Matrix Reduction Techniques for Ordinary Differential Equations in Chemical Systems

Varad Deshmukh

University of California, Santa Barbara

April 22, 2013

Contents

1 Introduction

Many Ordinary Differential Equations (ODE's) arising from real-world systems display the phenomenon of stiffness [15]. One such class is that belonging to chemical reaction systems in various fields such as biochemistry and fuel combustion kinetics. Chemical systems often consist of a large number of species, whose concentrations vary as various reactions occur simultaneously. Using the theory of chemical kinetics, it is possible to represent the chemical system as an ODE system of species concentrations. In case of chemical systems, stiffness arises due to the presence of both fast chemical reactions that go quickly to chemical equilibrium and extremely slow reactions. In the case of reaction-diffusion systems, diffusion is an additional source of stiffness. Due to this stiffness, extremely small time-steps are required to meet the stability requirements, increasing the computational workload.

To address the stiffness problem, model reduction techniques may be employed. Some of these techniques involve replacing the system by a reduced model that removes the stiff portions of the system. A number of existing approaches for model reduction of chemical systems can be found in the literature. Bhattacharjee et al. [5] and Okino et al. [21] discuss the extreme stiffness present in systems such as the modeling of pollutants and byproducts in chemical reactions. Bhattacharjee et al. provide an optimization-based reduction strategy to split the entire system and solve locally-accurate reduced kinetic models. Similarly, a paper by Petzold and Zhu [22] presents an optimization-based reduction approach that reduces the number of reactions in stiff chemical kinetic models.

To reduce the number of time-steps in simulation of stiff systems, implicit numerical methods are used. The most computationally intensive part of solving a ODE system with an implicit method involves solution of a linear system in the Newton iteration. In this report, we investigate model reduction strategies that speed up the linear solver process, by approximating the linear system with a system that might be faster to solve. An important advantage of large chemical systems is that the matrices in these linear systems are extremely sparse. We can therefore use a sparse linear solver to improve the simulation time. We investigate the degree to which reductions together with sparse strategies can speed up the simulation.

2 Chemical Models

For the experiments, we considered three models in the field of biochemistry, which involve sparse linear systems.

1. **Heat Shock Model:** The Heat Shock model was proposed by Kurata et al. [19] to represent the dynamical nature of the heat-shock response in Escherichia Coli. The system consists of 28 species and 61 reactions. The iteration matrix contains 116 non-zero elements. The model is run for 500 seconds of simulated time.

2. **Coagulation Model:** This model was proposed by Luan et al. [20] to represent the coagulation cascade in the human body. This is a much larger system of 193 species and 301 reactions. The iteration matrix has 938 non-zero elements. The model exhibits better sparsity than the Heat Shock model. The model is run for 700 seconds of simulated time.

3. **Hockins Coagulation Model:** This model, proposed by Hockins et al. [17], is a smaller version of the Coagulation model, consisting of 34 species and 45 reactions. We use this model

specifically for Michaelis-Menten reduction. As with the Coagulation model, we simulate the Hockins model for 700 seconds of simulated time. The Michaelis-Menten reduction requires adding extra pseudo-species to the system, and therefore, the iteration matrix has a larger dimension than the number of species.

The characteristics of all the models are summarized in Table 1.

Model	Species	Reactions	Iteration Matrix Dimension	Number of Non-zeros	Simulation time (s)
Heat Shock	28	61	28×28	116	500
Coagulation	193	301	193×193	938	700
Hockins Coagulation	34	45	37×37	158	700

Table 1: Summary of the characteristics of the chemical models.

3 Proposed Techniques – An Overview

To handle the stiffness in ODEs, numerical methods that have a large region of absolute stability are used. Commonly used implicit ODE solvers for stiff problems are Backward Difference Formula (BDF) methods [8] and implicit Runge-Kutta methods such as Radau [14]. Consider an ODE of the form

$$\frac{dy}{dt} = f(t, y). \tag{1}$$

The expression for the $(n+1)^{th}$ step of the k^{th} order BDF scheme is given by [3]

$$\sum_{i=0}^{k} \alpha_i y_{n+1-i} = h\beta f(t_{n+1}, y_{n+1}). \tag{2}$$

For an implicit scheme, we cannot directly solve for y_{n+1} using the previous simulation steps. A Newton iterative method needs to be used to solve the above nonlinear system. The Newton method finds the roots of nonlinear system (2) for y_{n+1}, given by

$$g(y_{n+1}) = \sum_{i=0}^{k} \alpha_i y_{n+1-i} - h\beta f(t_{n+1}, y_{n+1}) = 0. \tag{3}$$

Starting with an initial guess y_{n+1}^0 which comes from evaluating the interpolant of the BDF method at t_{n+1}, Newton iteration yields

$$\left(\left. \frac{\partial g}{\partial y} \right|_{y_{n+1}^\nu} \right) (y_{n+1}^{\nu+1} - y_{n+1}^\nu) = -g(y_{n+1}^\nu), \tag{4}$$

where the iteration matrix is $\frac{\partial g}{\partial y} = I - h\beta \frac{\partial f}{\partial y}$, and is generally evaluated only at the initial guess. Henceforth, we write $J = \frac{\partial f}{\partial y}$ for the Jacobian of f, and $M = I - h\beta J$ for the iteration matrix. The iteration proceeds until y_{n+1}^ν converges, i.e. $||y_{n+1}^{\nu+1} - y_{n+1}^\nu||$ is less than an error tolerance.

Thus, we solve a linear system at each iterative step, and use the solution to compute $y_{n+1}^{\nu+1}$. The cost of a Newton iteration step is largely dominated by the cost of solving the linear system.

3.1 Matrix Reductions

The chemical systems we study lead to sparse iteration matrices. The time required to factor the matrix and to do the forward and backward solves directly depends on the sparse structure of the matrix M. By intelligently deleting certain elements of the iteration matrix, we can create a better sparse structure, speeding up the linear solver. Since the Newton iteration is an approximate method, the iteration matrix M does not have to be exactly accurate. By making some small changes in the matrix, it is still possible to reach convergence for y_{n+1} without significantly affecting the number of iterations required. This forms the basis for all our model reduction schemes.

We explore different forms of reductions on this matrix. For the systems we consider in the paper, the matrix M is asymmetric and not so well-conditioned. The Heat Shock and Coagulation models produce iteration matrices with condition numbers of order 10^7 or larger. The Hockins model produces matrices with condition numbers of order 10^3.

3.1.1 Global Threshold Reduction

We consider the absolute value of each element of the matrix $h\beta J$ individually, compare it with a user-specified threshold - τ_{global} , and set it to zero if it is smaller than the threshold.

```
foreach (i,j)
if(abs(hβJ(i,j)) < τ_global)
  J(i,j) = 0
end
M = I - hβJ
```

3.1.2 Local Threshold Reduction

We consider each element of the iteration matrix J, and determine the maximum absolute value in its row i and column j. If the magnitude of $J(i,j)$ is much less than a given fraction of the smaller of those values, i.e. $abs(J(i,j)) < \tau \times min(rowmax(i), colmax(j))$, we set it to zero. The intuition is that, if a reaction involving the pair of species (A, B) is much slower than the fastest reactions involving A or B, we remove the reaction for this pair.

```
foreach (i)
  rowmax(i) = max(abs(J(i,:)))
  colmax(i) = max(abs(J(:,i)))
end
foreach (i,j)
if(abs(J(i,j)) < τ_local× min(rowmax(i), colmax(j)))
  J(i,j) = 0
end
M = I - hβJ
```

3.1.3 Michaelis-Menten Reduction

The Michaelis-Menten (MM) scheme [18] is a reduction that changes the chemical system itself, rather than changing the matrix. It replaces a sequence of reactions of the form

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C \overset{k_3}{\rightarrow} E + P \tag{5}$$

with the simpler reaction

$$S \overset{k'}{\rightarrow} P. \tag{6}$$

An assumption for the scheme, proposed by G.E. Briggs and J.B.S. Haldane [7], is that the complex C is in the Quasi-Steady-State (QSS) [25], i.e. its concentration remains almost constant on the time scale with which P is formed. Since the enzyme E and the complex C are not part of the ODE system (6), the Jacobian has zeros for entries corresponding to the species. Thus, the reduction involves creating a sparse Jacobian by reducing the chemical system. The concentrations for E and C can be computed algebraically. Often, these two species also participate in non-MM reactions. In such a case, we track their concentrations using a single pseudo-species per MM reaction. The supporting derivations for the Michaelis-Menten reduction are described in detail in Appendix A.

The MM reduction may be applied in several places within a system, by identifying the reactions that exhibit the Michaelis-Menten form and determining the corresponding QSS conditions. We use the techniques of Wu et al. [26] to identify the reactions and the time instances for their QSS activation.

3.2 Linear Solvers Considered

We consider two methods to solve the linear system – the direct and the iterative solver. The type of solver determines where the reduced matrix is used.

3.2.1 Direct Solver

In the direct solver, the reduced matrix is directly used as part of the linear system to be solved. To take advantage of the sparse structure, a sparse linear solver, KLU [10], is used. The direct solver approach is used for the threshold-based methods.

3.2.2 Iterative Solver

We also experiment with the reduced matrix as a preconditioner and solve the linear system via the General Minimized Residual method (GMRES)[23]. Many existing preconditioners are incomplete factorizations of a matrix, which means that certain elements are deleted in the factorized preconditioner. Our preconditioner is the reverse – we delete certain elements first, and then factorize the preconditioner. This is similar in spirit to the "support graph" preconditioners [6][4][13]. The iterative solver approach is used for both the threshold-based methods and the Michaelis-Menten reduction.

4 Experimental Design

The simulations for the reduction schemes and models were run on an Intel(R) Core i5-2410M CPU with a 2.3 GHz clock-rate and 6 GB DDR3 memory. We now describe the different setups involved. We also list the parameters being measured in the different setups.

4.1 Verification Experiments : MATLAB Setup

4.1.1 Direct Solver Reductions

To determine the effectiveness of the matrix reduction methods, we designed a restricted MATLAB ODE implementation. The ODE numerical method was a fixed-order implicit scheme : Backward Euler, with a fixed step-size. The Newton iteration scheme was implemented with the KLU sparse linear solver (via a MATLAB interface).

