Basis Design for Polynomial Regression with Derivatives.

Yiou Li (Illinois Institute of Technology) ,

Mihai Anitescu and Oleg Roderick (MCS, Argonne)

Also acknowledging the contributions of Fred Hickernell (IIT), Mihai Alexe (Virginia Tech), Zhu Wang (Virginia Tech)

CSE 2011

MS-17 Managing dimension and complexity in surrogate modeling

(Some ?) Components of the Uncertainty Quantification.

- Uncertainty analysis of model predictions: given data about uncertainty parameters $u \in R^p$ and a code that creates output from it $y = f(u)$ characterize y.
- Validation: Use data to test whether the UA model is appropriate or otherwise fit it or both.

- Challenge one: create model for $u \in R^p$ from data. (Mihai's definition of UQ) It does not need to be probabilistic (see Helton and Oberkampf RESS special issue) but it tends to be. What is the statistical model*?
- Challenge two: uncertainty propagation. Since f is expensive to compute, we cannot expect to compute a statistic of y very accurately from direct simulations alone (and there is also curse of dimensionality; exponential growth of effort with dimension). How do I propagate the model if the code is very computationally intensive?
- Challenge three: validation of uncertainty propagation model; or fitting/validation/testing. What are the statistics I choose to test?

Challenge 2: Faster Uncertainty Propagation by Using Derivative Information?

- Uncertainty propagation requires multiple runs of a possibly expensive code.
- On the other hand, adjoint differentiation adds a lot more information per unit of cost (O(p), where p is the dimension of the uncertainty space; though needs lots of memory).
- Q: Can I use derivative information in uncertainty propagation to accelerate its precision per unit of computing time.
- We believe the answer is yes.
- Q: How?
- This is what this presentation is about.

Outline

- 1. Polynomial regression with derivative PRD information: Uncertainty propagation using sensitivity information.
- 2. Obtaining derivative information: Automatic Differentiation of Codes with Substantial Legacy components.
- 3. PRD-based uncertainty propagation: Numerical examples.
- 4. What is a good basis in PRD? Limitations and Construction of Tensor Product Bases. Benefits.

If time permits

- 5. Better Models Gaussian Processes for Quantifying Uncertainty Propagation Error.

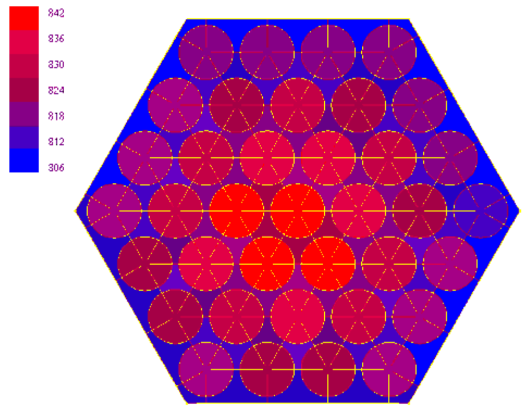
1. Polynomial regression with derivative PRD information: Uncertainty propagation using sensitivity information.

Why a hybrid sensitivity - sampling method for UQ?

- Brute force sampling suffers from the curse of dimensionality: it's efficiency decreases exponentially with the effective dimension of the uncertainty space.
- On the other hand, sensitivity in UQ provides a lot of information but tends to be used ONLY in conjunction with linearization models. As we will demonstrate, this may not go the distance.
- It is reasonable to assume that a hybrid approach will inherit the strengths of both. But how to do it?
- Our answer: polynomial regression with derivative information (PRD).

