INL/EXT-15-34246

Off-diagonal Jacobian support for Nodal BCs

John W. Peterson, David Andrs
Derek R. Gaston, Cody J. Permann
Andrew E. Slaughter

January 2015

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-15-34246

Off-diagonal Jacobian support for Nodal BCs

John W. Peterson, David Andrs
Derek R. Gaston, Cody J. Permann
Andrew E. Slaughter

January 2015

Idaho National Laboratory

Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Off-diagonal Jacobian support for Nodal BCs

John W. Peterson David Andrš Derek R. Gaston
Cody J. Permann Andrew E. Slaughter

January 30, 2015

1 Introduction

In this brief note, we describe the implementation of off-diagonal Jacobian computations for nodal boundary conditions in the Multiphysics Object Oriented Simulation Environment (MOOSE) [1] framework. There are presently a number of applications [2–5] based on the MOOSE framework that solve complicated physical systems of partial differential equations whose boundary conditions are often highly nonlinear. Accurately computing the on- and off-diagonal Jacobian and preconditioner entries associated to these constraints is crucial for enabling efficient numerical solvers in these applications.

Two key ingredients are required for properly specifying the Jacobian contributions of nonlinear nodal boundary conditions in MOOSE and finite element codes in general:

1. The ability to zero out entire Jacobian matrix rows after "normal" assembly has taken place, and efficiently replace the contents of those rows with the contributions associated to the nodal boundary conditions.

2. An interface for allowing users to specify both the residual and Jacobian contributions associated to their nonlinear boundary condition.

Prior to this work, MOOSE partially implemented item 1 and completely lacked item 2. By "partially," we mean that while MOOSE was indeed capable of zeroing the relevant Jacobian rows, it subsequently placed a 1 on the matrix diagonal by default. This approach is sufficient for implementing nodal boundary conditions of the form $u = g$, where g is a (possibly time and space-varying) constant, but not generalized conditions of the form $f(u) = 0$, where f is a nonlinear function of u and possibly other coupled variables.

In this report, we briefly discuss the definition of a simple model problem with a nonlinear nodal boundary condition, and discuss the performance of MOOSE on said application both before and after the proper nodal boundary condition Jacobian contributions were implemented. For details of the implementation, including the code and surrounding discussion by the MOOSE developers, see GitHub Pull Request #4570[1]. Once this code is merged into MOOSE, it will close long-standing Issue #656[2].

[1] `https://github.com/idaholab/moose/pull/4570`
[2] `https://github.com/idaholab/moose/issues/656`

2 Model Problem

Let $\Omega = [0,1]^2$ be the unit square, and consider the problem of finding u and v such that

$$-\nabla^2 u - v = 0 \tag{1}$$
$$-\nabla^2 v = 0 \tag{2}$$

in Ω, combined with the boundary conditions

$$u = u_\ell \text{ on left} \tag{3}$$
$$v = v_\ell \text{ on left} \tag{4}$$
$$au + u^2 + v^2 = b \quad \text{on right} \tag{5}$$
$$\frac{\partial v}{\partial x} = 0 \quad \text{on right} \tag{6}$$
$$\frac{\partial u}{\partial y} = \frac{\partial v}{\partial y} = 0 \quad \text{on top and bottom} \tag{7}$$

where a, b, u_ℓ, and v_ℓ are given constants whose values are constrained by several factors which we now describe in some detail.

The on- and off-diagonal Jacobian contributions associated with the coupled, nonlinear boundary condition (5) are given by

$$k_{ii} = a + 2u \tag{8}$$
$$k_{ij} = 2v \tag{9}$$

respectively, where i is the degree-of-freedom index associated to variable u for a given node on the right side of Ω, and j is the degree of freedom index associated to variable v on the same node. The Jacobian matrix row associated to row i is therefore given by

$$\left[0, \ldots, \underbrace{a+2u}_{i}, \ldots, \underbrace{2v}_{j}, \ldots, 0\right] \tag{10}$$

If $a = 0$, and an initial guess of $u = v = 0$ is passed to the nonlinear solver (as is done by default in MOOSE), the entire row in (10) will be zero, and the resulting matrix will be singular. We therefore require that $a \neq 0$ in (5), while noting that this restriction obviously does not guarantee $a + 2u \neq 0$ for *all* possible values of u. It is nevertheless sufficient for our purposes here, and clearly illustrates that care must be taken in defining nonlinear boundary conditions, even in relatively simple toy problems.

Since Eqn. (2) is not coupled to u, Ω is a simple domain, and the boundary conditions for v are trivial, the solution to (2), $v = v_\ell$, is also trivial. This further implies that the value of u on the right side of Ω is given by

$$u_r = \frac{-a \pm \sqrt{a^2 - 4\left(v_\ell^2 - b\right)}}{2} \tag{11}$$

In order to have a real-valued solution, we therefore also require that

$$a^2 \geq 4\left(v_\ell^2 - b\right) \tag{12}$$

a further constraint on the constants in (3)–(5), which is relatively simple to enforce in practice by choosing $b > 0$ large enough.

