Model-Order Reduction Using Interval Constraint Solving Techniques

Leobardo Valera and Martine Ceberio
Computer Science Department, The University of Texas at El Paso, El Paso, TX, 79968, USA,
{*lvalera, mceberio*}*@utep.edu*

Abstract: Many natural phenomena can be modeled as ordinary or partial differential equations. A way to find solutions of such equations is to discretize them and to solve the corresponding (possibly) nonlinear large systems of equations; see (Li and Chen, 2008).

Solving a large nonlinear system of equations is very computationally complex due to several numerical issues, such as high linear-algebra cost and large memory requirements. Model-Order Reduction (MOR) has been proposed as a way to overcome the issues associated with large dimensions, the most used approach for doing so being Proper Orthogonal Decomposition (POD); see (Schilders and Vorst, 2008). The key idea of POD is to reduce a large number of interdependent variables (snapshots) of the system to a much smaller number of uncorrelated variables while retaining as much as possible of the variation in the original variables.

In this work, we show how intervals and constraint solving techniques can be used to compute all the snapshots at once (I-POD); see (Granvilliers and Benhamou, 2006; Kreinovich and Ceberio, 2006; Moore and Kearfott, 2009). This new process gives us two advantages over the traditional POD method: 1. handling uncertainty in some parameters or inputs; 2. reducing the snapshots computational cost.

Keywords: Model-Order Reduction; Proper Orthogonal Decomposition; Large Nonlinear Systems of Equations; Interval Constraint Solving Techniques

1. Introduction

Many real life phenomena or situations can be represented by mathematic models. Because of the dynamic nature of these phenomena, the associated models could be, for instance, partial differential equations (PDEs). These kind of equations arise in many engineering problems describing phenomena such as the distribution of heat in a given rod or plate over time (heat equation), the description of waves like those of vibrating strings, and sound and water waves (wave equation), gas dynamics and traffic flow (Burgers' equation); see (Sharan and Pradhant, 2013; White, 2013).

A way to find an approximation of the solution of differential equations, either ordinary or partial, is to discretize the domain of the solution and to form a system of algebraic equations: this system of equations can be linear or nonlinear depending on the nature of the PDE.

In order to obtain a good accuracy in the approximation of the sought solution, the domain has to be discretized in many elements and nodes, leading to a large system of equations inheriting

several issues, such as: 1. not knowing about the existence and/or uniqueness of the solution of the system of equations, 2. storage, 3. high computational cost, 4. rounding errors.

To overcome these issues, several techniques have been developed to find a subspace where the an acceptable approximation of the solution of the system of equations lie. This process of identifying such subspace and reducing a large problem to a smaller one is known as **Model-Order Reduction (MOR)**.

Proper Orthogonal Decomposition (POD) is a broadly used and effective method to identify a reduced subspace and reduce the original large problem to a much smaller one. This method is based on Principal Component Analysis (PCA) (Cai and White, 2003; Marquez and Espinosa, 2013; Rathinam and Petzold, 2003). The idea behind POD consists in the following. First, identifying the function $U(x, t; \lambda)$, where x for solving the high dimensional Full-Order Model (FOM) for different values of its parameter, λ (chosen in a given interval or cartesian product of intervals). The function $U(x, t; \lambda)$ is defined in a space-temporal domain, where x represents the spatial variable and t the temporal one. For each parameter λ, after the FOM has been solved, m samples, $U(x, t_i, \lambda)$ are selected, commonly referred as snapshots, with $1 \leq i \leq m$. In this work, the instants of time at which the samples were selected were chosen in a uniform distribution. These snapshots are then arranged into a matrix A, which is factored: $A = U\Sigma V^T$, using Singular Value Decomposition (SVD). The basis of the reduced subspace is made of the first k columns, which are the ones that preserve most of the correlation.

Although POD is one of the most popular approaches to MOR, it presents several disadvantages, the main drawback being that it requires a series of offline computations in order to form the matrix of snapshots. The quality of the resulting reduced basis heavily depends on the choice of parameters and inputs, and on the accuracy of these over which the snapshots are computed.

In this work, we explored the idea of computing snapshots as a result of interval computations that we then sample: this leads us to one – interval – solving process as opposed to computing snapshots as a result of computations on sample parameters (many solving processes). This led to what we called an Interval-POD approach (IPOD), which has advantages beyond the mere computations of snapshots: if POD can handle intervals, it can therefore handle uncertainty as well. This would allow models to factor in uncertainty while still making it possible to process via MOR. Our preliminary tests show promise.

2. Background

Let us start by recalling the type of problems that we are attempting to solve. Many real-life phenomena are modeled and result in very large (most likely) nonlinear systems of equations that need to be solved. Solving these problems boils down to finding the zeroes of large-dimensional functions. Traditionally, finding zeroes of functions is achieved via the use of Newton methods.

In this section, we review basic notions about the components that motivate and make up our approach. Namely, we start by recalling the Newton's approach, which motivates the need to Model-Order Reduction. We then go over the MOR concept. Finally, we review interval computations and interval constraint solving techniques, as they are essential to our proposed IPOD.

2.1. The Newton Method

The Newton method is an iterative procedure that finds the zeroes of continuously differentiable functions $F : \mathbb{R}^n \to \mathbb{R}^n$. The formulation of the method is given by:

$$J_F(x_n)(x_{n+1} - x_n) = -F(x_n) \tag{1}$$

where $J_F(x_n)$ is the $n \times n$ Jacobian matrix of F.

If F is twice differentiable and the Hessian $\nabla^2 F(x)$ is Lipschitz continuous in a neighborhood of a solution x^* then:

1. if the initial point x_0 is sufficiently close to x^*, the sequence of iterations converges to x^*; and
2. the rate of convergence of $\{x_k\}$ is quadratic.

