INL/CON-20-60467-Revision-1



# BISON Robustness and Performance Improvements

October 2020

Daniel Schwen, Benjamin W Spencer, Stephanie A Pitts, Dylan James McDowell

Changing the World's Energy Future


*INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC*

INL/CON-20-60467-Revision-1

# BISON Robustness and Performance Improvements

**Daniel Schwen, Benjamin W Spencer, Stephanie A Pitts, Dylan James McDowell**

**October 2020**

**Idaho National Laboratory**
**Idaho Falls, Idaho 83415**

**http://www.inl.gov**

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

# Bison Robustness and Performance Improvements

Daniel Schwen,* Alexander D. Lindsay,† Benjamin W. Spencer,* Stephanie A. Pitts,* Dylan J. McDowell,*

*Computational Mechanics and Materials Department, Idaho National Laboratory, Idaho Falls, ID 83415, daniel.schwen@inl.gov, stephanie.pitts@inl.gov, benjamin.spencer@inl.gov, dylan.mcdowell@inl.gov
†Computational Frameworks, Idaho National Laboratory, Idaho Falls, ID, 83415, alexander.lindsay@inl.gov

## INTRODUCTION

BISON is a modern finite-element based nuclear fuel performance code that has been under development at the Idaho National Laboratory (USA) since 2009 [1]. The code is applicable to both steady and transient fuel behavior and can be used to analyze 1D (spherically symmetric), 2D (axisymmetric and generalized plane strain) or 3D geometries. BISON is the fuel performance code used within CASL for LWR fuel under both normal operating and accident conditions.

BISON is built using the INL Multiphysics Object-Oriented Simulation Environment, or MOOSE [2, 3]. MOOSE is a massively parallel, finite element-based framework to solve systems of coupled non-linear partial differential equations using the Jacobian-Free Newton Krylov (JFNK) method [4]. This enables investigation of computationally large problems, for example a full stack of discrete pellets in a LWR fuel rod, or every rod in a full reactor core. MOOSE supports the use of complex two and three-dimensional meshes and uses implicit time integration, important for the widely varied time scale in nuclear fuel simulation. An object-oriented architecture is employed which greatly minimizes the programming effort required to add new material and behavioral models.

The flexibility of the implicit and fully coupled multiphysics approach comes with a need for constructing suitable approximations for the Jacobian matrix of the coupled system used for either preconditioning a Krylov solve or in a direct Newton solve. Preconditioning options for Bison problems need to be revisited with new preconditioning methods becoming available.

### Theory of Automatic Differentiation

Automatic differentiation (AD) is a numerical technique for evaluating the derivatives of functions specified by a computer code. In contrast to symbolic differentiation, no closed form of the differentiable function needs to be provided. This numerical technique enables the differentiation of results of iterative methods such as Newton solves.

We utilize *forward mode* automatic differentiation to obtain derivatives of the residual vector entries with respect to the FEM degrees of freedom (DOFs), and use these derivatives to populate the Jacobian matrix of the system. In forward mode AD the algebra of real numbers is augmented with a new number type, the dual number, consisting of the undifferentiated function value and a vector of function derivatives. Mathematically these numbers represent truncated Taylor series of the form $x + \epsilon \boldsymbol{x}'$, where $\epsilon$ is an infinitesimal element with $\epsilon^2 = 0$, and $\boldsymbol{x}'$ is the vector of derivatives. Addition and multiplication can be written as

$$(x + \epsilon \boldsymbol{x}') + (y + \epsilon \boldsymbol{y}') = (x + y) + \epsilon(\boldsymbol{x}' + \boldsymbol{y}') \quad (1)$$

$$(x + \epsilon \boldsymbol{x}')(y + \epsilon \boldsymbol{y}') = (xy) + \epsilon(x\boldsymbol{y}' + \boldsymbol{x}'y) \quad (2)$$

$$f(x + \epsilon \boldsymbol{x}') = f(x) + \epsilon f'(x)x'. \quad (3)$$

Note that the quadratic term in Equation 2 drops out and the coefficients on the $\epsilon$ term are equivalent to the corresponding symbolic derivative rules. Equation 3 is the general form of function derivatives, for example as required for the trigonometric functions. In this framework the chain rule works naturally (as can be verified by applying Equation 3 twice).

