PERFORMANCE OF ITERATIVE EQUATION SOLVERS FOR CONVECTION-DIFFUSION-ADSORPTION-PROBLEMS IN THREE-DIMENSIONAL SPHERE PACKINGS IN COMSOL

Birgit Stute[1], Wolfgang Joppich[2], Wolfgang Wiechert[1], Eric von Lieres[1]

[1]Research Center Jülich, Institute of Biotechnology 2,
Wilhelm-Johnen-Strasse 1, 52425 Jülich, Germany
[2]University of Applied Sciences Bonn-Rhein-Sieg, Department of Electrical Engineering,
Mechanical Engineering and Technical Journalism, Grantham-Allee 20,
53757 Sankt Augustin, Germany

e.von.lieres@fz-juelich.de (Eric von Lieres)

Abstract

Packed bed chromatography is commonly applied for the separation of large molecules in biopharmaceutical industry. A technical chromatography system is typically composed of a cylindrical column that is filled with porous spheres. Particularly in small columns, the impacts of inhomogeneous packing and wall effects on separation performance can be quite significant. We hence study convection, diffusion and adsorption in three-dimensional sphere packings. Random packings are externally generated and imported into COMSOL where the model equations are easy to implement. However, the COMSOL algorithms for automatic meshing and for iteratively solving the resulting equation systems fail to work with default settings. We have previously established a semi-automated and half-manual meshing procedure that works with the direct PARDISO solver. The present contribution addresses the evaluation and optimization of the iterative equation solvers that are provided by COMSOL for the given spatial geometry with up to six million degrees of freedom. The given results illustrate that we can iteratively solve systems with up to 600 instead of only 150 spheres using less memory and less computational time.

Keywords: sphere packing, convection-diffusion-adsorption, equation solver, COMSOL

Presenting Author's biography

Birgit Stute received her engineering diploma from the Bonn-Rhine-Sieg University of Applied Sciences in 2010. She is currently a technical assistant for modeling and simulation at the Research Center Jülich, Germany.

1 Introduction

Packed bed chromatography is commonly applied for the separation of large molecules in biopharmaceutical industry. A technical chromatography system is typically composed of a cylindrical column that is filled with porous spheres, also referred to as beads [1]. These systems are usually analyzed and optimized on small scales in order to safe valuable material. However, the impacts of inhomogeneous packing and wall effects on band spreading and consequently on separation performance are much more significant in small columns. We hence study convection, diffusion and adsorption in three-dimensional sphere packings.

Our model based studies are performed with COMSOL 3.5a on a PC with 16 compute cores and 64 GB of main memory. On this machine, COMSOL can handle up to 600 spheres. The random packings are externally generated with a self-written MATLAB code according to the algorithm by Mueller [2] and imported into COMSOL multiphysics [3]. The differential equations and boundary conditions for flow profile as well as for the convection-diffusion-adsorption problem on this geometry are easy to implement in COMSOL. However, the provided algorithms for automatic meshing and for iteratively solving the resulting equation systems fail to work with their default settings.

We have established a semi-automated and half-manual meshing procedure that works with the direct PARDISO solver, as reported in a previous publication [3]. However, the spatially discretized system has more than six million degrees of freedom, and in this region iterative solvers are usually more efficient. The present contribution is, hence, focused on enabling the application and on evaluating and optimizing the performance of the various iterative equation solvers that are provided by COMSOL for the given spatial geometry.

2 Theory

In this section we briefly introduce the general approach for solving multiphysics problems with COMSOL. The provided linear equation solvers and preconditioners are described in more detail and compared with respect to their commonalities and differences. The most important options and settings and their default settings in COMSOL are critically discussed in section 4.

2.1 COMSOL multiphysics

COMSOL multiphysics is a graphical simulation environment for solving models that are internally defined by partial differential equations (PDE) on spatially structured domains. The PDE are discretized in the space variables using the Finite Element (FE) method, and time variant problems are subsequently integrated with the implicit Backward Differentiation Formulas (BDF) method. Each time step yields a non-linear equation system that is linearized and iteratively solved by Newton´s method. A large fraction of the overall computational effort is spent for repeatedly solving these linear systems, because two or three iterative methods are effectively nested for solving the original PDE:

1) Spatial discretization using the FE method
2) Iterative time integration by the BDF method
3) Iterative solution of the resulting non-linear equations systems with a Newton method
4) Direct or iterative solution of the resulting linear equation systems

2.2 Iterative equation solvers

The speed and memory requirements of the used linear solver are most crucial for the efficiency of the entire model solution procedure. Direct equation solvers are good for small and medium sized problems, but cannot be used for large equation systems, due to memory restrictions. Moreover, iterative methods are usually more efficient for large scale computations. The following iterative solvers for linear equation systems are implemented in COMSOL:

