Formal Identification of DC Operating Points in Integrated Circuits and some Lessons in (Ir)Reproducible Research in Computational Math

Ian Mitchell
Department of Computer Science
University of British Columbia

Joint work with
Mohammed Zaki & Mark Greenstreet

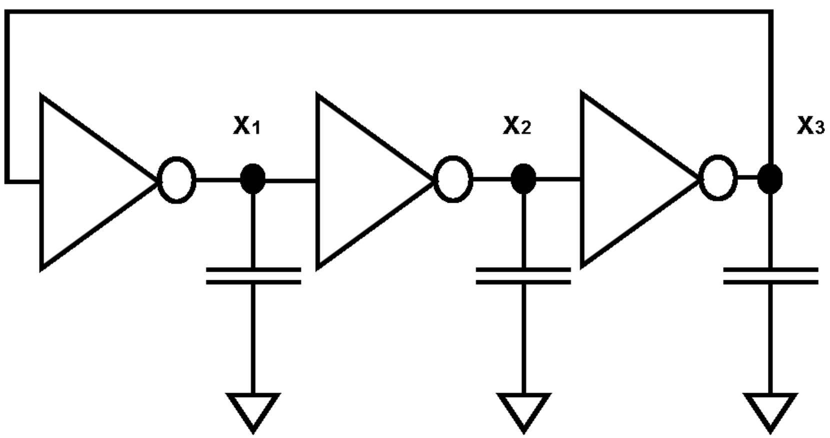
research supported by
NSERC & Intel

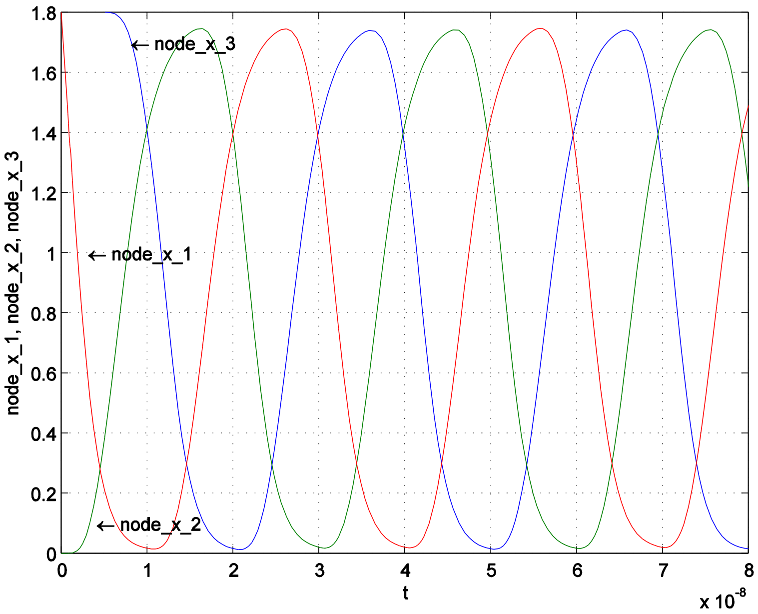
DC Operating Point(s) of a Circuit

- State (node voltages) towards which the system will settle when any inputs are held constant
  - Small perturbations of state will decay
- Inverter: A single known state
  - Output is high if input is low or vice versa
- Oscillator: No DC operating point(s)
  - Oscillation requires node voltages to keep changing
- D flip-flop: Depends on clock
  - Transparent: A single known state such that output matches D input
  - Latched: two possible states, depending on last D input
- Typical applications
  - Initial conditions for transient simulation
  - Linearization point for small signal analysis
  - Determine qualitative stable behaviour of the circuit (eg: memory or oscillator lockup)

Ring Oscillators

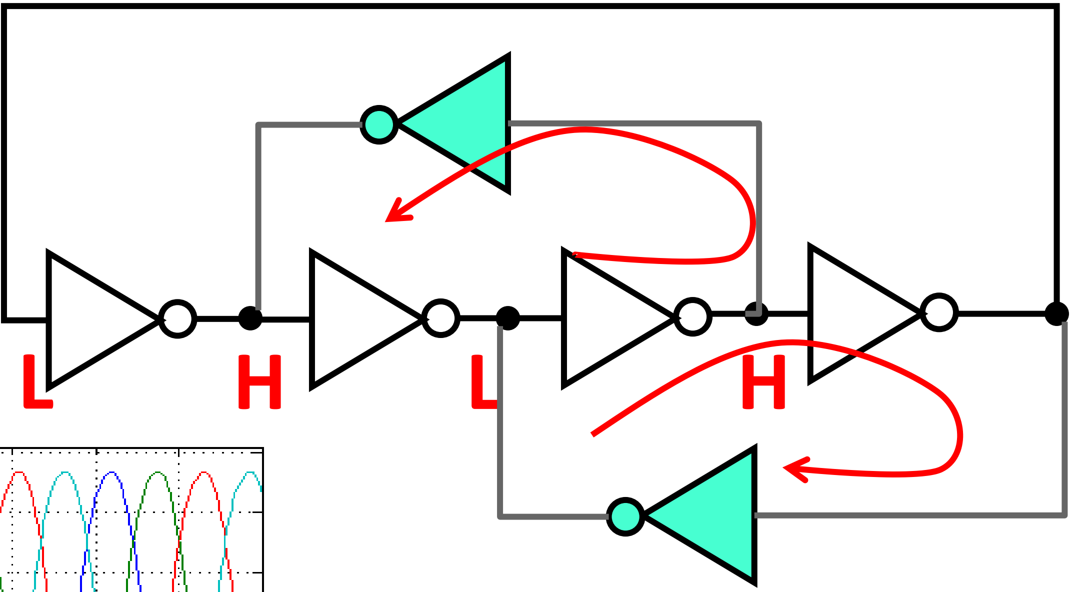
- Output oscillates between high and low values at fixed frequency
- Example implementation: Series inverters with feedback
  - No obvious stable state for odd number of inverters
  - Each inverter generates a signal with different phase

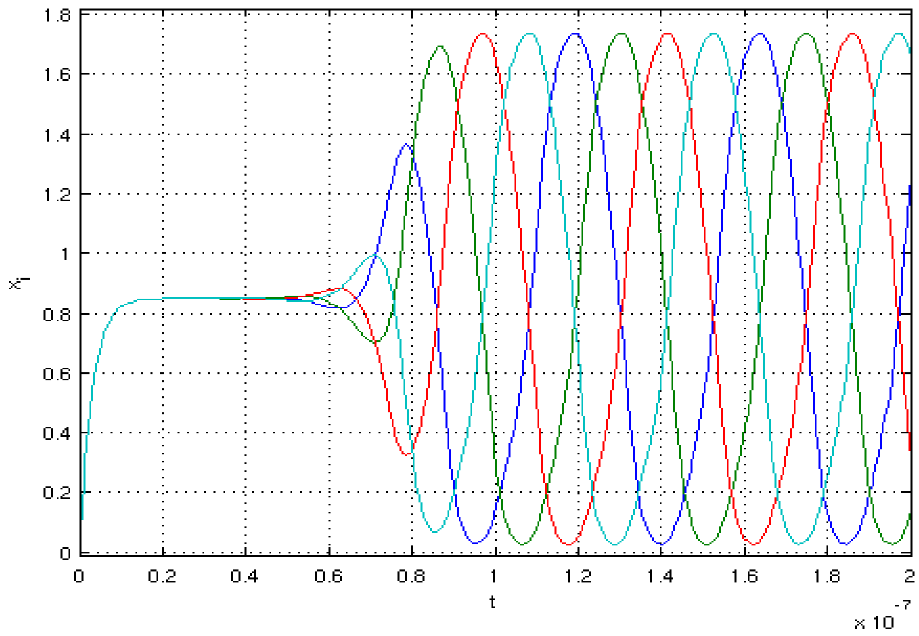




Rambus Ring Oscillator

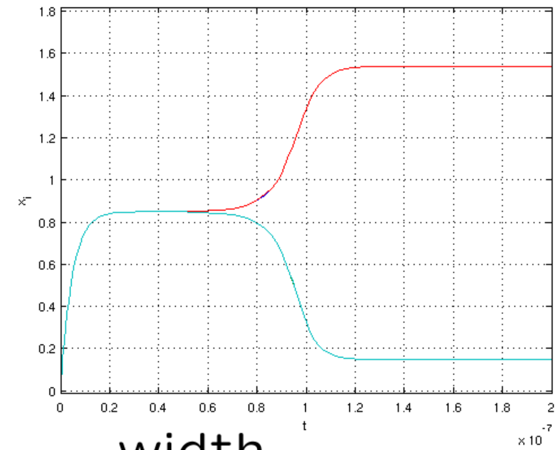
- Is it possible to generate a four phase signal?



