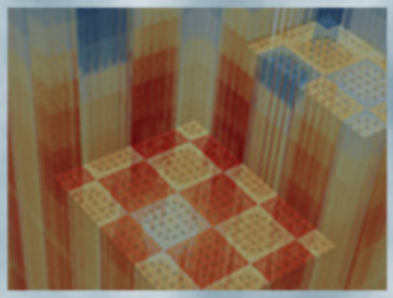
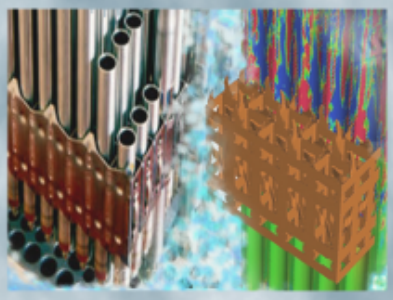
CASL-U-2019-1865-000





Bison Robustness and Performance Improvements L2:FMC.FUEL.P19.02

Daniel Schwen, INL
Benjamin W. Spencer, INL
Stephanie A. Pitts, INL
Dylan J. McDowell, INL
Shane Stafford, Southwestern Scientific

June 30, 2019

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0	06/30/2019	All	Initial Release

Document pages that are:

Export Controlled _____NONE_____

IP/Proprietary/NDA Controlled_____NONE_____

Sensitive Controlled_____NONE_____

Requested Distribution:

To: FMC FA Lead

Copy: CASL PM

Bison Robustness and Performance Improvements

Daniel Schwen, Benjamin W. Spencer, Stephanie A. Pitts, Dylan J. McDowell
Department of Fuel Modeling and Simulation
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840

Shane Stafford
Southwestern Scientific Ltd. Co.
3 Metate Dr.
Sandia Park, NM 87047-8508

June 30, 2019

1 Introduction

In last year's Bison robustness report [1] we have shown that improving the Jacobian matrix can drastically increase solution convergence rates. The development of an automatic differentiation (AD) capability in the MOOSE framework [2, 3] has created the possibility of having exact Jacobian matrices for all of our material models. A large focus in this year's report is the application of AD to MOOSE physics modules, such as the tensor mechanics module, and the application to Bison fuel and cladding material models. A second main focus in this year's report is the investigation of a set of algorithmic solver changes to improve the robustness and performance of Bison LWR simulations.

Beyond providing increased solver robustness, the ability to solve strongly coupled multiphysics problems, and solving large deformation problems with ease, these automatically generated perfect Jacobians also drastically reduce development time for new models. In the models we have converted to AD, Jacobian computation made up the majority of the code base. With this code removed both readability and maintainability of the MOOSE/Bison code base are improved.

The work documented here significantly improved the robustness of the Bison LWR assessment case suite. Through algorithmic changes in solver infrastructure, such as the application of predictors and improvements in timestepping, we were able to make tangible performance improvements. We document these improvements using a a new standard set of metrics that we collect in all assessment cases, and a suite of statistical analysis tools developed for this report.

We note that the work in this report is an important milestone in the improvement of Bison robustness and performance. This work is nonetheless part of an ongoing effort. Some benefits of this work will only be fully realized when additional necessary future work is incorporated into MOOSE/Bison. This work includes automatic variable scaling and a new mortar based thermo-mechanical contact system.

2 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 $x + \epsilon\vec{x}'$, where ϵ is an infinitesimal element with $\epsilon^2 = 0$, and $\vec{x}'$ is the vector of derivatives. Addition and multiplication can be written as

$$(x + \epsilon\vec{x}') + (y + \epsilon\vec{y}') = (x + y) + \epsilon(\vec{x}' + \vec{x}') \tag{1}$$

$$(x + \epsilon\vec{x}')(y + \epsilon\vec{y}') = (xy) + \epsilon(x\vec{y}' + \vec{x}'y) \tag{2}$$

$$f(x + \epsilon\vec{x}') = f(x) + \epsilon f'(x)x'. \tag{3}$$

Note that the quadratic term in Equation 2 drops out and the coefficients on the ϵ 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 ith 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 ith DOF.

Through the MOOSE framework we use the *MetaPhysicL* [4] 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. There is room for future algorithmic performance improvements in the forward AD implemented by MetaPhysicL. Currently the derivative vectors are sized at compile time, setting the computational complexity of AD independently of the problem complexity. It is planned to look into sparse and dynamically (run time) sized derivative vectors which have the potential of reducing the computational overhead from AD.

2.0.1 Application to MOOSE and Bison

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.

The prime target for AD conversion was the tensor mechanics module [5], for which we created 33 material classes

- `ADComputeStrainBase`
- `ADComputeVariableIsotropicElasticityTensor`
- `ADComputeRSphericalFiniteStrain`
- `ADComputeAxisymmetricRZSmallStrain`
- `ADComputeAxisymmetricRZFiniteStrain`
- `ADComputeIncrementalStrainBase`
- `ADComputeFiniteStrainElasticStress`
- `ADCompute2DFiniteStrain`
- `ADComputeRSphericalIncrementalStrain`
- `ADCompute2DSmallStrain`
- `ADComputeAxisymmetricRZIncrementalStrain`
- `ADCompute1DFiniteStrain`
- `ADCompute1DSmallStrain`
- `ADSingleVariableReturnMappingSolution`
- `ADPowerLawCreepStressUpdate`
- `ADCompute2DIncrementalStrain`
- `ADComputeThermalExpansionEigenstrainBase`
- `ADCompute1DIncrementalStrain`
- `ADComputeFiniteStrain`
- `ADComputeEigenstrain`
- `ADComputeSmallStrain`
- `ADComputeRSphericalSmallStrain`
- `ADComputeLinearElasticStress`
- `ADComputeMultipleInelasticStress`
- `ADComputeGreenLagrangeStrain`
- `ADComputeEigenstrainBase`
- `ADComputeStressBase`
- `ADComputeThermalExpansionEigenstrain`
- `ADRadialReturnCreepStressUpdateBase`
- `ADStressUpdateBase`
- `ADComputeElasticityTensorBase`
- `ADComputeIncrementalSmallStrain`

- ADRadialReturnStressUpdate

4 kernels

- ADStressDivergenceRZTensors
- ADStressDivergenceTensors
- ADStressDivergenceRSphericalTensors
- ADGravity

and 1 boundary condition (ADPressure).

In Bison so far 14 materials

- ADUPuZrFastNeutronFlux
- ADUPuZrThermal
- ADZryCreepUpdateBase
- ADZryThermalExpansionMATPROEigenstrain
- ADUPuZrCreepUpdate
- ADZryCreepLOCAErbacherLimbackHoppeUpdate
- ADHT9CreepUpdate
- ADUPuZrElasticityTensor
- ADZryCreepLimbackHoppeUpdate
- ADUPuZrBurnup
- ADThermalZry
- ADUPuZrFissionRate
- ADUPuZrFissionGasRelease
- ADHT9Thermal

1 kernel (ADFissionRateHeatSource) and 1 boundary condition (ADFissionRateHeatSource) have been converted. The tensor mechanics classes implement AD versions of small and large strain in total and incremental forms for cartesian, cylindrical, and spherical coordinate systems. The Bison classes cover a variety of fuel and clad thermo-mechanical models.

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.

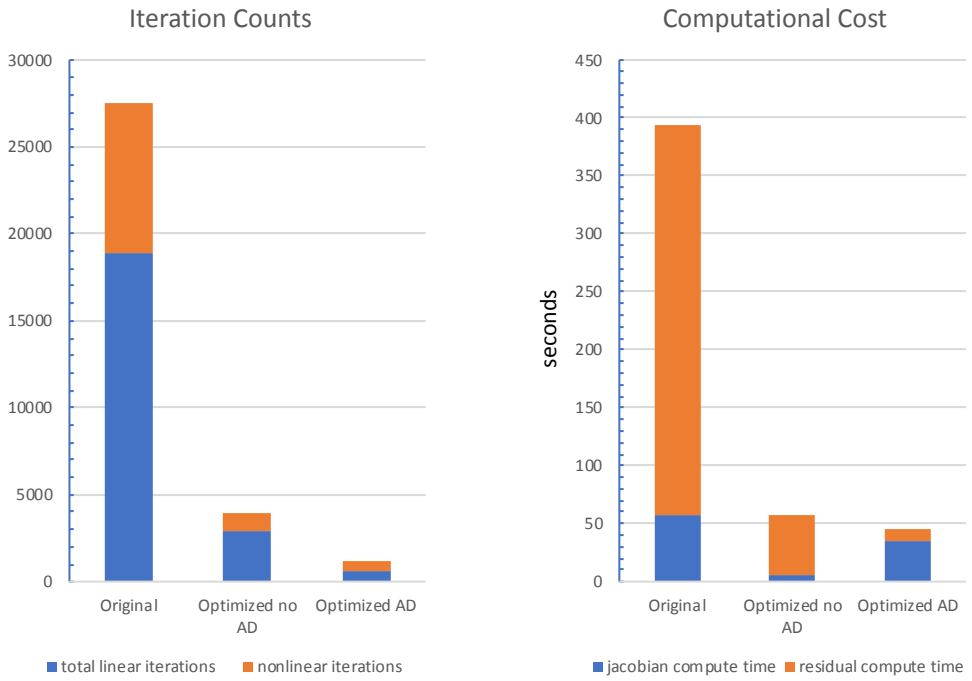


Figure 1: Iteration counts (left) and walltime (right) the REBEKA assessment case with original, optimized, and automatic differentiation enabled runs.

