SCARC is ready for use in FDS

Dr. Susanne Kilian

hhpberlin, Ingenieure für Brandschutz GmbH, D-10245 Berlin

s.kilian@hhpberlin.de

Abstract

The parallel computation of FDS multi-mesh scenarios is closely linked to the efficient solution of the underlying pressure equation. The default FDS pressure solver is essentially based on the use of local discretizations for the individual sub-meshes of the computational domain. Thereby the global pressure solution is composed of respective local solutions which are obtained from the mesh-wise execution of highly optimized *Fast Fourier Transformations* (FFT) in each time step. Due to the lack of a global discretization and the restriction of FFT to structured grids, this approach can cause the normal components of velocity to differ at adjacent mesh interfaces and to penetrate into immersed obstructions. A possible remedy for the reduction of these velocity errors is to solve the pressure equation multiple times within each individual time step. Although this *pressure iteration* works extremely well for a wide variety of cases, it can lead to considerable effort in special situations.

As alternative approach the solver package **Sca***lable* **R***ecursive* **C***lustering* (SCARC) was developed, using iterative solution strategies based on *Krylov* and *Multigrid* techniques in combination with various *Schwarz preconditioners*. Since SCARC fundamentally relies on a global discretization, there are no more errors related to mesh interfaces. Furthermore, internal obstructions can be addressed through both a structured and an unstructured discretization. To prevent penetration into internal obstructions, an additional pressure iteration for velocity correction may also be required for the structured variant, but on the other hand it still allows the use of the optimized local FFT-solvers. In contrast, the unstructured variant offers the correct treatment of internal obstructions without the need for auxiliary corrections, but the local FFT's can no longer be applied and are replaced by optimized local *LU-decompositions* which however turn out to be less efficient than the local FFT's.

This article gives a basic overview of the underlying concepts and the current state of development accompanied by some illustrative comparisons of the different variants with each other and with the default pressure solver.

1 MOTIVATION

The solution of elliptic partial differential equations (PDEs) is an integral part of the numerical solution of physical processes. Research into the development of numerical solvers for elliptic problems has a very long history and has produced a number of highly powerful approaches which, however, were originally tailored for purely serial applications of mostly small or moderate size. In view of the immense magnitude and complexity of today's CFD-problems, the efficient design of suitable parallel algorithms which fully exploit the current (super)computing power of modern high performance architectures seems inevitable and is a particularly great challenge.

But especially for elliptic problems, it is extremely difficult to develop algorithms which are numerically efficient and highly scalable at the same time: For this type of PDEs the solution at inner parts of the computational domain is strongly determined by all of its boundary values, that is, information must travel through the entire grid at least once. At the same time, these problems are characterized by an extremely high propagation speed for information such that even strictly localised effects immediately impact the overall solution in the whole domain. Thus, achieving fast convergence along with a high level of computational accuracy (*numerical efficiency*) is closely linked to the speed at which information can be exchanged across the entire domain. In contrast to that, the efficient parallelization of a numerical algorithm is mainly determined by the amount of interprocessor-communication needed to coordinate the single processors to ultimately produce a global solution. Achieving a good scalability to large processor counts (*parallel efficiency*) requires as much data locality as possible in order to minimize the communication overhead between the single processors. So, a proper compromise has to be found between those two very contradictory requirements.

A very natural approach for the parallel solution of PDEs based on finite difference discretization like in FDS is the use of *domain decomposition* techniques: The computational domain is subdivided into smaller subdomains which are assigned to the different processors of a parallel computer and simultaneously perform a series of local computations which are coordinated by synchronized data exchanges among each other. This parallel procedure may reduce the required runtime for a given constellation and enlarge the class of computable problems comprehensively.

Special care must be taken to ensure that this artificial subdivision does not impair the inherent global character of the underlying problem which especially holds true for elliptic problems. In particular, it must be guaranteed that the parallelization process preserves the convergence order and approximation quality of well-proven serial algorithms as best as possible. However, the efficiency of many serial elliptic solvers mainly relies on highly recursive data dependencies strongly connecting all parts of the domain which implies an extremely low level of intrinsic parallelism. In order to achieve a better parallel efficiency they must be split off and adapted to the new architectural features which typically requires extensive structural modifications to the numerical core and is often associated with more or less considerable losses of numerical efficiency (worse approximation quality, dependencies on the number of subdomains or the refinement parameters). The effects of this situation on the FDS pressure solution are described in more detail below and different solution techniques are presented.

2 DISCRETIZATION OF THE FDS PRESSURE EQUATION

The pressure equation in FDS, an elliptic partial differential equation of second order which is commonly referred to as *Poisson equation*, reads as

$$\nabla^2 H = -\frac{\partial(\nabla\cdot\mathbf{u})}{\partial t} - \nabla\cdot\mathbf{F}. \tag{1}$$

Detailed information on its derivation can be found in in the FDS Technical Reference Guide [1]. The pressure term $H \equiv |\mathbf{u}|^2/2 + \tilde{p}/\rho$ incorporates the density ρ as well as the perturbation pressure $\tilde{p}$ by which the fluid motion is driven. Obviously, H is strongly coupled with the velocity field $\mathbf{u}$ and the force term $\mathbf{F}$ which includes the thermodynamic contributions from the previous time step such as radiation, combustion, pyrolysis, etc. Thus, the quality of the pressure solution is of great importance for the accuracy of the entire numerical scheme in FDS.

In order to obtain a corresponding system of equations which can be solved on a computer system the spatial derivative of H in Equation (1) is discretized by a cell-centered *finite difference method* of second-order accuracy. Its numerical solution requires the specification of appropriate boundary conditions and has to be done at least twice in every step of the FDS time marching scheme, namely in the respective predictor and corrector parts. The efficient parallelization of Equation (1) must necessarily ensure that the strong global coupling induced by the underlying elliptic character is mapped as closely as possible.

2.1 *Global versus local discretization*

In order to describe the basic functionality of the various Poisson solvers in FDS, Figure 1 introduces a simple pipe-shaped geometry in 2D with a small internal obstruction, a prescribed inflow from the left, an open outflow on the right and a middle pressure device. This so-called `poisson2d` geometry will be used as a demonstration case throughout the entire paper.

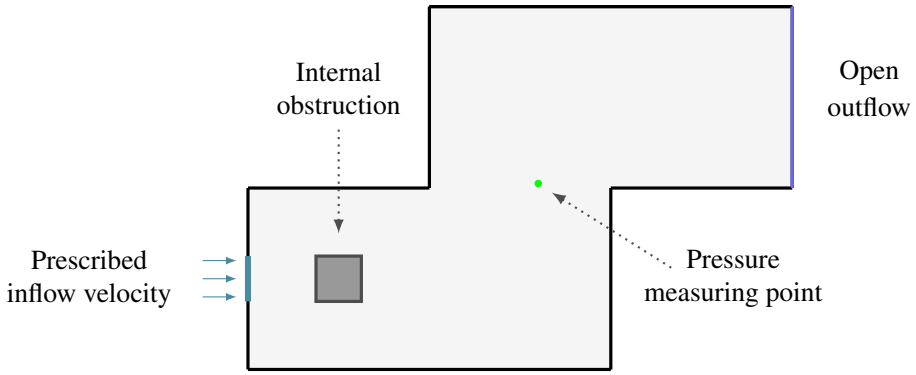

Figure 1: Simple pipe-shaped 2D-geometry `poisson2d` with an internal obstruction, a prescribed inflow from the left, a measuring point for the pressure and an open outflow on the right.

If the whole domain is discretized in one piece, one global system of equations is obtained,

$$Ax = b\,, \tag{2}$$

where n is the number of total grid cells, $x, b \in \mathbb{R}^n$ denote the corresponding solution and right hand side vectors and $A \in \mathbb{R}^{n\times n}$ the overall Poisson matrix.

However, since FDS is limited to cubic or rectangular meshes, typically a subdivision into M individual sub-meshes is used with n_m local cells each. In this case every sub-mesh holds its own system of equations

$$A_i x_i = b_i\,, \qquad i = 1,\ldots,M, \tag{3}$$

with corresponding local solution and right hand side vectors $x_i, b_i \in \mathbb{R}^{n_i}$ and locally defined Poisson matrices $A_i \in \mathbb{R}^{n_i\times n_i}$. The resulting global and local discretizations are shown in Figure 2.

