Dynamic Two-Way Parallelization of Non-Intrusive Methods for Uncertainty Quantification

C. Thiem and M. Schäfer

Abstract This work deals with a two-way parallelization method of numerical flow simulations under uncertainties with sampling methods for the uncertainty quantification. In general, for such simulations, there are two possibilities to distribute the computation on multiple processors in order to reduce the overall computing time. One approach is to divide the flow domain into several blocks which can be calculated by different processors. The other is to solve the various independent deterministic problems that arise from the sampling method for uncertainty quantification in parallel. Both methods have advantages and disadvantages and can be applied simultaneously, depending on the provided number of processors for the simulation. The presented method assigns the available processors dynamically to both parallelization methods during the simulation of the uncertain flow problem. The aim is to reduce the overall computing time compared to a static parallelization strategy.

1 Introduction

In computational fluid dynamics (CFD), even simple problems can not be computed analytically. For this reason over the last decades, efficient numerical methods have been developed. The continuous improvement of these algorithms combined with increasing compute power makes it possible that the problems to be simulated become more complex and more realistic. As a result, the subject of parameter uncertainty got into the spotlight of interest.

In this research field, parameters are not assumed to be exactly known because as a matter of fact quantities of physical processes are subject to certain fluctuations. A reason for these uncertainties might be natural fluctuations such as the outdoor temperature or atmospheric pressure. There are also human-made fluctuations, e.g.

C. Thiem (✉) • M. Schäfer
Institute of Numerical Methods in Mechanical Engineering, Graduate School Computational Engineering, Technische Universität Darmstadt, Dolivostr. 15, D-64293, Darmstadt, Germany
e-mail: thiem@gsc.tu-darmstadt.de; schaefer@fnb.tu-darmstadt.de

M. Mehl et al. (eds.), *Recent Trends in Computational Engineering - CE2014*,
Lecture Notes in Computational Science and Engineering 105,
DOI 10.1007/978-3-319-22997-3_6

certain tolerances in manufacturing processes, so that products are always slightly different. With the so-called non-intrusive methods for uncertainty quantification, it is possible to use highly developed common flow solvers and additionally take into account uncertainties. These methods divide a problem with uncertainties into a set of deterministic problems, which are represented by the so-called sample points. For each sample point, a CFD simulation has to be executed. Thereby, the computational effort increases tremendously such that all opportunities should be used to shorten the computational time. A very efficient approach to achieve an acceptable computing time is the parallel usage of multiple processors. In general, there exist two different parallelization methods for this kind of problems, on the one hand the domain parallelization [8] of the CFD problem and on the other hand the simultaneous calculation of the independent sample points [9].

This work presents a method that dynamically allocates processors during, the simulation of flow problems under uncertainties with these two separate parallelization methods. Additionally, an adapted optimization method for the initial solution of each sample point is applied.

A precondition for this method is that a sampling method for the uncertainty quantification of the flow simulation is used and that the flow problem is steady.

To explain the basic idea of the dynamic two-way parallelization, the theoretical fundamentals are described in the next section. Section 3 deals with the method for the optimization of the initial solution of the sample points. Subsequently, the concept of the two-way parallelization method is explained in more detail before the performance of the method is examined for test cases.

2 Theoretical Fundamentals

This section deals with theoretical basics that are necessary for the understanding of the main topic of this work. First, a general outline of flow problems under uncertainty and the numerical modeling is given. The following section explains the quantification of the stochastic system responses due to uncertainties. Furthermore, the two underlying parallelization strategies are explained in detail. Finally, the influence on the initial solution on the computing time of a steady CFD simulation is discussed.

2.1 Flow Problems Under Uncertainty

In this work we focus on incompressible, steady and laminar flow problems. These flows can be modeled by the Navier-Stokes equations [8]. The corresponding

equations for the conservation of mass and momentum can be written as:

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{1}$$

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} - \mu \frac{\partial^2 u_i}{\partial^2 x_j} = -\frac{\partial p}{\partial x_i} + \rho f_i, \tag{2}$$

where x_i is the ith component of the position vector, u_i is the ith component of the flow velocity and p is the pressure. The fluid density and dynamic viscosity are represented by ρ and μ. f_i describes the body force in the ith direction. In order to close the equation system, proper boundary conditions have to be applied. For simplicity, we write the governing PDE system in the general form

$$\begin{aligned} \mathscr{A}(\boldsymbol{\phi}(\mathbf{x}), \mathbf{x}) &= f(\mathbf{x}) \qquad \text{in } D, \\ \mathscr{B}(\boldsymbol{\phi}(\mathbf{x}), \mathbf{x}) &= g(\mathbf{x}) \qquad \text{in } \partial D, \end{aligned} \tag{3}$$

where $\mathscr{A}$ is the differential operator which also includes all parameters such as fluid properties. $\boldsymbol{\phi}$ describes the flow field solution. The boundary conditions operator is denoted by $\mathscr{B}$. D is a spatial domain in $\mathbb{R}^d$ with its boundary ∂D. $\mathbf{x}$ refers to the elements of D. The forcing term and the boundary conditions are denoted by f and g, respectively. For numerical simulations, we solve this equation system with finite-volume method (FVM) based solvers. The most important aspects of the FVM in the context of this work is explained in Sect. 2.4. For a more detailed description of this method, we refer, e.g., to [8].

The formulation (3) assumes that all system parameters are exactly known, but actually in quantities of physical processes there is always a certain amount of fluctuation. Uncertainties in simulations can also arise from a lack of knowledge, if some effects are too complicated, such that values can be estimated only approximately.

In order to account uncertainties in flow simulations, the system (3) has to be extended to a stochastic system of the form

$$\begin{aligned} \mathscr{A}(\boldsymbol{\phi}(\mathbf{x}, \omega), \mathbf{x}, \omega) &= f(\mathbf{x}, \omega) \qquad \text{in } D \times \Omega \\ \mathscr{B}(\boldsymbol{\phi}(\mathbf{x}, \omega), \mathbf{x}, \omega) &= g(\mathbf{x}, \omega) \qquad \text{in } \partial D \times \Omega. \end{aligned} \tag{4}$$

The stochastic system (4) includes the random parameter space (RPS) Ω, which contains possible realizations for the uncertain parameters. These realizations can be mapped to the random parameter ω. The uncertain system parameters result in a stochastic system output which can be described with methods from the field of uncertainty quantification (UQ). A brief general description of UQ is given in Sect. 2.2. A more detailed overview and description of these methods are given by [2, 6, 9].

2.2 Uncertainty Quantification

The system response of a system with uncertain input parameters is stochastic in nature. As a consequence, the quantities of interest can no longer be represented by a nominal value, but with a probability density function (PDF). In case of numerical simulation, special methods must be applied, which normally results in a sharp increase of the computational effort. This subsection provides a brief description of such methods.

Due to the fact that in addition to the spatial domain D the RPS Ω exists, the former dimension of the system increases by the number of uncertain parameters m. The stochastic response of the system can be characterized with methods from the field of uncertainty quantification. Methods for UQ can be categorized into two classes. On the one hand there are non-intrusive methods, which only need a set of solutions of the deterministic problem and on the other hand there are intrusive methods, which require a reformulation of the original model to a stochastic version. In this work we focus only on non-intrusive methods. For a detailed description of intrusive methods, we refer to [6].

Sampling methods belong to the non-intrusive methods. Although there are a variety of different methods, they are similar in the general procedure:

1. Choose a set ξ of P realizations of ω.
2. Perform a deterministic simulation for each $\omega \in \xi$.
3. Calculate statistics of the stochastic system output.

A major advantage of this method is that, due to the decomposition of the stochastic problem into a finite number of deterministic problems, common well-developed flow solvers can be used. A disadvantage is that, depending on the required accuracy, a lot of simulations are necessary. Thus, the solution of the stochastic problem is computationally highly intensive. However, the deterministic problems can be calculated independently of each other, whereby an effective parallelization is possible [9]. In addition, the fact of independence builds the basis of the method that is presented in Sect. 3. As a concrete example of a sampling method, the Monte-Carlo Simulation (MCS) is presented. Furthermore, the MCS is used for UQ in the course of this work. In principle, other sampling methods could be used, such as the in many cases much more efficient stochastic collocation.

