DOE FY16 Phase II SBIR
Topic 32d: Modeling and Simulation
Project Title: Automated Solver Selection for Nuclear Engineering Simulations

DOE Office of Science ASCR
Phase II Contract: DE-SC0013869
DUNS: 141943030
Project Period: August 2016-August 2019
Report Date: May 31st 2018
Report Period: Semi annual

Gerald Sabin[1] Boyana Norris[2] Vijay Mahadevan[3]

[1] RNET Technologies
240 W Elmwood Dr
Dayton, OH 45459-4296, US
gsabin@rnet-tech.com

[2] Dept. of Computer and Information Science
307 Deschutes Hall,
1202 University of Oregon,
Eugene, OR 97403, USA
norris@cs.oregon.edu

[3] 9800 Cass Ave
Lemont, IL, 60439, USA
mahadevan@anl.gov

1 Executive Summary

This report provides a report on DOE SBIR/STTR Phase II project: Automated Solver Selection for Nuclear Engineering Simulations with award number DE-SC0013869. The original project period was August 1st 2016 - 31st July 2018. The project was granted a 12 month no cost time extension for the period August 1 2018 - July 31st 2019. As such, this document serves as a semi-annual interim report for the six month period of December 1st 2017 - May 31st 2018.

The project aims to address various challenges in increasing the adoption of NEAMS tools by novice users by enabling automatic selection of appropriate solvers for the given problem. Although this requires the analysis of the matrix A in the linear system Ax=b, there are many matrix-free methods and preconditioners which could benefit from this approach as well, which the framework will also support. Of further interest are optimizations in quantities such as the energy consumed by the operations and the resiliency of the solver to soft errors.

The Nuclear Energy Advanced Modeling and Simulation (NEAMS) program by DOE is developing predictive models for the advanced nuclear reactor and fuel cycle systems using leading edge computational methods and high performance computing technologies. The SolverSelector will enable automatic selection of the optimal solver for the characteristics of the linear system being solved as well as considering the compute hardware present.

The major accomplishment of this reporting period has been on the development of feature extraction techniques for matrix-free methods. Feature extraction for Matrix-free methods is difficult because one does not have direct access to the matrix elements. The methods developed in this reporting period produce low-cost estimates of the matrix features that can be used to train machine learning algorithms and predict the optimal linear solver with a high level of accuracy.

In addition to this, the project team has also continued its efforts towards developing the SolverSelector API and in obtaining robust real-world training data sets from Proteus line of tools.

2 Accomplishments

2.1 Goals

The main goals and objectives of the project are:

- Make the automatic solver selection technology applicable to a broad range of nuclear reactor simulations by enabling solver selection throughout the spectrum of NEAMS tools.

- Extend the automatic solver selection to a special category of numerical subproblems (physics based preconditioning, matrix free methods etc.) and leadership platforms that are important to NEAMS applications.

- Investigate other performance objectives such as accuracy, resiliency, and energy efficiency for optimal solver selection in addition to CPU time. Cater to a wide range of simulation problems and customers with varying performance criteria.

- Integrate the above techniques into a pluggable software API and Library that can be leveraged in codes developed by NEAMS. Design and implement the software indirections that will enable plug-n-play integration of the proposed features into other numerical software.

