December 19, 2000

Adaptive Model Reduction for Chemical Kinetics

P. F. Tupper

SCCM Program,
Stanford University, Stanford CA 94305, USA.
e-mail: tupper@sccm.stanford.edu

Abstract

An adaptive model reduction algorithm is proposed for systems of ODEs from chemical kinetics. Its goal is to provide an accurate approximation to the solution of these systems faster than could be obtained through straightforward numerical integration. The algorithm approximates a system with a sequence of reduced models, each one appropriate to the dynamics of the system during a period of the trajectory. Reduced models are identical to the original system except for the deletion of some chemical reactions. This saves the cost of computing unimportant reaction coefficients. Both the reduced models and the durations for which they are used are selected adaptively in order to efficiently yield an accurate approximate solution. The performance of the algorithm is assessed through numerical experiments.

AMS subject classification: 65L05.

Key words: Ordinary differential equations, chemical kinetics, model reduction, adaptivity.

1 Introduction

The chemical kinetics of a homogeneous reacting system can be described by a system of first-order ordinary differential equations:

$$\frac{dz}{dt} = \sum_{i=1}^{M} s_i F_i(z), \tag{1.1}$$

where $z \in \mathbb{R}^N$ is a vector of chemical species concentrations and thermodynamic variables. Each $F_i : \mathbb{R}^N \to \mathbb{R}$ gives the reaction rate of one of M chemical reactions of the system. Each s_i is a vector that determines how the progress of a reaction effects the variables of the system. Equation (1.1) can be very stiff, and hence computationally expensive to solve. One of the dominant costs of the solution is the evaluation of the right-hand-side.

Zhu and Petzold [9] have developed a technique for replacing the right-hand-side of (1.1) with an expression of fewer terms that is easier to evaluate. They do this in such a way that the solution of the new system is approximately the same as that of the original system. The new system is

$$\frac{dy}{dt} = \sum_{i=1}^{M} d_i s_i F_i(y), \tag{1.2}$$

where each integer d_i is either zero or one. Such a system is referred to as a *reduced model* of the *full model* (1.1). In [9] an optimisation strategy is used to select an appropriate reduced model. They first compute the solution $y(t)$ of the full model accurately over the interval of interest $[0, t_{\text{final}}]$. They choose a positive integer $k < M$, which is to be the number of reactions in the reduced model. They then solve the integer programming problem of selecting $d_i \in \{0, 1\}$ such that

$$\int_0^{t_{\text{final}}} (y(t) - z(t))^2 dt$$

is minimised while maintaining $\sum_{i=1}^{M} d_i = k$.

This approach has the disadvantage that $y(t)$ needs to be computed over $[0, t_{\text{final}}]$ initially. However, it is hoped that the reduced model generated for a particular set of initial conditions will also be valid for other, nearby initial conditions. If this holds true, once the reduced model is generated, it can be reused with no additional cost, and further problem instances can be solved more readily.

This paper describes an alternative approach to model reduction for chemically reacting systems. Rather than discover one reduced model that is an accurate approximation to the full system for the entire time-interval $[0, t_{\text{final}}]$, we use a sequence of different reduced models, each of which is deemed to be accurate for a shorter period of the trajectory. As we proceed with the integration, we periodically assess which reactions are important to the system at that time, and integrate our system using a reduced model with these reactions. Hence, we are able to avoid the initial cost associated with solving the full system. Though there are other motivations for model reduction — namely the identification of important reactions and parameters— our goal is to produce an accurate approximation of the solution of Equation (1.1) faster than by integrating it directly.

At present there are many different approaches to model reduction. This work and [9] may be both considered *reaction subset methods*, in that our reduced models have the same state space but use only a subset of the reactions of the original system. Others utilise reduced models that are more sophisticated, and more computationally challenging to derive. One example is the work of Maas and Pope [3]. They determine an invariant manifold on which most of the dynamics takes place, and derive an approximating ODE system on this manifold. This allows the investigator to work with a system with a lower dimensional state space than that of the full system. Like [9] this requires extensive pre-computation. Another approach is that of Lam and Goussis [2]. They integrate the system through a technique known as *Computational Singular Perturbation* (CSP). Here, fast and slow modes of the dynamics are identified. The fast modes are modeled by algebraic constraints and the slow modes can then by solved by an explicit integrator. Identification is performed on-the-fly; thus the method is adaptive. The present work captures the simplicity of [9] with the benefits of the adaptivity of [2].

In Section 2 we begin by describing the adaptive model reduction algorithm in its most basic form. We apply it to a simple system and discuss the results in Section 3. In Section 4 we introduce several modifications to the basic algorithm that serve to make it more efficient. In Section 5 we apply this enhanced algorithm to a system from nuclear astrophysics. The algorithm enhances efficiency considerably in this case, while still maintaining accuracy of the solution. We determine that in order for the algorithm to be effective, the number of reactions that are important to the dynamics of the system at any point in time must be small. Otherwise, the computational overhead outweighs the savings from using a reduced model, as is the case in Section 6, where we apply it to system used to model combustion. In Section 7 we conclude with a general assessment of the algorithm and a discussion of future applications of it.

Later, we will need a way to quantify the accuracy of the algorithm. For each system that we integrate, there will be two types of errors, those due to the accuracy of the reduced models, and those due to the integrator used. In order to isolate the first source of error we choose a small value for the tolerance of our integrator, and assume that the error incurred by numerical integration is negligible. At each time $t \in [0, t_{\text{final}}]$, we denote the solution from straightforward integration by $z(t) \in \mathbb{R}^N$, and that from model reduction by $y(t) \in \mathbb{R}^N$. We compute the relative root mean squared (RRMS) difference between the two solutions in the following manner. First we interpolate values of the solution computed by straightforward integration so that they are defined at the same times, $t_k, k = 0 \ldots K$, as the solution with model reduction. Then we compute, for each variable $n = 1 \ldots N$,

$$RRMS_n = \sqrt{\frac{\sum_{k=1}^{K}(t_k - t_{k-1})(y_n^k - z_n^k)^2}{\max\left(\sum_{k=1}^{K}(t_k - t_{k-1})(z_n^k)^2, \delta\right)}}.$$

The factor of δ in the denominator allows us to weight variables that have small values less than ones that have high values. In this paper, $\delta = 10^{-20}$. We then compute the significant digits of accuracy (SDA) by

$$SDA = -\log_{10}\left(\frac{1}{N}\sum_{n=1}^{N} RRMS_n\right). \tag{1.3}$$

The software and data files necessary to repeat the numerical experiments in this paper can be found, along with instructions for their use, on the author's webpage [7].

