The Personal Programming Project (PPP)

Rules for the Winter Semester 2023/24

B. Eidel[1], A. Prakash, S. Prüger

version: Thursday 26.October 2023

[1]Responsible for the Module

Contents

Chapter 1

Introduction

1.1 Scope

The module Personal Programming Project (PPP) is described in the module handbook on https://tu-freiberg.de/media/2363/download. The workload is 210h, the final report has to be delivered within 22 weeks, 7 credit points are awarded for a successful completion of the project.

The grade for the PPP follows from an evaluation of (i) the source code, of (ii) the documentation and of (iii) the oral defense.

As a companion we provide here detailed guidelines for all stages of the PPP. This is intended to support CMS students in being successful in their project.

The main aim of the PPP is that students obtain expertise in programming by working on a topic of computational materials science (CMS) and closely related fields. The focus of the report should hence be on the programming aspect of the project. The project topic needs not to live at or even push the cutting edges of research. The focus on programming work implies that mere simulations applying an existing software can not serve as a proper PPP.

Chapter 2

From Planning to Enrolment

2.1 Preparations

A careful and sufficiently early planning of your project is important and strongly recommended. It is very advantageous for a successful PPP.

It is assumed that these preparations cover an initial literature search providing not only a good idea on the problem you wish to work on, but also a sufficient understanding of the way to attack and solve the programming task.

This implies the reading and understanding of documentations for finite element frameworks such as Abaqus (for programming a User Material UMAT or a User Element UEL), FEniCs, and alike, if your PPP shall be placed into these frameworks.

We strongly recommend looking for a supervisor. This should happen at an early stage of planning. Going alone you might miss something important.

2.2 Enrolment

1. **Enrolment, action and status.**

   Enrolment is mandatory for the PPP. Submission without enrolment is not possible.

   Enrolment is to be done on OPAL via the following link:

   `https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/17694195712`

   De-registration (from a group, see below) is not allowed.

   Once enrolled, you are automatically expected to deliver the final report of your PPP. Failure to do so will result in a "Fail" grade (5.0). This underpins the relevance of sufficient preparations.

2. **Group.**

   Ensure that you have enrolled yourself to the right group. The groups are always named with the corresponding semester as the reference. For example, students intending to execute their PPP in WS 2023/24 will have to choose the group "Course group PPP - Personal Programming Project WS2023/24". Those intending to do it in the SS 2024 will have to choose the group "Course group PPP - Personal Programming Project SS2024". The groups have no relation to when you started your study program!

If the group for the semester in which you intend to perform your PPP does not exist , then wait until the corresponding group is created or contact the supervisors. Usually, the group is created a couple of weeks before the proposal defence.

3. **Retake.**

   If a retake is necessary, you have to enrol anew to the corresponding group of that semester.

   The topic of the previous PPP-try cannot be resumed. For equal chances among the CMS students, the topic must considerably deviate from the previous one.

Chapter 3

Proposal, its Submission and Defense

You will need to defend your proposal in front of a panel of judges (usually the lecturers responsible for PPP). This defense will take place at the beginning of a semester's lecture period (typically winter semester). The exact date will be let known a few weeks before the start of lecture period of the winter semester.

To facilitate easy organization, the OPAL course provides two tasks are usually created in the student area for the corresponding semester.

(i) Schedule for proposal defense: This provides students with a time slot to choose for the proposal defense. A student may choose only one time slot. Note, however, that the chosen time slot might be changed so as not to have gaps between meetings.

(ii) Submission of proposal slides: This task is to be used to submit the slides for your proposal defense. All students will have to submit their slides by the given deadline, even if the proposal defense is scheduled for multiple days.

3.1 Proposal Slides

For the defense meeting, please prepare slides (ppt, odp, pdf) detailing the planned PPP. You are allowed a maximum of **5 slides** with a strict given structure. Please ensure that this is maintained. Slight deviations from the structure are allowed if it is necessary to ensure a good leitmotif. But please try and adhere to this structure as much as possible.

The structure is as follows:

- **Slide 1**: Motivation
- **Slide 2**: Aim/goals of the current work. Must contain the following details:
  - Theoretical background, key equations
  - Details on what exactly will be programmed
  - Approach to be used
  - How will this be programmed?
  - Scope - why does this work warrant a PPP (i.e., does it have substantial amount of programming, does it encompass 22 weeks of work etc.)

- **Slide 3**: Methods. Must contain the following details
  - Programming language(s) to be used
  - Third party open source and commercial software to be used
  - Third party libraries to be used
  - List/Details on sub tasks to be performed as part of the final goal
- **Slide 4**: Details on Testing/Verification, see Chap. 4
  - List of all tests that will be performed to verify the sanity of the code, with a clear identification of which of these are standardized tests and which are customized (or self thought). This is an important part of your proposal and requires care and diligence. It is true that you may accidentally overlook a test at this early stage of your PPP.
- **Slide 5**: Schedule
  - Clearly identified milestones to gauge the progress of work
  - Calendar (in terms of days/weeks) showing sub-tasks and the estimated time for each sub-task

3.2 Submissions

The following documents must be uploaded to OPAL by the given deadline. Later submissions can not be accepted. Submissions via email will also not be accepted.

