INTRODUCTION

Nuclear engineering is an inherently “multi-physical” field: neutron transport, heat transfer, solid and fluid dynamics, chemistry, and more are all interrelated in nuclear reactor systems. Multiphysics applications attempt to solve two or more interrelated problems with one combined simulation.

As multiphysics applications are developed, it is important to understand how linking various “single-physics” applications affects overall computational costs. The individual components can be connected by a variety of methods, each with its own strengths and weaknesses. In general, it is easier to link applications together with looser coupling methods. However, it is often the case that performance is improved by combining the applications with tighter coupling methods.

In this summary, several new coupling strategies are examined. The new methods—Solution Interruption, Alternating Nonlinear, and Residual Balance—for coupling two solvers are developed as alternatives to the widely used Picard iteration coupling method. They are implemented in the Multiphysics Object Oriented Simulation Environment (MOOSE) finite element framework [1].

To demonstrate the concepts behind the new coupling methods, an example conjugate heat transfer problem is solved with each of the methods. The example consists of two boundary value problems defined on adjacent spatial domains. Different physical processes happen on each domain and correlate with each other through the shared interface of the two domains, thereby forming a multiphysics problem. Two solvers, one for each spatial domain, were developed and compiled using the MOOSE framework. Each solver uses the Jacobian-Free Newton Krylov (JFNK) method, thus in this work Picard iteration is analyzed. Other methods, such as a monolithic application solved by JFNK, are less popular because they are more invasive and they involve significant changes to existing code. The rewritten (or altogether new) code should then be verified and validated, which can be labor intensive. However, the fully coupled JFNK method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

In the standard Picard iteration method, each physics component is solved completely multiple times before the global solution has been reached. Fig. 1 shows the residuals of each component solver as well as the total residual in the two domain multiphysics example problem, which is described in detail in the results section. From this plot it is apparent that the problem convergence rate is determined by the initial residuals rather than the final residuals; that is, the top rather than the bottom envelope of the curves. Thus there is no need for one solver to strive for perfection while the overall residual remains high due to the other solver. Inexact Newton methods commonly avoid over-solving by using a forcing term to govern the tolerance of the linear iterations [2]. This work develops new coupling methods by applying similar logic to Picard iteration. These modifications are designed to retain the flexibility of Picard iteration, but reduce its solution time.

Alternating Nonlinear Method

The Alternating Nonlinear method prevents solvers from traveling too far in the wrong direction by forcing communications between solvers before each physics component is solved. It is also possible to use a Picard iteration method that allows the two solvers to communicate at the end of each iteration. This method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

Coupling Methods

Picard iteration is a common way to couple multiple physics codes together. The algorithm is straightforward: first solve one application with the values from the other application held fixed, then solve the other application with the new values obtained from the first application, and repeat the process until the coupled problem has converged.

Picard iteration is simple to implement and quite robust. Its popularity is due to the fact that each single-physics component can be a separately developed black box program. To create a multiphysics program from these components, all that is necessary are data transfer methods and a wrapper program to control the successive solutions of the component programs. Other methods, such as a monolithic application solved by JFNK, are less popular because they are more invasive and they involve significant changes to existing code. The rewritten (or altogether new) code should then be verified and validated, which can be labor intensive. However, the fully coupled JFNK method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

In the standard Picard iteration method, each physics component is solved completely multiple times before the global solution has been reached. Fig. 1 shows the residuals of each component solver as well as the total residual in the two domain multiphysics example problem, which is described in detail in the results section. From this plot it is apparent that the problem convergence rate is determined by the initial residuals rather than the final residuals; that is, the top rather than the bottom envelope of the curves. Thus there is no need for one solver to strive for perfection while the overall residual remains high due to the other solver. Inexact Newton methods commonly avoid over-solving by using a forcing term to govern the tolerance of the linear iterations [2]. This work develops new coupling methods by applying similar logic to Picard iteration. These modifications are designed to retain the flexibility of Picard iteration, but reduce its solution time.

Alternating Nonlinear Method

The Alternating Nonlinear method prevents solvers from traveling too far in the wrong direction by forcing communications between solvers before each physics component is solved. It is also possible to use a Picard iteration method that allows the two solvers to communicate at the end of each iteration. This method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

Coupling Methods

Picard iteration is a common way to couple multiple physics codes together. The algorithm is straightforward: first solve one application with the values from the other application held fixed, then solve the other application with the new values obtained from the first application, and repeat the process until the coupled problem has converged.

