

Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations

Yipeng Huang, Ning Guo, Kyle Mandli,
Mingoo Seok, Yannis Tsividis, Simha Sethumadhavan

Columbia University

Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations

Yipeng Huang, Ning Guo, Kyle Mandli,
Mingoo Seok, Yannis Tsividis, Simha Sethumadhavan

Columbia University

2000	2016
The Best of the 20th Century: Editors Name Top 10 Algorithms Society for Industrial and Applied Mathematics Compiled by Barry Cipra	Most mentioned in The Princeton Companion to Applied Mathematics Compiled by Nick Higham
-	Newton methods
Matrix factorizations (e.g., LU decomposition)	
Eigenvalue algorithms (e.g., QR algorithm)	
Monte Carlo method	
Fast Fourier transform	
Krylov subspace iteration methods (e.g., conjugate gradients)	
Simplex method for linear programming	
Others: Fortran compiler, Quicksort, integer relation detection, fast multipole	Others: JPEG, PageRank, Kalman filter

2000

The Best of the 20th Century: Editors Name Top 10 Algorithms
Society for Industrial and Applied Mathematics
Compiled by Barry Cipra

2016

Most mentioned in
The Princeton Companion to Applied Mathematics
Compiled by Nick Higham

-

Newton methods

Matrix factorizations (e.g., LU decomposition)

Eigenvalue algorithms (e.g., QR algorithm)

Monte Carlo method

Fast Fourier transform

Krylov subspace iteration methods (e.g., conjugate gradients)

Simplex method for linear programming

Others: Fortran compiler, Quicksort,
integer relation detection, fast multipole

Others: JPEG, PageRank, Kalman filter

2000

The Best of the 20th Century: Editors Name Top 10 Algorithms
Society for Industrial and Applied Mathematics
Compiled by Barry Cipra

2016

Most mentioned in
The Princeton Companion to Applied Mathematics
Compiled by Nick Higham

-

Newton methods

Matrix factorizations (e.g., LU decomposition)

Eigenvalue algorithms (e.g., QR algorithm)

Monte Carlo method

Fast Fourier transform

Krylov subspace iteration methods (e.g., conjugate gradients)

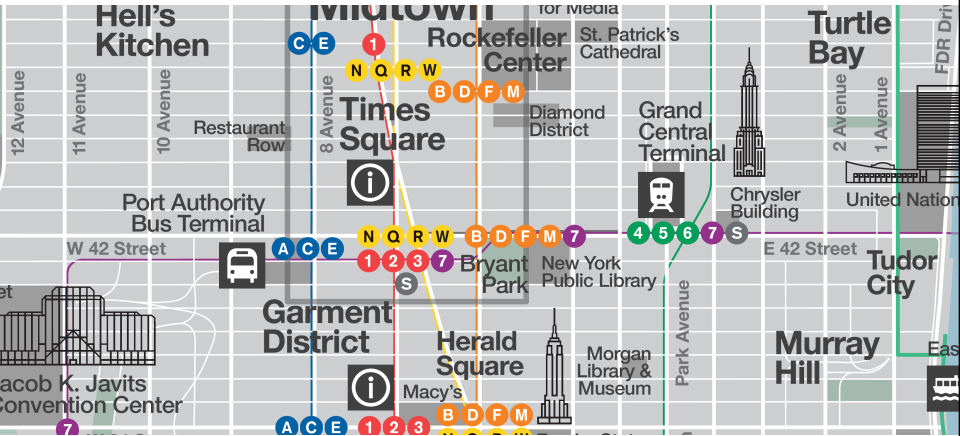
Simplex method for linear programming

Others: Fortran compiler, Quicksort,
integer relation detection, fast multipole

Others: JPEG, PageRank, Kalman filter

Discipline	Nonlinear PDEs	Cause of nonlinear behavior
Applied physics	Plasma & fluids	Multiphysics
Applied mathematics	Nonlinear waves	Dispersive effects
Chemical engineering	Combustion models	Multiphysics
Civil engineering	Solid mechanics	Nonlinear spring forces
Electrical engineering	Circuit simulation	Nonlinear components
Operations research	Convex optimization	Nonlinear objectives & constraints
Mechanical engineering	Optimal control	Nonlinear Euler-Lagrange equations

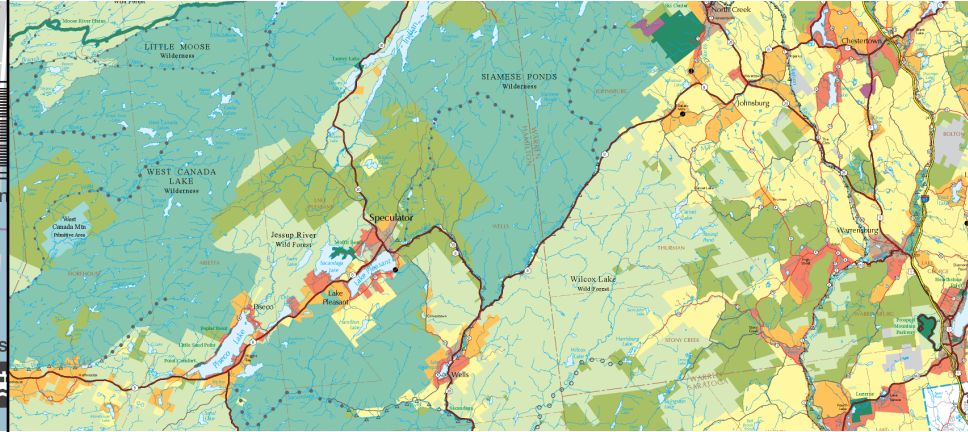
Linear



Simple map

Few turns

Nonlinear



Check map often





Need to start close

Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations

Yipeng Huang, Ning Guo, Kyle Mandli,
Mingoo Seok, Yannis Tsividis, Simha Sethumadhavan

Columbia University

Motivation for hybrid analog-digital architecture

	Ability to solve nonlinear problems	Solution accuracy & precision	Technique diversity	Problem sizes
Analog				
Digital				

Both analog and digital needed for target workload

Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

Hybrid analog-digital system combines both for speed & precision
How to map a PDE problem to a prototype analog accelerator

Evaluation:

100× faster for approximate solution

5.7× faster and 11.6× less energy when analog helps GPU

Outline

Motivation:

Analog avoids *digital pitfalls in nonlinear problems*

Architecture:

Hybrid analog-digital system combines both for speed & precision
How to map a PDE problem to a prototype analog accelerator

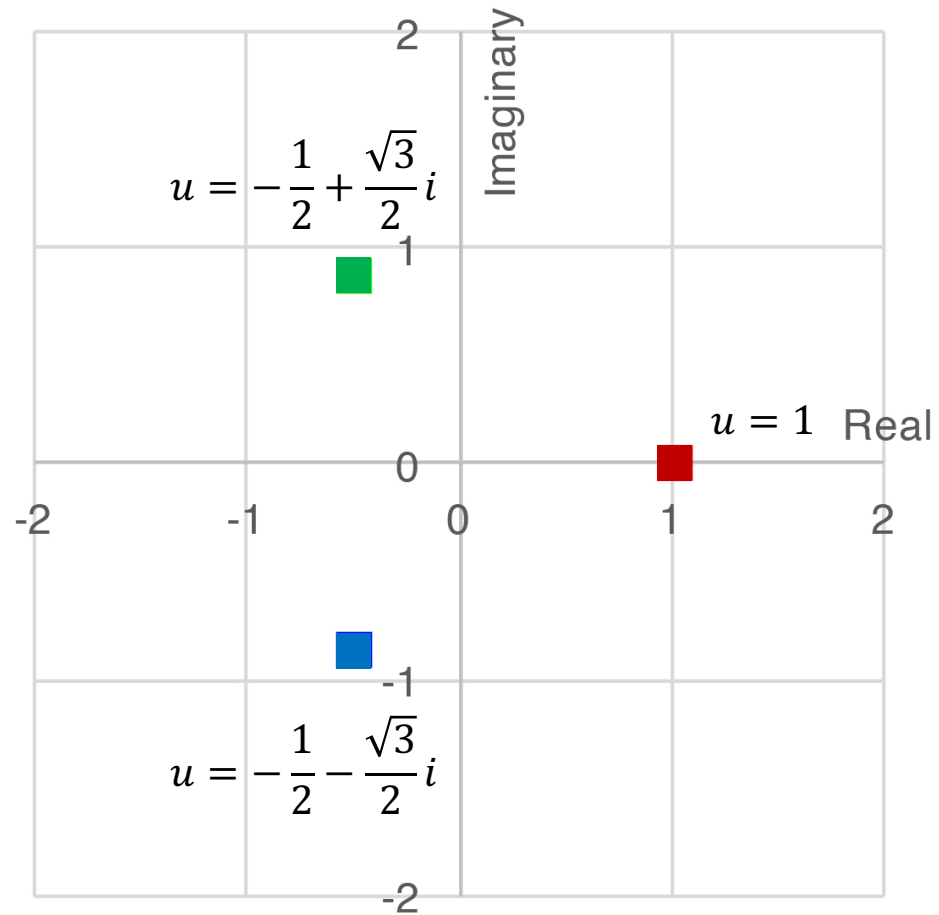
Evaluation:

100× faster for approximate solution

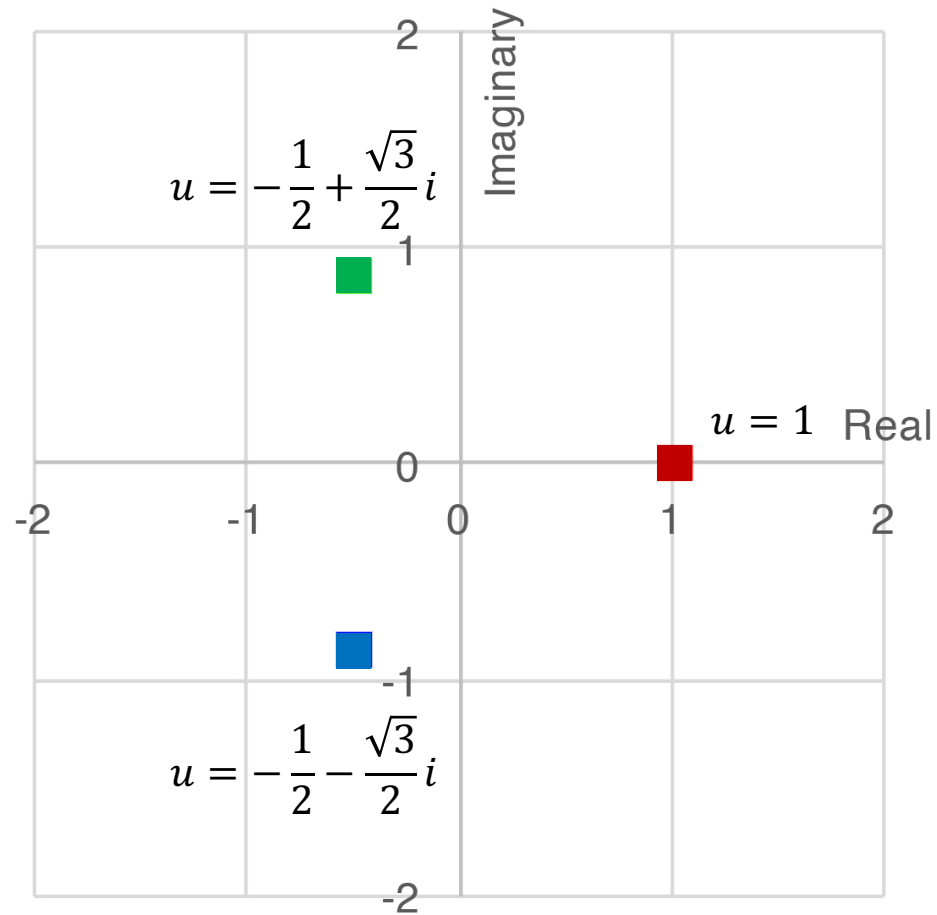
5.7× faster and 11.6× less energy when analog helps GPU

Solve for u :

$$f(u) = u^3 - 1 = 0$$



Solve for u :
 $f(u) = u^3 - 1 = 0$
nonlinear



Solve for u :

