CFD and OPTIMIZATION 2011
An ECCOMAS Thematic Conference
23-25 May 2011, Antalya TURKEY

SUNDANCE PDE SOLVERS ON CARTESIAN FIXED GRIDS IN COMPLEX AND VARIABLE GEOMETRIES

Janos Benk*, Miriam Mehl† and Michael Ulbrich††

*Department of Computer Science, Technische Universität München
Boltzmannstraße 3, 85748 Garching, Germany
email: benk@in.tum.de
†Institute for Advanced Study, Technische Universität München
Lichtenbergstraße 2a, 85748 Garching, Germany
email: mehl@in.tum.de
††Department of Mathematics, Technische Universität München
Boltzmannstraße 3, 85748 Garching, Germany
email: mulbrich@ma.tum.de

Key words: PDE Toolbox, Fluid Dynamics, Cartesian Grids, Adaptivity, Complex Geometries, Immersed Boundary

Abstract. *Structured adaptive Cartesian grids are a very memory-efficient alternative to unstructured computational grids and have been implemented as a new available grid type in the Sundance toolbox [3]. Our implementation includes an appropriate handling of hanging nodes as well as a parallelization via domain decomposition. Immersed boundary methods are essential for the representation of complex geometries on such Cartesian computational grids with more than first order accuracy. Only for rather simple setups, a deformation of a standard Cartesian grid (or several blocks of Cartesian grids) can be considered as an alternative. This becomes even more evident in scenarios with changing geometries such as shape optimization or multiphysics simulations. In this paper we present an implementation of an immersed boundary method in the Sundance toolbox based on Nitsche's method [5]. Tests with fluid dynamics applications show the potential of Cartesian grids in terms of memory efficiency as well as fast and robust treatment of complex geometries using Nitsche's method.*

1 INTRODUCTION

PDE- (partial differential equation) constrained optimization for multiphysics applications induces a two-fold computational challenge: First, multiphysics simulations already require highly efficient and flexible solvers to tackle both the large computational costs and the need to change or enhance scenarios regarding the model, the solver, or the coupling algorithm. Second, optimization algorithms further increase the computational costs. We aim at establishing a user-friendly, flexible, but still highly efficient framework for PDE-constrained optimization. This toolbox is based on Sundance, which has been

developed for the prototyping of PDE solvers and PDE-constrained optimization on a collection of grid types [3]. Sundance allows for the intuitive formulation of PDE problems in a weak-form-like notation and offers full flexibility in choosing various finite element discretizations of arbitrary approximation order. The task of the work presented in this contribution was to enhance Sundance with an adaptive Cartesian grid that provides a higher memory efficiency, a relatively easy parallelization, and a faster adaption to geometry changes as compared to unstructured grids. As both optimization and multiphysics applications such as shape optimization or fluid-structure interactions involve complex and in particular changing geometries, a substantial part of the work concentrates on geometry representation and handling of boundary conditions in a fixed grid setting using Nitsche's method [5]. As an example application, we present fluid dynamics simulations, and calculate drag and lift coefficients for the stationary DFG-Benchmark 'flow around a cylinder' [7].

2 SUNDANCE BASICS

In this section, we briefly describe the Sundance toolbox, in which we have implemented our Cartesian grid as an additional available grid type together with Nitsche's method for an accurate treatment of boundary conditions on complex geometries.

Sundance is a finite element method (FEM) PDE toolbox within the Trilinos library [3]. It incorporates the whole "simulation pipeline" of a classical PDE problem, as it is illustrated on Fig. 1. For the user, the most important part is the "problem formulation" section, where the user implements the PDE-problem in Sundance. Sundance uses operator overloading to provide a weak formulation of the respective PDEs directly in C++ code and, thus, relieves the user from all further details such as integration of basis functions, mapping of basis functions and operators from a reference element to concrete grid cells, or matrix assembly. To demonstrate this for an example, we consider the weak

