DwarfElephant

May 7, 2020

1 Introduction

For this tutorial, we are using an application, namely the DwarfElephant package, to the Multiphysics Object-Orientated Simulation Environment (MOOSE) (Alger et al., 2019; Gaston et al., 2015), implementing the reduced basis (RB) method. The RB method is a model order reduction technique. Meaning the method aims at significantly reducing the dimensionality of the finite element (FE) problems. This is realized via a separation resulting in an offline and online stage. During the offline phase, all expensive pre-computations are performed allowing a fast execution of the online stage. Therefore, the online stage can be effectively used in, for instance, parameter studies, which require many forward simulations. For further details refer to Section 2.

MOOSE, primarily developed by the Idaho National Laboratory, offers a built-in parallelization. The framework builds upon the libMesh (Kirk, Peterson, Stogner, & Carey, 2006) and PETSc (Balay et al., 2017) libraries. The RB implementation mainly uses the rbOOmit package provided by libMesh. A built-in parallelization is especially advantageous since constructing parallelized problems for HPC infrastructures is often anything but trivial. For further details regarding the software packages, refer to Section 5.

2 The Reduced Basis Method

Note, that the following explanation about the RB method is very brief for further details, especially regarding the mathematical background, refer to (Prud'homme et al., 2002), (Quarteroni, Manzoni, & Negri, 2015), (Hesthaven, Rozza, Stamm, et al., 2016), and for more information about the method in a geoscientific context we refer to (Degen, Veroy, & Wellmann, 2019).

In general, the RB method is independent of the discretization scheme; however, since MOOSE is an FE solver, we explain the background of the RB approach by considering an FE problem that can be defined by a parametric coercive partial differential equation (PDE). We further assume throughout the entire tutorial that the PDE is affine parameter-dependent, i.e., it can be expressed as a sum of terms that are compromised of a parameter-dependent and -independent part. For a thermal conduction problem, the diffusive part of the equation is always the same, only the thermal conductivity is varying for, for instance, different rock types. That does not mean that both parts can be evaluated independently of each other. Hence, the thermal diffusion ($\nabla^2 T$) is the parameter-independent part of the PDE, whereas the thermal conductivity (λ) is the parameter-dependent part. In contrast to the Dirichlet boundary conditions, Neumann and Robin boundary conditions appear in the integral form and are generally dependent on our parameters. In our specific example, however, we consider a parameter-independent Neumann boundary condition only.

The principal idea of the RB approach is to take advantage of the decomposition by implementing an offline-online procedure. During the offline stage, all the expensive pre-computations are performed. The cost of these pre-computations is dependent on the high-dimensional FE solution because several solves of the full FE problem are required to build the low-order approximations, which will be explained in the following. Hence, it is advisable to perform the offline stage for large-scale models, where the solution of a single FE problem is already extremely time-consuming, on HPC infrastructures if available.

In the offline-stage, performed once, we solve for the RB functions. Hence, the computational cost depends on $\mathcal{N}$. In the online-stage, performed many times – for each new μ, the computational cost is of the dimension N. Considering $N \ll \mathcal{N}$, this means that the online stage has a much lower computational cost because of the largely reduced dimensionality of the problem (Prud'homme et al., 2002; Veroy & Patera, 2005; Quarteroni et al., 2015; Hesthaven et al., 2016). In case of our geothermal conduction example, we can perform the online-stage for any new thermal conductivity that falls within the range of the training set.

3 Problem Description

In the following we will consider a geothermal conduction problem as given in (1):

$$-\lambda \nabla^2 T = 0, \tag{1}$$

where λ is the thermal conductivity and T the temperature. In our case, we consider a steady-state problem without sources and sinks, thus ending up with an elliptic PDE.

For performance and convergence reasons we take only dimensionless parameters and variables into account. Therefore, we nondimensionalize the length by (2):

$$l^* = \frac{l}{l_{\text{ref}}}. \tag{2}$$

Throughout the entire tutorial the stars or asterisks denote the dimensionless quantities; the quantities without any asterisks refer to the dimensional quantities and the subscript "ref" denotes the reference parameter used for the nondimensionalization. Here, l_{ref} is 1000 m.
The thermal conductivity is nondimesionalized by applying (3):

$$\lambda^* = \frac{\lambda}{\lambda_{\text{ref}}}. \tag{3}$$

The reference thermal conductivity is, in our example, 1.05 W m^{-1} K^{-1} (i.e., the smallest thermal conductivity).

