Nonparametric Adaptive Control

Recent Advances in Model Reference Adaptive Control: Theory and Applications

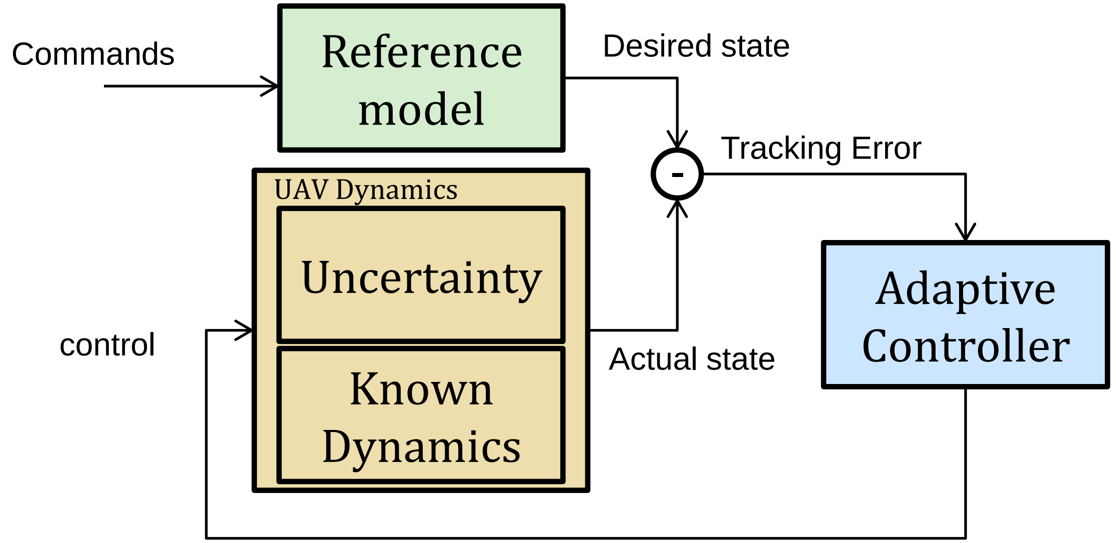
ORGANIZERS

ANTHONY CALISE, *Georgia Institute of Technology*
ERIC JOHNSON, *Georgia Institute of Technology*
TANSEL YUCELEN, *Georgia Institute of Technology*
GIRISH CHOWDHARY, *Massachusetts Institute of Technology*

AIAA Guidance, Navigation, and Control Conference

11-12 Aug 2012, Minneapolis, MN

Model Reference Adaptive Control (MRAC)



- Want to make the uncertain system behave like the reference model
- Need quantifiable metrics on performance and stability
- Need to enable agile and health-aware UAV operation
- Approach:
  - Simultaneously track the reference model and learn the uncertainty

Existing Relevant work in MRAC

- Classic work in Model reference adaptive control (MRAC) Narendra, Ioannou, Aström, Boyd etc.
  - Not guaranteed to be robust in presence of noise
  - Not guaranteed to learn the uncertainty without persistent excitation (PE)
- Classic robustifying modifications to MRAC: σ-mod (Ioannou 84), e-mod (Narendra 86), projection based adaptation
  - Not guaranteed to learn the uncertainty
- L_1 adaptive control (Cao, Hovakimyan 08)
  - Not concerned with *learning* uncertainty, tracking performance guaranteed through point wise uncertainty domination
- Intelligent excitation (Cao 07):
  - Learns the uncertainty at the cost of added control effort
- Modern modifications to MRAC: Q-modification (Volyanksyy 09), C-MRAC (Lavretsky 2009), Least squares adaptation (Ngyuen 2006), DF-MRAC (Yucelen 2010),….
  - Require PE operation to guarantee convergence: added control effort
  - Do not provide quantifiable (exponential) stability guarantees

Approximate Model Inversion (AMI) Based MRAC

Plant

$$\dot{x} = f(x, u),$$

- $x \in \Re^n$: system states
- $u \in \Re^m$: control inputs (multiple input), controllable
- f satisfies conditions for a unique solution, unknown

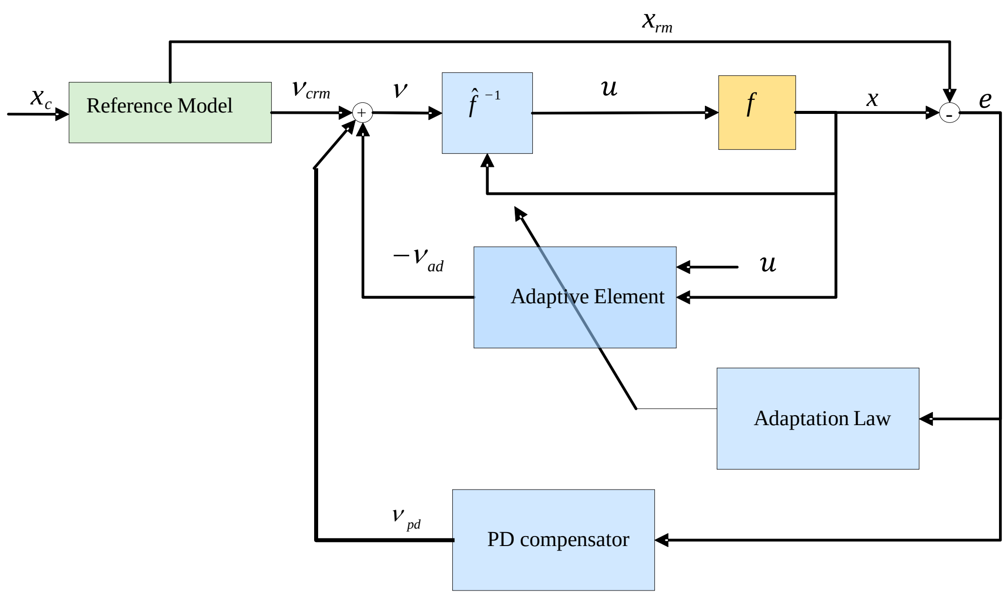
Problem Statement: Model Reference Adaptive Control

Design a control law $u(t)$ such that the plant tracks the reference model

$$\dot{x}_{rm} = f_{rm}(x_{rm}, r)$$

- $x_{rm} \in \Re^n$: system states
- $r \in \Re^l$: Exogenous inputs
- f_{rm} bounded input bounded output stable and continuously differentiable

Overview of Inversion Based MRAC



- Approximate inversion model $\hat{f}$
- Design a pseudocontrol ν to minimize the tracking error: $e = x - x_{rm}$

Modeling error $\Delta \in \Re^n$

$$\dot{x} = \hat{f}(x,u) + \boxed{\left[f(x,u) - \hat{f}(x,u)\right]}$$

- Combined pseudo-control action:

$$\nu = -Ke + \dot{x}_{rm} - \nu_{ad}$$

Choice of adaptive element

Structured Uncertainty

Given that there exists a constant matrix W^* and known basis functions $\sigma(x,u) \in \Re^m$ such that

$$\Delta(x,u) = {W^*}^T \sigma(x,u)$$

Then, choose adaptive element:

$$\nu_{ad} = W^T \sigma(x,u)$$

Unstructured Uncertainty (more general)

Given $\Delta(x,u)$ is continuous and defined over a compact set, then :

$$\Delta(x,u) = {W^*}^T \Phi(x,u) + \tilde{\epsilon}$$

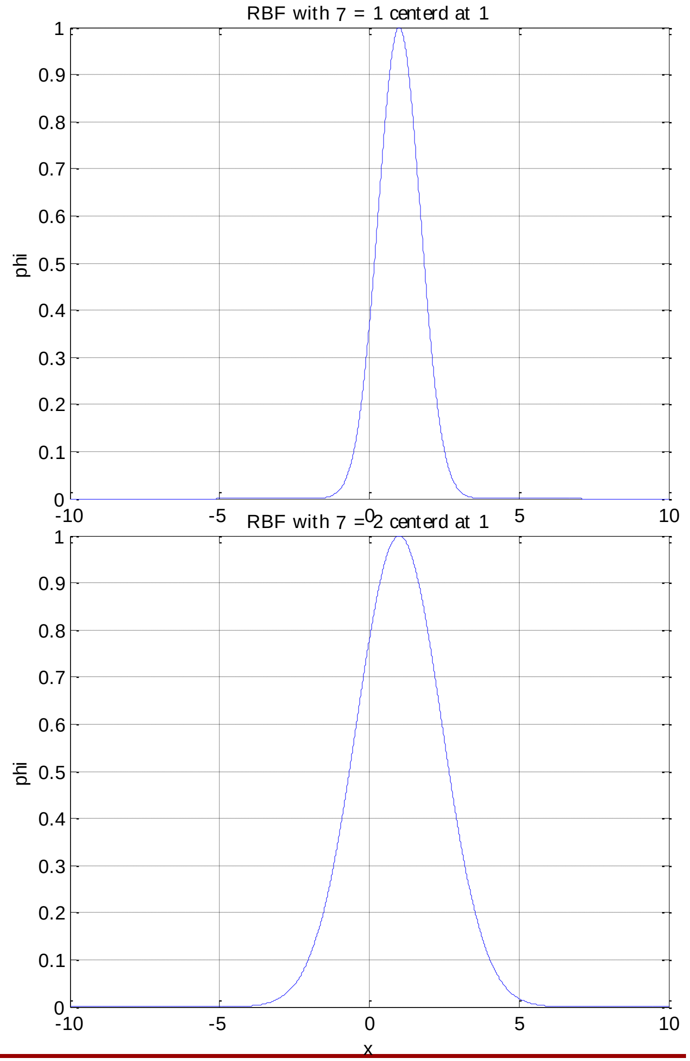
where $\Phi(x,u)$ are basis function of a Neural Network adaptive element:

$$\nu_{ad} = W^T \Phi(x,u)$$

Radial Basis Function NN

- Radial Basis Functions (RBFs) are Gaussian Kernels
- Select n_2 centers x_{c_j}
- Select a width parameter
- Add a bias term, then $\Phi(\bar{x}) \in \Re^{n_2+1}$

$$\Phi(\bar{x}) = \begin{bmatrix} b_w \\ e^{\frac{-||x_{c_1} - \bar{x}||^2}{2\mu^2}} \\ \vdots \\ e^{\frac{-||x_{c_{n_2}} - \bar{x}||^2}{2\mu^2}} \end{bmatrix}$$