Picard iteration is simple to implement and quite robust. Its popularity is due to the fact that each single-physics component can be a separately developed black box program. To create a multiphysics program from these components, all that is necessary are data transfer methods and a wrapper program to control the successive solutions of the component programs. Other methods, such as a monolithic application solved by JFNK, are less popular because they are more invasive and they involve significant changes to existing code. The rewritten (or altogether new) code should then be verified and validated, which can be labor intensive. However, the fully coupled JFNK method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

In the standard Picard iteration method, each physics component is solved completely multiple times before the global solution has been reached. Fig. 1 shows the residuals of each component solver as well as the total residual in the two domain multiphysics example problem, which is described in detail in the results section. From this plot it is apparent that the problem convergence rate is determined by the initial residuals rather than the final residuals; that is, the top rather than the bottom envelope of the curves. Thus there is no need for one solver to strive for perfection while the overall residual remains high due to the other solver. Inexact Newton methods commonly avoid over-solving by using a forcing term to govern the tolerance of the linear iterations [2]. This work develops new coupling methods by applying similar logic to Picard iteration. These modifications are designed to retain the flexibility of Picard iteration, but reduce its solution time.

Alternating Nonlinear Method

The Alternating Nonlinear method prevents solvers from traveling too far in the wrong direction by forcing communications between solvers before each physics component is solved. It is also possible to use a Picard iteration method that allows the two solvers to communicate at the end of each iteration. This method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

Coupling Methods

Picard iteration is a common way to couple multiple physics codes together. The algorithm is straightforward: first solve one application with the values from the other application held fixed, then solve the other application with the new values obtained from the first application, and repeat the process until the coupled problem has converged.

Picard iteration is simple to implement and quite robust. Its popularity is due to the fact that each single-physics component can be a separately developed black box program. To create a multiphysics program from these components, all that is necessary are data transfer methods and a wrapper program to control the successive solutions of the component programs. Other methods, such as a monolithic application solved by JFNK, are less popular because they are more invasive and they involve significant changes to existing code. The rewritten (or altogether new) code should then be verified and validated, which can be labor intensive. However, the fully coupled JFNK method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.

In the standard Picard iteration method, each physics component is solved completely multiple times before the global solution has been reached. Fig. 1 shows the residuals of each component solver as well as the total residual in the two domain multiphysics example problem, which is described in detail in the results section. From this plot it is apparent that the problem convergence rate is determined by the initial residuals rather than the final residuals; that is, the top rather than the bottom envelope of the curves. Thus there is no need for one solver to strive for perfection while the overall residual remains high due to the other solver. Inexact Newton methods commonly avoid over-solving by using a forcing term to govern the tolerance of the linear iterations [2]. This work develops new coupling methods by applying similar logic to Picard iteration. These modifications are designed to retain the flexibility of Picard iteration, but reduce its solution time.

Alternating Nonlinear Method

The Alternating Nonlinear method prevents solvers from traveling too far in the wrong direction by forcing communications between solvers before each physics component is solved. It is also possible to use a Picard iteration method that allows the two solvers to communicate at the end of each iteration. This method is desirable because of its nonlinear convergence properties [2], especially for strongly coupled problems.
This method was inspired by a specialized multi-scale method have similar magnitudes. The Residual Balance method speeds is faster toward the end of the solution when both residuals is designed to surpass the performance of the Alternating nonlinear iteration for a complete solve of the inner application.

Care should be taken when selecting the inner application (if no over-solving, as shown in Fig. 3.)

Due to the asymmetry of the Solution Interruption method, care should be taken when selecting the inner application (if both have the requisite iterative solver structure). The Solution Interruption method is ideal for coupling an inexpensive solver inside of a more expensive application. It will be demonstrated later that the Solution Interruption method outperforms the Alternating Nonlinear method if data transfer costs are dominant.

Solution Interruption Method

In the case where one of the solvers is a black box that cannot be divided into discrete iterations, the Solution Interruption method may provide an improvement on Picard iteration. As shown in Fig. 2, the outer application is interrupted after each nonlinear iteration for a complete solve of the inner application. This method was inspired by a specialized multi-scale method developed in MOOSE [4].

Due to the asymmetry of the Solution Interruption method, care should be taken when selecting the inner application (if both have the requisite iterative solver structure). The Solution Interruption method is ideal for coupling an inexpensive solver inside of a more expensive application. It will be demonstrated later that the Solution Interruption method outperforms the Alternating Nonlinear method if data transfer costs are dominant.

