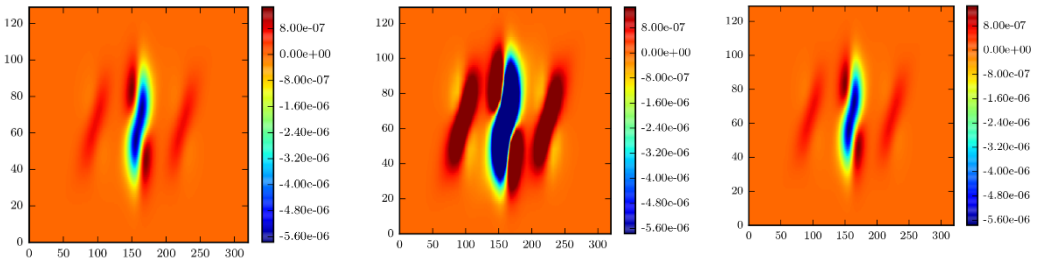
→ Lossy recovery but still very good result

→ No recomputing necessary

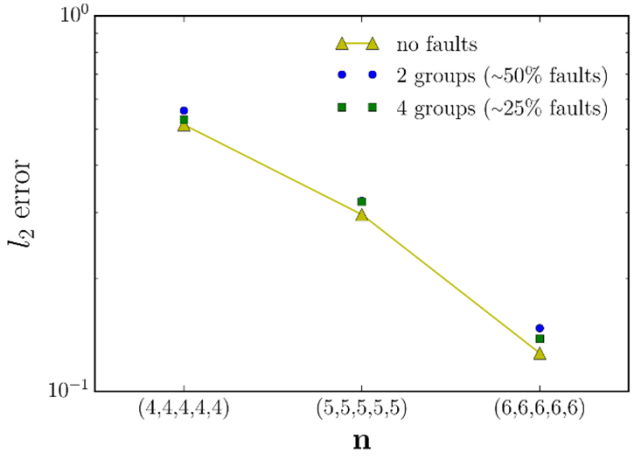
ABFT – Initial Results

Initial tests with GENE show a very good approximation quality after fault recovery …

