Performance of Preconditioners for Large-Scale Simulations Using Nek5000

N. Offermans, A. Peplinski, O. Marin, E. Merzari, and P. Schlatter

1 Introduction

The preconditioning of elliptic problems characterized by the propagation of information at infinite speed over the domain is a numerically challenging task. We study the case of the Poisson equation arising from the numerical resolution of the incompressible Navier–Stokes equations by operator splitting. We consider Nek5000, a code based on the spectral element method, as our framework. The current preconditioning strategy is based on an additive Schwarz method, which combines a domain decomposition method [5] and a so-called coarse grid problem [10]. The first step consists in solving directly local overlapping Poisson problems and is easily parallelizable. The second step corresponds to a Poisson-like problem over the whole domain and is hard to scale because of its relatively low number of degrees of freedom and the bottleneck induced by global communication.

A scalable solver for the coarse grid problem is critical to ensure strong scaling of the code. Existing strategies include a direct solution method similar to a Cholesky decomposition, called XX^T [14], and an algebraic multigrid (AMG) solver [6]. While the first choice works well for relatively small problems (typically $<100,000$ spectral elements on $<10,000$ cores), the second option is preferred for large scale simulations. The current AMG solver, which we will denote as the *in-house AMG*, is fast and scales well [11]. It has been shown that the use of AMG can speed up

N. Offermans (✉) · A. Peplinski · P. Schlatter
Linné FLOW Centre and Swedish e-Science Research Centre (SeRC), KTH Mechanics,
Stockholm, Sweden
e-mail: nof@mech.kth.se; adam@mech.kth.se; pschlatt@mech.kth.se

O. Marin · E. Merzari
Argonne National Laboratory, Lemont, IL, USA
e-mail: oanam@mcs.anl.gov; merzari@anl.gov

large-scale simulations by up to 10%. In addition, the XX^T has been designed for optimal performance on a number of cores which is a power of 2, whereas the AMG is insensitive to this parameter.

However, the AMG solver requires a setup phase, performed once for each mesh, by an external and serial code. Besides inducing an unwanted overhead, it also limits the use of the *in-house* AMG solver in the framework of mesh refinement, which is the main motivation for this work. Therefore, we propose to replace the *in-house* AMG by *BoomerAMG*, a parallel AMG solver for arbitrary unstructured grids from the *hypre* library for linear algebra [1, 4, 8]. *BoomerAMG* offers a number of parallel algorithms for the coarsening, interpolation and smoothing steps of the AMG setup, to accommodate various types of problems, meshes and architectures. The *BoomerAMG* solver will be tested in terms of scalability and time to solution.

Scaling tests for the *BoomerAMG* solver have been performed up to 4096 cores by Baker et al. [1]. Matrices arising from the finite element and finite difference discretizations of 2D and 3D scalar diffusion problems were considered. The authors used HMIS coarsening and extended+i interpolation and showed that l_1 -scaled Jacobi, l_1 -scaled Gauss-Seidel and Chebyshev smoothers are good choices for such problems.

Weak scaling up to 125,000 cores has been presented in Ref. [2], where *BoomerAMG* was used as a preconditioner for a conjugate gradient solver. The test case considered is that of a 3D Laplace operator. The parameters for the AMG solver were again HMIS coarsening, extended+i interpolation and symmetric hybrid Gauss-Seidel for the smoother. Aggressive coarsening with multipass interpolation was used on the finest grid, while the problem on the coarsest level was solved by Gaussian elimination. The authors show the impact of additional parameters such as the use of 64 bits for the integers or the use of an hybrid parallel strategy with OpenMP and MPI.

In the present work, we use the *BoomerAMG* from *hypre* to precondition a GMRES solver for the pressure equation arising from the spectral element discretization of the Navier-Stokes equations. We study strong scaling up to 131,072 cores on two different supercomputers: Mira, based on the IBM Blue Gene/Q architecture, and Hazel Hen, a Cray XC40 system. The first test case considered is the flow around a NACA4412 airfoil, which we use to identify a set of best parameters for the *BoomerAMG* solver. A second test case is employed for a strong scaling study: the turbulent flow in wire-wrapped pin bundles [3, 13].