2 The Basic Algorithm

In order to present some key ideas we will first consider a simple version of the proposed algorithm. Necessary modifications will become apparent once we analyze this case. We will denote an interval over which we integrate the system using a single reduced model a *macrostep*, in order to distinguish it from a step of the integrator which we will denote as an *integration step*. We divide the interval $[0, t_{\text{final}}]$ into K equal macrosteps of length t_{final}/K and denote the kth one by $I_k := [(k-1)t_{\text{final}}/K, kt_{\text{final}}/K]$. A small positive tolerance ϵ is selected. We compute the initial reaction rates $F_i(y_0), i = 1, \ldots M$, where $y_0 = y(0)$ is the vector of initial conditions of the system. For each reaction i, if $|F_i| < \epsilon$, we set $d_i = 0$, otherwise we set $d_i = 1$. We integrate (1.2) over the interval I_0 using this vector d. This procedure is repeated, generating the vector d from $F_i(y(kt_{\text{final}}/K))$, in order to integrate the system over $I_k, k = 1, \ldots K$. The concatenation of each of these numerical solutions is our approximate solution to the original ODE.

Below we provide a sketch of this simple version of the algorithm. It outputs a vector T of times, running from 0 to t_{final}, and an array Y of solution values. Y has the same number of rows as T has components. Each row of Y is the approximate solution vector for the corresponding time in the vector T. Both T and Y are formed by concatenating similar vectors and matrices, t and y respectively, that are output by the integrator. The integrator, denoted **odesolve** below, takes an initial condition (here, the final solution vector from the previous macrostep), a time interval to integrate over, and a vector of ones and zeros which indicates which reactions are included and excluded for that macrostep.

ALGORITHM 1 *Integration with Basic Model Reduction*

Input:
Initial condition y_0, tolerance ϵ, macrostep number K, end time t_{final}
Output:
Solution times T, solution array Y
begin
 $Y \leftarrow [y_0]$
 $T \leftarrow [0]$
 for $k = 1$ **to** K **do**
  $y_{\text{prev}} \leftarrow$ last row of Y
  for all i, if $|F_i(y_{\text{prev}})| < \epsilon$, then $d_i \leftarrow 0$, else $d_i \leftarrow 1$
  $[y, t] \leftarrow$ **odesolve**(y_{prev}, I_k, d)
  $Y \leftarrow [Y; y]$
  $T \leftarrow [T; t]$
 end
end

3 A Simple Application

We apply the algorithm to a small system used to model ozone chemistry in the upper atmosphere, see [9]. It consists of 6 reactions, 3 species (O,O_2,O_3), and one thermodynamic variable, temperature. Thus the components of the vector y are

$$y_1 = [\mathrm{O}], \quad y_2 = [\mathrm{O}_2], \quad y_3 = [\mathrm{O}_3], \quad y_4 = T,$$

where T denotes temperature. The reactions are

$$\begin{array}{lrcl} 1. & \mathrm{O_3} & \longrightarrow & \mathrm{O_2 + O}, \\ 2. & \mathrm{O_2 + O} & \longrightarrow & \mathrm{O_3}, \\ 3. & \mathrm{O_3 + O} & \longrightarrow & \mathrm{O_2 + O_2}, \\ 4. & \mathrm{O_2 + O_2} & \longrightarrow & \mathrm{O_3 + O}, \\ 5. & \mathrm{O + O} & \longrightarrow & \mathrm{O_2}, \\ 6. & \mathrm{O_2} & \longrightarrow & \mathrm{O + O}. \end{array} \tag{3.4}$$

The differential equation that describes this system is (1.1). For a full description of the s_i and F_i, see the appendix.

We choose the initial temperature to be 500K and the initial concentrations to be 0.0, 0.9, 0.1 for the three species respectively. Parameters $K = 10$ and $\epsilon = 0.01, 0.1, 1$ are used. We seek the solution on the time interval $[0, 0.1]$, where time is measured in seconds. The integrator used is ODE23TB from the MATLAB Version 5 ODE Toolbox [5]. The results are shown in Figure 3.1, plotted against the numerical solution of the full model.

The first run with $\epsilon = 0.01$ shows a fairly accurate reproduction of the solution of the full model. The time of ignition, i.e. the point at which the temperature rapidly shoots up, is delayed only slightly, and the steady state at the end appears to be accurately captured. However, the algorithm takes 130% of the time of direct integration, and hence would not be practical. Thus we increase the tolerance to see if, by omitting further reactions, we can improve efficiency.

The second run with $\epsilon = 0.1$ uses fewer reactions and is less accurate, as expected. The time at which ignition occurs is delayed by approximately 0.01 seconds, much more than in the first run. There also appear to be discontinuities in the temperature with respect to time at the ends of macrosteps 1 and 3. In Figure 3.2 we show a larger version of the central plot of Figure 3.1 so that these discrepancies are more visible. These rapid changes of temperature are caused by a fast reaction being turned on at the beginning of a new macrostep. Whenever we observe this at the beginning of a macrostep, we can be sure that the reaction was also important to the system in the previous macrostep, and should not have been omitted from it. We will discuss a strategy for coping with this difficulty in Subsection 4.2.