The Newton method is outlined in Table I:

Table I. Newton Method Outline.

Given an initial point x_0
`for i=1 until convergence`
Compute $F = F(x_0)$ and $J = J_F(x_0)$
Solve the linear system of equations: $J\Delta x = -F$,
Compute: $x_{i+1} = x_i + \Delta x$
`end for`

The Newton method converges if certain conditions are satisfied; for example, if a stationary initial point is chosen or if the approach oulined above enters in a cycle, the Newton method will not converge. Also, if the Jacobian matrix is singular or if any of its entries is discontinuous at the root, the convergence may fail. If the Jacobian is singular at the root of the function or the Hessian is not defined at it, the process may converge but not in q-quadratic order.

In addition to the above limitations of the Newton's approach, let us recall that the systems that are being considered for solution are of very large size. Not meeting a q-quadratic order of convergence is much more critical on such large spaces than it would be on smaller spaces.

To overcome all issues above mentioned, the solution is sought on a subspace where the convergence conditions are met, hence Model-Order Reduction.

2.2. Model-Order reduction

The main idea of the concept of Model Order Reduction (MOR) is as follows:

Let $T : V \to V$ be a bijective linear transformation. Then for every $b \in V$, there exists a unique $x \in V$ such that $T(x) = b$. Every linear transformation has a matrix representation (Hoffman, 1971). In this case, let us call A the matrix representation of T. Thus, finding x such that $T(x) = b$ is equivalent to solving the linear system:

$$Ax = b. \tag{2}$$

If the dimension of V is n, then Eq.(2) is a $n \times n$ linear system of n equations and n unknowns.

We can assure that there exists W, a subspace of V, whose dimension is $k \ll n$ and such that $x \in W$. This is true because, in particular, the subspace spanned by $\{x\}$ is a subspace of V, which contains x and whose dimension is $1 \ll n$.

Since W is a subspace of V, there exists a base $B = \{w_1, w_2, \ldots, w_k\}$ such that every element $w \in W$ can be expressed as a linear combination of the elements of B (Cotlar, 1974). In particular, if $w = x$,

$$x = \sum_{i=1}^{k} y_i w_i. \tag{3}$$

Since every base uniquely determines a subspace of V, we can, without loss generality, speak about subspace W and its base without difference. By writing Eq.(3) in matrix form, we obtain

$$Wy = x. \tag{4}$$

After substituting (4) in (2), we have:

$$(AW)y = b, \tag{5}$$

that can be solved using the normal equation (Bjork, 1996)

$$(AW)^T(AW)y = (AW)^T b, \tag{6}$$

which is itself a $k \times k$ linear system of equations. After we identify y, we can use Eq.(4) to find x.

Once the subspace is found, the approximation of the solution is given as the projection of it on the subspace obtained. This method truncates the solution of the original system to an appropriate basis. Let us illustrate this method by considering a basis transformation T that maps the original n-dimensional state space x into a vector that we will denote by

$$T(x) = \begin{pmatrix} T_1(x) \\ T_2(x) \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix}$$

where $\hat{x}$ is k-dimensional. Suppose that T has at least a right-inverse. Let us denote $S = T^{-1}$ then S can be written as

$$S = (S_1 \;\; S_2)$$

and

$$\begin{aligned} I &= \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} (S_1 \;\; S_2) \\ &= \begin{pmatrix} T_1 S_1 & T_1 S_2 \\ T_2 S_1 & T_2 S_2 \end{pmatrix} \qquad (7) \\ &= \begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix}. \qquad (8) \end{aligned}$$

Since $T_1 S_1 = I_k$, we have $\Pi = S_1 T_1$ is an oblique projection along the kernel of T_1 onto the k-dimensional subspace that is spanned by the columns of the matrix S_1.

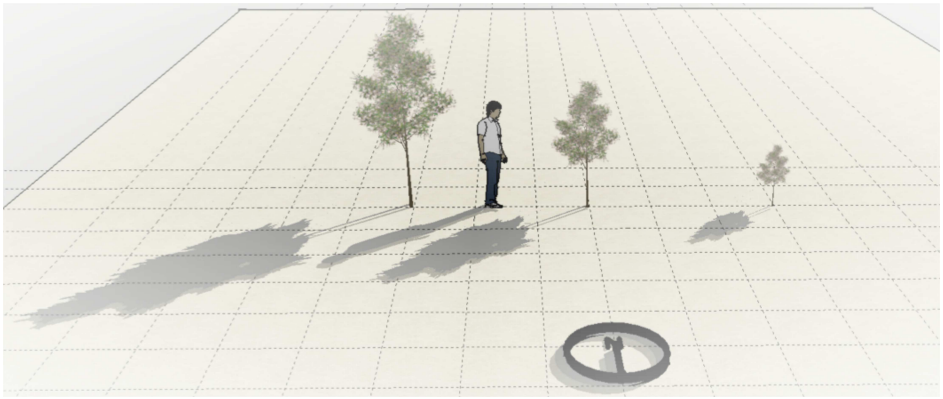

Figure 1. An oblique projection can be see as the shadow cast by objects on the ground when the sun is not directly vertical. Image taken from the site: http://www.schoolkitchengarden.com.au/design-your-garden/.

Let

$$\begin{array}{rcl} \dfrac{dx}{dt} & = & f(x,u), \\ y & = & g(x,u), \\ x(t_0) & = & x_0 \end{array} \tag{9}$$

be the dynamical system, where u is the input of the system, y is the output, x the so-called *state variable*. If we substitute the projection into the dynamical system Eq.(19), we obtain

$$\begin{array}{rcl} \dfrac{d\hat{x}}{dt} & = & T_1\hat{f}(S_1\hat{x} + S_2\tilde{x}, u), \\ y & = & \hat{g}(S_1\hat{x} + S_2\tilde{x}, u). \end{array} \tag{10}$$

The approximation occurs when we delete the terms involving $\tilde{x}$

$$\begin{array}{rcl} \dfrac{d\hat{x}}{dt} & = & T_1\hat{f}(S_1\hat{x}, u), \\ y & = & \hat{g}(S_1\hat{x}, u). \end{array} \tag{11}$$

In order to obtain a good approximation to the original system, the term $S_2\tilde{x}$ must be sufficiently small.