2.2.1 Monte-Carlo Simulation

A very widespread sampling method for UQ is the Monte-Carlo simulation (MCS). Its key idea is to create a set ξ of pseudo-random numbers representing the uncertain input. Once the deterministic solutions are calculated, statistics can be estimated using standard formulations as given in [2] such as for the expectation value of the

system output $\bar{y}$,

$$\bar{y} = \frac{1}{P} \sum_{j=1}^{P} y(j) \tag{5}$$

and the variance σ^2

$$\sigma^2 = \frac{1}{P-1} \sum_{j=1}^{P} (y(j) - \bar{y})^2, \tag{6}$$

where P is the number of sample points and $y(j)$ is the corresponding system output of the jth sample point. Thus, the method is the numerical equivalent of the approach of an experimental determination of the statistical quantities. The algorithm is easy to implement, but has the drawback of a low convergence rate of $1/\sqrt{P}$. Although this formula is independent of the dimensions, the required number of sample points P increases disproportionately with the number of uncertain parameters, when a required accuracy of the statistics should be achieved. Therefore, there is an indirect dependency in practice. In this paper, the accuracy is of minor relevance, so the MCS is still suitable because of the simplicity. The independence of the solutions $y(j) \in 1..P$ of the sample points can be seen from the formulas of the statistical quantities (5) and (6). The only requirement is that at the time of the calculation of the mean or standard deviation, all solutions must be available. The order in which these are calculated is of no importance.

2.3 Parallelization Strategies

If multiple CPUs are available, parallelization methods make it possible to divide simulations into several parts in order to calculate them simultaneously. Thereby, the computation time can be significantly reduced. However, parallelization overheads occur. A measure to determine the performance of parallelized computations is the speed-up S_P [8], which is defined as:

$$S_P = \frac{T_1}{T_P}, \tag{7}$$

where T_P is the total computing time with P CPUs. The nature of the losses essentially depends on the used parallelization methods.

For the combination of uncertainty quantification through sample methods and flow solvers based on FVM there exist mainly two different approaches for parallelization. The first approach is the decomposition of the flow domain into certain subdomains which can be calculated by different processors simultaneously. This parallelization strategy requires data transfer between all processors in order to

get an adequate result of the overall flow field [8]. In order to reduce the amount of communication between the processors, the introduction of ghost cells has been established. Ghost cells are auxiliary control volumes along inner boundaries between two subdomains. The variable values of the ghost cells correspond to the outer control volumes of the subdomain from the other side of the boundary. An update of the values is done after a sufficient number of iterations. Although this method can reduce the communication effort, the number of iterations until the solver converges is increased due to the additional artificial inner boundaries. Another issue that is affecting the efficiency of parallelization is the load-balance. So, if one processor is overloaded because for instance it has to compute more grid points, all other processors have to idle until the overloaded processor has finished its calculation.

The second approach is the parallel simulation of the sample points, as these can be solved independently. This means, if the set contains P problems, the computation could be parallelized on up to P processors or clusters. In this case, neither communication between processors is necessary nor additional arithmetic operations are needed. The main cause for a possible loss of performance is the load-balancing. The problem occurs when there are more processors than sample points left. A further effect which leads to performance losses is, that the start of the calculations can not be done at the same time, but only sequentially. However, this effect is usually negligible.

In general, the performance loss of the first approach is higher, however, with this approach the time for a single CFD simulation can be reduced significantly. Nevertheless, both strategies can easily be combined. Thereby, it is possible to compute n sample points simultaneously with m processors. However, specifying n and m has a major influence on the computation time.

2.4 Influence of the Initial Solution

If the finite-volume method is applied to simulate the flow problem, an algebraic system of equations is formulated which has to be solved. Due to the complexity of the system, it is necessary to solve it with an iterative method. This procedure generates a sequence $\{\boldsymbol{\phi}_n\}_{n\geq 1}$ of improving approximations of the solution $\boldsymbol{\phi}$ until a convergence criterion is satisfied. The number of iterations until convergence, and hence the computing time, depends among others on the initial solution $\boldsymbol{\phi}_0$. The process takes less time the more accurate the choice of the initial solution $\boldsymbol{\phi}_0$ is. So, in case of an uncertainty quantification a good initial solution for the computation of a sample point would be a flow field of a previously calculated sample point. Normally, uncertainties are small, and thus the flow fields are similar to each other. In [10] it has been shown that in practice it is often the case that a solution of a sample point is more suitable as initial solution for another sample point, the more similar the values of the uncertain parameters are, because the flow fields probably differ less from each other.

