Journal of Computational Physics 229 (2010) 1724–1738

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

On projected Newton–Krylov solvers for instationary laminar reacting gas flows

S. van Veldhuizen [a], C. Vuik [a,*], C.R. Kleijn [b]

[a] Delft University of Technology, Delft Institute of Applied Mathematics and J.M. Burgers Centre for Fluid Mechanics, Mekelweg 4, 2628 CD Delft, The Netherlands
[b] Delft University of Technology, Department of Multi Scale Physics and J.M. Burgers Centre for Fluid Mechanics, Prins Bernardlaan 6, 2628 BW Delft, The Netherlands

ARTICLE INFO

Article history:
Received 6 April 2009
Received in revised form 15 September 2009
Accepted 3 November 2009
Available online 11 November 2009

Keywords:
Inexact projected Newton methods
Positivity
Stiff systems
Preconditioned Krylov methods
Laminar reacting flows
Chemical vapor deposition

ABSTRACT

Numerical aspects of computational modeling of chemical vapor deposition are discussed. Large sparse strongly nonlinear algebraic systems are to be solved per time step. For this, inexact Newton methods and preconditioned Krylov subspace methods are suitable. To ensure positivity of concentrations, we propose a novel approach, namely a projected inexact Newton method. Unlike the commonly used method of clipping, this conserves mass. Efficiency of several preconditionings is compared. Our numerical tests culminate in an unusually large computation, namely a three-dimensional case with 17 species and 26 reactions.

1. Introduction

Multi-dimensional simulation of laminar reacting flow, such as for chemical vapor deposition (CVD) [8] and flows with combustion [19] requires the simultaneous solution of many strongly coupled advection–diffusion–reaction equations. Here we discuss only CVD, but our methods and considerations apply to laminar combustion as well.

Assuming continuum flow, the mathematical model consists of the Navier–Stokes equations accompanied by the energy equation and a usually large number (typically 10–100) of advection–diffusion–reaction equations that model the interaction of the reactive species, of the following form (for species with label i):

$$\frac{\partial(\rho\omega_i)}{\partial t} = -\nabla\cdot(\rho\mathbf{v}\omega_i) + \nabla\cdot(\rho\mathbb{D}_i\nabla\omega_i) + \nabla\cdot(\mathbb{D}_i^T\nabla(\ln T)) + m_i\sum_{k=1}^{K}\nu_{ik}R_k^g, \tag{1}$$

where ω_i is the ith species mass fraction, ρ the density of the gas mixture, $\mathbf{v}$ the mass averaged velocity, $\mathbb{D}_i$ the effective ordinary diffusion coefficient for species i [12], $\mathbb{D}_i^T$ the thermal diffusion coefficient for species i, T the temperature, m_i molar mass of species i, K the number of gas phase reactions, ν_{ik} the stoichiometric coefficient of species i in the kth reaction, and R_k^g the reaction rate of gas phase reaction k. Further details on R_k^g are presented in Section 4.

Often, the reactive gases in CVD are highly diluted in an inert carrier gas, such that the dilute mixture approach can be used, see [10]. Under those conditions the time evolution of the gas species can be computed from solving only the system of species equations (1) in time, whereas for the Navier–Stokes and energy equations the steady state solution suffice. Further

* Corresponding author.
E-mail addresses: c.vuik@tudelft.nl (C. Vuik), c.r.kleijn@tudelft.nl (C.R. Kleijn).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.11.004

details are given in Section 4 of this paper and in [18]. Under these circumstances most of the computing work is consumed by solving the system of Eq. (1). Due to large disparity in timescales for advection, diffusion and reaction, equations (1) are very stiff. In that case, most commercial CFD codes fail to solve Eq. (1) with acceptable efficiency. No successful computations of multi-dimensional, multi-species and multi-reaction CVD flow using commercial CFD codes have been reported in literature. Moreover, virtually all published multi-species, multi-reaction CVD simulations have been limited to two spatial dimensions and steady state. In our prior work [18] and this paper we aim at two-dimensional and three-dimensional instationary CVD simulations. In [18] we have shown that careful numerical analysis results in successful determination of time accurate solutions. However, in [18] we did not discuss efficient linear solvers. This is one of the topics of this paper.

For stability, temporal discretization of (1) has to be implicit, and for positivity of species concentrations it has to be first order [18]. We therefore use the first order Euler backward method. Hence, in each time step a large sparse strongly nonlinear algebraic system has to be solved. Usually, an initial guess sufficiently close to the solution for Newton's method to converge is not available. Application of direct solvers inside Newton's iteration is ruled out by the size of the systems involved. The linear systems generated by the Newton iteration will be solved iteratively with limited precision, resulting in what is called an inexact Newton method [4]. Because of inexactness, positivity of concentration is not guaranteed, leading to an unacceptable risk of breakdown of the computation. To prevent this we propose a projected Newton method, combined with backtracking algorithms, such that the returned solutions are guaranteed to be positive. This is the second topic of this paper, discussed in Section 2. In Section 3 we explore various preconditioning techniques and the way to use them in the context of chemically reacting flows. We conclude with test computations, including a three-dimensional case with 17 gas-phase species, 26 gas phase reactions and 14 surface reactions.

2. Nonlinear solver

2.1. Inexact Newton methods

Because of strong stiffness the species equations (1) are discretized implicitly in time. As a consequence a large strongly nonlinear algebraic system has to be solved in each time step. In a previous study [18] we found Newton's method to be more efficient than modified Newton methods. We denote a nonlinear algebraic system as $F(x) = 0$. Using Newton's methods, we have to solve

$$F'(x_k)s_k = -F(x_k). \tag{2}$$

For the sake of efficiency we solve (2) only approximately when far from convergence. This results in what is known as an Inexact Newton method, see for instance [4]. Independently of which approximation method is used, the convergence criterion to solve Eq. (2) approximately is

$$\|F(x_k) + F'(x_k)s_k\| \leqslant \eta_k \|F(x_k)\|, \tag{3}$$

for a certain 'forcing term' $\eta_k \in [0, 1)$. The local convergence behavior, which depends on the choice of η_k, is discussed in [4].

When the initial guess x_0 is not close enough to the solution x_* the (inexact) Newton method may diverge. We therefore augment Newton's algorithm and inequality (3) with an extra condition giving global convergence. We use backtracking methods as in [6]. The resulting method is presented as Algorithm 1.

In the while-loop, each λ is chosen to minimize the quadratic polynomial model of $\phi(\lambda) = \|F(x_k + \lambda s_k)\|_2^2$, subject to the safeguard that $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. How to build this quadratic polynomial model is discussed in [9]. In the numerical experiments of Section 5 we used the safeguard $\lambda \in [1/10, 1/2]$. Regarding this safeguard, it is important to choose the lower bound not too small, since it will lead to poor Newton updates, and, ultimately divergence. For any choice $\lambda_{\max} \in [1/2, 1)$ in this safeguard, the convergence speed will not be influenced.