1. **Slides**

   For the slides, the following name convention applies:

   ```
   <Matrikel Number>_<Name>_WS2023.pptx
   ```

2. **Supervisor consent form**

   Any supervisor must explicitly sign a form of consent for his supervision agreement. The form is attached at the end of this document.

3.3 Defense

The maximum time for presenting your proposal is **7 minutes**. It is very important, not to exceed the time slot.

Chapter 4

Coding and Code Testing

The source code has to be clearly structured and properly commented, and it must compile/run without any additional modifications. The packages, libraries, etc. necessary to compile/run and the corresponding versions should be given in the documentation, because the student should not worry about potential incompatibilities with already installed software packages on prospective users
Instructions on coding including the inclusion of coding aids (...) are detailed in the following. This will also encompass the documentation of the code in terms of comment lines and the manual.

How to avoid plagiarism. Your code must identify parts which were not programmed by you. If this is not properly done, it will be deemed as plagiarism. Plagiarizing will result in failing the project. In addition, the functions, subroutines or modules used from external sources, should be given in a list in your documentation.

Code parts taken from other sources, such as modules from libraries etc. are admitted for usage in the project, but they all must clearly, precisely and completely be declared along with their original source.

Building a novel software framework by heavily drawing on existing libraries can be accepted as a valid PPP, if and only if there is enough originality, which must be clearly demonstrated. Hence, a PPP need not necessarily consist of 100% code developed from scratch, but it can be.

4.1 The usage of chatbots in the PPP

ChatGPT and other chatbots have the potential to revolutionize the development of writing computer code in several ways listed below. This has consequences for the PPP module within CMS.

The usage of chatbots in programming your PPP is not only allowed. In contrast, students are encouraged to continue and expand their learning experience with ChatGPT obtained in their individual projects in "Research Seminar & Journal Club". (This refers to CMS student batches who learned to apply ChatGPT in the new module format introduced by Prof. Eidel, for the first time in the summer semester 2023.)

The condition for including Chatbots and related aids within the PPP is to accurately describe that usage. The introduction of a novel, extra part in the report reflects the importance of suchlike aids and their deliberate, critical usage.

Aspects of using chatbots in software development:

1. **Assistance in Coding**: Chatbots can serve as coding assistants by helping developers with syntax errors, suggesting code improvements, and providing explanations for programming concepts. This can be particularly helpful for novice programmers who are learning to code.

2. **Debugging Support**: Chatbots can assist in debugging code by analyzing error messages and offering suggestions for resolving issues. They can help identify the root causes of bugs and propose fixes.

3. **Code Generation**: Advanced chatbots can generate code snippets based on natural language descriptions or high-level requirements provided by developers. This can significantly speed up the coding process, especially for repetitive or boilerplate code.

4. **Code Review**: Chatbots can review code for adherence to best practices, coding standards, and security guidelines. They can provide feedback on code quality and suggest improvements.

5. **Knowledge Transfer**: Chatbots can facilitate knowledge transfer by answering questions and providing information about programming languages, libraries, and frameworks.

6. **Learning and Training**: Chatbots can be used as educational tools to teach coding concepts and programming languages. They can provide interactive lessons, quizzes, and exercises to help individuals improve their coding skills.

7. **Multilingual Support**: Chatbots can provide coding assistance in multiple languages, making coding resources more accessible to a global audience.

In this context it is important to note that chatbots are most effective as supplementary tools to human developers.
They can (i) enhance productivity, (ii) assist in learning, and (iii) improve code quality, but they are not a replacement for human expertise and creativity in software development. Developers should use chatbots as aids and leverage their capabilities to streamline coding tasks and improve their coding skills.

Once this kind of code-generation is involved, the testing of the resultant codes gains even more relevance, see Chap. 4.

4.2 Code Testing for Verification

This chapter serves to clearly define the requirements for tests that students must carry out when testing their computer codes within their Personal Programming Project (PPP) as an integral part of the CMS Master Course Computational Materials Science at the TUBA Freiberg.
The PPP puts strong emphasis on testing. The general rule for the PPP is "UNTESTED CODE IS NO CODE!" This rule calls for a corresponding diligence in writing up the report, see below.

4.2.1 Introduction: What is Verification? What is Validation?

Some terms shall be properly defined like verification and validation. This is even more important, since in different communities there are different definitions or notions of these terms.
In the context of computational physics, "verification" and "validation" are two distinct processes used to assess the accuracy and reliability of computational simulations or models. These terms are often used in fields where computer simulations are employed to predict the behavior of physical systems, such as structural engineering, fluid dynamics, and materials science.

