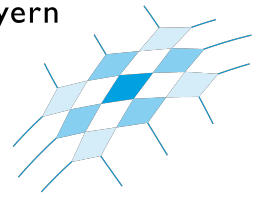




UNIVERSITÄT
BAYREUTH

Elitenetzwerk
Bayern



Book of Modules for the Elite Master's Programme

Scientific Computing

Faculty for Mathematics, Physics and Computer Science, University of Bayreuth, Germany

March 11, 2021

Preface

This Book of Modules describes all currently admissible modules for the elite master's programme *Scientific Computing* and their assignment to the module areas described in the examination regulations "Studien- und Prüfungsordnung". Each module description contains the frequency (and sometimes the term) with which the corresponding course is offered; the list of courses actually provided during the current semester is published via Campus Online. The coordinator of this master's programme will be glad to help you selecting suitable courses to satisfy the requirements formulated in the examination regulations.

The executive committee of the elite master's programme

March 11, 2021

Contents

1	Module Section A: Numerical Mathematics	3
1.1	Module A1: Numerical Methods for Differential Equations	3
1.2	Elective Modules A2: Advanced Topics in Numerical Mathematics	4
2	Module Section B: Modeling and Simulation	10
2.1	Module B1: Applied Functional Analysis	10
2.2	Elective Modules B2: Modeling and Simulation	11
2.3	Module B3: Industrial Internship	23
2.4	Module B4: Modeling and Status Seminar	25
3	Module Section C: High-Performance Computing	26
3.1	Elective Modules C1: High-Performance Computing	26
3.2	Module C2: Practical Course on Parallel Numerical Methods	33
4	Module Section D: Scientific Computing	34
4.1	Elective Modules D1: Complexity Reduction	34
4.2	Module D2: Special Skills in Scientific Computing	44
5	Module Section E: Soft Skills	45
6	Module Section F: Master's Thesis	46
7	Recommended Curriculum	47

1 Module Section A: Numerical Mathematics

1.1 Module A1: Numerical Methods for Differential Equations

<i>Title</i>	Numerical Methods for Differential Equations
<i>Module Label</i>	A1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chairs of Applied and Numerical Analysis, Applied Mathematics, Scientific Computing
<i>Learning Outcomes</i>	<p>This module lays ground to many modules of this programme. It should be attended during the first semester in order to be able to</p> <ul style="list-style-type: none"> • understand the way numerical algorithms for the solution of ordinary and partial differential equations work • choose a suitable algorithm for a given class of ordinary and partial differential equations • adapt standard algorithms to new problems • implement the algorithms discussed in the lecture in MATLAB or in a higher programming language. <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • One step methods for ordinary differential equations (convergence, Runge-Kutta- and extrapolation methods, methods for stiff differential equations, step size control) • multi step schemes for ODEs • finite difference methods for partial differential equations (transport, Poisson, heat, wave equation) • conforming finite element methods for elliptic PDEs (Galerkin method, convergence)
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	6
<i>Work Load</i>	4 h lectures plus 2 h post-processing per week = 90 h; 2 h discussion sections plus 3 h preparation/post-processing = 75 h; 15 h preparation for exam, in total: 180 h
<i>Recommended Prerequisites</i>	Introduction to Numerical Mathematics, Higher Skills in Numerical Mathematics; helpful but not required: Introduction to Ordinary Differential Equations, Introduction to Advanced Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year during winter term
<i>Can also be used as</i>	module B-MP or C1 in bachelor Mathematics and module A1 in master Mathematics

1.2 Elective Modules A2: Advanced Topics in Numerical Mathematics

<i>Title</i>	Numerical Methods for General Types of PDEs
<i>Module Label</i>	A2.1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chairs of Applied and Numerical Analysis, Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding the way numerical algorithms for the solution of special partial differential equations work • Ability to choose a suitable discretization technique for a given partial differential equation • Ability to choose a suitable algorithm • Ability to implement the algorithms discussed in the lecture in a higher programming language <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>This module is the continuation of the module A1: <i>Numerical Methods for Differential Equations</i>. It is focused on the numerical solution of more general types of partial differential equations arising from realistic applications such as fluid dynamics, electromagnetism, structural mechanics, etc. These require special discretization techniques:</p> <ul style="list-style-type: none"> • non-conforming and mixed finite element methods • finite element methods for (Navier-)Stokes equations • finite volume methods • edge elements <p>Special topics:</p> <ul style="list-style-type: none"> • adaptivity • smoothed particle hydrodynamics • mortar methods • level-set methods
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Discontinuous Galerkin Finite Element Methods
<i>Module Label</i>	A2.2
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding of important analysis techniques for Sobolev spaces and their modifications for discontinuous approximation spaces • Ability to choose and apply a suitable discontinuous Galerkin discretization for standard linear and nonlinear advection–diffusion problems • Ability to implement various types of discontinuous Galerkin discretizations in 1D and 2D using a programming language
<i>Content</i>	This course introduces the DG methods for hyperbolic and parabolic partial differential equations (PDEs) and systems of PDEs; it includes formulations for one- and multi-dimensional domains, for linear and nonlinear equations, and considers the stability and convergence analysis for these formulations. Also more advanced topics such as hybridized and ADER-DG methods as well as Riemann problems and their solution constitute a subject of this course.
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1,5 h post-processing per week = 52,5 h; 1 h discussion sections plus 2,5 h preparation/post-processing = 52,5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Constructive Approximation Methods
<i>Module Label</i>	A2.3
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Advanced Analysis and Applications / Numerical Mathematics
<i>Responsible</i>	Chair of Applied and Numerical Analysis
<i>Learning Outcomes</i>	<p>By the end of the course, a successful student should be able to</p> <ul style="list-style-type: none"> • explain the most important concepts of modern, multivariate approximation methods • explain the problems inherent to the reconstruction of multivariate functions from scattered data • prove and analyse the existence, the uniqueness, the computability and the quality of discrete reconstruction techniques • explain and implement the associated numerical schemes • understand the underlying mathematical theory <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • Jackson- and Bernstein theorems for classical univariate polynomial approximation • Multivariate reconstruction methods based upon radial basis functions, moving least-squares and partition of unity methods • Error and stability analysis of multivariate reconstruction methods • Development and implementation of efficient algorithms for such reconstruction methods • Optimal recovery for generalised interpolation with application to solving partial differential equations
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	Introduction to Advanced Analysis, Introduction to Numerical Mathematics
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B-MP or C1 in bachelor Mathematics and module A1 in master Mathematics