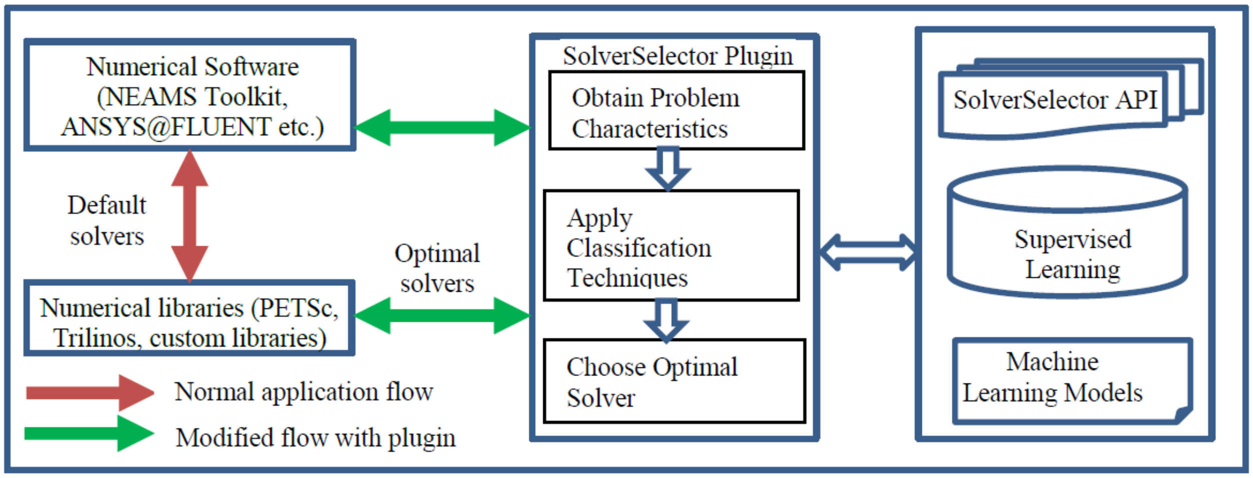

Numerical Software (NEAMS Toolkit, ANSYS@FLUENT etc.)
Default solvers
Numerical libraries (PETSc, Trilinos, custom libraries)
Optimal solvers
Normal application flow
Modified flow with plugin
SolverSelector Plugin
Obtain Problem Characteristics
Apply Classification Techniques
Choose Optimal Solver
SolverSelector API
Supervised Learning
Machine Learning Models

Figure 1: Overview of intended use of the proposed plugin.

The overall view of the proposed framework is outlined as shown in Figure 1. The major technical goals of the project will be accomplished by executing the list of technical tasks as stated in the proposal:

Task 1: Automatic Solver Selection for Fuels Product Line: Code instrumentations and data collections on MOOSE and BISON problems.

Task 2: Automatic Solver Selection for Reactors Product Line: Examination of problems that can benefit from this approach in Diablo and SHARP problems. Extending automatic solver selection approaches to the reactors product line.

Task 3: Matrix Free methods: Development of new approaches for matrix free methods. Develop novel methods for partial Jacobian and physics based preconditioning.

Task 4: Other Performance metrics: Evaluate automatic solver selection algorithms and its impact in terms of efficiency gain (CPU time), accuracy, energy efficiency, and resilience of overall NEAMS simulations.

Task 5: Multi-objective Optimization: Incorporate optimization of multiple objectives into the solver modeling and selection framework.

Task 6: Optimizations Targeting DOE Computing Facilities: Training data for Machine Learning models will be built using compute infrastructure at DOE labs.

Task 7: System Integration and Software Indirections: Integration of all features into a unified framework.

2.2 Recent Accomplishments

The report will outline relevant work carried out in the 6 month period December 2017-May 2018. During this reporting period, the project team continued to focus on Tasks 2, 3 and 7.

2.3 Task 2: Automatic Solver Selection for the Reactors Product Line

The NEAMS reactor product line consists primarily of PROTEUS, Nek5000 and Diablo physics codes. Among these, both Nek5000 and PROTEUS developed at ANL, have been shown to strong scale well (over 512K processes) on HPC architectures.

During this reporting period the project team continued its work related to extracting relevant and meaningful training data from the PROTEUS line of tools. This involved equipping the Proteus software with the necessary software indirections and interfaces for dumping the petsc-based linear solver matrices to file, as well as obtaining run-time logs of the linear solver. The modified Proteus software was then used to extract the matrices and solver performance information from a range of simulations using the Proteus benchmark problems.

The data obtained from these simulations will be fed into our machine learning algorithms in the upcoming reporting period. This will allow the project team to explore the effects domain specific features have on the accuracy of the machine learning algorithm, ultimately allowing us to develop tools and techniques for developing a robust machine learning algorithm for individual tools such as those in the NEAMS toolkit. The project team have also begun work to set up several larger, real-world Proteus simulations to further aid in this research. The results from these larger simulations are expected in the upcoming reporting period.

2.4 Task 3: Matrix Free Methods

A large majority of MOOSE based applications, including BISON and MARMOT, use matrix-free methods to solve the linear systems. Rather than storing the coefficient matrix explicitly, matrix-free methods access the matrix by evaluating matrix-vector products. This poses a difficulty for automatic solver selection because the majority of the classification features, i.e, the matrix norm, the trace and the row variance, require direct access to the matrix co-efficients. The extraction of features from systems defined using a matrix-free approach is cutting-edge and novel research. During this reporting period, the project team developed a sampling based technique for matrix-free feature extraction that has proven to be very successful. This is a very promising result as it extends the domain of the solver selection algorithm to include matrix-free nonlinear solvers such as the Preconditioned Jacboian Free Newton Krylov method (PJFNK) favored in MOOSE.