Perhaps the most troubling discrepancy is in the final macrosteps. In Figure 3.2 we can see that the solution of the full model attains an equilibrium, whereas the algorithm's solution exhibits oscillatory behaviour. This is purely an artifact of the model selection scheme. At the end of macrostep 6,

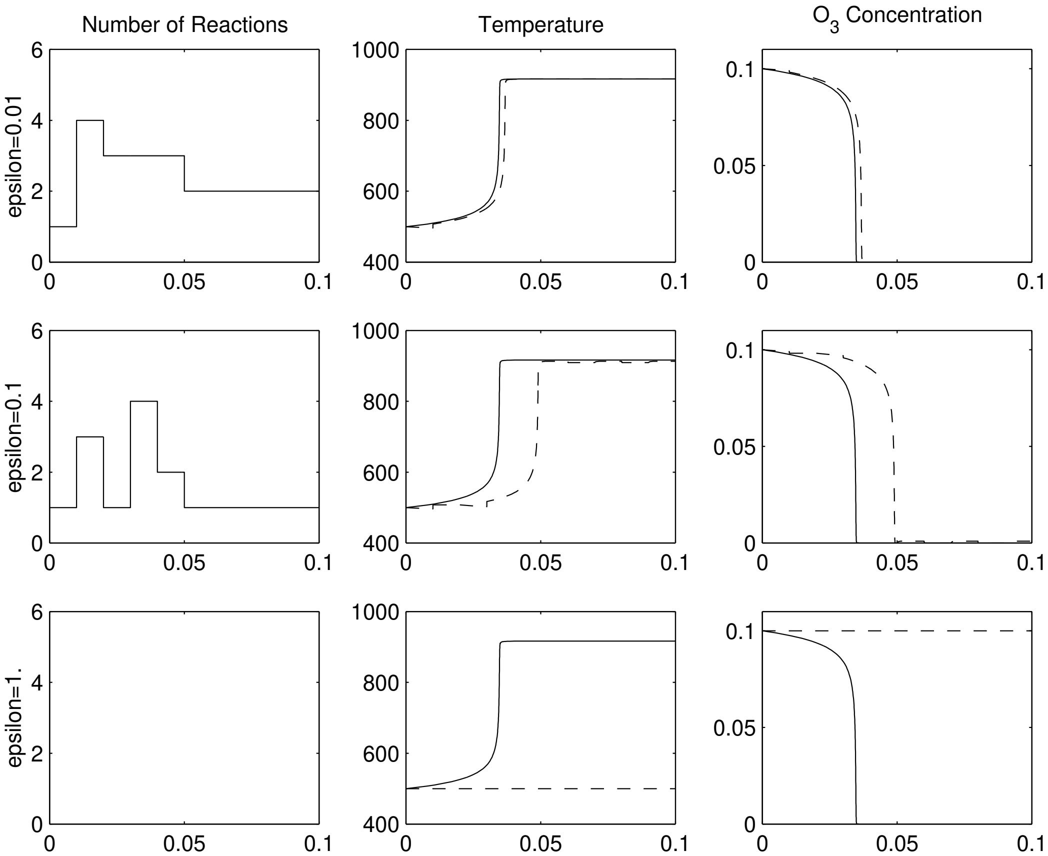

Figure 3.1: The number of reactions used in the reduced model, temperature, and O_3 concentration as a function of time. The solution with the full model is plotted with a solid line, and that with model reduction, with a dashed line. Results are shown for $\epsilon = 0.01, 0.1, 1$.

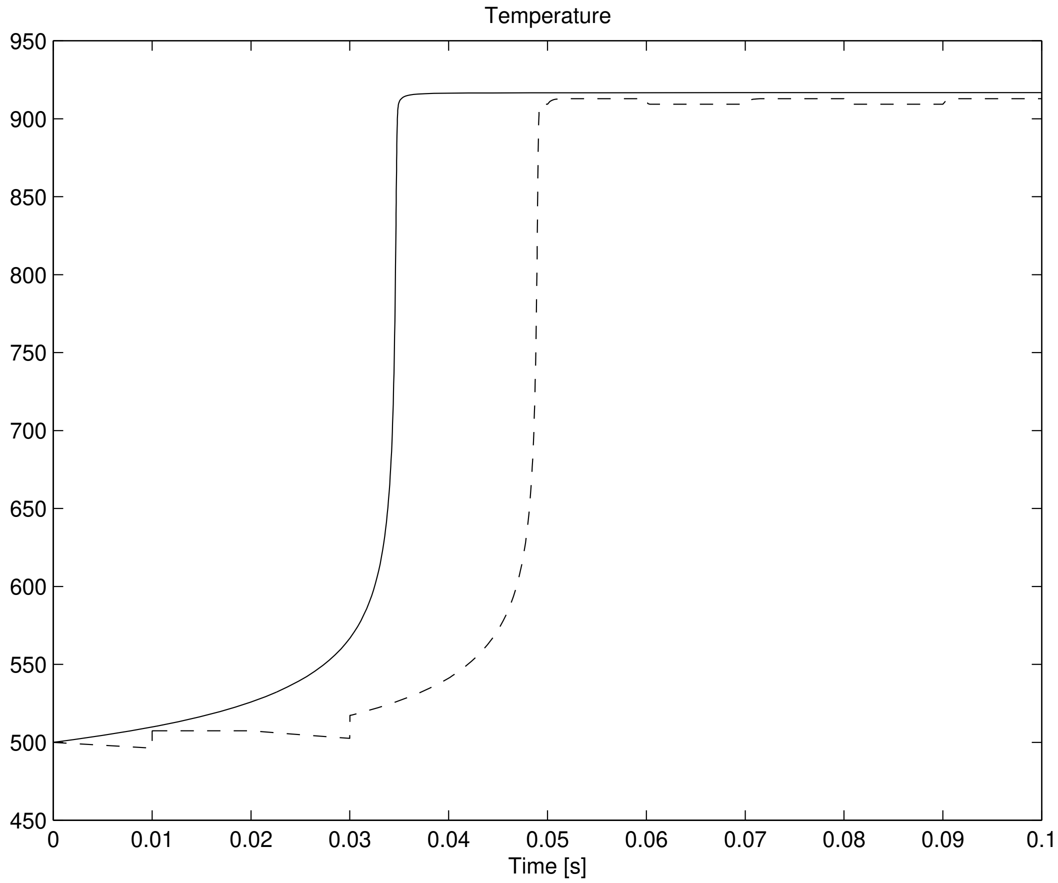

Figure 3.2: Temperature as a function of time, where the solution with the full model is plotted with a solid line, with model reduction, as a dashed line. Results shown for $\epsilon = 0.1$.

the system is close to the equilibrium of the full system and only reaction 2 is turned on. This causes O_3 to be formed. In the full system, an equilibrium would be attained through consumption of O_3 by reaction 1. However, with the algorithm, the production of O_3 is controlled only by a shortage of O. At the end of the macrostep, reaction 2 is exhausted but now reaction 1 is turned on since enough O_3 was produced. The algorithm proceeds by including reactions 1 and 3 at alternate macrosteps, never allowing an equilibrium to be attained. The algorithm integrates the system taking 175% of the time of direct integration, mostly because of the extra steps required to capture this unphysical oscillatory behaviour.