3 Optimal Choice of the Initial Solution of Sample Points

This section describes a method to reduce the absolute computational time of the UQ by optimizing the choice of the initial solutions for the CFD problems of the sample points. The basic idea is that the skillful use of previously computed flow fields as initial solutions for further sample points can reduce the computing time. In [10] a method is presented that reduces the computational effort purely by optimizing the order in which the sample points are calculated. Assuming that the solution of the preceding sample point is used as an initial solution for the next sample point.

In this section, a variation of this method is proposed. The requirements of this variation of the optimization method are that it is assumed that a sampling method for the UQ is used as well as an iterative solver for the corresponding steady-state CFD-problems. Furthermore, it is assumed that the flow solver is able to store a calculated flow field on the hard disk drive and if needed to reload it as an initial solution for another CFD simulation. The last points are the main differences compared to the original method and also are responsible for the improvement.

3.1 Key Idea and Weighting

As a foundation for the optimal choice of an initial solution, the assumption is made that a solution of a sample point is more suitable as an initial solution for another sample point, the more similar the values of the uncertain parameters are, because the flow fields probably differ less from each other. Of course, this can not be guaranteed, especially when it is a highly non-linear problem, but in practice it proves to be an appropriate assumption [10]. A rough estimation of the similarity of two sets of uncertain parameters can be made through the Euclidean distance in the RPS. However, the uncertain parameters have not necessarily the same order of magnitude, so all parameters should be scaled to the range $[-1 + 1]$. In addition, same nominal changes in various scaled uncertain parameters can have a different effect on the flow field and thus on the computing time. For this reason, it can be advantageous to weight the parameters. An algorithmic estimation of these weighting factors $\mathbf{W}$ can be done before or during the uncertain simulation. The preliminary calculation has the disadvantage that additional CFD simulations have to be performed. Although the results can be re-used for the uncertainty quantification, the calculations are performed with inappropriate initial solutions which finally leads to performance losses. An online estimation of the weighting factors does not require any additional simulations, but a suitable estimation is only achieved after several calculated sample points. An overcompensation of the effort for the estimation of the weighting factors might arise only for problems with numerous sample points, or if the uncertainty quantification belongs to a flow optimization process, where in each iteration an UQ has to be performed. Besides scalar weighting factors that describe only a linear relationship between

the similarity of uncertain parameters and the computing time, neural networks can be used to describe more complicated relationships. Neural networks are especially suitable in combination with the online estimation of the weighting factors. In the following, it is assumed that an appropriate weighting $\mathbf{W}$ of the uncertain parameters exists.

3.2 Optimization Problem

The optimization aim is to minimize the overall computation time. This is realized via an optimized choice of the initial solutions of the sample points. In order to describe the problem, the distance S_j^i between a sample point ω^j and the corresponding sample point used for the initial solution ω^i is defined as:

$$S_j^i = \left\| \mathbf{W}(\boldsymbol{\omega}^j - \boldsymbol{\omega}^i) \right\| . \tag{8}$$

With these considerations and preconditions, the optimization problem can be described as follows:

$$\min_{\mathbf{a},\mathbf{b}} \sum_{i=2}^{P} S_{a_i}^{b_i}, \qquad \mathbf{a} \in \mathbb{N}^{*P}, \qquad b_i \in a_{1..i-1}, \tag{9}$$

where $\mathbf{a}$, $\mathbf{b}$ are index vectors to the sample points $\omega^1 \dots \omega^P$. This gives a nested minimization problem in which the objective is to optimize the sequences in $\mathbf{a}$ and $\mathbf{b}$. $\mathbf{a}$ contains the sequence in which the sample points have to be calculated and $\mathbf{b}$ contains the indices to the associated initial solutions. In [10] this problem was interpreted as an Open Traveling Salesman Problem, a well-studied NP-hard combinatorial optimization problem [5]. In this work, the perception is different because any precalculated flow field can be used as an initial solution. In the next section, a new method based on graph theory is presented.