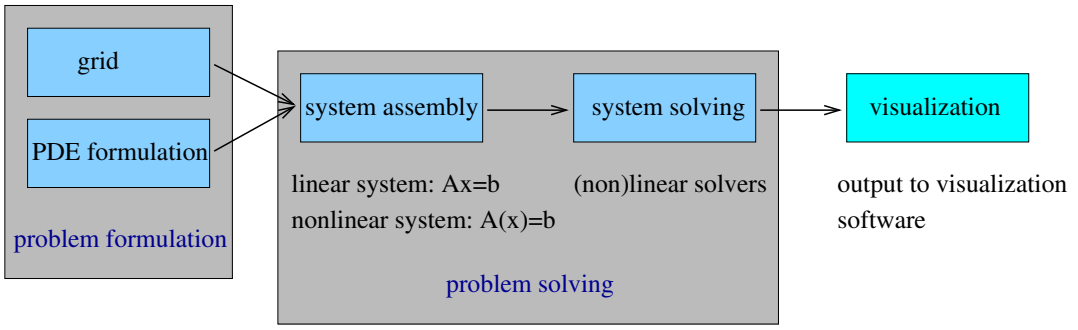


Figure 1: One representation of a PDE-toolbox simulation pipeline. This pipeline incorporates the processes from the PDE problem formulation till the visualization of the results.

form of the Poisson equation

$$\int_\Omega (\nabla u \cdot \nabla v - fv)\, dx = 0,$$

here u is the unknown function, v is the test function, and f is a given function. For u and v, a basis given by first order Lagrange polynomials (i.e. bilinear functions) is used.

To implement this example, the user has to provide the following C++ code in Sundance:

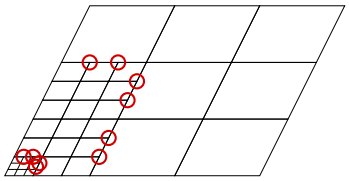
```
Expr dx = new Derivative(0);
Expr dy = new Derivative(1);
Expr grad = List(dx, dy);
Expr u = new UnknownFunction(new Lagrange(1), "u");
Expr v = new TestFunction(new Lagrange(1), "v");
QuadratureFamily quad = new GaussianQuadrature(2);
Expr integral = Integral( Omega , (grad*u)*(grad*v) - f*v, quad );
```

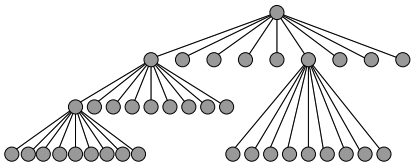
In the code above, the object *Omega* defines the computational domain where the integral should be evaluated for the assembly process of the system matrix. Such an object is called cell filter and is a general way for the user to define domains on the grid by selecting a subset of grid cells from the whole grid covering a possibly larger domain, in which the actual computational domain is embedded. Concerning the choice of basis functions and grid types, the user can freely combine different basis functions with various types of grids without major restrictions. The newly implemented adaptive Cartesian grid, which will be briefly described in Sect. 3 is also available for users in the current Sundance version.

The later stages of the simulation pipeline in Fig. 1, i.e. matrix assembly and solution of the resulting linear or nonlinear systems, are more or less hidden to the user. Sundance uses the linear and nonlinear solver packages from the Trilinos library. Results are written to a file with a defined format, such that they can be visualized with special software packages. Another major potential of Sundance is the distributed memory parallelism, which is transparent to the user. In [4], the authors present good weak scaling up to 256 processors for the Stokes matrix assembly.

3 CARTESIAN GRIDS

The tree-structure of our adaptive Cartesian grids makes the storage of any grid information such as relations between nodes, edges, faces, and elements obsolete. In addition, a tree-structured grid can be generated very fast based on octree or similar algorithms for arbitrary geometries. This immediately reduces the memory requirements and the overhead for grid generation. See Fig. 2 for a simple example grid in 2D together with the associated tree of grid cells. The same type of grids is used in the PDE framework Peano (see http://www5.in.tum.de/peano/releases/index.html, [9], and citations therein), that has been developed with the task to fully exploit the potential of this grid type using a sophisticated data management based on a particular depth-first grid traversal following a discrete iteration of a space-filling curve and data stacks in combination with matrix-free





grid	storage
Sundance-unstr.	698MB
Sundance-struc.	360MB
Cartesian	43MB

Figure 2: Left and middle: two-dimensional tree-structured grid together with the respective tree of grid cells at different refinement levels. Hanging nodes are marked with red circles. Right: comparison of storage requirements for the regular grid with 532.900 points implemented as an unstructured Sundance grid, a structured Sundance grid, and a tree-structured grid.