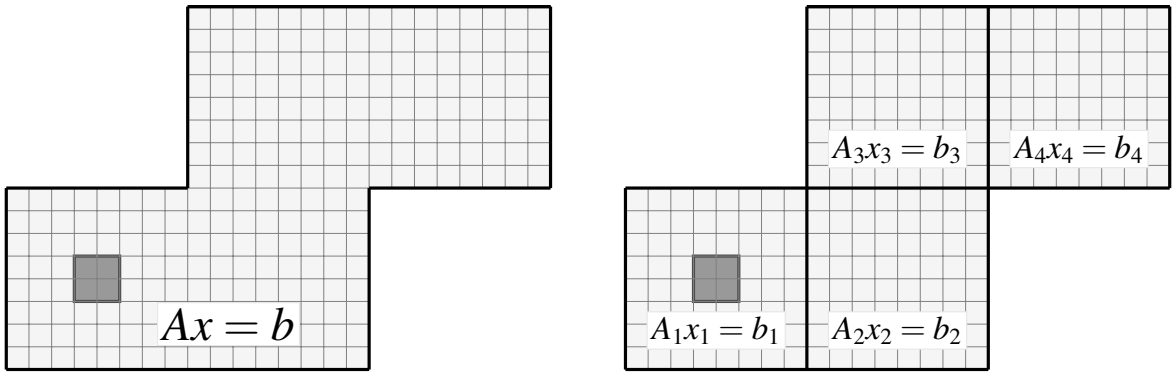

Figure 2: Different discretization types for the Poisson problem: (Left) Global discretization with one globally defined system of equations $Ax = b$. (Right) Local discretizations with a set of locally defined systems of equations $A_i x_i = b_i$.

Note that a global discretization can also be handled in a multi-mesh context, i.e. in a *data-parallel* sense. For this purpose, each mesh stores the corresponding part of the global matrix as well as the solution and right-side vector. By means of suitable data exchanges between adjacent meshes the individual matrix-vector operations are then performed in exactly the same way as would also be the case with an actual global execution.

With regard to parallelization the use of local discretizations seems to be the more natural approach. Thus, the crucial question is, if there are appropriate solutions strategies for a local discretization such that the obtained collection of mesh-wise solutions is comparably good as the overall solution for the corresponding global discretization, i.e. $\sum_{i=1}^{M} x_i \sim x$? This question will be addressed subsequently based on the example of the Poisson solvers used in FDS.

2.2 *Structured versus unstructured discretization*

Regarding the discretization of internal obstructions two possible approaches must be distinguished as well, which essentially differ in how they treat cells internal to solid objects and their neighboring gas-phase cells. Those different types have their pros and cons and also require different solution strategies which will be explained below.

- A **structured discretization** explicitly includes both gas-phase and solid cells and is the default type in FDS. The same matrix stencil is applied all over the domain. Without any exception all grid cells are incorporated into the resulting discretization matrix which therefore takes a very regular shape. As a major advantage this strategy allows the use of highly tuned solvers developed specifically for regular grid structures which can be performed with enormous computational efficiency. However, it is not possible to prescribe the required homogeneous no-flux condition at internal boundaries directly. Erroneously, the velocity field may contain non-zero contributions towards internal solids associated with a corresponding loss of accuracy which represents the greatest disadvantage of this strategy.

- An **unstructured discretization** only incorporates the gas-phase cells while omitting all solid cells from the system of equations. On gas-phase cells directly adjacent to the surface of an obstruction the homogeneous no-flux Neumann condition can explicitly be specified and included into the matrix so that there are no more penetration errors along internal obstructions. Thus, the major advantage of this strategy is that it offers more flexibility and achieves a higher degree of accuracy. But, in contrast to the structured case it requires the use of individual matrix stencils for the different grid cells depending on their location with respect to obstructions such that the regular matrix shape gets lost. The solution of such an irregular system places significantly higher demands on the robustness of the solver, which can comprehensively limit the achievable efficiency and represents the greatest disadvantage of this methodology.

In FDS, both the structured and unstructured discretization type can be used in combination with suitable solution strategies each, as will be shown subsequently. The differences between both are illustrated in Figure 3 where the respective matrix stencils are displayed. Both plots therein are related to the lower left mesh in case of the local discretization containing the internal obstruction as can be seen in the right plot of 2.

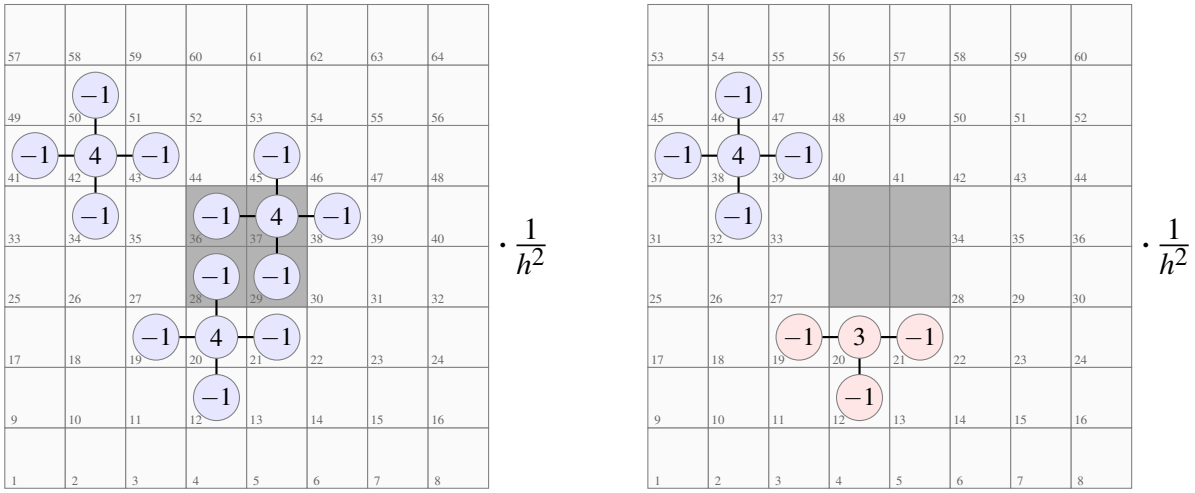

Figure 3: Matrix stencils for both discretization types: (Left) The structured discretization uses the same matrix stencil everywhere, including cells internal to the obstruction. (Right) The unstructured discretization uses individual matrix stencils, excluding all cells internal to the obstruction.

Figure 4 opposes the sparsity patterns of the resulting Poisson matrices to each other. As can be seen clearly, both discretization types lead to sparse matrices which basically have only very less non-zero elements. Based on the underlying stencils the non-zeros are restricted to only a few diagonal bands (5 in 2D, 7 in 3D) which is very less compared to the number of possible entries.

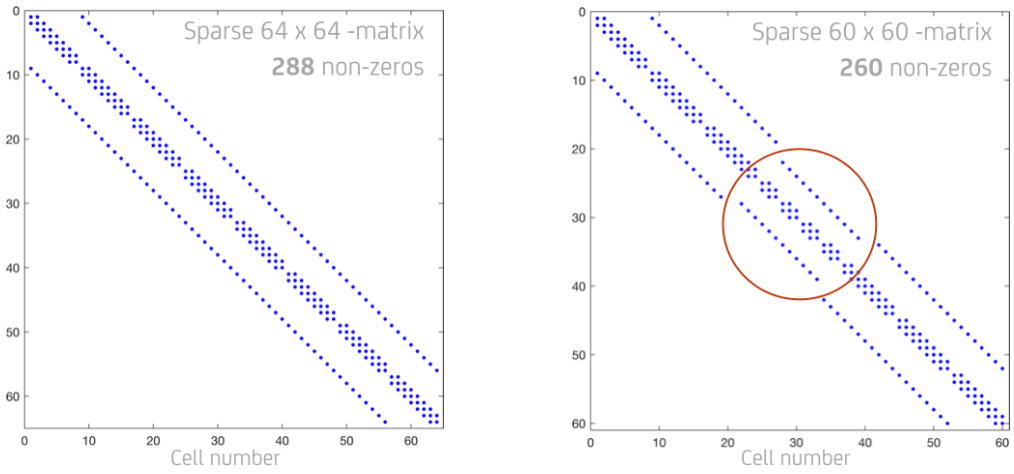

Figure 4: Sparsity patterns for the resulting Poisson matrices: (Left) The structured discretization leads to a completely regular pattern. (Right) The unstructured discretization shows irregularities related to the internal obstruction .

However, the structured discretization results in a highly regular matrix with a constantly repeating pattern while the unstructured discretization shows irregularities associated with the different treatment of the internal obstruction as indicated with the red circle in the right plot of Figure 4. For a simple obstruction like the one considered here, the differences are not big, but the situation is completely different for more realistic applications with complex geometric details. In any case, the mentioned optimized solvers for regular grids can no longer be applied in the unstructured case. Note, that the unstructured discretization even leads to a smaller dimensioned matrix since the obstruction is not included in the discretization process. But this supposed advantage typically does not carry weight due to the more complex structure as a whole.