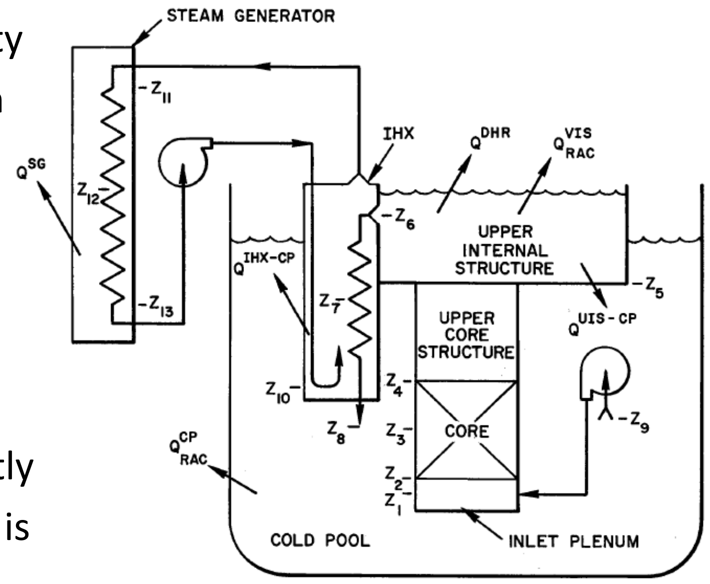
Uncertainty quantification, subject models

- Model I. Matlab prototype code: a steady-state 3-dimensional finite-volume model of the reactor core, taking into account heat transport and neutronic diffusion. Parameters with uncertainty are the material properties: heat conductivity, specific coolant heat, heat transfer coefficient, and neutronic parameters: fission, scattering, and absorbtion-removal cross-sections. Chemical non-homogenuity between fuel pins can be taken into account. Available experimental data is parameterized by 12-66 quantifiers.



- Model II. MATWS, a functional subset of an industrial complexity code SAS4A/SASSYS-1: point kinetics module with a representation of heat removal system. >10,000 lines of Fortran 77, sparsely documented.

MATWS was used, in combination with a simulation tool Goldsim, to model nuclear reactor accident scenarios. The typical analysis task is to find out if the uncertainty resulting from the error in estimation of neutronic reactivity feedback coefficients is sufficiently small for confidence in safe reactor temperatures. The uncertainty is described by 4-10 parameters.



Representing Uncertainty

- We use a hierarchical structure. Given a generic model with uncertainty

  $$F(T,R)=0$$

  $$R=R(T)\cdot(1+\Delta R(T,\alpha)) \qquad J=J(T)$$

  with model state $T=(T_1,T_2,\ldots,T_n)$ $R=(R_1,R_2,\ldots,R_N)$

  intermediate parameters and inputs

  that include errors $\Delta R=(\Delta R_1,\Delta R_2,\ldots,\Delta R_N)$

  An output of interest is expressed by the merit function $J(T)$

  The uncertainty is described by a set of stochastic quantifiers $\alpha=(\alpha_1,\alpha_2,\ldots,\alpha_m)$

- We redefine the output as a function of uncertainty quantifiers, $\Im(\alpha):=J(T)$

  and seek to approximate the unknown function $\Im(\alpha)$

Polynomial Regression with Derivatives, PRD

- We approximate the unknown response function by polynomial regression based on a small set of model evaluations. Both merit function *outputs* and merit function *derivatives* with respect to uncertainty quantifiers are used as fitting conditions.

- PRD procedure:

- choose a basis of multivariate polynomials $\{\Psi_q(\alpha)\}$

the unknown function is then approximated by an expansion $\Im(\alpha) \approx \sum_q x_q \Psi_q(\alpha)$

- choose training set $\{A\}$; $A_i = (\alpha_1^i, \alpha_2^i, \ldots, \alpha_n^i)$

- evaluate the model and its derivatives for each point in the training set

- construct a regression matrix. Each row consists of either the values of the basis polynomials, or the values of derivatives of basis polynomials, at a point in the training set.

- solve the regression matrix (in the least-squares sense) to find coefficients x_q

- Questions (for later):
  - How to best choose the polynomial basis?
  - How to obtain gradient information at computational cost comparable with that of a model run?
  - What to do if dimensionality of uncertainty space is very high?

Polynomial Regression with Derivatives, PRD

- PRD procedure, regression/ collocation* equations:

- Note: the only interaction with the computationally expensive model is on the right side!

- The polynomial regression approach without derivative information would provide *(n+1)* times LESS rows.

