Progress in Nuclear Energy 55 (2012) 12–32

Contents lists available at SciVerse ScienceDirect

Progress in Nuclear Energy

journal homepage: www.elsevier.com/locate/pnucene

A verification exercise in multiphysics simulations for coupled reactor physics calculations

Vijay S. Mahadevan [a], Jean C. Ragusa [b,*], Vincent A. Mousseau [c]

[a] Nuclear Energy Division, Argonne National Laboratory, Argonne, IL 60439, USA
[b] Department of Nuclear Engineering, Texas A&M University, College Station, TX 77843, USA
[c] Multiphysics Simulation Technologies, Sandia National Laboratories, Albuquerque, NM 87185, USA

ARTICLE INFO

Article history:
Received 16 November 2011

Keywords:
Multiphysics coupling
Code verification
Method of manufactured solutions
Convergence order

ABSTRACT

The modeling of nuclear reactors involves the solution of a multiphysics problem with various time and length scales. Mathematically, this requires solving a system of coupled, nonlinear, stiff, Partial Differential Equations (PDEs). This paper deals with the verification aspects associated with a multiphysics code, i.e., the substantiation that the mathematical description of the multiphysics equations are solved correctly (in time and space). Multiphysics applications have the added complexity that the solution field participates in various physics components, potentially yielding spatial and/or temporal coupling errors. We present a multiphysics framework that tightly couples the various physical models using the Jacobian-free Newton-Krylov technique (JFNK) and show that high-order convergence can be achieved in both space and time. Code verification results are provided.

1. Introduction

In reactor applications, multiphysics simulations combine models and algorithms from a diverse set of disciplines that need to be consistently coupled to yield accurate answers. Traditional coupling methods (also referred to as operator-split (OS) coupling techniques) lag nonlinearities, resulting in a loss of accuracy and stability (Strang, 1968; Marchuk, 1971; Mousseau et al., 2000). Recently, researchers have considered a different approach whereby the whole nonlinear multiphysics is solved at once, avoiding the lag in the coupled physics components (Knoll and Keyes, 2004; Knoll et al., 2003).

The present paper is concerned exclusively with the verification of numerical methods for multiphysics applications through the use of manufactured solutions. It is an exercise in mathematics, where one ensures that the equations are solved correctly, i.e., that the software has been coded and implemented correctly. This is an integral part of the development process for all modeling software along with validation and uncertainty quantification. The process of code verification usually involves a combination of unit tests and whole system tests. We do not discuss unit testing in this paper (one should always perform unit testing), but rather focus on solution verification efforts applied to multiphysics problems as a whole.

The process of code verification usually includes a combination of the following techniques: (1) comparison against analytical solutions (when such solutions are available; this often requires some drastic model and geometrical simplifications), (2) convergence order estimation using various space/time grid refinements, and, (3) convergence order estimation using the Method of Manufactured Solutions (MMS) (Roache, September, 2009) (this often requires that intricate volumetric and boundary source terms be computed accurately). In the context of nonlinear multiphysics applications, these same recipes apply with the added difficulty that the manner in which the physics components are coupled may degrade the convergence order in time (e.g., when using an operator-split technique) and/or the convergence order in space (e.g., when projections and interpolations of field variables between source and target meshes for different physics components are required).

Firstly, in order to eliminate the temporal pollution error due to operator-splitting, fully implicit numerical methods for tight coupling of various physics have been proposed for multiphysics applications (Knoll et al., 2003; Mousseau, 2004) and are employed in this paper. These techniques, based on Newton's method applied to the whole nonlinear problem, are briefly described in Section 3. With a fully implicit discretization, the temporal coupling error can be controlled and reduced to a small enough user-specified level, hence allowing for a proper verification of the temporal convergence orders. Secondly, each physics component may be discretized on its own mesh (the source mesh) and the solution field

* Corresponding author.
E-mail addresses: mahadevan@anl.gov (V.S. Mahadevan), ragusa@ne.tamu.edu (J.C. Ragusa), vamouss@sandia.gov (V.A. Mousseau).

from a given physics needs to be exported onto another mesh (the target mesh). L_2 projection or interpolation of the solution between the source and target meshes may cause non negligible spatial error (de Boer et al., 2007). In order to minimize the spatial coupling error due to the data transfer between the different physics defined on possibly non-overlapping meshes, several techniques have been developed (Grandy, 1999; Jiao and Heath, 2004). Jiao and Heath have derived rigorous cost estimates for different remapping methods along with the corresponding solution computational times (Jiao and Heath, 2004). The spatial coupling error due to, for instance, the use of different meshes, is still an ongoing topic of research (Tautges, 2004). A detailed study of this effect is beyond the scope of this paper and we will mostly use identical meshes for all physics components throughout the paper, with the exception of one numerical test case for which different spatial meshes are employed and convergence rates are measured. When different meshes are employed here, the spatial coupling error between physics components is controlled using an appropriate numerical quadrature rule, see Section 3.1.2.

The outline of the paper is as follows: in Section 2, we present the physical models employed in this study. In Section 3, we review numerical methods for fully implicit tightly coupled simulations based on the Jacobian-free Newton-Krylov technique, as well as the temporal and spatial discretization techniques. In Section 4, we present a test-bed software framework developed to perform multiphysics simulations for reactor applications. In Section 5, we review the techniques used for code verification. Numerical results are given in Section 6 and conclusions and recommendations are proposed in Section 7.

2. Physical models

In order to demonstrate the feasibility of high-order space/time methods for multiphysics applications in reactor analysis and to present the approach utilized to verify code implementation, the following physical models are employed. These models have been deliberately chosen to be “coarse-grained” because the purpose of the paper is not to validate a model but to present a multiphysics verification study. However, in Section 4, we describe how higher-fidelity physical models can be interchanged easily within the JFNK framework, when such models are deemed necessary. In this paper, we employ the following representations pertaining to the physics involved in reactor applications: 3D multigroup neutron diffusion, 3D nonlinear heat conduction, and (multiple) 1D channels for single-phase fluid flow.

2.1. Multigroup neutron diffusion model

The time-dependent neutron multigroup flux is obtained by solving the following equations

$$\frac{1}{v^g}\frac{\partial\phi^g(r,t)}{\partial t} - \vec{\nabla}\cdot D^g(r,t)\vec{\nabla}\,\phi^g(r,t) + \Sigma_{t,g}(r,t)\phi^g(r,t)$$
$$= \sum_{g'=1}^{G}\Sigma_s^{g'\to g}(r,t)\phi^{g'}(r,t) + \chi_p^g\sum_{g'=1}^{G}\left(1-\beta^{g'}\right)\nu\Sigma_f^{g'}(r,t)\phi^{g'}(r,t)$$
$$+\sum_{j=1}^{J}\chi_{d,j}^g\lambda_j C_j(r,t) \quad \forall g\in[1,G], \tag{1}$$

where G is the total number of energy groups, J is the total number of delayed neutrons groups. The notation used is standard in textbooks. The system of equations is closed with appropriate boundary and initial conditions. The neutronic model employs fuel assembly homogenized cross sections. Cross sections are either tabulated or provided in a closed form approximation, as a function of fuel temperature and coolant density (extension to additional parameters, such a boron concentration, void history, control rod history, etc., is straightforward). The tabulated cross-section values are obtained using table look-up and $\mathbb{R}^p$ interpolation, where p is the total number of parameters used. The energy production due to fission and radiative capture events is given by

$$q'''(r,t) = \sum_{g=1}^{G}\left[\kappa_f^g\Sigma_f^g + \kappa_c^g\Sigma_c^g\right](r,t)\phi^g(r,t), \tag{2}$$

where the κ coefficients represent the amount of energy released per reaction event. The ODEs for the evolution of delayed neutron precursor concentrations are given by

$$\frac{\mathrm{d}C_j(r,t)}{\mathrm{d}t} + \lambda_j C_j(r,t) = \beta_j\sum_{g'=1}^{G}\nu\Sigma_f^{g'}(r,t)\phi^{g'}(r,t). \tag{3}$$

2.2. Thermal heat conduction model

The nonlinear heat conduction model employed here represents the core as a porous medium and is described by the following PDE

$$\rho C_p\frac{\partial T}{\partial t} - \vec{\nabla}\cdot k(T)\vec{\nabla}\,T = q'''(r,t), \tag{4}$$

with appropriate boundary and initial conditions. q''' is the volumetric heat source in the fuel; the density ρ, the specific heat C_p, and the conductivity k may depend on the temperature T. In the current model, the heat conduction equation is solved on the same domain as the neutronic model. In the Results section, we provide verified solutions for cases with identical and separate spatial meshes for a coupled neutronics/heat conduction problem.

The fuel conduction physics can be coupled to the 1-D fluid flow physics in which conjugate heat transfer occurs at the fuel–fluid interface. The boundary term coupling the conducting solid to the fluid is given by

$$-k(T)\partial_n T|_w = h_c\left(T_w,T_f\right)\left(T_w-T_f\right), \tag{5}$$

where T_w is the (solid) wall temperature, T_f is the coolant temperature, and $h_c(T_w, T_f)$ is the convective heat transfer coefficient.

2.3. Coolant flow model

The coolant is modeled using a single-phase fluid flowing vertically in 1-D channels. The model allows for one or multiple 1-D channels (the maximum number of channels being equal to the number of fuel assemblies; a simple user-defined mapping is employed to assign channels and fuel assemblies). The governing equations for the fluid flow are solved in terms of the conservative variables and are as follows

$$\partial_t\rho + \partial_z(\rho u) = 0 \tag{6}$$

$$\partial_t(\rho u) + \partial_z\left(\rho u^2\right) + \partial_z P = -\frac{f_w}{D_{hy}}\rho|u|u \tag{7}$$

$$\partial_t(\rho E) + \partial_z(u(\rho E+P)) = S, \tag{8}$$

where ρ is the fluid density, ρu its momentum, ρE its total energy, P the pressure, and f_w/D_{hy} the wall-friction factor divided by the hydraulic diameter. S is the external source terms representing the energy convected from the fuel and is given by

$$S = h_c\left(T_w, T_f\right) \left(T_w - T_f\right) \frac{P_{\text{wet}}}{A_{\text{hy}}} \tag{9}$$

where $P_{\text{wet}}/A_{\text{hy}}$ is the ratio of the wetted perimeter to the hydraulic area of the flow. D_{hy}, P_{wet}, and A_{hy} are user-supplied values. An equation of state closes the system of equations

$$P = f^{\text{EoS}}(\rho, \rho e), \tag{10}$$

where the internal energy is $\rho e = \rho E - 1/2 \rho u^2$.

3. Numerical discretization

The above physical models yield a system of nonlinearly coupled equations of the form

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}), \tag{11}$$

where $\mathbf{u}$ is the solution vector, $\mathbf{f}$ is a nonlinear vector function, $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$, and N is the total number of unknowns. It helps to represent $\mathbf{u}$ as a vector comprised of the solution vector for each of the M physics components involved, i.e., $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]^T$. A similar definition holds for the nonlinear residual $\mathbf{f}(\mathbf{u})$ where its mth component is the nonlinear residual stemming from the m -th physics component. Note that $\mathbf{f}_m(\mathbf{u})$ may depend effectively on all other fields, e.g., $\mathbf{f}_m(\mathbf{u}) = \mathbf{f}_m(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M)$. In this paper, all the physical models are represented in the form of Eq. (11) for simplicity.

3.1. Spatial discretization

3.1.1. Continuous and discontinuous finite element discretizations

The spatial discretization is performed using the Finite Element library `libmesh` (Kirk et al., 2006). A continuous Galerkin (cG) method (Solin et al., 2003) with Lagrange basis functions is utilized for elliptic/parabolic PDEs (e.g., neutronics and heat conduction models), whereas a discontinuous Galerkin (dG) method (Hesthaven and Warburton, 2008) with Legendre basis functions is employed for hyperbolic systems (fluid flow equations). A wealth of documentation is available in the literature regarding cG and dG methods and we refer the reader to (Solin et al., 2003; Hesthaven and Warburton, 2008; Zienkiewicz et al., 2005) and the references therein for additional details. The weak forms for the equation systems are given in Appendix A.

3.1.2. Spatial coupling errors in multi-mesh approaches

Here, we provide some details regarding how the spatial discretization approach handles the source and target meshes when these are different. Data exchange may be complex to perform accurately and care must be taken to ensure that important quantities are conserved in that process, see, for instance, the discussions in (Grandy, 1999; Jiao and Heath, 2004; Timothy and Caceres, 2009). Recently, an approach based on adaptive mesh refinement technology has been proposed to handle multi-meshes simulations (Solin et al., 2010). As discussed in (Jiao and Heath, 2004), we employ high-order quadrature rules for the numerical integration of the terms residing on the target mesh such that the mass matrix and the load vectors are integrated exactly (Fix, 1972) in order to retain a conservative discretization and consistency in L_2. This strategy is generic and can be applied for arbitrary meshes, provided that the solution for a given physics and the properties can be evaluated at any given point in the target mesh. Hence, the multi-mesh errors are reduced as the order of the numerical quadrature is increased.