The development of the sampling based approach came after a detailed evaluation of a range of matrix-free feature extraction techniques that used only matrix-vector multiplications to extract features. This included an investigation into the three methods outlined in the previous progress report. Those methods included using a graph coloring algorithm to extract matrix values, an incomplete power iteration to estimate the eigenvalues, and a stochastic method for estimating the trace and the row variance. Each of those methods were tested and subsequently disregarded for being either to inaccurate (stochastic trace and row variance), to expensive (graph coloring) or, ineffective (incomplete power iteration). In contrast, the sampling based approach provided a good balance between accuracy and computational cost.

2.4.1 Sample Based Matrix-free Feature extraction

Define a $n \times n$ matrix A with columns $\mathbf{a}_j$ such that

$$A = \begin{bmatrix} | & & | \\ \mathbf{a}_1 & \dots & \mathbf{a}_n \\ | & & | \end{bmatrix},$$

Then, one can extract the vector $\mathbf{a}_j$ through a single matrix-vector multiplication with the $j'th$ Euclidean basis vector. That is to say,

$$\mathbf{a}_j = A\mathbf{e}_j$$

where $\mathbf{e}_j = \{0, 0, \dots, 1, 0 \dots\}$ represents the j'th standard basis vector. In fact, one can determine the effective value of each and every element in a $n \times n$ matrix-free system using at most n matrix-vector multiplications. However, this is an O(n^3) approach that returns optimal results in terms of feature accuracy, but is to expensive for use in a practical setting.

Our approach has been to use a small O(1) sample of the matrix columns to inform estimates of overall matrix features. Let S represent a set containing $m \ll n$ columns of A. Then, the set S can be formed in a matrix-free environment through the use of m matrix-vector multiplications at a cost of O(mn^2).

To ensure accuracy and to minimize bias, it is essential that the sample set S encompasses a good cross-section of the columns in A. This is particularly true for the matrices that arise from grid based PDE methods. In those cases, each column of A represents a point in the computational grid. As such, features such as boundaries can dramatically effect the size and number of elements in rows. To that end, it is advantageous to select a sample that includes a good mix of columns linked to interior and boundary based grid points. In fact, our testing has shown that it is beneficial to pick and choose which sample columns are used in each feature calculation. For example, due to the effects of boundary conditions on the contents of some columns, it is best to use columns that translate to interior mesh points to the estimate the number of non-zeros in the matrix. Various other features make similar decisions, using different subsets of the overall sample data to calculate the final value.

Another important aspect of the sample based feature extraction technique is the statistical scaling of the features. Where appropriate, all features calculated are scaled by the ratio n/m to ensure that the feature calculations reflect estimates of the overall matrix rather than just the sample set. Some features, such as those including maximums or minimums, do not use scaling, but instead assume that the value calculated using the sample set is representative of the entire matrix.

Table **??** outlines the features that we were able to calculate using this approach. The Sample column notes the subset of the data that was used to calculate that feature as explained above. One of five subsets were used; (1) full: the full sample data set, (2) interior: Only columns representing interior grid points (3) boundary: Only columns representing boundary grid points were used, (4) square: Only the elements A_{ij} for $i, j \in S$[1]. Features that were scaled are marked with a *.

[1]This forms a square, $m \times m$ sub-matrix that can be tested for properties such as symmetry

2.4.2 Implementation Details

The primary concern for automatic solver selection, both in the matrix-free and standard matrix based systems is that the cost of feature extraction acts to counteract the time saving benefits of choosing the optimal solver. Certainly, in situations where the best solver is already known, the cost associated with feature extraction will actually cause the method to be slower. Likewise, in situations where a poor default solver is used, the cost of feature extraction will be negligible when compared to the run-time reduction obtained by using the optimal solver.

As such, it is essential that the feature extraction techniques be as efficient as possible. To ensure this, the matrix-free feature extraction routine has been implemented directly in Petsc using a single loop. That is to say, for each column in the sample set, we complete a single matrix vector multiplication, followed by a single loop through the values of the column. If multiple cores are used, this is completed in parallel, with the only communication between processors being completed during the matrix-vector multiplication as required by Petsc.

A single MPI Reduce call is made at the completion of the feature extraction stage, at which point the root processor completes the final collation and calculation of the matrix-free features. In a practical setting, the root processor would then feed those features into the machine learning algorithm before scattering back the details of the optimal solver to the remaining processors.

