A Reduced-Order Model for the Solution of Diffusion Equations

Jaron P. Senecal and Wei Ji

Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute
110 8th St, Troy, NY 12180, USA. Emails: senecj@rpi.edu; jiw2@rpi.edu

INTRODUCTION

Despite the continually increasing capacity of computers, there are always problems that are too large to handle. Highly refined meshes quickly overcome the power of modern computing platforms due to what is known as the "Curse of Dimensionality." Standard discretization schemes scale exponentially with the number of dimensions. For highly refined applications, it may be unacceptable for a problem's degrees of freedom (DOFs) to scale according to a power of the number of elements per dimension, e.g. the third power for 3D.

One method of reducing the size of the problem, also known as model reduction, is Proper Generalized Decomposition (PGD). Its primary tenet is to express the solution with a separated representation (using a 2D solution as an example)

$$\phi(x,y) \approx \phi_N(x,y) = \sum_{i=0}^{N} F_i(x) \times G_i(y)\,. \tag{1}$$

This representation is also sometimes referred to as "finite sums decomposition" or "the tensor product approximation" [1]. The magnitude of N required to achieve an accurate solution depends on the suitability of this approximation to a given problem. Generally, satisfactory accuracy can be achieved with N on the order of tens, but a highly non-separable problem may require a higher-order representation (larger N). The separated representation allows one to apply separation of variables to the PDE at hand, which allows for the problem to be solved with coupled 1D solvers. Theoretically, then, the problem size will scale linearly with the number of elements per dimension.

Only recently has PGD been introduced for nuclear reactor analysis. PGD has been demonstrated as a 1D time-dependent diffusion solver [2]. It has also been applied to solve a 2D eigenvalue problem [3]. In this work we investigate the behavior of the PGD method for a 2D fixed source diffusion problem over a range of scattering ratios. Also, the linear scaling of PGD is demonstrated with a parametric examination of mesh size. The long-term goal of the work is to integrate PGD into a reactor multiphysics analysis framework and accelerate the overall simulations [4].

PGD METHOD OVERVIEW

The PGD method builds up a Reduced-Order Basis (ROB) on the fly. Each iteration starts by solving for and adding a new pair of basis functions (F_i, G_i) to Equation 1. This process is known as the "Enrichment Step." If the solution of a multi-dimensional problem can be represented by a separated form, the equation can be transformed into several coupled 1D equations—one for each dimension. The coupled one-dimensional equations are solved iteratively with Fixed-Point/Picard Iteration.

After a new basis pair has been found by the Enrichment Step, the "Projection Step" begins. First, the new basis functions are normalized, denoted by $\widehat{F_i}$ and $\widehat{G_i}$. Then the solution is projected onto the current Reduced-Order Basis, making the error orthogonal to the current basis. In other words, the ROB function pairs are optimally weighted to achieve a better solution of the form

$$\phi_N(x,y) = \sum_{i=0}^{N} \alpha_i \widehat{F_i}(x) \times \widehat{G_i}(y)\,. \tag{2}$$

Finding a solution in the ROB is computationally inexpensive because it only requires inverting an $(N+1)\times(N+1)$ matrix, where N is found in practice to be at most a few tens [1].

Finally, the current solution is checked against a termination criterion. We selected a termination criterion based on a small change in the solution, estimated by $\alpha_i \left\|\widehat{F_i}\right\|_\infty \left\|\widehat{G_i}\right\|_\infty$. This quantity is quick and easy to calculate, and our results suggest that it is a reasonable error estimate that is suitable for use as a termination criterion.

PGD IMPLEMENTATION

The PGD method is implemented with a code-and-wrapper scheme. The wrapper is written in Python 3 and it controls the PGD-specific logic and computations. The computationally intensive Enrichment Step is handled by a finite element application built in MOOSE [5]. The MOOSE framework takes care of the finite element machinery and also provides access to a variety of state-of-the-art tools and options that can be used to optimize performance. Figure 1 depicts the layout of the code.

We start with the one-group, fixed-source diffusion equation

$$\nabla \cdot D(x,y)\nabla\phi(x,y) + \Sigma_a(x,y)\phi(x,y) = q(x,y) \tag{3}$$

Multiplying by an arbitrary test function, $\varphi^*(x,y)$, and integrating over the domain, the following weak form is obtained.

$$\iint D(x,y)\nabla\phi(x,y)\cdot\nabla\varphi^*(x,y)\,\mathrm{d}x\,\mathrm{d}y + \iint \Sigma_a(x,y)\phi(x,y)\varphi^*(x,y)\,\mathrm{d}x\,\mathrm{d}y = \iint q(x,y)\varphi^*(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{4}$$

The next step of the PGD methodology requires the variables and any spatially varying coefficients to be expressed in a separated form so that the equation can be separated into one-dimensional problems. Spatially varying coefficients, e.g.