For illustration, let us consider two physics components, indexed by 1 and 2. In the weak formulation, the nonlinear residual of physics 1, $\mathbf{f}_1(\mathbf{u}_1, \mathbf{u}_2)$, is multiplied by a test function, $\mathbf{b}_1^j$. The following integral needs to be computed accurately for every cell K_1 of physics 1:

$$\int_{K_1} \mathbf{f}_1(\mathbf{u}_1(\mathbf{x}), \mathbf{u}_2(\mathbf{x}))\mathbf{b}_1^j(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{12}$$

Expanding the solution fields onto the basis functions, $\mathbf{u}_1(\mathbf{x}) = \sum_i \mathbf{b}_1^i(\mathbf{x})\hat{\mathbf{u}}_1^i$ and $\mathbf{u}_2(\mathbf{x}) = \sum_i \mathbf{b}_2^i(\mathbf{x})\hat{\mathbf{u}}_2^i$, and replacing the integral by a numerical quadrature (w_q, x_q) yields

$$\sum_q w_q \mathbf{f}_1\left(\sum_i \mathbf{b}_1^i(\mathbf{x}_q)\hat{\mathbf{u}}_1^i,\ \sum_i \mathbf{b}_2^i(\mathbf{x}_q)\hat{\mathbf{u}}_2^i\right)\mathbf{b}_1^j(\mathbf{x}_q). \tag{13}$$

For identical meshes, $\mathbf{b}_1^i(\mathbf{x}_q) = \mathbf{b}_2^i(\mathbf{x}_q)$ and these quantities are simple to obtain: mapping K_1 onto a reference element is advantageous since the basis functions need only to be evaluated once at the quadrature points of the reference element. However, when the meshes are different, (i) the numerical integration needs to be carried out on the physical element K_1, and, (ii) all the cells of physics 2 overlapping K_1 need to be retrieved and the basis functions $\mathbf{b}_2^i$ need to be evaluated at the quadrature points, ($\mathbf{x}_q$). For general unstructured meshes, one cannot obtain straightforwardly $\mathbf{b}_2^i(\mathbf{x}_q)$ in the reference element since this involves reverse lookups to find the correct target element for physics 2 containing the physical point. Hence, the numerical integration over a cell is carried out on the real geometry (the actual element itself) and not on its mapped reference element. High order quadrature rules for each physics are employed along with inverse mapping of the meshes in different physics in order to evaluate the basis functions at the given physical points. The data structure for this mapping is part of the `libmesh` library. This computation is necessary each time the residual for a given physics component needs to be evaluated and an efficient linked list data structure is created to store the required information in memory and speed up the integration over the cells.

3.2. Time discretization

The ability of achieve high order temporal discretization of coupled physics simulation depends on the methodology employed to tackle the nonlinear system of equations arising from the different physics components. Because of the great disparity of time scales involved, implicit time discretization is preferred for stability reasons. We have chosen time integrators based on the Diagonally Implicit Runge-Kutta (DIRK) family of methods. After spatial discretization, a large system of coupled nonlinear Ordinary Differential Equations (ODEs) is obtained

$$\mathbf{M}\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathbf{f}(\mathbf{U}). \tag{14}$$

$\mathbf{M}$ is the mass matrix resulting from the spatial discretization (the use of a finite volume technique in space or a lumped numerical spatial quadrature results in $\mathbf{M}$ being diagonal). In order to describe the DIRK methods used to discretize Eq. (14) in time, we introduce the Butcher tableaux (Hairer and Wanner, 1996)

$$\begin{array}{c|c} C & A \\ \hline & B^T \end{array},$$

with $B = [b_1, \ldots, b_s]^T$, $C = [c_1, \ldots, c_s]^T$, and $A = (a_{ij})_{i,j=1,\ldots,s}$. An s -stage DIRK method can be expressed as

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{15}$$

where vectors $\mathbf{k}_i$ ($i = 1, \ldots, s$) are obtained by solving the following s nonlinear systems

$$\mathbf{M}\mathbf{k}_i = \mathbf{f}\left(t_n + \Delta t c_i, \mathbf{U}^n + \Delta t \sum_{j=1}^{s} a_{i,j}\mathbf{k}_j\right). \tag{16}$$

Performing the following change of variables

$$\mathbf{Z}_i = \mathbf{U}^n + \Delta t \sum_{j=1}^{s} a_{i,j}\mathbf{k}_j \tag{17}$$

and substituting Eq. (17) in Eq. (16) yields a new set of s nonlinear problems of the form

$$\mathbf{F}(\mathbf{Z}) = \mathbf{M}\mathbf{U}^n + \Delta t (A \otimes I)\mathbf{f}(t_n + \Delta t C, \mathbf{Z}) - \mathbf{M}\mathbf{Z} = 0. \tag{18}$$

Eq. (18) represents the nonlinear system of equations obtained after space and time discretizations of the whole multiphysics problem under consideration. Methods based on either Picard or Newton iterations can be employed to solve Eq. (18) and are described in the next paragraph. The following temporal schemes have been chosen: Backward Euler (BE), a second-order singly DIRK with two stages (SDIRK2(2)), a third-order singly DIRK with three stages (SDIRK3(3)) and a fourth-order singly DIRK with five stages (SDIRK5(4)). All temporal schemes are A-, L-stable (non physical oscillations are damped for large time step sizes) and stiffly accurate (non-degradation of convergence order for stiff problems); for more details on these time integrators, we refer the reader to (Hairer and Wanner, 1996; Norsett, 1986). The Butcher tableaux for the schemes of interest are given below.

$$\begin{array}{c|c} 1 & 1 \\ \hline B^T & 1 \end{array}$$

BE1(1)

$$\begin{array}{c|cc} \gamma & \gamma & \\ 1 & 1-\gamma & \gamma \\ \hline B^T & 1-\gamma & \gamma \end{array}$$

SDIRK 2(2) with $\gamma = 1 - \frac{1}{\sqrt{2}}$

$$\begin{array}{c|ccc} \gamma & \gamma & & \\ \frac{1+\gamma}{2} & \frac{1-\gamma}{2} & \gamma & \\ 1 & \frac{-6\gamma^2+16\gamma-1}{4} & \frac{6\gamma^2-20\gamma+5}{4} & \gamma \\ \hline B^T & \frac{-6\gamma^2+16\gamma-1}{4} & \frac{6\gamma^2-20\gamma+5}{4} & \gamma \end{array}$$

SDIRK 3(3) with $\gamma = 0.435866521508459$

$$\begin{array}{c|ccccc} \gamma & \gamma & & & & \\ 3\gamma & 2\gamma & \gamma & & & \\ \frac{11}{20} & \frac{17}{50} & \frac{-1}{25} & \gamma & & \\ 2\gamma & \frac{371}{1360} & \frac{-137}{1320} & \frac{15}{244} & \gamma & \\ 1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \gamma \\ \hline B^T & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \gamma \end{array}$$

SDIRK 5(4) with $\gamma = 0.25$

3.3. Nonlinear solvers

The system of nonlinear equations, Eq. (18), is solved using OS and JFNK techniques. These are discussed next.

3.3.1. Operator-Splitting

Traditionally, the design and analysis of nuclear reactors is based on a solution procedure involving several mono-disciplinary physics codes, coupled together in a loose manner (Ober and Shadid, 2004; Ropp et al., 2004; Mahadevan, 2006; Ragusa and Mahadevan, 2009), a technique mathematically referred to as Operator-Splitting (OS). OS is still the most commonly employed coupling strategy in the reactor physics community. Data exchange between physics components is often carried out using parallel virtual machines (PVM) and message passing interfaces (MPI). In this coupling paradigm, the output data of one code is simply passed on as the input data to another code at each time step. Often, this strategy is non-iterative and the nonlinearities due to the coupling are not resolved over a time step, reducing the overall accuracy in the time stepping procedure to first-order in time, although high-order time integration might have been used for the individual physics components (Mahadevan, 2006; Ragusa and Mahadevan, 2009). Even though more accurate OS strategies exist (Strang, 1968; Marchuk, 1971). they are not in widespread use in the reactor analysis community.

3.3.2. Jacobian-free Newton Krylov (JFNK)

Brown and Saad proposed (Brown and Saad, 1990) a matrix-free version of Newton's method that alleviates the explicit storage of a Jacobian matrix in memory. By tackling the whole nonlinear as a system of equations that is amenable for the application of Newton's method, tight coupling between the various physics components can be preserved and the use of high-order methods for time integration becomes viable. At each Newton iteration, a linear system of equations involving the Jacobian matrix needs to be solved. As the number of physics components grows, so does the number of unknowns, resulting in possibly a large memory usage to store the Jacobian matrix. Furthermore, assembling this matrix when many physics components are involved is not a trivial task. However, employing a Jacobian-free approximation avoids the need for the Jacobian matrix storage since only the action of the Jacobian matrix on a vector is required to solve the linear system. The resulting Jacobian-free Newton Krylov (JFNK) method has enjoyed much success in recent years in several multiphysics applications; see the review articles (Knoll and Keyes, 2004; Gaston et al., 2009).

Eq. (18) describes the discretized multiphysics problem that needs to be solved for each of the s stages of the chosen DIRK method. Applying Newton's method to Eq. (18), we obtain the following sequence of linear systems to solve

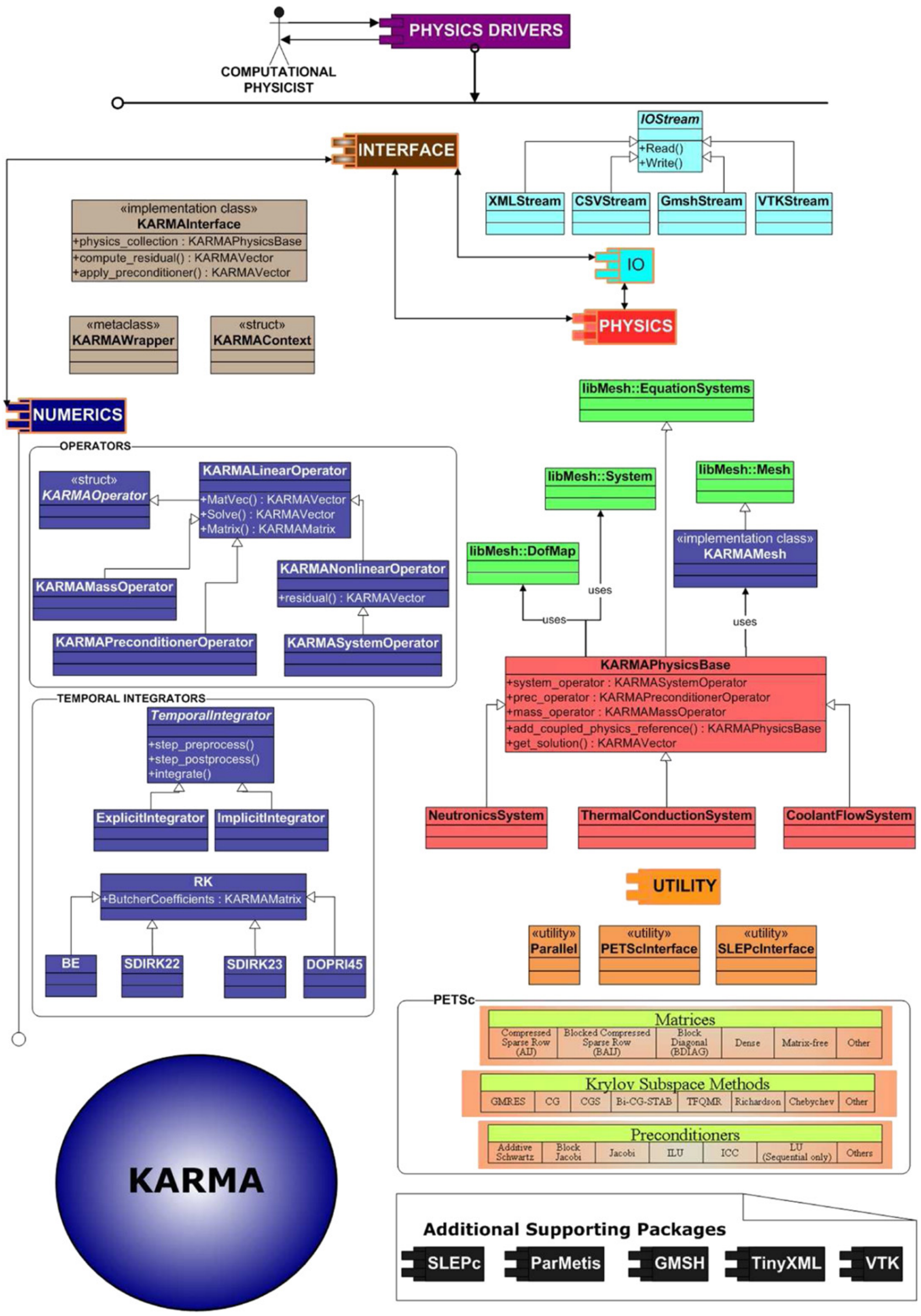


Fig. 1. Schematic diagram of the KARMA framework.