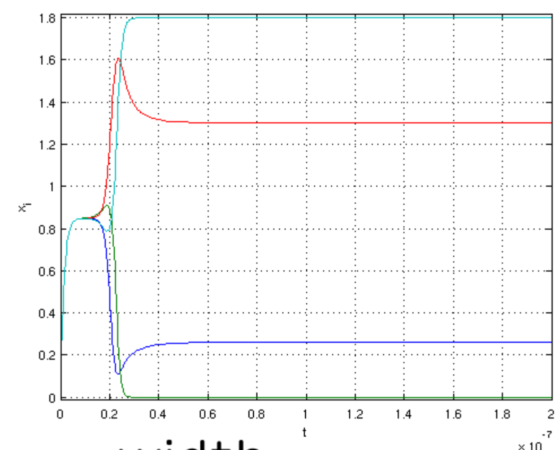


Rambus Ring Oscillator Failure Modes

- Behaviour depends critically on ratio of the widths of the ring and bridge inverters
  - Consistent oscillation if ratio is near one
  - Consistent failure to oscillate if ratio is very large or small
  - Behaviour depends on initial conditions for some ratios
- Example of actual design failure
  - Design passed typical analog validation procedure
  - Problems found during measurement of fabricated test chips



$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} \gg 1$$



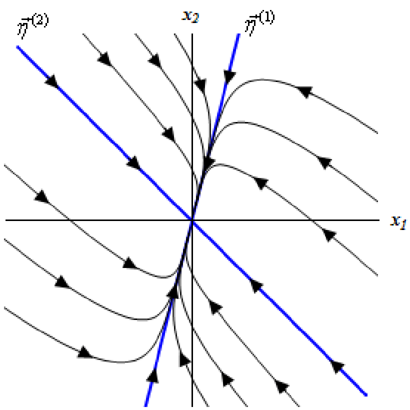
$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} \ll 1$$

Outline (Part 1)

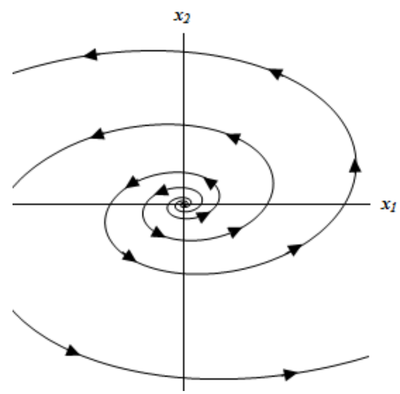
- Motivation
- Mathematical characterization of DC operating points
- Our approach
  - Circuit model construction with netlist & OOmspice / Chum
  - Analytic exclusion with symbolic model & HySAT
  - Numerical exclusion with interval model & IntLab
  - Stability analysis with pseudospectra & EigTool
- Examples
  - Schmitt trigger
  - Ring & Rambus oscillators

DC Operating Point Definition

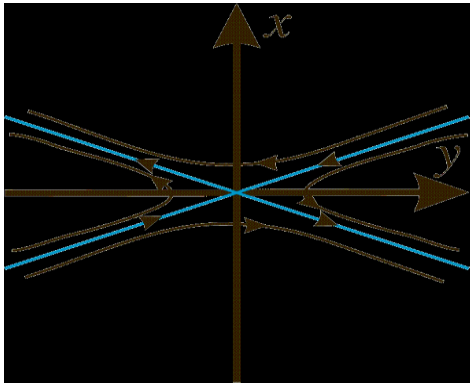
- Circuit modeled as an ordinary differential equation (ODE) with state $x(t)$: $dx(t) / dt = f(x(t))$
- Equilibrium state x_e such that: $f(x_e) = 0$
- DC operating point: Stable equilibrium state (toward which all neighbouring trajectories are attracted)
  – Stability determined by the sign of the real component of the eigenvalues $\boldsymbol{\lambda}_i$ of the Jacobian $\partial f(x_e) / \partial x$



Stable Equilibrium
(DC Operating point)
$\forall\lambda$ Re[λ] < 0



Unstable Equilibrium
$\forall\lambda$ Re[λ] > 0



Metastable Equilibrium
$\exists\lambda_1$ Re[λ_1] > 0 and
$\exists\lambda_2$ Re[λ_2] ≤ 0

Traditional DC Operating Point Analysis

- Solving non-convex, high-dimensional equation $f(x) = 0$ through traditional numerical methods is prone to failure
- Even if DC operating point is found, traditional numerical methods give no guarantee of uniqueness
- Common heuristic workarounds
  - Designer specifies the operating point
  - Transient simulation from powered off condition

Outline (Part 1)

- Motivation
- Mathematical characterization of DC operating points
- Our approach
  - Circuit model construction with netlist & OOmspice / Chum
  - Analytic exclusion with symbolic model & HySAT
  - Numerical exclusion with interval model & IntLab
  - Stability analysis with pseudospectra & EigTool
- Examples
  - Schmitt trigger
  - Ring & Rambus oscillators

Our Approach (conceptually)

- Construct an analytic ODE circuit model $dx(t) / dt = f(x(t))$
  - Requires circuit netlist and analytic transistor models
  - Accomplished by in-house OOmspice & Chum tools
- Rigourously exclude regions containing no equilibria
  - No solution to $f(x_e) = 0$
  - Restricted to bounded region of state space (eg: 0 to V_{dd})
  - First pass with unsatisfiability solver HySAT
  - Output of HySAT further refined through interval arithmetic in IntLab
- Examine system stability in regions which must or may contain equilibria
  - Construct interval Jacobian matrix $\partial f(x_e) / \partial x$ within region using IntLab
  - Examine eigenvalues of interval Jacobian through matrix pseudospectra with EigTool

Circuit Model

- OOmspice tool takes a Spice-like netlist and synthesizes ODE as either symbolic equations or a Matlab function
- Each MOSFET transistor is modeled as a three terminal (gate, source and drain), nonlinear voltage controlled current source $i_{ds} = f(v_{gs}, v_{ds})$
- Circuit nodes have capacitance (such as connected transistor gates)
  - Currently all capacitance is assumed to be constant and to ground
- KCL implies $i_{\text{transistors}} + i_{\text{capacitors}} = 0$ and standard capacitor model is $i_{\text{capacitors}} = C({dv}/{dt})$, so circuit ODE is given by

$$\frac{dv}{dt} = -C^{-1} i_{\text{transistors}}$$

where $i_{\text{transistors}}$ is constructed from i_{ds} equations for each separate transistor

Transistor Model

- Chum tool samples Hspice current data on a fine grid of gate, source and drain voltages
- Data can be fit to several different models
  - First-order global transistor model (used in FAC paper)
  - Piecewise quadratic polynomial
  - Slightly simplified global EKV model [Enz, Krummenacher & Vittoz, *Analog Integrated Circuits and Systems*, 1995]
- Global model is pre-constructed
- Local models may be constructed within specified regions or with specified terminal values (eg: drain is grounded)
- Error bounds are also generated for each model

Unsatisfiability

- HySAT attempts to prove that $f(x_e) = 0$ cannot be satisfied within a specified state space region
  - Initial region that is a box; for example, each node is between 0 and V_{dd}
  - Uses symbolic model
- Typically fails and returns a subset of the region which may contain a solution
  - Subset is dispatched for further analysis
  - New region is constructed by excluding the subset
  - HySAT is called again with new region
- Process is repeated until HySAT finds some subset of the original box which contains no equilibria

Interval Arithmetic Analysis

- Intlab evaluates Matlab function model using interval arithmetic in regions proposed by HySAT
- Three possible outcomes
  - Refutation: One or more components of derivative do not change sign, so no equilibrium is possible
  - Confirmation: Subset of region is identified which definitely contains an equilibrium
  - Inconclusive: Region may or may not contain an equilibrium
- In the latter two cases, Intlab uses automatic differentiation on the Matlab function model to generate a Jacobian interval matrix

Matrix Pseudospectrum

- Spectrum of matrix A is the set of eigenvalues of A

$$\Lambda(A) = \{z \in \mathbb{C} \mid \det(zI - A) = 0\}$$

- ϵ-pseudospectrum (ϵ-ps) of A are the set of eigenvalues of neighbouring matrices

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} \mid z \in \Lambda(A + E) \text{ for some } \|E\| \leq \epsilon\}$$

- Eigtool package uses numerical continuation to plot contours of ϵ-ps for values of $\epsilon = 10^\eta$
- For more details, see Embree & Trefethen, Pseudospectra Gateway, `http://www.comlab.ox.ac.uk/pseudospectra`

