DELFT UNIVERSITY OF TECHNOLOGY

REPORT 08-03

A class of projected Newton methods to solve laminar reacting flow problems

S. van Veldhuizen C. Vuik C.R. Kleijn

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2008

A class of projected Newton methods to solve laminar reacting flow problems

S. van Veldhuizen[1], **C. Vuik**[1], **C.R. Kleijn**[2]

[1] Delft University of Technology, Delft Institute of Applied Mathematics and J.M. Burgerscentrum, Mekelweg 4, 2628 CD Delft, The Netherlands
[2] Delft University of Technology, Department of Multi Scale Physics and J.M. Burgerscentrum, Prins Bernardlaan 6, 2628 BW Delft, The Netherlands

Received: February 19, 2008/ Revised version: February 19, 2008

Abstract The numerical modeling of the transport phenomena and the multispecies, multireaction chemistry in laminar reacting gas flow processes such as chemical vapor deposition typically involves the solution of large numbers of advection-diffusion-reaction equations, which are stiffly coupled through the reaction terms. Stability and positivity requirements of the solution basically reduce the time integration to be first order Euler Backward. This paper is devoted to the reduction of the computational costs within Newton's method by means of introducing preconditioned iterative linear solvers. However, solving nonlinear systems in an iterative way does not guarantee the positivity of the solution. In particular when the linear systems within Newton's method are not solved exactly, the nonlinear solution may have many small negative elements. To circumvent this, we introduce a projected Newton method. We conclude by comparing various preconditioners for the acceleration of the internal linear algebra problem.

1 Introduction

The class of nonlinear solvers that are a generalization of Newton's method in the following sense are called Inexact Newton solvers. Instead of factorizing the Jacobian and solving the Newton step directly, the Newton step is approximated by means of an iterative linear solver. Some well known references on Inexact Newton Methods are [3], [6] and [13], in which theoretical background and numerous applications and examples are given. However, for practical applications in for instance multi-dimensional industrial flow process simulations, it is relevant to explore the possibilities to get more

Correspondence to: s.vanveldhuizen@tudelft.nl

efficient linear solvers within an Inexact Newton method, because this is one of the cost drivers. Another important cost driver is the computation of the Jacobian. In this paper we explore possibilities to reduce the computational costs of time accurate transient simulations of laminar reacting gas flows as found in Chemical Vapor Deposition (CVD). For comprehensive descriptions of CVD we refer to [8,12]. As can be found in, for instance [21], implicit time integration is needed to positively integrate the stiff set of ODEs resulting from the discretization of the equations in the mathematical model. Consequently, huge nonlinear systems have to be solved. To reduce the computational costs efficient preconditioners for iterative linear solvers have to be developed. In this paper we explore several well known preconditioning techniques and suggest how they should be used in the context of chemically reacting flows. Further, we discuss the issue of positivity for iterative solvers.

In this paper CVD is modeled as a continuum reacting gas flow process. Its mathematical model describes flow, temperature distribution and chemistry within the reactor. The main gas flow is modeled by the continuity equation, the Navier Stokes equations and the equation of state (perfect gas law). The temperature part is described by the energy equation in terms of temperature. The chemistry part is described by the so-called species equations, which describe the transport of gas species and their conversion through various chemical reactions in the gas mixture in terms of mass fractions. Hence, the species equations are of the advection-diffusion-reaction type, i.e.,

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

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 diffusion coefficient for species i, $\mathbb{D}_i^T$ 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 4.1.

In this paper we will consider a CVD process in which the reacting gas species are highly diluted in an inert carrier gas. For such a process, the gas flow and temperature are not influenced by the reactions and the computation of the steady flow and temperature field is quite trivial. Only the species equations need to be solved in a time accurate manner.

The solutions of the other PDEs involving the mathematical model are computed using the codes developed by Kleijn [11]. Further details on computational algorithms in this code can be found in [11]. In [10] and [11] further details on CVD modeling can be found. In [20] and [21], further details can be found on the computational method of solving the species equations in this way.

Due to the different time scales involving the species transport and their conversion as consequence of chemical reactions, the system of species equations is a stiff system of PDEs. Therefore, implicit time integration will be needed to fulfill stability requirements. Besides the stability issue, the non-negativity of the species mass fractions is also important, because they represent a physical quantity. As we already discussed in previous work, fulfilling this extra constraint is impossible for higher order time integration schemes. Moreover, one can prove that unconditionally positive time integration schemes can be first order accurate only with the consequence that Euler Backward is the only known scheme which can be proven to be unconditionally positive. Therefore, we consider in this paper Euler Backward time integration only. Moreover, we investigate how this property can be maintained within the nonlinear solver.

The paper is organized as follows. First we discuss the Inexact Newton method and propose a generalization which maintains the non-negativity property. Second, various preconditioners are discussed, and if possible we mention their influence on the non-negativity preservation. We conclude with a short description of the chemistry models, the reactor geometry and corresponding boundary conditions, and numerical experiments.

2 Inexact Newton Solvers

Instead of solving the Newton step directly, i.e. $s_k = -[F(x_k)]^{-1}F(x_k)$, the Newton step in Inexact Newton solvers is approximated by an iterative linear solver, in our case a preconditioned Krylov method. This approximate Newton step s_k has to satisfy the so-called Inexact Newton condition

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|, \tag{2}$$

for a certain 'forcing term' $\eta_k \in [0, 1)$. In general form, the algorithm is presented as Algorithm 1.

Algorithm 1 Inexact Newton

Let x_0 be given.
for $k = 1, 2, \ldots$ until 'convergence' **do**
  Find some $\eta_k \in [0, 1)$ and s_k that satisfy

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|.$$

  Set $x_{k+1} = x_k + s_k$.
end for

Note that the inexact Newton condition (2) expresses

1. a certain reduction in the norm of $F(x_k) + F'(x_k)s_k$, which is the local linear model of F in a neighborhood of x_k, and,

2. a certain (relative) accuracy in solving the Newton equation $F'(x_k)s_k = F(x_k)$ iteratively.

Of course, the local convergence behavior of the inexact Newton method depends on the sequence of forcing terms η_k. As has been illustrated in [3], the intuitive idea that smaller values of the forcing terms leads to fewer Newton iterations holds. However, away from a solution, the function F and its local linear model may disagree considerably at a step that closely approximates the Newton step. When choosing η_k too small, this can lead to *oversolving* the Newton equation; meaning that imposing an accurate linear solution to an inaccurate Newton correction may result in a poor Newton update, and, therefore, little or no progress towards a solution. The latter has been experienced in, for instance, [14] and [16]. Moreover, for Newton solvers with forced global convergence algorithms, like line-search (or backtracking), in which additional accuracy in solving the Newton equation requires additional expense, it may entail pointless costs. Then, a less accurate approximation of the Newton step is cheaper, and probably more effective.

In the next section we give some proposed choices on the forcing terms, all taken from [6], that achieve desirable fast convergence and tend to avoid oversolving. All strategies to choose the forcing term incooperate information about F (and are scaling independent).