$$\mathbf{J}(\mathbf{U}^{\ell})\delta\mathbf{U} = -\mathbf{F}(\mathbf{U}^{\ell}), \tag{19a}$$

$$\mathbf{U}^{\ell+1} = \mathbf{U}^{\ell} + \delta\mathbf{U}, \tag{19b}$$

where ℓ is the Newton iteration index and $\mathbf{J}(\mathbf{U}^{\ell}) = \left.\frac{\partial\mathbf{F}(\mathbf{U})}{\partial\mathbf{U}}\right|_{\mathbf{U}^{\ell}}$ is the Jacobian matrix evaluated at $\mathbf{U}^{\ell}$. The linear system in Eq. (19) is solved by means of by means of a Krylov technique in which the Jacobian matrix is never explicitly formed (hence the terminology of Jacobian-free). To facilitate the Krylov linear solve, a right-preconditioned algorithm is used:

$$\left(\mathbf{J}(\mathbf{U}^{\ell})\mathbf{P}^{-1}\right)\mathbf{w} = -\mathbf{F}(\mathbf{U}^{\ell}), \tag{20a}$$

$$\mathbf{U}^{\ell+1} = \mathbf{U}^{\ell} + \mathbf{P}^{-1}\mathbf{w}, \tag{20b}$$

with **P** the preconditioning matrix. The linear solves are performed using a Krylov subspace method (here, we employ GMRes since **J** is usually not symmetric for multiphysics problems) and the action of $\mathbf{JP}^{-1}$ on a Krylov vector is obtained using the following finite-differenced definition for the matrix-vector product:

$$\mathbf{J}(\mathbf{U}^{\ell})\mathbf{P}^{-1}\mathbf{v} \simeq \frac{\mathbf{F}(\mathbf{U}^{\ell} + \varepsilon\mathbf{P}^{-1}\mathbf{v}) - \mathbf{F}(\mathbf{U}^{\ell})}{\varepsilon}, \text{ for any vector } \mathbf{v}, \tag{21}$$

where ε is a parameter used to control the magnitude of the perturbation; for additional details on JFNK technique, we refer the reader to (Knoll and Keyes, 2004; Brown and Saad, 1990). It is important to note that this procedure requires only the evaluation of the nonlinear residuals **F** and the entries of the Jacobian matrix are not needed. The computational cost of the JFNK technique mostly resides in evaluating the nonlinear vector-valued functions that are needed at every Krylov iteration. To reduce the dimension of the Krylov subspace, a right-preconditioner **P** has been introduced; this lessens both the memory and computational costs of the GMRes iterations. The matrix-free nature of this approach relies on (i) the fact that Krylov solvers build a solution subspace using matrix-vector operations only, (ii) these matrix-vector operations can be approximated using a finite difference formula that does not require knowledge of the matrix entries, and (iii) the Krylov method is insensitive to the matrix-vector product approximation. The coupling in between the various physics components is embedded in the nonlinear function **f** that is part of the definition of **F**, see Eq. (18).

The preconditioner matrix **P** is generally chosen to be a good approximation of the Jacobian matrix and should be easier to form and to solve as compared to the Jacobian matrix itself. Typically, a block-preconditioner is employed, where a single block represents a given physics, and the off-block-diagonal coupling terms are ignored (akin to a block-Jacobi preconditioner); block-Gauss-Seidel preconditioning can also be employed, thereby retaining some additional coupling in the preconditioner itself. These preconditioners are based on the individual physics and fall into the general category of physics-based preconditioners often applied in the JFNK approach; see also (Knoll et al., 2003; Mousseau et al., 2000).

At each Newton step, the solution of the linear solve, Eq. (19) or Eq. (20), does not need to be obtained with high accuracy, notably in the earlier steps of the nonlinear solve (when the Newton linearization point $\mathbf{U}^{\ell}$ still lies far from the solution of the nonlinear problem) (Knoll and Keyes, 2004; Lowrie, 2004; Tabesh and Zingg, 2009). In inexact Newton methods, the convergence of the linear iteration is given by the following criterion

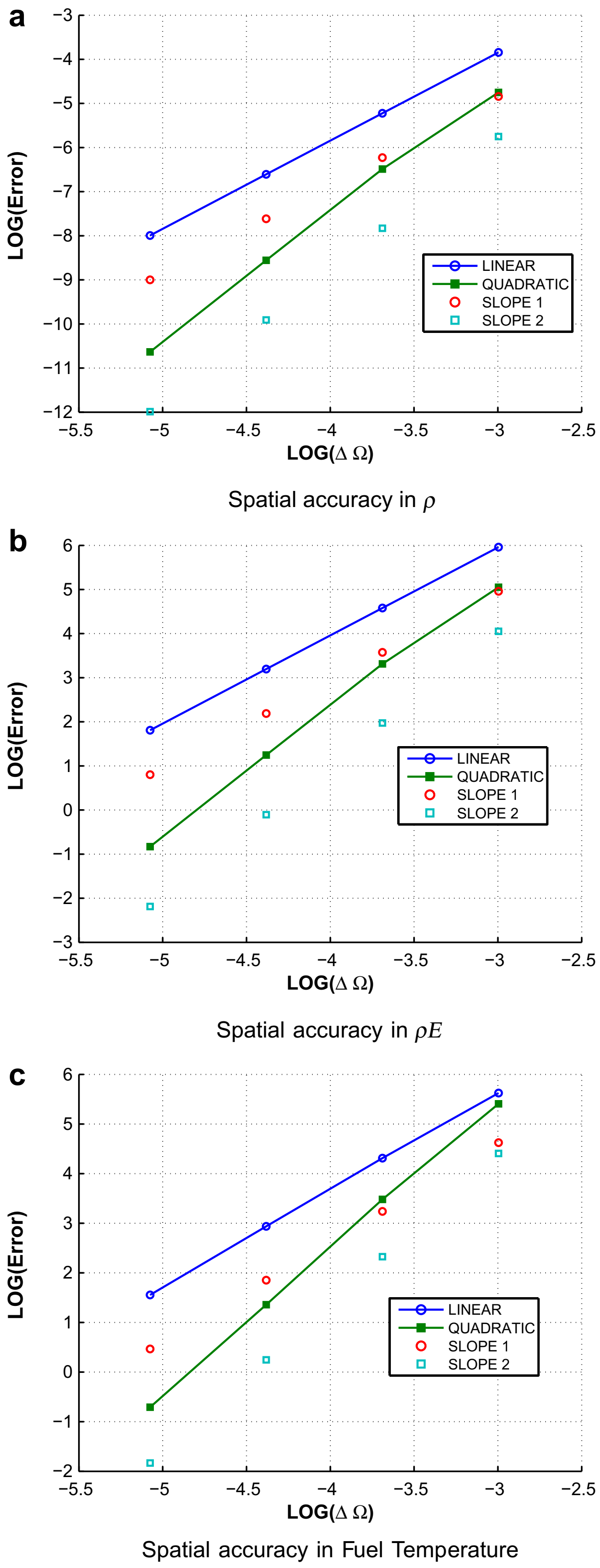


Fig. 2. Conjugate heat transfer example: Spatial convergence.

$$\|\mathbf{J}(\mathbf{U}^{\ell})\mathbf{P}^{-1}\mathbf{w}+\mathbf{F}(\mathbf{U}^{\ell})\|_2<\gamma\|\mathbf{F}(\mathbf{U}^{\ell})\|_2, \tag{22}$$

where the forcing term γ determines how accurately the linear solves are carried out. Variable values of γ can be employed as the nonlinear solution is approached, resulting in a looser linear tolerance when one is still far from the nonlinear solution and a tighter linear tolerance when one is near the nonlinear solution; we employ the strategy given by Eisenstat and Walker (Eisenstat and Walker, 1996) for selecting γ. In Section 6, we will consider the effect of the nonlinear error on the convergence orders of the temporal methods.

4. A multiphysics software perspective

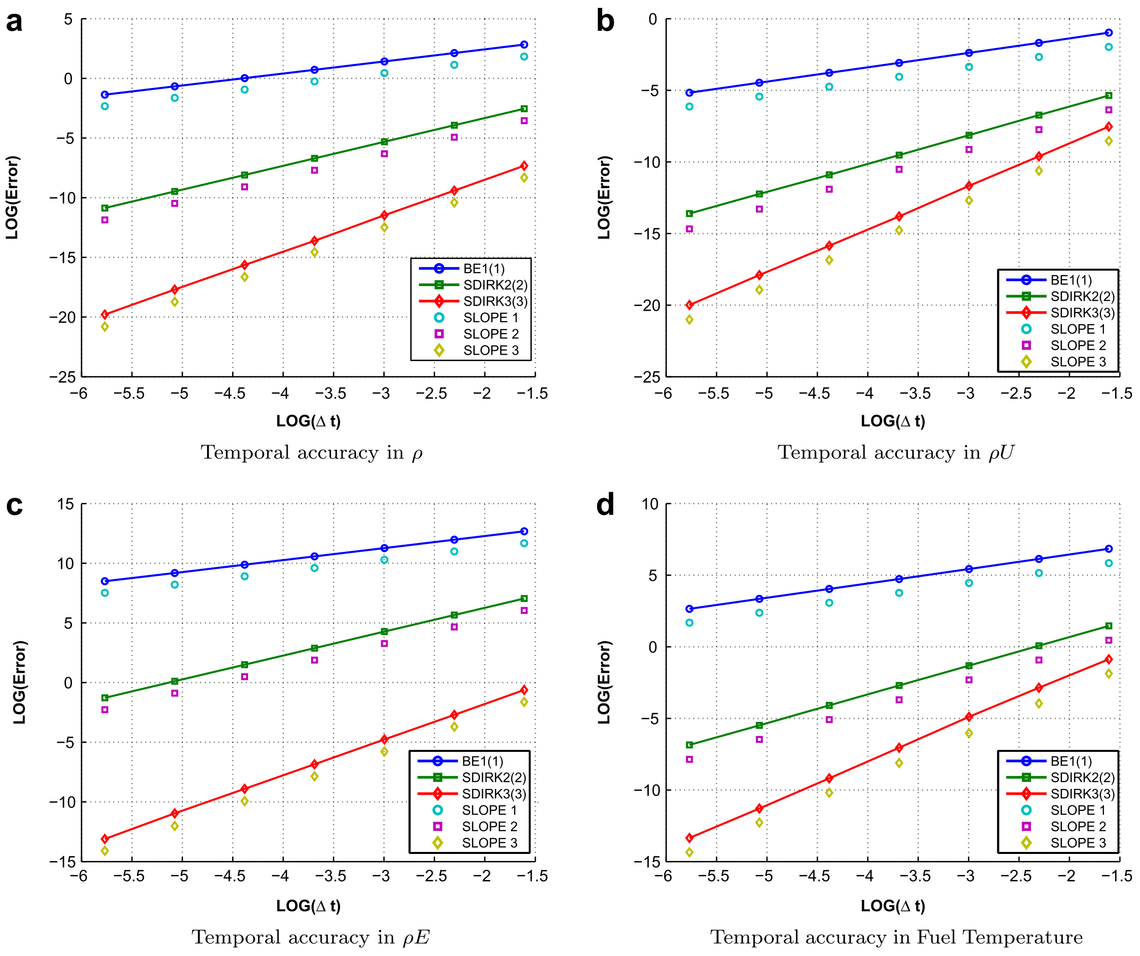
Next, we describe the multiphysics platform from a software perspective. Some of the software requirements we have chosen include: re-use of existing libraries to minimize development time, flexible data containers to write the nonlinear residuals $\mathbf{f}_m$ ($1\le m\le M$, where m is the single physics component index), the ability to use, within the same architecture, the OS or the JFNK approach separately for comparison purposes (since OS schemes are typically preconditioners in the JFNK approach, this requirement is equivalent to using OS schemes as solvers rather than as preconditioners).

The philosophy behind the software framework for multiphysics applications is to tightly solve coupled physical phenomena (i.e., avoiding the traditional OS schemes) using a “loosely coupled” software methodology. The loosely coupled architecture is made possible by requiring a software contract or a defined set of methods to be implemented. For instance, methods for updating the nonlinear residual, computing the Jacobian matrix (optional), and the preconditioner matrix provide the necessary objects needed to solve a multiphysics application. This effort has led to the development of the KARMA framework (c(K)ode for Analysis of Reactor and other Multiphysics Applications). Other examples of computational multiphysics frameworks can be found in (Gaston et al., 2009; Stewart and Edwards, 2004; Lawlor et al., 2006; Jiao et al., 2006).

KARMA is a fully implicit coupled multiphysics transient analysis test-bed code and currently provides the following features that are representative of current trends in coupled multiphysics frameworks:

1. state-of-the-art computer science toolbox and libraries for efficient spatial discretizations, handling and interfacing of the various physics components, fast and robust linear algebra for parallel computing platforms;
2. coarse grain physics models for rapid prototyping, testing and verification; finer grain models can replace the existing ones in a straightforward fashion through the common C++ interfaces;
3. several coupling methodologies (simple OS, OS with Picard iterations, Strang OS (Strang, 1968), Marchuk OS (Marchuk, 1971), JFNK).