How to choose the forcing term in inequality (3) is the central theme in [7], where it is found that for a wide range of PDE problems the following two forcing terms give optimal Inexact Newton convergence:

- Choose the initial forcing term equal to $\eta_0 = 1/2$,
- Choice 1: Given η_0, then choose η_k for $k \geqslant 1$ as

$$\eta_k = \frac{\big|\|F(x_k)\| - \|F(x_{k-1}) - F'(x_{k-1})s_{k-1}\|\big|}{\|F(x_{k-1})\|}, \tag{4}$$

- Choice 2: Given η_0, then choose for $k \geqslant 1$

$$\eta_k = \gamma \frac{\|F(x_k)\|^2}{\|F(x_{k-1})\|^2}, \quad \text{with} \quad \gamma \in [0, 1). \tag{5}$$

If the initial iterate x_0 is sufficiently near the solution x_*, then the sequence $\{x_k\}$ produced by Algorithm 1 and forcing terms (4) and (5), converges super-linearly towards a solution. Details on the convergence speeds are found in [7]. Further, a safeguard is implemented to avoid solving the Newton equation with more accuracy than necessary. Details on this safeguard are found in [7,17].

Algorithm 1. Globalized inexact Newton

Let x_0, $\eta_{\max} \in [0,1)$, $t \in (0,1)$ and $0 < \lambda_{\min} < \lambda_{\max} < 1$ be given.
for $k = 1, 2, \ldots$ until 'convergence' **do**
  Choose, using (4) and (5), $\eta_k \in [0, \eta_{\max}]$ and find s_k that satisfies
    $\|F(x_k) + F'(x_k)s_k\| \leqslant \eta_k \|F(x_k)\|$.
  while $\|F(x_k + s_k)\| > (1 - t(1 - \eta_k))\|F(x_k)\|$ **do**
    Choose $\lambda \in [\lambda_{\min}, \lambda_{\max}]$
    Set $s_k \leftarrow \lambda s_k$ and $\eta_k \leftarrow 1 - \lambda(1 - \eta_k)$
  end while
  Set $x_{k+1} = x_k + s_k$.
end for

2.2. Globalized projected Newton methods

Maintaining non-negativity of species concentrations while solving the species Eq. (1) is crucial to avoid breakdown of the computation. In this study spatial and temporal discretization ensure positivity for all mesh sizes and time step sizes. But solving the resulting implicit relation inexactly may result in violation of positivity of species concentrations as has also been confirmed by numerical experiments, see Section 5.

We therefore adapt the globalized inexact Newton method to preserve positivity. A sequence $\{x_n\}$ in the positive orthant is generated that converges to a solution x_* of the nonlinear problem $F(x) = 0$, by a so-called projected Newton method. Projected Newton methods originate from nonlinear optimization problems with constraints, and were first proposed by Bertsekas [2]. To the authors' knowledge, these kind of method have not been applied to PDEs. The idea is to replace negative elements in a new iterand $x_k + s_k$ by zero and to check whether this projected iterand satisfies the following sufficient decrease condition:

$$\|F(\mathcal{P}(x_k + s_k))\| \leqslant (1 - t(1 - \eta_k))\|F(x_k)\|, \tag{6}$$

where $\mathcal{P}$ is the projection on the positive orthant and t a small parameter. The ith entry of $\mathcal{P}(x)$ is defined by

$$\mathcal{P}_i(x) = \begin{cases} x_i & \text{if } x_i \geqslant 0 \\ 0 & \text{if } x_i < 0 \end{cases} \tag{7}$$

When condition (6) is not satisfied, the search direction s_k and η_k are adjusted by means of a backtracking procedure as described in Algorithm 1. The resulting algorithm, called globalized inexact projected Newton, is presented below as Algorithm 2.

Algorithm 2. Globalized inexact projected Newton

Let x_0, $\eta_{\max} \in [0,1)$, $t \in (0,1)$ and $0 < \lambda_{\min} < \lambda_{\max} < 1$ be given.
for $k = 1, 2, \ldots$ until 'convergence' **do**
  Find some $\eta_k \in [0, \eta_{\max}]$ and s_k that satisfy
    $\|F(x_k) + F'(x_k)s_k\| \leqslant \eta_k \|F(x_k)\|$.
  while $\|F(\mathcal{P}(x_k + s_k))\| > (1 - t(1 - \eta_k))\|F(x_k)\|$ **do**
    Choose $\lambda \in [\lambda_{\min}, \lambda_{\max}]$
    Set $s_k \leftarrow \lambda s_k$ and $\eta_k \leftarrow 1 - \lambda(1 - \eta_k)$
    If such λ cannot be found, terminate with failure.
  end while
  Set $x_{k+1} = \mathcal{P}(x_k + s_k)$.
end for

We cannot prove that inequality (6) can always be satisfied, except in the case that $x_k + s_k$ is in the positive orthant. Then, the standard convergence theory of inexact Newton methods with backtracking applies [6]. Neither can we derive conditions for which it surely does not hold. However, in practice we find it a useful extension. It is straightforward to prove the following when the augmented sufficient decrease condition (6) is satisfied and Algorithm 2 does not break down: If the iterates have a limit point interior to the positive orthant at which F' is non-singular, then that point must be a solution and the iterates must converge to it. The proof is analogous to the proof of Theorem 3.4 in [6], except that the sufficient decrease condition has to be replaced by the augmented sufficient decrease condition (6).

Often, positivity is achieved by clipping, i.e., replacing after completion of a time step negative concentrations by zero. Although this produces correct steady state solutions, it violates mass conservation during transients. The above approach conserves mass and is found to be more accurate than clipping in tests in Section 5.

3. Preconditioned linear solver

The Newton step (2) is approximated by a preconditioned Krylov subspace method. The partial derivatives of the chemistry terms cause the Jacobian matrix to be non-symmetric. Generally speaking, for non-symmetric linear systems there are three classes of methods: the class of GMRES methods [13], the class of Bi-CGSTAB methods [13], and IDR methods [14].

The convergence speed of Krylov solvers depends mainly on the preconditioner. For the type of problems considered in this study preconditioned GMRES and Bi-CGSTAB have a similar convergence behavior. Therefore, it is expected that the class of IDR methods will not perform significantly better. From the point of view of memory usage the Bi-CGSTAB method is favorable over the GMRES method, i.e., the number of vectors in memory are seven whereas the number of vectors in memory and the work for GMRES depends on the number of linear iterations. Moreover, in this study the matrices are structured, and thus the matrix–vector product is relatively cheap. All the above led to the choice of Bi-CGSTAB as the iterative linear solver in our codes.

3.1. Condition of the Newton equation

The large disparity in timescales for advection, diffusion and reaction leads to entries in the Jacobian matrix differing orders of magnitude from each other. Hence, a large spread in the eigenvalue distribution of the Jacobian might occur.

Definition 3.1. The condition number for matrix inversion with respect to a matrix norm $\|\cdot\|$ of a square matrix A is defined by $\kappa(A) = \|A\| \cdot \|A^{-1}\|$, if A is non-singular; and $\kappa(A) = +\infty$ if A is singular.

In [17] we computed the typical order of magnitude of the condition number of the Jacobian as a function of real time for the two-dimensional benchmark of Kleijn [11]. The Euler Backward time integration method was used, such that the time step size remains relatively large, and a projected Newton method was used to maintain positive approximated solutions. If more than one Newton equation per time step had to be solved, then the average of these condition numbers was taken. In particular, when the chemistry is most active, the condition numbers shoot up to $\mathcal{O}(10^{11})$. Therefore, application of Bi-CGSTAB without preconditioning is ruled out.

