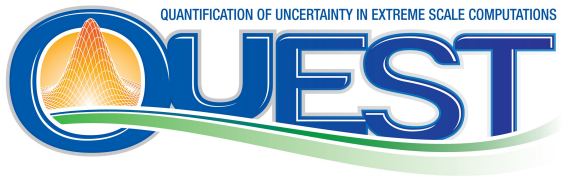
Polynomial Chaos Based Uncertainty Propagation

Lecture 2: Forward Propagation and Inverse Problems

Bert Debusschere†

†**bjdebus@sandia.gov**
Sandia National Laboratories,
Livermore, CA

Aug 11, 2014 – UQ Summer School, USC



Acknowledgement

Cosmin Safta Khachik Sargsyan Kenny Chowdhary Habib Najm	Sandia National Laboratories Livermore, CA
Omar Knio	Duke University, Raleigh, NC
Roger Ghanem	University of Southern California Los Angeles, CA
Olivier Le Maître	LIMSI-CNRS, Orsay, France
and many others ...	

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Scientific Discovery through Advanced Computing (SciDAC) and Applied Mathematics Research (AMR) programs.

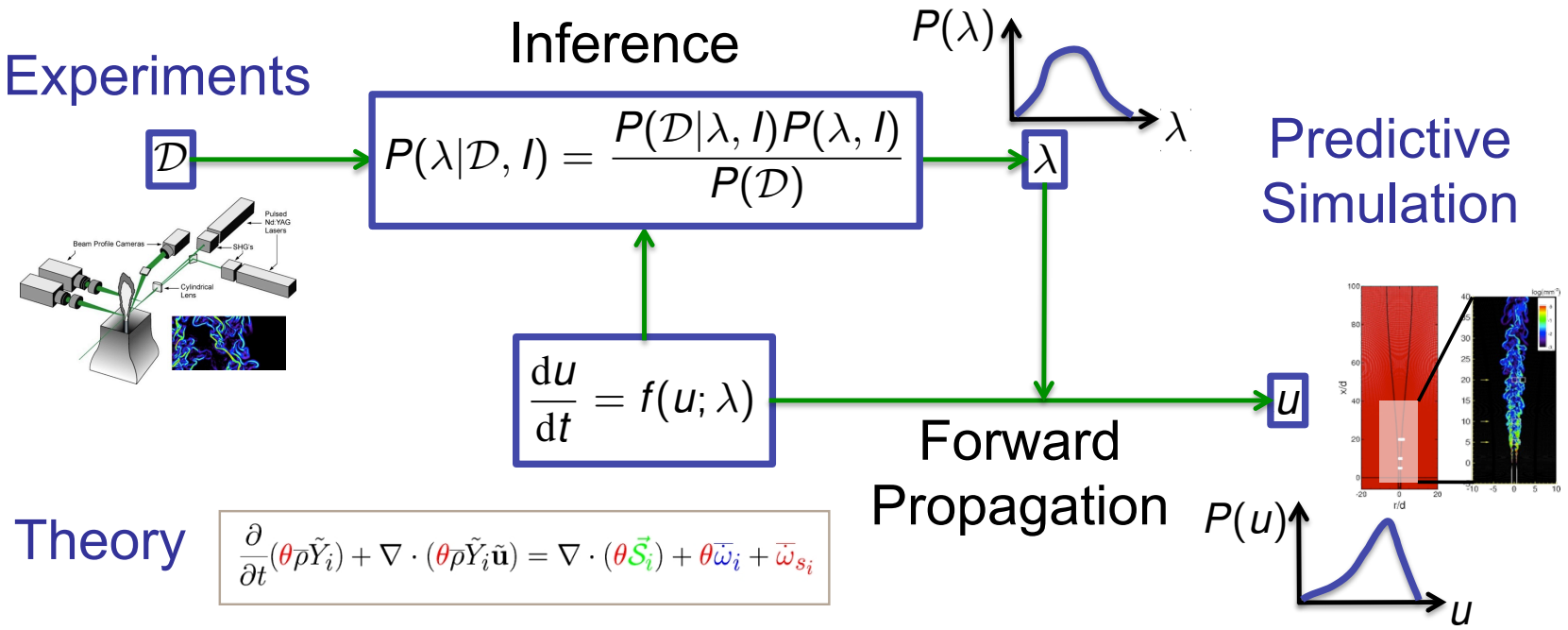
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Goals of this tutorial

- 2 Lectures
  - Lecture 1: Context and Fundamentals
  - Lecture 2: Forward Propagation and Inverse Problems
    - Intrusive Approaches
    - Non-Intrusive Approaches
    - High-dimensional Systems
    - Sensitivity Analysis
    - Inverse Problems

Outline

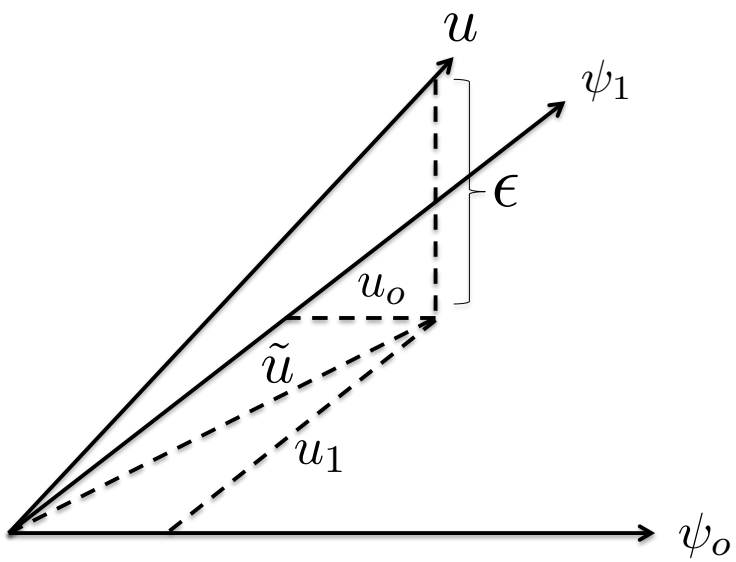
This section focuses on propagating uncertainty through computational models.



- Assume uncertain parameters λ have been characterized with PCEs
- The goal is to obtain PCEs for output quantities u

Propagation of Uncertain Inputs Represented with PCEs

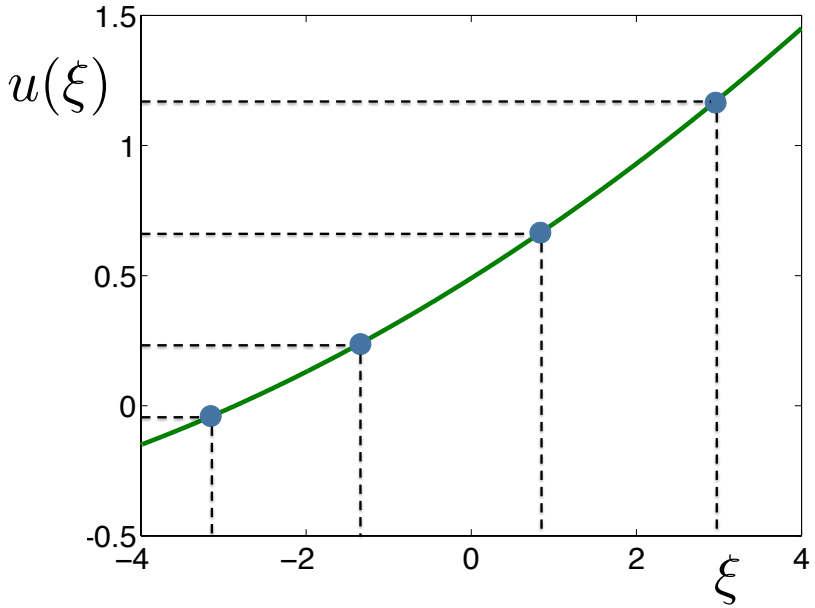
Galerkin Projection



$$u_k = \frac{\langle u\Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

Residual orthogonal to space covered by basis functions

Collocation



Match PCE to random variable at chosen sample points: interpolation or regression

Galerkin projection methods are either intrusive or non-intrusive

- Use same projection but in different ways

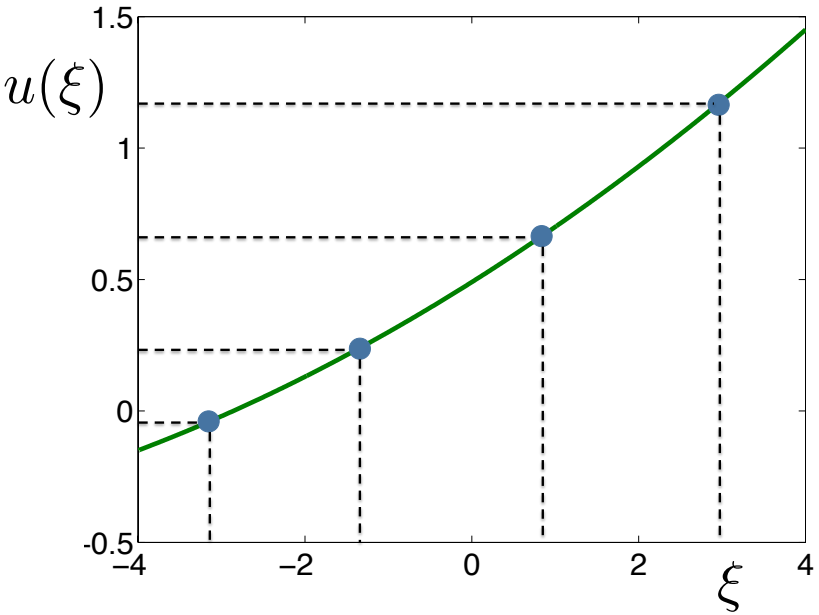
$$u_k = \frac{\langle u\Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

- Intrusive methods apply Galerkin projection to governing equations
  - Results in set of equations for the PC coefficients
  - Requires redesign of computer code
  - PCEs for all uncertain variables in system
- Non-intrusive approaches apply Galerkin projection to outputs of interest
  - Sampling to evaluate projection operator
  - Can use existing code as black box
  - Only computes PCEs for quantities of interest

Collocation approaches are non-intrusive and minimize errors at sample points

$$\sum_{k=0}^{P} u_k \Psi_k(\xi_i) = u(\xi_i)$$

$$i = 1, \ldots, N_c$$



- Use functional representation point of view
- Can use interpolation, e.g. Lagrange interpolants
- Or use regression approaches: $P + 1$ degrees of freedom to fit N_c points
- Can position points where most accuracy desired

Remainder of this section focuses on Galerkin projection methods

- Intrusive Galerkin projection
- Non-intrusive Galerkin projection

Intrusive Galerkin projection reformulates original equations

- Assume $v = f(u; a, \lambda)$, with
  - a deterministic parameter(s)
  - λ uncertain parameter(s)
  - u, v variables of interest (deterministic or uncertain)
- Represent uncertain variables with PCEs

$$\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k(\xi), \qquad v = \sum_{k=0}^{P} v_k \Psi_k(\xi)$$

- Apply Galerkin projection to get PC coefficients of v

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle f(u; a, \lambda) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

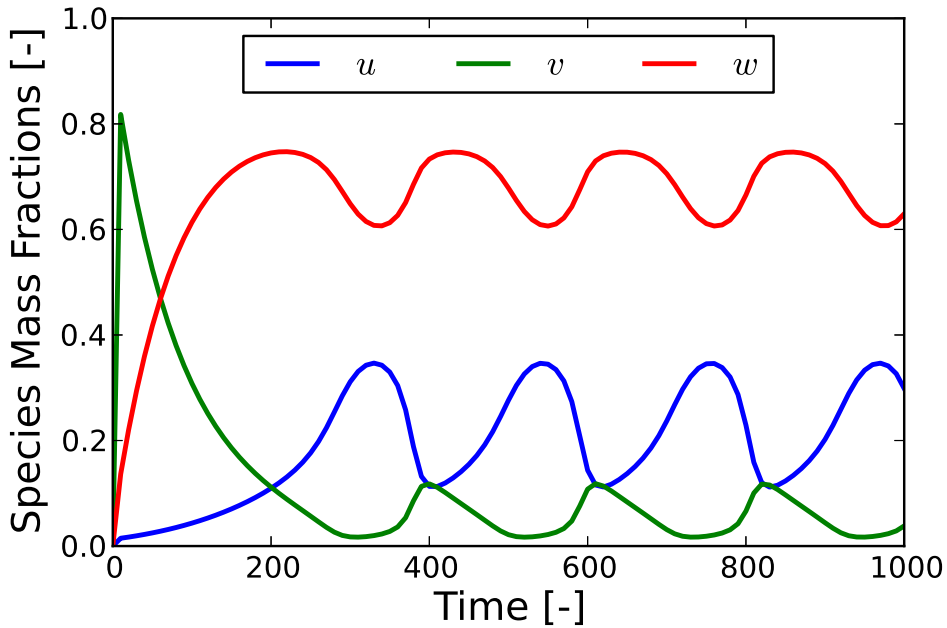
- Results in larger, but deterministic set of equations

Surface Reaction Model

3 ODEs for a monomer (u), dimer (v), and inert species (w) adsorbing onto a surface out of gas phase.

$$\begin{aligned}
\frac{\mathrm{d}u}{\mathrm{d}t} &= az - cu - 4duv \\
\frac{\mathrm{d}v}{\mathrm{d}t} &= 2bz^2 - 4duv \\
\frac{\mathrm{d}w}{\mathrm{d}t} &= ez - fw \\
z &= 1 - u - v - w
\end{aligned}$$

$$u(0) = v(0) = w(0) = 0.0$$

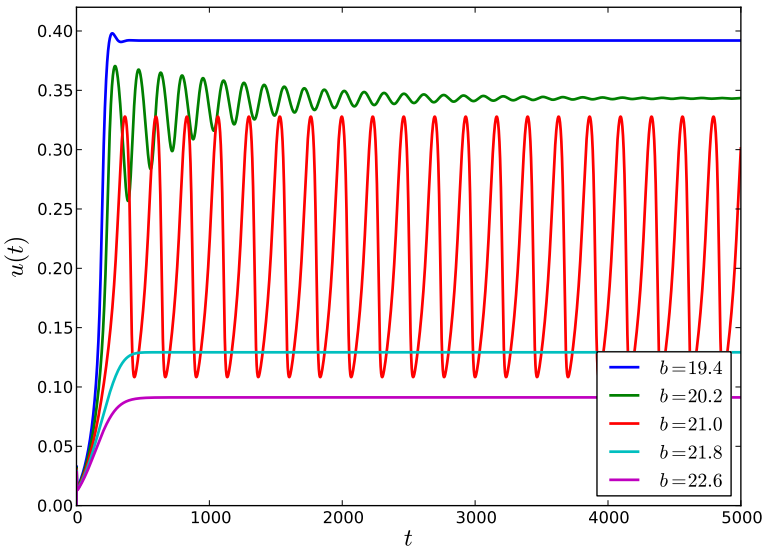


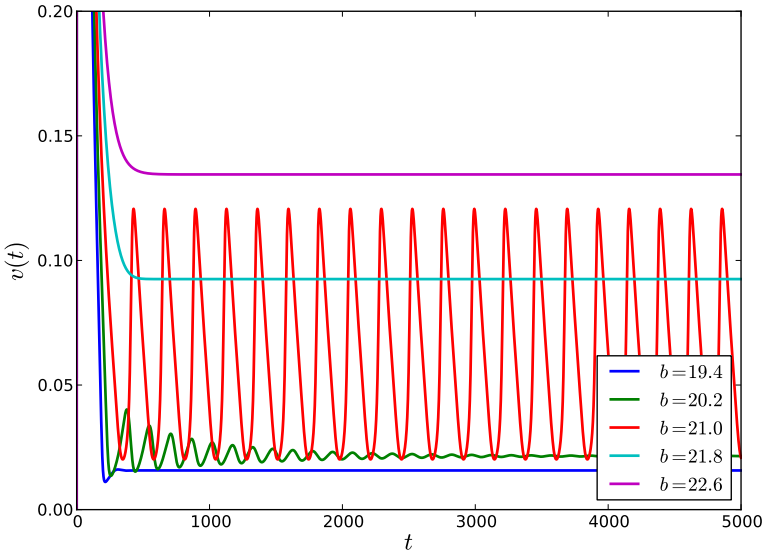
Oscillatory behavior for $b \in [20.2, 21.2]$

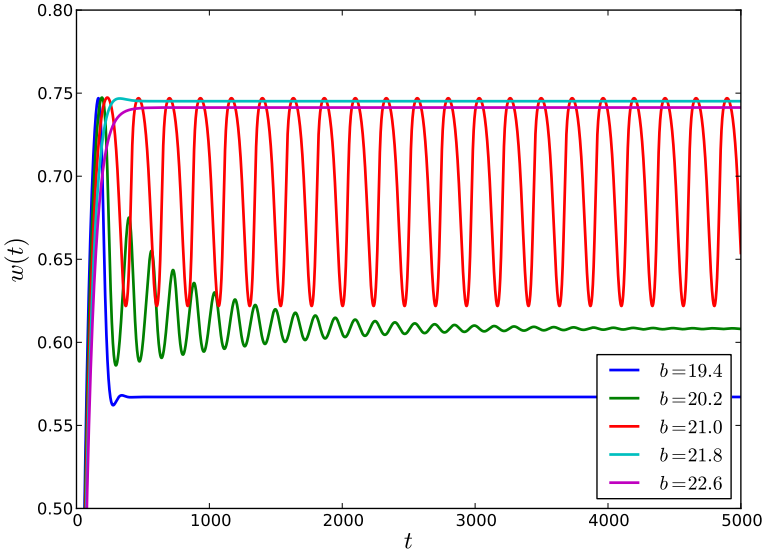
$$\begin{aligned}
a &= 1.6 \quad b = 20.75 \quad c = 0.04 \\
d &= 1.0 \quad e = 0.36 \quad f = 0.016
\end{aligned}$$

[Vigil *et al.*, Phys. Rev. E., 1996; Makeev *et al.*, J. Chem. Phys., 2002]

Surface reaction model shows wide range of dynamics







$$a = 1.6 \quad b = [19.4 \ldots 22.6]$$
$$c = 0.04 \quad d = 1.0$$
$$e = 0.36 \quad f = 0.016$$

Surface Reaction Model: Intrusive Spectral Propagation (ISP) of Uncertainty

- Assume PCE for uncertain parameter b and for the output variables, u, v, w
- Substitute PCEs into the governing equations
- Project the governing equations onto the PC basis functions
  - Multiply with Ψ_k and take the expectation
- Apply pseudo-spectral approximations where necessary

Surface Reaction Model: Specify PCEs for inputs and outputs

Represent uncertain inputs with PCEs with known coefficients:

$$b = \sum_{i=0}^{P} b_i \Psi_i(\xi)$$

Represent all uncertain variables with PCEs with unknown coefficients:

$$u(t) = \sum_{i=0}^{P} u_i(t)\Psi_i(\xi) \qquad v(t) = \sum_{i=0}^{P} v_i(t)\Psi_i(\xi)$$

$$w(t) = \sum_{i=0}^{P} w_i(t)\Psi_i(\xi) \qquad z(t) = \sum_{i=0}^{P} z_i(t)\Psi_i(\xi)$$

Surface Reaction Model: Substitute PCEs into governing equations and project onto basis functions

$$\frac{\mathrm{d}u}{\mathrm{d}t} = az - cu - 4duv$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} u_i\Psi_i = a\sum_{i=0}^{P} z_i\Psi_i - c\sum_{i=0}^{P} u_i\Psi_i - 4d\sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j$$

$$\left\langle \Psi_k \frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} u_i\Psi_i \right\rangle = \left\langle a\Psi_k \sum_{i=0}^{P} z_i\Psi_i \right\rangle - \left\langle c\Psi_k \sum_{i=0}^{P} u_i\Psi_i \right\rangle$$

$$- \left\langle 4d\Psi_k \sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j \right\rangle$$

Surface Reaction Model: Reorganize terms

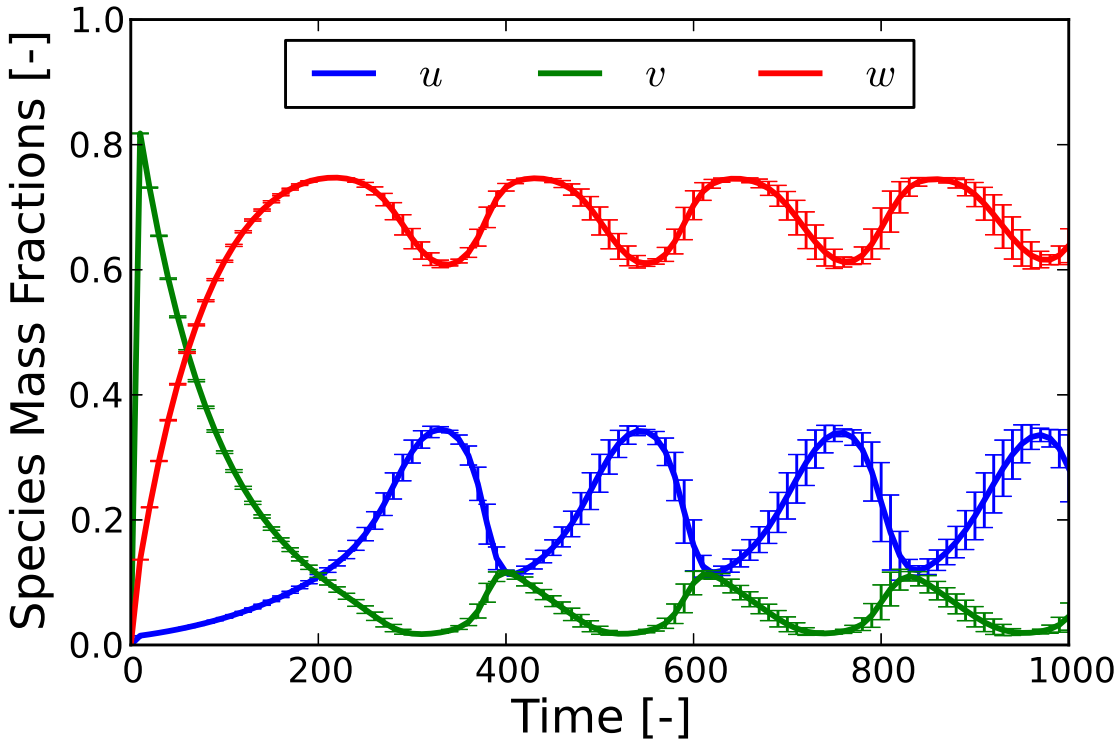
$$\frac{\mathrm{d}}{\mathrm{d}t} u_k \left\langle \Psi_k^2 \right\rangle = a z_k \left\langle \Psi_k^2 \right\rangle - c u_k \left\langle \Psi_k^2 \right\rangle - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \left\langle \Psi_i \Psi_j \Psi_k \right\rangle$$

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k = a z_k - c u_k - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k = a z_k - c u_k - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j C_{ijk}$$

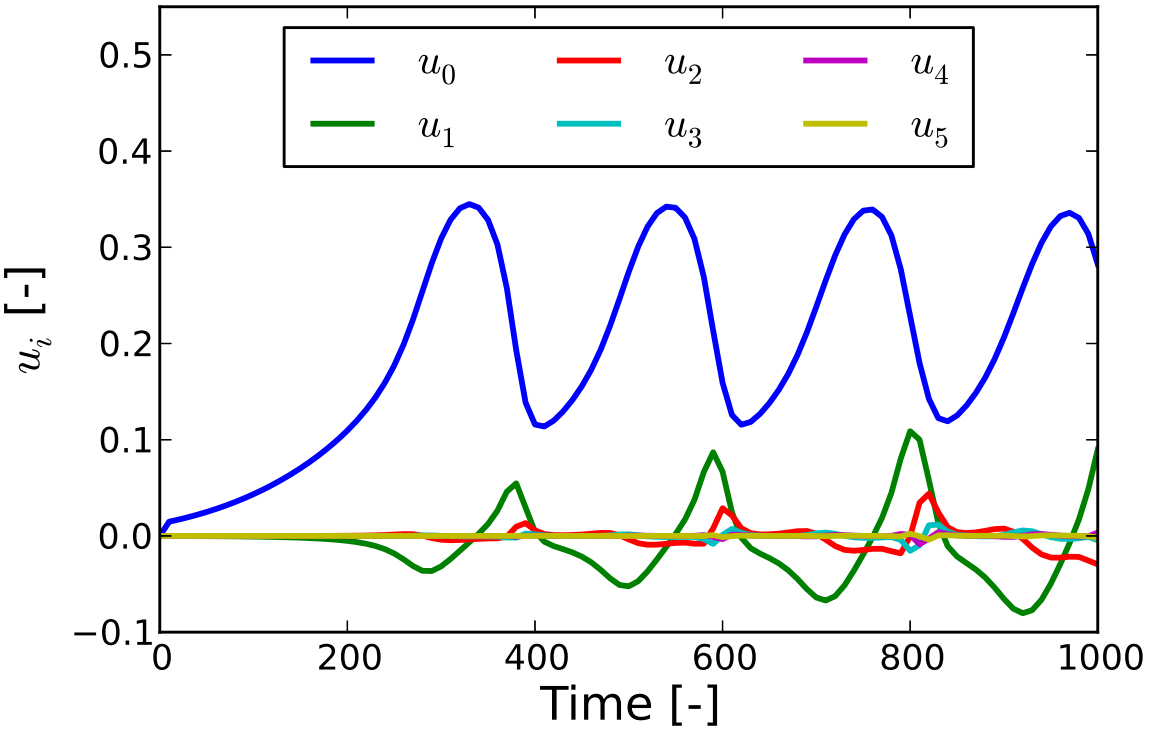
- Triple products $C_{ijk} = \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle}$ can be pre-computed and stored for repeated use