2.3. Interval Constraint Solving Techniques

The method that we are proposing in this article consists in expanding POD-based MOR techniques to interval computations (as we will describe in Section (4). Indeed, we aim to group the (possibly many) computational processes over the reals required to generate all snapshots into one single computational process over intervals that encompasses all computations over the reals.

In this subsection, we give a brief overview of interval computations and how to solve systems of equations that involve intervals; for more details about the field, please see (Moore and Kearfott, 2009).

2.3.1. *Computations with Intervals*
Let us start by pointing that in what follows, when mentioning intervals, we actually mean *closed intervals*. In addition, for simplicity, when we talk about intervals, we will talk about real-value-bounded intervals (not just floating-point-bounded intervals as is commonly the case when implemented on a computer).

So in this work, an interval X is defined as follows:

$$X = [\underline{X},\ \overline{X}] = \{x \in \mathbb{R} : \underline{X} \leq x \leq \overline{X}\}. \tag{12}$$

Operations on intervals are simply defined as follows: Since $x \in X$ means that $\underline{X} \leq x \leq \overline{X}$, and $y \in Y$ means that $\underline{Y} \leq y \leq \overline{Y}$ the followings operations are defined based on its infimum and supremum:

$$\textbf{Addition:}\ \ X + Y = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] \tag{13}$$

$$\textbf{Substraction:}\ \ X - Y = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}] \tag{14}$$

$$\textbf{Multiplication:}\ \ X \cdot Y = [\min S, \max S],\ \text{where } S = \{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}\} \tag{15}$$

As we observe above, combining intervals with addition, subtraction, and multiplication, always results in one interval. However, it is not always the case without extra care. For instance, the division of an interval by another one that contains 0 should result in two disjunct intervals. To avoid such cases with compromise the nature of traditional interval computations (according to which combining intervals should result in an interval), we generalize the combination of two intervals as follows:

$$\forall X, Y \text{ intervals},\ \ X \lozenge Y = \square\{x \lozenge y,\ \text{where } x \in X \text{ and } y \in Y\} \tag{16}$$

where $\lozenge$ stands for any arithmetic operator, including division, and $\square$ represents the hull operator.

More generally, when carrying out more general computations involving intervals, e.g., computing the interval value of a given function $f : \mathbb{R}^n \to \mathbb{R}$ on interval parameters (or a mix of interval and real-valued parameters), we have the following property:

$$f(X_1, \ldots, X_n) \subseteq \square\{f(x_1, \ldots, x_n),\ \text{where } x_1 \in X_1, \ldots, x_n \in X_n\} \tag{17}$$

where $f(X_1, \ldots, X_n)$ represents the range of function f over the domain $X_1 \times \ldots \times X_n$ and $\square\{f(x_1, \ldots, x_n),\ \text{where } x_1 \in X_1, \ldots, x_n \in X_n\}$ represents the smallest closed interval enclosing this range. Computing the exact range of f over intervals is therefore a very hard problem and instead, we approximate the range of f over domains using what we call an interval extension of f, which is in fact a surrogate interval function F.

Interval extensions of a given function f have to satisfy the following (very lose) property:

$$f(X_1, \ldots, X_n) \subseteq F(X_1, \ldots, X_n) \tag{18}$$

which to some extent would allow F to be the function that maps any input to the interval $[-\infty, +\infty]$. More pragmatically, the aim is to identify a function F that does not dramatically overestimate the range of our original function f (the closer to the range the better of course, but cost of achieving better range is also an issue). Many interval extensions exists. The most common one is the so-called natural extension, which is a simple interval extension of the syntactical expression of f: arithmetic operations are evaluated using interval rules as shown above, and any other single operator – e.g., power – has its own interval extension; see (Moore and Kearfott, 2009) for more details. Other extensions include Trombettoni et Al.'s occurrence grouping approach (Araya, Neveu, and Trombettoni, 2012). In this work, we use interval computations provided in RealPaver (Granvilliers and Benhamou, 2006) and the natural extensions this software provides.

2.3.2. *How to Solve Nonlinear Equations with Intervals?*
The premise of our approach is that we will replace several real-valued computational processes by one interval-based computational process by abstracting one real-valued parameter into an interval parameter. Each process (real-valued or interval) consists in solving a (most likely) nonlinear system of equations. In this subsection, we give the reader an overview of the way we proceed to solve a nonlinear system of equations that involves intervals.

We choose to solve nonlinear equations using interval constraint solving techniques. Constraint solving techniques allow to solve systems of constraints. Generally speaking, a constraint describes a relationship that its variables need to satisfy. A solution of a constraint is an assignment of values to the variables of the given constraint such that the relationship is satisfied.

In our case, each of our nonlinear equations $f_i(x_1, \ldots, x_n) = 0$ is a constraint: it establishes a relationship that the values of the variables should satisfy, in this case so that $f_i(x_1, \ldots, x_n)$ be equal to 0. Our system of nonlinear equations is therefore a system of constraints and our goal is to find values of the variables of this system that are such that: $\forall i,\ f_i(x_1, \ldots, x_n) = 0$.

Constraint solving techniques allow us to identify such values of the parameters that satisfy the constraints. Interval constraint solving techniques (Mackworth, 1977; Jaffar and Maher, 1994) produce a solution set (set of the solutions of the constraint system) that is interval in nature (this is what you will see in the graphs plotting our experimental results in Section (5)): it is a set of multi-dimensional intervals (or boxes whose dimension is n, the number of variables) that is guaranteed to contain all the solutions of the constraint problem (in our case, of the nonlinear system of equations).