The paper is organized as follows. In Sect. 2, we introduce the discretization method and describe the preconditioning strategy and the *hypre* library. In Sect. 3, we study which set of parameters gives the fastest time to solution for the problem at hand. Using those parameters, we perform a strong scaling study for the flow in wire-wrapped pin bundles in Sect. 4. We finish with conclusions and outlook in Sect. 5

2 Problem Description

Considering an operator splitting strategy to solve the Navier–Stokes equations, the consistent pressure p is the solution to a Laplace problem of the form $\Delta p = r$. In Nek5000, this equation is discretized using the spectral element method [7] and it has been shown that it is well preconditioned by an overlapping additive Schwarz method. The preconditioner combines local problems $\mathbf{R}_k^T \mathbf{A}_k^{-1} \mathbf{R}_k$ and a coarse grid problem $\mathbf{R}_0^T \mathbf{A}_0^{-1} \mathbf{R}_0$ and is expressed as

$$\mathbf{M}^{-1} = \mathbf{R}_0^T \mathbf{A}_0^{-1} \mathbf{R}_0 + \sum_{k=1}^K \mathbf{R}_k^T \mathbf{A}_k^{-1} \mathbf{R}_k ,$$

where $\mathbf{R}_0$ and $\mathbf{R}_k$ are restriction operators, $\mathbf{A}_k$ are local stiffness matrices and K is the total number of spectral elements. The matrix $\mathbf{A}_0$ corresponds to a Laplace operator defined on the element vertices only. Because of its low number of degrees of freedom and global extent, the scalability of the coarse grid problem is mostly limited by communication and latency. We note that the term “coarse grid” here refers to the fact that $\mathbf{A}_0$ is defined on the vertices of the spectral elements only. The problem is therefore “coarse” in comparison to the solution fields, which are expanded on the Gauss–Lobatto–Legendre points inside each spectral element (typically order 10 quadrature points in each direction). When talking about the different levels arising from the coarsening phase of the AMG setup, we will use the term “coarse level” to avoid confusion.

As mentioned before, the solver of choice for large problems is currently an *in-house AMG*, developed specifically for Nek5000 [6], whose main drawback is a setup phase by an external and serial code. The default option for the setup step of the *in-house AMG* is a Matlab code, which uses Ostrowski coarsening with norm bound, a diagonal Chebyshev smoother, applied on the second branch of the V-cycle only, and an energy-minimizing interpolation, all described in Ref. [9]. Other properties of the AMG include no smoothing on the finest level, a number of smoothing steps predefined for each level during the setup phase and a coarsest level made of one variable only. The good scalability of the *in-house AMG* is due to the fact that it automatically chooses, at run time, the fastest communication strategy at each level of the coarsening process, between three options: a pairwise exchange, a crystal router method or an allreduce operation. Previous work has shown that, when far from the strong scaling limit, the total time spent in the pressure solver is typically 85–90% of the total computational time, including the time spent in the coarse grid solver, which amounts to about 5–10% of that [11].

As an intermediate step in a previous work [12], the coarsening and interpolation steps from this Matlab code have been transferred to *BoomerAMG*, while the smoother and the solver were left unchanged. This “hybrid” serial setup was shown to significantly reduce the setup time without affecting the rapidity and scalability of the *in-house AMG* solver.

In the present work, we completely replace the *in-house AMG* by *BoomerAMG*, which allows for the whole AMG problem (setup + solver) to be performed online and in parallel. The existing code requires only limited modifications. Local contributions to the coarse grid operator are built on each process and then handed over to *BoomerAMG*, which takes care of assembling the global operator and of communication. If the operator possesses a nullspace, the solution is normalized such that the mean of the solution entries is 0. Apart from that, the critical aspect of switching to *BoomerAMG* is the choice of parameters for the setup and for the solver that match the performance of the *in-house AMG*.