KARMA 's plug-in architecture makes it easy to modify, add, and couple additional physics components. It also serves as a framework



Temporal accuracy in ρ Temporal accuracy in ρU

Temporal accuracy in ρE Temporal accuracy in Fuel Temperature

Fig. 3. Conjugate heat transfer example: Temporal convergence.

to conduct experiments on code architectures and software design. A prime concern in the code design is to achieve a high level of efficiency while still maintaining the object-oriented philosophy. We have employed well tested linear algebra and other general purpose libraries in order to reduce the overhead in design and implementation. Similar motivations have also led to the recent development of other multiphysics codes (Gaston et al., 2009; Stewart and Edwards, 2004; Lawlor et al., 2006; Jiao et al., 2006). Some of the supporting libraries used by KARMA include `PETSc` (Balay et al., 1997) (linear and nonlinear algebra library with parallel data structures based on MPI), `libmesh` (Kirk et al., 2006) (finite element library), `SLEPc` (Hernandez et al., 2005) (suite of eigensolvers), `Gmsh` (Geuzaine and Remacle, 2009) (mesh generation), `ParMETIS` (Kumar and Kumar, 1998) (parallel domain decomposition), and the visualization software ViSit (Schroeder et al., 1998). A schematic diagram showing the different parts of the KARMA framework and its interaction with the above mentioned packages is shown in Fig. 1.

A new physics model can be straightforwardly added into the KARMA framework simply by deriving from the `KARMAPhysicsBase` class and implementing the three operators that are essential to solve the physics. These operators are

1. SystemOperator: Implement the nonlinear residual **f**, i.e., the spatially discretized PDE representing the physics. This operator may also provide Jacobian matrix if desired; otherwise a Jacobian-free solve is performed;
2. PreconditionerOperator: Implement a preconditioner where either the matrix **P** is formed in memory or the action of $\mathbf{P}^{-1}$ is provided in order to reduce the total number of linear iterations required at each Newton iteration;
3. MassOperator: Implement a mass matrix, or a lumped version of it, for the time implicit non-autonomous PDE.

Once a physics system implements these three operators, the framework has all the necessary information required to solve the problem. All the material properties (cross-section data, conductivity, equation of state, etc.) are given as function pointers and thus arbitrary user-specified material properties can be chosen.

5. Code verification

Typically, the accuracy and convergence of numerical models are established by using simplified problems for which analytical solutions are available. For time-dependent coupled multiphysics problems, analytical solutions are more difficult to obtain and we need to resort to the Method of Manufactured Solutions (MMS) (Roache, 2002; Knupp and Salari, 2002) or semi-analytical methods (Ganapol, 2008). In the current work, the MMS is used extensively to analyze and prove code correctness. The basic philosophy behind MMS is that an exact solution U_{ref} with enough smoothness in space and time is chosen and substituted into the PDE to obtain a suitable forcing function (i.e., right-hand-side of the PDE) that is then employed in the numerical simulation. Therefore, the numerically obtained values can be compared to the exact ones, providing a measure of the error. This procedure can be applied to nonlinear and coupled physics problems and is an useful tool for code verification purposes. Examples are provided in Section 6. The global error in a numerical solution U_{num} is usually measured in the L_2 norm as

$$\|\text{Error}\|_2 = \|U_{\text{ref}} - U_{\text{num}}\|_2 = \sqrt[2]{\frac{1}{|\Omega|}\int_\Omega \left(U_{\text{ref}}(\mathbf{r}) - U_{\text{num}}(\mathbf{r})\right)^2 \mathrm{d}\mathbf{r}}, \quad (23)$$

where Ω is the spatial domain. A method is of order p_s in space and order p_t in time if the error varies as $\mathcal{O}(\Delta r^{p_s})$ and $\mathcal{O}(\Delta t^{p_t})$, respectively. The order of convergence in space is measured by computing the global error in a transient simulation for which the spatial mesh is successively refined. A small temporal grid is necessary to ensure that the temporal error is small enough so that the error observed is due to the spatial grid only. A similar procedure holds for the temporal error calculation, where the spatial mesh is fine enough so that the spatial errors do not pollute the global error and successive simulations are performed with various time step sizes. The measurement of the error and the comparison of the space/time accuracy orders obtained with expected convergence rates prove that the code implementation is consistent with the mathematics and that the numerical solution converges to the true solution of the PDEs.

6. Results

Manufactured solutions are devised for two coupled physics examples: (1) a conjugate heat-transfer problem and (2) a coupled neutronics and heat conduction problem. The MMS forcing functions used in the simulations are given in Appendices B and C for the different coupled physics scenarios. The accuracy achieved

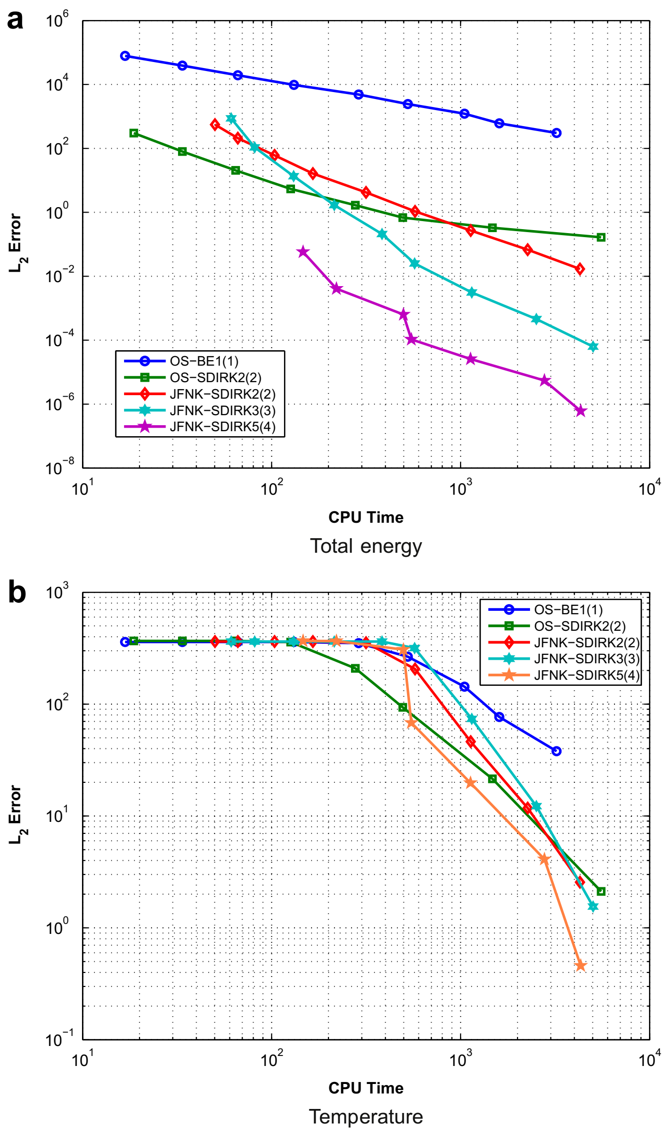


Fig. 4. Conjugate heat transfer example: Efficiency plots.

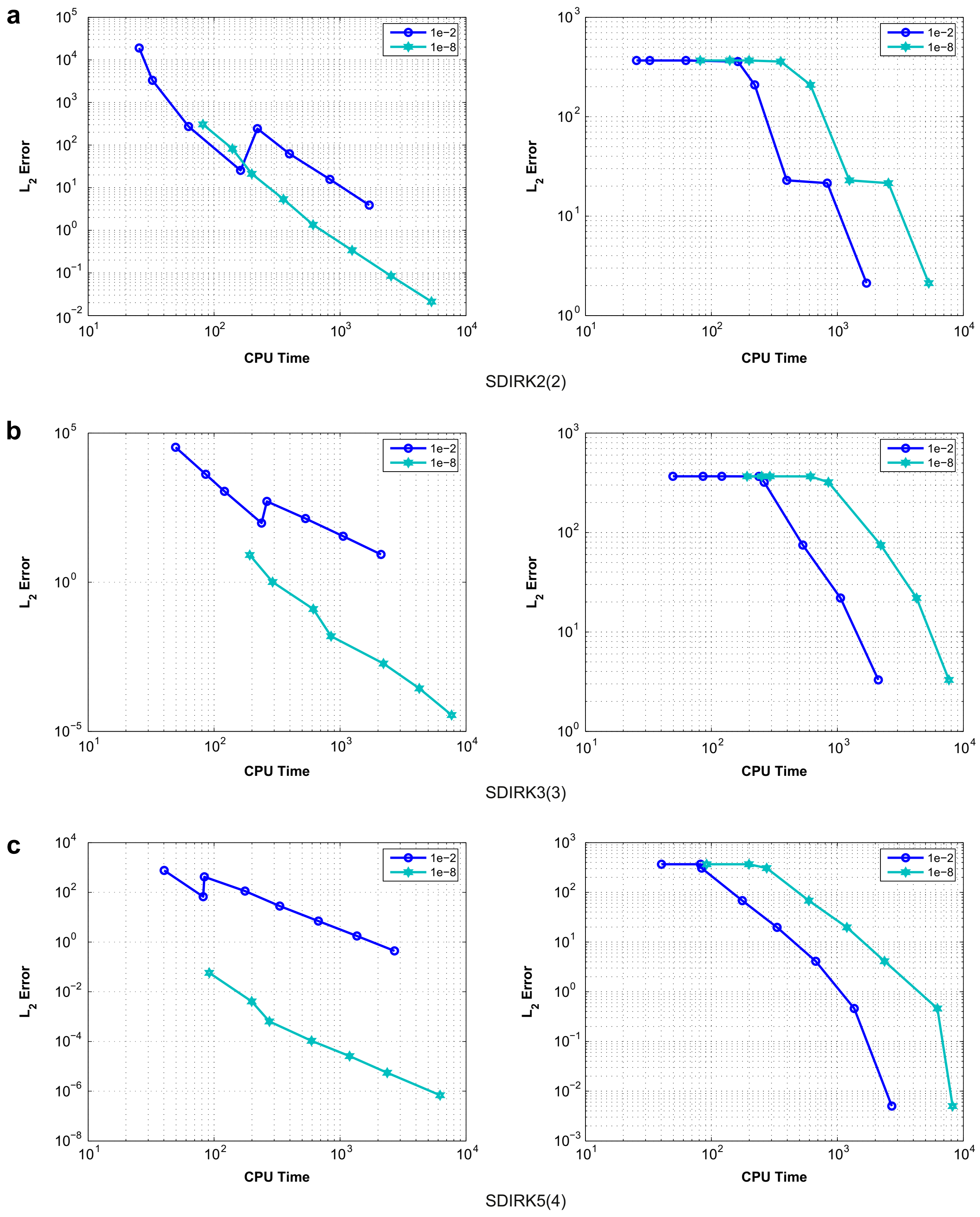


Fig. 5. Influence of the nonlinear tolerance on the accuracy attained versus CPU time for the conjugate heat transfer example. Left column: total fluid energy; right column: fuel temperature.

using the tightly coupled JFNK solver and the traditional OS strategy are compared and efficiency results (accuracy versus CPU time) are shown for problems with disparate time scales.

6.1. Coupled conjugate heat transfer example

6.1.1. Manufactured solution

Conjugate heat transfer between a conducting solid and a fluid is modeled. The flow is assumed to be turbulent with a constant inlet mass flux; Blasius' correlation is used to evaluate the friction factor $f_w = 0.3164/Re^{0.25}$, where Re is the Reynolds number of the fluid. Energy is supplied to the fluid at the wall surface (fluid–solid interface). The exact solutions for the fuel and fluid temperatures, T_{fuel} and T, on a rectangular domain are given below.

$$T_{\mathrm{fuel}}(x,z,t) = r_F\left(1+\tanh\left(C_{tf}t\right)\right)\left(\frac{1}{2}+\sin\left(\frac{\pi x}{2L_x}\right)\right) \times\left(1+\tanh\left(\frac{2w}{3}-\frac{wz}{L_z}\right)\right)+T_{f0} \tag{24}$$

$$T(z,t) = r_T\left(1+\tanh\left(C_{tc}t\right)\right)\left(a+b\tanh\left(-cw+\frac{wz}{L_Z}\right)\right) \tag{25}$$

$$\rho(z,t) = \rho_c+f\left(1-\frac{T(z,t)}{T_{c0}}\right)+g\sqrt{1-\frac{T(z,t)}{T_{c0}}}. \tag{26}$$

The internal energy and total energy are given by

$$\rho e = \rho C_v T \tag{27}$$

$$\rho E = \rho e+\frac{1}{2}\frac{G^2}{\rho}, \tag{28}$$

where $r_T, r_F, T_{f0}, T_{c0}C_{tf}, C_{tc}, w, a, b, c, f, g$, are parameters to control the spatial variations and time scales of the solution. C_v is the specific heat at constant volume and G is the mass flow rate. The equation of state employed to close the system of equations is of the form

$$P = P_0+\alpha(\rho-\rho_0)+\beta(T-T_0), \tag{29}$$

where α, β are constant parameters. Note that α is related to the speed of sound in the flowing medium and provides a simple way to alter the Mach number in the calculations.