Finally, we note that there are two possible solutions to (1), one each for the positive and negative roots in (11). That is, despite the fact that the differential equations (1)–(2) are linear, the nonlinear character of the boundary condition makes it possible for multiple solutions to exist, a basic characteristic of nonlinear problems in general. The numerical solution which is actually obtained depends on the initial guess, and the convergence behavior of the nonlinear solver (MOOSE employs the inexact Newton method with either preconditioned Jacobian-free Newton-Krylov or traditional matrix-vector products) is affected by the presence of multiple roots. In the following section, we discuss the numerical results, particularly the nonlinear convergence behavior, for solving this model problem with the old and new nodal boundary condition Jacobian implementations in MOOSE.

3 Results

Let $u_\ell = 1$, $v_\ell = 2$, $a = 1$, and $b = 9$ for the constants in (3)–(5). This yields the two possible values

$$u_r = \frac{-1 \pm \sqrt{21}}{2} \approx 1.79, -2.79 \tag{13}$$

for u on the right side of Ω. In the course of our investigation of the model problem, it was determined that an initial guess of $u = v = 0$ causes the Newton iterations to converge to the positive root in (13). The numerical solution for these constants on a 10×10 grid is shown in Fig. 1.

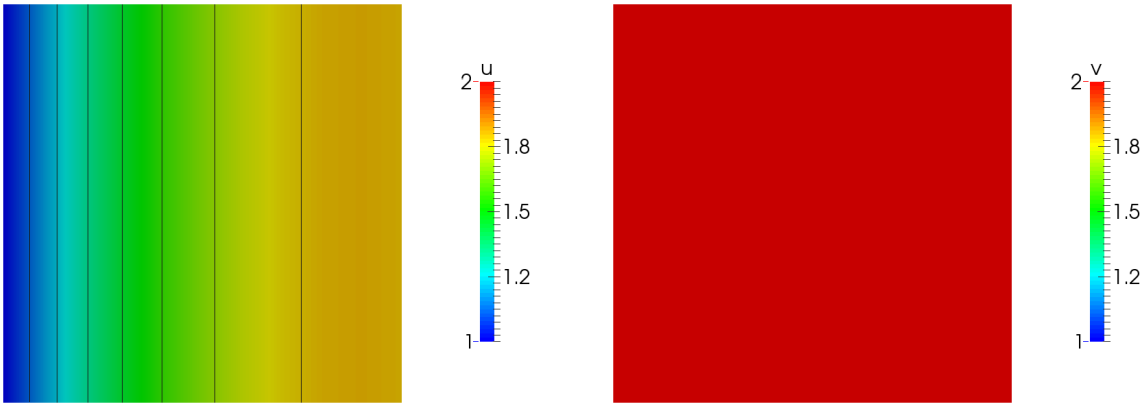

Figure 1: Computed solution to the model problem with the constants described in this section. Ten equally-spaced contours between 1 and 2 are drawn on the u plot to highlight the quadratic character of the solution.

We computed numerical solutions to the model problem with both the "original" (as described in the introduction, i.e. incorrect) and corrected Jacobian terms implemented. Here, we only present results for the "traditional" inexact Newton method (where the Jacobian matrix is explicitly assembled) as the existence of incorrect nodal boundary condition Jacobian terms does not seem to strongly affect the convergence rate of the PJFNK solution method on this model problem[3]. We also attempted solves both with and without the "linesearch" feature of PETSc enabled. The "linesearch" algorithm prevents the nonlinear residual norm from increasing at each Newton step, but does not guarantee convergence for poor initial guesses.

The results of these computations, plotted as the log of the nonlinear residual norm, $\|R\|$, vs. the number of Newton iterations, are shown in Fig. 2. Under the original code path, the Newton solver quickly diverges when linesearch is disabled, and stagnates when it is enabled. In a real code, stagnation of the nonlinear residual is typically just as unacceptable as outright divergence, and also has the negative side effect of requiring many more CPU cycles before reporting failure. In the corrected code, the Newton solver converges with linesearch either on or off. When linesearch is off, we observe an initial small *increase* in the residual norm, followed by a rate of decrease similar to the case where linesearch is on. The case with linesearch enabled converges one iteration sooner than the case with linesearch disabled, the latter case obtains a correspondingly smaller final norm. In both cases, the convergence rate in the final Newton iteration is nearly quadratic, which is a characteristic of the inexact Newton method when the Jacobian is correctly implemented.

4 Conclusions and Future Work

The newly-implemented nodal boundary condition Jacobian terms were shown to perform well on a model problem when the traditional inexact Newton method was employed as the solver. Through the definition of a model problem, we observed that nonlinear nodal boundary conditions can easily lead to ill-posed problems if they are not formulated carefully, and that they induce the full characteristics of general nonlinear equations, including the existence of multiple solutions. Finally, we showed conclusively that, for the inexact Newton solver method, implementing the correct nodal boundary condition Jacobian terms may be essential for preventing divergence of the nonlinear iterations.