<i>Title</i>	Mathematical Control Theory
<i>Module Label</i>	A2.4
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Optimization / Advanced Analysis and Applications
<i>Responsible</i>	Chair of Applied Mathematics
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • knowledge of methods and concepts of mathematical control theory • ability to solve selected problems from mathematical control theory • ability to apply these solution strategies to practical problem formulations <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • definition and classification of control systems • qualitative analysis of control systems • methods for controller design, e.g. <ul style="list-style-type: none"> – methods from linear algebra – methods from optimal control – methods based on Lyapunov functions
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	Introduction to Numerical Mathematics, A1: Numerical Methods for Differential Equations
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Nonlinear Optimization
<i>Module Label</i>	A2.5
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Optimization
<i>Responsible</i>	Chairs of Applied Mathematics, Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • understanding of optimality conditions for nonlinear optimization • understanding of the most important algorithms for the numerical solution of nonlinear optimization problems • ability to model and solve given practical problems in nonlinear optimization • ability to use and develop software for nonlinear optimization <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • modeling of nonlinear optimization problems • advanced algorithms for unconstrained optimization • optimality conditions for nonlinear optimization problems • algorithms for constrained optimization • outlook on further problem classes
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	Introduction to Numerical Mathematics, Higher Skills in Numerical Mathematics
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B-MP or C1 in bachelor Mathematics and module A1 in master Mathematics

<i>Title</i>	Optimization of Differential Equations
<i>Module Label</i>	A2.6
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics / Optimization
<i>Responsible</i>	Chairs of Applied Mathematics, Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • ability to derive and analyse optimality conditions • ability to apply theory to concrete applications • ability to solve the arising problems numerically <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • Introductory examples and concepts • General existence theory and first order optimality conditions • Linear-quadratic problems • Introduction to some non-linear problems • Basic numerical methods • Examples from applications
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	B1: Applied Functional Analysis, A1: Numerical Methods for Differential Equations
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module A1 in master Mathematics

2 Module Section B: Modeling and Simulation

2.1 Module B1: Applied Functional Analysis

<i>Title</i>	Applied Functional Analysis
<i>Module Label</i>	B1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Partial Differential Equations
<i>Responsible</i>	Chairs of Applied and Numerical Analysis, Applied Mathematics, Scientific Computing, Nonlinear Analysis and Mathematical Physics
<i>Learning Outcomes</i>	<p>This module lays ground to many modules of this programme. It should be attended during the first semester. By the end of the course, a successful student should</p> <ul style="list-style-type: none"> • know the basic solution spaces and understand their uses in the theory of partial differential equations; • master the concept of a weak solution; • be able to apply functional analysis methods to problems in partial differential equations; • understand how functional analysis concepts develop out of applications. <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>Basic solution spaces and methods from functional analysis which are used for analysing partial differential equations, in particular</p> <ul style="list-style-type: none"> • Sobolev spaces, embedding theorems • weak solutions of elliptic equations, Lax-Milgram lemma, Fredholm alternative • regularity of weak solutions of elliptic equations • spectral theory for compact operators
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	Introduction to Advanced Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year during winter term
<i>Can also be used as</i>	module B-MP or C1 in bachelor Mathematics and module A1 in master Mathematics

2.2 Elective Modules B2: Modeling and Simulation

<i>Title</i>	Partial Differential Equations and Integral Equations
<i>Module Label</i>	B2.1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Partial Differential Equations
<i>Responsible</i>	Chair of Nonlinear Analysis and Mathematical Physics
<i>Learning Outcomes</i>	<p>By the end of the course, a successful student should</p> <ul style="list-style-type: none"> • understand the origin of the treated equations in the modeling process; • know fundamental results on existence and uniqueness of their solutions; • understand qualitative properties of their solutions; • master key methods of their analysis. <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>Existence, uniqueness, and properties of solutions for integral equations and for various types of partial differential equations that are eminent for modeling in the sciences, in particular</p> <ul style="list-style-type: none"> • parabolic equations • wave equations and symmetric hyperbolic systems • Schrödinger equations • integral operators related to elliptic equations
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam, in total: 240 h
<i>Recommended Prerequisites</i>	Introduction to Advanced Analysis, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Modeling with Differential Equations
<i>Module Label</i>	B2.2
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Partial Differential Equations
<i>Responsible</i>	Chairs of Applied Mathematics, Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Ability to identify suitable mathematical models for a given application • Knowledge about important modeling principles • Ability to apply modeling techniques to basic practical applications <p>In contrast to students who have passed bachelor module C1, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • General modeling principles • Mathematical Models based on Ordinary Differential Equations from, e.g., <ul style="list-style-type: none"> – Mathematical Biology – Mechanics • Mathematical Models based on Partial Differential Equations from, e.g., <ul style="list-style-type: none"> – Mathematical Physics – Mathematical Finance
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	B1: Applied Functional Analysis, A1: Numerical Methods for Differential Equations
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Mathematical Modeling for Climate and Environment
<i>Module Label</i>	B2.3
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Partial Differential Equations
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Knowledge of important physical principles and their representation in mathematical models for main types of climate and environmental models • Ability to identify the key interactions between different compartments of a climate model and to express them in mathematical form • Ability to formulate simple environmental and climate models and skills to implement them using e.g. Matlab
<i>Content</i>	<ul style="list-style-type: none"> • Physical principles, mathematical models, and selected numerical methods in climate and environmental sciences • Earth system: Main components, driving forces, scales, feedbacks • Hierarchy of climate models, regional and global focus • Environmental modeling: Main applications and problem settings
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 2 h post-processing per week = 90 h; 120 h practical course; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	B1: Applied Functional Analysis, A1: Numerical Methods for Differential Equations, basic programming skills in Matlab
<i>Grading</i>	Oral exam; active participation in the tutorials
<i>Frequency</i>	Once per year, summer term
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Pattern Recognition
<i>Module Label</i>	B2.4
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Applied Computer Science III
<i>Learning Outcomes</i>	This course imparts advanced, systematic comprehension and methods to recognize or classify patterns in a set of data. E. g. applications are in the fields of object recognition, recognition of hand writing, speech, or gestures, and facial recognition.
<i>Content</i>	Bayesian classification, Parameter estimation, Nonparametric techniques, Linear classification, Feedforward neural networks, Feedback neural networks, Nonmetric methods, Supervised Learning, Unsupervised Learning
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1 h post-processing per week = 45 h; 60 h practical course; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	none
<i>Grading</i>	Oral exam
<i>Frequency</i>	Once per year, winter term
<i>Can also be used as</i>	