3 DIFFERENT POISSON SOLVERS

Subsequently the different Poisson solvers available in FDS are described. They are based on different combinations of global/local and structured/unstructured discretizations.

3.1 Mesh-wise FFT

Spectral solvers like the *Fast Fourier Transformation* (FFT) belong to the class of direct solvers and exploit a very special property of the underlying Poisson problem, namely that sine and cosine functions are eigenvectors of the Laplace operator. They expand the solution as a Fourier series which can be quickly performed at rather low complexity.

In practice, FFT-solvers have proven to be highly efficient and are used in many different fields of application. However, they are restricted to structured grids which may impede their use for complex geometries.

As described in detail in the FDS Technical Guide [1], the pressure equation (1) is obtained from the momentum equation by a sequence of substitutions and simplifications. The major advantage of this derivation is that it finally leads to a system of equations which has constant coefficients (i.e. is separable). In its default version FDS is based on the use of local structured discretizations where the single meshes are of cubic shape with uniform grid resolutions each. Correspondingly, a highly optimized FFT solver for regular grids from the package CRAYFISHPAK [2] is used as the default Poisson solver in FDS.

For a single-mesh application only one globally acting FFT-method is performed which has proven to be extremely powerful and unbeatably fast over the past years since its highly recursive character extremely well captures the fast global information transfer within the only grid used. For multi-mesh applications, however, there is no longer one global rectangular discretization, but only a collection of local ones as explained above, such that the global connectivity is split off. Now, every sub-mesh performs its own local FFT to compute an exact solution to the related local Poisson problem, see Figure 5. The global solution is then composed from the local ones.

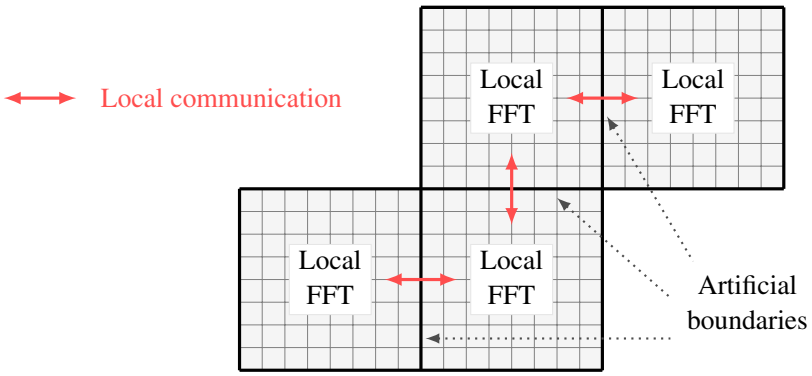

Figure 5: Mesh-wise FFT-method for the `poisson2d` geometry: First each mesh performs its own local FFT method, which requires data exchanges between neighboring meshes to define the boundary conditions along the new artificial boundaries. Then the global solution results from the composition of the local solutions.

A drawback associated with this approach is that the mathematical solvability of the local problems requires the definition of appropriate boundary conditions all over the local boundary. This particularly includes the cells along the new artificial boundaries between the meshes, but the correct boundary conditions are not known there at all. Instead, they are only approximately defined as Dirichlet boundary values, consisting of the mean values of neighboring cells in adjacent meshes taken from the last time step, which may lead to losses of accuracy.

On the other hand, this strictly locally oriented approach possesses an extremely high parallel efficiency because the required values for the definition of the boundary data have already been computed in the preceding time step and the exchange of the internal boundary values only requires next-neighbor communications which can be performed with great computational efficiency on today's parallel architectures.

There might be another disadvantage associated with this procedure: As already described, the most fundamental characteristic of the pressure equation (1) is a very fast propagation speed for information. Only a single time step may suffice to spread new information over the whole computational domain, i.e. local effects or perturbations have immediate impact on the overall solution. Due to the purely local character of the mesh-wise FFT solver, however, new information can only be transferred mesh-by-mesh, successively using the data exchanges between adjacent meshes as illustrated in Figure 6.

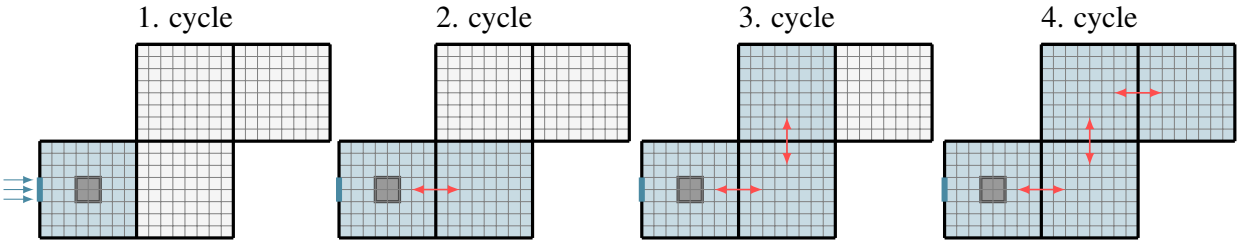

Figure 6: Delayed information transfer of the mesh-wise FFT solver for the `poisson2d` geometry: New information can only be passed across the whole domain by successively using the next-neighbor communications between adjacent meshes.

This necessarily involves a time delay compared to the real physical propagation speed and brings dependencies on the number of subdomains into play. The higher the number of subdomains and the associated artificial fragmentation of the global connectivity is, the stronger this effect may be with corresponding negative impacts on the accuracy and stability of the whole method. Nevertheless, for small numbers of subdomains this effect has proven to be very weak and the efficiency of the local FFT-methodology mainly dominates.

Finally, the following possible difficulties must be taken into account: Although the pressure term H is continuous at mesh interfaces, the finite-difference of its gradient is not such that the normal components of velocity may be different at mesh interfaces. Furthermore, as already described in the context of the structured discretization type, the normal components of velocity may be non-zero towards solid internal boundaries, or in other words, the velocity field may penetrate into internal obstructions.

In order to compensate these undesired effects an additional iterative correction strategy based on a *Direct Forcing Immersed Boundary Method* [3] is used in FDS: In every time step the mesh-wise FFT algorithm is not only applied once but multiple times until a specified tolerance for the velocity error along internal obstructions and mesh interfaces has been reached. For both, the internal obstructions and the mesh interfaces, the required velocity tolerance and the maximum allowed number of pressure iterations can be specified by the quantities `VELOCITY_TOLERANCE` and `MAX_PRESSURE_ITERATIONS` in the `&PRES`-namelist. By default, a velocity tolerance of "characteristic mesh size" divided by 10 and a maximum of 10 iterations is used. Certainly, the increased number of mesh-wise FFT solutions leads to a higher computational effort. Nevertheless, for a multitude of cases, especially in case of small mesh numbers or in steady-state like situations, only less pressure iterations are usually needed and convergence is achieved very quickly.

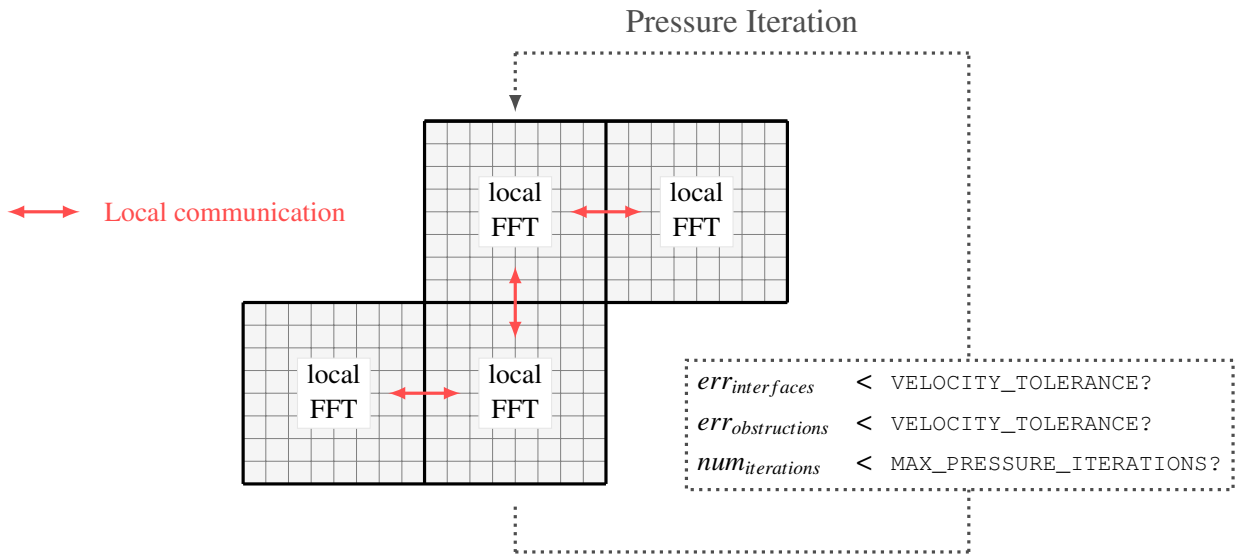

Figure 7: Repeated use of the mesh-wise FFT solver for the `poisson2d` geometry: In the scope of a surrounding pressure iteration the local FFT's are performed multiple times until the error related to the normal components of the velocity field along mesh interfaces and internal obstructions has been driven below a specified velocity tolerance.