3.2. Ordering of unknowns

Essential for the performance of direct linear solvers and iterative linear solver combined with an incomplete factorization type preconditioners is the ordering of unknowns. For the reacting flow problems studied here, the number of unknowns is equal to $N \cdot n$, where N is the number of species in the gas mixture and n the total number of gridpoints resulting from a multi-dimensional spatial discretization.

The *natural ordering* of the unknowns is per species equation. The corresponding nonzero pattern, from a two-dimensional example, is illustrated in Fig. 1. Along the diagonal blocks, which are of dimension equal to the number of grid points, we find the partial derivatives of the discretized advection, diffusion and reaction term with respect to of that particular species. The partial derivatives of the reaction terms in Eq. (1) with respect to the remaining $(N-1)$ species appear as extra sub- or superdiagonals. For multi-dimensional computational meshes with n grid points the bandwidth of the Jacobian matrix is then $(N-1)n$.

The bandwidth decreases considerably by ordering the unknown species mass fractions per grid point. For a two-dimensional computational grid with nr grid points in radial direction and nz grid points in axial direction, the bandwidth of the Jacobian matrix equals $nr \cdot N$. Remark that in this case we label the unknowns first in radial direction and thereafter in axial direction. The corresponding nonzero pattern of the Jacobian is shown in Fig. 2.

3.3. Incomplete LU factorization preconditioners

Since the computational grids are regularly structured, the Jacobian matrix is regularly structured as well. This property can be exploited to formulate incomplete LU factorization preconditioners in a simple way. In standard texts like that of Barret et al. [1] or Saad [13], this has been illustrated for the inhomogeneous steady state advection–diffusion equation on a rectangular domain. Spatial discretization is done by central Finite Volumes, where of course it is assumed that this discretization is stable.

These algorithms are easily extended for the species equations (1) with a number of species larger than one. The extra nonzero sub- and superdiagonals should be treated in the same way as the off-diagonals for the advection–diffusion case described above. For both orderings discussed in Section 3.2 with corresponding nonzero structures as in Figs. 1 and 2, this extension is straightforward.

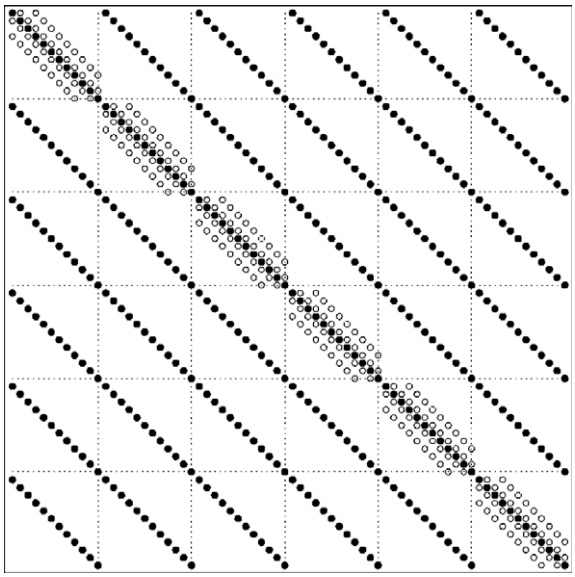


Fig. 1. Nonzero pattern of the Jacobian-matrix for $N = 6$ and the unknowns ordered in a natural way. The circles are partial derivatives with respect to advection and diffusion. In the case of 'lumping' this matrix, see Section 3.4.1, these diagonals marked by circles are added to the main diagonal.

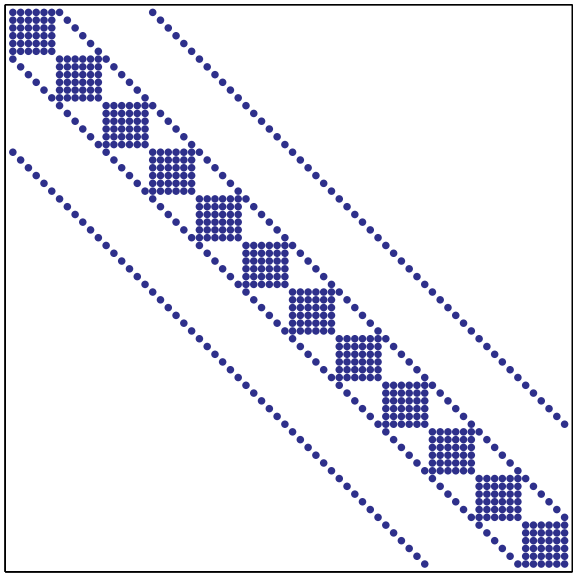


Fig. 2. Nonzero pattern of the Jacobian-matrix for $N = 6$ for the per grid point ordering.

Basic iterative methods like Jacobi or Gauss–Seidel converge more quickly if the diagonal entry is relatively large compared to the off-diagonals in its row or column. Techniques like block iterative methods can benefit if the entries in the diagonal blocks are large. For preconditioning techniques it is intuitively evident that large diagonals should be beneficial, see [5]. Comparing both orderings, it is seen that for the per grid point ordering, the partial derivatives of the reaction terms are clustered in the diagonal blocks, see Fig. 2. Numerical experiments reveal that this ordering indeed enhances the convergence speed of the iterative linear solver. The results are presented in Section 5.

3.3.1. Block incomplete factorization

For both orderings there is a natural block structure in the nonzero pattern of the Jacobian matrix. For the natural ordering, see Fig. 1, a block structure with blocks of dimension n, with n the number of grid points, over the species is present. Building an incomplete factorization on block level for this nonzero structure converts the diagonal blocks in the strictly low-

er triangular part of the nonzero structure into band submatrices. For fine multi-dimensional meshes, and a large number of reactive species in the gas mixture, it is impossible to store the strictly lower triangular part in computer memory.

Building an incomplete LU factorization on block level for the per grid point ordering appears to be efficient. In this section it will be illustrated for a rectangular computational grid on which the species Eq. (1), in cylindrical coordinates, are discretized by means of the hybrid Finite Volume scheme [11,18].

Denote nr as the number of grid points in radial direction and nz the number of grid points in axial direction, such that the total number of grid points is $n = nr \cdot nz$. Ordering the unknown species mass fractions per grid point generates a Jacobian matrix with a nonzero structure consisting of blocks with a dimension equal to the number of species N. The blocks on the diagonal $A_{ii},\ i = 1,\ldots,n$, are full. The off-diagonal nonzero blocks $A_{i-1,i}, A_{i,i-1}, A_{i-nr,i}$ and $A_{i,i-nr}$ are diagonal (sub)matrices, see Fig. 2.

The Jacobian matrix can be split into three matrices, namely,

1. a matrix D, containing all blocks A_{ii} on the main diagonal,
2. the strictly upper part U, containing the blocks $A_{i-1,i}$ and $A_{i-nr,i}$, and,
3. the strictly lower part L, containing the blocks $A_{i,i-1}$ and $A_{i,i-nr}$.

