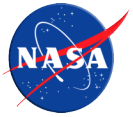
# *Feature Identification and State Estimation for Nonlinear Systems*

**Michael Turmon (JPL/Caltech)**

**Joint work with:**
**Michael Chin, Jeff Jewell (JPL)**
**Michael Ghil (UCLA/ENS Paris)**

**UCLA IPAM Workshop**
**May 26, 2010**

JPL



# Problem Context

Problems in ocean and atmosphere prediction with:

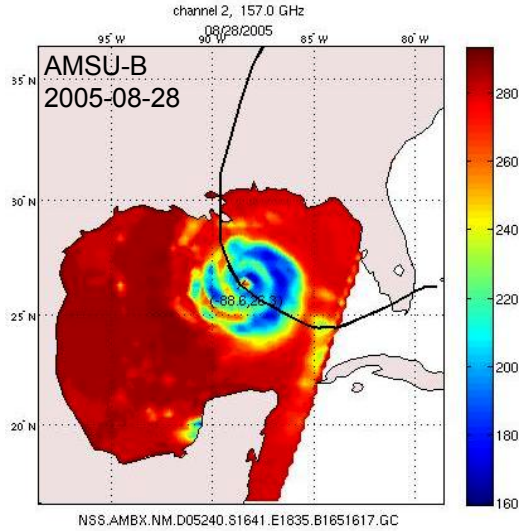
Known, gridded, nonlinear dynamics…

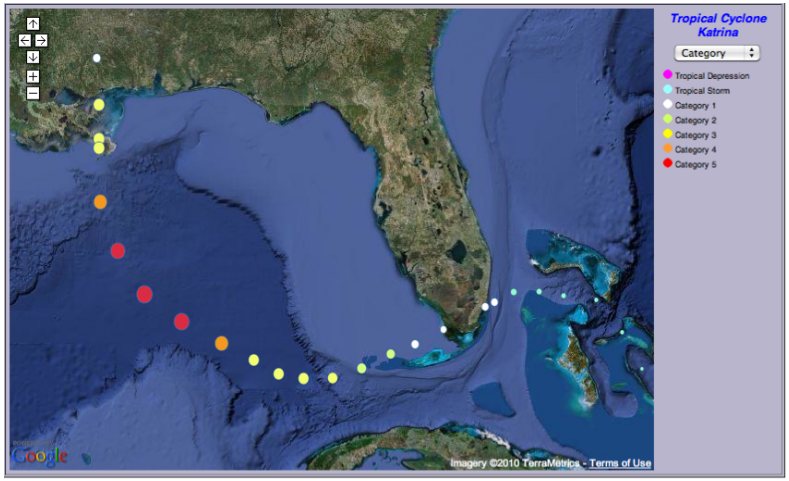
… producing complex emergent (Lagrangian) features…

…where the behavior of the features is predictable.

System evolution is subject to chance.

Small perturbations can cause huge differences in outcome and risk (Lorenz 1963 studied weather).



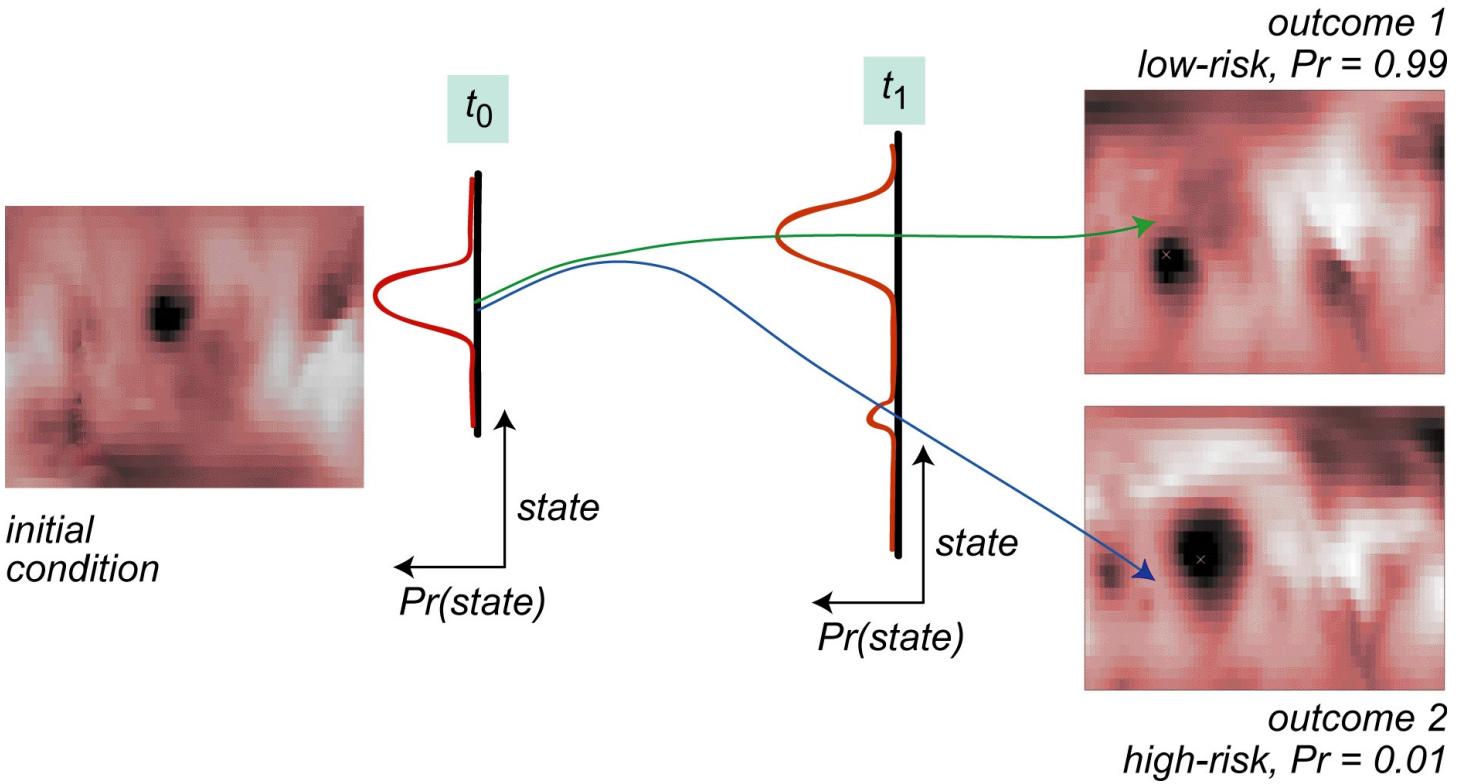
# Problem Context

The problem of divergent outcomes has been addressed using ensembles of independent, parallel evolution paths

- Comprehensible
- Less pervasive infrastructure changes to implement
- Many probes to spot highly-unlikely outcomes

We describe learning an object-level dynamical model to guide large-scale gridded simulations.

# Touchstone: Conditional Forecasting



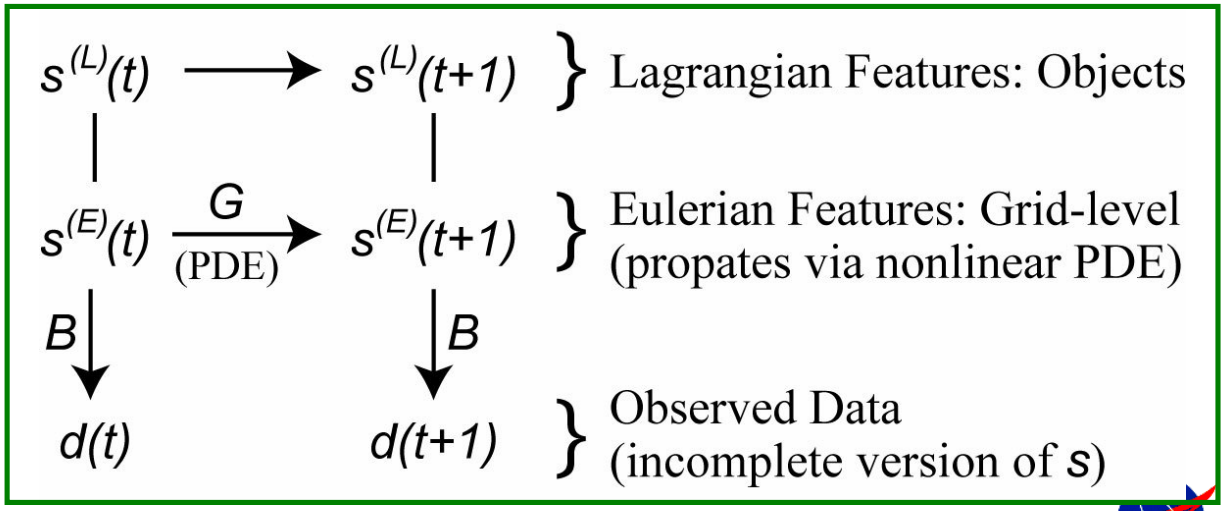

- Nonlinear evolution produces highly divergent final states
- Knowing Lagrangian dynamics allows physically-realistic end-scenarios to be posed, and their probabilities computed
- Conventional approaches require dart-throwing: huge ensembles of forward simulations to capture one end scenario of interest