For the MATLAB setup, the direct solver approach was used, and the reduced iteration matrix M was evaluated at every iteration. The reduction of the iteration matrix was done according to a user-specified threshold value (local and global). To prevent an excessive reduction and production of a poor approximation of the Jacobian, the threshold values were restricted to less than 0.1. For the MATLAB setup, we considered the set of thresholds $\tau = \{0.0, 0.001, 0.01, 0.1\}$. This would give us an idea about the behaviour for threshold values of different orders. The Coagulation model was run for a simulated time of 700 seconds, and the Heat Shock for a simulated time of 500 seconds.

KLU is a sparse high performance linear solver that employs hybrid ordering mechanisms and elegant factorization and solve algorithms. It achieves a low level of fill-in and beats many existing solvers in run time, when used for arbitrarily sparse matrices arising in circuit simulations. Popular direct solvers such as SuperLU [11] and UMFPACK [9] take advantage of supernodes – adjacent columns containing identical non-zero patterns. For very sparse matrices not providing any supernodes, KLU gives better performance. Our Jacobians are good candidates to be used by KLU.

The initial step in the matrix factorization by KLU is decomposing the matrix into Block Triangular Form (BTF). The BTF decomposition reorders the matrix into a set of diagonal blocks. Only the diagonal blocks need to be factored. The smaller the diagonal blocks, the faster the factorization process. An example of the BTF form for the reduced Heat Shock model is shown in Fig. 1. Finding the BTF decomposition is equivalent to finding the strongly connected components of a directed graph. KLU uses the Tarjan algorithm [12], which requires a complexity of $O(n+m)$, where n is the matrix dimension, and m is the number of off-diagonal non-zeros in the original matrix.

In the MATLAB code, we measured the number of Newton iterations throughout the simulation and the linear solver time. The aim of the MATLAB experiments was to decide whether the reductions provide a speedup in the linear solver, without affecting the Newton iterations significantly. For each case we measured a Figure of Merit (FOM) value – the total linear solver time, equal to the product of the total number of Newton iterations and the average linear solving time. The smaller the FOM, the more effective the reduction. Both the Coagulation and Heat Shock models were run using a similar setup.

4.2 Matrix Reductions : Sundials Implementation

4.2.1 Direct Solver

After preliminary experiments in MATLAB, we also implemented the matrix reductions in an adaptive time-step, adaptive-order ODE solver, together with a sparse linear system solver. We ported our MATLAB code to Sundials [16][2], an ODE solver package that provides these features. Sundials provides many optimizations, such as reusing the previously computed Jacobian and support for scaled preconditioners with iterative linear solvers. Sundials also provides for tracking of different solver statistics, such as the number of function evaluations, number of Newton iterations, total simulated time steps, etc.

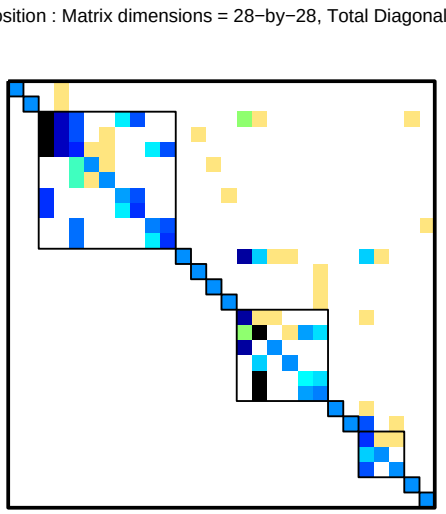

Figure 1: BTF decomposition of a reduced iteration matrix. The model is Heat Shock. The number of diagonal blocks determines the computation time.

However, Sundials lacks support for a sparse direct linear solver. Since it is crucial for our reduction process, we integrated the open source KLU implementation [1] with the Sundials package. This was done by replacing the calls to the built-in dense solver inside Sundials with corresponding calls to KLU sparse routines. Two main functions were edited :

1. **Iteration matrix setup:**
   This function computes the iteration matrix M from either the previously saved or a newly computed Jacobian J, reduces it with a threshold based reduction, and then factorizes it. The default dense solver decomposes it into LU form. This is replaced by the KLU factorization, which reduces the matrix according to the threshold schemes, reorders it into BTF and then factorizes the diagonal blocks.

2. **Linear system solve:**
   This function uses the factorized form to solve the linear system. The LU dense solve routine was replaced by the KLU solve routine.

We integrated the Sundials and KLU libraries and created a user program implementing both the Coagulation and Heat Shock models. As we shall see in the next section, the local threshold method showed no significant benefit in the MATLAB experiments. Therefore, in Sundials, only the global threshold reduction was implemented, for the range $\tau = \{0.0, 0.001, 0.005, 0.01, 0.05, 0.1\}$. To compare across thresholds, we measured the execution time using the $gettimeofday()$ function. We measured the total number of Newton iterations, the linear solver times and the total number of simulation time steps. We stored the intermediate iteration matrices, whose BTF structure could be separately visualized using the $cs\_dmspy()$ MATLAB function. We compared the overall performance of our reduced sparse solver implementation with the default dense solver implementation in Sundials. To check the effect on the accuracy due to the reduction, we measured the error norm of the difference between the obtained solution and the near-exact reference solution. The reference solution was determined by running the simulation with a very small fixed step-size (0.001) and

variable order. The Figure of Merit was the total execution time. For all the experiments, the absolute error tolerance co-efficient was set to 10^{-10} and the relative error tolerance was set to 10^{-6}.

4.2.2 Iterative Solver

As mentioned, Sundials also provides an option for using an iterative solver, along with a preconditioner. For our experiments, we used a left preconditioner for the GMRES method. As in the case of the direct solver, the iterative solver involves two implementation steps.

1. **Preconditioner matrix setup:**
   This function computes the iteration matrix M from either the previously saved or a newly computed Jacobian J, reduces it, and then factorizes it. This is used as a preconditioner.

2. **Preconditioner solve:**
   This function uses the matrix factorization of the preconditioner to solve a linear system.

Both of these functions are callbacks for Sundials, implemented in the user program. The remaining procedure was similar to the direct solver process. The same thresholds were used as in the direct solver to compare the corresponding results.

4.3 Michaelis-Menten Reduction

The Michaelis-Menten (MM) reduction was implemented on the Hockins model as a preconditioner reduction in Sundials. Since the Quasi-Steady-State for every MM reaction was valid only for certain portions of the simulation, a mechanism was designed to handle both the reduced and unreduced cases. The time instances at which Quasi-Steady-States for each MM reaction went from invalid to valid and vice versa (called switching points) were predetermined using the methods of Wu et al.

The Hockins model contains 3 MM-reactions. Three switching points were recognized in the simulated time range based on the QSS states for the MM-reactions. At each switching point the Jacobian and RHS function were changed appropriately. The system vector is also changed at the switching point outside the ODE setup, causing discontinuities. Therefore, Sundials requires that the ODE solver be restarted at each switching point. To account for one extra pseudo-species per MM-reaction, the Hockins model with 34 species was implemented as an ODE system of 37 elements.

For MM reactions sharing the enzyme E and the complex C with other non-MM reactions, an appropriate modification to the Jacobian and RHS function was applied. The details of the switching mechanism and the various modifications are described in Appendix (B). A code generator was implemented to auto-generate the modified Jacobian and RHS function. For the Coagulation model, the interdependencies among the reactions were too complex to be handled by our code-generator. The Michaelis-Menten reduction was therefore restricted to the simpler Hockins model.

In the Sundials setup, we measured the total execution time, the ODE solver statistics, and the linear solver statistics. We also stored the intermediate preconditioners to study their BTF structure, and recorded the number of non-zeros. To study the advantage of the reduction, these metrics were compared with those of the unreduced preconditioner.

5 Results

We now describe the results of the various experiments.

5.1 Matrix Reduction with Direct Solver: MATLAB Experiments

For the preliminary verification of the matrix reductions, we ran a MATLAB implementation for the Heat Shock and Coagulation models. We present here the results for the global and local threshold reduction methods on the Jacobian $h\beta J$. The simulated times for the Heat Shock and Coagulation models were 500 seconds and 700 seconds respectively. The step-size was fixed at 0.1. We measured the Newton iterations and the KLU solver time for four different thresholds: 0 (no reduction), 0.001, 0.01 and 0.1. For each threshold, we measured the relative speedup as the ratio of the linear solver time for the base case ($\tau = 0$) to the linear solver time for that threshold.

5.1.1 Global threshold

Tables 2 shows the results for the Heat Shock model.

Threshold	**Newton iterations**	**Linear solver time** (μs/iteration)	**Linear solver speedup**	**Total linear solver time (s)**
0	8535	28.50	1	0.24
0.001	9945	26.10	1.09	0.25
0.01	10934	24.50	1.16	0.26
0.1	11168	22.20	1.28	0.25

Table 2: Global threshold for step-size = 0.1: Heat Shock model.

As seen from the table, the linear solving time reduces with increasing threshold values, indicating that a better sparse structure is created. However, at the same time, we see that the number of Newton iterations increases due to the use of an approximate iteration matrix. On comparing the linear solver speedup for the largest threshold (0.1) with the relative increase in Newton iterations, the reduction up to thresholds = 0.1 is not beneficial. This is indicated by the total linear solver time. The total time increases as the threshold value increases, which is clearly undesirable.

Threshold	**Newton iterations**	**Linear solver time** (μs/iteration)	**Linear solver speedup**	**Total linear solver time (s)**
0	14002	188.20	1	2.63
0.001	14003	58.03	3.24	0.81
0.01	14004	48.82	3.85	0.68
0.1	15813	46.47	4.05	0.73

Table 3: Global threshold for step-size = 0.1: Coagulation model.

Table 3 shows the results for the Coagulation model. The linear solver experiences a better speedup with reduction on increasing threshold values. At the same time, the approximations don't significantly affect the Newton iterations. The total linear solver results clearly shows an overall improvement due to the reduction. Thus, the Coagulation model shows the benefits of the global threshold reduction.

5.1.2 Local threshold

We ran the exact same setup for the local threshold method. The results for the Heat Shock model are given in Table 4.