1) Conjugate Gradient (CG)
2) Generalized Minimal Residual (GMRES)
3) Stabilized Bi-Conjugate Gradient (BiCGStab)
4) Geometric Multigrid (GM)

The first three solvers belong to the class of Krylov subspace methods, which develop the solution of a linear equation system $\mathbf{A} \cdot x = b$ into a sequence of approximations $x^{(k)}$. The residuals $r^{(k)}$ and a series of Krylov subspaces K_k for the system matrix $\mathbf{A}$ are defined as follows:

$$r^{(k)} = b - \mathbf{A} \cdot x^{(k)} \tag{1}$$

$$K_k\left(r^{(0)}, \mathbf{A}\right) = span\left\{r^{(0)}, \mathbf{A} \cdot r^{(0)}, \mathbf{A}^2 \cdot r^{(0)}, \ldots, \mathbf{A}^{k-1} \cdot r^{(0)}\right\} \tag{2}$$

All Krylov subspace methods have in common that the differences between the approximations $x^{(k)}$ and the initial approximation $x^{(0)}$ are in the corresponding Krylov subspace K_k:

$$x^{(k)} - x^{(0)} \in K_k\left(r^{(0)}, A\right) \tag{3}$$

Moreover, in all Krylov subspace methods the k^{th} residuum $r^{(k)}$ is orthogonal to a k-dimensional subspace Γ_k of $\mathbf{R}^n$, where n is the number of unknowns of the linear equation system:

$$r^{(k)} \perp \Gamma_k \tag{4}$$

Hence, the dimension of the residual $r^{(k)}$ will be reduced by one in every iteration, and consequently the exact solution is found in the n^{th} iteration at least.

The discussed Krylov subspace methods differ in the choice of the subspaces and in the iterative definition of the approximations. Each method is designed to exploit specific features of the equation systems. The convergence of Krylov subspaces methods generally depends on the numerical condition and on the eigenvalues of the system matrix **A**.

2.2.1 Conjugate Gradient (CG)

The CG method is a Krylov subspace method with a so called Galerkin condition (Eq. 5). The k^{th} residuum $r^{(k)}$ is orthogonal to the k-dimensional Krylov subspace K_k of the matrix **A**:

$$r^{(k)} \perp \Gamma_k = K_k\left(r^{(0)}, \mathbf{A}\right) \tag{5}$$

The CG method solves the linear equation system $\mathbf{A}\cdot x = b$ by minimizing a functional $F(x)$:

$$F\left(x\right) = \frac{1}{2}\left\langle x, \mathbf{A}\cdot x\right\rangle - \left\langle x, b\right\rangle \tag{6}$$

This functional can be geometrically interpreted as n+1-dimensional paraboloid. The functional F is in each iteration minimized along a search direction $p^{(k)}$ starting from the previous approximation $x^{(k)}$.

$$x^{(k+1)} = x^{(k)} + s_k \cdot p^{(k)} \tag{7}$$

Here, s_k is a factor that minimizes the functional F in one dimension, and the directions $p^{(k)}$ are conjugated vectors:

$$\left\langle p^{(k+1)}, \mathbf{A}\cdot p^{(k)}\right\rangle = 0 \tag{8}$$

The k^{th} approximation $x^{(k)}$ minimizes the functional F within the subspace that is spanned by all search directions, due to their conjugateness.

The mathematical theory of the CG method is only valid for symmetric and positive definite system matrices **A** [4]. However, the method often converges when applied to more general linear equation systems. The required memory and the computational effort remain constant for all iterations.

2.2.2 Generalized Minimal Residual (GMRES)

The GMRES method also solves the linear equation system by minimizing a functional, namely the Euclidean norm of the residuum:

$$J\left(x\right) = \left\|\mathbf{A}\cdot x - b\right\|_2^2 \tag{9}$$

The functional J is minimized not only along the latest search direction, but also along all previous directions:

$$x^{(k+1)} = x^{(k)} + \sum_{i=1}^{k} s_i \cdot p^{(i)} \tag{10}$$

In the GMRES method the search directions $p^{(k)}$ are orthogonal scaled bases of the Krylov subspace K_k. In positive contrast to the CG method, the mathematical theory of the GMRES method is valid for all regular system matrices **A** [5]. In negative contrast to the CG method, the memory requirement and the computational effort of the GMRES method increase from iteration to iteration. This is mainly because all search directions $p^{(k)}$ are saved and reconsidered in each of the following iterations.