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.

2.1 Impacts on an Assessment Case

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.

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 [7].

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 1 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.

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 1. We confirmed that the calculated stresses, temperatures,

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

Table 1: 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

and predicted burst times are within a fraction of a percent for all runs.

2.1.1 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 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.

3 Performance Metric Output and Analysis

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. Unlike the implementation of automatic differentiation and the detailed study of the new code impact on specific cases, we applied the algorithm changes, discussed in Section 4, to the complete Bison LWR assessment case suite.

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. Among these Bison LWR assessment cases there are 53 assessment cases for which both a solid mechanics module system input file and a tensor mechanics module system input file exists. These duplicate assessment case input files are retained in the Bison assessment suite as part of the process of migrating the Bison

code base from the older solid mechanics module system to the newer tensor mechanics module system. The tensor mechanics module system offers more flexibility to include anisotropy, better correspondence to mathematical notation for formulas, and more integration with other MOOSE modules. Additionally the tensor mechanics module enables better control over which mechanics materials are included in the simulation by separating each mechanics physics component into individual material models. The presence of the 53 solid mechanics and tensor mechanics assessment case input file pairs affords the opportunity to evaluate the impact of each algorithmic change on both mechanics module systems. This additional evaluation of the four algorithmic changes allows us to ensure the continued robustness of the Bison simulations as we finalized the transition of the code base to the tensor mechanics system.

In this section we discuss the selection and implementation of these performance metrics, in a standard manner across the complete assessment case suite, and the analysis of these metrics to determine the impact of the algorithmic changes. The selection, implementation, and impact analysis of these algorithmic changes are discussed in Section 4.

3.1 Performance metric output

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 2. 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.

Table 2: 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

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.

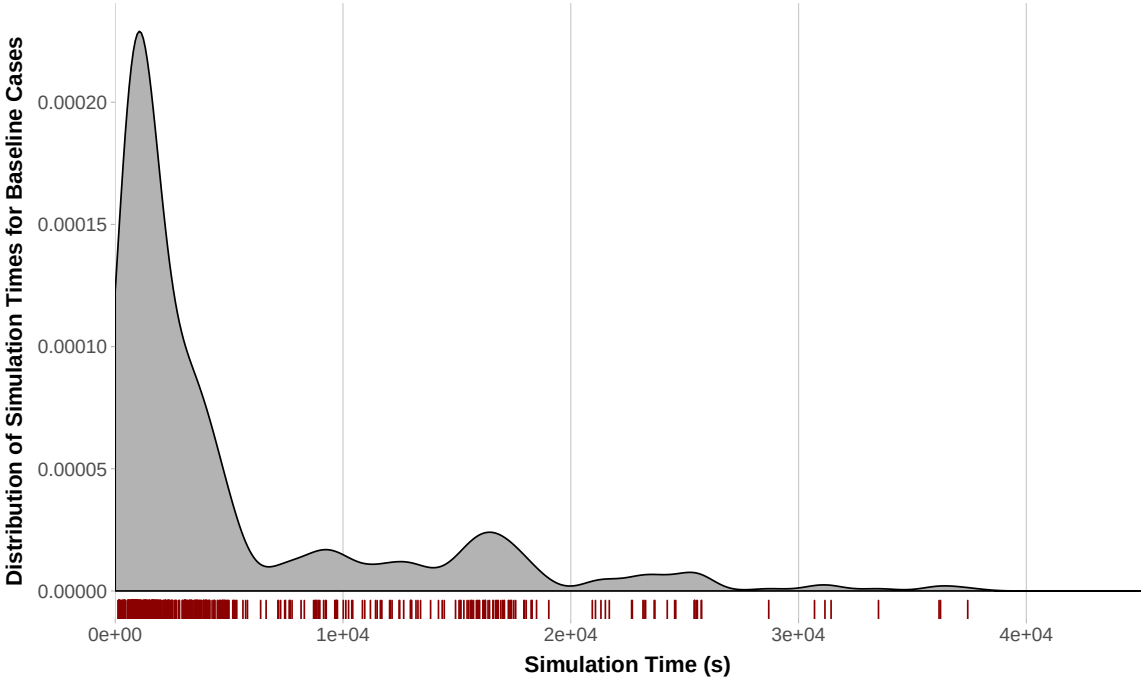


Figure 2: Rug plot detailing the the distribution of simulation time of the baseline cases.

3.2 Analysis of performance metrics

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 2 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 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^{th} case in the assessment suite, $\bar{y}$ represents the mean run time of the treatment group for the i^{th} case, and $\bar{x}$ represents the mean run time of the baseline cases (i.e. control group) for its i^{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.

4 Algorithmic Changes for Solution and Timestepping

The algorithmic changes presented in this section involve code modifications focused on specific aspects of the Bison simulation. In contrast to the material model code design and implementation work completed with the automatic differentiation discussed in Section 2, the algorithmic changes discussed here focused on modification to the simulation executioner, including the amount of boundary condition function change permitted in a single time step size, the finite element analysis (FEA) solver options used, and the prediction of an initial guess for the solution vector at next time step. The algorithmic changes considered here also included modifications to the method used to calculate thermal contact between the clad and fuel. Each modification or *intervention* presented here is enabled by an optional parameter. We employ statistical analysis of a set of thrice repeated runs of the Bison assessment suite to determine the impact of each intervention using the performance metrics discussed in Section 3. These cases were all run on the Lemhi high performance computing cluster at Idaho National Laboratory using PETSc 3.10.5 except as otherwise noted for preliminary benchmarking.

In total, 209 individual Bison LWR assessment cases were run, each three times per optional algorithmic change, to collect robustness performance data for each of the four algorithmic changes against a baseline Bison performance state. An assessment case was considered to pass reliably if the simulation completed for all three runs. Cases which failed repeatedly or intermittently were considered as failed for the purpose of this analysis.

The baseline behavior of the Bison assessment suite was established first by running the LWR suite three times without any of the optional algorithmic changes discussed below. Of the 209 total assessment cases included in this study, 196 cases were considered to pass and 13 cases were considered failed as the baseline behavior. The failures in the baseline behavior runs were caused by a mixture of convergence errors and timeouts: a situation in which the assessment case runtime would exceed an upper limit and thus is terminated before the simulation completes. While there are multiple potential reasons for a simulation timeout termination, the occurrence of these types of simulation failures highlights the importance of reducing simulation runtime in increasing Bison robustness.

4.1 Improved procedure for limiting power function change in a time step

In standalone Bison calculations the reactor power curves are an input provided as time dependent functions. These functions are most often implemented as piecewise linear function generated from tabulated data. Users provide the power for selected times and MOOSE performs a linear interpolation at all other times. The time derivative of these functions can change at, and only at, the time nodes through which the function is specified. At strong changes in the derivative, which may happen during quick power changes such as shutdowns and start-ups, non-linearities are introduced (as shown in Figure 3). The default iteration adaptive Bison timestepper can inspect an external driving force function, such as the reactor power, and can enforce timestep boundaries to coincide with the time nodes of the specified function. That is if a timestep would cause the simulation time to advance past a function time node, the timestep will be cut so that the simulation advances exactly to the function time node.

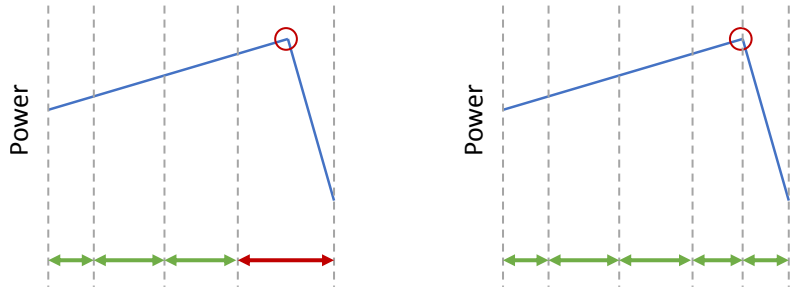


Figure 3: Timestep size in Bison can be controlled through a function, such as the reactor power curve. By default the timstepper avoids stepping over time nodes in tabulated and linearly interpolated functions.