# Technical Tools

- Learning of dynamics from example object paths
  - Fit state-based models, like Kalman filters, to observational data
- Fine-tuning of object-based (Lagrangian) paths in the gridded (Eulerian) space using Markov chain Monte Carlo (MCMC)
  - Conditional forecasting:  two-sided boundary value problem
  - MCMC fine-tunes candidate paths according to physical model

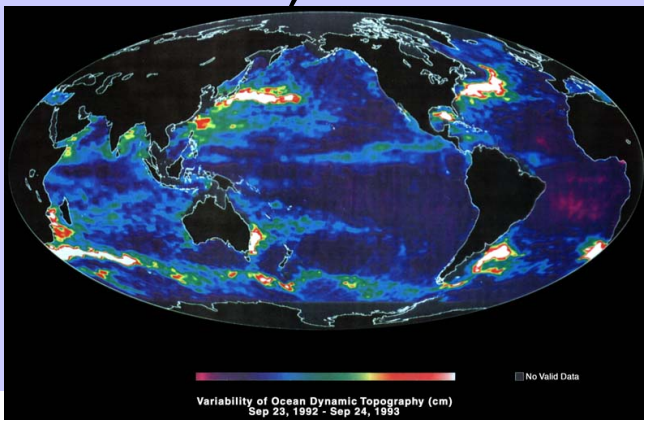
# Features Constrain Grid Dynamics

- Conventional dynamics are grid-level (Eulerian)
- Alternately, an object-level (Lagrangian) dynamic parallels the Eulerian dynamical system
- New tools exist to model dynamics on Lagrangian spaces
  - Faster computations due to fewer variables in object-level space
  - More intuitive domain for end users (fronts, vortices)
- *Develop object templates* and *learn dynamics* for these features



# System Decomposition

- Identify and track vortices (simulation data)
  - Tracking in shallow-water simulation exhibiting mid-latitude meso-scale vortices *(next slide)*
  - Similar vortices prominent in remote sensing data

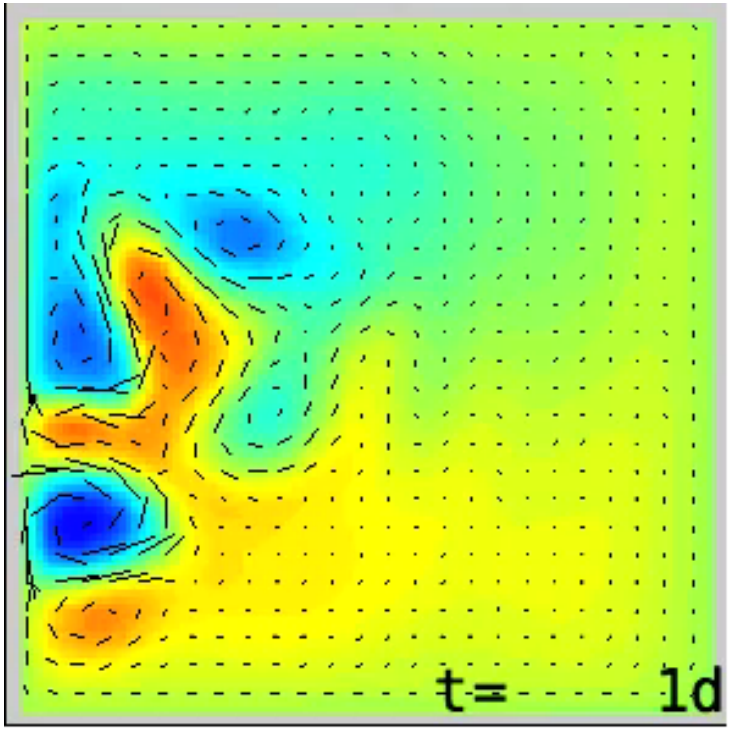



- Define Lagrangian object templates
  - Parameterized blobs with scale and orientation state variables representing pressure field

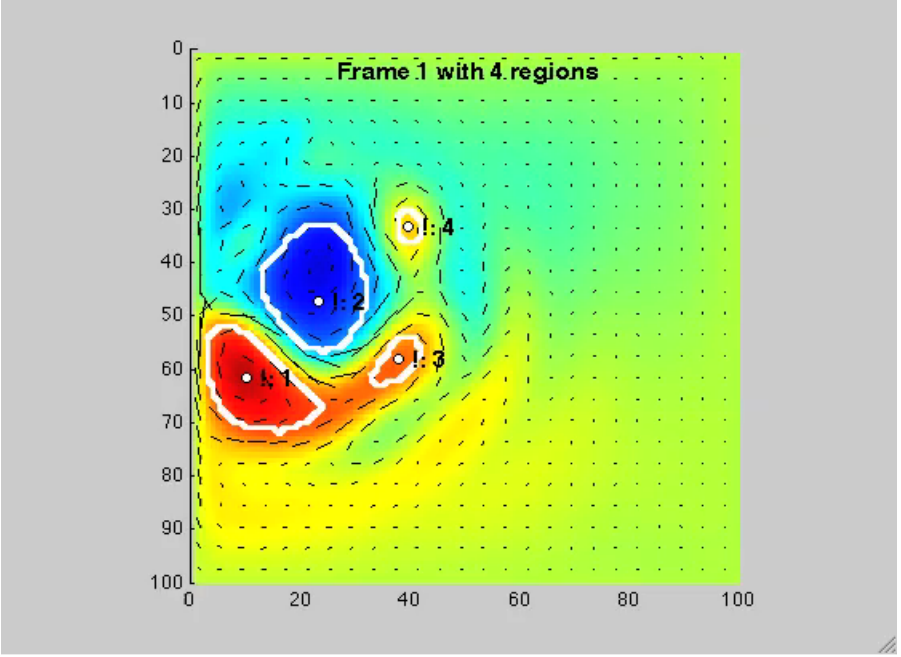
- Learn forward dynamics (simulation data)
  - Model object state variables based on tracks above
  - Perform data assimilation on tracked objects

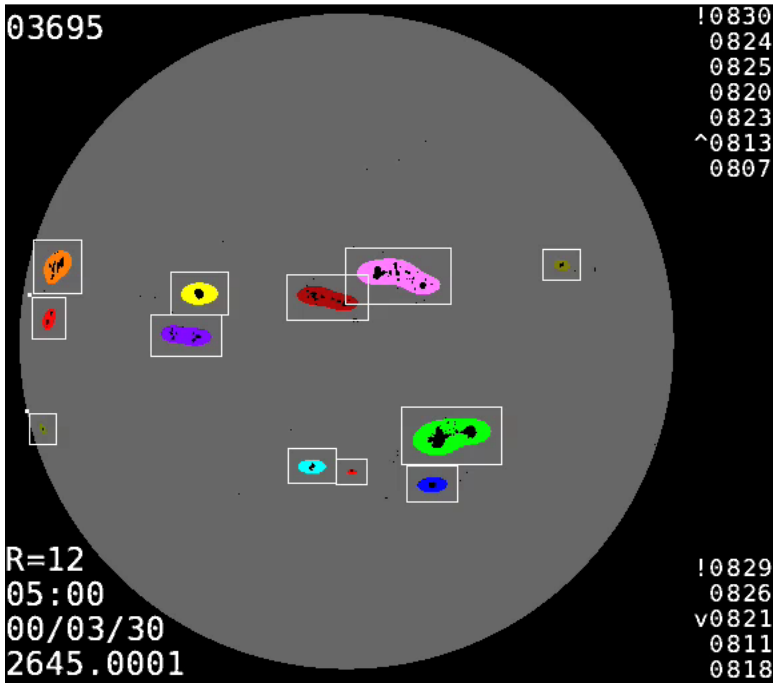
# Model: MICOM

- Single-layer *shallow water equation* dynamics exhibiting “double gyre” circulation pattern
  - Horizontal velocity and geopotential fields (no thermodynamic variables, temperature or salinity)
  - Arakawa C grid with leap-frog time-stepping (the advection scheme used by MICOM)
  - 2000 x 2000 km domain, 20 km grid resolution (100 x 100 x 3 state)
  - Used to study vortex (“ring”) formation in Gulf Stream



