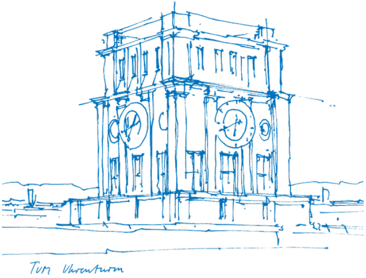
Algorithms of Scientific Computing

Overview and General Remarks

Michael Bader
Technical University of Munich

Summer 2022



Classification of the Lecture – Who is Who?

Students of Informatics:

- Informatics Bachelor & Master
- Informatics: Games Engineering (Bachelor & Master)
- Information Systems (Wirtschaftsinformatik)
- Data Engineering and Analytics

Students of Mathematics:

- Mathematics in Data Science
- Bachelor, Master, as minor?

Students of (Computational) (Science and) Engineering:

- Computational Science and Engineering (CSE): elective (cat. E)
- Engineering Science
- Physics in various "flavours"

... anyone else? *Warm Welcome!*

Lectures

Lecturers:

- Michael Bader
- Felix Dietrich, Christian Mendl, Tobias Neckel
  (recorded lectures from summer 2020)

Time & Day:

- by default, on **Mondays** (14 c.t.) and **Wednesdays** (12.00)

"Style": online-enhanced presence teaching

- recorded lectures from summer 2020
  → can be watched to prepare for (or rework) presence lectures
- presence lectures
  → aim at focusing on use cases, examples, **questions**, discussions, etc.
  → live streaming (and recording) via live.rbg.tum.de on Mondays
  → presence-only on Wednesdays

Tutorials

Tutor:

- Mario Wille

Time & Day:

- by default, tutorials will be on **Fridays** (10 c.t., MI HS 2)
- first tutorial on Apr 29:
  includes introduction to **Jupyter Notebook**

“Style”:

- worksheets with applications & examples
- no compulsory part

Lecture Slides: Color Code for Headers

Black Headers:

- for all slides with regular topics

Green Headers:

- summarized details: will be explained in the lecture, but usually not as an explicit slide; “green” slides will only appear in the handout versions

Red Headers:

- advanced topics or outlook: will not be part of the exam topics

Blue Headers:

- background information or fundamental concepts that are probably already known, but are important throughout the lecture

Algorithms in Scientific Computing

Scientific Computing

similar: *Computational Science and Engineering, Wissenschaftliches Rechnen, Simulation-based Science & Engineering, . . .*

Attempt of a definition:

Scientific Computing is . . .

- (numerical) simulation of problems from science or engineering using High Performance Computing (Bungartz, TUM)
- the interdisciplinary conjunction of mathematical and computer science methods as well as different applications of the natural sciences and engineering disciplines, e.g. (TU Darmstadt)
- is concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems (Wikipedia, 2015)
- an *interdisciplinary discipline*
- the focus at our chair SCCS (Informatics V)

Algorithms in Scientific Computing?

Central Question: What do I “get” from this lecture?

- …in particular in the field of Scientific Computing?
- …in general in the field of Informatics?

$\Rightarrow$ What could/should/do I want to learn in

- … Informatics?
- … Computing?

Cross-topical aspects: What central …

- problems
- techniques, methods
- analytical questions

… of Informatics/Computing/…play a (major) role?

What are our tools?

Representation of Information

Claim:

Informatics is the science (or art) of storing information such that it can be used (processed) efficiently.

Examples for information and storage technique:

- tables (data bases of all kind)
- trees, graphs (path searching, ...)
- objects
- multi-dimensional arrays (raster data, etc.)

Our topic:

How do we store *continuous* data (mathematical functions)?

For Comparison: Representation of Scalars

A brief history of the representation of numbers:

- "tally marks": $|, ||, |||, ||||$
  (still successfully used to count drinks in bars & restaurants)
- number symbols such as $\mathrm{I}, \mathrm{V}, \mathrm{X}, \mathrm{MMIV}$:
  compact but tedious for computing
- positional notation (decimal numbers, binary system, etc.):
  ease of arithmetics up to machine computing

Crucial ideas:

- Hierarchy (different "value" of digits depending on their position)
- Structure (concept of 0 as a placeholder!)

Representation of Mathematical Functions

Possibilities of representation (historical):

- *analytical functions*: $f(x) = e^x \sin(x)$
- *tabulated values*
  (historic example: logarithm tables; modern: rastered data/sampling)
- *interpolation* (also piecewise):
  (polygonal chain/curve, polynomial interpolation,
  spline interpolation, trigonometrical interpolation, ...)