solvers and dynamical grid adaptivity. Due to the restrictions introduced by the iterator concept of Sundance that requires a grid to offer the possibility to iterate over user-defined subsets of cells, faces, edges, or vertices (e.g. all boundary faces), the grid implementation can not apply the same data management as Peano and, thus, exploits only parts of the full saving potential. On the other hand, the implementation in Sundance offers the fast prototyping functionality of the Sundance toolbox and therefore allows to develop and test new numerical methods much faster than in Peano.

Complex geometries are embedded into a square or cube in our approach resulting in three types of grid cells in case of a fluid dynamics simulation, e.g.: fluid cells, obstacle/outer cells, and cells intersected by the domain boundary. For moving geometries, an Eulerian approach is applied, i.e. the grid remains constant (up to dynamically adaptive refinement or coarsening) while the cell markers change according to geometry changes. This approach allows to tackle large movements or even topology changes in a very natural way.

Besides their poor approximation property at complex geometry boundaries that will be resolved in Sect. 4, the second drawback of adaptively refined Cartesian grids is the occurrence of so-called hanging nodes, i.e. vertices of a cell that are not a vertex of all adjacent cells. Hanging nodes are marked with red circles in Fig. 2. They need a special treatment in the matrix assembly process. This problem is solved in a very natural way in our implementation using element-wise operator assembly in combination with suitable interpolation and restriction operators for the transport of operator contributions from hanging nodes to neighboring non-hanging father nodes or, vice versa, of unknowns from father nodes to hanging nodes. Hanging nodes themselves are not allowed to have associated degrees of freedom in order to ensure the continuity of the resulting solutions. We have implemented a version of this method that works independent from the actual PDE and discretization.

Figure 2 shows the reduction of storage requirements for the pure grid storage as compared to the unstructured and structured Sundance grids. The full storage saving potential of our structured grids however can only be exploited in combination with matrix-free solvers, which is a task for the future. In the current implementation, Sundance explicitly assembles the system matrices in order to provide them to the solver library. A domain decomposition for our Sundance structured grid has been implemented using a greedy

method, which estimates the load of the coarsest level of the tree and distributes the leafs in a load-balanced manner to the processors.

4 BOUNDARY CONDITIONS FOR COMPLEX GEOMETRIES WITH NITSCHE'S METHOD

As mentioned above, we embed a complex geometry in a rectangle. As a basis for the marking of cells according to that geometry and for the special treatment of cells cut by the domain boundary, an analytical or numerical description of this boundary is required. We implemented two variants—one based on an analytical description (for a sphere, e.g.) and one based on polygons in the two-dimensional case[1]. In the examples presented in this paper, we used polygons, since they are the more general approach.

As a result of the complex geometry and the required flexibility in terms of boundary conditions, we cannot use finite element spaces fulfilling the boundary conditions. Instead, we use standard finite element spaces and enforce non-homogeneous Dirichlet boundary conditions using Nitsche's method. Nitsche's method has originally been established for the Poisson equation in [5], but [1] shows that it can be extended to more general problems such as the Navier-Stokes equations. Essentially, Nitsche's method includes the boundary integrals resulting from integration by parts and non-zero basis function values at boundaries in the discrete operators and, additionally, enforces the boundary conditions by adding extra terms with an h-dependent weighting in the weak form of the respective PDE.