2.1 Choosing the forcing term

2.1.1 Choice 1 The first choice, taken from [6],is the following. Given the initial forcing term $\eta_0 \in [0, 1)$, then choose

$$\eta_k = \frac{\Big|\|F(x_k)\| - \|F(x_{k-1}) - F'(x_{k-1})s_{k-1}\|\Big|}{\|F(x_{k-1})\|}, \quad k = 1, 2, \ldots. \tag{3}$$

Observe that (3) directly reflects the agreement between F and its local linear model at the previous step. If the initial iterate x_0 is sufficiently near a solution x_*, then the sequence $\{x_k\}$ produced by Algorithm 1 and the forcing term as in (3), converges super-linearly towards a solution. As in the classical case of the secant method, it follows that the order of convergence equals $\frac{1+\sqrt{5}}{2}$; see, for instance, [15], page 293. The irrational number $\frac{1+\sqrt{5}}{2}$ is known as the golden ratio, see [7].

Usually the forcing term (3) avoids oversolving, but it might happen that it is chosen too small. As a safeguard we restrict η_k to be no less than a certain minimal value, which depends on η_{k-1} as $\eta_{k-1}^{(1+\sqrt{5})/2}$. Note that this safeguard should only be activated as η_{k-1} is relative large. Therefore we first check whether $\eta_{k-1}^{(1+\sqrt{5})/2}$ is larger than a certain threshold, and if so, the safeguard is active. As was done in [6], the threshold we use is 0.1. It appeared that this threshold value worked fine in our experiments, and therefore it was not necessary to change it. To summarize:

$$\text{Modify } \eta_k \leftarrow \max\{\eta_k, \gamma\eta_{k-1}^{(1+\sqrt{5})/2}\} \text{ whenever } \gamma\eta_{k-1}^{(1+\sqrt{5})/2} > 0.1.$$

2.1.2 Choice 2 Another way to base the forcing term on residual norms is

$$\eta_k = \gamma \frac{\|F(x_k)\|^2}{\|F(x_{k-1})\|^2}, \tag{4}$$

with $\gamma \in [0, 1)$ a parameter. In our experiments we put $\gamma = 0.5$, which worked fine. Again, we have the safeguard:

$$\text{Modify } \eta_k \leftarrow \max\{\eta_k, \gamma\eta_{k-1}^2\} \text{ whenever } \gamma\eta_{k-1}^2 > 0.1.$$

Note that for the choice of (4) as forcing term, the order of convergence of Inexact Newton equals 2, see [6]. In [9], (4) is chosen as forcing term. A brief discussion on the use of this forcing term can be found in [9].

2.2 The Globalized Inexact Newton Algorithm

It is necessary to introduce a globalized inexact Newton algorithm, because the initial approximate solutions are not always in the neighborhood of a solution. The method that we used to ensure global convergence of the Newton algorithm is line-search, or backtracking as it was called in [6]. The algorithm is given below; it is called Algorithm 2.

Algorithm 2 Globalized Inexact Newton

1: Let x_0, $\eta_{\max} \in [0, 1)$, $\alpha \in (0, 1)$ and $0 < \lambda_{\min} < \lambda_{\max} < 1$ be given.
2: **for** $k = 1, 2, \ldots$ until 'convergence' **do**
3: Find some $\eta_k \in [0, \eta_{\max}]$ and s_k that satisfy
4: $$\|F(x_k) + F'(x_k)s_k\| \le \eta_k\|F(x_k)\|.$$
5: **while** $\|F(x_k + s_k)\| > (1 - \alpha(1 - \eta_k))\|F(x_k)\|$ **do**
6: Choose $\lambda \in [\lambda_{\min}, \lambda_{\max}]$
7: Set $s_k \leftarrow \lambda s_k$ and $\eta_k \leftarrow 1 - \lambda(1 - \eta_k)$
8: **end while**
9: Set $x_{k+1} = x_k + s_k$.
10: **end for**

In the implementation of Algorithm 2 we choose every first initial forcing term equal to one half (1/2) and determine an approximation of the first Newton step s_0 using the Bi-CGSTAB algorithm [17]. The parameters used in the implementation of Algorithm 2 were $\eta_{\max} = 0.9$, $\alpha = 10^{-4}$, $\lambda_{\min} = 1/10$ and $\lambda_{\max} = 1/2$. In the while-loop, each λ was chosen such that it is the minimizer of the quadratic polynomial model of

$$\phi(\lambda) = \|F(x_k + \lambda s_k)\|_2^2, \tag{5}$$

subject to the safeguard that $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. Convergence of the globalized Newton method is declared when $\|F(x_k)\| < \mathrm{TOL}_{\mathrm{rel}}\|F(x_0)\| + \mathrm{TOL}_{\mathrm{abs}}$, where $\mathrm{TOL}_{\mathrm{rel}}$ and $\mathrm{TOL}_{\mathrm{abs}}$ are, respectively, the relative termination tolerance and the absolute termination tolerance of the Newton process. Failure, or divergence, of Algorithm 2 is declared when

- k reached the maximum number of Newton iterations, of which the default value is 100 in our code,
- the Bi-CGSTAB algorithm does not succeed in finding a suitable Newton step within 1000 iterations, or
- the linesearch algorithm is not able to find a suitable Newton step after 10 iterations. (Taking more linesearch iterations into account wouldn't make sense, because then the Newton update s_k would be too small to obtain convergence in the next iterations)

In the case that Newton's method diverges, then the common way to overcome divergence is to decrease the time-step size. In our code, as in many other codes, we halve the time step size and repeat Newton's process.

2.3 Globalized Projected Newton Methods

Preservation of non-negativity of species concentrations in the solution of the species equations crucial in order to avoid blow up of the solution. In earlier work, see [21], we discussed classes of time integration methods that perform well with respect to positivity. The first order Euler Backward method is the only method being unconditionally positive for all time step sizes. However, solving the resulting implicit relation by means of a Globalized (Inexact) Newton Method, see Section 2.2, does not guarantee positivity of the solution vector of species concentrations. Moreover, we observed that for certain preconditioned Krylov methods the returned nonlinear solution repeatedly contains negative species concentrations. Thus, in practice, even for the unconditional positive Euler Backward method, repetitions of negative species concentrations can be observed. For this lacking property of the (Globalized) (Inexact) Newton method we present an adaptation to the algorithm such that it preserves positivity.

The basic idea is to generate sequences $\{x_n\}$ in the positive orthant which converge to a solution x^* of the nonlinear problem $F(x) = 0$. The fact that $\{x_n\}$ is in the positive orthant, gives that the solution x^* contains positive entries. These so-called Projected Newton methods originate from nonlinear optimization problems with constraints, and were first proposed by [1]. To the authors' knowledge, these kind of ideas have not been applied into the field of PDEs.

Application of Projected Newton in the field of PDEs can be done as follows. Suppose we have computed a Newton direction s_k and that the new solution vector $x_k + s_k$ contains negative entries. In Figure 1 this is illustrated for the 2D case. Then, in order to maintain positivity of these entries we

project the negative entries to zero and check whether this projected solution is still in the steepest descent direction. More specific, we test whether the projected solution suffices the sufficient decrease condition, i.e.,

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

where $\mathcal{P}$ is the projection on the positive orthant and α a typical small parameter. The i^{th} entry of $\mathcal{P}(x)$ is given as

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

When condition (6) is not satisfied, the search direction s_k and η_k will be adjusted by means of a linesearch procedure as described in Section 2.2 and Algorithm 2.

