Electives Artificial Intelligence and Data Science Summer Semester

Faculty Computer
Science Date: 04.02.2025

Table of Contents

Electives
Artificial Intelligence and Data Science
Summer Semester

Faculty Computer
Science Date: 04.02.2025

Generell Information

The module handbook applies to Electives 1–4. Electives from the bachelor's course Artificial Intelligence AIN-B can only be chosen for Electives 1 and 2 in the first semester.

In addition to the subject-specific compulsory elective modules, which are offered in English, students may choose individual an advanced language course A2 as elective 4, in the same language that you have your language course A1.

The list includes only electives offered at DIT. Students can also choose additional electives offered at USB.

Important: If German is specified as the language of instruction for an elective course, then the examination will also be conducted in German!

AIN-B-11 Computational Logic

Module code	AIN-B-11
Module coordination	Thomas Ewender
Course number and name	AIN-B-11 Computational Logic
Lecturers	Prof. Dr. A Admin Thomas Ewender
Semester	2
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Undergraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	written ex. 90 min.
Duration of Examination	90 min.
Weighting of the grade	5/120
Language of Instruction	English

Module Objective

Students aquire understanding and hands-on experience of various logical systems and their usage in artificial intelligence applications..

Specifically, students will have achieved the following outcomes upon completion of the module:

Subject competency

Students understand the significance of logic for intelligent problem-solving.

Methodological competency

Students select the most appropriate logical system for solving a concrete practical problem, and use it to implement software-based solutions.

Personal competency

Students understand complex theoretical concepts and apply them to problems arising in practice.

Social competency

Students communicate clearly, argue and criticize logically and constructively, contribute to reasoned, team-oriented problem solving processes in the group.

Applicability in this and other Programs

Logic is foundational for all computer science courses and programmes. This module is a pre-requisite for the more advanced artificial intelligence lectures that build upon it.

Entrance Requirements

Recommended:

- Mathematics 1
- Foundations of Computer Science

Learning Content

Formal Logic: Syntax and Semantics

- Introduction to logical languages
- Basic concepts of logic
- Propositional logic
- Predicate (first-order) logic
- Formal proofs
- Set theory
- Classical semantics for first-order logic
- Resolution for propositional and first-order logic
- Semantics of logic programming

Logical Programming

- Prolog
- Answer Set Programming

Teaching Methods

- Interactive lectures
- Practical exercises using automatic proof checkers and theorem provers
- Software implementation of application-oriented examples

Recommended Literature

- Barwise, J und Etchemendy, J: Language, Proof and Logic , CSLI 2003 (or newer)
- Lifschitz, V.: Answer Set Programming , Springer Verlag 2019
- Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice , Morgan & Claypool Publishers, 2013

AIN-B-22 Computer Vision

Module code	AIN-B-22
Module coordination	Prof. Dr. Patrick Glauner
Course number and name	AIN-B-22 Computer Vision
Semester	4
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Undergraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	project work
Weighting of the grade	5/210
Language of Instruction	English

Module Objective

The aim of this class is to discuss Computer Vision (CV), which allows computers to process visual inputs. We deal every day dozens of times with CV, such as facial recognition, real-time translating camera input or auto-tagging friends in photos. Modern CV algorithms are strongly based on machine learning methods, in particular deep neural networks. Students will acquire knowledge in CV and be able to elaborate it further in the future, for example in projects or further studies. Overall, CV is a cutting-edge eld, with many high-pay opportunities for graduates.

Specifically, students will have achieved the following learning outcomes upon completion of the module:

Subject competency
Students will understand the concepts of the most common methods in computer vision. (2 - Understanding)

Methodological competency
Students will have the ability to develop high-quality programs using computer vision technologies. (3 - Apply)

Personal competency
Students will be able to implement their own algorithms and defend them against competing approaches. (6 - Create)

Social competency
Programming exercises take place as part of the course. Students are thus able to understand, critique, and complement programs of other students. (5 - Assess)

Applicability in this and other Programs

Including, but not limited to, the following modules:

- AI Project
- Deep Learning/Big Data

Entrance Requirements

- Programming, ideally in Python
- Algorithms and data structures
- (Some) mathematics

Learning Content

- Introduction: applications, computational models for vision, perception and prior knowledge, levels of vision, how humans see
- Pixels and filters: digital cameras, image representations, noise, filters, edge detection
- Regions of images and segmentation: segmentation, perceptual grouping, Gestalt theory, segmentation approaches, image compression
- Feature detection: RANSAC, Hough transform, Harris corner detector
- Object recognition: challenges, template matching, histograms, machine learning
- Convolutional neural networks: neural networks, loss functions and optimization, backpropagation, convolutions and pooling, hyperparameters, AutoML, efficient training, selected architectures
- Image sequence processing: motion, tracking image sequences, Kalman filter, correspondence problem, optical flow
- Foundations of mobile robotics: robot motion, sensors, probabilistic robotics, particle filters, SLAM

- Outlook: 3D vision, generative adversarial networks, self-supervised learning, vision transformers

Teaching Methods

- Lectures
- Projects

Recommended Literature

- C. Bishop and H. Bishop, " Deep Learning: Foundations and Concepts ", Springer, 2024.
- R. C. Gonzalez and R. Woods, " Digital Image Processing ", Pearson, 4th edition, 2018.
- I. Goodfellow, Y. Bengio and A. Courville, " Deep Learning ", MIT Press, 2016.
- S. Russell and P. Norvig, " Artificial Intelligence: A Modern Approach ", Pearson, 4th edition, 2021.