Here we observe a feature of adaptive model reduction for a stiff system that will be important in the sequel. Restarting the integration can significantly increase the number of integration steps required. When integrating a stiff system the most costly part of the trajectory to compute is the transient phase before the system settles upon the stiff manifold. Since any two reduced models may have different stiff manifolds, each new macrostep requires the computation of a transient trajectory. Thus there is an advantage to taking few macrosteps. This problem is addressed in Subsection 4.4.

During the final run with $\epsilon = 1$, since no reaction rate is larger than ϵ, d is just a vector of zeros, and remains so through all the macrosteps. Clearly this is an inadequate result, as all dynamics of the original system are lost. In this case, the algorithm took 40% of the time that direct integration took to approximate the solution. We can expect this to be an upper bound on efficiency improvement for this system, since no reaction rates are computed at all.

4 The Full Algorithm

With sufficiently small ϵ, the basic algorithm was able to approximate the solution accurately, but not efficiently. Here we describe modifications made to yield a more practical algorithm. We discuss the motivation for these modifications and describe how they are implemented. In particular, the full algorithm adaptively selects a macrostep size and reduced model, obviating many of the difficulties we saw in Section 3. In Subsection 4.6 we will give a full pseudocode specification of the algorithm. Throughout, y_{prev} and t_{prev} denote system state and time at the beginning of the current macrostep, while $y_{\text{next}}, t_{\text{next}}$ denotes these values at the end of the macrostep.

4.1 Reaction Rate Tolerances

The basic algorithm makes use of a tolerance ϵ in order to select a reduced model. The absolute magnitude of each reaction rate is compared with this tolerance.

As the system evolves there may be times when there is comparatively little activity compared to other times. Tolerances which are appropriate to times of much activity may be inappropriate during times of little activity, as we may omit nearly all the reactions from the model, and miss the behaviour of the system entirely. Hence we use a combination of absolute and relative tolerances, as is the case in many ODE solvers. We choose positive parameters ϵ_{abs}, ϵ_{rel}. To characterise the amount of activity of the system, we compute the sum of the magnitude of all the reaction rates. Then, for a reaction j to be considered negligible, we require that

$$|F_j| \leq \min(\epsilon_{abs}, \epsilon_{rel} \sum_{i=1}^{M} |F_i|). \tag{4.5}$$

This proves effective in providing fairly uniform error control throughout $[0, t_{\text{final}}]$.

In the main algorithm, the subroutine **select** sets the vector d. When this procedure is given a vector F of reaction rates and tolerances ϵ_{abs}, ϵ_{rel}, it returns a vector d of ones and zeros. Specifically, the jth component of d is set to 0 if (4.5) holds, and 1 otherwise. In the algorithm this is expressed as

$$d \leftarrow \mathbf{select}(F(y_{\text{prev}}), \epsilon_{abs}, \epsilon_{rel}).$$

It is called at the beginning of every macrostep.

4.2 Error Control

Ideally, every macrostep taken would yield a valid approximation to the solution of the full system in that interval. However, as seen in the Section 3, this is not always the case. We must find some way to determine whether a macrostep is valid and, if not, how to mitigate the situation. We begin by assuming that when a reaction rate is small in magnitude at a particular point of the trajectory, it can be safely deleted from the model at that point. Given that we delete reactions at the beginning of a step only if they have a small rate, a step becomes invalid when a reaction rate that was small at the beginning of the step becomes large during the course of the step. One simple, though fallible, way to detect this is to examine the rates of the deleted reactions at the end of the step, and see if they are still small. If they are not, we can infer that the step was inaccurate, and that we cannot accept the values computed for this macrostep in our solution. In practice, we choose a parameter $\phi > 1$ and compute

$$\mathit{mustinclude} \leftarrow \mathbf{select}(F(y_{\text{next}}), \phi\epsilon_{abs}, \phi\epsilon_{rel}).$$

If for each i, $\mathit{mustinclude}_i = 1$ only when $d_i = 1$, we accept the macrostep. The factor ϕ prevents unnecessarily frequent macrostep rejections. We may have the situation that we have included a reaction that only barely satisfies the criteria. With $\phi = 1$, if it grows even a small amount, it will trigger a step rejection. Setting $\phi > 1$ ensures that the model will be considered valid for some reasonable positive period of time. For all experiments in this paper ϕ is set to 10.

If the macrostep is rejected, we must choose some action to correct for the invalid step. One possibility, which we adopt, is to repeat the macrostep, including the reactions that had become important. If the step is now valid, we proceed to the next time-interval. If not, we iterate this procedure, including problematic reactions, until we have a reduced model that remains valid for the entire macrostep.

4.3 Steplength Selection

In the basic algorithm, the macrostep length was constant throughout $[0, t_{\text{final}}]$. This is not ideal since at times we may wish to use a reduced model that would not be valid for this entire period. Conversely, we may have a reduced model that will be effective for a longer period of time, and it is wasteful to generate a new one sooner than necessary. We would like to predict for how long a reduced model will be valid, and set the macrostep to be of that length. We define l_{macro} to be the length of the macrostep that we will use.