Proving Equilibrium Stability / Instability

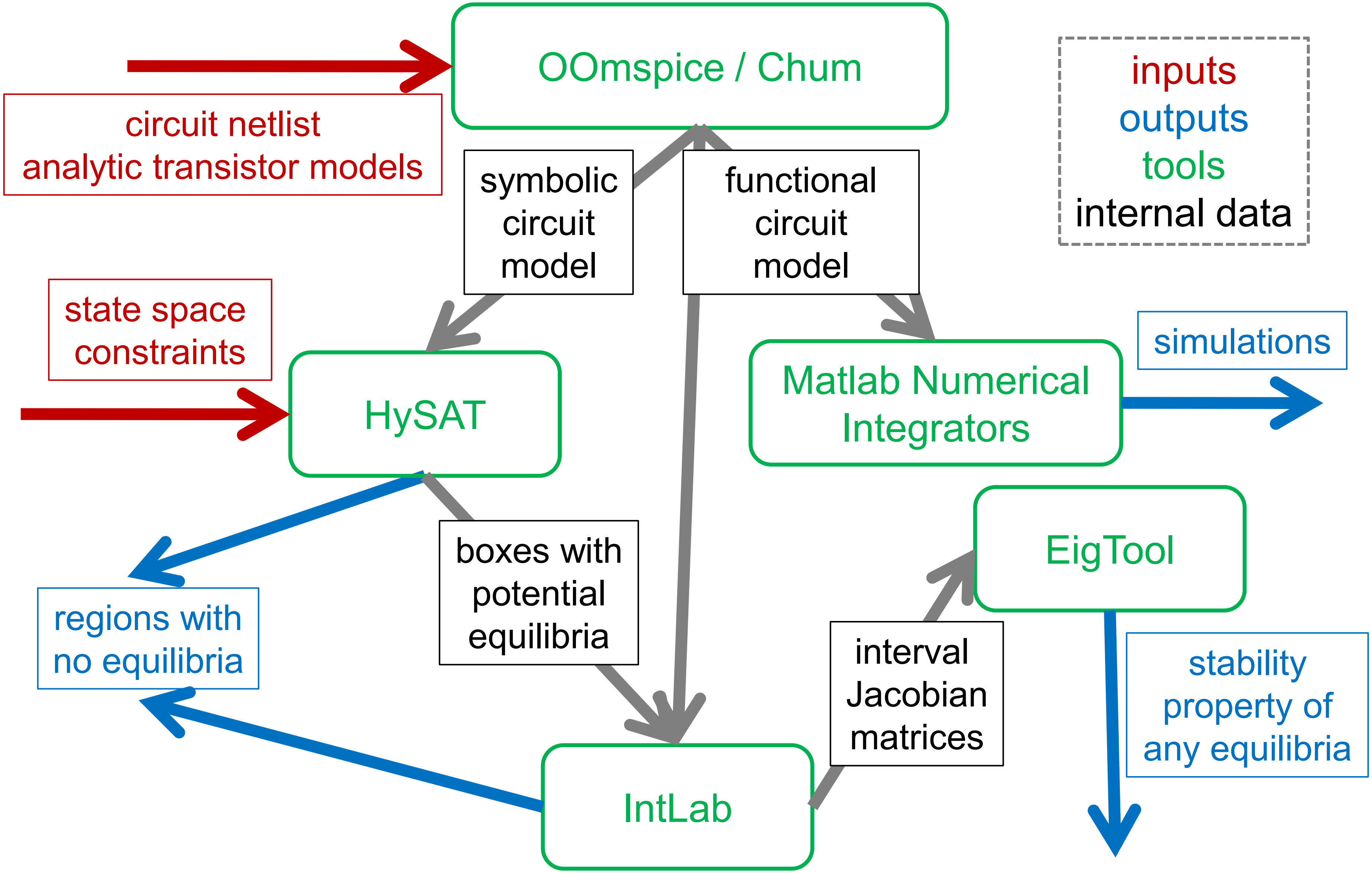
- For some region H of the state space, IntLab returns interval matrix $J_H = [A_C - \Delta, A_C + \Delta]$ containing $\partial f(\hat{x})/\partial x$ for all $\hat{x} \in H$
- Hurwitz test: J_H is stable if

$$\lambda_{\text{max}}(A_C^{(S)}) + \rho(\Delta^{(S)}) < 0$$

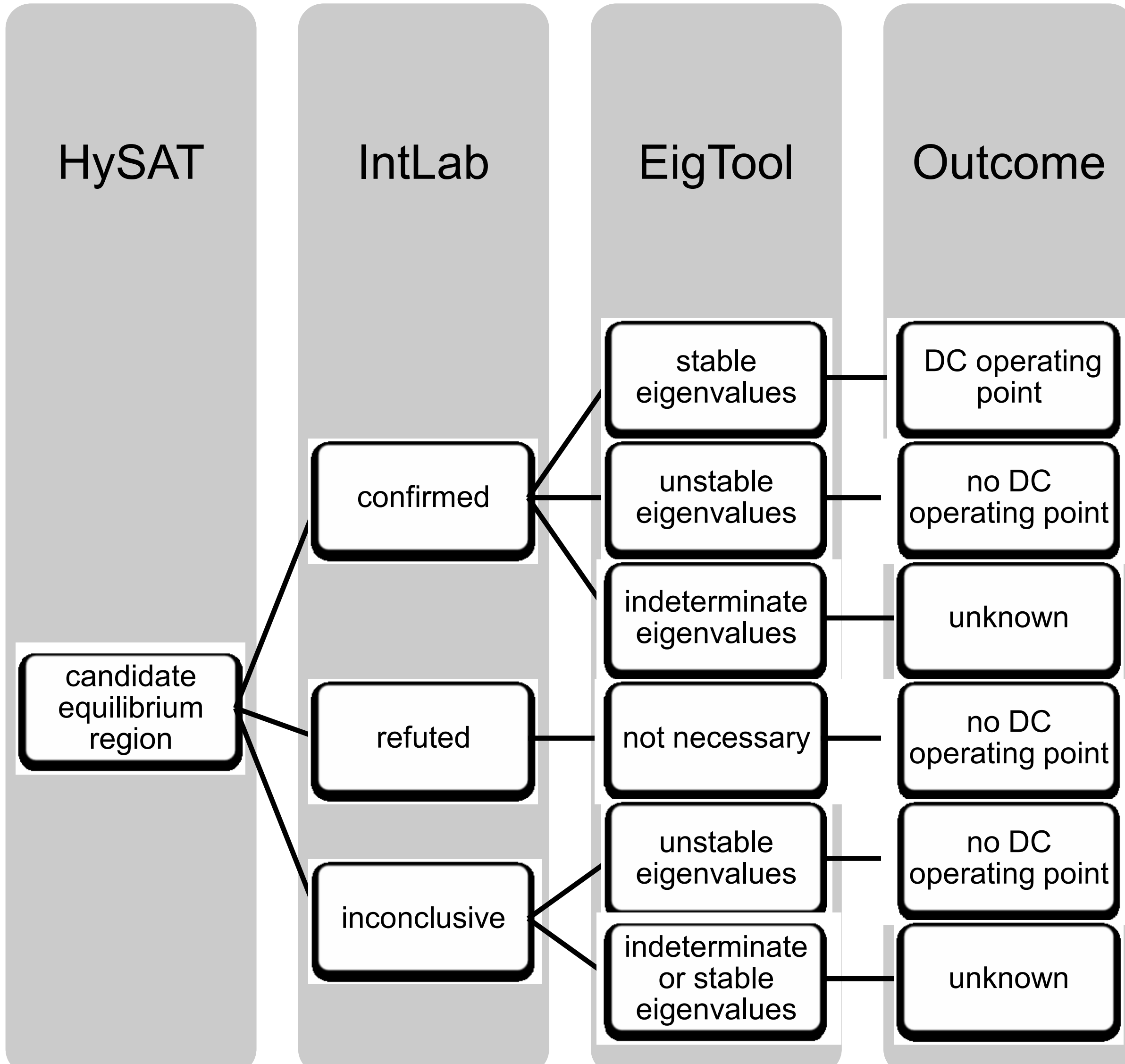
  where for matrix A, $A^{(S)} = \frac{1}{2}(A + A^T)$ and $\rho(A)$ is the spectral radius of A
- Can reach more general conclusions about stability / instability with ϵ-ps for $\epsilon_\Delta = \|\Delta\|_2$
  - No DC operating point: At least one isolated component of ϵ_Δ-ps in right-half plane
  - DC operating point(?): All ϵ_Δ-ps in left-half plane
  - Not a guarantee of a DC operating point:

$$\{A \mid A \in [A_C - \Delta, A_C + \Delta]\} \subset \{A_C + E \mid \|E\|_2 \leq \|\Delta\|_2\}$$

Our Approach (graphically)



Analysis Outcomes



Outline (Part 1)

- Motivation
- Mathematical characterization of DC operating points
- Our approach
  - Circuit model construction with netlist & OOmspice / Chum
  - Analytic exclusion with symbolic model & HySAT
  - Numerical exclusion with interval model & IntLab
  - Stability analysis with pseudospectra & EigTool
- Examples
  - Schmitt trigger
  - Ring & Rambus oscillators

Schmitt Trigger Netlist

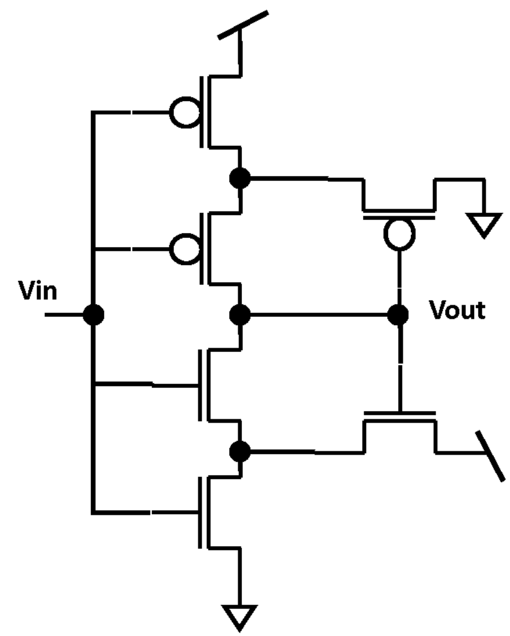
- Input to OOmspice

```
function   m = Schmitt_Trigger

%Parameters for First Order Transistors model
trn_params = 'Trans_params';
% Gate, Source and Drain Capacitance
C = [0.55e-15, 0.55e-15/2, 0.55e-15/2];
% Input Signal parameters
Pulse_params = 'pulse_param';

Vin1 = Voltage_Source_Pulse('Vin1', Pulse_params);
Vdd = Supply_Voltage('Vdd', 1.8);
Gnd = Ground('Gnd', 0);
NMos1 = NmosCell('NMos1', trn_params, C);
PMos1 = PmosCell('PMos1', trn_params, C);
PC = Trans_series('PC', trn_params, C, -1);
NC = Trans_series('NC', trn_params, C, 1);

m = connect_pins(m, Vdd, NMos1.D, 'node_vdd');
m = connect_pins(m, Vdd, PC.Vs);
m = connect_pins(m, Gnd, PMos1.D, 'node_gnd');
m = connect_pins(m, PC.Vgnd, Gnd);
m = connect_pins(m, NC.Vgnd, Gnd);
m = connect_pins(m, Gnd, Vin1.N);
m = connect_pins(m, Gnd, NC.Vs);
m = connect_pins(m, PC.Out, PMos1.S, 'node_y1');
m = connect_pins(m, NC.Out, NMos1.S, 'node_y2');
m = connect_pins(m, Vin1.P, PC.In, 'node_in');
m = connect_pins(m, Vin1.P, NC.In);
m = connect_pins(m, PMos1.G, PC.Vd, 'node_out');
m = connect_pins(m, NMos1.G, PC.Vd);
m = connect_pins(m, PC.Vd, NC.Vd);

m = end_circuit(m);
```

