Fast solver techniques for algebraic equations resulting from the Quasi Steady State Approximation

Fabian Mauss*
*Brandenburg University of Technology / Thermodynamics and Thermal Process Engineering , Cottbus, Germany

I. Introduction

In this abstract we discuss strategies for efficient solving ordinary differential equations together with non-linear algebraic equations. This set of equations is typical for kinetic system for which a high number of quasi steady state species have been defined. There are two possible strategies which can be applied with regular chemistry solvers. 1) operater splitting methods, and 2) nested solver technolgies. For operator splitting techniques there is no high demand on the speed of the solver for the set of algebraic equations. Each timestep will be calculated faster compared to original problem, since the size of each problem is reduced, and the stiffness of the unsteady problem is reduced, since the timescales have been clearly seperated. However operator splitting techniques are not fully implicit, and a reduction of the timestep size is necessary. This decreases the performance of the reduced chemical mechanism. Nested solver strategies are fully implicit, and do not experience the problem of limited timestep sizes. However there is a high demand on the speed and accuarcy of the algebraic solver.

In most chemistry software implicit Newton solvers are applied. Just in the field of DNS explicit Runge Kutta solvers are found. For the latter the application of operator splitting method is the method of choice, since the explicit Runge Kutta solver will not experience further restrictions in time step size from the application of operator splitting methods. For Newton methods applied for the outer solver the inner solver for the set of algebraic equations will be called for each species once, during the build of the Jacobian matrix applied by the outer solver. There is further one call of the inner solver for each iteration step of the outer solver. Hence the calls of the inner solver are an order of magnitude higher than calls of the outer solver, which explains the high demand in speed on the inner solver. In addition a high accuracy of the inner solver is needed to calculate the Jacobian of the outer solver with the accuracy needed for convergence.

In the presentation we compare two solution methods with each other: 1) fixpoint iteration, and 2) a highly optimized Newton solver. The shortcoming of method 1) is that the convergence of the method is not guaranteed. In case of divergence other methods need to be applied to find a solution. One possibility is a random pertuabation of the initial guess, which is successful for most applications. The shortcoming of method 2) is the high CPU demand for the build of the inner solver. To reach a faster solution we developed a sparse matrix method, for which the sparsity is optimized by simply ordering the list of chemical species. We further use the fact, which the sparsity pattern of the Jacobian matrix is only dependent on the chemical species, which are chosen to be in steady state. This allows performing the Gaussian elimination in a preprocessing step, before starting the calculation of the physical problem. This behavior is independent on the complexity of the physical system. Beyond the huge speed up, that can be reached through chemical lumping, and species elimination, the QSSA resulted in a further 40% gain in CPU time. This advantage is of interest for many complex CFD applications. The accuracy of the resulting mechanism is very high. In the following this technique is further discussed. In the presentation we compare the different solution techniques with each other.

II. The numerical method

The numerical method for the system of DAE is a combination of two numerical methods, one for the system of ODE and one for the system of NAE, the latter representing the QSS species. The numerical method for the system of ODE is a predictor-corrector method, where the predictor is based on Gears backward differencing scheme and the corrector is a damped Newton method. The system of NAE is solved by a modified Newton method.

The Newton solver for the system of ODE (in the following called the outer solver), is iterated until convergence for each time step. An inner loop for the system of NAE (in the following called the inner solver) is iterated until convergence, for each evaluation of the chemical source terms. This solver is called several times while solving the system of DAE. The inner solver must be very accurate to allow an accurate prediction of the Jacobian, needed for the outer solver. In the past we tested fix point solver, since they are highly efficient. However, it was found that this solver

combination was not robust. In this work a Newton solver is chosen as an inner solver. The computational cost of the Newton solver scales roughly as n square if the decomposed Jacobian is reused for many iteration steps. The inner Newton solver must be optimized for computational cost, to gain a speed up by the total procedure. The modifications of the inner Newton solver that lead to a total speed up of the solver combination are explained in the following sections.

