Analog Computing for Linear Systems

Yipeng Huang

December 15, 2015

A Simple Linear ODE and Digital Solution…

$$\frac{\mathrm{d}\vec{u}}{\mathrm{d}t} = A\vec{u}(t) + \vec{b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} u_0(t) \\ u_1(t) \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

With initial conditions:

$$\vec{u}(0) = \vec{u_{init}}$$

A Simple Linear ODE and Digital Solution…

$$\frac{d\vec{u}}{dt} = A\vec{u}(t) + \vec{b}$$

$$\frac{d}{dt}\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} u_0(t) \\ u_1(t) \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

With initial conditions:

$$\vec{u}(0) = \vec{u_{init}}$$

```
double time = 1.0;
int steps = 100;
double step_size = time / (double)steps;
for (int step=0; step<steps; step++) {
    delta_0 = b_0 + A_00 * u_curr0 + A_01 * u_curr1;
    delta_0 *= step_size;
    u_curr_0 += delta_0;
    delta_1 = b_1 + A_10 * u_curr0 + A_11 * u_curr1;
    delta_1 *= step_size;
    u_curr_1 += delta_1;
}
```

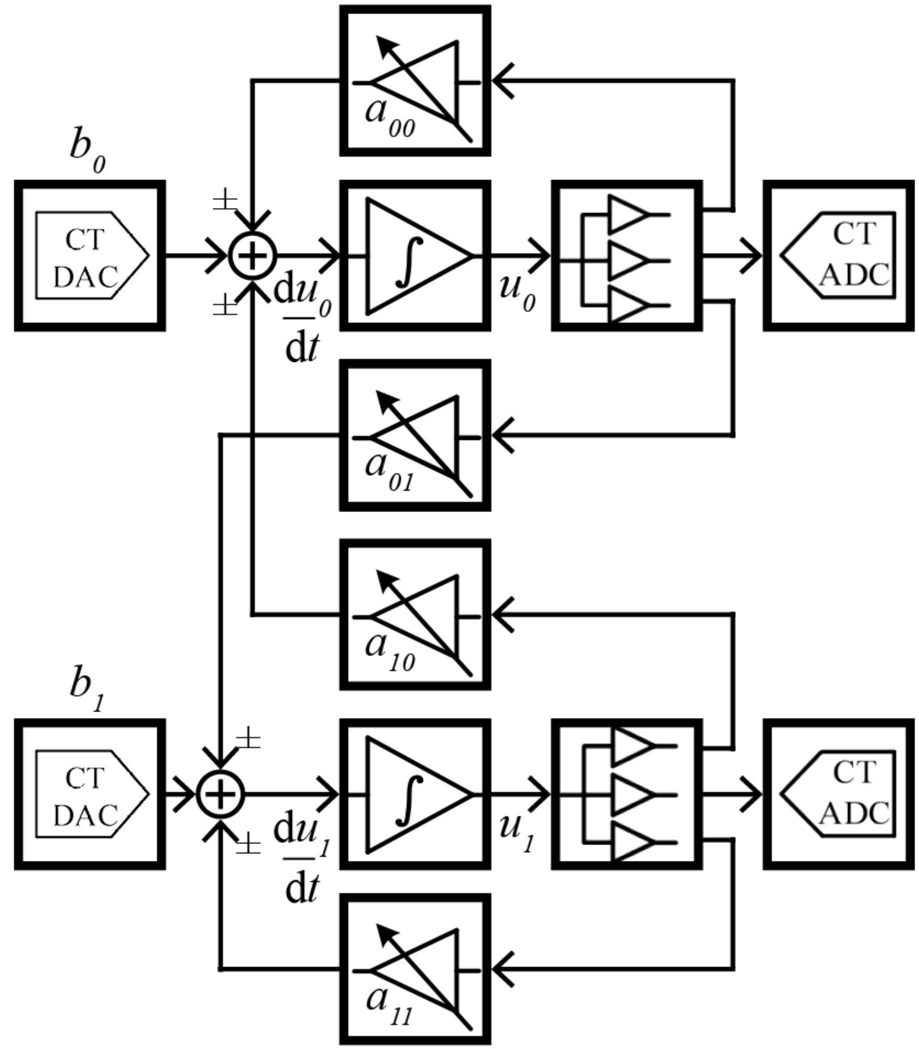
…and Equivalent Analog Solution

$$\frac{d\vec{u}}{dt} = A\vec{u}(t) + \vec{b}$$

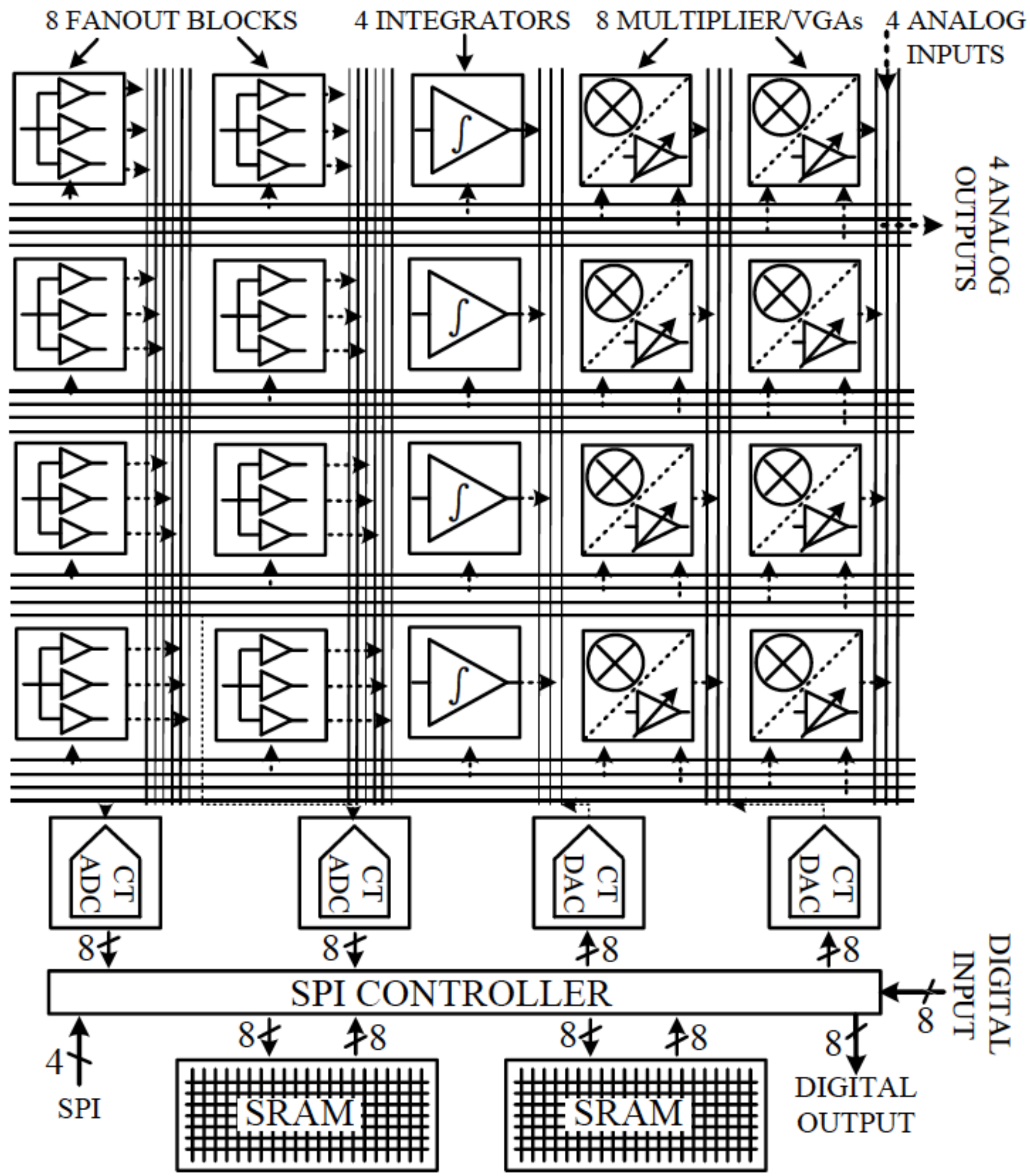
$$\frac{d}{dt}\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} u_0(t) \\ u_1(t) \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

With initial conditions:

$$\vec{u}(0) = u_{\vec{init}}$$

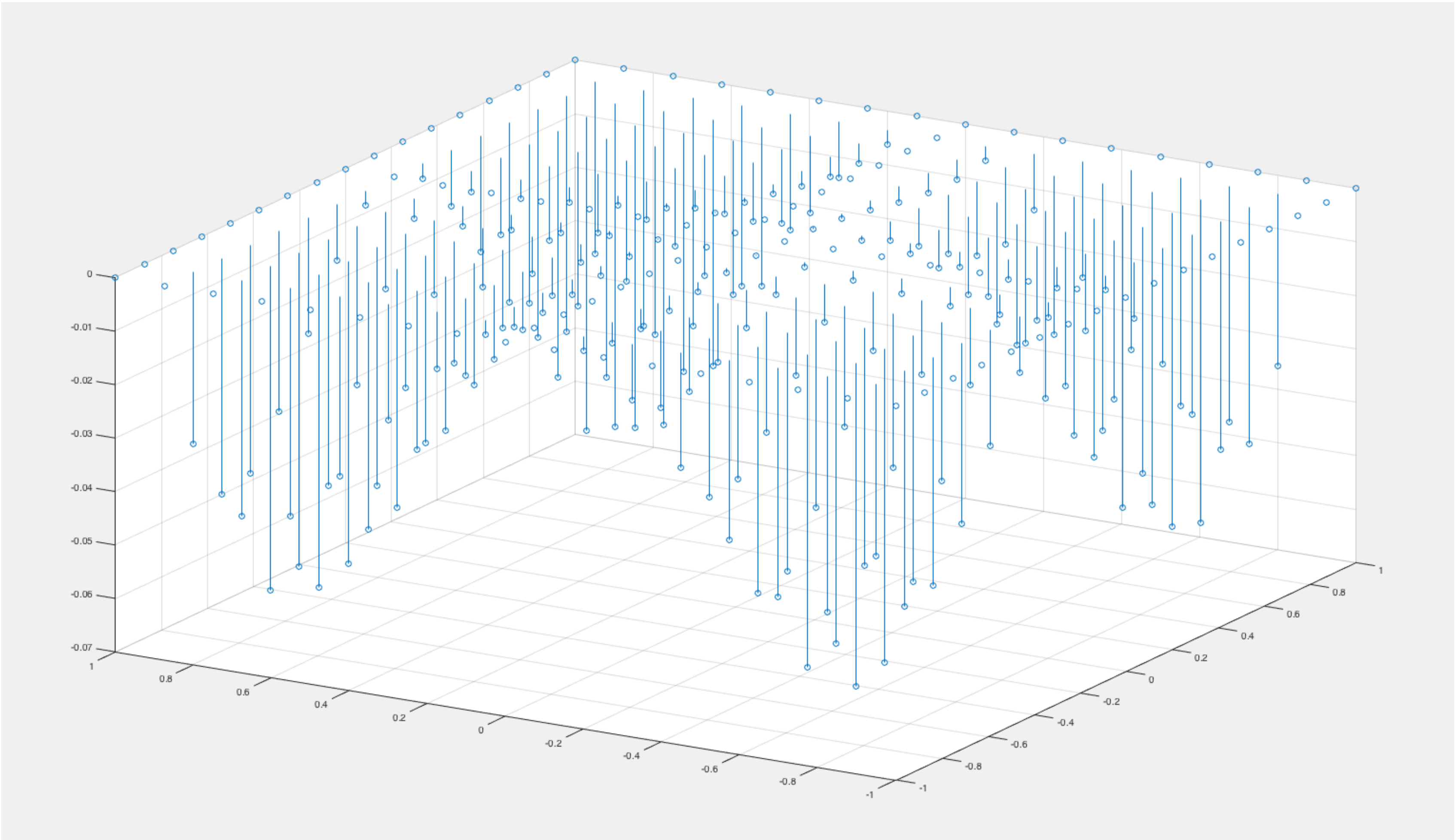


Physical Hardware



A Simple Linear PDE

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = b(x, y)$$



A Simple Linear PDE

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = b(x,y)$$

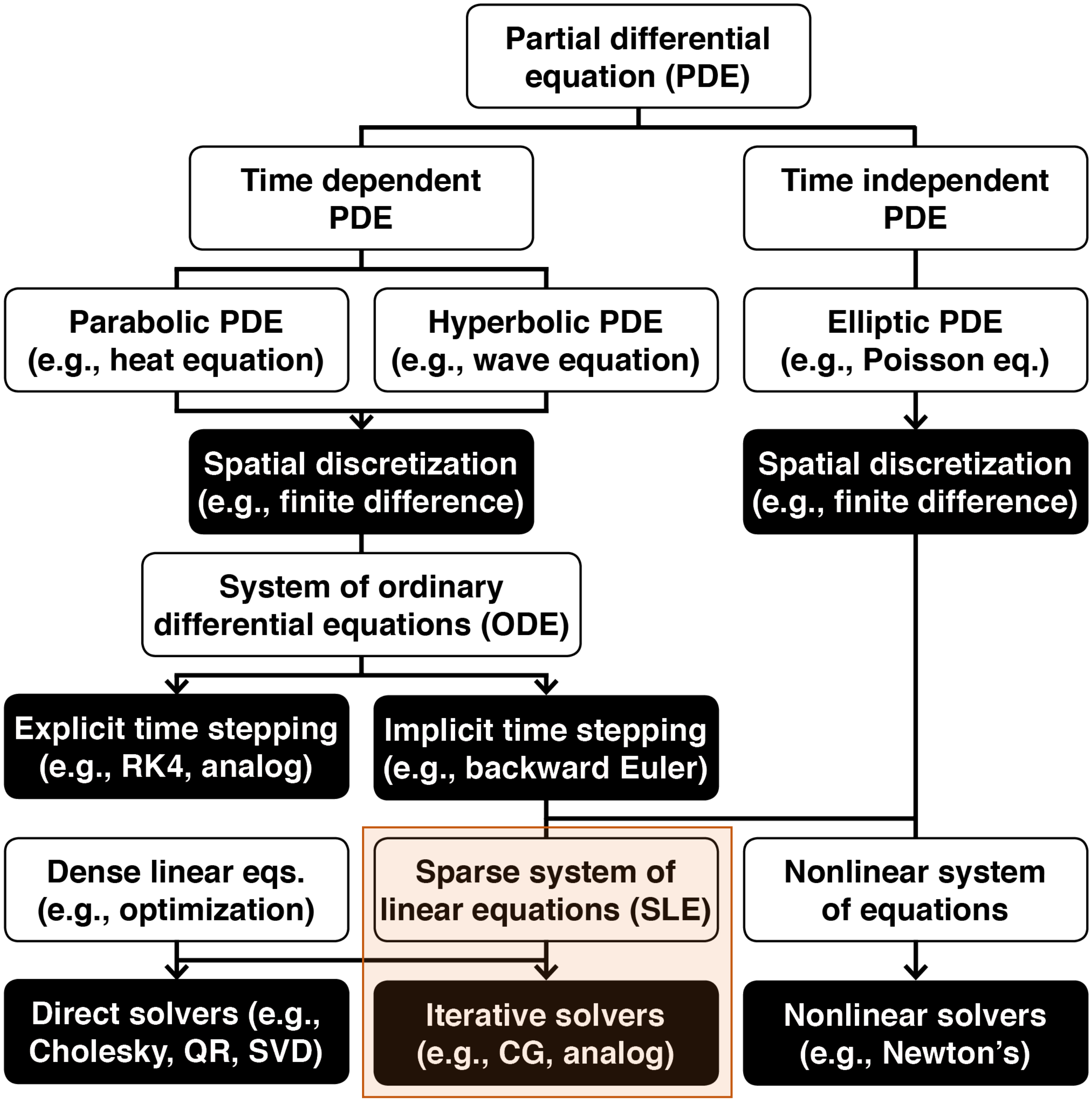
u_0	u_1	u_2
u_3	u_4	u_5
u_6	u_7	u_8

$$A\mathbf{u} = \mathbf{b}$$

$$A = \frac{1}{\frac{1}{3^2}} \begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}$$