The block incomplete LU factorization preconditioner is then written as

$$P = (D + L)D^{-1}(D + U), \tag{8}$$

where D is the block diagonal matrix containing the block pivots generated. Since the upper and lower triangle parts of the matrix remain unchanged, only storage space for D is needed. Its construction is described in Algorithm 3

Algorithm 3. Block ILU

Put $D_{ii} = A_{ii}$ for all $i = 1,\ldots,n$
for $i = 2,\ldots,n$ **do**
  if $\mathrm{mod}(i, nr) \neq 0$ **then**
    $D_{i+1,i+1} = D_{i+1,i+1} - A_{i+1,i}D_{ii}^{-1}A_{i,i+1}$
  end if
  if $i + nr \leqslant N \cdot n$ **then**
    $D_{i+nr,i+nr} = D_{i+nr,i+nr} - A_{i+nr,i}D_{ii}^{-1}A_{i,i+nr}$
  bf end if
end for

The preconditioning step $Px = b$ within Bi-CGSTAB is solved using the equivalent formulation: First solve z from $(D + L)z = b$, then solve x from $(I + D^{-1}U)x = z$. It is outlined in Algorithm 4.

Algorithm 4. Preconditioner solve of a system $Px = b$, with $P = (D + L)D^{-1}(D + U)$

for $i = 1,\ldots,n$ **do**
  Solve $D_{ii}z_i = b_i - \sum_{j<i}L_{ij}z_j$
end for
for $i = n,\ldots,1$ **do**
  Solve $D_{ii}y = \sum_{j>i}U_{ij}x_j$
  Put $x_i = z_i - y$
end for

For the right multiplication of D_{ii}^{-1} and the diagonal matrix $A_{i,i+1}$, as found in Algorithm 3, we proceed as follows. The inverse of D_{ii} is computed exactly using the Gauss–Jordan decomposition, see for instance [15]. The resulting inverse matrix is then multiplied by the diagonal matrix $A_{i,i+1}$. Consequently, to compute the solutions of the linear systems

$$D_{ii}y = \sum_{j>i}U_{ij}x_j, \quad \text{and} \quad D_{ii}z_i = b_i - \sum_{j<i}L_{ij}z_j, \tag{9}$$

in Algorithm 4, the inverse matrices of D_{ii} are reused.

An alternative approach to compute $D_{ii}^{-1}A_{i,i+1}$ is to build the LU factorization of D_{ii} and subsequently solve N linear systems. In terms of floating point operations (flops), this approach costs $2/3N^3 + N \cdot N^2 = 5/3N^3$ flops. The approach using the Gauss–Jordan decomposition needs N^3 flops to compute the exact inverse, and N^2 for the multiplication with the diagonal matrix. Based on the amount of flops we choose the Gauss–Jordan decomposition.

3.4. Block diagonal preconditioners

Using the per grid point ordering of unknowns an alternative approximation of the Jacobian matrix is the block diagonal matrix, obtained by omitting the off-block diagonal elements. The resultant approximate Jacobian is easily invertible, because it consists of small, easily factorizable subsystems on the diagonal blocks.

3.4.1. Lumping

By 'lumping' the Jacobian matrix, where it is important to lump the same species, a block diagonal nonzero structure is obtained as well. Thus, for the nonzero structure as in Fig. 2 the four off-block diagonals are added to the main diagonal.

Mathematically, the off-block diagonal elements represent the contributions of the discretized advection–diffusion operator of the neighboring points of a certain grid point C in the computational grid. Since these approximations are mostly second order accurate, such a contribution of a neighbor point of grid point C equal to the value of the solution in grid point C up to a first order truncation error. Thus, the contribution of this neighbor can be replaced by this first order approximation. Hence, a first order accurate approximation of the Jacobian matrix has been constructed.

This lumping approach can also be applied to the Jacobian matrix with the unknowns ordered in the natural ordering, see Fig. 1. In that case, the diagonals marked by circles in Fig. 1 should be added to the main diagonal. When constructing the resulting lumped matrix, which is a matrix with $(N-1)$ superdiagonals and $(N-1)$ subdiagonals, the LU factors have the same nonzero pattern as the lumped matrix. However, the implementation for this ordering is more difficult than for the per grid point ordering.

3.5. Comparison of costs: flops

To indicate the amount of work for the above preconditioners, we present for each of them the number of floating point operations (flops) needed to build the preconditioner P and to solve $Px = b$. Note that per Newton iteration the preconditioner is built once, and that $Px = b$ is solved twice in each Bi-CGSTAB iteration. From Table 1 it is concluded that the lumped Jacobian and the block diagonal are, in terms of flops, the cheapest to build, i.e., the number of flops scales linearly in the total number of grid points n and cubically in the number of gas-phase species N. The most expensive preconditioner to build is the ILU(0) preconditioner.

The extra fill-in for block ILU results also in extra computational costs for solving $Px = b$. The preconditioned systems that are the cheapest to solve, in terms of flops, are those belonging to the lumped Jacobian and the block diagonal.

4. Test problems

Numerical simulations presented in Section 5 are done for the CVD process of silicon from silane. The gas phase and surface chemistry are modeled according to the classical model as published by Coltrin and coworkers [3], which includes 17 gas species, 26 reversible gas phase reactions and irreversible 14 surface reactions. Reaction rate constants and thermochemical data have been taken from [3] as well. The general form of the gas phase reaction term in species Eq. (1) is

$$R_k^g = A\left[\frac{\rho\omega_1}{m_1}\right]^{\beta_1}\cdots\left[\frac{\rho\omega_N}{m_N}\right]^{\beta_N} T^{\delta}\exp(-E/RT), \tag{10}$$

where A, β_i, δ and E are fit parameters, R the universal gas constant and T the local temperature. The exact fit parameters for the forward reaction rates and the thermochemical data used to compute the backward reaction rates can be found in Table 1 in [16]. At the wafer surface it is assumed that irreversible, unimolecular decomposition reactions take place. Detailed descriptions of the surface chemistry model are found in [3,11,16,18].

4.1. Reactor configurations

For the two reactor configurations considered in this paper the reactor inlet gas mixture consists of 0.1 mole% silane diluted in the inert carrier gas helium. Further, the inflow temperature of the gas mixture is 300 K, and the susceptor is heated up to 1000 K. The pressure in the reactor is equal to the atmospheric pressure.

4.1.1. Two-dimensional reactor

The first reactor configuration is illustrated in Fig. 3. As computational domain has been taken, because of axisymmetry, one half of the $(r–z)$ plane. The boundary conditions are summarized in Fig. 3 as well. The susceptor at the bottom of the reactor is *not* rotating.

Table 1
Number of floating point operations to build the preconditioner P and to solve $Px = b$. The total number of grid points is denoted as n and N denotes the number of species.

	Building P	Solving $Px = b$
ILU(0)	$8N^3n$	$2(N^2+4N)n$
Block ILU	$2(N^3+3N^2)n$	$6N^2n$
Lumped Jacobian	$2/3N^3n$	$2N^2n$
Block diagonal	$2/3N^3n$	$2N^2n$

Because we use the dilute mixture approach [10], the flow fields can be considered in steady state during the instationary calculations. The streamlines and temperature field are shown in Fig. 3.

4.1.2. Three-dimensional reactor

The reactor configuration resulting in a three-dimensional computational domain is illustrated in Fig. 4. The boundary conditions for inflow, outflow, solid walls and reacting surface, are identical to those imposed on the two-dimensional reactor.