AIN-B-19 Natural Language Processing

Module code	AIN-B-19
Module coordination	Prof. Dr. Udo Garmann
Course number and name	AIN-B-19 Natural Language Processing
Semester	4
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Undergraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	Exercise Performance, written ex. 90 min.
Duration of Examination	90 min.
Weighting of the grade	5/210
Language of Instruction	English

Module Objective

The goal of this module is to learn Natural Language Processing (NLP), which enables computers to process human language. We engage in NLP dozens of times a day, such as performing a Google search, correcting spelling on a smartphone, classifying email as spam, or recognizing handwriting. Modern NLP algorithms are heavily based on machine learning methods. The students acquire knowledge of NLP and can deepen this in the future, e.g. in projects or further studies.

The students know terms from linguistics such as syntax, semantics, etc. They understand the different structures of language. Understand and apply regular expressions (analysis and application) in Python. The students know the Natural Language Toolkit (NLTK). You can use the NLTK for different forms of language processing.

In detail, the students have achieved the following learning outcomes after completing the module:

Professional competence

Students understand the concepts of the most common approaches to language processing. (2 - understanding)

Methodical competence

Students have the ability to create high quality programs using speech understanding technologies. (3 - Apply)

Personal competence

The students can implement their own methods and defend them against competing approaches. (6 - Create)

Social skills

Programming exercises take place as part of the course. The students are thus able to understand, criticize and complement the programs of other students. (5 - judge)

Applicability in this and other Programs

AI-Project

Deep Learning/Big Data

Entrance Requirements

Recommended:

Mathematics 2

Programming 2

Algorithms and Data structures

Learning Content

Basics: stemming, stopwords, n-grams
Text classification: Naïve Bayes, spam filtering, speech recognition, logistic regression spelling correction
Search engines: ranking, vector space model, PageRank
Basics of formal languages (related to NLP problems)
Regular Expressions and Finite State Machines (Related to NLP Problems)
Context-free grammars (related to NLP problems)
Analysis of the speech signal
Outlook: Embeddings, current advances in NLP

Teaching Methods

Lectures
Discussion of scientific articles and breaking news
Exercises, including computer exercises (proof of achievement)

Recommended Literature

- S. Bird, E. Klein and E. Loper, " Natural Language Processing with Python Analyzing Text with the Natural Language Toolkit ", Online at [NLTK website](https://www.nltk.org/book), visited 20/03/31.
- C. Bishop, " Pattern Recognition and Machine Learning ", Springer, 2006.
- D. Jurafsky, " Speech and Language Processing, An Introduction to Natural Language Processing ", Computational Linguistics, and Speech Recognition, Third Edition draft, available online at [Jurafsky:Homepage](https://web.stanford.edu/~jurafsky), visited 20/03/31.
- C. Manning, P. Raghavan and H. Sch#ütze, " Introduction to Information Retrieval ", Cambridge University Press, 2008.
- B. Pfister und T. Kaufmann, " Sprachverarbeitung, Grundlagen und Methoden der Sprachsynthese und Spracherkennung ", 2., aktualisierte und erweiterte Auflage, Springer-Verlag GmbH Deutschland 2017, ISBN 978-3-662-52837-2.
- S. Russel and P. Norvig, " Artificial Intelligence: A Modern Approach ", Prentice Hall, third edition, 2009.

AIN-B-20 Human Factors and Human-Machine Interaction

Module code	AIN-B-20
Module coordination	Prof. Dr. Christina Bauer
Course number and name	AIN-B-20 Human Factors and Human-Machine Interaction
Semester	4
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Undergraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	Portfolio
Weighting of the grade	5/210
Language of Instruction	English

Module Objective

Students understand and communicate the fundamental concepts of human-machine Interaction.

Specifically, students will have achieved the following outcomes upon completion of the module:

Subject competency

- Application of human factor principles to a specific domain
- Identification of various influences on the quality of work and interaction

Methodological competency

- Knowledge of various methodological approaches for investigating and evaluating human-machine interaction

- Systematic analysis and classification of situational influences
- Systematic analysis of error sources and types

Personal competency

- Realistic assessment of systemic influences on the work situation
- Improvement of team skills through knowledge of group mechanisms

Social competency

Students evaluate different user interface designs in exercise sessions. Thus, they able to understand and criticize different design decision and can justify their analyses.

Applicability in this and other Programs

All modules in which the consideration of human-computer-interaction mechanisms is a central subject.

Entrance Requirements

none

Learning Content

Introduction to the field of human-machine interaction

- Design of everyday objects
- Cognitive fundamentals
- Phenomena and mechanisms of attention

Information design

- Presentation of information
- Display design principles

Usability, UX

- Terms, models, processes
- Analysis methods
- Evaluation methods

Teaching Methods

- Interactive lectures
- Exercise sessions
- Group work

Recommended Literature

- Krug, S. (2013), Dont Make Me Think: A Common Sense Approach to Web Usability, 3rd revised edition, New Riders
- Norman, D. A. (2013), The design of everyday things, Basic Books, New York, NY
- Shneiderman, B., & Plaisant, C. (2010), Designing the user interface: strategies for effective human-computer interaction, Addison-Wesley, Boston

AIX-M-2 Datacenter Network Programming

Module code	AIX-M-2
Module coordination	Prof. Dr. Andreas Kassler
Course number and name	AIX-2 Datacenter Network Programming
Lecturer	Prof. Dr. Andreas Kassler
Semester	2
Duration of the module	1 semester
Module frequency	annually
Course type	compulsory course
Level	
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 90 hours self-study: 60 hours Total: 150 hours
Type of Examination	written ex. 90 min.
Duration of Examination	90 min.
Weighting of the grade	5/210
Language of Instruction	English

Module Objective

Students acquire understanding and hands-on experience of how the data plane of modern datacenter networking equipment can be programmed using the high-level and popular programming language P4 (see http://p4.org). They learn the basic concepts of the P4 language and understand, how offloading simple computational tasks to the data plane of programmable networking devices (such as datacenter routers or network cards) can be used to speed up the performance of Deep Learning, Big Data Analytics use-cases within modern datacenters. They understand, how the data plane can be used to accelerate distributed high-performance computing (HPC) building blocks including distributed key-value stores, where load-balancing and network monitoring of the datacenter networking fabric is important for achieving high speed and low latency.