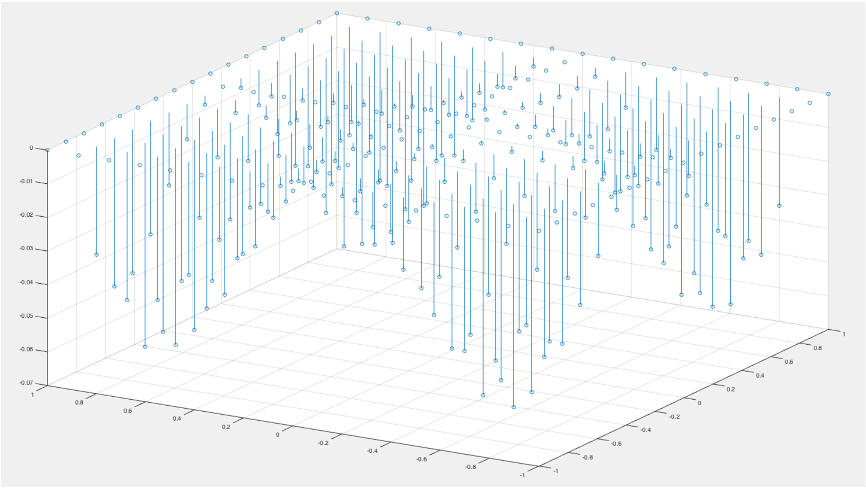
$$\mathbf{u} = [u_0, u_1, \ldots, u_8]^\top, \mathbf{b} = [b_0, b_1, \ldots, b_8]^\top$$