parameters

components

connectivity



Schmitt Trigger Circuit Model

- Output from OOmspice

```
function dydt = Schmitt_Trigger_model(t,y)

Cap_node_out_V_Port = y(1,:);
Cap_node_y2_V_Port = y(2,:);
Cap_node_y1_V_Port = y(3,:);

Vin1_V_Port = fun_pulse(t,'pulse_params')';
node_gnd_V = 0;
node_vdd_V = 1.80;
node_in_V = node_gnd_V+Vin1_V_Port;
node_y2_V = node_gnd_V+Cap_node_y2_V_Port;
node_y1_V = node_gnd_V+Cap_node_y1_V_Port;
node_out_V = node_gnd_V+Cap_node_out_V_Port;
Pmos1_PC_Ids = fun_mos(node_in_V,node_vdd_V,node_y1_V,-1,'Trans_Params');
Nmos1_NC_Ids = fun_mos(node_in_V,node_gnd_V,node_y2_V,1,'Trans_Params');
Nmos2_NC_Ids = fun_mos(node_in_V,node_y2_V,node_out_V,1,'Trans_Params');
Pmos2_PC_Ids = fun_mos(node_in_V,node_y1_V,node_out_V,-1,'Trans_Params');
PMos1_Ids = fun_mos(node_out_V,node_y1_V,node_gnd_V,-1,'Trans_Params');
NMos1_Ids = fun_mos(node_out_V,node_y2_V,node_vdd_V,1,'Trans_Params');
Cap_node_out_I_Port = -Pmos2_PC_Ids-Nmos2_NC_Ids;
Cap_node_y2_I_Port = -Nmos1_NC_Ids+Nmos2_NC_Ids+NMos1_Ids;
Cap_node_y1_I_Port = -Pmos1_PC_Ids+Pmos2_PC_Ids+PMos1_Ids;

dydt = [Cap_node_out_I_Port/1.650000e-15;...
        Cap_node_y2_I_Port/8.250000e-16;...
        Cap_node_y1_I_Port/8.250000e-16];
```

state space

provided by Chum

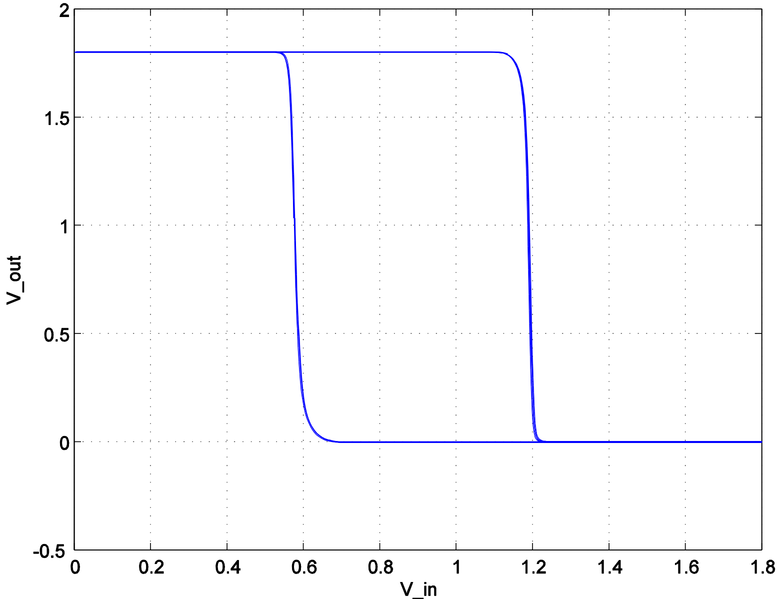
Equations

KCL

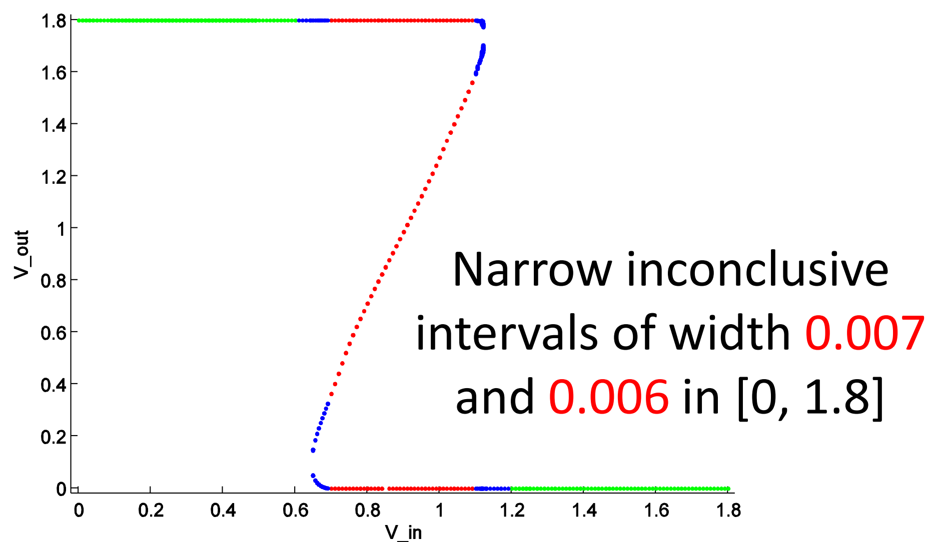
ODE

(Inverting) Schmitt Trigger Behaviour

- Input to output mapping is like an inverter, but with hysteresis
- Analysis determines location and stability of each equilibrium for each fixed input value
  - green: one equilibrium of known stability
  - red: three equilibria of known stability
  - blue: multiple equilibria of indeterminate stability



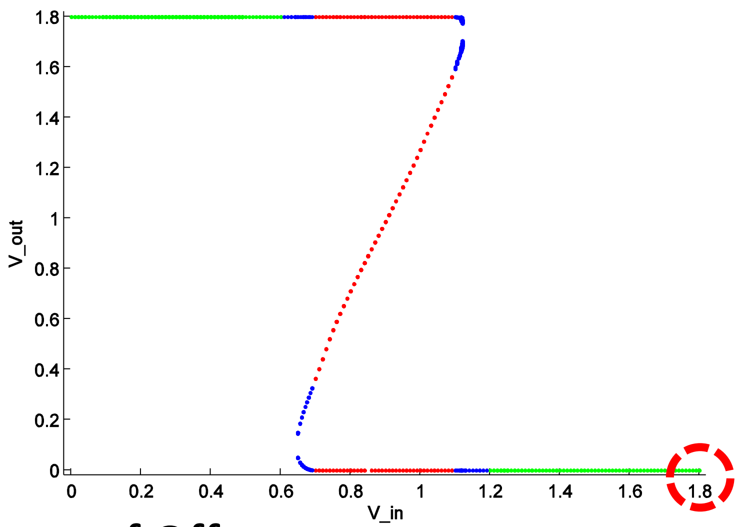
Simulation



DC equilibria detected

Schmitt Trigger Pseudospectrum V_{in} = 1.8

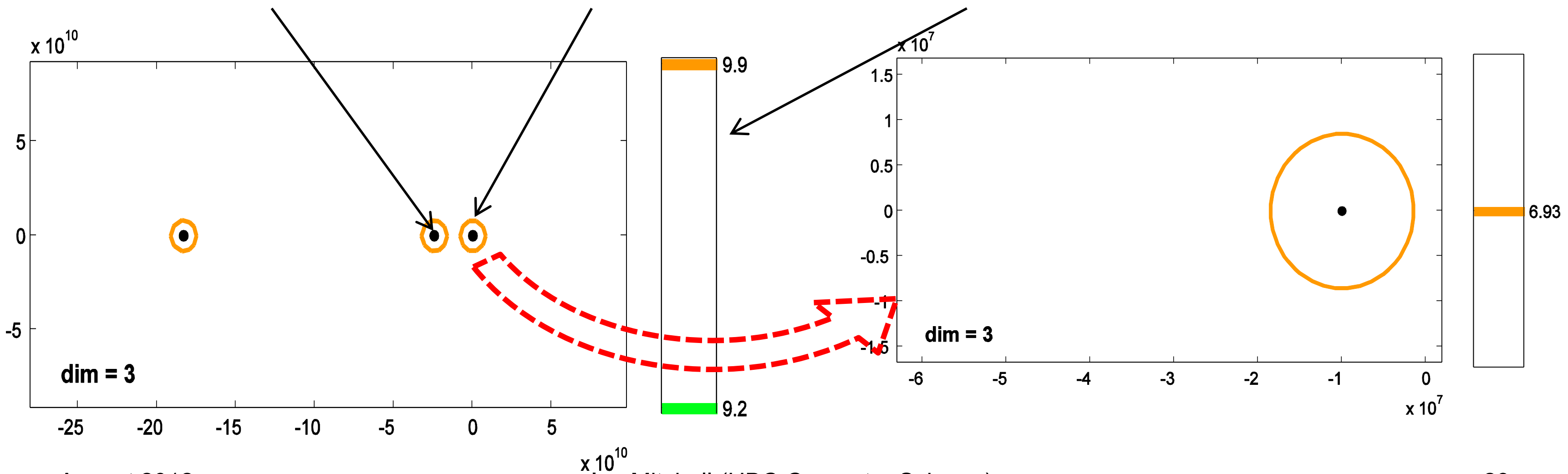
- Pseudospectrum of overapproximation of interval Jacobian lies in left half plane
- Equilibrium is a DC operating point