They setup their own development environment in the network emulator Mininet and implement simple data plane programs in the P4 language. They know how to use P4 to parse packet headers, apply different actions and modify packets before forwarding them. They know basic P4 constructs, how to store stateful information (e.g. parts of a neural network) and how to perform simple computational tasks in the data plane.
Based on this knowledge and understanding, students implement a small-scale project in a team. They use their acquired knowledge on P4 and programmable datacenter networking. They evaluate the results of other project groups and get evaluated by other groups. For this project work, they have used standard tools (Mininet, P4 toolchain, command line interface) for programming the data plane of an (emulated) datacenter router.
After finishing this module, students can design, implement and evaluate their own P4 programs using the network emulator Mininet.

Specifically, students will have achieved the following outcomes upon completion of the module:
Subject competency
Students understand the significance of datacenter network programming and how datacenter network programming can be used to improve the performance of distributed high-performance applications such as distributed training and inference of large-scale ML models.
Methodological competency
Students select the most appropriate P4 constructs for solving a concrete practical problem and use it to implement a concrete use-case of a data plane program.
Personal competency
Students understand complex theoretical concepts and apply them to problems arising in practice.
Social competency
Students communicate clearly, argue and criticize logically and constructively, contribute to reasoned, team-oriented problem-solving processes in the group.

Applicability in this and other Programs

This Module is suitable for the following programs:
- Master in Angewandte Informatik/Infotronik
- Master in Artificial Intelligence and Data Science
- Master in High Performance Computing/Quantum Computing

Entrance Requirements

Students should have basic understanding of Network Technologies and/or Communication Networks. Basic knowledge of Programming and basic knowledge in Python helps in the Project Part of the course.

Learning Content

The Module is decomposed into two parts:
Part I: ?Introduction to Datacenter Network Programming? and Part II ?Project in Datacenter Network Programming?
Content Part I:
(1) Introduction to Programming the Data Plane of a Datacenter networking device:
- Difference between Data and Control Plane
- Introduction to P4 language
- P4 programming model
- Compiling and deploying P4 programs
- P4 Targets: Behavioral Model (BMv2), Programmable Switching ASIC Intel Tofino, Mellanox Bluefield DPU, Netronome SmartNIC
- Basic P4 concepts: header parsing, applying tables and actions, header rewriting.
- Workshop: Setup Development environment with Mininet and Command Line Interface (CLI), implement, test and debug simple P4 language constructs and programs using the Mininet network emulator

(2) Datacenter Networking and Load Balancing:
Faculty Computer Science
Artificial Intelligence and Data Science
Date: 22.11.2022
- Datacenter networking fundamentals, routing and forwarding within the datacenter networking fabric
- Workshop: Advanced P4 concepts: stateful information, register arrays, counters and meters.
- Loadbalancing in Datacenter networks, Equal Cost Multipath Routing, Conga, Hula
- Workshop: Implementing ECMP in P4

(3) In Network support for Monitoring and Caching:
- Active and passive network monitoring
- Inband Network Telemetry (INT) for fine-granular network monitoring
- Accelerating Distributed Key-value stores in the data plane of the data center
- Using telemetry for fine-grained loadbalancing
- Workshop: Implementing Hula and INT in P4

(4) In Network support for Distributed Machine Learning:
- Role of the datacenter network for distributed training and inference
- In network support for Distributed Machine Learning Inference for in-switch traffic classification
- Mapping trained machine learning models (decision trees, SVMs, neural networks) to programmable data plane devices
- In network support for distributed training within a datacenter network

Content Part II:

Project: Implementation of your own small dataplane program in P4 and testing it in the Mininet network emulator.

Teaching Methods

- Interactive lectures
- Practical workshop style exercises using the network emulator Mininet
- Software implementation of your own P4 program

Remarks

The module is comprised of two parts. The second part (project work) can be done in groups of max. 3 students.

Recommended Literature

Recommended Literature will be provided at the start of the course by a set of research and practical oriented articles that are available online.

LSI-12 Data Visualization

Module code	LSI-12
Module coordination	Prof. Dr. Phillipp Torkler
Course number and name	LSI-12 Data Visualization
Lecturers	Prof. Dr. Phillipp Torkler Prof. Dr. Javier Valdes
Semester	2
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 45 hours virtual learning: 45 hours Total: 150 hours
Type of Examination	written student research project
Weighting of the grade	5/90
Language of Instruction	English

Module Objective

Data Visualization is the graphic representation of a data analysis to achieve clear and effective communication of results and insights. Complex ideas are presented in charts and graphs with the goal of quickly and easily disseminating key, actionable information. Data visualization is an essential part of data science and analytics, especially when working with large, complicated data sets like sequencing data. The visualization tells a story, whether as a stand-along graph or combined with other graphs, charts and design elements in an infographic or dashboard.

After completing the Data Visualization module, students will have obtained the following learning competencies:

Professional competence

After successfully completing the module, students will:

- know the data visualization principles.
- be familiar with file formats and their usage in the different analysis approaches.
- know about common data analysis workflows and be able to interpret and visualize the achieved results.

Methodological competence

After successfully completing the module, students will:

- know how to use ggplot2 in R to create custom plots.
- know how to use matplotlib and Python to create custom plots.

Social competence

- Interdisciplinary and interpersonal collaboration when working together in small groups on developing R and Python scripts for data analysis and data visualization.
- Working together with fellow-students in small groups on designing and developing biostatistical validation of biomedical datasets within R and/or Python.