Universal Approximation with RBFs

- Given fixed number of RBFs with a fixed width μ there exists an ideal set of weights W^* such that

$$\bar{\epsilon} = \sup_{\bar{x} \in \mathrm{D}} ||{W^*}^T \Phi(\bar{x}) - \Delta(z)||$$

- $\bar{\epsilon}$ can be made arbitrarily small given sufficient number of RBFs
- RBF NN $\nu_{ad} = W^T(t)\Phi(\bar{x})$
- Ideally, we would like $W(t) \rightarrow \mathrm{W}^*$
- Traditionally, adaptive control is happy with keeping e bounded

RBF adaptive law

- The following adaptive law guarantees that the tracking error stays bounded in a compact neighborhood

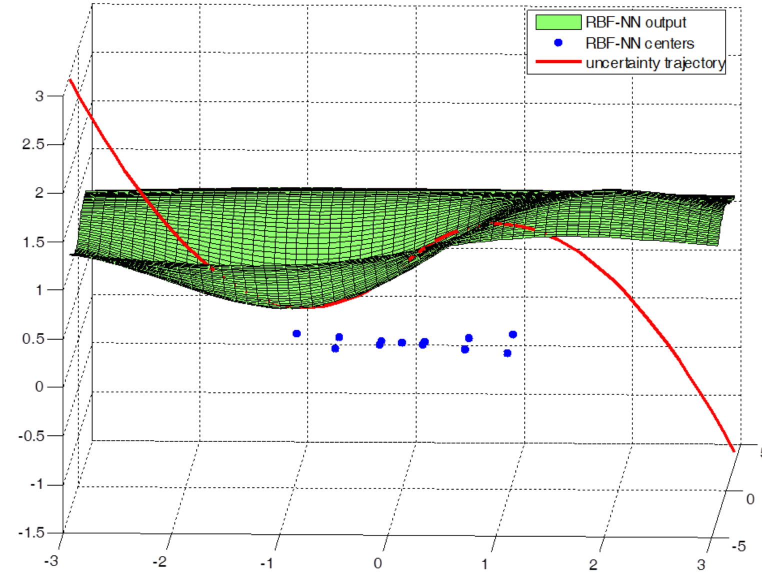
$$\dot{W} = -\Gamma\Phi(\bar{x})e^T PB$$

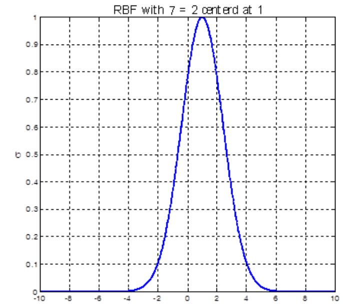
- Can use $\sigma -$ mod, or $e -$ mod or proj to guarantee boundedness in presence of noise:

$$\dot{W} = -\Gamma\Phi(\bar{x})e^T PB - \kappa W$$

Limitations of RBF based neuro-adaptive control

- RBFs: linear-in-the-parameters, easier to analyze and easier to tune
- Current approach: pre-allocate RBFs based on prior domain knowledge
- Problem: How to assign the centers of the RBFs (e.g. in fault tolerant control)
- Problem: How many RBFs needed
- Problem: How to guarantee long-term learning





(left) A one-dimensional RBF kernel with center $z_i = 1$ and width $\mu = 2$

No matter how good your adaptive control algorithm, if the adaptive element's representation is bad, you are out of luck

The problem of locality

- Nardi (2000) and Sundararajan (2002) tune RBF centers using the instantaneous tracking error
  - Rank-1 (greedy) update, moves all centers together in one direction
  - Ultimate boundedness guaranteed, but no guarantee that centers are actually moved to improve the representation
- Sanner and Slotine (1996), used heuristics to assign centers across a bounded domain: must assume bounded operating domain
- We offer a Reproducing Kernel Hilbert Space based approach that tackles the problem at its root

Reproducing Kernel Hilbert Spaces

- A mercer kernel $k(x_1, x_2)$ is a continuous, symmetric, positive semi-definite function for $x_1, x_2 \in D \subset \Re^n$
- Can think of a kernel as a measure on how different two points are
- (Mercer) There exists a Hilbert space H of functions and a mapping $\psi: D \to H$
- $\psi(x)$ takes you from a finite dimensional *input* space to an infinite dimensional *feature* space
- $\psi(x)$ need not be unique and is often unknown

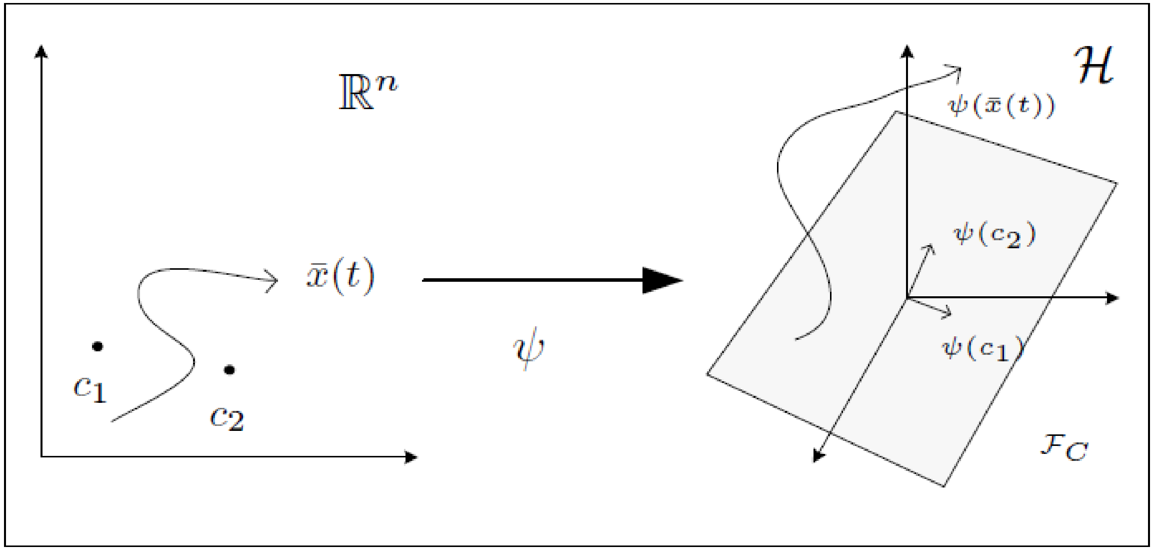
$$k(x_1, x_2) = \langle \psi(x_1), \psi(x_2) \rangle_H$$

PE signals and the RKHS

- The linear subspace generated by l RBF centers c_i

$$F_c = \left\{ \sum_{i=1}^{l} \alpha_i k(c_i, .) : \alpha_i \in \Re \right\}$$

- A trajectory in the $\Re^n$ is mapped to the feature space H
- The nonlinear trajectory is transported to the feature space H by the mapping ψ
- In H the trajectory is linearly parameterized over F_c

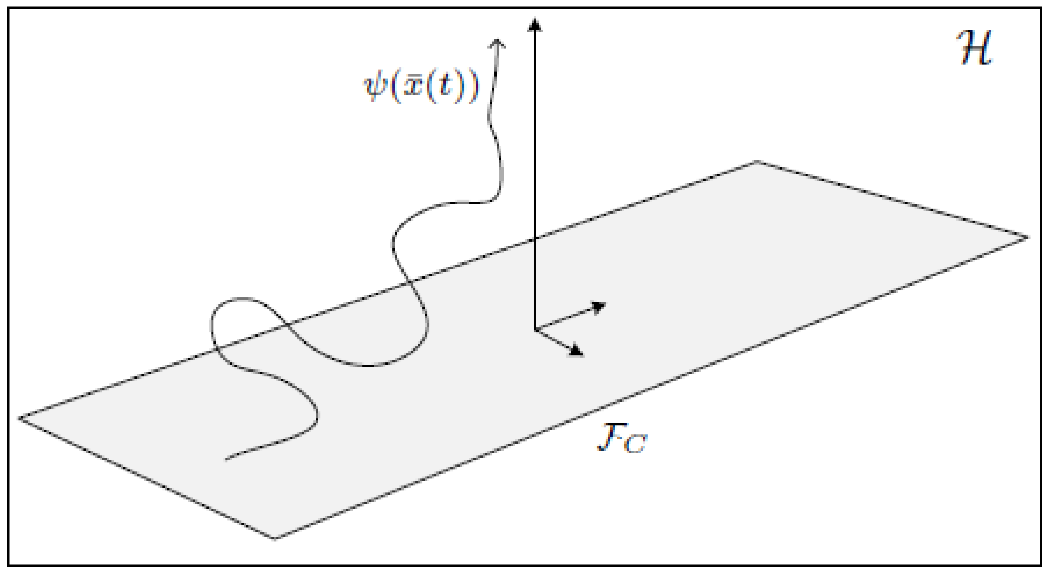


PE signals and the RKHS

Lose PE if trajectory is out of the range of F_C

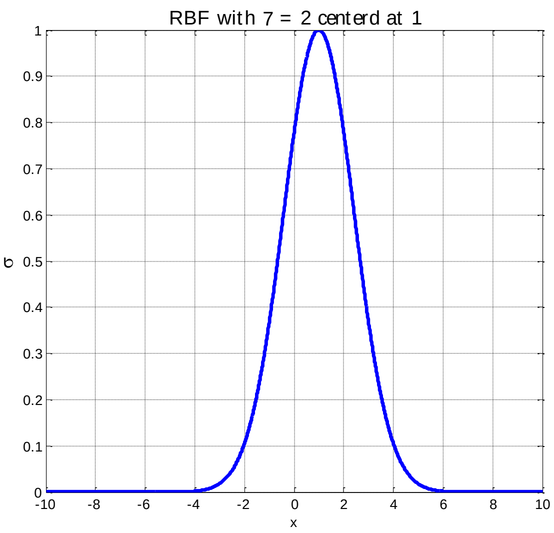
Let x evolve according to $\dot{x} = f(x, u)$, then, if there exists a time t_f s.t. the mapping $\psi\big(x(t)\big)$ is orthogonal to F_c for all $t \geq t_f$ then the signal $\Phi\big(x(t)\big)$ is not PE

- Key realization: Just because $x(t)$ is PE, doesn't mean $\Phi\big(x(t)\big)$ is PE too! (Thm 2)
- If $x(t)$ stops evolving, $\phi\big(x(t)\big)$ stops being PE (Thm 3)