The guarantee of completeness provided by interval constraint solving techniques comes from the underlying solving mode: a branch-and-bound (Kearfott, 2007) (or branch-and-prune for faster convergence (Caroa, Chablata, and Goldsztejnb, 2014)) approach that uses the whole search space as a starting point and successively assess the likeliness of finding solutions in the given domain (via interval computations) and possibly (if Branch and Prune) reduce it, and discard domains that are guaranteed not to contain any solution. *Note: while Branch-and-Bound algorithms only assess domains for likeliness of containing a solution (it is a keep or discard approach), Branch-and-Prune algorithms first use the constraints to reduce the domains to consistent domains (using appropriate*

consistency techniques based on interval computations) and the outcome (empty domain or not, small enough or not to be called a solution) decides whether to continue exploring the domain or not.

For instance, if on a given domain $D \subset \mathbb{R}$, any of the f_i is such that $0 \notin F_i(D)$, where F_i is an interval extension of f_i, then we can conclude that there is no zero of our system of equations in D and discard it altogether. In Table II, we outline the generic Branch-and-Bound approach, which is the underlying principle of search in interval constraint solving techniques, and allows to guarantee completeness of the search.

Table II. Generic Branch-and-Bound Algorithm.

Input: System of constraints $C = \{c_1, \ldots, c_k\}$, a search space D_0.
Output: A set *Sol* of interval solutions (boxes of size n, the number of variables)
Set *Sol* to empty
If $\forall i$, $0 \in F_i(D_0)$ then:
Store D_0 in some storage S[1]
While (S is not empty) do:
Take D out of S
If ($\forall i$, $0 \in F_i(D)$) then:
If (D is still too large[2]) then:
Split[3] D in D_1 and D_2
Store D_1 and D_2 in S
Else:
Store D in *Sol*
Return *Sol*

Using interval computations carries a lot of advantages, one of which being that the search can be guaranteed to be complete and that since intervals are used (interval computations to assess whether a domain is a viable option of not), uncertainty can easily be added and seamlessly handled. This however comes at a cost: interval solving processes are usually more computationally taxing that regular real-valued ones. Nevertheless, in what follows we will show that, when comparing our interval-based approach to real-valued processes that have to be repeated countless times, then the extra cost of interval computations is counterbalanced and our approach more computationally effective (as shown in Section 5).

[1] S could be a queue, a stack, a priority queue, etc.

[2] The size of the domain is a stopping criterion: once we have achieved a given precision ϵ, there is no need to keep exploring the domain, it is considered an interval solution.

[3] Literature is full of approaches to splitting, including splitting in more than two pieces. Here we assume that we split in two. The type of splitting (where? on which domain? or even in how many pieces?) does not affect the generic algorithm.

3. Proper Orthogonal Decomposition

In this section, we study the statistical procedure of Principal Component Analysis (PCA), which uses orthogonal transformation to convert a set of observations of possibly correlated Random Variables into a set of linearly uncorrelated ones with the largest possible variance, named principal components. The number of principal components is less than or equal to the number of the original random variables.

Using the same procedure as in (PCA), it is possible to find a set of linearly independent vectors from a set of linearly dependent ones, whose spanned space is practically the same. This procedure is named Proper Orthogonal Decomposition (POD), which we also describe here.

3.1. Principal Component Analysis

When information from a data sample is collected, usually we take the maximum number of variables. However, if we take too many variables from a data sample, for instance 20 variables, we must consider $\binom{20}{2} = 190$ possible correlation coefficients. If you have 40 variables that number is increased to 780. Obviously, in this case it is difficult to visualize relationships between variables. Another problem that arises is the strong correlation that often occurs between variables: if we take too many variables (which generally happens when much is not known about data, or we are only interested in exploratory tests), it is normal that they are related or they measure the same thing under different viewpoints. For example, in medical studies, blood pressure at the heart's outlet and out of the lungs are strongly related.

Therefore, it is necessary to reduce the number of variables. It is important to highlight that the concept of major information is related to the greater variability of the data or variance. The greater the variability (variance) of the data, the more information this data has.

Studying the relationships that exist between p correlated variables (which commonly measure information) transforms the original set of variables in another new set of uncorrelated variables together (that has no repetition or redundancy on the information) called a set of principal components.

3.2. Principal Components

Let us consider a number of variables $X = (x_1, x_2, \ldots, x_n)$ describing a group of objects or individuals and to calculate, from them, a new set of variables $(y_1, y_2, \ldots, y_n)$ uncorrelated with each other, whose variances will decrease gradually.

Each y_j (where $j = 1, \ldots, n$) is a linear combination of the original variables $x_1, x_2, \ldots, x_n$, i.e

$$\begin{aligned} y_j &= v_{1j}x_1 + v_{2j}x_2 + \ldots + v_{pj}x_n \\ &= Xv_j, \end{aligned}$$

where $v_j^{\mathrm{T}} = (v_{1j}, v_{2j}, \ldots, v_{pj})$ is a constant vector.

To keep the orthogonality of the transformation, we impose $||v_j|| = 1$.

The first component v_1 is calculated so y_1 has the greatest variance subject to the constraint that $||v_1|| = 1$. The second principal component v_2 is calculated so that the variables y_1 and y_2 are uncorrelated. Similarly are chosen $y_1, y_2, \ldots, y_p$, uncorrelated with each other.

The full principal components decomposition of X can therefore be given as

$$Y = XV,$$

where V is a $p \times p$ matrix whose columns are the eigenvectors of $X^T X$.