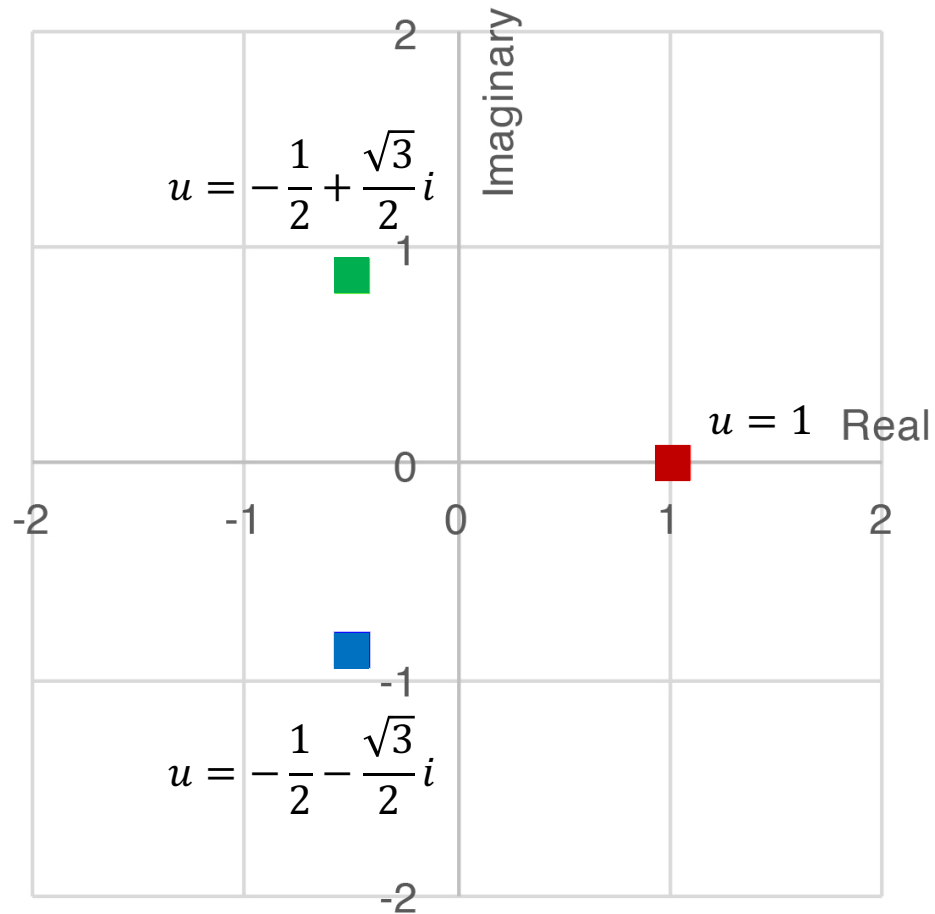
$$f(u) = u^3 - 1 = 0$$

Using Newton's method:

$$u_{next} = u_{curr} - \frac{f(u_{curr})}{f'(u_{curr})}$$

With some initial guess u_0

Returns one of three final solutions



Solve for u :

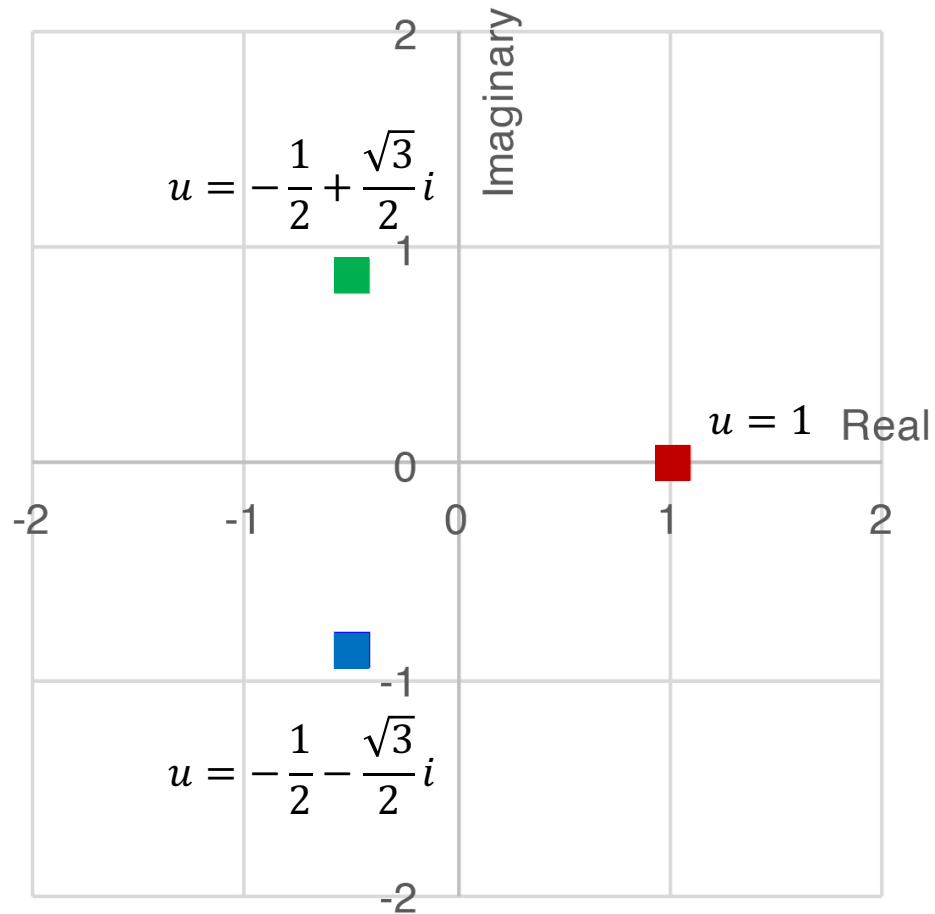
$$f(u) = u^3 - 1 = 0$$

Using Newton's method:

$$u_{next} = u_{curr} - \frac{f(u_{curr})}{f'(u_{curr})}$$

With some initial guess u_0

Returns one of three final solutions



Solve for u :

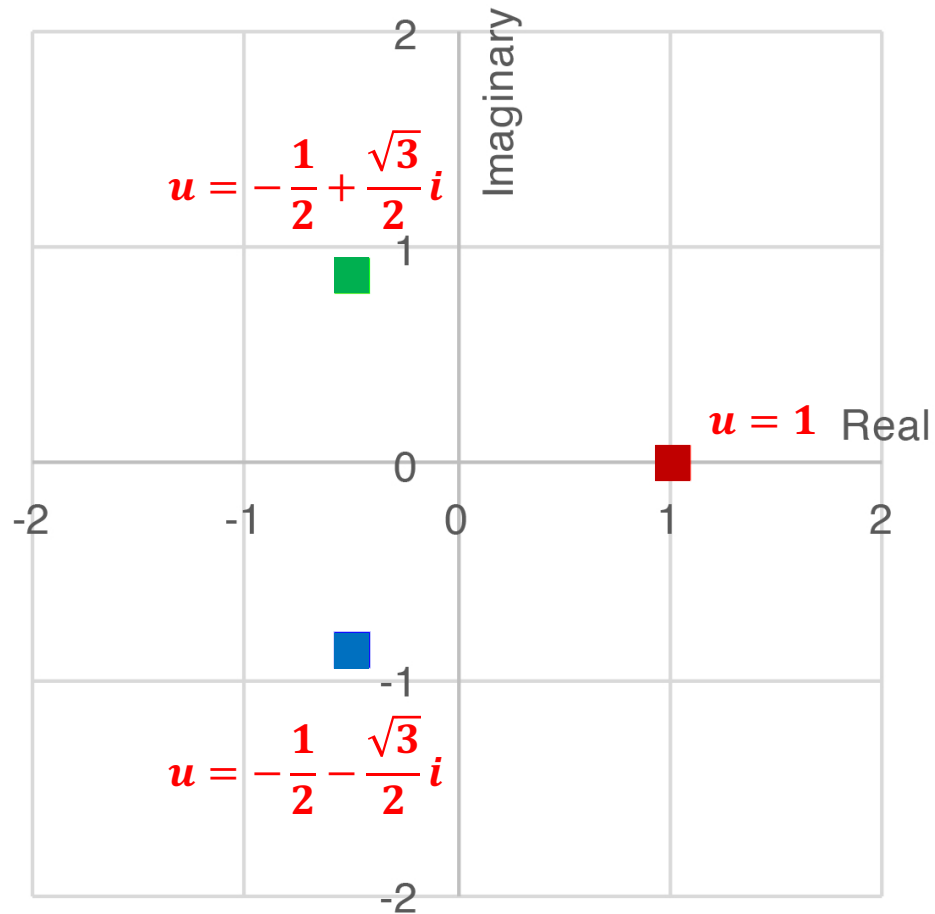
$$f(u) = u^3 - 1 = 0$$

Using Newton's method:

$$u_{next} = u_{curr} - \frac{f(u_{curr})}{f'(u_{curr})}$$

With some initial guess u_0

Returns one of three final solutions



Solve for u :

$$f(u) = u^3 - 1 = 0$$

Using Newton's method:

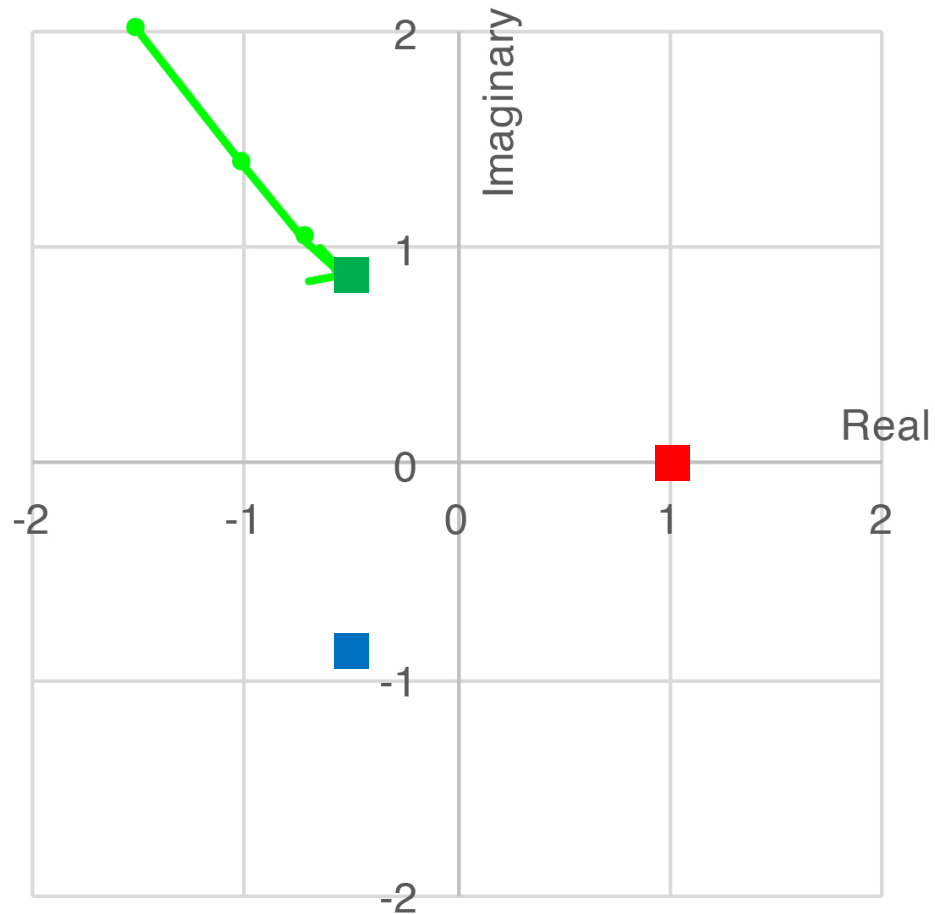
$$u_{next} = u_{curr} - \frac{u_{curr}^3 - 1}{3u_{curr}^2}$$

With initial guess:

$$u_0 = -1.5 + 2i$$

Returns final solution:

$$u_{final} = -\frac{1}{2} + \frac{\sqrt{3}}{2}i$$



Solve for u :
 $f(u) = u^3 - 1 = 0$

Using Newton's method:

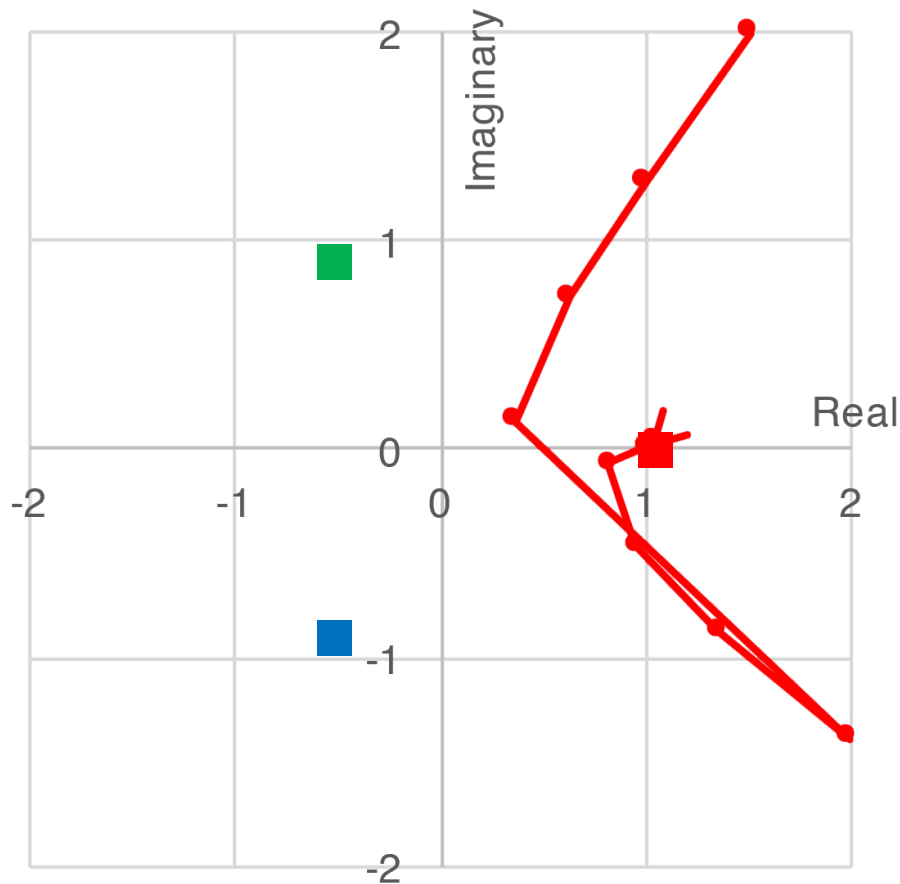
$$u_{next} = u_{curr} - \frac{u_{curr}^3 - 1}{3u_{curr}^2}$$

With initial guess:

$$u_0 = 1.5 + 2i$$

Returns final solution:

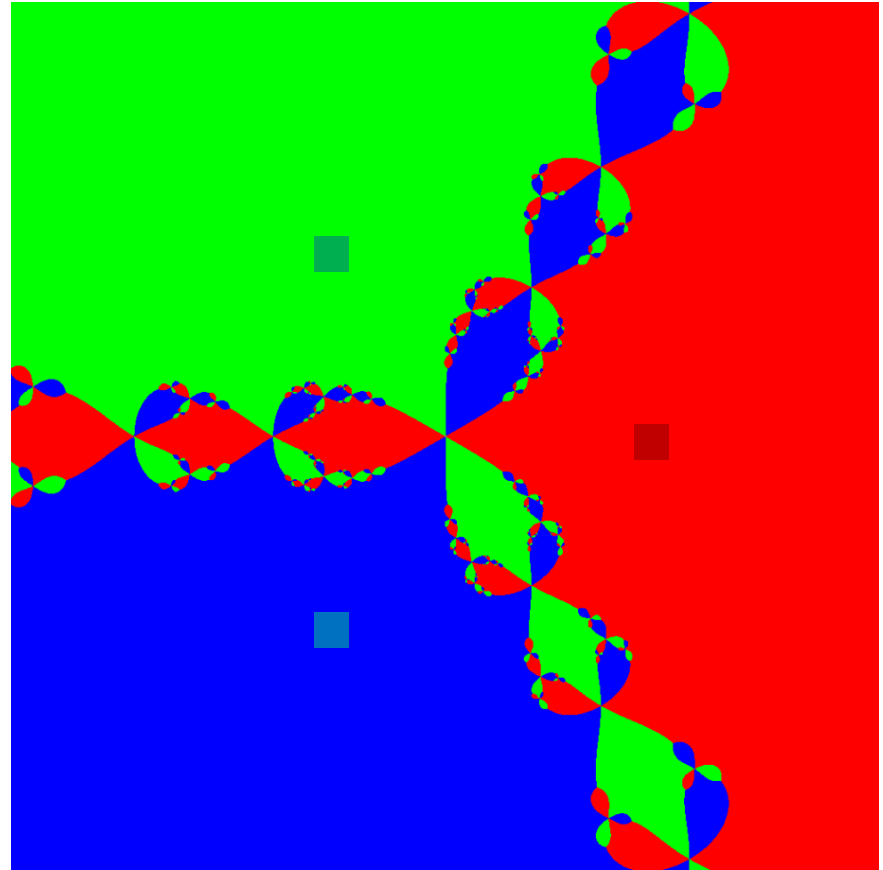
$$u_{final} = 1$$



Plot of final solution vs. initial guess for Newton's method

Three colors indicate which of the three
roots the initial guesses go to

**Like hiking in wilderness,
important to start close to destination
...or else easy to get lost**



Solve for u :

$$f(u) = u^3 - 1 = 0$$

Using damped Newton's method:

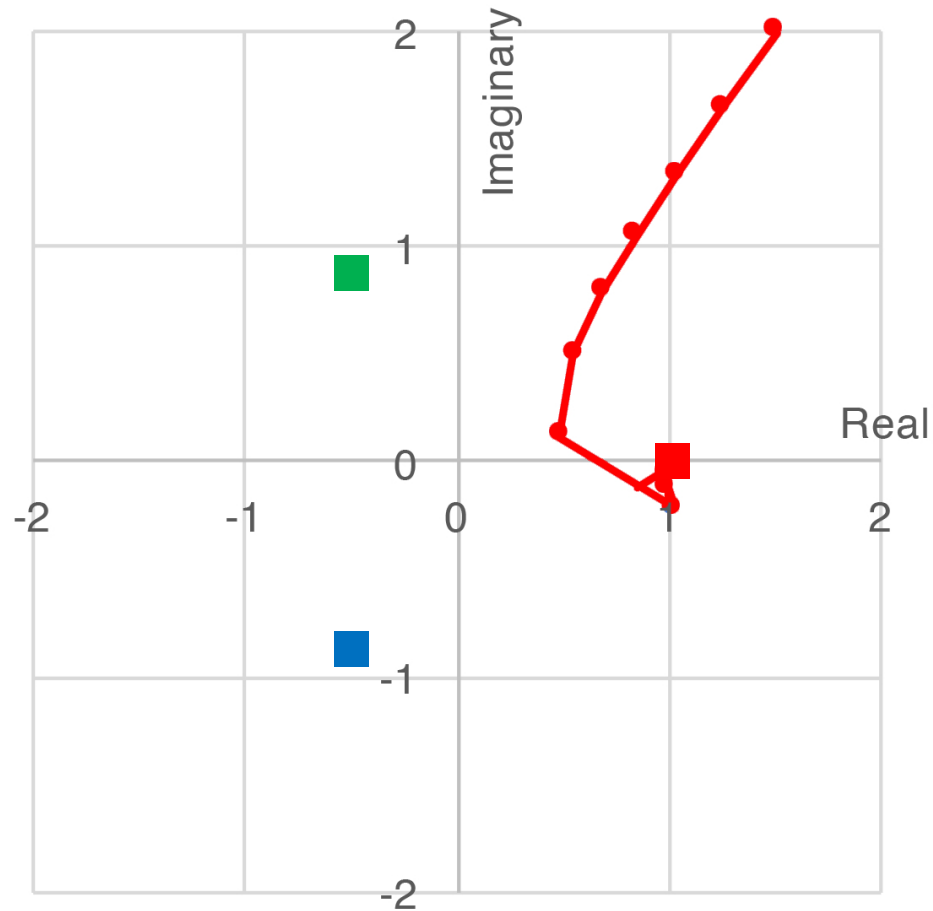
$$u_{next} = u_{curr} - \frac{1}{2} \frac{u_{curr}^3 - 1}{3u_{curr}^2}$$

With initial guess:

$$u_0 = 1.5 + 2i$$

Returns final solution:

$$u_{final} = 1$$



Solve for u :

$$f(u) = u^3 - 1 = 0$$

Using damped Newton's method:

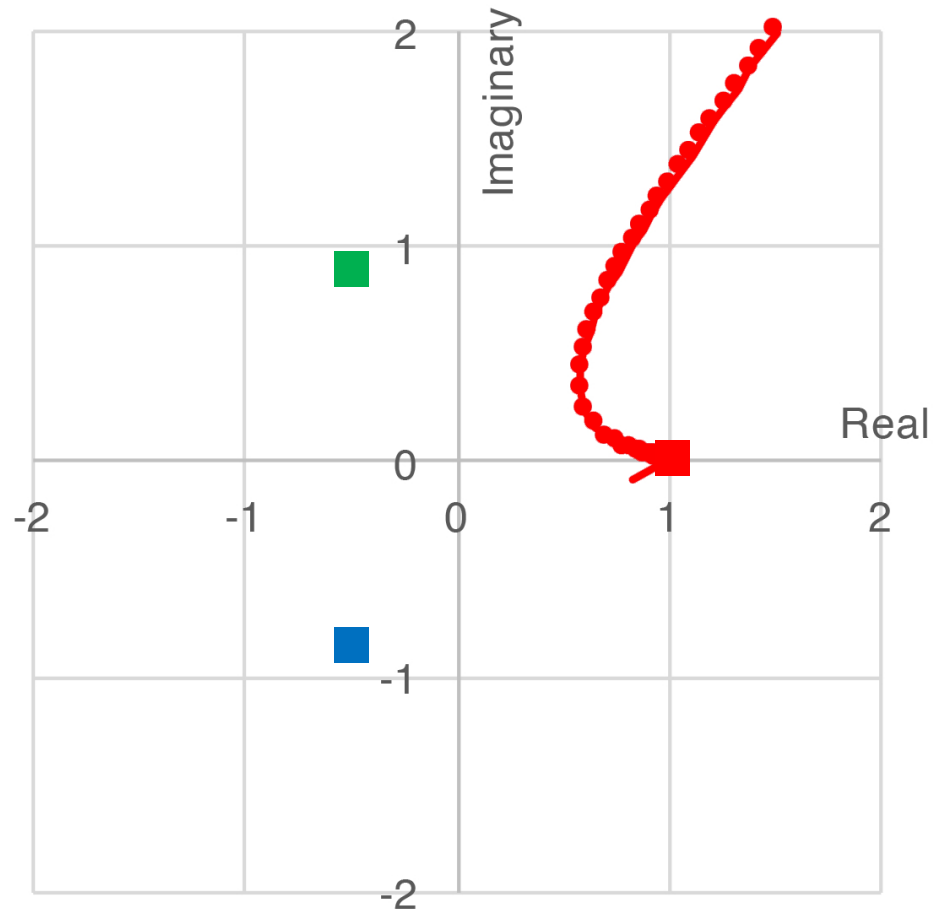
$$u_{next} = u_{curr} - \frac{1}{8} \frac{u_{curr}^3 - 1}{3u_{curr}^2}$$