3 Optimal Parameter Selection

The choice of parameters for the *BoomerAMG* solver is done by testing a set of parameters on a medium-sized test case and looking for the optimal combination. We consider the turbulent flow around a NACA 4412 airfoil at $Re_c = 400,000$ [13] on a mesh made of 253,980 elements, with polynomial order 11, and we run the simulation for 30 timesteps. The best set of parameters is defined as the one which minimizes the time to solution for the pressure equation. This is achieved by balancing two competing aspects: the accuracy of the coarse grid solution and the total number of iterations of the GMRES solver used for the pressure equation. Since the AMG is used as a preconditioner, a high level of accuracy is not paramount. Yet, it should be sufficient to ensure efficient preconditioning. Based on results obtained with the *in-house AMG*, the initial error on the coarse grid problem should be reduced by approximately one order of magnitude. While the *in-house AMG* is designed to ensure that a given reduction in the error is attained at minimal cost, the *BoomerAMG* is designed to ensure the maximum reduction in the error occurs at a given cost. Therefore, the best choice of parameters for the *BoomerAMG* is case dependent; here we optimize this choice in the case of large 3D simulations, when the use of AMG is most relevant.

All tests are run on 4096 processors on the Blue Gene Mira at the Argonne National Laboratory. We test a total of 96 combinations of the following parameters:

- Coarsening type: classical Ruge–Stueben (C1), Falgout (C2), PMIS (C3), HMIS (C4), CGC (C5) and CGC-E (C6),
- Interpolation method: extended (I1) and extended+i (I2),
- Relaxation type: l_1 -Gauss–Seidel forward solve on the down cycle + backward solve on the up cycle (R1), l_1 -scaled hybrid symmetric Gauss–Seidel (R2), Chebyshev (R3) and l_1 -scaled Jacobi (R4),
- AMG strength threshold: 0.25 and 0.5.

We assign a letter and a number to each option which will be used to identify the method when comparing the results.

Table 1 Five best timings for the *BoomerAMG* with corresponding parameters as compared to the *in-house AMG*

Run ID.	AMG solver	Coars.	Intp.	Relax.	Thresh.	No. pres. it.	Pres. time (s)	CGS time (s)
a	<i>in-house AMG</i>	–	–	–	–	3737	666.06	31.51
b	<i>BoomerAMG</i>	C3	I2	R2	0.25	3677	719.77	94.93
c	<i>BoomerAMG</i>	C4	I2	R1	0.5	3762	734.35	95.11
d	<i>BoomerAMG</i>	C3	I1	R1	0.25	3839	746.28	94.7
e	<i>BoomerAMG</i>	C3	I2	R4	0.25	3814	747.04	99.28
f	<i>BoomerAMG</i>	C3	I2	R1	0.5	3866	756.48	100.13

3 V-cycles for the *BoomerAMG*. Total number of iterations for the pressure solver and timings are reported for 30 timesteps

Table 2 Five best timings for the *BoomerAMG* with corresponding parameters as compared to the *in-house AMG*

Run ID.	AMG solver	Coars.	Intp.	Smo.	Thresh.	No. pres. it.	Pres. time (s)	CGS time (s)
a	<i>in-house AMG</i>	–	–	–	–	3737	666.06	31.51
d	<i>BoomerAMG</i>	C3	I1	R1	0.25	3752	675.52	38.09
c	<i>BoomerAMG</i>	C4	I2	R1	0.5	3786	683.65	41.0
b	<i>BoomerAMG</i>	C3	I2	R2	0.25	3877	698.44	40.31
f	<i>BoomerAMG</i>	C3	I2	R1	0.5	3990	721.03	44.35
e	<i>BoomerAMG</i>	C3	I2	R4	0.25	4476	803.74	46.24

1 V-cycle for the *BoomerAMG*. Total number of iterations for the pressure solver and timings are reported for 30 timesteps

In all cases, we set a relative tolerance of 0.1 on the solution of the coarse grid problem and a maximum of 3 V-cycles for the AMG. Moreover, the problem on the coarsest level is solved by Gaussian elimination. Total timings and number of pressure iterations for 30 timesteps are presented in Table 1 for the *in-house AMG* and the five fastest combinations of parameters for the *BoomerAMG*. We assign a letter to each run for comparison later. The time spent in the pressure solver is reported for process 0 and the time spent in the coarse grid solver (CGS) is the maximum value over all processors for a single run. Since an allocation on Mira is always made of cores that are physically contiguous and isolated from the rest of the network, the timings suffer little noise and uncertainty. We see that the number of pressure iterations is on par with the *in-house AMG*, but the time spent in the coarse grid solver is more that three times as much. As a result, the time spent in the pressure solver is between 8 and 13% higher.