<i>Title</i>	Mechanics of Continua
<i>Module Label</i>	B2.5
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Fluid Mechanics
<i>Responsible</i>	Theoretical Physics
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understand fundamental concepts of continuum mechanics such as stress tensors, dynamic equations • Assess which physical effects are relevant/irrelevant for a given situation • Propose a theoretical description and solution approach for a given continuum mechanical problem
<i>Content</i>	<ul style="list-style-type: none"> • Fundamental concepts in fluid mechanics: pressure, stress tensor • Euler, Stokes and Navier-Stokes equations • Dimensionless numbers • Applications to <ul style="list-style-type: none"> – Turbulence – Microfluids
<i>Duration</i>	1 semester
<i>Language</i>	German / English on demand
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	Understanding of classical Newtonian mechanics
<i>Grading</i>	Oral or written exam
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Molecular Dynamics Simulations of Biophysical Systems
<i>Module Label</i>	B2.6
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Physics
<i>Responsible</i>	Biofluid Simulation and Modeling
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding of the theoretical background in biophysical simulations • Practical application to example problems, e.g. protein folding
<i>Content</i>	<p>Molecular Dynamics computer simulations have become an invaluable tool in many areas of the natural sciences. Their widespread applications range from protein folding and membrane dynamics in biological physics, solute-solvent interactions in theoretical chemistry to nanofluidics in process engineering. The aim of this course is (i) to understand the physics behind frequently employed simulation methods and (ii) to gain practical experience in using these methods for a range of sample problems. We will start by considering the basic ingredients for atomistic simulations such as Verlet integration, force fields, long-range electrostatic interactions, thermostating, etc. We will then move on to cover more advanced topics such as free energy methods and sampling techniques for rare events. Finally, we will discuss systematic coarse-graining methods which allow the simulation of larger and more complicated processes, e.g. the unfolding of a protein in shear flow. The course includes a lab part in which actual simulations with the GROMACS package are conducted.</p>
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2h/week) and additional practical project
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 2 h post-processing per week = 60 h; Practical project = 40 h; 20 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Knowledge of statistical mechanics is helpful
<i>Grading</i>	Oral or written exam
<i>Frequency</i>	Approximately every two years
<i>Can also be used as</i>	

<i>Title</i>	Bioinformatics: Molecular Modeling
<i>Module Label</i>	B2.7
<i>Module Type</i>	Lecture with practical course
<i>Area of Research</i>	Biochemistry
<i>Responsible</i>	Bioinformatics / Structural Biology
<i>Learning Outcomes</i>	Knowledge on modeling and analysis of biomolecular processes such as enzymatic reactions In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.
<i>Content</i>	The lecture treats the theoretical foundations of molecular modeling (molecular force fields, biomolecular electrostatics, classical and statistical mechanics), their numerical designs (molecular dynamics simulations, energy minimization and normal mode analysis, Monte Carlo simulations), fundamentals quantum chemical methods as well as the modeling of biochemical reactions and ligand binding. In the practical course, various techniques (including analysis of biomolecular structures, computation of electrostatic properties of biomolecules, normal-mode analysis) will be exemplified by selected case studies to provide students with practical demonstrations of these methods.
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and practical course (3 weeks)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1 h post-processing per week = 45 h; 60 h practical course; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Basic knowledge of chemistry and some biochemistry
<i>Grading</i>	Oral or written exam; active participation in the practical course
<i>Frequency</i>	Every year during winter term
<i>Can also be used as</i>	

<i>Title</i>	Foundations of Bioinformatics
<i>Module Label</i>	B2.8
<i>Module Type</i>	Lecture with practical course
<i>Area of Research</i>	Biochemistry
<i>Responsible</i>	Bioinformatics / Structural Biology
<i>Learning Outcomes</i>	<p>Students should acquire the basics of bioinformatics and get to know them in theory and practice.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>The lecture covers the basic bioinformatic applications. Namely, the application of various theoretical methods in the analysis of molecular biological data in the foreground (databases and database search, sequences and sequence alignments, phylogenetic trees) as well as fundamentals of molecular modeling, structure prediction and drug design. In the practical course, the students have hands-on sessions for the different methods, the use of internet tools for sequence data analysis, web-based databases and creation of sequence alignments. Moreover some basic introduction to molecular visualization and an introduction to the UNIX operating system are provided.</p>
<i>Duration</i>	1 semester
<i>Language</i>	German
<i>Teaching Method</i>	Lectures (2 h/week) and practical course (3 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 2 h post-processing per week = 60 h 3 h practical course = 45 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Basic knowledge of chemistry and some biochemistry
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year during summer term
<i>Can also be used as</i>	