Applicability in this and other Programs

master seminar, master thesis

Entrance Requirements

Recommended or advantageous:

Basic Knowledge in R

Module: LSI-04 *Biostatistics I*

Learning Content

1. R Packages for data visualization
2. Open access visualization tools
3. Matplotlib and other Python packages for data visualization
4. Theoretical Background
5. Perception And Interpretation

Teaching Methods

Tutorial, practical exercises, application examples

The module consists of an interactive theoretical part with blended learning components. Within the tutorial the students use example NGS datasets to perform the biomedical data visualization. In the practical part of the tutorial the students should learn to find various visualization tools, possibilities and methods and discuss their advantages and disadvantages to represent statisitical significance.

Remarks

The iLearn teaching and learning platform provides students with additional literature references and learning material to prepare for the lectures.

Recommended Literature

Detailed lecture notes are available online for preparation and follow-up work

- The Biostars Handbook: Bioinformatics Data Analysis Guide; 2019; https://www.biostarhandbook.com/

HPC-04 Software Engineering

Module code	HPC-04
Module coordination	Prof. Dr. Christoph Schober
Course number and name	HPC-4 Software Engineering
Lecturer	Prof. Dr. Christoph Schober
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	written ex. 90 min.
Duration of Examination	90 min.
Weighting of the grade	5/90
Language of Instruction	English

Module Objective

Software Engineering for HPC/QC aims at bringing the students of various backgrounds to a common understanding of the processes required to deliver software fulfilling the requirements with high quality. This includes theoretical knowledge about classical software engineering, but focus clearly on practical knowledge and skills required in modern software development such as version control, automated testing, containerization and Continuous Integration and Delivery (CI/CD).

Professional skills

Students will know how software projects are managed and which different methodologies exists. Within the context of HPC/QC they are able to understand advantages and disadvantages of each method and know the differences to other fields of software engineering. They learn how software requirements are collected, prioritized and planned for implementation in an agile development process. Students learn about modern technology and tooling such as version control, automated testing, containerization and DevOps.

Methodological skills

Within the course the students will apply the knowledge of each theoretical block in short in-class and homework exercises, enabling them to work through a practical software project from requirements engineering to productive deployment using CI/CD. They get a hands-on impression of a set of tools and frameworks used in the industry and can use this knowledge to quickly understand any similar tool.

Social skills

Students understand the importance of communication and cooperation between stakeholders (internal and external) and the development team. They experience and practice this with exercises in small groups.

Personal skills

With the knowledge of this course the students will understand the importance of modern engineering technology to deliver high quality software. This enables them to work both in academic or industrial settings with ease and focus on the value of their work delivered to their stakeholders.

Applicability in this and other Programs

Software design and programming lectures

Entrance Requirements

None

Learning Content

The module is organized along the stages of the Software Development Lifecycle.

Introduction

- Software Engineering and HPC/QC
- Scientific software development

Requirement Analysis and Planning

- What is a project?
- How are software projects managed? (Waterfall, Agile)
- Agile by example: Scrum
- User stories, estimation, priorization and planning

Software Development and Testing

- Version control with Git
- Platforms for working with Git
- Code Reviews and Code Quality
- Testing
  - Unit-, Integration- and End2End-testing
  - Test automation
  - Test coverage
- Writing testable code
  - The role of architectures (MVC, Hexagonal,)
  - Design patterns

Deployment

- Containerization
- Introduction to DevOps
- CI with Gitlab
  - Test automation
  - Test coverage
  - Code Quality Metrics
- CD with Gitlab
  - Build and Package
  - Automated Deployment

Teaching Methods

Lecture with exercises

Recommended Literature

Online Resources

- Introduction to Git: https://git-scm.com/docs/gittutorial
- Introduction to Gitlab: https://docs.gitlab.com/ee/tutorials/
- Introduction to Gitlab CI/CD: https://docs.gitlab.com/ee/ci/

Books

- Software Engineering: Basic Principles and Best Practices (Ravi Sethi)
- Scrum for dummies (ISBN 978-1-119-90467-0): https://ebookcentral.proquest.com/lib/th-deggendorf/detail.action?docID=7109023 (English)
- Scrum: kurz & gut (ISBN 9783868998337) (German)
- Andrew S. Tanenbaum; Herbert Bos. Modern Operating Systems. Prentice Hall, 4th ed. 2014
- Evi Nemeth, Garth Snyder, Trent R. Hein et al. Unix and Linux System Administration Handbook. Addison-Wesley, 5th ed. 2018

- Christine Bresnahan, Richard Blum. Mastering Linux system administration. Wiley. 2021. https://ebookcentral.proquest.com/lib/th-deggendorf/detail.action?docID=6658986

HPC-07 HPC/QC Technology

Module code	HPC-07
Module coordination	Prof. Dr. Helena Liebelt
Course number and name	HPC-7 HPC/QC Technology
Lecturer	Prof. Dr. Helena Liebelt
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	StA
Weighting of the grade	5/90
Language of Instruction	English

Module Objective

In this module, students delve into the intricacies of High Performance Computing (HPC) and/or Quantum Computing (QC) systems, gaining a comprehensive understanding of the technological challenges specific to these cutting-edge fields. Through a blend of theoretical learning and hands-on experience, students familiarize themselves with the hardware technologies essential to these domains. Through practical sessions, they acquire invaluable skills in system assembly and configuration, empowering them to proficiently set up components of modern HPC systems. Additionally, students learn the nuances of system installation and configuration on a smaller scale, equipping them with the capability to effectively deploy and manage these advanced computing systems.