With initial guess:

$$u_0 = 1.5 + 2i$$

Returns final solution:

$$u_{final} = 1$$

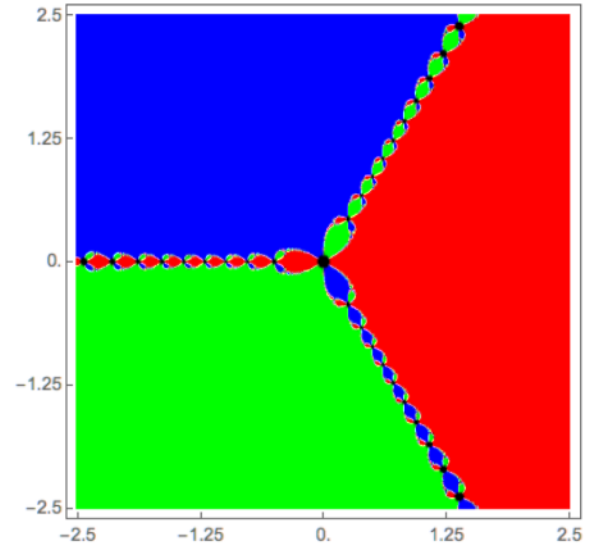
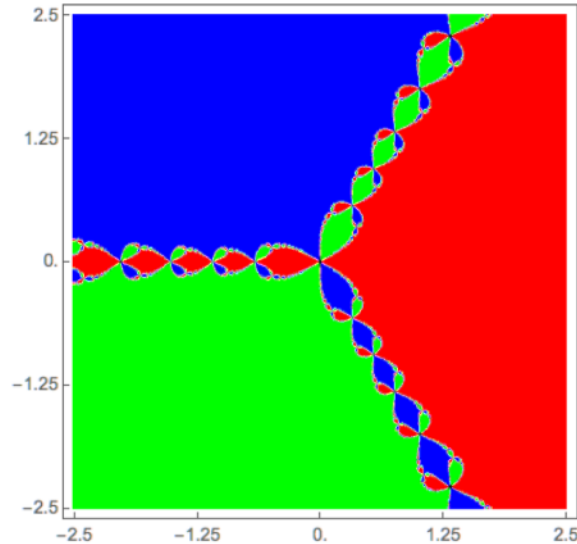
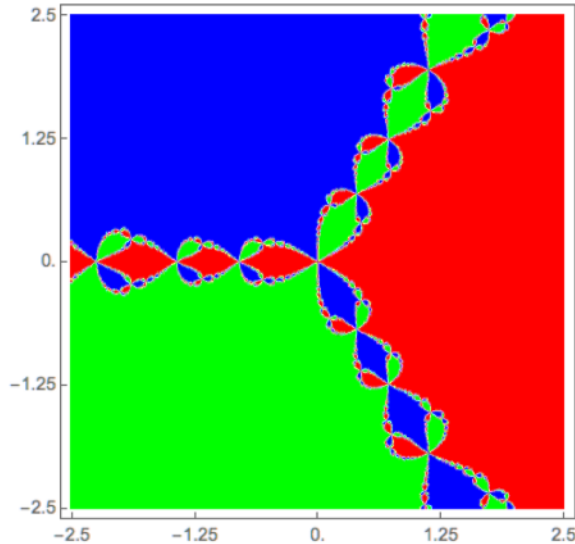


Like hiking in wilderness, better to check map often

Don't go too far between checking map, or else easy to get lost

With smaller damping parameter, plot becomes more contiguous...

...at the cost of more digital computation time



Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

Hybrid analog-digital system combines both for speed & precision
How to map a PDE problem to a prototype analog accelerator

Evaluation:

100× faster for approximate solution

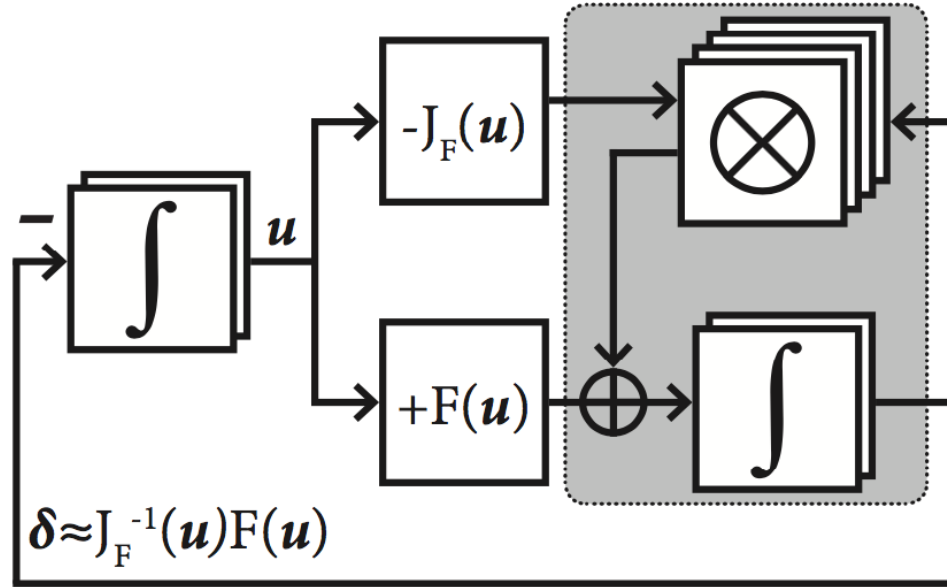
5.7× faster and 11.6× less energy when analog helps GPU

Push damped Newton's method to limit, get continuous Newton's method

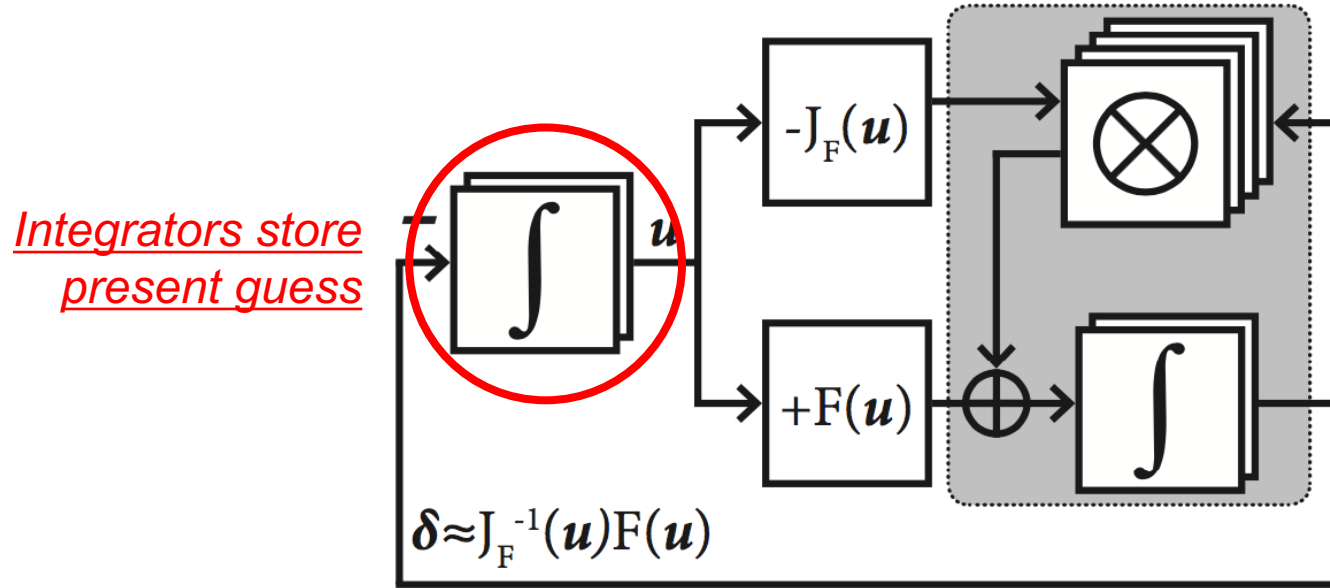
Take (nearly) infinitely many (nearly) infinitesimal steps

Like continuously checking map while hiking in wilderness

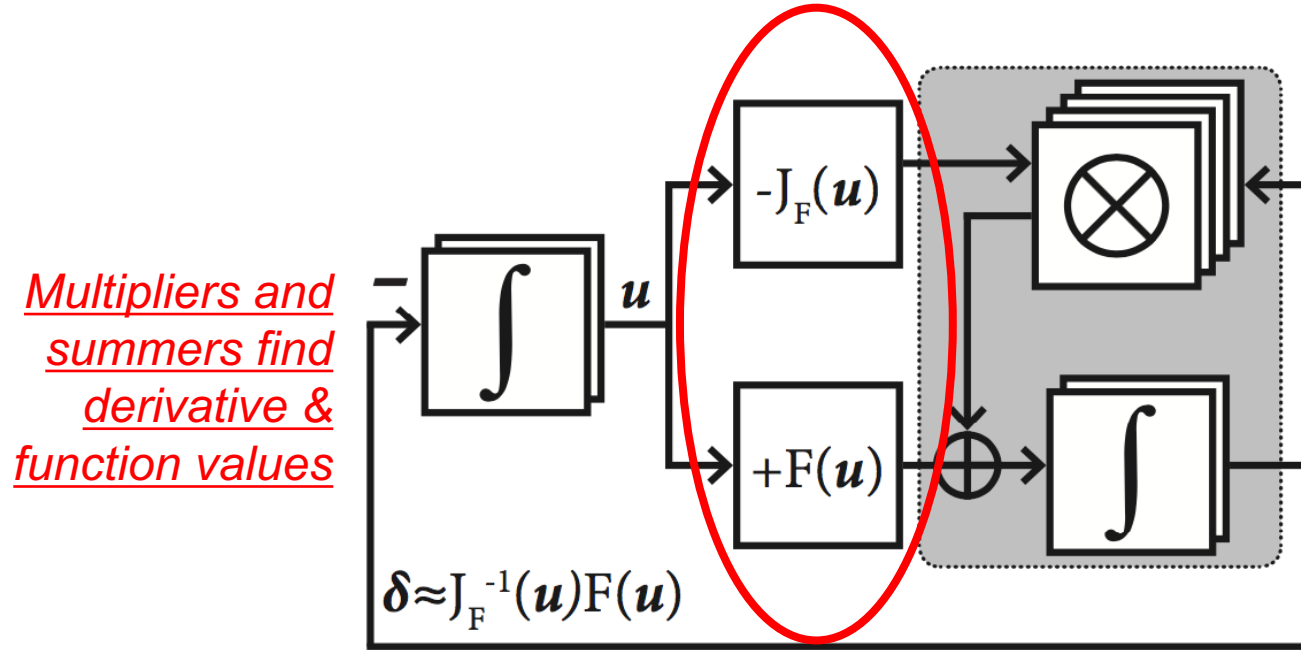
Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



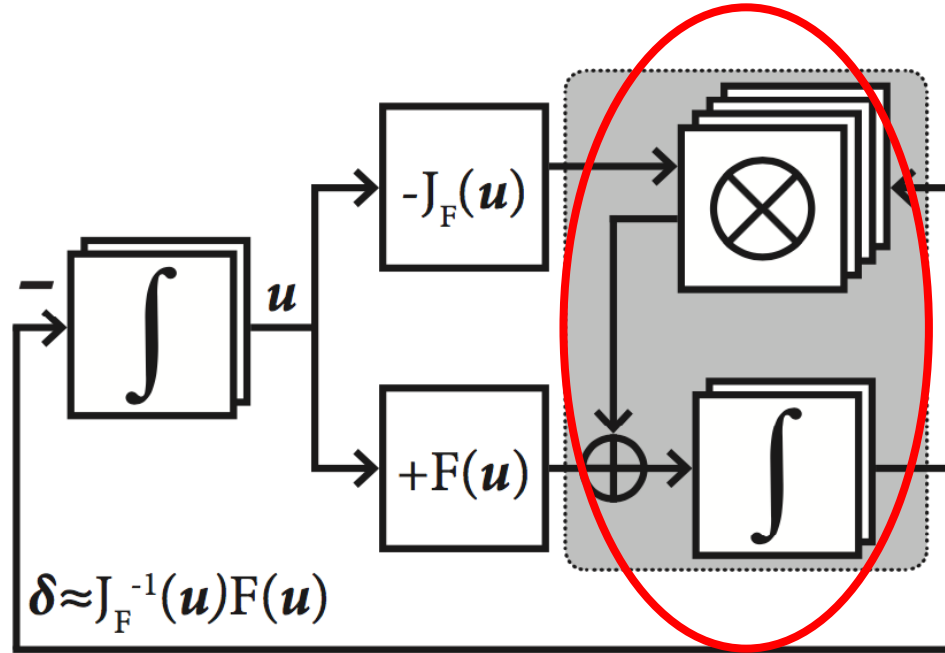
Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps

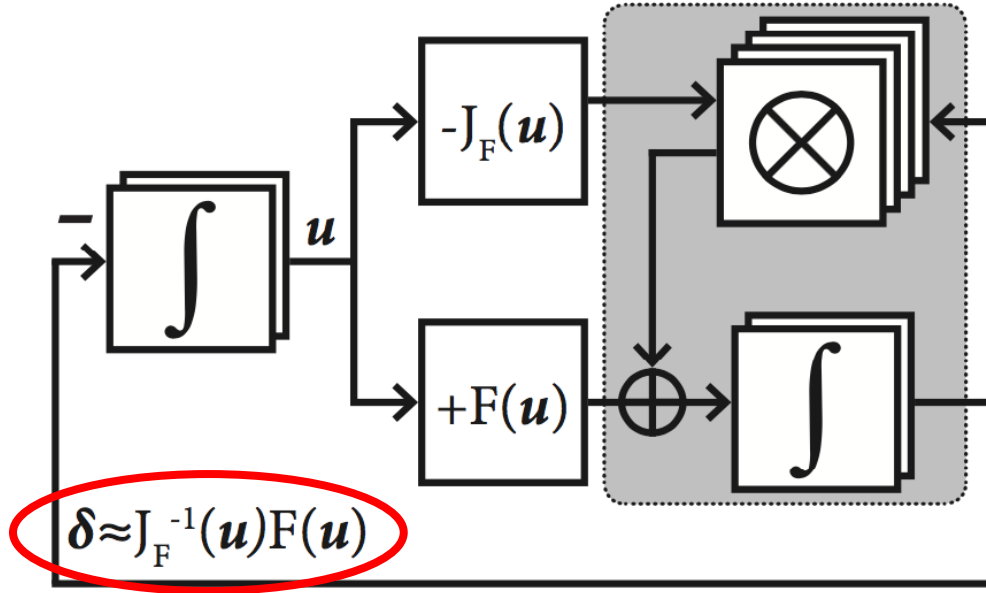


Negative feedback
solves linear
algebra problem

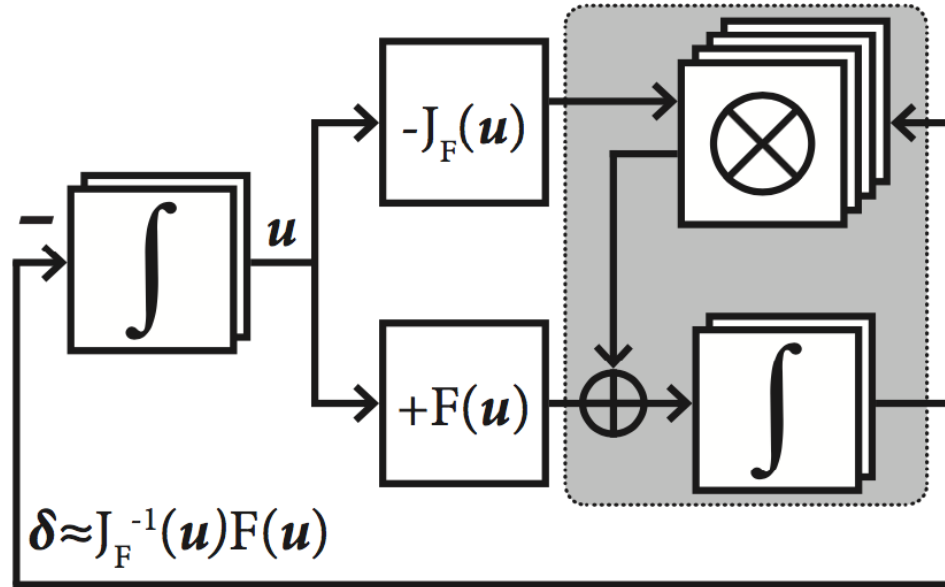
Huang et al.,
ISCA 2016

Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps

Integrators
accumulate
increments for
Newton's method



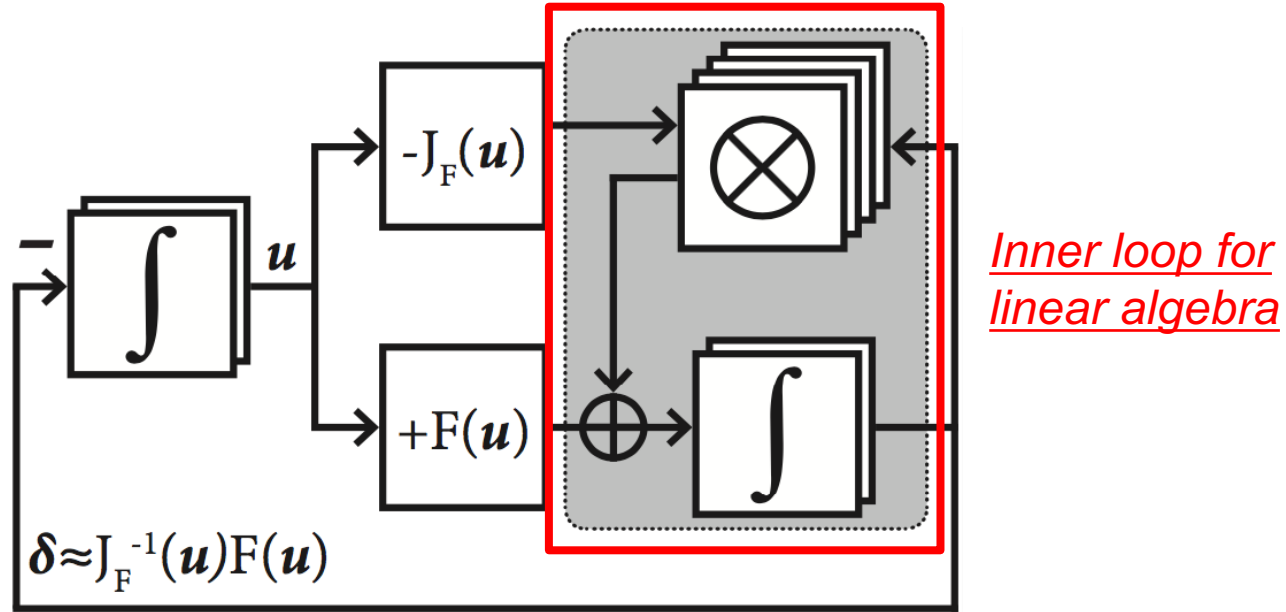
Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



Values:
analog current
and voltage

Works
continuously:
no clock or steps

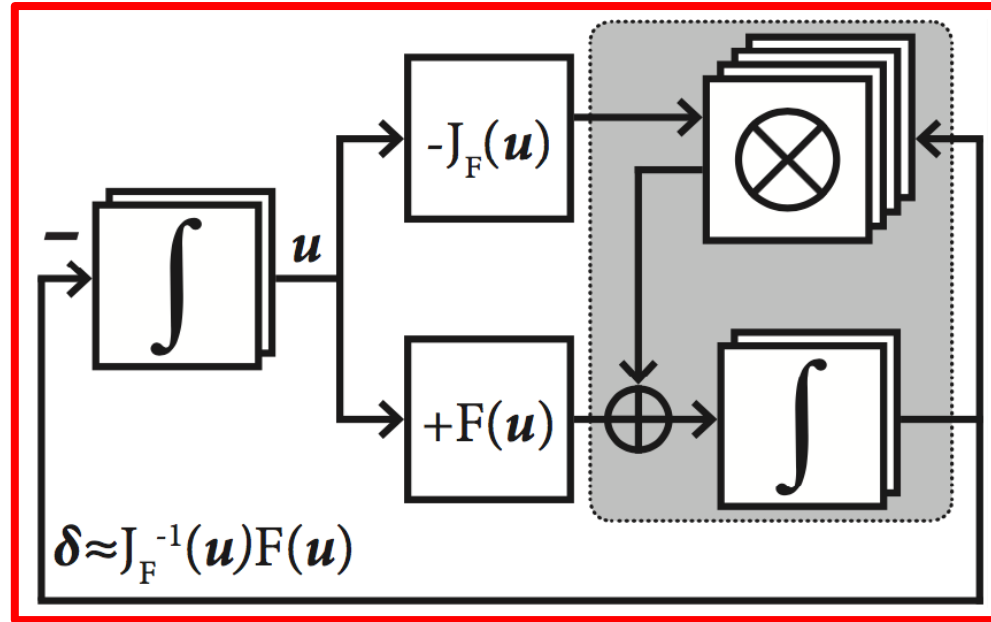
Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



Takeaway 1 of 4: how to do more work in analog?

Create analog inner loops by nesting circuits with different convergence rates

Push damped Newton's method to limit, get continuous Newton's method
Take (nearly) infinitely many (nearly) infinitesimal steps



*Outer loop for
Newton's method
invokes inner loop*

Takeaway 1 of 4: how to do more work in analog?

Create analog inner loops by nesting circuits with different convergence rates

Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

Hybrid analog-digital system combines both for speed & precision

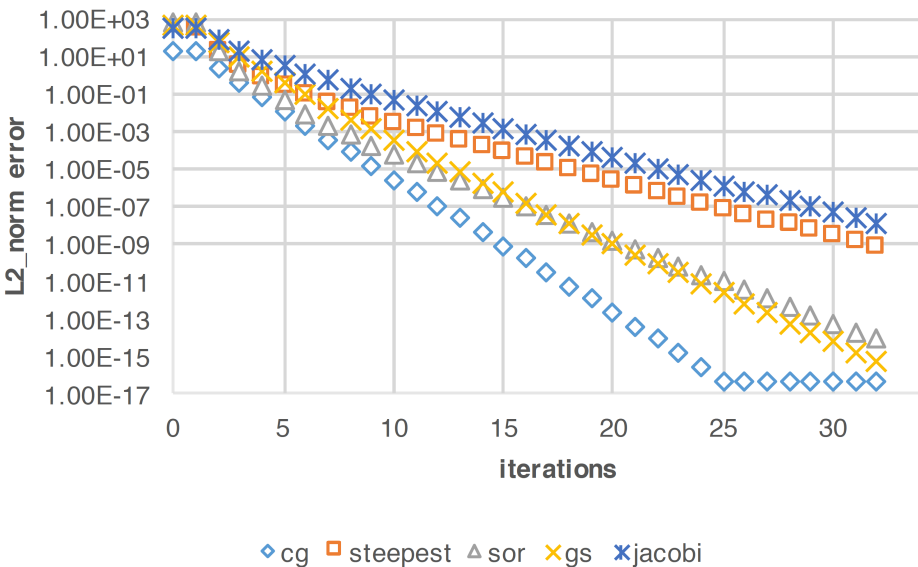
How to map a PDE problem to a prototype analog accelerator

Evaluation:

100× faster for approximate solution

5.7× faster and 11.6× less energy when analog helps GPU

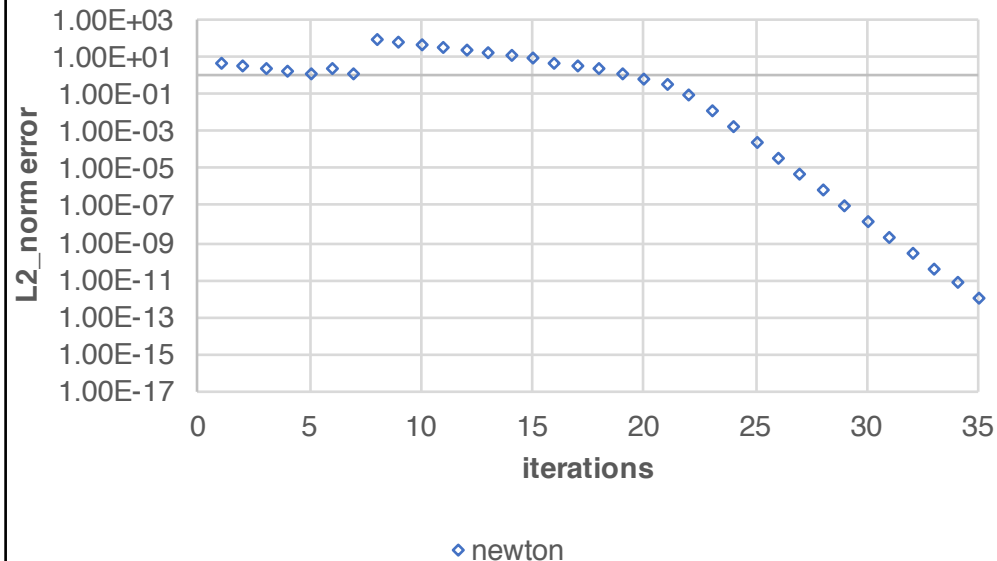
Linear



First few digits of solution cheap

Last few digits of solution expensive

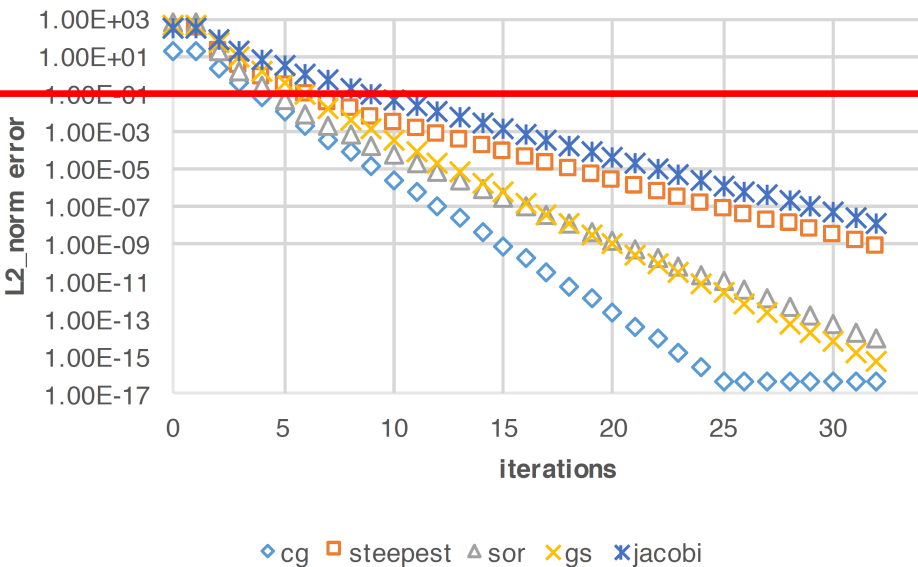
Nonlinear



Rough guess solution expensive

Refined solution from guess cheap

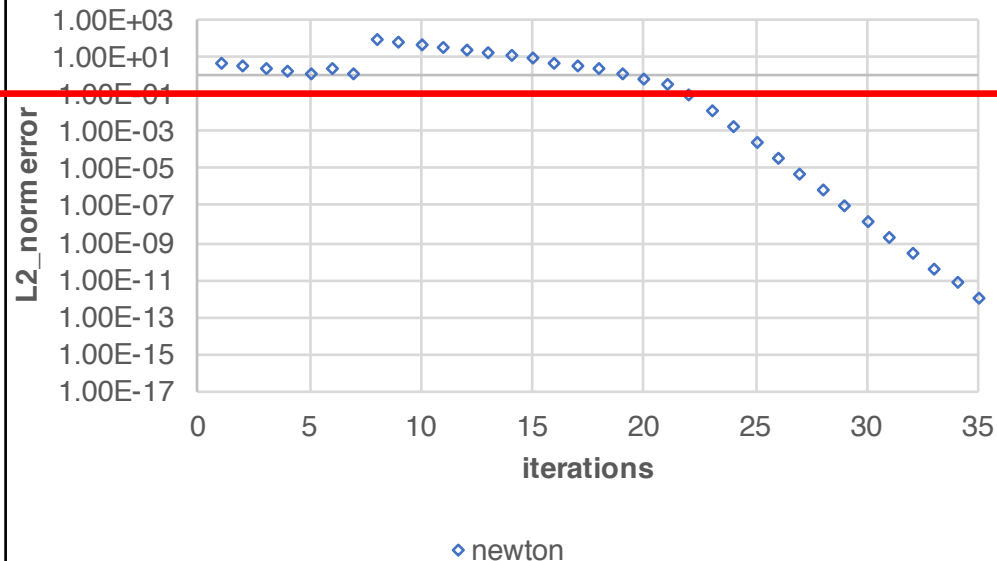
Linear



First few digits of solution cheap

Last few digits of solution expensive

Nonlinear



Rough guess solution expensive

Refined solution from guess cheap

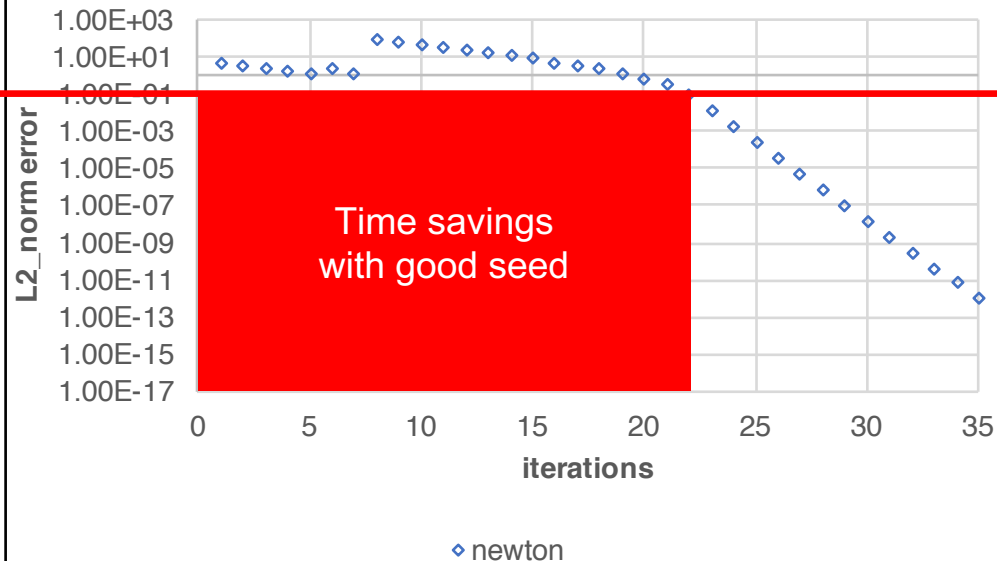
Linear



First few digits of solution cheap

Last few digits of solution expensive

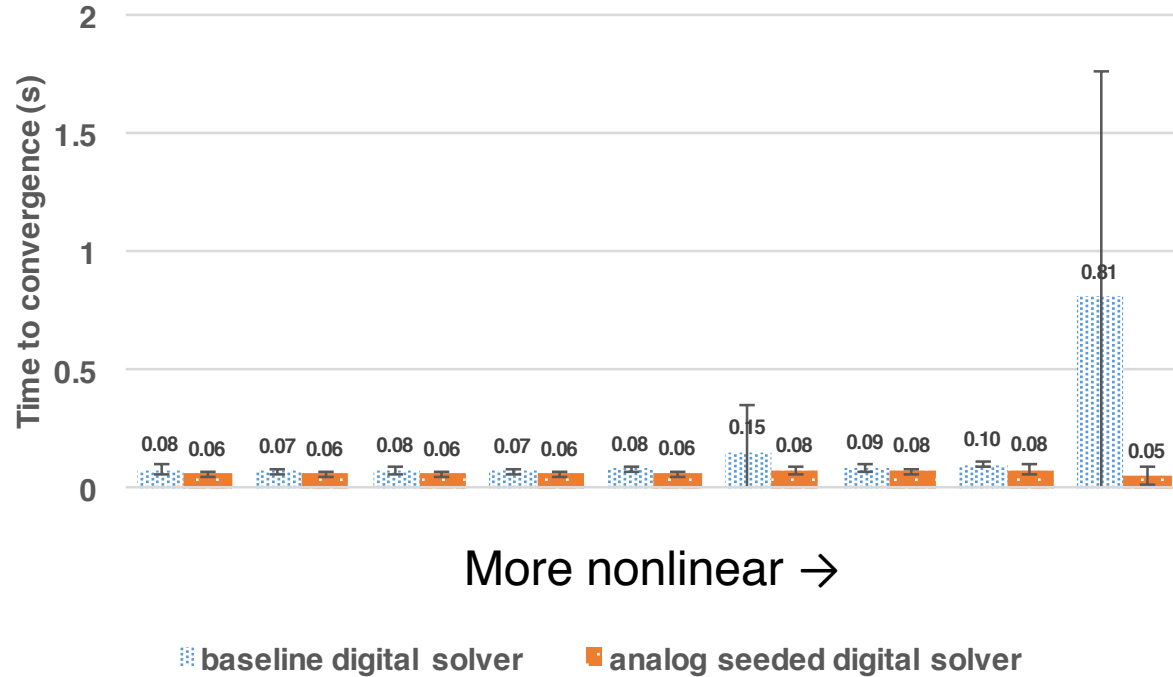
Nonlinear



Rough guess solution expensive

Refined solution from guess cheap

Solve time for digital and seeded digital solver



Takeaway 2 of 4: how to increase solution accuracy?
Cheap analog approximation → good digital initial guess

Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

Hybrid analog-digital system combines both for speed & precision

How to map a PDE problem to a prototype analog accelerator

Evaluation:

100× faster for approximate solution

5.7× faster and 11.6× less energy when analog helps GPU

Digital: lots of methods and code for breaking down PDEs

We want to keep existing code while using accelerator

Analog: can only solve ordinary differential equations...

...and algorithms phrased as ODEs, such as continuous Newton's

Takeaway 3 of 4: how to minimize reprogramming?

Retarget kernels in existing digital code to accelerate in analog

Case study: viscous 2D Burgers' equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\text{Re}} \nabla^2 \mathbf{u} = \text{RHS}$$
$$\begin{cases} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = \text{RHS}_0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = \text{RHS}_1 \end{cases}$$

Core nonlinear part of the Navier-Stokes equations for fluid dynamics
u and v are velocity of fluid in x- and y- dimensions, respectively

Case study: viscous 2D Burgers' equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\text{Re}} \nabla^2 \mathbf{u} = \text{RHS}$$
$$\begin{cases} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = \text{RHS}_0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = \text{RHS}_1 \end{cases}$$

Core nonlinear part of the Navier-Stokes equations for fluid dynamics

u and v are velocity of fluid in x - and y - dimensions, respectively

Why nonlinear: because derivatives of u and v have u and v as coefficients

Case study: viscous 2D Burgers' equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\text{Re}} \nabla^2 \mathbf{u} = \text{RHS}$$
$$\begin{cases} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = \text{RHS}_0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = \text{RHS}_1 \end{cases}$$

Core nonlinear part of the Navier-Stokes equations for fluid dynamics

u and v are velocity of fluid in x - and y - dimensions, respectively

Why nonlinear: because derivatives of u and v have u and v as coefficients

how nonlinear the PDE is depends on choice of Re , Reynolds number

Case study: viscous 2D Burgers' equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\text{Re}} \nabla^2 \mathbf{u} = \text{RHS}$$
$$\begin{cases} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = \text{RHS}_0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - \frac{1}{\text{Re}} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = \text{RHS}_1 \end{cases}$$

Core nonlinear part of the Navier-Stokes equations for fluid dynamics

u and v are velocity of fluid in x - and y - dimensions, respectively

Why nonlinear: because derivatives of u and v have u and v as coefficients

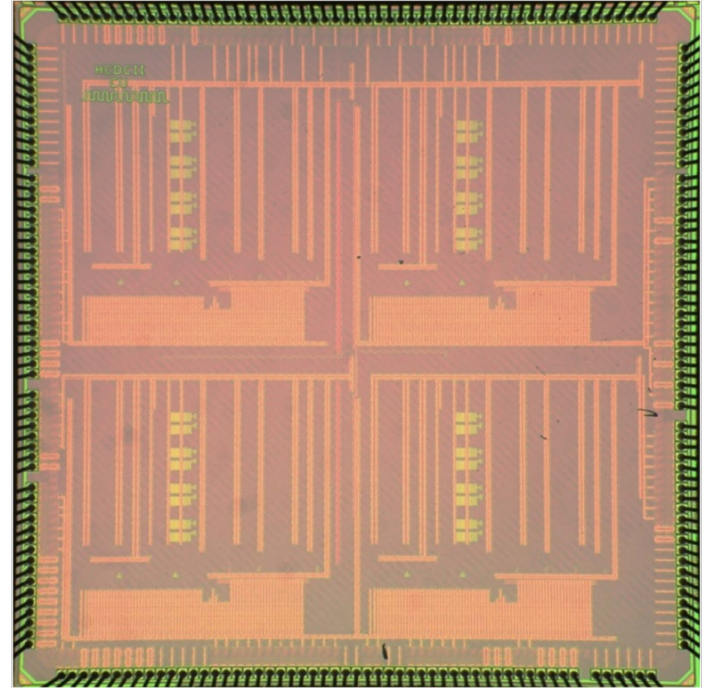
how nonlinear the PDE is depends on choice of Re , Reynolds number