To determine a good l_{macro}, we need to be able to predict how the omitted reaction rates will behave in the subsequent time interval. In particular, we need to know when they will be large enough to trigger a macrostep rejection. We compute an estimate of the rate of change of reaction j using finite differences:

$$G_j \approx dF_j(y_{\text{prev}})/dt,$$

for $j = 1, \ldots, M$. We predict that the reaction most likely to cause a macrostep rejection is the one with the largest G_j. Accordingly, we define

$$G_{max} = \max_j G_j$$

and j_{max} to be the index of the maximum G_j. We then assume that the reaction corresponding to G_{max} will grow linearly over the course of the macrostep. Accordingly we estimate the value of $F_{j_{max}}$ at time $t_{\text{prev}} + l_{macro}$ to be

$$F_{j_{max}}(y_{\text{prev}}) + G_{max} l_{macro}.$$

To prevent our estimated reaction rate from causing a macrostep rejection we then require that this quantity be less than

$$\min(\phi\epsilon_{abs}, \phi\epsilon_{rel} \sum_{i=1}^{M} |F_i|),$$

where ϕ is specified as in the previous subsection. Since we know that $F_{j_{max}}(y_{\text{prev}}) \leq \min(\epsilon_{abs}, \epsilon_{rel} \sum_{i=1}^{M} |F_i|)$, we set

$$l_{macro} = \frac{\sigma}{G_{max}} \min((\phi - 1)\epsilon_{abs}, (\phi - 1)\epsilon_{rel} \sum_{i=1}^{M} |F_i|),$$

where σ is a safety factor: $0 < \sigma < 1$. In the code, this is accomplished by the command

$$l_{macro} \leftarrow \textbf{getsteplength}(G_{max}, \epsilon_{abs}, \epsilon_{rel}, \phi, \sum_{i=1}^{M} |F_i|).$$

4.4 Model Selection

If we select our model as in the basic algorithm, we may exclude a reaction whose rate is below tolerance, but which is predicted to become large in a very short time. The procedure of the previous section will accordingly select a very short macrostep. However, this may be quite inefficient. It forces us to restart a macrostep again almost immediately, incurring a non-negligible overhead cost, as we saw in Section 3. It may be much more efficient to include the reaction to begin with so that the algorithm may take a larger step, though with an additional reaction. We propose a more sensible way of selecting a reduced model through the subroutine **getmodel**.

Our goal is to take a macrostep that is both long and efficient. We wish to minimise $\mathcal{C}/l_{macro}$, where l_{macro} is macrostep length and $\mathcal{C}$ is the computational expense of the macrostep. We assume that

$$\mathcal{C} = R + kn_{reac}l_{macro}$$

where R is the cost of restarting a macrostep, n_{reac} is the number of included reactions, and k is some constant parameter. Rewriting, we get

$$\frac{\mathcal{C}}{l_{macro}} = \frac{R}{l_{macro}} + kn_{reac}.$$

Extracting a constant factor, we want to minimise $w/l_{macro} + n_{reac}$ where w is a constant parameter that can be estimated or tuned. The simple optimisation algorithm used in **getmodel** below determines which reactions to add in order to minimise this objective function. However, once the new reactions are added, they change the values of dF/dt. So new reactions, still not included in the model, may now be predicted to have rapidly increasing rates. We again need to check if we should add more reactions. This procedure is iterated until we arrive at a reduced model for which the algorithm adds no additional reactions.

PROCEDURE **getmodel** *Model Selection*

Input:
state of system for which we need reduced model y,
tolerances ϵ_{abs}, ϵ_{rel},
error factor ϕ, weight w
Output:
reduced model vector d, macrostep length l_{macro}
begin
  evaluate F, dF/dt at y
  $d \leftarrow$ **select**$(F(y),\epsilon_{abs},\epsilon_{rel})$
  finished $\leftarrow$ false
  while *finished* $=$ false **do**
    if $d_i = 0$ **then** $G_i \leftarrow dF_i/dt$ **else** $G_i \leftarrow 0$

```
        sort G in decreasing order
        J ← largest j for which G_j > 0
        for k = 1 to J do
            l_macro ← getsteplength(G_k, ε_abs, ε_rel, φ, Σ_{i=1}^{M} |F_i|)
            optim_k ← w/l_macro + k
        end
        k_min ← k for which optim_k is minimum
        if k_min = 1
            then finished ← true
                 G_max ← G_{k_min}
            else d_i ← 1 for all i such that dF_i/dt > G_max
        endif
    end
    l_macro ← getsteplength(G_max, ε_abs, ε_rel, φ, Σ_{i=1}^{M} |F_i|)
end
```

4.5 Integration Step Limitations

One problem that arose frequently in experiments was that the algorithm would choose to take a macrostep of such a length that many integration steps were required to finish it. This was fine when the macrostep turned out to be valid, but when it was invalid, all the time spent on integration was wasted. To address this, a limit was placed on the number of integration steps that could be taken in a macrostep.

With the new restriction, macrostep length and reduced model are chosen as usual and integration is begun. Integration is terminated when the duration of the macrostep is reached, *or* the number of integration steps exceeds a fixed parameter *maxsteps*, whichever comes first. We indicate the use of this by adding the argument *maxsteps* to **odesolve**. This restriction proves to be quite crucial for maintaining efficiency.

4.6 Pseudocode

Below is the pseudocode specification of the full algorithm. It uses the procedures described in the previous subsections. In particular, it uses the routine **getmodel** described in Subsection 4.4 to select a reduced model d, and a macrostep length l_{macro}. It integrates with this model until either a time l_{macro} is attained, or a *maxsteps* number of integration steps is exceeded (Subsection 4.5). It then examines the solution at the end of the macrostep to assess whether it is sufficiently accurate. If it is, the algorithm accepts the macrostep and proceeds with a new one. Otherwise it includes any problem reactions and repeats the step (Subsection 4.2).