… which holds for the Picture Norm ☺, but also in a more quantitative sense



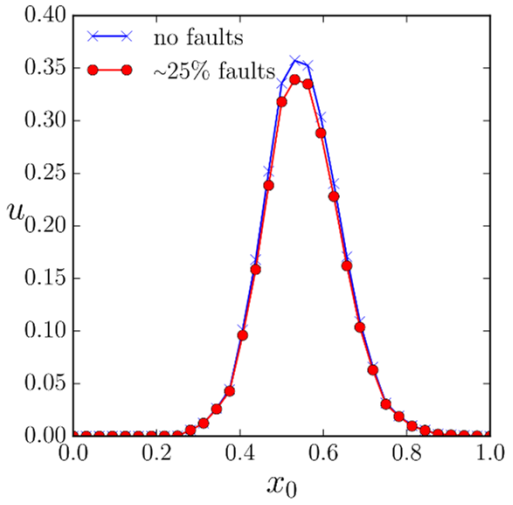
Original *With faults* *After recovery*



1 core fails

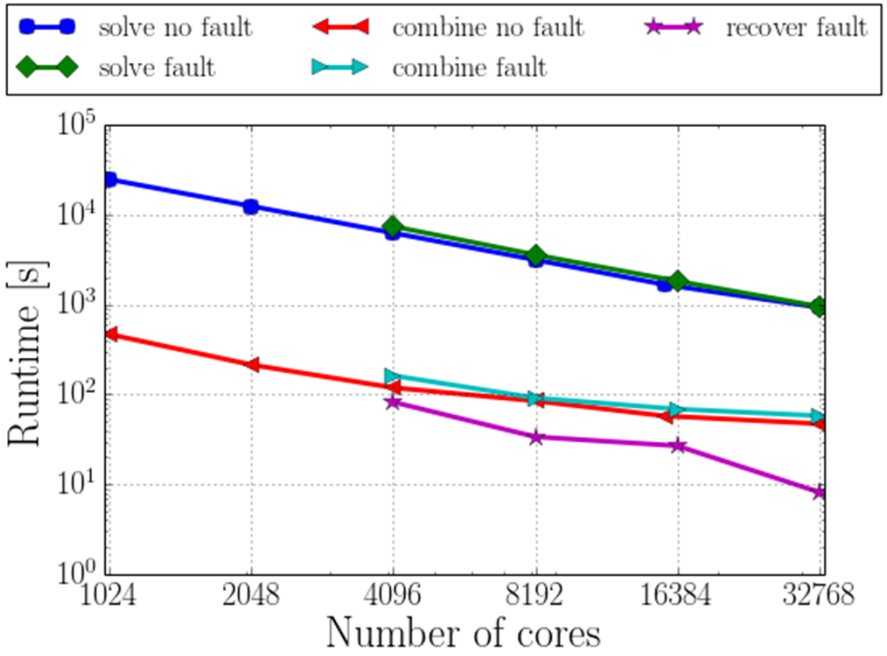
➔

whole processor group excluded

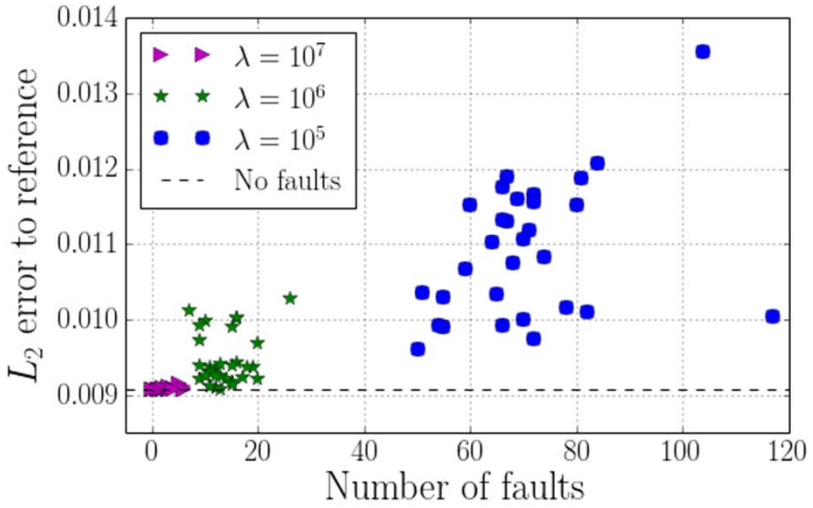


ABFT – More Sophisticated

Parallel tests on *Hazel Hen* (HLRS): **similar scaling with faults, small overhead**