Eigenvalue of central matrix

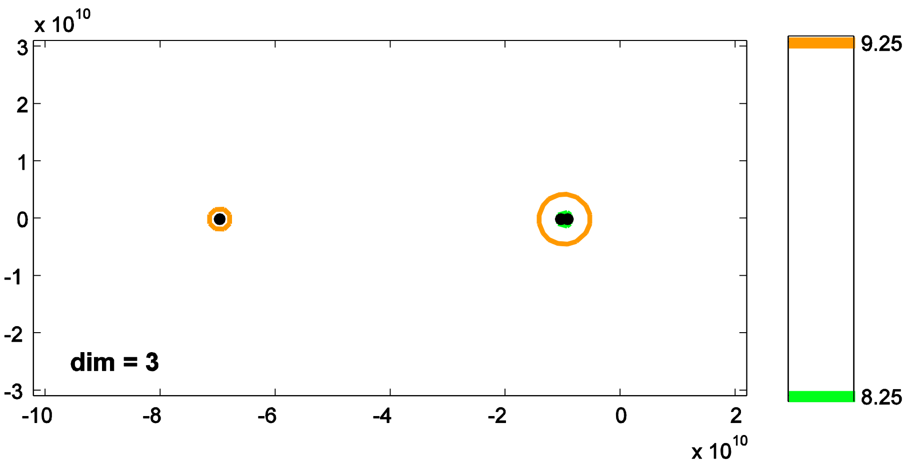
Bounds on eigenvalue

Norm of Offset Matrix (log scale)

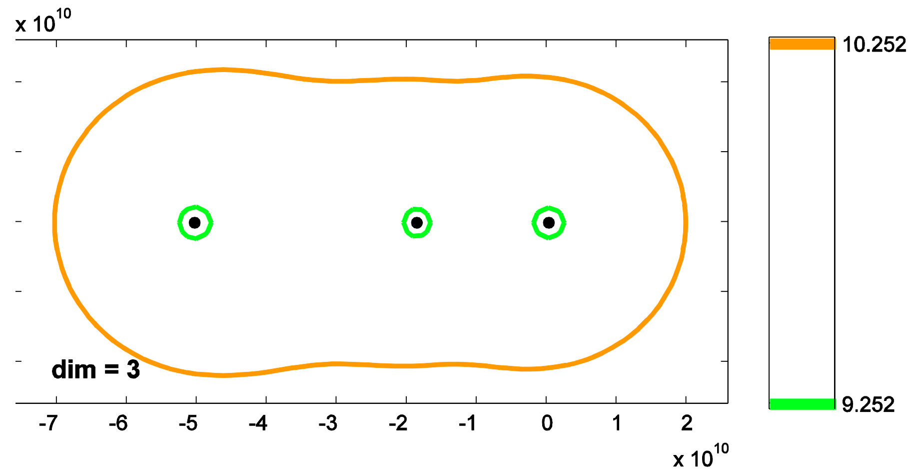


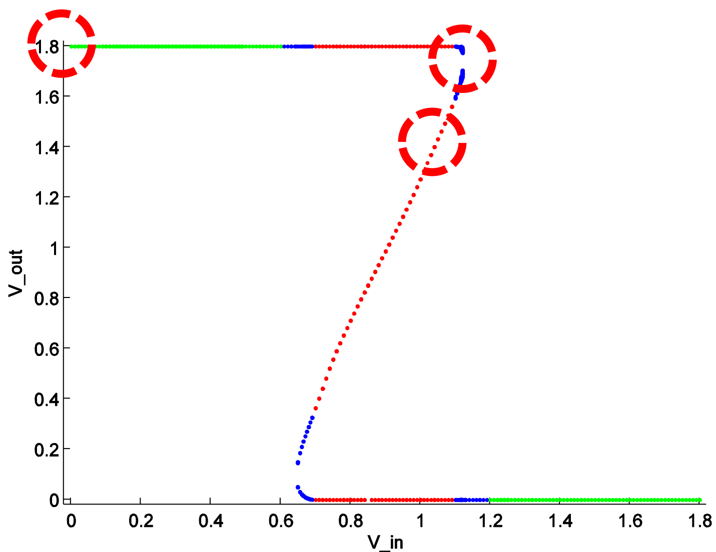
More Schmitt Trigger Pseudospectra

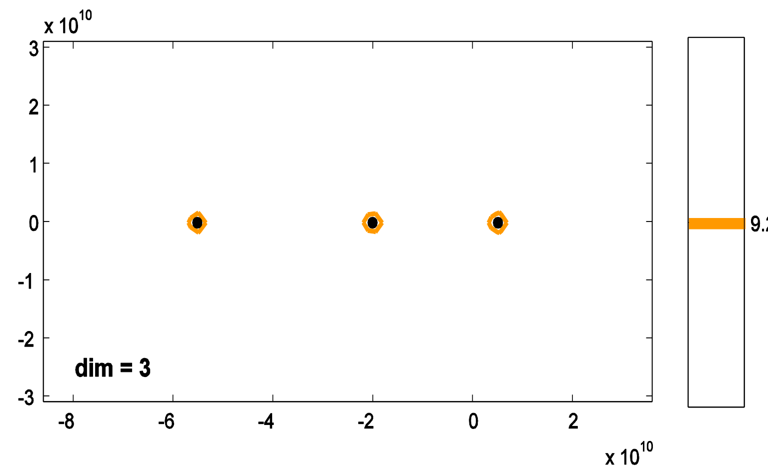
$V_{\text{in}} = 0$



$V_{\text{in}} = 1.124$ upper region



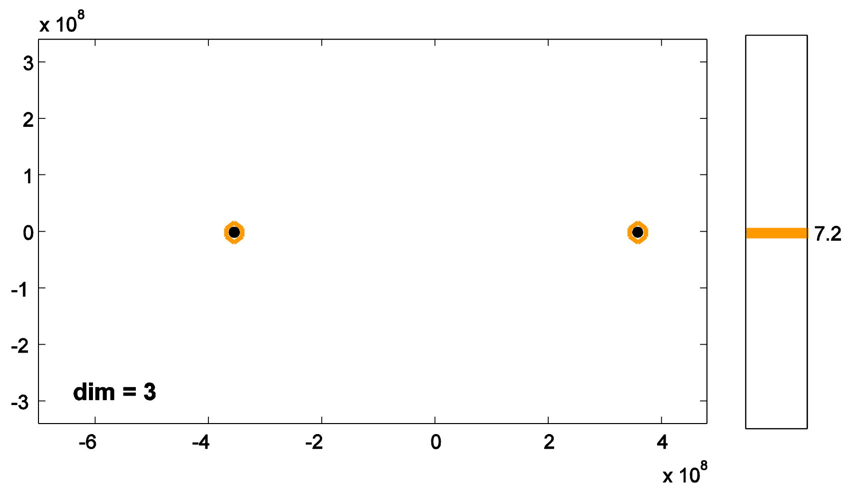




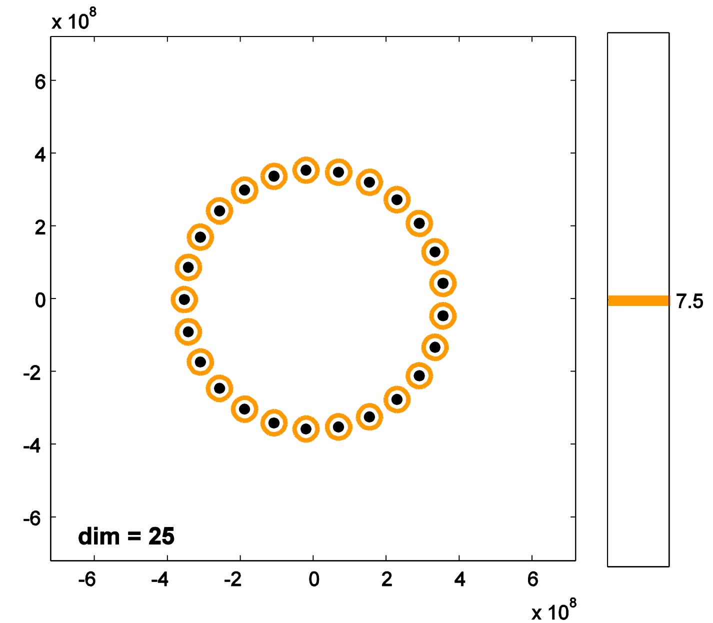
$V_{\text{in}} = 1$ middle equilibrium

Ring Oscillators' Results

- Ring oscillators with an odd number of inverters (up to 25) do not have DC operating points
  - However, a sufficiently large load capacitance on the output stage can create a DC operating point
- Ring oscillators with an even number of inverters (up to 24) always have DC operating points