Surface Reaction Model: ISP results



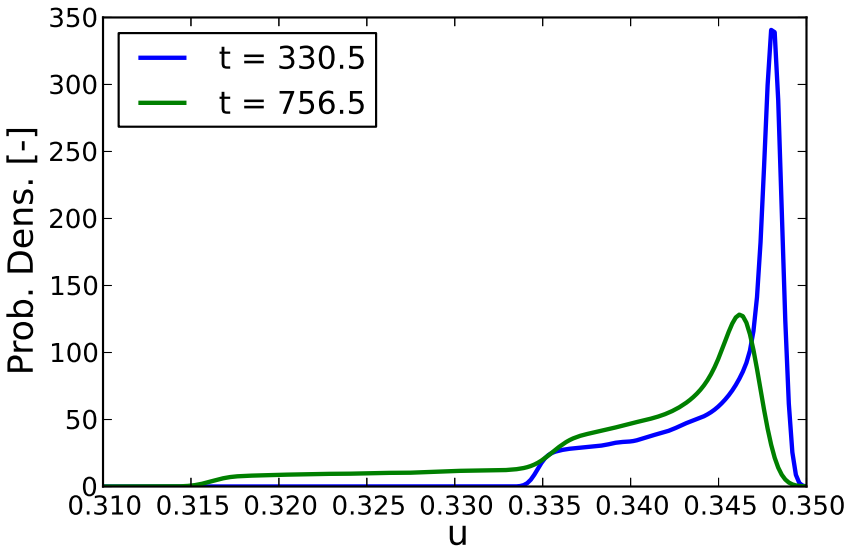
- Assume 0.5% uncertainty in *b* around nominal value
- Legendre-Uniform intrusive PC
- Mean and standard deviation for *u*, *v*, and *w*
- Uncertainty grows in time

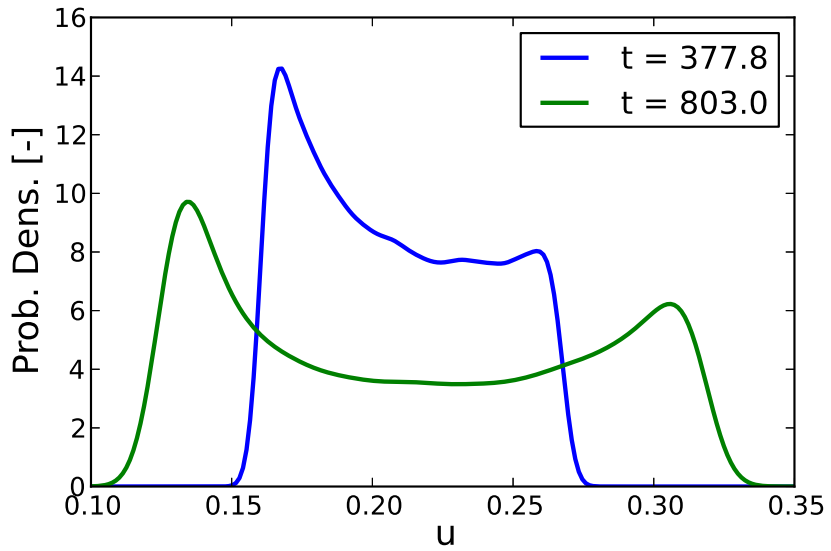
Surface Reaction Model: ISP results



- Modes of *u*
- Modes decay with higher order
- Amplitudes of oscillations of higher order modes grow in time

Surface Reaction Model: ISP results: PDFs





- Pdfs of u at maximum mean (left) and maximum standard deviation (right)
- Distributions get broader and multimodal as time increases
  - Effect of accumulating uncertainty in phase of oscillation

Remainder of this section focuses on Galerkin projection methods

- Intrusive Galerkin projection
- Non-intrusive Galerkin projection

Non-intrusive Galerkin projection

$$u_k = \frac{\langle u\Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{1}{\langle \Psi_k^2 \rangle} \int u\Psi_k(\xi) w(\xi) d\xi, \quad k = 0, \ldots, P$$

Evaluate projection integrals numerically

- Pick samples of uncertain parameters, e.g. $b(\xi) = \sum b_k \psi_k(\xi)$ by sampling ξ
- Run deterministic forward model for each of the sampled input parameter values $b^i = b(\xi^i)$
- Integration using random sampling or quadrature methods

Reconstruct uncertain model output

$$u(x, t; \theta) = \sum_{k=0}^{P} u_k(x, t)\Psi_k(\xi(\theta))$$

Random sampling approaches for Galerkin projection

- Evaluate integral through sampling

$$\int u\Psi_k(\xi)w(\xi)d\xi = \frac{1}{N_s}\sum_{i=1}^{N_s} u(\xi^i)\Psi_k(\xi^i)$$

- Samples are drawn according to the distribution of ξ
  - Monte-Carlo (MC)
  - Latin-Hypercube-Sampling (LHS)
- Pros:
  - Can be easily made fault tolerant
  - Sometimes random samples is all we have
- Cons: slow convergence, but less dependent on number of stochastic dimensions

Quadrature approaches for Galerkin projection

- Use numerical quadrature rules to evaluate integrals

$$\int u\Psi_k(\xi)w(\xi)d\xi = \sum_{i=1}^{N_q} q^i u(\xi^i)\Psi_k(\xi^i)$$

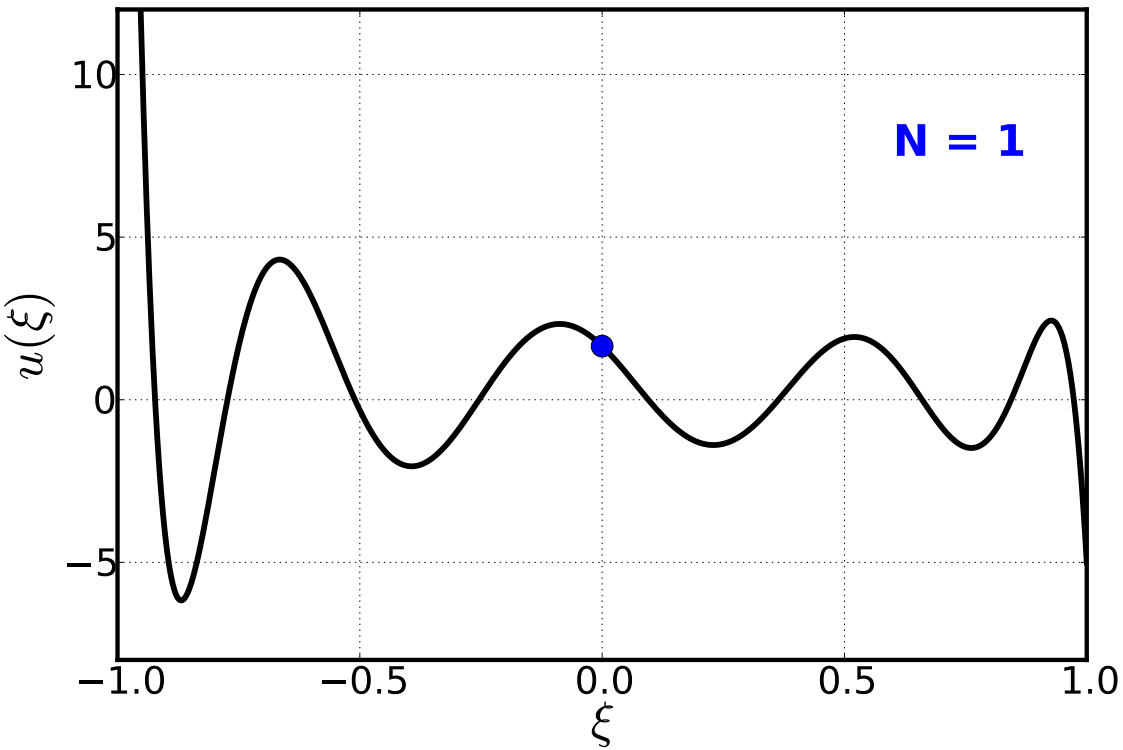
- The N_q ξ^i are quadrature points, with corresponding weights q^i
- Choice of quadrature points important for accuracy
  - Also referred to as *deterministic sampling* approach
- Pros:
  - Can use existing codes as black box to evaluate $u(\xi^i)$
  - Embarrassingly parallel
- Cons: Tensor product rule for d dimensions requires N_q^d samples

Gauss quadrature rules are very efficient

$$\int u\Psi_k(\xi)w(\xi)d\xi = \sum_{i=1}^{N_q} q^i u(\xi^i)\Psi_k(\xi^i)$$

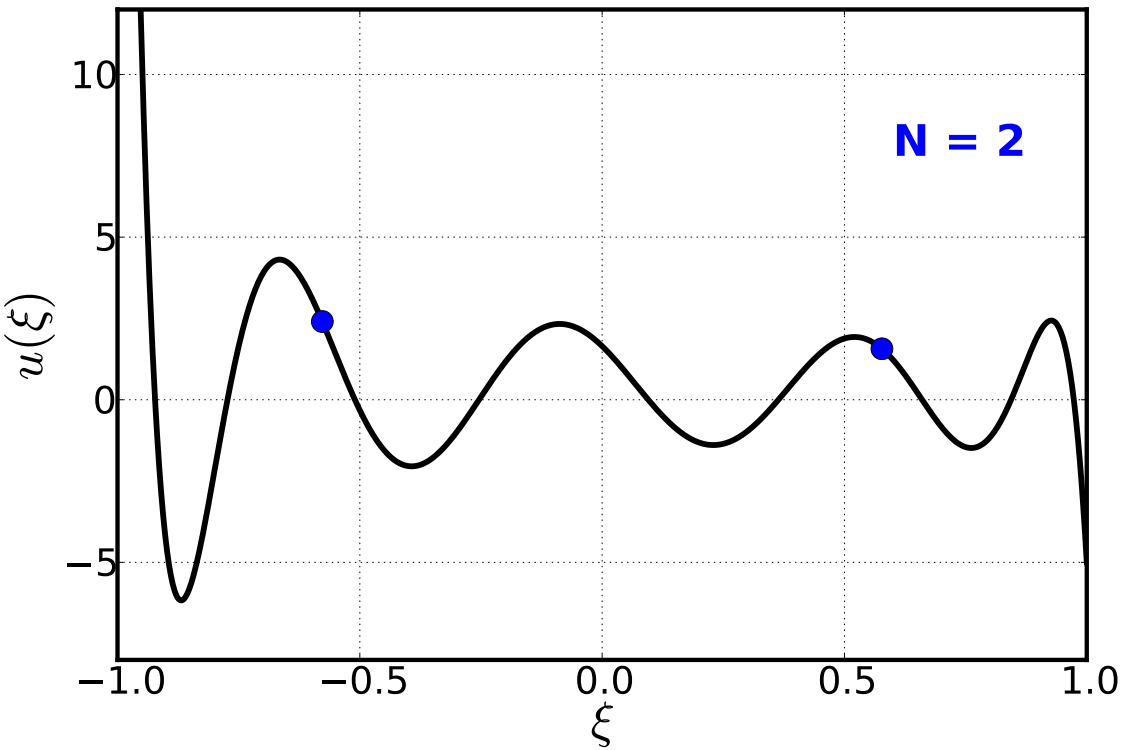
- N_q quadrature points can integrate polynomial of order $2N_q - 1$ exactly
- Gauss-Hermite and Gauss-Legendre quadrature tailored to specific choices of the weight function $w(\xi)$

Example quadrature integration of polynomial



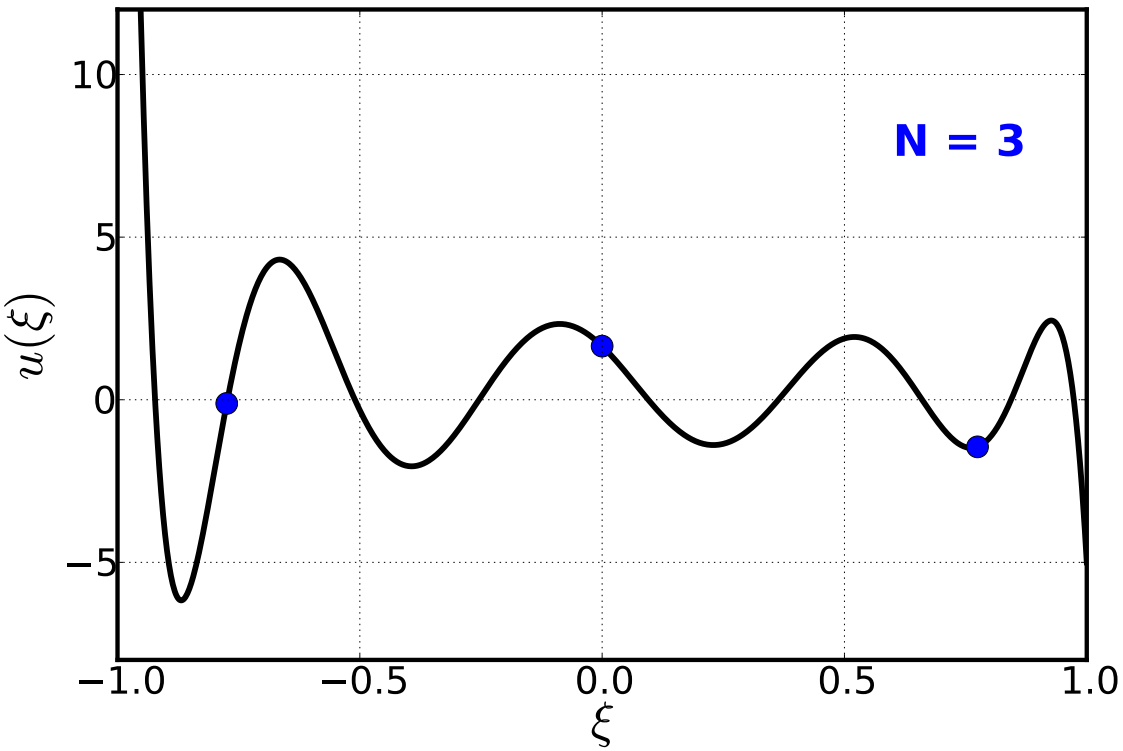
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 1 quadrature point gives integral = 1.65

Example quadrature integration of polynomial



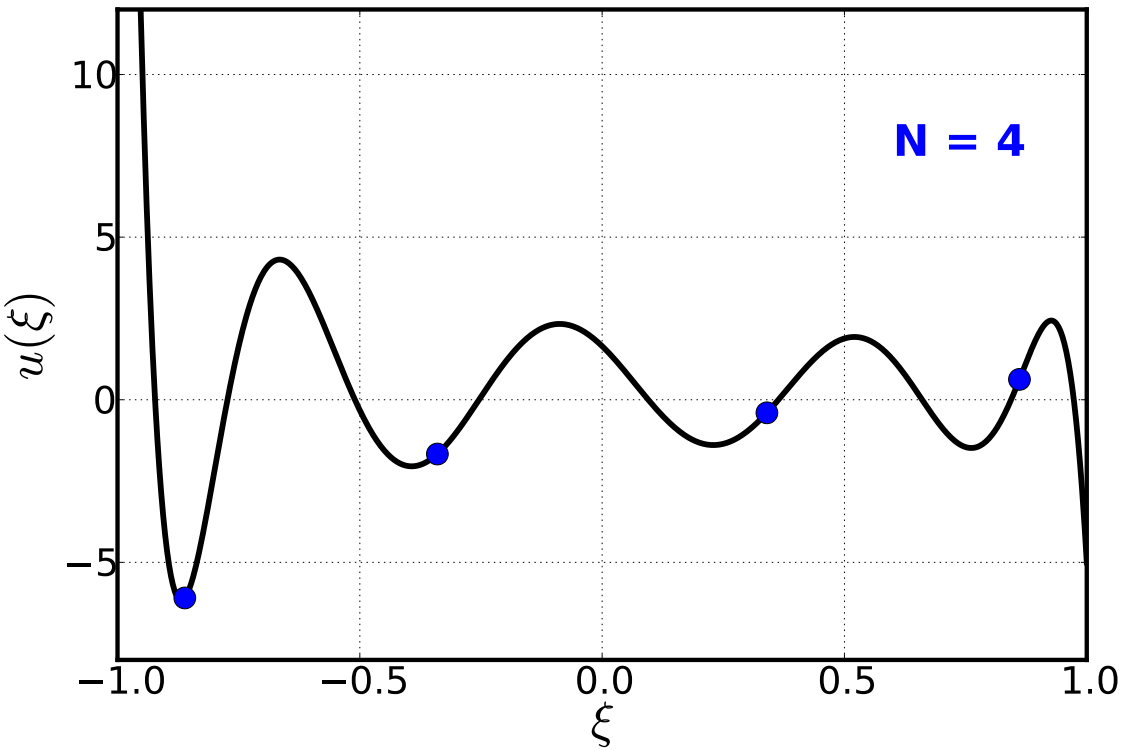
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 2 quadrature points gives integral = 1.99

Example quadrature integration of polynomial



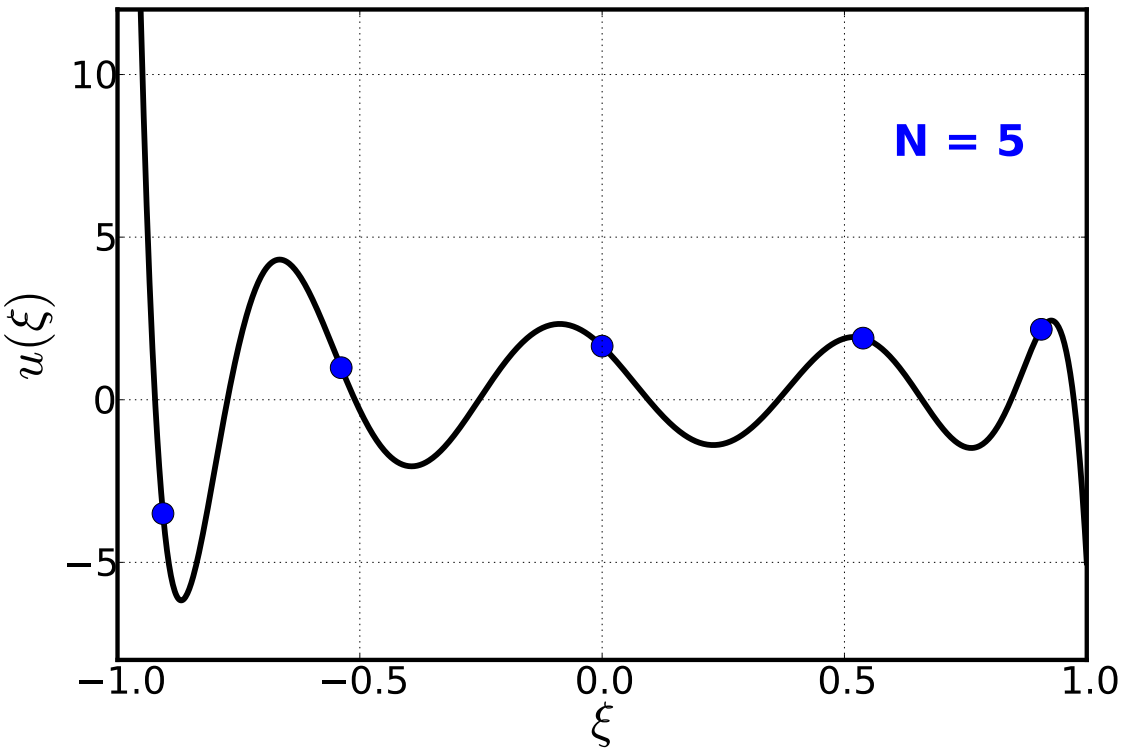
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 3 quadrature points gives integral = 0.30

Example quadrature integration of polynomial



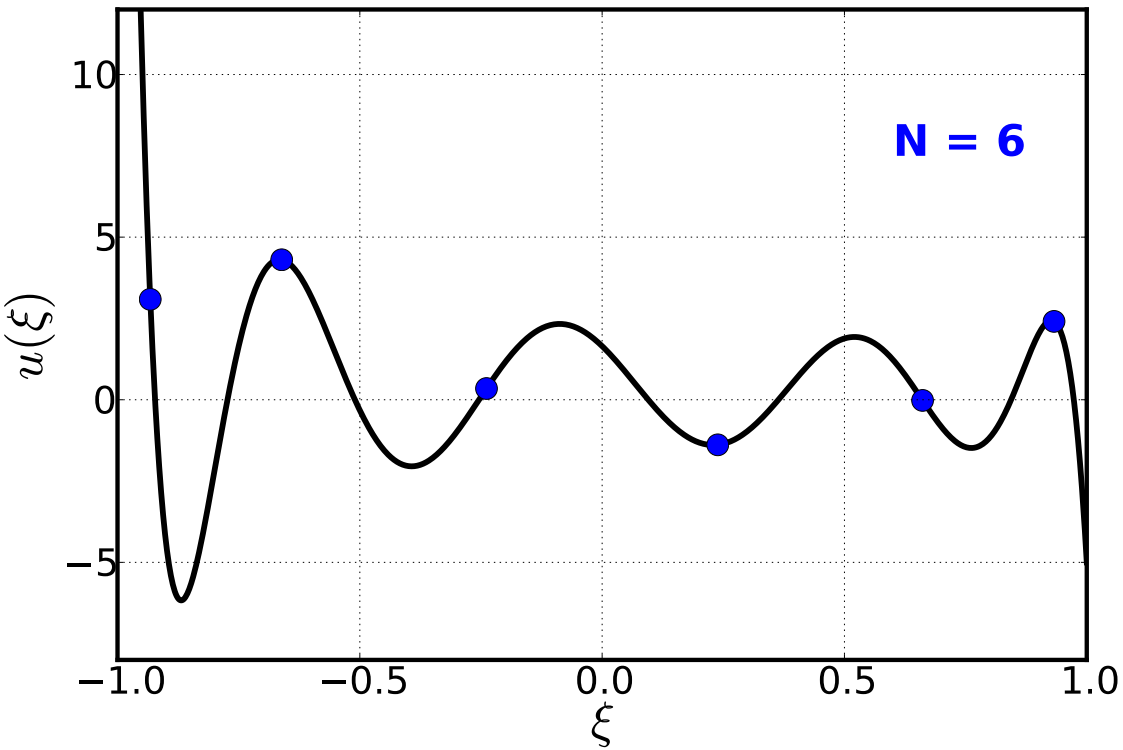
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 4 quadrature points gives integral = -1.63

Example quadrature integration of polynomial



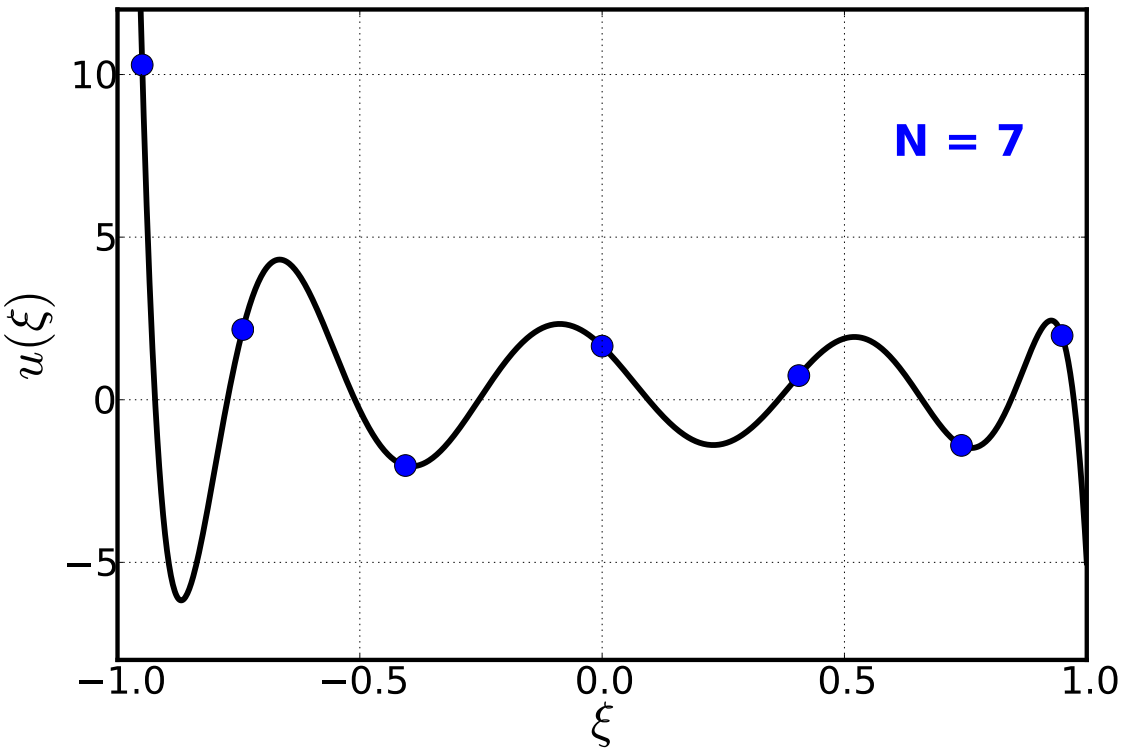
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 5 quadrature points gives integral = 1.00