Goals: access and use information efficiently!

- more compact storage
- identification of certain properties (information)
- more efficient algorithms for processing/computations

Our Key Formula: ("coefficients and basis functions")

$$f(x) \approx \sum c_i \phi_i(x)$$

Multi-Dimensional Data

Examples for multi-dimensional data structures:

- Matrices, tensors
- Image data (images, tomography, movies, , ...)
- Discretization based on grids (discretization of physical models / partial differential equations)
- Coordinates of any kind (often going along with graphs)
- Tables (relational databases)
- In financial mathematics: baskets of stocks/options/...

Multi-Dimensional Data

Core topic #1: linearization/sequentialization

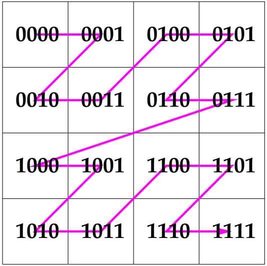
- Storage of data structures in memory
- Data processing (traversal)

Demands on linearization ("efficiency"):

- Maintain neighborhood ⇒ locality of data, "clustering"
- Simple, fast computation of indices
- "Continuity", regularity
- Symmetry w.r.t. single dimensions

Space-Filling Curves and Octrees:

- functions on structured adaptive grids
- ordered by space-filling curves

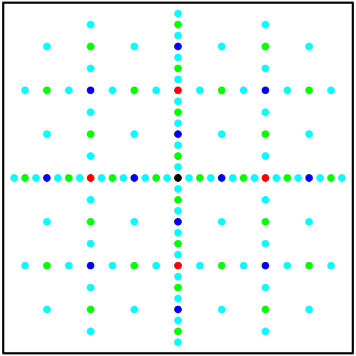


Multi-Dimensional Data

Core topic #2: "curse of dimensionality"

- arrays in 1D, 2D, 3D require n, n^2, n^3 unknowns (in general: n^d)
- how about problems, where $d > 3$ (or $d > 10$, or $d > 100$)

"Sparse Grids:"



Recursive Algorithms and Hierarchical Data Structures

"Traditional" style of algorithms in scientific computing:

- FORTRAN programs; procedural/iterative programming
- strongly based on loops and arrays

Nowadays:

- *Recursive* and *hierarchical*:
  w.r.t. algorithms (partitioning of problem) and data structures (trees, object orientation)
- *Adaptive*: invest effort, where most benefit can be achieved
- *"Optimal Complexity"*: high approximation order, etc.
- *Distributed:* Computing on parallel and distributed systems
- *Hardware-oriented:* $\rightarrow$ High Performance Computing

$\Rightarrow$ generally applicable concepts and ideas

What are our problems?

Interpolation/Representation of Data

Given:

- a set of values: f_i, f_{ij}, $\ldots$ (1D, 2D, $\ldots$)
- at grid/sampling points x_i, x_{ij}, $\ldots$

Wanted:

- function $f(x) = \sum a_i \phi_i(x)$ such that $f(x_i) = f_i$ (similar in 2D)
- we need to compute the coefficients a_i

Important aspects:

- solution depends on clever choice of the basis functions ϕ_i (recall: Lagrange/Newton interpolation)
- can we skip coefficients/basis functions a-priori/a-posteriori? $\rightsquigarrow$ adaptivity, compression, etc.

Recall: Interpolation Problem

For given values b_i and points x_i ($i = 1, \ldots, n$) find a function $f(x)$, such that:

$$f(x_i) = b_i \text{ for all } i = 1, \ldots, n \qquad \text{where } f(x_i) = \sum_{j=1}^{n} a_j g_j(x_i)$$

The functions $g_j(x)$ ($j = 1, \ldots, n$) are suitably selected (polynomials, e.g.).

With $G_{ij} := g_j(x_i)$, we can write the problem as a system of linear equations:

$$\sum_{j=1}^{n} a_j g_j(x_i) = b_i \text{ for all } i = 1, \ldots, n$$

$$\Leftrightarrow \sum_{j=1}^{n} G_{ij} a_j = b_i \text{ for all } i = 1, \ldots, n$$

Corresponds to solving a linear system of equations: $Ga = b$.