<i>Title</i>	Higher Strengths of Materials
<i>Module Label</i>	B2.9
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Engineering Science
<i>Responsible</i>	Chair of Design and CAD
<i>Learning Outcomes</i>	The module enables the students to calculate complex technical products by use of linear elasticity approach. It deepens the knowledge in the area of strengths of materials calculation. In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.
<i>Content</i>	Selected subjects in higher strengths of materials area, e.g. multiaxial stress and deformation, theories of thin shells, mechanical vibrations with relation to typical applications.
<i>Duration</i>	1 semester
<i>Language</i>	German
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 2 h discussion sections plus 1.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Basic engineering knowledge funding on Bachelor Engineering Science studies, especially in technical mechanics and strengths of materials.
<i>Grading</i>	Written examination (60 min)
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Computer Aided Engineering
<i>Module Label</i>	B2.10
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Engineering Science
<i>Responsible</i>	Chair of Design and CAD
<i>Learning Outcomes</i>	CAE1: ability to create CAD models and generate design proposals using optimization algorithms. CAE2: mastery of modern methods of calculation of statics and their application to constructive tasks; knowledge of associated software In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.
<i>Content</i>	CAE1: mastery of modern calculation methods and their application to constructive tasks; knowledge of associated software. Ability to design independently using CAD. CAE2: theory and application of finite element methods to static problems with a focus on the constructive point of view and modeling.
<i>Duration</i>	2 semesters
<i>Language</i>	English
<i>Teaching Method</i>	CAE1: lectures (2 h/week) and CAE2: seminar (2 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 2 h seminar plus 1.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Basic technical understanding A1: Numerical Methods for Differential Equations
<i>Grading</i>	Written examination; For the admission to the written exam a vivid participation in the exercises is required.
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Model Building and Simulation of Mechanical Systems
<i>Module Label</i>	B2.11
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Engineering Science
<i>Responsible</i>	Chair of Design and CAD
<i>Learning Outcomes</i>	The industrial standard CAD software CATIA enables students to create virtual models of products. The course Higher Finite Element Analysis enables students to build-up models for the dimensioning of complex technical products by use of sophisticated Finite Element Analysis methods. The knowledge is used in wide areas of advanced product development. In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.
<i>Content</i>	FEA: Handling of great and complex structures, use of shell and volume elements. Solving non-linearity, vibration and heat transfer problems. CATIA: Part and assembly creation; generation of technical drawings; surface modeling with Generative Shape Design.
<i>Duration</i>	2 semesters
<i>Language</i>	German
<i>Teaching Method</i>	FEA: lectures (2 h/week), seminar (1 h/week) and CATIA: seminar (2 h/week)
<i>Credit Points</i>	6
<i>Work Load</i>	FEA: 45 h lecture + follow up work 45 h seminars + preparation and follow-up work CATIA: 60 h seminars + preparation and follow-up work Examination: 30 h examination preparation; Module in total: 180 hours
<i>Recommended Prerequisites</i>	Basic engineering knowledge funding on Bachelor Engineering Science studies, especially in technical mechanics, construction design and mechanical engineering
<i>Grading</i>	Written examination (120 min)
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Foundations of Data Management
<i>Module Label</i>	B2.12
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Data Management, Data Science
<i>Responsible</i>	Chair of Theoretical Computer Science
<i>Learning Outcomes</i>	Students will learn the mathematical foundations of data management (which includes databases and data science). They will understand the connections between logic, expressivity, computational complexity, and efficient algorithms in this area. They will learn the formal tools to be able to understand and interpret recent scientific developments in the area.
<i>Content</i>	The lecture starts with formal definitions of databases and query languages. After showing that there is a deep connection between first-order logic and SQL when it comes to querying relational databases, it investigates the computational complexity (or efficient algorithms) for evaluating and analyzing SQL or first-order logic queries on databases. We then investigate conjunctive queries as a practically relevant special case, treat their evaluation and optimization problems, and connections with graph theory. (Knowledge of SQL is helpful but is not required.)
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h exercises plus 1 h post-processing per week = 30 h; 37.5 h preparation for the exam; in total: 120 h
<i>Recommended Prerequisites</i>	Theoretische Informatik 1 or mathematical skills equivalent to those obtained in a BSc degree in mathematics or physics
<i>Grading</i>	Oral or written exam; participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

2.3 Module B3: Industrial Internship

<i>Title</i>	Industrial Internship
<i>Module Label</i>	B3
<i>Module Type</i>	Internship
<i>Area of Research</i>	Scientific Computing
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<p>By the end of the internship, a successful student should have</p> <ul style="list-style-type: none"> • gained hands-on experience in a non-academic environment • applied mathematical techniques and techniques from computer science to real applications • acquired new ideas for his/her own research • written a short report
<i>Content</i>	Industrial applications in the area of Scientific Computing
<i>Duration</i>	6 weeks
<i>Language</i>	English
<i>Teaching Method</i>	Practical course
<i>Credit Points</i>	8
<i>Work Load</i>	6 weeks of internship = 230 h 10 h preparation of report; in total: 240 h
<i>Recommended Prerequisites</i>	Basic lectures of this programme (after first year)
<i>Grading</i>	Written report of at least 10 pages (to be submitted not later than 4 weeks after the internship)
<i>Frequency</i>	Every semester
<i>Can also be used as</i>	

2.4 Module B4: Modeling and Status Seminar

<i>Title</i>	Modeling and Status Seminar
<i>Module Label</i>	B4
<i>Module Type</i>	Seminar
<i>Area of Research</i>	All areas
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<p>Successful students can</p> <ul style="list-style-type: none"> • Modeling and numerical solution: transfer real-world problems to a mathematical model pave their way into a scientific topic; work in small groups select suitable efficient numerical methods and implement them on parallel computers • Talk: choose and master suitable presentation techniques speak freely about a subject and illustrate important structures instructively answer spontaneous questions from the audience in a reliable manner • Discussion: phrase appropriate subject-specific questions express constructive criticism for a talk exploit constructive criticism for their future talks • Handout: expose an advanced mathematical subject briefly, concisely, and memorably in writing efficient usage of scientific publication systems (e.g., \LaTeX)
<i>Content</i>	<p>Modeling Seminar:</p> <ul style="list-style-type: none"> • Students receive real-world projects and work (in small groups) their way into them • Each group prepares a presentation for its subject (duration: 30–60 minutes) and talks about it in front of the plenum • Each group prepares and distributes a report (at least 10 pages) using a scientific text system (e.g., \LaTeX) <p>Status Seminar:</p> <ul style="list-style-type: none"> • Each student prepares a presentation on the status of his/her studies and results of his/her research (duration: 15–30 minutes) and talks about it in front of the plenum <p>For both seminars there will be a discussion on the subject and on the presentation.</p>
<i>Duration</i>	4 semesters
<i>Language</i>	English
<i>Teaching Method</i>	Modeling seminar (1 week) and status seminar (2 days) each year
<i>Credit Points</i>	8
<i>Work Load</i>	70 h practical course and 30 h seminar each year = 200 h 40 h preparation/post-processing for seminar, in total: 240 h
<i>Recommended Prerequisites</i>	At least one module of A2 and D1, respectively; C2: Practical Course on Parallel Numerical Methods
<i>Grading</i>	Oral presentation and written report of at least 10 pages (to be submitted not later than 4 weeks after the seminar)
<i>Frequency</i>	Each year (modeling seminar during summer break, status seminar during winter break)
<i>Can also be used as</i>	

3 Module Section C: High-Performance Computing

3.1 Elective Modules C1: High-Performance Computing

<i>Title</i>	Algorithms and Data Structures II
<i>Module Label</i>	C1.1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Algorithms and Data Structures
<i>Learning Outcomes</i>	<p>This module teaches advanced techniques for the design and analysis of algorithms and data structures.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>Possible topics are:</p> <ul style="list-style-type: none"> • design principles • graph algorithms • advanced data structures • approximation algorithms • parameterized algorithms • randomized algorithms
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	Elementary programming skills, Basic skills in the design and analysis of algorithms.
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Algorithms and Data Structures III
<i>Module Label</i>	C1.2
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Algorithms and Data Structures
<i>Learning Outcomes</i>	<p>This module teaches specialized techniques for the design and analysis of algorithms and data structures</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>Possible topics are:</p> <ul style="list-style-type: none"> • geometric algorithms • algorithms for data analysis • streaming algorithms • external-memory algorithms
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	<p>2 h lectures plus 1.5 h post-processing per week = 52.5 h;</p> <p>1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h;</p> <p>15 h preparation for exam; in total: 120 h</p>
<i>Recommended Prerequisites</i>	Advanced programming skills, Advanced skills in the design and analysis of algorithms
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