To accelerate the *BoomerAMG* solver, we set the maximum number of V-cycles to 1, instead of 3, and perform another test using the optimal parameters. This should significantly accelerate the resolution of the coarse grid solver but reduce the accuracy of the solution, therefore increasing the number of pressure iterations. The corresponding results are presented in Table 2. A surprising result comes from run d, where the number of pressure iterations has actually decreased. Since this might be the sign of an unstable solver, we discard this choice of parameters. The second

best set of parameters corresponds to HMIS coarsening, extended+ i interpolation, l_1 -Gauss–Seidel forward solve on the down cycle + backward solve on the up cycle for the smoother and an AMG strength threshold of 0.5, which we use for the rest of our simulations.

We also experimented with more aggressive non-Galerkin coarsening to change the communication pattern on the largest AMG levels and reduce communication time. However, this caused a drop of accuracy for the coarse grid solver and an increase of iterations for the pressure solver, which led to slower overall timings as a result.

We note that the time to setup the coarse grid problem, which includes building the matrix A_0 and performing the *BoomerAMG* setup, is negligible. In the present configuration, it amounts to less than a second, whereas a single timestep takes about 20 s. Since the setup is performed once at the beginning of the simulation, we do not discuss the matter further. Furthermore, it is orders of magnitude lower than the serial versions of the setup [12].

Further analysis of the results shows that the use of classical Ruge-Stueben, Falgout, CGC or CGC-E significantly slows down the solver. Moreover, another valid choice for the smoother could have been l_1 -scaled hybrid symmetric Gauss–Seidel, whose speed is on par with our choice. Other relaxation methods, Chebyshev and l_1 -scaled Jacobi, are also consistently slower. Finally, the choice of the interpolation method does not have a significant impact and both methods give very similar results.

4 Scaling Results

As is often the case with numerical simulations, we look for the fastest path to solution for a given problem. Therefore, a strong scaling study, where the total amount of work is fixed and the number of processes is increased, is a relevant measure of the efficiency of the *BoomerAMG*. This is opposed to a weak scaling analysis, where the amount of work per process is kept constant, which is not carried out here.

We consider the turbulent flow inside a reactor assembly made of 61 wire-wrapped pins, a configuration appearing in a nuclear reactor core [3]. The mesh consists of 1,650,240 elements, uses polynomial order 7 and has a complex, fully three-dimensional topology, making it a relevant test case for evaluating preconditioning strategies. The initial velocity field is turbulent and we run the simulation for 10 timesteps. Two series of tests were conducted on two supercomputers: Mira and Hazel Hen. The number of compute nodes considered is 512, 1024, 2048, 4096 and 8192 on the former machine and 256, 512, 1024, 2048, 4096 on the latter one. On both computers, the number of MPI processes per node is equal to the number of available compute cores, i.e. 16 on Mira and 24 on Hazel Hen. We use our previous defined optimal parameters for the setup of the *BoomerAMG* and we also include non-Galerkin coarsening. A drop-tolerance of 0.05 for sparsification is set as default

on all levels, with the exception of the five finest levels, which have respective drop-tolerances of 0.0, 0.01, 0.02, 0.03 and 0.04. This choice of parameters is motivated by the fact that, unlike with the wing case, the time for the coarse grid solver is reduced by about 25%, as tests on 6,144 cores on Hazel Hen have shown, without impacting the number of pressure iterations.

First, let us mention that the setup time for the *BoomerAMG* solver is once again negligible in comparison to the time for the entire simulation, requiring less than 3 s on any number of cores on any machine. It is also significantly lower than reading the data of the *in-house AMG*, a serial process, which takes about 80–90 s. Therefore, it does not represent a bottleneck and we do not investigate timings for the setup phase in details.

Next, we present the strong scaling results, based on a single run per core count. The reported value for the time spent in the pressure solver is the timing from core 0. The time spent in the coarse grid solver is measured on each processor and we consider the maximum value among all processes. The average time per timestep for the pressure solver on Mira is shown in Fig. 1, left plot. Unlike what was observed for the wing simulation, the choice of AMG solver does not impact the number of pressure iterations. The *in-house AMG* is slightly faster than the *BoomerAMG* on all core counts and it seems to scale marginally better. Since all other timings are the same, the reason is a faster coarse grid solver, as can be seen in Fig. 1, right plot. The *in-house AMG* achieves a better performance because it optimizes the communication process independently on each level of the AMG. On the coarsest levels, it is able to take advantage of the fast allreduce operation offered by the network of the Blue Gene architecture in hardware. On the finest levels, it picks up the fastest method between a crystal-router strategy or a pairwise exchange.