1. **Verification**:
    - **Meaning**: Verification is the process of confirming that the numerical implementation of a computational model accurately represents the underlying mathematical equations. In other words, it ensures that the software code used for simulation correctly solves the governing equations.
    - **Focus**: Verification primarily focuses on checking the correctness of the numerical methods, algorithms, and software used in the simulation. It aims to identify errors or bugs in the code that might lead to incorrect results.
    - **Methods**: Verification is often achieved through a series of mathematical tests and benchmark problems. These tests include analytical solutions, method of manufactured solutions, and code-to-code comparisons to verify that the software behaves as expected.
    - **Goal**: The ultimate goal of verification is to ensure that the simulation code faithfully reproduces the mathematical model, so that any discrepancies between model and simulation results can be attributed to the model itself, not the numerical implementation.
2. **Validation**:
    - **Meaning**: Validation is the process of assessing whether a computational model accurately represents the real-world physical system it is intended to simulate. It evaluates the model's predictive capabilities by comparing its results to experimental or observed data.
    - **Focus**: Validation focuses on the model's ability to capture the physical behavior of the system being studied. It seeks to answer the question: "Is the model a good representation of reality?"
    - **Methods**: Validation often involves conducting experiments or collecting real-world data to compare with simulation results. The agreement between simulation and experimental data is used to assess the model's accuracy and reliability.
    - **Goal**: The goal of validation is to gain confidence in the model's predictive capabilities. It verifies that the model, as implemented in the simulation, provides meaningful and accurate predictions about the real-world system.

In summary, verification is concerned with confirming the correctness of the simulation code and its ability to solve mathematical equations accurately, while validation assesses the model's ability to represent and predict the behavior of a physical system. Both processes are essential in ensuring the credibility and usefulness of computational simulations in fields like computational mechanics, where accurate predictions of physical phenomena are critical for engineering and scientific applications.

4.2.2 List of Tests: mandatory and optional ones

The range of tests to be carried out depends on the type of software. Testing code may serve very different purposes. A test may serve the purpose to (a) verify the functionality of a code part, (b) to verify the performance of a numerical algorithm (e.g. does a method truly converge in the order predicted by theory? Does it converge at all?), (c) to validate the modeling, i.e. the set of equations describing the physical model, which are cast into a numerical solution framework.
Validation can be carried out against experimental results or in comparison with a more accurate physical model as e.g. validating an empirical potential against a DFT- or ab-initio simulations.
Although validation is most important for obvious reasons, it is not in the focus of the PPP, since PPP focuses on programming; code development, testing, optimization etc.
However, physical/mechanical nonsense must be identified as such – this is CMS! Don't expect mercy, if you oversee gross physical nonsense. Such observation may identify the deficit of a model (truly validation), or, even more likely, it may hint at a mistake in the solution method or implementation.

In the following we provide a list of tests. In their description we fulfill what we expect from students in that we completely clear all relevant questions on What? Why? How? including Which result is expected?
Note the distinction between necessary/mandatory tests, which the students **must** carry out. And other tests, which can be optionally done. Equally note the distinction in the grade of reliability of each test. For instance, the fact that a test based on a comparison with other software systems, or benchmark tests, etc. are of limited reliability – but frequently truly worth to go.

The list is –by its very nature– incomplete. We restrict to rather basic tests which –as a rule of thumb– show up in at least one PPP every season.

Note that e.g. finite element solvers (like Abaqus, Ansys, FEniCS etc.) or other software systems with their particular environments and settings impose extra rules which have to be considered. It goes beyond the scope of this guide to describe these rules in all details. Since this kind of PPP must have a supervisor, this supervisor will then provide domain-/software-specific material or give extra guidance.

1. **Testing of a numerical solution method. – MANDATORY**

   (a) **Newton-Raphson Method – MANDATORY**

   In a very large variety of applications the Newton Method (or Newton-Raphson Method) is used to solve nonlinear algebraic equations (based on linearization and the sequential solution of the linearized problems.)
   • **Why?** Since a particular appeal of Newton's method is its quadratic convergence close to the solution point, it must be tested.
   • **What?** Test, whether quadratic convergence is achieved, or not.
   • **How precisely?** The norm of the residual force vector is to be displayed in tabular form as a function of the iteration. The tolerance as the stopping criterion must be equally given.

   (b) **Time integration methods – MANDATORY**

Time integration algorithms are used in different fields of computational mechanics and computational material science, for the integration of evolution equations for inelastic constitutive laws (viscoelasticity, elasto(visco)plasticity, damage, but also for phase field models, and for dynamics, where Newton's equation of motion are integrated as in structural dynamics or molecular dynamics ...

– **Stability – Mandatory. Explicit** time integration schemes typically are only conditionally stable, which means that there is a critical time step size maxΔt, which are derived by theoretical considerations and are well-known, must not be exceed to preserve the stability of the method.

• **Why and What?** Since the instability may spoil your simulations, the critical time step size and the stability of the solution must be investigated. (Recall the analysis of the critical time step size of FTCS finite difference scheme for Fick's 2nd law in the FUMS-lecture by Prof. Eidel.)

• **How?** Choose the critical time step size (and beyond), which are derived from theoretical considerations and check the solution against the solution for a step size below the critical value.

– **Convergence order – Mandatory.** For time integration of inelastic evolution equations, implicit Backward Euler is almost always a good choice; of course, it is only linear, but it exhibits very nice stability properties – in contrast to explicit methods.

• **Why and What?** If you use a **higher-order (**$p > 1$**)** time integration scheme, the higher numerical costs must be justified by the improved higher order.

• **How?** The convergence order of the method can be obtained by displaying the error for a given discretization over the characteristic step size in double logarithmic scaling. The error is computed by means of an overkill solution which means by a grid or time step size $h \to 0$.