Example quadrature integration of polynomial



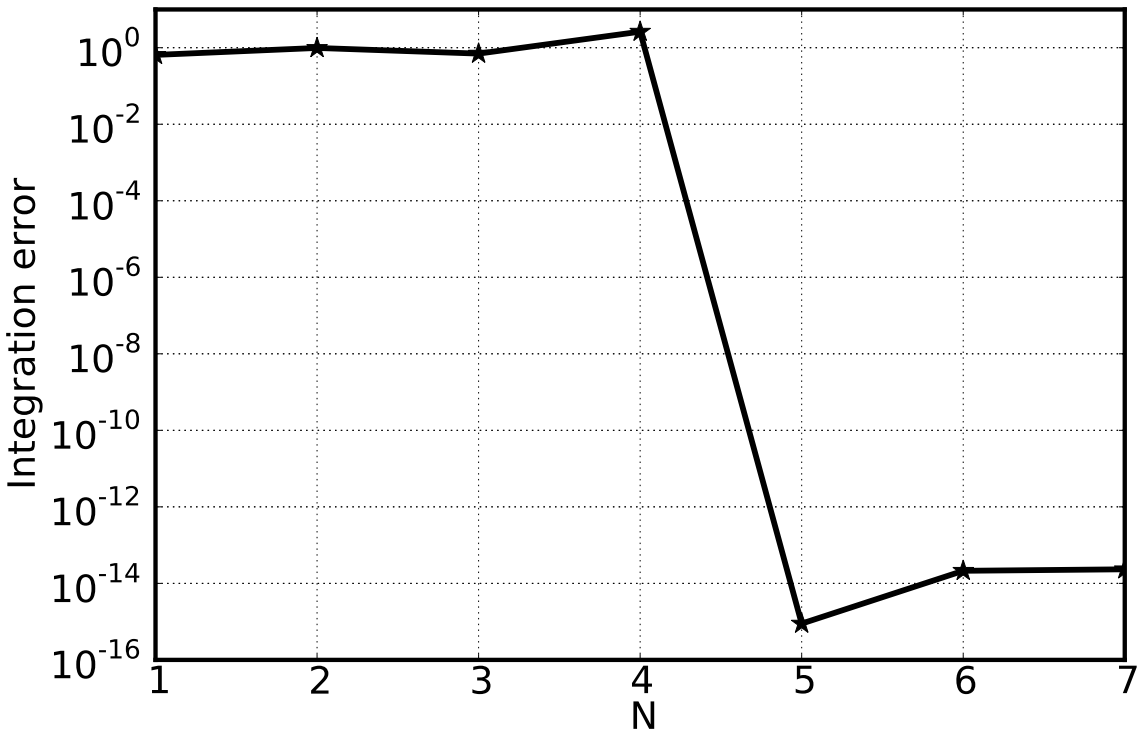
- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 6 quadrature points gives integral = 1.00

Example quadrature integration of polynomial



- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 7 quadrature points gives integral = 1.00

Example quadrature integration of polynomial



- Integral of 9^{th} order polynomial
- Gauss-Legendre quadrature with 5 or more points is exact

Minimum number of quadrature points for Galerkin projection

$$\int u\Psi_k(\xi)w(\xi)d\xi = \sum_{i=1}^{N_q} q^i u(\xi^i)\Psi_k(\xi^i)$$

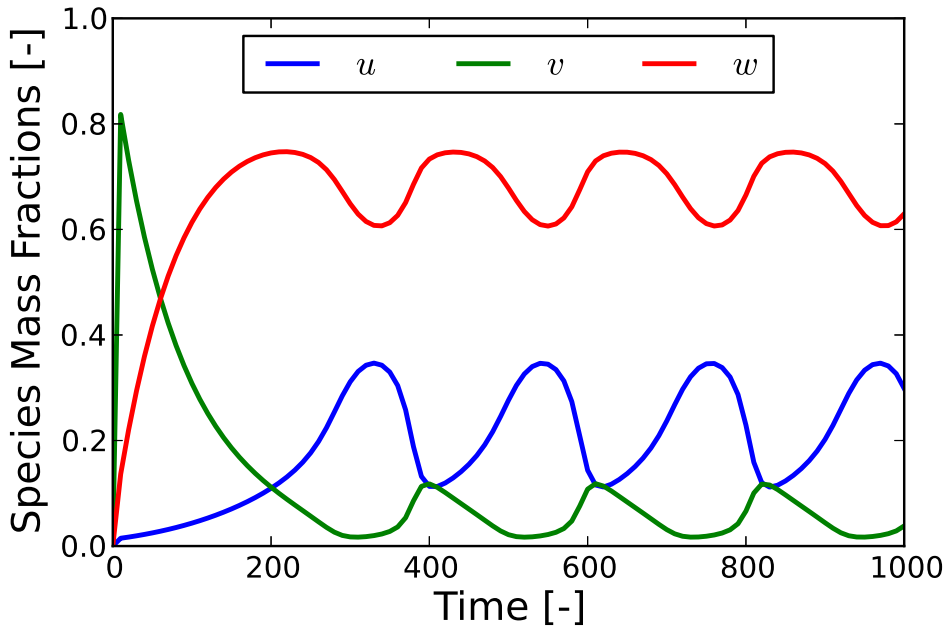
- As a rule of thumb, $p+1$ quadrature points are needed for Galerkin projection of PCE of order p
  - If both u and Ψ_k are of order p, then integrand is of order $2p$
  - $2p \leq 2N_q - 1$ or $N_q \geq p + \frac{1}{2}$
  - Only exact if u is indeed a polynomial of order $\leq p$

Surface Reaction Model

3 ODEs for a monomer (u), dimer (v), and inert species (w) adsorbing onto a surface out of gas phase.

$$\begin{aligned}
\frac{\mathrm{d}u}{\mathrm{d}t} &= az - cu - 4duv \\
\frac{\mathrm{d}v}{\mathrm{d}t} &= 2bz^2 - 4duv \\
\frac{\mathrm{d}w}{\mathrm{d}t} &= ez - fw \\
z &= 1 - u - v - w
\end{aligned}$$

$$u(0) = v(0) = w(0) = 0.0$$

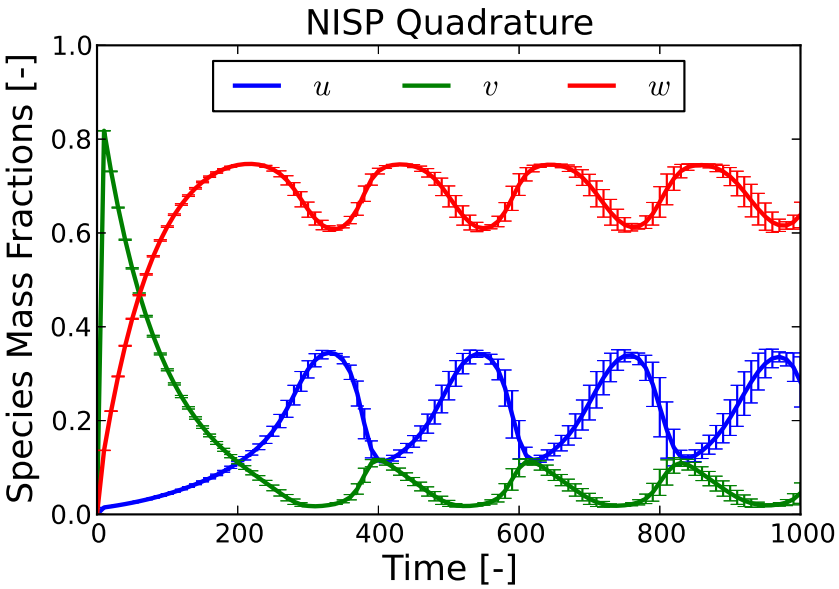


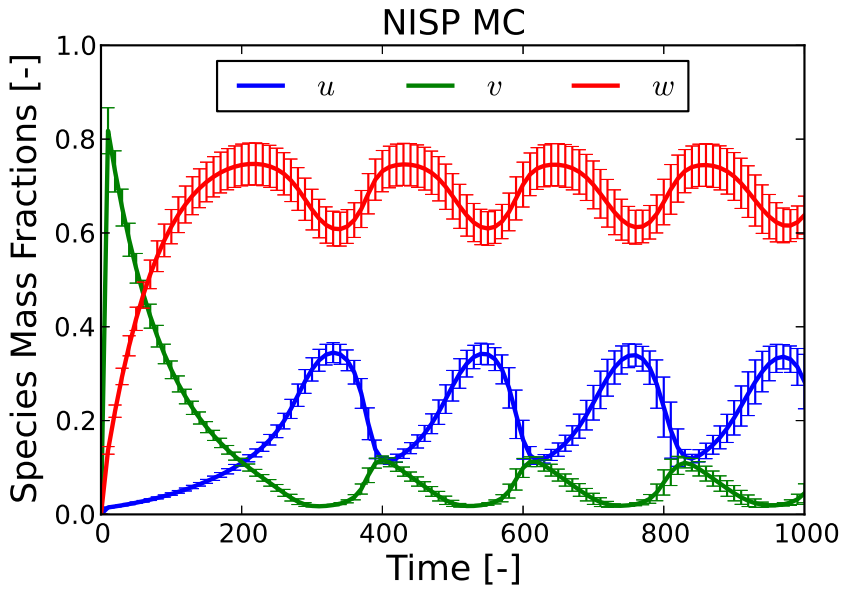
Oscillatory behavior for $b \in [20.2, 21.2]$

$$\begin{aligned}
a &= 1.6 \quad b = 20.75 \quad c = 0.04 \\
d &= 1.0 \quad e = 0.36 \quad f = 0.016
\end{aligned}$$

[Vigil *et al.*, Phys. Rev. E., 1996; Makeev *et al.*, J. Chem. Phys., 2002]

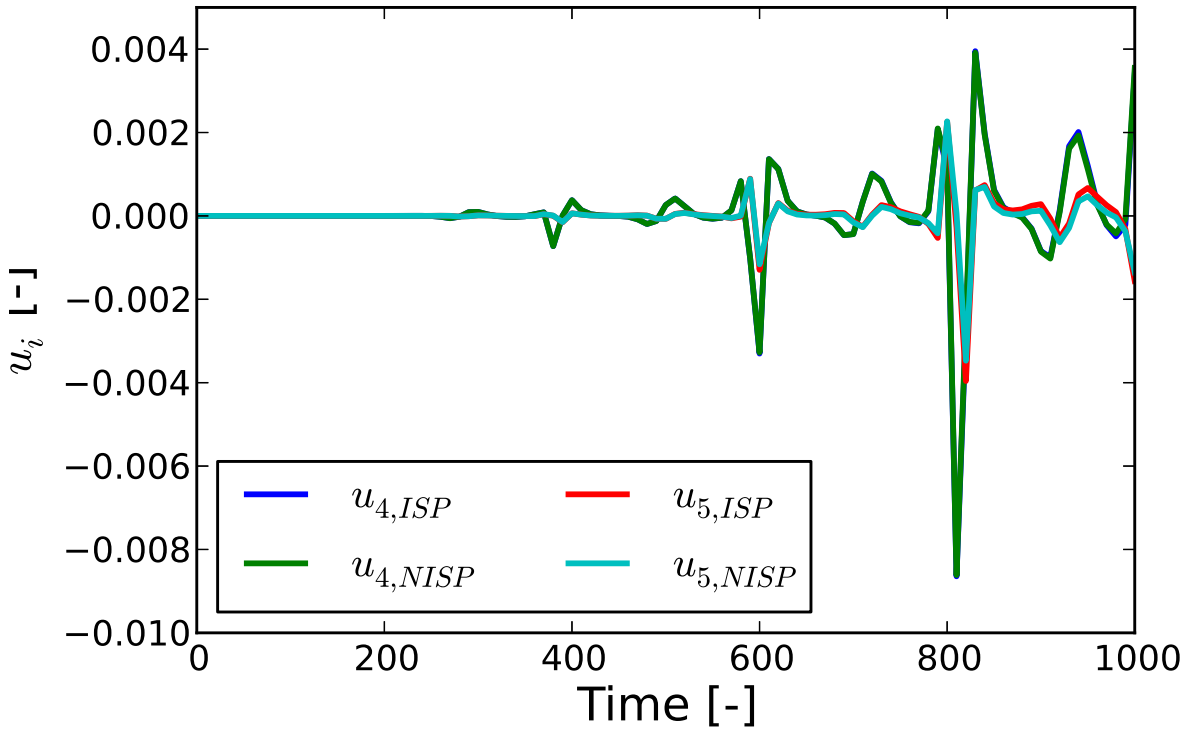
Surface Reaction Model: NISP results





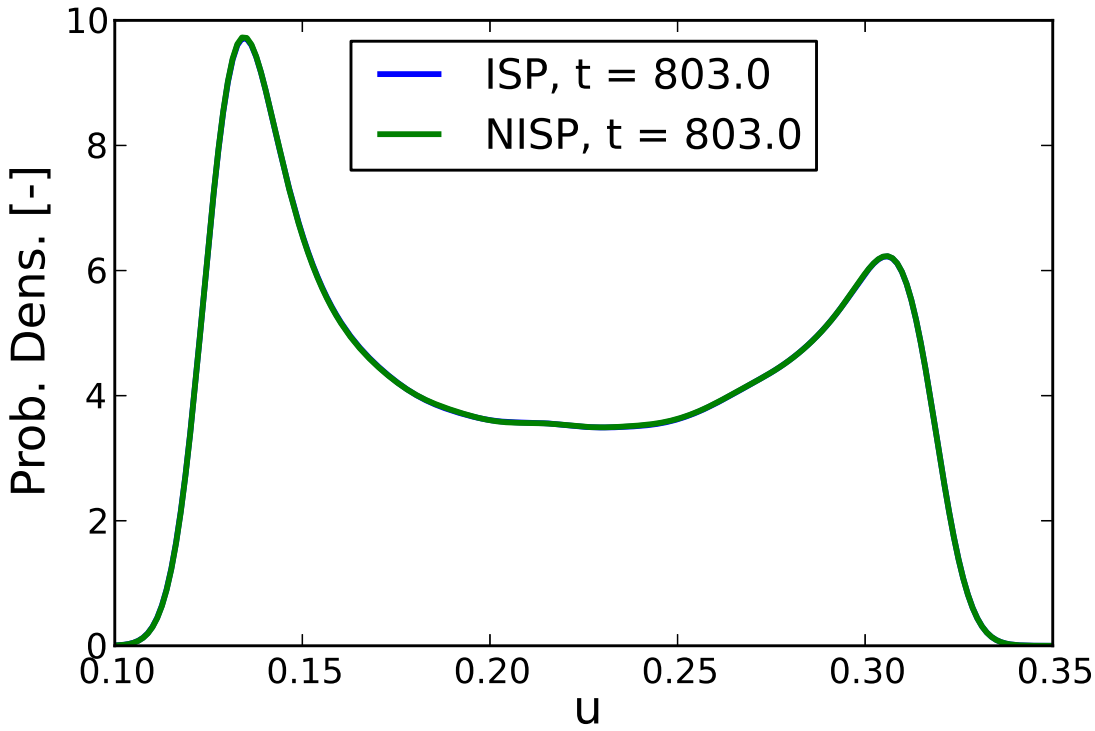
- Mean and standard deviation for u, v, and w
- Quadrature approach agrees well with ISP approach using 6 quadrature points
- Monte Carlo sampling approach converges slowly
  - With a 1000 samples, results are quite different from ISP and NISP

Surface Reaction Model: Comparison ISP and NISP



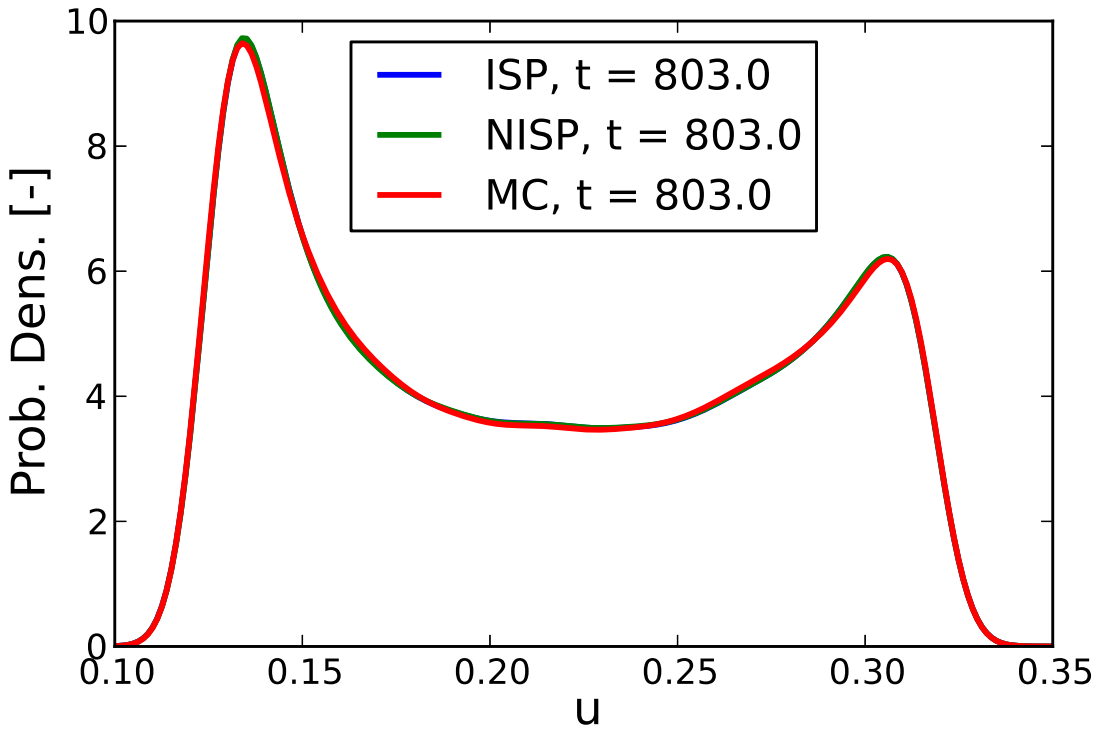
- Lower order modes agree perfectly
- Very small differences in higher order modes
  - Difference increases with time

Surface Reaction Model: Comparison ISP and NISP



- All pdf's based on 50K samples each and evaluated with Kernel Density Estimation (KDE)
- No difference in PDFs of sampled PCEs between NISP and ISP

Surface Reaction Model: Comparison ISP, NISP, and MC



- All pdf's based on 50K samples each and evaluated with Kernel Density Estimation (KDE)
- Good agreement between intrusive, non-intrusive projection, and Monte Carlo sampling

ISP pros and cons

- Pros:
  - Elegant
  - One time solution of system of equations for the PC coefficients fully characterizes uncertainty in all variables at all times
  - Tailored solvers can (potentially) take advantage of new hardware developments
- Cons:
  - Often requires re-write of the original code
  - Reformulated system is factor (P+1) larger than the original system and can be challenging to solve
  - Challenges with increasing time-horizon for ODEs
- Many efforts in the community to automate ISP

NISP pros and cons

- Pros:
  - Easy to use as wrappers around existing codes
  - Embarassingly parallel
  - Can be used even when there is no explicit equation for the observable
- Cons:
  - Most methods suffer from curse of dimensionality $N_q = n^{N_d}$
- Many development efforts for smarter sampling approaches and dimensionality reduction
  - (Adaptive) Sparse Quadrature approaches
  - Compressive Sensing
  - ...
- Sampling methods have found very wide spread use in the community

Software for intrusive and non-intrusive UQ

- DAKOTA: `http://dakota.sandia.gov/`
  - Wide variety of non-intrusive methods for uncertainty propagation, sensitivity analysis, *etc.*
  - Mature and well-supported
- UQ Toolkit (UQTk):
  `http://www.sandia.gov/UQToolkit/`
  - Intrusive and non-intrusive
  - Geared towards algorithm development and educational use
- Sundance:
  `http://www.math.ttu.edu/~klong/Sundance/html/`
  - UQ-enabled finite-element solution of PDEs
- Stokhos:
  `http://trilinos.sandia.gov/packages/stokhos/`
  - Package for intrusive Galerkin based UQ
- ...

Uncertainty Quantification Toolkit (UQTk)

- A library of C++ and Python functions for propagation of uncertainty through computational models
- Mainly relies on Polynomial Chaos Expansions (PCEs) for representing random variables and stochastic processes
- Target usage:
  - Rapid prototyping
  - Algorithmic research
  - Tutorials / educational
- Version 2.1 released under the GNU Lesser General Public License
  - C++ Tools for intrusive and non-intrusive UQ
  - Bayesian inference tools
  - Bayesian compressed sensing
  - Python postprocessing and analysis tools
- Available at `http://www.sandia.gov/UQToolkit/`

Outline

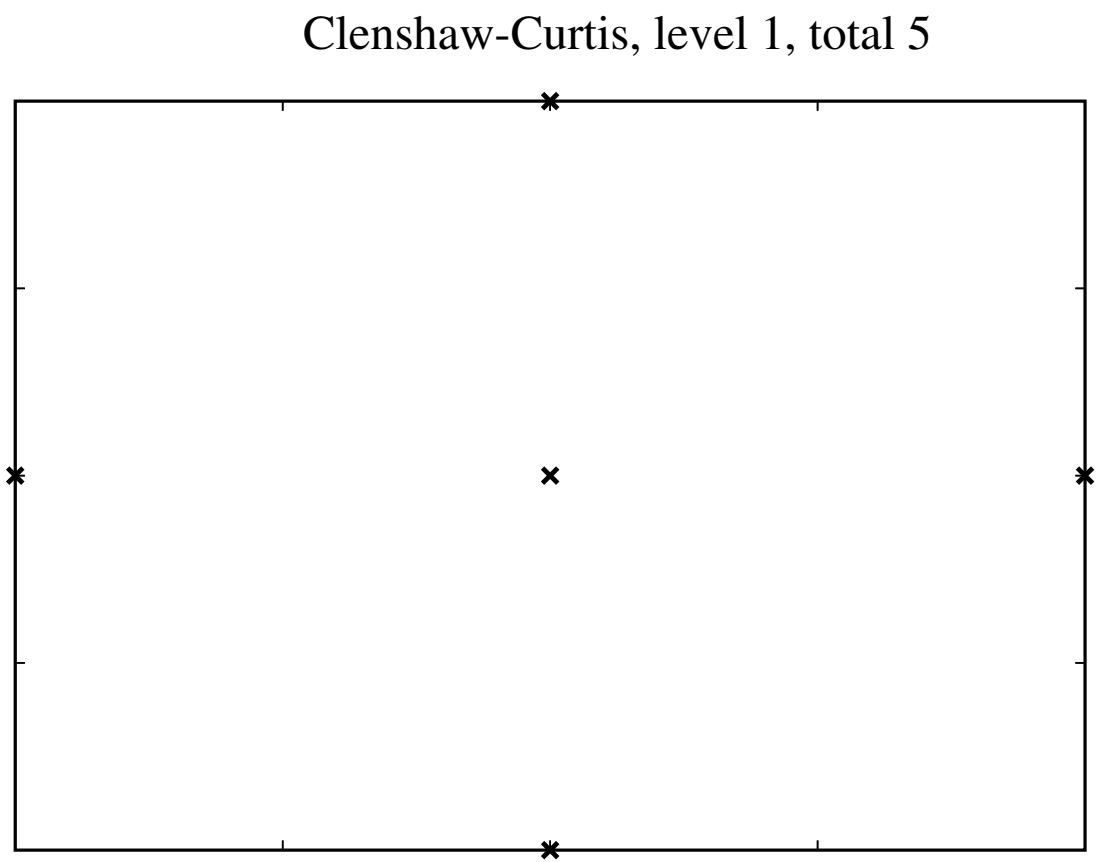
Sparse Quadrature Approaches for High-Dimensional Systems

- Need for sparse quadrature
- Sparse quadrature grids
- Application to heat transfer example