This computation-then-communication pattern is extremely efficient in terms of latency, but is somewhat bandwidth heavy. For example, the entire square data set (m^2 elements, where m is the number of samples) must be sent through to the root process so that the symmetrical features can be calculated. Given that we require m to be small, we do not foresee this being a problem because, in cases where the need for a large m arises, the cost of the m matrix-vector multiplications, both in terms of time and memory, will likely overshadow any costs associated with the high bandwidth MPI Reduce call.

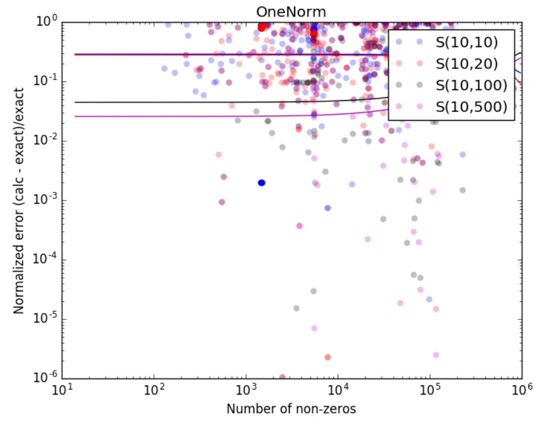
2.4.3 Results

In the following tests we highlight some results of our matrix-free feature extraction algorithms. To ensure we can check and verify our results, real matrices with known matrix-values were used. These matrices were obtained from the Florida sparse matrix collection. Note that there is no algorithmic difference between using the matrix-free feature extraction techniques on a matrix in which the matrix elements are known, and in a matrix-free system where only the action of the matrix on a vector is available [2] All of the following tests were performed on a linux based virtual machine (Windows host) using a single processor and a standard hp Desktop. Parallel tests using RNETs multi-node cluster are planned for the upcoming reporting period.

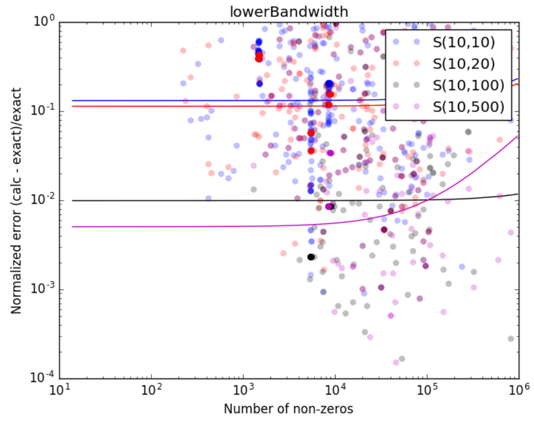
Figures **??**-**??** plot the error in four of the matrix-free features against the number of non-zeros in the matrix for every matrix in our database. Each figure presents the accuracy for a different sample set $S(i, e)$, where e indicates the number of boundary columns used and i indicates the number of interior point columns. In this case, the interior columns were selected randomly while the boundary columns were chosen to be columns 0 to $\lfloor e/2 \rfloor$ and $n - \lfloor e/2 \rfloor$ to n. The error in a feature is defined to be the difference between the exact and calculated feature value divided by

[2] In the case that the elements are known, one could reduce the computational cost by simply extracting the sample columns from memory, however, we chose to complete the matrix-vector multiplications to ensure the performance results are representative of a real-world matrix-free problem.

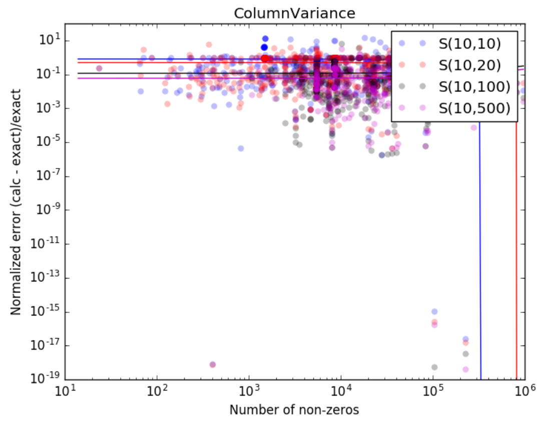
the magnitude of the exact feature value. The four features presented where chosen because they were deemed to be important features in our previous tests for standard matrix based systems.