Threshold	**Newton iterations**	**Linear solver time** (μs/iteration)	**Linear solver speedup**	**Total linear solver time (s)**
0	8535	28.50	1	0.243
0.001	9063	26.10	1.09	0.236
0.01	9063	25.90	1.1	0.234
0.1	33002	25.20	1.13	0.831

Table 4: Local threshold for step-size = 0.1: Heat Shock model.

As seen from the table, the total linear solver does not gain a significant speedup. For $\tau_{local} = 0.1$, the number of Newton iterations becomes large, thus indicating that the approximated Jacobian is unfavourable for Newton iteration convergence. Therefore, the local reduction does not benefit the Heat Shock model.

Table 5 shows the results for the Coagulation model.

Threshold	**Newton iterations**	**Linear solver time** (μs/iteration)	**Linear solver speedup**	**Total linear solver time (s)**
0	14002	188.20	1	2.63
0.001	14002	128	1.46	1.79
0.01	14002	120	1.56	1.68
0.1	14003	100	1.88	1.40

Table 5: Local threshold for step-size = 0.1: Coagulation model.

We observe that the local threshold method produces a speedup less than 2. The improvement in the total linear solver time is correspondingly small, when compared to the global threshold. The local threshold results improve with increasing thresholds. We also measured the total linear solver time for thresholds up to 0.9, and the values dropped only to 1.04 seconds. However, this is still larger than the best total time for the global threshold at $\tau_{global} = 0.1$.

5.2 Matrix Reduction with Direct Solver: Sundials Implementation

5.2.1 Global Threshold

For the direct solver method, we ran the Sundials implementation for both the models, using the threshold values $\{0, 0.001, 0.005, 0.01, 0.05, 0.1\}$. The Sundials framework recorded the total Newton iterations and the total time steps. We measured the linear solver times for each threshold case as well as the total execution time.

Heat Shock

The Heat Shock model was run for a simulated time of 500 seconds. Figures 2 and 3 show the BTF structure for the unreduced and reduced iteration matrices respectively.

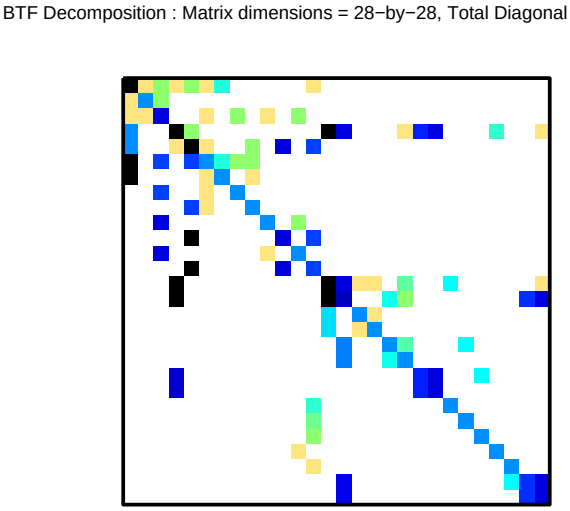

Figure 2: BTF structure of the Heat Shock iteration matrix with threshold = 0 (no reduction). Total number of diagonal blocks generated = 1 (no decomposition.)

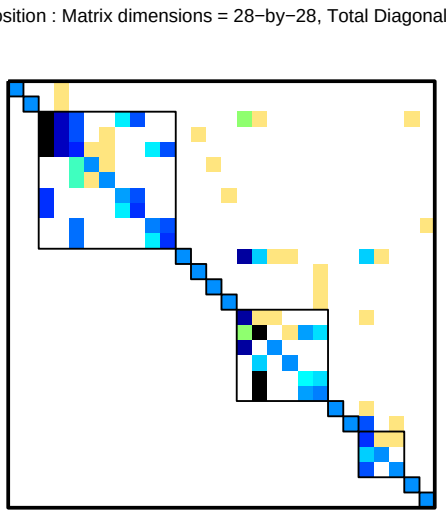

Figure 3: BTF structure of the Heat Shock iteration matrix with threshold = 0.01. Total number of diagonal blocks generated = 9.

The reduced matrix has many more diagonal blocks. Hence we expect that the reduction will provide a good speedup in the linear solver. Figures 4, 5 and 6 show the timings for the linear solver, the BDF solver, and the overall performance respectively for the Heat Shock Model.

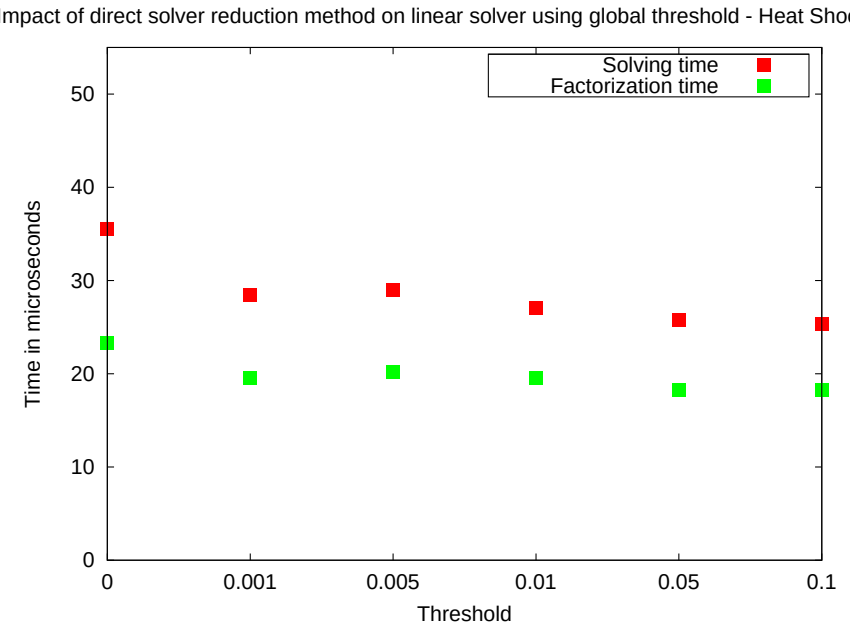

Figure 4: The time required for KLU factorization, and solving the factorized Jacobian. The drop in timing values is gradual for increasing thresholds, and very small in magnitude.

Thus we conclude that the improvement in solving and factorization time for the reduced matrices is not very significant.

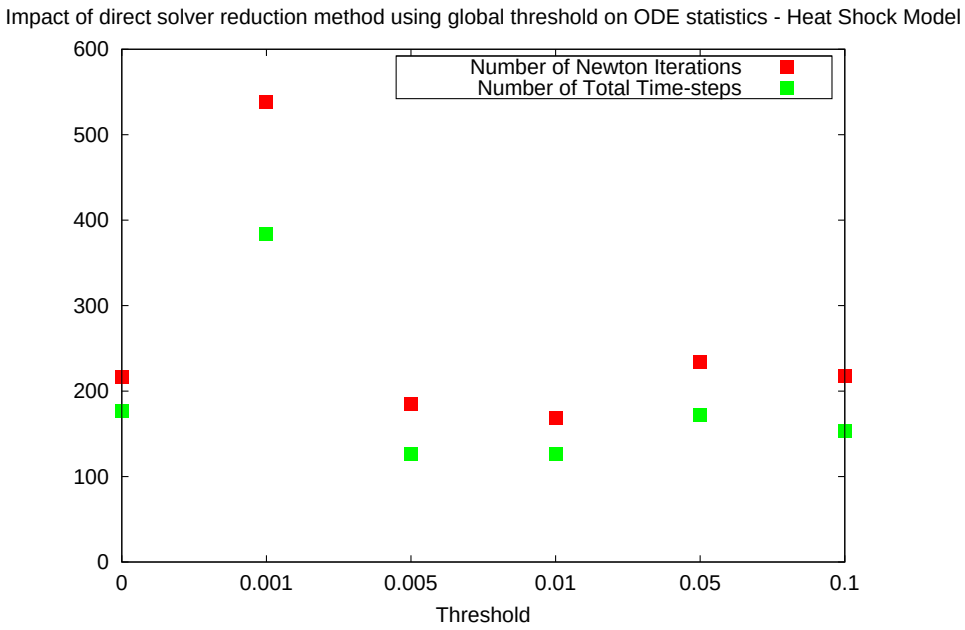

Figure 5: The total number of Newton iterations and simulation steps for the direct solver reduction method, as a function of threshold value.

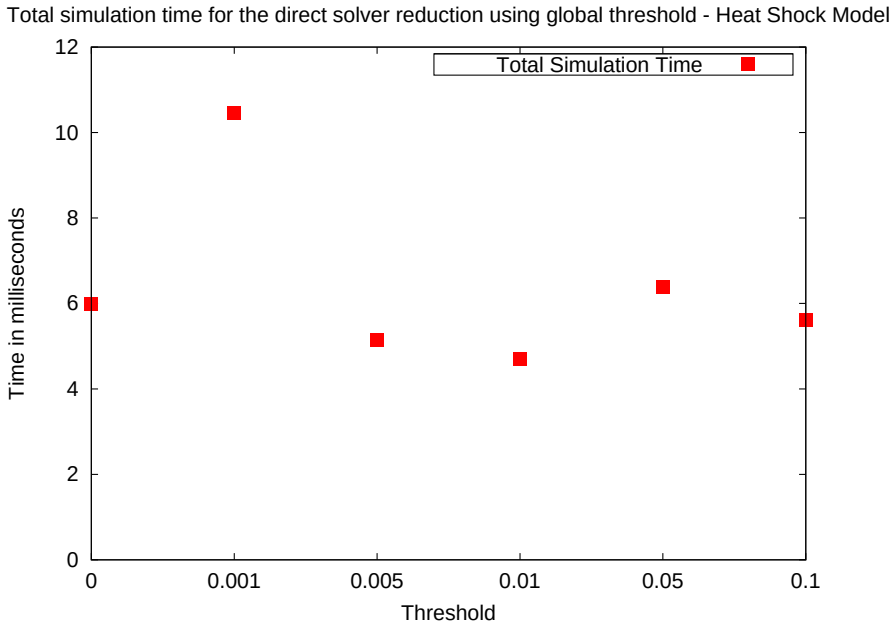

Figure 6: Total simulation time for the direct solver reduction.