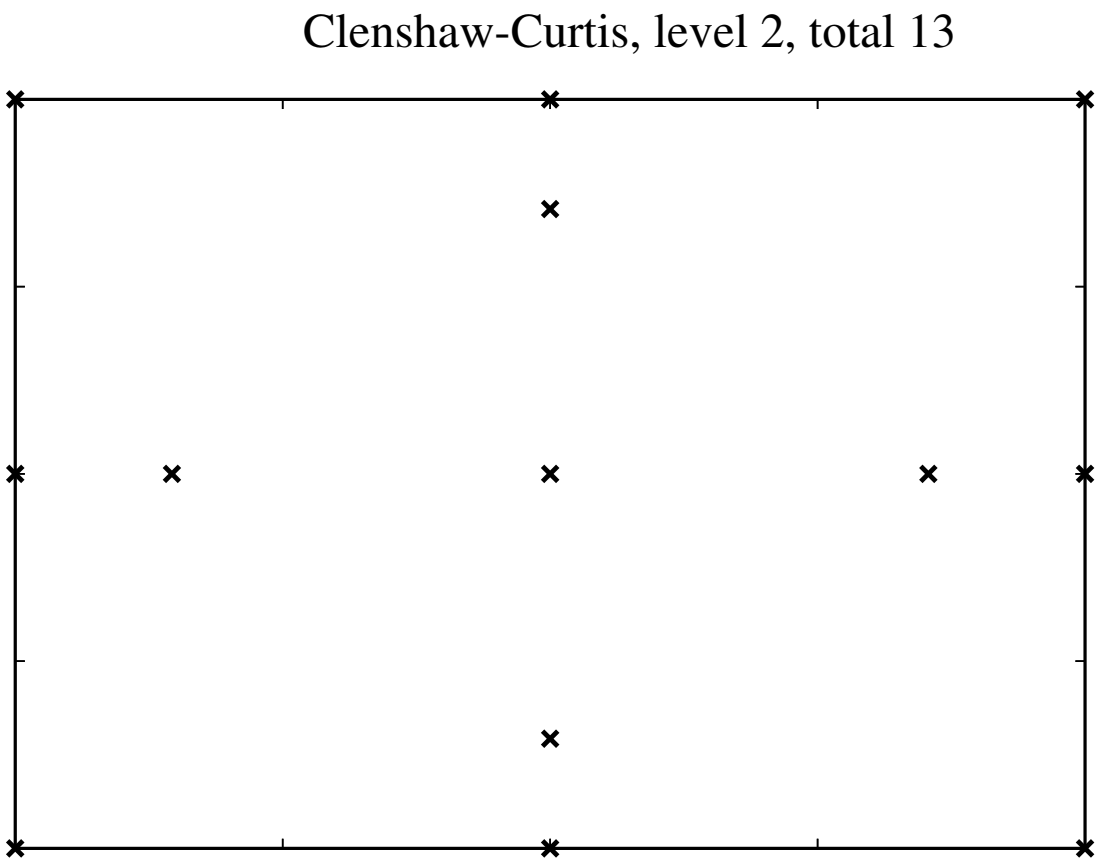
Full product quadrature is wasteful in high-dimensional systems

- Define the **precision** as the highest order of a polynomial that is integrated *exactly* by the quadrature rule
- Gaussian quadratures are optimal in 1d
  - N points achieve the highest possible precision of $2N - 1$
- In multi-d, full product quadrature is wasteful:
  - A 5 ppd (point per dimension) rule is of precision $P = 9$, but it integrates a polynomial $x^9 y^9$ exactly
- Sparse quadratures are built to achieve maximal precision with as few points as possible
  - *E.g.* Smolyak construction

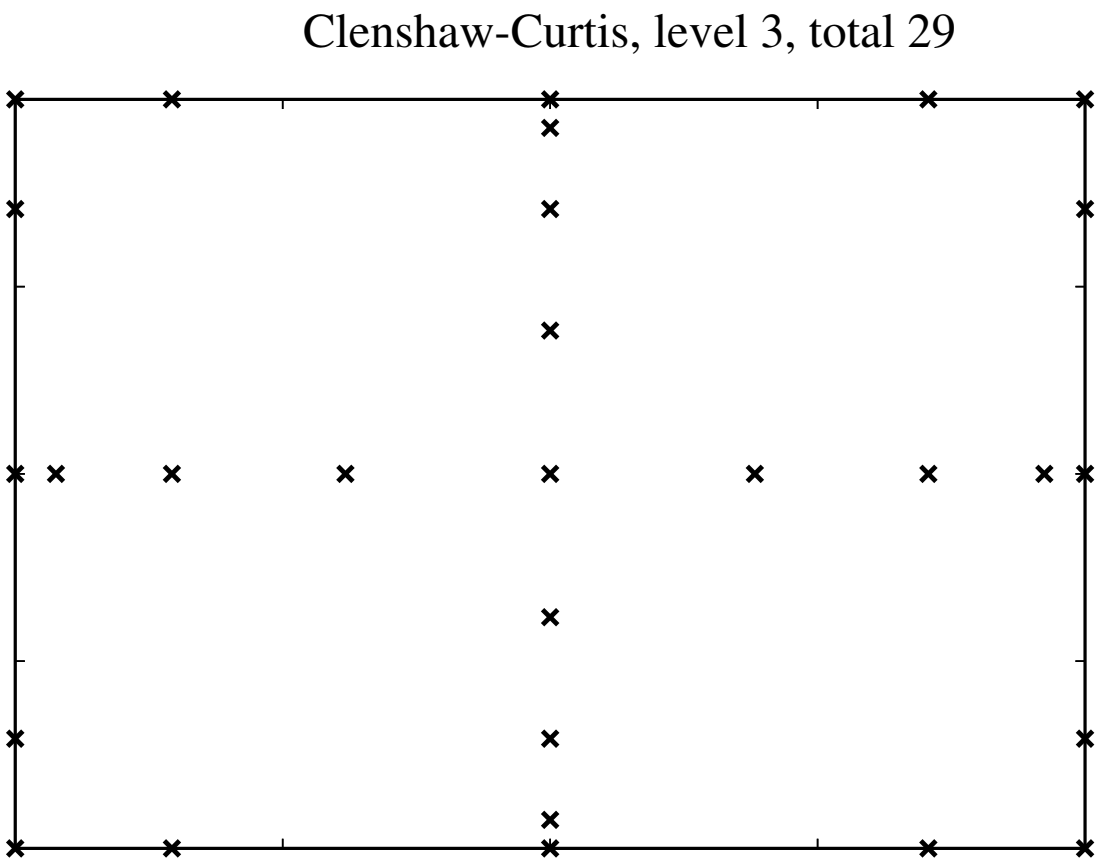
Clenshaw-Curtis nested quadrature rules allow reuse of function evalutions



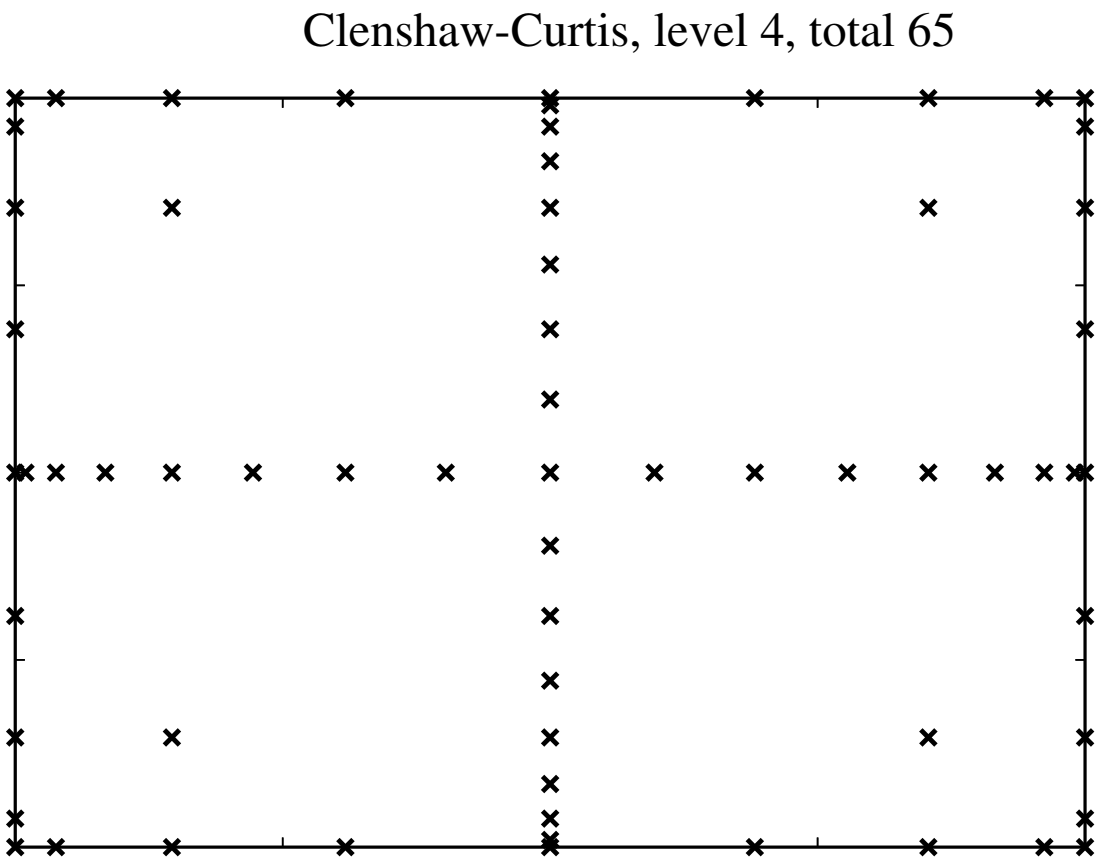
Clenshaw-Curtis nested quadrature rules allow reuse of function evalutions



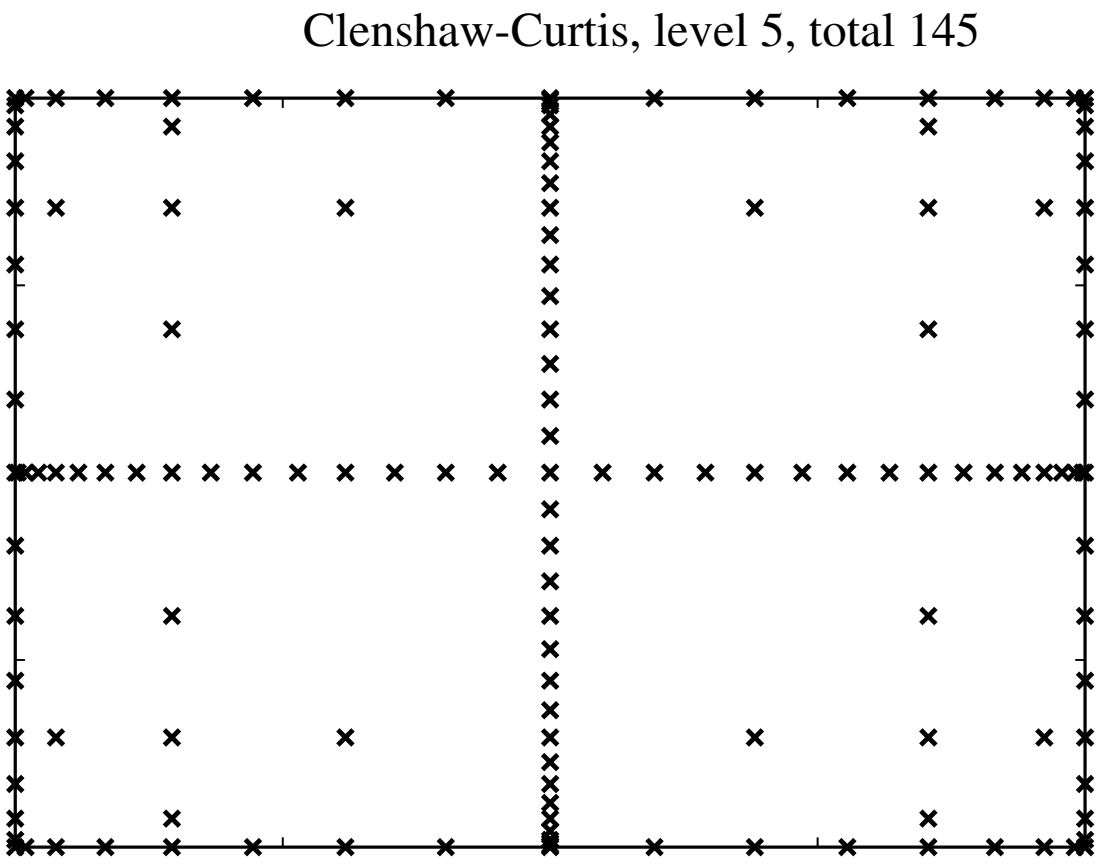
Clenshaw-Curtis nested quadrature rules allow reuse of function evalutions



Clenshaw-Curtis nested quadrature rules allow reuse of function evalutions



Clenshaw-Curtis nested quadrature rules allow reuse of function evalutions

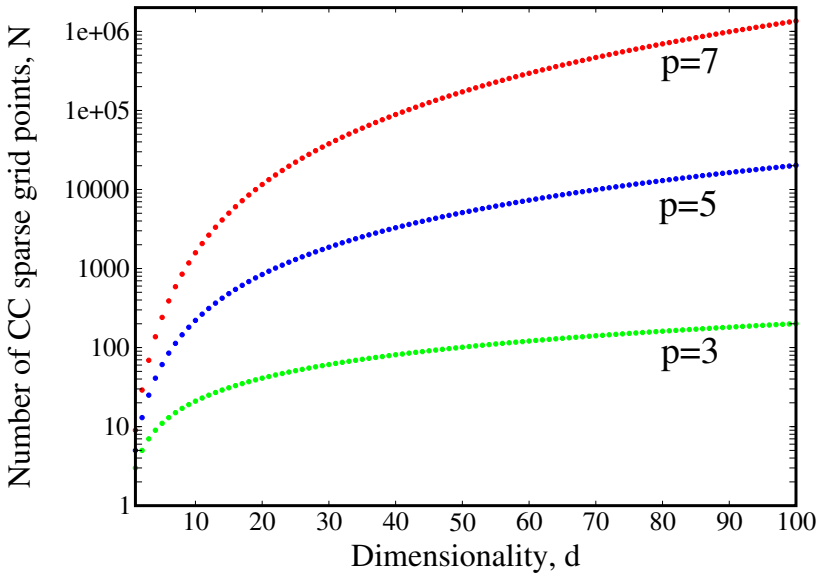


The number of function evaluations is drastically reduced compared to full quadrature

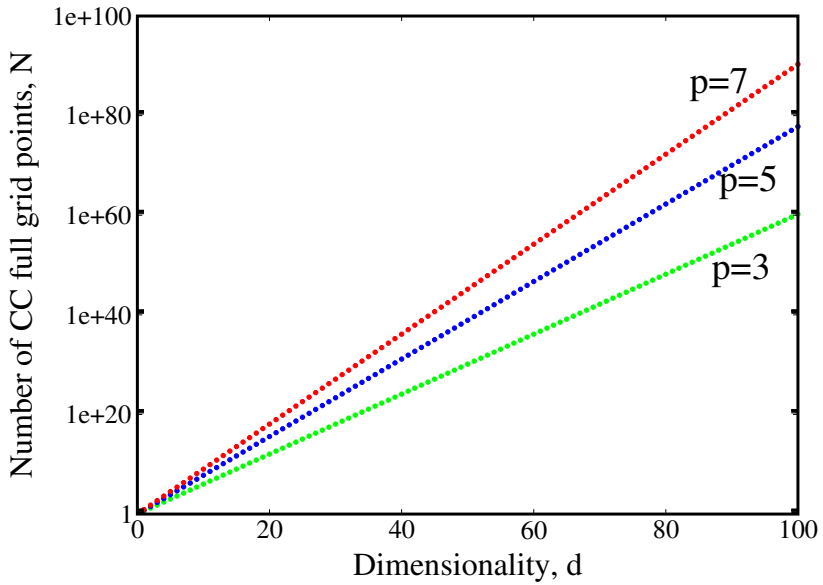
Table : The number of Clenshaw-Curtis sparse grid quadrature points for various levels and dimensionalities.

Level L	Precision $p = 2L - 1$	N $(d = 2)$	N $(d = 5)$	N $(d = 10)$	$N(d)$ General
1	1	1	1	1	1
2	3	5	11	21	$1 + 2d$
3	5	13	61	221	$1 + 2d + 2d^2$
4	7	29	241	1581	$1 + \frac{14}{3}d + 2d^2 + \frac{4}{3}d^3$
5	9	65	801	8801	$1 + \frac{20}{3}d + \frac{22}{3}d^2 + \frac{4}{3}d^3 + \frac{2}{3}d^4$
6	11	145	2433	41265	$\cdots$
7	13	321	6993	171425	$\cdots$
8	15	705	19313	652065	$\cdots$
9	17	1537	51713	2320385	$\cdots$

The number of function evaluations is drastically reduced compared to full quadrature

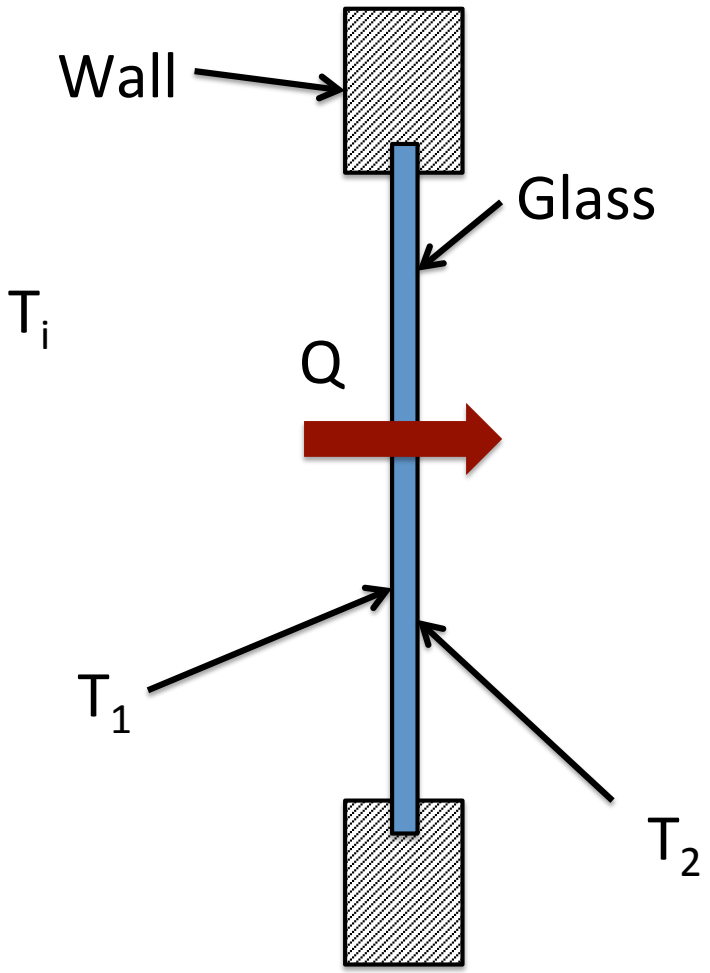


Sparse grid: polynomial growth, $\mathcal{O}(d^{\frac{p-1}{2}})$



Full grid: exponential growth, $(p+1)^d$

Heat Transfer through a Window



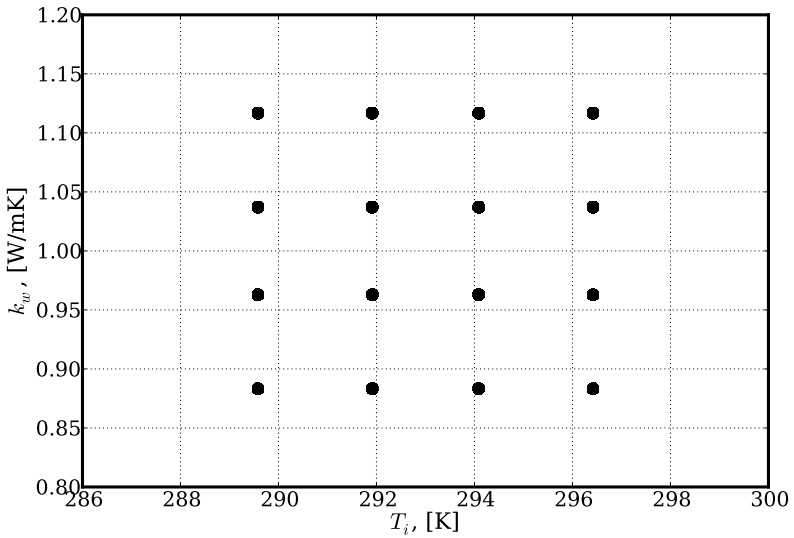
$$h_i(T_i - T_1) = k_w \frac{(T_1 - T_2)}{d_w}$$

$$k_w \frac{(T_1 - T_2)}{d_w} = h_o(T_2 - T_o)$$

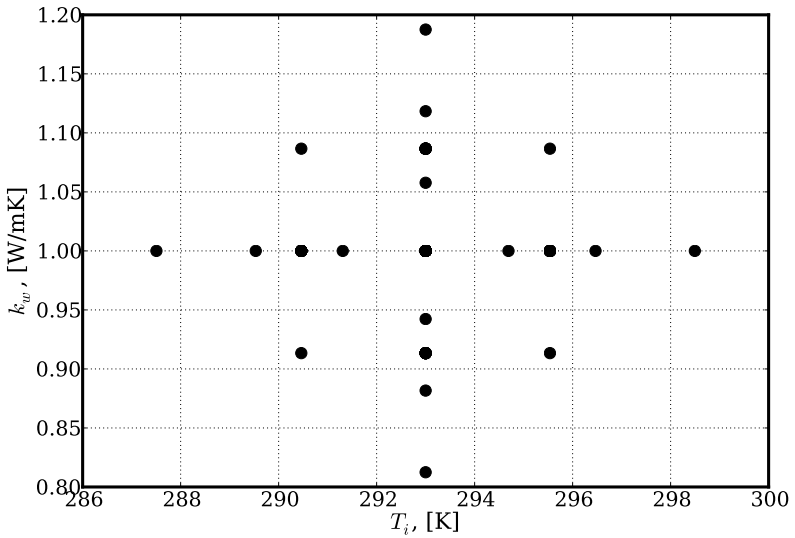
6 Uncertain, Gaussian parameters

$$T_i = 293\mathrm{K}, \sigma = 0.5\%$$
$$T_o = 273\mathrm{K}, \sigma = 0.5\%$$
$$d_w = 0.01\mathrm{m}, \sigma = 1\%$$
$$k_w = 1\mathrm{W/mK}, \sigma = 5\%$$
$$h_i = 2\mathrm{W/m^2K}, \sigma = 15\%$$
$$h_o = 6\mathrm{W/m^2K}, \sigma = 15\%$$

Sparse quadrature grid uses much fewer points than full tensor product



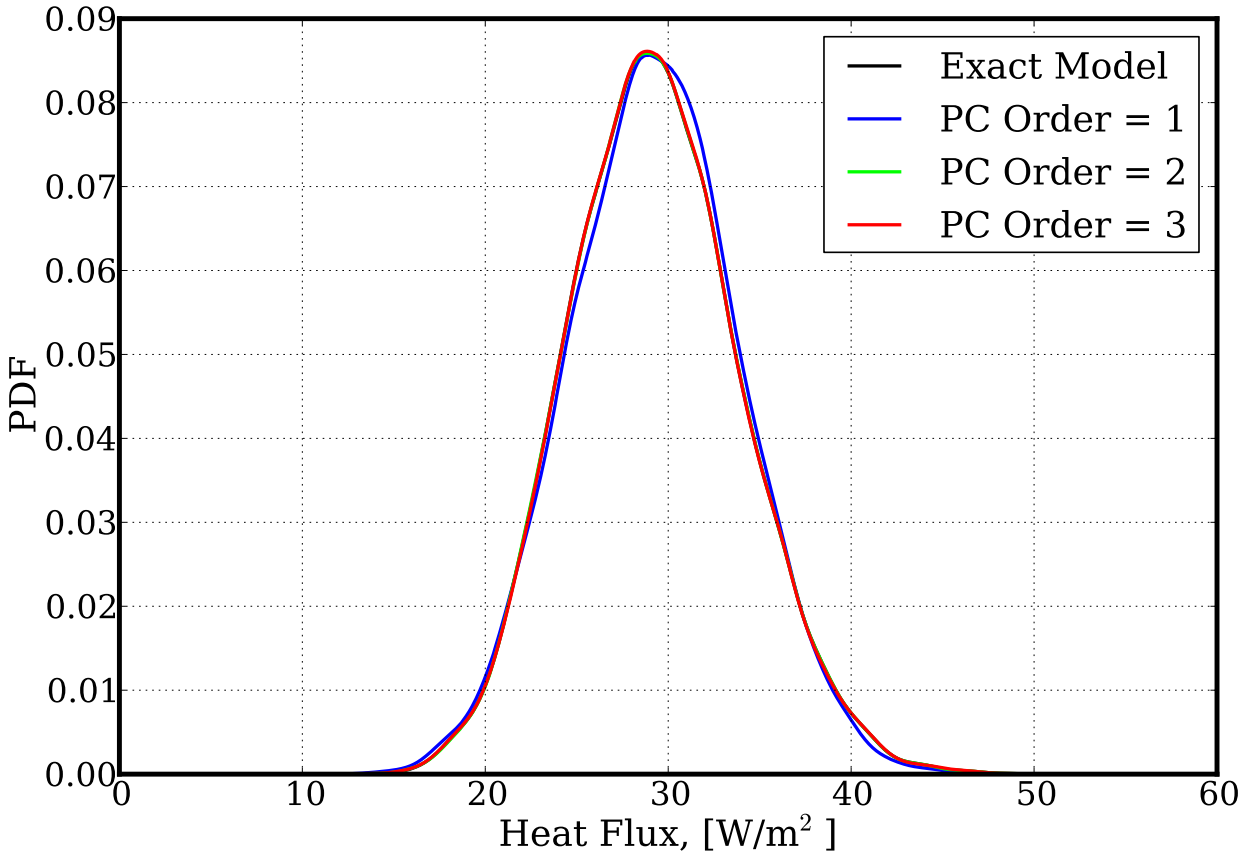
Full tensor product in 6D:
$N = 4^6 = 4096$