Finally, the nondimensionalisation of the temperature is performed with (4):

$$T^* = \frac{T - T_{\text{ref}}}{T_{\text{ref}}}, \tag{4}$$

where the reference temperature T_{ref} is taken as 10 °C. All model and simulation parameters are found in Table 1 and 2. The mesh is uniformly refined and has in total 1,658 nodes and 8,127 elements (Fig. 1).

4 Prerequisites and First Steps

Please, be aware that throughout the entire tutorial we are assuming that the MOOSE software package is correctly installed in the directory projects/moose. For further information on how to install MOOSE refer to (Alger et al., 2019).

As the first step download the DwarfElephant package from the Git Repository (`https://github.com/denise91/DwarfElephant`) into the folder projects/DwarfElephant. Afterwards compile the

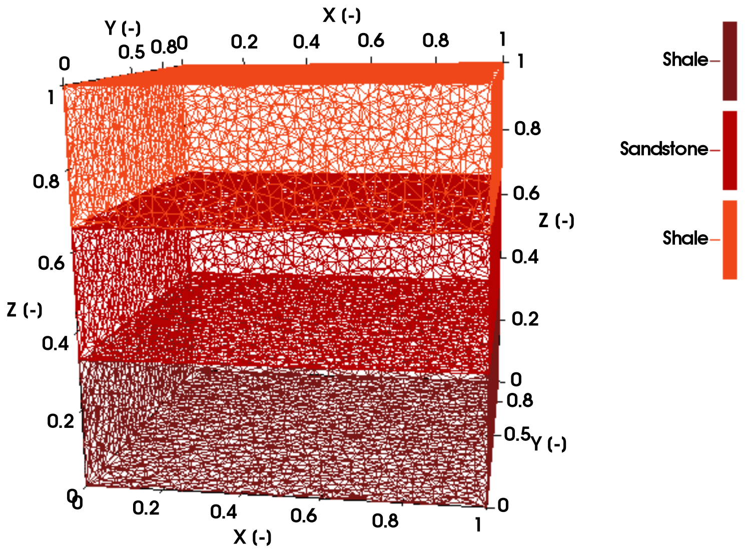

Figure 1: Mesh representation of a three-layer model. The model has a Dirichlet boundary condition at the top and a Neumann boundary condition at the bottom of the model. Note, that the mesh is dimensionless for performance reasons.

Table 1: Thermal and model properties for the synthetic model (Midttømme et al., 1998). Take into account that both the thermal conductivity and the length are dimensionless (λ_{ref} = 1.05 W m^{-1} K^{-1} and l_{ref} = 1,000 m).

unit	thermal conductivity [W m^{-1} K^{-1}] / thermal conductivity [-]	extension in		
		x-direction [-]	y-direction [-]	z-direction [-]
top layer (shale)	1.05 / 1.00	0.00 - 1.00	0.00 - 1.00	0.66 - 1.00
middle layer (sandstone)	2.50 / 2.38	0.00 - 1.00	0.00 - 1.00	0.33 - 0.66
bottom layer (shale)	1.05 / 1.0	0.00 - 1.00	0.00 - 1.00	0.00 - 0.33

Table 2: Simulation parameters for the reduced basis method of the synthetic model.

Simulation parameter	Value
Solver type	Newton
Linear solver tolerance	10^{-8}
Nonlinear solver tolerance	10^{-8}
Preconditioner type	hypre
Hypre type	boomerang
ksp_gmres_restart	101
Rel. training tolerance	10^{-5}
Parameter range (Layer 1 and 3)	0.5 to 5.0
Parameter range (Layer 2)	0.5 to 7.0
Number of parameters	3
Number of training samples	20
Maximum number of basis functions	20

package with the make command. Note, if you want to use the transient RB implementation you have to ensure that the SLEPc library (Hernandez, Roman, & Vidal, 2005) is enabled during the libMesh build.

After successfully setting up the DwarfElephant package we can start with our first RB example. Therefore create an empty input file with the name FirstRBProblem.i within the folder projects/DwarfElephant. The complete input file of this tutorial is also stored in the RBFirstProblem.i file.

5 Implementation

In this section, we will explain the implementation of the RB method within MOOSE. In order to use this code package, it is necessary that the problem can be described as a linear PDE.

5.1 Mesh, Variables, and Global Params

The DwarfElephant package has no additional mesh classes. Consequently, the usage of the Mesh and MeshModifier classes does not differ from MOOSE. Hence, for further explanation regarding the Mesh refer to the MOOSE examples and tutorials. Note that for efficiency reasons for both the solver and the method, we consider only dimensionless meshes.

As for the mesh classes, the here presented package does not offer any additional variable classes. Therefore, refer to the MOOSE examples and tutorials for further information about the usage of this feature.