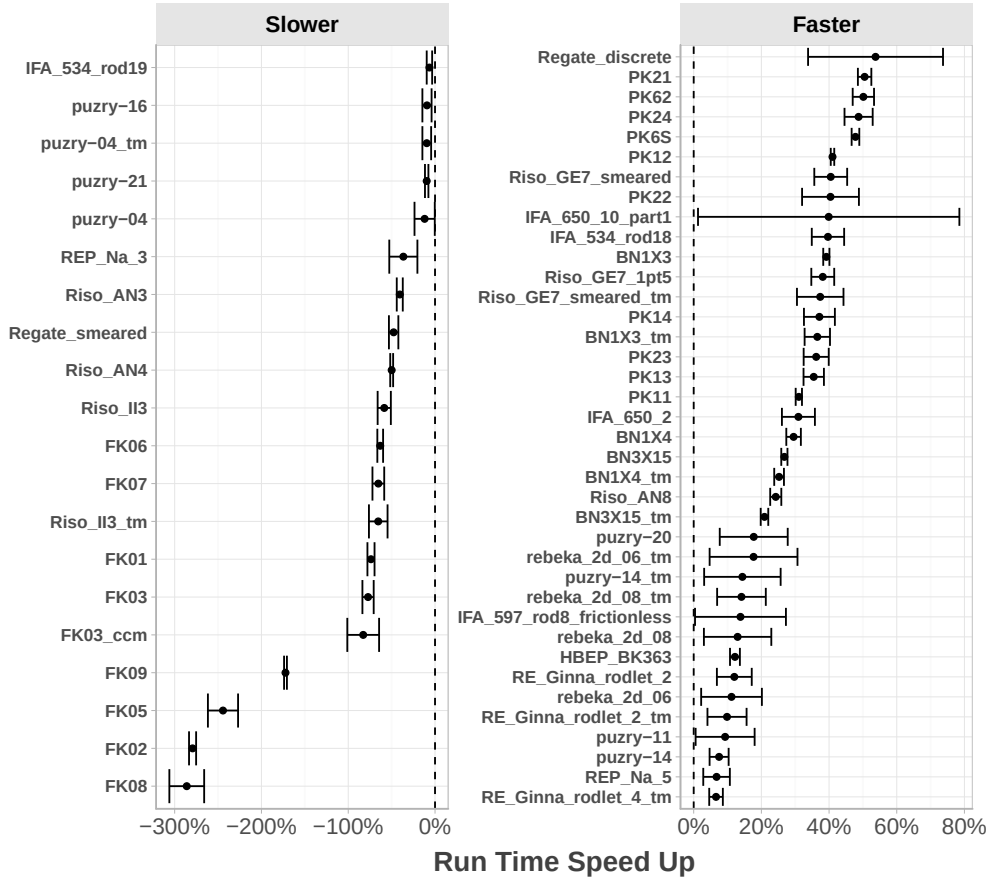


Figure 4: Assessment cases significantly accelerated or slowed down by applying the function change threshold timstepper intervention.

While this approach mitigates solver issues caused by rapid function changes, it may lead to overzealous reduction of the simulation timestep, especially if the provided function is tabulated with very small time intervals. We implemented a new way of reducing the timestep based on a maximum change of the function value in the interval given by the current timestep. This approach permits stepping over multiple time nodes of the power function as long as the power function changes only within a specified power interval.

The results of applying the timstepper intervention are shown in Figures 4 and 5. The improved limiting function change procedure option enabled seven cases that failed in the baseline behavior runs to pass. These baseline failures had occurred because of intermittent simulation timeout terminations: each failed assessment case in this group did not fail to complete because of a timeout termination in every baseline run. The assessment cases which passed with the

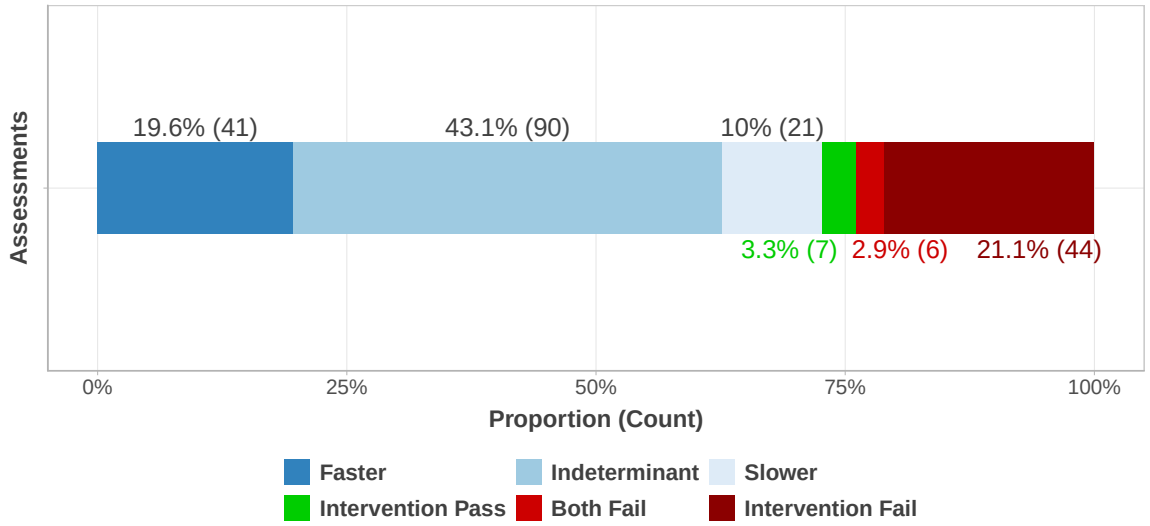


Figure 5: Overview of the number of assessment cases influenced by applying the function change threshold timstepper intervention. 7 cases passed that had previously failed in the baseline cases. 6 cases failed in both the baseline and intervention cases. 44 cases failed due to the intervention. A total of 57 cases were not used during comparison.

introduction of the limiting function procedure change investigated here include integral fuel rods and rodlets irradiated for differing numbers of years: IFA-515 Rod A1, IFA-562 rod 15, IFA-562 rod 16, IFA-562 rod 17, and IFA-636. We note that two additional cladding-only rodlets which failed in the baseline behavior runs did pass with the use of the improved function change procedure. These two assessment case input files, PUZRY-06 in both the solid mechanics module system and the tensor mechanics module system, failed in the baseline behavior runs because of an error associated with the job scheduler. As such the cause of the failure for these two assessment cases is likely unassociated with changes to the Bison code base and input files. We expect that the five assessment cases which were failing because of simulation timeout terminations do pass here because of the changes introduced by the new timestepper procedure. We hypothesize that the improved limiting function change procedure enabled the Bison simulation to more quickly navigate the complex power history function changes associated with these assessment cases. As a result, the simulation runtime of these assessment cases may have been decreased sufficiently to allow these assessment cases to reliably pass under the timeout limits.

Overall the modified timestepper does improve the execution time of a small set of about twenty assessment cases substantially with a runtime reduction of up to 50%.

4.2 Use of a multi-frontal direct solver

SuperLU has been the suggested package for preconditioning Bison simulations for several years now because it outperforms other preconditioners such as algebraic multi-grid (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.

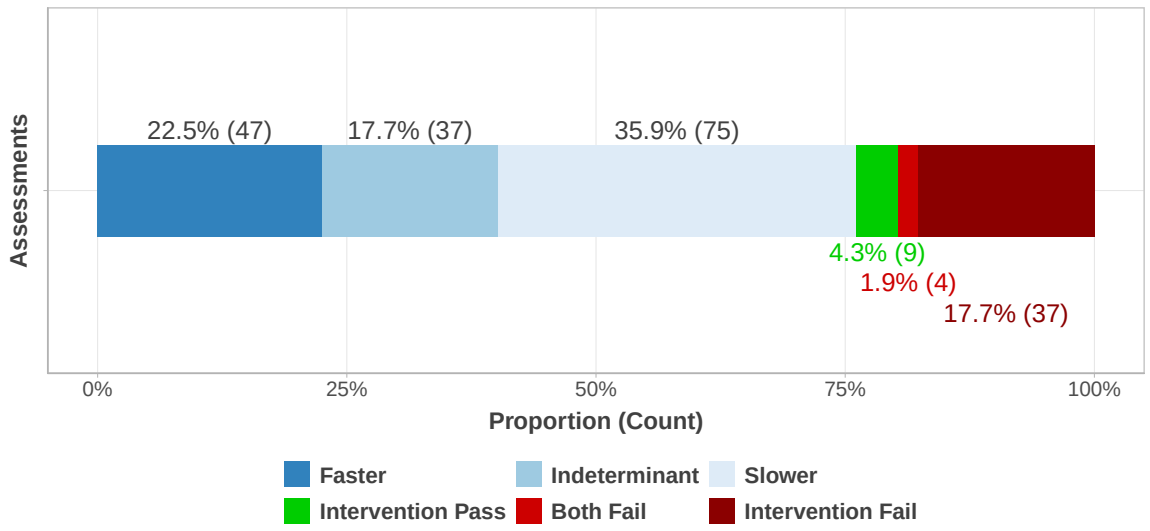


Figure 6: Overview of the number of assessment cases influenced by switching to the STRUMPACK multi-frontal direct solver. 9 cases passed that had previously failed in the baseline cases. 4 cases failed in both the baseline and intervention cases. 37 cases failed due to the intervention. A total of 50 cases were not used during comparison.

