Chaos in Computer Performance

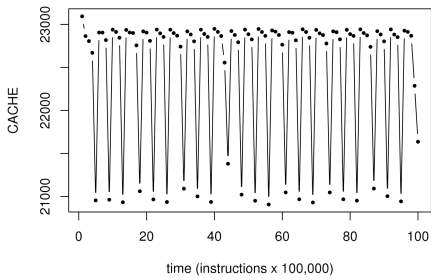
Liz Bradley & friends

University of Colorado Boulder

Department of Computer Science
www.cs.colorado.edu/~lizb

CNS-0720692

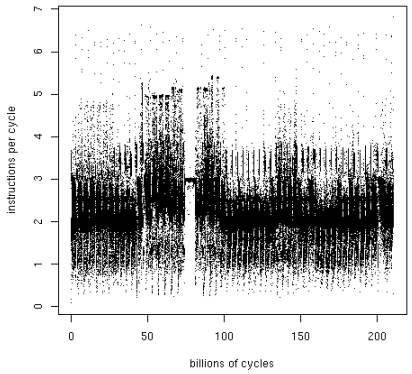
Computer dynamics




```
for (i=0; i<255; i++)
  for (j=i; j<255; j++)
    data[j][i]=0;
```

Intel Core2

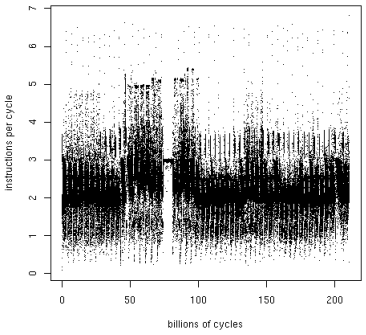
What about complex programs?



The `bzip2` SPEC benchmark

Intel Core2

How the computer systems community models this

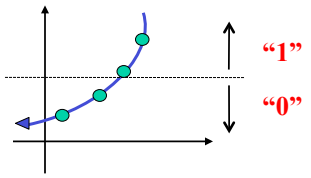


- End-to-end metrics (e.g., "FLOPS")
- Statistics (e.g., mean, variance)
- Stochastic processes
- Implicitly assumes linearity & time invariance…
- *…but computers are nonlinear, deterministic, and dynamic*

A better way to model computer performance

Flow: continuous time $x' = f(x)$

Map: discrete time $x_{n+1} = f(x_n)$



Underlying dynamics: continuous time & space — "flow" →

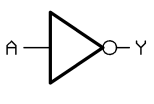
Clock abstraction: quantizes time — "map" ●●

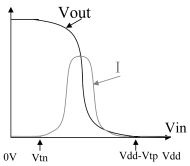
Digital abstraction: quantizes space — "symbol dynamics" 0/1

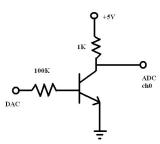
Modeling computer performance: linear or nonlinear?

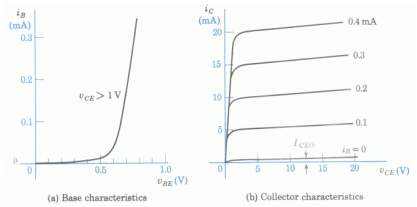
```
if x==0 then..else…

TEST x,0
BNE 0x10
```

abstraction ↑

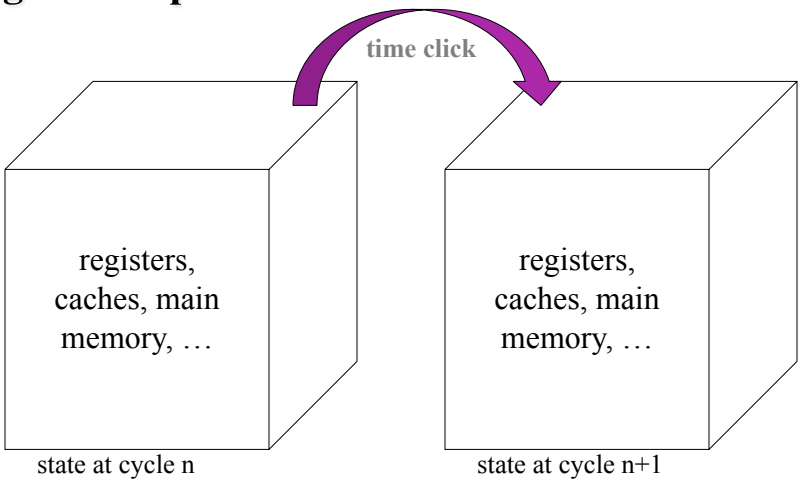






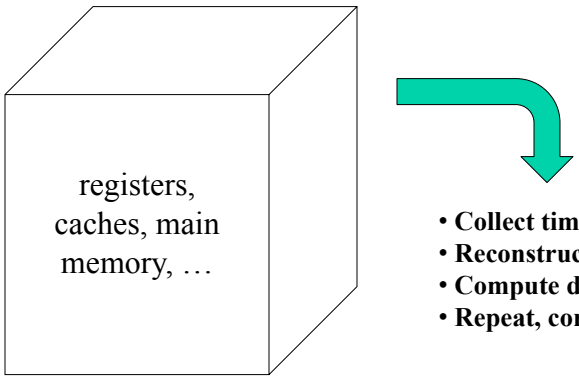


A nonlinear dynamics model of a digital computer



…a deterministic, nonlinear, high-dimensional map!