Increasing the number of subdomains may worsen the convergence and accuracy behaviour associated with a noticeable increase of computational costs. Particularly, for extended geometries with a large number of meshes (i.e. tunnels) and/or transient boundary conditions this purely local strategy may experience difficulties to reproduce the rapid propagation of information for elliptic problems fast enough. In particular, these situations make the development of alternative solution concepts appear necessary and have driven their development forward.

3.2 *UGLMAT*

Within the framework of the Gaussian elimination algorithm, many direct algorithms rely on the "lower-upper"-factorization of the system matrix, $LU = A$, with corresponding triangular matrices L and U, see Fig. 8. The solution of the global system of equations (2) can then simply be obtained using a forward substitution step, $Ly = b$, followed by a backward substitution step, $Ux = y$, which can be performed at low computational costs. The solver UGLMAT which is available as alternative Poisson solver in FDS is a representative of this methodology.

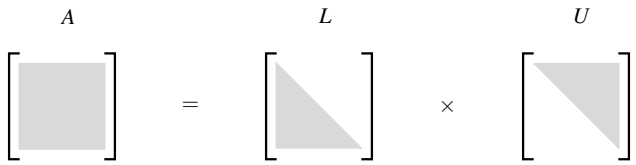

Figure 8: Decomposition of A into a lower triangular matrix L and an upper triangular matrix U.

Based on a global unstructured discretization of the overall system of equations (2), UGLMAT is able to compute the exact global solution up to machine precision. In particular, there are no errors along mesh interfaces and internal obstructions which is a major advantage of this approach. UGLMAT is basically built on the use of the optimized `CLUSTER_SPARSE_SOLVER` solver of the Intel® Math Kernel Library which can be applied for structured and unstructured discretizations. Compared to the optimized CRAYFISHPAK-FFT solver, however, it turns out to be slower by a factor of about 2 when applied for the same structured grid computation. Note, that UGLMAT currently can only be used for non-overlapping, non-stretched meshes at the same refinement level.

A main disadvantage of this approach is that less benefit can be drawn from the aforementioned sparsity of the Poisson matrix A. Unfortunately, the LU-factorization process leads to *fill-in*, i.e. it produces non-zero entries in L and U where A was zero before as illustrated in Figure 9. There, the sparsity patterns of the Poisson matrix A and its respective lower triangular matrix L are compared in case of a simple 3D-cube geometry with a refinement into 16^3 cells which already leads to the very unequal ratio of about 27 thousand non-zeros for A to about 1 million non-zeros for L. If the 3D-cube is further refined from 16^3 up to 128^3 cells this ratio even worsens dramatically and L requires about 238 times more memory than A.

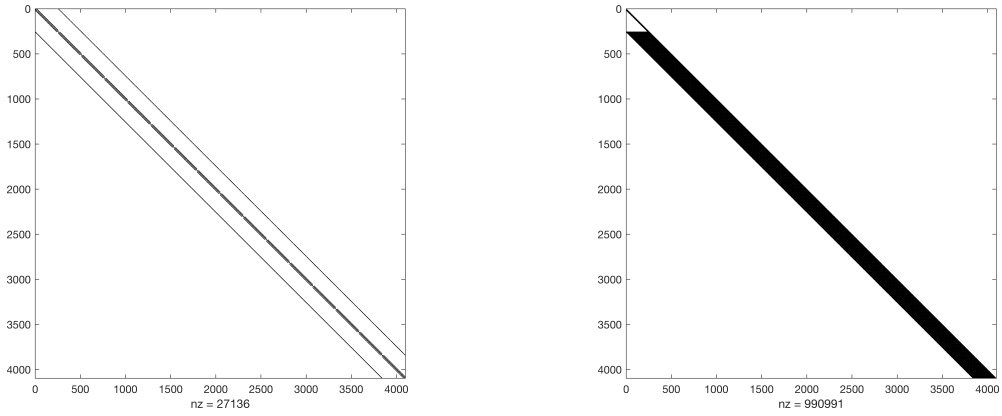

Figure 9: Sparsity patterns for the decomposition $A = LU$ in case of a refined 3D-cube with 16^3 cells: (Left) Poisson matrix A with about 27 thousand non-zeros; (Right) Lower triangular matrix L with nearly 1 million non-zeros;

Thus, the storage of L can become prohibitively expensive which especially holds true in case of huge systems of equations with many millions of unknowns. For realistically large CFD-problems, the associated additional storage requirements may exceed the capacity of the available computing system in the worst case. Hence, when considering the computational efficiency of LU-methods for a given application the resulting fill-in is a decisive criterion. All in all, the highly recursive, domain-spanning dependencies which have to be mapped in the LU-process make an efficient parallelization towards higher number of meshes rather difficult and decrease the scalability properties of this class of methods. For smaller numbers of meshes, however, UGLMAT can be a good alternative to the default FFT solver what needs to be tested in each individual case. As an example the `dancing_eddies` case from FDS Users's Guide can be considered.

3.3 ScaRC

With the aim to combine the advantages of the above Poisson solvers and to best possibly avoid their disadvantages, the solver package ScaRC was developed. In contrast to the upper two solvers FFT and UGLMAT, which both belong to the class of direct solvers, ScaRC is mainly based on iterative solution techniques.

While direct methods only require a single, but possibly very complex computational cycle to solve the system of equations exactly, iterative solvers perform multiple cycles producing a sequence of iterations which gradually improve an initial estimate of the solution until a predefined stopping criterion has been reached.

The computational complexity associated with each cycle of iterative methods is comprehensively less compared to the respective single cycle of direct methods. Thus, in comparison to direct methods, iterative methods are faced with the crucial question of how many cycles are ultimately required to achieve convergence.

3.3.1 Basic iteration with Schwarz preconditioning

The simplest form of an iterative methods is the so-called *basic iteration*,

$$x^k = x^{k-1} + B\left(b - Ax^{k-1}\right), \tag{4}$$

with iteration vectors $x^k, x^{k-1} \in \mathbb{R}^n$ and a *preconditioning matrix* matrix $B \in \mathbb{R}^{n\times n}$, which is used to increase the convergence speed by best possibly exploiting special knowledge of the underlying structure. Note, that B does not need to be built explicitly, but is only defined by its action.

Because the convergence properties of (4) itself are poor, more efficient generalizations are typically used such as the *preconditioned conjugate gradient method* (CG) and the *geometric multigrid method* (MG) which are able to significantly speed up convergence. For more detailed information on iterative methods please see Hackbusch [4] or Saad [5].

Usually, iterative methods are easier to implement than direct ones, because they can be reduced to a series of core components such as matrix-vector multiplications, scalar-products or linear-combinations of vectors, for which highly optimized program packages can be used, e.g. BLAS [6, 7]. Because they do not produce any fill-in, they preserve the sparsity structure of the system matrix and are often much less demanding with respect to their storage requirements than several direct methods .

With a view to appropriate parallelisation strategies iterative methods are of particular interest. This is mainly based on the fact that they allow to incorporate domain decomposition concepts in a very natural way by breaking up the preconditioning according to the underlying subdivision. The global preconditioning is now composed as the collection of locally defined preconditioning problems,

$$x^k = x^{k-1} + \sum_{i=1}^{M} B_i\left(b - Ax^{k-1}\right), \tag{5}$$

with suitably chosen local preconditioners B_i which is known as the additive version of the so-called *Schwarz preconditioning*. The local preconditioning problems are typically based on the more or less exact solution of the related local Poisson problems.

3.3.2 Inner core of SCARC

In the light of these considerations the alternative Poisson solver SCARC was developed and is mainly defined as symbiosis of global and local iterative techniques. In fact, SCARC is not only one single solver, but a whole class of different solvers and components which can be individually put together according to the situation at hand. The most fundamental philosophy behind is the use a global discretization based on a globally defined Poisson matrix. Furthermore the whole methodology can be applied for both structured and unstructured discretizations.