3.3 Optimization Method

In order to solve problem (9), in the first step the sample points $\omega^1 \dots \omega^P$ of the RPS are interpreted as nodes $\mathbf{V}$ from a connected graph G. The costs $\mathbf{C}$ of the associated edges $\mathbf{E}$ are estimated with formula (8). Then, based on G a minimum spanning tree (MST) $G' = (\mathbf{V}, \mathbf{E}')$ is generated by the algorithm of Prim [3]. As a result, the MST G' contains only the shortest edges $\mathbf{E}'$ between all sample points. This suggests that an optimal initial solution for a sample point V_{a_i} or rather ω^{a_i} must be connected to it by an edge of $\mathbf{E}'$. This provides the basis for the algorithm to determine the sequences of $\mathbf{a}$ and $\mathbf{b}$ to solve (9). The first element a_1 can be an arbitrary index

of a sample point, obviously without a particular initial solution b_1. The next point in the sequence must be connected to sample point a_1 via an edge out of $\mathbf{E}'$ such that $b_2 := a_1$. The third point must be connected to a_1 or a_2. According to this principle, the sequence of **a** and the associated initial solutions **b** can be determined. The solutions of **a** and **b** are not unique, but equivalent. For comparison, within the TSP approach the sequence **b** is determined with the rule $b_i := a_{i-1}$. It follows that for the TSP only the last flow field has to be stored. However, each point can only be used once as an initial solution, thus, a bigger middle distance between the initial solutions and sample points is to be expected. Results of a numerical test are shown in Sect. 5.3.1.

It should be noted that the above method is designed for processes without a sample point parallelization, but it is easily extendable for that case.

4 Dynamic Two-Way Parallelization

The aim of the dynamic two-way parallelization method is to exploit the strengths of each parallelization method from Sect. 2.3 adapted to a present situation during the UQ. The valuation of the situation is based on the quality of the available initial solutions for the remaining sample points. For this, the previously presented method of the MST is suitable. Therefore, it is assumed that, before using the two-way parallelization, an MST from the sample points of the RPS has been prepared.

The idea is that if there are multiple sample points with well-suited initial solutions, the sample point parallelization is preferred, because of the higher efficiency. If there are more CPUs than problems with well-suited initial solutions, the domain parallelization is additionally applied to these problems. Only if further CPUs remain sample points with poor initial solutions have to be calculated, where the domain parallelization is used primarily. The reason for this is that this calculation requires a higher computational effort, so that more time can be saved with this parallelization method.

In order to describe the algorithm of the two-way parallelization method, two arrays are introduced. The array **Q** contains all indices of sample points that have not yet been calculated and for which no flow field of a sample point exists that is connected by an edge of $\mathbf{E}'$. In addition, the array is sorted in descending order, with respect to the number of edges that a corresponding sample point has. The second array, **R**, contains all indices of sample points that have not yet been calculated, but for which a suitable predecessor has already been calculated. At the beginning of the algorithm **R** is empty and **Q** contains the indices of all sample points. Assuming that m processors are available and a single sample point can be calculated with up to p_{max} processors, the overall computation starts with the simultaneous calculation of the first $\lceil m/p_{\max} \rceil$ points of **Q**. Each of these problems is parallelized with p_{max} processors (if the number of processors does not fit, only the last problem will get fewer processors). The related indices are removed from **Q**. The idea behind this is, that the first simulation without a good initial solution is the most time consuming

one. All the following calculations converge much faster. So if the time for the first calculation is reduced, the computation time of the entire UQ can also significantly be affected. This reduction can be achieved only with the domain parallelization strategy. After initialization, the algorithm works according to the following scheme until all sample points are calculated:

1. Move all indices from **Q** to **R** whose associated sample points have a common edge with the just calculated sample points ω_i.
2. If **R** is not empty, distribute as many as possible idle processors uniformly to the sample points in **R** in order to simulate the underlying problems.
3. If there are still idle processors and **Q** is not empty, use them to calculate the first element of **Q**.
4. If there are still idle processors, use them to support a running computation through a dynamical change of domain partitioning.
5. Delete all indices of sample points from **Q** and **R** which are already calculated.

Steps 1 and 5 ensure that the arrays are updated in order to be able to evaluate the quality of the initial solutions. If the condition in step 2 holds, both parallelization methods with a preference for the sample point parallelization are combined. The maximum number of CPUs that can be used for this step is limited by the number of elements in R multiplied with maximum number of domain partitions p_{max} of the flow problem. The next condition ensures that rather points with poor initial solution are calculated than a CPU is idle. Step 4 is only relevant towards the end when all sample points have been calculated or are currently being calculated. Additionally, between steps 1 and 2 it is estimated whether it is worthwhile to wait for the completion of a running simulation. The reason is that the waiting time may be overcompensated by the skillful use of the free CPUs or the new initial solution. The estimate is based on a neural network that utilizes the information of previous simulations.