The streamlines and temperature distribution for the reactor, without inflow- and outflow pipes, are shown in Fig. 5.

5. Numerical experiments

This section is organized as follows. In Section 5.1 integration statistics are presented for the two-dimensional computational domain. In our prior work [18] the solutions of the two-dimensional problem have been validated. In Section 5.2 the projected Newton approach is compared with the clipping approach. The strength of the projected Newton method is even more clearly visible in the three-dimensional case, of which solutions and integration statistics are presented in Section 5.3.

5.1. Two-dimensional simulations

The simulations in this section run from inflow conditions until the steady state solution is reached. In all these simulations we allow the maximum number of time steps to be 1000. With respect to the number of allowed Newton iterations per time step we remark the following. In the time frame right before steady state is reached, we experienced that to find the correct search direction might take a few extra Newton iterations. Therefore, the maximum number of Newton iterations is set to 50.

The computational grids are equidistant in radial direction, whereas the grid spacing in axial direction gradually decreases towards the wafer surface. We consider three different grids, namely a grid with $nr = 35$ grid points in radial direction and $nz = 32$ in axial direction, an $nr = 35$ by $nz = 47$ grid, and an $nr = 70$ by $nz = 82$ grid.

In Tables 2 and 3 some relevant statistics are listed for the various forcing terms and preconditioners. Further, in Fig. 6 the total CPU times are shown. With respect to total CPU time it can be concluded that the incomplete factorization precondi-

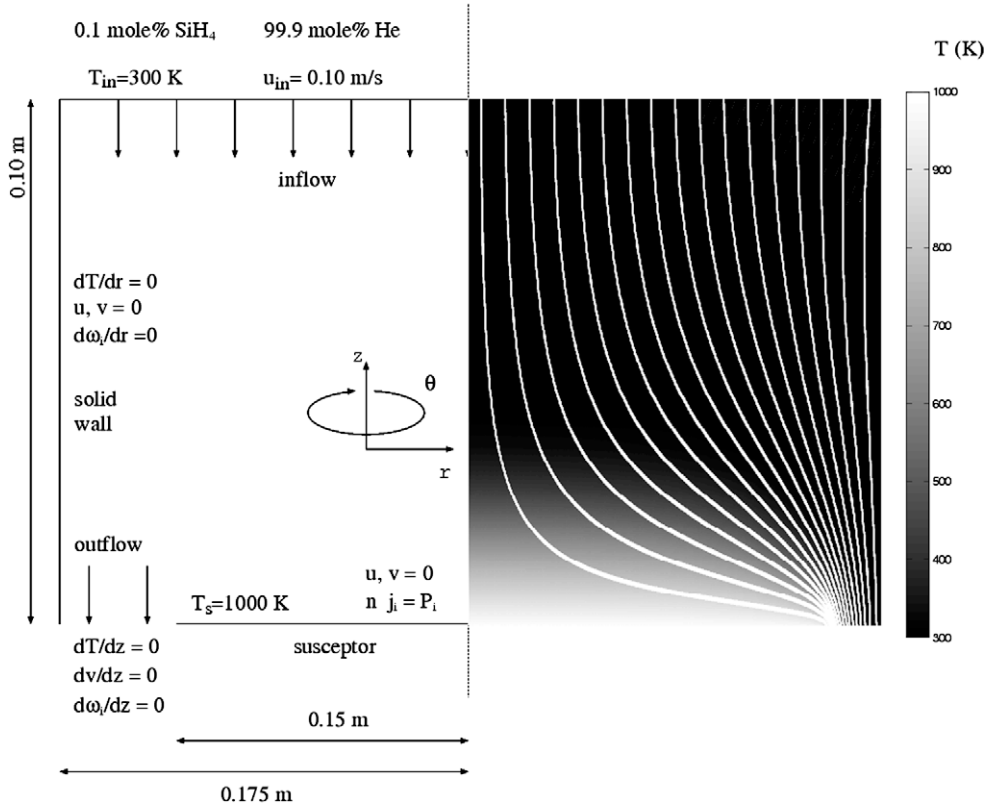
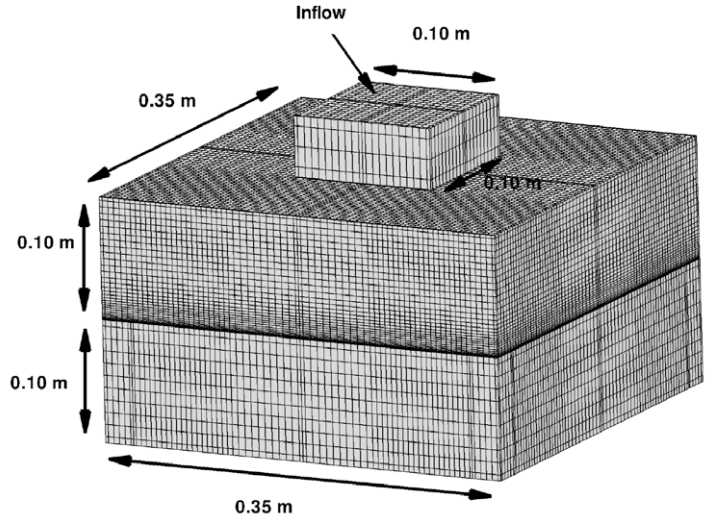
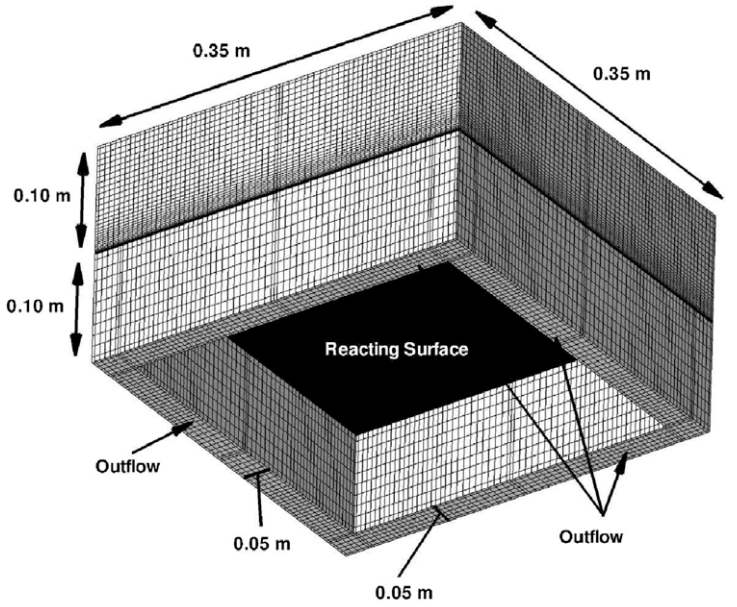


Fig. 3. Reactor geometry, dimensions, boundary conditions, streamlines and temperature field in Kelvin.



(a) Side view



(b) Bottom view

Fig. 4. Side and bottom view of the three-dimensional reactor geometry. Due to two symmetry planes the computational domain can be restricted to one quarter of the reactor.

tioners perform significantly better than the block diagonal preconditioners. On the finest grid the solver equipped with these preconditioners do not converge in the purely transient phase. The block incomplete factorization preconditioner is favorable over the ILU(0) preconditioner, when looking to CPU times. Furthermore, note that for most preconditioners using projected Newton instead of globalized inexact Newton leads to a slight improvement in terms of computational efficiency. However, combining the projected Newton method with the block diagonal preconditioners and forcing term (5) gives a considerable improvement of the computational effort needed.