Observing the linear solver time in Fig. 4, we see that the solver speedup is not significant with increasing reduction. Except for $\tau_{global} = 0.001$, the total iterations and simulation steps in Fig. 5 are roughly the same for the reduced matrices (w.r.t $\tau_{global} = 0$).

For $\tau_{global} = 0.01$, we get a small speedup of `~1.25` in the total execution time. There is a slowdown for $\tau_{global} = 0.001$ and $\tau_{global} = 0.05$. To summarize, the overall performance is not

very beneficial. We present an overall summary for the global reduction on the iteration matrix for different threshold values in Table 6. The error norm shows that the reductions maintain the accuracy of simulation.

Threshold	**Execution Time (FOM) (ms)**	**Error Norm (mol/L)**
0	5.98	2.88e-09
0.001	10.46	6.32e-11
0.005	5.15	1.78e-10
0.01	4.70	5.39e-11
0.05	6.39	5.51e-11
0.1	5.60	5.35e-11

Table 6: Global threshold reduction for direct solver reduction: Heat Shock Model. The total execution time serves as the Figure of Merit for the reduction.

Coagulation

We now present results for direct solver reduction on the Coagulation model. The model was run for a simulated time of 700 seconds. Figures 7 and 8 show the BTF structure for the iteration matrix for the unreduced and reduced cases, respectively.

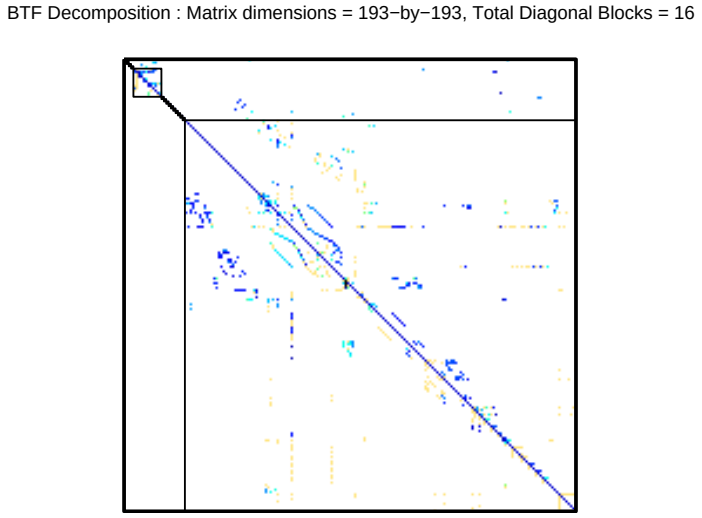

Figure 7: BTF structure of the Coagulation iteration matrix for threshold = 0 (no reduction). Total diagonal blocks generated in BTF structure = 16.

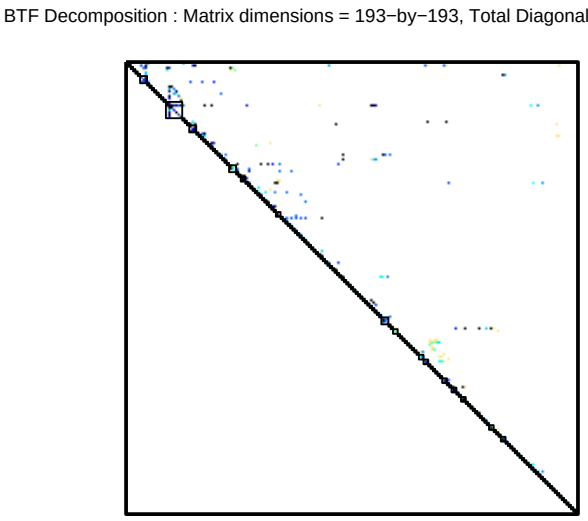

Figure 8: BTF structure of the Coagulation iteration matrix for threshold = 0.1 (reduction). Total diagonal blocks generated = 169.

The unreduced matrix has only a few diagonal blocks. On the other hand, the reduced iteration matrix has a large number of smaller blocks (169). The reduced iteration matrices for other thresholds have a similar decomposition structure, with nearly the same number of diagonal blocks. Fig. 9, which shows the time required for the factorization and solving of the iteration matrices reduced with different threshold values, confirms this.

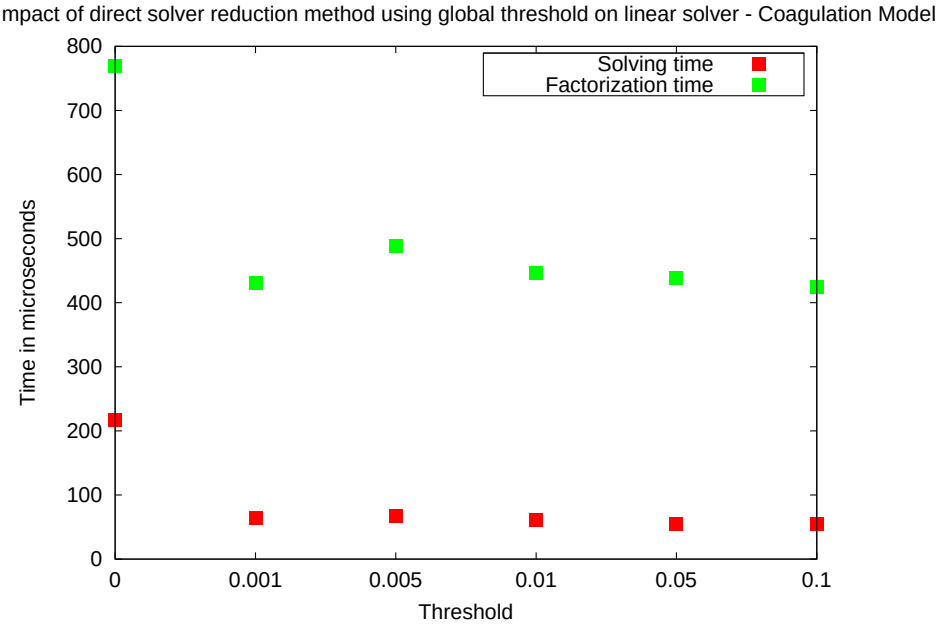

Figure 9: The time required for KLU factorization, and solving the factorized iteration matrix.

The factorization and solving time thus drop significantly with a threshold even as low as 0.001. Overall a best-case 3X speedup in solving time is observed due to the global reduction.

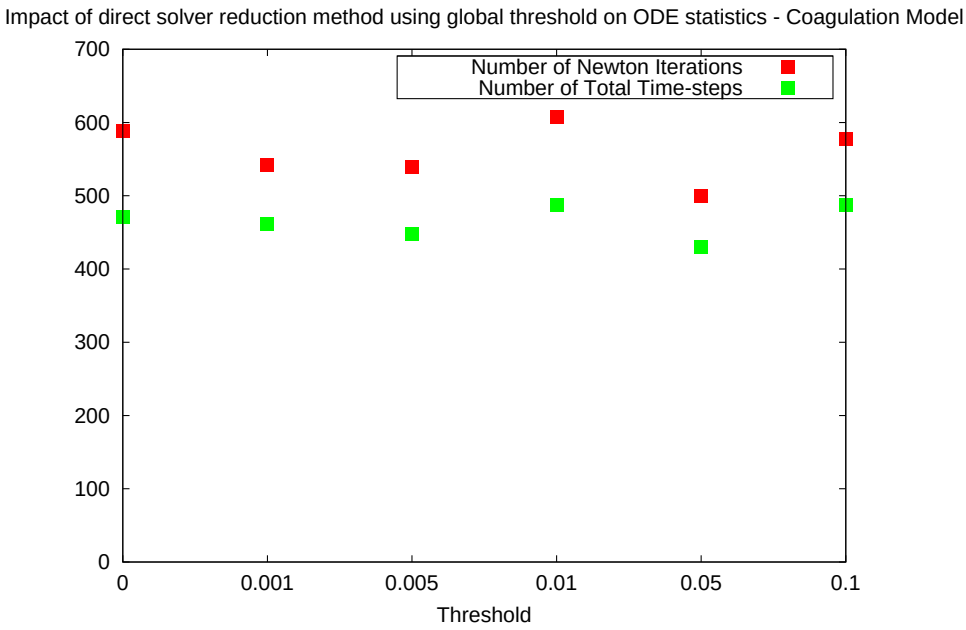

Figure 10: The total Newton iterations and simulation steps for the direct solver reduction method as a function of threshold size.

The total time for the entire simulation of the Coagulation model under different threshold values is shown in Fig. 11.

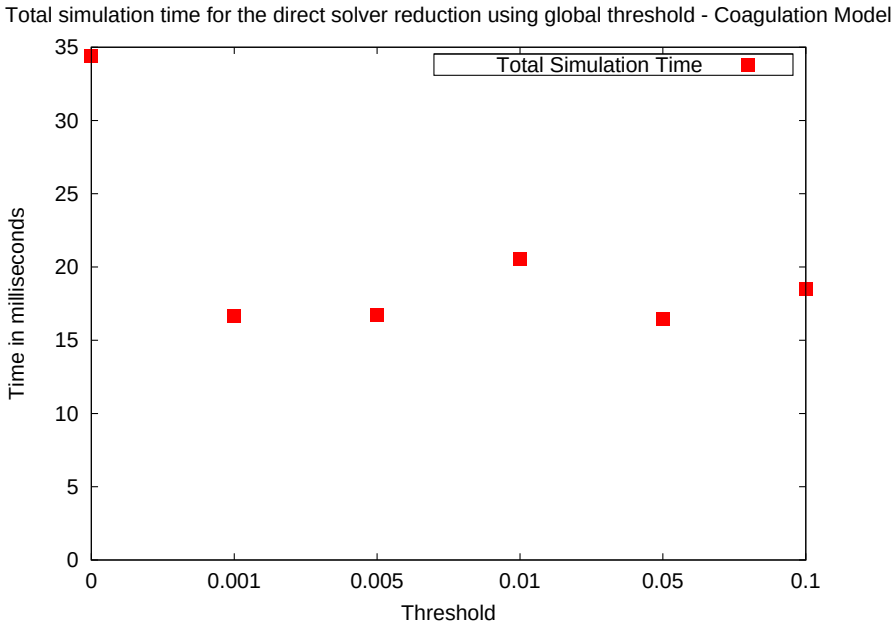

Figure 11: Measurement of the total simulation time for the direct solver reduction. We obtain a best speedup of `~2.07` at $\tau_{global} = 0.05$.