2. **Finite Elements – MANDATORY**

- Partition of Unity, Numerical Quadrature, etc.
  - Check for the Partition of Unity property of your ansatz function, i.e. $\sum_{i}^{N} \boldsymbol{N}_i \left(\boldsymbol{\xi}_p\right) = 1$ for any point $\boldsymbol{\xi}_p$ inside the unit domain of the element
  - As a consequence of the Partition of Unity property, the sum of the derivatives of the shape functions with respect to any of the local variables ξ, η, ζ vanishes for any point $\boldsymbol{\xi}_p$ inside the unit domain of the element, i.e. $\sum_{i}^{N} \frac{\partial \boldsymbol{N}_i}{\partial a} \left(\boldsymbol{\xi}\right) = 0$ with $a \in [\xi, \eta, \zeta]$.
  - Compute the Jacobian matrix of a single element and compare to the analytic value, e.g. in the case of an affine deformation being applied to the regular element geometry, in which case the Jacobian matrix is constant inside the finite element.
  - Use the numerical integration procedure to compute the length (1D), the area (2D) or the volume (3D) of the finite element from the determinant of the Jacobian and nodal coordinates. In the case, where an affine deformation is applied to the regular finite element geometry, the numerical result can be verified by analytical computations.

  - Due to the Partition of Unity property the sum of the derivatives with respect to the physical coordinates x, y, z should also vanish, at least for the regularly shaped elements.
  - Starting with a known field distribution (temperature, concentration, displacement,. . .) that can be exactly represented by the chosen finite element (e.g. a field varying linearly in each space direction), its gradient can be computed at each point in space. The exact gradient of the known fields has to be compared with the one computed from the B-matrix and the values of the nodal degree of freedom, chosen in accordance with the known field distribution. Such known field distributions may be either associated with rigid body motions, constant strain states or particular higher order gradients of the field variable (e.g. linearly varying strain along a certain direction), see [3, Sect. 4.2] for further information.
  - The computation of the external force / external heat flux vector requires the numerical integration over element edges or faces. Check that this numerical integration yields the length of element edges or the area of the element face, if the integrand is equal to one. Note that the order of the integration scheme may be different than the one applied to carry out the integration in the element domain. Further details on the computation of the Jacobian at element edges or surfaces can be found in [1, Sect. 5.3]

- Check the number of **zero eigenvalues of the element stiffness matrix**.
  - Note that the number of zero eigenvalues of a finite element stiffness matrix must equal the number of **regular rigid body motions** of that type of element, and the corresponding eigenvectors represent these rigid body motions (in 3d:6 for a volume element, in 2d: 3 for a quadrilateral or triangle element). If the actual number of zero eigenvalues equals the number of admissible rigid body motions, the test is fulfilled which implies that the element formulation can properly represent rigid body kinematics.
  - Note that "underintegration" of a finite element (the number of Gauss-points per direction of space is below the required number for a particular integration method as e.g. Gauss-Legendre) leads to a rank-deficiency and introduces local deformation modes without energy consumption, which are referred to as 'spurious', 'zero-energy'- or –for the resemblance to an hourglass-shape– 'hourglass'-modes (e.g. for the deformation of a plate or a shell) . This is indicated by an additional, spurious zero-eigenvalue and can be even visually confirmed as described (see [3, Sect. 8.1]).
  - Note that the number of zero eigenvalues also depends on the physics of the boundary value problem to be solved with the finite element method, e.g. in contrast to the situation in mechanics, described above, only one zero eigenvalue is observed in heat conduction or diffusion problems.

- Patch test: For iso-parametric element formulation.
  - **What and Why:** The patch test is an indicator of the quality of the programmed iso-parametric finite element and is used to assess the consistency and stability of the finite element formulation used. Provided the element has passed the patch test, it can be shown that any mesh using such finite elements will result in converged solution.
  - **How:** Construct a patch (mesh) of finite elements such that at least one node

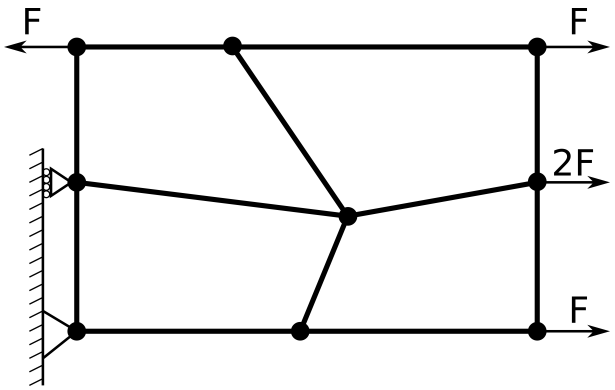

Figure 4.1: Mesh and boundary conditions for a possible patch test. Shown here is a patch test for σ_x in plane four node elements with linear shape functions. Adapted from Ref. [2]

.

is internal to the patch (see fig. 4.1). The element shapes must be irregular because some element types pass the test if they are rectangular, but fail otherwise. The patch is to be provided with just enough support to avoid rigid body motion. To one or more nodes on the boundary, we impose nodal forces or displacements that are consistent with a state of constant stress or strain. The internal node(s) is (are) left unrestrained.