<i>Title</i>	Parallel and Distributed Systems I
<i>Module Label</i>	C1.3
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Parallel and Distributed Systems
<i>Learning Outcomes</i>	<p>The goal of this course is to impart to the students basic techniques in parallel and distributed programming. By that, special methodical competences are acquired: By understanding basic problems such as load balancing and scalability and by learning synchronization and communication techniques, the students are enabled to design and, with the help of communication and thread libraries, to transform parallel algorithms into efficient parallel and distributed programs. By that, both shared and distributed address spaces are acquired. In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<ul style="list-style-type: none"> • Architecture and interconnection networks for parallel systems • Performance analysis and scalability of parallel programs • Programming and synchronization techniques for shared address space with multi-threading • Coordination of parallel and distributed programs • Application of programming techniques to complex examples from different areas • Programming techniques for distributed address spaces and message-passing
<i>Duration</i>	1 semester
<i>Language</i>	German
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	120 h in total (45 h presence, 60 h preparation/post-processing, 15 h preparation for exam)
<i>Recommended Prerequisites</i>	
<i>Grading</i>	Written exam; For the admission to the written exam a vivid participation in the exercises is required.
<i>Frequency</i>	Every year in winter term
<i>Can also be used as</i>	

<i>Title</i>	Parallel and Distributed Systems II
<i>Module Label</i>	C1.4
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Parallel and Distributed Systems
<i>Learning Outcomes</i>	<p>The goal of this course is to give the students a deep understanding of important techniques in parallel and distributed programming. The emphasis lies on the acquiring of methodical and technical competences. Based on a deep understanding of standard protocols for computer networks such as IP or TCP/UDP, the students are enabled to design and implement distributed programs. The course covers message-passing approaches such as MPI, passiv communication mechanisms such as sockets, and also active mechanisms such as RPC, RMI, or CORBA. The course also imparts design and implementation competences by applying the techniques to a variety of examples.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>The course covers the basics of parallel and distributed systems with an emphasis on distributed systems. Based on the first part of the course, the following topics are covered:</p> <ul style="list-style-type: none"> • Message-Passing programming (MPI) • Important communication protocols in distributed systems • Communication coordination and synchronization mechanisms in distributed systems (examples: Sockets, RPC, Java RMI) • Coordination with distributed objects (example: CORBA) • Security aspects and mechanisms in distributed systems
<i>Duration</i>	1 semester
<i>Language</i>	German
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	120 h in total (45 h presence, 60 h preparation/post-processing, 15 h preparation for exam)
<i>Recommended Prerequisites</i>	C1.3: Parallel and Distributed Systems I
<i>Grading</i>	Written exam; For the admission to the written exam a vivid participation in the exercises is required.
<i>Frequency</i>	Every year in summer term
<i>Can also be used as</i>	

<i>Title</i>	High-Performance Computing
<i>Module Label</i>	C1.5
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Parallel and Distributed Systems
<i>Learning Outcomes</i>	<p>The goal of this course is to give the students a deep understanding of important techniques of program analysis and program transformation. The emphasis lies on the acquiring of analytical and technological competences: the students are enabled to analyse arbitrary programs by applying the techniques of data and control dependency analysis and to perform optimizing program transformation based on these analysis techniques. Examples are the vectorization and parallelization of program parts or optimization towards a given memory hierarchy. Methodical and algorithmic competences are acquired by learning scheduling and load balancing algorithms and the underlying principles.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>The following topics are covered:</p> <ul style="list-style-type: none"> • Overview of current processor architectures and interconnection technologies • Control flow and data flow analysis, data flow equations and solution methods for data flow equations, optimizing transformations • Data dependency analysis, loop dependencies, data dependence equations and solution methods for them • Program transformations for vectorization, parallelization and cache optimization • Methods for scheduling and load balancing for instructions, loops, and tasks • OpenMP programming • Register allocation and program transformations for reducing the register need of programs • CPU programming with CUDA
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	240 h in total (90 h presence, 150 h preparation/post-processing with processing of worksheets)
<i>Recommended Prerequisites</i>	C1.3: Parallel and Distributed Systems I
<i>Grading</i>	Written exam; For the admission to the written exam a vivid participation in the exercises is required.
<i>Frequency</i>	Every year in summer term
<i>Can also be used as</i>	

<i>Title</i>	Parallel Algorithms
<i>Module Label</i>	C1.6
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Parallel and Distributed Systems
<i>Learning Outcomes</i>	<p>Students acquire in-depth knowledge about selected parallel algorithms from different fields of application. In particular, in connection with exercises, students gain analytical and methodological expertise, which empowers them to understand, implement, analyse, and design parallel algorithms.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	Selected parallel algorithms are presented. The range extends from basic, widespread algorithms (e.g., sorting) to complex algorithms from specific fields of application (e.g., computer graphics). Emphasis is put on algorithms from the field of scientific computing. The exercises cover theoretical problems as well as practical programming experience.
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	120 h in total (45 h presence, 60 h preparation/post-processing, 15 h preparation for exam)
<i>Recommended Prerequisites</i>	Algorithms and Data Structures I, C1.3: Parallel and Distributed Systems I
<i>Grading</i>	Oral or written exam
<i>Frequency</i>	Every year in summer term
<i>Can also be used as</i>	

<i>Title</i>	Programming and Data Analysis in Python
<i>Module Label</i>	C1.7
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair of Serious Games
<i>Learning Outcomes</i>	Students learn to quickly prototype and implement numerical programs in Python. They learn Python as a programming language and a scientific computing environment. They acquire knowledge of the basic programming language, as well as of important libraries for scientific computing, such as NumPy, SciPy, Matplotlib, Pandas, and TensorFlow/Keras. They develop practical and applied skills in exploratory computing, rapid prototyping, and implementation of numerical methods. In contrast to other environments, the Python scientific computing environment is open source, widely used, optimized for programmer productivity, and benefits from a large community and library ecosystem.
<i>Content</i>	The Python programming language: Programming philosophy in Python, data types, control structures, functions, object-oriented programming, debugging. Algorithms: Basic algorithms (e.g., searching and sorting), bisection, recursion, dynamic programming, Newton's method. Matrix methods: Linear Algebra with NumPy, matrix factorizations, eigenvectors and values, diagonalization, SVD, least squares and pseudoinverse. Data analysis: Pandas, clustering, plotting. Neural networks and deep learning.
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	120 h in total (45 h presence, 60 h preparation/post-processing, 15 h preparation for exam)
<i>Recommended Prerequisites</i>	None
<i>Grading</i>	Oral or written exam (85%), exercises (15%)
<i>Frequency</i>	Every year in winter term
<i>Can also be used as</i>	