We obtained a maximum `~2.07` overall speedup. The overall results are summarized in Table 7.

Threshold	Execution Time (FOM) (ms)	Error Norm (mol/L)
0	34.4	2.90e-05
0.001	16.6	4.91e-05
0.005	16.7	1.12e-03
0.01	20.5	1.12e-03
0.05	16.4	1.13e-03
0.1	18.5	1.67e-05

Table 7: Global threshold reduction for direct solver reduction: Coagulation Model.

5.2.2 Local Threshold

For the local threshold, we present the BTF structure obtained from the Sundials setup to explain the MATLAB results. Figures 12 and 13 show the structures for the local threshold reduction on the Heat Shock and Coagulation models respectively.

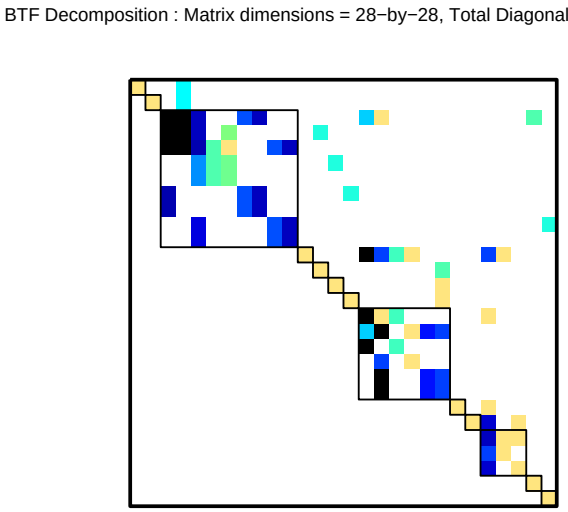

Figure 12: BTF structure of the Heat Shock iteration matrix with threshold = 0.1 (reduction). Total diagonal blocks generated = 13.

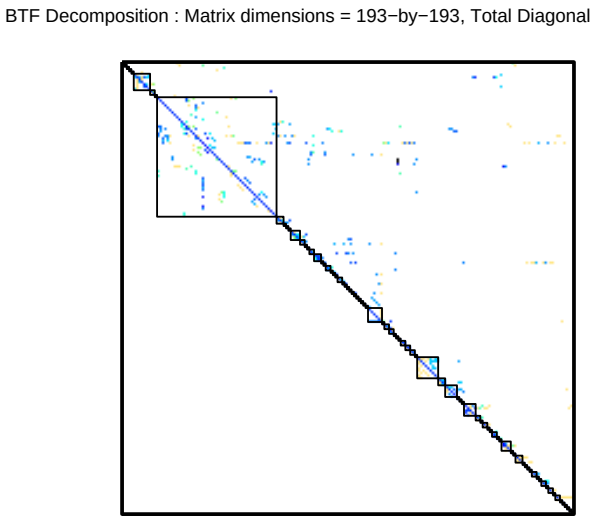

Figure 13: BTF structure of the Coagulation iteration matrix with threshold = 0.1 (reduction). Total diagonal blocks generated = 85.

For the Heat Shock model, the number of diagonal blocks is the same as in the global threshold method. The Coagulation model, however, produces much fewer blocks as compared to the global threshold method. Overall, the poor BTF structures explain the low linear solver speedup in both the models.

5.3 Iterative Solver with Preconditioner Reduction: Sundials Implementation

5.3.1 Global Threshold

For the iterative solver method with preconditioner, again implemented with Sundials, we experimented on both the Coagulation and Heat Shock models, and used the same thresholds as used for the direct solver reduction.

Heat Shock

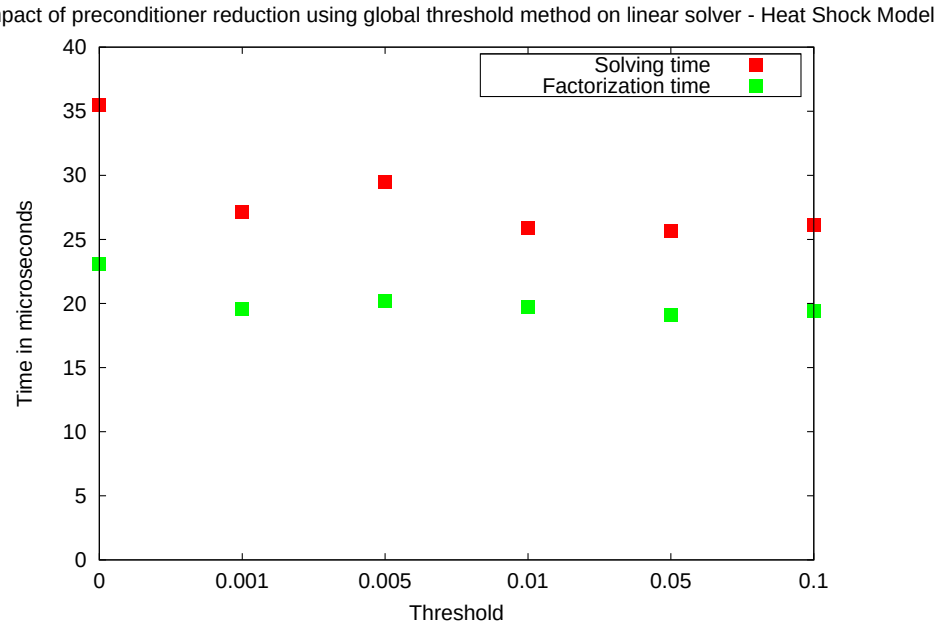

Figure 14: The time in seconds required for preconditioner factorization, and preconditioning solves time. The drop in timing values is gradual for increasing thresholds, and insignificant in magnitude.

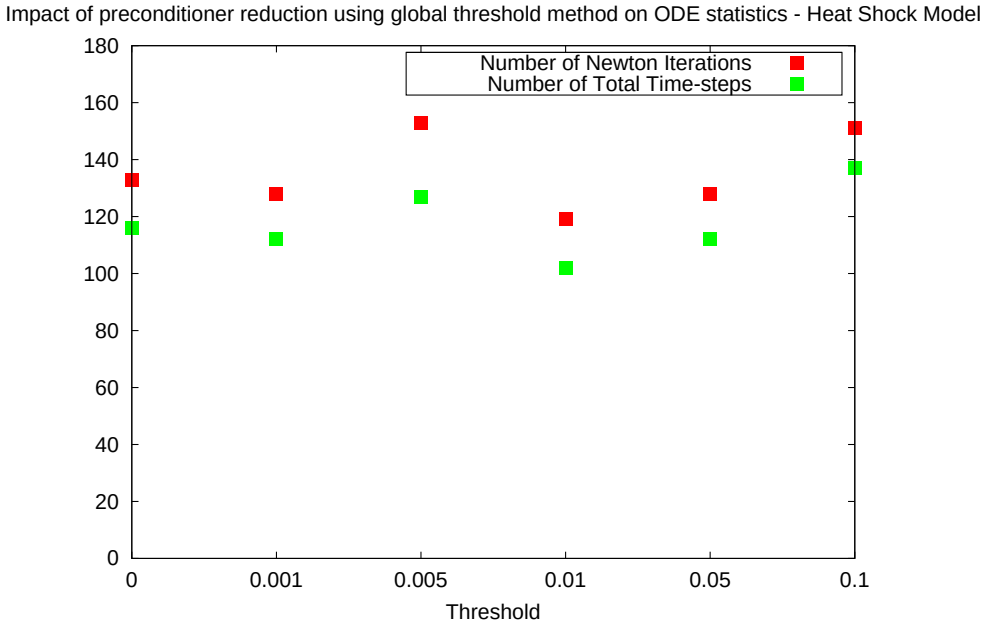

Figure 15: Total Newton iterations and simulation steps for the iterative solver reduction method.

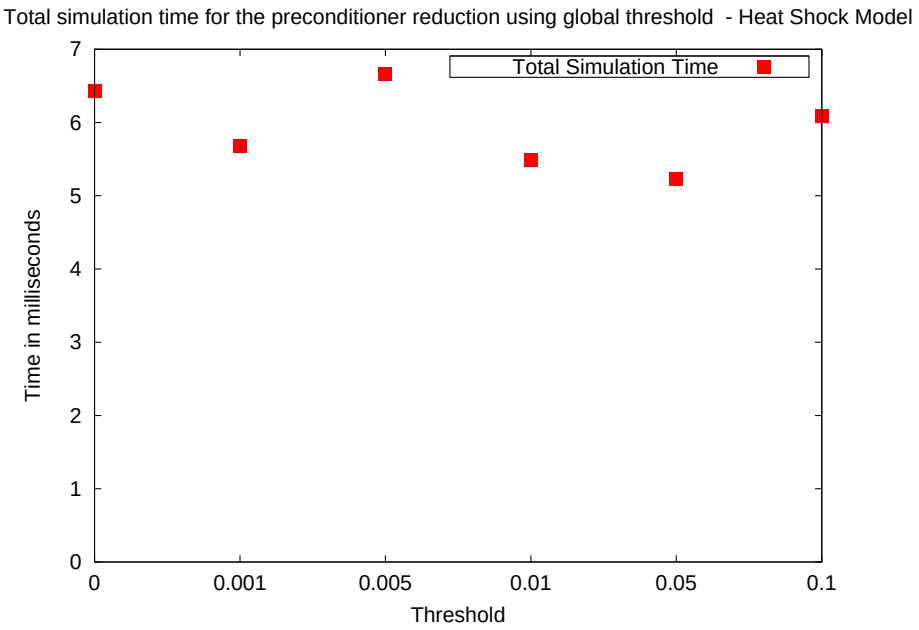

Figure 16: Total execution time for the iterative solver reduction.

The results in Fig. 14 show that, as with the direct solver reduction, reducing the preconditioner does not improve the linear solver time.