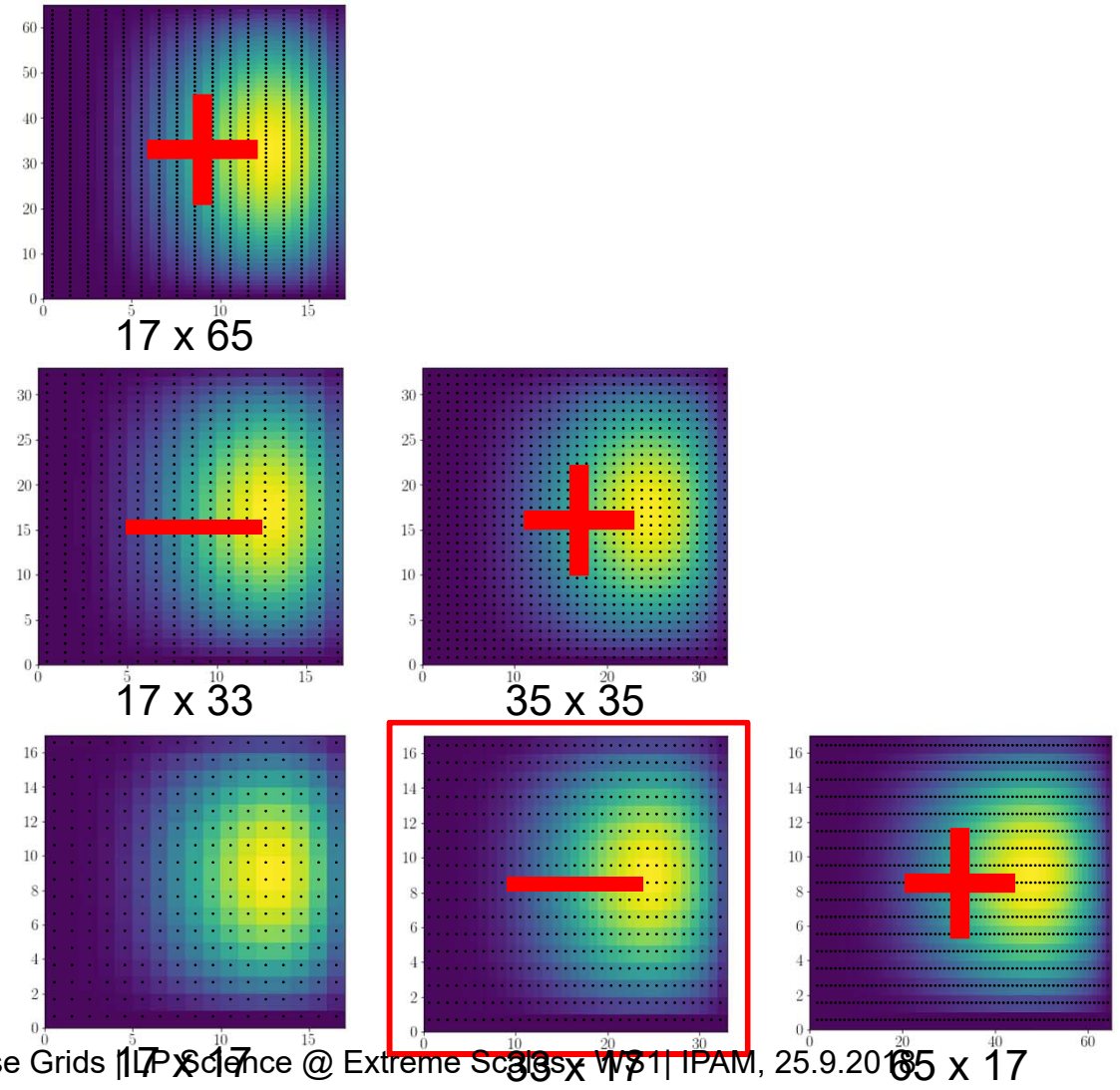
Error analysis with simulated faults: **low error even in massively faulty environment**