3.2 Module C2: Practical Course on Parallel Numerical Methods

<i>Title</i>	Parallel Numerical Methods
<i>Module Label</i>	C2
<i>Module Type</i>	Practical course
<i>Area of Research</i>	Computer Science, Scientific Computing
<i>Responsible</i>	Chairs of Parallel and Distributed Systems, Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Implementation of parallel algorithms: select suitable efficient numerical methods choose data structures that are suitable for the respective problem implement the numerical methods on a parallel computer using standard libraries • Presentation and discussion: choose and master suitable presentation techniques speak freely about a subject and illustrate important structures instructively answer spontaneous questions from the audience in a reliable manner
<i>Content</i>	In this practical course, students implement manageable numerical problems (such as Gaussian elimination, finite element discretization of 2d Laplacian, etc.) on parallel computers using the programming language C/C++ and standard software libraries (LAPACK/BLAS, OpenMP, OpenMPI). The resulting parallel efficiency is observed depending on the chosen implementation (naive or advanced such as Schwarz methods).
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Practical course (2 weeks)
<i>Credit Points</i>	2
<i>Work Load</i>	50 h practical course; 10 h preparation for exam; in total: 60 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, C1.3: Parallel and Distributed Systems I, D1.1: Efficient Treatment of Non-local Operators
<i>Grading</i>	Implementation and presentation of approaches; active participation and discussion
<i>Frequency</i>	Every year at the end of the winter term
<i>Can also be used as</i>	

4 Module Section D: Scientific Computing

4.1 Elective Modules D1: Complexity Reduction

<i>Title</i>	Efficient Treatment of Non-local Operators
<i>Module Label</i>	D1.1
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Scientific Computing
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding the way numerical algorithms for the solution of partial differential and integral equations work • Understanding that non-local operators may contain redundancies which can be used to reduce their asymptotic complexity • Ability to choose a suitable algorithm for a given class of partial differential and integral equations • Ability to implement the algorithms discussed in the lecture in a higher programming language on a parallel computer
<i>Content</i>	<p>State-of-the-art linear complexity treatment of partial differential and integral operators and parallelization techniques:</p> <ul style="list-style-type: none"> • fast multipole methods for the efficient treatment of multi-source potentials (one of the TOP10 algorithms from the 20th century) • hierarchical matrices (for the treatment of non-local operators with linear complexity) • Schwarz methods (additive and multiplicative) • BPX preconditioner • Domain decomposition (overlapping and non-overlapping), BPS and Neumann-Neumann preconditioners, FETI
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	module B-MP or C1 in bachelor Mathematics and module A1 in master Mathematics

<i>Title</i>	Fast Methods for Differential and Integral Equations
<i>Module Label</i>	D1.2
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Scientific Computing
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding the way numerical algorithms for the solution of partial differential and integral equations work • Detection of suitable structures which can be exploited for the complexity reduction of solution operators of elliptic boundary value problems • Ability to choose a suitable algorithm for a given class of partial differential or integral equations • Ability to implement the algorithms discussed in the lecture in a higher programming language
<i>Content</i>	<p>Numerical analysis of optimal complexity solvers for the treatment of boundary value problems; efficient treatment of parameter-dependent problems:</p> <ul style="list-style-type: none"> • subspace correction methods • hierarchical bases and BPX preconditioners • geometric and algebraic multigrid methods (convergence and implementation aspects) • reduced bases methods • analysis of hierarchical matrices
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Efficient Numerical Treatment of Multiscale Problems
<i>Module Label</i>	D1.3
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding of key mechanisms and challenges in scale interaction for the PDE-based equations and systems in scientific and technical problems • Ability to identify and apply a suitable modeling and/or numerical technique for a wide range of multiscale problems • Ability to implement the mathematical methods and numerical algorithms introduced in the course using a programming language
<i>Content</i>	<p>Modeling approaches:</p> <ul style="list-style-type: none"> • asymptotic analysis • homogenization • Reynolds-averaged Navier-Stokes (RANS), large eddy simulation (LES) <p>Numerical methods:</p> <ul style="list-style-type: none"> • multiscale finite element method (MsFEM) • variational multiscale method • wavelet-based discretizations • reduced-basis methods • heterogeneous multiscale methods (HMM)
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module A1 in master Mathematics

<i>Title</i>	Numerical Methods for Uncertainty Quantification
<i>Module Label</i>	D1.4
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	
<i>Content</i>	This is a cutting-edge area of Scientific Computing. In addition to Monte Carlo methods, stochastic collocation, polynomial chaos expansions, stochastic Galerkin methods, the Karhunen-Loève expansion, model order reduction, and multilevel quadrature recent developments in this area are to be discussed.
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (4 h/week) and tutorials (2 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 3 h post-processing per week = 52,5 h; 1 h discussion sections plus 2,5 h preparation/post-processing = 52,5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	High-dimensional Approximation
<i>Module Label</i>	D1.5
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chairs of Applied and Numerical Analysis, Scientific Computing
<i>Learning Outcomes</i>	By the end of the course, a successful student should <ul style="list-style-type: none"> • understand the curse of dimensionality • know several concepts to reduce the complexity in high-dimensional problems • be able to apply such concepts to typical examples from finance, physics and engineering
<i>Content</i>	<ul style="list-style-type: none"> • Introduction to problems from finance, physics and engineering leading to high-dimensional partial differential equations, such as Black-Scholes and Fokker-Planck • Modern concepts for high-dimensional problems including tensor product methods, sparse grids, kernel-based methods, Monte-Carlo and Quasi-Monte-Carlo methods • Error and stability analysis of such methods • Efficient algorithms for and implementations of such methods
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1,5 h post-processing per week = 52,5 h; 1 h discussion sections plus 2,5 h preparation/post-processing = 52,5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Data Analytics
<i>Module Label</i>	D1.6
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Computer Science
<i>Responsible</i>	Chair for Databases and Information Systems
<i>Learning Outcomes</i>	<p>Conceptual foundation of development of large databases (Big Data) and information systems with focus on modeling. Deepening of proficiency in databases in the context of large and complex database and web applications; imparting of interdisciplinary, analytical competences for reconstructing and modeling complex applications (mostly stemming from the application fields); technological competence for selecting and integrating heterogeneous modeling and implementation concepts for the design and realization of data and process based applications. Deepening of proficiency in the fields of data analytics. Realization of complex architectures in the application fields Bio Informatics, Environmental Informatics and Engineer Informatics will be discussed in all courses.</p> <p>In contrast to students who have passed the corresponding bachelor module, students of this modules can apply the previous techniques more autonomously and can relate them to formerly acquired advanced skills.</p>
<i>Content</i>	<p>first semester: Data Warehousing, Data Mining second semester: Data Visualisation, Machine Learning, Ontologies, NoSQL, Distributed Computing Concepts (MapReduce, Hadoop, etc.)</p>
<i>Duration</i>	2 semesters
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	8
<i>Work Load</i>	<p>4 h lectures plus 3 h post-processing per week = 105 h; 2 h discussion sections plus 5 h preparation/post-processing = 105 h; 30 h preparation for exam; in total: 240 h</p>
<i>Recommended Prerequisites</i>	Datenbanken und Informationssysteme I
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year during winter and summer term
<i>Can also be used as</i>	