The derivative vector entries are all treated equally in the dual algebra operations. The assignment of which derivative vector corresponds to the derivative with respect to a certain variable is done by *seeding* the vector properly for the initial values used in the computation. Constants will have a zeroed out derivative vector (and non-dual numbers will be implicitly treated as constants). A non-constant primal value $u_i$, the $i$th DOF value of the $u$ FEM variable, will have a derivative vector that is zero, except for a single 1 in the vector element corresponding to the $i$th DOF.

### Automatic Differentiation Implementation

Through the MOOSE framework we use the *MetaPhysicL* [5] metaprogramming and operator-overloaded class library for numerical simulations. MetaPhysicL is a C++library that introduces dual number types and overloads mathematical operators and elementary functions to operate on these dual numbers.

The real valued residual calculation can now be switched over to the dual number algebra implemented in MetaPhysicL resulting an a dual number residual with derivative vector entries representing the Jacobian. We note that the computation of the residual with dual numbers is computationally more complex due to the added operations involving the residual vectors. The MOOSE FEM solve consists of two nested sets of loops - the outer non-linear and the inner linear iterations. The Jacobian is needed only once per non-linear iteration. During the linear iterations the undifferentiated residual value is required, and a dual number computation would be a waste of computation time. Thus two versions of each residual computation routine are needed.

In MOOSE/Bison we implemented the two residual computation routines, with and without dual numbers, through C++templating. This approach avoids code duplication and ensures that both residuals are always mathematically identical. Each MOOSE class in the AD system takes an `enum` type template parameter that can either have the value `ComputeStage::RESIDUAL` or `ComputeStage::JACOBIAN`.

The number types used in the templated class all depend on this compute stage template parameter and resolve to real numbers in the residual case and dual numbers in the Jacobian case. The MOOSE framework calls the appropriate template instantiation at the right compute stage.

As a result the cost of linear iterations does not change when using the AD system in MOOSE. However the cost of a non-linear iteration, and thus a Jacobian computation, can increase significantly. Where in the hand-coded Jacobian approach off-diagonal Jacobian contributions can be approximated or left out, in the AD case all derivatives are computed at all times.

The conversion of MOOSE models covering the mechanics and thermal properties of fuel and cladding materials required creating automatic differentiation (AD) versions of many MOOSE materials and kernels. In the MOOSE tensor mechanics module AD versions of small and large strain models in total and incremental forms for cartesian, cylindrical, and spherical coordinate systems were added resulting in 33 new MOOSE material classes for mechanics with automatic difefrentiation. In Bison a variety of fuel and clad thermomechanical models ranging from Zircaloy to UPuZr metallic fuel were converted over to automatic differentiation, resulting in 14 new material models with AD support.

A key improvement provided by AD affects the inelastic stain models, such as plasticity and creep. Stresses that exceed a yield surface in a high dimensional parameter space need to be returned to the yield surface while resulting in inelastic relaxation of the material. This process, called *return mapping* requires nested iterative solves at every quadrature point in the simulation. The updated stress has a complex coupling to the displacements of the system, resulting in off-diagonal Jacobian contributions in the stress divergence kernels. Implementing this coupling for arbitrary inelastic models and their combinations is extremely cumbersome and error prone. This is why the state of the art in MOOSE/Bison has been to use elastic approximations to the Jacobian contributions instead, resulting in sub-optimal convergence properties of these models. With AD all derivatives of the return stress are automatically computed along with the return stress in the `ADSingleVariableReturnMappingSolution` and `ADRadialReturnStressUpdate` classes.

As forward mode automatic differentiation utilizes operator overloading for custom dual number types external libraries such as LAPACK are unable to provide derivative data. Libraries written in C++can be modified through templating, other functionality has to be reimplemented. We implemented a dual number Eigenvector/Eigenvalue decomposition for rank two tensors which is used when the derivative data for building the Jacobian is needed. During linear iteration, when only the residual vector is evaluated, the standard optimized LAPACK routines are utilized.

A number of Bison models call a legacy FORTRAN implementation of the MATPRO [6] materials properties library for use in the analysis of light water reactor fuel rod behavior. To enable automatic differentiation we started the process of converting these FORTRAN routines over to C++, where we use a templated class for each model. Through the use of unit tests we verify the new implementation, comparing the numerical results over the entire parameter space of each model between the original FORTRAN and the new C++implementation.