ALGORITHM 2 *Integration with Adaptive Model Reduction*

Input:
Initial y_0, tolerances ϵ_{abs}, ϵ_{rel},
error factor ϕ, weight w, end time t_{final}
Output:
Approximate solution vector Y, with solution times T
begin
    $Y \leftarrow y_0$
    $T \leftarrow 0$
    repeat
        $y_{\text{prev}} \leftarrow$ last value of Y
        $t_{\text{prev}} \leftarrow$ last value of T
        $[d, l_{macro}] \leftarrow$ **getmodel**(y_{prev}, ϵ_{abs}, ϵ_{rel},ϕ,w)
        $mustinclude \leftarrow 0$
        repeat
            $d \leftarrow \max(d, mustinclude)$
            $[y, t] \leftarrow$ **odesolve**(y_{prev}, $[t_{\text{prev}}, t_{\text{prev}} + l_{macro}]$, d, $maxsteps$)
            $y_{\text{next}} \leftarrow$ last value of y
            $t_{\text{next}} \leftarrow$ last value of t
            $mustinclude \leftarrow$ **select**($F(y_{\text{next}})$, $\phi\epsilon_{abs}$, $\phi\epsilon_{rel}$)
            if for any i, $mustinclude_i = 1$ and $d_i = 0$
                then *stepdone* $\leftarrow$ false
                else *stepdone* $\leftarrow$ true
            endif
        until *stepdone* = true
        $Y \leftarrow [Y; y]$
        $T \leftarrow [T; t]$
        $t_{\text{next}} \leftarrow$ final value of T
    until $t_{\text{next}} = t_{\text{final}}$
end

Run	Event	Initial Temperature 10^9K	Initial Density g/cm^3
1	Nova	0.2	5×10^3
2	Supernova	0.25	50
3	X-ray Burst	1.5	1.5×10^6
4	Thorne-Zytkow	2.0	3×10^3

Table 1: Four sets of initial conditions for the astrophysics system.

5 An Astrophysics Example

The first system to which we apply the full algorithm is from the field of nuclear astrophysics. Here researchers model the astrophysical events in which heavier isotopes are synthesised from lighter isotopes through nuclear reactions. Though such events do not occur in a spatially homogeneous fashion, researchers use ODE models to get an idea of what kind of behaviour may occur.

We consider one such system here, due to [6]. It consists of 394 species and 3933 reactions. See [7] for files containing reaction rate data. Rather than model the thermodynamics, we take temperature and pressure to decay exponentially, i.e. $T = T_0 e^{-0.005t}$, $\rho = \rho_0 e^{-0.005t}$ (see [4]). We start the system at initial conditions corresponding to concentrations in the sun, a typical choice, and solve the equations for a selection of different initial temperatures and pressures taken from [8] and shown in Table 5. Each set of initial conditions corresponds to one of four different astrophysical events. The solution is found on a time interval of 100 seconds.

None of MATLAB's integrators were able to integrate the system, due to its large instability. Hence, a simple adaptive Backward Euler method was written, and all experiments in this section were performed with it. Hence, we should be aware that results for this system may change given an optimised integrator.

The algorithm was applied with $\epsilon_{abs} = \epsilon_{rel} = 10^{-6}$. Table 2 shows how it performs in efficiency and accuracy. In each case the algorithm was able to improve significantly the speed with which the solution was computed. A more than 9-fold increase in efficiency was obtained in the second run, and the algorithm always reduced the running time by a factor of more than 2.

Event	Time [s] no m.r.	Time [s] m.r.	Ratio	Macrosteps	Rejections	SDA
1	2.30e3	265	0.12	27	1	2.03
2	1.96e3	206	0.11	22	0	2.23
3	7.17e3	3.03e3	0.42	130	25	2.09
4	4.76e3	1.81e3	0.38	85	37	1.61

Table 2: The performance of the model reduction (m.r.) algorithm on the astrophysics system, for four different sets of initial conditions. Time indicates the duration of the computation in seconds. Ratio indicates the duration with model reduction divided by the duration without model reduction.

Accuracy also appears to be quite good; an SDA (see (1.3)) of more than 1.5 was attained in each case. Accuracy can be assessed further by examining plots of the solution. Figure 5.3 shows plots of reduced model size and selected isotope concentrations versus time for the four events. The difference between the plots for the straightforward integration and integration with model reduction is not visible.

In order to examine the effect of varying tolerances on efficiency and accuracy, we set $\epsilon_{abs} = \epsilon_{rel} = \epsilon$, and vary ϵ between 10^{-7} and 10^{-1} as we apply the algorithm to the second set of initial conditions in Table 2. Results are shown in Figure 5.4. As expected, lowering error tolerances yields a solution that is more accurate and more expensive to compute.

6 A Combustion Example

We now apply the full algorithm to a smaller system and see if it is still efficient. We consider a set of reactions from the field of gas combustion: GRIMECH [1]. It consists of 32 species and 177 reactions. Reaction rates are given by expressions similar to those for the system in Section 3. For the purpose of simplicity, the reaction coefficients are altered slightly from those of this source. However, this should not qualitatively effect the behaviour of the system or the performance of the algorithm. A full description of the system can be found at [7]. We use ODE23TB as our integrator.

Figure 6.5 shows the results of the run. The qualitative behaviour of the system is captured fairly well, so that there is only a slight delay in the time