The memory requirements and the computational effort can be reduced with the GMRES[m] method by restarting the algorithm after m iterations with the current approximation as initial value. As the CG and GMRES methods, the GMRES[m] method yields a sequence of monotone falling residuals. However, convergence is not guaranteed.

2.2.3 Stabilized Bi-Conjugate Gradient (BiCGStab)

The Bi-Conjugate Gradient (BiCG) method replaces the Galerkin condition in the CG method with an oblique projection method due to which the k^{th} residuum $r^{(k)}$ is orthogonal to the k-dimensional Krylov subspace of the transposed of **A**.

$$r^{(k)} \perp \Gamma_k = K_k(r^{(0)}, \mathbf{A}^T) \tag{11}$$

As in the CG method, the required memory and the computational effort remain constant for all iterations, and as in the GMRES method, the mathematical theory of the BiCG method is valid for all regular matrices **A** [6].

In negative contrast to the CG and GMRES methods, the sequence of residuals is not monotonously decreasing, because the iterated approximations are not determined by minimizing a functional F or J, but computed on the basis of two residuals and two search directions. These residuals are constructed in the BiCG method such as to form a set of bi-orthogonal bases of the Krylov subspaces of **A** and $\mathbf{A}^T$.

The Stabilized Bi-Conjugate Gradient (BiCGStab) improves the convergence of the BiCG method, and also avoids possible breakdowns by restarting the method with the current approximation as initial value [6].

2.2.4 Geometric Multigrid (GM)

The GM method iteratively solves linear equation systems with a geometric reference to the spatial grid of the FE method. The equations are solved by recursively applying a sequence of two-grid schemes, each of which is composed of two sequential steps:

1) Smoothing, which eliminates error fractions of high frequency
2) Coarse grid correction, which removes error fractions of low frequency

The smoothing step performs several iterations of a conventional relaxation scheme, and the coarse grid correction step projects the current residual on a coarser grid, where the residual equation is solved for the coarse grid error. The estimated error is then transferred back to the finer grid, and used for improving the previous solution. Different cycle types can be chosen for the recursive application of the two grid schemes, and the V, W and F cycles are implemented in COMSOL.

In contrast to the discussed Krylov subspace methods, the GM method yields good convergence rates that are independent from the spatial discretization. Hence, this method is particularly suitable for fine meshes, provided that high quality meshes are also available for medium and coarse spatial discretizations.

2.3 Preconditioning

Preconditioning is usually applied in order to improve the convergence rate of the Krylov subspace methods. Preconditioning changes important characteristics of the system matrix **A** such as the numerical condition and the eigenvalue spectrum. This is achieved by multiplying the original equation system with a preconditioning matrix $\mathbf{P}^{-1}$ from the left or from the right hand side:

$$\mathbf{P}^{-1} \cdot \mathbf{A} \cdot x = \mathbf{P}^{-1} \cdot b \qquad \text{(left hand side)} \quad (12)$$

$$\mathbf{A} \cdot \mathbf{P}^{-1} \cdot y = b \quad \text{with} \quad y = \mathbf{P} \cdot x \qquad \text{(right hand side)} \quad (13)$$

The preconditioned equation systems have the same solution as the original system but better numerical properties and consequently the solvers converge faster.

The inverse $\mathbf{P}^{-1}$ of a preconditioner matrix **P** is often not efficient to compute. Consequently, the matrix products $\mathbf{P}^{-1}\cdot\mathbf{A}$ or $\mathbf{A}\cdot\mathbf{P}^{-1}$ are usually not explicitly formed, but instead linear equation systems with the system matrix **P** are solved. These solutions must of course be much cheaper to compute than the original equation system with matrix **A**. At the same time, the preconditioner matrix **P** should be similar to the original system matrix **A**. There are various preconditioning methods with different cost to benefit ratios. Symmetric preconditioning is also possible in order to preserve theoretical requirements of the CG method.

2.3.1 Jacobi and SSOR

The Jacobi preconditioner matrix $\mathbf{P}_{jac}$ contains only the diagonal elements of the original system matrix **A**. This preconditioner is an exception to the rule, because $\mathbf{P}^{-1}$ is cheap to compute. However, $\mathbf{P}_{jac}$ is usually a poor approximation to **A**, and the numerical properties of the preconditioned system are consequently not much improved.

The system matrix of the symmetric successive over-relaxation (SSOR) method is a better approximation of **A**. This matrix can also be used for preconditioning at the cost of solving an additional linear equation system at each evaluation of the preconditioned system.

The Jacobi and SSOR preconditioners in COMSOL are process matrices of the corresponding relaxation schemes. The same iteration schemes can also be used for smoothing within the GM method. However, the optimal relaxation factor can be different for preconditioning and smoothing even when solving the same PDE.