Solvers based on the PJFNK method were found to be largely unaffected by the improved accuracy of the preconditioning matrix, which is a secondary result of this project. We believe that more realistic applications than the model problem considered here will possibly recognize more significant improvements in preconditioning for the PJFNK method. More importantly, developers who previously could not rely on the traditional inexact Newton solution method due to the inability to specify nodal boundary condition Jacobian terms now have the option to explore its potential benefits in their applications.

In the initial GitHub Pull Request where the nodal boundary condition Jacobian terms were implemented, some approaches for optimizing and improving the new capability were

[3]Considering that the *action* of the Jacobian was already correct under the "original" code path—only the preconditioning matrix was wrong for the rows associated to the nodal boundary condition degrees of freedom—the lack of effect on PJFNK makes sense.

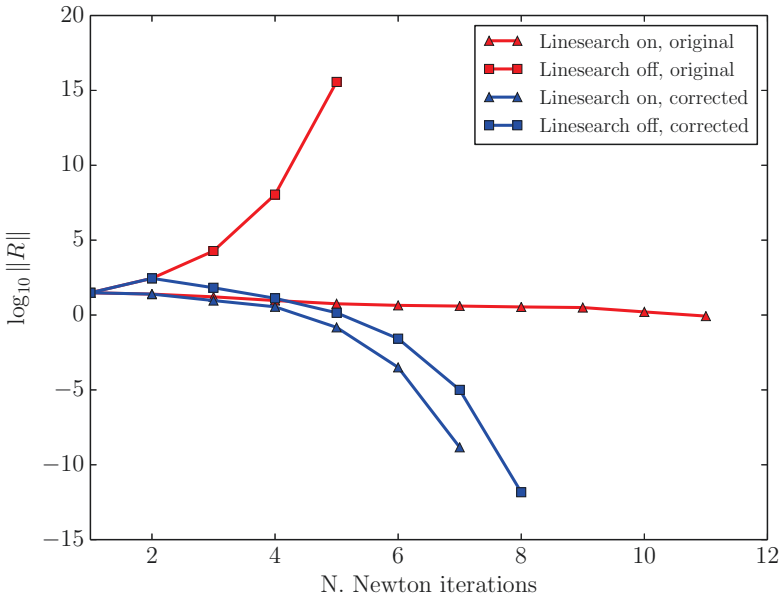

Figure 2: Newton solver convergence for both the "original" and "corrected" nodal boundary condition Jacobian contributions for the model problem defined by (1)–(7).

discussed. The primary optimization involves caching the nodal boundary condition Jacobian information in a threaded region, then "joining" the resulting data onto a single thread before the terms are finally inserted in the Jacobian or preconditioning matrix. This optimization will be beneficial in applications whose boundary condition Jacobian terms are especially costly to compute; one can envision scenarios in which a small simulation or a loosely-coupled external application is required for computing residual and Jacobian terms on the boundary. A secondary optimization involves the manner in which values are actually set in the parallel Jacobian matrix data structure: we currently make N function calls to set these values, where N is the number of nodes with nodal boundary conditions, but the PETSc library provides routines which would allow us to make only a single function call regardless of the number of boundary conditions. Both this and the threading optimization mentioned above will be conducted as future work.

References

[1] D. R. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering Design*, vol. 239, pp. 1768–1778, 2009, http://dx.doi.org/10.1016/j.nucengdes.2009.05.021.

[2] M. R. Tonks, D. R. Gaston, P. C. Millett, D. Andrš, and P. Talbot, "An object-oriented finite element framework for multiphysics phase field simulations," *Computational Materials Science*, vol. 51, no. 1, pp. 20–29, Jan. 2012, 10.1016/j.commatsci.2011.07.028.

[3] R. L. Williamson, J. D. Hales, S. R. Novascone, M. R. Tonks, D. R. Gaston, C. J. Permann, D. Andrš, and R. C. Martineau, "Multidimensional multiphysics simulation of nuclear fuel behavior," *Journal of Nuclear Materials*, vol. 423, no. 1–3, pp. 149–163, Apr. 2012, http://dx.doi.org/10.1016/j.jnucmat.2012.01.012.

[4] R. A. Berry, J. W. Peterson, H. Zhang, R. C. Martineau, H. Zhao, L. Zou, and D. Andrš, "RELAP-7 Theory Manual," Idaho National Laboratory, Tech. Rep. INL/EXT-14-31366, Feb. 2014, https://inlportal.inl.gov/portal/server.pt/document/155280/relap-7_theory_manual.

[5] Y. Wang, "Nonlinear diffusion acceleration for the multigroup transport equation discretized with S_N and continuous FEM with RattleSnake," in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*, Sun Valley, ID, May 5–9, 2013.