Professional skills

Professional Skills: Students develop a range of professional skills essential for success in the field of High Performance Computing (HPC) and/or Quantum Computing (QC) systems. They refine their ability to analyze complex technological issues specific to these domains, employing critical thinking and problem-solving techniques to address challenges effectively. Through hands-on experience in system assembly and configuration, students enhance their technical proficiency, ensuring they are adept at deploying and managing modern HPC systems. Furthermore, students cultivate strong communication skills, enabling them to articulate their ideas and solutions clearly to peers and industry professionals.

Methodological skills

This module hones students' methodological skills, providing them with a structured framework for approaching problems in HPC and/or QC systems. Students learn systematic approaches to system setup, installation, and configuration, ensuring precision and efficiency in their work. They develop robust methodologies for troubleshooting and debugging, equipping them with the ability to identify and resolve issues promptly. Additionally, students learn to conduct thorough research, staying abreast of the latest advancements in hardware technologies relevant to the field.

Social skills

In addition to technical expertise, students refine their social skills, recognizing the collaborative nature of the HPC and/or QC ecosystem. Through group projects and collaborative tasks, students learn to work effectively as part of a team, leveraging each other's strengths to achieve common goals. They develop interpersonal skills such as active listening, constructive feedback, and conflict resolution, fostering a positive and productive team dynamic. Furthermore, students engage in networking opportunities with industry professionals, enhancing their ability to build and maintain professional relationships within the field.

Personal skills

This module also focuses on the development of personal skills essential for professional growth and success. Students cultivate traits such as adaptability and resilience, learning to navigate the dynamic landscape of HPC and/or QC systems with confidence. They hone their time management and organization skills, balancing academic coursework with hands-on practical sessions effectively. Additionally, students foster a growth mindset, embracing challenges as opportunities for learning and growth. By prioritizing self-reflection and continuous improvement, students develop into well-rounded individuals prepared to excel in the rapidly evolving field of high-performance computing.

Applicability in this and other Programs

Hardware / system design for complex modern computing systems

Entrance Requirements

Prerequisites for this module on a master's level would typically include:

1. Foundational Knowledge in Computer Science or Related Field: Students should have a solid understanding of computer science fundamentals, including data structures, algorithms, computer architecture, and operating systems.
2. Programming Proficiency: Proficiency in at least one programming language commonly used in high-performance computing, such as C/C++, Python, or Fortran, is essential. Students should be comfortable writing, debugging, and optimizing code.
3. Mathematical Background: A strong background in mathematics, particularly in areas such as linear algebra, calculus, and probability theory, is necessary for understanding the underlying principles of high-performance computing and quantum computing.
4. Understanding of Parallel Computing Concepts: Familiarity with parallel computing concepts and techniques is crucial, including parallel algorithms, parallel programming models (e.g., MPI, OpenMP, CUDA), and parallel computing architectures.
5. Basic Knowledge of Hardware Systems: Students should have a basic understanding of computer hardware components and architecture, including processors, memory systems, storage devices, and networking.
6. Prior Experience with Operating Systems: Familiarity with operating systems concepts and administration is beneficial, as students will be involved in setting up and configuring computing systems.
7. Experience with Command-Line Interface (CLI) Tools: Proficiency in using command-line interface tools and Unix/Linux operating systems is important for executing commands, managing files, and interacting with the computing environment.
8. Probability and Statistics: Some familiarity with probability and statistics is advantageous, especially for students interested in quantum computing, as it provides the foundation for understanding quantum algorithms and quantum information theory.
9. Critical Thinking and Problem-Solving Skills: Strong critical thinking and problem-solving skills are essential for analyzing complex technological issues and developing effective solutions in the context of high-performance and quantum computing systems.
10. Research Skills: Students should possess basic research skills, including the ability to gather, evaluate, and synthesize information from academic literature, technical documentation, and online resources related to high-performance and quantum computing.

These prerequisites ensure that students have the necessary background knowledge and skills to fully engage with the advanced topics covered in the module and to successfully complete hands-on practical sessions and assignments.

Learning Content

The aim of this course is to discover the technological paticularities of HPC and QC systems.

The module is divided into two parts, both covering theoretical as well as practical aspects including hands-on sessions:

- Hardware
  - Setting up a compute node
  - Rack technologies
  - Cooling aspects
- Software
  - Setting up an operating system
  - Middleware
  - Access and scheduling

Teaching Methods

Lecture with lab sessions / exercises

Recommended Literature

- Andrew S. Tanenbaum; Herbert Bos. Modern Operating Systems. Prentice Hall, 4th ed. 2014
- Evi Nemeth, Garth Snyder, Trent R. Hein et al. Unix and Linux System Administration Handbook. Addison-Wesley, 5th ed. 2018
- Christine Bresnahan, Richard Blum. Mastering Linux system administration. Wiley. 2021. https://ebookcentral.proquest.com/lib/th-deggendorf/detail.action?docID=6658986
- Further literature as indicated in the lecture

AIX-M-11 Quantum Chemistry

Module code	AIX-M-11
Module coordination	Prof. Dr. Christoph Schober
Course number and name	AIX-M-11 Quantum Chemistry
Lecturer	Prof. Dr. Christoph Schober
Semester	2
Duration of the module	1 semester
Module frequency	annually
Course type	compulsory course
Level	postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	oral ex. 20 min.
Weighting of the grade	according to ECTS
Language of Instruction	English

Module Objective

The module provides an introduction to simulation of materials for students of computer science. It aims to do so without
requiring deeper knowledge of chemistry or quantum mechanics, but a general understanding of chemistry/physics on highschool level is advantageous.

Questions that will be answered are:

- What are typical problems that are solved?
- Is quantum chemistry only suitable for high performance computers?
- Is quantum chemistry used in the industry?