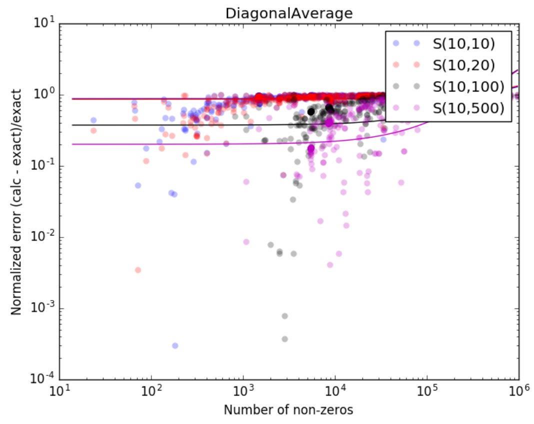
(a) One Norm



(b) Lower Bandwidth



(c) Column Variance

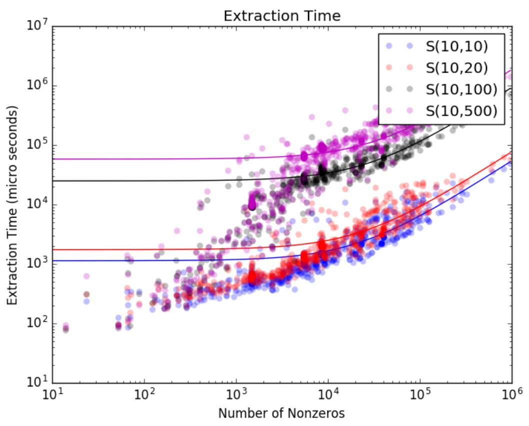


(d) Diagonal Average

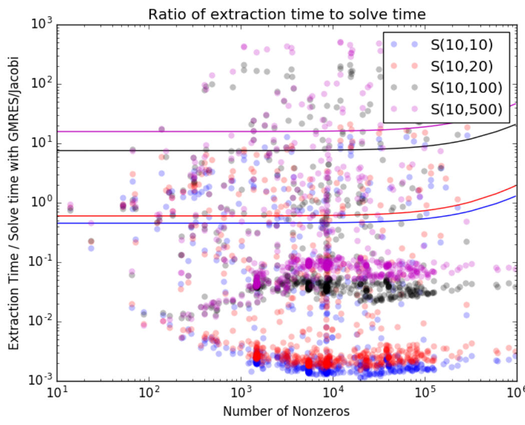
Figure 2: The normalized error four four of the features when calculated with using the matrix-free sampling approach. Each marker indicates a different matrix. The solid lines represent a the linear line of best fit through the data obtained for each sample set. Both the x and y axis are presented using a log scaling due to the large differences in values.

The important thing to note here is that the error increases as the sample size decreases. If one chooses to sample all rows, the feature set is 100% accurate, with the accuracy decreasing from there as the sample size drops.

Another important aspect of feature extraction is the cost. Figure **??** plots the time taken to extract the entire set of features against the number of non-zeros for the entire matrix data set, for the set of randomized sample-sets used in Figures **??**. Figure **??** takes this a step further, plotting the ratio of feature extraction time over the time taken to solve the matrix using GMRES preconditioned with Jacobi. GMRES preconditioned with Jacobi can be used to solve a wide range of problems, although Jacobi is not generally thought of as a fast preconditioner, with many better, albeit less robust, options available.



(a) Time required to extract feature plotted against number of non zeros for matrices in the moose dataset.



(b) Time required to extract feature over the time required to solve the system using GMRES preconditioned with ILU plotted against number of non zeros for the matrices in the moose dataset.

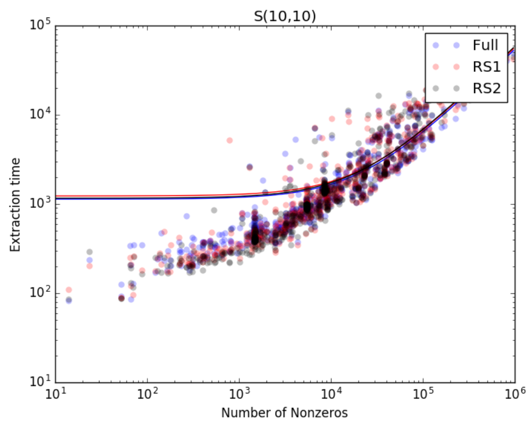
Based on these figures it is clear that reducing the size of the feature set also reduces the cost. This should not be surprising as a smaller sample set means less matrix-vector multiplications, less data to process and a smaller memory requirement. Most importantly, for the majority of the matrices, the time taken to extract the features was much less than the solve time. This is extremely promising indicating that large run-time reductions should be possible in real world situations.