**Algorithmic Interventions**

In addition to the automatic differentiation discussed above, we explored the impact of a number of other algorithmic changes on the robustness and run time of the Bison LWR assessment cases. In this paper only the use of a multifrontal direct solver will be presented.

The Bison LWR assessment case suite consists of a variety of fuel performance simulations, from long running integral fuel rod simulations to shorter duration fuel rodlets and cladding only simulations. As such, the assessment suite provides the opportunity to test the robustness improvements provided by different algorithmic changes to the Bison simulations across a wide spectrum of fuel performance simulation types. A set of 209 different assessment case input files were used in this study.

We identified a set of performance metrics which allow us to monitor different aspects of the Bison simulation and measured the impact of the algorithmic changes on the Bison LWR assessment case suite. These metrics were selected to identify different components of a Bison fuel performance analysis that contribute to simulation robustness. Since simulation code robustness can encompass a variety of factors, we have focused here on measuring increases in simulation speed. Foremost among these performance metrics is thus the total simulation run time: the amount of wall time required for each Bison simulation to complete.

We created a new Bison input file block, called the Performance Metric Outputs action, which automatically generates all of the performance metric postprocessors listed in Table I. This action also generates a separate CSV output file which contains all of the performance metric output information from each timestep of the Bison LWR assessment simulations.

The performance metric outputs action was added to each of the assessment case input files within the Bison suite. This standard action enables the consistent generation of robustness measurement data across all of the different algorithmic changes, which will be discussed in the next section. The outputs generated by the performance metric action were used to analyze the impact of each algorithmic change.

To determine if an intervention had a significant impact on run times for a given assessment case, a pairwise t-test was performed using a set of baseline runs as the control and intervention runs as the treatment. These results were then analyzed in terms of the mean percent difference of run times between the two groups. We chose to compare the impact of the different algorithmic changes on each Bison LWR assessment case individually.

Figure 1 shows the distribution of run times for the baseline cases (excluding failed tests). A majority of the Bison assessment cases are clustered below 5,000 seconds with a relatively high number of outlier simulations running for 40,000 seconds or 11.1 hours. With this large spread in simulation time, commenting on the overall effect of the different algorithmic changes across the entire Bison LWR assessment suite is difficult. Since each intervention is optional, we elected to

TABLE I. Standardized metrics and generated outputs created by the 'PerformanceMetricOutputs' action.

| Functionality | Generated Output Quantity Name | Associated Code Class Name |
|---|---|---|
| Total simulation run time | `simulation_alive_time` | PerfGraphData |
| Linear iterations per timestep | `number_linear_iterations` | NumLinearIterations |
| Nonlinear iterations per timestep | `number_nonlinear_iterations` | NumNonlinearIterations |
| Time step size | `time_step_size` | TimestepSize |
| Physical memory usage | `physical_memory_use` | MemoryUsage |
| Total linear iterations | `total_linear_iterations` | CumulativeValuePostprocessor |
| Total nonlinear iterations | `total_nonlinear_iterations` | CumulativeValuePostprocessor |
| Number of degrees of freedom | `number_dofs` | NumDOFs |
| Number of nonlinear variables | `number_nonlinear_variables` | NumVars |
| Residual compute time per timestep | `residual_compute_time` | PerfGraphData |
| Jacobian compute time per timestep | `jacobian_compute_time` | PerfGraphData |
| Number of residual calls | `number_calls_residual` | PerfGraphData |
| Number of Jacobian calls | `number_calls_jacobian` | PerfGraphData |

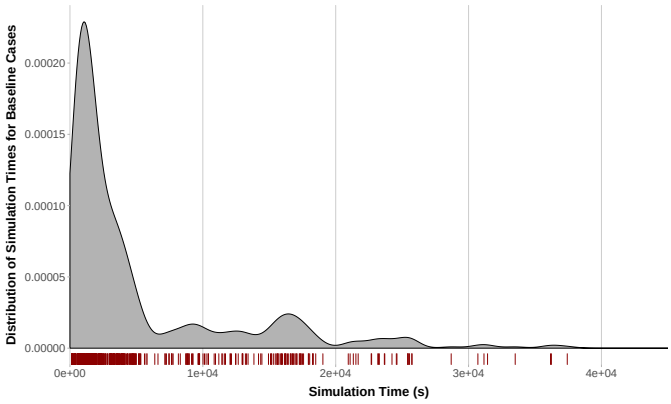


Fig. 1. Rug plot detailing the the distribution of simulation time of the baseline cases.

analyze which exact cases were improved by the algorithmic changes.