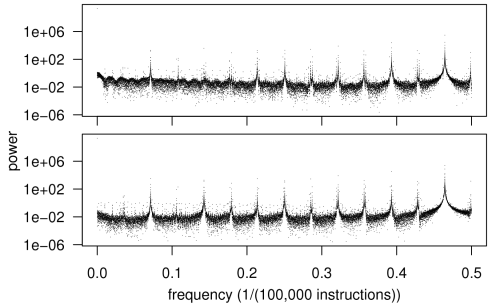
Validating that model



- **Collect time-series data**
- **Reconstruct the dynamics**
- **Compute dynamical invariants**
- **Repeat, corroborate, …**

Measuring computer dynamics

- Custom measurement infrastructure
  Use HPMs, turn off other processes, and grab data every n cycles
- Change n and see if results change:

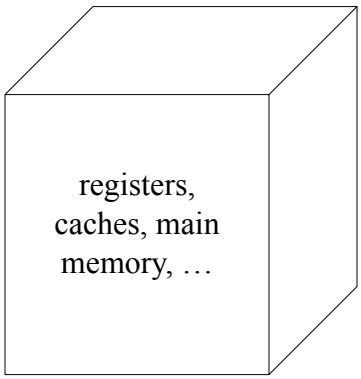


T_{sample} = 100,000 cycles

T_{sample} = 50,000 cycles

- Corroborate all calculations using other metrics
- *See Todd's thesis & papers for lots more about the associated effects & pitfalls*

Problem…

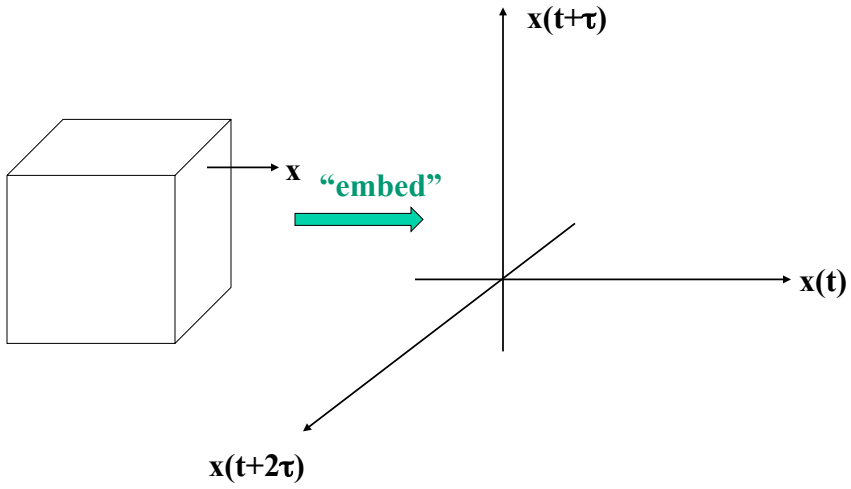


- We don't know what the state variables are
- Let alone can we measure them
- And there may be a lot of them

Control theory's "observer problem."

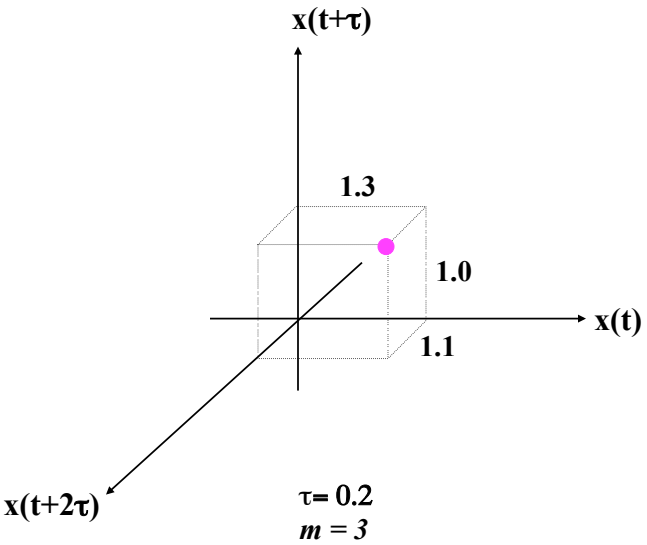
A partial solution:

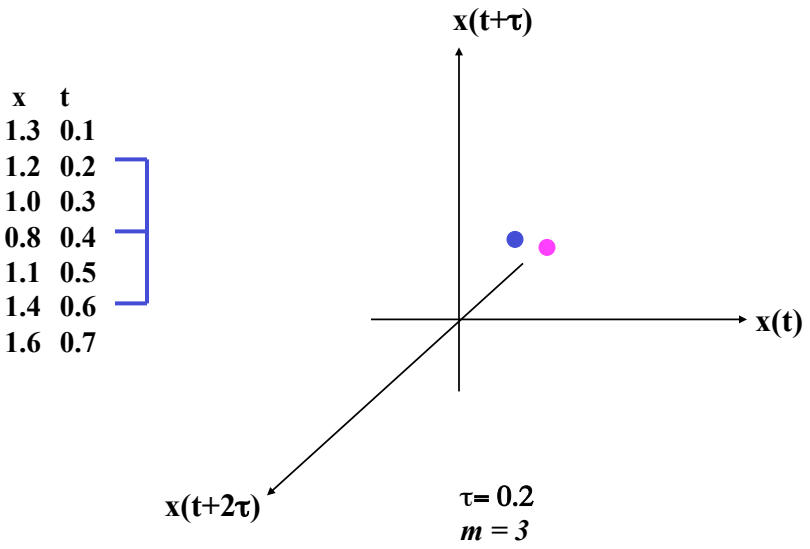
delay-coordinate embedding



Mechanics:

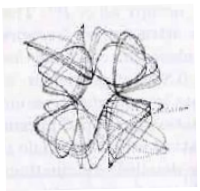
x	t
1.3	0.1
1.2	0.2
1.0	0.3
0.8	0.4
1.1	0.5
1.4	0.6
1.6	0.7

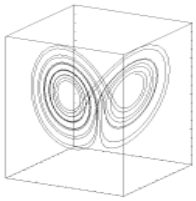




Reconstructing the dynamics: delay-coordinate embedding

Takens(*): for the right τ and enough dimensions, the embedded dynamics are diffeomorphic to (have same topology as) the original state-space dynamics.

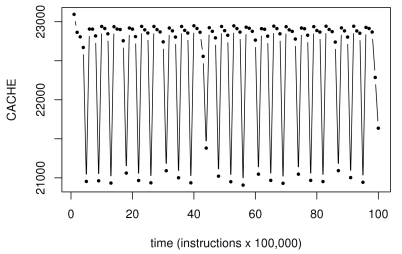




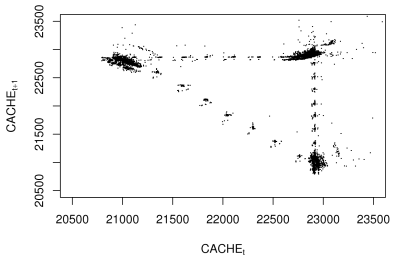
* Whitney, Mane, …

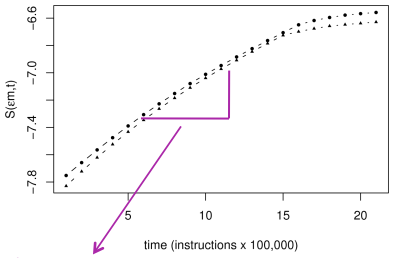
Note: this also requires a smooth, generic measurement function. The HPMs effect that as long as they do not overflow.

`col_major` dynamics on an Intel Core2



embed
$\tau = 1^*$
$m = 12$





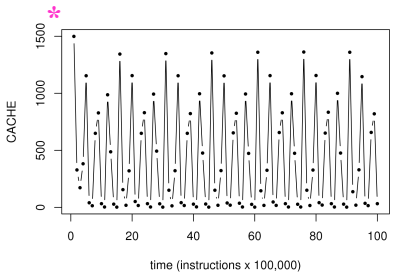
$\lambda = 0.08^*$

d_{corr}=0.83 (graph not shown)

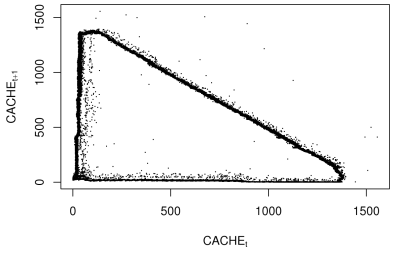
```
for (i=0; i<255; i++)
  for (j=i; j<255; j++)
    data[j][i]=0;
```

* All times scaled by 100,000 cycles

`col_major` dynamics on an Intel Pentium4



embed
$\tau = 1^*$
$m = 12$

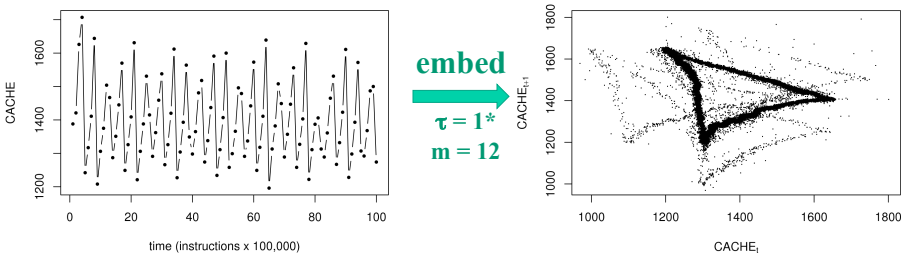


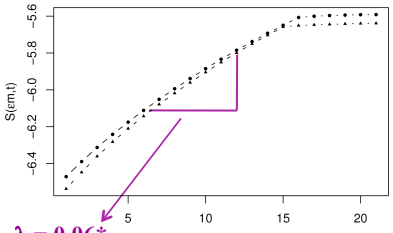
Periodic!

```
for (i=0; i<255; i++)
  for (j=i; j<255; j++)
    data[j][i]=0;
```