Level 2 sparse rule in 6D:
$N = 109$

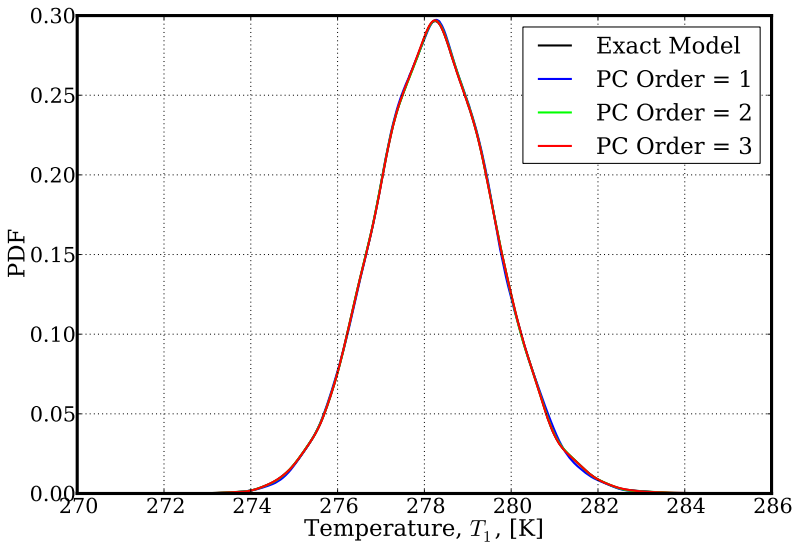
- Wiener-Hermite base rule in both cases
- Constructed for exact 3^{rd} order Galerkin projection

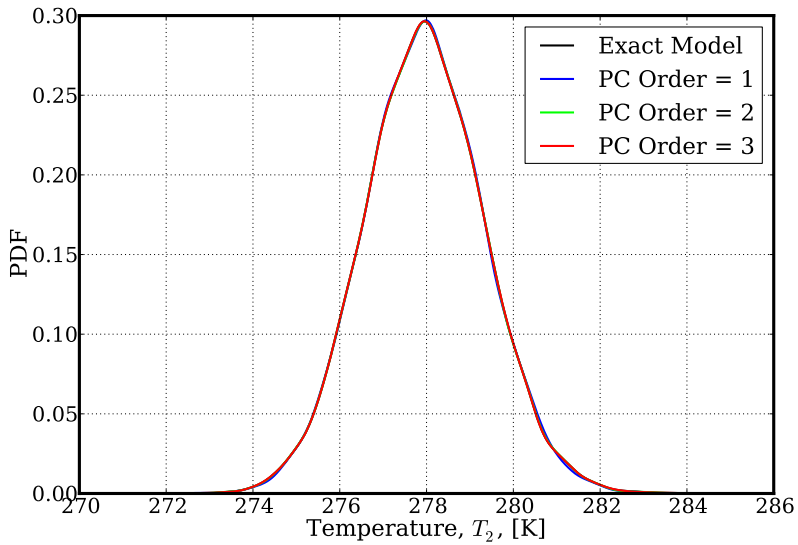
Second order PC is sufficient for forward propagation



- Second and third order PCE results coincide

Second order PC is sufficient for forward propagation





- Not much nonlinearity in temperatures
- Temperature drop across glass is small

Advantages and caveats of sparse quadrature approaches

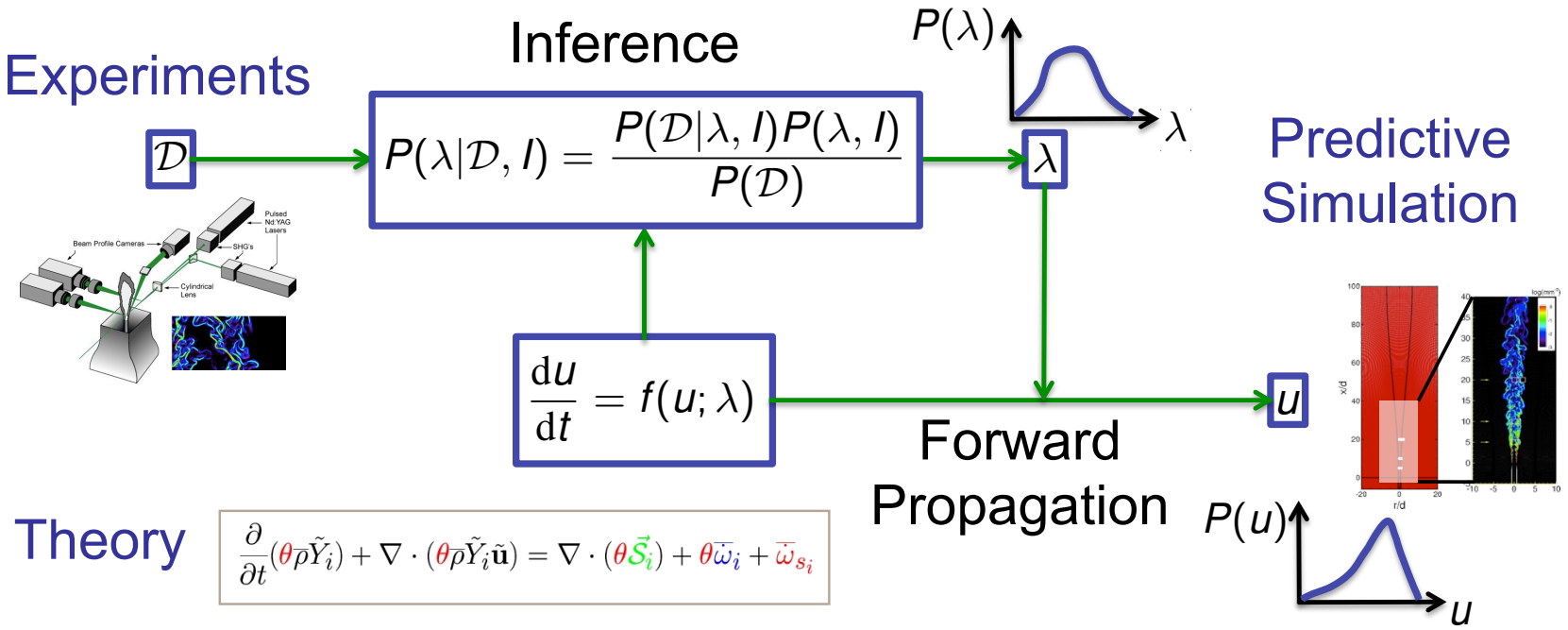
- Pro: number of required samples scales much more gracefully with number of dimensions than full tensor product quadrature rule
- Caveats:
  - Function to be integrated needs to be smooth
  - Due to negative quadrature weights, integrating a noisy positive function can give a negative answer
  - For very high dimensions, even sparse quadrature is too expensive

Taking Advantage of Sparsity in the System

- For really high dimensional systems, even sparse quadrature requires too many function evaluations
  - For 80-dimensional climate land model, $L = 4$ requires $\approx 10^6$ points
- Such systems can only be tackled with dimensionality reduction and/or adaptive order
  - Sensitivity analysis
  - High Dimensional Model Representation (HDMR)
  - Adaptive sparse quadrature approaches
- More generally, use only the basis terms needed to represent the physics / information in the system / data
  - (Bayesian) Compressive Sensing (CS) approaches
- If information content is sparse, it can be represented at reasonable cost
  - If not, you need to pay the price

Outline

Sensitivity analysis gives insight into key sources of uncertainty



- Obtaining global sensitivity analysis from PCEs
  - Identify dominant sources of uncertainty
  - Attribution

PC postprocessing: global sensitivity information is readily obtained from PCE

$$g(\xi_1, \ldots, \xi_d) = \sum_{k=0}^{P} c_k \Psi_k(\boldsymbol{\xi})$$

- Global sensitivity analysis $\equiv$ Variance decomposition
- Total variance

$$Var[g(\boldsymbol{\xi})] = \sum_{k>0} c_k^2 ||\Psi_k||^2$$

PC postprocessing: global sensitivity information is readily obtained from PCE

- Main effect sensitivity indices

$$S_i = \frac{Var[\mathbb{E}(g(\boldsymbol{\xi}|\xi_i)]}{Var[g(\boldsymbol{\xi})]} = \frac{\sum_{k\in\mathbb{I}_i} c_k^2 ||\Psi_k||^2}{\sum_{k>0} c_k^2 ||\Psi_k||^2}$$

  $\mathbb{I}_i$ is the set of bases with only ξ_i involved. S_i is the uncertainty contribution that is due to i-th parameter only.

- Joint sensitivity indices

$$S_{ij} = \frac{Var[\mathbb{E}(g(\boldsymbol{\xi}|\xi_i,\xi_j)]}{Var[g(\boldsymbol{\xi})]} - S_i - S_j = \frac{\sum_{k\in\mathbb{I}_{ij}} c_k^2 ||\Psi_k||^2}{\sum_{k>0} c_k^2 ||\Psi_k||^2}$$

  $\mathbb{I}_{ij}$ is the set of bases with only ξ_i and ξ_j involved. S_{ij} is the uncertainty contribution that is due to (i,j) parameter pair.

PC postprocessing: sampling-based approaches

$$g(\xi_1, \ldots, \xi_d) = \sum_{k=0}^{P} c_k \Psi_k(\boldsymbol{\xi})$$

- In some cases, need to resort to Monte-Carlo estimation, e.g.
  - Piecewise-PC with irregular subdomains
  - Output transformations, e.g. build PC for $\log g(\boldsymbol{\xi})$, but inquire sensitivity with respect to $g(\boldsymbol{\xi})$
- A brute-force sampling of $Var[\mathbb{E}(g(\boldsymbol{\xi}|\xi_i)]$ is extremely inefficient.

PC postprocessing: sampling-based approaches

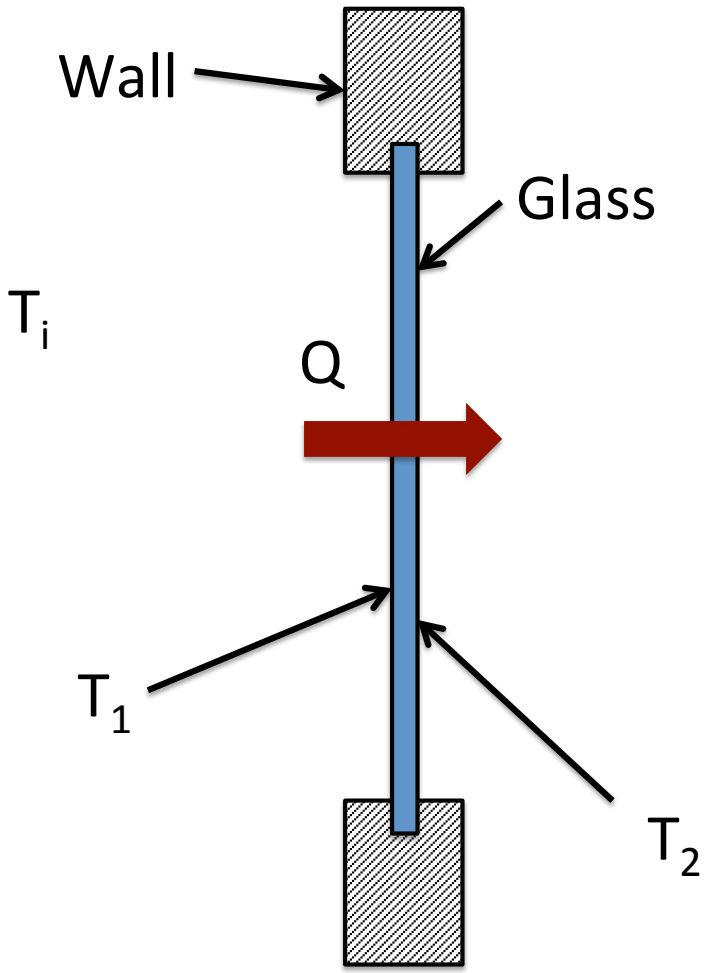
- Tricks are available, given a single set of sampled input [Saltelli, 2002]. E.g., use

$$\mathbb{E}[g(\boldsymbol{\xi}|\xi_i)^2] = \mathbb{E}[g(\boldsymbol{\xi}|\xi_i)g(\boldsymbol{\xi}'|\xi_i)] = \frac{1}{N-1}\sum_{r=1}^{N} g(\boldsymbol{\xi}^{(r)})g(\tilde{\boldsymbol{\xi}}^{(r)}),$$

  where $\tilde{\boldsymbol{\xi}}$ is $\boldsymbol{\xi}'$ with i-th element replaced by ξ_i.
- Similar formulae available for joint sensitivity indices.

- Con: as all Monte-Carlo algorithms, converges slowly.
- Pro: sampling is cheap.

Heat Transfer through a Window



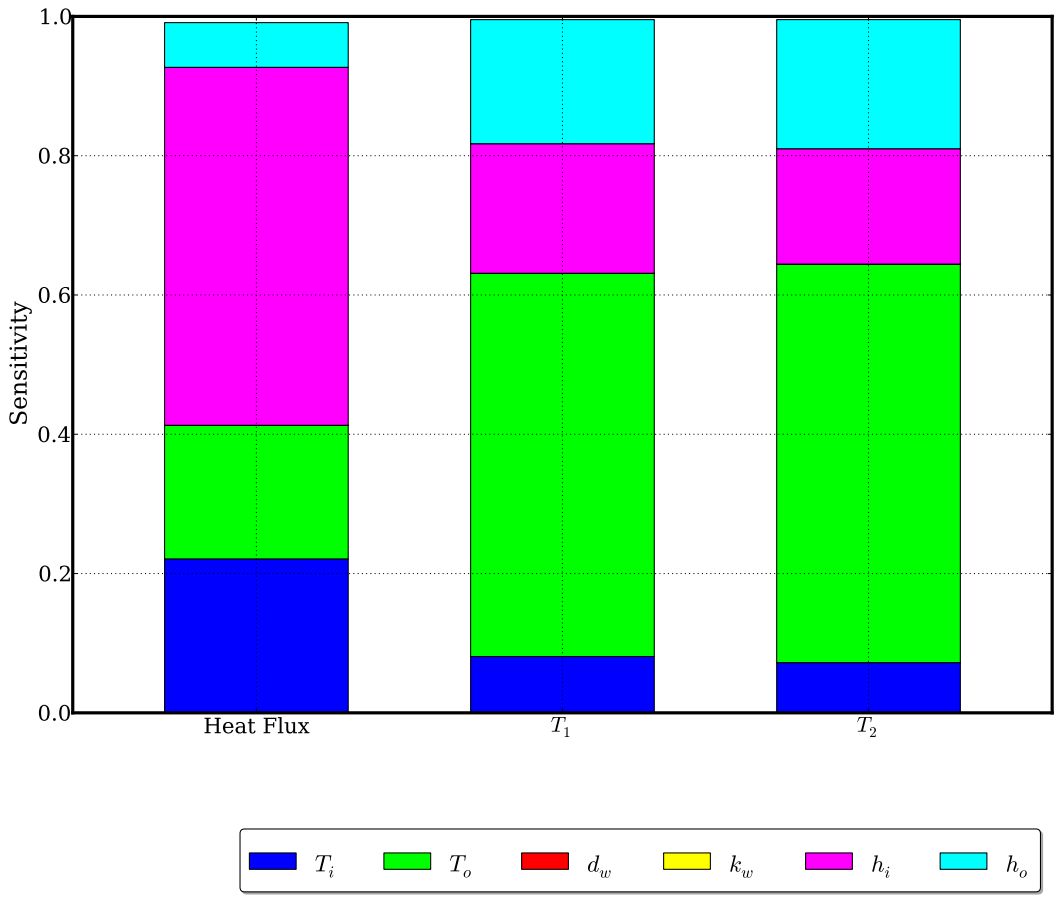
$$h_i(T_i - T_1) = k_w \frac{(T_1 - T_2)}{d_w}$$

$$k_w \frac{(T_1 - T_2)}{d_w} = h_o(T_2 - T_o)$$

6 Uncertain, Gaussian parameters

$$T_i = 293\text{K}, \sigma = 0.5\%$$
$$T_o = 273\text{K}, \sigma = 0.5\%$$
$$d_w = 0.01\text{m}, \sigma = 1\%$$
$$k_w = 1\text{W/mK}, \sigma = 5\%$$
$$h_i = 2\text{W/m}^2\text{K}, \sigma = 15\%$$
$$h_o = 6\text{W/m}^2\text{K}, \sigma = 15\%$$

Outputs are most sensitive to ambient temperatures and convective heat transfer coefficients

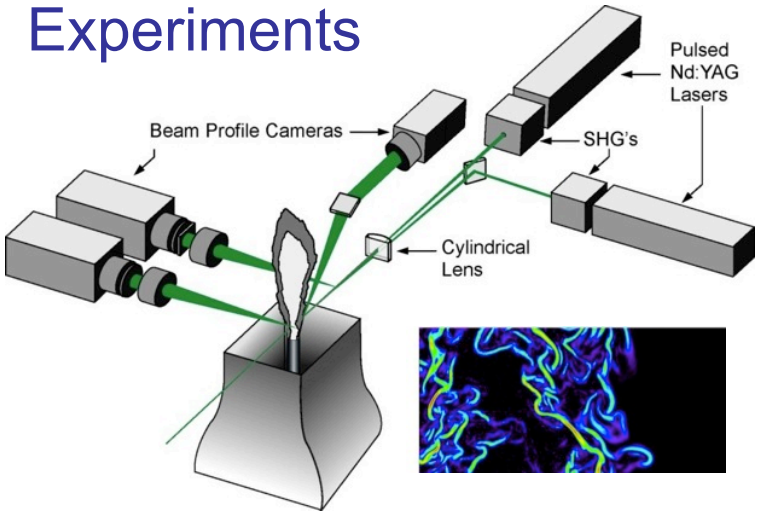


- Main effect sensitivities
  - Sum to 1 only if coupling terms do not matter
- k_w has minimal contribution due to its low uncertainty

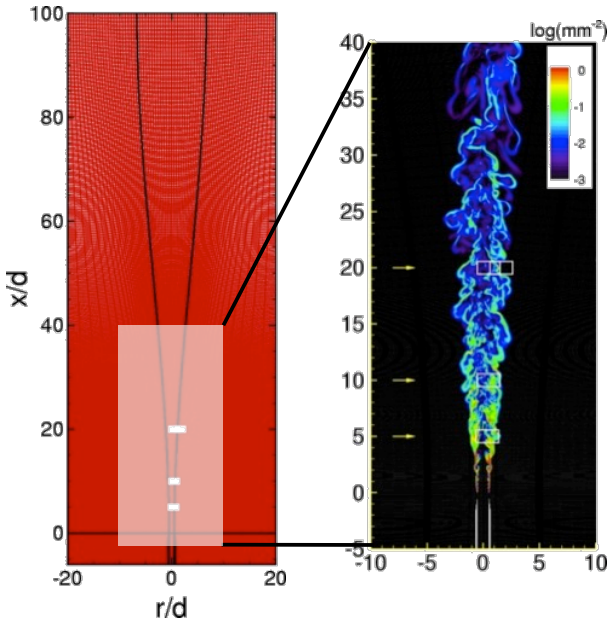
Outline

Experimental data is used to calibrate models

Experiments



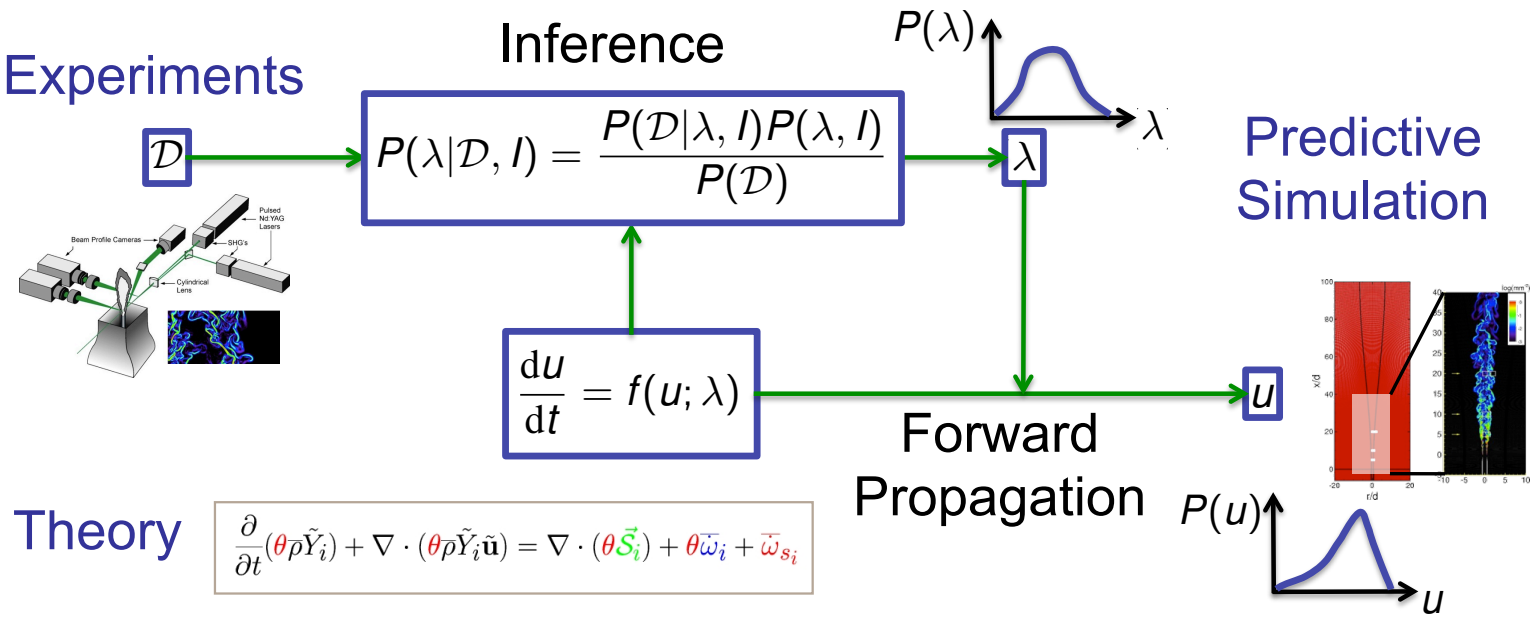
Predictive Simulation



$$\frac{\partial}{\partial t}(\theta\overline{\rho}\tilde{Y}_i) + \nabla\cdot(\theta\overline{\rho}\tilde{Y}_i\tilde{\mathbf{u}}) = \nabla\cdot(\theta\vec{\mathcal{S}}_i) + \theta\overline{\dot{\omega}}_i + \overline{\dot{\omega}}_{s_i}$$

Theory

Bayesian inference



- Bayesian inference can handle various sources of data
- Probabilistic formulation readily accommodates various sources of uncertainty

Bayes' rule updates prior belief with information extracted from data

- Bayes' formula

$$\overbrace{P(\boldsymbol{c}|\mathcal{D})}^{\text{Posterior}} \propto \overbrace{P(\mathcal{D}|\boldsymbol{c})}^{\text{Likelihood}} \overbrace{P(\boldsymbol{c})}^{\text{Prior}}$$

- Update prior distribution/knowledge about parameter $\boldsymbol{c}$ to posterior distribution given data $\mathcal{D}$, using likelihood function $\mathcal{L}(\boldsymbol{c}) \equiv P(\mathcal{D}|\boldsymbol{c})$
- Data $\mathcal{D} = \{d_i\}_{i=1}^{N}$ - measurements of *some* quantities of interest (QoIs)
- Prior distribution $P(\boldsymbol{c})$ is based on expert opinion/previous literature

Bayes' rule updates prior belief with information extracted from data

- Bayes' formula

$$\overbrace{P(\boldsymbol{c}|\mathcal{D})}^{\text{Posterior}} \propto \overbrace{P(\mathcal{D}|\boldsymbol{c})}^{\text{Likelihood}} \overbrace{P(\boldsymbol{c})}^{\text{Prior}}$$