Algorithm 3 Globalized Inexact Projected Newton

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

As in the case of linesearch, or backtracking methods, we cannot prove that (6) can always be satisfied. Neither can we derive conditions for which it surely does not hold. However, we can plead on the fact that it is a useful extension. The unconditional positivity of Euler Backward ensures that a non-negative solution exists. If we start with a positive initial guess in a neighborhood of the positive solution, then we may expect that the algorithm converges towards this solution. However, due to the use of approximate Jacobians and/or preconditioned Krylov solvers the solution is most likely approached from a non-positive direction. By projecting the negative entries to zero, it is still likely that we remain in a neighborhood of the solution. The Globalized Inexact Projected Newton method is presented as Algorithm 3.

It is a straightforward exercise to prove that when (6) is satisfied and Algorithm 3 does not break down, it converges to a solution. The proof is an analogue of the proof of Theorem 3.4 in [5], except that the sufficient decrease condition has to be replaced by (6). By overcoming the positivity issue, the remaining problem to be solved is the system of linear equations. The next section is devoted to the preconditioning the linear systems appearing in Newton's method.

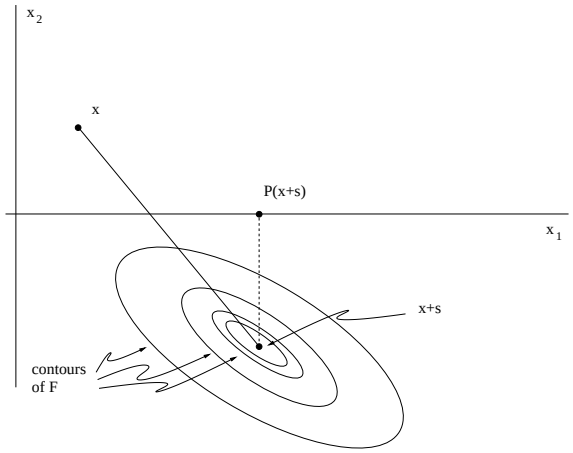


Fig. 1 Illustration of the Projected Newton Method for a nonlinear problem of 2 variables, where $x = [x_1, x_2]^T$ and s the Newton search direction.

3 Preconditioned Linear Solver

Right preconditioned Bi-CGSTAB is used as iterative linear solver. It is well known that a preconditioner M^{-1} should be chosen in such a way that the eigenvalues of AM^{-1} are efficiently clustered in comparison with those of A. Typically, in reacting flow simulations where the linear systems have huge condition-numbers, an effective clustering is needed to get convergence at all. In Figure 2 the typical order of magnitude of the condition-number of the Jacobian is shown as a function of (real) time (in seconds). Application of Krylov solvers without (effective) preconditioning is ruled out.

The ordering of unknowns and equations influences both the performance and construction of a preconditioner. In this paper two orderings are considered:

1. the so-called 'natural' ordering, where the unknowns are ordered per species equation, and,
2. the alternate blocking per grid point ordering, where the unknown species are ordered per grid point.

For both orderings the corresponding non-zero pattern of the Jacobian in the Newton iteration is shown in Figure 3(a) and 3(b).

3.1 Incomplete Factorization

Since the computational grids are structured, it is rather easy to build the incomplete factorization without fill-in for both orderings. Preconditioning techniques can benefit if the entries in the diagonal block are large relative to the off-diagonal blocks, see [4]. Since indeed the 'heavy' elements are in

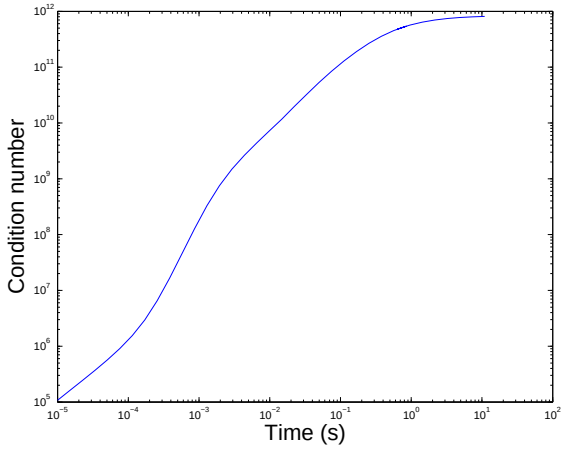
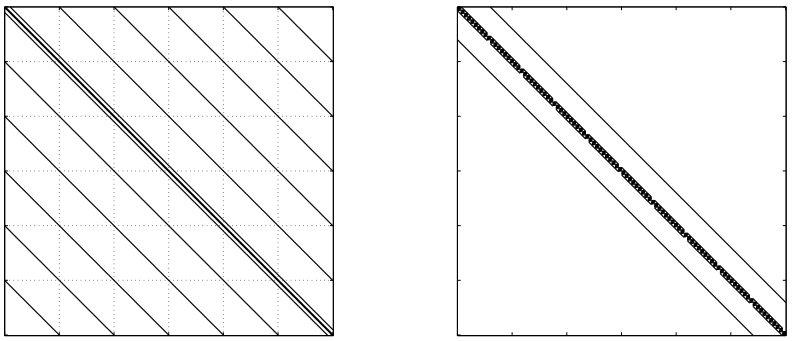


Fig. 2 Condition-number of the Jacobian as function of time (in seconds)



(a) Natural ordering (b) Alternate blocking per grid point

Fig. 3 Nonzero structures of the Jacobian within Newton's method for different orderings

the diagonal blocks in the alternate blocking per grid point ordering, it is expected that the incomplete LU preconditioner for this ordering should be more beneficial than the incomplete LU preconditioner for the natural ordering. Numerical experiments with ILU(0) preconditioners for both orderings have confirmed this. In Table 1 the integration statistics for time accurate simulations running from initial inflow conditions until steady state are presented. Details of the chemical model consisting of 16 species and 26 reactions used in these experiments are discussed in Section 4.

Number of	alternate block	natural
F	197	220
F'	111	124
Newton	111	124
linesearch	7	12
Rej. time	0	0
Acc. time	36	36
lin iter	346	444
CPU time	300	400

Table 1 Integration statistics for GIN with ILU(0) as preconditioner for two orderings of the unknowns.