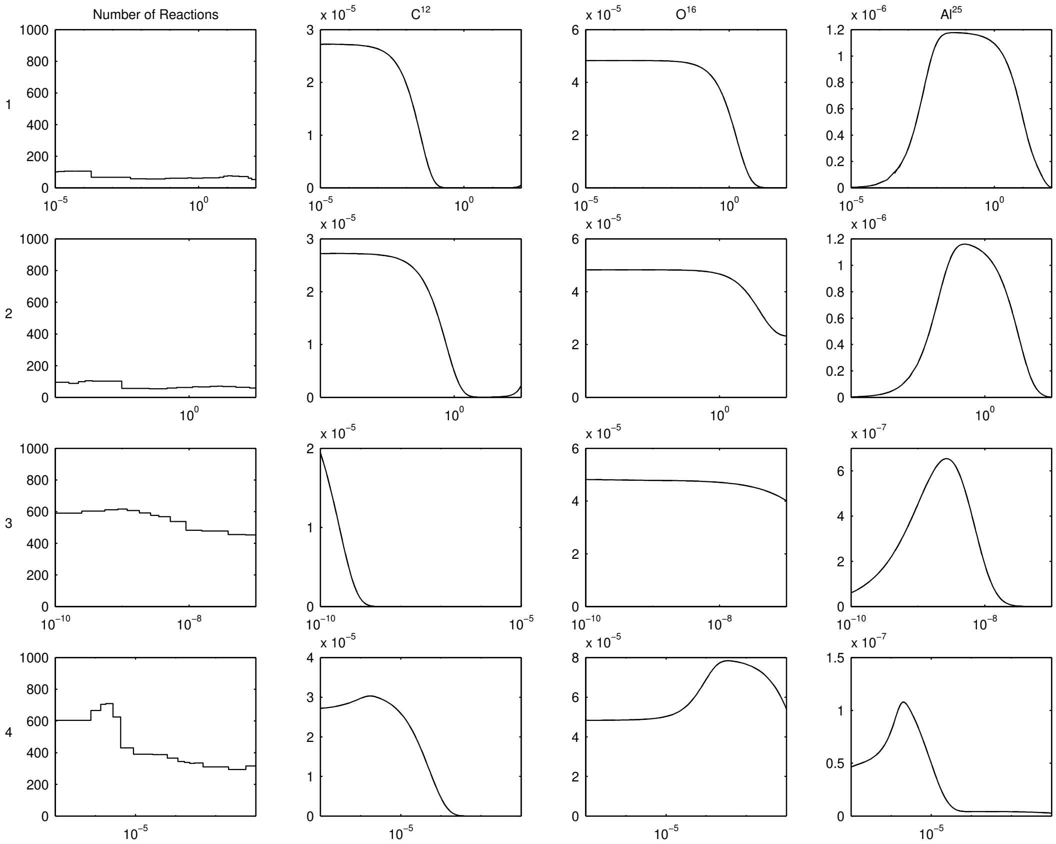

Figure 5.3: Number of reactions used in reduced model, C^{12}, O^{16}, and Al^{25} concentration as a function of time, for four astrophysical events. The difference between the solution using model reduction and without is not visible.

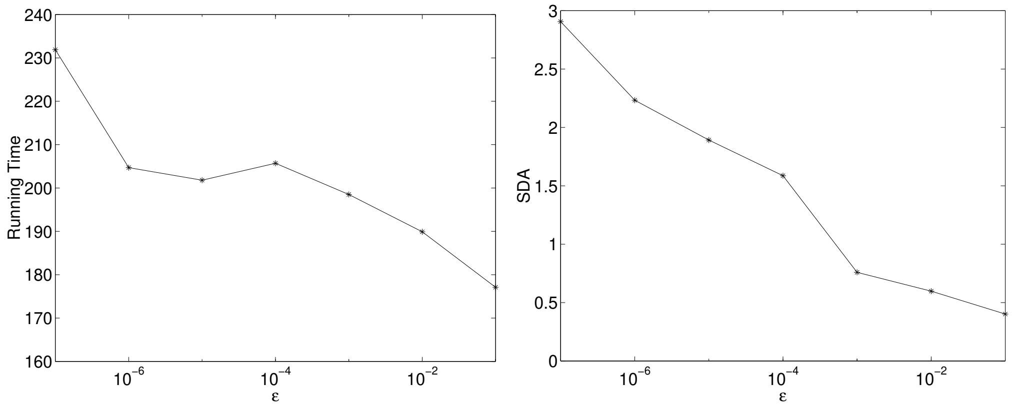

Figure 5.4: Running time in seconds and significant digits of accuracy (SDA) versus tolerance ϵ for the second set of initial conditions. The running time without model reduction was 1900 seconds.

of ignition. The algorithm used 30 macrosteps to integrate the system and there were 4 macrostep rejections. The algorithm took 120% of the time that direct integration took. A significant increase in efficiency only came at the cost of raising tolerances too high, worsening the accuracy unacceptably. It appears that our algorithm is not practical for this system.

One of the reasons that the algorithm was not as successful as one might hope is that the fraction of the reactions that were used at any point of the trajectory was large. At worst the algorithm excluded only two of the reactions. Typically, the reduced model used more than a third of the reactions. So a right-hand-side evaluation with the reduced model was often almost as expensive as with the full model. Given the overhead cost of model reduction and of periodically restarting the integration, the algorithm was slower than straightforward integration. Thus the algorithm has a better chance of succeeding if we consider a larger system as in the previous section, where a smaller fraction of the total reactions is important at any given time.

7 Conclusion

We have shown that adaptive model reduction of the type examined can be effective in improving the speed at which chemical kinetics systems are integrated. A system was found to be appropriate for the algorithm if few of the system's reactions were important to the dynamics at any point in time.