One final approach for reducing the cost of feature extraction is to reduce the number of features in the feature set, with the downside being that reducing the number of features will likely reduce the effectiveness of the machine learning models. A similar approach was used for standard matrix-based systems with good results. Table **??** shows the features included in the reduced feature sets found for standard matrix based systems. Figures **??**-**??** plot the time required to extract the features for the full and reduced feature sets for four different sample sets. Interestingly, there is little to no difference between the time taken to extract the features when the sample size is small. However, once the sample size increases, the differences become quite large.

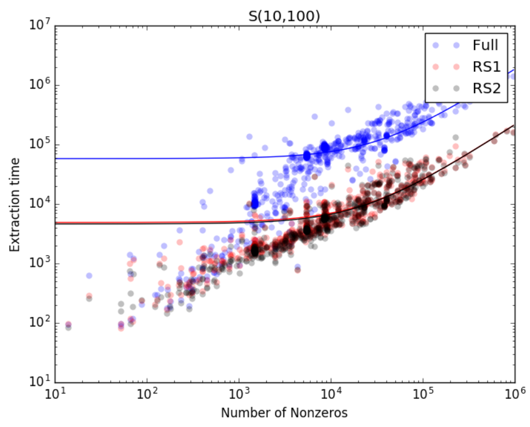
Table 1: Matrix-free: Reduced Feature Sets

Features	Reduced Set 1	Reduced Set 2
Min. Non zeros/row	X	X
Lower Bandwidth	X	X
NonZeroPatternSymmetryV1	X	
Infinity Norm	X	
Column Variance	X	X
Diagonal Non Zeros	X	X
Diagonal Average	X	X

All in all, these results show that, when carefully tuned, the sample based feature extraction method developed and implemented during this reporting period is more than capable of producing an accurate set of features at a cost that is small when compared to a basic Krylov solver. The next



(a) $S(10,10)$



(b) $S(10,100)$

Figure 4: Time required to extract the features in each feature set for two different sample sets.

step is to determine the effectiveness of each feature set. That is, how good are the machine learning models at classifying solvers when trained and tested using these feature sets. The project team is currently running these tests with results expected within the first month of the next reporting period; however, early results using a matlab implementation of the extraction algorithm have produced models with between 90 and 100% overall accuracy depending on the learning algorithm used.

2.5 Task 7: SolverSelector API

Task seven addresses the issue of developing the software indirections that allow for plug-and-play integration of the solver selector features into the NEAMS solvers. During this reporting period, the project team continued the development of it SolverSelector API.

In particular, the project team focused on integrating the aforementioned matrix-free feature extraction techniques into the petsc interface of the API. While the API is designed to be applicable to a variety of linear solver apckages, Petsc was chosen for testing and development because it is a mature framework that supports a wide range of Krylov solvers and practitioners. Moreover, Petsc is used widly across the numerical simulation community and is the go to tool when it comes to krylov based linear solvers. In particular, the MOOSE framework, and hence a large number of NEAMS tools, use Petsc for its linear algebra and linear solver algorithms.

As of the end of this reporting period, the Solver selector API allows the user to:

- Build a machine learning database using matrices stored in the Petsc binary format using a simple input file format.
- Extend the machine learning database at run-time through the use of a simple input file format. This is particularly useful for matrix-free methods, where one cannot build the database externally using saved matrix files.
- Use machine learning to automatically select the best linear solver based on a pre-built machine learning database and model (i.e., online linear solver optimization)

- Perform cross-validation testing to access the validity and accuracy of the machine learning model based on the current database
- Save the optimal solver choice for use at a later point, increasing efficiency for problems where the matrix values do not change dramatically between runs or linear solves (i.e., offline linear solver choice optimization).

The project team has integrated and used each of these features in both matrix-free and standard matrix systems from both the Petsc and Moose example sets. In both cases, utilizing the solver selection features requires less than five lines of additional code and two command line parameters.

2.6 Training and Professional Development

This project has provided opportunities for research development and collaboration with Dr. Boyana Norris at The University of Oregon (UO). The UO subcontract includes the support of graduate students. We are also closely working with Dr. Vijay Mahadevan from Argonne National Laboratory who is interacting with the Diablo team and also looking at PROTEUS to generate relevant operators for problems of interest.