5 Numerical Test Cases

In order to analyze the efficiency for the presented optimization method for the initial solutions and the dynamic two-way parallelization, we investigate a 2D-flow around a NACA 4-digit series airfoil at low Reynolds numbers with various configurations.

The section begins with a brief description of the used numerical framework. Afterwards, the CFD test case of the airfoil is explained in more detail. Finally, the numerical results of various test cases are presented. The first test case deals with the efficiency of the method for the optimal choice of the initial solutions (see Sect. 3) compared to a random choice and the method from [10]. Subsequently, two results are shown for the dynamic two-way parallelization under various conditions. The investigations primarily concern the computing time. The different parallelization

methods have no influence on the result of the UQ, therefore no values for stochastic quantities are given.

5.1 Numerical Framework

As a flow solver, we utilize our in-house flow solver FASTEST [4]. The software uses a fully conservative FVM in order to discretize the incompressible Navier-Stokes equations [8]. The discretized system is solved by an iterative pressure-correction procedure with an SIP solver for inner iterations. The communication of the parallelization of the flow domain decomposition is done via the Message Passing Interface. For UQ with MCS, the sample points are created with a pseudo-random number generator from MATLAB [7]. The implementation of the dynamic two-way parallelization including the generation of the MST is also written in MATLAB.

5.2 Problem Description

The basis of the test case is a 2D flow around a NACA 5320 airfoil with an angle of attack of 0° at a Reynolds number of $Re = 100$. The flow region and the detail of the airfoil are shown in Fig. 1. At the curved inlet and on the airfoil surface Dirichlet boundary conditions are applied. Neumann boundary conditions are used at the outlet. For this purpose, it is ensured that the distance between the outlet and the airfoil is sufficiently large. A mirror boundary condition is applied at the upper and lower boundaries. For this it is guaranteed that near to these areas the fluid flows only in x-direction. The domain has a total length of 32m and a height of 24m. The distance between the leading and trailing edge of the airfoil is 1m. The fluid is imaginary with a density $\rho = 1\text{kg/m}^3$ and a dynamic viscosity $\eta = 1\text{kg/(ms)}$. The flow field is discretized with 30,720 control volumes, which are distributed in 10 structured blocks. The test case has five uncertain parameters of which the first three are related to the shape of the NACA airfoil. These are the maximum camber position c_{pos}, the maximum camber c_{max} and the thickness t of the airfoil as defined in [1]. The other two uncertain parameters are the inlet velocity v_{in} and the angle of attack α. Furthermore, the amount of uncertainty is specified in percent of the total domain of the corresponding uncertain parameter. The upper and lower values of these domains are shown in Table 1.

5.3 Results

This subsection includes numerical results of three different test cases.

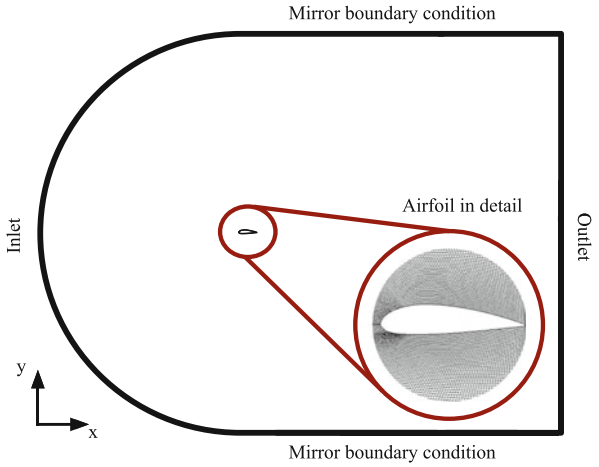

Fig. 1 Flow region and detail view of the NACA5320 test case

Table 1 Dimensionless minimum and maximum values and total range for the uncertain parameters

Parameter	c_{pos}	c_{max}	t	v_{in}	α
Min. value	0	0	0.01	50	−50
Max. value	0.9	0.95	0.4	150	50
Range	0.9	0.95	0.39	100	100

5.3.1 Test Case 1: Optimal Choice of Initial Solutions

In this setup the number of used sample points is varied (values 10, 50, 100, 150 and 200) in order to examine the influence on the efficiency of the methods for optimizing the choice for the initial solution of the sample points. The simulations are performed with one CPU. The results are shown in Fig. 2. The time is normalized with respect to the resulting time if no optimization method is used. TSP is the method which is proposed in [10] and MST the method from Sect. 3. It can be seen that the relative computing time decreases with increasing number of sample points, if the choice for the initial solution is optimized. Comparing TSP and MST it is notable that with the MST the computing time can be significantly more reduced. In case of 200 sample points, the time can be reduced to 64 % with the MST and to 83 % with the TSP. However, the MST requires a lot more hard disk space.

