Towards Complexity Classification of Partial Differential Equations

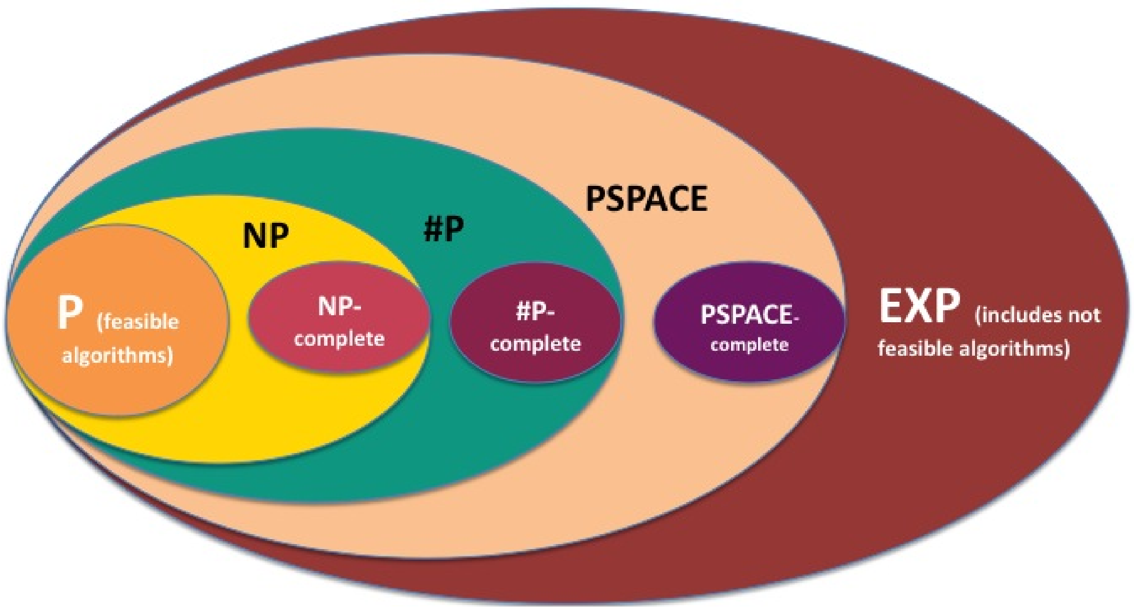
Svetlana Selivanova

KAIST

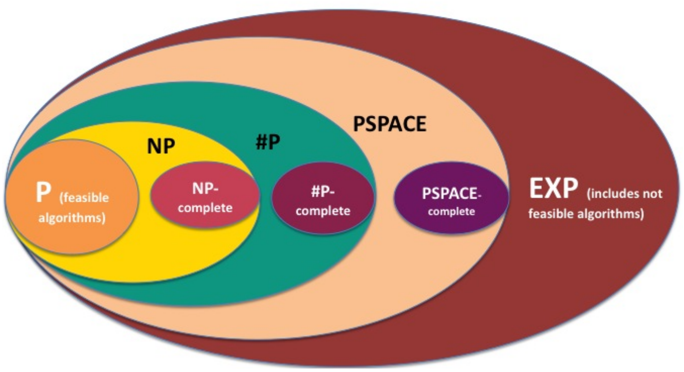
The first Korea Logic Day 2021, January 14

KAIST

Complexity hierarchy

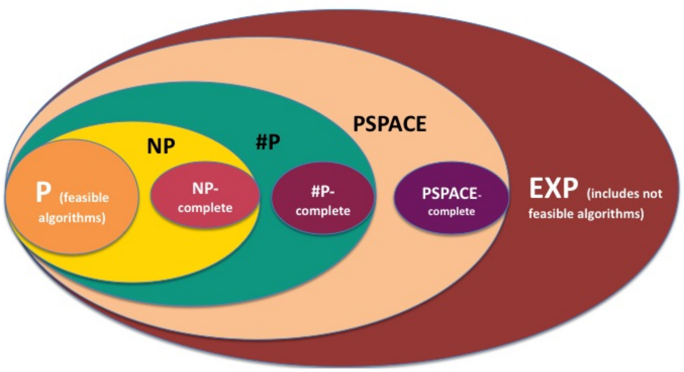


“Real” / “Continuous” Computability and Complexity Theory



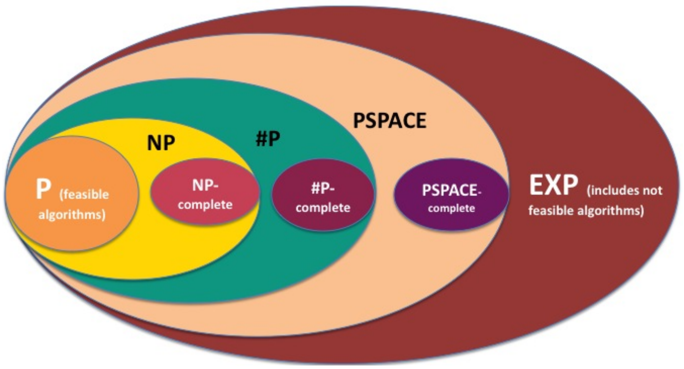
- Ker-I Ko. Complexity Theory of Real Functions, 1991.
- Klaus Weihrauch. Computable Analysis, 2000.