What’s happening

- PE of the RBFs is dependent on center location
- If $x(t)$ is too far away from the centers c, then $\sigma_i\big(x(t)\big) = e^{\frac{-||x-c_i||^2}{2\mu^2}} \approx 0$
- That means, no matter what you weights are $\nu_{ad} = W^T\sigma\big(x(t)\big) \approx 0$
- Therefore the weights can end up bursting!
- Need a method to select centers online to ensure the RBF representation is valid
- This leads to a nonparametric approach: in which the number of RBFs are not fixed a-priori



Linear independence in RKHS

- Add $x(t)$ to the dictionary of centers $C_l = \{c_i\}_{i=1}^{l}$ using a linear independence test (Nguyen-Tong et al.)

$$\gamma = \left\| \sum_{i=1}^{l} a_i \phi(c_i) - \phi\big(x(t)\big) \right\|$$

- Let K be the Kernel matrix s.t. $K_{i,j} = \Phi\big(x_i, x_j\big)$ then

$$\gamma = k\big(x(t), x(t)\big) - [K(C_l, x(t)]^T \hat{a}_l$$

- The Budgeted Kernel Restructuring Algorithm (BKR):
  - Add $x(t)$ if $\gamma > \eta$ and increase size of dictionary, recalculate γ
  - If dictionary size exceeds budget, add center by removing an existing center with the least γ, recalculate γ's

Recap of BKR-CL (CDC 11, TNNLS 12)

- A linear independence test in a Reproducing Hilbert Space used to select new centers online (Budgeted Kernel Restructuring (BKR))
- If $\gamma = \left\|\sum_{i=1}^{l} a_i \psi(c_i) - \psi\big(x(t)\big)\right\| > \epsilon$ then add $x(t)$ as a center to the kernel dictionary $D_t = [c_1, c_2, \dots, c_b]$
- Correspondingly update the history stack
- When budget reached ($|D_t|$=b), remove the point with minimal γ
- Update NN weights using Concurrent Learning (CL) weight update law

Concurrent Learning adaptive law

- Use online recorded data concurrently with current data
- For the stored data point x_k assume that $\dot{x}_k$ is available and let

$\epsilon_k(t) = W^T(t)\Phi(x_k) - \Delta(x_k) = W^T(t)\Phi(x_k) - (\dot{x}_k - \nu_k)$, since $\dot{x} = \hat{f} + \Delta$

Concurrent learning adaptive law

$$\dot{W}(t) = \underbrace{-\Gamma\Phi\big(x(t)\big)e^T(t)PB}_{\text{Instantaneous update, } \dot{W}_t} \underbrace{- \Gamma\sum_{k=1}^{p}\Phi(x_k)\epsilon_k^T(t)}_{\text{Update on recorded data, } \dot{W}_b}$$

- Online history stack matrix $Z = [\Phi(x_1),\ \Phi(x_2), \ldots, \Phi(x_{p_{\max}})]$
- Max dimension of Z is fixed, so the data is recorded sparsely
- Z is not a moving window of data (although it could be)

Weight error dynamics

- Key insight comes from analyzing the weight error dynamics:

$$\dot{W}(t) - \dot{W}^* = -\Gamma\Phi\big(x(t)\big)e^T(t)PB \quad - \Gamma\sum_{k=1}^{p}\Phi(x_k)\epsilon_k^T\,(t)$$

But, $\epsilon_k = \nu_{ad}(x_k) - \Delta(x_k) = W^T(t)\Phi(x_k) - \Delta(x_k)$

However, $\Delta(x_k) = {W^*}^T\Phi(x_k)$, and recall that $\tilde{W} = W - W^*$, so,

$$\epsilon_k = \tilde{W}^T\Phi(x_k)$$

- This yields:

$$\dot{\tilde{W}}(t) = -\Gamma\Phi\big(x(t)\big)e(t)^T P - \Gamma\sum_{k=1}^{p}\Phi(x_j)\left(\tilde{W}^T(t)\Phi(x_k)\right)^T$$

$$\dot{\tilde{W}}(t) = -\Gamma\Phi\big(x(t)\big)e(t)^T P - \Gamma\sum_{k=1}^{p}\Phi(x_k)\Phi^T(x_k)\,\tilde{W}(t)$$

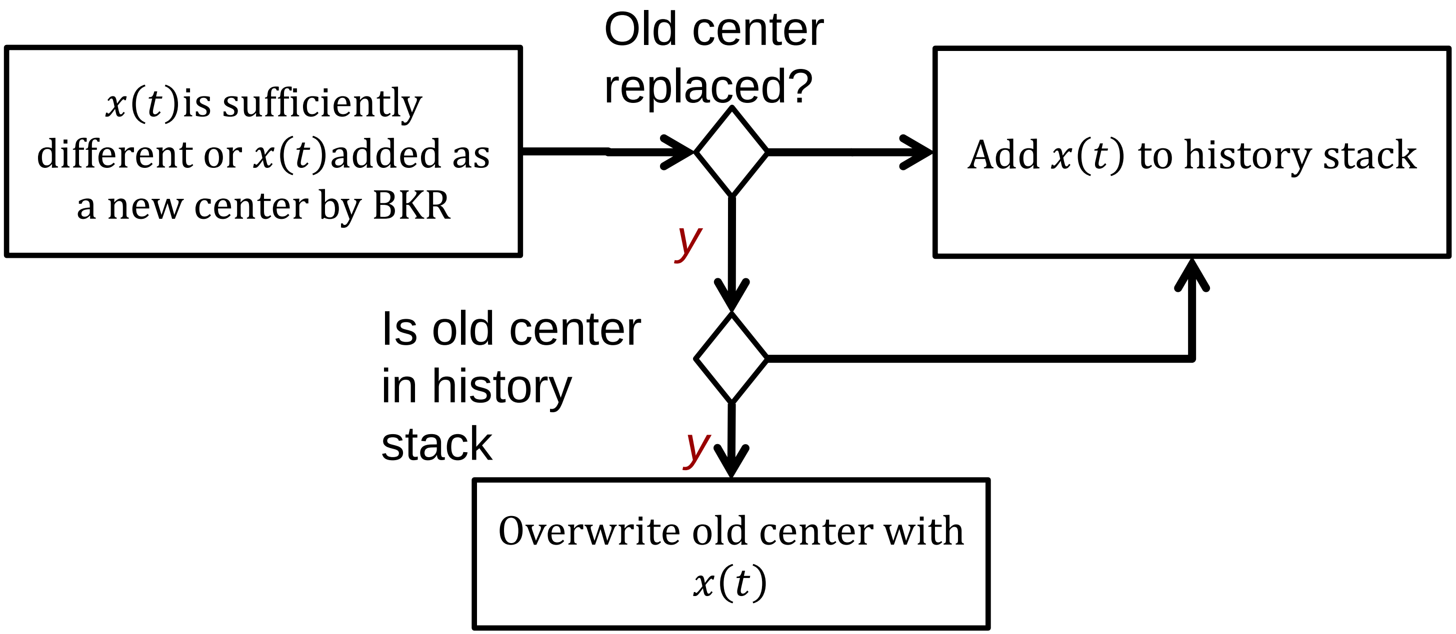
Theoretical impact of BKR on CL

THM: UUB of BKR-CL

The switched close loop system defined by the weight error dynamics and the tracking error dynamics is globally uniformly ultimately bounded.

- Let $\Omega_t = \sum_{k=1}^{p} \Phi_t(x_k)\Phi_t^T(x_k)$, where $\Phi_t(x_k) = [k(x_k, c_1), k(x_k, c_2), \ldots, k(x_k, c_l)]$, with l the number of centers in the dictionary
- Then the rank-condition is typically automatically satisfied because a new kernel is only added if it is not in the span of the space generated by existing centers

BKR-CL: a non parametric approach on a budget



- Once history stack is full, singular value maximizing algorithm (ACC 11) used
- The dictionary and history stack updates are coupled
- SVD calculation is the most expensive part, can be further improved
- Theorem: UUB of the *switched* closed loop system guaranteed

Wing Rock dynamics with BKR-CL

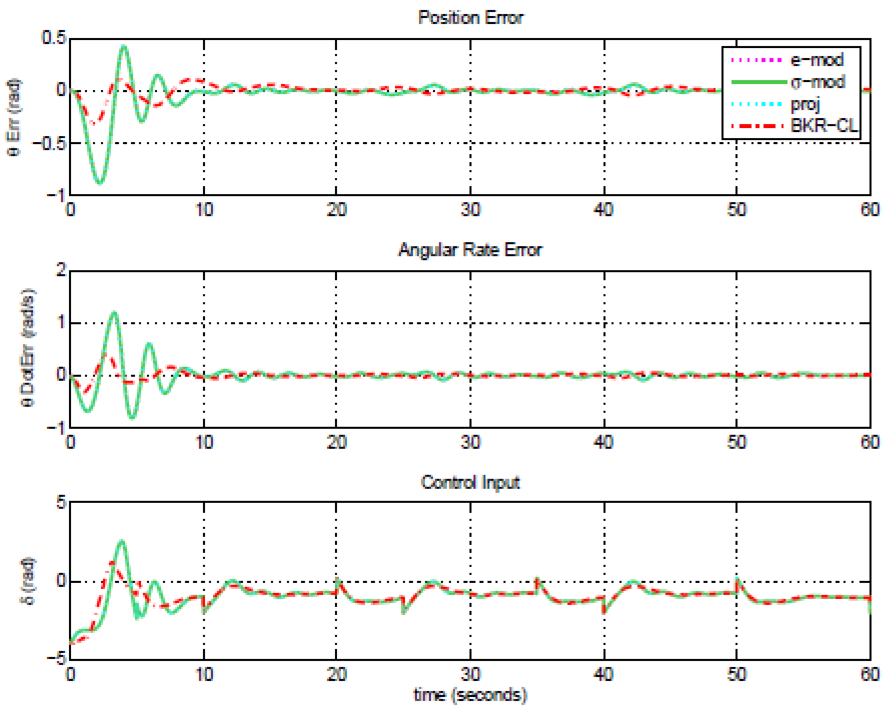
- Highly swept-back aircraft are susceptible to Wing Rock: lightly damped oscillations
- ϕ: roll angle, p roll rate, δ_a aileron

A model for Wing Rock dynamics (Monahemi 96)