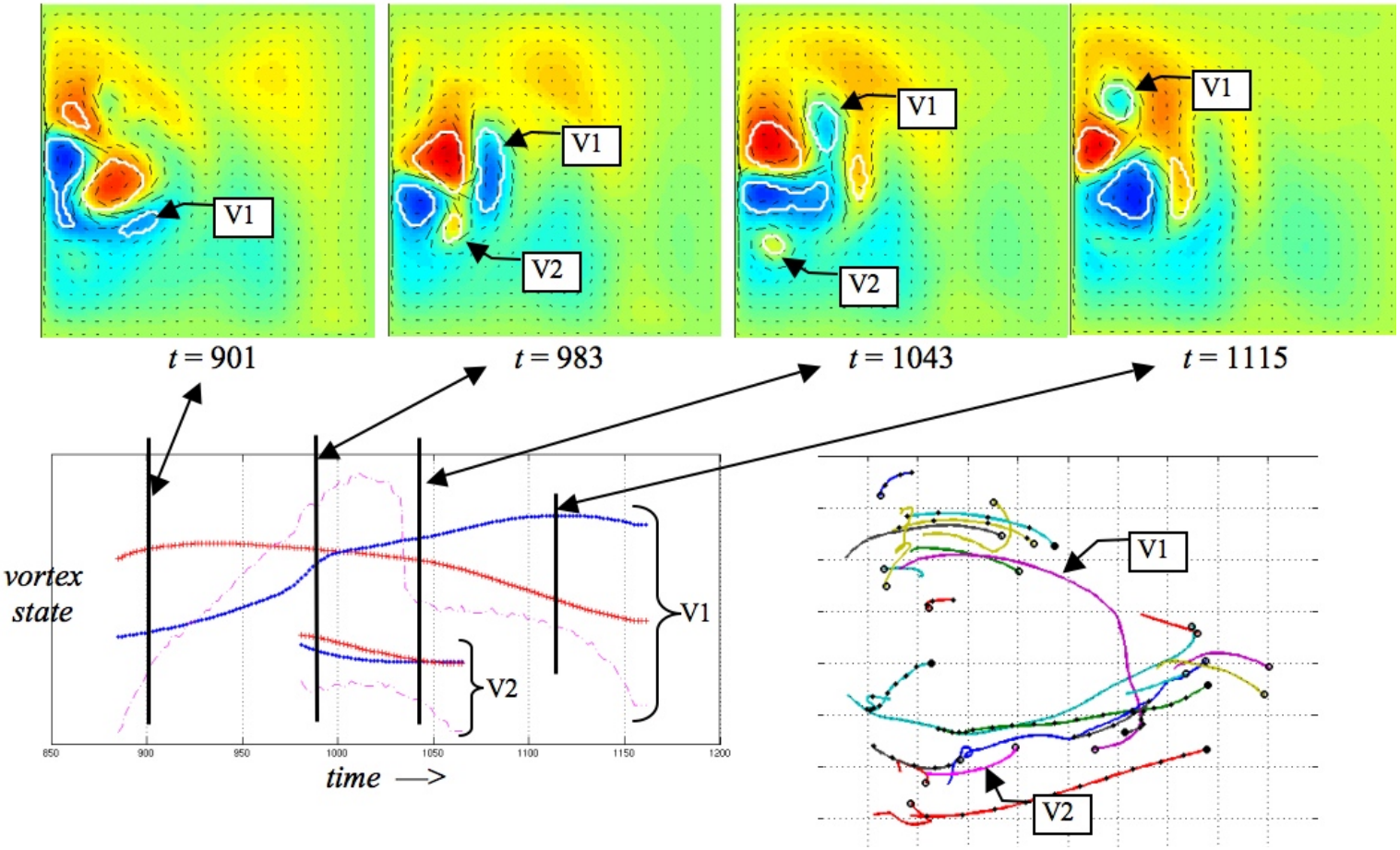
# Tracking objects




- Tracking algorithm
  - Object: threshold exceeded by Gaussian-smoothed pressure field
  - Association: single-link-most-likely tracker
  - Association criterion is area-of-overlap
- MICOM vortices:  top left (~100 objects over 10000 days)
- Solar active regions: top right (~10000 objects over 12 years)

# Quantifying the Dynamics



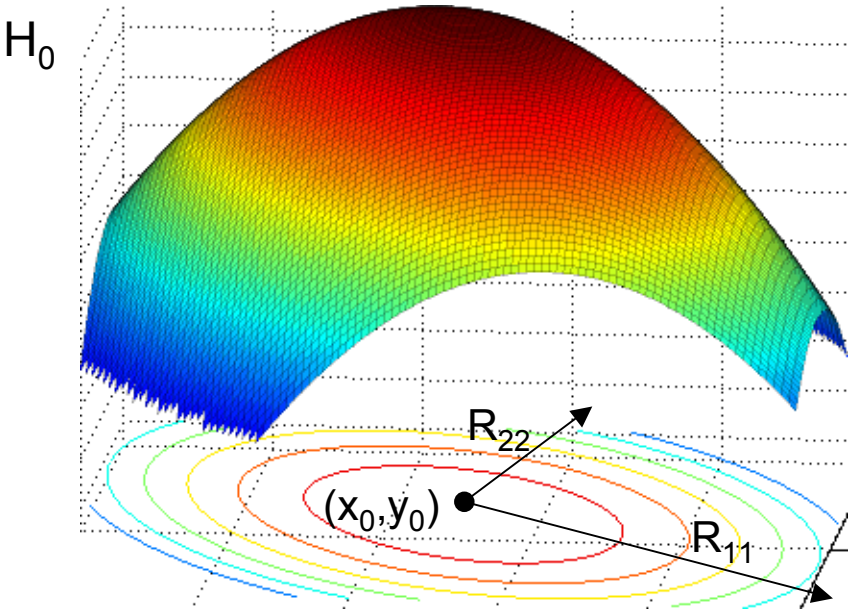

- Tracked vortices (top)
- Lagrangian state (position, size) of V1 and V2, bottom left
- Two object families evident, bottom right

# System Decomposition

- Identify and track vortices (simulation data)
  - Tracking in shallow-water simulation exhibiting mid-latitude meso-scale vortices
  - Similar vortices prominent in remote sensing data
- Define Lagrangian object templates
  - Parameterized blobs with scale and orientation state variables representing pressure field *(next slide)*
- Learn forward dynamics (simulation data)
  - Model object state variables based on tracks above
  - Perform data assimilation on tracked objects

# Templates for Lagrangian Features

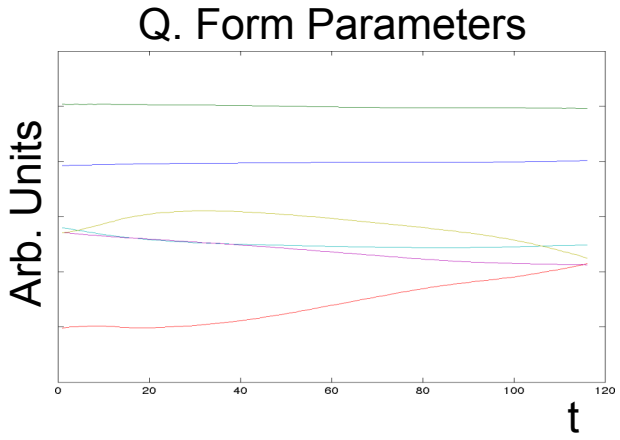
- Approximate the geopotential as a quadratic form

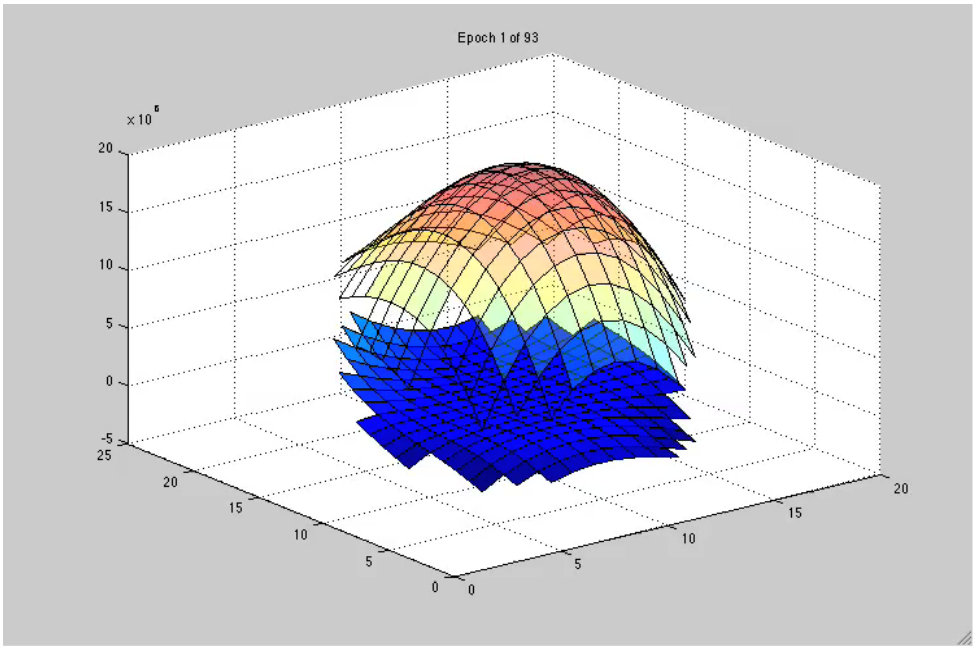



$$h(x,y) = H_0 + \Big([x\,y] - [x_0\,y_0]\Big)\,R\,\Big([x\,y] - [x_0\,y_0]\Big)^{\mathsf{T}}$$