The use of the STRUMPACK solver option produced a significant impact on the converged solution behavior of the Bison LWR assessment suite as shown in Figures 6 and 7. 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

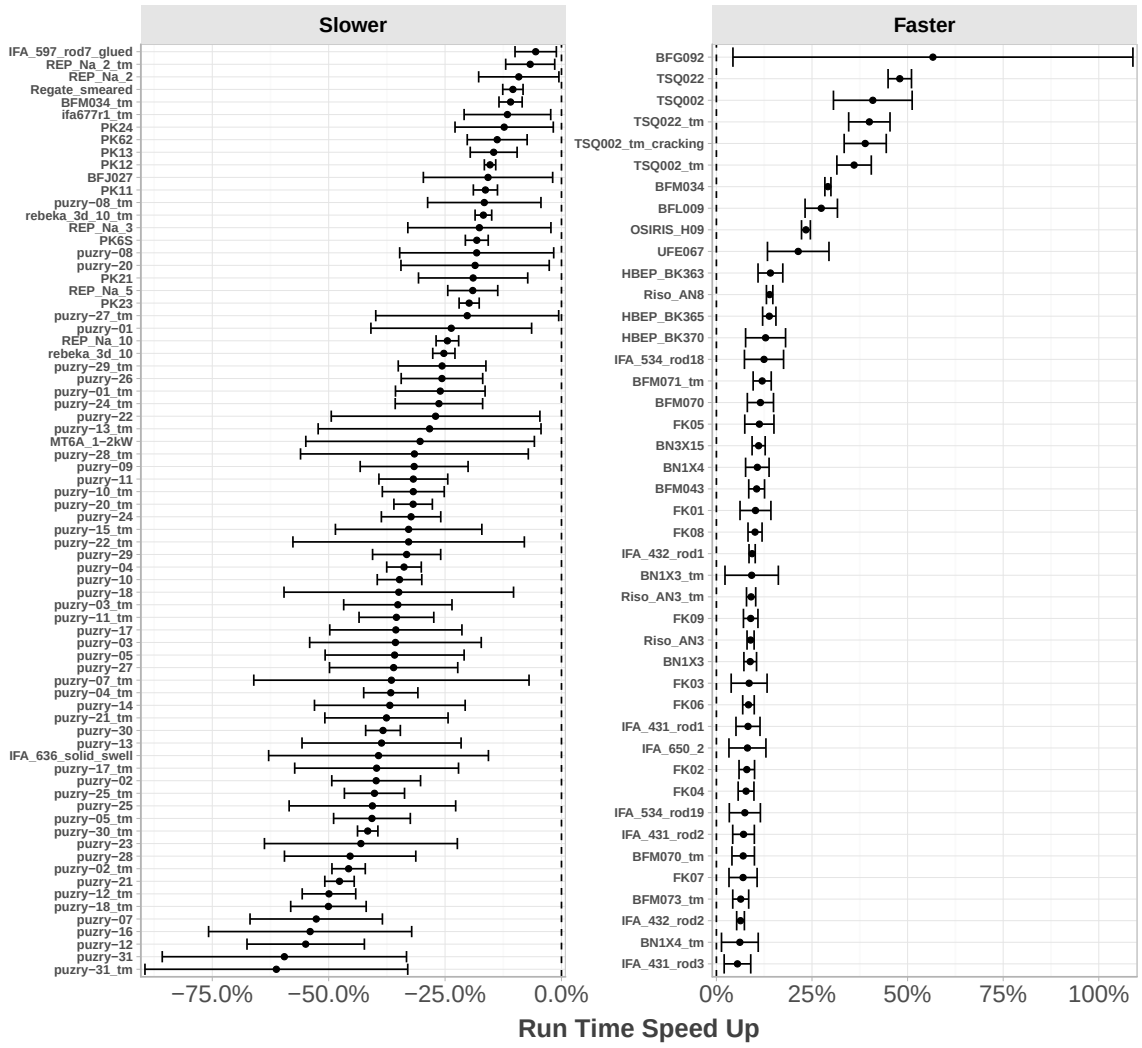


Figure 7: Assessment cases significantly accelerated by switching to the STRUMPACK multi-frontal direct solver.

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.

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.

4.3 Use of an adaptive predictor

In a transient simulation the initial guess for the solution of a new timestep is taken from the solution of the previous time step by default. The MOOSE Predictor system can be used to customize this initial guess. A predictor within the MOOSE context is an object that provides an initial guess for the solution at a given timestep. Currently two predictor classes exist in MOOSE: the linear simple predictor and the second order Adams-Bashforth predictor. The simple predictor goes into effect after the second timestep. It uses the solution vectors of the previous two timesteps and linearly extrapolates all DOF values to the current time. This predictor is particularly effective if the solution progresses in a linear way, such as displacements under linear loading or linear temperature increases. MOOSE uses a scaling value that ranges from zero to one, to perform a weighted average between the modified and unmodified solution vectors. A value of one results in using the extrapolated solution vector, and a value of zero in discarding the prediction entirely.

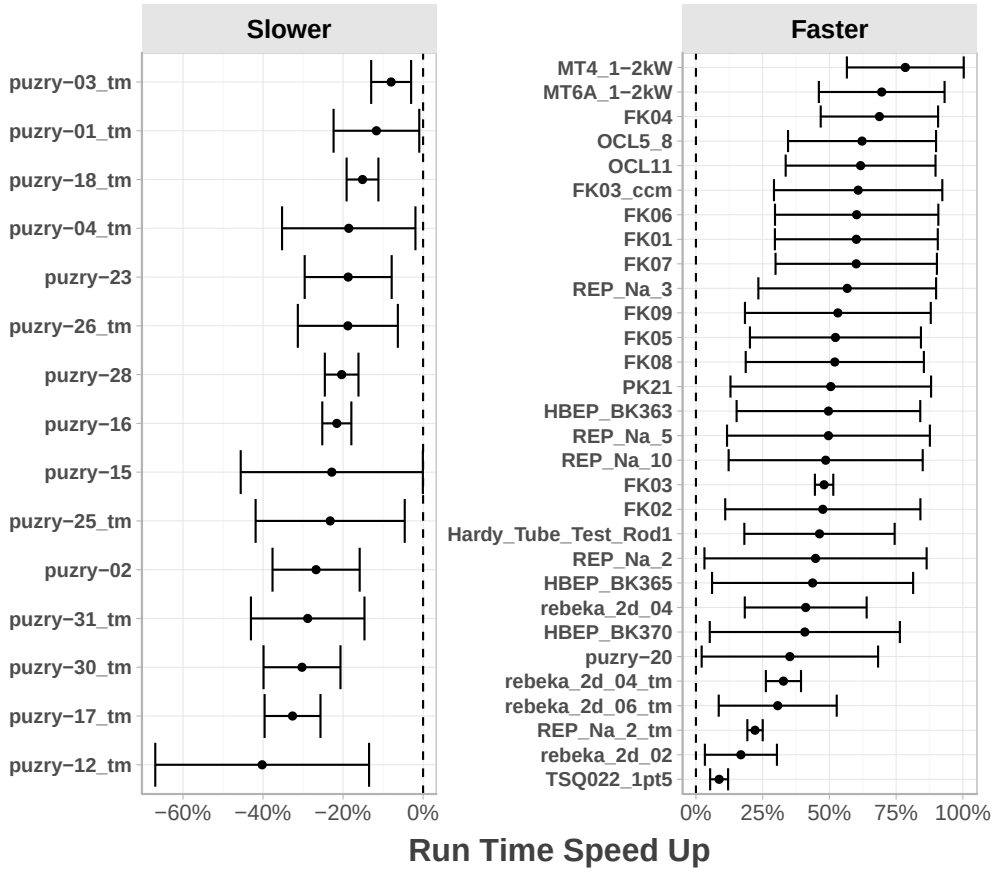


Figure 8: Assessment cases significantly accelerated or slowed down by adding the stock MOOSE simple predictor with a scale factor of 1.0.

We first apply the regular stock MOOSE simple predictor with a scale factor of 1.0 to all assessment cases. The results are shown in Figures 8 and 9. While a number of assessment cases see a significant reduction in runtime we also observe a high failure rate. Further study is required to investigate the effect of lower scale factors.

In transient simulations linear and non-linear behavior in the solution progression can appear during different stages. We propose an adaptive predictor to increase the applicability of the MOOSE predictor system. We perform a residual norm evaluation before and after applying the predictor. The unmodified solution vector from the previous time step is saved off and can be rolled back in case the solution vector updated by the predictor is a worse initial guess, i.e. has a higher residual norm than the unmodified solution.