Complexity classification of "continuous" problems



- max f: **NP**-complete
- $\int_0^x f(t)dt$: $\sharp$**P**-complete
- $\int_0^1 f(t)dt$: $\sharp\mathbf{P}_1$-complete
- Solutions of ODEs $[\frac{du}{dt} = f(t,u), \quad u(0) = u_0]$: **PSPACE**-complete in general

Complexity classification of "continuous" problems



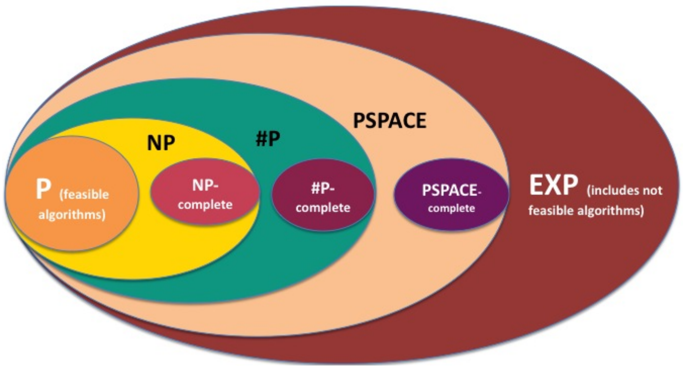
- PDEs: (elliptic) Dirichlet problem for the Laplace equation

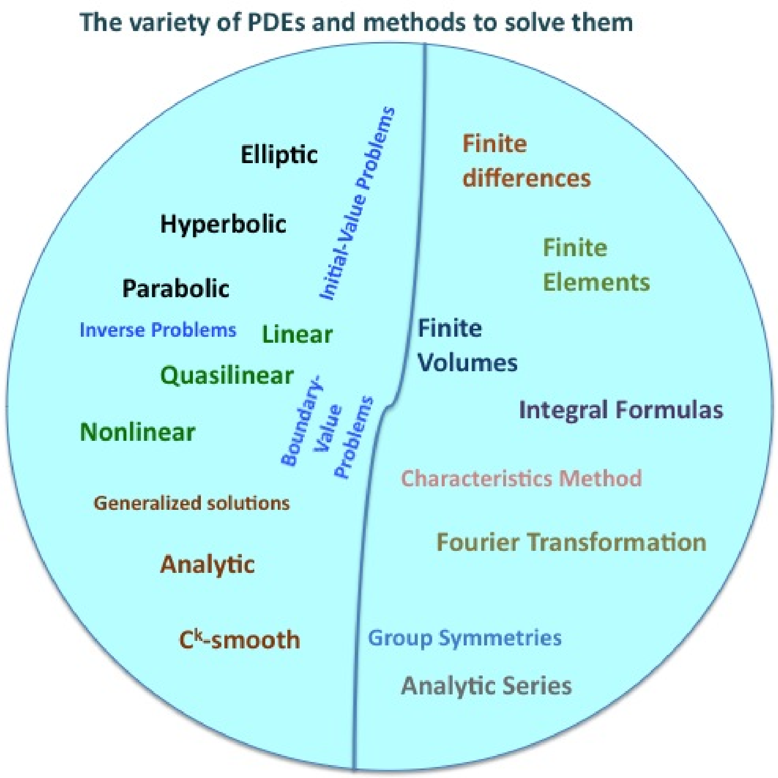
$$\Delta u = f \text{ on } B_d(0,1);$$

$$u = 0 \text{ on } \partial B_d(0,1)]$$

in $\sharp\mathbf{P}$, $\sharp\mathbf{P}_1$-hard [Kawamura, Steinberg, Ziegler 2017]

Complexity classification of "continuous" problems





- PDEs: (elliptic) Poisson problem in $\sharp$P, $\sharp P_1$-hard [Kawamura, Steinberg, Ziegler 2017]

- <u>How about other PDEs?</u>

Outline of the talk

- Motivation
- Real Complexity Classes
- Current progress for PDEs
  - ▷ Finite Approximation method and Exponential Linear Algebra
  - ▷ Analytic series and PTIME computability
  - ▷ A hardness result
- Perspectives and future work

MOTIVATION

- ▷ Exact Real Computation (ERC) and Partial Differential Equations (PDEs)
- ▷ Very brief reminder about PDEs
- ▷ How does Classifying PDEs by their Algorithmic Complexity help?

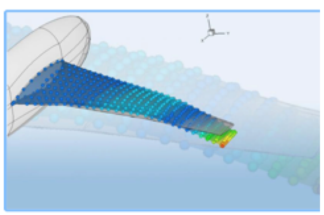
◇ Motivation for **Exact Real Computation**

▷ Computing solutions with **guaranteed prescribed precision** is important

– For safety critical applications: accumulation of errors can lead to disasters!

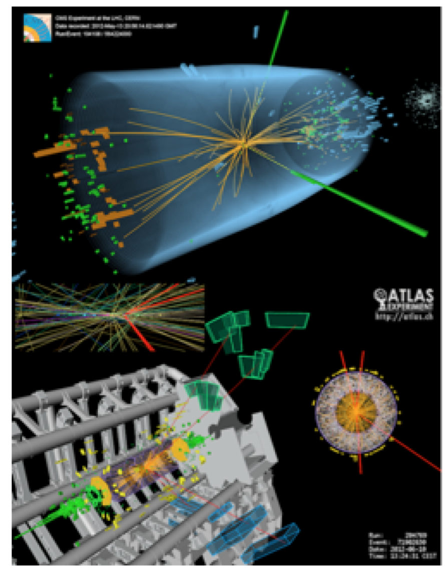
– For small scale applications like particle physics: high precision needed!

Narrows Bridge (Tacoma, Wash.) broke down in 1940. Im credit: Public domain image, from the Seattle Post-Intelligencer 1940



Under flutter effects, aircraft wings can bend or break off, leading to numerous plane crashes. Image credit: Netherlands Aerospace Center / NRL

Space shuttle Challenger exploded in 1986. Image credit: Michael Hindes – West Springfield, MA



Candidate Higgs boson events from collisions between protons. Image credit: CERN Document Server

◇ Motivation for Exact Real Computation

▷ It would be great for numerical package users to have the result **without thinking about error bounds while still having it accurate**

▷ Most software packages

- Are restricted by **floating points**: at most 53 digits of output with double precision.
- Contain complicated sequences of computation hidden from the users: it is hard to control the error propagation.

▷ For problems with big modulus of continuity or discontinuities, **there can be a wrong result!** Users need to be careful!