  - Magnitude ($H_0$), position ($x_0$,$y_0$), scale, orientation (R)
  - Six free parameters describe feature
  - Time series of these six parameters gives full path

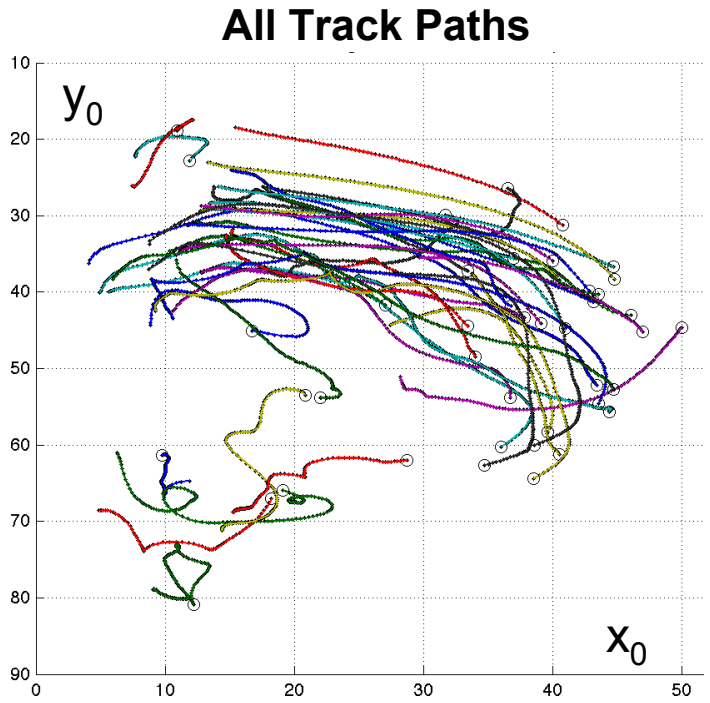
# Templates in Practice

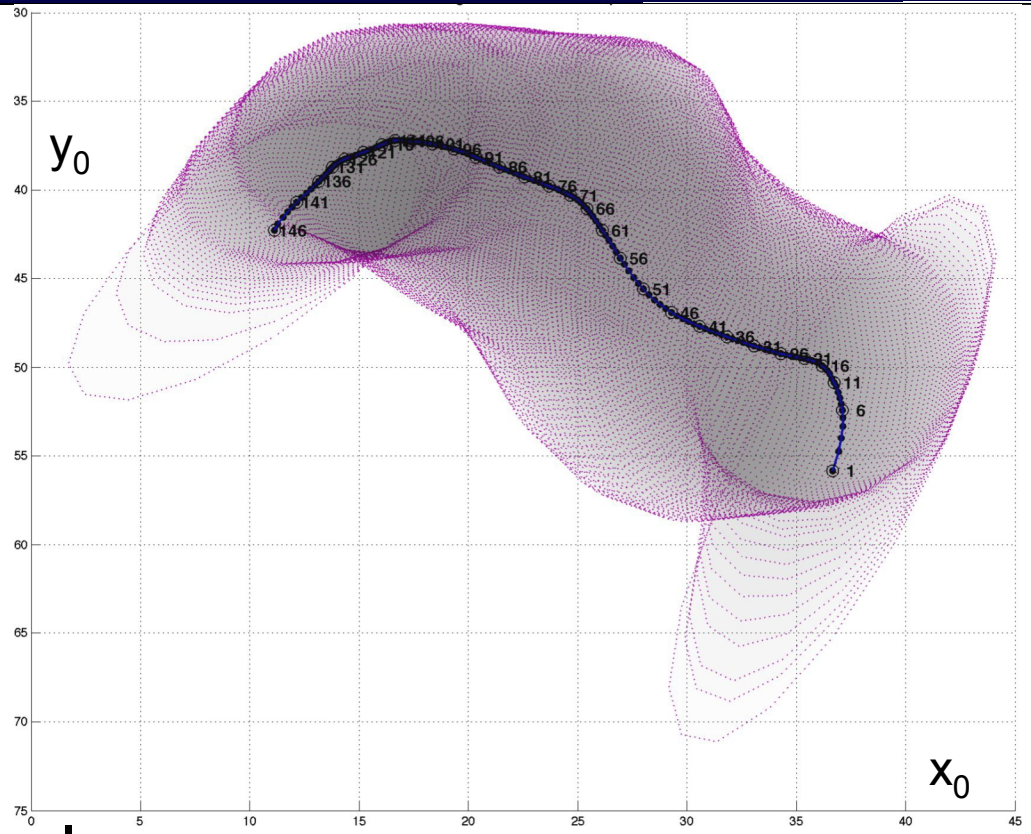



- Use quadratic form parameters (left) to reconstruct pressure field (right)
- Movie shows typical pressure field and reconstruction error
  - Error typically less than 10%

# Ensemble of objects




- Typical paths and scales
  - Evolution of $(x_0,y_0)$ and *R* is smooth
- There are subclasses of typical objects

# System Decomposition

- Identify and track vortices (simulation data)
  - Tracking in shallow-water simulation exhibiting mid-latitude meso-scale vortices
  - Similar vortices prominent in remote sensing data
- Define Lagrangian object templates
  - Parameterized blobs with scale and orientation state variables representing pressure field
- Learn forward dynamics (simulation data)
  - Model object state variables based on tracks above *(next slide)*
  - Perform data assimilation on tracked objects

# Models for Tracks

- Quantitatively model evolution of Lagrangian features
  - Ability to draw plausible tracks from this model
- Learn a Kalman filter model from tracks
  - Kalman filter model for six-dimensional parameterized feature H(t).

$$z(t+1) = A\,z(t) + B\,u(t) + v(t) \qquad v(t) \sim N(0, Q)$$
$$H(t) = C\,z(t) + D\,u(t) + w(t) \qquad w(t) \sim N(0, R)$$

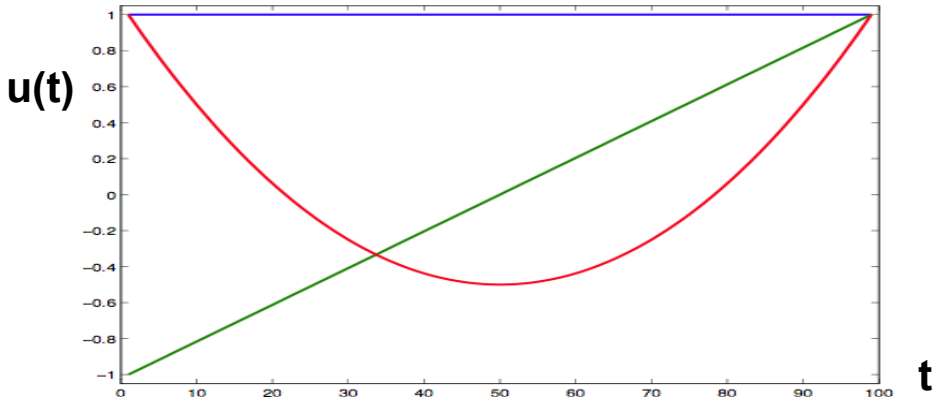
  - Hidden variables z(t) of dimension 2
  - Fit by maximum-likelihood by EM algorithm

# Models for Tracks (II)

- Kalman filter model

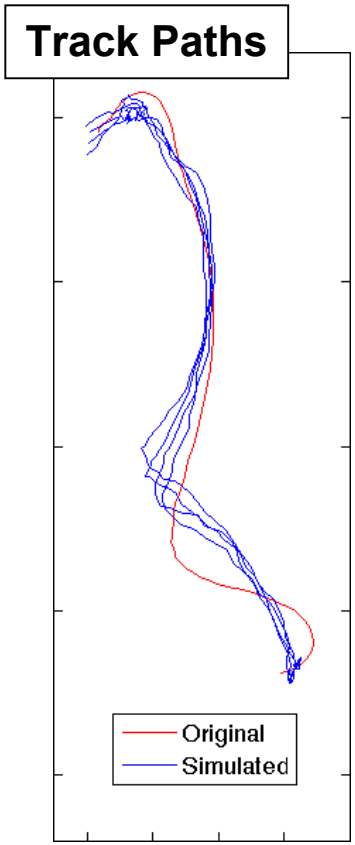
$$z(t+1) = A\,z(t) + B\,u(t) + v(t) \qquad v(t) \sim N(0, Q)$$
$$H(t) = C\,z(t) + D\,u(t) + w(t) \qquad w(t) \sim N(0, R)$$