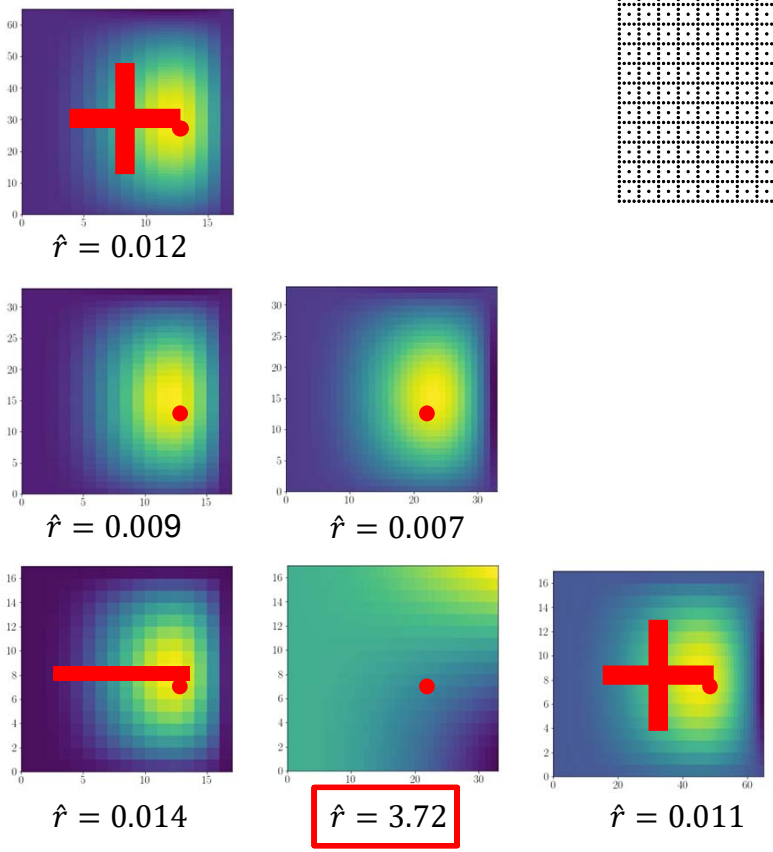
For the study: faults are distributed according to the **Weibull distribution** (k=0.7):

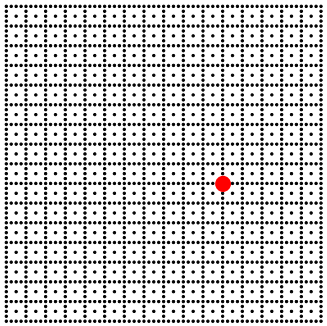
$$f(x;k,\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{(k-1)} e^{-\left(\frac{x}{\lambda}\right)^k}$$

And what about Silent Data Corruption?



Algorithmic Solution: Detect Outlier Fields





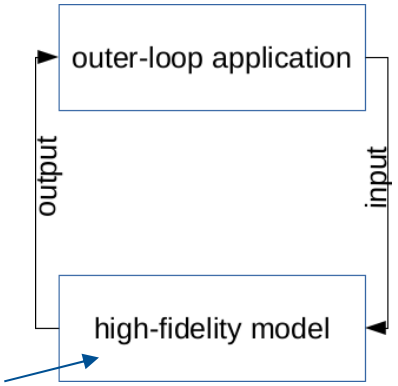
1. Identify grid point with largest difference
2. Perform robust regression to fit values

→ Find outliers
→ No assumptions about SDC
→ General advantage of hierarchical / multi-level schemes

Application #7: Uncertainty Quantification

General outer-loop applications: challenges

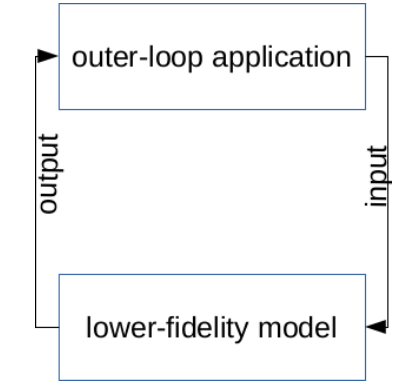
- "outer-loop scenarios": optimization, data assimilation, multi-physics problems, control, **uncertainty propagation**, inverse problems, ...
- usually depend on "high-fidelity" models (e.g., PDEs) showing
  - high computational demands
  - high dimensionality
  - high data throughput
  - ...