2.3.2 Incomplete LU factorization (ILU)

The complete LU factorization is a direct solution method that factorizes the original system **A** into an upper triangular matrix **U** and a lower triangular matrix **L**. The ILU preconditioner can be adjusted to specific demands of the applied iterative equation solver. Non-zero elements of the matrices **L** and **U** are dropped in the elimination phase of the LU factorization according to specific rules, for example the *fill ratio drop* rule or the *threshold drop* rule. Both rules retain the largest absolute vales in the columns of the matrices **L** and **U**. Diagonal elements are never dropped.

In the *fill ratio* rule, a pre-specified *fill ratio* factor times the number of non-zero elements in each column of **A** controls the number of non-zero elements in the respective columns of **L** and **U**. In the *threshold drop* rule, all elements with absolute values below a pre-specified *drop tolerance* factor times the Euclidean norm of the corresponding column are dropped to zero.

COMSOL allows to specify a desired preconditioning quality and to iterate the ILU preconditioning matrix. The preconditioner matrix can also be divided by a specified relaxation factor. In addition to the choice of a drop rule and of a relaxation parameter, COMSOL allows to tune the ILU preconditioner by *threshold pivoting* and by an *respect pattern* option. *Threshold pivoting* exchanges rows of **A** in order to avoid zero elements on the diagonal and to improve the diagonal dominance. The option *respect pattern* limits both drop rules by restricting the dropping of elements in the matrices **L** and **U** to positions where **A** has zero entries.

The required memory, the computational effort, but also the quality of the ILU preconditioner decrease with an increasing number of zero elements in the factorization. Hence, a well adjusted preconditioner should balance the computational resources between the iterative solution of the preconditioned system and the preconditioning itself.

2.3.3 Vanka

The Vanka preconditioner was developed in particular for saddle point problems that can occur in the spatial

discretization of PDE with the FE method. The resulting linear equation systems in the time integration are underdetermined and zeros elements appear on the diagonal of **A**.

The Vanka preconditioner is composed of the process matrices of the SSOR method and of a modified symmetric Gauß-Seidel method [7]. The latter blocks the unknowns of the linear equation system by physical proximity, and COMSOL allows to specify variables that are to be used as origin for building these blocks.

Vanka yields very good results as preconditioner and also as smoother for the GM method in solving the Navier-Stokes equation. The blocked versions of the Vanka and SSOR preconditioners are available for parallel computation with COMSOL.

3 Benchmark examples

Four benchmark examples with increasing complexity and 20, 50, 150 and 600 packed spheres are defined for successively evaluating and tuning the solver performance (see Fig. 1). The velocity profile of the buffer solvent as well as convection, diffusion, and adsorption of solute molecules are computed with COMSOL 3.5a for each of these geometries.

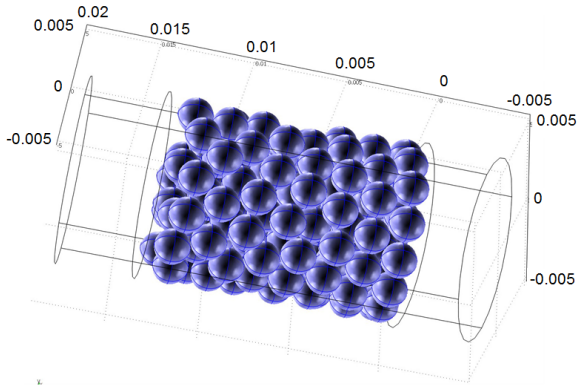

Fig. 1 Irregular packing of 150 spherical beads in a cylindrical tube

3.1 Mathematical equations

The Navier-Stokes equations for incompressible flow are defined in the interstitial volume between the spheres ($\vec{u}$: velocity, p: pressure, η: dynamic viscosity, ρ: fluid density):

$$\frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho} \cdot \nabla p + \eta \cdot \Delta \vec{u} \qquad (14)$$

$$\nabla \cdot \vec{u} = 0 \qquad (15)$$

A convection-diffusion equation is defined on the same domain (c: molecule concentration, D: diffusion coefficient):

$$\frac{\partial c}{\partial t} = -\vec{u} \cdot \nabla c + D \cdot \Delta c \qquad (16)$$

Within the porous beads, Fickian diffusion is the only transport mechanism. In this domain, adsorption to the local pore walls also takes place (c_p and q: molecule concentrations in mobile and adsorbed phase, D_e: effective diffusivity, ε: porosity, k_a and k_d: adsorption and desorption rates, q_{max} maximum capacity):

$$\frac{\partial c_p}{\partial t} = D_e \cdot \Delta c_p - \frac{1-\varepsilon}{\varepsilon} \cdot \frac{\partial q}{\partial t} \qquad (17)$$

$$\frac{\partial q}{\partial t} = k_a \cdot c_p \cdot \left(q_{\max} - q\right) - k_d \cdot q \qquad (18)$$