In its most basic form SCARC simply consists of the nested combination of a global basic iteration in the style of Equation (5) with additional local basic iterations on each single sub-mesh as sketched in Figure 10.

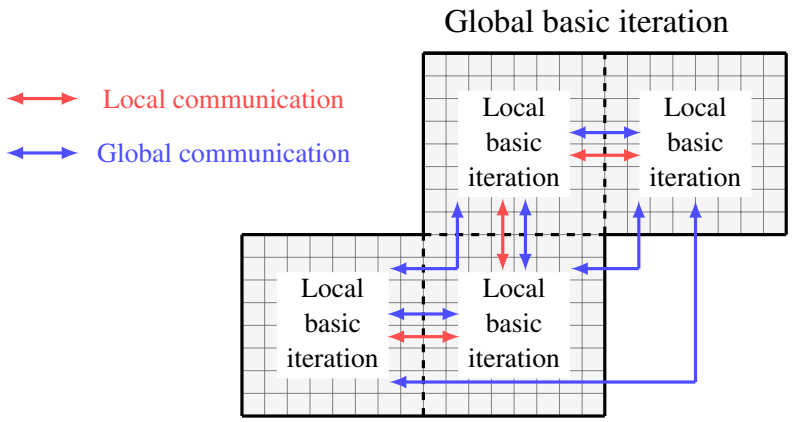

Figure 10: Inner core of SCARC consisting of a nested combination of a global data-parallel basic iteration defined on a global Poisson-matrix for the entire domain with mesh-wise preconditioning by local basic iterations.

The global basic iteration is distributed over the individual sub-meshes and serves to best possibly map the global dependencies. Since the matrix-vector products therein are related to the global Poisson-matrix and are only computed in a data-parallel way, there are no more errors along the mesh interfaces related to the normal components of velocity. This fact represents the most significant difference to the mesh-wise default FFT solver. However, the computation of these globally consistent matrix-vector products requires the local communication of mutually required data between adjacent meshes. Furthermore, to check the convergence criterion for the global basic iteration, overall defect norms have to be computed which requires global communication between all meshes as well.

Imbedded in this surrounding global basic iteration, the individual meshes perform their own local basic iterations until a predefined local stopping criterion has been reached. However, the respective local results are not used as solutions for their own, but only to correct the global solution and to guarantee a sufficiently high degree of local accuracy. In particular, the preconditioners for the local basic iterations can be chosen individually corresponding to the local situation which of course requires appropriate strategies to properly balance the computational load across processors.

3.3.3 Default 1-level generalizations of SCARC

To improve the overall convergence speed, the global and local iterations are usually substituted by more efficient solution strategies as already mentioned above. The global iteration is typically replaced by a data-parallel CG- or MG-method. Apart from using the aforementioned globally defined matrix-vector products, these methods incorporate even more globally acting features such as global scalar-products in case of CG or an additional coarse grid problem in case of MG which furthermore contribute to a stronger global coupling. In turn, there is a great freedom in the choice of the local solution techniques again. For example, the local iterations can be replaced by local MG-methods with different types of smoothing. Alternatively, optimized direct local solvers (according to the underlying discretization type) can be used leading to the exact solution of the local Poisson problems.

The resulting combination of global and local solvers can be used for structured and unstructured discretizations. In FDS the following two default variants for both cases are available.

Global structured discretization: The default variant for the structured case, simply abbreviated with SCARC, is illustrated in Figure 11. It can be called by setting `SOLVER='SCARC'` in the `&PRES` name list and is based on a global, data-parallel CG method. Due to the structured discretization type, the mesh-wise preconditioning is done by local FFT methods based on the optimized CRAYFISHPAK package. But for exactly that reason, there still may be velocity penetration errors towards internal obstructions. Thus, this variant is embedded into the already known pressure iteration. However, in contrast to the default FFT solver the velocity field is already correct along mesh interfaces and the pressure iteration only relates to internal obstructions.

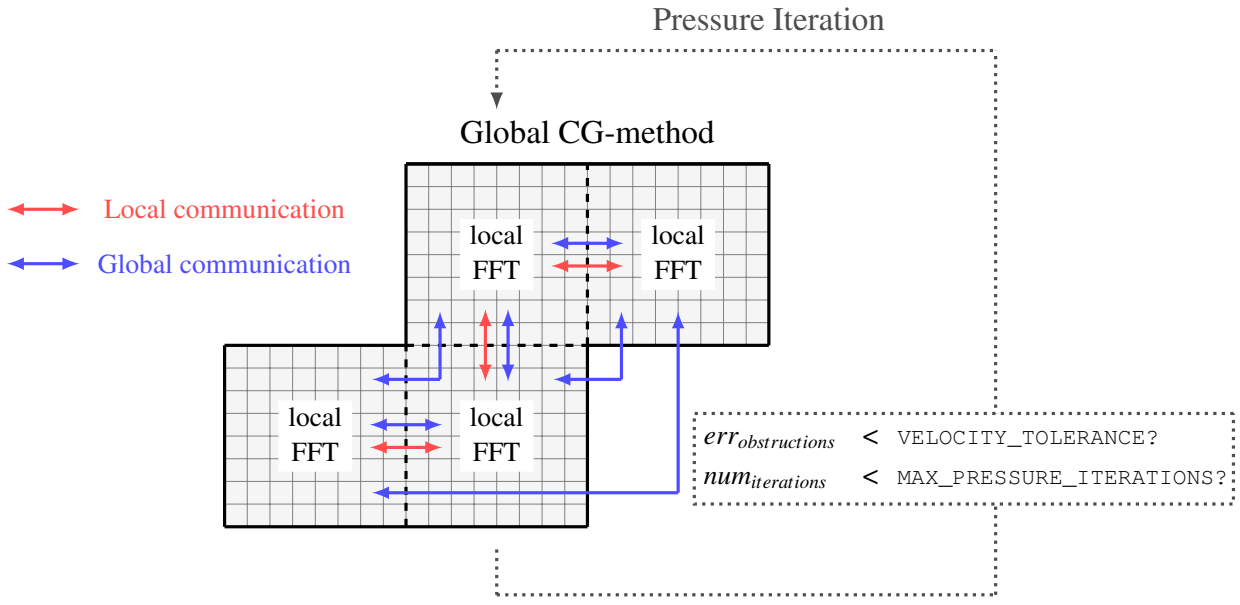

Figure 11: Default SCARC solver for a global structured discretization, consisting of a global data-parallel CG-method with mesh-wise preconditioning by local optimized FFT-methods, embedded in a pressure iteration to correct velocity penetration towards internal obstructions.

Global unstructured discretization: The default variant for the unstructured case, simply abbreviated with USCARC, is illustrated in Figure 12. It can be called by setting `SOLVER='USCARC'` in the `&PRES` name list and is also based on a global data-parallel CG method. But due to the unstructured discretization type, the mesh-wise preconditioning can no longer be done by local FFT's, but local *LU*-decompositions are used instead which are based on the optimized `PARDISO` solver of the Intel® Math Kernel Library. Apart from the already correct transitions of the velocity field along mesh interfaces, the treatment of internal obstructions is also correct and no pressure iteration is needed at all.

Certainly, the use of the local *LU*-decompositions also requires additional memory. However, these are only built and stored mesh-wise and do not act across the entire domain. Thus, the associated storage requirements seem to be acceptable or do at least not grow with increasing number of meshes. Nevertheless, as already mentioned, the local *LU*'s are not as efficient as the local FFT's, such that unstructured alternatives are currently being developed here.

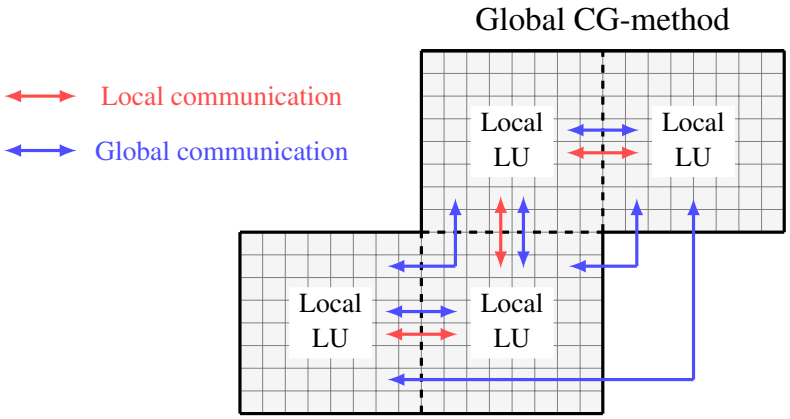

Figure 12: Default USCARC solver for a global unstructured discretization, consisting of a global data-parallel CG-method with mesh-wise preconditioning by local optimized *LU*-decompositions.