Knowledge and understanding

The students will obtain a broad overview of the field of quantum chemistry and its different flavors (such as wave-function based methods and density based methods to calculate properties of materials). They gain an understanding of the computational complexity and scalability of different levels of theory and their requirements in terms of computational power. Knowing the constraints of classical quantum chemistry students will be able to understand the potential of Quantum Computing for quantum chemistry. They understand the ideas behind the Variational Quantum Eigensolver for NISQ (Noisy Intermediate Scale Quantum) devices.

Application, utilisation and generation of knowledge

The students are able to assess the basic applicability of a computational quantum chemistry method for different simulation tasks in material science or pharmaceutical research. With their broad overview students are able to dive deeper into specific topics by using the appropriate scientific literature.

Communication and Cooperation

Students learn to express requirements and translate them from an non-domain perspective (e.g., a scientist requiring a computational scanning probe image to compare with an experimental image) to a domain specific solution (e.g., calculating the electron density using DFT to gennerate a STM image)

Applicability in this and other Programs

This module can be used as elective (master level) for other degrees

Entrance Requirements

- Linear Algebra (matrices, dot product, ...)
- Familiarity with Python or other scripting languages
- Basic knowledge of quantum mechanics is advantageous, but not a requirement

Learning Content

The course will start with an introduction of quantum chemistry for non-(quantum)-chemists, a brief history and applications in academia and industry.

We will then look at one of the foundations of quantum chemistry, the Hartree-Fock method, to solve the Schrödinger equation. You will create a basic implementation of Hartree-Fock (almost) from scratch using Python and calculate the properties of some simple systems.

Building up on this knowledge we look at the current "zoo" of quantum chemistry methods and their possibilities (and limitations). You will then use ASE, the "atomic simulation environment" (https://wiki.fysik.dtu.dk/ase/), to calculate properties of materials (such as the structure of water) using different methods and existing implementations.

In the final part of the course we revisit the limitations of classical quantum chemistry and take a look at quantum computing and the promises it holds for solving some really hard technological questions that are currently out of reach for classical quantum chemistry.

Teaching Methods

Lecture with exercises, coding exercises

Remarks

The lecture will be held in English.

Recommended Literature

- Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory - Attila Szabo, Neil S. Ostlund (ISBN 0486691861)
- Introduction to Computational Chemistry - Frank Jensen (ISBN 1118825993)
- A Chemist's Guide to Density Functional Theory - W. Koch, M. Holthausen (ISBN 9783527303724)

AIX-M-16 ChatGPT et al.: Generative AI with Transformers

Module code	AIX-M-16
Module coordination	Prof. Dr. Andreas Fischer
Course number and name	AIX-M-16 ChatGPT et al.: Generative AI with Transformers
Lecturers	Zineddine Bettouche Prof. Dr. Andreas Fischer
Semester	2
Duration of the module	1 semester
Module frequency	annually
Course type	compulsory course
Level	
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	written student research project
Weighting of the grade	
Language of Instruction	English

Module Objective

Entrance Requirements

AIX-M-16 ChatGPT et al.: Generative AI with Transformers

Entrance Requirements

Substantial background in artificial intelligence

Learning Content

The module will give an introduction to the transformer technology which drives modern large language models. Covered topics are:

- Foundations of Language Models
- Word Embeddings
- Attention Mechanism
- Architectures of Transformer Models
- Popular Open Source Transformer Models
- Limitations of Large Language Models
- Applications of Transformers in and beyond NLP
- Optimization of Transformer Models

Type of Examination

written student research project

Methods

Seminaristic education

Recommended Literature

- Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

- Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

MET-01 ADVANCED PROGRAMMING TECHNIQUES

Module code	MET-01
Module coordination	Prof. Dr. Andreas Wölfl
Course number and name	MET 1101 Advanced Programming Techniques
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	written examination
Weight	5/90
Language of Instruction	English

Module Objective

Students of this course extend their software programming abilities by creating and maintaining a complex computer program in a development team. They learn the interplay between the design, maintenance and extension steps as applied to a complex software project.

The students achieve the following learning objectives:

Professional Skills

The students know the elementary workings as well as the application area of versioning control software. They are able to make good use of such a system in the context of a software development process.

The students extend their knowledge in the area of object-oriented programming and are able to confidently apply this programming paradigm to solve complex problems. The know the basic UML tools and can use them to design an appropriate software architecture to solve simple problems.

The students are familiar with basic programming patterns. They are able to implement them where appropriate in their own code. They know about the development method of test-driven development and are able to create software tests with which they can estimate the reliability of the software they are developing.

Methodological Skills

The students are able to realize and extend a software project. They can quickly acquaint themselves with a pre-existing code-base and identify appropriate points for extending this code-base. They are able to perform a requirements analysis for these extensions and to develop the respective solutions.

Soft Skills

The students realize a complex software project embedded in a development team. They are able to coordinate the development process appropriately with their team members. They can take professional feedback and implement the appropriate changes to their work.

Applicability in this and other Programs

For this degree program:

Compulsory subject in Electrical Engineering and Information Technology (Master); joint study, both main subjects

For any other degree program:

Elective for Master Applied Research in Engineering Sciences

Entrance Requirements

Formally: none

In terms of content: Basic education in computer science, proficiency in an object-oriented programming language

Learning Content

Using versioning control software

The software development process

Requirements analysis

Software architecture with UML

Software design patterns

Unit tests

Test-driven development

Teaching Methods

Lecture with practical exercises

Remarks

Contribution to open-source projects

Recommended Literature

R. Martin: Clean Code: A Handbook of Agile Software Craftsmanship, 1. Auflage, Prentice Hall 2008.

M. Fowler: Patterns of Enterprise Application Architecture, 1. Auflage, Addison Wesley 2002.

E. Gamma / R. Helm / R. Johnson / J. Vlissides: Design Patterns. Elements of Reusable Object-Oriented Software, 1. Auflage, Prentice Hall 1994.