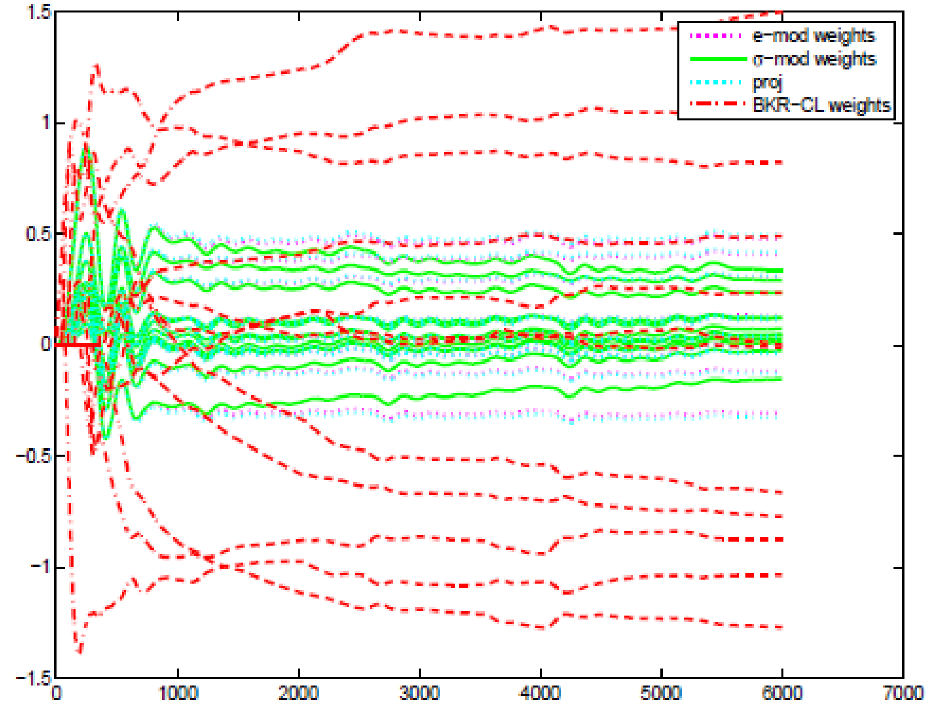
$$\dot{\phi} = p$$
$$\dot{p} = \delta_a + \Delta(\phi, p)$$

- Inversion model: $\nu = \delta$
- Task: track roll commands in presence of wing rock dynamics: $\Delta(\phi, p) = 0.8 + 0.23\phi + 0.69p - 0.62|\phi| + 0.01|p|p + 0 + .02p^3$
- Second order reference model used
- History stack max size: 32, RBF dictionary max size: 20

Comparison with traditional methods

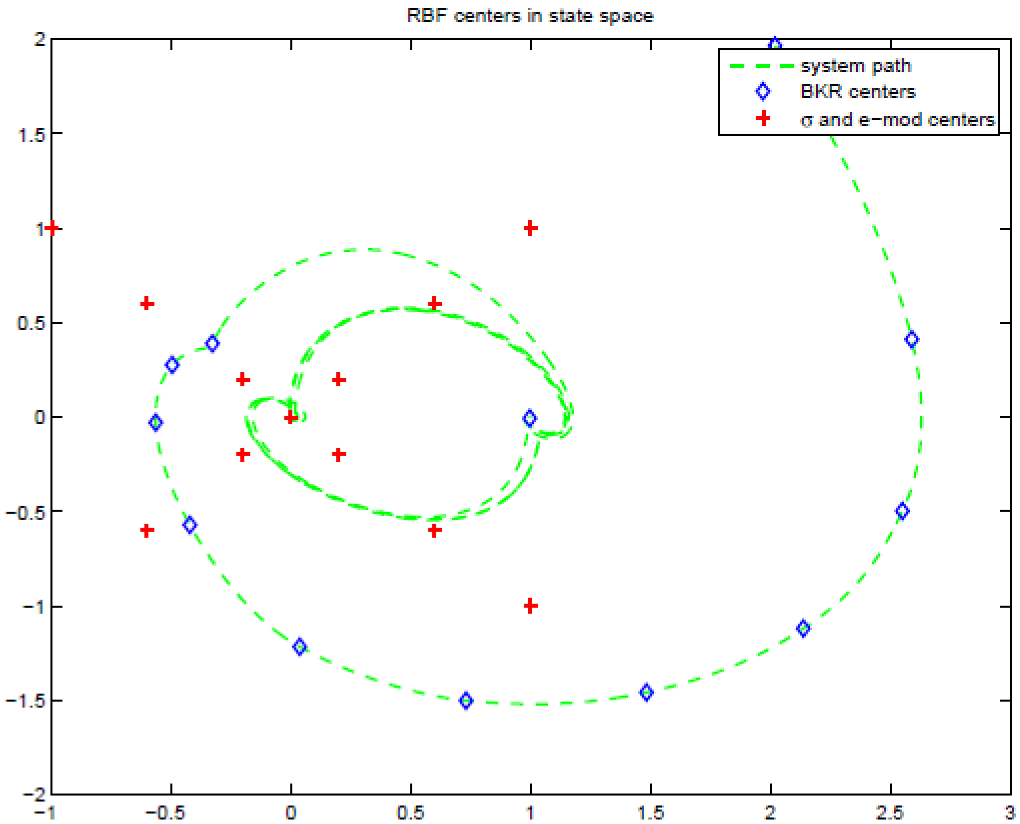


Tracking error comparison



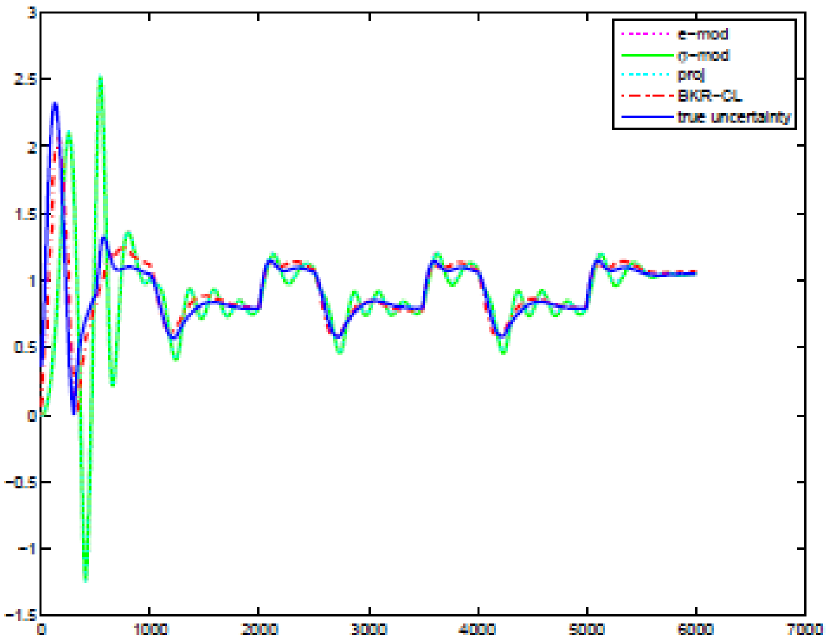
Weight evolution comparison

Final center assignment with 12 centers

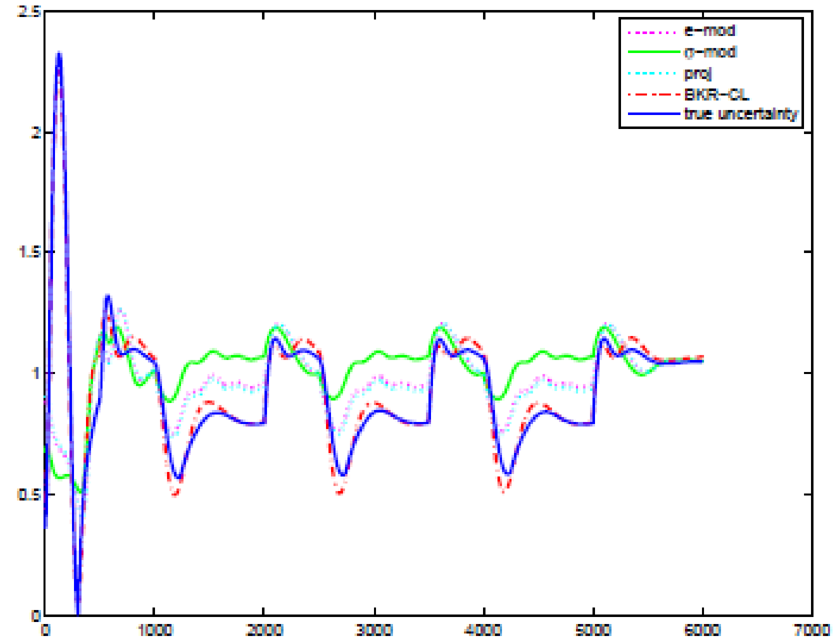


- Centers are assigned along the path of the system
- Video of the moving centers: http://www.youtube.com/watch?feature=player_detailpage&v=0_CGXmSXegs

Long-Term learning *(TNN 11)*



Online estimate of uncertainty



Estimate of uncertainty with weights frozen post-simulation

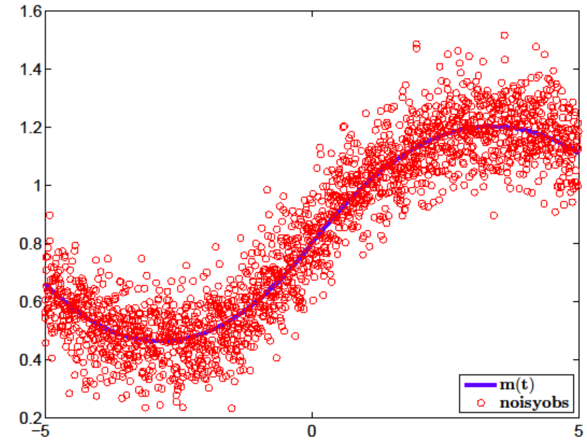
- Important insight: Good online tracking of uncertainty doesn't mean you are actually *learning* the uncertainty
- BKR-CL tracks the uncertainty and learns it over the long term
- Reference: *Kingravi, Chowdhary, Vela, Johnson CDC 2011 (Dec), and TNN 2011 (submitted)*