A comparison of the run times was obtained by computing the average run time across observations and within each intervention. The mean percent difference (MPD) was then computed to standardize interpretation for each case.

$$MPD_i = \frac{\bar{y}_i - \bar{x}_i}{\bar{x}_i} \times -100 \tag{4}$$

where $i$ represents the $i^{\text{th}}$ case in the assessment suite, $\bar{y}$ represents the mean run time of the treatment group for the $i^{\text{th}}$ case, and $\bar{x}$ represents the mean run time of the baseline cases (i.e. control group) for its $i^{\text{th}}$ case. MPD is also multiplied by $-100$ to convert the result into a percent where positive numbers indicate faster run times and negative numbers indicate slower run times.

Mean standard error bars were computed to determine significance of impact. The standard error was added and subtracted from both sides of the MPD to create an interval. If that standard error interval contained zero, we determined that the treatment had no significant impact on that particular case's run time. The MPD standard error was computed as

$$SE(MPD)_i = \sqrt{\frac{Var(y_i)\bar{x}_i^2 - 2Cov(x_i, y_i)\bar{x}_i\bar{y}_i + Var(x_i)\bar{y}_i^2}{\bar{x}_i^4}} \times 100 \tag{5}$$

where $x_i$ and $y_i$ represent each observation within a specific case. The observations are interdependent since the same Bison LWR assessment cases were run for both the baseline and algorithmic changes, and thus the covariance is required to account for repeated measurements in the dataset.

We note that due to constraints in computing power and time, only three runs were used as observations for each case in both the control and treatment groups. This lower number of runs may contribute to an over-inflated estimated error for each case. This potential larger error means cases that were determined to have no significant impact could have an impact on run time. Further investigation could provide more information about these cases and could refine conclusions regarding impact of the algorithmic changes.

## RESULTS AND ANALYSIS

We tested the new automatic differentiation capabilities on a REBEKA LOCA Bison assessment case [1]. This type of assessment case was chosen for simplicity, as it only contains a cladding component and no fuel. The number of physics models in this simulation is low, and in particular, no thermal or mechanical fuel/clad contact needs to be considered. Both mechanical and thermal contact are in the process of being converted over to AD.

As a second intervention we tested the use of the multi-frontal direct solver STRUMPACK [7]. We ran the entire Bison assessment test suite multiple times to obtain statistically

[1] `bison/assessment/LWR/validation/LOCA_REBEKA_cladding_burst_tests/analysis/rebeka_2d_14MPa/rebeka_singlerod_2d_14MPa_tm.i`

meaningful on the performance changes of each assessment case.

### Automatic Differentiation

We tested the new automatic differentiation capabilities on a REBEKA LOCA Bison assessment case [2]. This type of assessment case was chosen for simplicity, as it only contains a cladding component and no fuel. The number of physics models in this simulation is low, and in particular, no thermal or mechanical fuel/clad contact needs to be considered. Both mechanical and thermal contact are in the process of being converted over to AD.

The REBEKA assessment case is a validation case that models an actual clad burst experiment. The simulation includes an internally pressurized and heated Zircaloy clad tube with a Erbacher-Limback-Hoppe secondary thermal creep model under LOCA conditions [8].

As we discuss in the conclusions below, the benefits of AD are most evident in models that do not converge easily. Our first step was thus to make the existing assessment case a more challenging solve by increasing the permitted time step size by 50%. With a perfect Jacobian available in the AD case it is possible to use a direct Newton solve. We added a predictor to the assessment case which has been shown to improve Newton convergence by providing a better initial guess based on an extrapolation of the solution variables at previous time steps. As a final measure we changed the time integration scheme to second order implicit Euler. As shown in Figure 2 these optimizations to the *Original* assessment case reduce both the iteration counts as well the total execution time shown in the *Optimized no AD* bars.