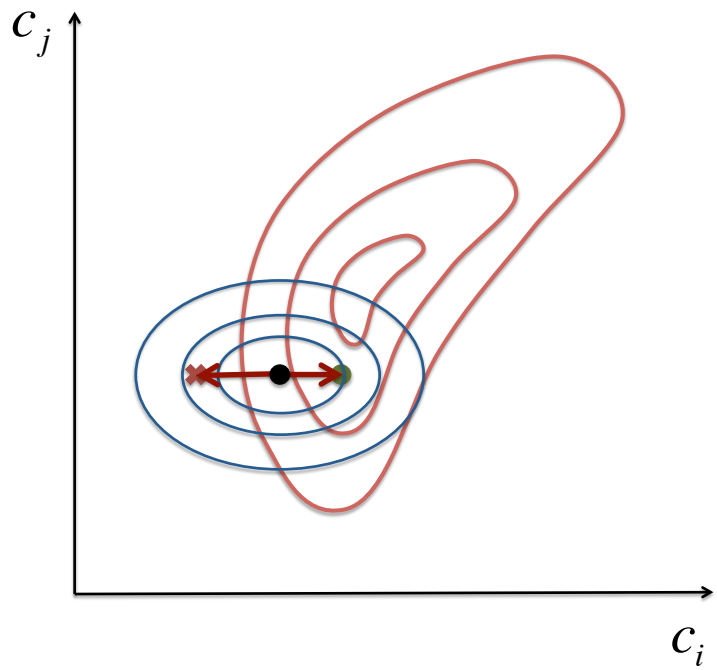
- Likelihood function measures goodness-of-fit and is the key component that connects the model inputs to measured QoIs, e.g.

$$\mathcal{L}(\boldsymbol{c}) = P(\mathcal{D}|\boldsymbol{c}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\sum_{i=1}^{N} \frac{(d_i - f_i(\boldsymbol{c}))^2}{2\sigma^2} \right)$$

- Input parameter $\rightarrow$ output QoI functions $f_i(\cdot)$ could be expensive or not even available
- Posterior distribution generally not analytically tractable: resort to Markov Chain Monte Carlo

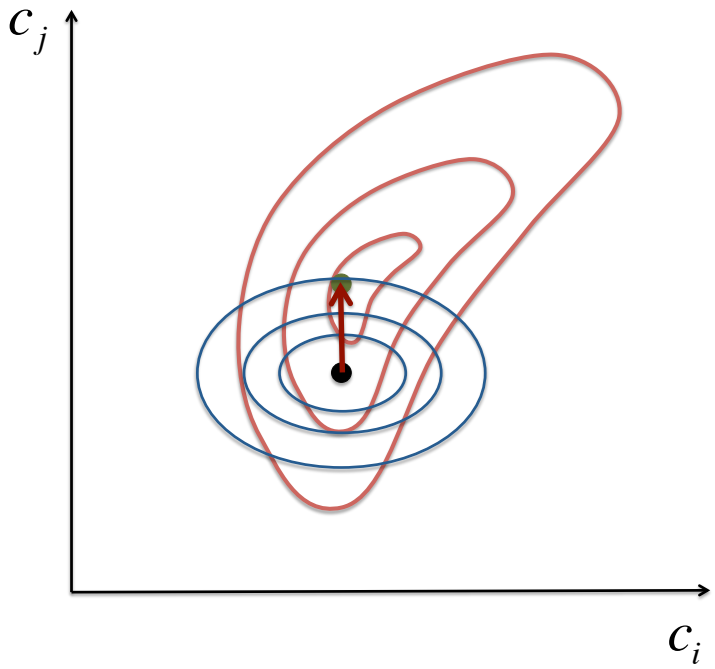
Markov Chain Monte Carlo: Single Site

- Set current chain state $\boldsymbol{c}$ at *initial chain state* $\boldsymbol{c}^{(0)}$,
- Repeat for a predefined number (N_{MCMC}) of times,
  - For $k = 1, \ldots, K$,
    - generate a single-site proposal c'_k from a Gaussian distribution centered at the current chain state value of site c_k with *proposal width* σ_k,
    - compute $\alpha = \min\{1, P(\boldsymbol{c}'|D)/P(\boldsymbol{c}|D)\}$,
    - update the current chain state's k-th element $c_k = c'_k$ with probability α,
  - End
- End



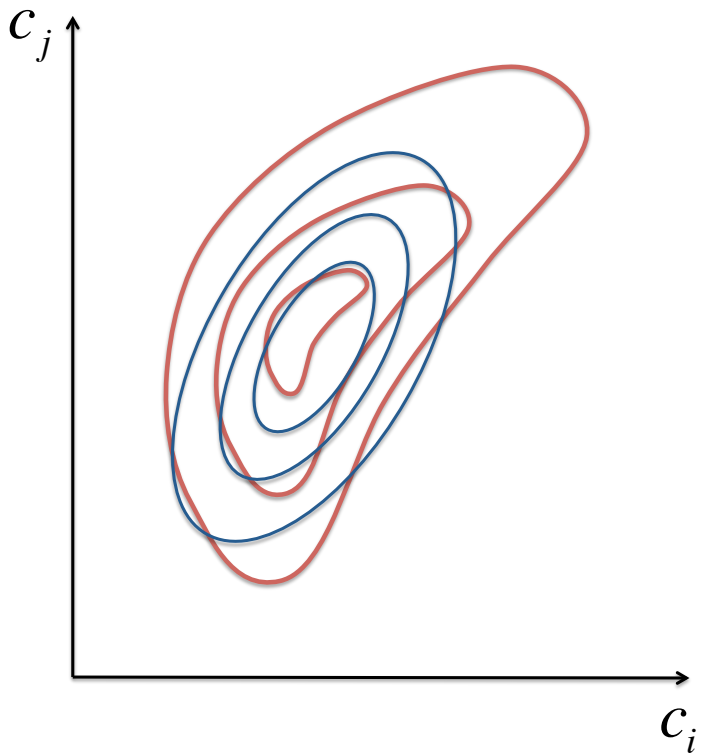
Markov Chain Monte Carlo: Single Site

- Set current chain state $\boldsymbol{c}$ at *initial chain state* $\boldsymbol{c}^{(0)}$,
- Repeat for a predefined number (N_{MCMC}) of times,
  - For $k = 1, \ldots, K$,
    - generate a single-site proposal c'_k from a Gaussian distribution centered at the current chain state value of site c_k with *proposal width* σ_k,
    - compute $\alpha = \min\{1, P(\boldsymbol{c}'|D)/P(\boldsymbol{c}|D)\}$,
    - update the current chain state's k-th element $c_k = c'_k$ with probability α,
  - End
- End



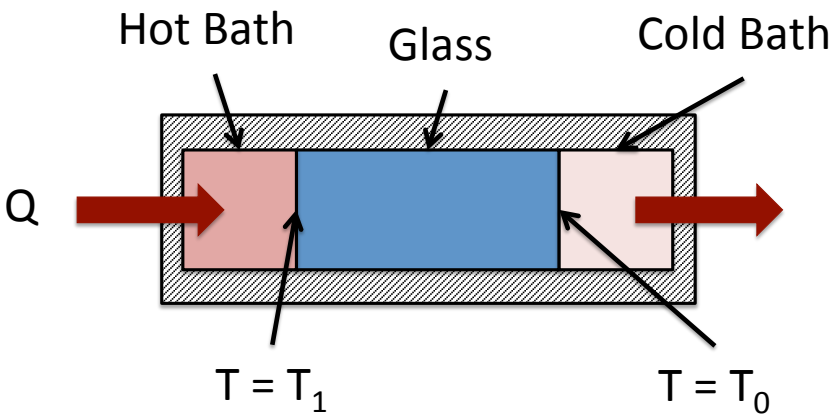
Markov Chain Monte Carlo: Adaptive

- Set current chain state $\boldsymbol{c}$ at *initial chain state* $\boldsymbol{c}^{(0)}$,
- Repeat for a predefined number (N_{MCMC}) of times,
  - generate a proposal $\boldsymbol{c}'$ from a multivariate Gaussian distribution centered at the current chain state value $\boldsymbol{c}$ with *proposal covariance* that is learnt from previous chain states,
  - compute $\alpha = \min\{1, P(\boldsymbol{c}'|D)/P(\boldsymbol{c}|D)\}$,
  - update the current chain state $\boldsymbol{c} = \boldsymbol{c}'$ with probability α,
- End



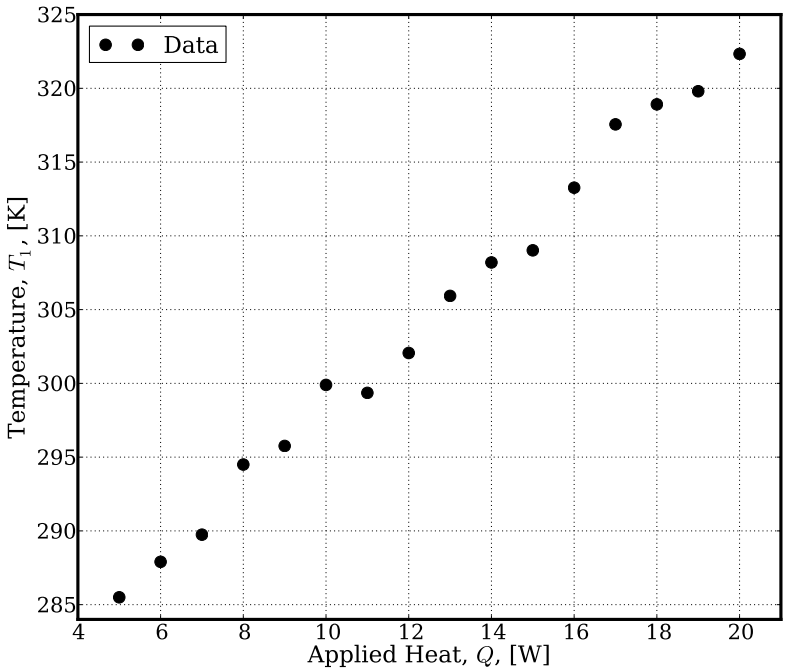
[Haario,2002]

Inference of Conductive Heat Transfer Coefficient



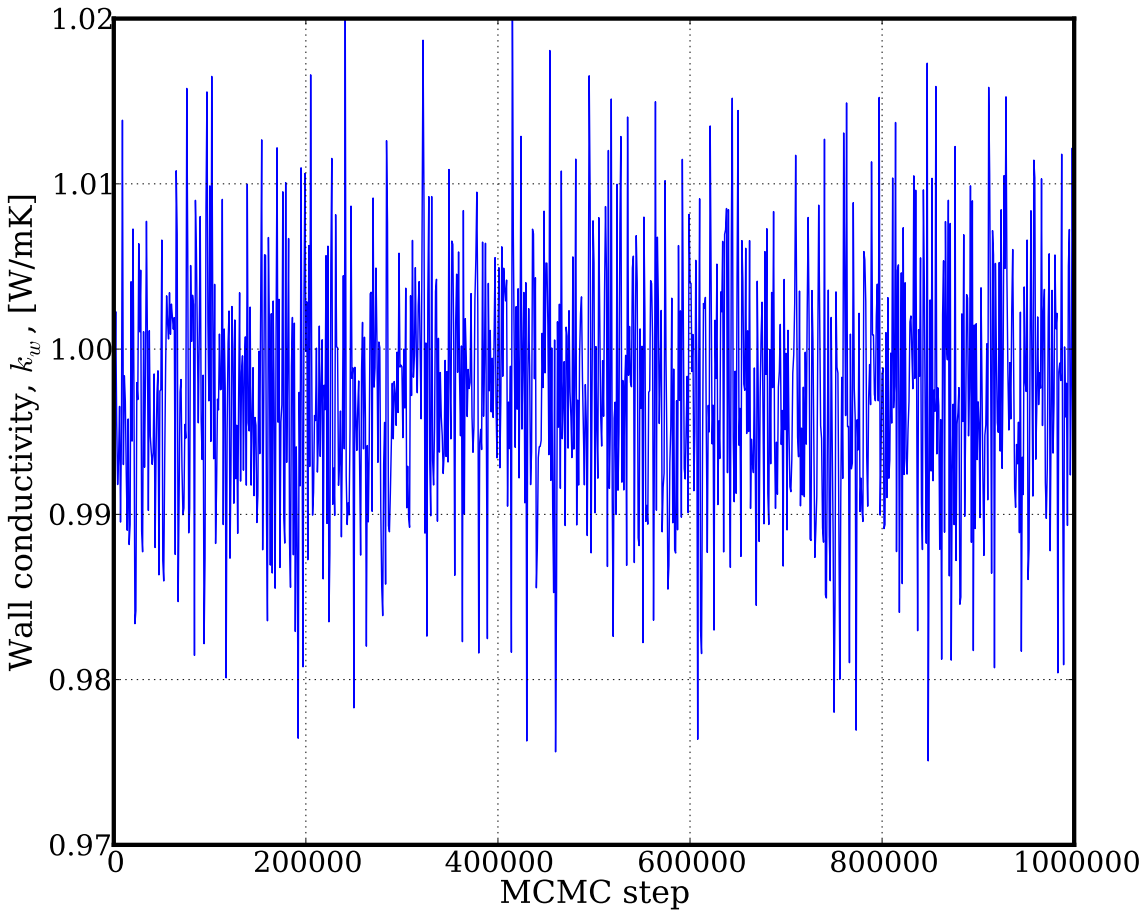
$$Q = k_w A_w \frac{(T_1 - T_0)}{d_w}$$

$$T_0 = 273 \text{ K}, A_w = 0.04 \text{ m}^2$$



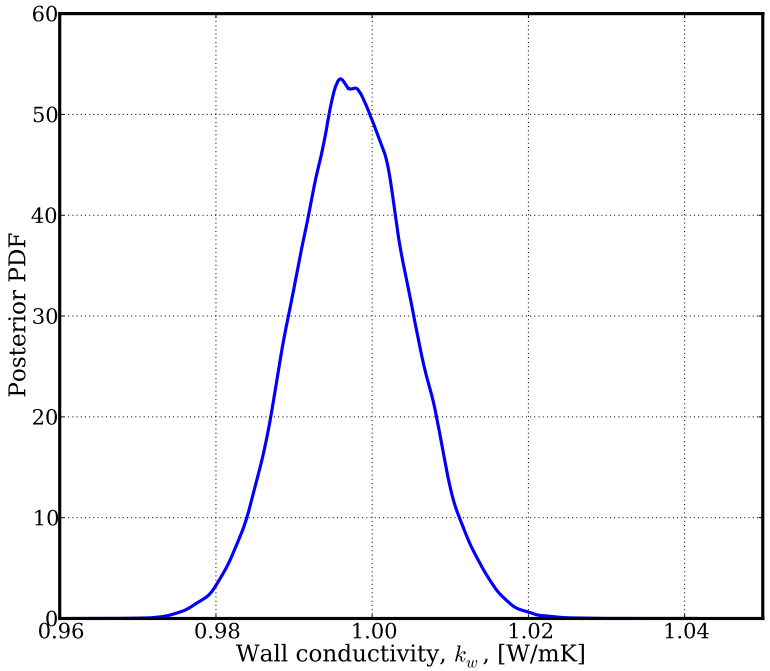
- Piece of glass between two isothermal baths
- Measure T_1 for various applied heat loads Q
- Assume Gaussian measurement noise with $\sigma = 1$ K
- Uniform prior on k_w between 0 and 5 W/mK

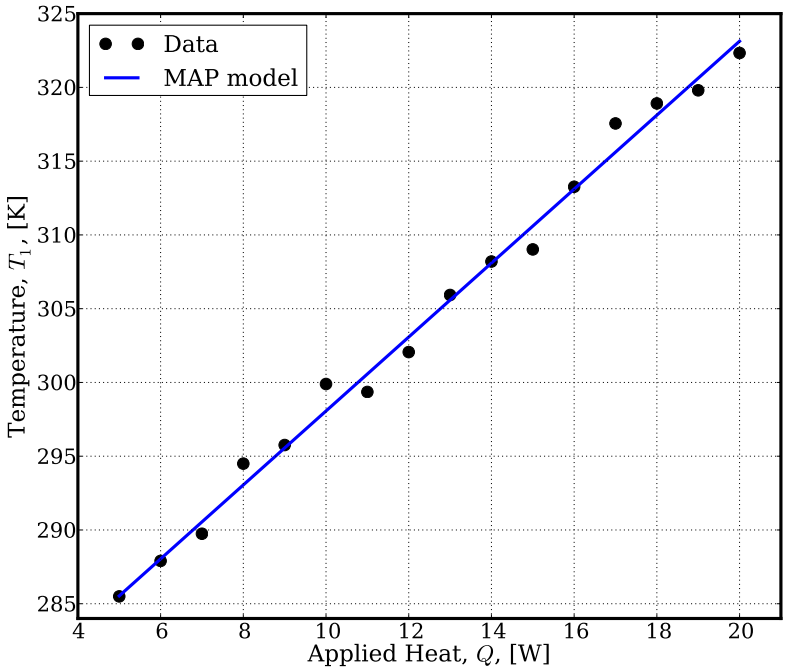
MCMC chain is well mixed



- Used adaptive MCMC with $\gamma = 1$

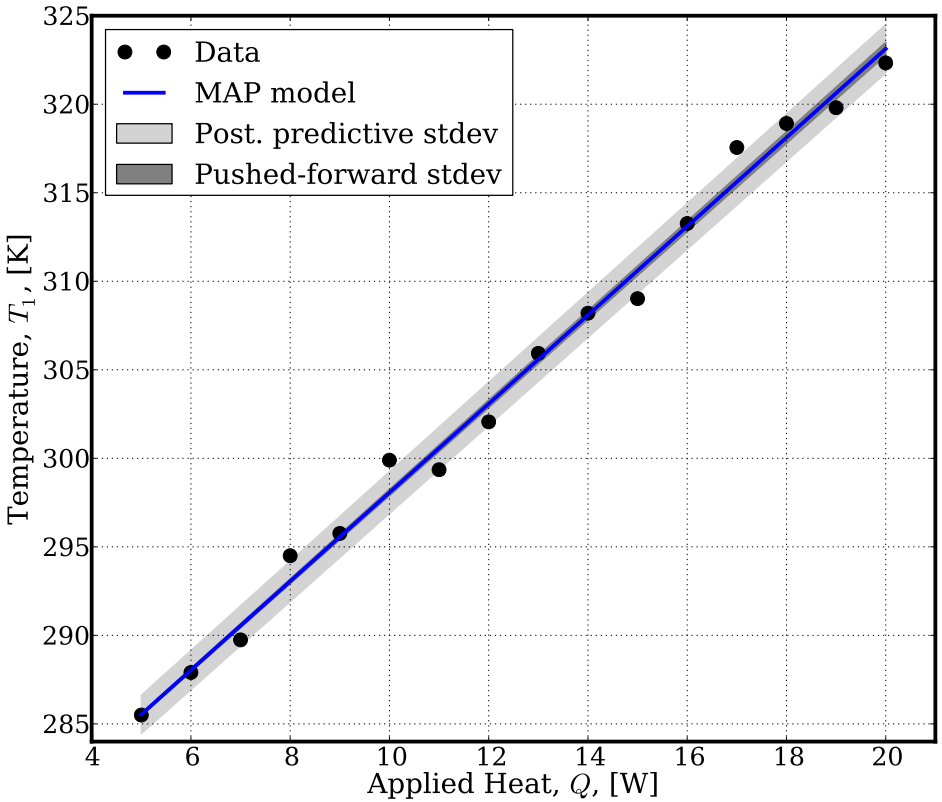
Posterior Distribution and MAP Fit





- Small amount of uncertainty around exact value $k_w = 1$ W/mK
- Maximum A Posteriori (MAP) value for k_w produces good fit to data

Posterior predictive distribution covers original data set



- Pushed forward posterior distribution produces small amount of uncertainty around MAP fit
- Posterior predictive distribution combines posterior uncertainty and measurement noise

Further Reading

- N. Wiener, “Homogeneous Chaos", *American Journal of Mathematics*, 60:4, pp. 897-936, 1938.
- M. Rosenblatt, “Remarks on a Multivariate Transformation", *Ann. Math. Statist.*, 23:3, pp. 470-472, 1952.
- R. Ghanem and P. Spanos, “Stochastic Finite Elements: a Spectral Approach", Springer, 1991.
- O. Le Maître and O. Knio, “Spectral Methods for Uncertainty Quantification with Applications to Computational Fluid Dynamics", Springer, 2010.
- D. Xiu, “Numerical Methods for Stochastic Computations: A Spectral Method Approach", Princeton U. Press, 2010.
- O.G. Ernst, A. Mugler, H.-J. Starkloff, and E. Ullmann, “On the convergence of generalized polynomial chaos expansions,” ESAIM: M2AN, 46:2, pp. 317-339, 2011.
- D. Xiu and G.E. Karniadakis, “The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations", *SIAM J. Sci. Comput.*, 24:2, 2002.
- Le Maître, Ghanem, Knio, and Najm, *J. Comp. Phys.*, 197:28-57 (2004)
- Le Maître, Najm, Ghanem, and Knio, *J. Comp. Phys.*, 197:502-531 (2004)
- B. Debusschere, H. Najm, P. Pébay, O. Knio, R. Ghanem and O. Le Maître, “Numerical Challenges in the Use of Polynomial Chaos Representations for Stochastic Processes", *SIAM J. Sci. Comp.*, 26:2, 2004.
- S. Ji, Y. Xue and L. Carin, “Bayesian Compressive Sensing", *IEEE Trans. Signal Proc.*, 56:6, 2008.
- A. Saltelli and S. Tarantola and F. Campolongo and M. Ratto, “Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models”. John Wiley & Sons, 2004.
- K. Sargsyan, B. Debusschere, H. Najm and O. Le Maître, “Spectral representation and reduced order modeling of the dynamics of stochastic reaction networks via adaptive data partitioning". *SIAM J. Sci. Comp.*, 31:6, 2010.
- K. Sargsyan, B. Debusschere, H. Najm and Y. Marzouk, “Bayesian inference of spectral expansions for predictability assessment in stochastic reaction networks," *J. Comp. Theor. Nanosc.*, 6:10, 2009.
- D.S. Sivia, “Data Analysis: A Bayesian Tutorial,” Oxford Science, 1996.