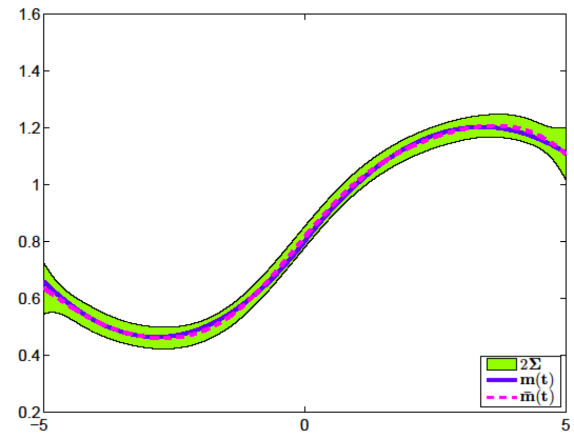
Nonparametric Bayesian Models in Adaptive Control

- A stochastic representation of the uncertainty may be more natural, e.g. Gaussian Processes (GP):

$$\Delta(x) \sim GP\big(m(x), k(x, x')\big)$$

  - $m(x)$ is the mean function and $k(x, x')$ is the covariance function
- New Approach: GP-MRAC
  - Let the prior be a Gaussian Process
  - Perform Bayesian posterior inference to estimate the mean ($\bar{m}(x)$) using observed data
  - The adaptive control: $\nu_{ad} = \bar{m}(x)$
- Bayesian inference needs to be performed in a sequential manner online on a Budget: Remove data points using information theoretic measures (K-L divergence)
- Benefits:
  - Kernels evolve with data: globally applicable
  - Paradigm shift: represent uncertainty as a distribution over functions
  - Inherently handles measurement noise





GP representation of uncertainty

Gaussian Processes

- A Gaussian Process is a collection of random variables, any finite subset of which has a joint Gaussian distribution
- A Gaussian process is completely characterized by the mean $m(x)$ and the covariance function $k(x,x')$: $f(x) \sim GP\big(m(x), k(x,x')\big)$
- GPs are a Bayesian Nonparametric (BNP) model widely studied for supervised learning problems[3]
- Nonparametric: The number of parameters are not fixed a-priori, rather they grow in response to the data
- Offline learned dynamic models using GPs have been used in robotics[3,4]

Some references:

1. **Rasmussen** C. E., Williams C. K. I. Gaussian processes for machine learning, the MIT press 2005
2. **Csató** L., and Opper M., Sparse on-line Gaussian processes, Neural Computations, 14(3):641-668, 2002.
3. Ko J., **Fox** D., GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models, Autonomous Robots, 27(1), 2009.
4. Ko J., Klein D., **Fox** D., Haehnel D., Gaussian process and reinforcement learning for identification and control of an autonomous blimp. IEEE ICRA, 2007.

Rass

GP Inference

- $Z_t = \{z_1, z_2, \dots, z_t\}$ a set of recorded states up to time t
- For each i the measurements of the modeling error are:

$$y_i = (\dot{z} + \epsilon_i) - \nu = \Delta(z_i) + \epsilon_i$$

Where $\epsilon_i \sim \mathcal{N}(0, \omega^2)$ Gaussian white noise

- $Y_t = \{y_1, y_2, \dots, y_t\}$
- Bayesian inference:

$$posterior = \frac{(prior * likelihood)}{total\ probability}$$

GP Inference

- Let $K_t \in \Re^{t\times t}$ be the kernel matrix s.t. $K_{i,j} = k(z_i, z_j)$
- The joint distribution of training inputs Y_t and y_{t+1} is

$$\begin{bmatrix} Y_t \\ y_{t+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_t(Z_t, Z_t) + \omega^2 I & \bar{k} \\ \bar{k}^T & k_* \end{bmatrix}\right)$$

- Where $\bar{\mathrm{k}} = K(z_{t+1}, Z\_t), k_* = k(z_{t+1}, z_{t+1})$
- The closed form for Gaussian distributions makes inference easy

GP Inference

- Predicting the value at a point z_{t+1}

$$\Delta_{t+1} \sim \mathcal{N}\left(m(\mathrm{z_{t+1}}), \mathrm{cov}(\mathrm{z_{t+1}}, \mathrm{Z_t})\right)$$

- With the predictive mean given by

$$m(\mathrm{z_{t+1}}) = K(z_{t+1}, Z_t)(K(Z_t, Z_t) + \omega^2 I)^{-1} Y_t$$

- The predictive covariance given by

$$\mathbb{V}(\mathrm{z_{t+1}}) = k_* - \bar{k}^T (K_t + \omega^2 I)^{-1} \bar{k}$$

- The inversion is well defined for Gaussian kernels due to Mercer’s theorem
- Can avoid the inverse and do a sequential update (see Csato 2002, Sparse Online Gaussian Processes, Neural Computation)

The need for Sparsification

- Quick recap of GP-MRAC
  - Model the adaptive element as the mean of a GP
  - Perform nonparameteric inference on the GP using data pairs $(\widehat{\Delta}, x)$ noisy modeling error estimates $\left(\widehat{\Delta} = \dot{\hat{z}} - \nu\right)$
  - GP adds a kernel $k(z_t, .)$ at every input point
- Adding a kernel at every input point can quickly become intractable in online applications
- Need a way to filter through and only pick relevant points as kernels

Budgeted GP inference

- Need to limit the max allowable size of the kernel dictionary: Budget
- After the budget is reached, new data incorporated only by replacing old data
- Two possible methods
  - Remove the oldest point from kernel dictionary
  - Use KL divergence between prior and posterior after getting a measurement $(z, \dot{\hat{z}})$

GP MRAC algorithm recap

- Algorithm: while new measurements $(z_t, \dot{\hat{z}}_t)$
  - Prior is a Gaussian Process
  - Perform online Bayesian posterior inference to estimate the mean $(\bar{m}(x))$
  - Adaptive element output: $\nu_{ad} = \bar{m}(z)$
  - Determine whether to add the current state z in the dictionary of kernels
  - If budget has been reached remove an old point using either KL divergence (KD) or oldest point (OP) method
- Benefits:
  - Kernels evolve with data: globally applicable nonparametric method
  - Paradigm shift: represent uncertainty as a distribution over functions
  - Inherently handles measurement noise, analysis requires Ito calculus

Wing Rock dynamics with BKR-CL

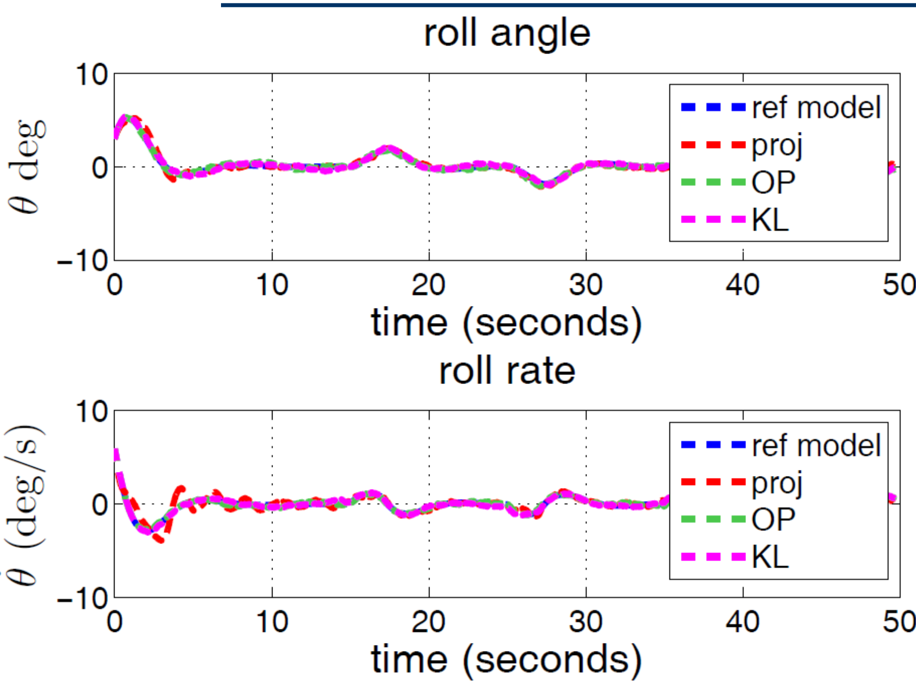
- Highly swept-back aircraft are susceptible to Wing Rock: lightly damped oscillations
- ϕ: roll angle, p roll rate, δ_a aileron

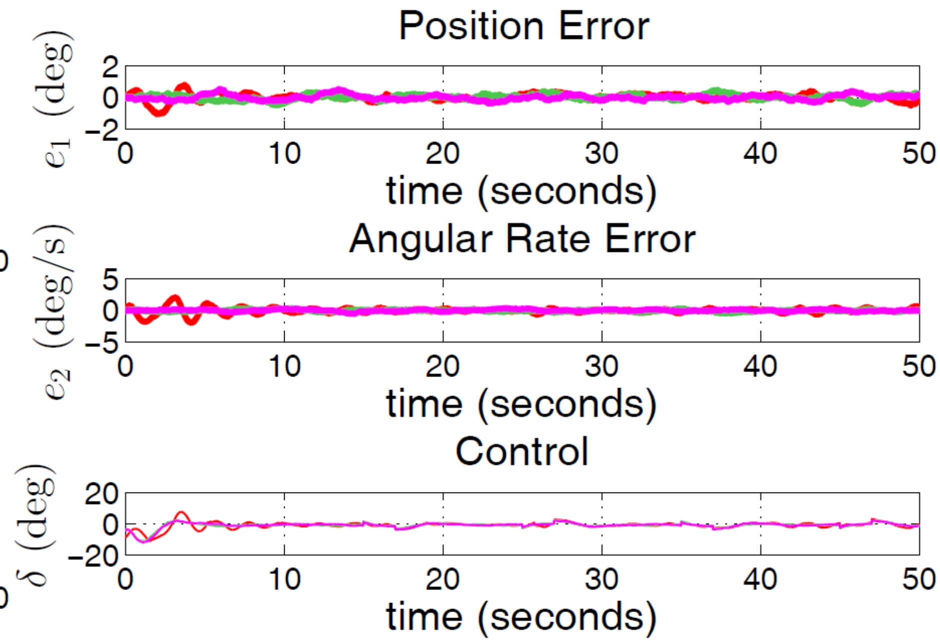
A model for Wing Rock dynamics (Monahemi 96)

$$\dot{\phi} = p$$
$$\dot{p} = \delta_a + \Delta(\phi, p)$$

- Inversion model: $\nu = \delta$
- Task: track roll commands in presence of wing rock dynamics: $\Delta(\phi, p) = 0.8 + 0.23\phi + 0.69p - 0.62|\phi| + 0.01|p|p + 0 + .02p^3$
- Second order reference model used
- History stack max size: 32, RBF dictionary max size: 20

