Continuous Math Taxonomy, ISCA & JSSC

Yipeng Huang

December 7, 2015

Application Taxonomy

- **For organizing what we've explored and planning new applications**
- **Based on a survey of analog books**
  - Karplus, Jackson, Korn, Fifer, Ullmann
- **Based on a survey of CS, applied math, operations research**

Berkeley Dwarfs

- **Continuous mathematics (core solvers)**
  - Sparse linear algebra
  - Dense linear algebra

- **Continuous mathematics (hierarchical methods)**
  - Spectral methods (FFT)
  - Structured grids (multigrid)
  - Unstructured grids (finite elements)
  - N-body (fast multipole method)

- **Continuous + discrete mathematics**

- **Discrete mathematics**

Broad picture of continuous math

- **Trefethen & Bau Numerical Linear Algebra**
  - Epilogue on "heart of numerical analysis":

an analytical point of view, and to do it with lightning speed. For guidance to the future we should study not Gaussian elimination and its beguiling stability properties, but the diabolically fast conjugate gradient iteration—or Greengard and Rokhlin's $O(N)$ multipole algorithm for particle simulations—or the exponential convergence of spectral methods for solving certain PDEs—or the convergence in $O(1)$ iteration achieved by multigrid methods for many kinds of problems—or even Borwein and Borwein's magical AGM iteration for de-

- **Prof. David Keyes (Columbia, KAUST)**
  - "Facing the Four Algorithmic Frontiers of Exascale":
    - **Some optimal hierarchical algorithms**
      - **Fast Fourier Transform (1960's)**
      - **Multigrid (1970's)**
      - **Fast Multipole (1980's)**
      - **Sparse Grids (1990's)**
      - **$\mathcal{H}$ matrices (2000's)**

Berkeley Dwarfs (continued)

- **Continuous mathematics (core solvers)**

- **Continuous mathematics (hierarchical methods)**

- **Continuous + discrete mathematics**
  - Monte Carlo
  - Dynamic programming & graphical models (Viterbi algorithm, hidden Markov models)
  - Backtrack and branch and bound (non-convex optimization, constraint satisfaction problem, travelling salesman problem)

- **Discrete mathematics**
  - Combinational logic, finite state machines, graph traversal

Broad picture of continuous algorithms

- **Soc. Int. App. Math: 20th Century Top 10 Algorithms:**
  - Monte Carlo, Simplex, Conjugate Gradients, direct methods, FFT, FMM, etc.

- **Other sources of inspiration**
  - Numerical Recipes in C
  - Matlab standard routines
  - PETSc scientific computation library
  - …