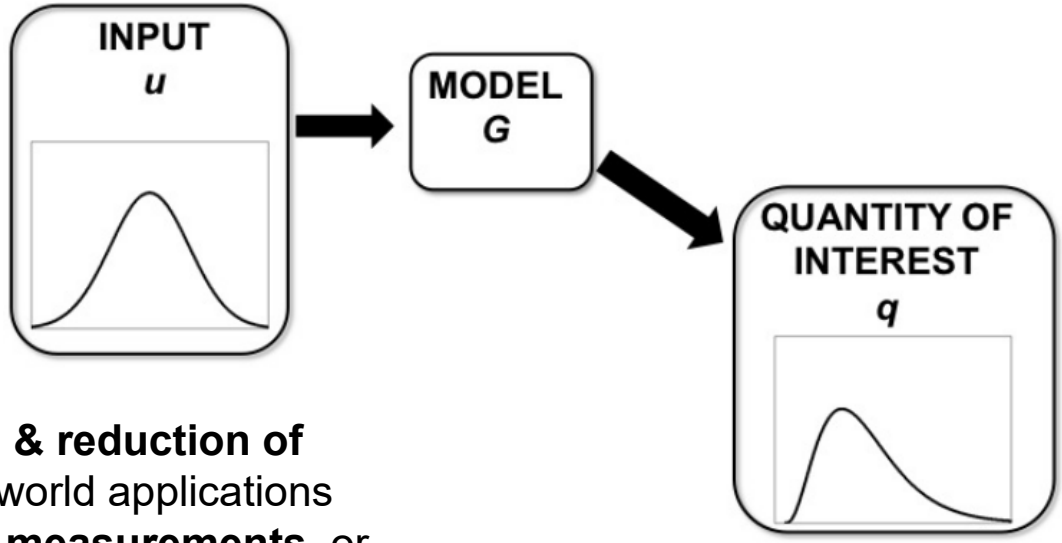
Classical HPC simulation

Outer-loop applications: solutions

- replace the hi-fi model with a "lower-fidelity" model that
  - is **non-intrusive**, i.e., the hi-fi code serves as a **black-box**
  - is less accurate, but **computationally cheap**
  - **exploits** the **structure** of the problem at hand
- to obtain lower-fidelity models, we need
  - enhanced, structure-exploiting algorithms → **adaptivity**
  - **HPC** → embarrassingly parallel calls of the hi-fi model



Uncertainty Quantification



UQ:

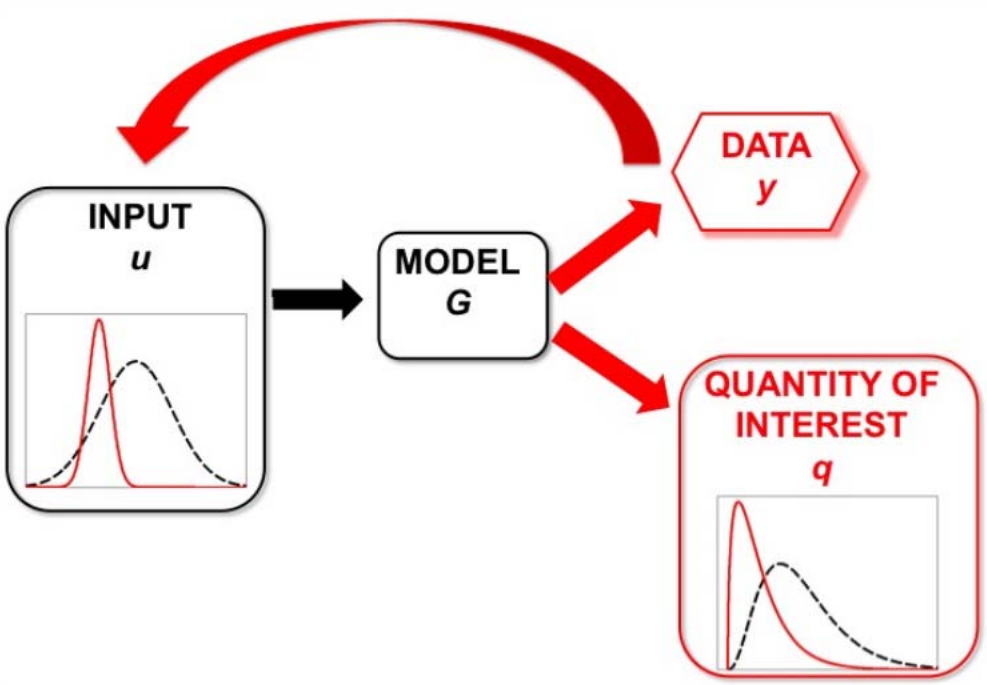
- Field of **quantitative characterization & reduction of uncertainties** in computations or real-world applications
- Consider uncertainties in **parameters**, **measurements**, or **models** in your system
- Classical appearance: **propagate uncertainties** *u* through model *G* and find solutions *q* usually in form of statistics, e.g. mean $\mathbb{E}$ or variance $\mathbb{V}$
- Goal: improve **reliability** of simulations/predictions
- Adds another layer of complexity compared to deterministic problems (“outer loop”)