- The overall computational savings depend on how cheaply the derivatives can be computed

$$\begin{pmatrix} \Psi_1(A_1) & \Psi_2(A_1) & \cdots \\ \frac{d\Psi_1(A_1)}{d\alpha_1} & \frac{d\Psi_2(A_1)}{d\alpha_1} & \cdots \\ \frac{d\Psi_1(A_1)}{d\alpha_2} & \frac{d\Psi_2(A_1)}{d\alpha_2} & \cdots \\ & \vdots & \\ \frac{d\Psi_1(A_1)}{d\alpha_m} & \frac{d\Psi_2(A_1)}{d\alpha_m} & \cdots \\ \Psi_1(A_2) & \Psi_2(A_2) & \cdots \\ \frac{d\Psi_1(A_2)}{d\alpha_1} & \frac{d\Psi_2(A_2)}{d\alpha_1} & \cdots \\ & \vdots & \\ \Psi_1(A_M) & \Psi_2(A_M) & \cdots \\ & \vdots & \\ \frac{d\Psi_1(A_M)}{d\alpha_m} & \frac{d\Psi_2(A_M)}{d\alpha_m} & \cdots \end{pmatrix} \cdot x = \begin{pmatrix} \Im(A_1) \\ \frac{d\Im(A_1)}{d\alpha_1} \\ \frac{d\Im(A_1)}{d\alpha_2} \\ \vdots \\ \frac{d\Im(A_1)}{d\alpha_m} \\ \Im(A_2) \\ \frac{d\Im(A_2)}{d\alpha_1} \\ \vdots \\ \Im(A_M) \\ \vdots \\ \frac{d\Im(A_M)}{d\alpha_m} \end{pmatrix}$$

Equivalence between various approaches.

- Collocation is equivalent to interpolation (if basis is uniquely defined)

$$F(\tilde{\mathbf{T}}(x_i), R(\tilde{\mathbf{T}}(x_i), x_i)) = 0, \quad i = 1,2,\ldots,m. \Leftrightarrow \sum_l \beta_l^T \Psi_l(x_i) = \tilde{\mathbf{T}}(x_i) = T_i, \quad i = 1,2,\ldots,m.$$

- If Response function is linear in state, the collocation is equivalent to surface response.

$$\sum_l \beta_l^J \Psi_l(x_i) = \sum_l J(\beta_l^T) d\Psi_l(x_i) = J(x_i) = J(\tilde{\mathbf{T}}(x_i)).$$

- If it has a unique solution, surface response is equivalent to Regression:

$$\min_{\beta_i^J} \sum_{i=1}^{M} \left(J(x_i) - \sum_l \beta_l^J \Psi_l(x_i) \right)^2,$$

- Same situation when derivatives are used.
- Therefore surface response, regression, collocation, interpolation are not far from each other. We may call regression matrix collocation matrix.

Cost versus benefit of using gradient information.

- Theory: Cost of gradient evaluation can be at most 5 times larger than cost of function evaluation. Therefore relative efficiency of using gradient information versus using one more sample point: at least n/5.

- This bound is achieved by adjoint calculations or reverse automatic differentiation mode.

- Sometimes, the bound is much smaller. Example: Coupling in multiphysics achieved by operator splitting/ Gauss Seidel since Newton method may not converge. Then compute adjoint at converged point.

- When instrumenting code for gradient with AD we are somewhere between intrusive and non-intrusive methods for UQ. Clearly not as simple as brute force sampling, but not as intrusive as Galerkin stochastic FEM methods either. But payoff of same accuracy for fewer samples is a great driver.

- In principle, can be operated without knowing what the code does, in practice, the latter helps.

2. Obtaining derivative information: Automatic Differentiation of Codes with Substantial Legacy components.

PRD, computation of derivatives

- Hand-coding derivatives is error-prone, has large development cost, code maintenance is a problem.
- Finite difference approximations introduce truncation errors, and Cost of gradient ~ Dimension X Cost of the function, and advantage of adjoints is lost.
- For most applied purposes, a more promising approach is Automatic (Algorithmic) Differentiation, AD. It also uses the chain-rule approach, but with minimal human involvement.
- Ideally, the only required processing is to identify inputs and outputs of interest, and resolve the errors at compilation of the model augmented with AD.