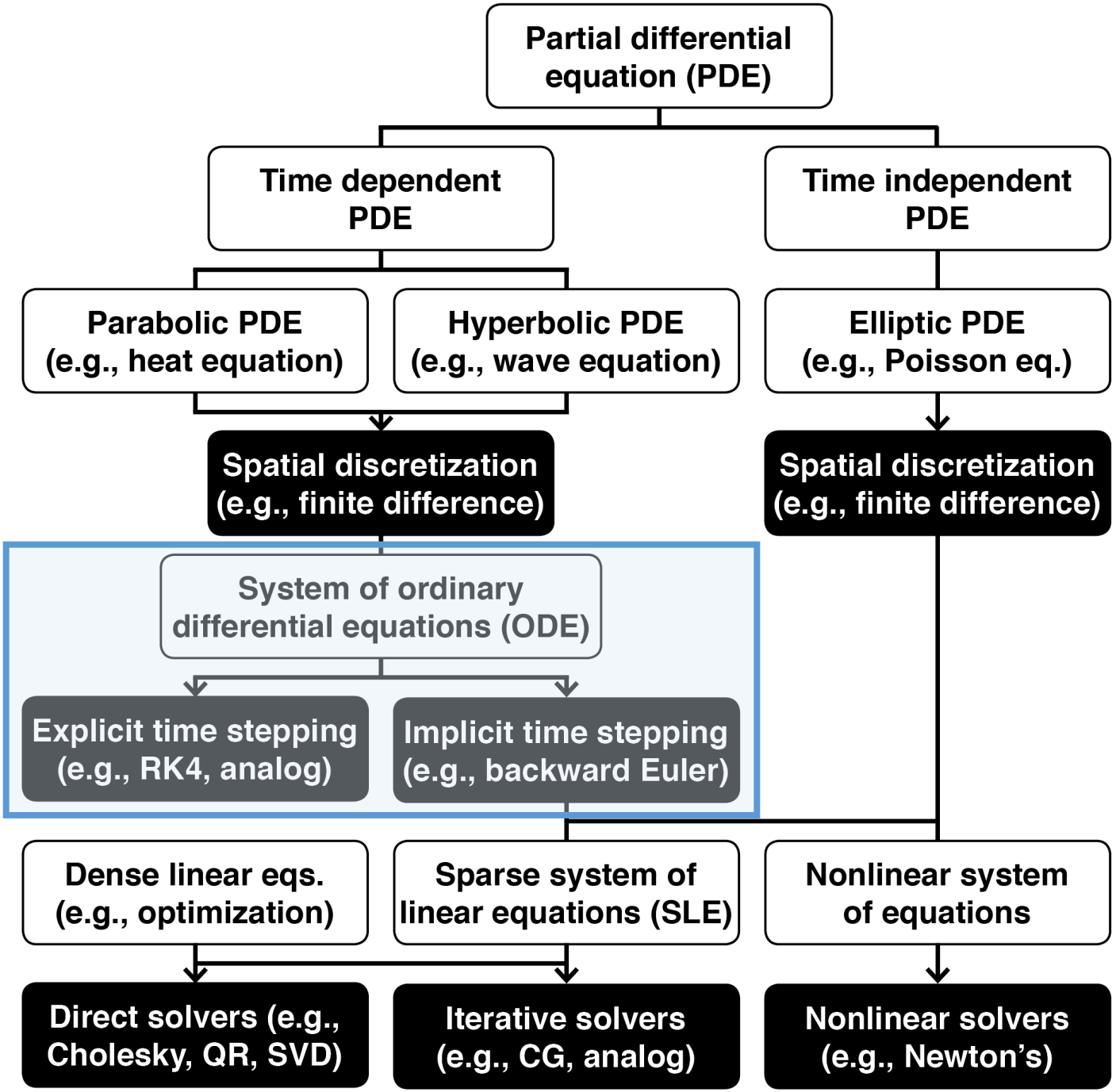
“Digital HPC” vs. “Analog CPS”

- **Analog must support algorithms and abstractions to succeed in modern digital world**
  - Demo fundamental algorithms, not specific applications

- **In terms of continuous math applications, CPS and HPC workloads are similar**
  - CPS workloads solve a lot of optimization problems: sensor fusion, path planning, actuator inverse kinematics
  - EEMBC, an embedded systems benchmark, has workloads categorized as linear algebra, spectral method problems

- **Will there be pure analog CPS workloads in future?**
  - Not sure; I imagine digital will be needed for functions such as communication, networking, programming, logging