A Neumann boundary condition is imposed at the interface of the beads and the interstitial volume ($\vec{n}$: outer normal vector, k_f: film diffusion coefficient):

$$\vec{n} \cdot (-D \cdot \nabla c + c \cdot \vec{u}) = k_f \cdot (c - c_p) \qquad (19)$$

The parameter values of the system equations and boundary conditions are defined in Tab. 1.

Tab. 1 Parameter values

Parameter	Description
u_0 = 2.13e-4 m/s	Normal inflow velocity
ρ = 1.00e3 kg/m^3	Density
η = 9.50e-4 Pa·s	Dynamic viscosity
c_0 = 7.14e-3 mol/m^3	Inlet concentration
D = 5.75e-8 m^2/s	Diffusion coefficient (fluid)
D_e = 5.75e-8 m^2/s	Diffusion coefficient (pore)
ε = 0.75	Sphere porosity
k_a = 1.14 m^3/mol/s	Adsorption coefficient
k_d = 2.00e-3 1/s	Desorption coefficient
k_f = 6.90e-4 m/s	Film diffusion coefficient

In all benchmarks models, the entire column is 2.2 cm long, including two void regions, each of which is 0.42 cm long. The column diameter is 1 cm, and the spheres diameters depend on the number of beads in the column: 1.07e-8 m for 20 beads, 4.46e-9 m for 50 beads, 1.65e-9 m for 150 beads, and 4.18e-10 m for 600 beads.

3.2 Solution procedure

The velocity profile can be considered time invariant for the studied solvent and solute properties. Hence, the Navier-Stokes equations for incompressible flow (Eq. 14-15) are first solved with a stationary solver,

and the resulting profile is stored. The coupled equations for the time variant convection, diffusion and adsorption processes (Eq. 16-19) are then computed on the stored velocity profile.

3 COMSOL options and settings

COMSOL allows to control and tune the implemented algorithms with specific options and settings. The default values are not optimal for all problems, and the impact of changes is not always easy to anticipate.

3.1 Relative tolerance

A *relative tolerance* can be specified for the iterative linear equation solvers. However, for time variant problems this setting is effective only when set larger than the *relative tolerance* of the time integrator. Otherwise the latter is also used for the linear equation solver. Moreover, the specification of a *relative tolerance* for the linear equation solver is overridden in nonlinear stationary problems by the *relative tolerance* of Newton's method. The default *relative tolerance* of the iterative linear equation system solvers, 1e-6, is four orders of magnitude larger than the default *relative tolerance* of the time integrator, 1e-2.

3.2 Factor in error estimate

Another important setting of the linear equation system solvers in COMSOL is denoted *factor in error estimate*. According to the COMSOL online documentation [7] this factor is used in place of the numerical condition number $\kappa(\mathbf{A})$ of the system matrix **A** for estimating the relative approximation error on basis of the residual *r*:

$$\frac{\|\bar{x}-x\|}{\|x\|} \le \kappa(\mathbf{A}).\frac{\|r\|}{\|b\|} \tag{20}$$

Here, *x* and $\bar{x}$ denote the exact and the approximated solution of the linear equation system. In practice, equation 21 allows to increase the relative tolerance of the solution of the linear equation system by increasing the factor above the actual condition of the matrix **A**. Conversely, the computational effort for each solution of the linear equation system increases with the *factor in error estimate*, and the default value, 400, can often be decreased by several orders of magnitude without any visible effect on the solution of the PDE. However, insufficient accuracy of the linear equation solver can cause problems with the convergence of the time integrator. Hence, an optimal choice of the *factor in error estimate* is intricate and must usually be based on a careful observation of the convergence rates not only of the linear equation solver but also of the time integrator and Newton solver.

3.3 Maximum iterations

COMSOL stops the entire solution procedure with an error message when the specified *maximum iterations* are exceeded. The default value, 1e3, is sufficient for most problems. However, the number of iterations of the Krylov subspace methods increases with the number of unknowns of the linear equation system, *n*, and hence with the number of elements in the space discretization of the PDE.

The GMRES[*m*] method additionally allows to specify the *number of iterations before restart*. The default value is $m = 50$. The memory requirement increases with *m*, and for values above *maximum iterations* the GMRES method is effectively used.