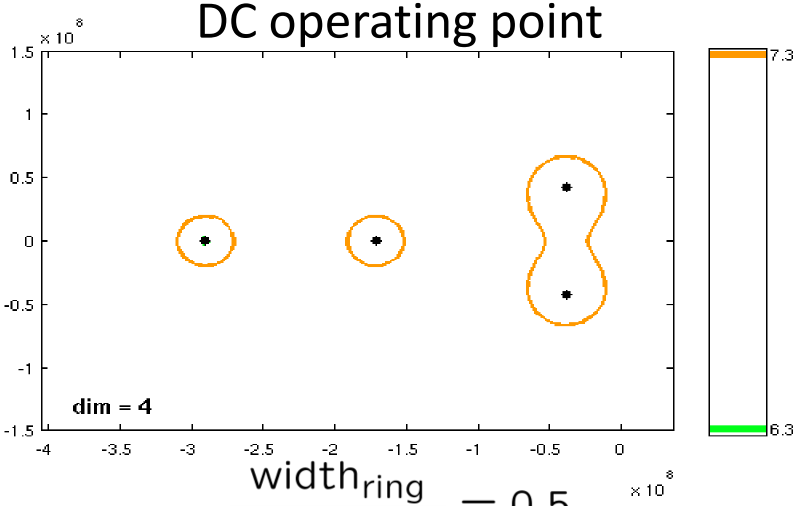
three inverters



25 inverters

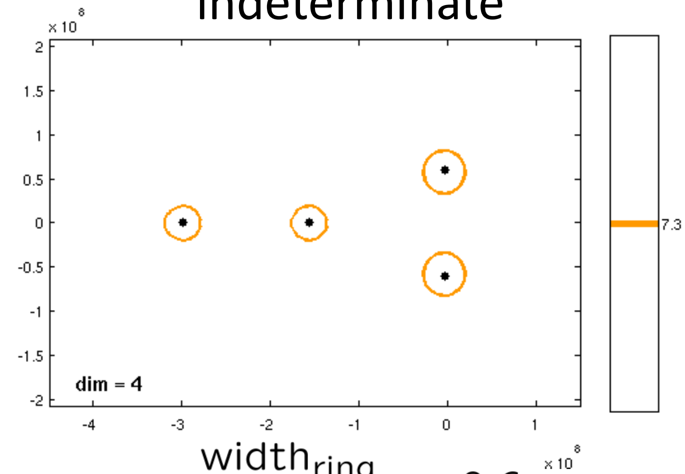
Rambus Oscillator’s Results

DC operating point



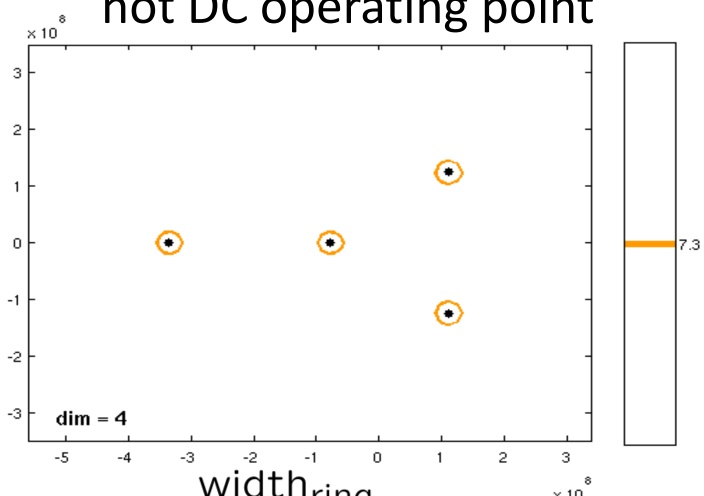
$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} = 0.5$$

indeterminate



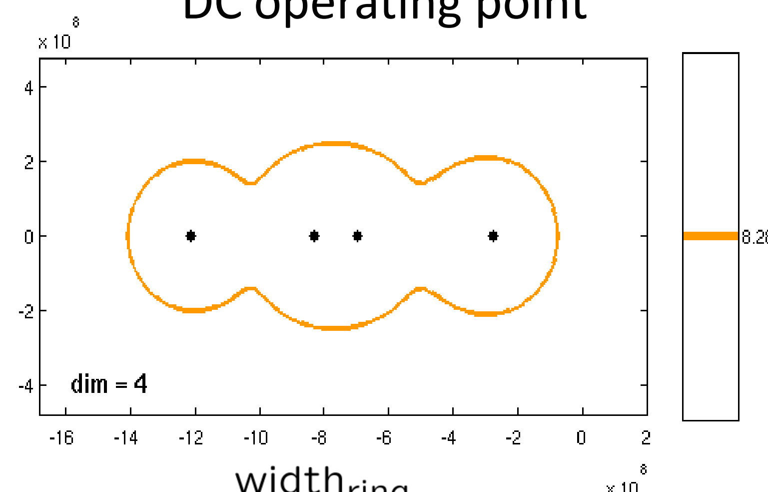
$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} = 0.6$$

not DC operating point



$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} = 0.9$$

DC operating point



$$\frac{\text{width}_{\text{ring}}}{\text{width}_{\text{bridge}}} = 3.5$$

Conclusions (Part 1)

- DC analysis can be approached formally and with rigour
- DC equilibria can be localized and their stability properties analyzed to find DC operating points using a collection of (almost) freely available tools
  - Models constructed by OOmspice / Chum (not released)
  - State space regions excluded by HySAT
  - Remaining regions confirmed or refuted (or not) by IntLab
  - Remaining regions’ stability examined by EigTool
- Demonstrated on Schmitt trigger, ring oscillators and Rambus oscillator
- Remaining challenges
  - Improved circuit models
  - Rigourous treatment of multiple roots
- Not a challenge: High dimensions

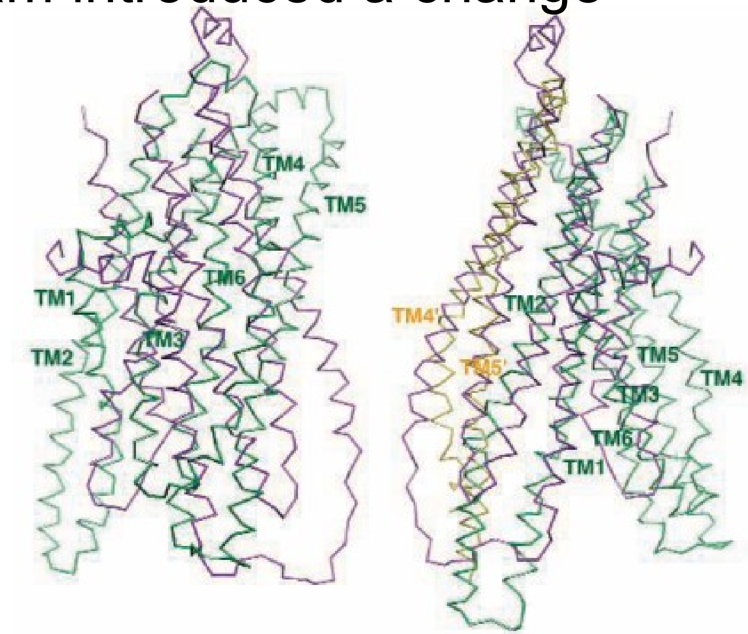
The Real Problem

- I can’t reproduce any of these results
- These aren’t my only such results
- I’m not alone

Accurate to within ±One (Hundred Percent)

- 2001–2005: Geoffrey Chang and colleagues published a number of high profile protein structures
  - 2001 paper on MsbA cited 360+ times by 2006
- September 2006: A dramatically different structure for a related protein is published
- December 2006: Chang et al retract five papers because "An in-house data reduction program introduced a change in sign…"

Image from:
Miller, "A Scientist's Nightmare: Software Problem leads to Five Retractions" in *Science* 314(5807): 1856-1857 (22 December 2006)



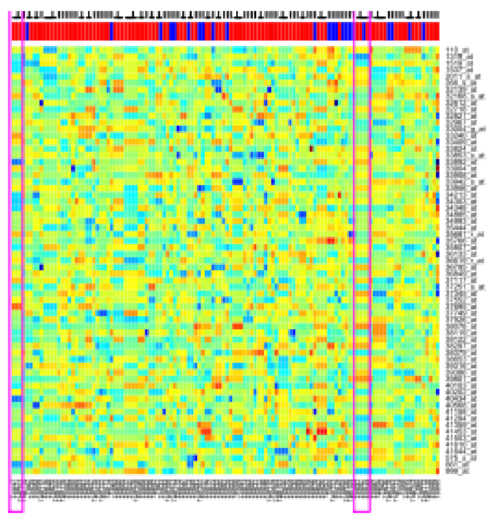
Flipping fiasco. The structures of MsbA (purple) and Sav1866 (green) overlap little (*left*) until MsbA is inverted (*right*).

A Simple Labelling Mistake?