On 131,072 cores, the actual speed up for the coarse grid solver is 3.11 and the parallel efficiency is 0.194 for the *in-house AMG*, when the timings on 8,192

Fig. 1 Rod-bundle test case on Mira, with 1,650,240 elements, leading to 12 or 13 elements per core for the largest core count. AMG parameters: HMIS coarsening, extended+i interpolation, l_1 -Gauss–Seidel forward solve on the down cycle + backward solve on the up cycle for the smoother and an AMG strength threshold of 0.5. Left: time spent in the pressure solver; value for process 0, averaged over the total number of timesteps. Right: time spent in the coarse grid solver; maximum value over all processes, normalized by the total number of timesteps

Fig. 2 Rod-bundle test case on Hazel Hen, with 1,650,240 elements, leading to 16 or 17 elements per core for the largest core count. AMG parameters: HMIS coarsening, extended+i interpolation, l_1 -Gauss–Seidel forward solve on the down cycle + backward solve on the up cycle for the smoother and an AMG strength threshold of 0.5. Left: time spent in the pressure solver; value for process 0, averaged over the total number of timesteps. Right: time spent in the coarse grid solver; maximum value over all processes, normalized by the total number of timesteps

cores are used as references. These quantities are respectively 2.44 and 0.153 for the *BoomerAMG*. As mentioned before, the network on Mira is characterized by little noise and the uncertainty on the timings is low. Yet, these numbers are based on a single run and are only indications.

The same timings on Hazel Hen are shown in Fig. 2. Unfortunately, the node allocation on this machine can be scattered and the interconnect noise, which is shared with the rest of the computer, can be high and unpredictable. Therefore, a thorough analysis of the scaling results is not possible from a single run and the data from Fig. 2 are only indicative. Nevertheless, the runs for each AMG solver on the same number of cores are obtained using the same node allocation. This makes a comparison between the two solvers on a given core count somewhat relevant. In contrast to the results on Mira, the *BoomerAMG* is slightly faster than the *in-house AMG* on most core counts; the exception is on 6144 cores, where the timings are almost equal. Furthermore, it is quite clear that the coarse grid solver on Hazel Hen does not scale at all. Indeed, the time spent in the coarse grid solver is almost constant from the lowest amount of cores considered.

Based on the available data, we see that beyond 24,576 thousand cores, the time spent in the coarse grid solver accounts for roughly half of the time spent in the pressure solver. At that point, there is about 67 elements and 35,000 grid points per core, which is consistent with the strong scaling limit on a similar computer as identified in Ref. [11].

5 Conclusions

We used the *BoomerAMG* solver from the *hypre* library for linear algebra to solve a global coarse problem that is part of the preconditioner for the pressure equation arising when time-integrating the Navier–Stokes equations. The set of parameters for the *BoomerAMG* setup that leads to the lowest solver time for the pressure equation is HMIS coarsening, extended+ i interpolation, l_1 -Gauss–Seidel forward solve on the down cycle + backward solve on the up cycle for the smoother and an AMG strength threshold of 0.5. We also used non-Galerkin coarsening, with more aggressive drop-tolerance on the coarser levels, to speed up the solver. This new method replaces an existing AMG solver, which is fast and scales well but requires a setup phase done externally in serial. Strong scaling was assessed for both AMG solvers on a real large-scale test case on two supercomputers: an IBM Blue Gene/Q (Mira) and a Cray XC40 (Hazel Hen). On Mira, the *in-house AMG* leads to a faster pressure solver than the *BoomerAMG* on all core counts. The maximum difference is about 10% on 131,072 cores. This is because the *in-house AMG* is able to take advantage of the fast hardware allreduce operation on this machine at the coarsest levels of the AMG solver; in that sense the present result was expected. On Hazel Hen, however, the *BoomerAMG* is consistently faster than the *in-house AMG* and we observe the strong scaling limit to be reached at about 24,576 cores. Overall, the *BoomerAMG* is a valid alternative to the *in-house AMG*; both methods are close in terms of performance and the *BoomerAMG* has the advantage to be set up online and in parallel. In particular for modern architectures, the *BoomerAMG* is even faster than the *in-house AMG* with obvious advantages in the setup phase. All the codes developed in this work are available from the https://github.com/nicooff/nek5000/tree/amg_hypre_c Github repository.