  - **Expected result:** Since a state of constant stress or strain has been imposed on the patch, the values computed in the Gauss points must display the same. For more details, please consult Refs. [2, 7], [3, Sect. 8.2].

- Check of **constitutive response under full deformation control**
  - The term full deformation control refers to a homogeneous deformation state, in which all components of the total strain or the deformation gradient (in a finite deformation framework) are prescribed, which can be obtained by a single finite element with all displacement degrees of freedom being prescribed consistently. In general, Dirichlet boundary conditions are specified at all degrees of freedom of the nodes located at the boundary.
  - If the determination of the stress tensor and the internal state variables requires the solution of a single or a system of nonlinear equations, ensure that the applied numerical method shows the desired convergence behavior, e.g. quadratic convergence in case of the Newton-Raphson method.
  - The consistent algorithmic tangent obtained from the linearization of the incremental form of the material model should be compared to a finite difference approximation of the tangent. This approximation can be obtained by repeated computations of the stress tensor for linearly independent, perturbed strain states, see [6] for the small strain and [5, 4] for the finite deformation framework.
- Check of **constitutive response under stress control**
  - The term stress control refers also to a homogeneous deformation state, but the loading is prescribed in terms of a known traction vectors and sufficient Dirichlet boundary conditions to prevent rigid body motions. Once again, a single finite element is sufficient for this test. In general, the loading is prescribed in terms of Neumann boundary conditions and the minimal number of Dirichlet boundary conditions to prevent zero eigenvalues in the tangent matrix.

  - Check for the convergence behavior of the numerical solution method applied to solve the system of nonlinear equations for the nodal degrees of freedom.

- **Incorporation and testing of** different **nonlinear effects** in the material behavior or **coupling** of different fields has to be done one after another.
  - If the material model includes simple models as special cases, e.g. a pressure-dependent, non-associative plasticity model may be reduced to a standard J_2-plasticity model, these special cases have to be tested by appropriate selection of the material parameters.
  - The step-wise application of different nonlinear effects also applies to problems in finite deformation. So whenever the implementation of a nonlinear constitutive behavior in a geometrically nonlinear formulation is sought, testing a physically linear material behavior (St. Venant Kirchhoff elasticity) with geometrically nonlinear effects activated, before considering tests being both physically and geometrically nonlinear is recommanded. The properties of the element stiffness matrix and the element's capability to represent rigid body motions as well as constant strain states have to be checked also in the case of finite deformations, see [3, Sect. 13.9].

3. **Testing against analytical solutions – OPTIONAL**

   Analytical solutions are a very valuable test for testing the numerical solution, definitely worth to go and a plus! However, care must be taken, whether all relevant conditions coincide between the chosen model along with the chosen numerical solution method and the corresponding conditions for the analytical solution.

   Examples: Deflection of a cantilever beam as simulated by finite elements against the analytical solution. Although this task seems to be trivial, it is not. Make sure, that you choose an appropriate analytical model for beam bending which is fully consistent with the simulation you carry out. (Criteria: Bernoulli-beam theory (restricted to bending deformation) versus Timoshenko-theory (adds shear deformations) with its link to the geometry of the beam, most notably the length-to-height ratio ($L : H$ – this ratio should be large). Are deflections small enough such that geometrical linear theory is applicable? Note that for larger deflections (rule of thumb, larger than the height H) geometrical nonlinearity must be considered.)

4. **Testing against existing (commercial) simulation codes – OPTIONAL**

   Suchlike tests by comparison with established (commercial) simulation codes can give an orientation, but must be taken with extreme care since the outcome may be misleading in either case.

   If it cannot be assured from the documentation of the commercial software that the mathematical model implemented in the commercial software is identical to the one to be implemented in the PPP, an observed deviation falsely indicates a potential error. For illustration purpose, we refer to the following example. The linear quadrilaterial elements in Abaqus have been used to verify the corresponding UEL implementation in a previous PPP. However, the results of the two implementation agreed in the case of homogeneous deformations, but in the inhomogeneous case, there was a deviation, which lead to a desperate search for a bug, where there was none. The reason was that the student in the PPP implemented a linear finite element from the standard text book,

whereas the Abaqus "linear" element had features implemented to avoid for instance locking[1].

5. **Test by comparison with results in publications – OPTIONAL**

   In a large variety of computational mechanics disciplines different models and methods are assessed and compared in test sets in terms of very particular (Initial) Boundary Value Problems. Some of them are referred to as "benchmark problems". In many cases, the simulation results are compared by e.g. Force-displacement curves since they are an overall, global response for the behavior of the model/method and thereby easy to compare.

   If suchlike test sets are available in literature, it is valuable to carry out these tests with your own software. Again, care and diligence must be employed to check the comparability based on fulfilled conditions jointly shared by the own model with the reference.

   Note in this context, that results in literature can be inaccurate or even wrong. You can not consider suchlike tests as reliable in the strict sense of verification!

   In either case a good training for a student striving for becoming a Master of his or her discipline. Discuss the results with your supervisor!