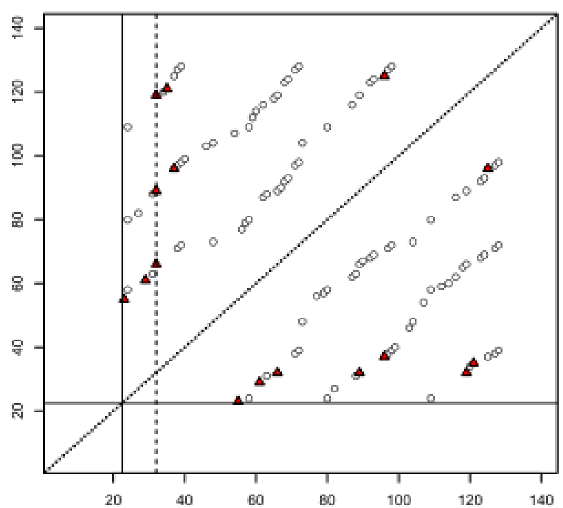
- 2006: Anil Potti and colleagues announce method for predicting patient response to chemotherapy drugs based on gene microarray data
  - 200+ citations by 2009
- 2007: Clinical trials begin
- 2007–2009: Baggerly, Coombes and colleagues try to reproduce results, but find frequent inconsistencies
- 2010–2011: Trials stopped, Potti resigns, 7+ retractions

Images from:
Baggerly & Coombes, "Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology" in *Annals of Applied Statistics* 3(4): 1309-1334 (2009)

Response Labelling +
Gene Expression Heatmap



Repeated Columns
(Δ: inconsistent labels)



Why so Secretive?

- 2005: Jelte Wicherts and colleagues requested data from 49 papers recently published in two highly ranked American Psychological Association journals (part of a larger study)
  - Corresponding authors had signed publication form agreeing to share data
  - 21 shared some data, 3 refused (lost or inaccessible data), 12 promised to later but did not, and 13 never responded
- 2011: Wicherts and colleagues analyze internal consistency of p-values reported from null hypothesis tests
  - Willingness to share is correlated with fewer reporting errors and relatively stronger evidence against NH

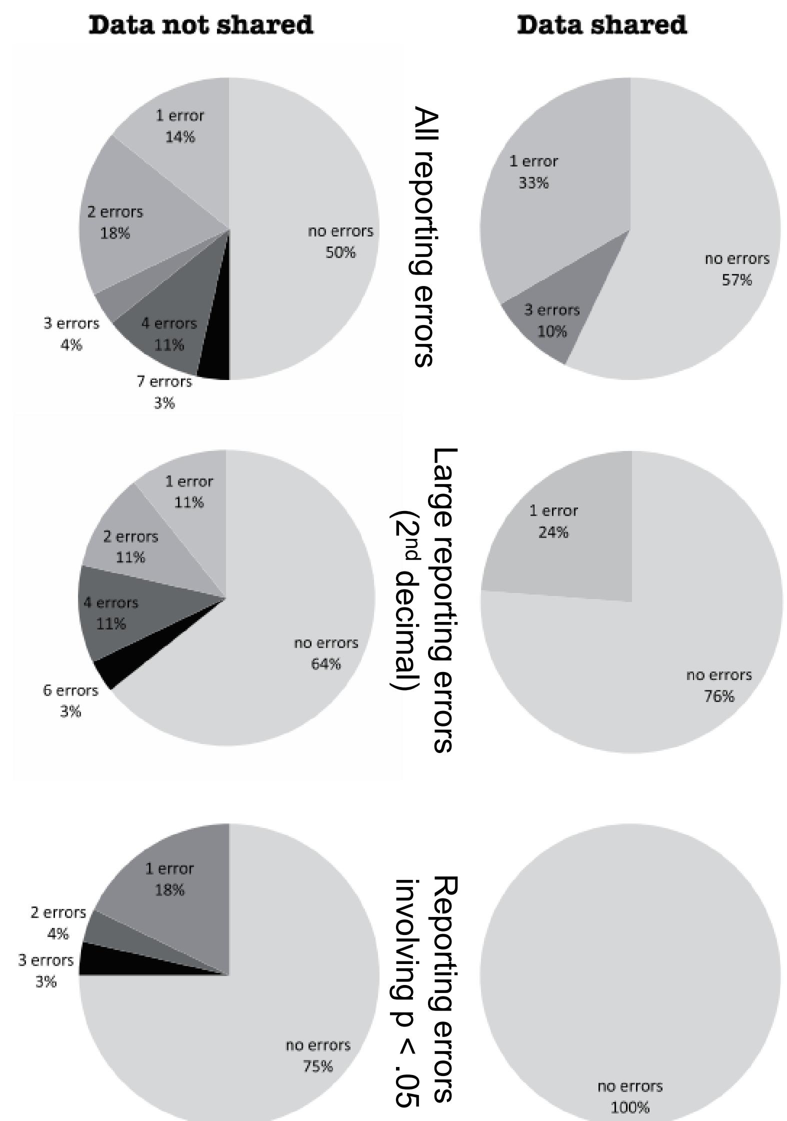


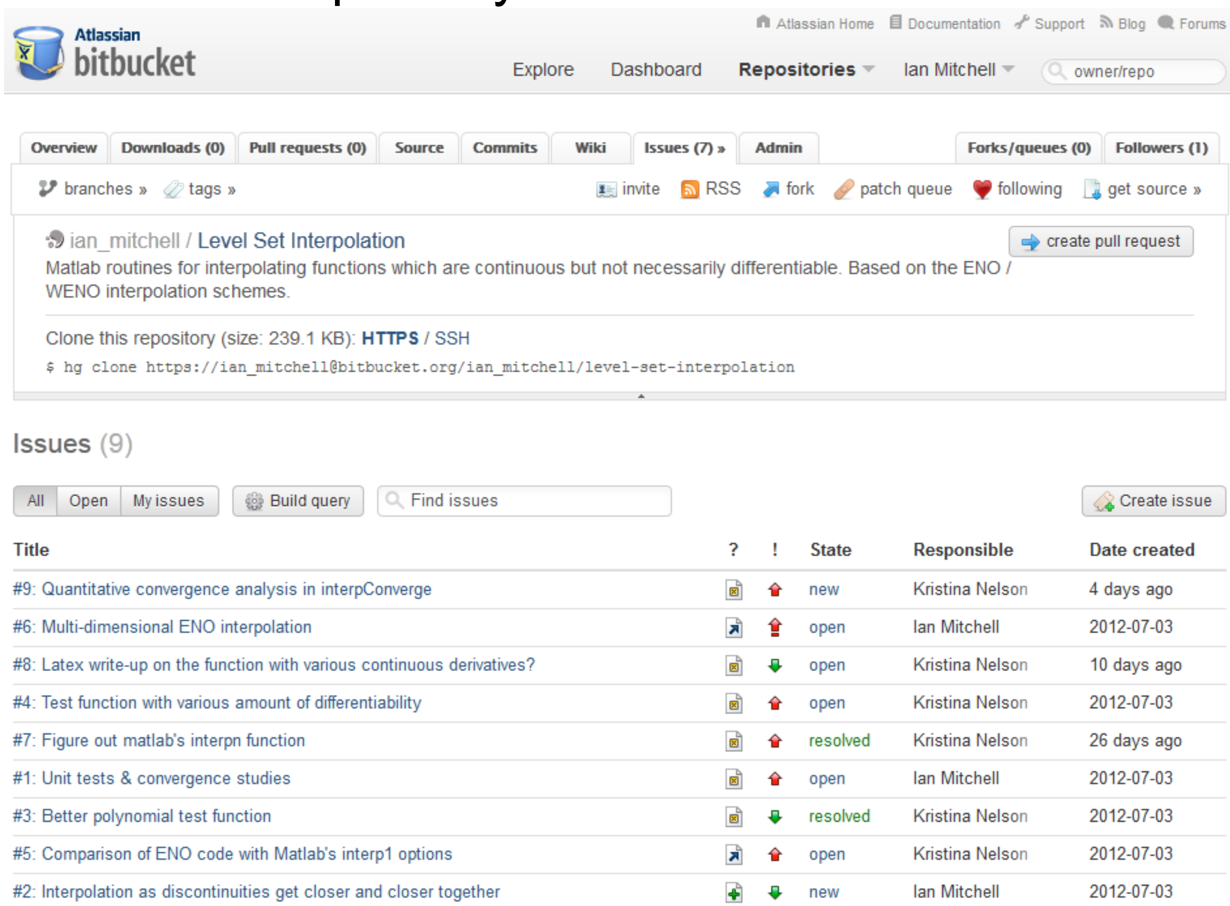
Image from:
Wicherts, Bakker & Molenaar, "Willingness to share research data is related to the strength of the evidence and the quality of reporting of statistical results" in *PLoS ONE* 6(11), Nov. 2011.

Some Suggestions for Doing It Better

- Use a (modern) version control system
  - Online repositories (eg: bitbucket, github, google code, sourceforge) include wikis and issue trackers
- Document with the data (and code is data)
  - You will forget how and why you did things, files and directories will get separated and lost
- Write tests first and run them often
  - Bugs are inevitable and “static” code isn’t
- If you do it twice, automate it
  - Computers are better at repetition, you can automate a person with a checklist, and automation is documentation
- Look at the code together
  - Code review and pair programming lead to demonstrable improvements in code quality
- Improve your process gradually, but continually
  - Every little bit helps

Example: Source and Issue Management

- Summer project with NSERC USRA Kristina Nelson: ENO interpolation in multiple dimensions
- Managed with Mercurial (hg) version control software and bitbucket online repository



Atlassian Home · Documentation · Support · Blog · Forums

Atlassian bitbucket

Explore · Dashboard · Repositories · Ian Mitchell · owner/repo

Overview | Downloads (0) | Pull requests (0) | Source | Commits | Wiki | Issues (7) » | Admin | Forks/queues (0) | Followers (1)

branches » tags » invite RSS fork patch queue following get source »

ian_mitchell / Level Set Interpolation

create pull request

Matlab routines for interpolating functions which are continuous but not necessarily differentiable. Based on the ENO / WENO interpolation schemes.