5.3.2 Test Case 2: Parallelization Methods with 15 CPUs

The focus of this test is to examine the performance of the dynamic two-way parallelization compared to static parallelization strategies. Therefore, the RPS contains 45 sample points. For the calculation of the uncertain flow simulation, 15 CPUs are available. A single sample point can be calculated with up to $p_{max} = 5$ CPUs. The results are shown in Fig. 3. The term **stat. *n* x *m* CPU** means that a static parallelization method is used, where n sample points can be calculated

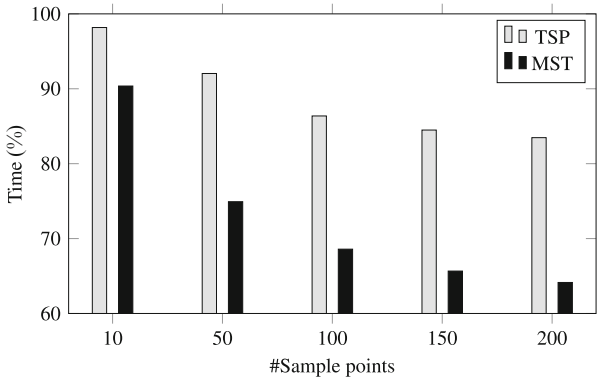

Fig. 2 Results of test case 1: Comparison of the minimum spanning tree method (MST) and the traveling salesman problem method (TSP) for the optimization of initial solutions with various numbers of sample points. The relative time on the y-axis is normalized with respect to the overall computing time if no optimization method is used

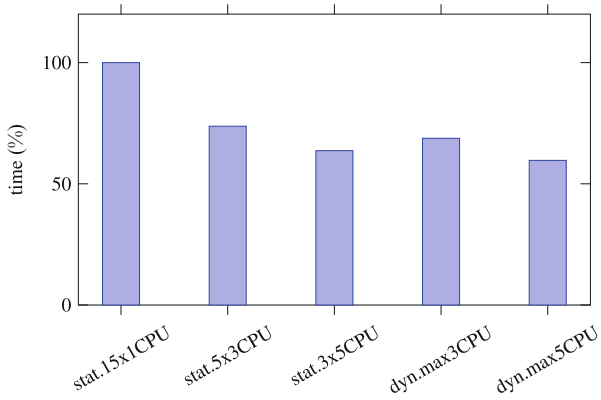

Fig. 3 Results of test case 2: Dependence of the relative computing time on different static and dynamic parallelization methods. The overall computing time is normalized to the resulting time of a static sample point parallelization method with 15 CPUs

simultaneously with m CPUs each. The term **dyn. max *m* CPU** means that the dynamic two-way parallelization is applied whereby a sample point is calculated with a maximum of n CPUs. The static methods also use the optimization of the initial solutions, but this does not change the number of CPUs for the domain parallelization. It is apparent that the methods stat.3x5CPU and dyn.max5CPU show the best performance. In Table 2, the ratio of good to poor initial solutions is illustrated. The relatively high number of poor initial solutions used by the dynamic

Table 2 Results of test case 2: Ratio of good to poor initial solutions of a static and dynamic parallelization method

Method	Good init.	Poor init.
Stat.3×5CPU	39	6
Dyn.max5CPU	30	15

Table 3 Results of test case 2: Overview of the distribution of the number of CPUs to the sample points for the dynamic two-way parallelization with a maximum of 5 CPUs

#CPU	#Sample points
1	21
2	5
3	4
4	4
5	11

method is the reason that the theoretical advantage is hardly effective. Further investigations revealed that the ratio of CPUs to sample points is responsible for this effect. Table 3 provides an overview of the distribution of the CPUs during the dynamic two-way parallelization.