6. **Miscellaneous**

   - **Microstructure generation** For a programming task like Voronoi microstructure generation, check if your code can generate regular mono-dispersive microstructures. In other words, can your program generate a microstructure with only cubes/squares or icosahedra/hexagons given the corresponding regular grid of points?

     For physics-based microstructure models like grain growth models, check if the predicted results follows the expected physics. For instance, if the model is based on interface curvature, check if grain shrinks with positive curvature and grows with a negative curvature. In other words, a spherical grain should shrink whilst a hyperboloid grain should grow. Recall that a curved boundary is expected to migrate towards its center of curvature. Furthermore, the rate of migration should be directly proportional to the local curvature. For a model based on energies, the domain of lower energy should ideally grow into the domain of higher energy for energy minimization.

More Comments

Furthermore, any testing must be done on the same code that will be used for actual production runs. In other words, there cannot exist two pieces of code, one for production runs and one for testing. Even though these aspects have been mentioned many times to the students, many students still test their code by copying sections of the code to a different file and testing it independent of the main program.

[1] locking refers to artificial stiffening effects, which are either introduced (i) by the numerical solution method or (ii) by a particular material behavior along with the numerical solution method. For the first case (i) we mention as an example the so-called 'shear'-locking for elements having a displacement ansatz of low order, most pronounced for $p = 1$ but also visible for higher order ansatz functions. The second type (ii) is volume-locking or incompressibility locking which may occur for materials having a Poisson's contraction coming close to 0.5 or for elasto-plasticity, where the plastic part of deformation is incompressible.

Chapter 5

Report

The grade for the PPP follows from an evaluation of (i) the source code, of (ii) the documentation and of (iii) the oral defense. The source code has to be clearly structured and properly commented, and it must compile/run without any additional modifications. The packages, libraries, etc. necessary to compile/run and the corresponding versions should be given in the documentation, because the student should not worry about potential incompatibilities with already installed software packages on prospective users.
The report to be written with LaTeX must fulfill the rules of scientific writing you have learned in "Research Seminar and Journal Club" and must contain the following sections:

1. Front/Title page with Title, Author, Matr.-number
2. Abstract (0.5 page): Concise account of the present work
3. Introduction (Not more than 1 page): Explains the Motivation for and the Objective of this work.
4. Theory (not more than 4 pages): Theory with key equations is presented with a sound link to the physical background of the considered problem. Reference to the most relevant literature (usually $<$10 references).
5. Numerical Methods and Implementation Details ($>$2 pages): Explain all relevant numerical methods that you use (in case of doubt talk to your supervisor how much detail is required). Explain the choice of the programming language. Are there tricky aspects in the implementation? - discuss! If you're using external libraries this should be stated here as well (which version, where to get, etc.)
6. External libraries, software (packages), Interfaces: Here any piece of software adopted/adapted from other sources like libraries and packages must be listed with a short description of their functionality.
7. Program flowchart: This is a plus and helps to visually explain the code structure and functionalities.
8. Usage of Chatbots and Other Tools: It must be declared, if and how Chatbots like ChatGPT have been used for fabricating the PPP in whatever aspect of application; code development, testing, writing parts of the report including pimping the text, etc.
9. Testing for Verification ($>$2 pages), **cf. Chap 4**: This section shall contain a concise description of tests carried out along with a discussion of the result.

Structure each test case as follows:

Test Case XX:

- Aim: Explain the purpose of the test case. What or which feature does this test case aim to verify?
- Setup: Provide all information for the setup, e.g. the boundary value problem with all parameters (geometry, material parameters, etc.) Sketches are instrumental.
- How to run the test case: Provide the command to run your program in order to execute this test case. This must include all options, parameters, files, etc. required for execution.
- Expected result: What is the expected/correct result of this test? Provide enough details for the reader to justify why this is indeed the expected result.
- Obtained result: What was the result obtained after execution of your code? Does it compare well with the expected result? In case of deviation, discuss the reasons for such a deviation. (Even details matter, e.g. if there is a little "wiggle" in a curve that you expected to be smooth – where does it come from?)

The results are typically documented in terms of diagrams and/or tables, to which the detailed descriptions and explanations refer.

Each test case can be placed in a separate folder containing a README file with the above details. The folder should also contain all the input/parameter/data files required to reproduce the test case.

As mentioned above in Chap. 4, "Validation" in the sense of computational physics is here not a mandatory part of the testing actions. Efforts to even validate the modeling goes beyond the task of PPP. Sound comments on validation are a plus.

10. Computation Time Study. (1-2 pages) This is optional and a plus, additional bonus points may be given. For this you will need to include the computation times of each function of your program and identify where the bottleneck is. The timing functions must be a part of your program itself (times determined by, say, using a stop watch will not be entertained.)

11. Conclusion. (Not more than page) It contains a concise summary of the PPP. From content it goes beyond the abstract in that obtained results are included a bit more precisely. Likewise, conclusions can and shall be drawn with respect to possible improvements, further tests, future extensions etc. but going beyond the scope of a PPP.

12. Manual (>2 pages) Provide execution instructions: Explain how to use and run your code. What are the files needed for the execution, what format must the files (both data and parameter files) exist in, keywords to be used etc. Use a few simple examples for this.