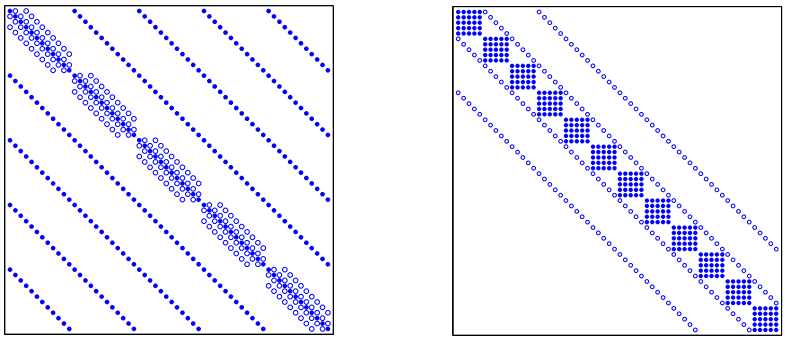
3.2 Lumped Jacobian

Iterative methods will converge more quickly if the diagonal elements are relative large compared to the off-diagonal elements in its row and column. Consider, for instance, the 2D discrete advection-diffusion operator A, for which the advection and diffusion terms are discretized by means of central differences. This operator A could be approximated by a diagonal matrix where the entry on the diagonal of the lumped mass matrix is the sum of all entries of one row of the mass matrix. In the case of a system of species equations one should not sum up all entries of one row of the Jacobian, because then also contributions of other species are added to the diagonal entry. Instead, one should sum up the entries of one row that are related to the same species. This means that for the natural ordering of unknowns the diagonals marked by circles in Figure 4(a) should be added to the main diagonal. Hence, the resulting nonzero structure of the Jacobian is not easy factorizable (or invertible) due to fill-in.

However, for the alternate blocking per grid point ordering the lumped version of the Jacobian is constructed by adding the diagonals marked by circles in Figure 4(b) to the main diagonal. The result is that the lumped approximation consists of small, uncoupled systems on the diagonal. Each of these small subsystems has dimension equal to the number of species, and, hence, is easily decomposed in LU factors.

3.3 Block D-ILU

Denote nr as the number of grid points in radial direction and nz the number of grid points in axial direction for a 2D finite difference discretization in cylindrical coordinates. The total number of grid points is denoted as $n = nr \cdot nz$. Ordering the unknowns according to the alternate blocking per grid point ordering generates a matrix consisting of blocks with a dimension equal to the number of species. The blocks on the diagonal A_{ii}, $i = 1, \ldots, n$ are not sparse. The other nonzero blocks $A_{i-1,i}, A_{i,i-1}, A_{i-nr,i}$ and $A_{i,i-nr}$ are diagonal (sub)matrices.



(a) Natural ordering. (b) Alternate blocking per grid point.

Fig. 4 Nonzero pattern of the lumped approximations to the Jacobian matrix for different orderings of unknowns. The super- and sub-diagonals marked by circles should be added to the main diagonal.

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 lower part L, containing the blocks $A_{i-1,i}$ and $A_{i-nr,i}$, and,
3. the strictly upper part U, containing the blocks $A_{i,i-1}$ and $A_{i,i-nr}$.

The block D-ILU preconditioner is then written as

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

where D is the block diagonal matrix containing the block pivots generated. Generating this preconditioner is described in Algorithm 4. Since the upper and lower triangle parts of the matrix remain unchanged, only storage space for D is needed.

Algorithm 4 Block D-ILU

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

To solve $Mx = b$, with M defined as in (8), we implemented the following equivalent formulation:

$$(D + L)z = b, \qquad (I + D^{-1}U)x = z. \tag{9}$$

Solving $Mx = b$ using this formulation is outlined in Algorithm 5.

Algorithm 5 Preconditioner solve of a system $Mx = b$, with $M = (D + L)D^{-1}(D + U)$

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

With respect to solving systems

$$D_{ii}y = \sum_{j>i} U_{ij}x_j, \tag{10}$$

and

$$D_{ii}z_i = b_i - \sum_{j<i} L_{ij}z_j, \tag{11}$$

as formulated in Algorithm 5, is done by building an LU factorization of D_{ii}. Since the dimension of D_{ii} equals the number of species, and is small with respect to the number of grid points, this is a cheap operation.

For the right multiplication of D_{ii}^{-1} and the diagonal matrix $A_{i,i+1}$ we proceed as follows. We compute the inverse of D_{ii} exactly, with a Gauss-Jordan decomposition, and multiply it with the diagonal matrix. The dimension of D_{ii}, and $A_{i,i+1}$, is denoted as s; s is the number of species in the gas mixture. The total flop count for this multiplication operation is s^3 for the Gauss-Jordan decomposition and s^2 for the multiplication with the diagonal matrix. Another possibility is to use an LU factorization and solve s linear systems, which needs $2/3s^3 + s \cdot s^2$ flops. Based on the amount of flops we use the first approach.

3.4 Block Diagonal Preconditioner

If the unknowns are ordered according to the alternate blocking per grid point ordering, then the non-zero pattern of the Jacobian matrix is as in Figure 4(b). Instead of adding the diagonals marked by circles to the main diagonal, they can also be omitted. The resultant approximate Jacobian is easily invertible, because it consists of small, easily factorizable subsystems on the diagonal blocks.

3.5 Comparison of Flops

To indicate the amount of work for one of the above preconditioners, we present for each of them the number of floating point operations (flops) needed to build the preconditioner P, and the number of flops to solve $Px = b$. Note that per Newton iteration the preconditioner is built once, and the $Px = b$ is solved twice in each Bi-CGSTAB iteration. From Table 3.5 it can be concluded that the incomplete LU-factorization, the lumped Jacobian and the block diagonal are, in terms of flops, the cheapest to build, i.e., the number of flops scales linearly and n and cubically in s. The most expensive preconditioner to build is the blocked version of D-ILU. Thus, the extra fill-in in this preconditioner expresses itself in, of course, extra computational costs.

Table 2 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 s denotes the number of species.

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

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

4 Test Problems

Numerical tests are performed for two CVD processes, which are both shortly described in this section. In the present paper we are only interested in solving the species equations

$$\frac{\partial(\rho\omega_i)}{\partial t} = -\nabla\cdot(\rho\mathbf{v}\omega_i) + \nabla\cdot[(\rho\mathbb{D}_i'\nabla\omega_i) + (\mathbb{D}_i^T\nabla(\ln T))] + m_i\sum_{k=1}^{K}\nu_{ik}R_k^g, \quad (12)$$

and to study the transient behavior of the deposition rate along the wafer. Since the reactants are highly diluted in the inert carrier gas helium, it is justified to assume that the velocity-, temperature-, density- and pressure fields are the steady state fields.

The computational domain for both processes is the same, i.e., both processes occur in the same reactor. The reactor geometry and the matching

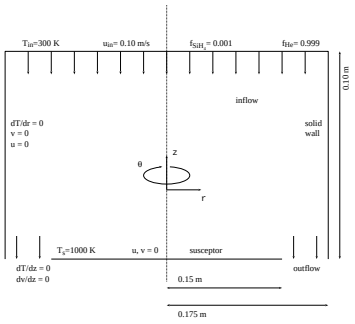


Fig. 5 Reactor geometry and boundary conditions

boundary conditions are shown in Figure 5. As computational domain we take, because of axisymmetry, one half of the (r-z) plane.

The pressure in the reactor is 1 atm. Through the inlet top plane of the reactor a gas mixture, consisting of 0.1 mole% silane diluted in helium, enters the reactor chamber with a uniform temperature $T_{\rm in} = 300$ K and a uniform velocity $u_{\rm in} = -0.10\frac{\rm m}{\rm s}$. At a distance of 0.1 m below the inlet a susceptor of diameter 0.3 m is placed with temperature T_s. The susceptor does not rotate. The outer walls of the reactor are adiabatic and do not rotate.