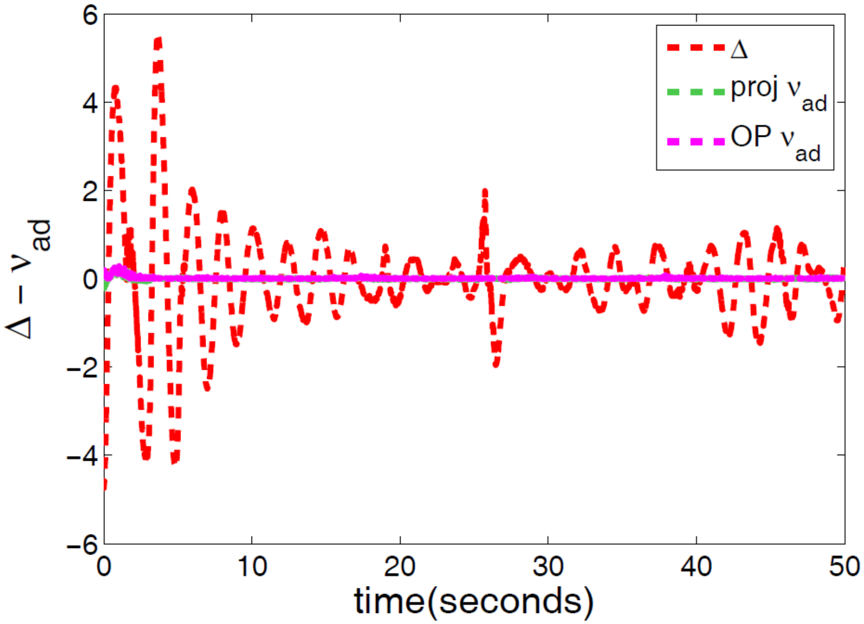
Case 1: System operates in expected domain



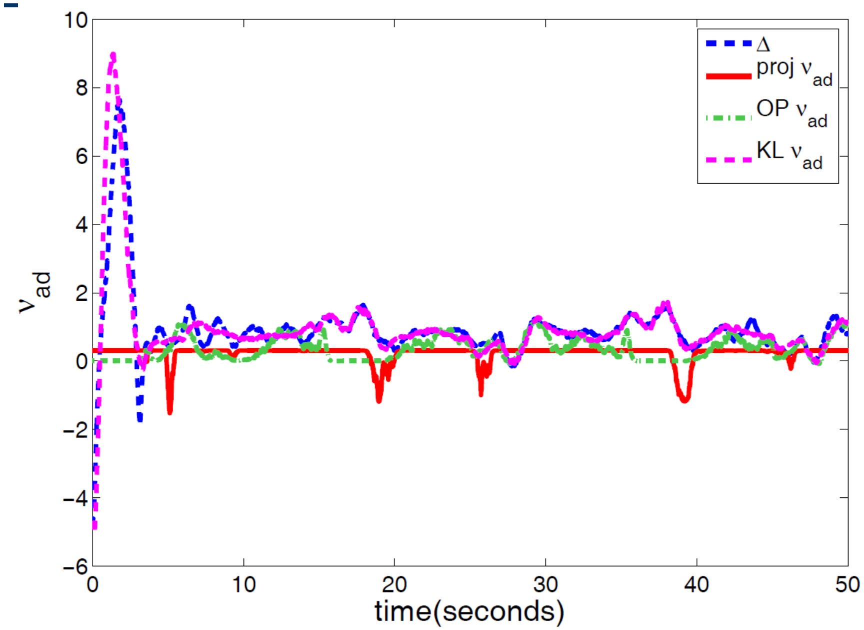


- RBF centers spread in expected domain of operation, system does not leave that domain
- Tracking performance comparable, GP-MRAC has less oscillations

Case 1: System operates in expected domain



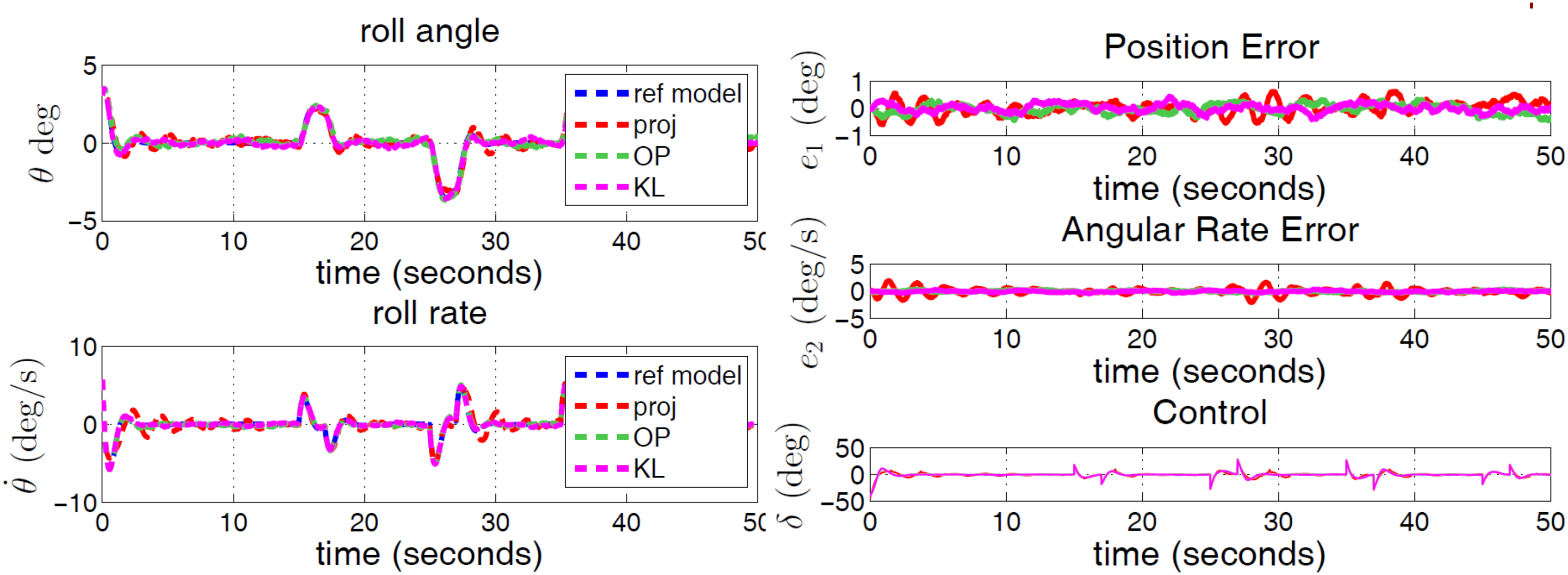
Difference between actual uncertainty and adaptive element output during simulation



Estimate of uncertainty post-simulation: metric on learning

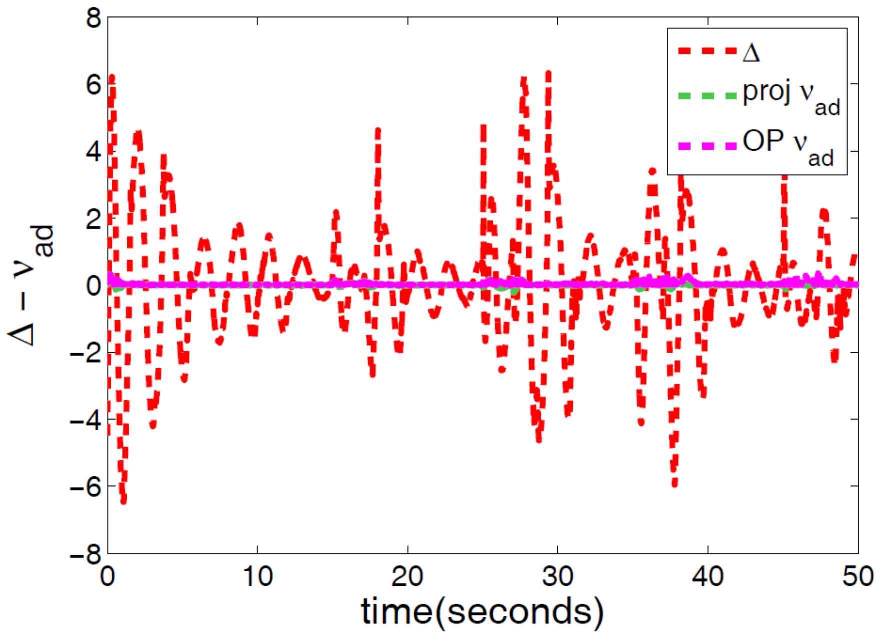
- Uncertainty captured better with GP-MRAC online
- Noise leads to oscillatory behavior with current choice of gains and bounds on the projection operator for traditional MRAC
- GP-MRAC (KD) exhibits long-term learning of the uncertainty

Case 2: System leaves expected domain

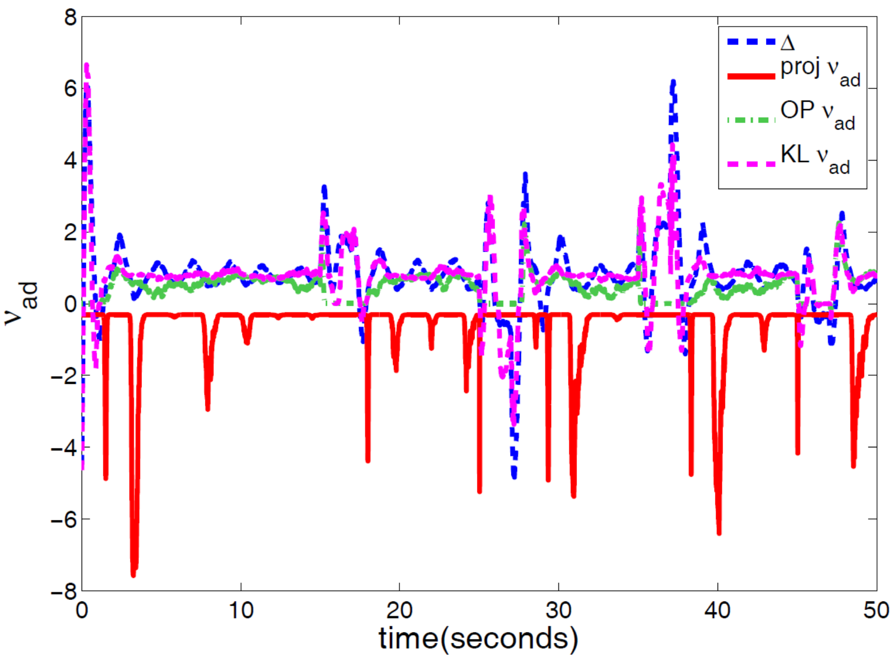


- The reference commands drive the system out of $[-1,1]$, where centers were distributed
- More oscillation seen without GP-MRAC