A. Hunt / David Thomas / W. Cunningham: The Pragmatic Programmer. From Journeyman to Master, 1. Auflage, Addison Wesley 1999.

MET-13 ADVANCED MODELLING AND SIMULATION

Module code	MET-13
Module coordination	Prof. Dr. László Juhász
	Automatisierungstechnik (AT)
Course number and name	MET 1106 Advanced Modelling and Simulation
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	written ex. 120 min.
Duration of Examination	120 min.
Weight	5/90
Language of Instruction	English

Module Objective

The students deal first with problems related to mathematical modelling, parameter identification, simulation and digital control of technical and especially mechatronics systems. Furthermore, they learn about the basics of model-based control-design techniques. Here, not only the user's sight, but also the developer's tasks are discussed. Thus, students learn how to perform successful digital control-design and -testing using the model-based control-design methods. They are in position to successfully synthetize such control design as well as to critically evaluate it.

The students achieve the following learning objectives:

Professional Skills

The students are able to describe technical systems by means of mathematical modelling. They know the basic methods for parameter identification of technical and mechatronics systems and they apply such methods in practical exercises. They have the knowledge to create and verify parametrized mathematical models of technical systems.

Students have deep knowledge about digital control systems and their application during control-design for mechatronic systems.

Students learn about the basics and advanced methods of model-based digital control-design.

Students are familiar with the individual elements of the model-based design technology according to the V-cycle and understand the common elements and the differences in between them. They are able to evaluate the designed digital controller by means of offline- and real-time simulation according to the V-cycle standards. On this way they are able to uncover errors in the control design in the early development stage.

The Students learn about the software tool-chains based on MathWorks and dSPACE tools and can apply such tools using the earned knowledge and experience independently and holistically for the tasks of digital control-design and -testing.

Methodological Skills

Students are familiar with the common methods for mathematical modelling of technical and mechatronics systems and are able to apply such methods successfully.

Students learn about the methods of parameter identification in time and frequency domain and use these methods for practical exercises.

Furthermore, students are familiar with the most important methods used by digital control-design and are able to use these methods successfully.

Students are familiar with the common methods and tools used by the model-based control-design and are able to apply these methods successfully. Particularly, the stability criteria of digital simulation used for investigation of analogous and discrete plants and control systems are well-known. Students are familiar with guidelines for appropriate design of individual control functions regarding their later application in RCP, HIL and automatic production code. The earned skills are consolidated through practical exercises dealing with modelling, code generation and control of an example application.

Students are familiar with the meaning of real-time requirements and its impact on control-design and testing by RCP. They are able to apply thins knowledge both for software and hardware requirements during RCP process successfully. Thus, students are able to successfully perform function-prototyping by means of RCP for CPU-based systems and test their design appropriately. Especially he can clarify and analyze the problems of tasking, configuration of I/O devices and their impact on the real-time capability.

Students can overview the problems which may arise by the automatic production code generation and are able to apply optimization methods for minimization of the CPU-load and memory consumption of the ECU. Especially the design of the optimal numerical representation of the controller by means of fixed-point data types and scaling is treated here with emphasis. He is able to create optimal production code based on a functional model and to perform all the necessary steps on this way in successful manner. The student is familiar with the testing of the created production code by means of various simulation types, like MIL, SIL and PIL. He knows the basics of the integration of the ECU code towards the production prototype.

Students are familiar with the common methods of the HIL-Simulation and they are able to design and execute a HIL-Simulator for testing of production ECUs. Students can understand the synergies between the RCP and HIL and are able to apply test-automatization and virtualization.

Soft Skills

The students are aware their responsibility when work as developer in tasks of model-based control design and –testing. They are able to assess individual development steps and are prepared to give feedback and successfully work together in development teams.

Applicability in this and other Programs

For this degree program:

Compulsory subject within Master-Program Electrical Engineering and Information Technology, focus automation engineering (AT)

For other degree program:

Optional subject for General Engineering.

Elective for Master Applied Research in Engineering Sciences

Entrance Requirements

Formally: none

Essential thematic prerequisites: Mathematical modelling of linear time-invariant systems, physical basics and modelling approaches for mechanical and electrical systems, analogous and digital control design, advanced knowledge of programming language C.

Learning Content

1. Mathematical modelling of technical systems

    1.1. Common approaches for mathematical modelling of technical systems

    1.2. Mathematical modelling of mechanical systems

    1.3. Mathematical modelling of electrical systems

    1.4. Mathematical modelling of hydraulically systems

    1.5. Mathematical modelling of heat-transfer systems

1.6. Mathematical modelling of mechatronic systems

1.7. Linearization of non-linear systems in steady-state

1.8. Description of technical systems by means of state-space equations

2. Digital control

2.1. Discrete description of technical systems and the digital control loop

2.2. Discretization of analog plants

2.3. Difference equations and the Z-transformation

2.4. Stability of discrete systems

2.5. Methods of digital control-design

2.6. Discrete state-space equations

3. Parameter-identification of technical and mechatronical systems

3.1. Overview of methods for parameter-identification

3.2. Parameter-identification using time-domain

3.3. Parameter-identification using frequency-domain

3.4. Parameter-identification methods based on spectroscopy

3.5. Parameter-identification methods based on spectral analysis

4. Elements of model-based control-design and -testing

4.1. Model-based control-design according to the V-model

4.2. Offline Simulation

4.3. Rapid Control Prototyping

4.4. Production code generation

4.5. Hardware-in-the-Loop Simulation

4.6. Measurement and calibration

5. Practical exercises

5.1. Modelling and simulation of technical systems: among others example of an electrical throttle-valve

5.2. Model-based control design and –testing for the position-control of an electrical throttle-valve

5.3. Testing with RCP through example application: control of an electrical throttle-valve

5.4. Production-code generation, various examples (among others, control of an electrical throttle-valve)

Teaching Methods

Teaching lessons, practical exercises (modelling, simulation, control design, testing), individual and group work

Remarks

Tutorial

E-learning plattform

Recommended Literature

R. Woods / K. Lawrence: Modeling and Simulation of Dynamic Systems. Prentice Hall 1997.

D. Abel / A. Bollig: Rapid Control Prototyping (in German). Springer 2013.

H. Schildt: C++ The Complete Reference, Part I: the C subset. Springer 2013.

Ljung: System Identification: Theory for the User, 2/E. Prentice Hall 1999.