4.1 Mathematical Model of CVD

The composition of the N-component gas mixture is described in dimensionless mass fractions, which sum up to one. As initial condition we take a steady state velocity and temperature field of a pure helium flow in the reactor. These laminar flow and temperature fields are computed by respectively the continuity eq., the Navier-Stokes eqs. and the transport eq. for thermal energy, see [10] and [11]. At $t = 0$, an inlet mole fraction $f_{\rm SiH_2} = 0.001$ is fed into the reactor.

We assume that in the gas-phase K reversible reactions of the form

$$\sum_{i=1}^{N} \nu'_{ik} \mathcal{A}_i \underset{k^g_{k,\rm backward}}{\overset{k^g_{k,\rm forward}}{\rightleftarrows}} \sum_{i=1}^{N} \nu''_{ik} \mathcal{A}_i \tag{13}$$

take place. In (13) $\mathcal{A}_i$ are the species in the gas mixture, ν'_{ik} the forward stoichiometric coefficient for species i in reaction k, ν''_{ik} the backward stoichiometric coefficient for species i in the $k^{\rm th}$ reaction. The net stoichiometric coefficient is defined as $\nu_{ik} = \nu''_{ik} - \nu'_{ik}$. For the $k^{\rm th}$ reaction the *net* molar reaction rate R^g_k $\left(\frac{\rm mole}{m^3 \cdot s}\right)$ is defined as

$$R^g_k = k^g_{k,\rm forward} \prod_{i=1}^{N} \left(\frac{P\omega_i m}{RTm_i}\right)^{\nu'_{ik}} - k^g_{k,\rm backward} \prod_{i=1}^{N} \left(\frac{P\omega_i m}{RTm_i}\right)^{\nu''_{ik}}, \tag{14}$$

where the forward reaction rate constant $k^g_{k,\text{forward}}$ is fitted according to a modified Arrhenius expression

$$k^g_{k,\text{forward}}(T) = A_k T^{\beta_k} e^{\frac{-E_k}{RT}}, \tag{15}$$

where A_k, β_k and E_k are fit parameters. The backward reaction rate constants $k^g_{k,\text{backward}}$ are computed self-consistently from thermo-chemistry, according to

$$k^g_{\text{backward}}(T) = \frac{k^g_{\text{forward}}(T)}{K^g(T)} \left(\frac{RT}{P^0}\right)^{\sum_{i=1}^N \nu_{ik}}, \tag{16}$$

with $K^g(T)$ the reaction equilibrium constants. To facilitate easy reproduction of the solutions presented in the present paper, the reaction equilibrium constants are fitted to a modified Arrhenius expression

$$K^g(T) = A_{\text{eq}} T^{\beta_{\text{eq}}} e^{\frac{-E_{\text{eq}}}{RT}}, \tag{17}$$

with $A_{\text{eq}}, \beta_{\text{eq}}$ and E_{eq} are fit parameters.

4.2 CVD process 1

The first CVD process considered in this paper is a 6 species/5 gas phase reactions chemistry model, which was also considered in [18]. The 5 reactions are listed in Table 3 in which also the fit parameters of (15) and (17) are shown.

Table 3 Gas phase reaction mechanism and fit parameters for the 6 species/5 reactions model of Section 4.2. The parameters β_k and β_{eq} are dimensionless, while E_k and E_{eq} have unit $\frac{\text{kJ}}{\text{mol}}$. The dimensions of the parameters A_k and A_{eq} depend on the order of the reaction, but are expressed in terms of mol, m^3 and s.

Reaction	A_k	β_k	E_k	$A_{k,eq}$	$\beta_{k,eq}$	$E_{k,eq}$
$SiH_4 \rightleftarrows SiH_2 + H_2$	1.09×10^{25}	-3.37	256	6.85×10^{5}	0.48	235
$Si_2H_6 \rightleftarrows SiH_4 + SiH_2$	3.24×10^{29}	-4.24	243	1.96×10^{12}	-1.68	229
$Si_2H_6 \rightleftarrows H_2SiSiH_2 + H_2$	7.94×10^{15}	0	236	3.70×10^{7}	0	187
$SiH_2 + Si_2H_6 \rightleftarrows Si_3H_8$	1.81×10^{8}	0	0	1.36×10^{-12}	1.64	-233
$2SiH_2 \rightleftarrows H_2SiSiH_2$	1.81×10^{8}	0	0	2.00×10^{-7}	0	-272

4.3 CVD process 2

The other process considered in this paper is the well known 16 species/ 26 reactions chemistry model by [2]. In Table 4 the reactions and fit parameters are listed. The fit parameters are taken from [11], in which steady state benchmark solutions are presented for the present reactor geometry. Note

that the corresponding table in [11] contains a number of typographical and printing errors, which have been corrected the present table.

In our prior work [19] we presented the transient deposition rates for wafer temperatures varying from 900 K to 1100 K. In Figure 6, taken from [19], total transient deposition rates on the symmetry axis are presented for wafer temperatures in this range. In [19] it has been concluded that our steady state growth rates agree excellently with the ones computed with the software of Kleijn [11]; for all studied temperatures the steady state growth rates were found to differ less than 5% from each other.

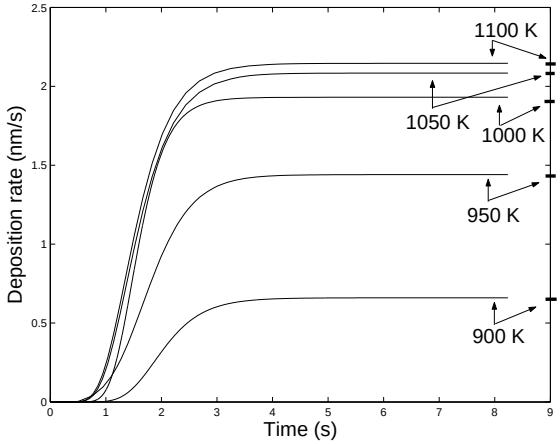


Fig. 6 Transient total deposition rates on the symmetry axis for wafer temperatures varying from 900 K up to 1100 K. On the right vertical axis: steady state total deposition rates obtained with Kleijn's steady state code [11].

5 Numerical Experiments

Numerical experiments are done for the two CVD processes mentioned in the previous section. Tests are done on three different grid sizes, namely, a grid with $nr = 35$ grid points in radial direction and $nz = 32$ grid points in axial direction, an $nr = 35$ by $nz = 47$ grid, and an $nr = 70$ by $nz = 82$ grid. In all grids the grid points in radial direction are equidistant. The grid points in axial direction have a gradually decreasing grid spacing towards the wafer surface. For the finest grid, i.e., the 70×82 grid, the axial distance from the wafer to the first grid point equals $1 \cdot 10^{-6}$ m, and thereafter the grid space gradually increases to $\Delta z = 5 \cdot 10^{-3}$ m for $z \geq 0.04$ m. For the coarsest grid the distance between the wafer and the first grid point is $2.5 \cdot 10^{-4}$ m. The 35×32 grid is illustrated in Figure 7.

Table 4 Fit parameters for the forward reaction rates (16) and gas phase equilibria constants (17) for the benchmark problem. The parameters β_k and β_{eq} are dimensionless, while E_k and E_{eq} have unit $\frac{\text{kJ}}{\text{mol}}$. The dimensions of the parameters A_k and A_{eq} depend on the order of the reaction, but are expressed in terms of mol, m^3 and s.