3.3.4 Further multi-level generalizations of SCARC

The SCARC variants shown so far were only defined on one single grid level, namely the fine level on which FFT and UGLMAT are also built. However, a very important generalization is based on the use of additional coarser grid leves, which basically can contribute to a much better global coupling. The two most important multi-level representatives of ScaRC are presented below.

SCARC-multigrid: A special variant for structured discretizations, SCARC-multigrid, is based on the use of a global MG-method instead of the CG-method. As illustrated in Figure 13 for the `poisson2d` geometry, it not only uses one fine grid level, but a complete hierarchy of levels with increasingly coarser degree of grid resolution. The total possible number of levels depends on the grid resolution. All levels work together in a well tuned choreography to finally produce a common global solution. Due to the structured nature of discretization an additional surrounding pressure iteration is used which, however, only relates to the internal obstructions as for SCARC.

The combination of approximate solutions on the different refinement levels (based on a small number of SSOR-preconditioned basic iterations each) with an exact solution on the coarse grid level (based on a suitable direct solver), provides a much stronger global coupling which is reflected in significantly better convergence rates as can be seen for many test cases. Certainly, the computational effort is increased at the same time, which has to be set in relation to the resulting gain of numerical efficiency.

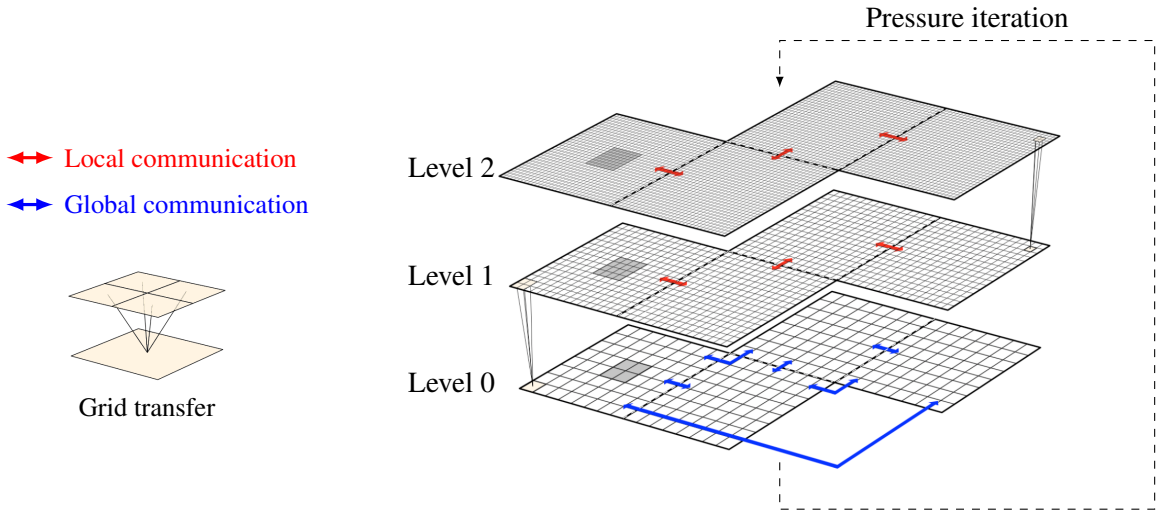

Figure 13: Alternative version SCARC-multigrid for a global structured discretization, consisting of a global data-parallel MG-method with mesh-wise SSOR-smoothing, embedded in a pressure iteration to correct velocity penetration towards internal obstructions.

SCARC**-twolevel:** Another structured multi-level variant, SCARC-twolevel, is based on the use of exactly two grid levels. In the same way as the default SCARC version, it performs a global CG-method on the finest grid level. In addition, exactly one further level with a coarser degree of resolution is used, on which the associated restricted problem is solved by a globally acting solver (mostly a direct one) with the highest possible accuracy. This level may consist of one or more coarsenings of the finest grid level whereby the domain decomposition itself is the coarsest possible level. As for all structured SCARC variants, a surrounding pressure iteration is used which only relates to the internal obstructions.

Again, both fine and coarse grid levels closely work together to finally produce a common solution which incorporates a much higher level of global coupling compared to the 1-level counterpart due to the additional coarse information.

A graphical representation for SCARC-twolevel is not used at this point. However, as an example for the `poisson2d` geometry, Level 2 and Level 0 from Figure 13 could be taken as fine and coarse grid level for the SCARC-twolevel variant.

3.3.5 Preliminary assessment and application notes

In the light of the above discussions, the acronym SCARC stands for:

- **Sca***lable*, because the number of global and local iterations can be varied,
- **R***ecursive*, since it can be applied recursively to more than 2 levels with different refinements,
- **C***lustering*, since neighboring meshes can be merged into larger clusters with special treatment.

Due to the complexity of the topic, an all-encompassing presentation was not possible here, but only a rough overview of the basic functionality and the most important representatives should be given. A detailed description of the underlying algorithmic concepts can be found in [8, 9, 10, 11].

In order to access the most recent version of SCARC it is necessary to download a clone of the FDS repository and to compile the code yourself. Please note, that there are still some limitations in the use of SCARC which actually correspond to those for UGLMAT, namely that it can only be applied for non-overlapping, non-stretched meshes at the same refinement level. While the use of an overlapping decomposition isn't necessary by design, the other two restrictions are only temporary in nature and corresponding enhancements are currently in progress.

Clearly, SCARC is not a pure black-box solver and its application requires a careful assessment of the underlying problem characteristics. However, its major advantage is that it allows a very diverse combination of different components with regard to the selection of the global solvers (CG, MG or combinations of both), the local solvers (from simple basic iterations to highly optimized MG) and the local accuracy requirements (from gaining only one digit up to machine precision) such that any available information about the underlying problem can be exploited.

With respect to the choice of the local solvers, there is great potential for further variations and optimization. In the most recent test calculations, the FFT's have turned out to be the fastest local solvers so far. In particular, they are comprehensively faster than the local *LU*'s which may lead to the following unexpected circumstances: Although USCARC doesn't need the pressure iteration (due to the correct treatment of internal obstructions) its runtime may be longer than that of SCARC explicitly performing the pressure iteration, namely just in such cases where only a small number of pressure iterations are needed to drive the penetration error below the specified tolerance. This imbalance is to be addressed by developing more efficient local solvers for the unstructured case.

In conclusion it must be pointed out that SCARC is still in a beta development state. Test calculations for a large number of cases have shown that it basically is able to achieve a high level of accuracy. This is mostly due to the fact that a global discretization is used so that the correct velocity field along inner mesh interfaces is calculated and no additional pressure iteration is needed there. In case of the unstructured variant USCARC, the penetration errors towards inner obstructions are also eliminated.

However, especially in case of longer lasting validation cases, it has also been observed that the previous runtimes are not yet optimal. This remaining weakness is mainly based on the fact that the focus of the development has been on the fundamental algorithmic correctness of the individual solution components and less on runtime performance so far. Nevertheless, there is still a great potential for runtime improvements and corresponding improvements are currently in progress.

4 NUMERICAL TESTS

During the previous sections different Poisson solvers were presented, namely the structured FFT and different variants for structured SCARC as well as the unstructured UGLMAT and USCARC. To describe their algorithmic functionality, the pipe-shaped demonstration case `poisson2d` from Figure 1 was used throughout the entire paper.

Subsequently, this case will also be used as verification case to check how well the individual solvers can deal with internal obstructions and multi-mesh decompositions, and how quickly they can transmit global information. To this end, air is blown into the domain whereby the inflow velocity is continuously varied within a range of 0 to 2 m/s such that the flow pattern in the entire domain frequently changes. As will be seen below this special setting causes a very characteristic stair-like course of the pressure trace which is recorded in the middle pressure device up to a final simulation time of 0.5 s. The necessity to continuously adapt to varying global situations poses a particular challenge to the pressure solver and is intentionally used to analyze the related scalability of the different variants with respect to an increasing number of meshes.

4.1 2D Poisson test case based on a subdivision into 4 meshes

First, the `poisson2d` geometry is subdivided into 4 meshes with a side length of 40 cm each. The single meshes are refined into 16^2 cells corresponding to a grid resolution of 2.5 cm. The side length of the internal obstruction amounts to 10 cm. The resulting flow field at time $t = 0.31$ s is illustrated in Figure 14.