5.3.3 Test Case 3: Parallelization Methods with 5 CPUs

In this test case 5 CPUs are used to compute 45 sample points with various parallelization methods, otherwise the test case is identical to test case 2. Due to the reduced number of processors, only three methods were used for parallelization. These are a pure sample point parallelization, a flow domain parallelization and a dynamic two-way parallelization with up to 5 CPUs per sample point. The relative computing times of each method can be found in Fig. 4. Additional information about the quality of the used initial solutions are shown in Table 4. The pure sample point parallelization requires the most time and is therefore the reference. The weakness arises in the first cycle, in which five sample points without a suitable initial solutions are computed in parallel. This step requires as much time as the calculation of the remaining 40 points. The first step of the pure domain parallelization requires only about one-fifth of that time and thus the total time can be reduced to 78 %. The dynamic method performs significantly better in this test case and requires only 66 % of the computing time. Only three sample points were started with poor initial solutions.

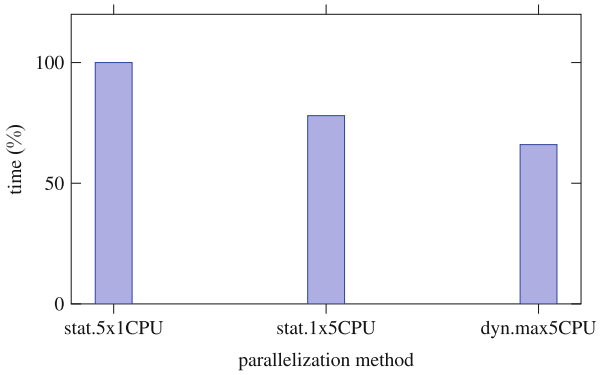

Fig. 4 Results of test case 3: Comparison of the relative computation time of different static and dynamic parallelization methods. The overall computing time is normalized to the resulting time of a static sample point parallelization method with 5 CPUs

Table 4 Results of test case 3: Ratio of good to poor initial solutions of each parallelization method

Method	Good init.	Poor init.
Stat.5x1CPU	40	5
Stat.1x5CPU	44	1
Dyn.max5CPU	42	3

6 Conclusion

We have presented a dynamic two-way parallelization method for numerical flow problems under uncertainty. First, necessary theoretical fundamentals were explained. Subsequently, a method for the optimal choice of initial solutions for sample points was introduced. A possible reduction of computing time due to this method has been demonstrated. This technique is further used within the dynamic two-way parallelization. Based on two test cases, it has been shown that the dynamic two-way parallelization method has significant potential to shorten the computation time compared to static parallelization methods. The major advantage of the dynamic two-way parallelization method is that it automatically provides a good balancing whereas the best static solution for a certain scenario can be found by trial-and-error, only. A major factor for the efficiency of the dynamic approach is the ratio of the number of available CPUs to sample points. The strengths of this kind of parallelization arise mainly with a large number of sample points.

A generalization of this method for non-stationary flows could be possible, but further investigation how to handle flow field information of time-steps of an initial sample point in order to reduce the computation time of a subsequent sample point is needed.

Acknowledgement This work is supported by the 'Excellence Initiative' of the German Federal within State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt.

References

1. Abbot, I.H., von Doenhoff, A.E.: Theory of Wing Sections, Dover, New York (1960)
2. Harzheim, L.: Strukturoptimierung, Harri Deutsch, Frankfurt (2008)
3. Horowitz, E., Sahni, S.: Fundamentals of Computer Algorithms. Computer Science Press, Mayland (1978)
4. Institut of Numerical Methods in Mechanical Engineering: FASTEST-Manual. Technische Universität Darmstadt, Darmstadt (2005)
5. Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York (1985)
6. Le Maitre, O., Knio, O.: Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics. Springer, Berlin (2010)
7. MATLAB: MATLAB version R2013a, The MathWorks Inc., Natick, MA (2013)
8. Schäfer, M.: Computational Engineering, Springer, Berlin (2006)
9. Schieche, B.: Unsteady Adaptive Stochastic Collocation Methods on Sparse Grids. Dr. Hut (2012)
10. Thiem, C., Schäfer, M.: Acceleration of Sampling Methods for Uncertainty Quantification in Computational Fluid Dynamics. Proceedings of the Ninth International Conference on Engineering Computational Technology (2014)