Reaction	A_k	β_k	E_k	$A_{k,eq}$	$\beta_{k,eq}$	$E_{k,eq}$
$SiH_4 \rightleftharpoons SiH_2 + H_2$	1.09×10^{25}	-3.37	256	6.85×10^{5}	0.48	235
$SiH_4 \rightleftharpoons SiH_3 + H$	3.69×10^{15}	0.0	390	1.45×10^{4}	0.90	382
$Si_2H_6 \rightleftharpoons SiH_4 + SiH_2$	3.24×10^{29}	-4.24	243	1.96×10^{12}	-1.68	229
$SiH_4 + H \rightleftharpoons SiH_3 + H_2$	1.46×10^{7}	0.0	10	1.75×10^{3}	-0.55	-50
$SiH_4 + SiH_3 \rightleftharpoons Si_2H_5 + H_2$	1.77×10^{6}	0.0	18	1.12×10^{-6}	2.09	-6
$SiH_4 + SiH \rightleftharpoons Si_2H_3 + H_2$	1.45×10^{6}	0.0	8	1.82×10^{-4}	1.65	21
$SiH_4 + SiH \rightleftharpoons Si_2H_5$	1.43×10^{7}	0.0	8	1.49×10^{-10}	1.56	-190
$SiH_2 \rightleftharpoons Si + H_2$	1.06×10^{14}	-0.88	189	1.23×10^{2}	0.97	180
$SiH_2 + H \rightleftharpoons SiH + H_2$	1.39×10^{7}	0.0	8	2.05×10^{1}	-0.51	-101
$SiH_2 + H \rightleftharpoons SiH_3$	3.81×10^{7}	0.0	8	2.56×10^{-3}	-1.03	-285
$SiH_2 + SiH_3 \rightleftharpoons Si_2H_5$	6.58×10^{6}	0.0	8	1.75×10^{-12}	1.60	-241
$SiH_2 + Si_2 \rightleftharpoons Si_3 + H_2$	3.55×10^{5}	0.0	8	5.95×10^{-6}	1.15	-225
$SiH_2 + Si_3 \rightleftharpoons Si_2H_2 + Si_2$	1.43×10^{5}	0.0	68	2.67×10^{0}	-0.18	59
$H_2SiSiH_2 \rightleftharpoons Si_2H_2 + H_2$	3.16×10^{14}	0.0	222	1.67×10^{6}	-0.37	112
$Si_2H_6 \rightleftharpoons H_3SiSiH + H_2$	7.94×10^{15}	0.0	236	1.17×10^{9}	-0.36	235
$H_2 + SiH \rightleftharpoons SiH_3$	3.45×10^{7}	0.0	8	1.42×10^{-4}	-0.52	-183
$H_2 + Si_2 \rightleftharpoons Si_2H_2$	1.54×10^{7}	0.0	8	7.47×10^{-6}	-0.37	-216
$H_2 + Si_2 \rightleftharpoons SiH + SiH$	1.54×10^{7}	0.0	168	1.65×10^{3}	-0.91	180
$H_2 + Si_3 \rightleftharpoons Si + Si_2H_2$	9.79×10^{6}	0.0	198	1.55×10^{2}	-0.55	189
$Si_2H_5 \rightleftharpoons Si_2H_3 + H_2$	3.16×10^{14}	0.0	222	1.14×10^{6}	0.08	210
$Si_2H_2 + H \rightleftharpoons Si_2H_3$	8.63×10^{8}	0.0	8	3.43×10^{-4}	-0.31	-149
$H + Si_2 \rightleftharpoons SiH + Si$	5.15×10^{7}	0.0	22	1.19×10^{3}	-0.88	29
$SiH_4 + H_3SiSiH \rightleftharpoons Si_3H_8$	6.02×10^{7}	0.0	0	7.97×10^{-16}	2.48	-233
$SiH_2 + Si_2H_6 \rightleftharpoons Si_3H_8$	1.81×10^{8}	0.0	0	1.36×10^{-12}	1.64	-233
$SiH_3 + Si_2H_5 \rightleftharpoons Si_3H_8$	3.31×10^{7}	0.0	0	1.06×10^{-14}	1.85	-318
$H_3SiSiH \rightleftharpoons H_2SiSiH_2$	1.15×10^{20}	-3.06	28	9.58×10^{-3}	0.50	-50

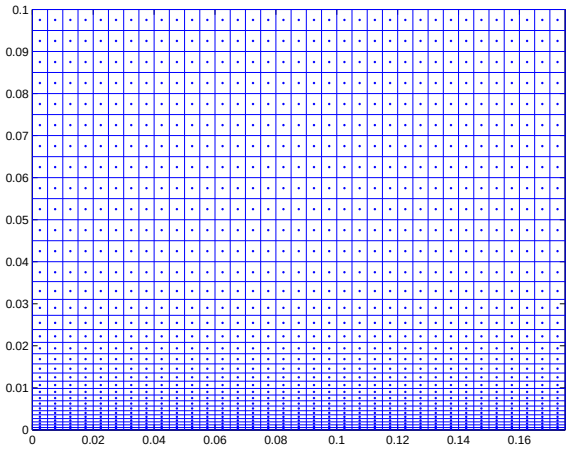


Fig. 7 Computational 35×32 grid. Species mass fractions are computed in the cell centers (represented as dots).

5.1 Experiments for the 6 species/5 reactions problem

We present results for time accurate transient simulations that start from the inflow conditions and run until the process is in steady state. For these simulations we allow the maximum number of time steps to be 1000, whereas the default maximum number of Newton iterations is 80. The latter seems to be rather large, but in our simulations we experienced that the strong non-linear reaction terms cause difficulties to find the correct search direction. Mostly, it is found within 10 Newton steps and converges then towards a solution. However, in the time frame right before steady state is reached, we experienced that finding the correct search direction might take some extra Newton steps. For simulations on the 35×32 grid the wafer temperature was set to 1000 K. In Table 5 relevant integration statistics are presented for the Euler Backward method, with a Globalized Inexact Newton (GIN) method, or Globalized Inexact Projected Newton (GIPN) method, equipped with the presented preconditioners. The number of function evaluations, by which we mean the evaluations of the right hand side in (12) with the reaction terms included, are for all preconditioners between 180 and 300, except for the block diagonal preconditioner; the number of function evaluations is in the order of 2300 for the GIN, and 280 for the GIPN. The number of time step rejections for the block diagonal preconditioner is due to multiple time step rejections caused by negative species concentration in the solution vector. As a result, the total computational costs are an order of magnitude higher for the GIN combined with a block diagonal preconditioner. The bold face numbers between brackets in Table 5 are the numbers belonging to the GIPN simulations for the cases where they differ from the simulations with GIN.