Clone this repository (size: 239.1 KB): **HTTPS** / SSH

```
$ hg clone https://ian_mitchell@bitbucket.org/ian_mitchell/level-set-interpolation
```

Issues (9)

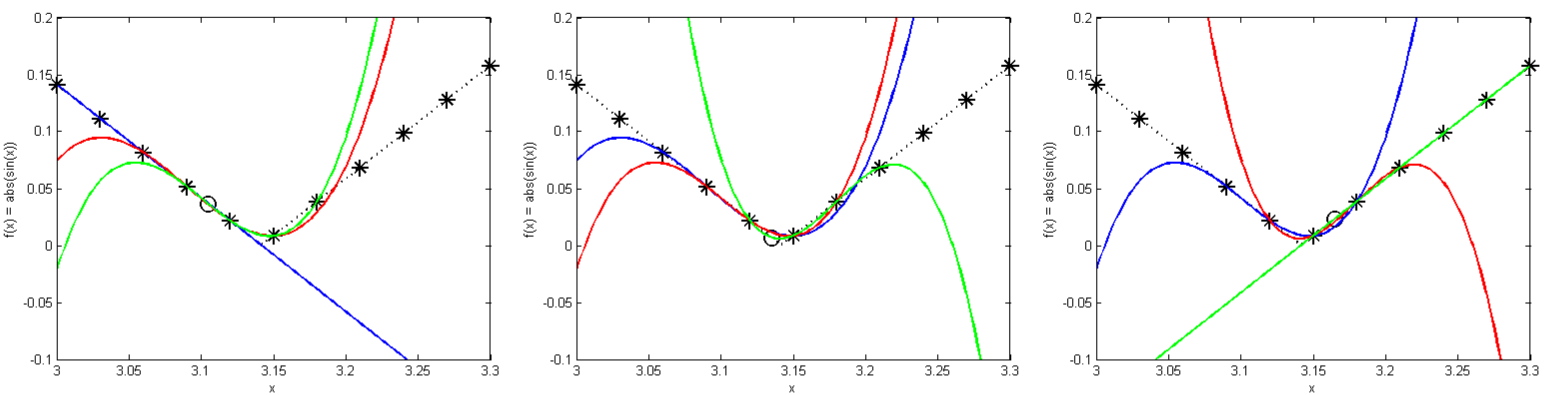
All · Open · My issues · Build query · Find issues · Create issue

Title	?	!	State	Responsible	Date created
#9: Quantitative convergence analysis in interpConverge			new	Kristina Nelson	4 days ago
#6: Multi-dimensional ENO interpolation			open	Ian Mitchell	2012-07-03
#8: Latex write-up on the function with various continuous derivatives?			open	Kristina Nelson	10 days ago
#4: Test function with various amount of differentiability			open	Kristina Nelson	2012-07-03
#7: Figure out matlab's interpn function			resolved	Kristina Nelson	26 days ago
#1: Unit tests & convergence studies			open	Ian Mitchell	2012-07-03
#3: Better polynomial test function			resolved	Kristina Nelson	2012-07-03
#5: Comparison of ENO code with Matlab's interp1 options			open	Kristina Nelson	2012-07-03
#2: Interpolation as discontinuities get closer and closer together			new	Ian Mitchell	2012-07-03

Example: Verifying Numerical Code

- For most scientific computing algorithms, testing for exact answers is not possible
  - Discretization, truncation and rounding are unavoidable
  - How can we automate testing?
- Background: Essentially Non-Oscillatory Interpolants
  - High order local polynomial interpolants for insufficiently smooth data often become oscillatory and hence inaccurate
  - ENO idea: Several different high order local polynomial interpolants can be constructed for a given set of data, so use the least oscillatory [Harten et. al., *J. Computational Physics* 71(2): 231–303, 1987]

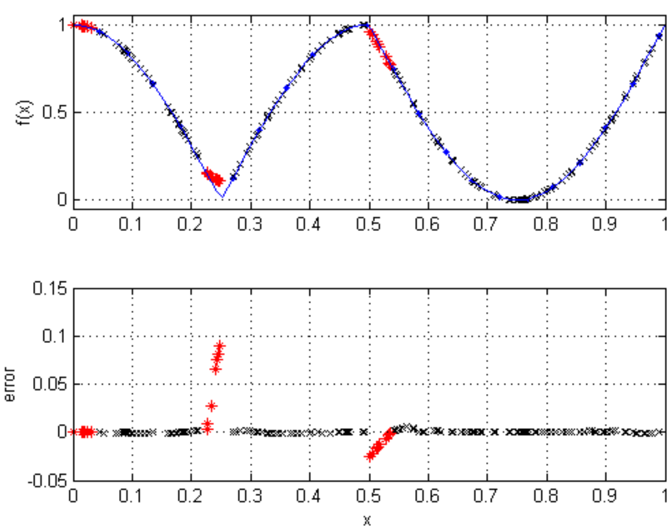
cubic polynomial interpolants of $|\sin(x)|$, $x_i = 3 + 0.03i$, for evaluation points in various intervals near $x = \pi$



Verification with Convergence Rate Test

- Theoretical order of accuracy for polynomial interpolants is easily derived (for smooth data)
- Given known test function, experimental order of accuracy for interpolants is easily measured
  - Experimental order of accuracy is very sensitive to bugs
  - Poor choice of test functions may not properly exercise code

test function & error (ENO3)

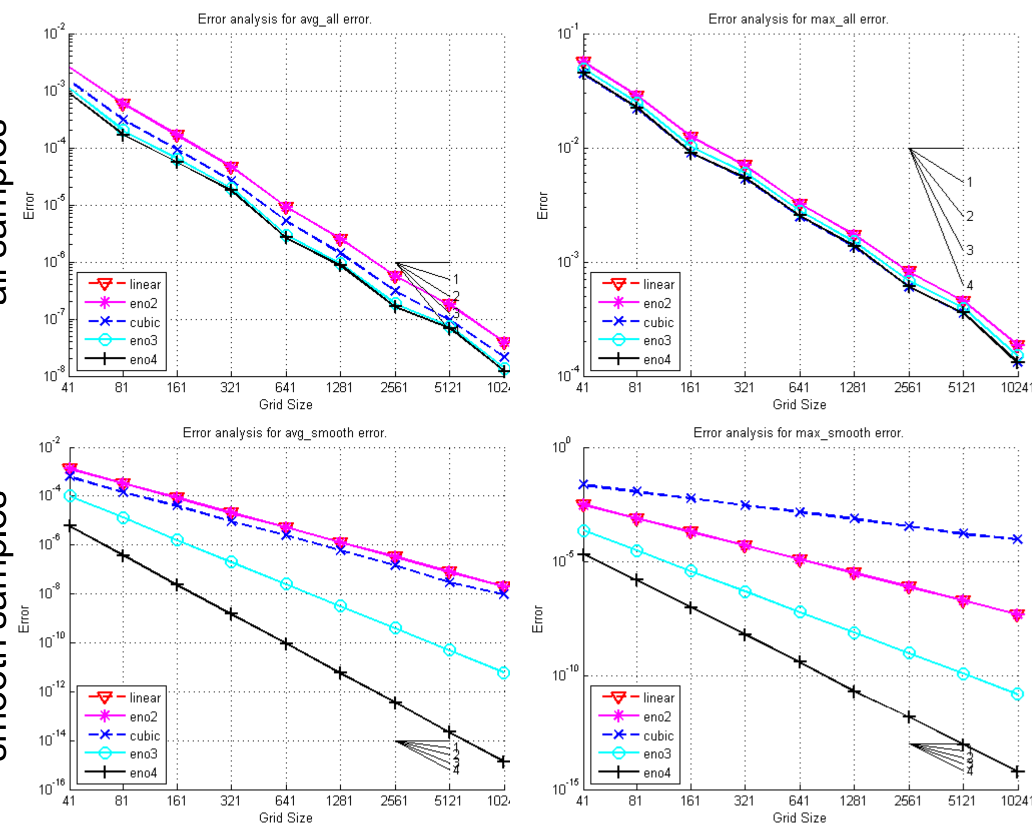


average error

maximum error

all samples

smooth samples



Citations and Links

- Software Carpentry website
  - Upcoming bootcamp at UBC October 18-19, 2012
- Ideas from software engineering
  - Heroux & Willenbring, "Barely Sufficient Software Engineering: 10 Practices to Improve Your CSE Software" in *ICSE Workshop on Software Engineering for Computational Science & Engineering*, pp.15-21 (2009)
  - Wilson et. al., "Best Practices for Scientific Computing" in preparation
  - Sink, *Version Control by Example*, 2011
- Testing differential equation codes
  - Oberkampf & Roy, *Verification and Validation in Scientific Computing*, 2010
  - Roy, "Review of Code and Solution Verification Procedures for Computational Simulation", *J. Comp. Physics* 205:131-156
  - Knupp & Salari, *Verification of Computing Codes in Computational Science and Engineering*, 2003
- Additional links and readings (CPSC 535Z at UBC): https://sites.google.com/site/ubccpsc535zwinter2011/

Formal Identification of DC Operating Points in Integrated Circuits and some Lessons in (Ir)Reproducible Research in Computational Math

For more information contact

Ian Mitchell

Department of Computer Science

University of British Columbia

mitchell@cs.ubc.ca

http://www.cs.ubc.ca/~mitchell