In the following we formulate the stationary Navier-Stokes equations as our main test application and use Nitsche's method for the Dirichlet boundary conditions. We consider a domain $\Omega \subset \mathbb{R}^2$ and the unknown functions $u = (v, p) \in H^1(\Omega)^2 \times L^2_0(\Omega)$ with the velocities v and the pressure p. Further, let $\Gamma = \partial\Omega$ be the boundary of the domain Ω, $g \in H^{1/2}(\Gamma)$ the non-homogeneous Dirichlet boundary conditions, and $f \in L^2(\Omega)^2$ the external forces acting on the fluid. With these definitions, the incompressible fluid flow is described by the stationary Navier-Stokes equations

$$-\nu\Delta v + (v \cdot \nabla)v + \nabla p = f \ \ in\ \Omega \tag{1}$$

$$\nabla \cdot v = 0 \ \ in\ \Omega \tag{2}$$

$$v = g \ \ on\ \ \partial\Omega \tag{3}$$

To transform (1)-(3) to the weak form, we test the PDE with the functions $\phi = (\psi, \xi) \in$

[1]The accuracy of the latter is sufficient for second order discretizations. For solvers based on higher order finite elements, also the accuracy of the geometry description should be enhanced accordingly using for example spline functions.

$H^1(\Omega)^2 \times L_0^2(\Omega)$, and integrate by parts:

$$\begin{aligned}(-\nu\Delta v, \psi) &= \nu \int_\Omega \nabla v : \nabla \psi \, dx - \nu \int_\Gamma \partial_n v \psi \, dS(x) \\ &= \nu(\nabla v, \nabla \psi) - \nu \langle \partial_n v, \psi \rangle, \\ (\nabla p, \psi) &= -\int_\Omega p \, (\nabla \cdot \psi) \, dx + \int_\Gamma p \, n \cdot \psi \, dS(x) \\ &= -(p, \nabla \cdot \psi) + \langle p \, n, \psi \rangle.\end{aligned}$$

Here, we denote L^2-pairings on Ω by $(\cdot, \cdot)$ and on Γ by $\langle \cdot, \cdot \rangle$. Collecting all domain integrals yields the functional a and a right hand side f with

$$a(u, \phi) := \nu(\nabla v, \nabla \psi) + ((v \cdot \nabla)v, \psi) - (p, \nabla \cdot \psi) + (\nabla \cdot v, \xi),$$

whereas the boundary integrals define a functional c with

$$c(u, \psi) := -\nu \langle \partial_n v, \psi \rangle + \langle pn, \psi \rangle.$$

This gives a weak formulation of the stationary Navier-Stokes equation

$$a(u, \phi) + c(u, \psi) = (f, \psi) \ \ \forall \ \phi, \tag{4}$$

where the boundary conditions are not yet included. Dirichlet conditions are now enforced weakly using Nitsche's method by penalty-like terms and additional terms that maintain the skew-symmetry of the Stokes operator [1]. The skew symmetric counter part of c is

$$\hat{c}(v, \phi) := -\nu \langle \partial_n \psi, v \rangle - \langle \xi n, v \rangle.$$

We use a Taylor-Hood setting and choose biquadratic finite element functions for the discrete velocity space V_h and bilinear finite element functions for the discrete pressure space Z_h. Adding the skew-symmetrization $\hat{c}(v, \phi)$, the penalty term $\nu \frac{\gamma_1}{h} \langle v, \psi \rangle + \frac{\gamma_2}{h} \langle v \cdot n, \psi \cdot n \rangle$, and the inflow stabilization $-\langle (v \cdot n)^- v, \psi \rangle$ on the left, where $(t)^- = \min\{t, 0\}$, and compensating this by adding $\hat{c}(g, \phi) + \nu \frac{\gamma_1}{h} \langle g, \psi \rangle + \frac{\gamma_2}{h} \langle g \cdot n, \psi \cdot n \rangle - \langle (g \cdot n)^- g, \psi \rangle$ on the right gives the final Nitsche formulation of the stationary Navier-Stokes equation (4):

$$\begin{aligned}&a(u_h, \phi) + c(u_h, \psi_h) + \hat{c}(v_h, \phi_h) + \nu \frac{\gamma_1}{h} \langle v_h, \psi_h \rangle + \frac{\gamma_2}{h} \langle v_h \cdot n, \psi_h \cdot n \rangle - \langle (v_h \cdot n)^- v_h, \psi_h \rangle \\ &= (f, \psi_h) + \hat{c}(g, \phi_h) + \nu \frac{\gamma_1}{h} \langle g, \psi_h \rangle + \frac{\gamma_2}{h} \langle g \cdot n, \psi_h \cdot n \rangle - \langle (g \cdot n)^- g, \psi_h \rangle \ \ \forall \ \phi_h \in V_h \times Z_h.\end{aligned} \tag{5}$$