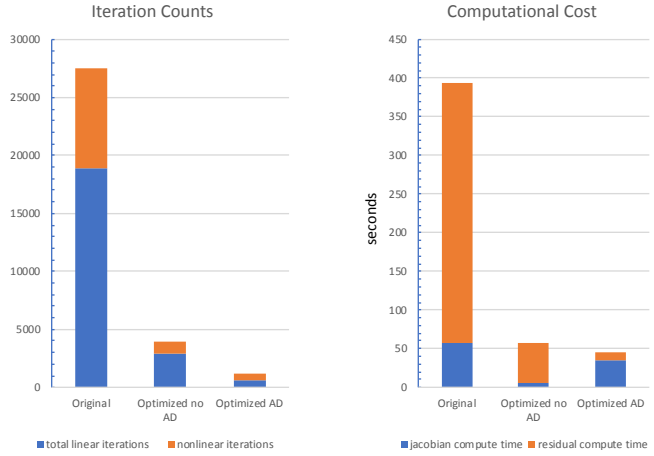


Fig. 2. Iteration counts (left) and walltime (right) the REBEKA assessment case with original, optimized, and automatic differentiation enabled runs.

We note that the time step increase has the side effect of lowering the accuracy of the solve slightly. A selection of cladding parameters at the time of bursting is shown in Table II. We confirmed that the calculated stresses, temperatures, and predicted burst times are within a fraction of a percent for all runs.

[2] `bison/assessment/LWR/validation/LOCA_REBEKA_cladding_burst_tests/analysis/rebeka_2d_14MPa/rebeka_singlerod_2d_14MPa_tm.i`

### Multifrontal Solver

SuperLU has been the suggested package for preconditioning Bison simulations for several years now because it outperforms other preconditioners such as algebraic multigrid (AMG) methods and additive Schwartz on the problems for which Bison is typically used. SuperLU performs a super-nodal (directed-acyclic graph) LU factorization with $O(N^2)$ FLOPS and has an $O(N^{4/3})$ memory requirement when used in the context of solving 3D partial differential equations (PDEs). SuperLU is an optional package for PETSc and requires no other options to make it work well for LWR simulations.

More recently, multi-frontal solvers have been developed that outperform the best super-nodal solvers. Among these is STRUMPACK's multi-frontal solver for hierarchically semi-separable matrices (in the following, we use "STRUMPACK" to denote the factorization). STRUMPACK has $O(N^{4/3} \log N)$ FLOPS and $O(N)$ memory usage for 3D elliptic PDEs, and should therefore outperform SuperLU for most BISON simulations. STRUMPACK is also optional for PETSc and can be used with several different sparsity reordering packages.

We tested STRUMPACK on some representative Bison assessment cases on the Falcon high performance computing (HPC) cluster at Idaho National Laboratory (INL). In particular, we chose to benchmark STRUMPACK versus SuperLU using the USPWR 16x16 TSQ002 2D full rod simulation because that case has all major physics used in LWR simulations and includes a realistic power profile over the full service life of the rod. The USPWR 16x16 TSQ002 case has approximately 280,000 degrees of freedom (DOFs), or about 10k DOFs per core when run on a typical 2-socket Xeon workstation or single node in an HPC cluster. This loading is well within the range of published results that show SuperLU, STRUMPACK, and other packages scaling to well over 1M DOFs when used with 10k DOFs per core.

We found that STRUMPACK does indeed outperform SuperLU for these problems. For the benchmark case, STRUMPACK took roughly 40% less time to factorize than did SuperLU, while memory usage was comparable to SuperLU. We found the best results when ordering was completed by PARMETIS, but other ordering packages work nearly as well. In addition to providing better speed, STRUMPACK also exhibits better parallel strong scaling than SuperLU for this problem, which implies that it will perform progressively better on larger problems.

Nevertheless, preconditioning cost — even with full LU factorization — is not the dominant cost in many Bison simulations, and therefore, run time can be expected to decrease only modestly with a change of preconditioner. For the benchmark USPWR 16x16 case, preconditioning is less than 25% of the total run, and STRUMPACK reduced overall run time by approximately 12%. Further reduction of the preconditioning time is unlikely for runs of this size, although for very large 3D runs, we expect further improvement.