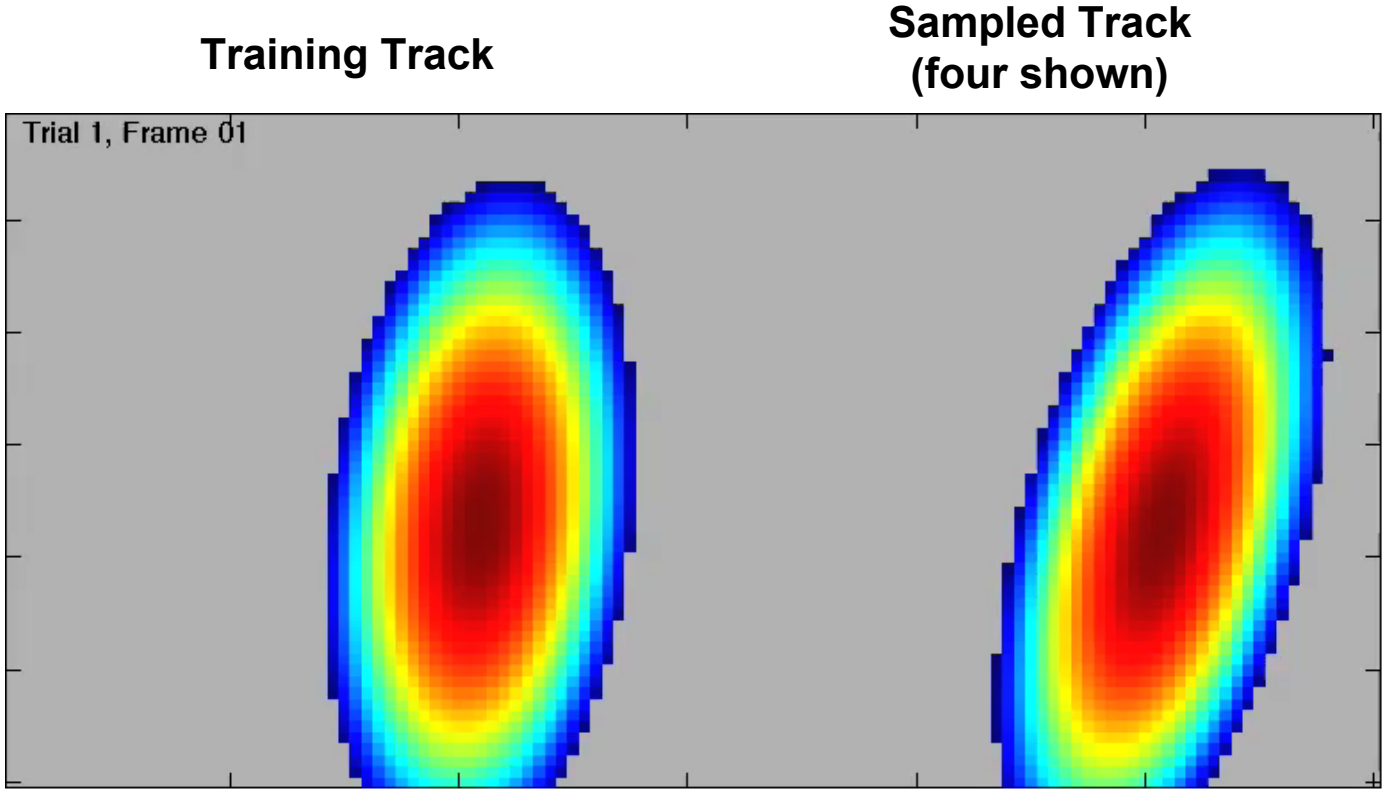
  - The six-dimensional output H(t) collects magnitude, location, x/y scale, and eccentricity
  - Scale is non-negative, so we fit the log of scale
  - The exogenous inputs u(t) model growth and decay



# Learning Models for Tracks

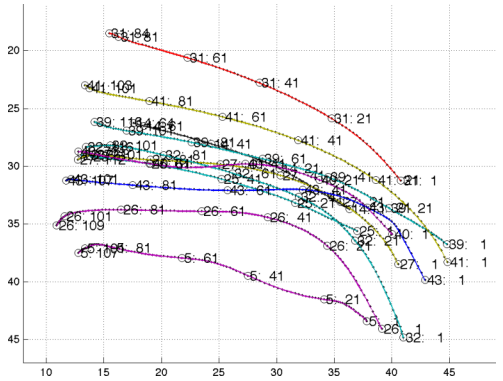
- Quantitatively model evolution of Lagrangian features
  - Kalman filter model for six-dimensional parameterized feature
  - Shown below as trained with one track (left: path; right: shape)



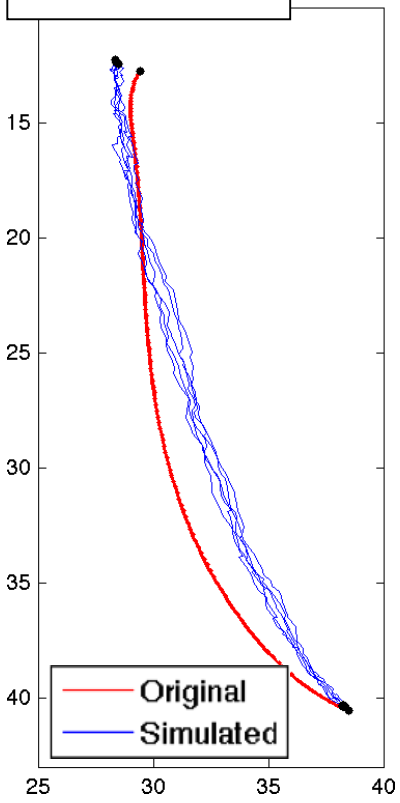



# Learning Models from Ensembles

- Train on an ensemble of 11 similar tracks (right)
  - Same Kalman filter track model setup
  - Condition on *endpoints of another track* (left: path; right: shape)

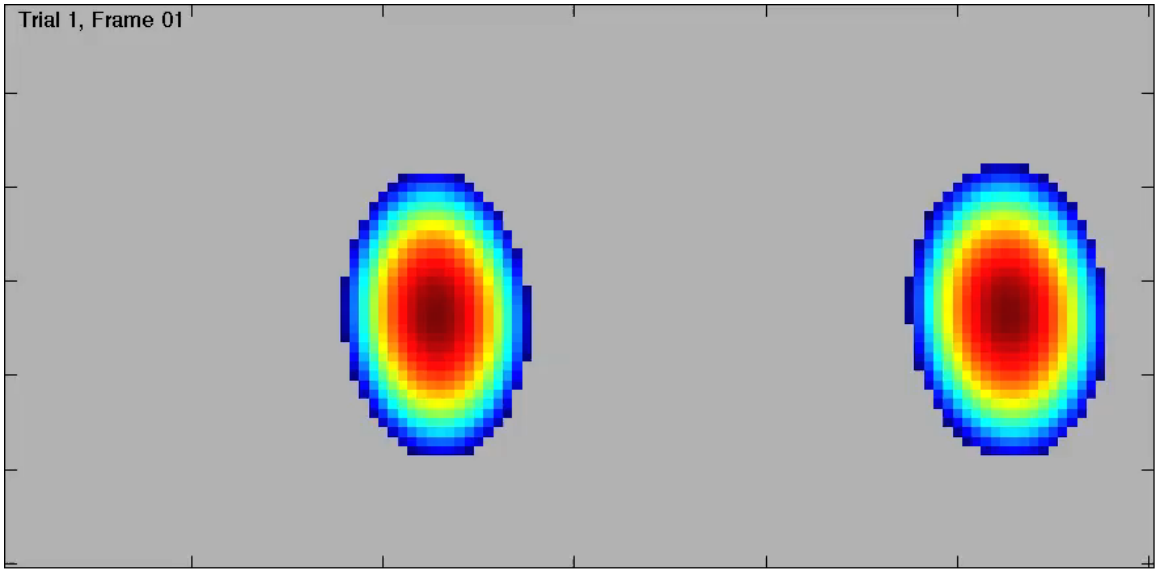


**Track Paths**



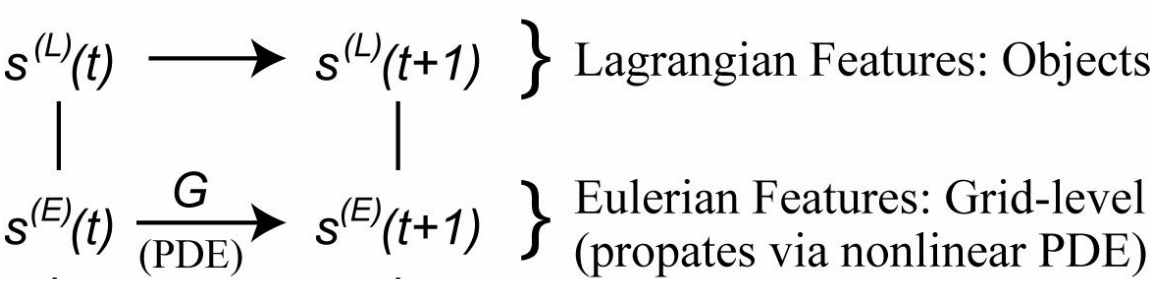

**Track Conditioned Upon**

**Sampled Track (five shown)**



# Adapting the Path

- This is enough to interpolate coarse-scale dynamics
  - Single-sided or double-sided interpolation
- Use MCMC for physics-based fine tuning of path