To solve this PDE, do space discretization and time stepping

results in system of nonlinear algebraic equations, which we solve in analog

Prototype analog accelerator for nonlinear systems of equations

Tiled programmable analog accelerator
Enhanced calibration for all analog units



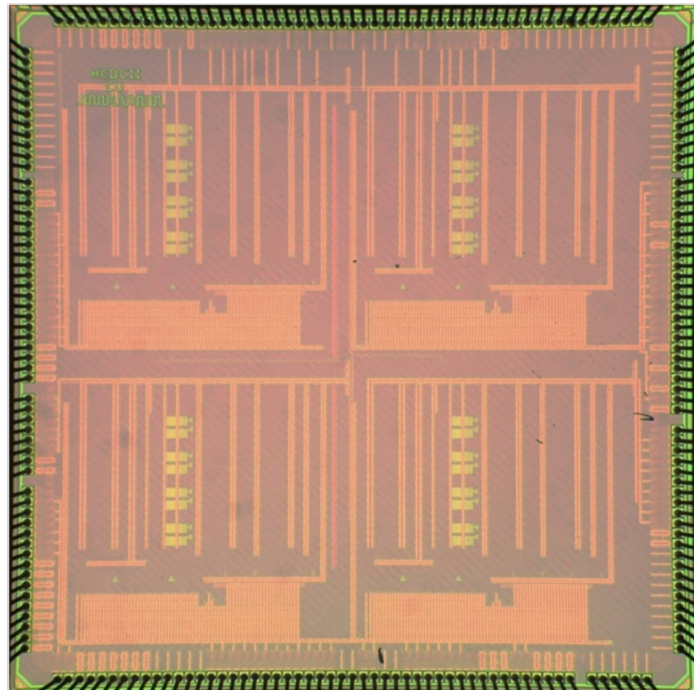
Prototype analog accelerator for nonlinear systems of equations

Tiled programmable analog accelerator

Enhanced calibration for all analog units

Analog in/out for multi-chip operation

Sparse global connectivity b/w tiles for PDEs;
Dense local connectivity b/w function units for
variety of nonlinear functions, derivatives



Prototype analog accelerator for nonlinear systems of equations

Tiled programmable analog accelerator

Enhanced calibration for all analog units

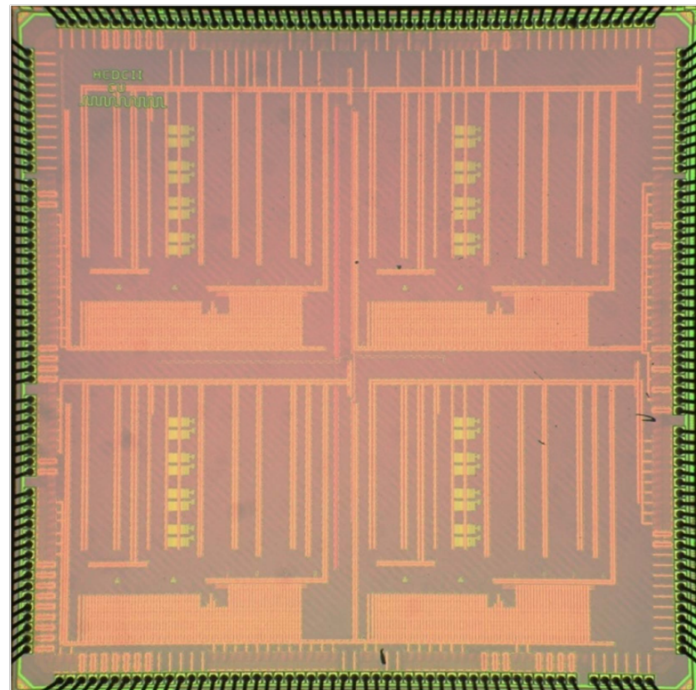
Analog in/out for multi-chip operation

Sparse global connectivity b/w tiles for PDEs;
Dense local connectivity b/w function units for
variety of nonlinear functions, derivatives

Solve 2D Burgers' equation with 2 chips

x-velocity field in one chip

y-velocity field in the other



Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

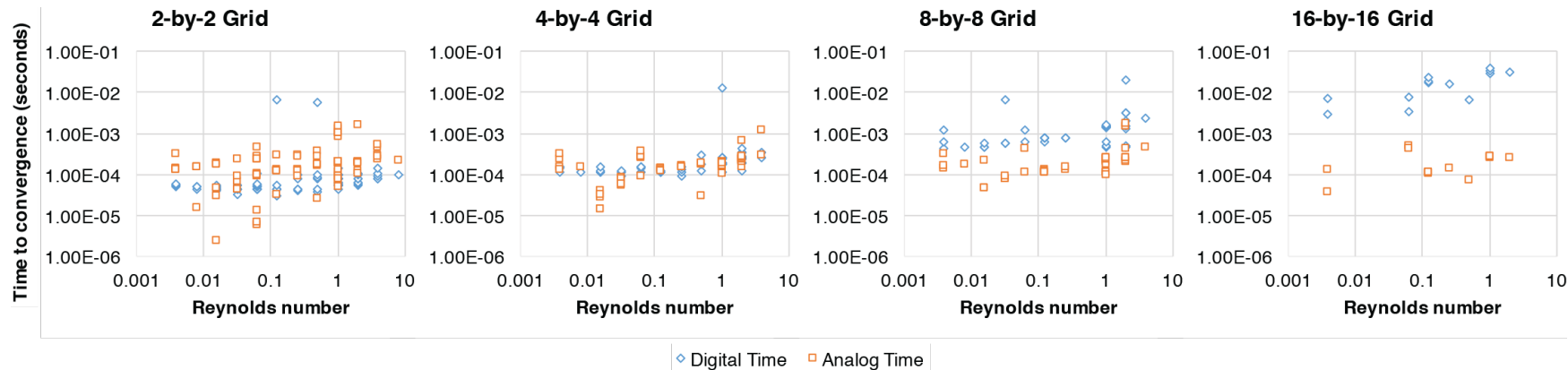
Hybrid analog-digital system combines both for speed & precision
How to map a PDE problem to a prototype analog accelerator

Evaluation:

100× faster for approximate solution

5.7× faster and 11.6× less energy when analog helps GPU

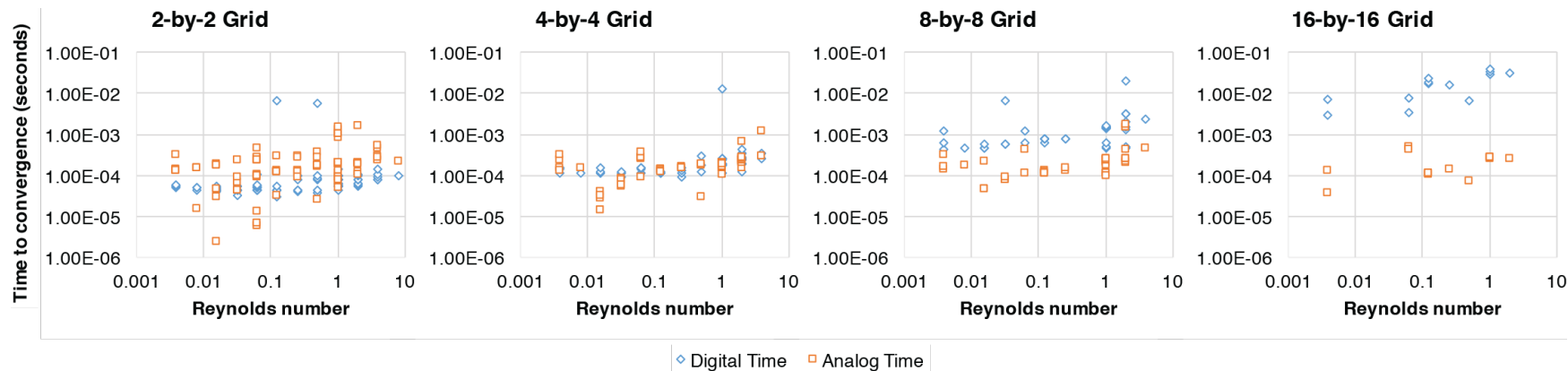
Performance simulation of scaled-up analog accelerators



Analog accelerator performance vs. digital for larger problem sizes

Plot of solution time to same accuracy vs. how nonlinear problem is

Performance simulation of scaled-up analog accelerators



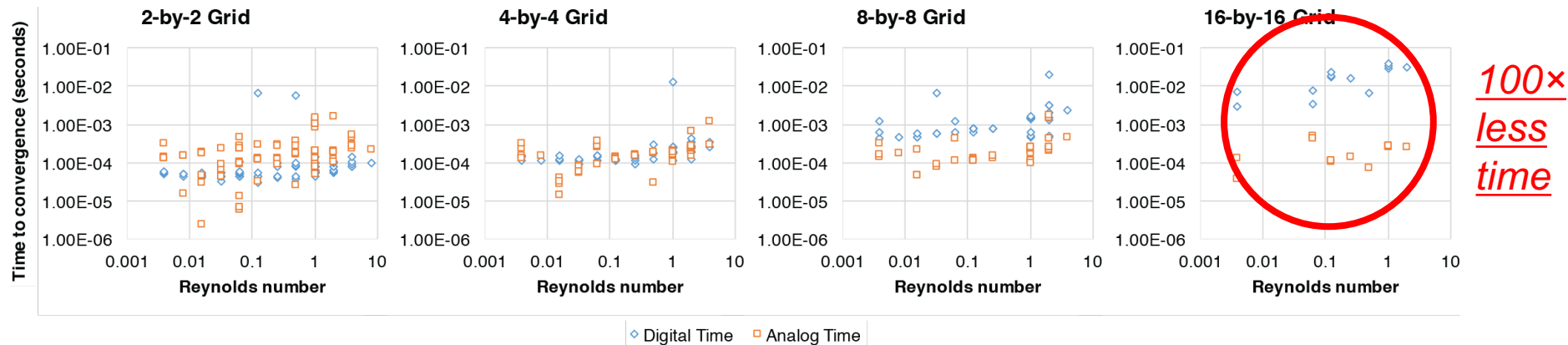
Analog accelerator performance vs. digital for larger problem sizes

Plot of solution time to same accuracy vs. how nonlinear problem is

Analog: solution time for 2-by-2 grid validated using prototype chip

Digital: solution time from damped Newton's method solver

Performance simulation of scaled-up analog accelerators



Analog accelerator performance vs. digital for larger problem sizes

Plot of solution time to same accuracy vs. how nonlinear problem is

Analog: solution time for 2-by-2 grid validated using prototype chip

Digital: solution time from damped Newton's method solver

Outline

Motivation:

Analog avoids digital pitfalls in nonlinear problems

Architecture:

Hybrid analog-digital system combines both for speed & precision
How to map a PDE problem to a prototype analog accelerator