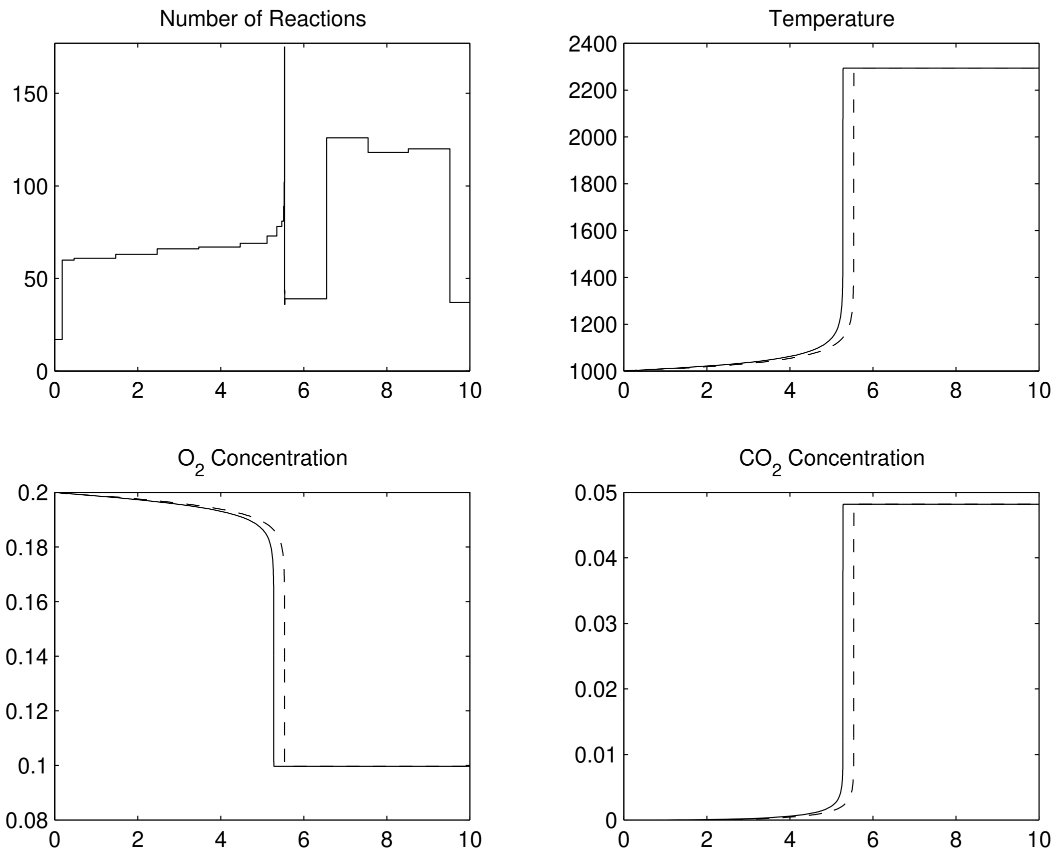

Figure 6.5: The number of reactions used in the reduced model, temperature, oxygen and carbon dioxide concentration as a function of time. The solution with the full model is plotted with a solid line, and that using model reduction, with a dashed line.

In this case the additional cost of computing reduced models and assessing their accuracy was more than compensated by the computational savings of model reduction.

There are two directions of further research which may prove of interest: (1) to examine the effectiveness of the algorithm in a broader set of chemical kinetics situations, (2) to apply a similar algorithm to other complex systems where piece-wise approximations by simpler systems may prove valuable. One example of the latter case is that of a nonlinear sytem which we approximate adaptively by a series of linear systems. We may be able to solve the linear systems quite accurately, in comparison to the non-linear system, but we need to know how often to update our approximation and when to reject solutions given by it. Many of the issues considered in this paper would be germane to such a project.

Acknowledgements. The author would like to thank Linda Petzold and Andrew Stuart for many helpful discussions. During the preparation of this paper, the author was supported by the Thomas V. Jones Stanford Graduate Fellowship.

Appendix

Here we provide the full description of the simple system in Section 3. It consists of $M = 6$ reactions and $N = 4$ variables. The variables are

$$z_1 = [\mathrm{O}], \quad z_2 = [\mathrm{O}_2], \quad z_3 = [\mathrm{O}_3], \quad z_4 = T,$$

where T denotes temperature and [X] denotes the concentration of species X. The differential equation for the system is

$$\frac{dz}{dt} = \sum_{i=1}^{M} s_i(z) F_i(z),$$

where $s_i : \mathbb{R}^N \to \mathbb{R}^N$ and $F_i : \mathbb{R}^N \to \mathbb{R}$. For each i the vector function s_i is given by the ith column of the following matrix:

$$\begin{bmatrix} 1 & -1 & -1 & 1 & -2 & 2 \\ 1 & -1 & 2 & -2 & 1 & -1 \\ -1 & 1 & -1 & 1 & 0 & 0 \\ -28.55/C_p & 28.55/C_p & 90.55/C_p & -90.55/C_p & 119.1/C_p & -119.1/C_p \end{bmatrix}$$

where the only variable is $C_p = [5.242z_1 + 7.148z_2 + 9.364z_3]/1000$.

To define the F_i we need to specify some other variables first. We define

$$n_0 = z_1 + z_2 + z_3$$

and

$$n_i = \begin{cases} 0.44z_1 + 0.44z_2 + z_3, & i = 1, 2, 5, 6 \\ 1, & i = 3, 4. \end{cases}$$

Then we let

$$V = n_0 RT/P$$

where $R = 1.986$ and $P = 0.0242$. For each reaction, F_i is given by

$$F_i(z) = A_i T^{b_i} \exp(-E_i/RT) P_i(z) n_i V^{c_i}$$

where all remaining parameters are defined in the table below.

Reaction	A_i	b_i	E_i	$P_i(z)$	c_i
1	4.31×10^{14}	0	92790.0	z_3	-1
2	1.20×10^{13}	0	-8110.0	$z_1 z_2$	-2
3	1.14×10^{13}	0	19120.0	$z_1 z_3$	-1
4	1.19×10^{13}	0	420690.0	z_2^2	-1
5	1.38×10^{18}	-1	1420.0	z_1^2	-2
6	2.75×10^{19}	-1	496610.0	z_2	-1

Table 3: Reaction coefficients for the system in Section 3.

References

[1] Gas Research Institute, *http://www.me.berkeley.edu/gri_mech/.*

[2] S. H. Lam, *Using CSP to understand complex chemical kinetics* , Combust. Sci. and Tech. 89, (1993), pp. 375–404.

[3] U. Maas, S. B. Pope, *Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space*, Combustion and Flame, 88, (1992), pp. 239–264.

[4] N. Prantzos, M. Arnold, J.-P. Arcoragi, *Neutron Capture Nucleosynthesis During Core Helium Burning in Massive Stars,* Astrophys. J., 315, (1987), pp. 209–228.

[5] L. F. Shampine, M. W. Reichelt, *The MATLAB ODE suite,* SIAM J. Sci. Comput., 18, (1997), no. 1, pp. 1–22.

[6] F.-K. Thielemann, *http://quasar.unibas.ch/personal/thiel.html/.*

[7] P. F. Tupper, *http://www-sccm.stanford.edu/Students/tupper/.*

[8] L. Van Wormer, J. Gorres, C. Iliadis, and M. Wiescher, *Reaction Rates and Reaction Sequences in the rp-Process,* Astrophys. J., 432, (1994), pp. 326–350.

[9] W. Zhu, L. Petzold. *Model Reduction for Chemical Kinetics: An Optimization Approach*, AIChE Journal (April 1999), pp. 869–886.