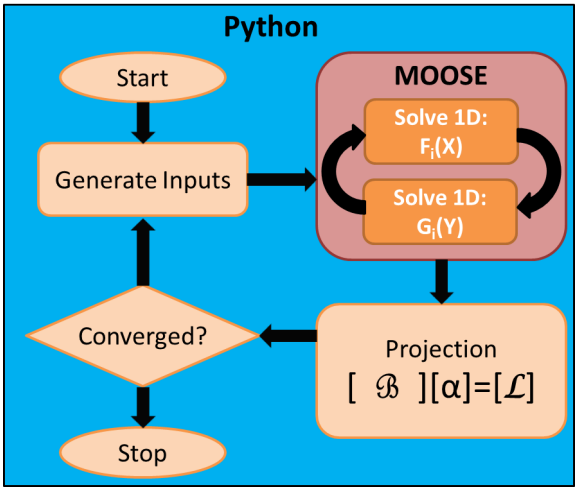

Fig. 1. Schematic diagram of the PGD solver. The Enrichment Step can be performed by an existing code, MOOSE in this case, while everything else is handled by the wrapper code.

$q(x, y)$, can be decomposed by applying Singular Value Decomposition (SVD) to the discrete representation of the coefficients. The following definitions are needed in order to separate Equation 4

$$\phi_i(x,y) = \sum_{j=0}^{i-1} F_j(x) \times G_j(y) + F_i(x) \times G_i(y) \quad (5)$$

$$\varphi^*(x,y) = F^*(x)G_i(y) + F_i(x)G^*(y)\,. \quad (6)$$

Note that at this point, $F_j(x)$ and $G_j(y)$ for $j < i$ are known as the result of previous iterations, and $F^*(x)$ and $G^*(y)$ are arbitrary functions. Equation 4 must hold for any value of either $F^*(x)$ or $G^*(y)$, so it can be separated into two equations. The X solver finds $F_i(x)$ by solving the equation with the $F^*(x)$ terms, holding $G_i(y)$ fixed. Similarly the Y solver finds $G_i(y)$ by taking $F_i(x)$ to be fixed. We arrive at two non-linearly coupled 1D equations, note that the integrals in parentheses are calculated from known or guessed values.

X solver:

$$\sum_{j=0}^{i}\sum_{k=1}^{N_D}\left(\int D_k(y)\frac{\mathrm{d}G_j}{\mathrm{d}y}\frac{\mathrm{d}G_i}{\mathrm{d}y}\,\mathrm{d}y\right)\int D_k(x)F_jF^*\,\mathrm{d}x$$
$$+\sum_{j=0}^{i}\sum_{k=1}^{N_D}\left(\int D_k(y)G_jG_i\,\mathrm{d}y\right)\int D_k(x)\frac{\mathrm{d}F_j}{\mathrm{d}x}\frac{\mathrm{d}F^*}{\mathrm{d}x}\,\mathrm{d}x$$
$$+\sum_{j=0}^{i}\sum_{k=1}^{N_\Sigma}\left(\int \Sigma_{ak}(y)G_jG_i\,\mathrm{d}y\right)\int \Sigma_{ak}(x)F_jF^*\,\mathrm{d}x$$
$$=\sum_{k=1}^{N_q}\left(\int q_k(y)G_i\,\mathrm{d}y\right)\int q_k(x)F^*\,\mathrm{d}x \quad (7)$$

Y solver:

$$\sum_{j=0}^{i}\sum_{k=1}^{N_D}\left(\int D_k(x)\frac{\mathrm{d}F_j}{\mathrm{d}x}\frac{\mathrm{d}F_i}{\mathrm{d}x}\,\mathrm{d}x\right)\int D_k(y)G_iG^*\,\mathrm{d}y$$
$$+\sum_{j=0}^{i}\sum_{k=1}^{N_D}\left(\int D_k(x)F_jF_i\,\mathrm{d}x\right)\int D_k(y)\frac{\mathrm{d}G_i}{\mathrm{d}y}\frac{\mathrm{d}G^*}{\mathrm{d}y}\,\mathrm{d}y$$
$$+\sum_{j=0}^{i}\sum_{k=1}^{N_\Sigma}\left(\int \Sigma_{ak}(x)F_jF_i\,\mathrm{d}x\right)\int \Sigma_{ak}(y)G_iG^*\,\mathrm{d}y$$
$$=\sum_{k=1}^{N_q}\left(\int q_k(x)F_i\,\mathrm{d}x\right)\int q_k(y)G^*\,\mathrm{d}y \quad (8)$$