Case 1: System operates in expected domain



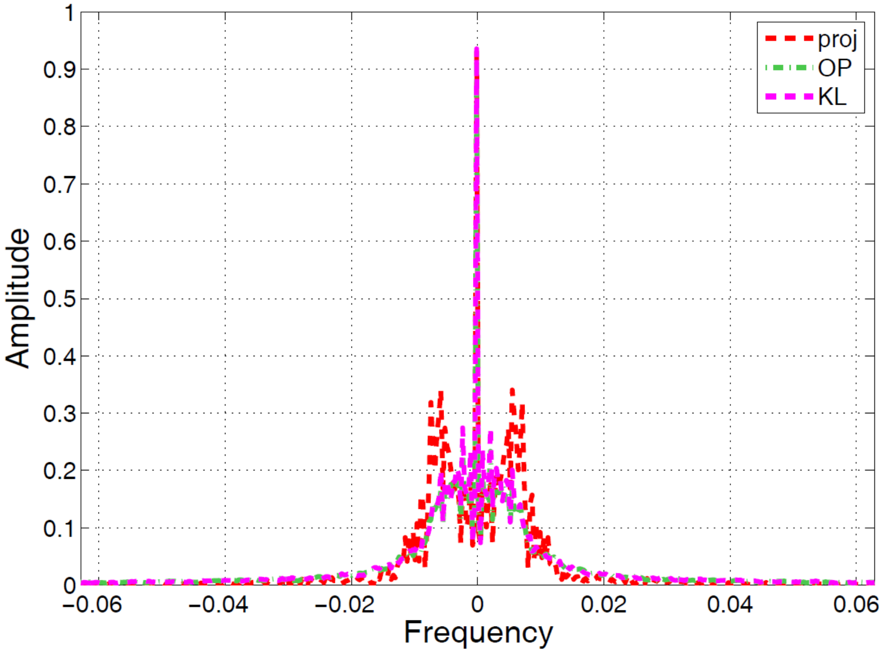
Difference between actual uncertainty and adaptive element output during simulation



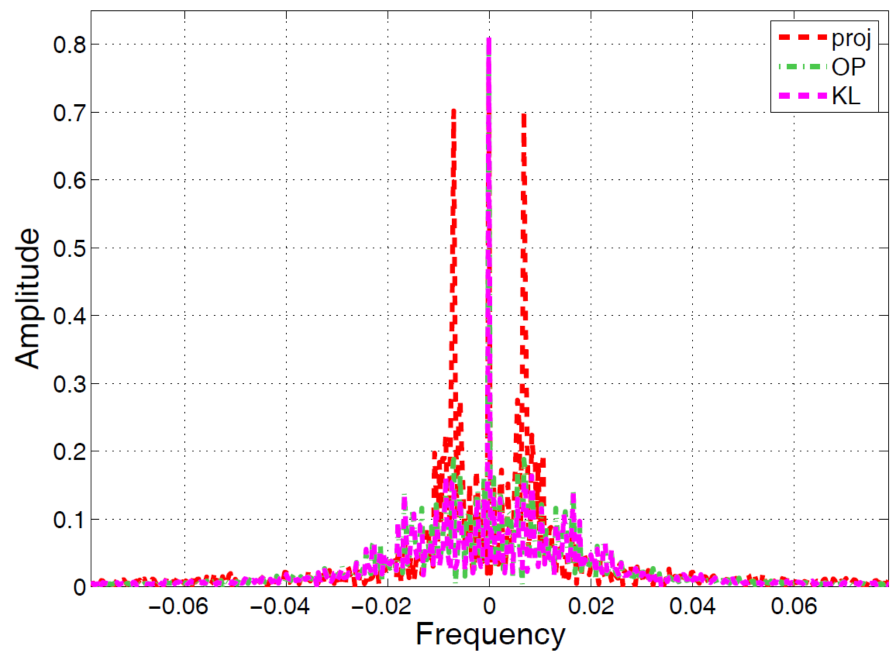
Estimate of uncertainty post-simulation: metric on learning

- Uncertainty captured better with GP-MRAC online
- Noise leads to oscillatory behavior with current choice of gains and bounds on the projection operator for traditional MRAC
- GP-MRAC (KD) exhibits long-term learning of the uncertainty

The benefit in terms of reduced oscillations



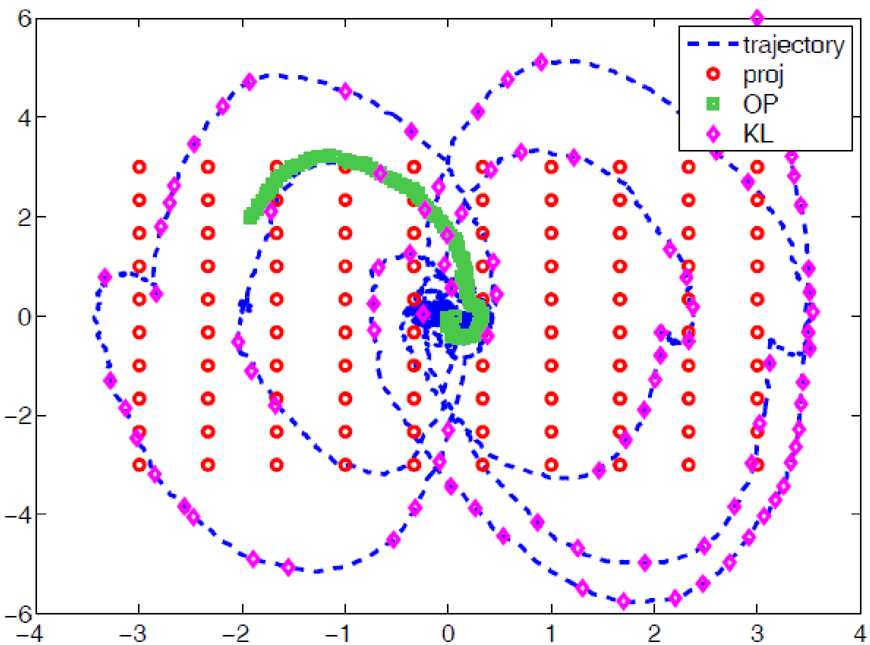
FFT of error for Case 1

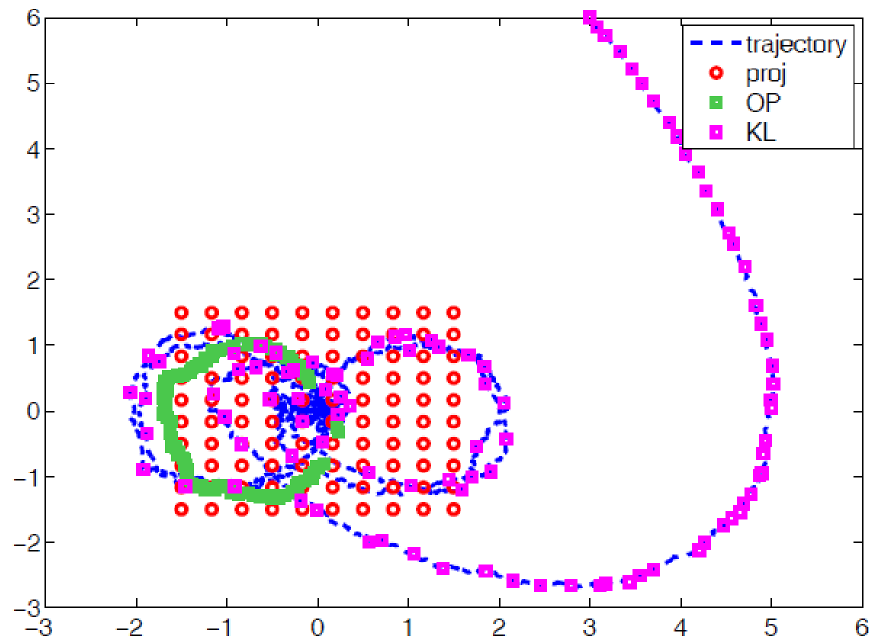


FFT of error for Case 2

- Significant reduction in oscillations with GP-MRAC
- Indicates that the adaptive element can predict and cancel out the uncertainty better

Where the centers are





- Both KL and OD place centers along the trajectory
- KL retains older centers, OD only keeps recent centers: forgets older data (good for time-varying uncertainties)