<i>Title</i>	Complexity Reduction in Control
<i>Module Label</i>	D1.7
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Applied Mathematics
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Understanding why many numerical approaches to control problems suffer from the curse of dimensionality • Recognition of redundancies, such as the turnpike property in optimal control problems, and their use for complexity reduction • Knowledge about state-of-the-art model order reduction methods for linear and nonlinear control systems and how they rely on the particular input-output-structure • Ability to identify suitable techniques for particular applications
<i>Content</i>	<ul style="list-style-type: none"> • State-of-the-art techniques for the reduction of complexity in control problems, as for instance <ul style="list-style-type: none"> – Model predictive control and its use as a complexity reduction technique – Sparse grid and parallel computing approaches for Hamilton-Jacobi-Bellman equations – Model order reduction methods for control systems, such as balanced truncation or proper orthogonal decomposition • Implementation of selected algorithms in the tutorials
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	A2.4: Mathematical Control Theory, A1: Numerical Methods for Differential Equations
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Meshfree Methods
<i>Module Label</i>	D1.8
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Applied and Numerical Analysis
<i>Learning Outcomes</i>	<p>By the end of the course, a successful student should</p> <ul style="list-style-type: none"> • know fundamental techniques to reduce the complexity in meshfree discretisations for solving partial differential equations numerically • be able to choose a suitable meshfree method for a given problem • master key methods of their analysis and implementation
<i>Content</i>	<ul style="list-style-type: none"> • Short review of kernel-based collocation and particle methods for solving partial differential equations • Discussion of various cost reducing methods including: <ul style="list-style-type: none"> – Fast summation techniques based on far field expansions of radial basis functions (RBF) and completely monotone functions – Local Lagrangian methods, RBF-FD – Generalised alternating projection methods – multilevel methods for compactly supported RBF – Adaptive and greedy variants of the above methods • Discussion of data structures and implementational details for meshfree methods
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam, in total: 120 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis, A2.3: Constructive Approximation Methods
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Boundary Element Methods
<i>Module Label</i>	D1.9
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics
<i>Responsible</i>	Chair of Scientific Computing
<i>Learning Outcomes</i>	<p>Exterior boundary value problems are difficult to treat via finite element discretizations due to the unboundedness of the computational domain. This lecture presents a different approach by which the boundary value problem is reformulated as an integral equation on the boundary. In particular, this offers the advantage that only a lower-dimensional set has to be discretized. As a consequence, the resulting linear systems are significantly smaller but fully populated in general. The latter difficulty can be treated by local low-rank approximation.</p> <p>By the end of the course, a successful student should</p> <ul style="list-style-type: none"> • know fundamental techniques to reduce suitable boundary value problems to boundary integral equations • master key methods of the analysis and implementation of boundary integral methods • be able to reduce the complexity of suitable discrete non-local operators
<i>Content</i>	<ul style="list-style-type: none"> • Sobolev spaces on manifolds • fundamental solutions of partial differential operators • boundary integral operators and their properties • boundary integral equations and their finite element discretization • generating the matrix coefficients • fast boundary element methods • applications: potential equation, linear elasticity, Stokes equations
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam, in total: 120 h
<i>Recommended Prerequisites</i>	A1: Numerical Methods for Differential Equations, B1: Applied Functional Analysis
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B1 in master Mathematics

<i>Title</i>	Optimization Methods in Machine Learning
<i>Module Label</i>	D1.10
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Optimization, Data Analysis
<i>Responsible</i>	Chair of Applied Mathematics
<i>Learning Outcomes</i>	<ul style="list-style-type: none"> • Knowledge about the special nature of optimization problems in machine learning • Understanding of optimization algorithms in machine learning • Ability to apply optimization algorithms in machine learning properly
<i>Content</i>	<ul style="list-style-type: none"> • Introduction to machine learning • Stochastic gradient methods • Proximal gradient methods • Acceleration techniques
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Basic knowledge in numerics and optimization
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every two years
<i>Can also be used as</i>	module B1 in master Mathematics

4.2 Module D2: Special Skills in Scientific Computing

<i>Title</i>	Special Skills in Scientific Computing
<i>Module Label</i>	D2
<i>Module Type</i>	Lecture
<i>Area of Research</i>	Numerical Mathematics, Scientific Computing
<i>Responsible</i>	Chairs of Applied and Numerical Analysis, Applied Mathematics, Scientific Computing
<i>Learning Outcomes</i>	The goal of this module is to provide students with special skills in the areas of Numerical Mathematics and Scientific Computing, relevant for current research activities.
<i>Content</i>	<p>An active field of mathematical research, in which specialized techniques are applied or in which known techniques from different areas are combined in an original way. Examples are:</p> <ol style="list-style-type: none"> 1. Fractional order differential operators 2. Optimization of nonsmooth problems 3. Advanced topics of optimization with partial differential equations 4. mathematical control theory for infinite-dimensional systems
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Lectures (2 h/week) and tutorials (1 h/week)
<i>Credit Points</i>	4
<i>Work Load</i>	2 h lectures plus 1.5 h post-processing per week = 52.5 h; 1 h discussion sections plus 2.5 h preparation/post-processing = 52.5 h; 15 h preparation for exam; in total: 120 h
<i>Recommended Prerequisites</i>	Advanced status of study
<i>Grading</i>	Oral or written exam; active participation in the tutorials
<i>Frequency</i>	Every year
<i>Can also be used as</i>	