Sources of uncertainties:

- Measurement errors
- Experimental errors
- Modeling errors
- Numerical errors
- …

One more layer of complexity – inverse problems

- **Large amount of output data *y***
- **Task: find matching input parameters *u***
- Problem: often ill-posed (no unique solution)
- Possible approaches:
  - Apply Bayes' apparatus to estimate input distribution
  - Formulate as optimization problem and fit output to data
- Examples:
  - Groundwater flow problems
  - Identification of cracks from surface measurements
  - Medical imaging
  - …



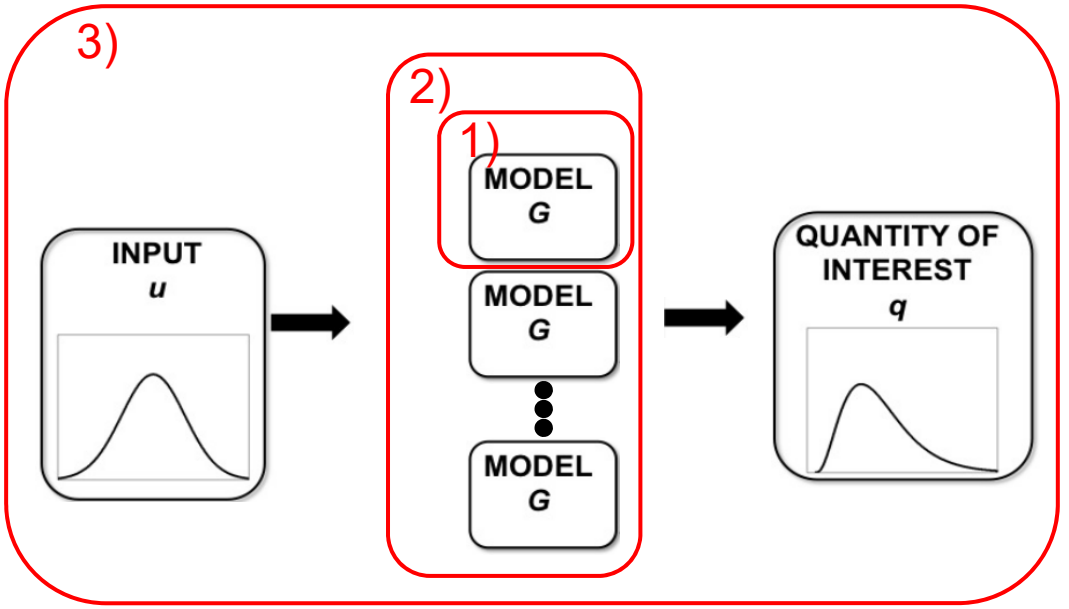
UQ introduces additional HPC demand

Level of parallelism in UQ:

1) Parallel implementation of deterministic model
2) Evaluations of statistics, e.g. through Monte Carlo approach:

$$q = \mathbb{E}[u] = \frac{1}{N}\sum_{i=1}^{N} G(u_i)$$

3) Parallel processing of different
   - input data,
   - deterministic parameters
   - models
   - …

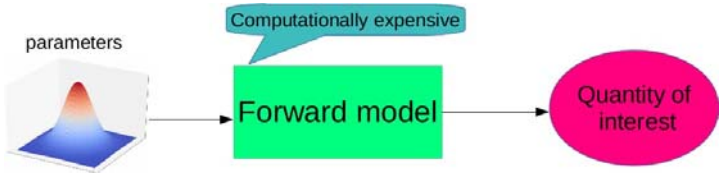


Trade-off:
Use limited resources for deterministic solver or parallel UQ runs?