Future modifications may include automatic determination of the optimum scaling value and the detection of me-

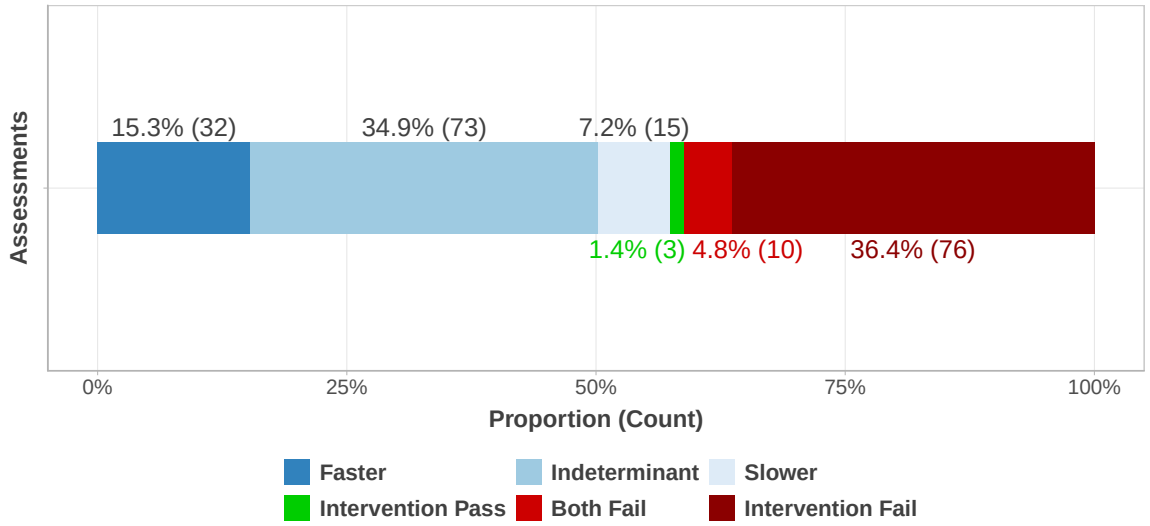


Figure 9: Overview of the number of assessment cases influenced by adding the stock MOOSE simple predictor with a scale factor of 1.0. 3 cases passed that had previously failed in the baseline cases. 10 cases failed in both the baseline and intervention cases. 76 cases failed due to the intervention. A total of 89 cases were not used during comparison.

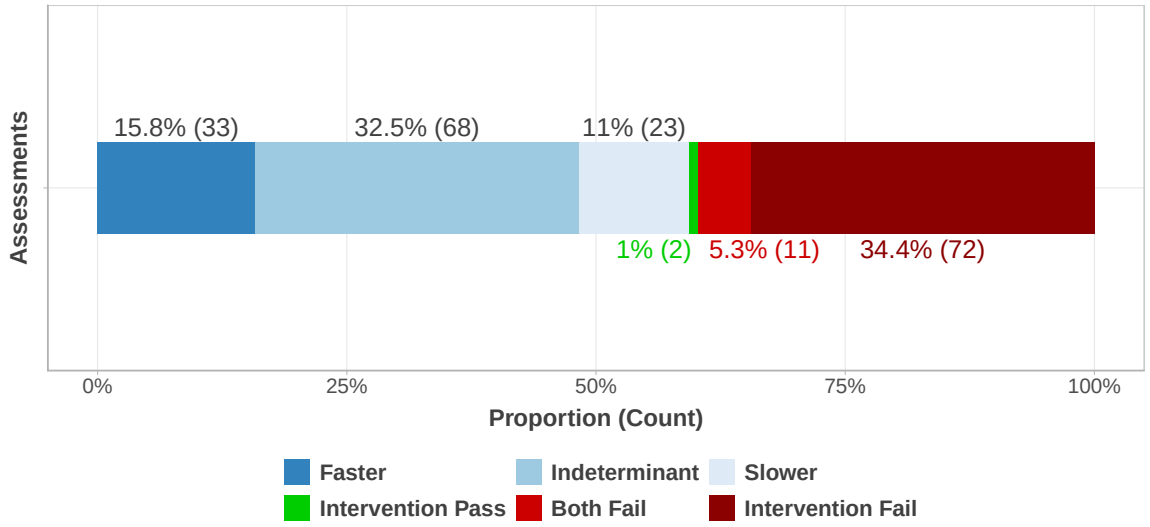


Figure 10: Overview of the number of assessment cases influenced by adding an adaptive predictor. 2 cases passed that had previously failed in the baseline cases. 11 cases failed in both the baseline and intervention cases. 72 cases failed due to the intervention. A total of 85 cases were not used during comparison.

chanical contact events. Mechanical contact presents a unique challenge for solution prediction, as the predictor may discretely change the solution state from out-of-contact to in-contact (or vice versa). The in-contact or penetration state is associated with a very large residual norm, which is likely to lead to the rejection of the solution prediction. However, the lower residual out-of-contact guess is not necessarily improving the solver performance or stability, because a transition to in-contact may now occur during the solve, leading to a discontinuity in the residual. Detection of mechanical contact could help circumvent this issue and force the use of the predicted solution. This area is a subject of ongoing work.

We next applied the smart predictor intervention, which applies the prediction only whenever it results in a lowered residual vector norm. The results are presented in Figures 10 and 11. Compared to the stock predictor the adaptive predictor algorithmic change had only a limited impact on the the test runtimes and the pass rate of the assessment cases which failed in the baseline behavior runs. The only assessment cases which passed with this optional change are the two cladding only rodlets: PUZRY-06 in both the solid mechanics module system and the tensor mechanics module system. Since these failures occurred because of a HPC scheduler error in the baseline runs, the passing behavior of these two cases are likely unrelated to the use of the optional adaptive predictor.

The adaptive predictor at this point needs further research and is not a significant improvement over the existing MOOSE predictors that are applied at every timestep. Nonetheless we have shown that a significant number of assessment cases can be substantially accelerated by using the existing MOOSE predictors.

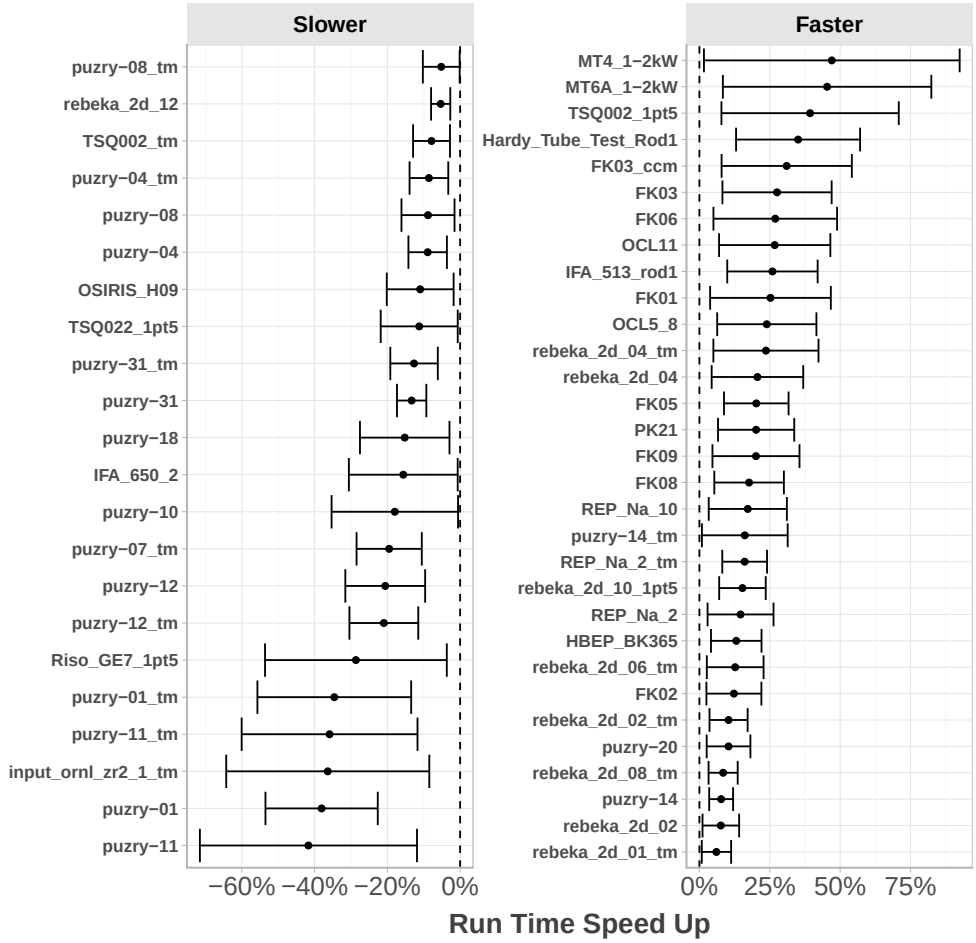


Figure 11: Assessment cases significantly accelerated or slowed down by adding an adaptive predictor that only goes into affect whenever the prediction lowers the residual vector norm.

4.4 Linearization of gap conductance model

In Bison simulations the gap between the fuel and clad domains is not meshed. Heat transfer through this unmeshed region is modeled using a thermal contact model. Geometric search algorithms determine the width of the gap, and through boundary conditions (and in the future mortar based constraints) the heat flux across the gap is modeled. The conductive component of the heat transfer through a gas filled gap, such as the fuel/clad gap in light water reactor (LWR) fuel pins follows the highly non-linear and asymptotic $\frac{k}{L}$ law, where k is the thermal conductivity of the gap and L is the gap width. During LWR operation the fuel/clad gap closes due to thermal expansion and fuel swelling. This causes the gap length L to approach zero and the gap conductance to diverge to infinity, presenting numerical challenges for the solver.