2.7 Dissemination

Currently, the work is in the development phase. Once completed, we will leverage our contacts in various National Laboratories and Government Agencies to demonstrate our SolverSelector plugin and the performance benefits, among others. The ultimate plan is to develop and commercialize a flexible automatic solver API. A research paper based on the matrix-free feature extraction techniques and the associated machine learning models is being written by our collaborators in Oregon for submission to the Copper Mountain special issue of SISC.

2.8 Plans

In the upcoming no cost time extension period, the Solver selector API will be hardened, tested, documented and extended. However, the majority of the work will be focused on improving the underlying machine learning models; a task that will be made much easier through the solver selector API. This will include detailed testing and analysis of our run-time solver selection algorithms in real-world examples starting with the examples in Petsc and Moose and moving up through to examples using NEAMS tools such as PROTEUS and BISON. Impressive run-time reductions for these simulations, especially in the matrix-free case, will be a key selling point for the Solver Selector API.

In addition to this, the project team will begin to investigate some of the additional features and algorithms outlined in the no-cost-time-extension request, such as a model for extrapolating training data obtained using a small number of processors for use on large-scale compute resources. Building the training data sets is expensive, hence these extrapolation techniques and algorithms will extremely useful in removing the computational burden associated with setting up the machine learning models.

3 Products

3.1 Publications

None this period; however, a paper based on the matrix-free feature extraction techniques is currently being written.

3.2 Intellectual Property

None this period.

3.3 Technologies and Techniques

None this period.

4 Participants and Collaborators

4.1 Participants

Name: Gerald Sabin
Project Role: PI
Nearest Person Month Worked: 1
Contribution to Project: Gerald is the PI of the project and has been leading the projects technical direction at RNET, and working to develop its commercial roadmap. He has also been working closely with RNET technical staff on the development of software produced during this period.
Funding Support: this project
Collaborated with individual in foreign country: No
Country(ies) of foreign collaborator: N/A
Traveled to foreign country: No
If traveled to foreign country(ies), duration of stay: N/A

Name: Ben O'Neill
Project Role: Research Scientist
Nearest person month worked: 2
Contribution to Project: Ben is the lead developer for the Solverselector API. In particular, Ben is working on developing the underlying framework as well as the interface for Petsc, Moose and the NEAMS tools.
Funding Support this project: This project
Collaborated with individual in foreign country: No
Country(ies) of foreign collaborator: N/A
Traveled to foreign country: No
If traveled to foreign country(ies), duration of stay: N/A

4.2 Partners

Organization Name: University of Oregon
Location of Organization: Eugene, OR
Contribution to the project: Staff at the University of Oregon are leading the basic machine learning research for this project.
Financial support: This project
In-kind support: None
Facilities: None
Collaborative research: Yes
Personnel exchanges: None

Organization Name: Argonne National Laboratory
Location of Organization: Lemont, IL
Contribution to the project: Staff at Argonne are working to extract relevant linear operators from a spectrum of solvers which will be used to test the SolverSelector on the NEAMS reactor product line.
Financial support: This project
In-kind support: None
Facilities: None
Collaborative research: Yes
Personnel exchanges: None

4.3 Other Collaborators

None this period

5 Impact

5.1 Impact on Principal Discipline

The SolverSelector will significantly impact the speed and reliability of NEAMS simulations. The product will facilitate easy adoption of the NEAMS tools by the non-experts in the academia and industry on a variety of compute platforms ranging from high-end workstations to large computing clusters. The NEAMS tools play a significant role in enabling pellet-to-plant simulation capability of high performance nuclear engineering simulations. It is often necessary to leverage domain expertise to get the most out of these tools, like, say, choosing the right solvers to obtain the best performance. The impact of our work would be to let the underlying code choose the appropriate solver, based on the input matrix structure. Such a feature would be very convenient for novice users and hence drive adoption rates and perpetuate beneficial practices. Potentially, we can also extend the work to optimize other characteristics such as energy and examine the resiliency of the numerical code to soft errors. This can allow power constrained devices to react accordingly when it is necessary to solve such linear systems, with minimal power consumption

5.2 Impact on other Disciplines

An automatic solver selector can be applied in a range of applications, outside the scope of the NEAMS tools. In general, any steady-state or time-dependent numerical simulations can benefit from our SolverSelector. The computational scientists, numerical software developers, and electrical engineers will be able to let the software decide the best performing method without the need for labor-intensive experimentation, and thus focus on more scientific aspects of their applications.