Automatic Differentiation, AD

- AD is based on the fact that any program can be viewed as a finite sequence of elementary operations, the derivatives of which are known. A program *P* implementing the function *J* can be parsed into a sequence of elementary steps:

$$P: \quad J = f_k(f_{k-1}(\dots f_1(\alpha)))$$

The task of AD is to assemble a new program *P'* to compute the derivative. In *forward* mode:

$$P': \quad (\nabla_\alpha J)_i = \frac{\partial f_k}{\partial f_{k-1}} \cdot \frac{\partial f_{k-1}}{\partial f_{k-2}} \cdot \dots \cdot \frac{\partial f_1}{\partial \alpha_i}$$

- In the forward (or *direct*) mode, the derivative is assembled by the chain rule following computational flow from an input of interest to all outputs. We are more interested in the *reverse* (or *adjoint*) mode that follows the reversed version of the computational flow from an output to all inputs:

$$P': \quad (\nabla_\alpha J) = \left(\frac{\partial f_1}{\partial \alpha}\right)^T \cdot \left(\frac{\partial f_2}{\partial f_1}\right)^T \cdot \dots \cdot \left(\frac{\partial f_k}{\partial f_{k-1}}\right)^T$$

In adjoint mode, the complete gradient can be computed in a single run of P', as opposed to multiple runs required by the direct mode.

- For inherently non-differentiable components of code, it is possible to construct a smooth interpolation. THIS is one of the many cases where it helps UQ to be integrated with a physics team which we believe and we practice (We will not discuss nondifferentiability here).

AD tools, Fortran

- TAF (FastOpt)
  - Commerical tool
  - Support for almost all of Fortran 95
  - Used extensively in geophysical sciences applications

- Tapenade
  - Support for many Fortran 95 features
  - Developed by a team with extensive complier experience

- OpenAD/F
  - Support for many Fortran 95 features
  - Developed by a team with expertise in combinatorial algorithms, compilers, software engineering, and numerical analysis
  - Development driven by climate modeling and astrophysics applications

- ADIFOR
  - Mature, very robust tool. Support for all of Fortran 77 :forward and adjoint modes
  - Hundreds of users, over 250 citations

AD tools, Capabilities

- Fast O(1) computation of
  - Gradient (in adjoint mode)
  - Derivative matrix-vector products

- Efficient computation of full Jacobians and Hessians, able to exploit sparsity, low-rank structure

- Efficient high-order directional derivative computation

- Minuses: it is still not a mature technology (after 30 years !!!) except for very specific cases (e.g codes written entirely in Fortran 77+ STANDARD).

- We believe in (and we practice) close integration with an AD development team (Jean Utke, Mihai Alexe)

Applying AD to code with major legacy components

- We investigated the following question: are AD tools now at a stage where they can provide derivative information for realistic nuclear engineering codes? Many models of interest are complex, sparsely documented, and developed according to older (Fortran 77) standards.

- Based on our experience with MATWS, the following (Fortran 77) features make application of AD difficult:

- Not supported by AD tools (since they are nonstandard) /need to be changed.
  - machine-dependence code sections need to be removed (i/o)
  - Direct memory copy operations needs to be rewritten as explicit operations (when LOC is used)
  - COMMON blocks with inconsistent sizes between subroutines need to be renamed
  - Subroutines with variable number of parameters need to be split into separate subroutines

- EQUIVALENCE, COMMON, IMPLICIT* definitions are supported by most tools though they have to be changed for some (such as OpenAD). (for Open AD statement functions need to be replaced by subroutine definitions, they are not supported in newer Fortran)

- Note that the problematic features we encountered have to do with memory allocation and management and i/o, not mathematical structure of the model! We expect that (differentiable) mathematical sequences of any complexity can be differentiated.

Validation of AD derivative calculation

- Model II, MATWS, subset of SAS4A/SASSYS-1. We show estimates for the derivatives of the fuel and coolant maximum temperatures with respect to the radial core expansion coefficient ,obtained by different AD tools, and compared with the Finite Differences approximation, FD.

All results agree with FD within 0.001% (and almost perfectly with each other).