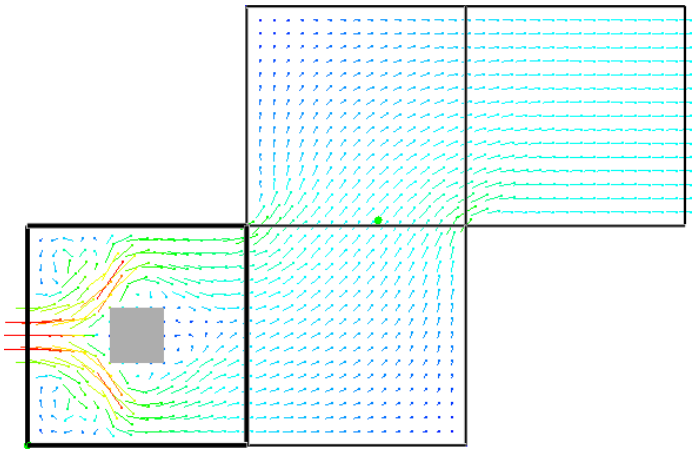

Figure 14: Flow field for a 4-mesh subdivision of the `poisson2d` test case at $t = 0.31$ s.

Both structured solvers, FFT and SCARC were applied in two different versions, namely with the respective default velocity 0.0125 m/s abbreviated as FFT-default and SCARC-default, and with a tight velocity tolerance of 0.00001 m/s abbreviated as FFT-tight and SCARC-tight. No velocity tolerance had to be specified for the unstructured variants UGLMAT and USCARC. The measured pressure traces for these six different solvers are illustrated in Figure 15.

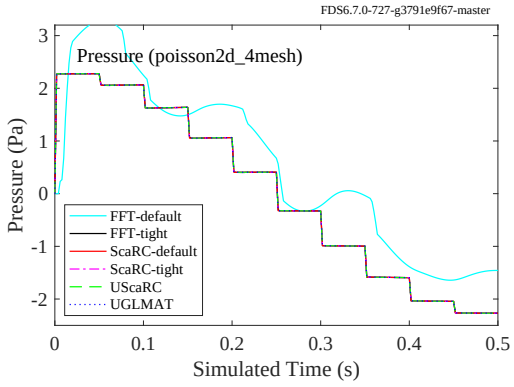

Figure 15: Pressure traces of the different pressure solvers for a 4-mesh subdivision of the `poisson2d` test case.

As a globally operating direct method, UGLMAT provides an exact solution to the global Poisson problem in each time step. Thus, its pressure trace is regarded as a reference solution for all other solvers. Obviously FFT-default (cyan) has troubles to map the pressure trace correctly while SCARC-default (red) already seems to match exactly. Both are structured solvers which are not able to set the correct boundary values along the inner obstruction by design. However, this basically doesn't seem to carry much weight here, as can be seen for SCARC-default for which the obstruction is the only possible error influence in this test case.

With a view to the constantly changing global velocity field, the multi-mesh decomposition seems to cause much more troubles which is reflected in the serpentine line of FFT-default. The steady changes of the inflow conditions can only be spread across the entire domain by successive mesh-by-mesh communications. Thus, the new inflow information reaches the pressure device in the third mesh only with delay. Evidently, this cannot be sufficiently remedied by the default pressure iteration.

However, as observed for FFT-tight (black), the tighter pressure iteration works great and is able to completely resolve these troubles leading to a correct pressure trace if only a higher number of pressure iterations is performed. In contrast to this, there seems to be no reason at all to apply SCARC-tight (magenta dashed dotted) in this case. Since all SCARC variants provide correct transitions at mesh interfaces by design, the overall error influences are significantly less pronounced here. Finally, USCARC (green dashed) operates independently of all these effects and produces the same correct pressure trace as UGLMAT (blue dotted).

So far, Figure (15) reflects the accuracy which is achieved by the different solvers in the approximation of the pressure trace, but it does not say anything about the numerical effort required for this. While for both unstructured solvers UGLMAT and USCARC only one pressure iteration is needed by construction, the structured variants require the execution of different numbers of pressure iterations until the respective velocity tolerance is met.

To analyze this situation in more detail, the achieved accuracies for the velocity errors, as displayed in Figure 16, are set in relation to the number of pressure iterations required to reach them, as displayed in Figure 17. Note, that Figure 16 only illustrates the velocity tolerances for the structured solvers because UGLMAT and USCARC achieve machine precision in the range of

10^{-16} by design which is omitted here to allow a more diversified view of the remaining solvers. Similarly, while the left plot in Figure 17 compares the pressure iterations for all six solvers, the right plot omits FFT-tight in order to give a more detailed overview of the others.

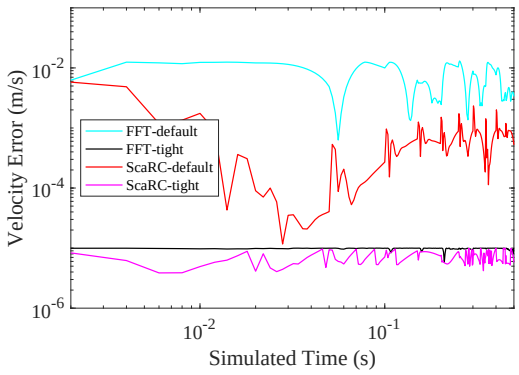

Figure 16: Velocity tolerances of the structured pressure solvers for a 4-mesh subdivision of the `poisson2d` test case.

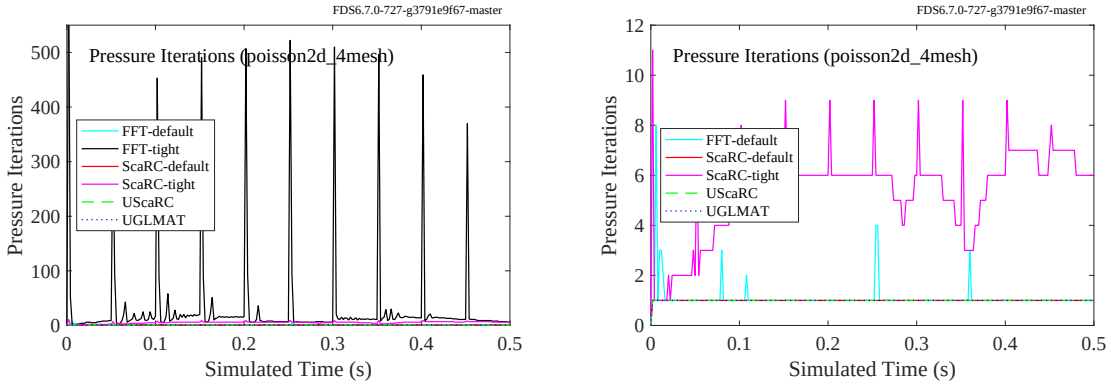

Figure 17: Pressure iterations of the different pressure solvers for a 4-mesh subdivision of the `poisson2d` test case. (Left) Comparison of all solvers. (Right) Zoomed view without FFT-tight.

According to the size of the default velocity tolerance, FFT-default (cyan) leads to a coarse velocity error in the range of 10^{-2} as can be seen in Figure 16. This error consists of two parts, namely the error along the internal obstruction and the error at the mesh interfaces. The related pressure iteration mostly converges within 1 cycle. Only with changing inflow conditions occasionally up to 4 cycles are needed as can be seen at the cyan line in the right plot of Figure 17. The maximum number of 8 cycles is only required at the beginning of the simulation until the flow field has built up for the first time, but never again after. Thus, convergence is fast, but it is also associated with the error of the pressure trace displayed in Figure 15.

Although its surrounding pressure iteration is based on the same default velocity tolerance, SCARC-default (red) already provides a basically smaller velocity error compared to FFT-default in the range of about 10^{-4} up to 10^{-3}, see Figure 16 again. This is because the only part that makes up this error relates to the inner obstruction, whereas nothing is added at mesh interfaces any more. Convergence can always be achieved in just 1 pressure iteration independently of the inflow changes and there is no major fluctuation at the beginning, see the right plot of Figure 17.

Figure 16 also proves that FFT-tight (black) and SCARC-tight (magenta) in fact succeed to fulfill the required velocity tolerance of 10^{-5}. However, when looking to Figure 17 the biggest difference between both solvers becomes apparent: While FFT-tight needs relatively many pressure iterations (averagely 35) to fulfill the fine tolerance, SCARC-tight requires considerably less (averagely 6). Note, that the average calculation was restricted to the time interval $[0.05, 0.5]$ such that the higher fluctuations which only occur at the beginning were neglected in order not to falsify the whole picture. In particular, at every change of the inflow conditions, the latency of FFT-tight gets obvious in a short-term increase of the number of iterations (up to 522 in the worst case) which basically relies on the fragmentation induced by the subdivision.