Taxonomy of PDE / ODE, Grid / Stencil Problems



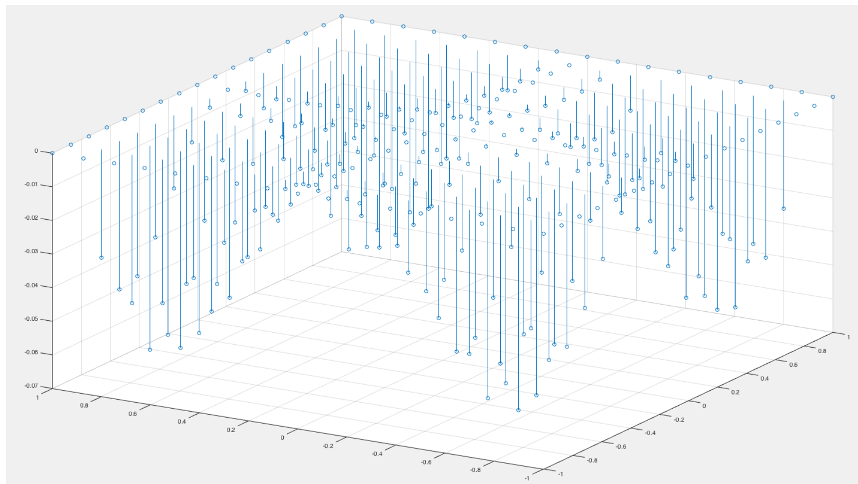
Overview of Approach and Findings

- **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



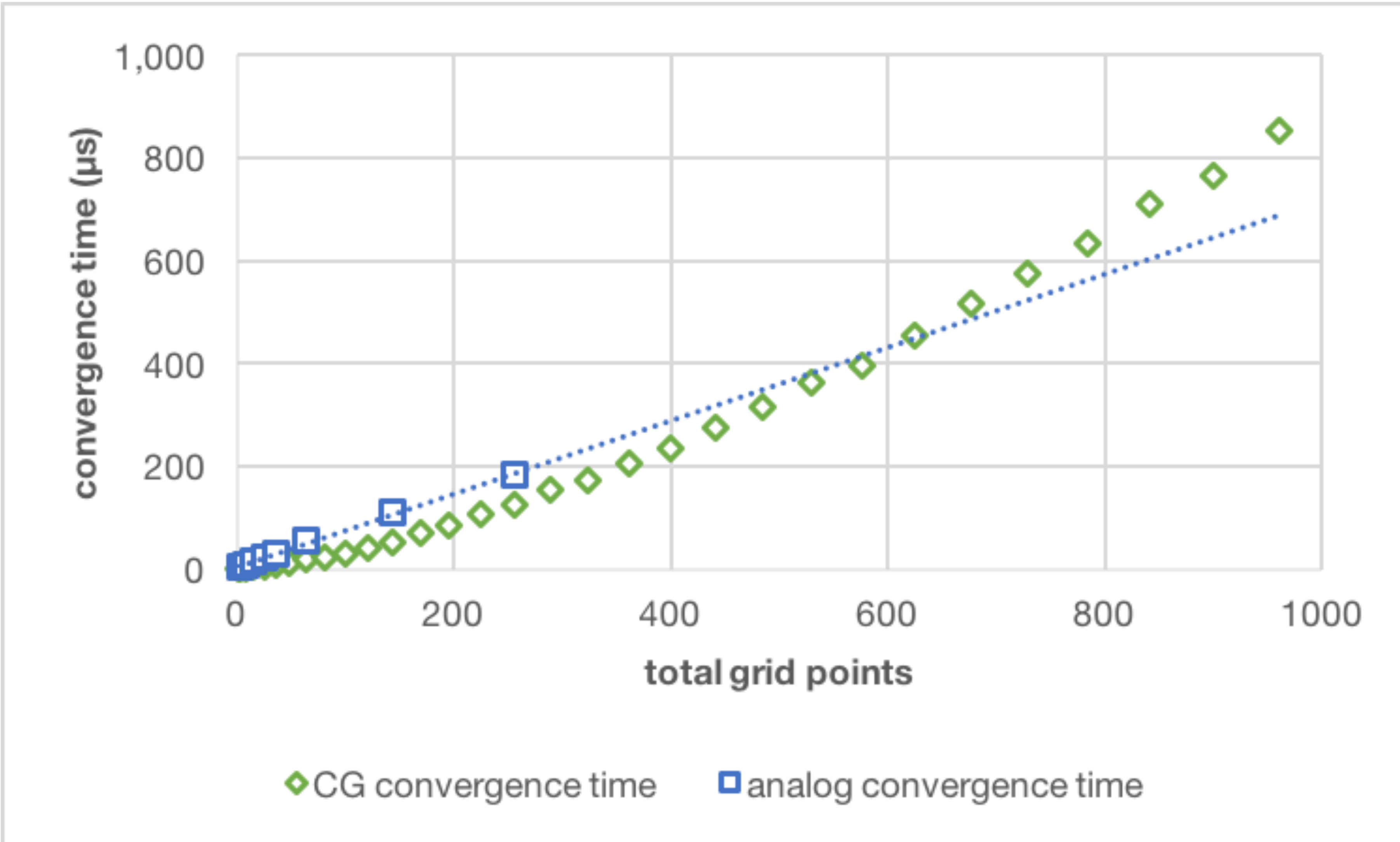
Overview of Approach and Findings

- **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

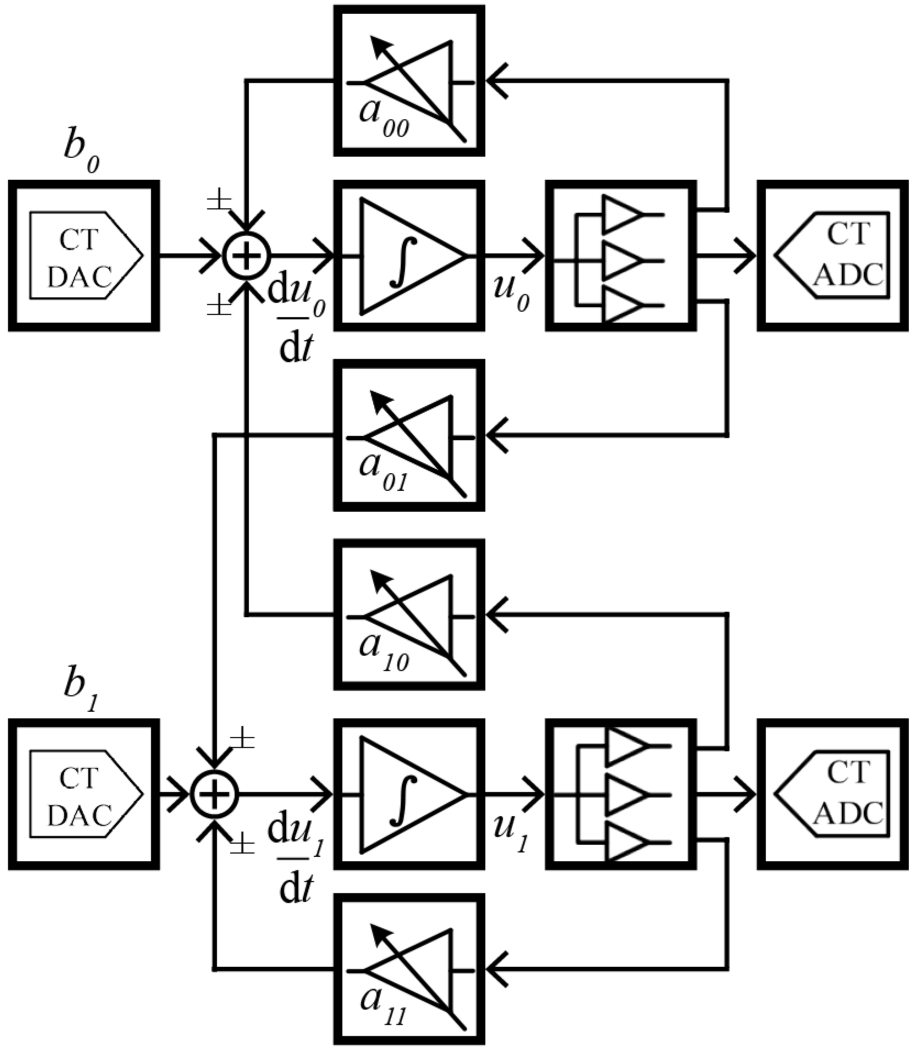


- **Findings**
  - Analog is fast, but not game-changing; Digital has advantage in CG algorithm
  - Analog area consumption is high
- **What it needs to succeed**
  - Optimum gradient method, discussed in in Karplus and Korn
  - I’ll do in depth experiments using HCDC v2
  - Vary problem size (up to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