The ODE characteristics are shown in Fig. 15. Since the reduced matrix was used as a preconditioner, we expected only an improvement in the linear solver time without any significant change in the Newton iterations and the time steps. On investigation into Sundials, it was found that the solution of the linear system under different preconditioners produced different error estimates in the ODE solver. The error estimates at each step are used by the solver to determine the step-size and order for the next step. Due to the difference in error estimates, the ODE solver varied in the order and step-size decisions across different preconditioners. The total time-steps vary by at most 12% w.r.t. the base case ($\tau_{global} = 0$).

From the total timing results in Fig. 16, we see that the speedup is not significant with increasing threshold values. The best speedup obtained is 1.23 for the $\tau_{global} = 0.05$. Table 8 summarizes the results for the Heat Shock model. The error norm column indicates that the iterative solver results are roughly similar to the direct solver results.

Threshold	**Execution Time (FOM) (ms)**	**Error Norm (mol/L)**
0	6.42	4.62e-10
0.001	5.67	6.31e-10
0.005	6.65	6.82e-11
0.01	5.49	7.29e-10
0.05	5.23	2.08e-09
0.1	6.08	5.27e-09

Table 8: Global threshold reduction for iterative solver reduction: Heat Shock model. The total execution time serves as the Figure of Merit for the reduction.

Coagulation

The Coagulation model, on the other hand, gives far more consistent results for the linear solver timings, the ODE statistics and the overall timings, as indicated by the Figures 17, 18 and 19.

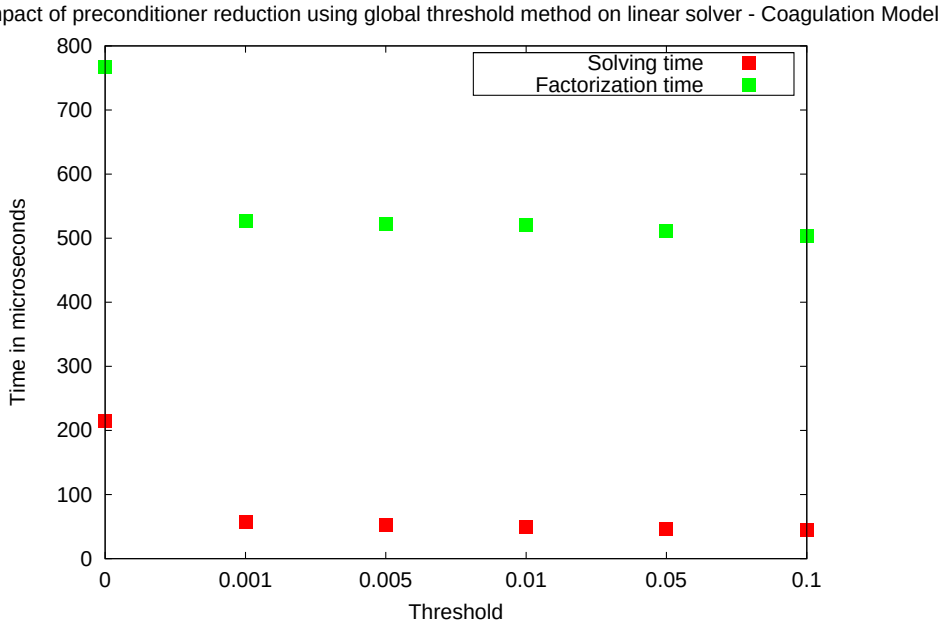

Figure 17: The time in seconds required for preconditioner factorization, and preconditioning solves time.

The solving time improves by a factor of 3 with the reduction of even such a low threshold as 0.001. The statistics are similar to those of the iteration matrix reductions for the global threshold method shown in Fig. 9.

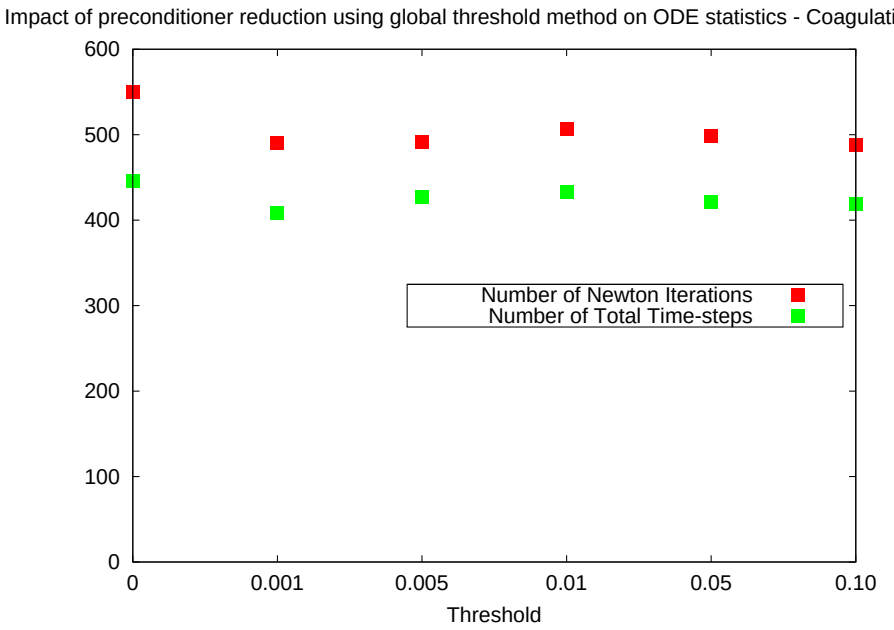

Figure 18: The variation of the number of simulated time steps and the total Newton iterations.

Fig. 18 displays the effect of the reduction on the total number of simulation steps and Newton iterations. As in the case of Heat Shock, there is a small variation in the number of time-steps and Newton iterations across different thresholds. The largest difference w.r.t. the base case is 8.5%.

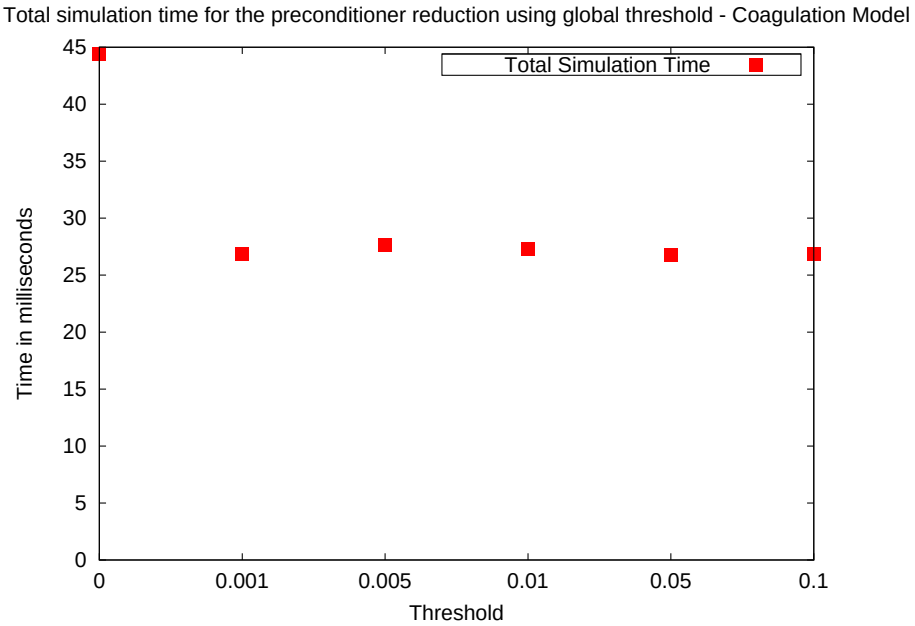

Figure 19: Measurement of the total execution time for the iterative solver reduction for different threshold values. We get a maximum of `~1.65` speedup at $\tau_{global} = 0.001$.

The total execution timing results are consistent with the linear solver and ODE results. The general trend is a speedup for τ_{global} greater than 0. Comparing Figs. 11 and 19, we also note

that the iterative solver method is more expensive than the direct solver method in terms of total execution time. The speedup obtained in the iterative method is also less than that obtained in the direct solver method. The final summary of the preconditioner reduction is shown in Table 9. The error norm results are roughly comparable to direct solver results for the Coagulation model.

Threshold	**Execution Time (FOM) (ms)**	**Error Norm (mol/L)**
0	44.3	7.45e-05
0.001	26.8	9.73e-05
0.005	27.6	1.57e-04
0.01	27.2	1.20e-04
0.05	26.8	3.42e-04
0.1	26.5	4.57e-04

Table 9: Global threshold reduction for iterative solver reduction: Coagulation Model.

5.4 Michaelis-Menten Reduction: Sundials Implementation

We ran the Michaelis-Menten reduced and unreduced implementation of the Hockins model in Sundials. Tables 10, 11 and 12 summarize the results for the different cases. We recorded the ODE characteristics provided by Sundials – the total number of simulation steps, Newton iterations, function evaluations, preconditioner evaluations, and preconditioner solves. We also stored the intermediate preconditioner structures, and measured the number of non-zero elements and the factorization and solving time for those preconditioners.

Preconditioner	**Simulation steps**	**Newton iterations**	**Preconditioner evaluations**	**Preconditioner solves**
Unreduced	392	437	7	1026
Reduced	415	548	9	1139

Table 10: ODE performance comparison for Michaelis-Menten reduction.

Preconditioner	**Factorization time (μs)**	**Solving time (μs)**	**Non-zeros**
Unreduced	33.2	42.2	158
Reduced	31.6	40.9	149

Table 11: Solver performance comparison for Michaelis-Menten reduction.

Preconditioner	Total Simulation Time (FOM) (ms)
Unreduced	12.8
Reduced	14.4

Table 12: Overall performance comparison for Michaelis-Menten reduction.

The ODE solver statistics shown in Table 10 do not differ much in the two cases. Due to the ODE solver resets in the reduction, at each switching point the step-size was reinitialized to a small value. Though the step-size eventually increased to a large value (even greater than the unreduced case), extra time-steps were spent in doing so. Therefore, the total time-steps and Newton iterations are more in the reduced case. The ODE solver reset is necessary because the modification of the system vector outside the ODE setup creates function discontinuities. It may be possible to consider alternative implementations where such discontinuities might be avoided. Without the resets, the total number of time steps may be reduced considerably.