5 Module Section E: Soft Skills

<i>Title</i>	Soft Skills
<i>Module Label</i>	E
<i>Module Type</i>	Seminar
<i>Area of Research</i>	
<i>Responsible</i>	Programme coordinator
<i>Learning Outcomes</i>	Key skills for personal development
<i>Content</i>	Participation in seminars devoted to presentation skills, data processing, literature research, handling of foreign-language literature, teamwork, etc.; for instance, the seminars <i>academic writing</i> , <i>embodied communication</i> , and <i>time- and self-management</i> are offered regularly by the Elite Network of Bavaria; the programme coordinator and your mentor will give individual recommendations
<i>Duration</i>	1 semester
<i>Language</i>	English
<i>Teaching Method</i>	Participation in seminars
<i>Credit Points</i>	2
<i>Work Load</i>	In total 60 h of seminars, i.e. 3–4 seminars
<i>Recommended Prerequisites</i>	None
<i>Grading</i>	Proof of attendance
<i>Frequency</i>	
<i>Can also be used as</i>	

6 Module Section F: Master's Thesis

<i>Title</i>	Master's Thesis
<i>Module Label</i>	F
<i>Module Type</i>	Thesis
<i>Area of Research</i>	all areas
<i>Responsible</i>	Programme coordinator
<i>Learning Outcomes</i>	Ability to prepare a scientific work (larger than a bachelor's thesis)
<i>Content</i>	Scientific work in the area of Scientific Computing that should have a connection with application-driven questions and with the focus of this master's programme. In particular, interdisciplinary problems should be treated.
<i>Duration</i>	2 semesters
<i>Language</i>	English or German
<i>Teaching Method</i>	
<i>Credit Points</i>	30
<i>Work Load</i>	900 h (editing time: at most 10 months)
<i>Recommended Prerequisites</i>	
<i>Grading</i>	Written thesis
<i>Frequency</i>	
<i>Can also be used as</i>	

7 Recommended Curriculum

The following recommended curriculum assumes that the studies start in winter term.

a) Full-time study

Sem.	Numerical Maths	SWS	ECTS	Modeling	SWS	ECTS	Computer Science	SWS	ECTS	ECTS
1	A1 Numerical Methods for Differential Equations	4+2	6	B1 Applied Functional Analysis	4+2	8	C1-I	2+1	4	30
	D1-I	4+2	8	B2-I (non-math.)	2+1	4				
2	A2-I	4+2	8	B2-I (non-math.)	2+1	4	C1-II	4+2	8	30
	D1-II, B2-II or A2-II	4+2	8				C2 Parallel Num. Methods	P 2 w.	2	
3	F Master's Thesis		14	B3 Industrial Internship or D1-IV	P 6 w.	8				30
	D1-III	4+2	8							
4	F Master's Thesis		16	B4 Modeling and Status Seminar	S	8	E Soft Skills	S 60h	2	30
	D2 Special Skills in Scientific Computing	2+1	4							
	Numerical Maths		72	Modeling		32	Computer Science		16	120

b) Part-time study

Sem.	Numerical Maths	SWS	ECTS	Modeling	SWS	ECTS	Computer Science	SWS	ECTS	ECTS
1	A1 Numerical Methods for Differential Equations	4+2	6	B1 Applied Functional Analysis	4+2	8				14
2	D1-I	4+2	8				C1-I	4+2	8	16
3	A2-I	4+2	8	B2-I (non-math.)	2+1	4	C1-II	2+1	4	16
4	D1-II, B2-II or A2-II	4+2	8	B2-I (non-math.)	2+1	4	C2 Parallel Num. Methods	P 2 w.	2	14
5	D1-III	4+2	8	B3 Industrial Internship or D1-IV	P 6 w.	8				16
6	F Master's Thesis		15							15
7	F Master's Thesis		15							15
8	D2 Special Skills in Scientific Computing	2+1	4	B4 Modeling and Status Seminar	S	8	E Soft Skills	S 60h	2	14
	Numerical Maths		72	Modeling		32	Computer Science		16	120

The following recommended curriculum assumes that the studies start in summer term.

a) Full-time study

Sem.	Numerical Maths	SWS	ECTS	Modeling	SWS	ECTS	Computer Science	SWS	ECTS	ECTS
1	A2-I	4+2	8	B2-I (non-math.)	2+1	4	C1-I	4+2	8	28
	D1-I	4+2	8							
2	A1 Numerical Methods for Differential Equations	4+2	6	B1 Applied Functional Analysis	4+2	8	C1-II	2+1	4	32
	D1-II, I, B2-II or A2-II	4+2	8	B2-I (non-math.)	2+1	4	C2 Parallel Num. Methods	P 2 w.	2	
3	F Master's Thesis		14	B3 Industrial Internship or D1-IV	P 6 w.	8				30
	D1-III	4+2	8							
4	F Master's Thesis		16	B4 Modeling and Status Seminar	S	8	E Soft Skills	S 60h	2	30
	D2 Special Skills in Scientific Computing	2+1	4							
	Numerical Maths		72	Modeling		32	Computer Science		16	120

b) Part-time study

Sem.	Numerical Maths	SWS	ECTS	Modeling	SWS	ECTS	Computer Science	SWS	ECTS	ECTS
1	D1-I	4+2	8				C1-I	4+2	8	16
2	A1 Numerical Methods for Differential Equations	4+2	6	B1 Applied Functional Analysis	4+2	8				14
3	D1-II, B2-II or A2-I	4+2	8	B2-I (non-math.)	2+1	4	C2 Parallel Num. Methods	P 2 w.	2	14
4	A2-II	4+2	8	B2-I (non-math.)	2+1	4	C1-II	2+1	4	16
5	D1-III	4+2	8	B3 Industrial Internship or D1-IV	P 6 w.	8				16
6	F Master's Thesis		15							15
7	F Master's Thesis		15							15
8	D2 Special Skills in Scientific Computing	2+1	4	B4 Modeling and Status Seminar	S	8	E Soft Skills	S 60h	2	14
	Numerical Maths		72	Modeling		32	Computer Science		16	120