* Different cache measure

`row_major` dynamics on an Intel Core2





λ = 0.06*

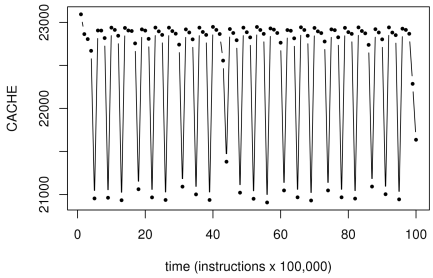
d_{corr} = 1.16 (graph not shown)

```
for (i=0; i<255; i++)
  for (j=i; j<255; j++)
    data[i][j]=0;
```

* All times scaled by 100,000 cycles

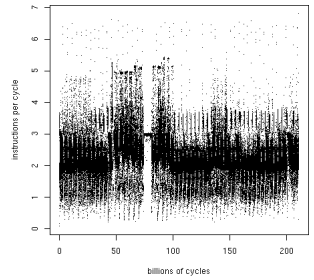
Realistic and representative?

```
for (i=0; i<255; i++)
  for (j=i; j<255; j++)
    data[j][i]=0;
```

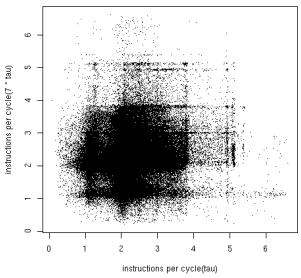


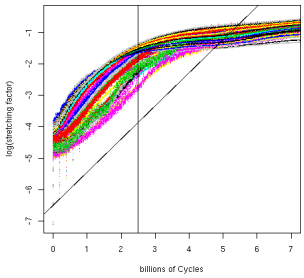
- This "microkernel" experiment is carefully crafted to isolate the asymptotic dynamics of a simple piece of code
- Real programs don't act like that
- Does this NLTSA approach give us any traction with them?

bzip2 dynamics on an Intel Core2



embed
τ = 194*
m = 10

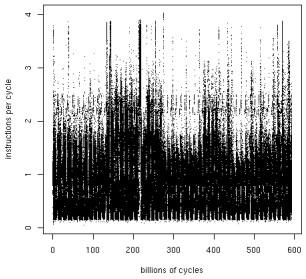




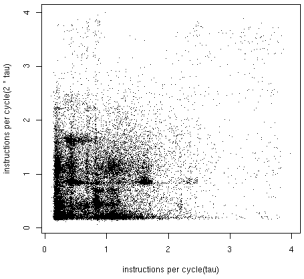
λ = 1.06*

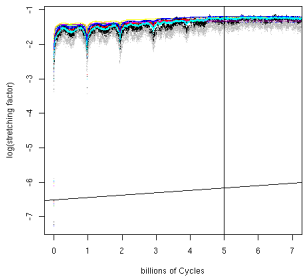
* All times scaled by 100,000 cycles

bzip2 dynamics on an Intel Pentium 4



embed
τ = 973*
m = 12





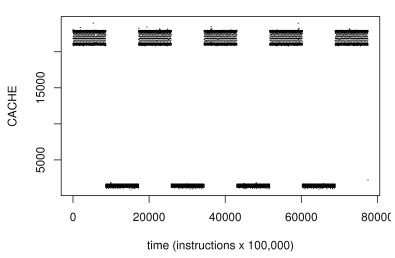
λ = 0.07*

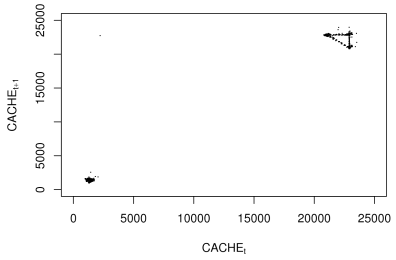
* All times scaled by 100,000 cycles

Composition dynamics

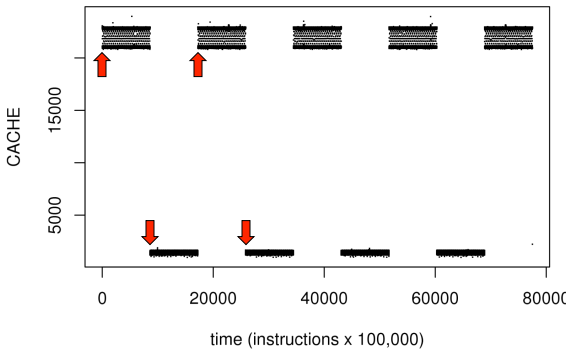
```
for (a=0; a<50000; a++)
 for (i=0; i<255; i++)
  for (j=i; j<255; j++)
   data[j][i]=0;
```

```
for (a=0; a<50000; a++)
 for (i=0; i<255; i++)
  for (j=i; j<255; j++)
   data[i][j]=0;
```