We approached this issue by specifying a lower bound L_{min} for the gap length below which we transition to a Taylor expansion of the gap conductance. At the zeroth order we obtain a small but constant value for gap lengths below L_{min}. The zeroth order approximation has the advantage of a vanishing derivative with respect to the displacement variables on either side of the gap. This vanishing derivative makes a correct Jacobian trivial to implement. The zeroth order approach is physically justifiable as long as the constant conductance, corresponding to a small L_{min}, is large compared to the conductance of the fuel and clad material. In this case the temperature step at the closed gap interface is negligible.

A first order (linear) Taylor expansion of the gap conductance was introduced to ensure differentiability of the gap conductance at the L_{min} point. A linear model also allowes for a larger choice of L_{min} and retains the thermal feedback effect of the closing gap. This feedback is important as a closing gap leads to faster heat removal from the fuel, which can halt or even reverse the dimensional change of the fuel, leading to a gap opening. However this reverse feedback makes the solve numerically difficult, leading to oscillations in the gap width.

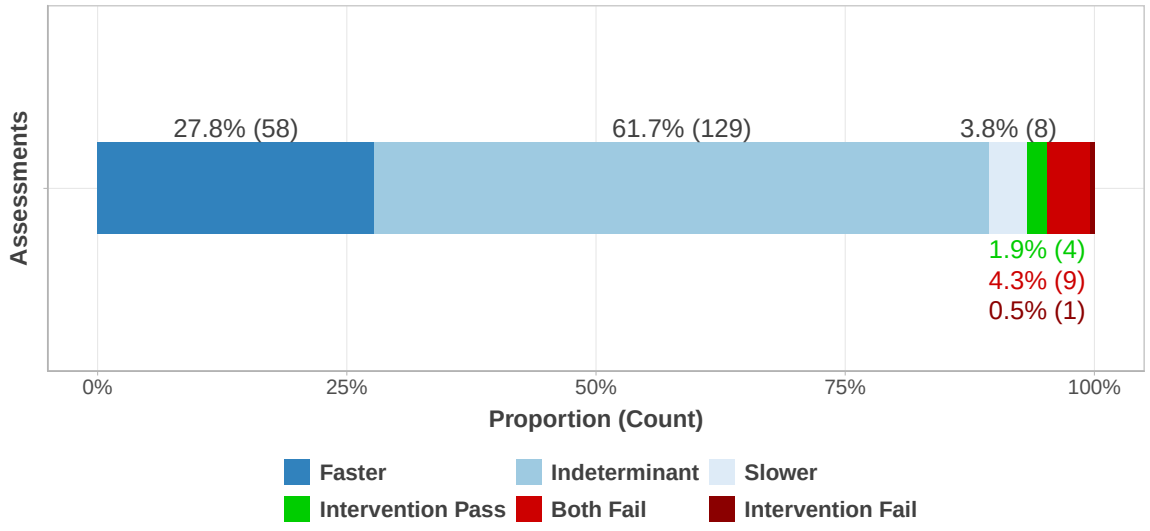


Figure 12: Overview of the number of assessment cases influenced by the linearized gap conductance intervention. 4 cases passed that had previously failed in the baseline cases. 9 cases failed in both the baseline and intervention cases. 1 cases failed due to the intervention. A total of 14 cases were not used during comparison.

The result of the linearized gap conductance intervention are shown in Figures 12 and 13. The gap conductance linearized model enabled four assessment cases which had failed in the baseline behavior runs to pass: IFA-515 Rod A1, OSIRIS-J12, and PUZRY-06 (both solid mechanics and tensor mechanics input files). The first two assessment cases failed because of intermittent simulation timeout terminations. As noted before, the two PUZRY assessment case

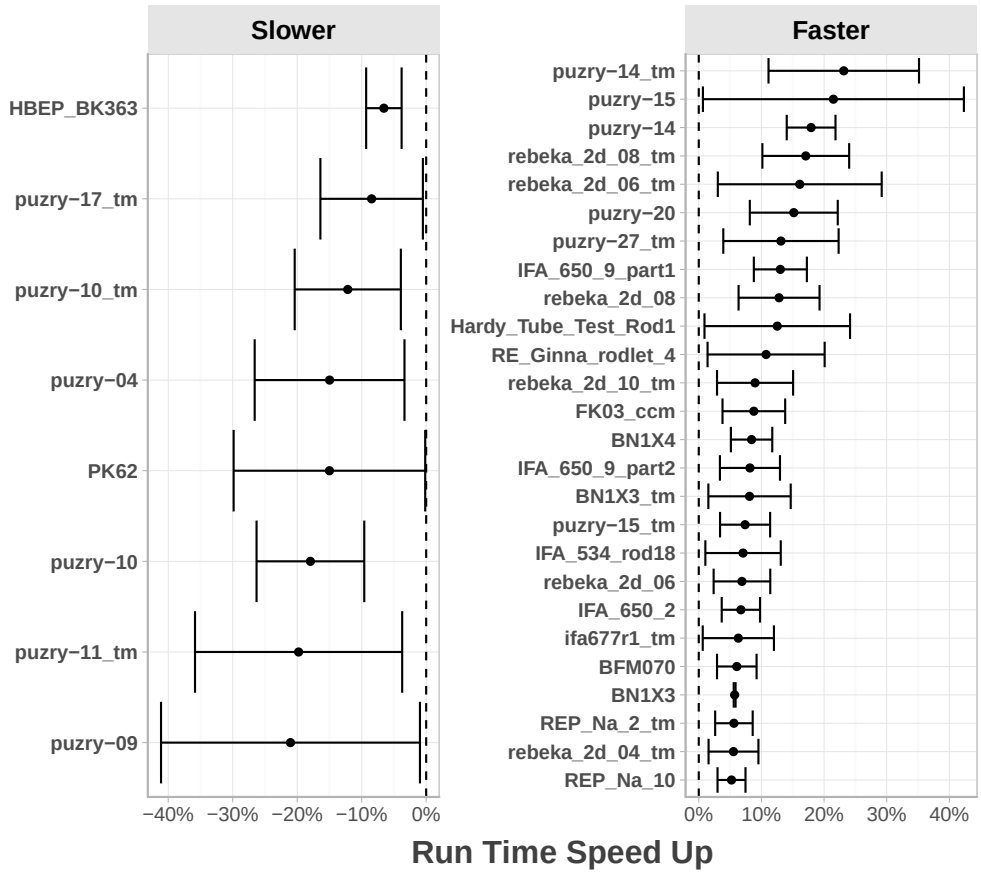


Figure 13: Assessment cases significantly accelerated or slowed down by applying the linearized gap conductance. The relative change in execution time is computed from three independent executions of the assessment suite and compared to a common baseline case comprising seven independent executions of the assessment suite.

current passes are likely unrelated to changes in the Bison code base or input files. For the assessment cases which experienced simulation timeouts, we hypothesize that the the linearizing modification to the gap conductance model maybe have reduced or stabilized the simulation runtime such that these cases could reliably pass under the time limit.

The linearized gap conductance intervention has a modest impact on the execution speed of a subset of Bison assessment cases, providing a speed up of up to 20%.

4.5 Combined effects of all algorithmic changes

We conducted a preliminary investigation into the combined effect of the four different algorithmic changes by simultaneously applying all four interventions in the Bison LWR assessment suite. An analysis of the impact of the simultaneous application of all four interventions is shown in Figures 14 and 15. These results demonstrate that the four interventions we have considered here do not interact in a linear fashion: simulation speedup gains from the individual algorithmic changes do not consistently correspond to larger simulation runtime speedups in the simultaneous combination runs.