3.4 Preconditioning

The Krylov subspace methods require to declare the preconditioning method and direction. The CG method is always symmetrically preconditioned, and the preconditioning direction only affects the accuracy check of the linear solver. In linear stationary problems, the accuracy for right hand side preconditioning is checked according to equation 22 (ρ: factor in error estimate, *tol*: relative tolerance of the linear solver):

$$\rho \cdot \|b - \mathbf{A} \cdot x\| < tol \cdot \|b\| \tag{21}$$

For left hand side preconditioning the preconditioner matrix **P** occurs in the accuracy check:

$$\rho \cdot \left\|\mathbf{P}^{-1} \cdot b - \mathbf{A} \cdot x\right\| < tol \cdot \left\|\mathbf{P}^{-1} \cdot b\right\| \tag{22}$$

In nonlinear stationary or in time variant problems the above estimations are modified by several factors which for example depend on the convergence of the Newton solver. The best preconditioning direction depends on the linear solver method and also on the PDE and their space discretization. For example, preconditioning from the right hand side causes less computational effort for the BiCGStab method.

The preconditioners in COMSOL have two common settings, the *relaxation parameter* and the *number of iterations* (see section 2). For the studied convection-diffusion-adsorption problem, the Jacobi and SSOR preconditioners were observed to work best with relaxation parameters of 0.3 and 0.7, respectively. The performance of the ILU preconditioner showed little dependency on the relaxation parameter.

The *number of iterations* setting specifies how often the preconditioner is iteratively reapplied. More iterations enhance the convergence of the linear equation solver, but on the cost of longer calculation times. For the studied benchmarks, the Jacobi preconditioner was most efficient with 2 to 5 iterations. For all other discussed preconditioners more than 3 iterations did not significantly improve the convergence of the linear solver.

Several additional options and settings of the ILU preconditioner have been introduced in section 2. The *drop rule*, the *drop tolerance* factor or the *fill ratio* factor, and the *respect pattern* option were found to be most important for efficiently solving the convection-diffusion-adsorption problem.

The default drop rule is *threshold drop* with a *tolerance* of 0.01. The latter can be numerically entered or adjusted with a slider between 0.2 and 1e-6. Positive numbers can be specified for the factor of the *fill ratio* rule, and 2 is the default value. Smaller *drop tolerances* or larger *fill ratios* yield more accurate LU factorizations and vice versa. The *respect pattern* option decreases the number of dropped elements, and consequently increases the accuracy but also the computational effort of the LU factorization.

The *variables* setting is most important for tuning the performance of the Vanka preconditioner/ smoother. This setting specifies which variables are used for building the blocks of the modified Gauß-Seidel method, and the remaining unknowns are preconditioned by SSOR. For the Navier-Stokes equations the pressure *p* has been observed to be a good choice for the *variables* setting.

4 Benchmark Results

Iterative solvers are more memory efficient and allow to import, mesh and solve both the Navier-Stokes equations and the convection-diffusion-adsorption problem in COMSOL not only faster but also for significantly more spheres than direct solvers. However, adequate preconditioning of the underlying linear equation systems that are solved in each time step of the differential equation solver is crucial. Moreover, the default values of some algorithm settings in COMSOL had to be changed by up to several orders of magnitude in order to gain satisfactory numerical speed.

4.1 Navier Stokes equations

Tab. 2 shows the memory usage and runtimes of different linear solvers with optimized settings for the Navier-Stokes equations. The benchmark model with 50 spheres was solved on a fine mesh with 694.284 degrees of freedom (DoF). PARDISO is a direct solver, and Vanka with the pressure *p* specified as *variables* was found to be the most effective preconditioner/smoother for all applied iterative solvers used, namely GMRES, BiCGStab and GM.

Good performance of all iterative solvers was observed with a *factor in error estimate* of 1. Optimal values of the *relaxations parameter* and of the *number of iterations* were also found to be 1 for the modified Gauß-Seidel and for the SSOR preconditioning in all Krylov subspace methods, namely GMRES and BiCGStab. However, in the GM method the Vanka smoother performed best with a *relaxation parameter* of 0.7.

The GM method was applied with two meshes only, due to quality limitations in the coarse meshes that were generated by COMSOL. Two pre-smoother and one post-smoother steps were performed in a V-cycle, and the convergence of the GM method was found to strongly depend on the mesh quality. Unfortunately, both the creation of coarse meshes, and the refinement of coarse meshes into intermediate meshes with common points, which is favorable for the GM method, yields poor mesh qualities in COMSOL. The results in Tab. 2 were computed with a coarse mesh with minimum element quality of 0.20, and a finer mesh with minimum element quality of 0.25, that has also been used with other linear solvers. The results could not be improved with the GM method using three or more grid levels.