Where ODE solvers fit in



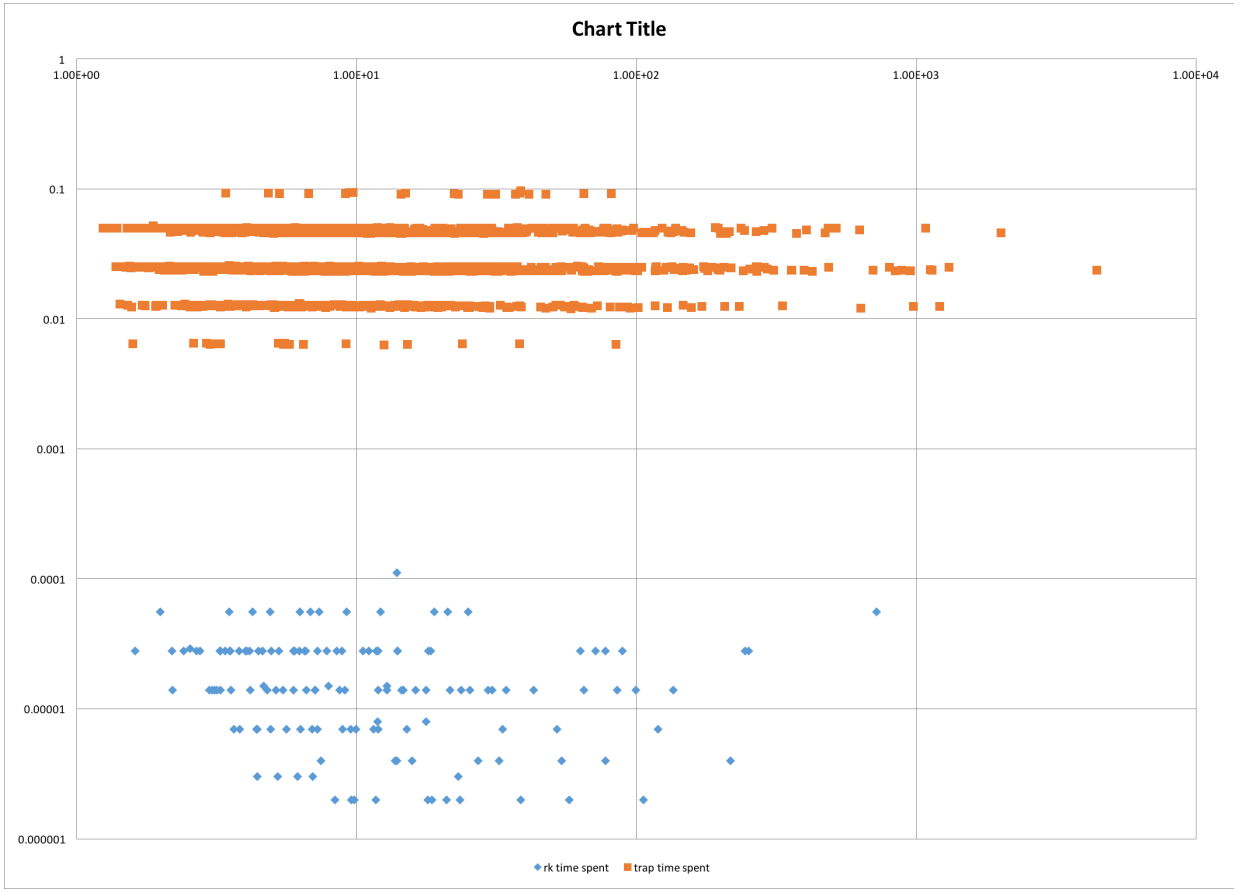
ODE explicit time stepping

- **Examples**
  - Euler's, RK4, Adams-Bashforth, analog differential analyzers
- **Strengths**
  - Simple, fast, straightforward for nonlinear ODEs
- **Weaknesses**
  - Inefficient at stiff problems
  - Accumulation of errors
  - When explicit solver is hardware, difficulty in splitting the problem
- **Mitigations**
  - To split problem: waveform relaxation, needs accurate timebase readout and ability to replay variables x(t) as a function of time
  - To address accumulation of errors: Korn discusses “resubstitution checks,” which requires ability to replay variables x(t)
  - Something we can try with HCDC v2 with FPGA

ODE implicit time stepping

- **Examples**
  - Backward Euler, trapezoidal, Heun's, predictor-corrector, predictor-modifier-corrector
  - discussed in analog computing only in context of hybrid computation, such as Karplus's 1968 book

- **Strengths**
  - Provides stronger guarantee that each step is accurate
  - Therefore, can take larger step sizes
- **Weaknesses**
  - Needs to solve a lot of inverse problems of the form x = h^-1(0)
  - Take 100x number of compute cycles per step