For the mid-size grid, 35×47 grid points, the behavior is found to be similar. In Table 6 the most relevant integration statistics are listed. The wafer temperature for the simulations on the 35×47 grid was set to 950 K. It has to be remarked that the different wafer temperature does not influence the behavior of the present computational method; it has been done to present different deposition rates for various temperatures to the reader. On the finest grid we found that for the block diagonal and lumped preconditioner the solution was not in steady state after 1000 time steps. For both preconditioners series of repeated time step rejections are obtained as a result of negative solution components. These preconditioners combined with the GIPN give multiple Newton divergence per time step, resulting in very small time step sizes. Therefore, we omit these results from Table 7. The wafer temperature for the simulations on the 70×82 grid was set to 900 K.

5.2 Experiments for the 16 *species/*26 *reactions system*

For time accurate simulations on the classical 16 species/26 reactions system, see also [19, 21], generally the same behavior in the integration statistics

Table 5 Number of operations for the 6 species/5 reactions problem with various preconditioners. Grid size: 35 × 32. Wafer temperature: 1000 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq. (4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Precond	ILU(0)		Lumped Jacobian	
Forcing term	1	2	1	2
Newton iter	113 (**102**)	106 (**100**)	175 (**149**)	143 (**137**)
Negative	1 (**0**)	1 (**0**)	2 (**0**)	2 (**0**)
Acc timestep	38 (**36**)	38 (**36**)	39 (**36**)	39 (**36**)
line search	13 (**12**)	14	24	22
lin iters	821 (**722**)	940 (**960**)	3,862 (**3,734**)	3,549 (**3,571**)
CPU time (sec)	180 (**170**)	175 (**170**)	300 (**260**)	250 (**240**)
Precond	block D-ILU		block diag	
Forcing term	1	2	1	2
Newton iter	86 (**81**)	86 (**80**)	165 (**161**)	144 (**148**)
Negative	1 (**0**)	1 (**0**)	0	1 (**0**)
Acc timestep	38 (**36**)	38 (**36**)	36	38 (**36**)
line search	9	9	24 (**20**)	19 (**22**)
lin iters	389 (**367**)	461 (**430**)	3,442 (**2,948**)	3,315 (**3,386**)
CPU time (sec)	150 (**140**)	152 (**140**)	300 (**290**)	270 (**260**)

Table 6 Number of operations for the 6 species/5 reactions problem with various preconditioners Grid size: 35×47. Wafer temperature: 950 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq. (4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Preconditioner	ILU(0)		Lumped Jacobian	
Forcing term	1	2	1	2
Newton iter	114	102	437 (**441**)	453 (**371**)
Negative	0	0	4 (**0**)	9 (**0**)
Acc time step	36	36	100 (**103**)	111 (**96**)
line search	18	15	15 (**17**)	18 (**15**)
lin iters	1,028 (**1,026**)	1,010	95,029 (**105,486**)	106,306 (**94,306**)
CPU time	270	250	1,700 (**1,800**)	1,875 (**1,600**)
Preconditioner	block D-ILU		block diag	
Forcing term	1	2	1	2
Newton iter	96 (**100**)	89	442 (**385**)	322 (**341**)
Negative	0	0	1 (**0**)	1 (**0**)
Acc time step	36	36	103 (**93**)	87 (**90**)
line search	6 (**7**)	6	27 (**13**)	12 (**17**)
lin iters	1,175 (**1,231**)	1,154 (**1,137**)	105,547 (**96,764**)	79,159 (**89,593**)
CPU time (sec)	250	230	1,900 (**1,700**)	1,400 (**1,500**)

Table 7 Number of operations for the 6 species/5 reactions problem with various preconditioners Grid size: 70×82. Wafer temperature: 900 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq.(4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Preconditioner	ILU(0)		block D-ILU	
Forcing term	1	2	1	2
Newton iter	102	89	102	91
Negative	0	0	0	0
line search	5	5	8	7
lin iters	1,108	1,206	909	1,003
CPU time (sec)	650	580	760	690

can be found. As in the previous section, we tested on three grid sizes, where for the finer grids the wafer temperature was set to lower temperatures. Remarkable is the consistent behavior of the block D-ILU preconditioner for all grid sizes. The number of function evaluations remains the same for all grids, and even more important for the CPU time, the number of Jacobian evaluations remains low for finer grids. For the finer grids the extra fill-in, with regard to ILU(0), for this preconditioner pays off with respect to the number of Bi-CGSTAB iterations.

The block diagonal preconditioner combined with forcing term 2 (4), which inverts the reaction terms per grid point, performs very badly if no projection is applied in the Newton method. Repeated negative species concentrations drive the time step size to such small values, that no steady state solution is obtained within the maximum allowed number of 1000 time steps, see Table 8 and 9. However, when GIPN is used, for all grids except the finest one, block diagonal performs much better, and might even compete with the block D-ILU in simulations with large number of species and fine (3D) grids.

Again, on the finest grid both the lumped Jacobian and the block diagonal preconditioner result in multiple Bi-CGSTAB and (projected) Newton divergence, such that the solution is not computed until steady state within 1000 time steps. For the 70×82 grid the grid cell sizes along the reacting surface are considerably smaller than for both other grids, causing a much stiffer problem. Consequently, the condition numbers of the Jacobian grow several orders of magnitude over the ones for simulations on coarser grids. The lumped Jacobian and block diagonal preconditioners are not capable to reduce these huge condition numbers to enable fast Bi-CGSTAB convergence, and thus (projected) Newton diverges. For this reason, we present only results for the ILU(0) and block D-ILU, see Table 10.

Table 8 Number of operations for the 16 species/26 reactions problem with various preconditioners. Grid size: 35×32. Wafer temperature: 1000 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq. (4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Preconditioner	ILU(0)		Lumped Jacobian	
Forcing term	1	2	1	2
Newton iter	108 (**101**)	101 (**94**)	152 (**149**)	149 (**127**)
Negative	1 (**0**)	1 (**0**)	0	2 (**0**)
Acc time step	38 (**36**)	38 (**36**)	37 (**36**)	41 (**36**)
line search	9 (**6**)	9 (**7**)	13 (**17**)	16
lin iters	848 (**825**)	1,129 (**1,009**)	4,987 (**4,654**)	7,927 (**5,819**)
CPU time (sec)	300 (**270**)	265 (**235**)	470	530 (**410**)
Preconditioner	block D-ILU		block diag	
Forcing term	1	2	1	2
Newton iter	111 (**97**)	104 (**93**)	149 (**133**)	1379 (**125**)
Negative	2 (**0**)	2 (**0**)	1 (**0**)	403 (**0**)
Acc time step	39 (**36**)	39 (**36**)	38 (**36**)	724 (**36**)
line search	5	6 (**4**)	10 (**15**)	7 (**16**)
lin iters	624 (**556**)	838 (**718**)	4,219 (**4,313**)	13,371 (**6,275**)
CPU time (sec)	330 (**290**)	320 (**270**)	460 (**410**)	3610 (**450**)