The principal component decomposition of X can be expressed in terms of singular value decomposition of X. Given

$$X = U\Sigma V^T,$$

then we have

$$\begin{aligned} Y &= XV \\ &= U\Sigma V^T V \\ &= U\Sigma. \end{aligned}$$

In practice, we initiate computations with p variables and we are left with a number of much smaller components that collect a large percentage of the variability. For instance, we take r variables, where r is the minimum positive integer such that:

$$\frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{p} \sigma_i} > tol.$$

where tol is an approximation of 1 by defect.

3.3. Proper Orthogonal Decomposition Method

Consider a parameterized static computational model described by large-scale linear system of discrete equations

$$A(\lambda)x = b. \tag{19}$$

Here we can see Eq.(19) as an input-output system, where λ is the input and the solution, $x(\lambda) \in \mathbb{R}^n$, is the output.

The idea behind this method is that, given a certain input, the solution $x(\lambda)$ of a system contains the behavior of the system (Schilders and Vorst, 2008). Therefore, the set of outputs serves as a starting-point for POD. The outputs are called *snapshots* and these must be given or be computed first.

Assume the set of snapshots S and the solution $x(\lambda^*)$ of Eq.(19) for a particular λ^* is in the subspace spanned by S. We assume that the columns of S are highly correlated, so we can apply principal components analysis (PCA) to obtain an uncorrelated number of columns, see 3.1, and thus to reduce the size of linear system of equations.

Consider the SVD of S

$$S = U\Sigma V^T \tag{20}$$

and

$$T = V\Sigma^{-1}U^T. \tag{21}$$

Define

$$T_1 = \sum_{i=1}^{k} v_i \sigma_i^{-1} u_i^T; \quad T_2 = \sum_{i=k+1}^{n} v_i \sigma_i^{-1} u_i^T,$$
$$S_1 = \sum_{i=1}^{k} u_i \sigma_i v_i^T; \qquad S_2 = \sum_{i=k+1}^{n} u_i \sigma_i v_i^T. \tag{22}$$

Conditions given by Eq.(22) are a particular case of conditions given in Eq.(7). We conclude that we get a good approximation of Eq.(19) if $S_2\tilde{x}$ is sufficiently small ($\tilde{x} = T_2(x)$) or equivalently if $\sigma_i \approx 0$ for $k+1 \leq i \leq n$.

To obtain a basis of W we have the algorithm in Table IV.

Table III. Computing a Proper Orthogonal Decomposition Basis.

In: Parameter λ's and *input-output system*
Out: Base of the subspace W
Solve the full-order model to several λ's.
For each λ, take one or more **snapshots**, which is the solution of Eq.(19) for some values of t, and store such snapshots in a matrix S. Compute the SVD of S: $[W, \Sigma, V] = svd(S)$.
Find k such that $\sigma = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{n} \sigma_i} > 0.99$.
Consider only the k first columns and redefine $W = W(:, [1:k])$.

Several problems have been solved by using this method (Willcox, 2002). As it has been said before, the POD is based on Principal Components Analysis. The reader who wants to read a little more about this can find a good source of information in (Jolliffe, 1986).

4. Interval Proper Orthogonal Decomposition (I-POD)

In this section, we present our Interval POD approach to solving large nonlinear systems of equations in a reduced subspace. Let us first recall once again the problem that we are solving.

Given a parametric system of equations (also known as the Full Order Model):

$$R(x, \lambda) = 0, \quad \lambda \in \mathbf{I} \tag{23}$$

where R can be either linear or nonlinear function $R : \mathbb{R}^n \to \mathbb{R}^n$, that might arise from the discretization of a set of partial differential equations and $\mathbf{I}$ is a fixed interval. The idea behind POD is to solve Eq.(23) for a sequence of values $\lambda_i \in \mathbf{I}$, i.e.,

$$R(x, \lambda_1) \; = \; 0,$$

$$\begin{aligned} R(x, \lambda_2) &= 0, \\ \vdots \quad &\quad \vdots \\ R(x, \lambda_n) &= 0, \end{aligned} \tag{24}$$

where $\lambda_i \in \mathbf{I}$, for $i = 1, 2, \ldots, n$. The main idea of this method is based on the high correlation between solutions for such values λ_i, so PCA techniques can be applied to obtain a smaller number of columns uncorrelated with the greatest of accumulated variance.

In this work, we propose an interval version of POD. The original idea behind this new Interval POD is that we aim to reduce the amount of work in solving the Full Order Model for many different values of the input parameters (λ). Instead we suggest and experimented solving the Full Order Model once on the entire interval containing all desirable values of λ.

This slight change in concept (many processes solving for many different values of λ vs. one process solving for an entire interval instead) has consequences in our ability to solve the Full Order Model. Now that an interval is part of the problem we are bound to use interval-computation-based solving techniques and we found interval constraint solving techniques to be very practical to do so.

More specifically, we are now solving:

$$R(x, \mathbf{I}) = 0, \tag{25}$$

which is a nonlinear system of equations with explicit uncertainty in the shape of an interval.

We called this variation of POD the **Interval Proper Orthogonal Decomposition (I-POD)** method.

5. Numerical Results

In this section, we describe and report on preliminary experiments of our IPOD method on two well-known problems: the Burgers' equation and the Transport equation. In each of these experiment, we aim to assess the ability of IPOD to generate snapshots that yield a reduced basis of quality enough that the solution of the reduced-order model yields a very small error (w.r.t. FOM solution) in comparison to what a similar process using POD achieves. Our experiments were conducted using MATLAB R2012b (8.0.0.783) on a laptop with 1.7 GHz intel core i7 and 8GB of RAM.