Recap of ISCA paper's direction

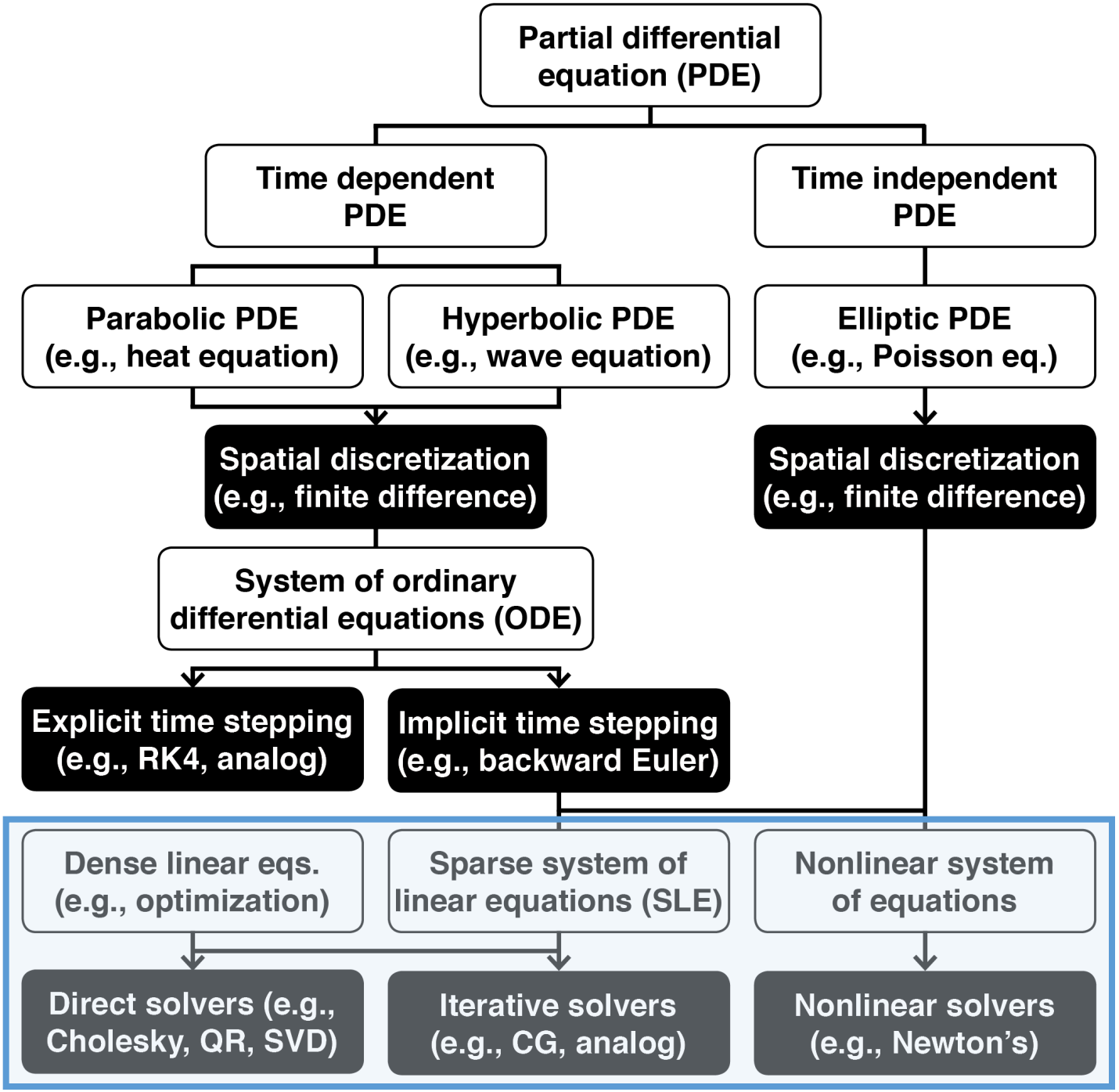
- **Analog computing solving one class of inverse problems**
- **What are inverse problems?**
  - given heuristic h(x)
  - find x such that heuristic = 0
  - x = h^-1(0)
  - h(x) can be linear or nonlinear
  - if h(x) is linear, it can be sparse, or dense

- **From Prof. David Keyes lecture:**

Dominant cycle burners

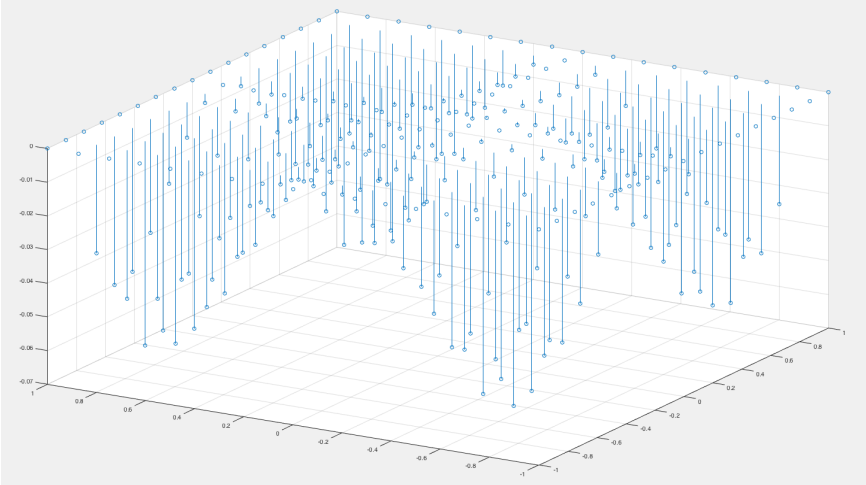
- **Poisson-like elliptic problems**
  - **Highest order operators in many PDE-based models in fluids, solids, E&M, radiation, DFT, MD, etc.**
- **Linear algebra on dense symmetric/Hermitian matrices**