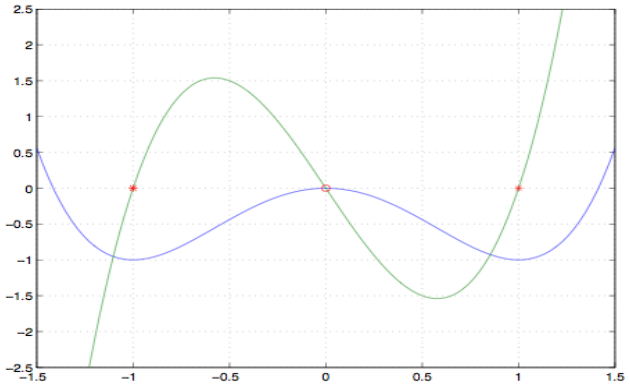



- The MCMC will start from the coarse representation and adapt a path that conforms to the physical model
- Describe how this works for a simpler system

# Double-well System

- Particle in a potential well given by

$$f(x) = -2x^2 + x^4 \quad \text{and} \quad g = f'$$




- Trivial climate system
  - Miller et al. 1994,1999, Pham, 2001, Kim et al., 2003

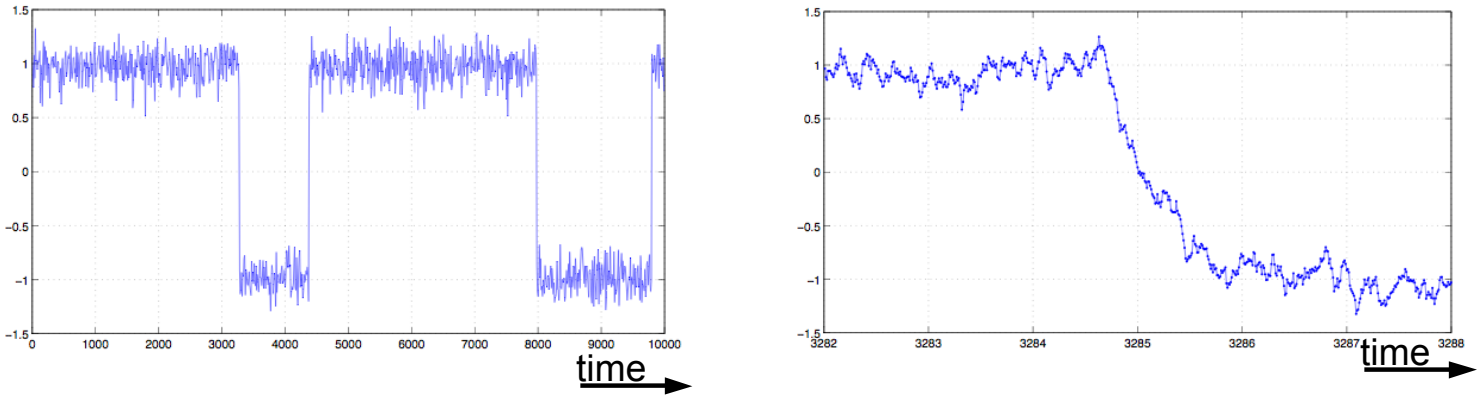
- Continuous diffusion with width-$\delta$ discretization:

$$x_{t+1} = x_t - \delta g(x_t) + r_t \qquad r_t \sim N(0, \delta\kappa^2)$$

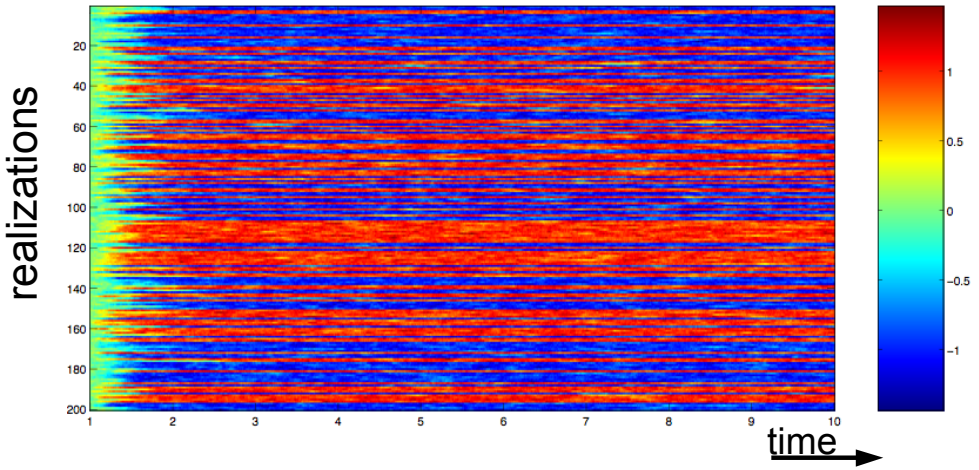
$$y_t = x_t + s_t \qquad s_t \sim N(0, \sigma^2)$$

# Sample Paths

- $\kappa^2 = 0.24$, $\delta = 0.01$, mean switching time ~ 4600:




- Stacked ensemble of 200, starting at $x_1 = 0$

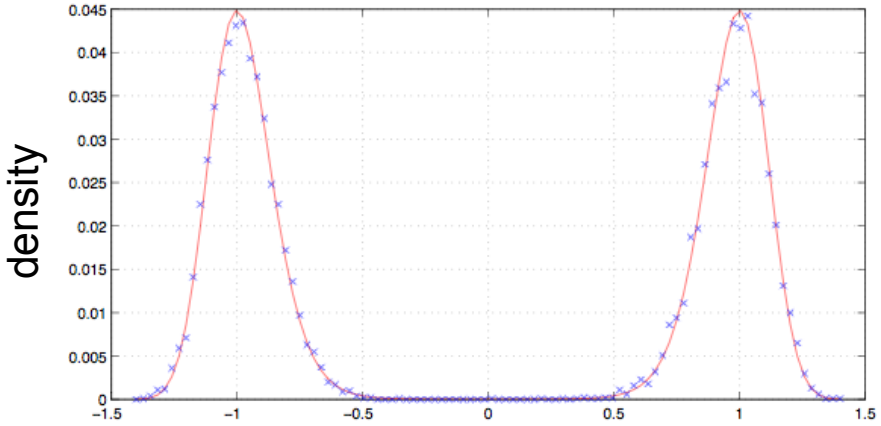


# Limit Densities

- Limiting density

$$p_\infty(x) \propto \exp(-2f(x)/\kappa^2) \qquad f(x) = -2x^2 + x^4$$

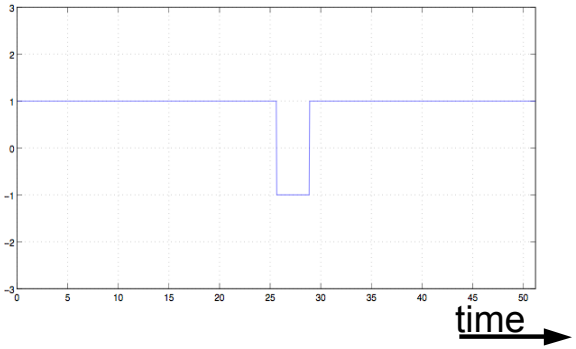
- Histogram:
  - Ensemble of 10000 taken at T = 1000
  - This is ordinary Monte Carlo

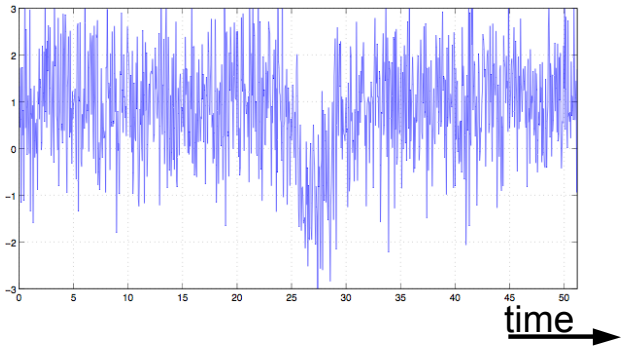



- Posterior by Markov Chain Monte Carlo

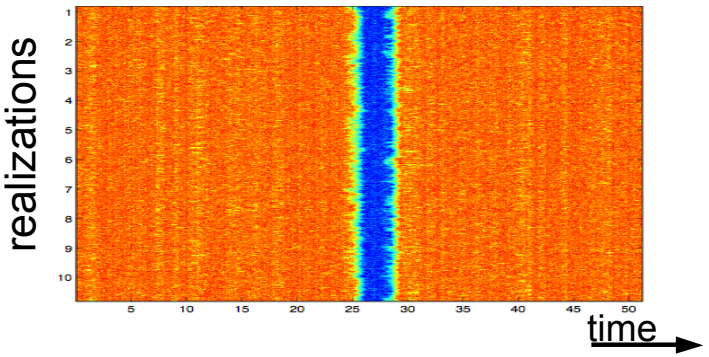
# MCMC Posteriors: Low Noise

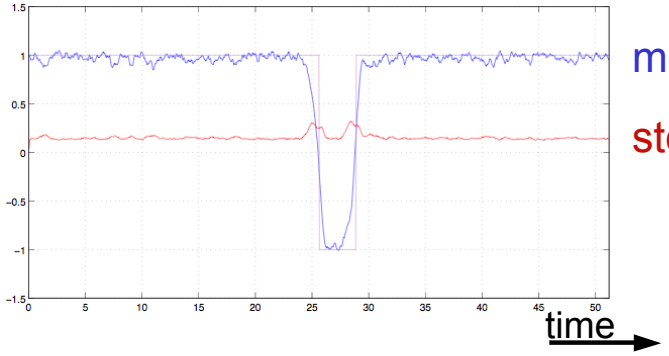
- True $x_t$ with observed data ($\sigma = 1$)