5.1. Burgers' Equation

Consider the Burgers' equation:

$$\frac{\partial U(x,t)}{\partial t} + \frac{\partial f(U(x,t))}{\partial x} = g(x), \tag{26}$$

where U is the unknown conserved quantity (mass, density, heat etc.), $f(U) = 0.5U^2$ and in this example, $g(x) = 0.02 \exp(0.02x)$. The initial and boundary conditions used with the above PDE

are: $U(x;0) \equiv 1$; $U(0;t) = 4$, for all $x \in [0;100]$, and $t > 0$.

Below, in Tables IV and V, we describe the procedure to obtain the snapshots and the reduced basis in the POD method for the Burgers' equation. We will then compared it with **I-POD**.

Table IV. Computing a Proper Orthogonal Decomposition Basis.

Initialize an empty matrix where we will collect the snapshots: Snap = [], and an initial $\lambda = 3.5$.

`For i= 2:100`,

Solve:
$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} + \dfrac{\partial f(U(x,t))}{\partial x} = g(x), \\ g(x) = 0.02\exp(0.02x) \\ U(x;0) \equiv 1, \text{ for all } x \in [0;100], \\ U(0;t) = \lambda_i, \text{ for } t > 0. \end{cases} \tag{27}$$

Collect snapshots:

From $t_1, t_2, \ldots, t_n$, select a subsequence[4] $t_{i1}, t_{i2}, \ldots, t_{ip}$.

Add new columns to the snapshot matrix Snap = [Snap $U(x,t_{i1})$ $U(x,t_{i2})$ $\ldots U(x,t_{ip})$].

Update λ: $\lambda_i = \lambda_{i-1} + 0.01$

Apply the principal component analysis, (SVD). Snap $= W\Sigma V^T$.

Select from W the principal components with the greatest accumulated variance:

$\sigma = 0$.

`for k=1:n` compute:

$\sigma = \sigma + \frac{\sigma_k}{\sum_{j=1}^{n} \sigma_j}$

If $\sigma > Tol$, $0 < Tol < 1$, break.

Select the first k columns of W and redefine it. $W = W(:,[1,2,\ldots,k])$.

The new W will be the reduced basis to apply the POD method.

We applied both previous procedures, POD and IPOD, to solve Eq. (26) and we obtained the results reported in the Table VI. We observe that there is no significant difference between the traditional method (using POD) and the method we propose (using IPOD) w.r.t. (1) the dimension of the subspace, (2) the time it takes to solve the problem once we have identified the reduced basis, and (3) the relative error compared with the FOM solution. The major two advantages of our proposed method are:

- the computational time it requires to obtain the snapshots: Our approach requires 68.52% less time than the original one and the quality of the snapshots our method generates is comparable to that generated by POD as observed in the relative error; and
- the ability to handle uncertainty: the interval that contains λ, handled at once by IPOD, is similar to uncertainty and is handled without problems. Further experiments will aim to

[4] In the experiment done in this work, $p = 5$ and the subsequence was a uniform random selection.

Table V. Computing a Proper Orthogonal Decomposition Basis Using IPOD.

Initialize a empty matrix where to collect the snapshots: Snap = [], and an initial $\lambda = 3.5$.

Solve:
$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} + \dfrac{\partial f(U(x,t))}{\partial x} = g(x), \\ g(x) = 0.02\exp(0.02x) \\ U(x;0) \equiv 1, \text{ for all } x \in [0;100], \\ U(0;t) = \mathbf{I}, \text{ for } t > 0. \end{cases} \tag{28}$$

The solution of Eq. (28) is an interval solution, i.e, for any $1 \leq x_0 \leq 100, 0 \leq t_0 \leq 50$, the value $U(x_0, t_0)$ is an interval. The infimum of such interval is defined $U_l(x_0, t_0)$ and $U_r(x_0, t_0)$ is the supremum. In that case, for all $1 \leq x \leq 100,\ 0 \leq t \leq 50$, $U(x,t) \in [U_l(x,t), U_r(x,t)]$, see Figure 3.

`For i=1:100`

Compute:

$U(x,t) = (U_r(x,t) - U_l(x,t))(\lambda - 3.5) + U_l(x,t)$

Collect snapshots:

From $t_1, t_2, \ldots, t_n$, select a subsequence[5] $t_{i1}, t_{i2}, \ldots, t_{ip}$.

Add new columns to the snapshot matrix Snap = [Snap $U(x,t_{i1})\ U(x,t_{i2})\ \ldots U(x,t_{ip})$].

Update λ: $\lambda_i = \lambda_{i-1} + 0.01$

Apply the principal component analysis, (SVD). Snap $= W\Sigma V^T$.

Select from W the principal components with the greatest accumulated variance:

$\sigma = 0$.

`for k=1:n` compute:

$\sigma = \sigma + \frac{\sigma_k}{\sum_{j=1}^{n} \sigma_j}$

If $\sigma > Tol$, $0 < Tol < 1$, break.

Select the first k columns of W and redefine it. $W = W(:, [1, 2, \ldots, k])$.

The new W will be the reduced basis to apply the POD method.

Table VI. Comparing POD and I-POD methods in solving a particular example of Burgers Equation.