The reduction does not provide any advantage in the preconditioner solving time. Table 11 indicates that the number of non-zero elements was only reduced by 9 with MM reduction. Thus, the total simulation time therefore does not reduce with the MM reduction. The BTF structure plots for the full and MM-reduced preconditioners are shown in the Figures 20 and 21.

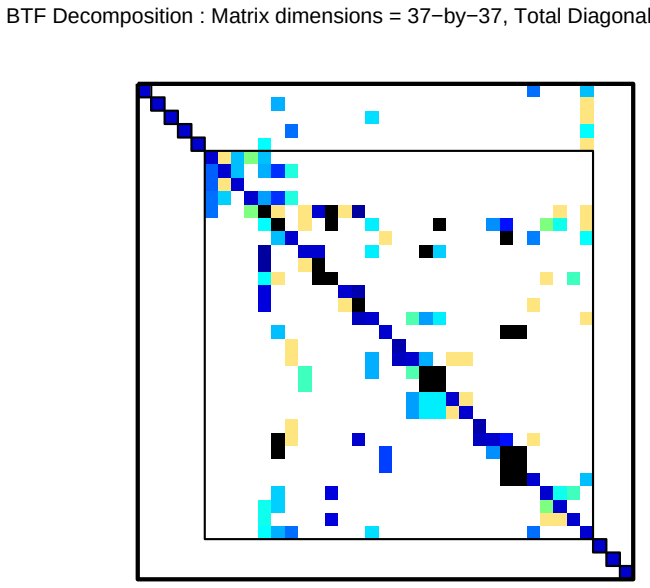

Figure 20: BTF structure for iteration matrix of Hockins model without reduction. Total diagonal blocks = 9.

The unaffected structure with the reduction confirms the absence of improvement in the linear solving time. In retrospect, we see that the basic MM reduction (Equation (6) in Section 3.1.3) does not change the vertex reachability relationships in the reaction graph. Therefore it is not surprising that the effect on the BTF structure is small.

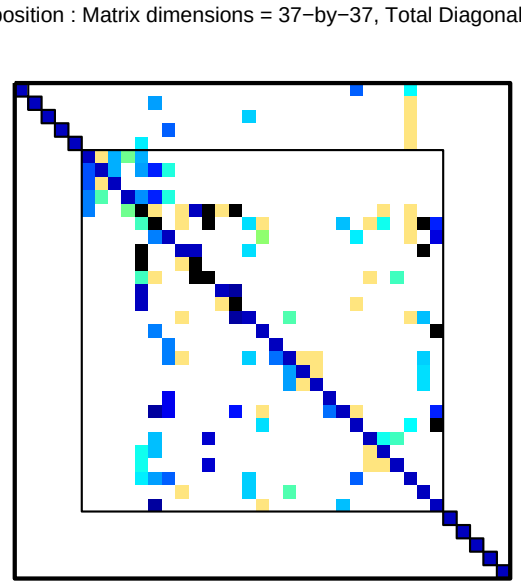

Figure 21: BTF structure for iteration matrix of Hockins model with MM-reduction. Total diagonal blocks = 11.

6 Comparison among Proposed Techniques

We have considered different type of reductions on three models – Heat Shock, Coagulation and Hockins Coagulation. We now provide a comparison amongst all the reductions.

6.1 Threshold-based Reduction

We investigated two reductions – a local row-wise and column-wise threshold, and a global threshold method in a direct linear solver MATLAB implementation. The global threshold reduction showed promising results for the Coagulation model, with a considerable improvement in the linear solves, without affecting the number of Newton iterations. The reductions did not provide a significant improvement for the Heat Shock model. The local threshold, on the other hand, was ineffective for both models.

The results from the Sundials implementation of the matrix reductions were similar to the results generated with the MATLAB setup. Table 13 shows the performance of the iteration matrix reduction and preconditioner reduction using the global threshold. We also show the dense direct and iterative solver results for the unreduced matrices.

Type of reduction	Characteristic	Coagulation model	Heat Shock Model
Direct Dense Solver	Execution time (ms)	84.9	3.5
Direct Sparse Solver	Execution time for unreduced matrix (ms)	34.4	5.98
	Best execution time for reduced matrix (ms)	16.4	4.77
	Best speedup v/s sparse unreduced	2.07	1.25
	Best speedup v/s dense unreduced	5.17	0.73
Iterative Dense Solver	Execution time (ms)	120.0	4.1
Iterative Sparse Solver	Execution time for unreduced preconditioner (ms)	44.3	6.42
	Best execution time for reduced preconditioner (ms)	26.8	5.23
	Best speedup v/s sparse unreduced	1.65	1.23
	Best speedup v/s dense unreduced	4.47	0.78

Table 13: Comparison of matrix reductions for the Coagulation and Heat Shock models, using Sundials.

The matrix reduction with the direct solver provides a better speedup and is faster than the preconditioner reduction in the iterative solver. The BTF structure of the reduced matrices for both the models exhibited an increase in the number of fine diagonal blocks. The increase was from 1 to 13 diagonal blocks for the Heat Shock model, and from 16 to 169 blocks for the Coagulation model with the global reduction method. However, the Heat Shock model, being a much smaller one, and generating fewer sub-matrices in the decomposition, did not benefit significantly from the linear solver speedup. The local threshold method created fewer diagonal blocks for both the models as compared to the global threshold method, thus explaining the MATLAB results. The speedup against the dense implementation shows that the Heat Shock model did not benefit from using the KLU solver.

6.2 Michaelis-Menten Reduction

Finally, we implemented the Michaelis-Menten nonlinear model reduction to improve the Block Triangular Form (BTF) of the preconditioner by exploiting the structure of the underlying chemical system. On a model containing 34 species and 3 Michaelis-Menten reactions, it was found that the preconditioner of the reduced system did not lead to a significantly better matrix structure. The ODE solver statistics were not affected significantly, and no overall speedup was obtained.

7 Conclusion

We explored the effectiveness of various matrix reduction techniques for ordinary differential equations arising from chemical reaction systems. The threshold based methods created an improved sparse structure in the reduced matrices. The speedup obtained in the sparse matrices depended on the number of diagonal blocks produced in their BTF structure. In the Coagulation model, the global threshold method produced almost twice the number of blocks than the local threshold method, and therefore was more effective. The Heat Shock model produced fewer diagonal blocks under both reductions, and did not benefit from threshold reductions.

The Michaelis-Menten reduction was performed on the chemical system to produce a sparse matrix. The number of elements reduced depended on Michaelis-Menten-type reactions in the system. However, the total diagonal blocks was unaffected as a result of reduction. On studying the effect of the reduction, it was found that the Michaelis-Menten reduction does not change the total strongly connected components in the equivalent graph of the Jacobian. Therefore, the linear solver did not gain any advantage due to reduction.

Since we have experimented on only a few models, it is hard to draw conclusions for the effectiveness of the reductions for the entire class of models they represent. However, based on the results, the threshold reductions are recommended for any large chemical reaction system that produces matrices of considerable sparsity. Matrices containing a large number of extremely small elements can be well reduced by the global threshold method. It is hard, however, to determine which threshold value provides the best reduction, though it is recommended to experiment with smaller threshold values (< 0.1).

To conclude, we have explored a few matrix reductions on a certain class of ODE problems. The next step is to evaluate these methods on other problems, and determine if the reductions are beneficial for the entire class. Determining the best threshold based on developed theory, or using an automatic method can help reduce the development time for manual performance-tuning for each model.

A Michaelis-Menten Reduction : Details

Consider the following chemically reacting system :

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C \overset{k_3}{\to} E + P. \tag{7}$$

This is a typical example of an Enzyme-Substrate-Complex-Product system. The corresponding differential equations are given by

$$\frac{d}{dt}\begin{pmatrix}[E]\\[S]\\[C]\\[P]\end{pmatrix} = \begin{pmatrix}-k_1[E][S] + k_2[C] + k_3[C]\\ -k_1[E][S] + k_2[C]\\ k_1[E][S] - k_2[C] - k_3[C]\\ k_3[C]\end{pmatrix}. \tag{8}$$

The solving of the 4-species system requires a 4×4 Jacobian as part of the Newton iteration, given by

$$J = \begin{pmatrix}-k_1[S] & -k_1[E] & k_2 + k_3 & 0\\ -k_1[S] & -k_1[E] & k_2 & 0\\ k_1[S] & k_1[E] & -k_2 - k_3 & 0\\ 0 & 0 & k_3 & 0\end{pmatrix}. \tag{9}$$

Instead of working on the entire 4×4 system, the nature of the reaction-rates and the species concentrations in the reaction allows us to approximate the system shown in (7) by a smaller system. This type of approximation is based on the Michaelis-Menten kinetics [24]. The equivalent reduced reaction is given by

$$S \overset{k'}{\to} P. \tag{10}$$

The concentrations for E and C need to be tracked and we also need to determine the value for the new rate constant k'.

Let $[E_0]$ be the initial concentration of E at the start of reduction. Using the QSS assumptions on the complex C and the fact that the total molecules of E remain constant, the kinetics for S and P can be written (according to Michaelis-Menten) as

$$\frac{d[P]}{dt} = k_3[C] = k_3\frac{[S][E_0]}{K_m + [S]}, \tag{11}$$

$$\frac{d[P]}{dt} = -k_3[C] = -k_3\frac{[S][E_0]}{K_m + [S]}, \tag{12}$$

where $K_m = \frac{k_2+k_3}{k_1}$. Rewriting, we have

$$\frac{d[P]}{dt} = k'[S], \tag{13}$$

$$\frac{d[S]}{dt} = -k'[S]. \tag{14}$$

where, $k' = \frac{k_3[E_0]}{K_m+[S]}$.