5.3 Impact on Human Resources

None

5.4 Impact on Infrastrucure

None

5.5 Impact on Technology Transfer

This project will transfer basic research in developing machine learning models that automatically select linear solvers and preconditioners into an easy to use API that can be integrated with state-of-the-art simulation tools in academia, national labs, and industry.

5.6 Impact on Society

Improving the performance and accuracy of the linear solvers, will allow for faster and more accurate simulation of many products that are crucial to the United States Government and its Citizens, including airplanes, nuclear reactors, medical devices, and many, many other products.

6 Changes

None.

7 Demographics

Table 2: Full Feature Set

Features	Description	Sample Set
Dimension	This feature gives the size of the matrix A. For a square matrix, the matrix size is the number of rows or columns of the matrix.	
Number of non-zeros *	The total number of non zeros in the coefficient matrix A.	interior
Minimum non-zeros per row	This property finds the minimum number of non zeros in any row of the matrix A.	boundary
Maximum non-zeros per row	This feature gives the maximum number of non-zeros in any row of the matrix A.	interior
Average non-zeros per row *	This feature provides the average number of non-zeros per row for the matrix A.	full
Row diagonal dominance	This feature computes the row diagonal dominance of the matrix A. It returns 0 if its not diagonally dominant.	full
Symmetricity	This feature finds whether the matrix is symmetric or not. For a matrix to be symmetric, it is equal to its transpose ($A = A^T$).	square
Trace *	This feature computes the sum of the elements of the main diagonal of the matrix A.	full
Absolute Trace *	This feature computes the sum of absolute values of the diagonal elements of the matrix.	full
Diagonal non-zeros *	This feature counts the number of non-zero elements on the diagonal.	full
One norm	This feature gives the maximum column sum of the matrix A. It is denoted by the formula $\|\|A\|\|_1 = max_j \sum \|A_{ij}\|_{i=1}^{m}$, where $j = 1, \ldots, m$.	full
Infinity norm	The infinity norm is computed as follows: $\|\|A\|\|_\infty = \max(\sum_{j=1}^{n} \|A_{1j}\|, \sum_{j=1}^{n} \|A_{2j}\|, \ldots, \sum_{j=1}^{n} \|A_{mj}\|)$	full
Symmetric Infinity norm	This feature provides the infinity norm of only the symmetric part of the matrix A.	square
Anti symmetric Infinity norm	This feature computes the infinity norm of the anti-symmetric part of the matrix A.	square
Frobenius norm *	This feature gives the square root of the sum of the absolute squares of its elements. It is denoted by $\sqrt{\sum_{i=1}^{m}(\sum_{j=1}^{n} \|A_{ij}\|^2)}$.	interior
Symmetric Frobenius norm *	Frobenius norm of the symmetric part of the matrix.	square
Anti symmetric Frobenius norm *	This feature gives the Frobenius norm of the anti-symmetric part of the matrix.	square
Absolute non-zero sum *	Sum of the absolute values of all the non- zeros in matrix.	interior
Diagonal Average *	This feature computes the average of the absolute values of the diagonal elements of a matrix.	full
Lower Bandwidth	The smallest number k, such that any entry $a_{i,j} = 0$ when $i - j > k$.	full
Upper Bandwidth	The smallest number k, such that any entry $a_{i,j} = 0$ when $j > i + k$.	full
Average diagonal distance *	This feature gives the average distance of nonzero diagonal to the main diagonal.	interior
Non zero pattern symmetry V1	Checks the nonzero pattern symmetry. If symmetric, returns 1 otherwise 0.	square
Column diagonal dominance	This feature computes the column diagonal dominance of the matrix. It returns 1 if it is diagonally dominant, and 0 otherwise.	full
Diagonal sign	This feature indicates the diagonal sign pattern of the matrix. The value is -2 for all negative elements, -1 for some non-positive elements, 0 if all elements are zero, 1 is some elements are non-negative, 2 if all elements are positive.	full
Row variance	This feature computes the row variance of the matrix	interior
Column variance	The feature computes the columns variance of the matrix	interior

Number	Name	Project Role	Email	Months
1	Gerald Sabin	PI	gsabin@rnet-tech.com	1
2	Ben O'Neill	Research Scientist	boneill@rnet-tech.com	1

Table 3: Demographic Information. Months represents the man months spent during the current reporting period.