Where inverse problems fit in



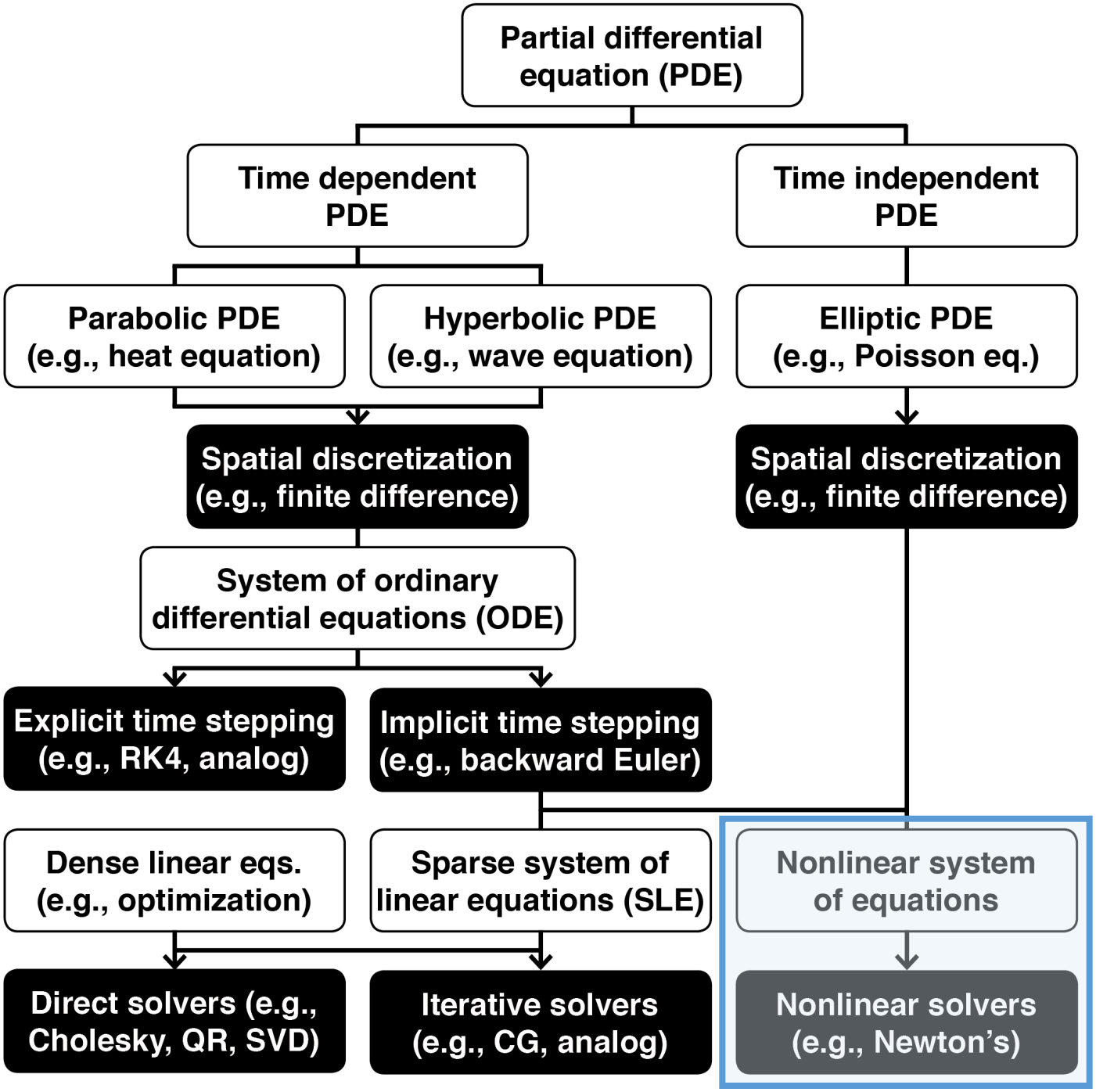
Recap of ISCA paper's direction

- **Use analog computing as a "super operator" for sparse problems**
  - Then, using multigrid, domain decomposition, and other higher level algorithms, analog can help solve larger problems



- **Findings**
  - Analog is fast, but not game-changing; Digital has advantage in CG algorithm
  - Analog area consumption is high
- **What it needs to succeed**
  - Optimum gradient method, discussed in in Karplus and Korn