The left-hand side of these equations we symbolize by the $\mathbf{B}(\cdot,\cdot)$ bilinear operator and the right-hand side we symbolize by the $\mathbf{L}(\cdot)$ linear operator. With this notation, Equations 7 and 8 are respectively written as

$$\mathbf{B}\left(\phi_{i-1} + F_i \times G_i,\, F^* \times G_i\right) = \mathbf{L}\left(F^* \times G_i\right) \quad (9)$$

$$\mathbf{B}\left(\phi_{i-1} + F_i \times G_i,\, F_i \times G^*\right) = \mathbf{L}\left(F_i \times G^*\right)\,. \quad (10)$$

The Projection Step begins by normalizing the newest basis function pair. Equation 2 is then inserted into Equation 4 with $\varphi^* = \sum_{j=0}^{i} \alpha^* \widehat{F}_j \times \widehat{G}_j$. This results in an $(i+1)\times(i+1)$ matrix equation that can be solved to obtain the α coefficients

$$\mathbf{B}\left(\sum_{j=0}^{i}\alpha_j^*\widehat{F}_j \times \widehat{G}_j,\, \sum_{k=0}^{i}\alpha_k\widehat{F}_k \times \widehat{G}_k\right) = \mathbf{L}\left(\sum_{j=0}^{i}\alpha_j^*\widehat{F}_j \times \widehat{G}_j\right). \quad (11)$$

Note that at this point, all of the $\widehat{F}$'s and $\widehat{G}$'s are known. When the α's have been found, ϕ_i is known and convergence can be checked. Algorithm 1 summarizes how the PGD method is implemented.

Algorithm 1 Proper Generalized Decomposition Code in 2D.

```
Initial set-up, use SVD to decompose parameters
for i = 0, 1, 2, ... do
    Enrich the solution basis
        while ||R_F|| > picard_tol do
            Solve Equation 7 for F_i(x)
            Solve Equation 8 for G_i(y)
        end while
    Project solution onto reduced-order basis
        Normalize: F̂_i = F_i / ∫F_i^2 dx and Ĝ_i = G_i / ∫G_i^2 dy
        Solve B⃗α = L
        where B_jk = B(F̂_j × Ĝ_j, F̂_k × Ĝ_k)
        and L_j = L(F̂_j × Ĝ_j)
        φ_i = Σ_{k=0}^{i} α_k F̂_k × Ĝ_k
    Check Convergence
        if α_i ||F̂_i||_∞ ||Ĝ_i||_∞ < pgd_tol then
            break
        end if
end for
```

Table I lists some of the parameters used for the example problem. The problem has homogeneous cross sections and the source is constant inside of a central square region. The absorption cross section is varied from $\frac{1}{150}$ to 6.6 cm^{-1} in order to examine the performance of PGD as a function of scattering ratio, $c = \Sigma_s/\Sigma_t = 1 - \Sigma_a/\Sigma_t$. The mesh size is also varied from 50 to 1000 elements per direction in order to observe the scaling behavior of PGD in comparison to standard full-order FEM solvers.

TABLE I. Example problem parameters.

$x \in [0, 1]$
$y \in [0, 1]$
$D = 0.05$
$q = 1.0$ on $x, y \in [0.3, 0.7]$
$\phi = 0$ on boundaries

RESULTS

TABLE II. PGD solver performance with respect to scattering ratio.

c	PGD Time, s	Iterations	FEM Time, s
0.01	3.5	5	30.7
0.1	4.6	6	30.1
0.25	3.6	5	33.3
0.5	3.4	5	35.5
0.85	4.5	6	50.6
0.99	3.5	6	70.0
0.999	3.4	6	71.2

First, we examine the performance of the PGD method as a function of the scattering ratio, c. Table II lists the performance results for a variety of scattering ratios. The PGD performance is effectively constant for any scattering ratio. Whereas the standard 2D MOOSE FEM solver with default block diagonal preconditioning starts to run slower as the scattering ratio increases. The results are given for 200 elements per dimension and a 1E-4 absolute termination criterion, but with different values the trends are similar.