13. Literature (0.5 page)

14. Git log: Provide a log of the git commits that were made. To get such a log, just execute "`git log`" on your git repository.

Further important details on the project report+code

1. The goals, tasks and milestones announced in your proposal defense must be called upon in the report. Clearly identify which of them were actually achieved and which were not. For the latter, some reasoning of the failure and justification for further course of action must be provided. The goal of this report part is to show compliance to what was promised in the proposal.

2. The report must contain documentation of the implemented features. This is in addition to the documentation found in the code itself. (see best practices!)

Chapter 6

The Final

6.1 Required Final Uploads

- **What has to be uploaded?**

  To be submitted, i.e. uploaded to OPAL

  1. Code: According to the following name convention for your files:

     `PPPws2023_<Matr.Nr.>_<Name>.<zip/tgz/tar.gz>`

  2. Report: pdf-file.
  3. Presentation slides: To be uploaded onto OPAL by the given deadline. You will only be allowed to use the uploaded slides during the examination.

- **Which is the deadline?**

  It will be communicated via OPAL. As a rule (see module description) 22 weeks after the PPP start. In exceptional cases (such as a cyber attack with negative consequences for working on a PPP), there will be exceptional measurements such as adequate extensions, which will be equally communicated via OPAL.

6.2 Final presentation / PPPdefense

Recall, that the grade for the PPP is a result of the quality of (i) the source code, (ii) the documentation/report and (iii) the oral defense.
The oral defense (Colloquium) will consist of a presentation and subsequently a viva-voce.
Total planned duration: **30** minutes
Presentation: **15** minutes
Oral examination: $\geq$**15** minutes. Here we will go through your code, documentation, testing etc. and discuss them with you. We recommend that you to bring your laptop for this purpose, in order to show the working capabilities of your program.

It is expected that the number of slides is adequate to the limited time of presentation.

How you structure the presentation is your own choice. You have received some advice on the same in the lecture "Research Seminar and Journal Club". Make sure that you have a coherent story to tell.

The overarching theme of the presentation is that the essential content of the report must be

captured.
The presentation must in either case contain at least the following points:

1. Introduction/motivation: briefly explain why this project if of interest
2. Tasks and aims of the project: make sure that you are referring to the tasks and aims that were presented in the proposal
3. Details on the model (including theory) that is implemented
4. Details on the implemented code, the challenges, modularity etc.
5. Details on the tests performed: Ensure that the results of all tests discussed in the proposal are presented here
6. Summary and Conclusions

Please ensure that you refer to your own proposal in all relevant cases, and show that the task has been accomplished (be it testing or coding). If otherwise, please also explain why this was not achievable.

Important Notes.

- Your program should be able to run without any modifications to the code for any case that we ask you to run. To give an example, in case of PPPs related to the finite element method, the code should allow for an adjustment of the time step, tolerances, boundary conditions, physical properties, material parameters, etc.
- Do not exceed the time limit for your presentation.
- Recommendation: Carry out rehearsals of your presentation, including the interview of yours. It can be helpful, to team-up with other students of your batch, or seniors ...
- Recommendation: Show your slides to your supervisor –early enough– for some hints. But do not forget, it is your show! (i.e. your effort!)

Chapter 7

Topics

7.1 Some generic topics

An incomplete list of possible generic topics for PPP are listed here. This list should give you (Students) an idea on the complexity of the problem expected in the framework of the PPP.

1. Develop a self-written finite element, finite difference or finite volume code for a particular physical problem. The developed code must be applicable to at least two-dimensional problems. Examples include:
   (a) Static mechanical problems with irreversible material behavior (e.g. non-linear viscoelastic, J2-plasticity with combined isotropic-kinematic hardening, damage mechanics for monotonic or cyclic damage, etc.)
   (b) Static elastic problems with adaptive mesh refinement
   (c) Wave propagation with implicit time integration
   (d) Convective and/or diffusive transport problems (heat, concentration, etc.)
2. Developing a 2D microstructure evolution code (e.g. solidification) using phase field
3. Extension of existing free or commercial finite element codes
   (a) Implementation of element routines for a particular boundary value problem, like e.g., large displacements hyperelastic theory, plate theory, cohesive elements etc.
   (b) Implementation of material routines for irreversible materials laws beyond that already discussed in different lectures
   (c) Implementation of multi-physics problems, like coupled electrical/mechanical or chemo-mechanical problems
4. Extension of in-house developed code to include additional features
5. Implementation of smoothed particle hydrodynamics
6. Implementing 2D/3D Molecular dynamics or statics code with possible extension towards
   (a) Comparison of different optimization algorithms for static relaxation
   (b) benchmarking of time integration schemes
   (c) Computation of energy and stress/stain field of a penny-shaped crack, edge dislocation

7. Implementation of Monte Carlo methods (Metropolis, Potts, Ising etc.) and application to specific material science problem
8. Implementation of cellular automaton and application to relevant problems like recrystallization
9. Data analysis in field of atomistic simulations, image correlation etc.
10. Visualization and analysis of complex 3D datasets

7.2 Machine Learning Topics

Many students have expressed interest in doing machine learning related projects. You are very much welcome to pursue ideas on this front.
Several particular conditions must be fulfilled.