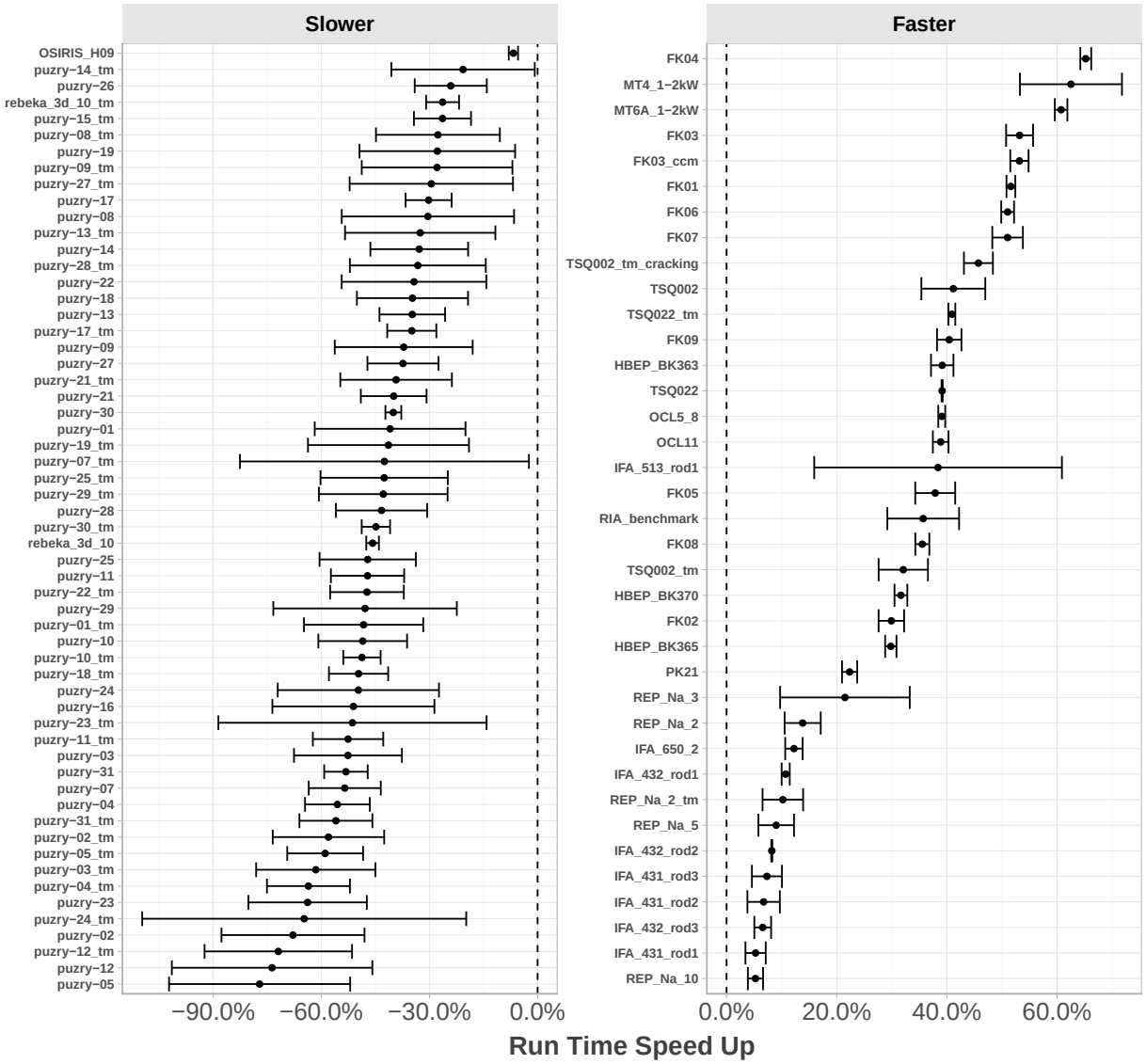


Figure 14: Assessment cases significantly accelerated by applying all four algorithmic interventions simultaneously.

The use of all four optional algorithmic changes enables only three assessment cases which had failed in the baseline runs to pass. These cases include a single integral fuel rod, IFA-513 rod 6, and the two cladding only PUZRY-06 rodlets. As before we expect that only the integral fuel rod assessment case, which had failed because of intermittent simulation timeout terminations, is affected by the optional changes in the Bison code base and input files. The small number of assessment cases in this group further indicates that these four optional algorithmic changes interact with each other such that the runtime reductions from individual algorithmic changes do not simply sum in a linear fashion. Additional work to understand the interaction among the promising algorithmic changes is required.

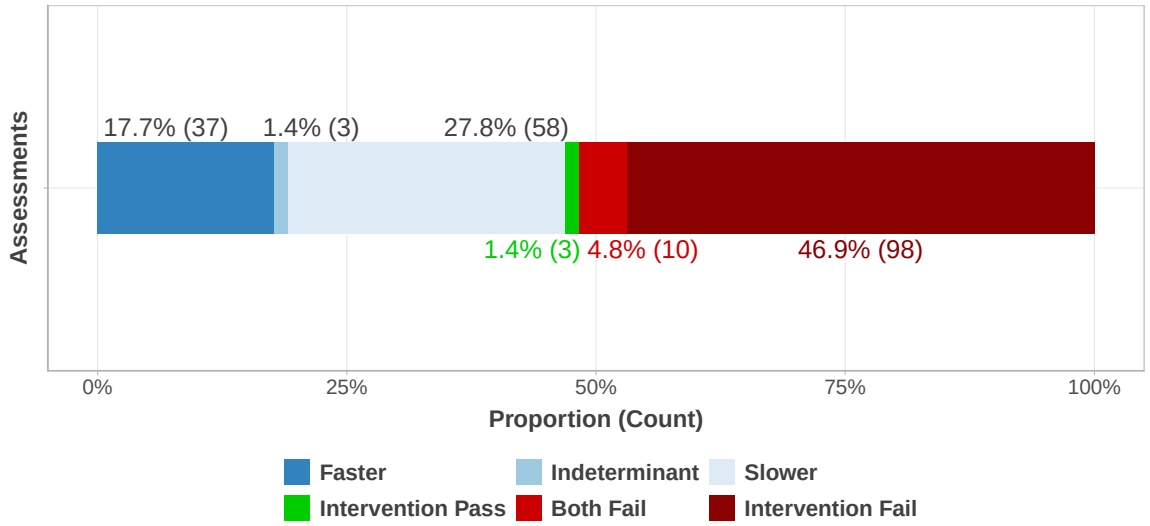


Figure 15: Overview of the number of assessment cases influenced by applying all four algorithmic interventions simultaneously. 3 cases passed that had previously failed in the baseline cases. 10 cases failed in both the baseline and intervention cases. 98 cases failed due to the intervention. A total of 111 cases were not used during comparison.

4.5.1 Comparison of Algorithmic Change Interactions

To compare the impact of each algorithmic change against the simultaneous combination of the changes on a more granular level we have formed a set of ten individual assessment cases. These ten assessment cases were selected as representative of the complete Bison LWR assessment suite, consisting of integral fuel rods and cladding only rodlets which have been compared to different types of experimental data in previous Bison validation efforts [8]. The set of ten assessment cases, along with the validation focus of each, are given in Table 3.

Table 3: The set assessment cases selected as representative of the Bison LWR assessment suite.

Assessment Case	Rod Type	Validation Focus
MT4 1-2kW	integral fuel rod	LOCA conditions
IFA 650-2 (tm)	integral fuel rodlet	LOCA conditions
PUZRY-14 (tm)	cladding only rodlet	LOCA burst test
RIA-NSRR FK03	integral fuel rodlet	RIA conditions
IFA 431-1	integral fuel rodlet	Beginning of life fuel centerline temperature
IFA 432-2	integral fuel rodlet	Beginning of life fuel centerline temperature
HBEP-BK370	integral fuel rod	Fission gas release
SuperRamp PK21	integral fuel rod	fission gas release, PCMI
USPWR TSQ002 (tm)	integral fuel rod	Fuel centerline temperature, fission gas release, PCMI
USPWR TQ022 (tm)	integral fuel rod	Fuel centerline temperature, fission gas release, PCMI

In Table 3 the notation (tm) indicates that the assessment case input file uses the tensor mechanics module system. The USPWR TSQ002 case was selected because it includes inelastic material models for the fuel, including smeared cracking and creep behavior, and utilizes many of most up-to-date physics material models available in Bison.

We compare the simulation runtime speed up among the function change limited timestepper, the solver STRUMPACK,

the adaptive predictor, the linearized thermal contact gap model, and all four simultaneous combined changes in Table 4. The values presented in Table 4 are calculated in the same manner as the values from the graphs above. In contrast to these figures above, in Table 4 we present the runtime change data for all four individual interventions and the simultaneously combined interventions run, including minimal differences, for the ten assessment cases identified in Table 4. This survey of the impact of the different interventions allows us to draw preliminary conclusions both about which type of assessment cases may be helped by which type of interventions and about how the different individual interventions may interact with each other in the combined run.

Table 4: Speedup of the simulation runtime from each algorithmic intervention for the ten representative Bison LWR assessment cases. The change in simulation runtime is presented below as a percent speedup with respect to the baseline runtime. Speedups greater than five percent are shown in bold text.