Extra Material on Forward Propagation

- Intrusive operations on PCEs
- Simple ODE example
- Intrusive UQ of incompressible flow
- Uncertainty Quantification Toolkit (UQTk) implementation of intrusive and non-intrusive UQ in surface reaction example (3 ODE system)

Intrusive Galerkin projection reformulates original equations

- Assume $v = f(u; a, \lambda)$, with
  - a deterministic parameter(s)
  - λ uncertain parameter(s)
  - u, v variables of interest (deterministic or uncertain)
- Represent uncertain variables with PCEs

$$\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k(\xi), \qquad v = \sum_{k=0}^{P} v_k \Psi_k(\xi)$$

- Apply Galerkin projection to get PC coefficients of v

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle f(u; a, \lambda) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

- Results in larger, but deterministic set of equations

Projection of linear operations: $v = \gamma + \lambda$

- Assume $v = \gamma + \lambda$, with $\gamma = \sum_{k=0}^{P} \gamma_k \Psi_k$, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Galerkin projection

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle (\gamma + \lambda) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

$$\begin{aligned} \langle (\gamma + \lambda) \Psi_k \rangle &= \left\langle \sum_{j=0}^{P} \gamma_j \Psi_j \Psi_k \right\rangle + \left\langle \sum_{j=0}^{P} \lambda_j \Psi_j \Psi_k \right\rangle \\ &= \sum_{j=0}^{P} \gamma_j \langle \Psi_j \Psi_k \rangle + \sum_{j=0}^{P} \lambda_j \langle \Psi_j \Psi_k \rangle \\ &= \gamma_k \langle \Psi_k^2 \rangle + \lambda_k \langle \Psi_k^2 \rangle \end{aligned}$$

- $\Rightarrow \quad v_k = \gamma_k + \lambda_k$

Projection of linear operations: $v = a + \lambda$

- Special case of $v = \gamma + \lambda$, with
  - $\gamma = a\,\Psi_0 = a$
  - $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$
  - $v = \sum_{k=0}^{P} v_k \Psi_k$
- Resulting in
  - $v_0 = a + \lambda_0, \quad v_{k>0} = \lambda_k$
- Deterministic parameter is a special case of a PCE with 0^{th} term only
- Adding a deterministic value to a PCE just shifts its mean

Projection of linear operations: $v = a + \lambda$

- Assume $v = a + \lambda$, with
  a deterministic, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Galerkin projection

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle (a+\lambda)\Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

$$\begin{aligned} \langle (a+\lambda)\Psi_k \rangle &= \langle a\Psi_k \rangle + \left\langle \sum_{j=0}^{P} \lambda_j \Psi_j \Psi_k \right\rangle \\ &= a \langle \Psi_k \rangle + \sum_{j=0}^{P} \lambda_j \langle \Psi_j \Psi_k \rangle \\ &= a\, \delta_{0k} + \lambda_k \langle \Psi_k^2 \rangle \end{aligned}$$

- $\Rightarrow \quad v_0 = a + \lambda_0, \quad v_{k>0} = \lambda_k$

Projection of linear operations: $v = a\,\lambda$

- Assume $v = a\,\lambda$, with
  a deterministic, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Galerkin projection

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle (a\,\lambda) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

$$\begin{aligned} \langle (a\,\lambda) \Psi_k \rangle &= \left\langle a \sum_{j=0}^{P} \lambda_j \Psi_j \Psi_k \right\rangle \\ &= \sum_{j=0}^{P} a\,\lambda_j \langle \Psi_j \Psi_k \rangle \\ &= a\,\lambda_k \langle \Psi_k^2 \rangle \end{aligned}$$

- $\Rightarrow \quad v_k = a\,\lambda_k$

Projection of product: $v = \gamma\ \lambda$

- Assume $v = \gamma\ \lambda$, with $\gamma = \sum_{k=0}^{P} \gamma_k \Psi_k$, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Galerkin projection

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle (\gamma\ \lambda) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

$$\begin{aligned}
\langle (\gamma\ \lambda) \Psi_k \rangle &= \left\langle \left(\sum_{i=0}^{P} \gamma_i \Psi_i \sum_{j=0}^{P} \lambda_j \Psi_j \right) \Psi_k \right\rangle \\
&= \left\langle \sum_{i=0}^{P} \sum_{j=0}^{P} \gamma_i \lambda_j \Psi_i \Psi_j \Psi_k \right\rangle \\
&= \sum_{i=0}^{P} \sum_{j=0}^{P} \gamma_i \lambda_j \langle \Psi_i \Psi_j \Psi_k \rangle
\end{aligned}$$

Projection of product: $v = \gamma\,\lambda$

$$\Rightarrow v_k = \sum_{i=0}^{P}\sum_{j=0}^{P}\gamma_i\lambda_j\frac{\langle\Psi_i\Psi_j\Psi_k\rangle}{\langle\Psi_k^2\rangle} = \sum_{i=0}^{P}\sum_{j=0}^{P}\gamma_i\lambda_j C_{ijk}$$

- $C_{ijk} = \frac{\langle\Psi_i\Psi_j\Psi_k\rangle}{\langle\Psi_k^2\rangle}$
- The C_{ijk} tensor can be computed up front for any given PC order and dimension and stored for use whenever two RVs are multiplied
- This tensor is sparse, *i.e.* many of its elements are zero

1D 4^{th}-Order C_{ijk} Example : Hermite polynomials

$\langle \Psi_i \Psi_j \Psi_k \rangle$	value
$\langle \Psi_0 \Psi_0 \Psi_0 \rangle$	1
$\langle \Psi_0 \Psi_1 \Psi_1 \rangle$	1
$\langle \Psi_0 \Psi_2 \Psi_2 \rangle$	2
$\langle \Psi_0 \Psi_3 \Psi_3 \rangle$	6
$\langle \Psi_0 \Psi_4 \Psi_4 \rangle$	24
$\langle \Psi_1 \Psi_1 \Psi_2 \rangle$	2
$\langle \Psi_1 \Psi_2 \Psi_3 \rangle$	6
$\langle \Psi_1 \Psi_3 \Psi_4 \rangle$	24
$\langle \Psi_2 \Psi_2 \Psi_2 \rangle$	8
$\langle \Psi_2 \Psi_2 \Psi_4 \rangle$	24
$\langle \Psi_2 \Psi_3 \Psi_3 \rangle$	36
$\langle \Psi_2 \Psi_4 \Psi_4 \rangle$	192
$\langle \Psi_3 \Psi_3 \Psi_4 \rangle$	216
$\langle \Psi_4 \Psi_4 \Psi_4 \rangle$	1728

k	$\langle \Psi_k^2 \rangle$
0	1
1	1
2	2
3	6
4	24

- $C_{ijk} = \langle \Psi_i \Psi_j \Psi_k \rangle \, / \, \langle \Psi_k^2 \rangle$
- and,

$$\begin{aligned} \langle \Psi_i \Psi_j \Psi_k \rangle &= \langle \Psi_i \Psi_k \Psi_j \rangle = \\ \langle \Psi_j \Psi_i \Psi_k \rangle &= \langle \Psi_j \Psi_k \Psi_i \rangle = \\ \langle \Psi_k \Psi_i \Psi_j \rangle &= \langle \Psi_k \Psi_j \Psi_i \rangle \end{aligned}$$

- with other not-reported $\langle \Psi_i \Psi_j \Psi_k \rangle$ zero

1D 4^{th}-Order C_{ijk} Example : Legendre polynomials

$\langle \Psi_i \Psi_j \Psi_k \rangle$	value
$\langle \Psi_0 \Psi_0 \Psi_0 \rangle$	1
$\langle \Psi_0 \Psi_1 \Psi_1 \rangle$	1/3
$\langle \Psi_0 \Psi_2 \Psi_2 \rangle$	1/5
$\langle \Psi_0 \Psi_3 \Psi_3 \rangle$	1/7
$\langle \Psi_0 \Psi_4 \Psi_4 \rangle$	1/9
$\langle \Psi_1 \Psi_1 \Psi_2 \rangle$	2/15
$\langle \Psi_1 \Psi_2 \Psi_3 \rangle$	3/35
$\langle \Psi_1 \Psi_3 \Psi_4 \rangle$	4/63
$\langle \Psi_2 \Psi_2 \Psi_2 \rangle$	2/35
$\langle \Psi_2 \Psi_2 \Psi_4 \rangle$	2/35
$\langle \Psi_2 \Psi_3 \Psi_3 \rangle$	4/105
$\langle \Psi_2 \Psi_4 \Psi_4 \rangle$	$\approx$0.029
$\langle \Psi_3 \Psi_3 \Psi_4 \rangle$	$\approx$0.026
$\langle \Psi_4 \Psi_4 \Psi_4 \rangle$	$\approx$0.018

k	$\langle \Psi_k^2 \rangle$
0	1
1	1/3
2	1/5
3	1/7
4	1/9

- $C_{ijk} = \langle \Psi_i \Psi_j \Psi_k \rangle \, / \, \langle \Psi_k^2 \rangle$
- and,

$$\begin{aligned} \langle \Psi_i \Psi_j \Psi_k \rangle &= \langle \Psi_i \Psi_k \Psi_j \rangle = \\ \langle \Psi_j \Psi_i \Psi_k \rangle &= \langle \Psi_j \Psi_k \Psi_i \rangle = \\ \langle \Psi_k \Psi_i \Psi_j \rangle &= \langle \Psi_k \Psi_j \Psi_i \rangle \end{aligned}$$

- with other not-reported $\langle \Psi_i \Psi_j \Psi_k \rangle$ zero

Projection of triple product: $v = \gamma\ \lambda\ u$

- Assume $v = \gamma\ \lambda\ u$, with $\gamma = \sum_{k=0}^{P} \gamma_k \Psi_k$, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, $u = \sum_{k=0}^{P} u_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Galerkin projection

$$v_k = \frac{\langle v \Psi_k \rangle}{\langle \Psi_k^2 \rangle} = \frac{\langle (\gamma\ \lambda\ u) \Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \ldots, P$$

$$\begin{aligned}
\langle (\gamma\ \lambda\ u) \Psi_k \rangle &= \left\langle \left(\sum_{l=0}^{P} \gamma_l \Psi_l \sum_{i=0}^{P} \lambda_i \Psi_i \sum_{j=0}^{P} u_j \Psi_j \right) \Psi_k \right\rangle \\
&= \left\langle \sum_{l=0}^{P} \sum_{i=0}^{P} \sum_{j=0}^{P} \gamma_l \lambda_i u_j \Psi_l \Psi_i \Psi_j \Psi_k \right\rangle \\
&= \sum_{l=0}^{P} \sum_{i=0}^{P} \sum_{j=0}^{P} \gamma_l \lambda_i u_j \langle \Psi_l \Psi_i \Psi_j \Psi_k \rangle
\end{aligned}$$

Projection of triple product: $v = \gamma\ \lambda\ u$

$$\Rightarrow v_k = \sum_{l=0}^{P}\sum_{i=0}^{P}\sum_{j=0}^{P}\gamma_l\lambda_i u_j \frac{\langle\Psi_l\Psi_i\Psi_j\Psi_k\rangle}{\langle\Psi_k^2\rangle} = \sum_{l=0}^{P}\sum_{i=0}^{P}\sum_{j=0}^{P}\gamma_l\lambda_i u_j D_{lijk}$$

- $D_{lijk} = \frac{\langle\Psi_l\Psi_i\Psi_j\Psi_k\rangle}{\langle\Psi_k^2\rangle}$
- The D_{lijk} tensor can also be computed up front for any given PC order and dimension and stored for use whenever three RVs are multiplied
- While this tensor is sparse, it can be expensive to compute and store
- This fully spectral formulation is even less practical for higher order products

Pseudo-spectral triple product: $v = \gamma\ \lambda\ u$

- Assume $v = \gamma\ \lambda$, with $\gamma = \sum_{k=0}^{P} \gamma_k \Psi_k$, $\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k$, $u = \sum_{k=0}^{P} u_k \Psi_k$, and $v = \sum_{k=0}^{P} v_k \Psi_k$
- Perform product in sub-steps

$$\begin{aligned} v = \gamma\ \lambda\ u &= ((\gamma\ \lambda)u) \\ &= \tilde{v}u \end{aligned}$$

- Each sub-step can be performed with regular binary product formula

$$\begin{aligned} \tilde{v} &= \gamma\ \lambda \\ v &= \tilde{v}u \end{aligned}$$

Pseudo-spectral product readily generalizes to higher order products

- Decompose higher order products or powers in sequences of binary products
  - Equivalent to successive multiplication (accounting for terms up to order $2p$) and projecting back to order p
- Efficient and convenient
- Can lead to aliasing errors due to loss of information in higher order modes
- See also [Debusschere *et al.*, *S*IAM J. Sci. Comp, 2004]

Intrusive propagation through non-polynomial functions

Addition, subtraction, and product allow (pseudo-)spectral evaluation of all polynomial functions

How to propagate PC expansions ($\{u_k\} \Rightarrow \{v_k\}$) through transcendental functions

$$v = \frac{1}{u}, \quad v = \ln u, \quad \text{or} \quad v = e^u$$

- Use local polynomial approximations, *e.g.* Taylor series
- Rework operation into a system of equations
- Integration approach
- Borchardt-Gauss Algorithm: Arithmetic-Geometric Mean (AGM) series

[Debusschere *et al.*, SISC 2004; McKale, Texas Tech, M.S. Thesis, 2011]

Taylor series allows computation of many transcendental functions

- Provides local polynomial approximation
  - *E.g.* $e^u = 1 + \frac{u}{1!} + \frac{u^2}{2!} + \frac{u^3}{3!} + \ldots$
- Expanding the series for $f(u)$ around u_0 speeds up convergence
- Works well in most cases, especially for small uncertainties
- Not very robust for larger uncertainties
  - Series can take too long to converge
  - High-order PC multiplications lead to aliasing
  - Instabilities if Taylor series range of convergence exceeded; *e.g.* $\log(u)$ expanded around u_0 only converges for $|u - u_0| < 1$

Inversion and division can be computed through a system of equations

- Assume three uncertain variables u, v, and w

$$w = \frac{u}{v} \Rightarrow \qquad v\ w = u$$

- Mode k of the stochastic product

$$\sum_{i=0}^{P}\sum_{j=0}^{P} C_{ijk}\, v_i\ w_j = u_k$$

- System of $P+1$ linear equations in w_j with known u_k and v_i,

$$V_{kj} = \sum_{i=0}^{P} C_{ijk}\, v_i \quad \Rightarrow \quad \mathbf{Vw} = \mathbf{u}$$

- More robust than Taylor series expansion for $1/u$

Integration approach for non-polynomial functions

- Consider the ODE $\frac{\mathrm{d}v}{\mathrm{d}u} = v$, with solution $v = e^u$ $\Rightarrow f(u) = e^u$ can be obtained from

$$\mathrm{d}v = v\,\mathrm{d}u \qquad \Rightarrow \qquad e^u - e^{u_o} = \int_{u_o}^{u} v\,\mathrm{d}u$$

- Similarly for e^{-u^2}, and $\ln(u)$

$$e^{-u^2} - e^{-u_o^2} = \int_{u_o}^{u} -2uv\,\mathrm{d}u, \qquad \ln(u) - \ln(u_o) = \int_{u_o}^{u} \frac{\mathrm{d}u}{u}$$

- Accurate if PC order is high enough to properly capture the random variable $v = f(u)$

More general formulation of integration approach for irrational functions

- To evaluate $v(u)$, $u = \sum_{k=0}^{P} u_k \Psi_k$, $v = \sum_{k=0}^{P} v_k \Psi_k$,
  - Use a deterministic IC u_a such that $v(u_a)$ is known
  - Express $\dot{v} = \mathrm{d}v/\mathrm{d}u = f(v, u)$;
    ... **require**: f is a rational function
    ... ensures that $(\dot{v})_k$ are found from v_k and u_k coeffs
  - Evaluate the integral:

$$v_k(u_b) - v_k(u_a) = \sum_{j=0}^{P} \int_{(u_a)_j}^{(u_b)_j} \sum_{i=0}^{P} C_{ijk} (\dot{v})_i \, \mathrm{d}u_j$$

- ok for e^u, e^{u^2}, and $\ln(u)$, with $\dot{v} = v$, $2uv$, and $1/u$ resp.
  - but not for $e^{\sin u}$, with $\dot{v} = v \cos u$
- More robust than Taylor series, but CPU-intensive

Overloading of operations

- Construction allows for a general representation using pseudo-spectral (PS) overloaded operations.
  - *E.g.* multiplication operation '$*$'

$$w = \lambda * u * u * v$$

    - Each deterministic function multiplication is transformed into a corresponding PC product
- Potential meta-code: take a general deterministic code function $F(u)$, produce a pseudo-spectral stochastic function $\tilde{F}(\tilde{u})$
  - Possibility of transforming legacy deterministic code into corresponding pseudo-spectral stochastic code.
  - UQToolkit: contains library of utilities for operations on random variables represented with PCEs

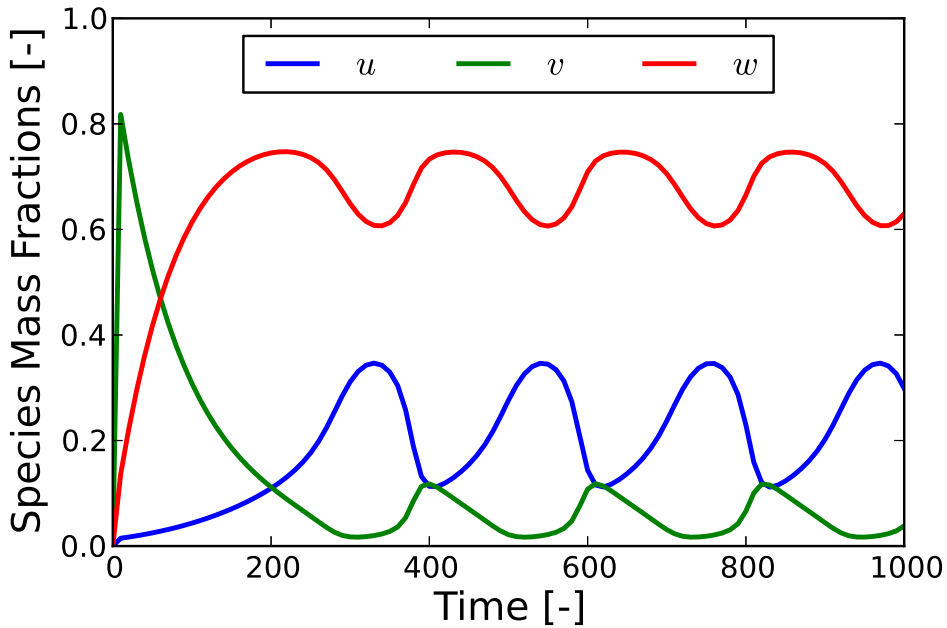
Surface Reaction Model

3 ODEs for a monomer (u), dimer (v), and inert species (w) adsorbing onto a surface out of gas phase.

$$\begin{aligned}
\frac{\mathrm{d}u}{\mathrm{d}t} &= az - cu - 4duv \\
\frac{\mathrm{d}v}{\mathrm{d}t} &= 2bz^2 - 4duv \\
\frac{\mathrm{d}w}{\mathrm{d}t} &= ez - fw \\
z &= 1 - u - v - w
\end{aligned}$$

$$u(0) = v(0) = w(0) = 0.0$$

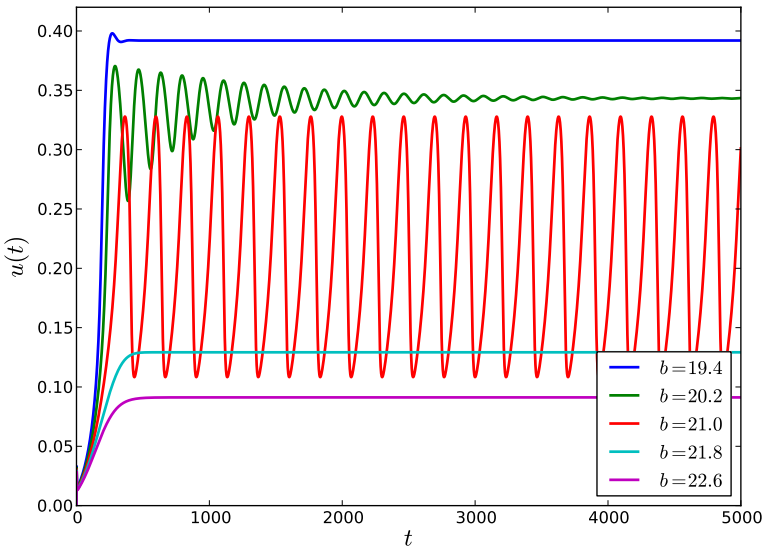
$$\begin{aligned}
a &= 1.6 \quad b = 20.75 \quad c = 0.04 \\
d &= 1.0 \quad e = 0.36 \quad f = 0.016
\end{aligned}$$

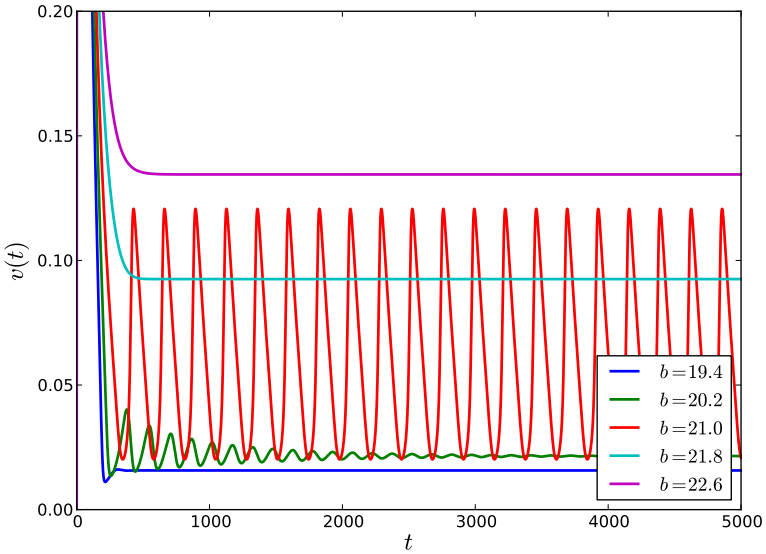


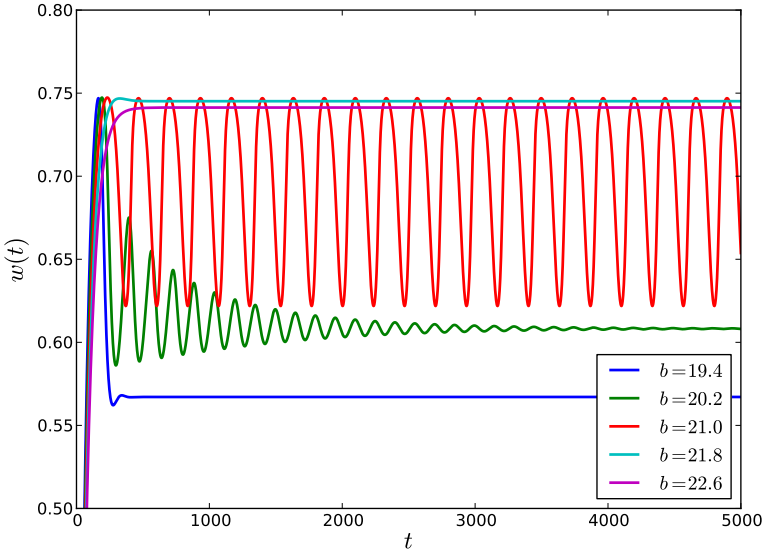
Oscillatory behavior for $b \in [20.2, 21.2]$

[Vigil *et al.*, Phys. Rev. E., 1996; Makeev *et al.*, J. Chem. Phys., 2002]

Surface reaction model shows wide range of dynamics







$$a = 1.6 \quad b = [19.4 \ldots 22.6]$$
$$c = 0.04 \quad d = 1.0$$
$$e = 0.36 \quad f = 0.016$$

Surface Reaction Model: Intrusive Spectral Propagation (ISP) of Uncertainty

- Assume PCE for uncertain parameter b and for the output variables, u, v, w
- Substitute PCEs into the governing equations
- Project the governing equations onto the PC basis functions
  - Multiply with Ψ_k and take the expectation
- Apply pseudo-spectral approximations where necessary

Surface Reaction Model: Specify PCEs for inputs and outputs

Represent uncertain inputs with PCEs with known coefficients:

$$b = \sum_{i=0}^{P} b_i \Psi_i(\xi)$$

Represent all uncertain variables with PCEs with unknown coefficients:

$$u(t) = \sum_{i=0}^{P} u_i(t)\Psi_i(\xi) \qquad v(t) = \sum_{i=0}^{P} v_i(t)\Psi_i(\xi)$$

$$w(t) = \sum_{i=0}^{P} w_i(t)\Psi_i(\xi) \qquad z(t) = \sum_{i=0}^{P} z_i(t)\Psi_i(\xi)$$

Surface Reaction Model: Substitute PCEs into governing equations and project onto basis functions

$$
\begin{aligned}
\frac{\mathrm{d}u}{\mathrm{d}t} &= az - cu - 4duv \\
\frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} u_i\Psi_i &= a\sum_{i=0}^{P} z_i\Psi_i - c\sum_{i=0}^{P} u_i\Psi_i - 4d\sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j \\
\left\langle \Psi_k \frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} u_i\Psi_i \right\rangle &= \left\langle a\Psi_k \sum_{i=0}^{P} z_i\Psi_i \right\rangle - \left\langle c\Psi_k \sum_{i=0}^{P} u_i\Psi_i \right\rangle \\
&- \left\langle 4d\Psi_k \sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j \right\rangle
\end{aligned}
$$

Surface Reaction Model: Reorganize terms

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k \left\langle \Psi_k^2 \right\rangle = a z_k \left\langle \Psi_k^2 \right\rangle - c u_k \left\langle \Psi_k^2 \right\rangle - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \left\langle \Psi_i \Psi_j \Psi_k \right\rangle$$

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k = a z_k - c u_k - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k = a z_k - c u_k - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j C_{ijk}$$

- Triple products $C_{ijk} = \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle}$ can be pre-computed and stored for repeated use

Surface Reaction Model: Substitute PCEs into governing equations and project onto basis functions

$$\frac{\mathrm{d}v}{\mathrm{d}t} = 2bz^2 - 4duv$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} v_i\Psi_i = 2\sum_{h=0}^{P} b_h\Psi_h \sum_{i=0}^{P} z_i\Psi_i \sum_{j=0}^{P} z_j\Psi_j - 4d\sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j$$

$$\left\langle \Psi_k \frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=0}^{P} v_i\Psi_i \right\rangle = \left\langle 2\Psi_k \sum_{h=0}^{P} b_h\Psi_h \sum_{i=0}^{P} z_i\Psi_i \sum_{j=0}^{P} z_j\Psi_j \right\rangle$$

$$- \left\langle 4d\Psi_k \sum_{i=0}^{P} u_i\Psi_i \sum_{j=0}^{P} v_j\Psi_j \right\rangle$$

Surface Reaction Model: Reorganize terms

$$\frac{\mathrm{d}}{\mathrm{d}t} v_k \left\langle \Psi_k^2 \right\rangle = 2 \sum_{h=0}^{P} \sum_{i=0}^{P} \sum_{j=0}^{P} b_h z_i z_j \left\langle \Psi_h \Psi_i \Psi_j \Psi_k \right\rangle - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \left\langle \Psi_i \Psi_j \Psi_k \right\rangle$$

$$\frac{\mathrm{d}}{\mathrm{d}t} v_k = 2 \sum_{h=0}^{P} \sum_{i=0}^{P} \sum_{j=0}^{P} b_h z_i z_j \frac{\left\langle \Psi_h \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle} - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} v_k = 2 \sum_{h=0}^{P} \sum_{i=0}^{P} \sum_{j=0}^{P} b_h z_i z_j D_{hijk} - 4d \sum_{i=0}^{P} \sum_{j=0}^{P} u_i v_j C_{ijk}$$