Summary

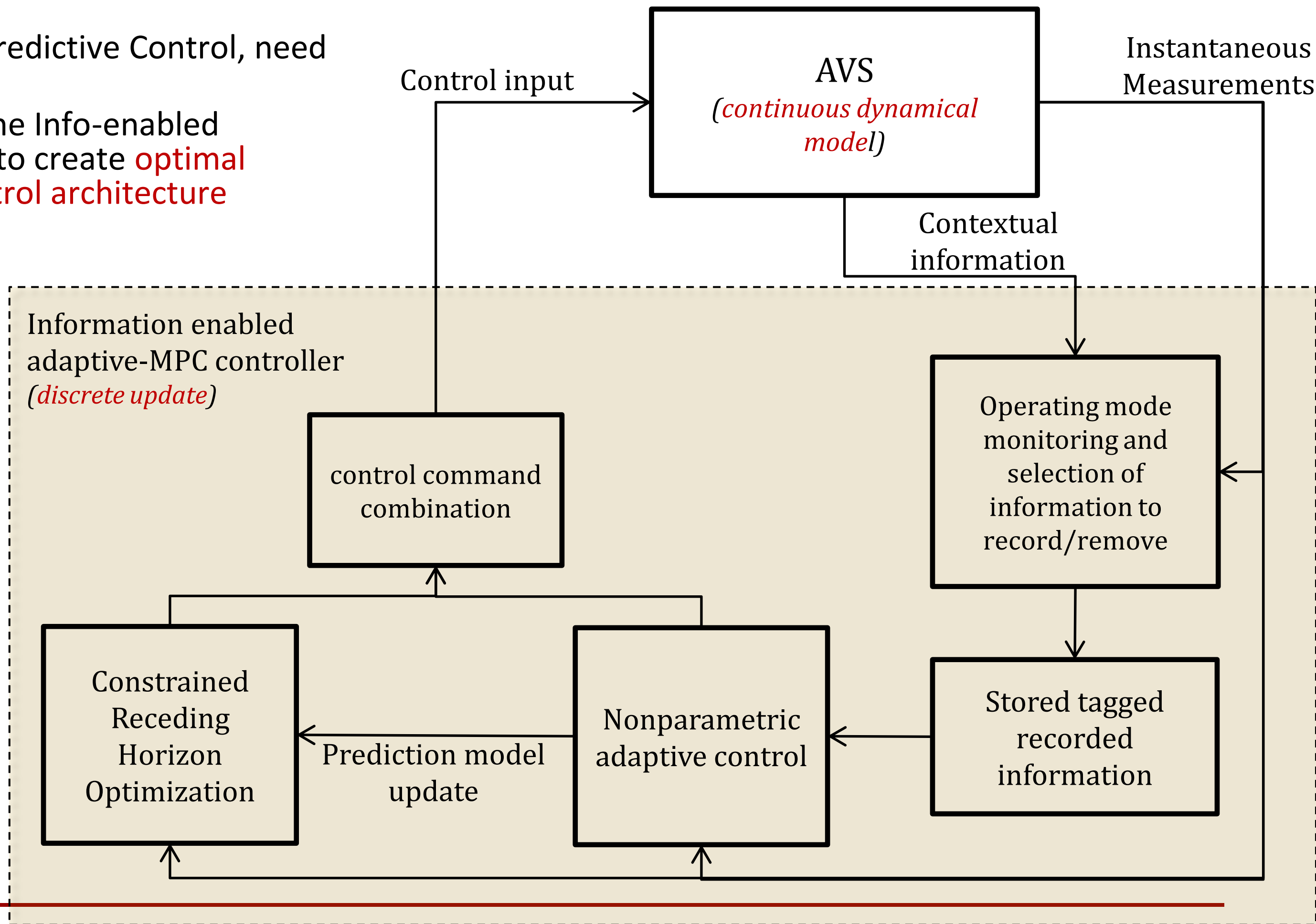
- Presented BKR-CL: a non parameteric approach to RBF NN adaptive element *on a budget*
  - CL helps in guaranteeing weights go to their ideal values
  - BKR helps in ensuring the centers are relevant
- Presented GP-MRAC a budgeted Bayesian Nonparameteric MRAC scheme
  - Bayesian inference used to estimate ideal weights, guaranteed optimal
  - Can extended operating domain almost globally
  - Can be thought of a stochastic nonparametric extension of BKR-CL

Examples: BKR-CL

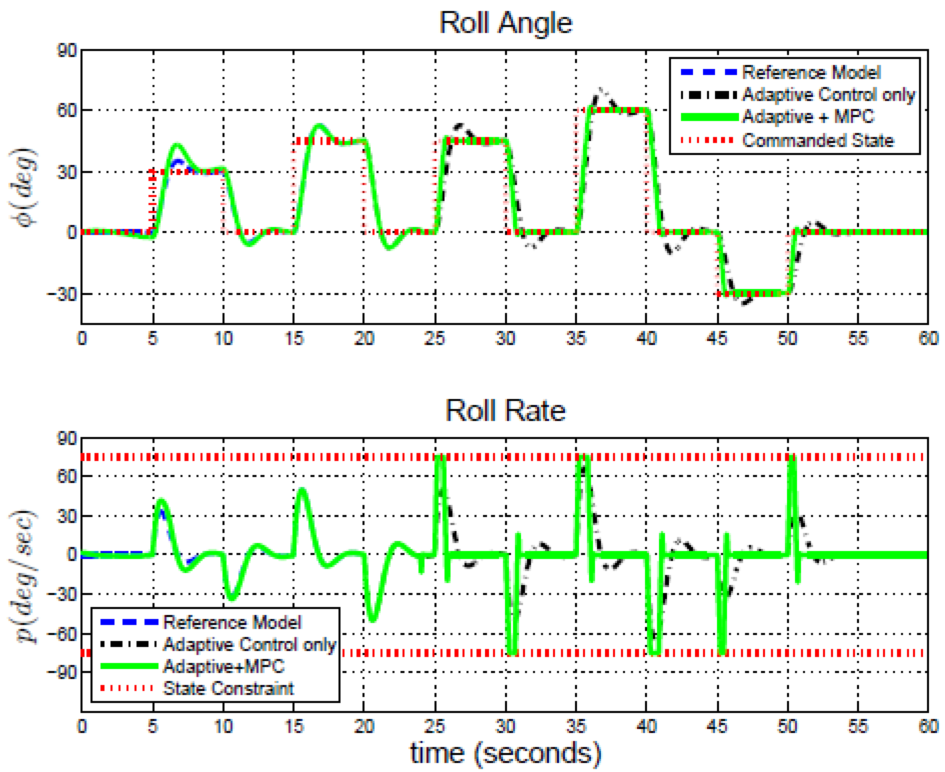
Example GP-MRAC

Future: Information enabled Optimal Control

- Need controllers with optimality guarantees in face of uncertainty
- Current approach: Model Predictive Control, need good system models
- Proposed approach: combine Info-enabled adaptive control with MPC to create optimal Integrate planning and control architecture
- NSF-CPS or AFOSR grant

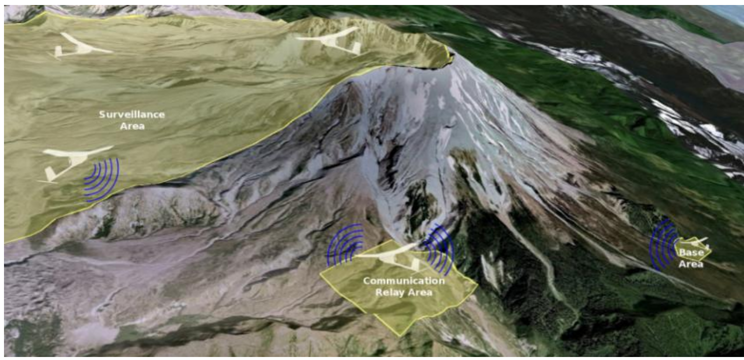


Adaptive-MPC



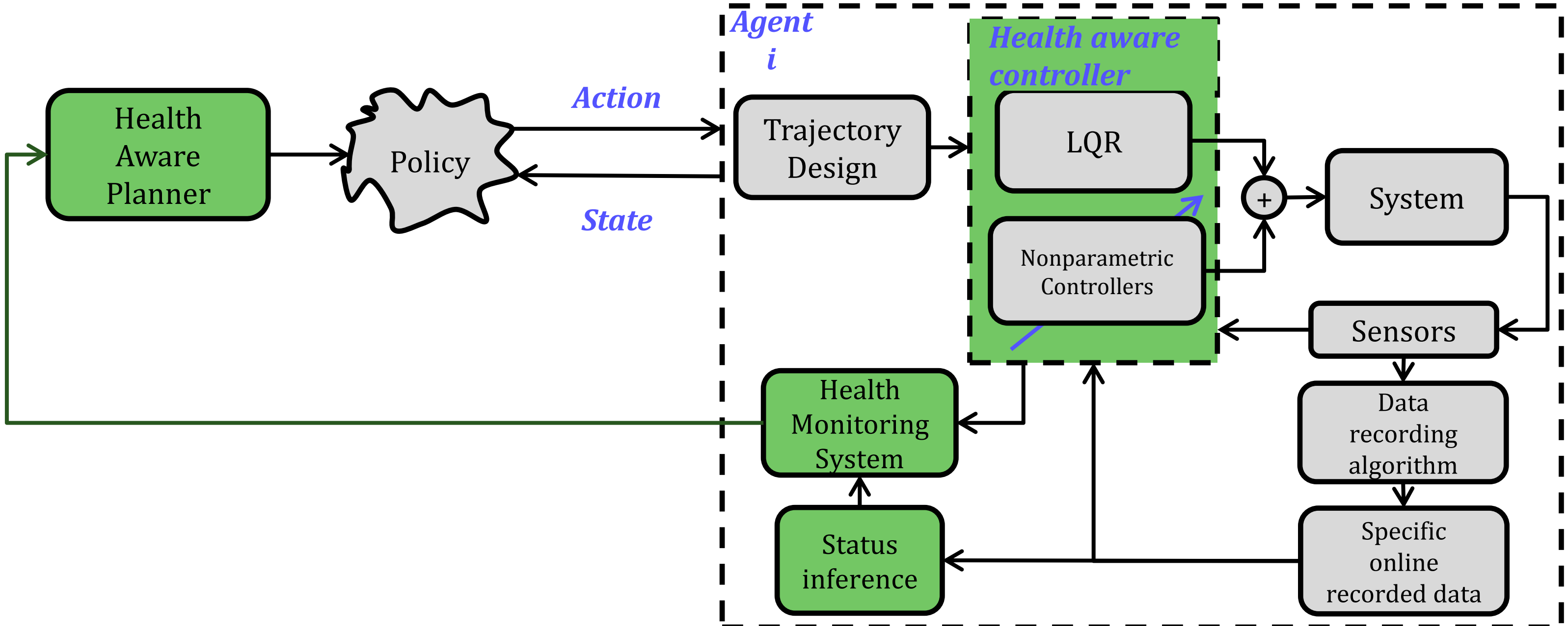
- CL-MRAC good at learning and simultaneously stabilizing unstable systems
- **Problem**: Difficult to guarantee optimality under constraints with MRAC
- **Idea:** learn the model using CL-MRAC, Switch to MPC when model learned
- Preliminary results: Combined CL MRAC-MPC approach guarantees stability and optimality in presence of state and actuator constraints

Ongoing: Technologies for Persistent UAS missions (Boeing R&D)



- Persistent UAV missions bring new challenges
  - How to achieve mission objective in presence of fuel and communication constraints
  - UAV dynamics and health may change during mission (fault tolerance)
- Ongoing work:
  - Improving planning by using information contained in the internal parameters of adaptive controllers
  - Improved planning algorithms in the framework dynamic programming/reinforcement learning
  - nonparametric Bayesian models for planning

Health Aware Planning



- How can we use learned system information for making better decisions?
- How can this information be shared across the fleet?

Questions??

Metrics based adaptive fault tolerant control GNC 09, 10; ACC 10; Infotech 10,11