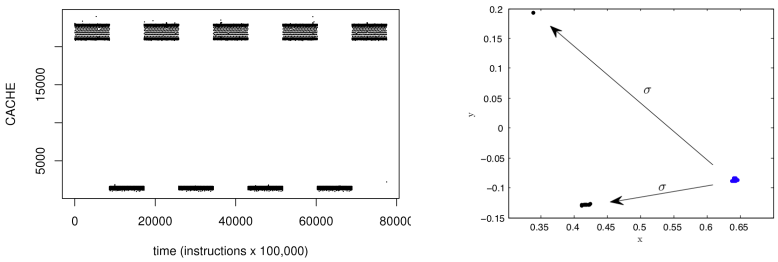


Decomposing and analyzing interleaved dynamics



- Detect the "phases" ⬇⬆
- Decide which chunks are "similar"
- Piece them together and analyze them

Topology-based signal separation



Idea:

- deterministic dynamics, if embedded correctly, are continuous

Conjecture:

- if the image of a connected set is not connected, more than one dynamics is at work

Approach:

- track connectedness over time

Applications:

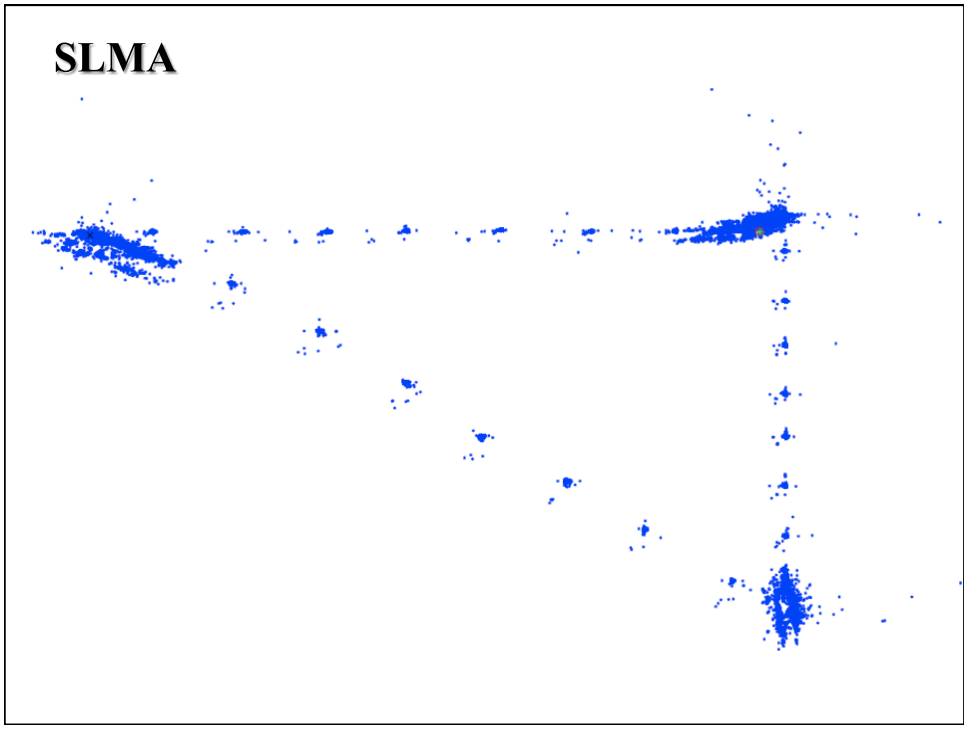
- pulling apart interleaved dynamics, removing noise…

Computers appear to be deterministic. So what about forecasting their dynamics?

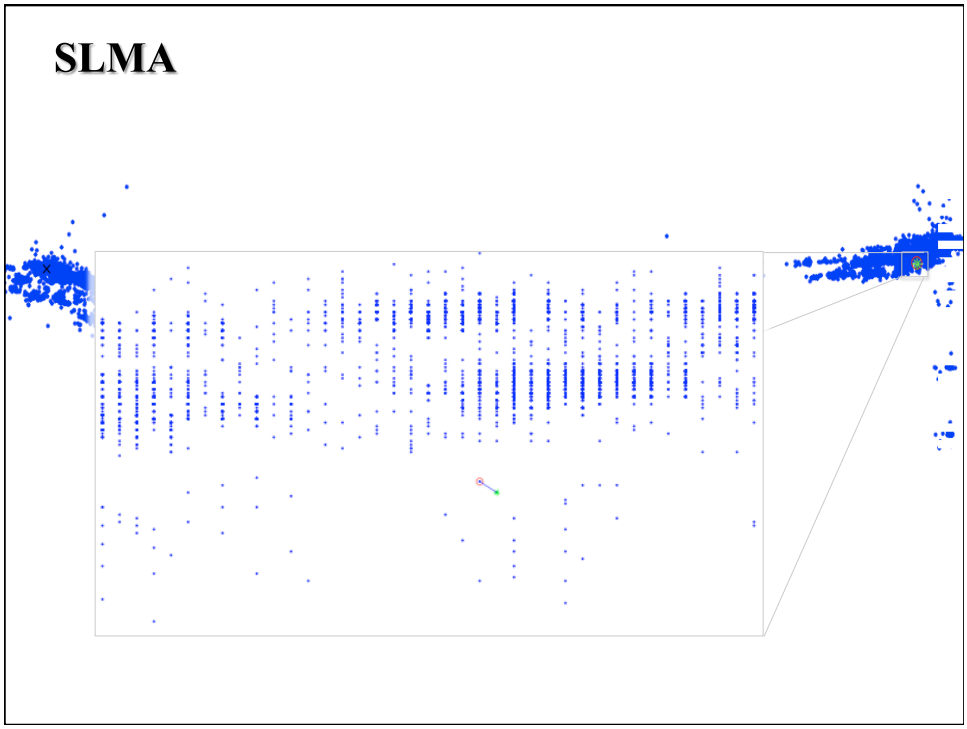
- **Sampled Lorenz Method of Analogues (SLMA)**
  - **Leverages the continuity of the observation function**
  - **Uses nearest-neighbor trajectory**