The mesh size was progressively refined in order to observe the actual scaling behavior of the PGD method. All tests were run without MPI or multi-threading. The results are displayed in Figure 2. As expected, the standard FEM solver scales quadratically and PGD scales linearly. The difference in scaling leads to a massive performance advantage for PGD on highly refined meshes. By 400 elements per dimension PGD is more than 55 times faster than FEM.

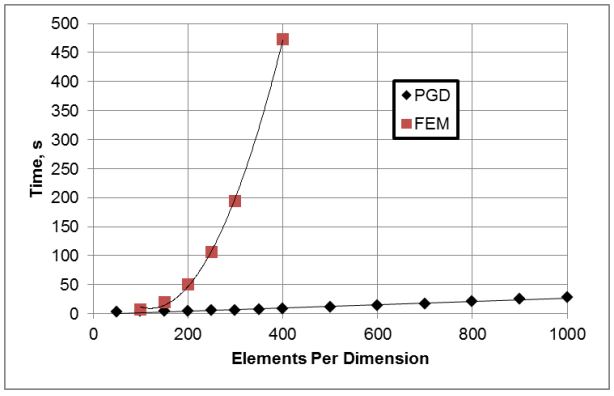

Fig. 2. Computer time required as a function of number of elements.

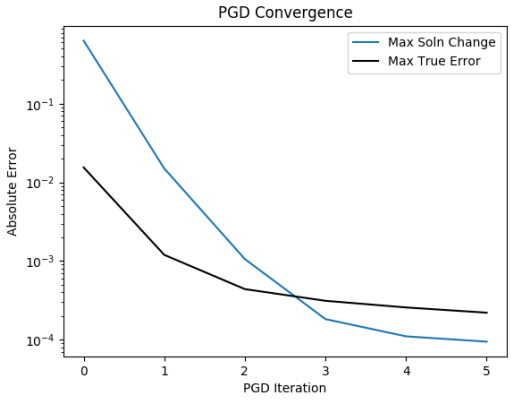

Fig. 3. The PGD error estimate plotted along with the true error for $c = 0.985$.

Figure 3 shows the convergence rate of the PGD algorithm. It shows that the maximum contribution of the most recent solution mode—namely $\alpha_i \left\|\widehat{F}_i\right\|_\infty \left\|\widehat{G}_i\right\|_\infty$—is a reasonably accurate estimator of the true maximum error. PGD solvers are susceptible to stagnation or underestimated errors if the selected top-level PGD tolerance is too small/tight relative to the tolerances at lower solver levels. Strategies for avoiding these problems will be explained in detail in future work.

The PGD solution of the problem with $c = 0.85$ and 200 elements per dimension is shown in Figure 4 and the normalized modes/basis functions are presented in Figure 5. The corresponding α's are listed in Table III. Figure 6 shows the error of the PGD solution with respect to the standard FEM solution. It only takes 6 iterations to reach a step size

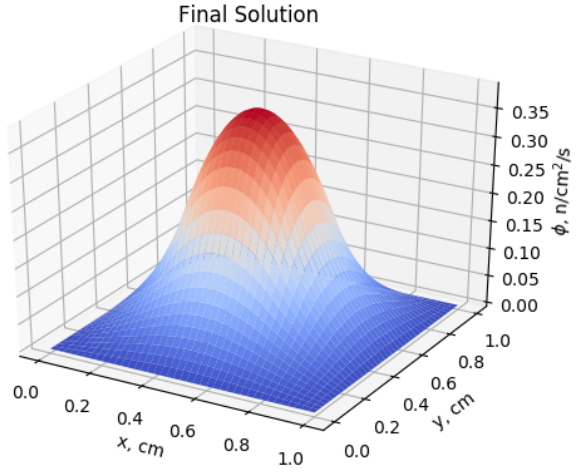

Fig. 4. Flux distribution solved with PGD.

less than 1E-4, thus the solution is only mildly non-separable. Table IV shows the performance data of the PGD solver. Note that although the wrapper is written in Python 3, which is not a compiled language, its performance is not a concern. This is because upwards of 90% of the time is spent in MOOSE for the Enrichment Step, which is consistent with other findings [6]. Thus future efforts will primarily focus on improving this part of the algorithm.