Exact Real Computation approach

- Exact Real Computation has $\mathbb{R}$ (real numbers) as **exact** data type
- Computations **approximate** output to **guaranteed** precision 2^{-n} given by the user (i.e., computes **any** n digits versus fixed 53 in double precision)
- Computing a function $t \mapsto \mathbf{u}(t)$:

$$|t - \frac{t_m}{2^m}| < 2^{-m} \rightarrow ||\mathbf{u}(t) - \frac{u_n}{2^n}|| < 2^{-\mathbf{n}}$$

  t_m, u_n integers, $m = m(n)$ modulus of continuity of $\mathbf{u}$

- Exact Real Computation packages: iRRAM, ARIADNE, Aern
- We aim to
  - Develop the necessary theory (complexity classification!)
  - Create Exact Real Computation solvers for PDEs to be used for applications

◇ Partial Differential Equations (PDEs)

- PDEs describe various processes evolving in several (e.g. time and space) directions
- Important phenomena from nature to engineering are modeled by PDEs
- Current vast methodologies for solving them still remain *highly limited* without an overall theoretical framework.
- Numerical methods and packages *suffer from* floating point errors and computational instabilities.

General form of a PDE:

$$\begin{cases} Lu(x) = f(x), x \in \Omega \subset \mathbb{R}^k \\ \mathcal{L}u(x)|_\Gamma = \varphi(x\,|_\Gamma), \Gamma \subseteq \partial\Omega. \end{cases}$$

Differential operator:

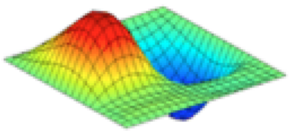
$$L\mathbf{u} = \sum_{|\alpha|=k} a_\alpha(D^{k-1}\mathbf{u}, \dots, \mathbf{u}, y) D^\alpha \mathbf{u} +$$

$$+ a_0(D^{k-1}\mathbf{u}, \dots, \mathbf{u}, x), \quad D^\alpha \mathbf{u} = \frac{\partial^{\alpha_1} \dots \partial^{\alpha_k}}{\partial x_1^{\alpha_1} \dots \partial x_k^{\alpha_k}}$$

▷ $\frac{\partial^2}{\partial t^2}u - \Delta u = 0$ *wave equation*

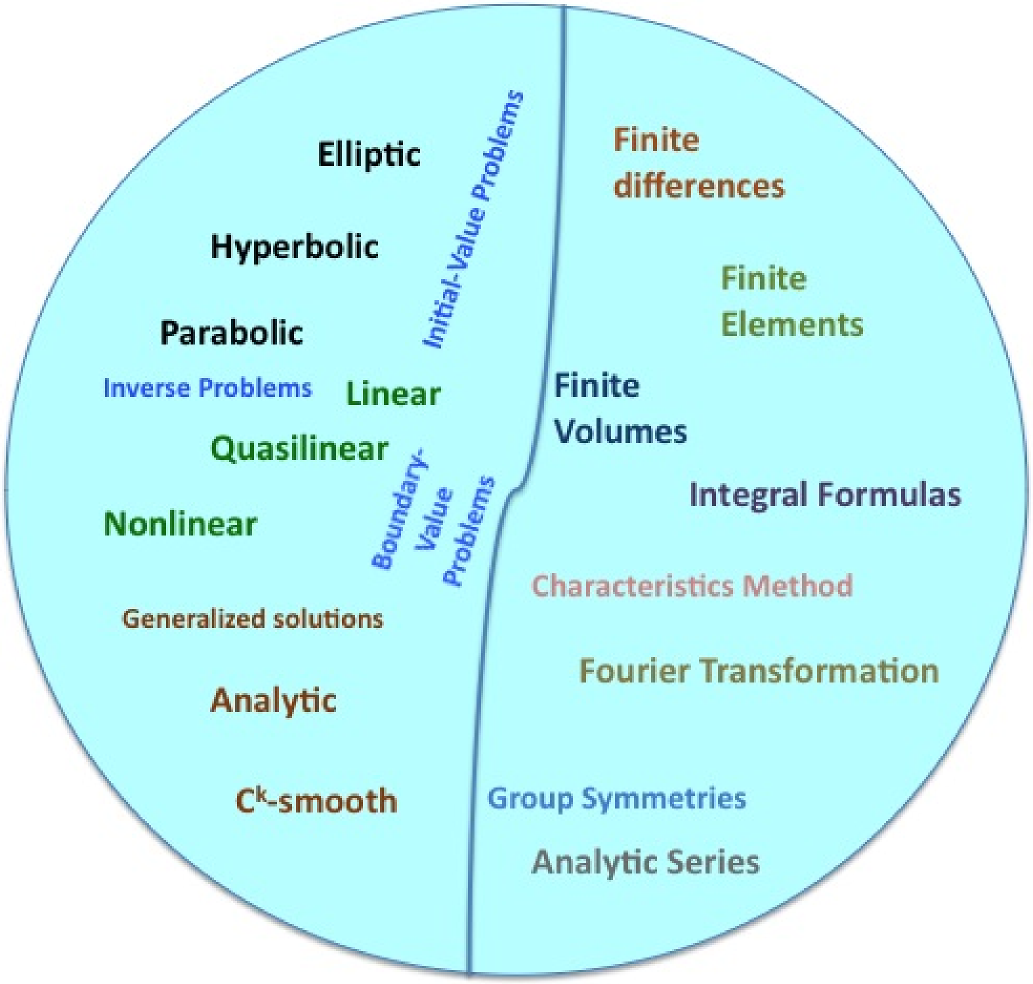
▷ $\frac{\partial}{\partial t}u - \Delta u = 0$ *heat equation*