AD tool	Fuel temperature derivative, K	Coolant temperature derivative, K
ADIFOR	18312.5474227	17468.4511373
OpenAD/F	18312.5474227	17468.4511372
TAMC	18312.5474248	17468.4511392
TAPENADE	18312.5474227	17468.4511372
FD	18312.5269537	17468.4315994

3. PRD-based uncertainty propagation: Numerical examples.

PRD UQ, tests on subject models 1.

- Model I, Matlab prototype code. Output of interest: maximal fuel centerline temperature.
- We show performance of a version with 12 (most important) uncertainty quantifiers. Performance of PRD approximation with full and truncated basis is compared against random sampling approach (100 samples)*:

	Sampling	Linear approximation	PRD, full basis	PRD, truncated basis
Full model runs	100	1*	72*	12*
Output range, K	2237.8 2460.5	2227.4 2450.0	2237.8 2460.5	2237.5 2459.6
Error range, K		-10.38 +0.01	-0.02 +0.02	-0.90 +0.90
Error st. deviation		2.99	0.01	0.29

* derivative evaluations required ~150% overhead

PRD, basis truncation

- Issue: we would like to use high-order polynomials to represent non-linear relationships in the model. But, even with the use of derivative information, the required size of the training set grows rapidly (curse of dimensionality in spectral space)

- We use a heuristic: we rank uncertainty quantifiers by importance (a form of sensitivity analysis is already available, for free!) and use an incomplete basis, i.e. polynomials of high degree only in variables of high importance. This allows the use of some polynomials of high degree (maybe up to 5?)
- Several versions of the heuristic are available, we choose to fit a given computational budget on the evaluations of the model to form a training set.

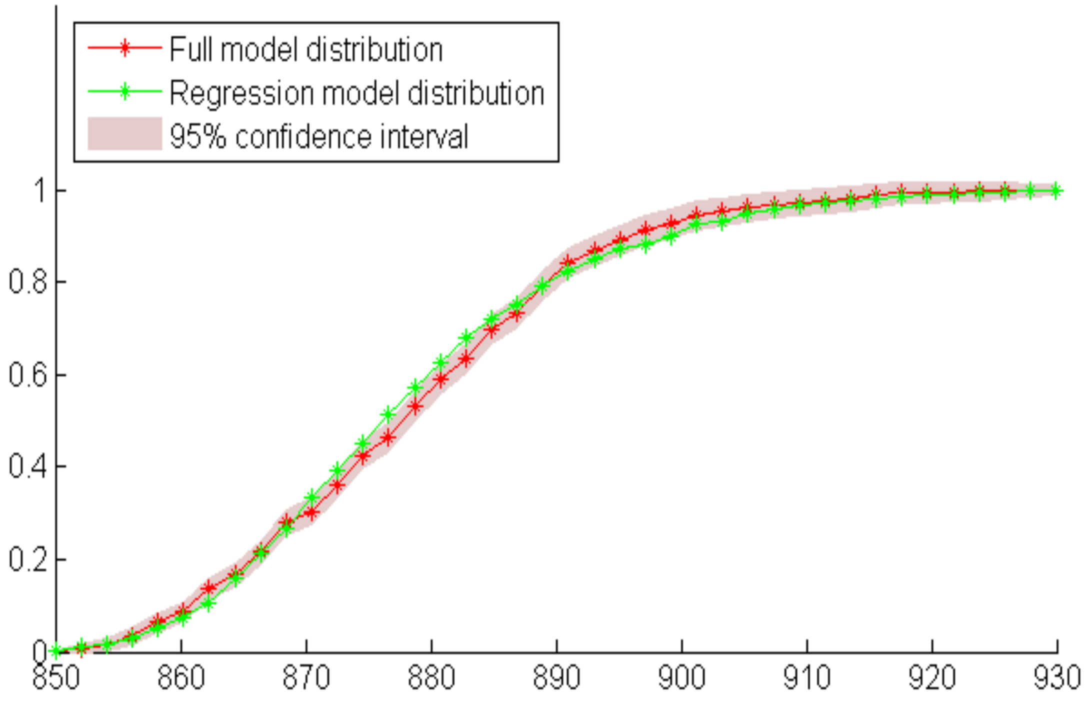
- In our first experiments, we use either a complete basis of order up to 3, or its truncated version allowing the size of training set to be within 10-50 evaluations.