From Tables 2 and 3 it can clearly be seen that for larger meshes the number of Bi-CGSTAB and Newton iterations increases considerably. Looking to the results of the ILU(0)- and Block ILU preconditioner, it can be concluded that both behave well with respect to positivity, and thus the differences between the projected and regular Newton method are minimal. As remarked above, the Block ILU preconditioner is overall computationally cheaper than ILU(0). This can be explained by the fact that a considerably less amount of linear iterations is needed, see Tables 2 and 3. For finer grids the ratio

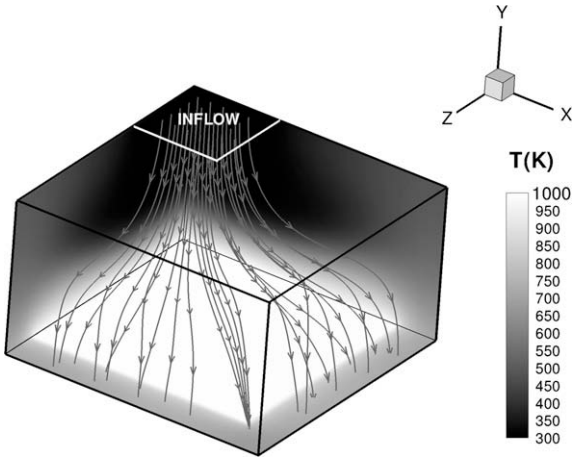


Fig. 5. Streamlines and temperature distribution in Kelvin for a quarter of the reactor chamber of Fig. 4, without inflow and outflow regions.

Table 2
Number of Bi-CGSTAB and Newton iterations for forcing terms Choice 1 (4) and Choice 2 (5) and preconditioners on three computational grids for the globalized inexact Newton method. If a steady state has not been reached then we write nf in the corresponding entry. Further, the number of rejected time steps due to negative species, denoted as # Neg., are specified.

	35×32			35×47			70×82		
	# lin. it.	# Newt.	# Neg.	# lin. it.	# Newt.	# Neg.	# lin. it.	# Newt.	# Neg.
Choice 1									
ILU(0)	848	108	1	2073	205	0	7522	353	0
Block ILU	624	111	2	772	153	0	2069	325	0
Lump	4987	152	0	72,091	1241	177	nf	nf	nf
Block diagonal	4219	149	1	22,498	285	2	nf	nf	nf
Choice 2									
ILU(0)	1129	101	1	2541	194	1	11,100	395	3
Block ILU	838	104	2	859	148	0	2144	299	0
Lump	7927	149	2	106,833	1391	122	nf	nf	nf
Block diagonal	13,371	1379	403	28,140	2054	583	nf	nf	nf

Table 3
Number of Bi-CGSTAB and Newton iterations for forcing terms Choice 1 (4) and Choice 2 (5) and preconditioners on three computational grids for the globalized inexact projected Newton method. If a steady state has not been reached then we write nf in the corresponding entry.

	35×32		35×47		70×82	
	# lin. it.	# Newt.	# lin. it.	# Newt.	# lin. it.	# Newt.
Choice 1						
ILU(0)	825	101	2086	211	7930	385
Block ILU	556	97	772	153	1921	308
Lump	4654	149	10,655	250	nf	nf
Block diagonal	4313	133	25,605	290	nf	nf
Choice 2						
ILU(0)	1009	94	2505	202	8895	351
Block ILU	718	93	859	148	2290	306
Lump	5819	127	23,598	196	nf	nf
Block diagonal	6275	125	29,253	223	nf	nf

$$\frac{\#\ \text{ILU(0) preconditioned Bi-CGSTAB iters}}{\#\ \text{Block ILU preconditioned Bi-CGSTAB iters}} \tag{11}$$

increases, which means that Block ILU performs better for finer grids. Apparently, the extra fill-in generated by the Block ILU preconditioner, which is a combination of large and small entries, gives a much better approximation of the Jacobian matrix than the ILU(0) preconditioner.

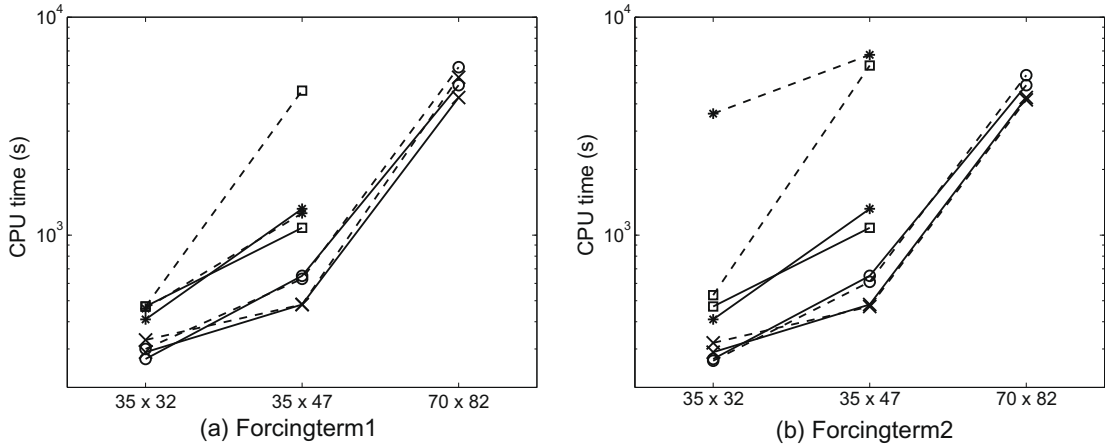


Fig. 6. Total CPU times for different grids and forcing terms. Solid lines correspond to the projected Newton method, whereas dotted lines correspond to the regular Newton method. Line mark ∘ corresponds to ILU(0) preconditioning, × corresponds to block-ILU, □ corresponds to lumping and ∗ corresponds to block diagonal preconditioning.

The block diagonal and lumping precondioner perform bad with respect to positivity, in particular combined with forcing term (5). In this case the projected Newton method brings relief. The computational costs decrease by a factor 10, but are still higher than for the incomplete factorization type preconditioners. Mainly, this is due to the total number of linear iterations, which is between a factor of 5–10 higher. Probably, the fact that the inverse of the Jacobian is approximated by inverting only the 'large' terms, is not good enough.

Furthermore, we remark that for the simulations without projected Newton, and in which the number of time step rejections is equal to zero, the integration statistics are not necessarily equal to those with projected Newton. In between not converged globalized inexact Newton iterations the species concentrations may be negative. The projected Newton does not allow these negative entries, and, hence, different integration statistics are found. Compare, for example, the results for the 70×82 grid in Tables 2 and 3.

To conclude, in Table 4 we compare the integration statistics of the incomplete LU factorization for the natural and per grid point ordering. It is quite clear that the different orderings have significant effect on the total computational effort needed to perform the simulation. Mainly, the computational costs are due to the linear solver. When less Bi-CGSTAB iterations are needed to find the solution, up to a certain accuracy, the total costs are expected to be lower. From Table 4 it can be seen that the total number of linear iterations for the per grid point ordering is significantly less than for the natural ordering. Moreover, the approximated linear solutions obtained in the per grid point ordering are more accurate than those obtained with the natural ordering. This results in a lower number of Newton iterations.