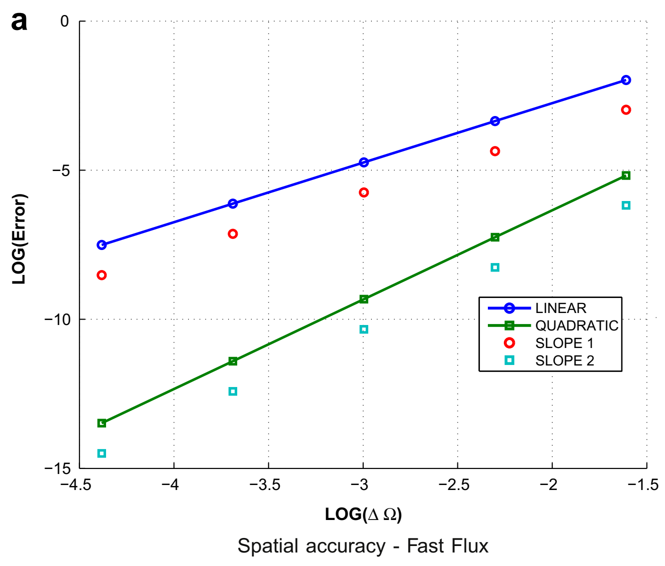
Based on this manufactured solution, the forcing functions have been generated and a study of the convergence order has been carried out for various levels of spatial and temporal discretizations. The numerical solutions approach the true solution as the spatial and temporal meshes are refined, as expected. The convergence graphs, shown in Figs. 2 and 3, prove that the implementation of the physics is verified and demonstrate that high-order accuracy is obtained using the JFNK technique.

6.1.2. Efficiency

The conjugate heat transfer problem introduced above is used to test the efficiency of the coupling methods. Here, we vary the temporal scales of the exact solutions by changing the time constants such that $C_{tf} = 1000C_{tc}$. This forces the temporal evolution of the fuel temperature to occur at a faster rate than that of the fluid. In order to resolve this stiffness, we use the traditional OS coupling strategy with BE and SDIRK2(2) temporal integration methods and JFNK tight coupling strategy with SDIRK2(2), SDIRK3(3), and SDIRK5(4) temporal methods. The accuracy results obtained for a fixed spatial mesh are plotted against the total computational time (work) in Fig. 4 (each curve was obtained by running various time step sizes to reach a given final time).

Fig. 4 presents the efficiency plots. The results indicate that the tight (JFNK) coupling scheme is optimal to obtain accurate solution fields in all physics, once the variation in fuel temperature variable is captured In Fig. 4 we observe that the JFNK strategy (with high-order time integration) asymptotically reduces the error in the solution more quickly (for high accuracy, i.e., small time step sizes, the JFNK-SDIRK3(3) and JFNK-SDIRK5(4) solutions are orders of magnitude better). In contrast, for low requirements on accuracy (larger time step sizes), the JFNK approach require more computational time than OS schemes, especially OS-SDIRK2(2). However, the rate of convergence of OS-SDIRK2(2) is quite poor, equivalent to that of a first-order scheme (OS-BE). In the plot for the temperature variable, Fig. 4, we note that for low CPU time values (i.e., when large time step sizes are used), the temperature variable is not resolved and no schemes are accurate. Once the dynamical physical scales have been resolved, JFNK strategies with high-order time integration become more efficient as the time step sizes become smaller.

In all of the JFNK results shown in Fig. 4, a tight nonlinear tolerance was employed for the inexact Newton solver ($\|\mathbf{F}(\mathbf{U}^{\ell})\|_2 < 10^{-10}$). Next, to investigate the effect of this tolerance



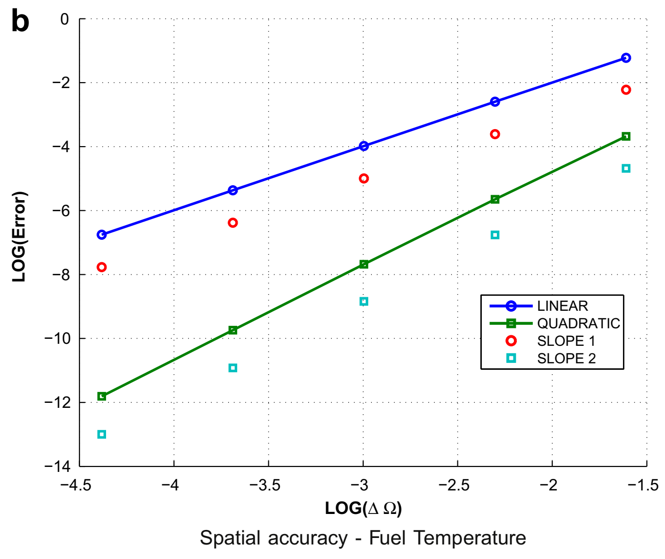


Fig. 6. Spatial accuracy, coupled neutronics/heat conduction.

on the efficacy using the JFNK coupling strategy, simulations are performed using relaxed values: 10^{-2}, and 10^{-8}. The results are provided in Fig. 5(a) for SDIRK2(2), Fig. 5 for SDIRK3(3) and Fig. 5(b) for SDIRK5(4). The plots indicate that increasing the nonlinear tolerance in general, increases the global accuracy in the solution even though the convergence trends remain similar. For coarse nonlinear tolerances, the convergence in the energy variable is not guaranteed to be monotonic. As the tolerances are made more stringent, the theoretical rates of convergence for all the methods are restored. For the temperature variable, increasing the accuracy only increases the total computational cost in general in the resolved regime and does not yield any gain.

6.2. Coupled neutronics/heat conduction example: method of manufactured solutions

Making use of the MMS technique once again, a test problem was constructed to verify convergence. Two neutron energy groups, two delayed neutron precursor groups and one scalar temperature field are considered for this coupled neutronics/heat conduction problem. The total number of solution fields is five. The exact solutions used in 2-D are

$$\phi_1(x,y,t) = A_\phi\left(1+\tanh\left(r_\phi t\right)\right)\sin(\pi x)\sin(\pi y)xy \tag{30}$$

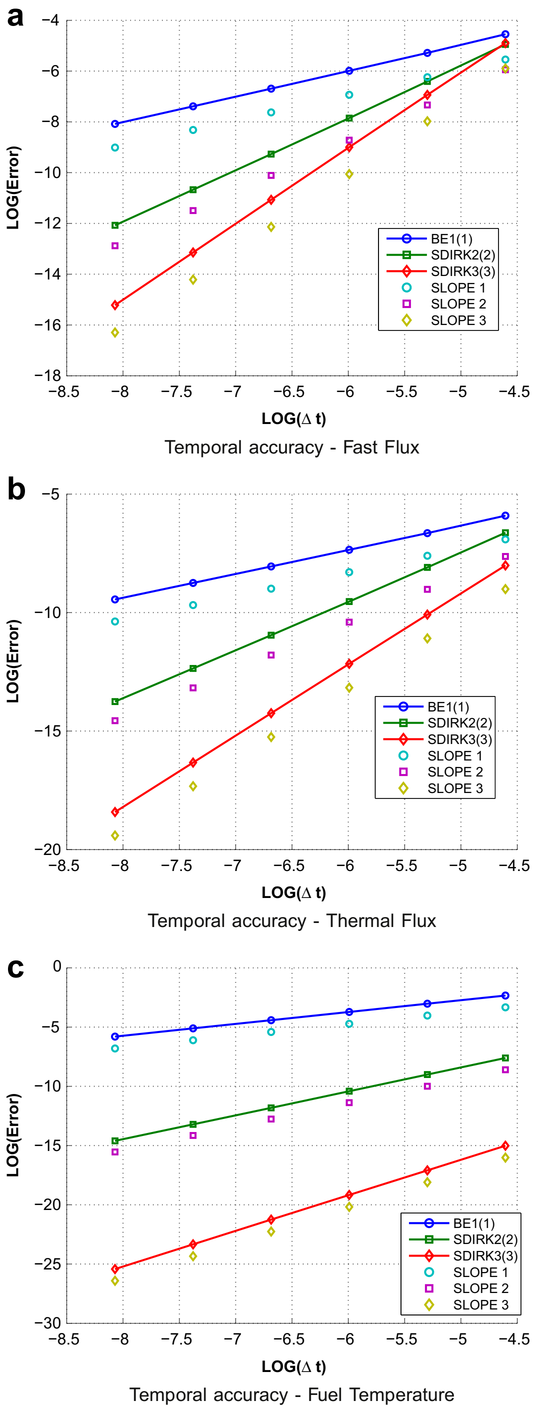


Fig. 7. Temporal accuracy, coupled neutronics/heat conduction.

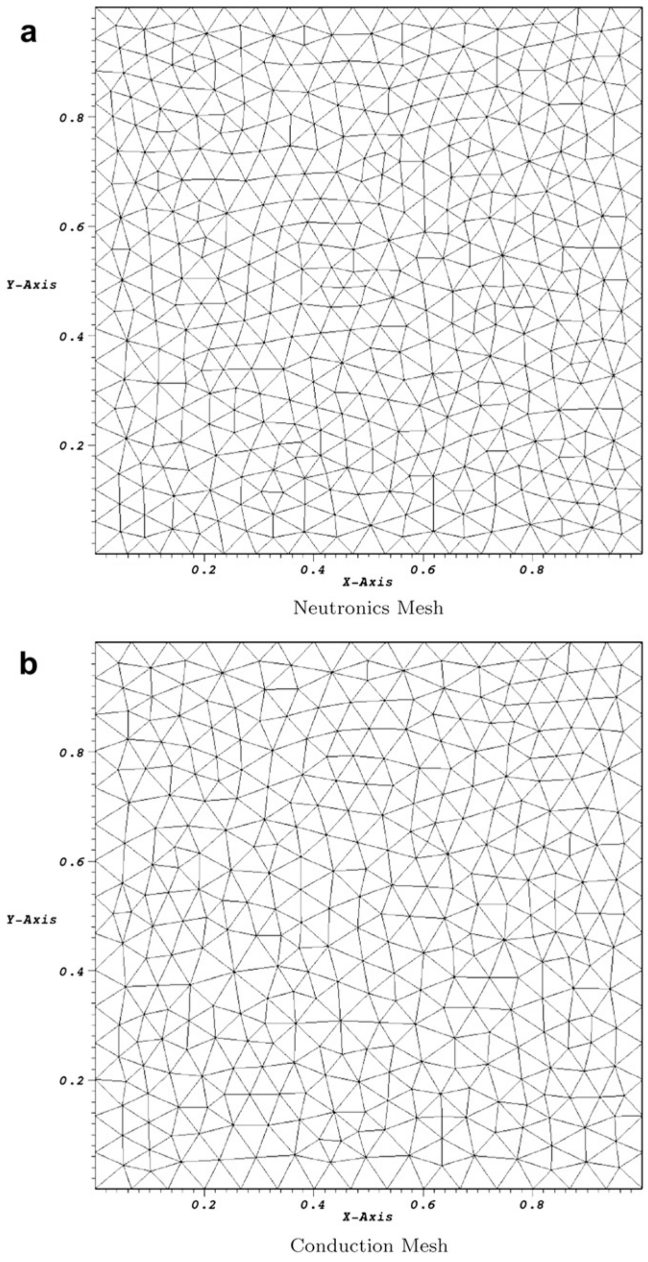


Fig. 8. Non-conforming meshes, neutronics–heat conduction problem.

$$\phi_2(x,y,t) = \phi_1(x,y,t)\times\frac{\Sigma_{s,1\to2}}{\left(\Sigma_{\mathrm{rem},2}+D_2B_g^2\right)} \tag{31}$$

$$C_i(x,y,t) = C_i(x,y,0)e^{-\lambda_i t} + \int_0^t \mathrm{d}s\; e^{\lambda_i(s-t)}\sum_{g=1}^{g=2}\beta_{i,g}\nu\Sigma_{f,g}\phi_g(x,y,s) \tag{32}$$

$$T(x,y,t) = A_T(1+\tanh\,(r_T t))\sin\,(\pi x)\sin\,(\pi y), \tag{33}$$

where $B_g^2 = (\pi/L_X)^2 + (\pi/L_Y)^2$ is the geometric buckling term, L_x and L_Y are the domain sizes in the x and y dimensions. A_ϕ, r_ϕ A_T and r_T are constant parameters. Using the exact solutions for the fluxes, the exact solutions for the precursors, $C_{i,}$ can easily be obtained. The initial precursors values are given by

$$C_i(x,y,0) = \frac{1}{\lambda_i}\sum_{g=1}^{g=2}\beta_{i,g}\nu\Sigma_{f,g}\phi_g(x,y,0),\;\; i = 1,2. \tag{34}$$

Doppler feedback is accounted for in the neutronics model through the removal cross section of group 1:

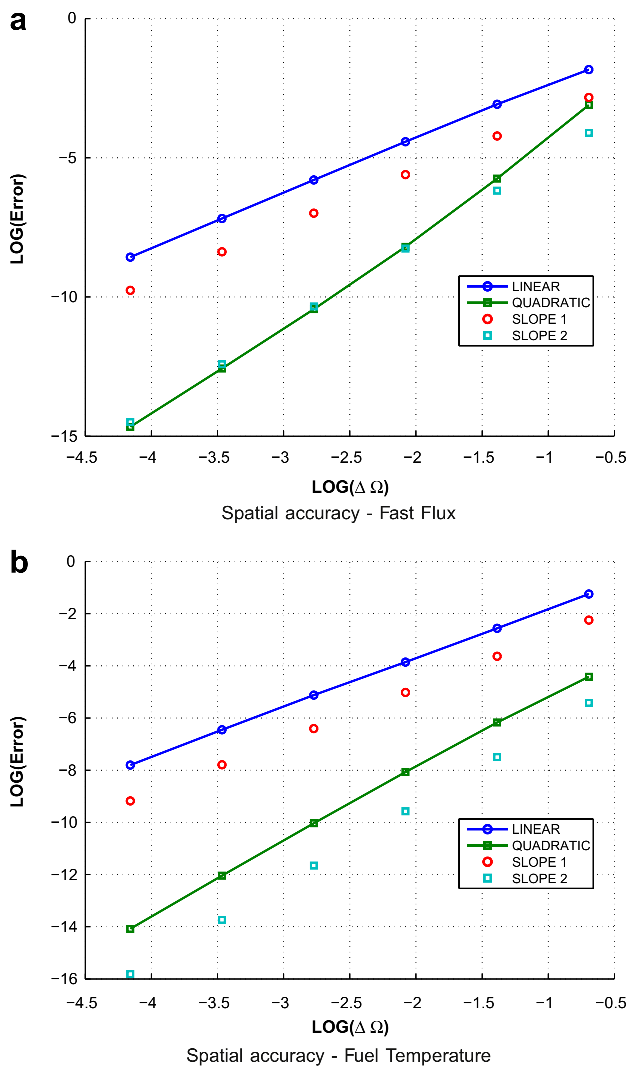


Fig. 9. Spatial accuracy, coupled neutronics/heat conduction with non-conforming meshes.

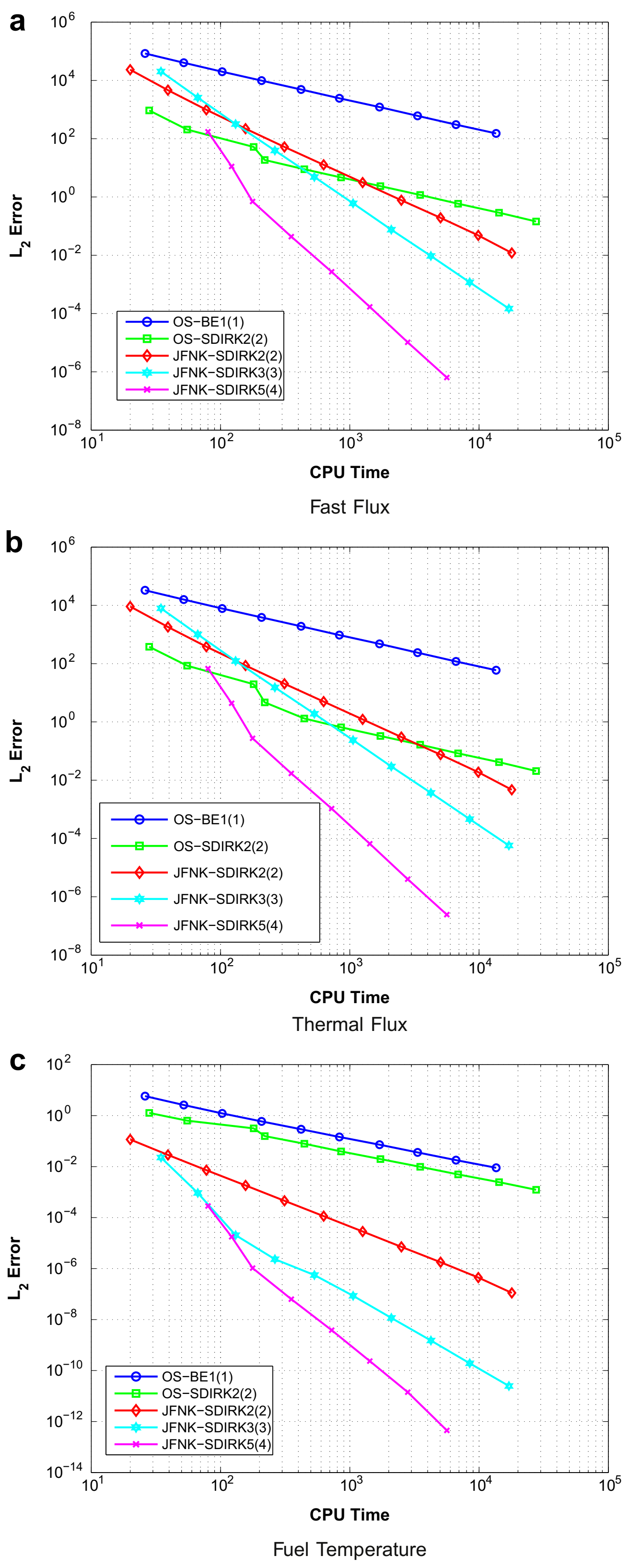


Fig. 10. Coupled neutronics/heat conduction problem: Efficiency plots (L_2 error vs computational work).

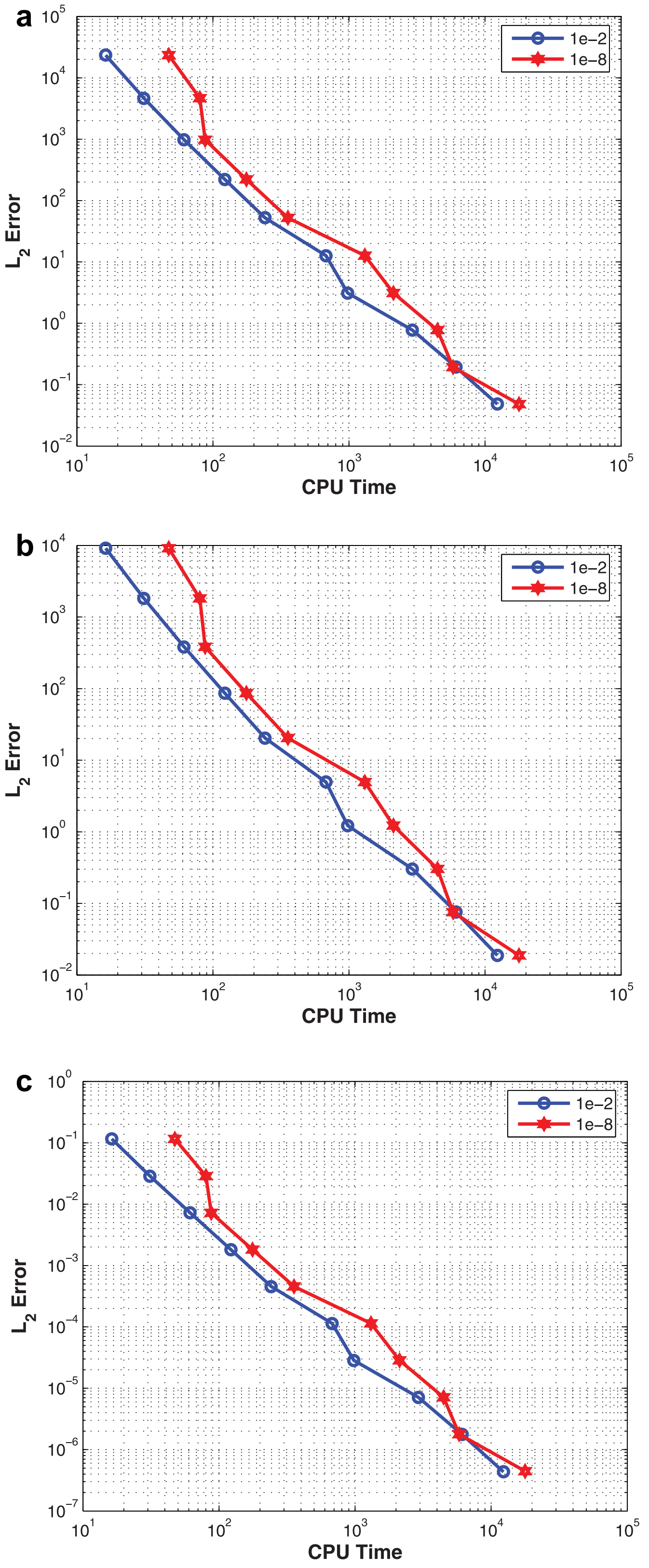


Fig. 11. Influence of the nonlinear tolerance on the accuracy attained versus CPU time (L_2 error vs work) using JFNK-SDIRK2(2).

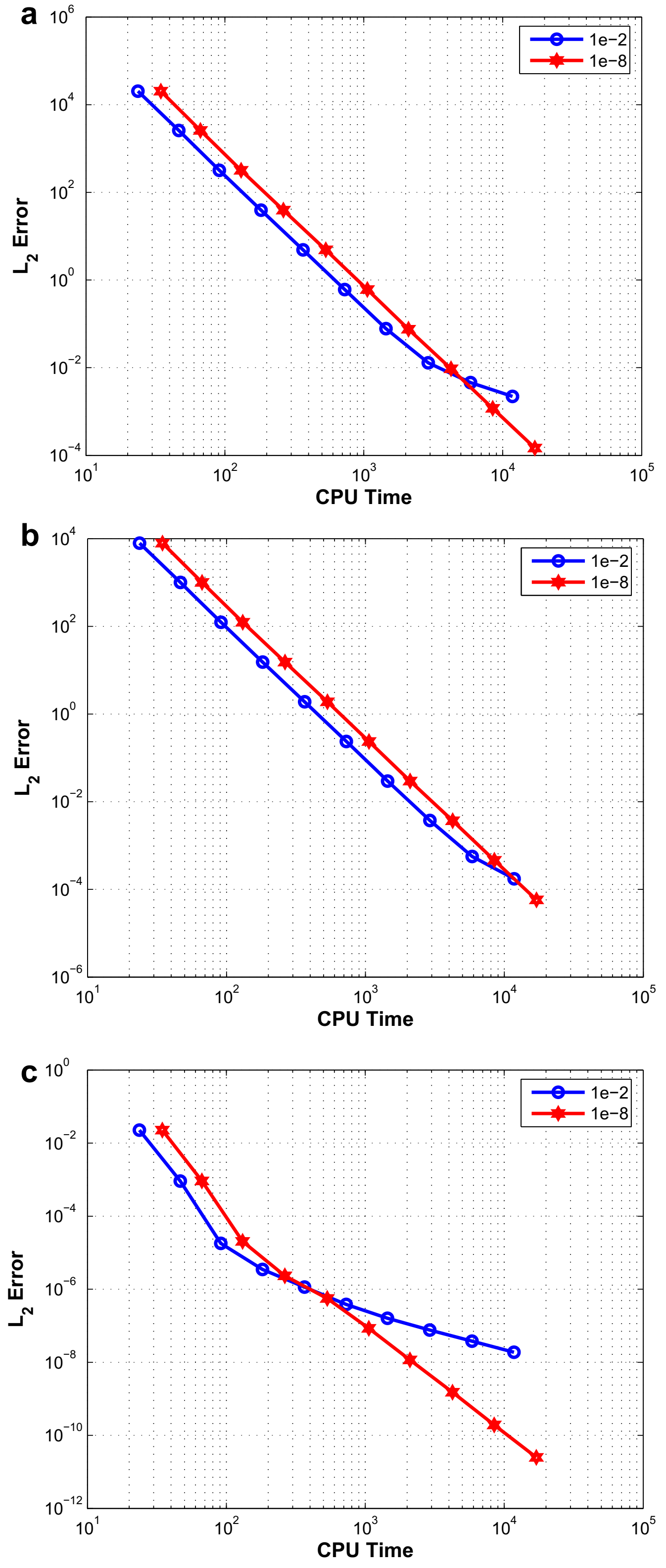


Fig. 12. Influence of the nonlinear tolerance on the accuracy attained versus CPU time (L_2 error vs work) using JFNK-SDIRK3(3).