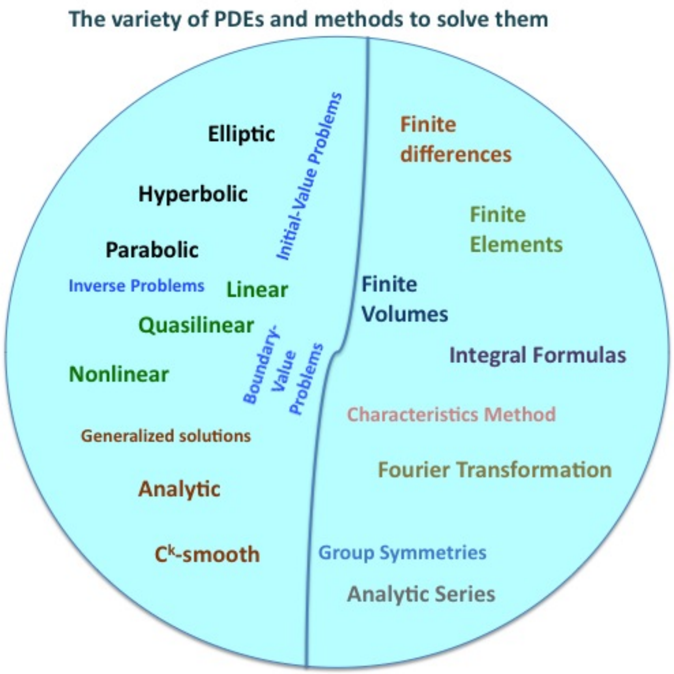
▷ *acoustics, elasticity, electromagnetism, fluid dynamics, neuroscience, quantum physics etc.*



Solution of the 2D Wave equation (Image credit: Wikipedia)

The variety of PDEs and methods to solve them





Goals:

- Develop a uniform framework for solving (important classes of) PDEs with guaranteed arbitrary precision given by the user, which is crucial for safety critical applications

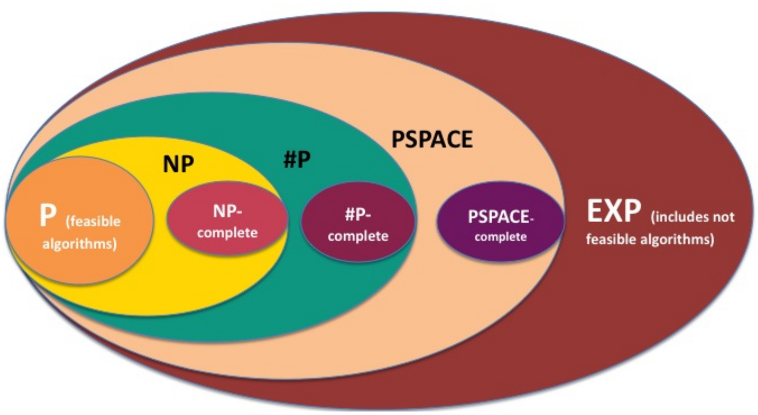
$$||u - u^{(n)}|| < 2^{-\mathbf{n}}$$

- Classify PDEs by their algorithmic complexity.

- Based on this classification we develop and implement optimal and reliable algorithms.

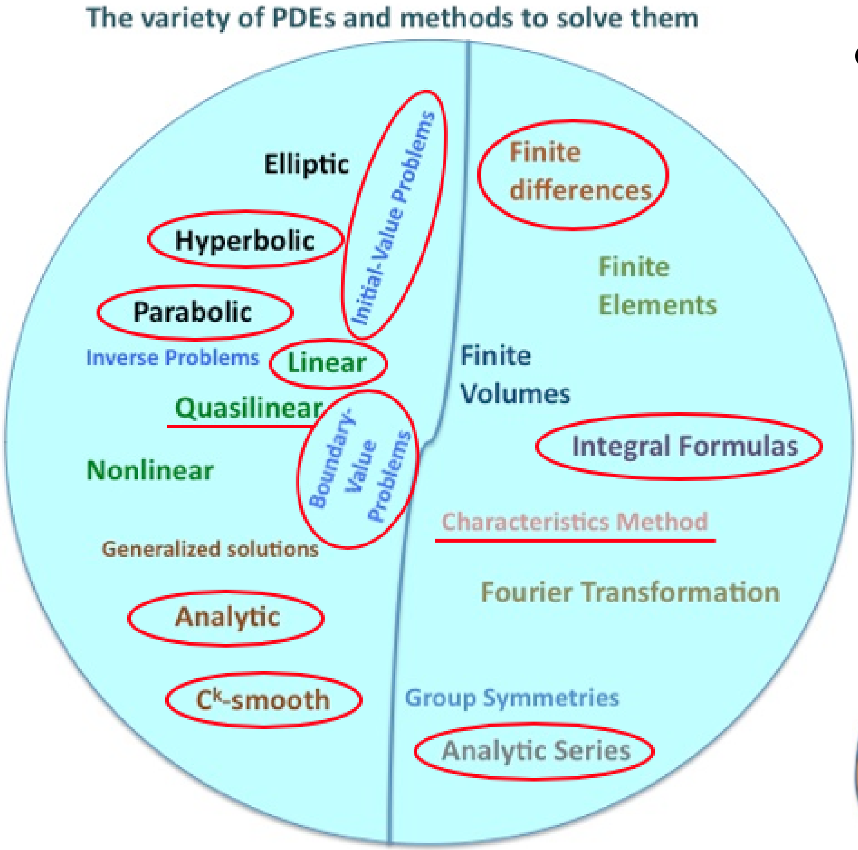
◇Classifying PDEs by their Algorithmic Complexity

- What amount of resources (time, memory cells) is needed to solve a particular problem?
- Examples: n^k (*feasible*), $\log \mathrm{n}$ (*runs fast*), 2^{n} (*can run a million years*)
- Which algorithm is **optimal**?
- To investigate these problems, we use the discrete complexity hierarchy.
  - It relates to the "**P=NP?**" Millenium problem.