Method	Tag 1	Tag 2	Tag 3	Tag 4
POD	300 secs	37	0.75 secs	$4.85E-4$
I-POD	94.45	36	0.75 secs	$5.76E-4$

Tag 1: Time computing the Reduced basis
Tag 2: Dimension of the Subspace
Tag 3: Time solving 26 using the obtained basis
Tag 4: $||u_{fom} - u_{rom}||/||u_{fom}||$

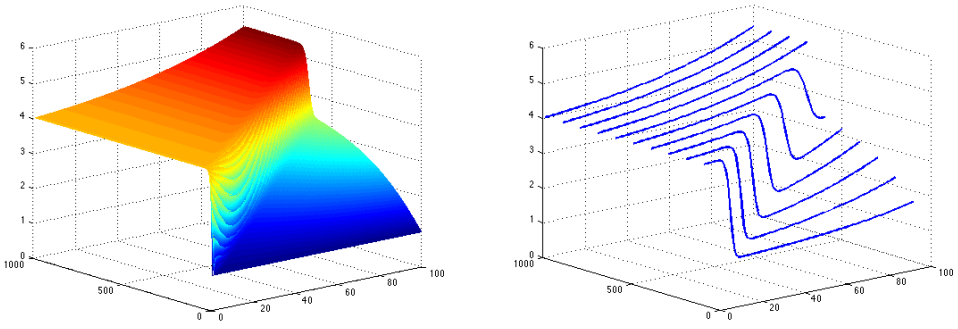

Figure 2. Solution of Eq.(27) for $\lambda = 4$, and some snapshots corresponding to this parameter.

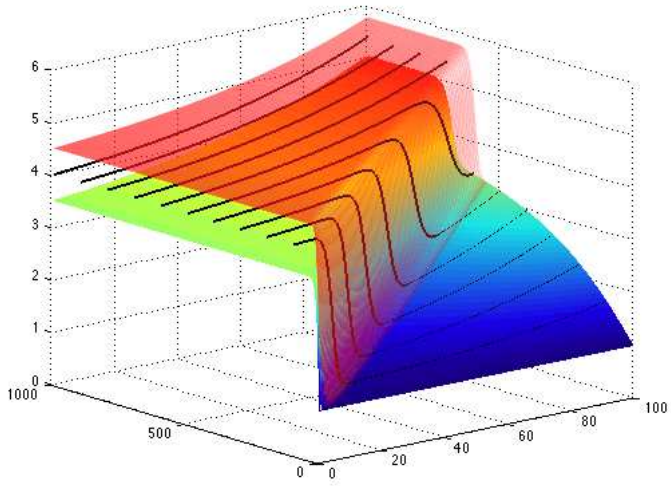

Figure 3. Infimum and supremum of the solution of Eq.(28), and some snapshots corresponding to $\lambda = 4$ are shown.

demonstrate that IPOD produces meaningful results also when there are other sources of uncertainty (beyond the interval for λ).

5.2. Transport Equation

The transport equation is a partial differential equation that models the concentration of a contaminant in the position x in at time t in a fluid that is flowing with velocity v in a thin straight tube whose ross section, denoted by A is constant. Such concentration will be denoted by $U(x,t)$. if the function $U(x,t)$ and its partial derivatives of order one are continuous functions of x and

[5] In the experiment done in this work, $p = 5$ and the subsequence was a uniform random selection.

t, and the fluid velocity v and the cross section of the tube, A, are constants, then the Transport Equation is reduced to:

$$\frac{\partial U}{\partial t} + v\frac{\partial U}{\partial x} = 0$$
$$(x,t) \in \Omega \tag{29}$$

Where Ω is a convex domain. In particular, we solve Eq.(29) with $U(x,t)$ subject to the following boundary and initial conditions:

$$U(0,t) = u(t) = -\sin(2\pi t) + \sin(\pi t) \tag{30}$$
$$U(x,0) = u(x) = \sin(2\pi x) + \sin(\pi x) \tag{31}$$

for all $t \in [0,1]$, and $x \in [0,1]$.

Using $v \in [0.5, 1.5]$ as the input parameter, we can proceed, similarly to how we did in the Burger Equation case, and compute, first, a basis using POD method, and later, using IPOD.

Comparative values are presented in Table VII:

Table VII. Comparing POD and I-POD methods in solving a particular example of Burgers Equation.

Method	Tag 1	Tag 2	Tag 3	Tag 4
POD	0.31 sec.	12	0.06 sec.	$7.97E-5$
I-POD	0.05 sec.	76	0.05 sec.	$1.81E-5$

Tag 1: Time computing the Reduced basis
Tag 2: Dimension of the Subspace
Tag 3: Time solving 29 using the obtained basis
Tag 4: $||u_{fom} - u_{rom}||/||u_{fom}||$

In this experiment, we observed that even when the dimension of the subspace is larger using the I-POD (76) than when using POD (12), the time needed to compute the reduced basis in I-POD is significantly less than the computing time needed using POD. Once, both basis are known, solving the reduced problem from POD or I-POD take the same time.

In Figure 4, we can observe the plot of the solution of the transport equation for time-steps 20, 40, 60, 80. In green and red are respectively the infimum and the supremum of the interval containing the solution.

6. Conclusions and Future Work

In this paper, we proposed and described a novel Model-Order Reduction approach that improves the well-known Proper Orthogonal Decomposition method (POD) by using Interval analysis and

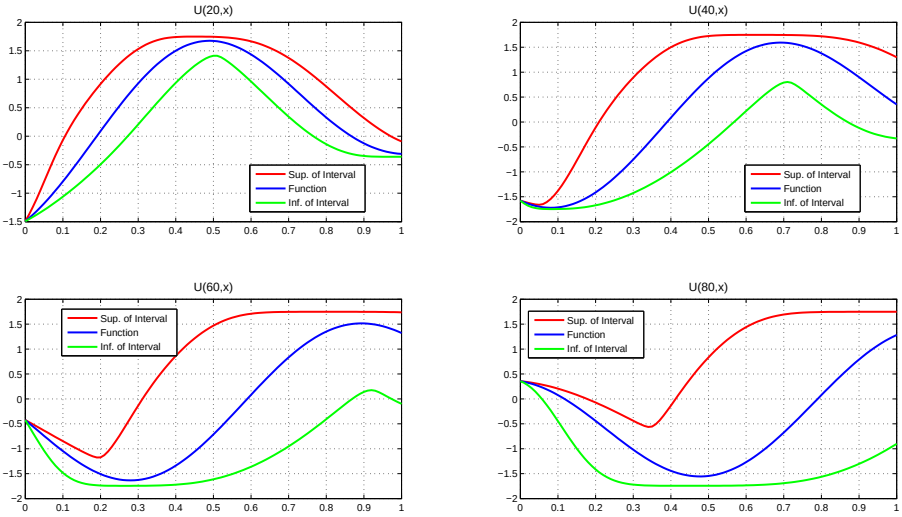

Figure 4. Solution of Eq.(29) for time-steps 20, 40, 60, 80, enclosed in the Interval solution.