The new chemical equation consists of a 3-species system – S, P and E_0, given as

$$S \overset{k'(E_0)}{\rightarrow} P. \tag{15}$$

The concentrations for E and C, eliminated from the ODE system, can be determined algebraically, as

$$[E] = \frac{K_m[E_0]}{K_m + [S]} \tag{16}$$

$$[C] = \frac{[S][E_0]}{K_m + [S]}. \tag{17}$$

Because we need a generalized ODE system that works in both the reduced and non-reduced cases, the corresponding Jacobian and the RHS function for the reduced systems contain all 5 species – $\{E, S, C, P, E_0\}$.

$$\frac{d}{dt}\begin{pmatrix}[E]\\[S]\\[C]\\[P]\\[E_0]\end{pmatrix} = \begin{pmatrix}0\\ -\frac{k_3[E_0][S]}{K_m+[S]}\\ 0\\ \frac{k_3[E_0][S]}{K_m+[S]}\\ 0\end{pmatrix} \tag{18}$$

$$J_{reduced} = \begin{pmatrix}0 & 0 & 0 & 0 & 0\\ 0 & -\frac{k_3[E_0]}{(K_m+[S])^2} & 0 & 0 & -\frac{k_3[S]}{K_m+[S]}\\ 0 & 0 & 0 & 0 & 0\\ 0 & \frac{k_3[E_0]}{(K_m+[S])^2} & 0 & 0 & \frac{k_3[S]}{K_m+[S]}\\ 0 & 0 & 0 & 0 & 0\end{pmatrix} \tag{19}$$

Note that the rows corresponding to E and C are all zeros, thus they are not updated during the Newton iteration of the reduced system. At the end of the simulation, the concentrations of E and C can be updated according to the equations (16) - (17). We can therefore solve a much sparser system, by eliminating the computation of two species – E and C (zero-ing out the Jacobian row), and keeping track of a new pseudo-species – E_0. This serves as the basis for the model reduction approach. We can similarly reconstruct $J_{unreduced}$ for the 5-species system using Equation (9), to obtain

$$J_{unreduced} = \begin{pmatrix}-k_1[S] & -k_1[E] & k_2+k_3 & 0 & 0\\ -k_1[S] & -k_1[E] & k_2 & 0 & 0\\ k_1[S] & k_1[E] & -k_2-k_3 & 0 & 0\\ 0 & 0 & k_3 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{pmatrix}. \tag{20}$$

The models we consider for the Michaelis-Menten (MM) reduction also contain other type of reactions that are not candidates for the Michaelis-Menten reduction. Additionally, the non-MM may reactions involve the E and C species on both the reactant and product side. Since these species are being tracked during the reduction process using a pseudo-species, E_0, such that $[E_0] = [E] + [C]$. We need to replace the occurrences of E/C with E_0. If the replacement is on the reactant side, the rate constant is modified, whereas if it is on the product side, the pseudo-species concentration is modified, in place of its corresponding species. The strategy is explained case-wise as follows.

As stated in Equation (7), consider an MM reaction R_1, given by

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C \overset{k_3}{\rightarrow} E + P, \tag{21}$$

and a non-MM reaction R_2,

$$A + B \overset{k}{\rightarrow} D. \tag{22}$$

1. Suppose an Enzyme E of an MM-reaction R_1 occurs on the reactant side of R_2 (i.e., $B = E$) :

   $$A + E \overset{k}{\rightarrow} D. \tag{23}$$

   When the reaction R_1 is being reduced by the MM-scheme, the rate of change of $[D]$ is given by

   $$\frac{d[D]}{dt} = k[A][E]. \tag{24}$$

   Using Equation 16, this is rewritten as,

   $$\frac{d[D]}{dt} = \frac{k[A][E_0]K_m}{K_m + [S]} = k''[A][E_0], \tag{25}$$

   where $k'' = \frac{kK_m}{K_m + [S]}$ Thus, we are now using the concentration of E_0 to compute the rate of change of $[D]$. Similar procedure is followed if $B = C$.

2. Suppose the enzyme of R_1 occurs on the product side of the reaction R_2 :

   $$A + B \overset{k}{\rightarrow} E. \tag{26}$$

   We keep track of E_0 instead of E, i.e.

   $$\frac{d[E_0]}{dt} = k[A][B]. \tag{27}$$

   A similar procedure is followed if $D = C$.

To summarize, we consider the effect of reduction of a MM reduction on other reactions that have E or C as a reactant or a product. For the MM reaction, with enzyme E and complex C, we consider a pseudo-species E_0 whose concentration is $[E_0] = [E] + [C]$. We halt the update of $[E]$ and $[C]$, and instead update $[E_0]$ as shown in Equation (27). The products of the non-MM reaction are updated based on $[E_0]$ as shown in Equation (25). At any point, $[E]$ and $[C]$ can be recovered using equations (16) and (17) respectively. Therefore, the rates of the dependent non-MM reactions now contain $[E_0]$ instead of $[E]$ and $[C]$, and $[E_0]$ also depends on the evolution of these reactions. The Jacobian and the RHS terms for the involved species are changed accordingly.

B Michaelis-Menten Reduction : Setup

As mentioned in Section 4.3 we need to implement a generic form for the right-hand side function and the system's Jacobian. Both these ODE components assume a different form when the Michaelis-Menten (MM) reduction is used. We should be able to switch to the reduced form when the Quasi Steady State conditions $[E_0] << [S]+K_m$ is met. and revert back to the unreduced form, when the valid condition is violated. To handle the switching, we maintain a flag that separates the two cases.

$$\frac{d}{dt}\begin{pmatrix}[E]\\[S]\\[C]\\[P]\\[E_0]\end{pmatrix} = (!flag)(RHS_{unreduced}) + (flag)(RHS_{reduced}). \tag{28}$$

Thus, when the flag is on, the unreduced RHS is used, and when the flag is off, the reduced right-hand side function is used. The Jacobian (now a 5×5 system) also is constructed in a similar manner using the same flag, i.e.

$$J = (!flag)(J_{unreduced}) + (flag)(J_{reduced}). \tag{29}$$

In this manner, we maintain a list of flags - $flag_1, flag_2 \cdots$, one per MM reaction.
Apart from this, we need to handle a number of special cases for the non-MM reactions. Consider a non-MM reaction of the form :

$$A + B \rightarrow C + D.$$

The following cases may occur –

1. Either A or B is an enzyme/complex of some MM-reaction.
2. A is an enzyme/complex for MM-reaction R_1 and B is an enzyme/complex for MM-reaction R_2.
3. C and D are enzymes/complexes of different MM-reactions R_1 and R_2.

We observed that due to the possibility of simultaneous occurrence of above cases, the resultant Jacobian and right-hand side would become extremely complex, For simplification purposes, we only consider the cases that actually happen in the model we are going to simulate. The model that we eventually adapted : the Hockins Coagulation model contains only cases 1 and 3. It consists of 34 species and 3 MM reactions, and no two MM-reactions share the same E/C species.

Even with such a small model with a lesser complexity, it was difficult to hand-code the right-hand side and Jacobians to handle all the cases. We therefore implemented a code-generator which creates these ODE components.

References

[1] KLU : Clark-Kent Factorization Package. `http://www.cise.ufl.edu/research/sparse/klu/`.

[2] SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers). `https://computation.llnl.gov/casc/sundials/main.html`.

[3] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM: Society for Industrial and Applied Mathematics, July 1998.

[4] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):930–951, December 2005.

[5] Binita Bhattacharjee, Douglas A. Schwer, Paul I. Barton, and William H. Green. Optimally-reduced kinetic models: Reaction elimination in large-scale kinetic mechanisms. *Combustion and Flame*, 135(3):191 – 208, 2003.

[6] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, March 2003.

[7] George E. Briggs and John B. S. Haldane. A note on the kinetics of enzyme action. *Biochem. J.*, 19:338–339, 1925.

[8] Peter N. Brown, G. D. Byrne, and A. C. Hindmarsh. VODE: A variable-coefficient ODE solver. *SIAM J. Sci. Stat. Comput.*, 10(5):1038–1051, September 1989.

[9] T. Davis and I. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):140–158, 1997.

[10] Timothy A. Davis and Ekanathan Palamadai Natarajan. Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Trans. Math. Softw.*, 37(3):36:1–36:17, September 2010.

[11] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[12] I. S. Duff and J. K. Reid. An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Trans. Math. Softw.*, 4(2):137–147, June 1978.

[13] Keith Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, October 1996. CMU CS Tech Report CMU-CS-96-123.

[14] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, Berlin, 2nd revised edition edition, 1991.

[15] Desmond J. Higham and Lloyd N. Trefethen. Stiffness of ODEs. *BIT*, 33:285–303, 1993.

[16] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, September 2005.

[17] M.F. Hockin, K.C. Jones, S.J. Everse, and K.G. Mann. A model for the stoichiometric regulation of blood coagulation. *J Biol Chem*, 277(21):18322–33, 2002.

[18] Kenneth A. Johnson and Roger S. Goody. The original Michaelis Constant: Translation of the 1913 Michaelis-Menten paper. *Biochemistry*, 50(39):8264–8269, 2011.

[19] H. Kurata, H. El-Samad, T.-M. Yi, M. Khammash, and J. Doyle. Feedback regulation of the heat shock response in E. Coli. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 837 –842 vol.1, 2001.

[20] D. Luan, M. Zai, and J.D. Varner. Computationally derived points of fragility of a human cascade are consistent with current therapeutic strategies. *PLoS Comput Biol*, 3(7):e142, 2007.

[21] Miles S. Okino and Michael L. Mavrovouniotis. Simplification of mathematical models of chemical reaction systems. *Chem Rev*, 98(2):391–408, 1998.

[22] Linda Petzold and Wenjie Zhu. Model reduction for chemical kinetics : An optimization approach. *AiChe Journal*, 45:869–886, 1999.

[23] Youcef Saad and Martin Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.

[24] W.R. Salzman. Chemical Kinetics (The Rates of Chemical Reactions) : Enzyme Kinetics. `http://www.chem.arizona.edu/~salzmanr/480a/480ants/enzymekn/enzymekn.html`, 2000.

[25] L. A. Segel and M. Slemrod. The quasi-steady state assumption: A case study in perturbation. *SIAM Rev.*, 31(3):446–477, September 1989.

[26] S. Wu, J. Fu, H. Li, and L. Petzold. Automatic identification of model reductions for discrete stochastic simulation. *J Chem Phys*, 137(3):034106, 2012.