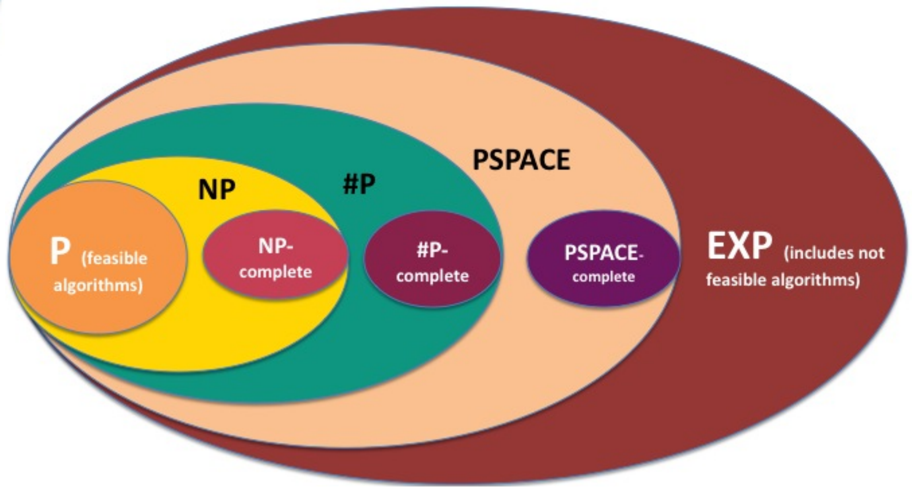
- **P**: algorithms running in polynomial time (feasible) n^k
- **PSPACE**: algorithms using polynomial amount of memory cells
- **EXP**: algorithms running in exponential time (not feasible) 2^{n}

Classifying PDEs by their Algorithmic Complexity



Our strategy:

- Investigate complexity of Exact Real Computation adaptations of various methods
- Then try to optimally match the PDE with a complexity class (with respect to the parameter n for precision 2^{-n})



REAL COMPLEXITY

- ▷ Brief reminder about discrete complexity classes
- ▷ Main real complexity classes
- ▷ Examples of what type of results we are proving / interested to prove

Discrete complexity classes

- $P = \{L \subseteq \mathbb{N} \mid$ decidable in polynomial time$\}$

- $FP = \{f : \mathbb{N} \to \mathbb{N} \mid$ computable by a deterministic Turing machine within time polynomial in the binary length of the input$\}$

- $NP = \{L \subseteq \mathbb{N} \mid$ verifiable in polynomial time$\}$
  (or: accepted by a non-deterministic Turing machine within polynomial time)

- $\sharp P = \{f : \mathbb{N} \to \mathbb{N} \mid f$ counts the number of accepting computations of a non-deterministic polynomial-time Turing machine$\}$

- $\sharp P_1 = \{f : 2^{\mathbb{N}} \to \mathbb{N} \mid f$ counts the number of accepting computations of a non-deterministic polynomial-time Turing machine$\}$

- $PSPACE = \{L \subseteq \mathbb{N} \mid$ decidable in polynomial space$\}$

- $EXP = \{L \subseteq \mathbb{N} \mid$ decidable in exponential time$\}$

Real complexity classes

◇ For real numbers

Def. Computing $r \in \mathbb{R}$ in time $t : \mathbb{N} \to \mathbb{N}$ means to output $a_n \in \mathbb{Z}$ (in binary) s.th.

$$|r - a_n/2^n| \leq 1/2^n,$$

in $\leq t(n)$ steps.

- **PTIME** if t(n)=poly(n)
- **EXP** if t(n)=exp(n)
- **PSPACE**: if the amount of memory s(n) is bounded polynomially in n

Real complexity classes

$\diamond$ For real functions

Def. Computing $f :\subseteq \mathbb{R} \to \mathbb{R}$ in time $t : \mathbb{N} \to \mathbb{N}$ means, on input $a_m \in \mathbb{Z}$ s.th.

$$|x - a_m/2^m| \leq 1/2^m,$$

to output $b_n \in$ s.th.

$$|f(x) - b_n/2^n| \leq 1/2^n,$$

in $\leq t(n)$ steps.

- **PTIME** if t(n)=poly(n)
- **EXP** if t(n)=exp(n)
- **PSPACE**: if the amount of memory s(n) is bounded polynomially in n

Examples

◇ The following are equivalent:

- FP=$\sharp P$

- For every polynomial time computable $h : [0,1] \to \mathbb{R}$, the function
$$x \to \int_0^x h(t)dt$$
is again polynomial time computable.

(In other words, indefinite Riemann integration is “$\sharp P$-complete”)

Examples

$\diamond$ The following are equivalent:

- $FP_1 = \sharp P_1$

- For every polynomial time computable $h : [0,1] \to \mathbb{R}$, the real number $\int_0^1 h(t)dt$ is again polynomial time computable.

(In other words, definite Riemann integration is "$\sharp P_1$-complete")

Examples

$\diamond$ PDEs: (elliptic) Dirichlet problem for the Laplace equation

$$\Delta u = f \text{ on } B_d(0,1);$$

$$u = 0 \text{ on } \partial B_d(0,1)]$$

(1) "in $\sharp$**P**", (2) "$\sharp$**P**$_1$-hard" [Kawamura, Steinberg, Ziegler 2017].

(here $\Delta u = \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2} u$)

CURRENT ACHIEVEMENTS OF COMPLEXITY OF PDEs

- $\triangleright$ General overview
- $\triangleright$ Finite Approximation method and Exponential Linear Algebra
- $\triangleright$ Analytic series and PTIME computability
- $\triangleright$ A hardness result: heat equation

$\diamond$ Hardness result: Heat equation

Theorem [Koswara, Pogudin, S., Ziegler'20]

$$\frac{\partial}{\partial t}u = \Delta u \text{ on } [0,1]^2;$$

$$u(0) = u(1), \quad u_x(0) = u_x(1)$$

(1) "in $\sharp$**P**", (2) "$\sharp\mathbf{P}_1$-hard" .

(here $\Delta u = \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2} u$)

Theorem [Koswara, Pogudin, S., Ziegler'20]

$$\frac{\partial}{\partial t} u = \Delta u \text{ on } [0,1]^2;$$

$$u(0) = u(1), \quad u_x(0) = u_x(1)$$

(1) "in $\sharp$**P**", (2) "$\sharp$**P**$_1$-hard" .

Proof sketch:

- in : using finite difference approach(see below)
- "$\sharp$**P**$_1$-hard": using smoothness properties of the solution operator and the following two facts [Ker I Ko'91]
  - ▷ There is a PTIME-computable h: $[0,1] \to [0,1]$ s. th. $\int_0^1 h(t)dy$ is not computable in PTIME unless $FP_1 = \sharp P_1$
  - ▷ For every PTIME-computable **analytic** function $g : [0,1] \to \mathbb{R}$, $\int_0^1 g(t)dt$is computable in $PTIME_1$.