Tab. 2 Memory usage and runtimes of different linear solvers for the Navier-Stokes equations and the 50 sphere benchmark

Solver	Memory [GB]	Runtime [s]
PARDISO	8,9	352
GM	6,5	986
GMRES	5,7	1332
BiCGStab	4,0	1670

In Tab. 2 the solvers are arranged by decreasing memory usage and by increasing computational time. The results perfectly agree with theory in that direct solvers outperform iterative solvers for small and medium sized equation systems at the cost of excessive memory usage. Consequently, the 600 sphere benchmark could not be computed on a machine with 64 GB of physical memory with the direct PARDISO solver, but with the GMRES method.

Tab. 4 shows the runtimes for the model with 600 randomly packed spheres. The Navier-Stokes equations with 6.412.071 DoF were solved with individually optimized GMRES options and settings.

4.2 Convection-diffusion-adsorption equations

In contrast to the Navier-Stokes equations, the direct PARDISO solver was outperformed by the iterative solvers for the convection-diffusion-adsorption equations. The BiCGStab, GMRES and GM methods with optimized options and settings (see Tab. 3) are faster and more memory efficient than the direct solver for 200.000 and more DoF (see Fig. 2 and Fig. 3). The CG method converges, though in theory only applicable to equation systems with symmetric matrices which is not given here. However, good

convergence is only achieved with comparably short steps below 0.01 s of the time integrator, which result in non-competitive computation times for the PDE solution. As already discussed in the previous section, the GM method can be efficiently used in COMSOL with two meshes only. This limits the performance of the GM method, as can be seen from Fig. 2.

Tab. 3 Optimized solver settings for the convection-diffusion-adsorption equations

Solver Precond./ Smooth.	Relaxation factor	Number of iterations
PARDISO	-	-
GMRES SSOR	0.7	1
GMRES SSOR blocked	0.7	1
GMRES ILU	0.7	1
BiCGStab SSOR blocked	0.7	1
GM SSOR	0.7 - 0.7	2 - 1

In addition to Tab. 3, all solutions of the convection-diffusion-adsorption equations were computed with a *relative tolerance* of the time integrator of 0.01, a relative tolerance of the linear solver of 1e-6, and a *factor in error estimate* of 1. All Krylov subspace methods were preconditioned from the right, and the ILU preconditioner was applied with a *drop tolerance* of 0.1 and a *pivot threshold* of 0.5. The *respect pattern* option was disabled.

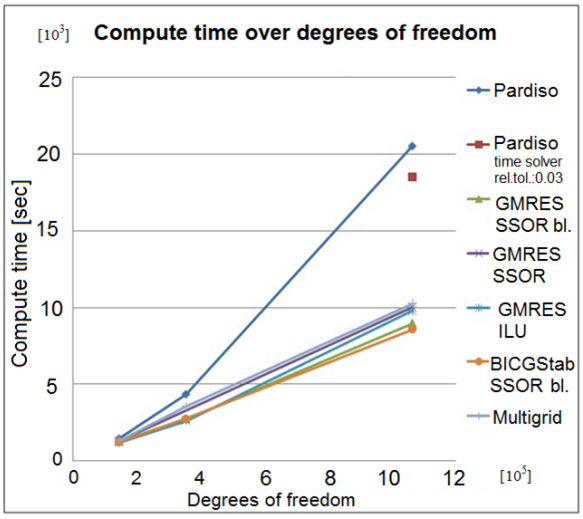

Fig. 2 Compute time over degrees of freedom for different linear solvers and preconditioners

In theory, the computational effort for solving a linear equation system with N degrees of freedom increases with N^2 for direct solvers but better for Krylov subspace methods. Nevertheless, the above statements are only valid in an asymptotic sense, and direct solvers can be more efficient for systems of small and medium size. Moreover, the performance of the iterative solvers varies between the different methods and strongly depends on the applied preconditioner.

Fig. 2 validates the discussed trends with runtimes for the convection-diffusion-adsorption equations and irregular packings of 20, 50 and 150 spheres. Fig. 4 illustrates that the iterative solvers are superior to the direct PARDISO solver also with respect to memory consumption.