Table 9 Number of opertations for the 16 species/26 reactions problem with various preconditioners Grid size: 35 × 47. Wafer temperature: 950 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq. (4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Preconditioner	ILU(0)		Lumped Jacobian	
Forcing term	1	2	1	2
Newton iter	205 (**211**)	194 (**202**)	1,241 (**250**)	1,391 (**196**)
Negative	0	1 (**0**)	96 (**0**)	122 (**0**)
Acc time step	36	38 (**36**)	177 (**38**)	221 (**40**)
line search	54 (**51**)	49 (**61**)	44 (**41**)	32 (**33**)
lin iters	2,073 (**2,086**)	2,541 (**2,505**)	72,091 (**10,655**)	106,833 (**23,598**)
CPU time (sec)	630 (**650**)	610 (**640**)	4,600 (**1,080**)	6000 (**1,020**)
Preconditioner	block D-ILU		block diag	
Forcing term	1	2	1	2
Newton iter	153	148	285 (**290**)	2,054 (**223**)
Negative	0	0	2 (**0**)	583 (**0**)
Acc time step	36	36	43	1,000 (**43**)
line search	29	28	47 (**46**)	0 (**37**)
lin iters	772	859	22,498 (**25,605**)	28,140 (**29,253**)
CPU time (sec)	480	470	1,260 (**1,320**)	6,720 (**1,240**)

Table 10 Number of operations for the 16 species/26 reactions problem with various preconditioners Grid size: 70 × 82. Wafer temperature: 900 K. Forcing term 1 refers to Eq. (3); forcing term 2 refers to Eq. (4). The ordinary numbers are for GIN, whereas the bold face numbers are for GIPN.

Preconditioner	ILU(0)		block D-ILU	
Forcing term	1	2	1	2
Newton iter	353 (**385**)	395 (**351**)	325 (**308**)	299 (**306**)
Negative	0	3(**0**)	0	0
Acc time step	37	41 (**37**)	37	37
line search	127 (**145**)	136 (**128**)	101 (**95**)	101 (**96**)
lin iters	7,522 (**7,930**)	11,100 (**8,895**)	2,069 (**1,921**)	2,144 (**2,290**)
CPU time (sec)	5,900 (**4,875**)	5,420 (**6,000**)	5,300 (**4,280**)	4,175 (**4,350**)

5.3 Discussion on the integration statistics

It can be seen from the simulations with GIN that simulations on finer grids are more sensitive with respect to positivity, see the number of rejected time steps due to negativity. For the incomplete factorization based preconditioners this is not harmful, but for the lumped Jacobian and block diagonal preconditioners it is. However, when using GIPN instead of GIN, then the integration statistics are better for all methods, and in particular we see that the block diagonal preconditioner improves considerably.

Despite the fact that per Newton iteration the amount of flops to build the block D-ILU preconditioner and to solve a preconditioned system is the highest, it performs the best in terms of computational time. Besides the fewer number of linear iterations needed in the transient simulation, it also

needs fewer Jacobian evaluations. The latter is probably the most important factor to reduce the computational costs, in particular for finer grids and a large amount of species, e.g., compare the difference in integration statistics between the 6 species problem and the 16 species problem.

Another interesting issue is the role of the forcing term. It is not obvious to choose the one or the other. For sure is that forcing term 2, see Eq. (4), is in general tighter than forcing term 1, see Eq. (3), which sometimes results in negative species concentrations, see for instance Table 9. In particular the lumped Jacobian and the block diagonal preconditioner have many rejections due to negative concentrations. Combined with projected Newton it gives satisfying results, enabling us to overcome this undesired result. For the finer grids, forcing term 2 of Eq. (4) gives the best results.

A validation of the steady state solutions obtained has been done in [19], in which the steady solutions obtained by this code are compared with the steady state solutions of the steady state code developed by [11]. As far as we know, no other references on time accurate transient results are known on problems of the same kind of complexity.

6 Conclusions

In this paper we proposed a collection of preconditioners for transient laminar reacting gas flow simulations. Moreover, we proposed an extension of Newton's method to overcome difficulties with positivity. From the numerical experiments, which are performed for two CVD processes on various computational grids, it can be concluded that for most preconditioners tested in this paper, the total computational costs decrease when using projected Newton methods. Moreover, the usage of projected Newton methods increases the robustness of the Euler Backward solver. The incomplete factorization based preconditioners perform excellent in terms of efficiency. In particular, with the block D-ILU preconditioner, see Section 3.4, the fewest number of Jacobian evaluations and Bi-CGSTAB iterations are needed to compute a time accurate solution of both CVD processes presented in this paper. Moreover, for this preconditioner, combined with projected Newton methods, we measured the shortest CPU times. Taking all these arguments together, makes the block D-ILU the best preconditioner for type of problems.

Acknowledgements S. van Velhuizen is supported by the Delft Center of Computational Science and Engineering.

References

1. D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20:221–246, 1982.

2. M. E. Coltrin, R. J. Kee, and G. H. Evans. A mathematical model of the fluid mechanics and gas-phase chemistry in a rotating Chemical Vapor Deposition reactor. *J. Electrochem. Soc*, 136:819–829, 1989.
3. R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.
4. I. S. Duff and 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):889–901, 1999.
5. S. C. Eisenstat and H. F. Walker. Globally convergent Inexact Newton methods. *SIAM J. Optimization*, 4:393–422, 1994.
6. S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput.*, 17:16–32, 1996.
7. R. Hertz-Fischler. *A mathematical history of the golden number*. Dover Publications, Inc., Mineola, NY, 1998.
8. M. L. Hitchman and 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. *Transport phenomena in Chemical Vapor Deposition reactors*. PhD thesis, Delft University of Technology, Delft, 1991.
11. C. R. Kleijn. Computational modeling of transport phenomena and detailed chemistry in Chemical Vapor Deposition- A benchmark solution. *Thin Solid Films*, 365:294–306, 2000.
12. C. R. Kleijn, R. Dorsman, K. J. Kuijlaars, M. Okkerse, and H. van Santen. Multi-scale modeling of chemical vapor deposition processes for thin film technology. *J. Cryst. Growth*, 303:362–380, 2007.
13. D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193:357–397, 2004.
14. J. N. Shadid, R. S. Tuminaro, and H. F. Walker. An inexact Newton method for fully coupled solution of the Navier-Stokes equations with mass and heat transport. *J. Comput. Phys.*, 137:155–185, 1997.
15. J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 1980.
16. R. S. Tuminaro, H. F. Walker, and J. N. Shadid. On backtracking failure in Newton-GMRES methods with a demonstration for the Navier-Stokes equations. *J. Comput. Phys.*, 180:549–558, 2002.
17. H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.
18. S. van Veldhuizen, C. Vuik, and C. R. Kleijn. Numerical methods for reacting gas flow simulations. In V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, and J. Dongarra, editors, *Computational Science–ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006. Proceedings, Part II*, pages 10–17, Berlin, 2006. Springer. Lecture Notes in Computer Science 3992.
19. S. van Veldhuizen, C. Vuik, and C. R. Kleijn. Comparison of numerical methods for transient CVD simulations. *Surf. Coat. Technol.*, 201:8859–8862, 2007.
20. S. van Veldhuizen, C. Vuik, and C. R. Kleijn. Numerical methods for reacting gas flow simulations. *Internat. J. Multiscale Eng.*, 5:1–10, 2007.
21. S. van Veldhuizen, C. Vuik, and C. R. Kleijn. Comparison of ODE methods for for laminar reacting gas flow simulations. *Num. Meth. Part. Diff. Eq.*, 2008. DOI: 10.1002/num.20305.