Analog to the mesh and variable classes, the DwarfElephant package does not add any additional features to the GlobalParam class. For further information about the global parameters please refer to the MOOSE examples and tutorials.

5.2 Kernels

Kernels are in MOOSE responsible for implementing the PDEs, so they define the "physics" of a problem. In contrast to the standard finite element (FE) method, the RB method needs the PDE decomposed into a parameter-dependent and independent part. Additionally, each parameter that should be variable during the online stage requires its own contribution to the stiffness matrix.

Therefore, we implemented the DwarfElephantRBKernel. This Kernel class implements additionally an automatic splitting of the stiffness matrix into the mesh subdomains. However, the IDs of the individual stiffness matrices can be directly defined over the input file and hence any kind of decomposition is possible. Note that the DwarfElephantRBKernel does not implement any specific PDE. This Kernel has the same role as the Kernel provided by MOOSE. Hence, if you want to implement a PDE and want to use the RB method for the simulations you have to ensure that your own Kernel inherits from the DwarfElephantRBKernel instead of the Kernel.

5.3 BCs

A similar scenario as for the Kernels is encountered for the boundary conditions. If the boundaries have a parameter-dependent part an affine decomposition has to be performed. MOOSE offers two types of boundary conditions: nodal and integrated boundary conditions. Nodal boundary conditions are forced boundary conditions (i.e., Dirichlet boundary conditions), in contrast, integrated boundary conditions are boundary conditions that appear in the weak form of the PDE (i.e., Neumann boundary conditions). Be aware that due to the implementation libMesh is using, which destroys the affine decomposition for Dirichlet boundary conditions, non-zero Dirichlet boundary conditions problematic. Hence, it is important to nondimensionalize the PDE in such a way that the Dirichlet boundary condition is equal to zero. This leads also to an improved solver tolerance.

If you want to implement your own BC, it has to inherit from the DwarfElephantRBNodalBC or the DwarfElephantRBIntegratedBC class. Both of these boundary condition classes are responsible for performing the necessary RB related operations.

5.4 Materials

As already mentioned the RB method requires a decomposition into a parameter-dependent and independent part. Materials are in general parameter-dependent. Hence, in most of the cases, it is not useful to define a Material class for the RB method. The parameter-dependent parts of the PDE will be later defined in a separate block. Nonetheless, the package has a few Material classes, which are used for validation purposes within the FE method.

5.5 Problem, Executioner and UserObjects

For the RB method, the assembly procedure had to be modified from the FE version MOOSE provides. Additionally, we have to perform the solve directly over the libMesh class RBConstruction and are accordingly bypassing the NonlinearSystem class from MOOSE. Therefore, a new Problem (DwarfElephantRBProblem), System (DwarfElephantSystem) and Executioner (DwarfElephantRB-Executioner) class are implemented. Note that the Executioner of the RB Problem is the same for the steady-state and transient case because the time-stepping has to be included in the basis construction. However, this tutorial covers the steady-state case only.

To use the RB method in MOOSE two types of UserObjects are required. The UserObject DwarfElephantInitializeRBSystemSteadyState/DwarfElephantInitializeRBSystemTransient and Dwarf-ElephantOfflineOnlineStageSteadyState/DwarfElephantOfflineOnlineStageTransient. The first User-Object, which has to be executed on initial, is responsible for initializing the RB System for the steady state and transient case, respectively. Here, all necessary parameter for the RB construction have to be defined for more information about these parameters we refer to Section 6.7.

The second UserObject performs both the offline and online stage, where both can also be performed separately. Be aware that this UserObject has to be executed on timestep_end because for this UserObject all matrices and vectors must have been assembled. The necessary parameters are related to the storing options and the online stage, again we refer to Section 6.7 for further details. Keep in mind that the number of parameters is problem specific and need to be defined at the beginning of each new problem.

5.6 PostProcessors

For the PostProcessors, we offer two ways either on the full or reduced model. In order to do the postprocessing on the full model you can use the PostProcessors provided by MOOSE but you have to execute them on the time_step 'custom'. However, performing the postprocessing on the full instead of the reduced model is not advisable in terms of efficiency. Hence, we strongly advise using the second option for the postprocessing. Keep in mind that the current implementation only supports postprocessing steps on the reduced model that are affinely decomposable (meaning linearly dependent on the variable). If you want to perform the postprocessing on the reduced model you have to decompose the operation into a parameter-dependent and -independent part (analog to the Kernel and BC). Currently, the package contains a reduced order version of the MOOSE PostProcessors ElementalVariableValue, NodalVariableValue, and PointValue. Note, that for implementation reasons these PostProcessors (DwarfElephantRBElementalVariableValue, DwarfElephantRBNodalVariableValue, and DwarfElephantRBPointValue) are defined as UserObjects. Furthermore, keep in mind that in the RB literature the post processed values are referred to as the output of interest.