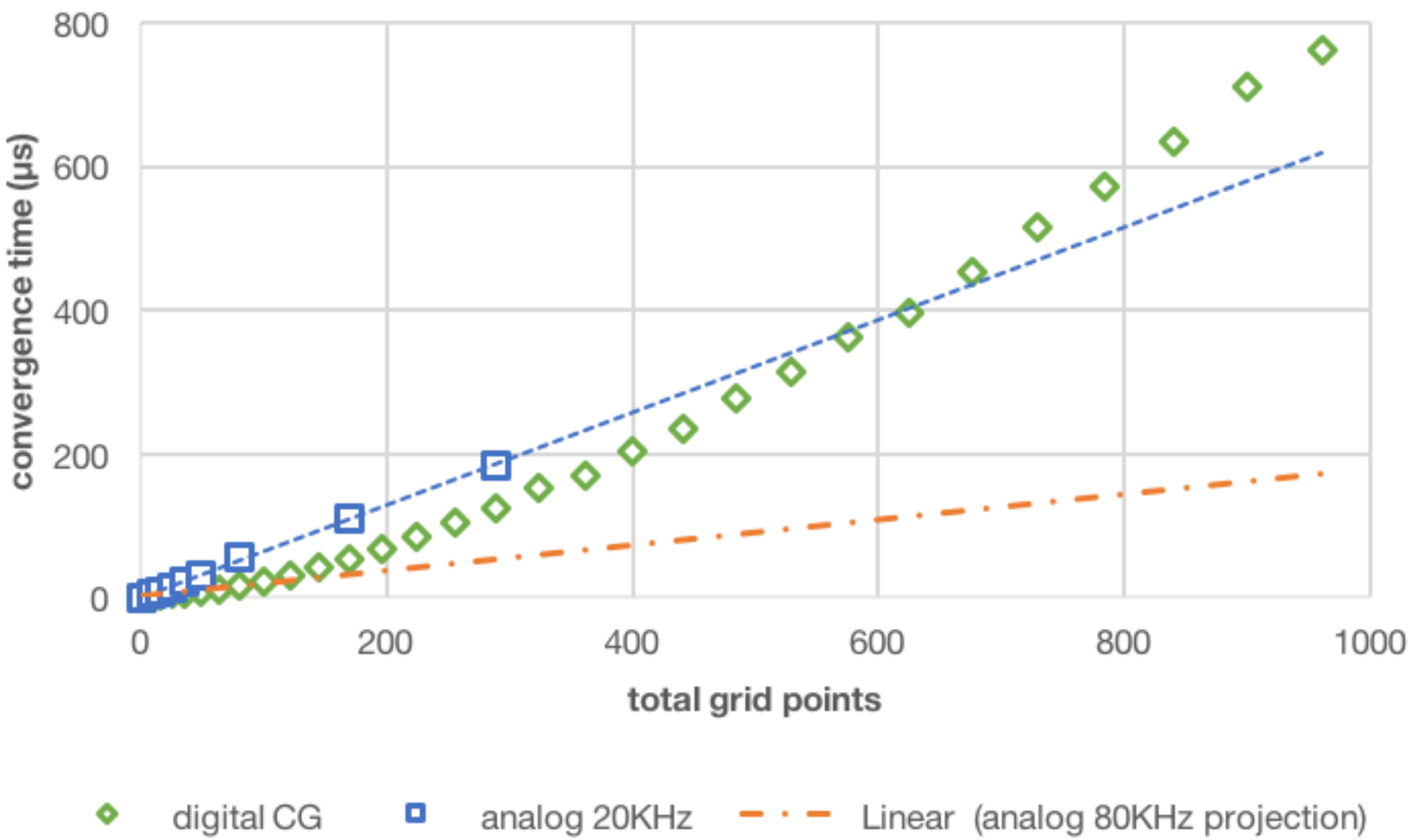
Performance Comparison



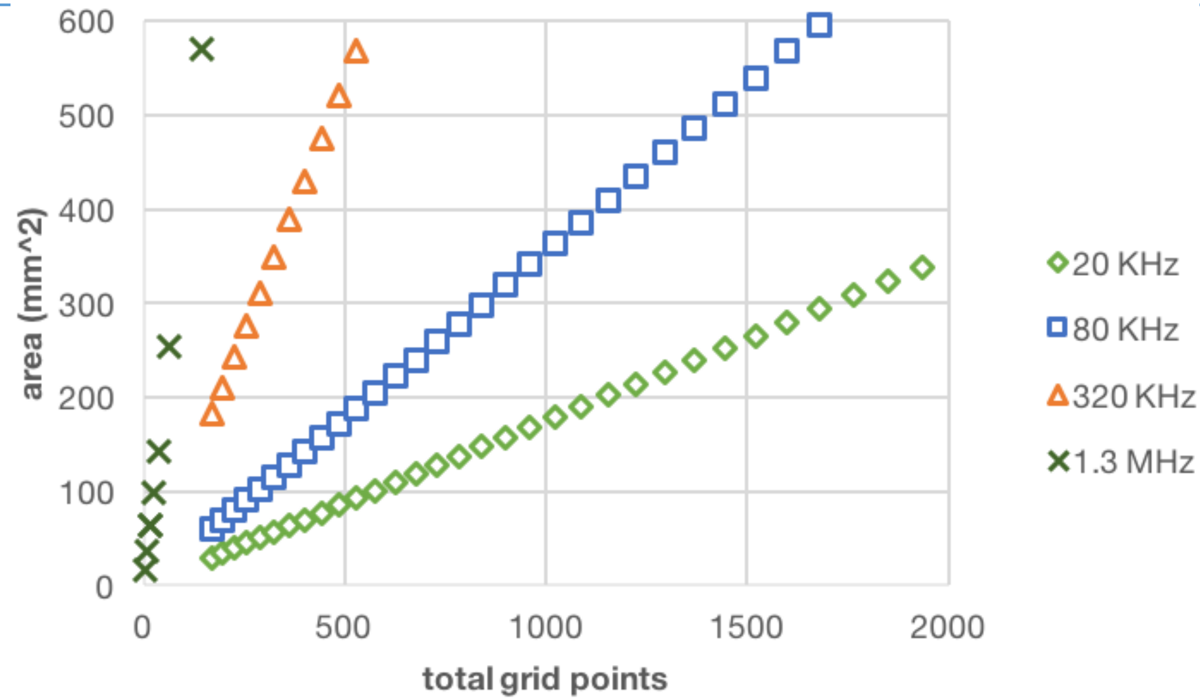
Analog Computer Bandwidth

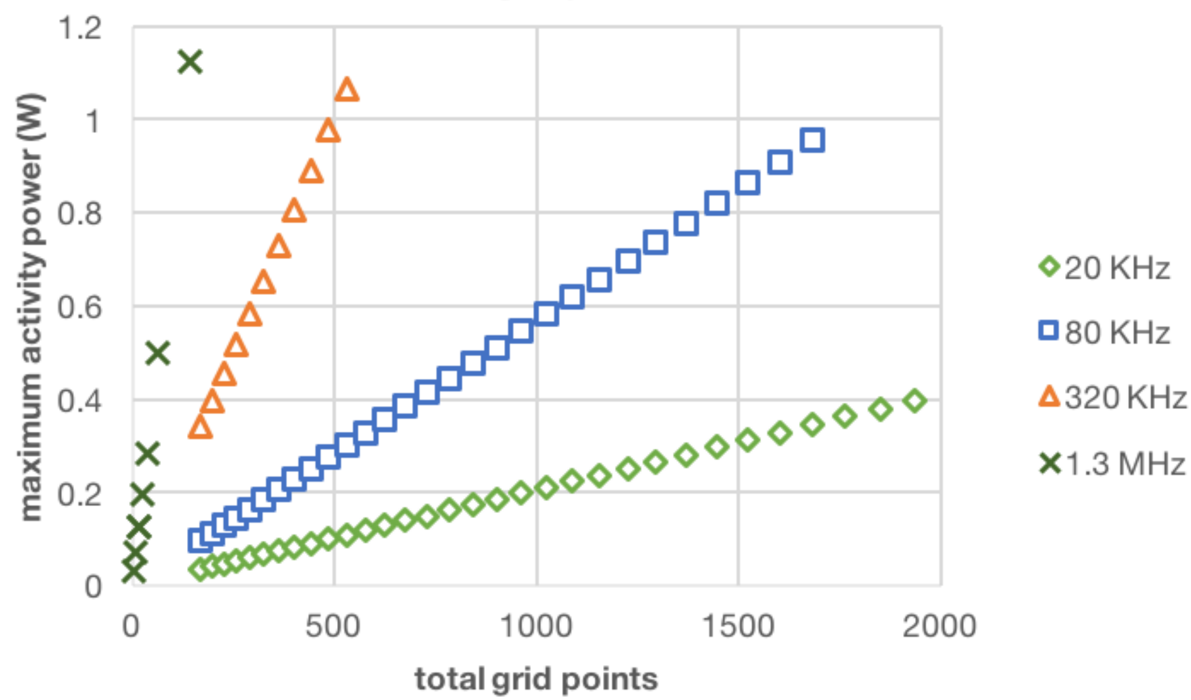


Performance Comparison