Here, h denotes the local mesh size. This formulation is consistent in the sense that the solution of the Navier-Stokes equations satisfies the variational problem. Further, convergence in the case of the Stokes system (i.e., without the nonlinear convective terms)

can be seen from the fact that for sufficiently large $\gamma_1, \gamma_2 > 0$, the penalty terms make the $(1,1)$-block coercive on the affine subspace of weakly div-free FEM functions. The formulation (5) was used in [1] only on edges of Cartesian grid cells, where boundary conditions could be enforced also in the classical way. We, in contrast, apply (5) to general non-conforming cases, where the cells boundaries do not represent Γ (see Sect. 6 for numerical results).

5 CUT-CELL INTEGRATION AND CURVE INTEGRALS

In the following we present a general method that allows the computation of complex integral terms such as in (5) on Ω in a toolbox manner, where the domain boundary Γ does not coincide with grid edges. These methods can also be used for other application fields (e.g. structural mechanics) within the Sundance toolbox.

For the operator evaluation in (5), Nitsche's method poses two challenges: the quadrature of basis functions or their respective derivatives over a subdomain of a cell that is intersected by boundaries and curve or surface integrals over the boundary itself. We have implemented solutions for both in Sundance. For the subdomain quadrature, we apply cut-cell integration, an efficient combination of geometry adaptivity and function adaptivity. We illustrate this problem in Fig. 3, where integrals must be computed only on Ω. In addition, we also implemented the curve and surface integrals in Sundance, which can be done in a straight-forward manner on structured forms such as polygons.

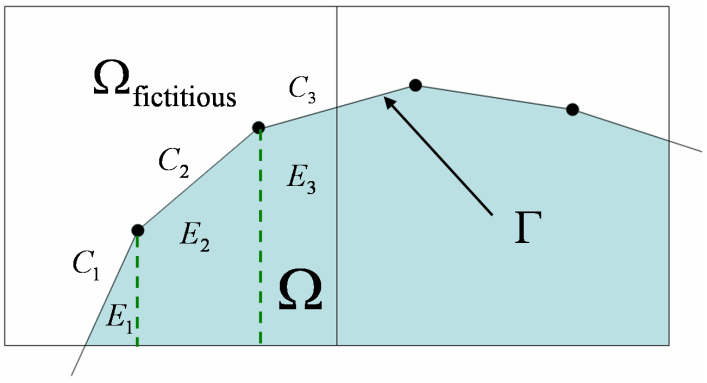


Figure 3: consider the left quad cell E intersected by the polygon. The operators should be evaluated only in the domain $E \cap \Omega$, the fictitious domain $\Omega_{fictitious} = E - \Omega$ is not part of Ω. The curve integrals have to be calculated on the polygon, which represents the boundary $\Gamma = \partial\Omega$.

The basic idea is to decompose such a cell E that is intersected by the polygon into smaller subcells $E_i, i = 1, \ldots, M_E$, (e.g. quadrilaterals and triangles) until $E \cap \Omega$ is represented exactly by $\cup_{i \in I_E} E_i$, $I_E \subset \{1, \ldots, M_E\}$. This is always possible in the case of polygons, as it is shown for the example in Fig. 3, where $E \cap \Omega$ is represented by the triangle E_1 and the quadrilaterals E_2 and E_3. These elements can be integrated up to numerical accuracy. Now, given quadrature points $x_{i,j}$ and weights $\omega_{i,j}$ on the subcells

$E_i \subset \bar{\Omega}$ and a given function f, it holds

$$\int_{E\cap\Omega} f(x)dx \approx \sum_{i\in I_E} \int_{E_i} f(x)dx \approx \sum_{i\in I_E} \sum_{j=1}^{N_i} \omega_{i,j} f(x_{i,j}).$$

This provides a first quadrature rule on $E \cap \Omega$. In a second step, we reduce the number of quadrature points, such that we can use a constant number of quadrature points for each cell E from the grid that is cut by the polygon. This property is necessary to vectorize the assembly process of the system matrix, which is done in the Sundance toolbox.