Interval Constraint Solving Techniques. This new method called Interval Proper Orthogonal Decomposition (IPOD) was tested on two nonlinear partial differential equations problems: the Burgers' equation and the Transport equation. We observed and reported promising performance of IPOD, when compared to POD.

From this preliminary work, we draw the following research activities and directions. First, we do plan to challenge IPOD on problems whose solution is highly nonlinear, e.g., the Fitz-Hugh-Nagumo (FHN) problem. We also need to assess its relevance in handling and meaningfully solving problems with other sources of uncertainty. Finally, when having to handle uncertainty, achieving a relevant reduced basis is not all that needs to be modified from traditional approaches: once the space reduced, solving techniques (currently, namely, Newton-based methods) need to be extended to intervals. Although Interval Newton has been around for a long time (Moore and Kearfott, 2009; Hansen and Walster, 2004), Interval Newton on a reduced subspace is expected to present its set of challenges, including in particular, the likeliness of an empty solution set.

Acknowledgements

This work was supported by Stanford's Army High-Performance Computing Research Center funded by the army Research Lab, and by the National Science Foundation award #0953339.

References

Li, J. and Y. Chen. *Computational Partial Differential Equations Using MATLAB.* CRC Press, Las Vegas (NV), 2008.

Schilders, W. H and H. A. Vorst. *Model Order Reduction: Theory, Research Aspects and Applications.* Springer Science & Business Media, 2008.

Granvilliers, L. and F. Benhamou. *RealPaver: An Interval Solver using Constraint Satisfaction Techniques..* ACM TRANS. ON MATHEMATICAL SOFTWARE, 2006.

Kreinovich, V., G. Xian, M. Ceberio, at al. *Towards Combining Probabilistic and Interval Uncertainty in Engineering Calculations: Algorithms for Computing Statistics under Interval Uncertainty, and Their Computational Complexity.* Reliable Computing, 2006.

Moore, R. E, R. B. Kearfott and M. J. Cloud. *Introduction to Interval Analysis (1 edition)*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009.

Sharan, M. and A. Pradhan. *A Numerical Solution of Burgers Equation Based on Multigrid Method*, International Journal of Advancements in Electronics and Electrical Engineering – IJAEEE, 2, 2013.

White. J. *A Trajectory Piecewise-Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micromachined Devices* in International Conference Computer Aided Design, pages 252–257, 2013.

Kelly, C. T. *Solving Nonlinear Equations with Newton's Method,* no 1 in Fundamentals of Algorithms, SIAM, ISBN 0-89871-546-6, 2003

Kelly, C. T. *Reduction of Model Order Based on Proper Orthogonal Decomposition for Lithium-Ion Battery Simulations,* Journal of the Electrochemical Society, 156:A154–A161, 2009.

Marquez, A. and J. Espinosa. *Model Reduction Using Proper Orthogonal Decomposition and Predictive Control of Distributed Reactor System, Model Reduction Using Proper Orthogonal Decomposition and Predictive Control of Distributed Reactor System,* Journal of Control Science and Engineering, Journal of Control Science and Engineering, p. e763165, 2013.

Rathinam, M. and A. Petzold. *A New Look at Proper Orthogonal Decomposition,* SIAM Journal on Numerical Analysis, 41, pages 1893–1925, 2003.

Björck, A. *Numerical Methods for Least Squares Problems*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, January 1996.

Hoffman, K. and R. Kunze. *Linear Algebra*, Prentice-Hall, 1971.

Cotlar, M. and R. Cignoli. *Introduction to Functional Analysis*, North Holland Publishing Company and Elsevier Publishing Company, first edition ed., April 1974.

Willcox, K and J. Peraire. *Balanced model reduction via the proper orthogonal decomposition.* AIAA journal, 40(11):2323–2330, 2002.

Jolliffe, I. T. *Principal Component Analysis* Springer-Verlag. p. 487. doi:10.1007/b98835, ISBN 978-0-387-95442-4, 1986.

Hansen, E. and W. Walster. *Global Optimization using Interval Analysis,* Second Edition, Revised and Expanded. Marcel Dekker, Inc. New York, 2004.

Araya, I., B. Neveu and G. Trombettoni. *An interval extension based on occurrence grouping* Computing, 94(2):173-188, March 2012.

Hailer, A. *Verification of Branch and Bound Algorithms Applied to Water Distribution Network Design* Logos Verlag, Berlin, Dissertation, Technische Universitt Hamburg, Harburg, 2006.

Kearfott, R. B. *Verified branch and bound for singular linear and nonlinear programs: An epsilon-inflation process,* Available from the author, April 2007.

Jaffar, J. and M. Maher. *Constraint Logic Programming: a Survey* Journal of Logic Programming, 19/20:503–581, 1994.

Mackworth, A. K. *Consistency in Networks of Relations* Artificial Intelligence, 8(1):99–118, 1977.

Caroa, S., D. Chablata, A. Goldsztejnb, D. Ishiic and C. Jermannd. *A branch and prune algorithm for the computation of generalized aspects of parallel robots* Artificial Intelligence, 211:34–50, June 2014.

Hentenryck, V. P., D. McAllester and D. Kapur. *Solving Polynomial Systems Using a Branch and Prune Approach* SIAM J. Numer. Anal., 34(2):797–827, 1997.