Residual Balance Method

The Residual Balance method, a variation on Picard iteration, is designed to surpass the performance of the Alternating Nonlinear method in problems where the residual from one solver is initially larger than that of the other. Fig. 3 shows that the convergence rate of the Alternating Nonlinear method is faster toward the end of the solution when both residuals have similar magnitudes. The Residual Balance method speeds up the initial part of the solution process by taking multiple nonlinear steps in the solver with the larger residual.

The Residual Balance method is straightforward to implement in a system that is capable of Picard iteration. Only two modifications are required: each solver must do at least one nonlinear iteration when it is called, and the absolute tolerance abs_tol must be adaptive. The solution process goes as follows. The first solver stores the initial L2 norm of the residual, \( L^2 \), then does a single nonlinear iteration. Next, the appropriate data transfers take place, including \( L^2 \). Then, the second solver calculates and stores the L2 norm of its residual, \( L^2 \). The second solver performs \( k \geq 1 \) nonlinear iterations until one of the following conditions is met:

\[
L^2 < abs_{tol} \quad \text{or} \quad \frac{L^2}{L^2_{s1}} < rel_{tol},
\]

where

\[
abs_{tol} = \max(A \vert^{0} S_1, min_{abs_{tol}}).
\]

Next, control is passed back to the first solver and the simulation continues in a similar manner until the norms of both residuals have been satisfactorily reduced.

The purpose of \( min_{abs_{tol}} \) is to retain the original function of the absolute tolerance. The parameter \( A \) in Eq. (2) ensures that the current solver will approach the magnitude of the other solver’s residual norm. A value on the order of \( A = 0.1 \) generally yields good results, which is in agreement with other observations [5]. The optimal value of \( A \) may be found, but the performance benefit may not merit the problem-specific effort.

The problem shown in Figs. 1 and 3 is solved with fewer nonlinear iterations by using the Residual Balance method, as shown in Fig. 4. Initially, the residual from Solver 2 is much larger, but the Residual Balance method quickly reduces this residual component until its contribution is commensurate with that of the other solver. From that point, the solution continues in the same manner as the Alternating Nonlinear method.

By adjusting the value of \( A \), the Residual Balance method behaves the same as other methods. If \( A \) is large, it guarantees
The governing equations are as follows

\[
\frac{d^2T}{dx^2} - \epsilon \frac{dT}{dx} = 0, \quad x \in [0, 1],
\]

\[
\kappa \frac{d^2T}{dx^2} = 0, \quad x \in [1, 2],
\]

with the boundary conditions

\[
T(0) = 0.98, \quad T(2) = 1.0,
\]

\[
T(1^-) = T(1^+), \quad \mathbf{n} \cdot q(1^+) = -\mathbf{n} \cdot q(1^-).
\]

The final two conditions specify that the temperature and heat flux must be continuous at the interface, \(x = 1\). \(\mathbf{n}\) is the unit outward normal vector for the respective sub-domain. For this non-dimensionalized problem, \(\epsilon\) is the Peclet number and \(\kappa\) is the ratio of the thermal conductivity in the right domain to that of the left. As in [6], the values \(\epsilon = 9\) and \(\kappa = 0.1\) are used.

Radiation heat transfer between the surfaces at \(x = 1\) and \(x = 2\) is implemented as an adjustment to the heat flux. This source of nonlinearity is governed by \(R\), a non-dimensional constant proportional to the Stefan-Boltzman constant. The definitions of the interface flux from the right and left domains, respectively, are

\[
q_{l=1^-} = -\frac{dT}{dx},
\]

and

\[
q_{l=1^+} = -\kappa \frac{dT}{dx} - R(T_2^4 - T_1^4).
\]

Here the interface spatial derivatives are calculated on the linear Lagrange elements with a first order finite difference formula.