- An even better scheme - adaptive basis truncation based on stepwise fitting is developed later, simultaneously with conditions for better algebraic form of multivariate basis,

Uncertainty quantification, tests on subject models

- Model II, MATWS, subset of SAS4A/SASSYS-1. We repeat the analysis of effects of uncertainty in an accident scenario modeled by MATWS + GoldSim. The task is to estimate statistical distribution of peak fuel temperature.
- We reproduce the distribution of the outputs correctly*;

regression constructed on 50 model evaluations thus replaces analysis with 1,000 model runs. We show cumulative distribution of the peak fuel temperature.



- Note that the PRD approximation is almost entirely within the 95% confidence interval of the sampling-based results.

- We also have error model now (Lockwood, MS72, Wed 10:30, Carson3)

4. What is a good basis in PRD?

PRD, selection of better basis

- We inherited the use of Hermite multivariate polynomials as basis from a related method: Stochastic Finite Elements expansion.

- While performance of PRD so far is acceptable, Hermite basis may not be a good choice for constructing a regression matrix with derivative information; it causes poor condition number of linear equations (of the Fischer matrix).

- Hermite polynomials are generated by orthogonalization process, to be orthogonal (in probability measure ρ; Gaussian measure is the specific choice):
$$\int_{\Omega} \Psi_j(A)\Psi_h(A)\rho(A)dA = \delta_{jh}$$

- We formulate new orthogonality conditions:
$$\int_{\Omega}\left(\Psi_j(A)\Psi_h(A) + \sum_{i=1}^{m} \frac{\partial\Psi_j(A)}{\alpha_i}\cdot\frac{\partial\Psi_h(A)}{\alpha_i}\right)\rho(A)dA = \delta_{ih}$$

  and ask the question: how does a good basis with respect to this inner product looks like?

Insight on the scalar product -- framework

- Define the linear mapping

$$\mathbf{L}_{\mathbf{x}} f = \left(f(\mathbf{x}), \frac{\partial f}{\partial x_1}(\mathbf{x}), \cdots, \frac{\partial f}{\partial x_d}(\mathbf{x}) \right)^T .$$

- For a vector function

$$\mathbf{L}_{\mathbf{x}} \mathbf{f}^T = \begin{pmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \cdots & f_k(\mathbf{x}) \\ \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_k}{\partial x_1}(\mathbf{x}) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_1}{\partial x_d}(\mathbf{x}) & \frac{\partial f_2}{\partial x_d}(\mathbf{x}) & \cdots & \frac{\partial f_k}{\partial x_d}(\mathbf{x}) \end{pmatrix} .$$

- The collocation matrix in this framework, for a given set of polynomials Ψ is

$$\mathbf{F} = \begin{pmatrix} \mathbf{L}_{\mathbf{x}_1} \Psi^T \\ \mathbf{L}_{\mathbf{x}_2} \Psi^T \\ \vdots \\ \mathbf{L}_{\mathbf{x}_m} \Psi^T \end{pmatrix} .$$

What guides me when I choose the polynomials?

- What is the least-squares matrix?

$$\frac{1}{m}\mathbf{F}^T\mathbf{F} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{L}_{\mathbf{x}_i}\Psi\left(\mathbf{L}_{\mathbf{x}_i}\Psi^T\right) \approx$$

$$\approx \int_\Omega \mathbf{L}_\mathbf{x}\Psi\left(\mathbf{L}_\mathbf{x}\Psi^T\right)\rho(\mathbf{x})d\mathbf{x} = \left(\int_\Omega\left(\Psi_j(\mathbf{x})\Psi_h(\mathbf{x}) + \sum_{i=1}^{d}\frac{\partial\Psi_j}{\partial x_i}(\mathbf{x})\frac{\partial\Psi_h}{\partial x_i}(\mathbf{x})\right)\rho(\mathbf{x})d\mathbf{x}\right)_{j,h=1}^{k}.$$

- For best use of information, I would like this matrix to be almost a multiple of the identity.
- In turn, this requires the polynomials to be orthogonal with respect to this new inner product.