- ***k*-ball Sampled Lorenz Method of Analogues**
  - **Leverages the continuity of the observation function**
  - **Uses *k*-nearest-neighbor trajectories and averages**

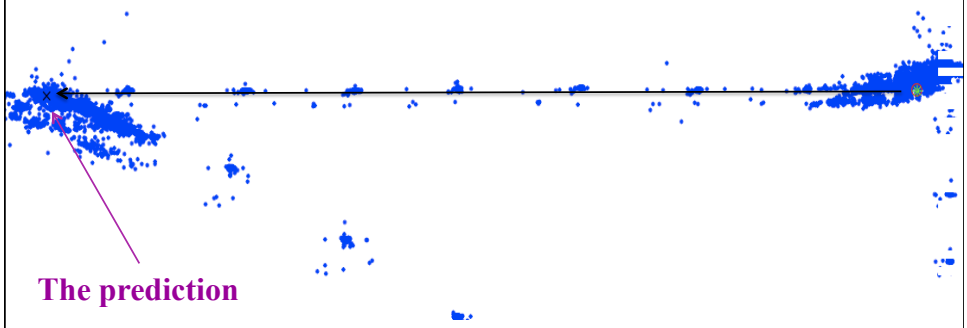
SLMA



SLMA



SLMA

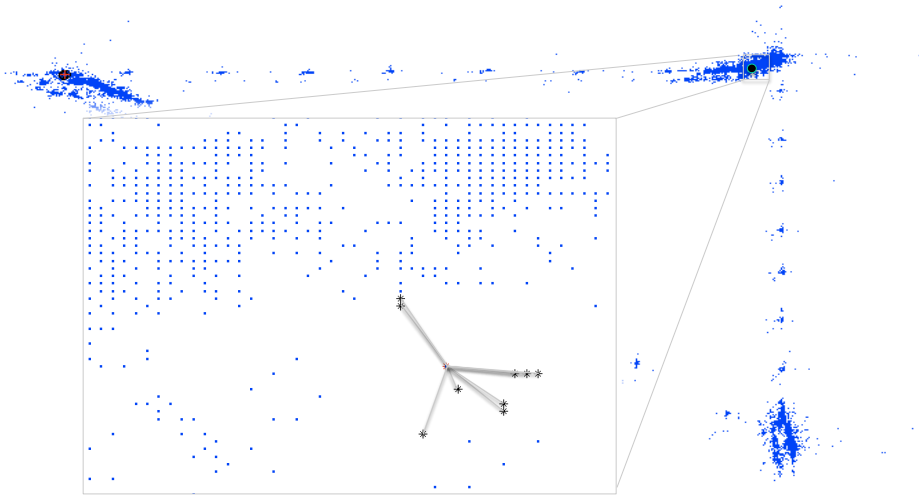


Prediction of computer dynamics: algorithms

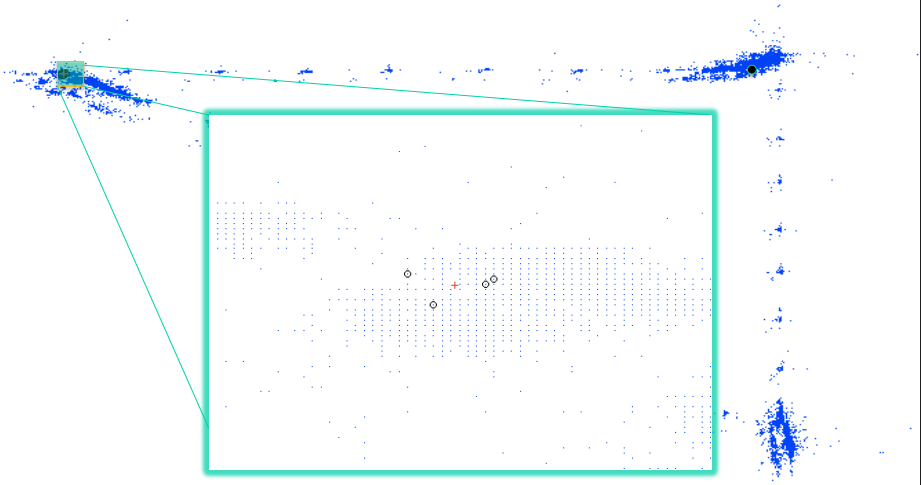
- Sampled Lorenz Method of Analogues (SLMA)
  - Leverages the continuity of the observation function
  - Uses nearest-neighbor trajectory