- Pre-computing and storing the quad product D_{hijk} becomes cumbersome
- Use pseudo-spectral approach instead

Surface Reaction Model: Pseudo-Spectral approach for products

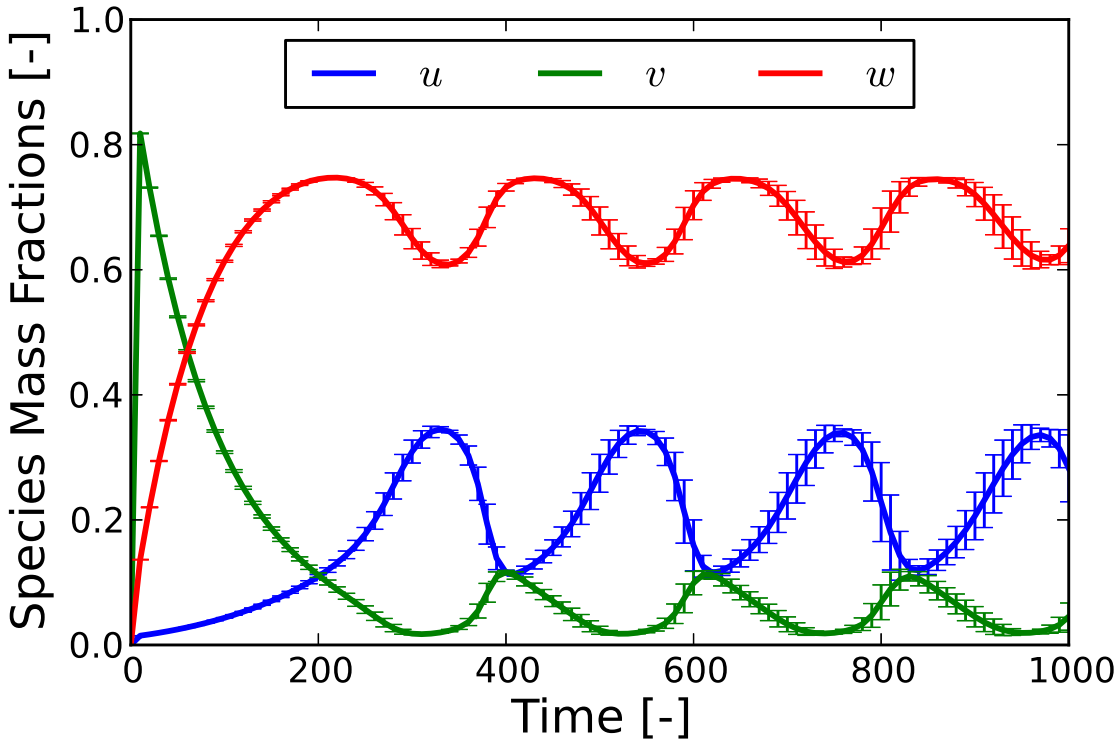
- Introduce auxiliary variable $g = z^2$

$$\begin{aligned} g &= z^2 \\ f = 2bz^2 &= 2bg \end{aligned}$$

$$\begin{aligned} g_k &= \sum_{i=0}^{P}\sum_{j=0}^{P} z_i z_j C_{ijk} \\ f_k &= 2\sum_{i=0}^{P}\sum_{j=0}^{P} b_i g_j C_{ijk} \end{aligned}$$

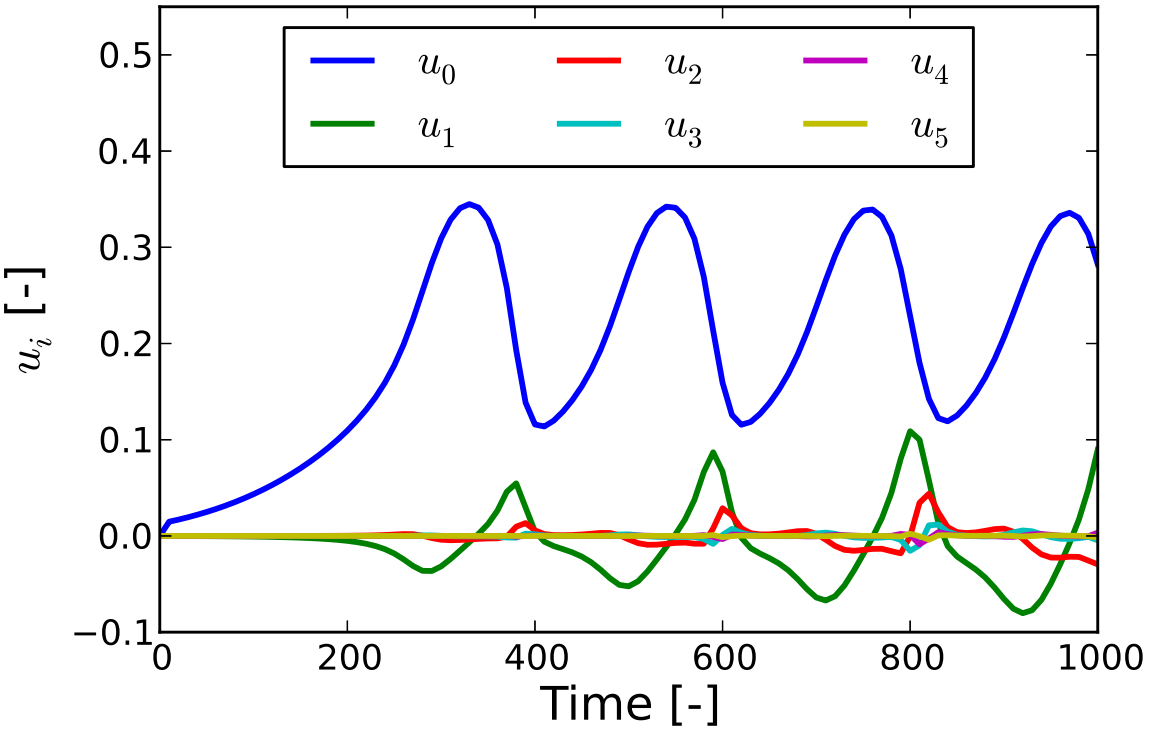
- Limits the complexity of computing product terms
  - Higher products can be computed by repeated use of the same binary product rule
- Does introduce errors if order of PCE is not large enough

Surface Reaction Model: ISP results



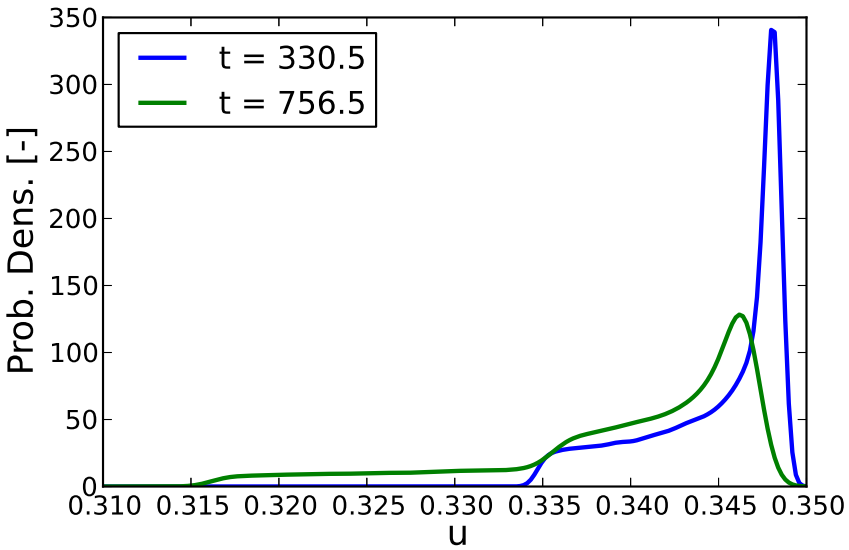
- Assume 0.5% uncertainty in b around nominal value
- Legendre-Uniform intrusive PC
- Mean and standard deviation for u, v, and w
- Uncertainty grows in time

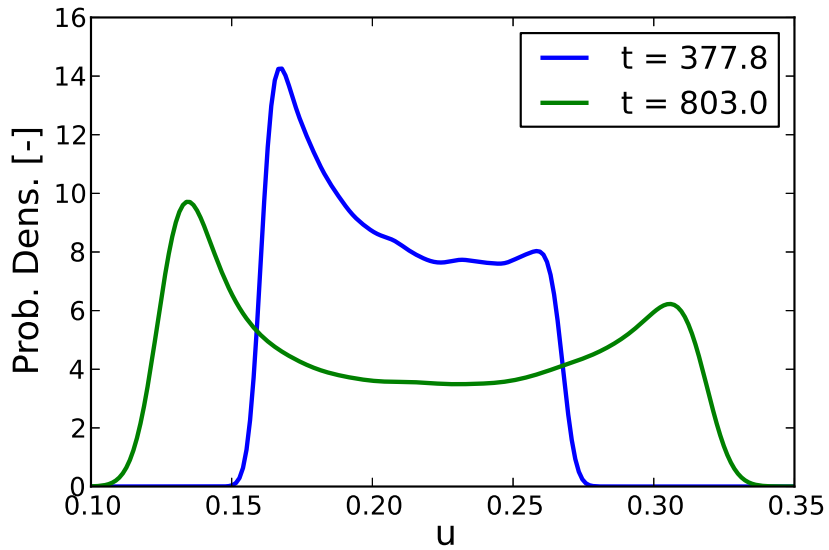
Surface Reaction Model: ISP results



- Modes of *u*
- Modes decay with higher order
- Amplitudes of oscillations of higher order modes grow in time

Surface Reaction Model: ISP results: PDFs





- Pdfs of u at maximum mean (left) and maximum standard deviation (right)
- Distributions get broader and multimodal as time increases
  - Effect of accumulating uncertainty in phase of oscillation

Extra Material on Forward Propagation

- Intrusive operations on PCEs
- Simple ODE example
- Intrusive UQ of incompressible flow
- Uncertainty Quantification Toolkit (UQTk) implementation of intrusive and non-intrusive UQ in surface reaction example (3 ODE system)

Intrusive Spectral Stochastic UQ Formulation: ODE Example

- Sample ODE with parameter λ:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u$$

- Let λ be uncertain; introduce $\xi \sim \mathcal{N}(0,1)$.
- Express λ and u using PCEs in ξ:

$$\lambda = \sum_{k=0}^{P} \lambda_k \Psi_k(\xi), \qquad u(t) = \sum_{k=0}^{P} u_k(t) \Psi_k(\xi)$$

- Substitute in ODE and apply a Galerkin projection on $\Psi_i(\xi)$,

Galerkin Projection on $\Psi_i(\xi)$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\sum_{k=0}^{P} u_k(t)\Psi_k(\xi)\right) = \left(\sum_{p=0}^{P}\lambda_p\Psi_p(\xi)\right)\left(\sum_{q=0}^{P} u_q(t)\Psi_q(\xi)\right)$$

$$\sum_{k=0}^{P}\frac{\mathrm{d}u_k(t)}{\mathrm{d}t}\Psi_k(\xi) = \sum_{p=0}^{P}\sum_{q=0}^{P}\lambda_p u_q(t)\Psi_p(\xi)\Psi_q(\xi)$$

$$\left\langle\sum_{k=0}^{P}\frac{\mathrm{d}u_k(t)}{\mathrm{d}t}\Psi_k(\xi)\Psi_i(\xi)\right\rangle = \left\langle\sum_{p=0}^{P}\sum_{q=0}^{P}\lambda_p u_q(t)\Psi_p(\xi)\Psi_q(\xi)\Psi_i(\xi)\right\rangle$$

$$\sum_{k=0}^{P}\frac{\mathrm{d}u_k(t)}{\mathrm{d}t}\left\langle\Psi_k(\xi)\Psi_i(\xi)\right\rangle = \sum_{p=0}^{P}\sum_{q=0}^{P}\lambda_p u_q(t)\left\langle\Psi_p(\xi)\Psi_q(\xi)\Psi_i(\xi)\right\rangle$$

$$\frac{\mathrm{d}u_i}{\mathrm{d}t}\left\langle\Psi_i^2\right\rangle = \sum_{p=0}^{P}\sum_{q=0}^{P}\lambda_p u_q\left\langle\Psi_p\Psi_q\Psi_i\right\rangle$$

Resulting Spectral ODE system

- $(P+1)$-dimensional ODE system

$$\frac{du_i}{dt} = \sum_{p=0}^{P}\sum_{q=0}^{P} \lambda_p u_q C_{pqi}, \quad i = 0, \ldots, P$$

  where $C_{pqi} = \langle \Psi_p \Psi_q \Psi_i \rangle / \langle \Psi_i^2 \rangle$
- The tensor C_{pqi} can be evaluated once and stored for any given PC order and dimension
- This tensor is sparse, i.e. many elements are zero

Pseudo-Spectral Construction–1

$$w = \lambda u^2 v, \qquad u = \sum_{k=0}^{P} u_k \Psi_k, \qquad \text{similarly for } \lambda \ \& \ v$$

Spectral:

$$\begin{aligned} w_i &= \left\langle \lambda u^2 v \right\rangle_i \\ &= \sum_{j=0}^{P} \sum_{k=0}^{P} \sum_{l=0}^{P} \sum_{m=0}^{P} \lambda_j u_k u_l v_m \left\langle \Psi_j \Psi_k \Psi_l \Psi_m \right\rangle_i , \quad i = 0, \ldots, P \end{aligned}$$

- The corresponding tensor of basis product expectations becomes too large to pre-compute and store

Pseudo-Spectral: Project each PC product onto a $(P+1)$-polynomial before proceeding further, thus:

Pseudo-Spectral Construction–2

$$\tilde{w} = uv \quad : \quad \tilde{w}_i = \langle uv \rangle_i = \sum_{j=0}^{P} \sum_{k=0}^{P} u_k v_j \langle \Psi_k \Psi_j \rangle_i , \quad i = 0, \ldots, P$$

$$\hat{w} = u\tilde{w} \quad : \quad \hat{w}_i = \langle u\tilde{w} \rangle_i = \sum_{j=0}^{P} \sum_{k=0}^{P} u_k \tilde{w}_j \langle \Psi_k \Psi_j \rangle_i , \quad i = 0, \ldots, P$$

$$w = \lambda\hat{w} \quad : \quad w_i = \langle \lambda\hat{w} \rangle_i = \sum_{j=0}^{P} \sum_{k=0}^{P} \lambda_k \hat{w}_j \langle \Psi_k \Psi_j \rangle_i , \quad i = 0, \ldots, P$$

- Aliasing errors
- Efficiency, and convenience

[Debusschere *et al.*, *SIAM J. Sci. Comp.*, 2004.]

Extra Material on Forward Propagation

- Intrusive operations on PCEs
- Simple ODE example
- Intrusive UQ of incompressible flow
- Uncertainty Quantification Toolkit (UQTk) implementation of intrusive and non-intrusive UQ in surface reaction example (3 ODE system)

Spectral UQ: Incompressible Flow - Stochastic Projection Method

- $(P+1)$ Galerkin-Projected Mom./Cont. Eqns, $q = 0, \ldots, P$:

$$\begin{aligned} \frac{\partial \boldsymbol{v}_q}{\partial t} + \nabla \cdot \langle \boldsymbol{v}\boldsymbol{v} \rangle_q &= -\nabla p_q + \frac{1}{\mathrm{Re}} \nabla \cdot \left\langle \mu[(\nabla \boldsymbol{v}) + (\nabla \boldsymbol{v})^T] \right\rangle_q \\ \nabla \cdot \boldsymbol{v}_q &= 0 \end{aligned}$$

- Projection: for $q = 0, \ldots, P$:

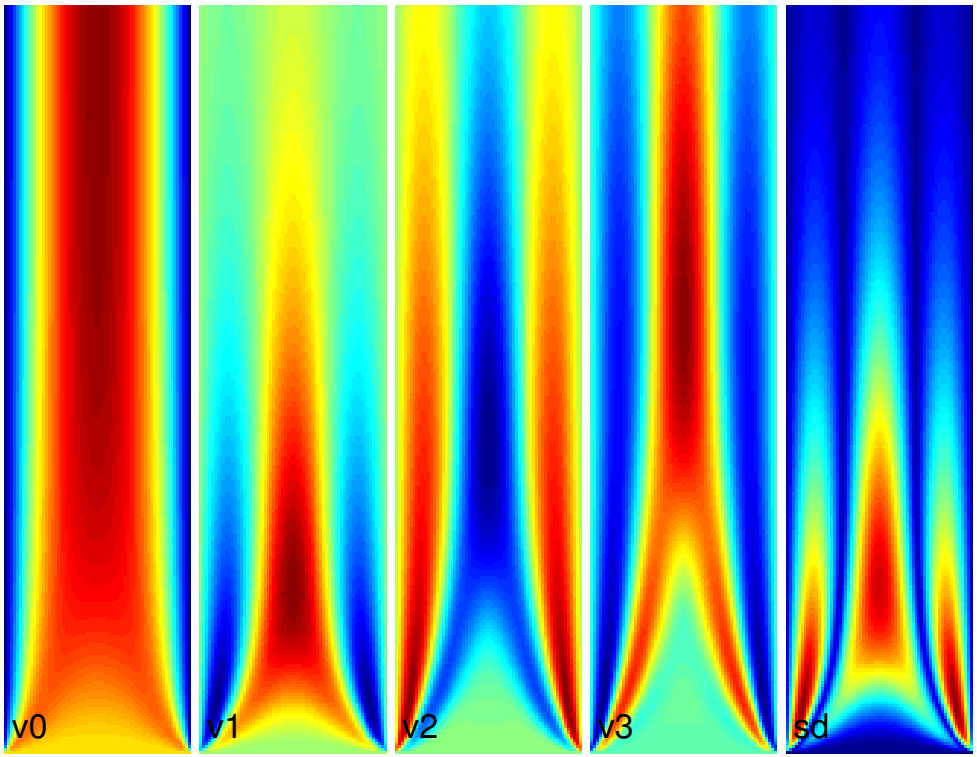
$$\begin{aligned} \frac{\tilde{\boldsymbol{v}}_q - \boldsymbol{v}_q^n}{\Delta t} &= C_q^n + D_q^n \\ \nabla^2 p_q &= -\frac{1}{\Delta t} \nabla \cdot \tilde{\boldsymbol{v}}_q \\ \frac{\boldsymbol{v}_q^{n+1} - \tilde{\boldsymbol{v}}_q}{\Delta t} &= -\nabla p_q \end{aligned}$$

- $P+1$ *decoupled* Poisson Eqns for the pressure modes

[Le Maître *et al.*, *J. Comp. Phys.*, 2001.]

Laminar 2D Channel Flow with Uncertain Viscosity

- Incompressible flow
- Gaussian viscosity PDF
  - $\nu = \nu_0 + \nu_1 \xi$
- Streamwise velocity
  - $v = \sum_{i=0}^{P} v_i \Psi_i$
  - v_0: mean
  - v_i: i-th order mode
  - $\sigma^2 = \sum_{i=1}^{P} v_i^2 \left\langle \Psi_i^2 \right\rangle$



v_0 v_1 v_2 v_3 σ

Extra Material on Forward Propagation

- Intrusive operations on PCEs
- Simple ODE example
- Intrusive UQ of incompressible flow
- Uncertainty Quantification Toolkit (UQTk) implementation of intrusive and non-intrusive UQ in surface reaction example (3 ODE system)

Uncertainty Quantification Toolkit (UQTk)

- A library of C++ and Matlab functions for propagation of uncertainty through computational models
- Mainly relies on Polynomial Chaos Expansions (PCEs) for representing random variables and stochastic processes
- Target usage:
  - Rapid prototyping
  - Algorithmic research
  - Tutorials / Educational
- Version 2.0 about to be released under the GNU Lesser General Public License
  - C++ Tools for intrusive and non-intrusive UQ
  - Matlab tools for intrusive and non-intrusive UQ
  - Karhunen-Loève decomposition
  - Bayesian inference tools
- Downloadable from

  `http://www.sandia.gov/UQToolkit/`

PCEs in the UQToolkit

```
// Initialize PC class
int ord = 5;   // Order of PCE
int dim = 1;   // Number of uncertain parameters
PCSet myPCSet("ISP",ord,dim,"LU"); // Legendre-Uniform PCEs
```

```
// Initialize PC class
int ord = 5;   // Order of PCE
int dim = 1;   // Number of uncertain parameters
PCSet myPCSet("NISP",ord,dim,"LU"); // Legendre-Uniform PCEs
```

- Currently support Wiener-Hermite, Legendre-Uniform, and Gamma-Laguerre (limited), Jacobi-Beta (development version)
- PCSet class initializes PC basis type and pre-computes information needed for working with PC expansions

Operations on PCEs in the UQToolkit

```
// PC coefficients in double*
double* a = new double[npc];
double* b = new double[npc];
double* c = new double[npc];

// Initialization
a[0] = 2.0;
a[1] = 0.1;
...
// Perform some arithmetic
myPCSet.Subtract(a,b,c);
myPCSet.Prod(a,b,c);
myPCSet.Exp(a,c);
myPCSet.Log(a,c);
```

```
// PC coefficients in Arrays
Array1D<double> aa(npc,0.e0);
Array1D<double> ab(npc,0.e0);
Array1D<double> ac(npc,0.e0);

// Initialization
aa(0) = 2.0;
aa(1) = 0.1;
...
// Perform arithmetic
myPCSet.Subtract(aa,ab,ac);
myPCSet.Prod(aa,ab,ac);
myPCSet.Exp(aa,ac);
myPCSet.Log(aa,ac);
```

- PC coefficients are either stored in `double*` vectors or in more advanced custom `Array1D<double>` classes
- Functions can take either data type as argument

Surface Reaction Model: UQTk implementation

```
// Build du/dt = a*z - c*u - 4.0*d*u*v
aPCSet.Multiply(z,a,dummy1);              // dummy1 = a*z
aPCSet.Multiply(u,c,dummy2);              // dummy2 = c*u
aPCSet.SubtractInPlace(dummy1,dummy2); // dummy1 = a*z - c*u
aPCSet.Prod(u,v,dummy2);                  // dummy2 = u*v
aPCSet.MultiplyInPlace(dummy2,4.e0*d); // dummy2 = 4.0*d*u*v
aPCSet.Subtract(dummy1,dummy2,dudt);   // dudt = a*z - c*u - 4.0*d*u
```

- All operations are replaced with their equivalent intrusive UQ counterparts
- Results in a set of coupled ODEs for the PC coefficients
  - u, v, w, z represent vector of PC coefficients
- This set of equations is integrated to get the evolution of the PC coefficients in time

Surface Reaction Model: Second equation implementation

```
// Build dv/dt = 2.0*b*z*z - 4.0*d*u*v
aPCSet.Prod(z,z,dummy1);                    // dummy1 = z*z
aPCSet.Prod(dummy1,b,dummy2);               // dummy2 = b*z*z
aPCSet.Multiply(dummy2,2.e0,dummy1);        // dummy1 = 2.0*b*z*z
aPCSet.Prod(u,v,dummy2);                    // dummy2 = u*v
aPCSet.MultiplyInPlace(dummy2,4.e0*d);      // dummy2 = 4.0*d*u*v
aPCSet.Subtract(dummy1,dummy2,dvdt);        // dvdt = 2.0*b*z*z - 4.0*d*u

// Build dw/dt = e*z - f*w
aPCSet.Multiply(z,e,dummy1);                // dummy1 = e*z
aPCSet.Multiply(w,f,dummy2);                // dummy2 = f*w
aPCSet.Subtract(dummy1,dummy2,dwdt);        // dwdt = e*z - f*w
```

- Dummy variables used where needed to build the terms in the equations
- Data structure is currently being enhanced to provide the operation result as the function return value
  - Will allow more elegant inline replacement of operators with their stochastic counterparts

Surface Reaction Model: NISP implementation in UQTk

Quadrature:

```
// Get the quadrature points
int nQdpts=myPCSet.GetNQuadPoints();
double* qdpts=new double[nQdpts];
myPCSet.GetQuadPoints(qdpts);
...
// Evaluate parameter at quad pts
for(int i=0;i<nQdpts;i++){
  bval[i]=myPCSet.EvalPC(b,&qdpts[i]);
}
...
// Run model for all samples
for(int i=0;i<nQdpts;i++){
  u_val[i] = ...
}
// Spectral projection
myPCSet.GalerkProjection(u_val,u);
myPCSet.GalerkProjection(v_val,v);
myPCSet.GalerkProjection(w_val,w);
```

Monte-Carlo Sampling:

```
// Get the sample points
int nSamples=1000;
Array2D<double> samPts(nSamples,dim);
myPCSet.DrawSampleVar(samPts);
...
// Evaluate parameter at sample pts
for(int i=0;i<nSamples;i++){
  ... // select samPt from samPts
  bval[i]=myPCSet.EvalPC(b,&samPt)
}
...
// Run model for all samples
for(int i=0;i<nSamples;i++){
  u_val[i] = ...
}
// Spectral projection
myPCSet.GalerkProjectionMC(samPts,u_val,u);
myPCSet.GalerkProjectionMC(samPts,v_val,v);
myPCSet.GalerkProjectionMC(samPts,w_val,w);
```