Tensor Product Bases

- These are possibly the most common polynomial basis sets.
- When one uses only function values (and not derivatives) we can construct orthogonal tensor product bases of arbitrary order and arbitrary number of variables.
- Does the same hold for this new orthogonal product? (it clearly does in one dimension)
- Surprise, no, as can be shown by a counterexample.
- Univariate orthogonal polynomials :

$$1, x_1, x_1^2 - \frac{1}{3}, x_1^3 - \frac{9}{10} x_1, x_2, x_2^2 - \frac{1}{3}, x_2^3 - \frac{9}{10} x_2 .$$

- Offenders:

$$\left\langle x_1 \cdot x_2 , x_1 \cdot \left(x_2^3 - \frac{9}{10} x_2 \right) \right\rangle$$

Do tensor products ever work?

- Yes, sometimes:

Corollary 2. **[ ]**
The following choices of Γ all satisfy the satisfy the criteria of Theorem 2 that guarantees an orthonormal multivariate basis:

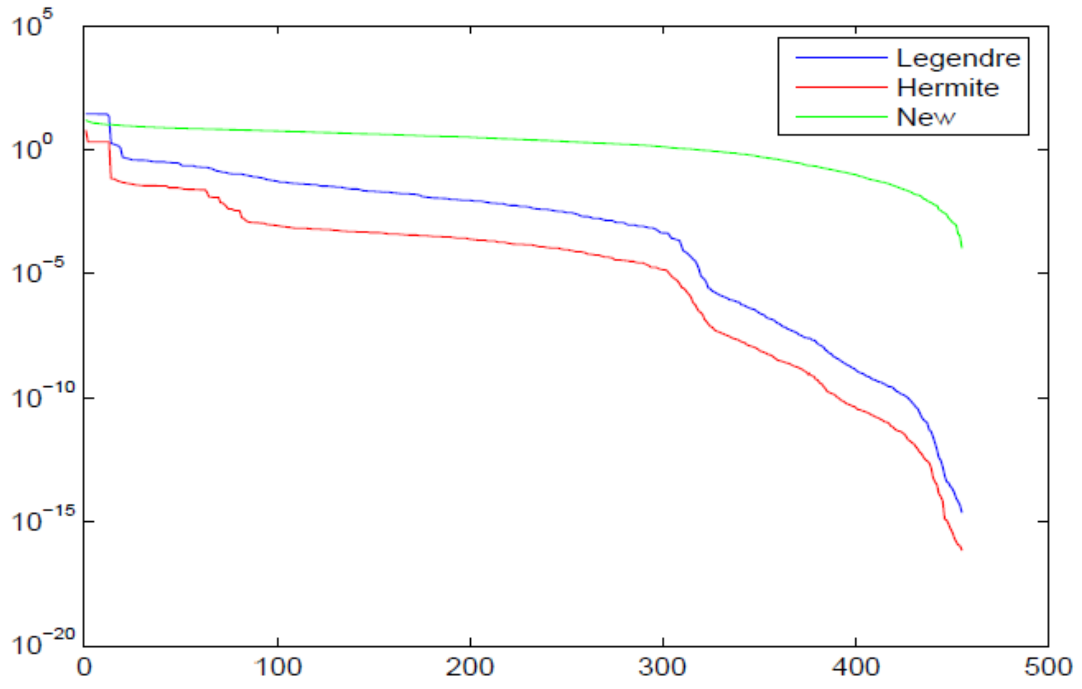
$$\Gamma_1 = \{\mathbf{p} \in \mathbb{N}_0^d : \|\mathbf{p}\|_1 \leq 3\}, \qquad \Gamma_2 = \{\mathbf{p} \in \mathbb{N}_0^d : \|\mathbf{p}\|_\infty \leq 2\},$$
$$\Gamma_3 = \Gamma_2 \cup \{\mathbf{p} = (0, \ldots, 0, p_i, 0, \ldots, 0)^T \in \mathbb{N}_0^d : i = 1, \ldots, d\}.$$

- For higher dimensions, perhaps use GS, but no easy solution here.
-

5. NUMERICAL RESULTS

PRD, selection of better basis

- Model I, Matlab prototype code. We compare the setup of PRD method using Hermite polynomial basis and the improved basis. We observe the improvement in the distribution of singular values of the collocation matrix.
- We compare numerical conditioning for Hermite, Legendre polynomials, and the basis based on new orthogonality conditions.
- We have 10^10 improvement in the condition number of the Fischer matrix *!!! In principle this results in much more robustness of the matrix.
- This will offer us substantial flexibility in creating the PRD model.