- ***k*-ball Sampled Lorenz Method of Analogues**
  - **Leverages the continuity of the observation function**
  - **Uses *k*-nearest-neighbor trajectories and averages**

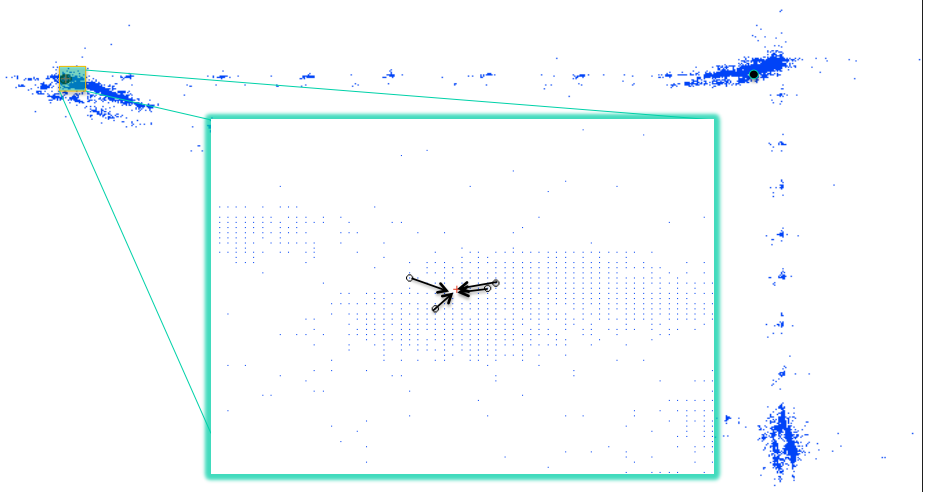
k-ball SLMA



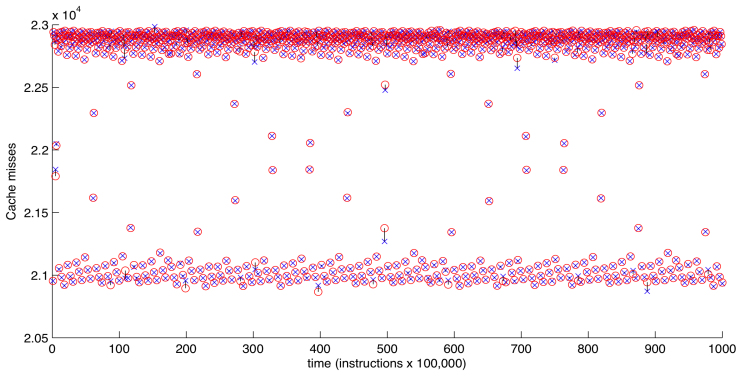
k-ball SLMA



k-ball SLMA



Prediction results



RMSPE = 11.566
(.07% of the average)

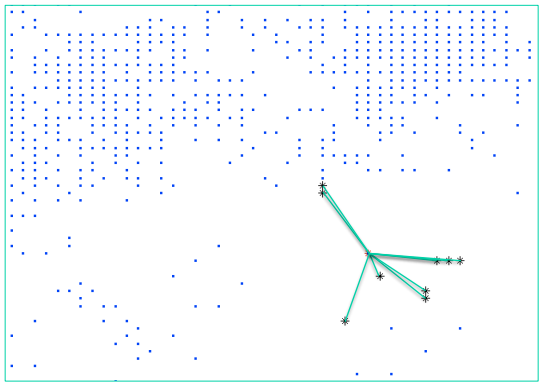
Original Signal
Predicted Signal
Error Between Signals

`column_major`
cache misses

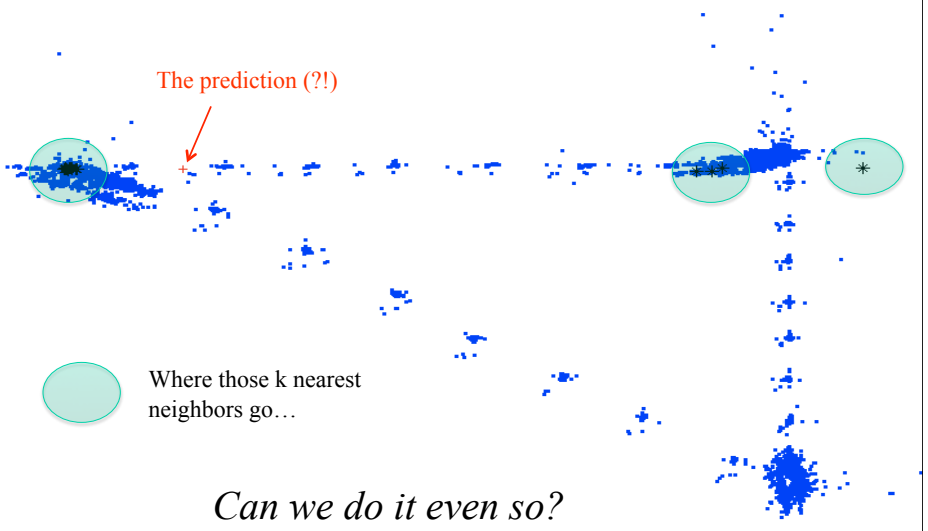
Can we predict *in projection*?