1. **Supervisor.** What is strongly recommended for any PPP-topic is mandatory for a Machine Learning Topic; to have a supervisor.
2. **Topic.** The project must deal with a topic of CMS and closely related fields of computational mechanics or computational physics.

   This includes the database used in the project. The usability in the CMS field must be clearly demonstrated. (Datasets such as e.g. handwritten digits, or pedestrian data or alike for image classification is not allowed. Merely saying that the algorithms can be used for problems in materials science too will not suffice.) You may either choose data available from online repositories or generate your own datasets.
3. **Programming.** Enough effort must be clearly demonstrated on the programming front; just building a neural network by applying libraries such as TensorFlow, KERAS etc. and training it on already available data is not sufficient for the PPP. If you decide to generate your own data, please make sure that you explicitly show the programming required (remember: simulation is not programming, however, postprocessing and management of data can be deemed as programming, if the needed programming is shown. All that needs a clarification based on the PPP-responsible staff and the supervisor).

   With this we are asking you to think about data acquisition and management, both of which are as important as the machine learning system itself. You are now forced to think about the data itself, particularly on how you can generate such data, what are the features that you need to have, what data formats you use etc. This is important, since materials scientists are rarely involved in algorithmic development, but rather utilize algorithms for solving materials science problems.

Chapter 8

FAQs

This page lists some questions that have been asked by students and should answer some of the doubts that you might have. Note that this is a page that is under constant development, with new questions (and answers) being added every time one is posed.
If you find that your question is not listed here, then post it in the forum!

- **Supervisor**
  Q: Is it mandatory to have a supervisor for my PPP?
  A: NO. There is no mandatory requirement for a supervisor. However, we strongly recommend having one. This helps in not only ensuring that you are on the right track with the project, but also that it conforms to the requirements of the PPP, particularly with regards to testing and documentation. Going alone, you simply may miss something important.

- **Becoming sick**
  Q: I am sick and the doctor has advised me rest for X days. Can I get an extension of the deadline for X days?
  A: YES. Submission deadlines are extended by the number of days in the medical certificate provided by the doctor. Note that the medical certificate must in principle be produced before the deadline.

Chapter 9

Appendix

In the following we provide some templates.

1. Title page of report (as an example. Requires adaption to the individual PPP.)
2. Supervisor consent form.
3. Registration form

Technische Universität Bergakademie Freiberg

Computational Materials Science
Fakultät IV

Personal Programming Project

Phase field modeling of diffusion-stress-fracture in Li-ion battery

Felix Daniel Rajendran

Matriculation Number: 66972
19.05.2023

Supervised by
Prof. Dr.-Ing. habil. Bernhard Eidel

Professor Incharge
Dr.-Ing. Arun Prakash
Dr.-Ing. Stefan Prüger

TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
The University of Resources. Since 1765.

Supervision of Personal Programming Project

Student: ……

Course: Computational Materials Science

Matriculation number: ……

Topic: ……

……

……

☐ Supervised Personal Programming Project

Supervisor Name: ……

Supervising Chair: ……

Date:

Signature of supervisor:

☐ Unsupervised Personal Programming Project

(I hereby declare that I intentionally did not choose a supervisor. This option is not available for Personal Programming Projects related to machine learning.)

Signature of student:

Bergakademie Freiberg

.....

(Stempel des Institutes)

Freiberg, ...

REGISTRATION of the Personal Programming Project (PPP)

Students Name: .....Matrikel-Nr.....

Master CMS

Topic

Date of Registration:	
Submission deadline::	
Final deadline (defence)	

Important regulations

-Maximum duration: 6 month
-Report submission after 22 weeks
-Presentation must be done within the 6 month period after registration

Supervisor:

Date	Examiners Title and Name	Signature
----------	----------	----------

Bibliography

[1] K.-J. Bathe. Finite Elemente Procedures. Prentice-Hall, Inc., Watertown, MA, 2nd edition, 2019.

[2] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt. Concepts and applications of finite element analysis. Fourth Edition. John Wiley & Sons, 2003.

[3] K. Knothe and H. Wessels. Finite Elemente. Springer, Berlin, Heidelberg, 2017.

[4] J. Kochmann, T. Brepols, S. Wulfinghoff, B. Svendsen, and S. Reese. On the computation of the exact overall consistent tangent moduli for non-linear finite strain homogenization problems using six finite pertubations. In 6th Conference on Computational Mechanics ECCM, Glasgow, 2018.

[5] F. Meier, C. Schwarz, and E. Werner. Determination of the tangent stiffness tensor in materials modeling in case of large deformations by calculation of a directed strain perturbation. Computer Methods in Applied Mechanics and Engineering, 300:628–642, Mar. 2016.

[6] A. Pérez-Foguet, A. Rodriguez-Ferran, and A. Huerta. Numerical differentiation for local and global tangent operators in computational plasticity. Computer Methods in Applied Mechanics and Engineering, 189(1):277–296, 2000.

[7] O. Zienkiewicz and R. L. Taylor. The finite element patch test revisited a computer test for convergence, validation and error estimates. Computer Methods in Applied Mechanics and Engineering, 149(1-4):223–254, 1997.