5.7 Outputs

All MOOSE Output classes can be used in the same way for the RB and FE Problem. We just offer an additional Output class DwarfElephantDakotaOutput to output PostProcessor values in a format that is compatible to the Dakota framework (Adams et al., 2017). In case you are using the RB please method set the boolean use_rb to true.

6 A Geothermal Conduction Problem

In this section, we will explain the usage of the RB method within MOOSE through the above described thermal conduction problem. Note, that although we are using in this tutorial a geophysical problem the DwarfElephant package is physics independent and is therefore not restricted to geophysical problems.

6.1 Mesh

For our example please copy the provided mesh file (RB_mesh_3layers.e) into the DwarfElephant folder and load the file with the following command into your simulation. The mesh capabilities are entirely from the MOOSE Framework. Hence, for further information, we refer to (Alger et al., 2019).

```
[Mesh]
    file = RB_mesh_3layers.e
[ ]
```

6.2 Variables

We define the temperature as our variable of interest since we are considering a geothermal conduction problem for our example simulation. Again, we did not add any functionality to this class and therefore we advise to visit (Alger et al., 2019) for further details.

```
[Variables]
    [./temperature]
    [../]
[ ]
```

6.3 GlobalParams

For this example we define the variable and the initial_rb_userobject as global parameters. Also the functionality of the GlobalParams is the one from MOOSE (check (Alger et al., 2019) for more information).

```
[GlobalParams]
    variable = temperature
    initial_rb_userobject = initializeRBSystem
[]
```

6.4 Kernels

As previously mentioned we have to separately implement the parameter-dependent and independent part. Therefore, we need to define only the diffusion part in the Kernel Block. If nothing else is defined, then the number of generated stiffness matrices corresponds to the number of subdomains. If that is not desired you have to set the boolean matrix_seperation_according_to_subdomains to false.

```
[Kernels]
    [./RBConduction]
        type = DwarfElephantRBDiffusion
    [../]
[]
```

6.5 BCs

The default settings to not consider any splitting in accordance with the mesh for the boundaries. In case you need to define several load vectors use the ID_Fq identifier. Per default, the ID is set to zero. In our example, we define a zero Dirichlet BC for the top and a Neumann BC for the bottom of the model. Both boundaries are stored in the load vector zero.

```
[BCs]
    [./top]
        type = DwarfElephantRBDirichletBC
        boundary = 2
        value = 0.00
    [../]

    [./bottom]
        type = DwarfElephantRBNeumannBC
        boundary = 1
        value = 3.71
    [../]
[]
```

6.6 Problem and Executioner

As mentioned above we need a separate Problem and Executioner class. Please define them in your input file as:

Table 3: Input parameters for the RB simulation.

Parameter	Definition
N_max	maximal number of basis functions
n_training_samples	number of samples in the training set
parameter_names	name of the parameters (for reusability please use mu_0 to mu_n)
parameter_min_values	lower bound of the parameter range
parameter_max_values	upper bound of the parameter range
store_basis_functions	defines whether the basis functions are stored in the offline data folder
online_mu	definition of the online parameters

```
[Problem]
    type = DwarfElephantRBProblem
[]

[Executioner]
    type = DwarfElephantRBExecutioner
    solve_type = 'Newton'
    l_tol = 1.e-8
    nl_rel_tol = 1.e-8}
[]
```

6.7 UserObjects

In this section, we define the parameter-dependent part of the problem. In order to run the simulation for this tutorial ensure that in the DwarfElephantRBClassesSteadyState.h file (projects\DwarfElephant\include\userobjects) the line under "FEProblemBase & fe_problem;" is equal to DwarfElephantRBP1T3EqualF1O1SteadyStateExpansion _rb_theta_expansion; To execute the offline-online procedure for the steady state and transient case the UserObjects DwarfElephantOfflineOnlineStageSteadyState and DwarfElephantOfflineOnlineStageTransient are needed. In this example we will use the steady state classes. The definition of the input parameters are found in Tab. 3 :

```
[UserObjects]
    [./initializeRBSystem]
        type = DwarfElephantInitializeRBSystemSteadyState
        execute_on = 'initial'
        N_max = 20
        n_training_samples = 100
        rel_training_tolerance = 1.e-5
        parameter_names = 'mu_0 mu_1 mu_2'
        parameter_min_values = '1.0 1.0 1.0'
        parameter_max_values = '5.15 7.15 5.15'
    [../]

    [./performRBSystem]
        type = DwarfElephantOfflineOnlineStageSteadyState
        store_basis_functions = true
        online_mu = '1.05 2.5 1.05'
        execute_on = 'timestep_end'
    [../]
[]
```

Note: Since the first UserObject has the task to initialize the RB System it has to be executed on the initial time step. The other UserObject has to be executed at the end of the timestep because the matrices and vectors have to be assembled first.