Approximation of Data

Given:

- a set of values: f_i, f_{ij}, $\ldots$ (1D, 2D, $\ldots$)
- at points x_i, x_{ij}, $\ldots$

Wanted:

- function $f(x) = \sum a_i \phi_i(x)$ such that $\sum \big(f_i - f(x_i)\big)^2$ is minimal
- we need to compute the coefficients a_i

Important aspects:

- typically more data f_i than coefficients (overdetermined)
- solution depends on clever choice of the basis functions ϕ_i
  $\rightarrow$ how many functions, and how should they look like?
- related to classification and learning ("big data")

Recall: Approximation Problem

For given values b_i and points x_i ($i = 1, \ldots, m$) find a function $f(x)$, such that:

$$f(x_i) \approx b_i \text{ for all } i = 1, \ldots, m \qquad \text{where } f(x_i) = \sum_{j=1}^{n} a_j g_j(x_i) \quad \text{with } m > n$$

We find the best approximation by minimising the quadratic error:

$$\sum_{i=1}^{m} \big(f(x_i) - b_i\big)^2 \stackrel{!}{=} \min \quad \Leftrightarrow \quad \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_j g_j(x_i) - b_i \right)^2 \stackrel{!}{=} \min$$

We set all derivatives w.r.t. our variables a_k to 0 (again with $G_{ij} := g_j(x_i)$) **for all $k = 1, \ldots, n$:**

$$\frac{\partial}{\partial a_k} \left(\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_j G_{ij} - b_i \right)^2 \right) = \sum_{i=1}^{m} \frac{\partial}{\partial a_k} \left(\sum_{j=1}^{n} a_j G_{ij} - b_i \right)^2 \stackrel{!}{=} 0$$

$$\Leftrightarrow \quad \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} a_j G_{ij} - b_i \right) G_{ik} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{m} G_{ik} \sum_{j=1}^{n} a_j G_{ij} = \sum_{i=1}^{m} G_{ik} b_i$$

Corresponds to solving a linear system of equations: $G^T G a = G^T b$.

Approximation with Regularization

Given:

- a set of values: f_i, f_{ij}, ... (1D, 2D, ...)
- at points x_i, x_{ij}, ...

Wanted:

- function $f(x) = \sum a_i \phi_i(x)$ such that $\sum \big(f_i - f(x_i)\big)^2 + \gamma \|Lf\|$ is minimal with $Lf = f'$, $Lf = f''$, or similar
- compute coefficients a_i (depend on parameter γ)

Important aspects:

- more or less data f_i available than required (over- or underdetermined)
- frequent approach to predict value $f(x)$
  $\rightsquigarrow$ related to classification and learning ("big data")
- solution depends on clever choice of the basis functions ϕ_i
  $\rightarrow$ how many functions, and how should they look like?

Find/Approximate a Function

Problem:

- approximate a function that cannot be represented exactly: $f(x) \approx b(x)$
- using a representation $f(x) = \sum a_i \phi_i(x)$

Leads to Question of Approximation between Functions:

- define a norm on functions: $\|f\|^2 = \int \left(f(x)\right)^2 dx$
- try to minimize $\|b(x) - \sum a_i \phi_i(x)\|^2$

Orthogonality Argument:

- norm derived from dot product on functions: $\langle f, g\rangle = \int f(x)g(x)\,dx$
- then demand that $b(x) - \sum a_i \phi_i(x)$ is orthogonal to all other functions:

$$\int v(x) \left(b(x) - \sum a_i \phi_i(x)\right) \mathrm{dx} = 0 \qquad \text{for all } v.$$

Find/Approximate a Function (2)

More Interesting Setup:

- solve a partial differential equation, e.g.: $\frac{\partial^2}{\partial x^2} f(x) = b(x)$
  (with initial and boundary conditions, as required)

Wanted:

- function $f(x) = \sum a_i \phi_i(x)$ that “solves” the equation
- depending on the choice of $\phi_i(x)$, only an approximate solution might be possible

Leads to Finite Element methods:

- again, demand that

$$\int v(x) \left(b(x) - \frac{\partial^2}{\partial x^2} f(x) \right) \mathrm{dx} = 0 \qquad \text{for all } v.$$

- solution depends on clever choice of the basis functions ϕ_i and “test functions” v; leads to system of equations for a_i

What are our algorithms?

Schedule

Fast Fourier Transform:

- discrete Fourier transform as 2D, 3D interpolation
- FFT as divide-and-conquer algorithm
- transform for data compression (images, audio and video data)

Hierarchical basis and sparse grids

- adaptive integration and Archimedes' quadrature
- hierarchical basis functions
- the curse of dimensionality $\rightarrow$ sparse grids
- wavelets

Space trees and space-filling curves

- sequential data structures and traversal of octrees
- definition and construction of space-filling curves
- adaptivity vs. parallelisation and partitioning