The use of the STRUMPACK solver option produced a significant impact on the converged solution behavior of the

TABLE II. REBEKA assessment case computed quantities and simulation metadata for the original, optimized, and automatic differentiation enabled runs.

| | Original | Optimized no AD | Optimized AD |
|---|---|---|---|
| average interior clad temperature | 927.48 K | 926.80 K | 926.80 K |
| max clad temp | 939.31 K | 938.59 K | 938.59 K |
| max hoop stress | 131.27 MPa | 132.27 MPa | 132.82 MPa |
| vonmises stress clad | 109.83 MPa | 109.75 MPa | 109.85 MPa |
| burst time | 366.31 s | 365.59 s | 365.59 s |
| physical memory use | 710 MB | 799 MB | 894 MB |
| total linear iterations | 18947 | 2949 | 603 |
| total nonlinear iterations | 8520 | 943 | 603 |
| residual compute time | 336.46 s | 50.24 s | 10.53 s |
| jacobian compute time | 58.03 s | 6.3 s | 34.84 s |
| simulation alive time | 504.77 s | 69.28 s | 56.16 s |

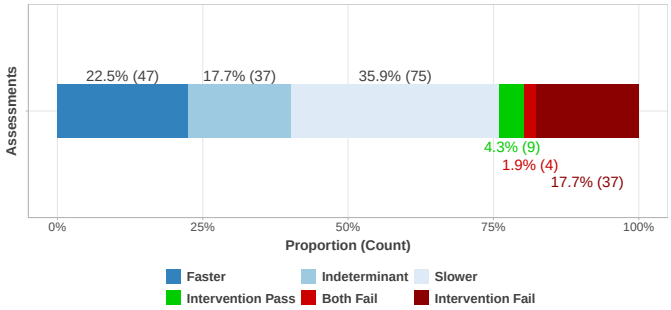


Fig. 3. Overview of the number of assessment cases influenced by switching to the STRUMPACK multi-frontal direct solver. 9 cases passed that had previously failed in the baseline cases. 4 cases failed in both the baseline and intervention cases. 37 cases failed due to the intervention. A total of 50 cases were not used during comparison.

Bison LWR assessment suite as shown in Figures 3 and 4. As in the algorithmic change discussed above, the STRUMPACK solver option allowed a set of assessment cases which had failed in the baseline behavior runs due to simulation timeout termination to pass: IFA-562 rod 15, IFA-562 rod 16, IFA-562 rod 17, IFA-636, and OSIRIS-J12.

Additionally the use of the STRUMPACK solver enabled an integral fuel rod which had failed in the baseline runs due to convergence failure, Calvert Cliffs BFM043 (tensor mechanics version), to successfully converge and solve. In the baseline behavior runs, this assessment case had consistently failed to converge during a short period of flat power immediately proceeding the last power adjustment in the power history. This significant improvement in the convergence behavior of this assessment case indicates that the STRUMPACK solver option may be a useful option to explore in Bison simulations of integral fuel rods which have difficultly converging with the traditional SuperLU solver.

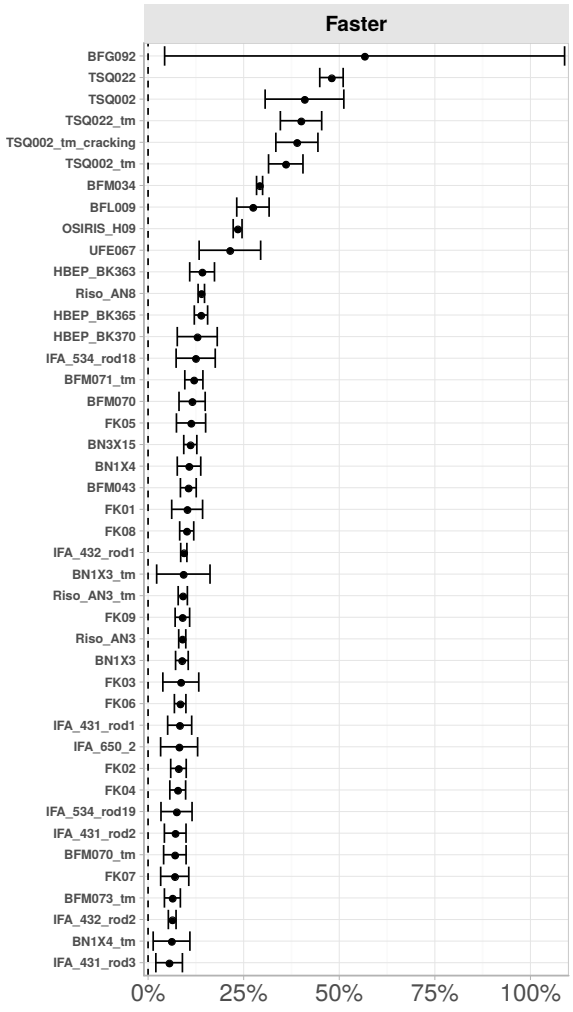


Fig. 4. Assessment cases significantly accelerated by switching to the STRUMPACK multi-frontal direct solver.