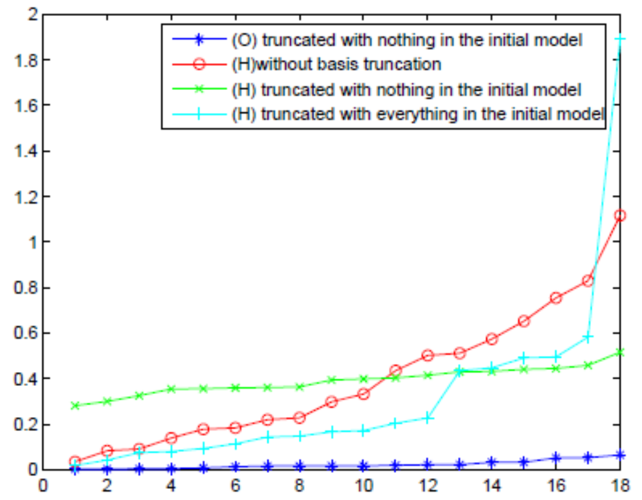


PRD, adaptive (stepwise fitting) basis truncation

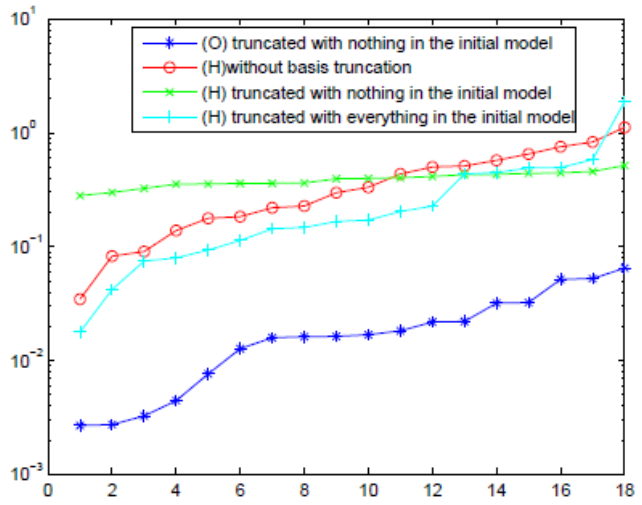
- We use a stepwise fitting procedure (based on F-test):

1. Create the PRD model as an expansion in the starting set of polynomials
2. Add one (estimated as most likely) polynomial to the set. An expansion term currently not in the model is added if, out of all candidates, it has the largest likelihood that it would have non-negligible coefficient if added to model.
3. Remove one (estimated as least likely) polynomial from the set. An expansion term in the model is removed if it has the highest likelihood to have negligible coefficient.

- It is possible to truncate the model starting with a full basis set (of fixed maximal polynomial order) or from an empty basis set (all polynomials of fixed maximal order are candidates to be added).



(Hermite basis error on 20 samples)



(Orthogonal basis error on 20 samples, log_10 plot)

- Orthogonal basis created starting "with nothing" in the expansion results in precision of up to 0.01 degree K (compare with errors of >10 K by linear model).

Conclusions.

- Polynomial Regression with Derivatives promises smaller error for the same number of runs of the model but has new challenges compared with standard polynomial regression:

- Standard inner product is not the most suitable, we introduced a new one that includes derivative info.
- Nevertheless, we demonstrate an orthogonal tensor product basis of arbitrary order may not exist in this inner product.
- Established new sufficient (and almost necessary) conditions for the larger tensor product basis of finite order.

- Future Research:
  - Explore other basis generation approaches, to circumvent the order limit.
  - Better error model (see Lockwood, MS72, Wed 10:30, Carson3 talk).
  - Need to find a balance between the need for high-order polynomials, and limitations on the number of samples