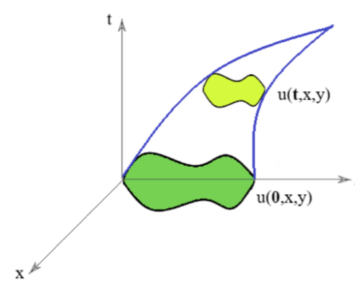
Our current findings on Complexity of PDEs

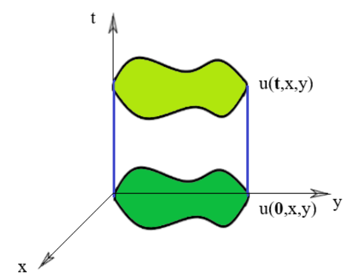
We made significant progress on classifying evolutionary systems of PDEs.

Joint work with: I.Koswara, D. Lim, M.Ziegler (KAIST) G.Pogudin (École Politecnique), A.Kawamura (Kyoto University).

Type of PDE/ Functional class	**Linear** Evolutionary Systems (including Hyperbolic and Parabolic), **our results:**	Linear Elliptic: Poisson Problem for Laplace equation, studied in [Kawamura Steinberg, Ziegler'17]	**Quasilinear** Evolutionary Systems, **our results:**
Analytic	P (method of analytic series)	P	#P
C^k-smooth, $k \geq 1$	in general PSPACE for periodic #P (method of finite differences); Heat equation is #P-complete	#P-complete	EXP
W_p^k(generalized, Sobolev spaces)	Currently working on developing the framework of real complexity for this case.		

$$\begin{cases} \frac{\partial}{\partial t}\vec{u} = \sum\limits_{i=1}^{m} B_i(\vec{x},\vec{u})\frac{\partial}{\partial x_i}\vec{u}, \vec{x} \in \Omega, \\ \vec{u}(0,\vec{x}) = \varphi(\vec{x}), \end{cases}$$





Complexity of linear PDEs: Difference Schemes

$$\frac{\partial}{\partial t}\vec{u} = \sum_{i=1}^{m} B_i(x)\frac{\partial}{\partial x_i}\vec{u}, \quad \vec{u}(0,x) = \varphi(x), \quad (\mathcal{L}\vec{u}\mid_{\partial\Omega}= 0).$$

Theorem. (Koswara, Pogudin, S., Ziegler) Suppose the given IVP and BVP be well posed and admit a converging finite difference approximation (with certain natural properties).

$B_i(x)$, $\varphi(x)$ fixed PTIME computable functions. Then:

1. The solution u is in **PSPACE**

2. For the periodic boundary condition u is **"in $\sharp P$"**.

Complexity of linear PDEs: Difference Schemes

Theorem. (Koswara, Pogudin, S., Ziegler) Suppose the given IVP and BVP be well posed and admit a converging finite difference approximation (with certain natural properties).

$B_i(x)$, $\varphi(x)$ fixed PTIME computable functions. Then:

1. The solution u is in **PSPACE**

2. For the periodic boundary condition u is **"in $\sharp P$"**.

Examples to which this theorem applies:

1. Heat equation $\frac{\partial}{\partial t}u = \Delta u$

2. Wave equation $\frac{\partial^2}{\partial t^2}u = \Delta u$

3. Symmetric hyperbolic systems (including acoustics, elasticity, Maxwell equations)

Complexity of linear PDEs: Difference Schemes

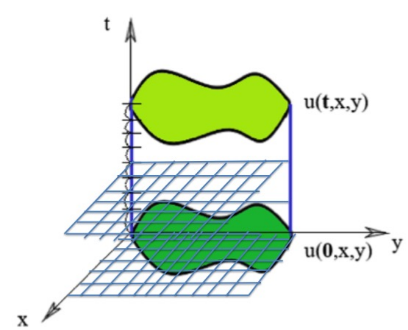
Some proof ideas

$$\frac{\partial}{\partial t}\vec{u} = \sum_{i=1}^{m} B_i(x)\frac{\partial}{\partial x_i}\vec{u}, \quad \vec{u}(0,x) = \varphi(x), \quad (\mathcal{L}\vec{u}\,|_{\partial\Omega} = 0).$$

Discretize with uniform grid steps τ, $h = 2^{-O(2^n)}$

$$\mathbf{u}^{(n)} = \mathbf{A_n}^{2^n}\varphi^{(n)}$$

Huge matrix powering!



Dimension of A_n is $O(2^n)$; powers are **uniformly bounded**

Complexity of linear PDEs: Difference Schemes

Lemmas

- 2^n vector $\times$ 2^n vector: $\#P$-complete
- 2^n matrix to the power 2^n: $PSPACE$-complete
- !!! for the special case of periodic PDEs, 2^n matrix to the power 2^n is in $\#P$ (for 2-band matrices also in $PTIME$)

- Structured matrices ($\mathcal{C}_{k,j,l}$ are circulant)

$$A_k \ := \ \sum_{j=1}^{J} Q_j \otimes \mathcal{C}_{k,j,1} \otimes \mathcal{C}_{k,j,2} \otimes \cdots \otimes \mathcal{C}_{k,j,L}$$

$$A_n = \left[\begin{array}{cccc|cccc|cccc|cccc}
\mu & \lambda & & \lambda & \lambda & & & & & & & & \lambda & & & \\
\lambda & \mu & \lambda & & & \lambda & & & & & & & & \lambda & & \\
 & & \cdots & & & & \cdots & & & & \cdots & & & & \cdots & \\
\lambda & & \lambda & \mu & & & & \lambda & & & & & & & & \lambda \\
\hline
\lambda & & & & \mu & \lambda & & \lambda & \lambda & & & & & & & \\
 & \lambda & & & \lambda & \mu & \lambda & & & \lambda & & & & & & \\
 & & \cdots & & & & \cdots & & & & \cdots & & & & \cdots & \\
 & & & \lambda & \lambda & & \lambda & \mu & & & & \lambda & & & & \\
\hline
 & & & & & & & & & & & & & & & \\
 & & & & & & \cdots & & & & \cdots & & & & \cdots & \\
 & & & & & & & & & & & & & & & \\
\hline
\lambda & & & & & & & & \lambda & & & & \mu & \lambda & & \lambda \\
 & \lambda & & & & & & & & \lambda & & & \lambda & \mu & \lambda & \\
 & & \cdots & & & & \cdots & & & & \cdots & & & & \cdots & \\
 & & & \lambda & & & & & & & & \lambda & \lambda & & \lambda & \mu
\end{array}\right.$$