  - I'll do in depth experiments using HCDC v2
  - Vary problem size (up to 4x4), 2d, 3d, dense connectivity
  - Randomized coefficients and boundary conditions

Possible direction: nonlinear algebra



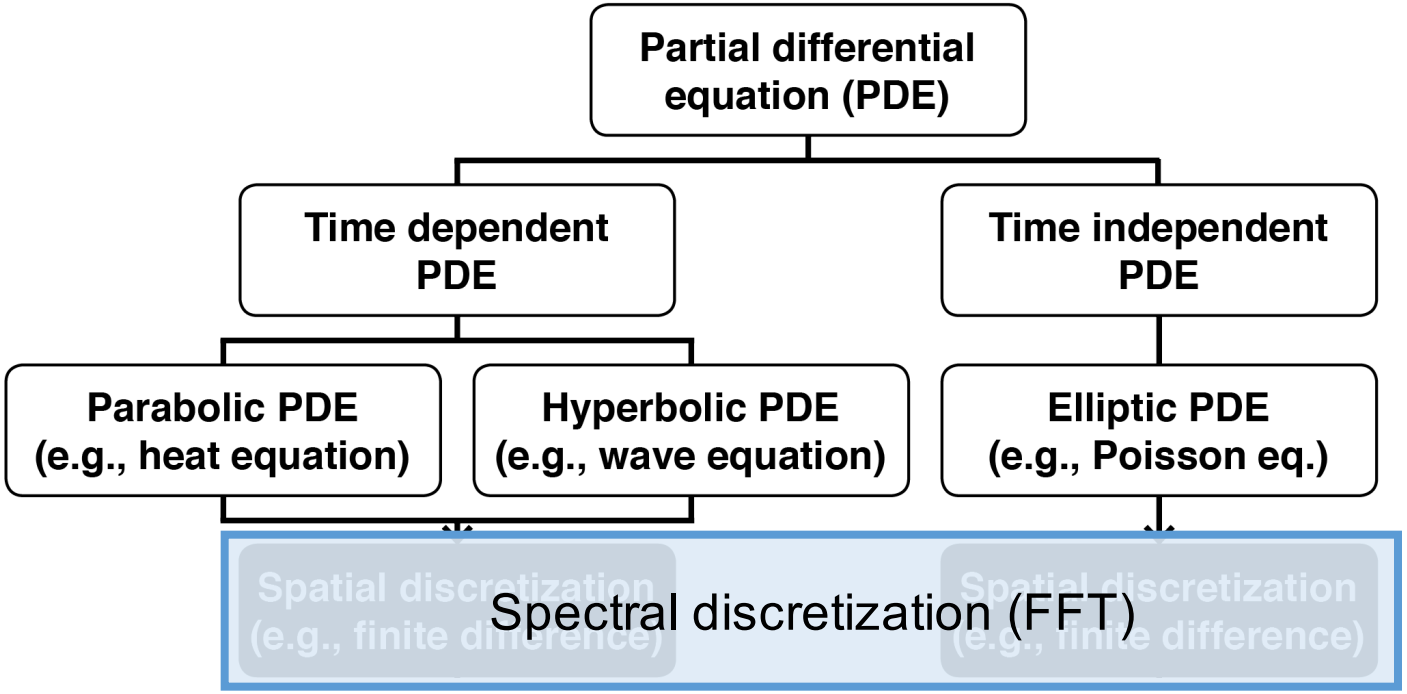
Possible direction: nonlinear algebra

- **Nonlinear PDEs yield nonlinear systems of equations**
  - The most basic class of nonlinear PDEs are semilinear PDEs such as advection-diffusion
  - Spatial discretization yields nonlinear ODEs
  - Time stepping on nonlinear ODEs requires solving nonlinear systems of equations