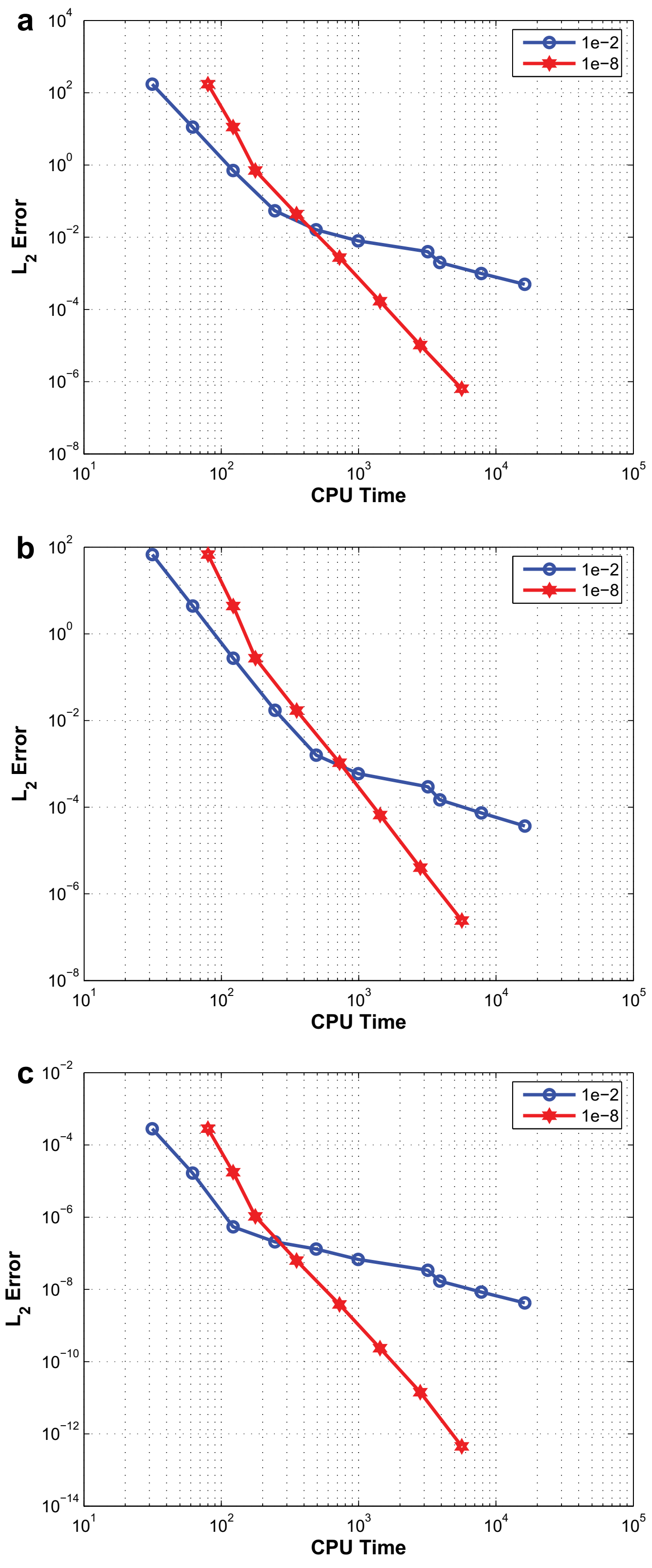


Fig. 13. Influence of the nonlinear tolerance on the accuracy attained versus CPU time (L_2 error vs work) using JFNK-SDIRK5(4).

$$\Sigma_{\text{rem},1}(T) = \Sigma_{\text{rem},1}(T_0) + \frac{\partial \Sigma_{\text{rem}}}{\partial \sqrt{T}}\bigg|_0 \left(\sqrt{T} - \sqrt{T_0}\right). \tag{35}$$

The following equation is employed to describe the temperature-dependent fuel conductivity:

$$k(T) = k_0 + \frac{\partial k}{\partial T}\bigg|_0 (T - T_0). \tag{36}$$

The Matlab script used to obtained the forcing functions is given in Appendix C. Additional cross sections could be made temperature-dependent using the provided script.

First, the same spatial mesh is utilized for the discretization of both physics components and, in the next sub-section, different (non-embedded) spatial meshes are used. Spatial accuracy is verified for both cases.

6.2.1. *Identical spatial meshes*

With the knowledge of the exact solution profiles and the corresponding forcing functions, a space/time convergence study has been carried out. The convergence plots for this example are shown in Figs. 6 and 7. It is clear from the results that the solution fields for all physics components are high-order accurate in space and time. By varying the coupling coefficient $\partial\Sigma_{\text{rem}}/\partial\sqrt{T}$ in Eq. (35) and other free parameters such as r_ϕ, r_T, $1/v_1$, $1/v_2$, stiffer transients were devised and the same convergence rates have been observed.

6.2.2. *Non-conforming spatial meshes*

Multiphysics applications may require that different physics components employ different spatial meshes. Even though a complete study of this question is outside the scope of this paper, we provide here an example where the neutronics and the heat conduction physics utilize different spatial meshes; see Fig. 8. Note that the meshes are *not* embedded (when meshes are embedded, projection and interpolation operators are relatively simple to define). As detailed in Section 3.1.2, the spatial coupling error that may occur due to inadequate projection/interpolation in between source/target meshes can be avoided by employing a high enough numerical quadrature in the finite element setting. It is important to verify then that mitigation of the interpolation errors is possible by using high-order quadratures. In this example, the same test problem with MMS from Section 6.2.1 is considered but different spatial meshes are employed.

The exact solution profiles for the two physics components show that their spatial scales vary differently. Without the use of high-order quadratures, the interpolation error started to dominate and corrupted the measured orders of convergence. In order to eliminate this, high-order quadrature rules have been employed, (see Section 3.1.2). Fig. 9 shows that spatial convergence is recovered, as expected, for the MMS by using high-order quadrature rules.

6.2.3. *Efficiency*

The manufactured solutions for coupled neutronics/conduction problem are used to test the efficiency of the coupling methods with disparate time scales. By changing the time constants as $r_\phi = 1000 r_T$ in Eqs. (30)–(33), the evolution of the neutron flux is forced to occur at a considerably faster rate. We utilize the traditional OS coupling strategy and the tight coupling strategy with JFNK for the $L-$ stable time integrators SDIRK2(2), SDIRK3(3), and SDIRK5(4). The accuracy results obtained are plotted against the total computational time for the different schemes. The plots for the efficiency of temporal resolution for the different solution fields are shown in Fig. 10. In this analysis, the reference solution was computed with the fourth-order SDIRK5(4) scheme.

It is evident from the results that the higher order SDIRK3(3) and SDIRK5(4) schemes prove to be efficient as compared to OS schemes. Clearly the OS-BE coupling strategy is the least efficient. The OS-SDIRK2(2) strategy exhibits a first-order convergence, similar to that of OS-BE, even though the error constant is smaller. For coarse tolerances, OS-SDIRK2(2) proves to be relatively efficient in comparison to the tightly coupled methods with SDIRK2(2) and SDIRK3(3). However, OS-SDIRK2(2) is only first-order accurate asymptotically and is less efficient than tightly coupled schemes for finer tolerances. All JFNK schemes present error reductions in accordance to the expected convergence rates. These results emphasize the gains that can be obtained from the use of tightly coupled strategies with high-order time integrators for multi-physics problems.

Similar to the experiments carried out for the conjugate-transfer problem, the nonlinear tolerance of the inexact Newton solver is varied from 10^{-2} to 10^{-8}. The results are plotted in Fig. 11 for SDIRK2(2), Fig. 12 for SDIRK3(3) and Fig. 13 for SDIRK5(4).

The figures indicate that, for lower-order methods (SDIRK2(2)), increasing the nonlinear tolerance does not provide significant gain in accuracy. Hence it would suffice to use coarser nonlinear tolerance for such methods. But for higher-order methods, a degradation in the order of convergence is clearly noticeable for coarser tolerances and this behavior is more pronounced as the order of the method increases. In general, care is needed when employing higher-order methods with relaxed tolerances since they become sensitive to solving the nonlinear problem more rigorously at every time step in order to maintain the theoretical convergence rates.

7. Conclusions

Various coupling strategies, loose OS coupling and tight JFNK based coupling, have been implemented in a multiphysics framework. The verification of several coupled multiphysics problems was performed using manufactured solutions and high-order spatial and temporal convergence rates were observed, in accordance with the theoretically expected orders of accuracy. Spatial discretization was based on discontinuous or continuous Galerkin Finite Elements, depending upon the nature of the PDE, and temporal integration methods used implicit Runge-Kutta schemes. Using tight coupling methods removes the dominant low-order temporal coupling error due to the operator-split linearization and high-order time integration schemes can then be implemented elegantly. An approach to tackle the spatial coupling error due to the use different meshes between physics components was tested and high-order convergence rates were recovered for multiphysics problems with non-embedded spatial meshes. This technique relied more heavily on accurately integrating the weak forms using high-order quadrature rules than than was required in the case of identical spatial meshes. These preliminary results are promising and further investigation is warranted. The effect of the nonlinear solve tolerance was observed: for low (second) order scheme, a coarse nonlinear tolerance may be sufficient but for high-order schemes, a tighter tolerance was required to recover the asymptotic convergence rate. Also for relaxed accuracy requirements, typically the OS-SDIRK2(2) method outperformed the tightly coupled methods in both the problems tested but asymptotically, the method still yields first-order convergence in contrast to the usage of higher order methods based on JFNK yielding higher order rates.

Studies employing high-fidelity physical models were beyond the scope of this study and have been left for future analyses since the versatility of the KARMA framework allows for different physics to be interchanged in a straightforward manner.

Appendix A. Spatial Discretization using Galerkin FEM

A.1. Elliptic equation systems

Consider an elliptic (e.g., diffusion-reaction) system of PDE given by

$$-\vec{\nabla}\cdot D(r,u)\vec{\nabla}u + c(r,u)u = q \quad \forall \vec{r}\in\Omega, \tag{37}$$

supplemented with boundary conditions and where D, c, q are the diffusion coefficient, reaction coefficient, and the source terms, respectively. For this elliptic system of equations, we employ a spatial discretization based on cG-FEM. The typical weak formulation is given as

$$\mathbf{f}(\mathbf{U}) = \sum_K \int_K \{vq\} - \int_K \{D(r,u)\vec{\nabla}v\cdot\vec{\nabla}u + c(r,u)vu\} + \text{boundary terms} \tag{38}$$

where $\mathbf{U}$ is the vector of unknowns that yields the solution field u expressed with Lagrange basis functions v. The discretization yields a discrete vector of nonlinear residuals, $\mathbf{f}$, which are then be utilized in the nonlinear solution procedure.

A.2 Hyperbolic equation systems

Consider a nonlinear hyperbolic conservation equation of the form

$$\vec{\nabla}\cdot\vec{F}(u) = S. \tag{39}$$

For hyperbolic systems of equations, we employ a spatial discretization based on dG-FEM. Because of the discontinuous nature of the finite element approximation (dG), the flux $\vec{F}(u)\cdot\vec{n}$ is undefined at the cell's boundaries and this quantity is replaced by a numerical flux. In the current work, the Rusanov or Local Lax-Friedrichs (LLF) flux function is used (Lax, 1954). The weak form of the dG method can then be written as

$$\mathbf{f}(\mathbf{U}) = \sum_K \int_K \{vS\} + \sum_K \left\{ \int_K \{\vec{F}(u)\cdot\vec{\nabla}v\} - \int_{\partial k} \left\{ H\left(u^+,u^-,\vec{n}\right)v^+ \right\} \right\}, \tag{40}$$

where $u^{\pm}$ represents the traces of u on both sides of the element interface; the LLF flux $H(u^+,u^-,\vec{n})$ is given by

$$H\left(u^+,u^-,\vec{n}\right) = \frac{1}{2}\left\{ \vec{F}\left(u^+\right)\cdot\vec{n} + \vec{F}\left(u^-\right)\cdot\vec{n} + \lambda\left(u^+ - u^-\right)\right\}, \tag{41}$$

with λ the largest eigenvalue (in absolute value) of the Jacobian matrix of $\vec{F}$.