5.2. Comparing projected Newton methods with clipping

Although both approaches compute identical steady states, differences are found when calculating the instationary solution. To show the differences between the clipping on time level and the projected Newton method we compare mass balances at time $t = 2$ s for the atoms e = Si, H and He. At time $t = 2$ s we compute for atom e

$$\frac{\int_0^2 Q_{\text{in},e} - Q_{\text{dep},e} - Q_{\text{out},e}\,dt - \int_{\text{reactor}} c_e(2,r,z)\,dS}{\int_0^2 Q_{\text{in},e}\,dt}, \tag{12}$$

where $Q_{\text{in},e}$ is the molar inflow rate of atom e in the inlet, $Q_{\text{dep},e}$ the molar deposition rate of atom e, $Q_{\text{out},e}$ the molar outflow rate of atom e in the outlet and $c_e(2,r,z)$ the molar concentration of atom e in the reactor at time $t = 2$ s and spatial coordinate

Table 4
Integration statistics for globalized inexact Newton combined with ILU(0) as preconditioner for two orderings of the unknowns.

Mesh size	Per grid point			Natural		
	35×32	35×47	70×82	35×32	35×47	70×82
F	203	407	675	205	476	811
F'	108	213	353	114	252	461
Newton	108	213	353	114	252	461
Linesearch	9	61	124	8	76	138
Lin. it.	848	2111	7171	1131	2455	7942

Table 5
Values of expression (12) for the Si, H and He atoms on different grids.

	35×32	35×47
Si	$-2.3 \cdot 10^{-2}$	$-1.02 \cdot 10^{-2}$
H	$-3.1 \cdot 10^{-2}$	$-2.4 \cdot 10^{-2}$
He	$\mathcal{O}(10^{-10})$	$\mathcal{O}(10^{-10})$

(r,z). For simulations with the projected Newton method on the 35×32 and 35×47 grids the absolute value of expression (12), which should be zero, is of order $\mathcal{O}(10^{-8})$ for all atoms e. However, when using the clipping strategy, in which mass is added when negative species concentrations are set to zero, the values found for expression (12) are listed in Table 5.

Thus, on the 35×32 grid 2% is added to the total moles of silicon atoms that entered the reactor, and 3.1% is has been added to the total moles of H atoms that entered the reactor. On the 35×47 grid this is 1% and 2.4% for the silicon and hydrogen atoms respectively. This result clearly shows that the projected Newton method preserves mass, whereas clipping fails. We believe that larger differences are found when performing numerical experiments for inherently transient chemically reacting flow problems.

5.3. Three-dimensional experiments

To the authors' knowledge, similar results for three-dimensional transient simulations with a chemistry model with 17 species/26 gas phase reactions/14 surface reactions, or a problem of similar complexity are not published. First, a validation of the steady state solution is done.

5.3.1. Validation of 3D steady state solution

In Fig. 7 we compare some selected species mass fractions along the intersection of the two symmetry planes with the two-dimensional axisymmetric solution at the symmetry axis at $r = 0$. Physically, it is expected that both solutions agree well. Since both steady state solutions agree well, we conclude that the three-dimensional solution found in our computations is correct.

In Fig. 8 the total deposition rate, the deposition rate due to the most important growth species along the diagonal from the center of the wafer to the cornerpoint of the wafer, as well as the radial deposition profiles from the two-dimensional axisymmetric simulations are shown. Comparing the two-dimensional and three-dimensional deposition profiles in the neighborhood of the symmetry axis it is concluded that all deposition rates agree very well. Towards the boundary of the wafer the flow fields of the two-dimensional and three-dimensional simulations differ too much to expect agreement on the deposition rates.

5.3.2. Transient solutions

In Fig. 9 the transient total deposition rate of solid silicon on 2 locations on the wafer are displayed. The deposition rates are monotically increasing in time. At the corner point of the wafer it takes much longer for the deposition rate to reach its steady state value.

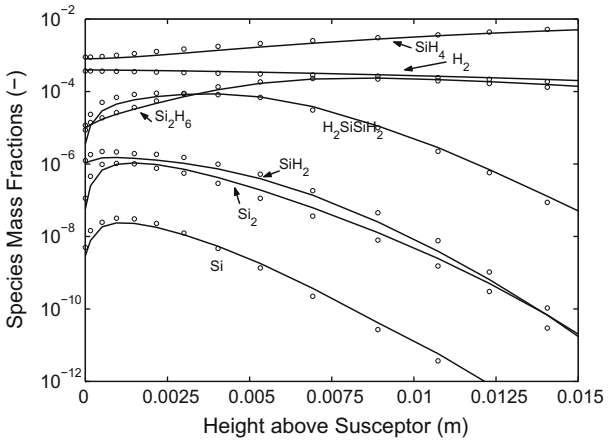


Fig. 7. Axial steady state concentration profiles along the intersection of the two symmetry planes. Solid lines are the profiles belonging to the three-dimensional simulations, circles are profiles along the symmetry axis belonging to the two-dimensional axisymmetric case.

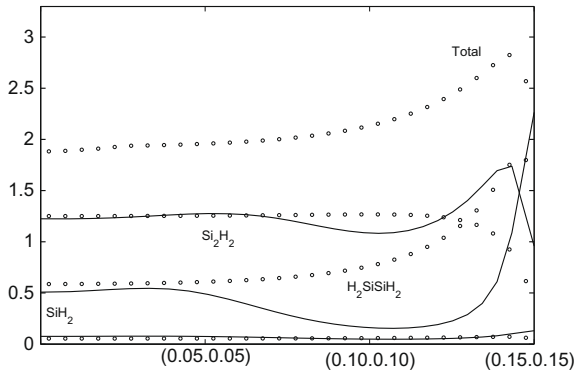


Fig. 8. Solid lines are steady state deposition rates along the intersection of the origin and the cornerpoint of the wafer (i.e., $(x, y, z) = (0.15, 0, 0.15)$). The circles are radial steady state profiles belonging to the two-dimensional axisymmetric case.

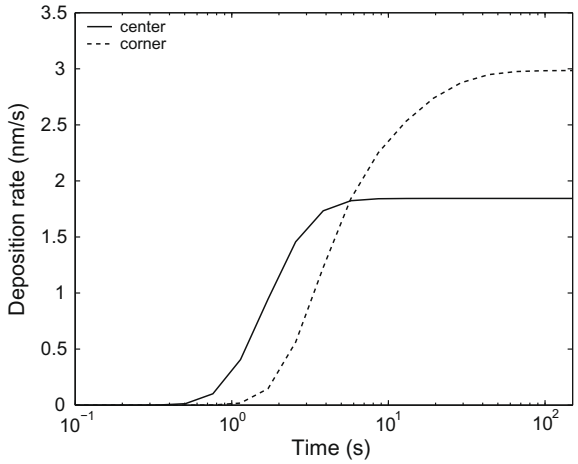


Fig. 9. Transient total deposition rate on the center and on the corner (that is $(x, y, z) = (0.15, 0, 0.15)$) of the susceptor.