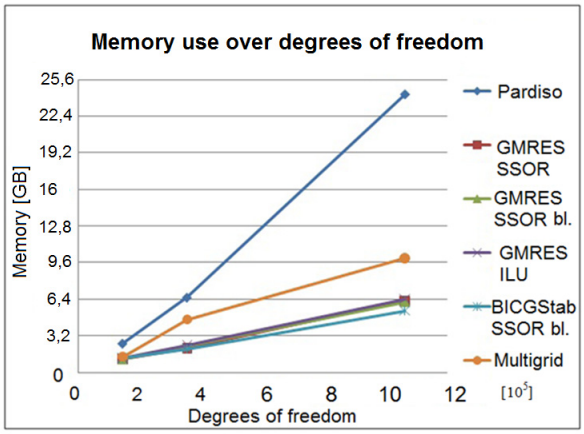

Fig. 3 Memory usage over degrees of freedom for different linear solvers and preconditioners

Fig. 2 and Fig. 3 reveal that the BiCGStab method outperforms the GMRES method with an optimized SSOR preconditioner. The observed discrepancies in computation times and memory usage can be reduced, but not eliminated by using alternative preconditioners for the GMRES method. This trend is expected to continue for increasing degrees of freedom, due to increasing numbers of required iterations for all Krylov subspace methods. However, in contrast to the BiCGStab method, not only the computation time but also the memory usage of the GMRES method is known to increase with the number of iterations.

Tab. 4 shows the runtimes for the model with 600 randomly packed spheres. The convection-diffusion-adsorption equations with 3.303.518 DoF were solved with individually optimized GMRES options and settings.

Tab. 4 Number of iterations and runtimes for the 600 sphere benchmark with GMRES

Equations	Iterations	Runtime [h]
Navier-Stokes	656	5:58
Conv.-Diff.-Ads.	3986	11:56

Fig. 4 shows a typical breakthrough curve for the 600 sphere benchmark at the outlet of the column. The steepness and shape of the breakthrough curve is strongly determined by local inhomogeneities of the packing density that appear particularly at the column walls.

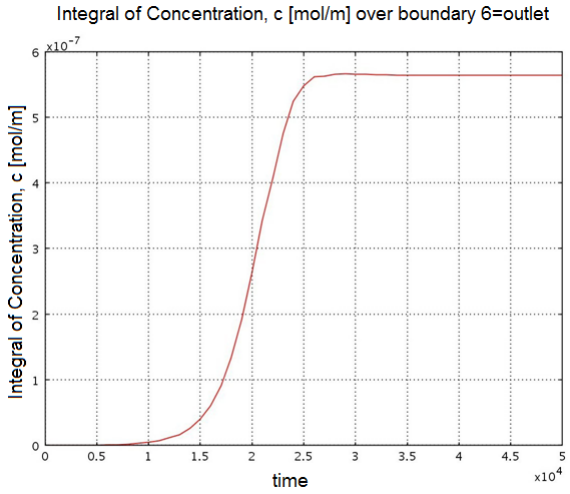

Fig. 4 Integral of outlet concentration over column cross section as function of time

5 Conclusions

The given results illustrate that we can iteratively solve systems with up to 600 instead of only 150 spheres using less memory and less computational time than the direct PARDISO solver. The GMRES method and the BiCGStab method were observed to perform significantly better than the other iterative solvers that are implemented in COMSOL when adequately preconditioned. Most crucially, the default values of the provided preconditioners had to be changed by several orders of magnitude in order to achieve satisfactory convergence rates. For instance, the *factor in error estimate* had to be reduced from 400 to 1.

Spatially resolved simulations of packed bed chromatography are important for analyzing and optimizing the performance of micro-columns with up to 10^5 beads. Systems with more beads can be spatially homogenized and described by the well known general rate model [1]. COMSOL will most certainly not allow to close the gap between 600 and 10^5 beads, but the presented results indicate that simulations with some 10^3 beads are possible on machines with huge amounts of physical memory. The number of beads is currently limited by the user interface of version 3.5a, which appear to be resolved in version 4.0.

6 References

[1] G. Guiochon, D. Shirazi, und A. Felinger. Fundamentals of Preparative and nonlinear chromatography. Elsevier Academic Press. 2006.

[2] G. Mueller. Numerically packing spheres in cylinders. *Powder Technology*, 159:105-110, 2005.

[3] S. Schnittert, R. Winz, and E. von Lieres. Development of a 3D model for packed bed liquid chromatography in micro-columns. In D. Al-Dabass, S. Katsikas, I. Koukos, A. Abraham, R. Zobel, editors, *Proceedings of UKSim 3rd European Symposium on Computer Modeling and Simulation*, pages 193-197, Athens, November 25-27 2009. IEEE Computer Society, Los Alamitos.

[4] J. R. Shewchuk. An introduction to the conjugate gradient method. Without the agonizing pain. Carnegie Mellon University. 1994

[5] Y. Saad, and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7: 856-869. 1986.

[6] 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: 631-644, 1992.

[7] COMSOL. COMSOL Multiphysics User Guide. Version 3.5a. COMSOL AB, Stockholm, 2008.