The right plot in Figure 17 also reveals slight increases for SCARC-tight for every inflow change, since it still has to prevent the velocity penetration to the obstruction. However this is significantly less than for FFT-tight and a maximum of 11 iterations is not exceeded. This difference again reflects the fact that SCARC already captures the subdivision correctly and that the only disturbing influences refer to the internal obstruction. As expected, UGLMAT (blue dotted) and USCARC (green dashed) produce machine precision accuracy in exactly one pressure iteration.

4.2 *2D Poisson test case based on a subdivision into 16 meshes*

Subsequently, the `poisson2d` geometry from Figure 1 is not only subdivided into 4 but into 16 meshes with a side length of 20 cm and 32^2 cells each (corresponding to the finer grid resolution of 0.625 cm). The resulting flow field is illustrated in Figure 18.

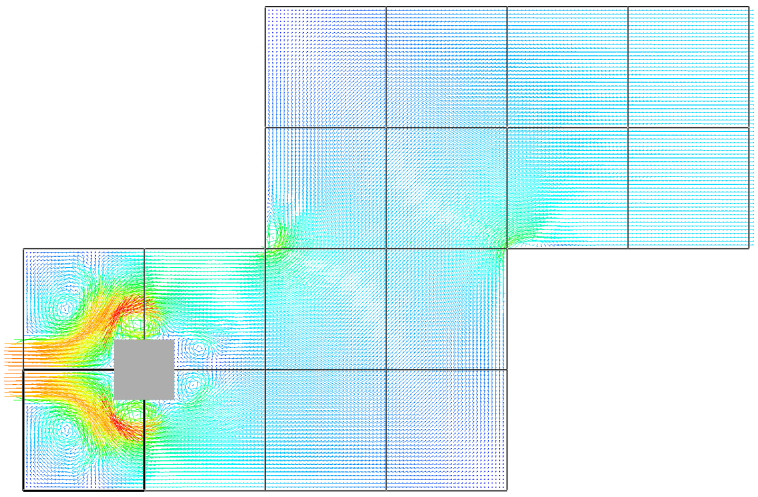

Figure 18: Flow field for a 16-mesh subdivision of the `poisson2d` test case at $t = 0.31$ s.

As Figure 19 shows, a fundamental improvement of numerical efficiency can be achieved by adding coarse grid information. The left plot therein displays the slow convergence rates which are achieved for default ScaRC and UScaRC (working both only on fine-grid level), the much better rate for ScaRC-twolevel (working on 2 different grid levels) and the very good rate for ScaRC-multigrid (working on 3 different grid levels as displayed in Figure 13). The related number of ScaRC iterations is shown in the right plot of Figure 19.

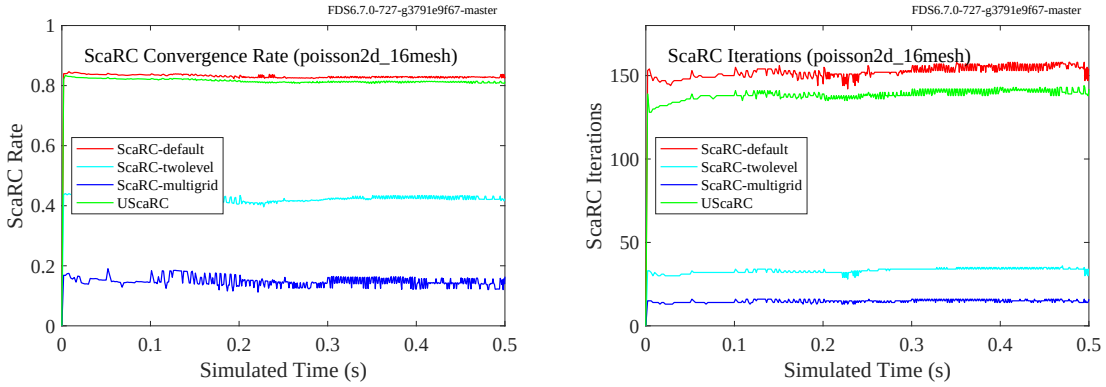

Figure 19: Comparison of different ScaRC variants for a 16-mesh subdivision of the `poisson2d` test case. (Left) ScaRC convergence rates. (Right) Number of required ScaRC iterations.

When looking to ScaRC-twolevel, the additional usage of a coarser grid level obviously leads to a halving of the convergence rate from 0.81 to about 0.42, see the left plot of Figure 19. The right plot therein shows that this reduction is also associated with a comprehensive reduction of the number of required global ScaRC iterations from 150 to about 40, which is quite much because it has to be performed twice per FDS time step. Even more significant convergence acceleration can be achieved with the ScaRC-multigrid variant. Here the number of global ScaRC iterations is reduced by a factor of 10 to about 15 while the final convergence rate is in the range of 0.15.

Please note that if only a 4-mesh subdivision instead of the 16-mesh subdivision is used for the `poisson2d` geometry while maintaining the same fine-grid resolution of 0.625 cm, a convergence rate of 0.7 with about 90 global iterations is achieved for the default ScaRC and UScaRC which still is not very good but significantly better than for the 16-mesh case. This observation reflects the fundamental design property of the 1-level variants that their convergence rate decreases with increasing number of sub-meshes.

This difference is far less pronounced for the two multi-level variants ScaRC-twolevel and ScaRC-multigrid which, due to the use of additional coarse grid information, show convergence rates that are significantly more independent of the number of sub-meshes.

Certainly, the ScaRC iterations incorporating multiple grid levels are more expensive than the purely 1-level related ones, since additional computations and communications are required. This has to be kept in mind when comparing the different approaches. All in all, it becomes clear that there is still potential for improvement with regard to the convergence rates of the default ScaRC variants, but that the use of corresponding coarse grid information points the way and a suitable balance between numerical and parallel efficiency needs to be further worked out here.

4.3 *Summary from the previous test calculations*

The purpose in developing the `poisson2d` test case was on the one hand to illustrate the functionality of the different SCARC variants and on the other hand to prove their basic correctness. The variants shown differ significantly in the type of the underlying discretization, the preconditioning mechanisms used, and whether they work only at the fine grid level or at additional coarser grid levels. The biggest difference to the current FFT solver consists in the fact that SCARC has no more errors along inner mesh boundaries. Thus, for the structured SCARC variants often much less pressure iterations are needed than for the default FFT solver, because it only has to reduce the penetration error towards internal obstructions. For the unstructured variant USCARC, no pressure iteration is needed at all.

The previous studies have shown that the addition of coarser grid levels can contribute to a significant improvement of numerical efficiency. However, the achieved convergence acceleration must be put in relation to the increased costs for each single iteration. A final assessment for the performance of the different variants can only be done on the base of additional measurements of the computational times. In order to arrive at meaningful estimates for the optimal parameters a-priori, different sensitivity studies are in work which should finally allow to identify an optimal set of parameters that delivers resilient and efficient results for a large number of general cases. The coming test series will concentrate on the official FDS verification and validation cases to examine the suitability of SCARC for more realistic cases.

References

[1] Kevin B. McGrattan, Simo Hostikka, Randall McDermott, Jason Floyd, and Marcos Vanella. Fire Dynamics Simulator Technical Reference Guide Volume 1: Mathematical Model. National Institute of Standards and Technology, 6.7 edition, June 2018. 2, 6

[2] A. Sweet, Roland. Crayfishpak: A vectorized fortran package to solve helmholtz equations. 6

[3] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations. Journal of Computational Physics, 161:35–60, 2000. 7

[4] W. Hackbusch. Iterative solution of large sparse systems of equations, volume 95 of Applied Mathematical Sciences. Springer-Verlag, New York, 1994. Translated and revised from the 1991 German original. 10

[5] Y. Saad. Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003. 10

[6] Jack J. Dongarra, Lain S. Duff, Danny C. Sorensen, Henk Vorst, and A. Vander. Numerical Linear Algebra for High Performance Computers. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. 10

[7] Jack J. Dongarra. Basic linear algebra subprograms technical forum standard, international journal of high performance. Applications and Supercomputing, ACM Transactions on Mathematical Software(16(1)):1–111, 2002. 10

[8] Susanne Kilian. SCARC: Ein verallgemeinertes Gebietszerlegungs-Mehrgitterkonzept auf Parallelrechnern. PhD thesis, Universität Dortmund, Berlin, 2001. 15

[9] Susanne Kilian and Stefan Turek. An example for parallel SCARC and its application to the incompressible navier-stokes equations. Preprint, Universität Heidelberg, June 1998. 15

[10] Dominik Göddeke. Fast and accurate finite-element multigrid solvers for PDE simulations on GPU clusters. PhD thesis, TU Dortmund, May 2010. 15

[11] H. Wobker. Efficient Multilevel Solvers and High Performance Computing Techniques for the Finite Element Simulation of Large–Scale Elasticity Problems. PhD thesis, TU Dortmund, March 2010. 15