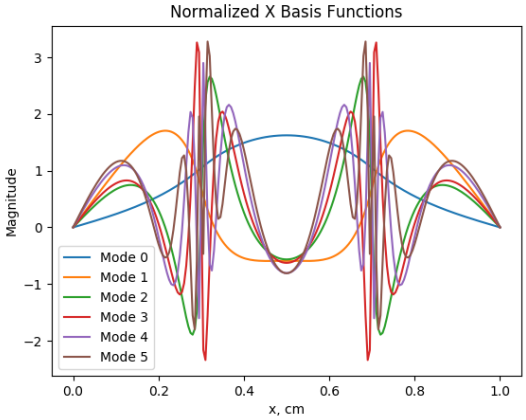

Fig. 5. The normalized x-dimension solution modes. This problem is symmetric, so the normalized y-dimension solution modes look the same.

TABLE III. Separation values, α_i. The decreasing values indicate that each mode contributes less to the solution than the last because the solution is approaching a fixed-point.

Mode	α
0	1.46E-1
1	3.72E-3
2	1.40E-4
3	4.48E-5
4	2.89E-5
5	1.38E-5

TABLE IV. PGD solver performance with $c = 0.85$.

Time	4.5s
MOOSE Time	94%
PGD Iterations	6
Maximum Error	-3.4E-4

CONCLUSION

PGD can be used for nuclear reactor analysis problems to avoid the "Curse of Dimensionality." This is important for highly refined multidimensional meshes where the number of degrees of freedom taxes computer resources. A mesh-refinement study demonstrated the linear scaling behavior of PGD. Also, the full range of scattering ratios can be handled effectively with PGD. The present work demonstrates that PGD can be implemented with existing tools by creating a wrapper code to control the execution of the PGD algorithm. Future work will incorporate a multi-group, three-dimensional PGD diffusion solver into a reactor multiphysics solver system.

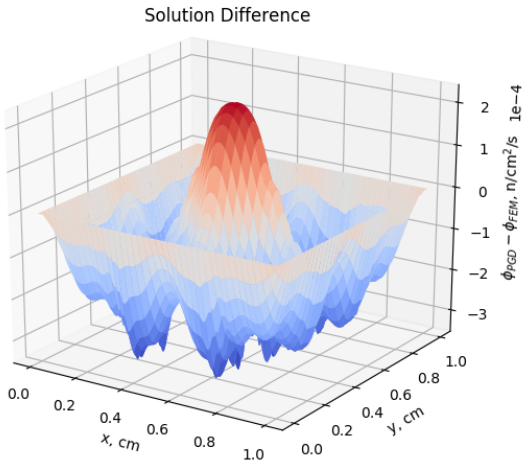

Fig. 6. Difference between the flux distribution solved with PGD and the standard 2D finite element solver.

ACKNOWLEDGMENT

The first author is supported by the Nuclear Regulatory Commission Fellowship Program under the Grant NRC-HQ-13-G-38-0035.

REFERENCES

1. F. CHINESTA, A. AMMAR, A. LEYGUE, and R. KEUNINGS, "An overview of the proper generalized decomposition with applications in computational rheology," *Journal of Non-Newtonian Fluid Mechanics*, **166**, *11*, 578–592 (2011), xVIth International Workshop on Numerical Methods for Non-Newtonian Flows.
2. A. J. ALBERTI and T. S. PALMER, "A-priori reduced order modeling for transient neutron diffusion," in "5th International Conference on Transport Theory," (2017), pp. 1–3, Monterey, CA, October 16–20.
3. S. GONZÁLEZ-PINTOR, D. GINESTAR, and G. VERDÚ, "Using proper generalized decomposition to compute the dominant mode of a nuclear reactor," *Mathematical and Computer Modelling*, **57**, *7*, 1807–1815 (2013).
4. J. P. SENECAL and W. JI, "Development of an efficient tightly coupled method for multiphysics reactor transient analysis," *Progress in Nuclear Energy*, **103**, 33–44 (2018).
5. D. R. GASTON, C. NEWMAN, G. HANSEN, and D. LEBRUN-GRANDIÉ, "MOOSE: a parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, **239**, 1768–1778 (2009).
6. D. GONZÁLEZ, A. AMMAR, F. CHINESTA, and E. CUETO, "Recent advances on the use of separated representations," *International Journal for Numerical Methods in Engineering*, **81**, *5*, 637–659 (2010).