Assessment Case	Timestepper	STRUMPACK	Predictor	Linear Gap	Combined
MT4 1-2kW	3.0 ± 53.5%	-6.6 ± 31.6%	**71.6 ± 4.3%**	16.2 ± 22%	**62.5 ± 9.3%**
IFA 650-2	**30.9 ± 4.9%**	**8.1 ± 4.8%**	-6.7 ± 22.9%	**6.7 ± 3.1%**	**12.2 ± 1.6%**
PUZRY-14 (tm)	**14.4 ± 11.3%**	-22.9 ± 24.1%	**19.1 ± 12.7%**	**23.1 ± 12.0%**	-20.7 ± 19.9%
RIA-NSRR FK03	-77.2 ± 6.5%	**8.6 ± 4.7%**	**41.9 ± 3.0%**	-0.4 ± 4.6%	**53.2 ± 2.4%**
IFA 431-1	0.9 ± 0.4%	**8.3 ± 3.1%**	0.5 ± 2.2%	0.7 ± 2.7%	**5.3 ± 1.8%**
IFA 432-2	0.3 ±1.5%	**6.3 ± 1.0%**	2.9 ± 1.3%	0.8 ± 1.4%	**8.2 ± 0.1%**
HBEP-BK370	3.6 ± 5.0%	**12.9 ± 5.2%**	**21.8 ± 2.3%**	0.0 ± 5.0%	**31.7 ± 1.2%**
SuperRamp PK21	**50.5 ± 2.0%**	-19.0 ± 11.7%	**29.0 ± 3.7%**	-4.5 ± 9.5	**22.3 ± 1.4%**
USPWR TSQ002 (tm)	-0.4 ± 6.9%	**38.9 ± 5.5%**	5.9 ± 4.3%	-4.3 ± 5.1%	**45.7 ± 2.6%**
USPWR TQ022 (tm)	1.5 ± 5.4%	**40.0 ± 5.4%**	-1.1 ± 6.0%	-1.7 ± 3.9%	**40.9 ± 0.6%**

Generally the function change limiting timestepper and the adaptive predictor speed up the accident scenario assessment cases more. In these RIA and LOCA assessment cases the loading conditions are generally linearly; both of these interventions are better suited for linear loading conditions. We also note that the function change limiting timestepper and the adaptive predictor appear to potentially interfere with each other in the simultaneous combined run. The assessment cases which demonstrated significant speedups with either the individual timestepper or adaptive predictor intervention demonstrate lower speedup changes in the simultaneous combined run. The STRUMPACK intervention resulted in a speedup of many of the integral fuel rod assessment cases. Furthermore, the STRUMPACK algorithm change option appears to general maintain speedups from the individual algorithmic change runs, indicating that the STRUMPACK intervention does not interfere with the other intervention studied here. Although further study is warranted, we are cautiously optimistic about the ability of the STRUMPACK intervention to speedup the run time in many of the integral fuel assessment cases in the Bison suite.

Table 5: Reduction of the total non-linear iterations from each algorithmic intervention for the ten representative Bison LWR assessment cases. The change in total non-linear iterations is presented below as a percent speedup with respect to the baseline total non-linear iterations. Changes in the total non-linear iterations corresponding to a reduction of five percent or more are shown in bold text, where negative numbers correspond reductions in the total number of non-linear iterations.

Assessment Case	Timestepper	STRUMPACK	Predictor	Linear Gap	Combined
MT4 1-2kW	0.0 ± 0%	0.0 ± 0%	**-43.0 ± 0.1%**	0.0 ± 0%	**-43.1 ± 0.0**
IFA 650-2	**-31.7 ± 0.6%**	0.5 ± 0.5%	-1.2 ± 0.5%	0.0 ± 0.6%	**-13.7 ± 0.5%**
PUZRY-14 (tm)	0.0 ± 0%	0.0 ± 0%	**-11.0 ± 0.0%**	0.0 ± 0%	**-11.0 ± 0.0%**
RIA-NSRR FK03	73.2 ± 0.0%	-1.0 ± 0.0%	**-35.4 ± 0.0%**	0.0 ± 0%	**-43.0 ± 0.0%**
IFA 431-1	0.0 ± 0%	-0.1 ± 0.2%	-0.3 ± 0.0%	0.0 ± 0%	0.0 ± 0%
IFA 432-2	0.0 ± 0%	0.0 ± 0%	-1.7 ± 0.0%	0.0 ± 0%	-1.7 ± 0.0%
HBEP-BK370	-4.2 ± 5.8%	**-5.8 ± 0.4%**	**-19.3 ± 0.7%**	-0.9 ± 7.0%	**-19.6 ± 0.3%**
SuperRamp PK21	**-49.2 ± 1.9%**	-1.1 ± 1.1%	**-33.0 ± 1.6%**	-0.4 ± 0.5%	**-36.3 ± 0.7%**
USPWR TSQ002 (tm)	0.0 ± 0.5%	-0.5 ± 0.8%	**-11.5 ± 1.2%**	-0.4 ± 0.8%	**-12.7 ± 0.7%**
USPWR TQ022 (tm)	34.2 ± 0.2%	0.0 ± 0.6%	-0.9 ± 1.7%	-0.2 ± 0.3%	-3.6 ± 0.2%

Given the potential connection between the function change limiting timestepper and the adaptive predictor to the total number iterations taken by a Bison simulation, we examine the change in the total number of non-linear iterations in the ten assessment case set, Table 3, across the four individual algorithmic changes and the simultaneous combined changes in Table 5. The total non-linear iteration reduction response of the combined simultaneous interventions run appear to be dominated by the adaptive predictor. Both the pattern and the value of the total non-linear iteration count reduction of the combined runs correspond closely with the results from the adaptive predictor runs for the ten individual assessment cases we examine in Table 5.

5 Summary and Future Work

Improvements made in the scope of this report fall under two categories. The first category is transitioning of the Bison physics models to take advantage of the new automatic differentiation system in MOOSE. This effort results in a perfect Jacobian, prepares Bison for multiphysics coupling, and improves the convergence of all converted Bison models. At present Bison simulations are limited by the models and subsystems that have not yet been converted over to automatic differentiation. The largest bottleneck is thermo-mechanical contact. In practice we can already observe a reduction in iteration counts in our fully converted assessment cases. On the MOOSE framework side there are still performance issues to be sorted out in the automatic differentiation system. The Jacobian evaluation in the non-linear timesteps takes substantially more time than our approximate hand-coded Jacobians. These future improvements in MOOSE will pay direct dividends to Bison without further development required on the Bison side.

The second category of improvements consists of a set of targeted specific algorithmic changes and the tools required for a systematic and standardized evaluation of the impact of each improvement.

- Performance metric output and analysis
- Improvements to time stepping procedure
- Use multi-frontal direct solver
- Adaptive Predictor
- Linearization of gap conductance models

These specific algorithmic interventions benefit very specific models and physics components that are either not used across the board in the Bison assessment cases or that can have negative performance impacts. Each proposed intervention form this report is therefore optional and should only be applied in cases where a performance or robustness improvement is observed.

Work is in progress for moving mechanical and thermal contact over to AD. The new contact system is based on the mortar method [9] and supports complete Jacobians populated by AD. This work will cover the bulk of the remaining inaccurate Jacobian contributions in Bison models. We expect the closing of this gap to help us fully realize the potential of the AD conversions performed in this report.

Preliminary work on automatic variable scaling has been committed to MOOSE. Automatic variable scaling will optimize the condition number of the Jacobian matrix by automatically adjusting the variable scaling factors for the residual vector contributions. This will improve user friendliness in particular for mechanics models, where the scaling of the displacement variables currently needs to be manually set to the inverse of the Young's modulus of the material. Automatic scaling can be enabled to rescale at every timestep to adapt to changing materials properties. Systematic testing of this new capability on the Bison assessment tests is still outstanding.

6 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] B. W. Spencer, R. L. Williamson, A. D. Lindsay, F. Kong, R. J. Gardner, J. D. Hales, A. Casagranda, D. Schwen, H. Chen, N. Prakash, C. Matthews, and C. Unal. Bison improvements for robustness and speed. Technical Report CASL-U-2018-1625-000, INL/EXT-18-45704, Idaho National Laboratory, June 2018.

[2] Brian Alger, David Andrš, Robert W. Carlsen, Derek R. Gaston, Fande Kong, Alexander D. Lindsay, Jason M. Miller, Cody J. Permann, John W. Peterson, Andrew E. Slaughter, and Roy Stogner. MOOSE Web page. `https://mooseframework.org`, 2019.

[3] Derek R. Gaston, Cody J. Permann, John W. Peterson, Andrew E. Slaughter, David Andrš, Yaqi Wang, Michael P. Short, Danielle M. Perez, Michael R. Tonks, Javier Ortensi, Ling Zou, and Richard C. Martineau. Physics-based multiscale coupling for full core nuclear reactor simulation. *Annals of Nuclear Energy*, 84:45–54, 2015.

[4] Roy Stogner and Alexander Lindsay. MetaPhysicL. `https://github.com/roystgnr/MetaPhysicL`, 2019.

[5] Deepak P. Adhikary, Chandana Jayasundara, Robert K. Podgorney, and Andy H. Wilkins. A robust return-map algorithm for general multisurface plasticity. *International Journal for Numerical Methods in Engineering*, 109:218–234, 01 2016.

[6] D.L. Hagrman and G.A. Reymann. Matpro-version 11: a handbook of materials properties for use in the analysis of light water reactor fuel rod behavior. Technical Report NUREG/CR-0497, Idaho National Engineering Laboratory, February 1979.

[7] 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.*, pages 271–283. American Society for Testing and Materials, 1982.

[8] RL Williamson, KA Gamble, DM Perez, SR Novascone, G Pastore, RJ Gardner, JD Hales, W Liu, and A Mai. Validating the BISON fuel performance code to integral LWR experiments. *Nuclear Engineering and Design*, 301:232–244, 2016.

[9] John W. Peterson. Progress toward a new implementation of the mortar finite element method in MOOSE. Technical Report INL/EXT-17-44034-Revision-0, Idaho National Laboratory, February 2018.