Uncertainty propagation showcase: Sensitivity-driven adaptive strategies

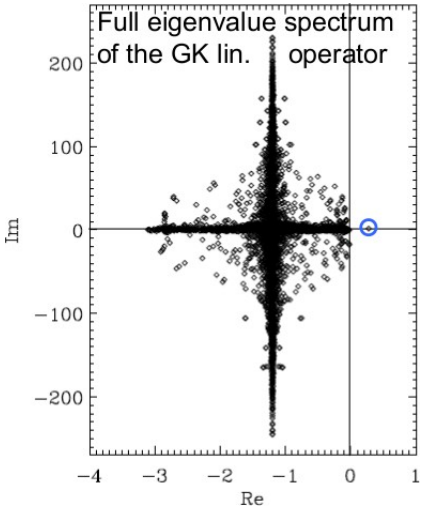
Problem description

- **uncertainty propagation**: given uncertain inputs, find output quantities of interest (QoI)
- example QoI: expectation, standard deviation, sensitivity coefficients



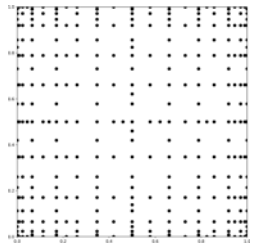
Target application

- gyrokinetic plasma micro-turbulence simulation
- 5D integro-differential Vlasov-Maxwell system of eqs.
- employ the **HPC**-ready **code GENE**
- here: restriction to linear physics
  - simpler testbed
  - insights into sensitivies of underlying micro-instabilities
- output of interest: growth rate of the micro-instability
- the quantification of uncertainty is paramount



Sensitivity-driven adaptive strategies – our approach

Design a **non-intrusive**, **adaptive**, **sensitivity-driven** approach that



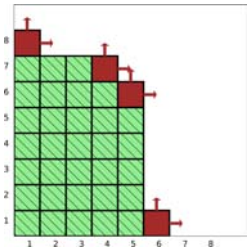
- delays the “curse of dimensionality”

  → **sparse approximations**, based on **interpolation**

- **reduces** the overall computational cost

  → **dimension-adaptivity:**

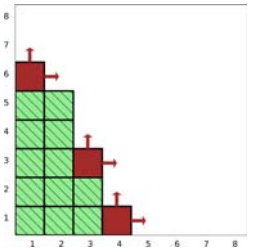
  - construct the approximation adaptively
  - add new subspaces based on suitable metrics
  - **HPC-**friendly: multiple layers of parallelism possible



- **exploits** the **structure** of the problem at hand

  → **sensitivity-driven adaptivity**

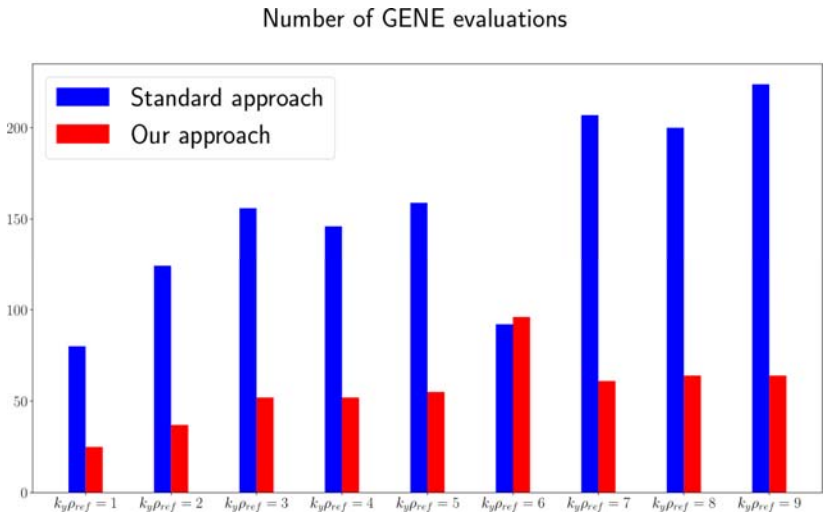
  - use sensitivity information to drive the adaptive process
  - prevents the algorithm to refine in “unimportant” directions