Gajic: Linear Dynamic Systems and Signals. Prentice Hall 2002.

N. Nise: Control Systems Engineering. John Wiley & Sons 2004.

R. Dorf / R. Bishop: Modern Control Systems. Pearson Educational International 2005.

R. Isermann: Grundlegende Methoden, Identifikation dynamischer Systeme, Bd.1. Springer-Verlag 1992.

R. Isermann: Identifikation dynamischer Systeme II. Besondere Methoden, Anwendungen. Springer-Verlag 1992.

M. Gipser: Systemdynamik und Simulation. Teubner-Verlag 1999.

R. Nollau: Modellierung und Simulation Technischer Systeme (in German). Springer 2009.

ASE-02 Digital Car / Innovation Management & Customer Design

Module code	ASE-02
Module coordination	Prof. Dr. Markus Straßberger
Course number and name	ASE-02 Digital Car / Innovation Management & Customer Design
Lecturer	Prof. Dr. Markus Straßberger
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	Postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	Portfolio
Weighting of the grade	5/90
Language of Instruction	German

Module Objective

Die Studierenden erhalten einen Einblick in die Herausforderungen und Anforderungen der aktuellen digitalen und vernetzten Automobiltechnik sowie in deren technologischen Ansätze und Lösungen. Darüber hinaus werden die methodischen Grundzüge des Innovationsprozesses in der Automobilindustrie, des nutzerorientierten Designs und des Lean-Development vermittelt.

Die Studierenden erreichen die folgenden Lernziele bzgl. Fach- und Methodenkompetenzen

Die Studierenden sind in der Lage, die Komplexität einer digitalen Fahrzeugfunktion, deren Abhängigkeiten und die wesentlichen Kostenfaktoren sowie die größten Fallstricke bei der Realisierung der jeweiligen Funktionalität im automobilen Umfeld zu verstehen. Sie können sich leicht in jedes digitale Fahrzeugprojekt einarbeiten.

Entrance Requirements

Learning Content

- Grundlagen des digitalen und vernetzten Fahrzeugs
- Abhängigkeiten und Komplexität in der Fahrzeugentwicklung
- Metoden des Innovationsmanagements im Automobilsektor
- Nutzerorientiertes Design und Lean Development im Kontext digitaler Fahrzeuge

Teaching Methods

Lehre in Form von seminaristischem Unterricht und Gastvorträgen aus der Automobilbranche.
Hands-On Gruppenarbeiten mit dem Ziel der Erarbeitung neuer Produktideen uaf Basis nutzenorientierten Desgins.

Type of Examination

project work

ASE-03 Advanced Driver Assistance Systems

Module code	ASE-03
Module coordination	Prof. Thomas Limbrunner
Course number and name	ASE-03 Advanced Driver Assistance Systems
Lecturer	Prof. Thomas Limbrunner
Semester	1
Duration of the module	1 semester
Module frequency	annually
Course type	required course
Level	postgraduate
Semester periods per week (SWS)	4
ECTS	5
Workload	Time of attendance: 60 hours self-study: 90 hours Total: 150 hours
Type of Examination	Portfolio
Weighting of the grade	5 ECTS
Language of Instruction	English

Module Objective

Students are given a basic overview of the systematics of driver assistance systems and the interaction of the components involved. The aim is to gain an overall system understanding of the topology in the vehicle and to highlight the key aspects of the development and function of driver assistance systems.

Applicability in this and other Programs

Master Automotive Software Engineering, Master Applied Research, Master AI, Bachelor Cybersecurity, B-AI, MT-B, M-AID

Entrance Requirements

Undergraduate studies

Learning Content

- Overview of driver assistance systems (definition, classification of relevant terms, classification, areas of application, legal aspects,
  NCAP, ...)
- System overview of the vehicle from the perspective of driver assistance, understanding the functional chains, K-matrix,
  mapping of signals
- Sensor technology, measurement and functional principle, such as camera (mono, stereo), lidar, radar, ultrasound, EGO data
- Central vehicle computer, domain controller, sensor fusion

Note: The content of the course may change over time and will be continuously adapted to current technological developments

Teaching Methods

Seminar based teaching combined with practical blocks, as well as some group work or research with presentation of results

Recommended Literature

[1] Winner, H.; Hakuli, S.: "Handbuch Fahrerassistenzsysteme"
Springer Vieweg Verlag 2012, 2015, 3. Auflage, ISBN: 978-3-658-05733-6
[2] Reif, K.: "Automobil Elektronik", Vieweg Verlag 2006, 1. Auflage, ISBN 3-528-03985-X
[3] Streichert, T.; Traub, M.: "Elektrik/Elektronik Architekturen im Kraftfahrzeug",
Springer Vieweg Verlag 2012, ISBN: 978-3-642-25478-9
[4] Schäufele, J.; Zurawka, T.: "Automotive Software Engineering",
Vieweg Verlag 2003, ISBN: 3-528-01040-1

Type of Examination

Portfolio