To achieve this, we consider a further set of geometry-independent points $p_k \in E$, $1 \le k \le K$, which are the quadrature points for cell E. Then the integration of the Lagrange polynomials l_k results in a quadrature rule $\int_{E\cap\Omega} f(x)dx \approx \sum_{k=1}^{K} w_k f(p_k)$ with precomputable weights $w_k = \sum_{i\in I_E} \sum_{j=1}^{N_i} \omega_{i,j} l_k(x_{i,j})$. Here, it is important that the quadrature rules on the E_i integrate all $l_k, k = 1, \ldots, K$ exactly. For each cell E that is cut by the polygon, we will precompute a set of special weights $w_k, k = 1, .., K$, which will be stored in the grid. However, if the geometry is changed, all these weights need to be recomputed. The advantages of the proposed method are the separation of geometry and function, such that precomputations based only on the geometry are possible.

According to the method described above, we integrate only over $E \cap \Omega$, which potentially could induce a numerical singularity in the system matrix if the area of $E \cap \Omega$ is numerically zero. To avoid this, we provide the possibility to choose scalings $\alpha_1, \alpha_2 \ge 0$ and to work with modified weights w_k^m that approximate $\alpha_1 \int_{E\cap\Omega} f(x)\,dx + \alpha_2 \int_{E\setminus\Omega} f(x)\,dx$. In this case, the weights are calculated in the following way:

$$w_k^m = (\alpha_1 - \alpha_2) w_k + \alpha_2 \sum_{j=1}^{N} \omega_j l_k(x_j) = (\alpha_1 - \alpha_2) w_k + \alpha_2 \int_E l_k(x)\,dx,$$

where on the cell E, the Lagrange polynomial l_k is integrated exactly with N quadrature points. In this formula on cell E, Ω has a weight of α_1, and $\Omega_{fictitious}$ is weighted by α_2. In our computations, we set $\alpha_1 = 1.0$ and $\alpha_2 = 1.0 \cdot 10^{-7}$.

For the curve integral within a cell E, we use line quadrature points for each piecewise linear portion of Γ. We denote the curve with C, and we are interested in the curve part $E \cap C$, where $E \cap C = \cup_{i\in J_E} C_i$. Each C_i represents one line segment. The curve integral of the function f is

$$\oint_{E\cap C} f(x)\,dS(x) = \sum_{i\in J_E} \oint_{C_i} f(x)\,dS(x) = \sum_{i\in J_E} \sum_{j=1}^{Q} w_{i,j}^C f(x_{i,j}).$$

The weights $w_{i,j}^C$ for the curve integrals are scaled with the length of the line, and similar to the special weights w_k^m, they are also stored in the grid.

6 NUMERICAL RESULTS

To demonstrate the capability of Nitsche's method in fluid mechanics, we set up the stationary DFG benchmark problem 2D-1 described in [7]. We implemented (5) using the quadrature methods described above. The implementation of such complex formula is surprisingly easy within the Sundance toolbox, once the weak form of the equations is defined.

The stationary scenario consist of a channel flow and a cylinder obstacle in the middle. We discretized the cylinder by a polygon with 63 points. In the computations, as described in the previous section, we use the polygon for cut-cell and curve integration.

The benchmark values are the lift and drag coefficients. The underlying lift and drag forces can be computed in different ways. The first approach uses the curve integral over the boundary C of the cylinder [7],

$$F_\phi = \oint_C \phi \cdot \sigma(u) \cdot n_C \, dS(x),$$

where $\sigma(u) = 2\nu\varepsilon(v) - pI$ is the stress tensor, $\varepsilon(v) = \frac{1}{2}(\nabla v + \nabla v^T)$ is the strain tensor, and ϕ is a unit vector pointing in flow direction (here, $\phi = (1,0)^T$) for the drag and perpendicular to the flow direction (here, $\phi = (0,1)^T$) for the lift. Further, n_C denotes the outward unit normal vector of the cylinder. The alternative approach is described in [2]. It transforms the curve integral into a domain integral. This approach proves to be more stable, since it will not be influenced by local errors on the boundary:

$$F_\phi = -\int_\Omega \left[((v \cdot \nabla)v - f) \cdot \Phi - p\nabla \cdot \Phi + 2\nu\varepsilon(v) : \varepsilon(\Phi)\right] dx,$$

where Φ is a smooth extension of the vector ϕ such that $\Phi|_C = \phi$ and $\Phi|_{\partial\Omega\setminus C} = 0$.

For the benchmark calculations, we use an adaptive Cartesian grid, which is refined one level further around the cylinder obstacle than in the inner part of the domain. Figure 5 shows the grid structure for a grid with 221×42 cells.

One important aspect is the choice of the parameters γ_1 and γ_2 that weight the stabilization terms. We calculated the benchmark with several parameter values and resolutions. We made two different convergence analysis for $\gamma_1 = \gamma_2 = 10^3$ and for $\gamma_1 = \gamma_2 = 10^4$. For each setup,we computed the lift and drag coefficients with the described approaches—once with curve integrals and once with domain (or cell) integrals. For the first case ($\gamma_1 = \gamma_2 = 10^3$), only the domain integrals converge to the reference interval, the lift value of the curve integral for the finest resolution still has around three percent relative error.

In the second case when $\gamma_1 = \gamma_2 = 10^4$, we get good convergence for both lift and drag calculation approaches. We also notice that with these higher penalty values $\gamma_1 = \gamma_2 = 10^4$, the domain integral delivers the same results as in the previous case. This is due to the local influence of the Nitsche's terms in the fluid domain. We can conclude here

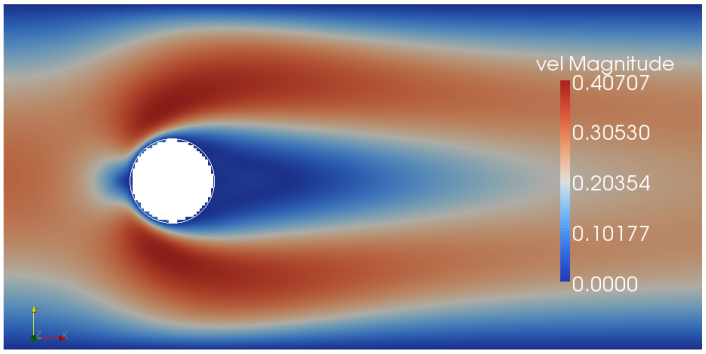


Figure 4: Velocity field magnitude for the 2D-1 stationary benchmark. The white polygon represents the cylinder obstacle. The picture shows only the left part of the flow channel.

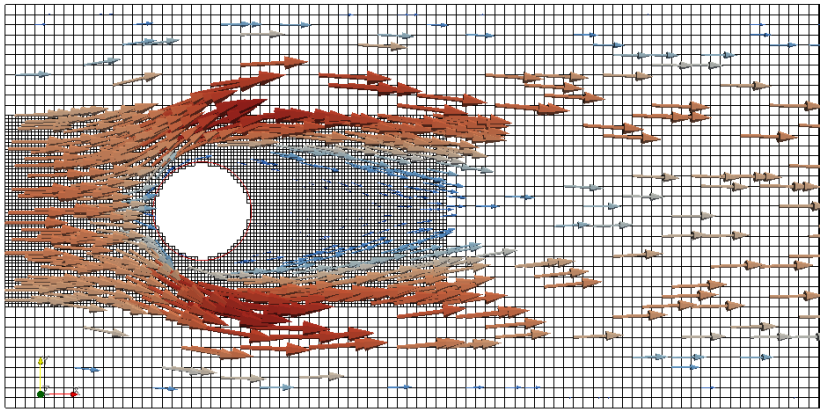


Figure 5: The velocity field for the 2D-1 stationary scenario benchmark. The red curve is the polygon, which represents the obstacle. The grid has a resolution of 221×42 and is refined around the cylinder.

that the domain integral gives more stable results for the drag and lift coefficients, and this approach is recommended for lift and drag computations unless the boundary forces themselves are needed.