• Kronecker products of circulant matrices correspond to polynomials

$$A_{h(n)}^{(2)} = \mu I + \lambda \begin{bmatrix} J_1 & & \\ & \cdots & \\ & & J_1 \end{bmatrix} + \lambda \begin{bmatrix} J_2 & & \\ & \cdots & \\ & & J_2 \end{bmatrix} + \lambda \begin{bmatrix} 0 & I & \\ & \cdots & \\ & & I \\ I & & 0 \end{bmatrix} + \lambda \begin{bmatrix} 0 & & I \\ I & 0 & \\ & \cdots & \\ & I & 0 \end{bmatrix},$$

where I is the identity matrix of a corresponding dimension;

$$J_1 \begin{bmatrix} 0 & 1 & \\ & \cdots & \\ & & 1 \\ 1 & & 0 \end{bmatrix}; \quad J_2 = \begin{bmatrix} 0 & & 1 \\ 1 & 0 & \\ & \cdots & \\ & 1 & 0 \end{bmatrix}.$$

We can write this in tensor form

$$A_{h(n)}^{(2)} = \mu(I \otimes I) + \lambda(J_1 \otimes I) + \lambda(J_2 \otimes I) + \lambda(I \otimes J_1) + \lambda(I \otimes J_2)$$

Or in polynomial form

$$p^{(2)}(X, Y) = \mu + \lambda X + \lambda X^{-1} + \lambda Y + \lambda Y^{-1}$$

Note that already for the quadratic polynomial $P(X) = (1+X+X^2)/3$, evaluation of the 'explicit' formula

$$\left(\tfrac{1}{3}+\tfrac{1}{3}X+\tfrac{1}{3}X^2\right)^K[X^J] \;=\; 3^{-K}\cdot \sum_{\substack{0\leq\mu,\nu\leq K\\ \mu+2\nu=M}} \frac{K!}{\mu!\cdot\nu!\cdot(K-\mu-\nu)!} \quad (1)$$

involves terms like $K!$ of value, and the sum with a number of terms, doubly exponential in k: not at all obvious to compute in $\sharp$P.

• For raising polynomials to huge powers Cauchy's integration formula is applicable!

We can express any single desired coefficient of P^M as loop integral over $P^M(z)/z^{M+1}$ for $|z| = 1$ running over the complex unit circle. And due to P having bounded powers, the values of $P^M(z)/z^{M+1}$ are bounded

• Integration is in $\sharp P$!

Complexity of linear PDEs: Analytic series

$$\frac{\partial}{\partial t}\vec{u} = \sum_{i=1}^{m} B_i(x)\frac{\partial}{\partial x_i}\vec{u}, \qquad \vec{u}(0) = \varphi(x).$$

Theorem (Koswara, S., Ziegler) If φ, B_i are analytic, then

$\diamond$ φ, $B_i \in$ PTIME $\Longrightarrow$ $\mathbf{u} \in$ PTIME

$$\vec{u}(t) \; = \; \sum_{k=0}^{\infty} \tfrac{t^k}{k!}\Big(\textstyle\sum_{i=1}^{m} B_i(x)\tfrac{\partial}{\partial x_i}\Big)^k \varphi(x)$$

CONCLUSIONS AND PERSPECTIVE

- ▷ Summary

- ▷ Future work

Our current findings on Complexity of PDEs

- For analytic systems, achieved poly-time algorithms **P** (feasible!)
$$\vec{u}(t) = \sum_{k=0}^{\infty} \frac{t^k}{k!}\left(\sum_{i=1}^{m} B_i(x)\frac{\partial}{\partial x_i}\right)^k \varphi(x)$$

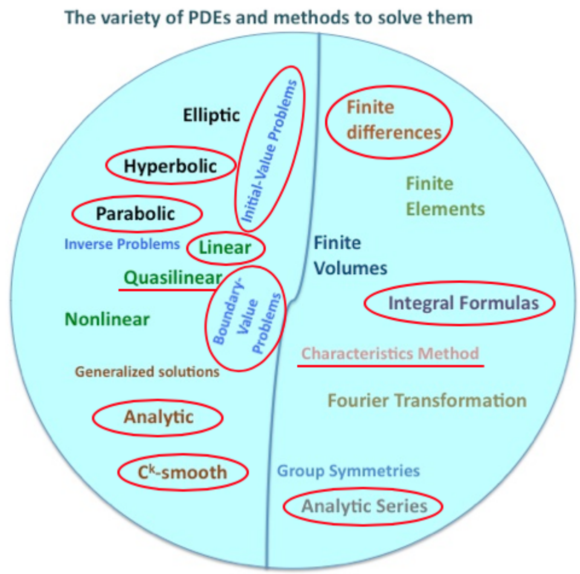
- For finite difference methods, achieved **PSPACE** $\mathbf{u}^{(n)} = \mathbf{A_n}^{2^n}\varphi^{(n)}$

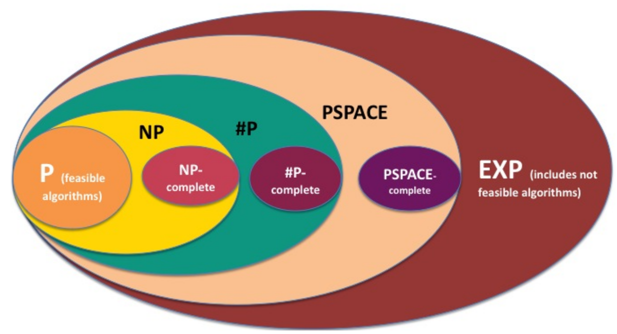
- For particular cases (e.g., periodic boundary conditions) improved to $\sharp$**P** (class between P and PSPACE)

- For heat equation, proved no better algorithm better than $\sharp\mathbf{P}_1$ (in the nonanalytic C^k-smooth case): optimality result

- For quasilinear systems, at best $\sharp$**P** algorithms for analytic case by now

- Development of a more general paradigm to include Sobolev functions is in progress





Our current progress on Exact Real Computation for PDEs

We have made progress in **implementation**.