5.3.3. Integration statistics

Numerical experiments with globalized inexact Newton methods and Euler Backward time integration reveal that simulations running from inflow conditions to steady state give multiple time step rejections due to negativity for all preconditioners presented in Sections 3.3 and 3.4. Negativity was mostly observed for species having small concentrations. Incidentally, especially during the transient part of the simulation, before reaching steady state, and in certain grid cells in the vicinity of the reacting surface, negativity was also observed for species present in larger concentrations, like silane. For all simulations on the three-dimensional meshes no solutions are found without application of the inexact projected Newton method.

Numerical experiments have been carried out on an $35 \times 32 \times 35$ grid and an $70 \times 70 \times 70$ grid. The integration statistics for the simulations from inflow conditions until steady state with the Euler Backward solver combined with the globalized inexact projected Newton method for the coarse grid are listed in Table 6, and in Table 7 for the $70 \times 70 \times 70$ grid.

Table 6
Number of operations for the 17 species and 26 reactions problem on the three-dimensional computational grid consisting of $35 \times 32 \times 35$ grid cells. The wafer temperature has been set to 1000 K.

	ILU(0)	Block ILU	Lumped Jacobian	Block diagonal
Newton	239	156	332	327
Linesearch	51	20	31	29
Newt. Diver.	3	0	0	0
Acc. time step	44	43	43	43
Lin. it.	3196	2481	17,472	18,392
CPU (s)	20,100	17,500	28,000	29,000

Table 7
Number of operations for the 17 species and 26 reactions problem on the three-dimensional computational grid consisting of $70 \times 70 \times 70$ grid cells. The wafer temperature has been set to 1000 K.

	ILU(0)	Block ILU	Lumped Jacobian	Block diagonal
Newton	539	436	366	367
Linesearch	223	142	114	107
Newt. Diver.	11	11	9	9
Acc. time step	55	53	52	52
Lin. it.	7830	5525	47,105	48,810
CPU (h)	200	167	203	260

Clearly, from Tables 6 and 7 it can be seen that two components of the solver cause increasing computational costs. First, effective preconditioning drops the number of preconditioned Bi-CGSTAB iterations. In Table 7 we see that the total computational costs are mainly determined by the number of linear iterations, i.e., for the block diagonal preconditioners the number of Newton iterations is low and linear iterations is high.

On the other hand, per Newton iteration the Jacobian is evaluated analytically. For the numerical experiments in the present paper the partial derivatives of the chemistry term can be calculated at low cost, such that the exact Jacobian matrix is relatively cheaply assembled. For the experiments on the course grid we see an increasing number of Newton iterations for the 'weaker' block diagonal preconditioners compared to the incomplete factorization preconditioners. Again, this will increase the computational costs for the block diagonal preconditioners. For the numerical experiments on the fine grid we observe larger number of Newton iterations for the incomplete factorization preconditioners. Apparently, for these effective preconditioners in some time steps the Newton step was oversolved. For the ILU(0) preconditioner this phenomena leads to somewhat longer simulation times.

To summarize, looking at computational time the Block ILU preconditioner combined with the globalized inexact projected Newton method is the best method to compute a fully time-accurate transient solution of laminar reacting gas flow problems. Further, the reduction of computational costs is most effectively done by reducing the computational costs of the linear solver by effective preconditioning.

6. Conclusions

In this paper two topics considering the efficient time accurate solution of laminar reacting gas flows are investigated. To conserve the non-negativity of the species concentrations we propose to use inexact projected Newton methods instead of regular inexact Newton methods. Projected Newton methods are widely applied in constraint optimization, but are generally unknown in the field of PDEs and reacting flow simulations. Numerical experiments revealed that this inexact projected Newton method combined with various preconditioners, while leading to slight improvement in computational efficiency for two-dimensional simulations, is indispensable for three-dimensional simulations. Further, we have shown through numerical experiments that traditional clipping methods, which are not mass conserving, give less accurate time dependent solutions than the projected Newton methods, which conserve mass.

The total computational costs are also determined by the efficiency of the linear solver. In this paper preconditioned Krylov solvers are used, and various preconditioners are presented and compared. Choosing the block ILU preconditioner combined with the project Newton methods enables us to compute time dependent solutions, from inflow until steady state, on a $70 \times 70 \times 70$ grid with 17 reactive species.

Acknowledgment

The work of S. van Veldhuizen was supported by the Delft Centre for Computational Science and Engineering. We gratefully acknowledge Professor Piet Wesseling for the helpful discussions and suggestions during the preparation of this work.

References

[1] R. Barret, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.A. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1994.
[2] D.P. Bertsekas, Projected Newton methods for optimization problems with simple constraints, SIAM J. Control Optim. 20 (1982) 221–246.
[3] M.E. Coltrin, R.J. Kee, G.H. Evans, A mathematical model of the fluid mechanics and gas-phase chemistry in a rotating Chemical Vapor Deposition reactor, J. Electrochem. Soc. 136 (1989) 819–829.
[4] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400–408.
[5] I.S. Duff, J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, SIAM J. Matrix Anal. Appl. 20 (4) (1999) 889–901.
[6] S.C. Eisenstat, H.F. Walker, Globally convergent Inexact Newton methods, SIAM J. Optim. 4 (1994) 393–422.
[7] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 16–32.
[8] M.L. Hitchman, K.F. Jensen, Chemical Vapor Deposition – Principles and Applications, Academic Press, London, 1993.
[9] C.T. Kelley, Solving Nonlinear Equations with Newton's Method, Fundamentals of Algorithms, SIAM, Philadelphia, 2003.

[10] C.R. Kleijn, Chemical vapor deposition processes, in: M. Meyyappan (Ed.), Computational Modeling in Semiconductor Processing, Artech House, Boston, 1995, pp. 97–229. Chapter 4.
[11] C.R. Kleijn, Computational modeling of transport phenomena and detailed chemistry in Chemical Vapor Deposition – A benchmark solution, Thin Solid Films 365 (2000) 294–306.
[12] K.J. Kuijlaars, C.R. Kleijn, H.E. A van den Akker, Multi-component diffusion phenomena in multiple-wafer chemical vapour deposition reactors, Chem. Eng. J. 57 (1995) 127–136.
[13] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, 2003.
[14] P. Sonneveld, M.B. van Gijzen, IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems, SIAM J. Sci. Comput. 31 (2008) 1035–1062.
[15] G. Strang, Introduction to Linear Algebra, Wellesley-Cambridge Press, Massachusetts, 2003.
[16] S. van Veldhuizen, C. Vuik, C.R. Kleijn, Numerical methods for reacting gas flow simulations, Int. J. Multiscale Eng. 5 (2007) 1–10.
[17] S. van Veldhuizen, C. Vuik, C.R. Kleijn, A class of projected Newton methods to solve laminar reacting flow problems. Report 08-03, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2008.
[18] S. van Veldhuizen, C. Vuik, C.R. Kleijn, Comparison of ODE methods for laminar reacting gas flow simulations, Numer. Meth. Part. Diff. Eq. 24 (2008) 1037–1054.
[19] J. Warnatz, U. Maas, R.W. Dibble, Combustion, second ed., Springer-Verlag, Heidelberg, 1999.