The governing equations can be simplified by using some notations commonly used in finite element method problems. Derivatives are denoted by subscripted commas, and \((\cdot)_\Omega\) denotes an integral over the domain \(\Omega\). Angle brackets are used for terms evaluated on the domain boundaries, \(\partial \Omega\). Thus the weak forms of the governing equations, after some integration by parts, are written as

\[
\left( -T_{,x}, w_{,x} \right)_\Omega + \left( \mathbf{n}_1 \cdot T_{,x}, w \right)_{\partial \Omega} + \left( -cT_{,x}, w \right)_\Omega = 0, \tag{7}
\]

\[
\left( -\kappa T_{,x}, w_{,x} \right)_\Omega + \left( \mathbf{n}_2 \cdot \kappa T_{,x}, w \right)_{\partial \Omega} = 0. \tag{8}
\]

As discussed by Yeckel [6], Robin boundary conditions ensure temperature and heat flux continuity at the interface. With the definitions in Eqs. (5) and (6), the boundary terms in Eqs. (7) and (8) are used to implement the Robin boundary conditions in the following manner

\[
\left( \mathbf{n}_1 \cdot T_{,x} \right)_{|x=1^-} \cdot w = \left( \mathbf{\kappa} T_{,x} \right)_{|x=1^-} \cdot w + \left( R \left( T_2^4 - T_1^4 \right), w \right) + \left( \frac{1 - \alpha}{\alpha} (T_{|x=1^-} - T_{|x=1^-}), w \right), \tag{9}
\]

\[
\left( \mathbf{n}_2 \cdot \kappa T_{,x} \right)_{|x=1^+} \cdot w = \left( -T_{,x} \right)_{|x=1^+} \cdot w + \left( R \left( T_2^4 - T_1^4 \right), w \right) + \left( \frac{1 - \beta}{\beta} (T_{|x=1^-} - T_{|x=1^-}), w \right). \tag{10}
\]

Selection of the values for the Robin boundary condition parameters \(\alpha\) and \(\beta\) is discussed elsewhere [6].

### Coupling Method Performance Comparison

In MOOSE each of the coupling methods can be applied to the example problem. Each domain is solved using JFNK with GMRES as the linear solver. The initial guess is \(T = 0\) everywhere in Case 1. Then in Case 2 the initial guess is manipulated so that one solver starts with a much higher residual than the other. This is done in order to highlight the benefit of the Residual Balance method, as depicted qualitatively in Figs. 1, 2, and 4. In Case 1, the Residual Balance method is equivalent to the Alternating Nonlinear method, because the...
residuals have generally the same magnitude. Table I summarizes the results of these two problems. Note that the solution is unchanged by the initial guess or the solution method.

For strongly coupled problems, fully coupled applications perform better and are more robust than multiple applications coupled by some other method. Thus the quadratic convergence of monolithic JFNK is a sort of gold standard that new coupling methods strive to emulate. The fully coupled solution is included as a reminder that tightly coupled methods should be avoided if possible in strongly coupled problems.

These results show that the Alternating Nonlinear method performs quite well—up to 20% better than Picard iteration. Furthermore, the Residual Balance method performs at least as well as the Alternating Nonlinear method.

Solution Interruption performs full solves of the right domain for each nonlinear iteration of the left domain. In Case 1, it outperforms Picard iteration because most of the nonlinear iterations are being performed on the right domain, which is faster to solve. This is the primary strength of the Solution Interruption method: the bulk of the work can be shifted to the solver that is less costly to solve. In Case 2, however, the left domain requires more work because it has been perturbed by a poor initial guess. Thus the asymmetric structure of the Solution Interruption method is both its weakness and its strength.

Next, we note that Picard iteration normally requires fewer data transfers than the other methods. In a problem where data transfer is a very large cost, Picard iteration or Solution Interruption may be preferable because the sub-application is called fewer times. Table II compares the tightly coupled methods in Case 3, which is dominated by expensive data transfers. Case 3 was created by intentionally modifying Case 1 to use inefficient data transfers.

### CONCLUSIONS

Fully coupled implementations are generally the ideal for multiphysics programs, but often tightly coupled implementations are necessary or much more convenient. Picard iteration permits both codes to be black boxes. The Solution Interruption method requires at least one of the solvers to have an iterative structure that can be interrupted. The Alternating Nonlinear and Residual Balance methods require both solvers to have an iterative structure and provide the residual. Implementing these methods is straightforward, but there may be additional considerations for codes that do not use the finite element method.

The Solution Interruption, Alternating Nonlinear, and Residual Balance methods are simple to use once the necessary codes have been written in MOOSE. These methods can be easily incorporated into other finite element frameworks as well. In light of the potential savings, the Residual Balance method should be tried as a replacement for Picard iteration whenever a fully coupled implementation is not feasible.

### ACKNOWLEDGMENTS

This material is based upon work supported under a Department of Energy, Office of Nuclear Energy, Integrated University Program Graduate Fellowship.

### REFERENCES