Joint work with: H.Thies (Kyushi University), F.Steinberg (TU Darmstadt), Jiman Hwang, Martin Ziegler (KAIST), P. Collins (Maastricht University)

- Implementation of the analytic series method for Cauchy-Kovalevskaya type systems.
- Currently tested for acoustics and elasticity systems up to precision 2^{-300}.

$$\begin{cases} \rho_0 \dfrac{\partial u}{\partial t} + \dfrac{\partial p}{\partial x} = 0, \\ \rho_0 \dfrac{\partial v}{\partial t} + \dfrac{\partial p}{\partial y} = 0, \\ \dfrac{\partial p}{\partial t} + \rho_0 c_0^2 \left(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} \right) = 0. \end{cases}$$

$$\begin{cases} \dfrac{1}{2\mu} \dfrac{\partial \sigma_{ij}}{\partial t} - \dfrac{\lambda}{2\mu(3\lambda + 2\mu)} \delta_{ij} \dfrac{\partial(\sigma_{11} + \sigma_{22} + \sigma_{33})}{\partial t} - \dfrac{1}{2}\left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i}\right) = 0, \\ \rho \dfrac{\partial u_i}{\partial t} - \dfrac{\partial \sigma_{ij}}{\partial x_j} = 0, \end{cases}$$

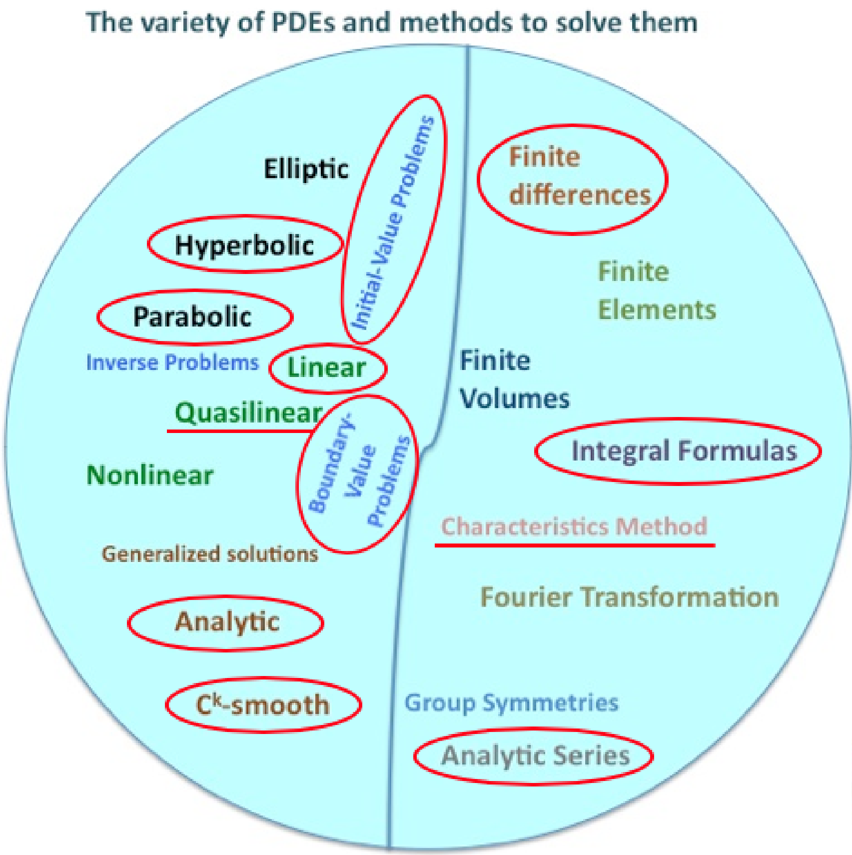
Concluding remarks

- Seems like PDEs are either PTIME or $\sharp P_1$-hard

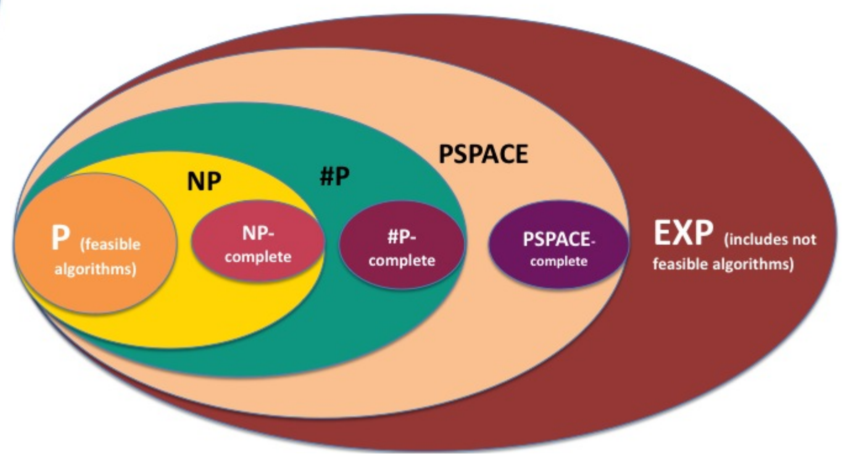
- The case of analytic initial data is much easier, allows PTIME algorithms

- The popular finite difference method is not quite suitable for Exact Real Computation

Future work



- Optimality for broader classes
- Development of PDE solvers
- Sobolev functions
- Nonlinear equations



THANK YOU FOR YOUR ATTENTION!