- **Using analog to assist Newton methods**
  - If we can generate f(x), then if we put f(x) in amplifier feedback path we can find f^-(x)

- **Mentioned in the literature**
  - Roots of polynomials discussed in Fifer 1961 chapter 22
  - Explored in Cowan's thesis
  - In digital, methods are diverse: Newton-Raphson, Gauss-Newton, L-M, Broyden's (but all are unreliable)

Possible direction: spectral methods



Is the HCDC design an efficient analog spectrum analyzer? Would it be as efficient as FFT, a hierarchical method?

Possible direction: probability & statistics

- **Mentioned in the literature**
  - Jackson 1960 chapter 10-5
  - Fifer 1961 chapter 26
  - Karplus 1972 chapter 11
  - Talks about correlations

- **Can analog solve routines in modern probability-based applications?**
  - Useful in dynamic programming & graphical models
  - Such as Viterbi & hidden Markov models

Possible direction: integral equations

- **Mentioned in the literature**
  - Mentioned in most analog books very briefly
  - Has a chapter in Numerical Recipes in C
  - Otherwise much less important than differential equations

- **Examples**
  - Fredholm equations
  - Volterra equations

Possible direction: N-body

- **Increasingly the dominant workload in HPC**
  - Pure PDE problems no longer the focus of attention
  - Driven by particle simulations, molecular dynamics

- **Example algorithms**
  - FMM: fast multipole method relies on the continuous Green's function
  - Barnes-Hut
  - Hartree-Fock

HCDC v2 FPGA Requirements

- Fold the instruction set into hardware (so configuration bitstream is generated by HW not SW)
- Ability to capture variables x(t) as a function of time
- Ability to replay variables x(t) as a function of time
- Ability to set up HCDC as linear solver using optimum gradient descent, or similar method
- Ability to execute multigrid algorithm for 2D, 3D PDEs, subject to randomized boundary conditions

JSSC Comparison Refinement

- **Sleepwalker core is built on the OpenMSP430 project**
  - OpenMSP430 is RTL code for an MSP430, hosted at OpenCores.org
  - Mature project with confirmed FPGA, ASIC implementations
  - Cycle by cycle the same as the official MSP430 instruction set manual
  - I have an MSP430 environment now and I'm working on timing Euler, RK4 code

Floating point operation energy, area for 1GHz 32nm ASIC

number representation	number precision	operation	latency (cycles)	total latency (seconds)	power (microJ/s)	operation energy (pJ)	area (um^2)	area (mm^2)
integer	64 bit	add subtract	1	1.04E-09	487.4012	0.51	3526.8	0.00353
integer	32 bit	add subtract	1	1.00E-09	216.5812	0.22	1457.7	0.00146
integer	64 bit	multiply	20	2.42E-08	1.08E+04	261.07	81016.7	0.08102
integer	32 bit	multiply	20	2.00E-08	3.22E+03	64.47	26247.4	0.02625
integer	32 bit	divide	30	3.09E-08	5.15E+03	159.00	83771.7	0.08377
floating point	double precision	add subtract	5	5.00E-09	947.4885	4.74	21029.7	0.02103
floating point	single precision	add subtract	4	4.64E-09	1.07E+03	4.98	9697.4	0.00970
floating point	double precision	multiply	32	3.30E-08	3.51E+03	115.54	67052.6	0.06705
floating point	single precision	multiply	16	1.60E-08	3.26E+03	52.17	25943.3	0.02594
floating point	double precision	divide	30	3.15E-08	1.29E+04	407.39	190673.8	0.19067
floating point	single precision	divide	30	3.36E-08	1.01E+04	337.68	76995.4	0.07700

For reference from Keckler et al., "GPUs and The Future of Parallel Computing"

- Intel Westmere CPU takes 1700 pJ per floating point op
- Nvidia Fermi GPU takes 225 pJ per floating point op
- Raw operation (no overheads) takes 50 pJ per double precision fused multiply