1) The optimisation of the inner solver: A modified Newton solver was chosen as an inner solver because of the high accuracy demands on the solution from the NAE. This implies that a Jacobian matrix must be calculated followed by a Gaussian elimination. The back substitution and the update of the source-vector must be called at each iteration step of the inner solver. The CPU time of the inner solver can therefore be decreased considerably by optimizing the building of the Jacobian, the Gaussian elimination, the back substitution and the update of the source vector. The sparsity pattern of the Jacobian of the inner solver is constant in time, since the chemical reaction scheme is invariant. The values of the matrix elements depend on the concentrations of the non-steady state species. The constant sparsity pattern implies that all operations made on the Jacobian will involve the same matrix elements of the Jacobian. Hence, every operation involved in the entire Gaussian elimination and the back substitution can be automatically written as source code in a preprocessing step. A highly optimized sparse matrix solver is generated. Only the non-zero elements of the Jacobian are used in the Gaussian elimination and back-substitution. The CPU-time needed for this solver is proportional to the number of operations in the automatically written source code. The sparsity pattern of the Jacobian, and thereby the number of operations in the inner solver, is strongly dependent on the order of the QSS species in the concentration vector. In order to find the optimum order of the QSS species vector, which corresponds to the minimum number of operations in the inner solver, the Metropolis algorithm is employed. The Metropolis algorithm accepts or declines the new QSS species vector, which is generated by a random shuffle of the QSS species order. The procedure is outlined as follows:

Initialize the concentration vector for the inner solver
Initialize the fictive temperature θ
Repeat until terminating condition is fulfilled

1) Change the places of two randomly chosen species in the concentration vector.
2) Create new Jacobian and perform Gaussian elimination.
3) Calculate the Sum Of Operations (SOP) in the Gaussian elimination and back substitution of the solver for the algebraic equations.
4) Calculate the SOP, based on the new and old concentration vector.
5) If the SOP is less than zero accept the new concentration vector. Otherwise accept the new concentration-vector with probability:

$$P\left(SOP,\theta\right) = exp\left(-\frac{SOP}{\theta}\right)$$

6) Decrease θ

The initial value of θ and the function used in order to decrease θ can be optimized for each case. In this work we used a predefined number of iterations as termination condition. The Metropolis algorithm allows leaving areas of local minima, and converging towards deeper local minima. With some probability it is accepting SOP ¿ 0. This algorithm does not guarantee that the global minimum is found, but in most cases a deep enough local minimum, corresponding to a very low number of operations in the inner solver, is found. Hence, the total CPU time needed for the integration of the system of DAE at high reduction levels will be reduced. It should be noted that the final value of SOP does not necessary correspond to the deepest minima. For this reason, the QSS concentration vector which corresponds to the deepest minima during the entire optimization procedure is saved instead in this work. The optimized Jabian matrix results in 317 non-zero elements after Gaussian elimination, while the original matrix resulted in 452 non-zero elements. We also calculated the worst sorting for the steady state species, which resulted in 923 non-zero elements. This is a factor 3 larger, than the optimum sorting. In Figure 1 we show the calculated speed up of the program as a function of the number of QSS species. The maximum speed up is 40% at 52 steady state species, this is about half of the total number of species. Considering a quadratic decrease of the CPU time with the number of species, a 75% speed up would have been expected. However, this would assume that the steady state species solver works at no CPU cost.

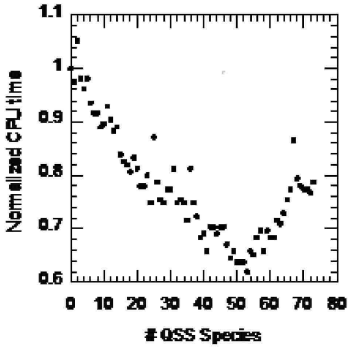

Fig. 1. Normalized CPU time as a function of the number of QSS species. The figure shows the reference case T = 900 K, ϕ = 1.0, p = 40 bar.