## CONCLUSIONS

Our general conclusion regarding the use of automatic differentiation is that models that take many linear iterations to converge will benefit most from the exact Jacobians provided by the automatic differentiation.

If a hand-coded Jacobian already exists and the impact of mesh displacements is low (small strain case), a small increase in linear iterations is likely to incur a smaller performance penalty than the cost of using automatic differentiation. If a hand-coded Jacobian does not yet exist, for example in a newly developed model, the developer time and effort of deriving such a Jacobian has to be carefully weighed against the computational pay-off. If large deformation occurs in the target model, no hand-coded Jacobian can be fully exact within the MOOSE framework at this time. Off-diagonal contributions of test function gradient derivatives, quadrature point weight derivatives, coordinate system factors (in case of radial coordinate systems), and element normal vector derivatives with respect to the mesh displacements are provided by only the automatic differentiation system.

The STRUMPACK solver is an exciting new solver that we recommend users to try on their models. Applying this intervention requires only a PETSc option change and it can provide up to a 50% reduction in runtime.

The vast majority of Bison assessment cases uses thermal and mechanical contact to connect the fuel and cladding domains. A transition of contact to a rewritten mortar based system is in progress, with framework level components under review for integration at the time of writing of this report. Contact currently is a robustness and performance bottleneck. We expect a major payoff of our work once we transition the Bison models over to the new AD based contact system and remove the final remaining bottleneck.

## ACKNOWLEDGMENTS

This work was sponsored by the U.S. Department of Energy, Office of Nuclear Energy, Consortium for Advanced Simulation of LWRs (CASL) and Advanced Fuels Campaign (AFC) programs.

This manuscript has been authored by Battelle Energy Alliance, LLC under Contract No. DE-AC07-05ID14517 with the U.S. Department of Energy.

## REFERENCES

1. R. L. WILLIAMSON, J. D. HALES, S. R. NOVASCONE, M. R. TONKS, D. R. GASTON, C. J. PERMANN, D. ANDRS, and R. C. MARTINEAU, "Multidimensional Multiphysics Simulation of Nuclear Fuel Behavior," *Journal of Nuclear Materials*, **423**, 149–163 (2012).
2. C. J. PERMANN, D. R. GASTON, D. ANDRS, R. W. CARLSEN, F. KONG, A. D. LINDSAY, J. M. MILLER, J. W. PETERSON, A. E. SLAUGHTER, R. H. STOGNER, and R. C. MARTINEAU, "MOOSE: Enabling Massively Parallel Multiphysics Simulation," (2019).
3. D. R. GASTON, C. J. PERMANN, J. W. PETERSON, A. E. SLAUGHTER, D. ANDRŠ, Y. WANG, M. P. SHORT, D. M. PEREZ, M. R. TONKS, J. ORTENSI, L. ZOU, and R. C. MARTINEAU, "Physics-based multiscale coupling for full core nuclear reactor simulation," *Annals of Nuclear Energy*, **84**, 45–54 (2015).
4. D. A. KNOLL and D. E. KEYES, "Jacobian-Free Newton-Krylov Methods: a Survey of Approaches and Applications," *SIAM Journal on Scientific and Statistical Computing*, **193**, *2*, 357–397 (2004).
5. R. STOGNER and A. LINDSAY, "MetaPhysicL," `https://github.com/roystgnr/MetaPhysicL` (2019).
6. D. HAGRMAN and G. REYMANN, "MATPRO-Version 11: a handbook of materials properties for use in the analysis of light water reactor fuel rod behavior," Tech. Rep. NUREG/CR-0497, Idaho National Engineering Laboratory (February 1979).
7. "STRUMPACK – STRUctured Matrices PACKage, Version 00," (12 2014).
8. F. J. ERBACHER, H. J. NEITZEL, H. ROSINGER, H. SCHMIDT, and K. WIEHR, "Burst criterion of Zircaloy fuel claddings in a loss-of-coolant accident," in "Zirconium in the Nuclear Industry, Fifth Conference, ASTM STP 754, D.G. Franklin Ed.", American Society for Testing and Materials (1982), pp. 271–283.