- Delay-coordinate embedding requires significant *post facto* analysis
- This is problematic for prediction
- But low-dimensional projection may destroy the continuity upon which the SLMA algorithms rely…

What can happen if you calculate neighbor relationships in a 2D projection



False-neighbor effects upon prediction



Can we do it even so?

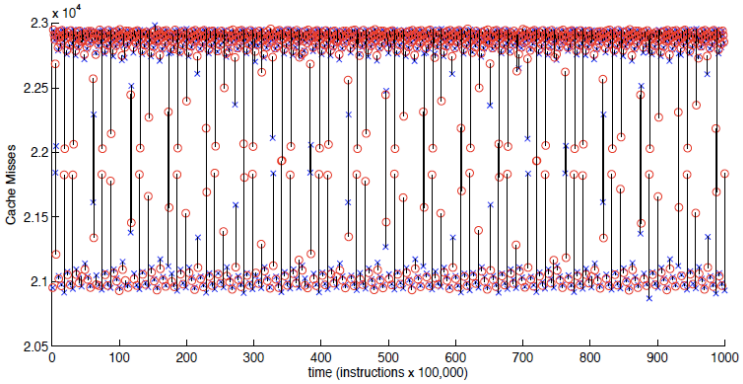
Projection questions

In an m-dimensional dynamical system, there are m-choose-two different 2D projection options.

- Do they produce different prediction results?
- How do those results compare to those computed using the full embedding?

Assess using root-mean squared prediction error (RMSPE)

Prediction results: SLMA on (1,2) projection

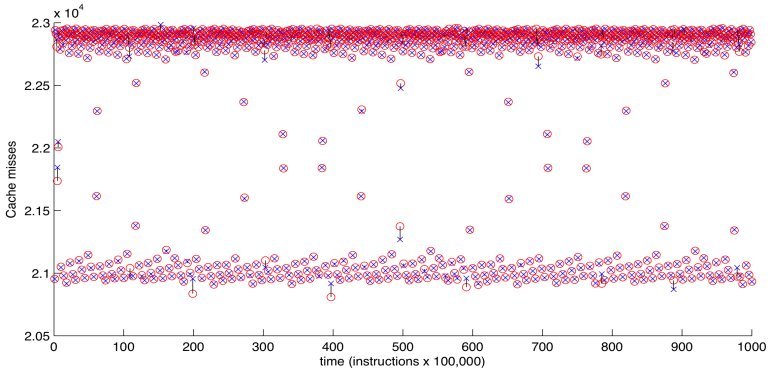


RMSPE=300.7126
(was 11.6 for full embedding)

Original Signal
Predicted Signal
Error Between Signals

`column_major` cache misses

Prediction results: SLMA on best 2D projection



(3,7) projection
RMSPE = 16.0508
(was 300.7126 for the (1,2) embedding)

Original Signal
Predicted Signal
Error Between Signals

`column_major` cache misses

Prediction results: summary

Cache-Miss Rate	SLMA RMSPE	k-Ball SLMA RMSPE
Full Embedding	15.2381	11.5660
Standard Projection	521.4182	300.7126
Best Projection	24.3713	16.0508
IPC	**SLMA RMSPE**	**k-Ball SLMA RMSPE**
Full Embedding	8.1577e-04	5.0020e-04
Standard Projection	8.1577e-04	5.5163e-04
Best Projection	6.9028e-04	5.2564e-04

- Average Cache-Miss Rate ~22,000
- Average IPC ~1.9
- "Standard Projection" = (1,2) projection

See Josh's papers for more about prediction.

What next?

- This is a microkernel experiment; what about real programs?
- Statistical methods to "learn" the right τ (*which is the only free parameter in this method!*)
  - Machine learning
  - *Kalman filters (KF, EKF, EnKF, UKF, …)

So what?

- Computers are nonlinear dynamical systems.
- Traditional analysis tools, which assume linearity and time invariance, don't really work on them (but the computer systems community continues to rely on them almost exclusively, which is starting to cause real problems)
- Nonlinear time-series analysis methods are quite effective…
  - reconstruct the dynamics using delay-coordinate embedding
  - produces surprisingly low-dimensional dynamics (m=12)
  - with clear & consistent dynamical invariants
  - dynamics dictated by hardware and software
  - exploit continuity of reconstructed dynamics to…
    - *filter out noise, do "phase detection"*
    - *find periodic orbits*
    - *make predictions!*

Amer Diwan

`Now at Google`

Todd Mytkowicz

`Now at Microsoft Research`

Zach Alexander

Now at Sybase

Joshua Garland

csel.cs.colorado.edu/~garlanjt