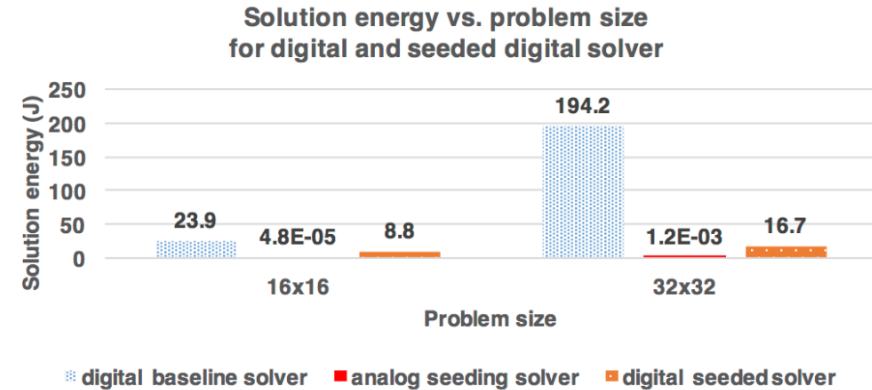
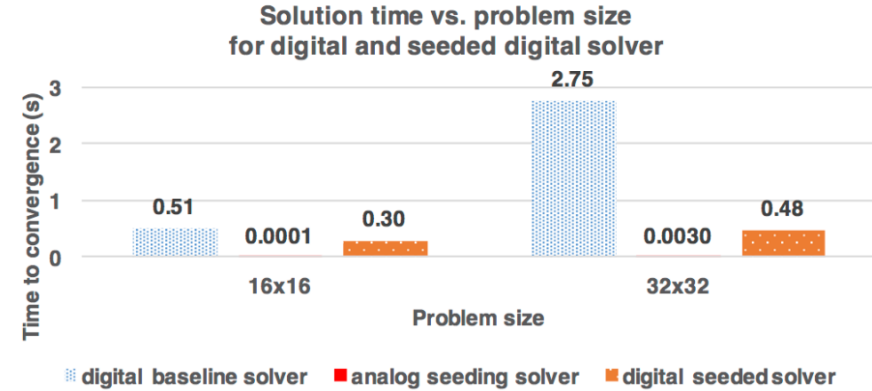
Evaluation:

100× faster for approximate solution

5.7× faster and 11.6× less energy when analog helps GPU

Using cheap analog approximation to reduce GPU solution time & energy

Plot of solution time & energy vs. problem size
32x32 2D problem has 2048 variables at play



Using cheap analog approximation to reduce GPU solution time & energy

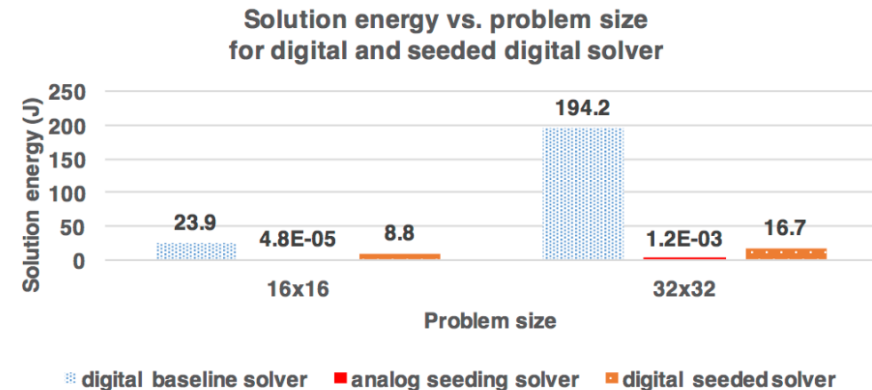
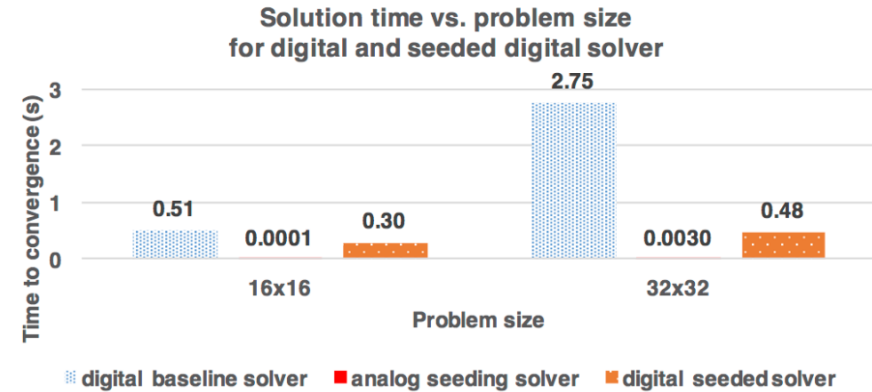
Plot of solution time & energy vs. problem size
32x32 2D problem has 2048 variables at play

Comparing:

GPU alone to high precision

Analog approximation

GPU with analog approx. to high precision



Using cheap analog approximation to reduce GPU solution time & energy

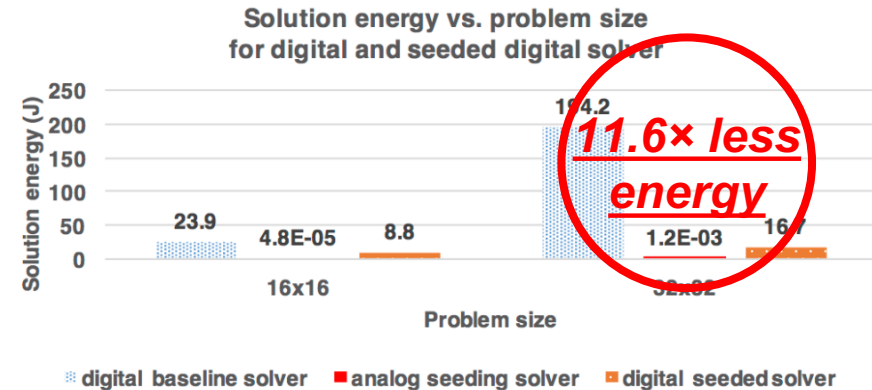
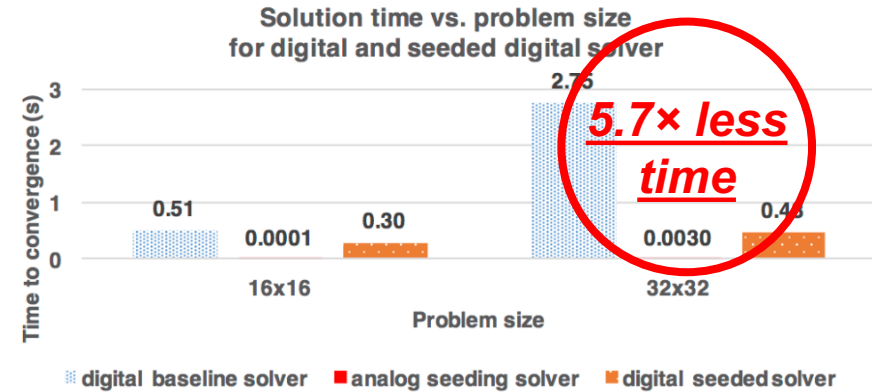
Plot of solution time & energy vs. problem size
32x32 2D problem has 2048 variables at play

Comparing:

GPU alone to high precision

Analog approximation

GPU with analog approx. to high precision



Using cheap analog approximation to reduce GPU solution time & energy

Plot of solution time & energy vs. problem size
32x32 2D problem has 2048 variables at play

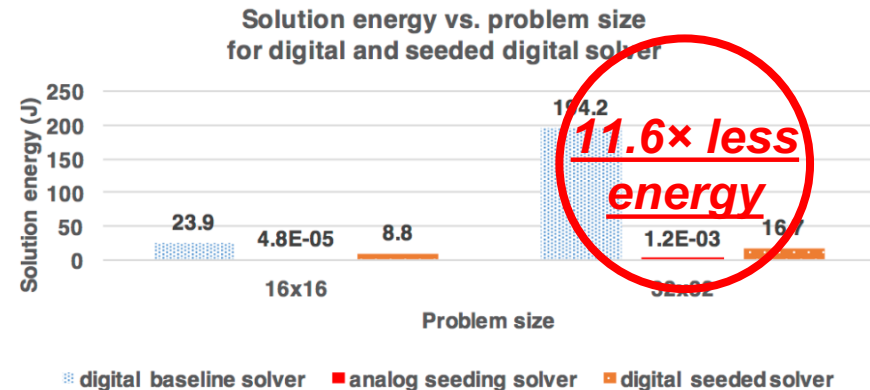
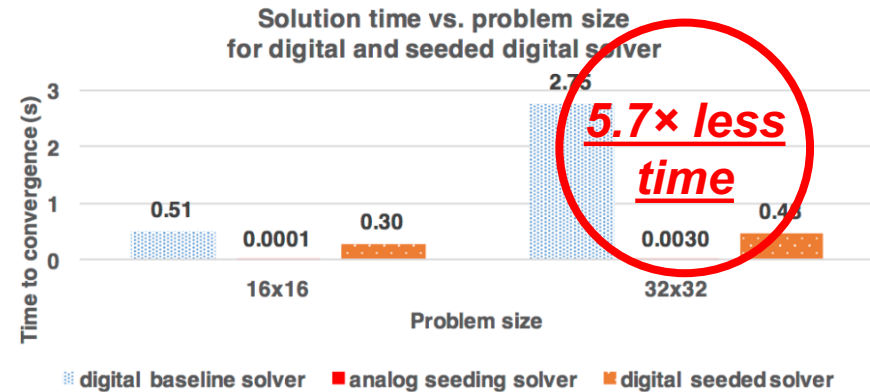
Comparing:

GPU alone to high precision

Analog approximation

GPU with analog approx. to high precision

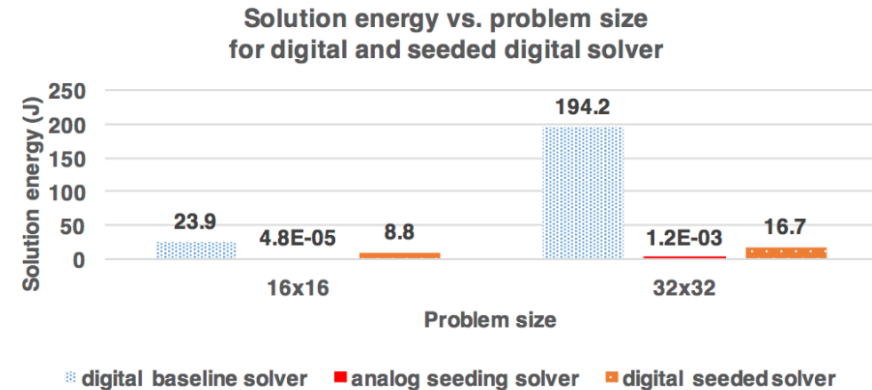
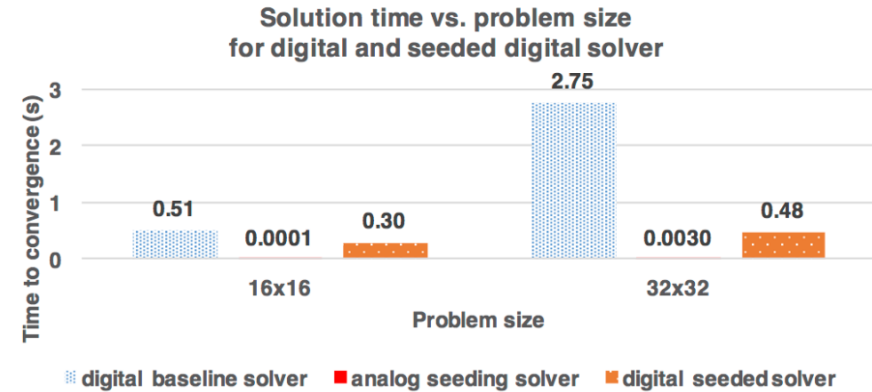
Time and energy saved here is significant
PDE solver workloads repeatedly call these innermost loops as dominant kernel



Using cheap analog approximation to reduce GPU solution time & energy

Scaling out beyond analog size constraint?
16-by-16 chip uses 350mm², 400mW

Takeaway 4 of 4:
how to increase problem size?
Extremely low power analog can scale out and stack differently than digital



Nonlinear is analog killer app!

Nonlinear is analog killer app! Why it works:

How to do more work in analog?

Create analog inner loops by nesting circuits with different convergence

How to increase solution accuracy?

Cheap analog approximation → good digital initial guess

How to minimize reprogramming?

Retarget kernels in existing digital code to accelerate in analog

How to increase problem size?

Extremely low power analog can scale out and stack differently than digital