Future work will extend the use of *BoomerAMG* to mesh refinement, where an online and parallel AMG setup phase is a requirement.

Acknowledgements We would like to thank Aleks Obabko for his advice and for sharing the wire-wrapped pin bundle case with us. Financial support by the H2020 EU Project “ExaFLOW: Enabling Exascale Fluid Dynamics Simulation” (grant reference 671571), and the Knut and Alice Wallenberg Foundation is gratefully acknowledged. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. Additional computer time was provided by ExaFLOW at HLRS Stuttgart, and by resources provided by the Swedish National Infrastructure for Computing (SNIC) at PDC Stockholm.

References

1. Baker, A.H., Falgout, R.D., Kolev, T.V., Yang, U.M.: Multigrid smoothers for ultraparallel computing. *SIAM J. Sci. Comput.* **33**(5), 2864–2887 (2011)
2. Baker, A.H., Falgout, R.D., Kolev, T.V., Yang, U.M.: Scaling hypre’s multigrid solvers to 100,000 cores. In: Berry, M.W., Gallivan, K.A., Gallopoulos, E., Grama, A., Philippe, B., Saad,

- Y., Saied, F. (eds.) *High-Performance Scientific Computing: Algorithms and Applications*, pp. 261–279. Springer, London (2012)
3. Brockmeyer, L.M., Sarikurt, F., Hassan, Y., Merzari, E.: CFD investigation of wire-wrapped fuel rod bundles and flow sensitivity to bundles size. In: *Proceedings of the 16th International Topical Meeting on Nuclear Reactor Thermalhydraulics (NURETH-16)* (2015)
 4. Falgout, R.D., Jones, J., Yang, U.: The design and implementation of hypre, a library of parallel high performance preconditioners. In: Bruaset, A.M., Tveito, A. (eds.) *Numerical Solution of Partial Differential Equations on Parallel Computers*, pp. 267–294. Springer, Heidelberg (2006)
 5. Fischer, P.F.: An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *J. Comput. Phys.* **133**(1), 84–101 (1997)
 6. Fischer, P.F., Lottes, J.W., Pointer, D., Siegel, A.: Petascale algorithms for reactor hydrodynamics. *J. Phys. Conf. Ser.* **125**(1), 012076 (2008)
 7. Fischer, P.F., Lottes, J.W., Kerkemeier, S.G.: Nek5000 (2008). <http://nek5000.mcs.anl.gov>
 8. Henson, V.E., Yang, U.M.: BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.* **41**(1), 155–177 (2002)
 9. Lottes, J.W.: *Towards robust algebraic multigrid methods for nonsymmetric problems*. Springer theses (2017)
 10. Lottes, J.W., Fischer, P.F.: Hybrid multigrid/schwarz algorithms for the spectral element method. *J. Sci. Comput.* **24**(1), 45–78 (2005)
 11. Offermans, N., et al.: On the strong scaling of the spectral element solver Nek5000 on petascale systems. In: *Proceedings of the Exascale Applications and Software Conference*. Stockholm (2016)
 12. Offermans, N., Peplinski, A., Marin, O., Fischer, P.F., Schlatter, P.: Towards adaptive mesh refinement for the spectral element solver Nek5000. In: *Proceedings of the DLES11 Conference*. Pisa (2017)
 13. Vinuesa, R., Negi, P.S., Atzori, M., Hanifi, A., Henningson, D.S., Schlatter, P.: Turbulent boundary layers around wing sections up to $Re_c=1,000,000$. *Int. J. Heat Fluid Flow* **72**, 86–99 (2018)
 14. Tufo, H.M., Fischer, P.F.: Fast parallel direct solvers for coarse grid problems. *J. Parallel Distrib. Comput.* **161**(2), 151–177 (2001)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.