HCDC: A Top-Down Perspective

Yipeng Huang
May 2, 2014
Outline

• HCDC & computational robotics
  – Speeding up inverse kinematics using hardware

• HCDC & smooth optimization
  – Solving linear programming using gradient descent

• HCDC & solving linear equations
  – Inverting matrices using gradient descent on analog circuits

• HCDC & solving ordinary differential equations
  – Putting the integrators to their best use

• HCDC host system
  – Performance evaluation
  – Digital microcontroller interface
Where to Look for HCDC Applications

• Use HCDC analog-digital chip to accelerate applications
  – Better than digital alone
  – Pick up where analog computers of 1960’s left off

• Look for problems that digital computers struggle at
  – Applications that have a continuous core problem
  – Emergent, computationally intensive programs that deal with real world
  – Robotics, sensor, and actuator programs

• Tackle problems that didn’t exist / impossible in 1960’s
  – Return to the classical analog programs: simulations, optimizations
  – How digital computing can assist where analog failed
A Core Robot Algorithm: Inverse Kinematics

- How to control a robot’s joints to achieve desired pose
  - Input: current robot geometry
  - Output: required joint increments

- Computationally intensive problem all limbed robots must solve

- Beyond controlling single arms and legs, many larger problems rely on inverse kinematics
  - redundant manipulators
  - multiple end effectors
  - inverse dynamics
A Digital Accelerator for Inverse Kinematics

• Inverse kinematics not well suited for normal digital architectures
  – Entirely floating point array, matrix operations
  – 40% of cycles in inverting matrices
  – 15% of cycles in sine, cosine operations

• We’ve created an accelerator to solve IK via damped least squares
  – Dedicated sine, cosine function generators
  – Parallel, fixed-point functional units
  – Solves IK problem in 4µs: compare against 10ms for general algorithm on CPU

• At coarse level, future HCDC chips may include similar accelerators
  – Include more core robotics algorithms
  – Map portions to adjacent analog circuitry
A Hierarchy of Optimization Problems

- Problem statement:
  - Optimize a utility function
  - Given a set of resources
  - Where each resource is subject to constraints

- Mixed integer non-linear programming
- Continuous variables: non-linear programming
- Convex utility function: convex programming
- Quadratic utility: quadratic programming
- Linearity of all variables: linear programming
Smooth Optimization: Linear Programming

- General form
  Maximize $z = c^T x$
  Subject to constraints $Ax \leq b$

- Example
  Maximize $z = 2x_1 - 3x_2 + 3x_2$
  Subject to constraints
    - $x_1 + 2x_2 - 2x_3 \leq 4$
    - $x_1 + x_2 - x_3 \leq 7$
    - $-x_1 - x_2 + x_3 \leq -7$
    - $x_1, x_1, x_1 \geq 0$

- All linear programming problems can transform to this form
  - Write as maximization problem
  - Ensure bounded constraints
Interior Point Method for Linear Programs

- Transform problem space so current point is “centered”
  - If you’re already near boundary, you might not reach global maximum

- Take a step in the direction that increases utility most
  - Intuitively, you should consume the resources that result in more utility

- Return solutions back to original space
  - Update the slack variables; how much of each resource do I have left?

- Iterate

- HCDC speeds this up: rapidly taking infinitely small steps
Considerations on Mapping to HCDC

• HCDC version of interior point method needs $|x| + |v|$ integrators
  – But we only have four, more if we team up chips
  – People already routinely solve linear programming problems with ~100K variables

• Try cyclic or block coordinate-descent decomposition
  – In contrast to the interior point method, a conjugate gradient descent
  – The literature on analog computing hints at using coordinate descent optimization

• Tackle non-convex optimization, which digital is slow at
  – Branch and bound method: divide and conquer a non-convex exploration space
  – Digital computers cannot tackle problems larger than ~100 variables
  – Use HCDC to accelerate the underlying linear program solver
Solving Differential Equations Using HCDC

• In addition to the proposed applications, we have already demonstrated solving ordinary differential equations
  – As part of verification and validation of chip before tapeout

• Equation tests: check solving whole equations
  – 2\textsuperscript{nd} order linear ODE
  – 2\textsuperscript{nd} order nonlinear ODE
  – 2\textsuperscript{nd} order transcendental ODE
    • Different datapaths for same equation
    • Different gain and range settings for same equation

• >40 unique chip tests
Performance Model

• Timing: we can now accurately model timing overhead costs of analog vs. digital computation
  – I will talk about this

• Power: we will rely on physical chip to measure power cost of analog vs. digital computation
  – Vastly improve previous estimates about analog efficiencies

• Area: we are building intuition on how well HCDC may scale
  – Ability to combine more integrators; communication costs

• Accuracy: we need to quantitatively measure error
  – Would analog excel at stiff, unstable, chaotic equations?
Timing: Startup & Calibration

• Conservative assumptions: 5MHz SPI clock
  – resulting in 208KHz HCDC controller clock

• Upon startup, all configuration bits have to be written zero
  – One time cost of 4ms

• Configuration portion of initial calibration would take 7ms
  – Assuming all tunable parameters must be tuned using binary search

• Each configuration takes 3.7ms in the worst case
  – Average case ~2ms for real equations
  – More aggressive SPI clock decreases this time linearly
  – Important for enabling time multiplexing on chip
HCDC Host System

- Arduino microcontroller
  - Bridge between computer and HCDC chip
  - Programmable through Arduino’s software tools
  - Native digital serial interface and software controlled serial interface
  - 10-bit ADCs
  - Memory for stored programs