6.8 Outputs

For the Output class the standard MOOSE classes can be used.

```
[Outputs]
    exodus = true
    execute_on = 'timestep_end'
    [./console]
        type = Console
        outlier_variable_norms = false
    [../]
[]
```

6.9 Run

Now, all parameters are set. Go to the folder projects/DwarfElephant/ (cd projects/DwarfElephant) and compile the code with the make command. Afterwards run the example with:

./DwarfElephant-opt -i RBFirstProblem.i

The resulting temperature distribution is stored in the file RBFirstProblem_out.e and should look like Fig. 2.

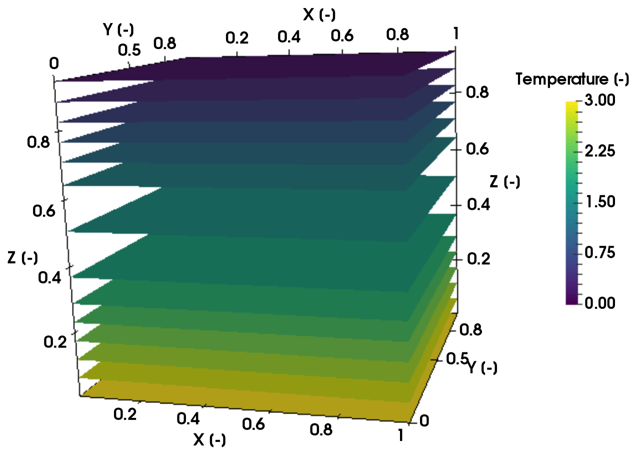

Figure 2: Temperature distribution for a three-layer model. Note that both the mesh dimensions and the temperature have been normalized for numerical reasons and are hence dimensionless. The model has a reference temperature of 10 °C, a reference length of 1,000 m and solver tolerances of 10^{-8}.

References

Adams, B. M., Ebeida, M., Eldred, M., Geraci, G., Jakeman, J., Maupin, K., . . . Rushdi, A. (2017). DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.6 User's Manual. *Sandia National Laboratories, Tech. Rep. SAND2014-4633*.

Alger, B., Andrš, D., Carlsen, R. W., Gaston, D. R., Kong, F., Lindsay, A. D., ... Stogner, R. (2019). *MOOSE Web page.* `https://mooseframework.org`. Retrieved from `https://mooseframework.org`

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., ... Zhang, H. (2017). *PETSc Web page.* `http://www.mcs.anl.gov/petsc`.

Degen, D., Veroy, K., & Wellmann, F. (2019, Jun). *Certified Reduced Basis Method in Geosciences Addressing the challenge of high dimensional problems.* EarthArXiv. Retrieved from `eartharxiv.org/neh9j` doi: 10.31223/osf.io/neh9j

Gaston, D. R., Permann, C. J., Peterson, J. W., Slaughter, A. E., Andrš, D., Wang, Y., ... Martineau, R. C. (2015). Physics-based multiscale coupling for full core nuclear reactor simulation. *Annals of Nuclear Energy*, *84*, 45–54.

Hernandez, V., Roman, J. E., & Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, *31*(3), 351–362.

Hesthaven, J. S., Rozza, G., Stamm, B., et al. (2016). *Certified Reduced Basis Methods for Parametrized Partial Differential Equations.* SpringerBriefs in Mathematics, Springer.

Kirk, B. S., Peterson, J. W., Stogner, R. H., & Carey, G. F. (2006). libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, *22*(3-4), 237–254.

Midttømme, K., Roaldset, E., & Aagaard, P. (1998). Thermal conductivity claystones and mudstones of selected from England. *Clay Minerals*, *33*(1), 131–145.

Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., & Turinici, G. (2002). Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods. *Journal of Fluids Engineering*, *124*(1), 70–80.

Quarteroni, A., Manzoni, A., & Negri, F. (2015). *Reduced basis methods for partial differential equations: An introduction.* Springer International Publishing.

Veroy, K., & Patera, A. (2005). Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Methods in Fluids*, *47*(8-9), 773–788.