7 CONCLUSION AND OUTLOOK

We showed that Cartesian grids are a very memory-efficient alternative to unstructured grids and at the same time offer full grid adaptivity and the possibility to accurately and efficiently represent complex and changing geometries as needed in shape optimization

$\gamma_1 = \gamma_2 = 10^3$	DragV	LiftV	DragC	LiftC
111×21	5.57919	0.012239	5.56063	0.0152477
221×42	5.57935	0.010597	5.57857	0.0098645
441×81	5.57935	0.010622	5.58070	0.0102580
Reference values [6]	5.579535	0.0106189	5.579535	0.0106189
Reference intervals [7]	$5.57 - 5.59$	$0.0104 - 0.0110$	$5.57 - 5.59$	$0.0104 - 0.0110$
$\gamma_1 = \gamma_2 = 10^4$	DragV	LiftV	DragC	LiftC
111×21	5.57936	0.0120811	5.57367	0.0171845
221×42	5.57936	0.0105961	5.57805	0.0105291
441×81	5.57936	0.0106214	5.57801	0.0110584
Reference values [6]	5.579535	0.0106189	5.579535	0.0106189
Reference intervals [7]	$5.57 - 5.59$	$0.0104 - 0.0110$	$5.57 - 5.59$	$0.0104 - 0.0110$

Table 1: Results of the 2D-1 benchmark computations. The values in the columns "DragV" and "LiftV" are the drag and lift coefficients computed with the domain integrals on Ω. The values in the columns "DragC" and "LiftC' are the drag and lift values computed with by the direct curve integrals on the boundary C of the cylinder.

and multiphysics problems. The next steps will be to test the implemented methods for multiphysics—in particular fluid-structure interaction—scenarios and to combine them with optimization algorithms. Furthermore, the implementation of matrix-free solvers in Sundance will substantially contribute to the exploitation of the full saving potential of structured adaptive Cartesian grids. Finally, methods developed in our Cartesian grid setting within Sundance profiting from the fast prototyping features of Sundance can be ported to high performance computing frameworks such as Peano [8] to further increase performance and parallel scalability.

Acknowledgements

The financial support of the Institute for Advanced Study (IAS) of the Technische Universität München and project 'B7:A High-End Toolbox for Simulation and Optimisation of Multi-Physics PDE models' within the 'Munich Centre of Advanced Computing' (www.mac.tum.de) is thankfully acknowledged. Part of the computations were done on a Linux cluster that is partially funded by the Deutsche Forschungsgemeinschaft.

REFERENCES

[1] R. Becker. Mesh adaptation for Dirichlet flow control via Nitsche's method. *Communications in Numerical Methods in Engineering*, 18(9):669–680, 2002.

[2] M. Giles, M. G. Larson, J. M. Levenstam, and E. Süli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. Technical Report NA-97/06, Oxford University Computing Laboratory, 1997.

[3] K. Long. Sundance 2.0 Tutorial, 2004.

[4] K. Long, R. Kirby, and B. van Bloemen Waanders. Unified embedded parallel finite element computations via software-based Fréchet differentiation. *SIAM Journal on Scientific Computing*, November 2010.

[5] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universitt Hamburg*, 36:9–15, 1971. 10.1007/BF02995904.

[6] R. Rannacher C. Waguet R. Becker, M. Braack. Fast and reliable solutions of the navier-stokes equations including chemistry. *Computer and Visualization in Science*, (2,3), 1999.

[7] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. In *Flow simulation with high-performance computers. Bd. 2.*, volume 52 of *Notes on numerical fluid mechanics*, pages 547–566. Vieweg, Braunschweig, January 1996.

[8] T. Weinzierl. *A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids*. Verlag Dr. Hut, 2009.

[9] T. Weinzierl and M. Mehl. Peano – a traversal and storage scheme for octree-like adaptive cartesian multiscale grids. *SIAM Journal on Scientific Computing*, 2011. accepted.