Appendix B. Matlab script for coupled conduction/fluid MMS problem

Listing 1: MMS forcing function for coupled Conduction/Fluid problem

```
clear all;  clc;

%          |=========|  |   yMAX
%    |     |   Tf    |  |
%    y     |   2-D   |  Tc (1-D: y)
%    |     |  (x,y)  |  |
%          |=========|  |   0
%          0  --x--  xMAX

syms x y t rF rT Ctf Ctc Tf Tc rho rhoU rhoE mu hc Dh
syms xMAX yMAXw del GAMMA Cp Cv RCONST Qconst G
syms dpdrho dpde kfluid Fw a b c p0 rho0 Tf0 Nu0 mu0

% Heat conduction PDE

% Ctf and Ctc are time constants -> Basically, you could
% make Ctf >> Ctc and this will introduce fast time scales
% due to heat conduction and slower scales due to heat
% removed by fluid.

% Fuel temperature exact solution
Tf = Tf0 + (1+tanh(Ctf*t)) * rF * ((0.5+ sin(pi/2*y/yMAX) )'
         * (1+tanh(w*2/3-w*x/xMAX))) ;

% Coolant temperature exact solution
Tc = (1+tanh(Ctc*t)) * rT * (a-b*tanh(c*w-w*y/yMAX)) ;

% viscosity variation with temperature
mu0 = 1.2075e-006 * subs(subs(Tc, t, 0), y, 0) ;

% Density profile
rhoc = 219 ; f = 275.32 ; g = 511.58 ; Tc0 = 2503.7 ;
rho = rhoc + f*(1-Tc/Tc0) + g*(1-Tc/Tc0).^0.5 ;

inte = Cv * Tc ;
% Velocity constant in space-time
v = G./rho ;

% momentum = mass flux
m = rho .* v ;
```

```
% Total energy
e = rho.*inte + 0.5*m.*m./rho ;

% linearized EOS
p = p0 + dpdrho * (rho-rho0) +
   dpde * (Tc-subs(subs(Tc, t, 0), x, 0)) ;

M = v ./ sqrt(dpdrho) ;

% viscosity variation with temperature
mu = 1.2075e-006 * Tc ;

% http://www.cheresources.com/convection.shtml
Nu = Nu0 * (mu./mu0).^0.14 ;

hc = Nu * kfluid / Dh ; % Heat Transfer Coefficient

Re = G * Dh / mu ; % Reynold's number
Fw = 0.3164 / Re^0.25 ;  % Blasius friction factor

% Thermal conductivity for fuel.
k = 6400.0./Tf^2 * exp(-16.35/Tf) ;

% % % % % % % FORCING FUNCTIONS % % % % % % % % % % %

STf = diff(Tf,t) - diff(k*diff(Tf,x), x)
         - diff(k*diff(Tf,y), y)
         + hc * (subs(Tf,x,xMAX)-Tc) ;

Scont = diff(rho, t) + diff(m, y);

Smom = diff(m, t) + diff(m*v, y) + diff(p, y)
         + Fw * v * abs(v) ;

Sener = diff(e, t) + diff((e+p)*v, y)
         - hc * (subs(Tf,x,xMAX)-Tc) ;

disp('STf');   ccode(STf)
disp('Scont'); ccode(Scont)
disp('Smom');  ccode(Smom)
disp('Sener'); ccode(Sener)
```

Appendix C. Matlab script for coupled neutronics/conduction MMS problem

Listing 2: MMS forcing function for coupled Neutronics/Conduction problem

```
clc
clear all;

syms x y z LZ t s T egroups dgroups k avgnu Bg2
syms LX LY CT CF rT rF PI normalization_const
syms DEFAULT_FUEL_TEMP doppler_coeff k0 k1 rho cp
syms beta_tot PHI1 PHI2 xsrem1 xsrem2 xsfiss1 xsfiss2
syms xsrem01 xsrem02 xsnufiss1 xsnufiss2
syms xsdiff1 xsdiff2 energy_per_fission1 energy_per_fission2
syms invvel1 invvel2 xsscatt12 xsscatt21

sx = sin(x/LX*PI) ;
sy = sin(y/LY*PI) ;

T = CT*(1+tanh(rT*t))*sx*sy ;
k = k0 + k1*(T - DEFAULT_FUEL_TEMP) ;

egroups = 2 ;   % energy groups
dgroups = 2 ;   % delayed precursor groups

xsnufiss1 = avgnu * xsfiss1 ;
xsnufiss2 = avgnu * xsfiss2 ;

xsrem1 = xsrem01 + doppler_coeff*(
            sqrt(T) - sqrt(DEFAULT_FUEL_TEMP) ) ;
xsrem2 = xsrem02 ;

beta(1) = sym('beta[1]') ;
beta(2) = sym('beta[2]') ;
lambda(1) = sym('lambda[1]') ;
lambda(2) = sym('lambda[2]') ;
beta_tot = beta(1) + beta(2) ;

% 2-D geometric buckling
Bg2 = PI*PI*(1/LX^2+1/LY^2) ;
PHI1 = CF*(1+exp(rF*t))*sx*sy*(x/LX*y/LY) ;
```

```
PHI2 = PHI1 * xsscatt12 / (xsrem2 + xsdiff2 * Bg2) ;

for dg = 1 : dgroups
   % Initial precursor concentration
    PREC_0(dg) = beta(dg)*(subs(xsnufiss1*PHI1+
                     xsnufiss2*PHI2, t, 0))/lambda(dg) ;
   % Analytically solve the precursor ODE
    PREC(dg) = ( PREC_0(dg) + beta(dg)*int(
             subs(
               (xsnufiss1*PHI1+
                  xsnufiss2*PHI2)*exp(lambda(dg)*s), t, s),
               s, 0, t) ) * exp(-lambda(dg)*t) ;
end

% FORCING FUNCTIONS

% Fuel Temperature
srcT = rho*cp*diff(T, t) - diff(k*diff(T,x),x) -
      diff(k*diff(T,y),y) - normalization_const*
      (energy_per_fission1*xsfiss1*PHI1+
         energy_per_fission2*xsfiss2*PHI2) ;

% Fast Flux
srcFLX1 = invvel1*diff(PHI1,t) - diff(xsdiff1*diff(PHI1,x),x)
         - diff(xsdiff1*diff(PHI1,y),y) + xsrem1*PHI1
         - (1-beta_tot)*(xsnufiss1*PHI1+xsnufiss2*PHI2)
         - xsscatt21*PHI2 ;
if dgroups > 0
    for dg = 1 : dgroups
         srcFLX1 = srcFLX1 - lambda(dg)*PREC(dg) ;
         srcPREC(dg) =  diff(PREC(dg),t) -
         beta(dg)*(xsnufiss1*PHI1+xsnufiss2*PHI2)
         + lambda(dg)*PREC(dg) ;
    end
end
% Thermal Flux
srcFLX2 = invvel2*diff(PHI2,t) - diff(xsdiff2*diff(PHI2,x),x)
         -diff(xsdiff2*diff(PHI2,y),y) + xsrem2*PHI2
         - xsscatt12*PHI1 ;

Power = normalization_const*(energy_per_fission1*xsfiss1*PHI1
         + energy_per_fission2*xsfiss2*PHI2) ;

% Total power in the domain as a function of time
totPower = int (int (Power, y, 0, LY), x, 0, LX) ;

codeT = ccode(T)
codeFLX1 = ccode(PHI1)
codeFLX2 = ccode(PHI2)
for dg = 1 : dgroups
    fprintf('\ncodePREC{%d} = \n\n', dg) ;
    disp(ccode(PREC(dg)))
end

codeST = ccode(srcT)
codeSFLX1 = ccode(srcFLX1)
codeSFLX2 = ccode(srcFLX2)

for dg = 1 : dgroups
    codeSPREC{dg} = ccode(srcPREC(dg)) ;
    fprintf('\ncodeSPREC{%d} = \n\n', dg) ;
    disp(codeSPREC{dg})
end

codePower = ccode(totPower)
```

References

Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F., 1997. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (Eds.), Modern Software Tools in Scientific Computing. Birkhauser Press, pp. 163–202.

Brown, P.N., Saad, Y., 1990. Hybrid Krylov methods for nonlinear systems of equations. SIAM Journal of Scientific and Statistical Computation 11 (3), 450–481.

de Boer, A., van Zuijlen, A.H., Bijl, H., 2007. Review of coupling methods for non-matching meshes. Computer Methods in Applied Mechanics and Engineering 196 (8), 1515–1525.

Eisenstat, S.C., Walker, H.F., 1996. Choosing the forcing terms in an inexact Newton method. SIAM Journal of Scientific Computation 17 (1), 16–32.

Fix, G.J., 1972. Effects of Quadrature Errors in Finite Element Approximation of Steady State Eigenvalue and Parabolic Problems. Academic Press, New York.

Ganapol, B., 2008. Analytical Benchmarks for Nuclear Engineering Applications: Case Studies in Neutron Transport Theory, Tech. Rep. OECD/NEA, Paris (.

Gaston, D., Newman, C., Hansen, G., Lebrun-Grandie, D., 2009. MOOSE: a parallel computational framework for coupled systems of nonlinear equations. Nuclear Engineering and Design 239 (10), 1768–1778.

Geuzaine, C., Remacle, J.-F., 2009. `Gmsh`: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79 (11), 1309–1331.

Grandy, J., 1999. Conservative remapping and region overlays by intersecting arbitrary polyhedra. Journal of Computational Physics 148 (2), 433–466.

Hairer, E., Wanner, G., 1996. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems of Computational Mathematics. Springer Series, New York. vol. 14.

Hernandez, V., Roman, J.E., Vidal, V., 2005. SLEPc: a scalable and flexible toolkit for the solution of Eigenvalue problems. ACM Transaction on Mathematical Software 31 (3), 351–362.

Hesthaven, J.S., Warburton, T., 2008. Nodal Discontinuous Galerkin Methods, of Algorithms, Analysis and Applications Series: Texts in Applied Mathematics. Springer, New York. vol. 54.

Jiao, X., Heath, M.T., 2004. Common-refinement-based data transfer between non-matching meshes in multiphysics simulations. International Journal for Numerical Methods in Engineering 61 (14), 2402–2427.

Jiao, X., Zheng, G., Alexander, P.A., Campbell, M.T., Lawlor, O.S., Norris, J., Haselbacher, A., Heath, M.T., 2006. A system integration framework for coupled multiphysics simulations. Engineering with Computers 22 (3), 293–309.

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.

Knoll, D.A., Keyes, D.E., 2004. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. Journal of Computational Physics 193 (2), 357–397.

Knoll, D.A., Chacon, L., Margolin, L.G., Mousseau, V.A., 2003. On balanced approximations for time integration of multiple time scale systems. Journal of Computational Physics 185 (2), 583–611.

Knupp, P., Salari, K., 2002. Verification of Computer Codes in Computational Science and Engineering (Discrete Mathematics and Its Applications). Chapman and Hall.

Kumar, G.K., Kumar, V., 1998. Vipin, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. Journal of Parallel and Distributed Computing 48, 71–95.

Lawlor, O., Chakravorty, S., Wilmarth, T., Choudhury, N., Dooley, I., Zheng, G., Kal, L., 2006. Parfum: a parallel framework for unstructured meshes for scalable dynamic physics applications. Engineering with Computers 22 (3), 215–235.

Lax, P.D., 1954. Weak solutions of nonlinear hyperbolic equations and their numerical computation. Communications on Pure and Applied Mathematics 7 (1), 159–193.

Lowrie, R.B., 2004. A comparison of implicit time integration methods for nonlinear relaxation and diffusion. Journal of Computational Physics 196, 566–590.

Mahadevan, V.S., 2006. Nonlinearly consistent schemes for coupled problems in reactor analysis, Master's thesis, Texas A&M University, College Station, TX.

Marchuk, G.I., 1971. On the theory of the splitting-up method. In: Numerical Solution of Partial Differential Equations. Academic Press, New York. vol. II.

Mousseau, V.A., Knoll, D.A., Rider, W.J., 2000. Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion. Journal of Computational Physics 160 (2), 743–765.

Mousseau, V.A., 2004. Implicitly balanced solution of the two-phase flow equations coupled to nonlinear heat conduction. Journal of Computational Physics 200 (1), 104–132.

Norsett, S.P., 1986. Local error control in SDIRK-methods. BIT 26 (1), 100–113.

Ober, C.C., Shadid, J.N., 2004. Studies on the accuracy of time-integration methods for the radiation-diffusion equations. Journal of Computational Physics 195 (2), 743–772.

Ragusa, J.C., Mahadevan, V.S., 2009. Consistent and accurate schemes for coupled neutronics thermal-hydraulics reactor analysis. Nuclear Engineering and Design 239 (3), 566–579.

Roache, P.J., 2002. Code verification by the method of manufactured solutions. Journal of Fluids Engineering 124 (1), 4–10.

Roache, P.J., September, 2009. Fundamentals of Verification and Validation. Hermosa Publishers.

Ropp, D.L., Shadid, J.N., Ober, C.C., 2004. Studies of the accuracy of time integration methods for reaction-diffusion equations. Journal of Computational Physics 194 (2), 544–574.

Schroeder, W.J., Martin, K.M., Lorenson, B., 1998. The Visualization Toolkit, first ed. Prentice-Hall, Englewood Cliffs, NJ.

Solin, P., Segeth, K., Dolezel, I., 2003. Higher-order Finite Element methods, Czech Republic Series: studies in Advanced Mathematics. Chapman and Hall/CRC Press.

Solin, P., Dubcova, L., Kruis, J., 2010. Adaptive hp-fem with dynamical meshes for transient heat and moisture transfer problems. Journal of Computational and Applied Mathematics 233 (12), 3103–3112.

Stewart, J.R., Edwards, H.C., 2004. A framework approach for developing parallel adaptive multiphysics applications. Finite Elements in Analysis and Design 40 (12), 1599–1617.

Strang, G., 1968. On the construction and comparison of difference schemes. SIAM Journal on Numerical Analysis 5 (3), 506–517.

Tabesh, M., Zingg, D. W., January 5-8, 2009. Efficient implicit time-marching methods using a Newton-Krylov algorithm. In: 47th Aerospace Sciences Meeting.

Timothy, T.J., Caceres, A., 2009. Scalable parallel solution coupling for multiphysics reactor simulation. Journal of Physics: Conference Series. doi:10.1088/1742-6596/180/1/012017. Vol.180, 012017.

Tautges, T.J., 2004. MOAB-SD: integrated structured and unstructured mesh representation. Engineering with Computers 20 (3), 286–293.

Zienkiewicz, O.C., Taylor, R.L., Zhu, J., 2005. The Finite Element Method, sixth ed. Butterworth-Heinemann.