Sensitivity-driven adaptive strategies – our approach

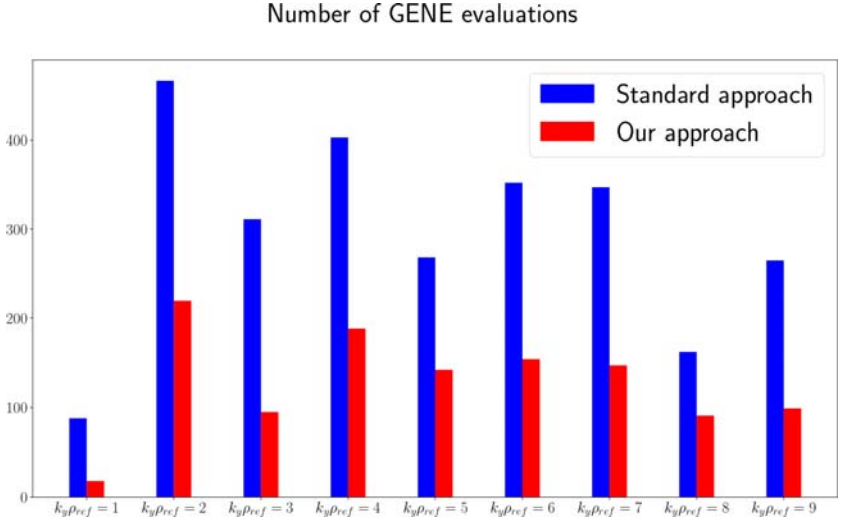
Results: benchmark problem

- 8 uncertain physical inputs
- 32 cores/simulation
- 2 layers of parallelism
- simulation time up to several minutes/sim
- 9 UQ simulations (9 wave numbers $k_y \rho_{ref}$)



Results: real-world problem

- 3 uncertain physical inputs
- 32 cores/simulation
- 2 layers of parallelism
- simulation time up to few hours/sim
- 9 UQ simulations (9 wave numbers $k_y \rho_{ref}$)



Contents

Concluding Remarks

Sparse grids are where HPC and BD meet

Sparse grids can't solve all problems … but hopefully some …

There is some software available:

- In particular: ***sg++*** , cf. http://sgpp.sparsegrids.org
- SG module in *Tasmanian* (ORNL)
- *swMATH*: SG interpolation toolbox (MATLAB)
- A couple of specific libraries, such as *fastsg*
- Included in *Dakota* (SNL)
- …

Acknowledgements

Input & Collaboration:

- ExaHD: D. Pflüger ++, M. Griebel ++, A. Parra, M. Obersteiner, F. Jenko ++
- Quadrature: M. Griebel ++, T. Gerstner, S. Dirnstorfer
- Analytics: M. Griebel ++, D. Pflüger ++, J. Garcke, B. Peherstorfer, V. Khakhutskyy
- Steering: D. Butnaru
- UQ: T. Neckel, I. Farcas, F. Menhorn

Compute resources:

- Mainly HLRS (Stuttgart) and LRZ (Munich)

Funding:

- DFG (PP SPPEXA-ExaHD, G8-NuFUSE)
- BMBF (FIDEUM, SimData)
- HGF (HEPP)
- Intel (IPCC @ TUM & LRZ), TUM Institute for Advanced Study

Sorry for all I forgot – it will probably happen again, but hopefully with others …

150 Jahre culture of excellence

Thanks for your attention!