- Stack of samples from the posterior (left)
  Posterior mean and variance versus time (right)



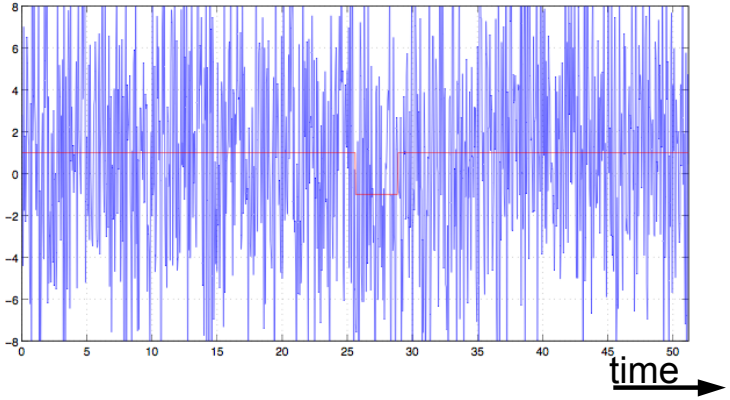



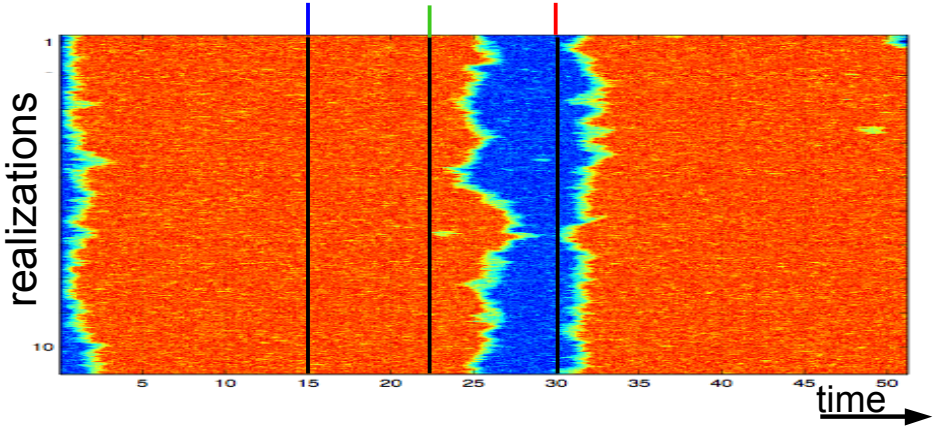

mean

std. dev.

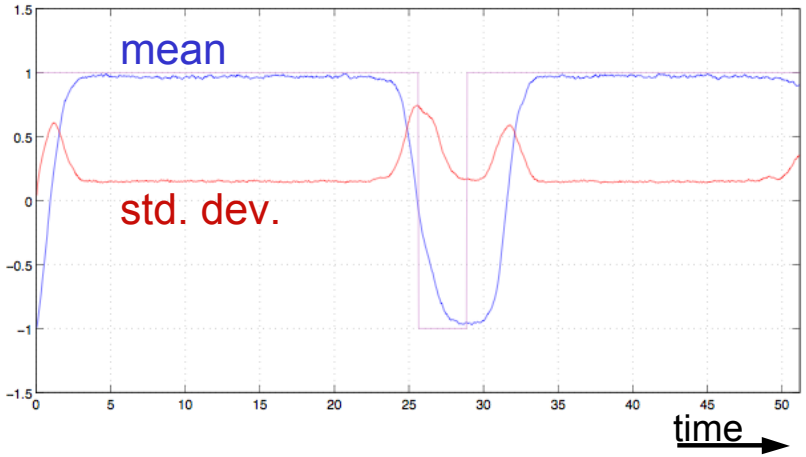
# MCMC Posteriors: High noise

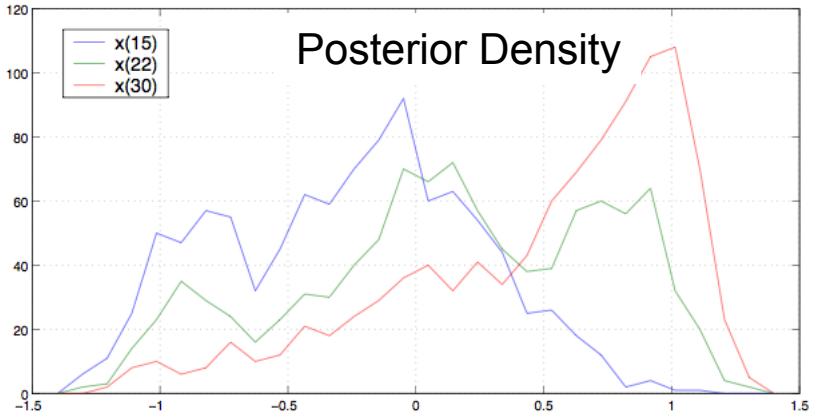
- Now $\sigma = 5$ versus $\sigma = 1$:



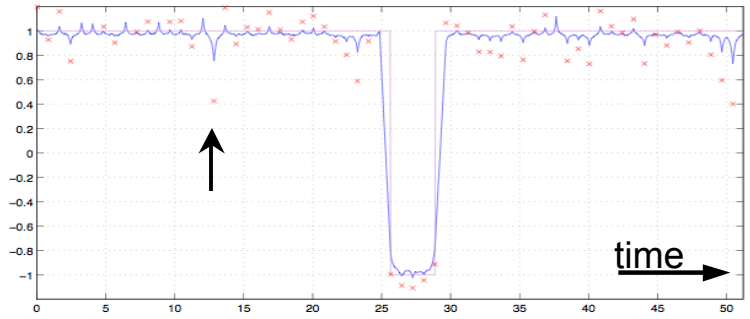

- Posterior density estimates (right) show non-Gaussian behavior

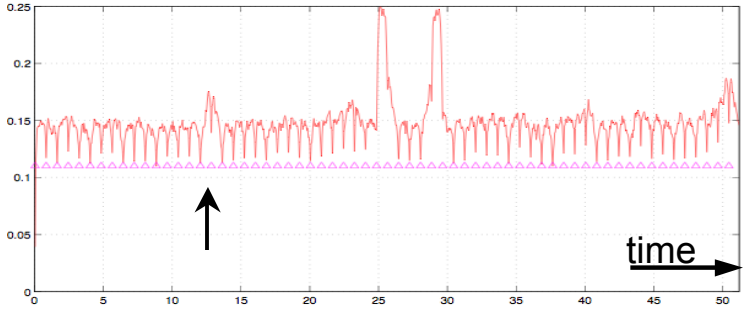


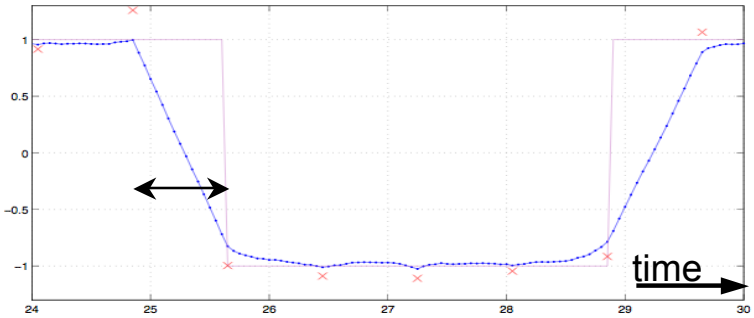
# MCMC Posteriors: Sparse Sampling

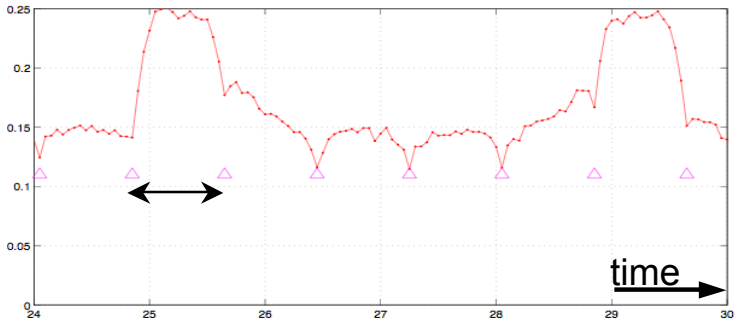
- Moderate noise $\sigma$ = 0.2, but downsample 16x:




**Zoom**



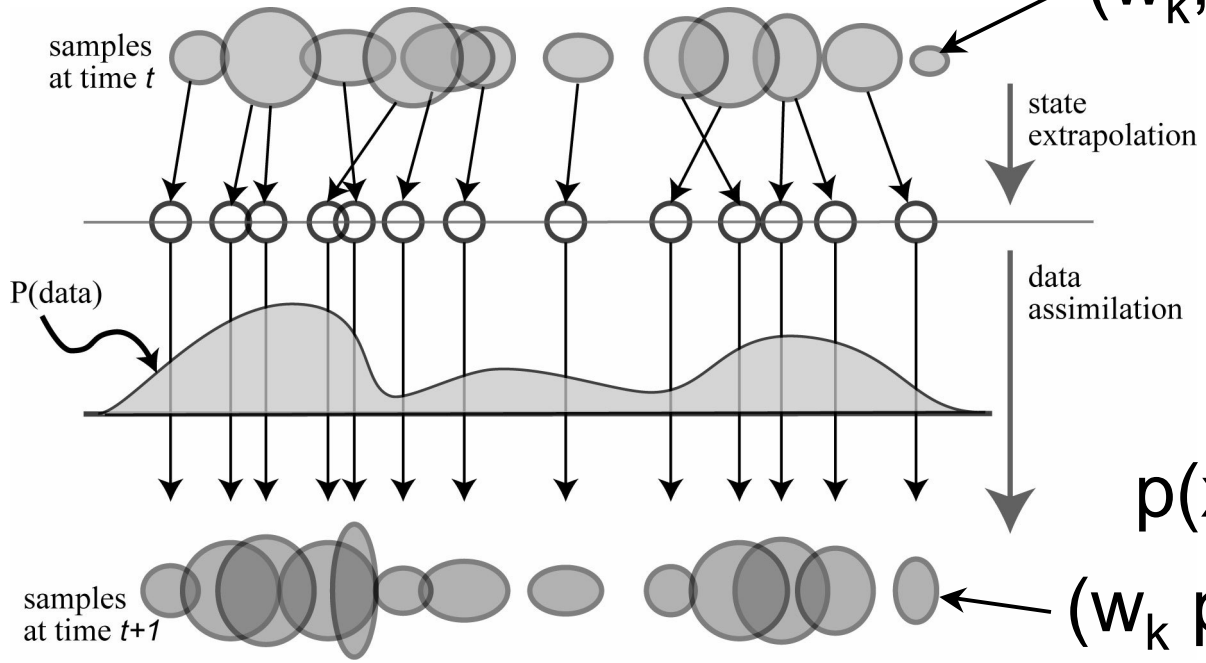

- Tradeoff between data assimilation and path smoothness
- Gaussian process noise disallows fast jumps

# Particle Filter

- MCMC represents the global posterior with a single, random sample
  - Sample evolves across simulation epochs
- Sequential Monte Carlo represents an instantaneous density with a set of random samples (“particles”)
- This set is pushed forward in physical time
- Particle filter can take the place of MCMC in finding posteriors

# Particle Filter Data Assimilation

- Posterior ~ weighted sample
  $p(x_t \mid y_{1:t}) \sim \{ (w_k, x^{(k)}) \}_{k=1\ldots K}$

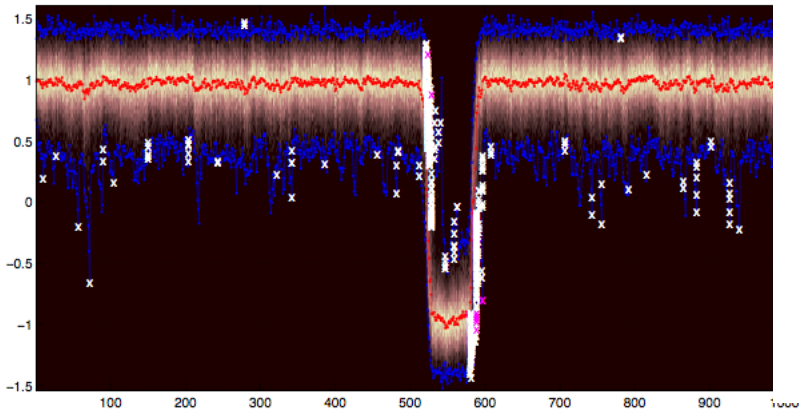


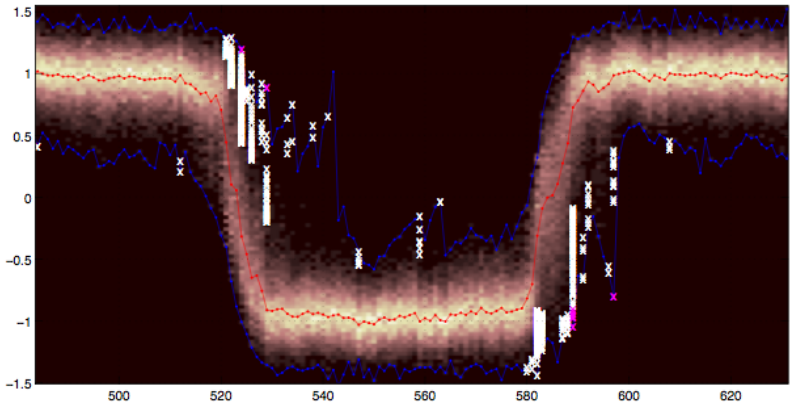

$p(x_t \mid y_{1:t}) \sim (w_k, x^{(k)})$

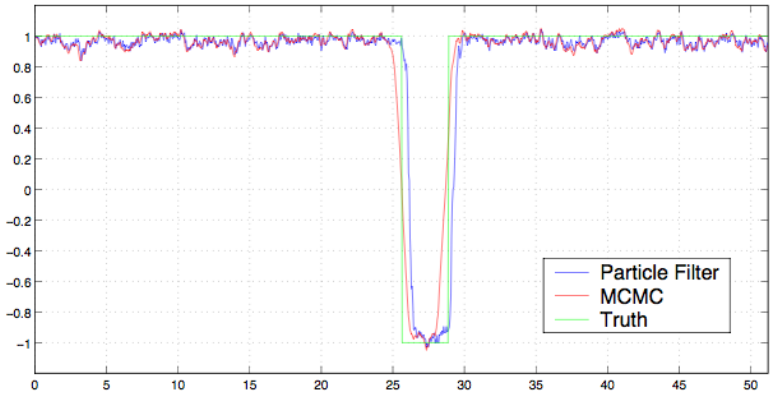
$p(x_{t+1} \mid y_{1:t+1}) \sim (w_k\, p(y_{t+1} \mid x^{(k)}), x^{(k)})$

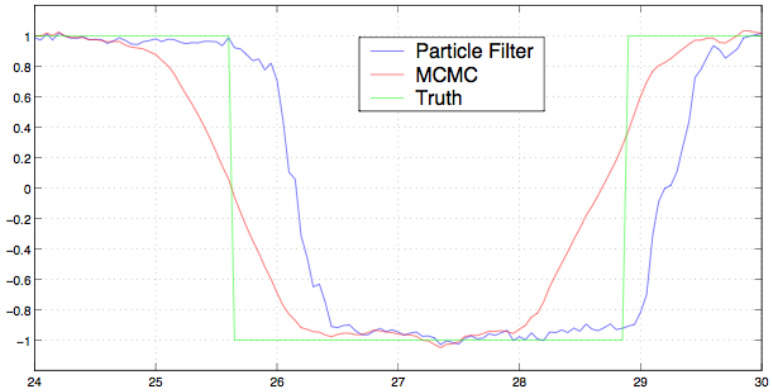
# Particle Filter for Double Well System

- Particle filter vs. MCMC
  - Loss of particle diversity signaled by white crosses



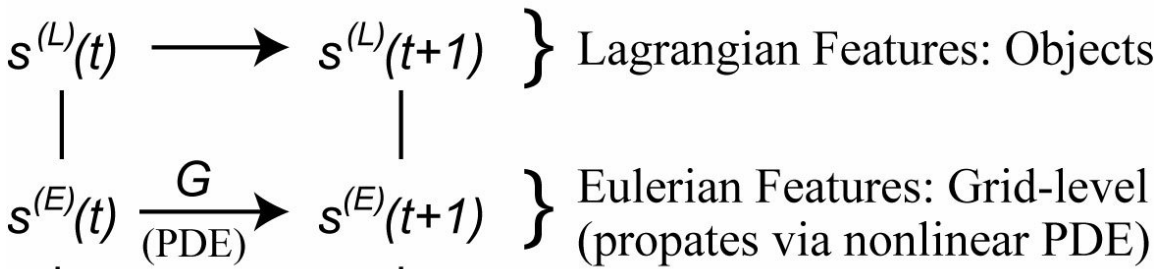



# Future Work

- Use MCMC for physics-based fine tuning of path
  - Evaluate computational efficiency




- Begin to handle object interactions
  - Objects governed by sparsely-parameterized per-region dynamical models should repel each other