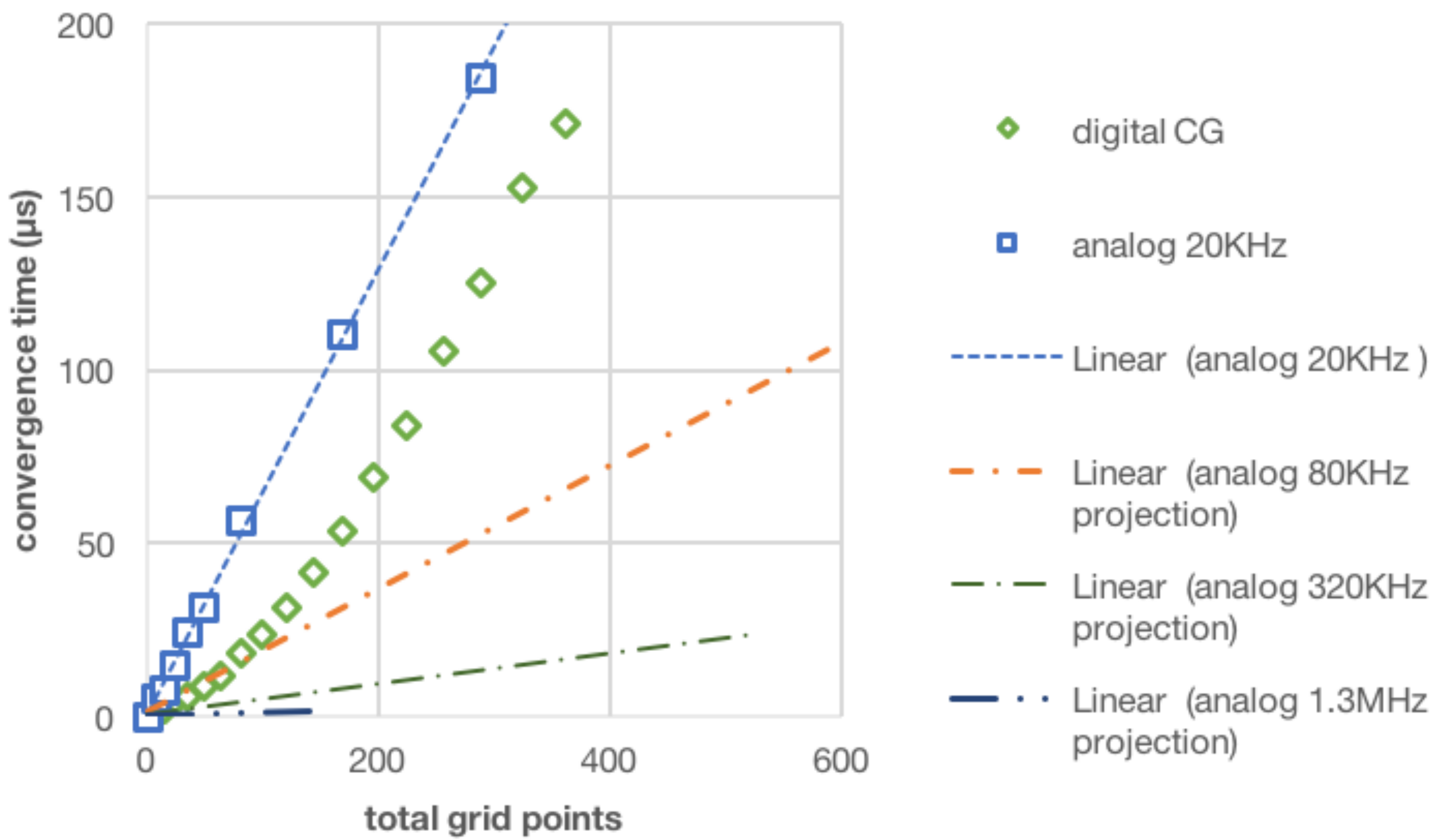


Area and Power

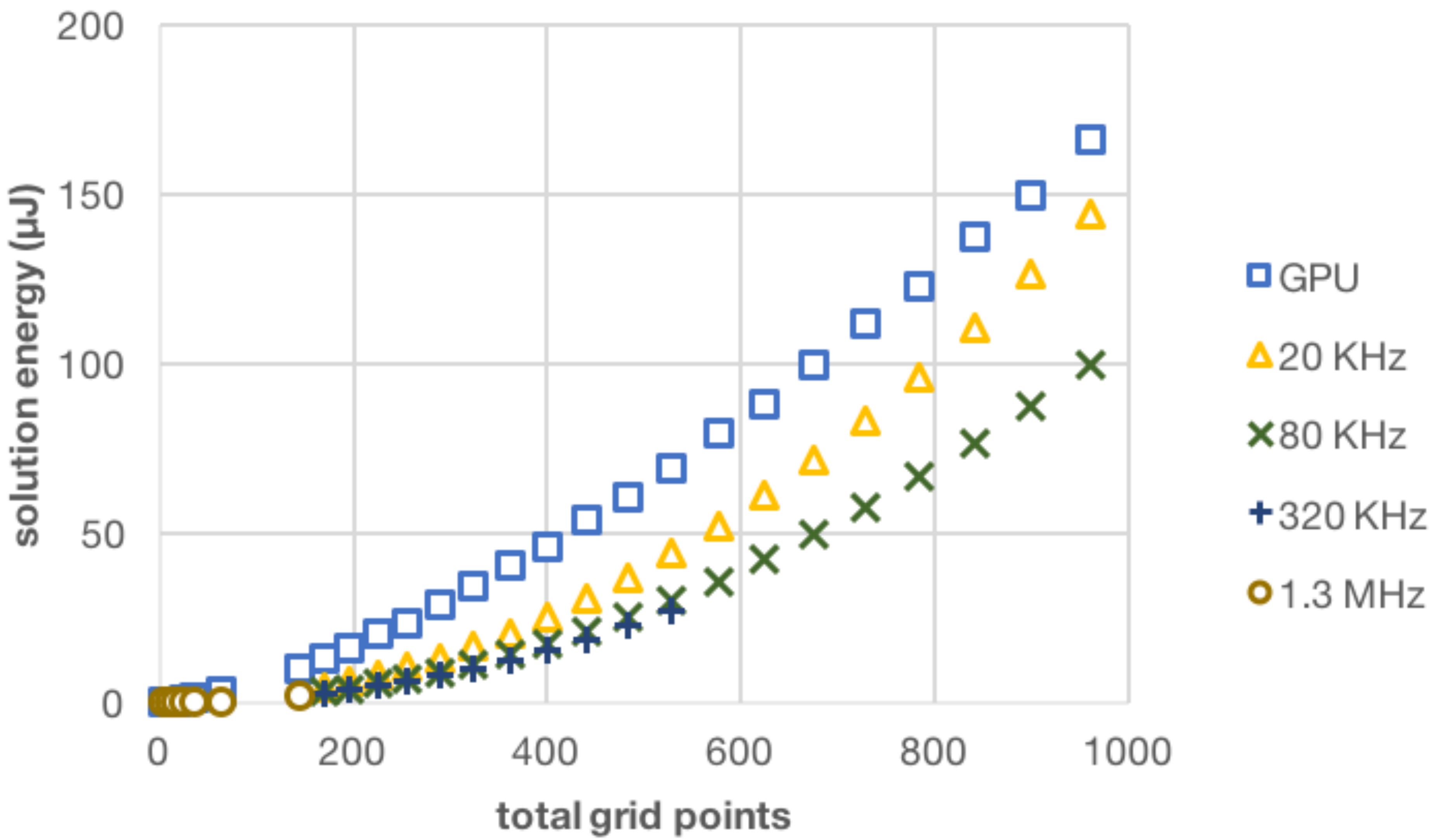




Design Space Exploration



Energy


200
150
100
50
0
0
200
400
600
800
1000
solution energy (µJ)
total grid points
GPU
20 KHz
80 KHz
320 KHz
1.3 MHz

Next Directions for Analog

- **Dense linear algebra**
- **Nonlinear systems of equations**
- **Stochastic simulation**