Mini Instant Winter University School (MIWUS)

A LLM-assisted course on Computational Metallurgy

Lecture 00: Practical Guide to Claude AI and DeepSeek LLMs

for MATLAB-Based Computational Metallurgy

Professor Fabio Miani

University of Udine, Italy

Department of Polytechnic Engineering and Architecture

Academic Year 2024/2025

January 6, 2026

Document Generation Notice:
This lecture material was **drafted using Anthropic Claude LLM** (Claude Sonnet 4.5) on January 6, 2026, through collaborative interaction between Prof. Fabio Miani and Claude AI. The document represents an example of LLM-assisted educational content creation, demonstrating the practical application of AI tools in academic course development.
All technical content has been reviewed for accuracy and appropriateness for graduate-level computational metallurgy education.

Note: This lecture provides hands-on experience with Large Language Models (LLMs) as research and coding assistants. All tools mentioned are accessible to students. All hyperlinks are clickable in the PDF version.

Contents

1 Introduction: The LLM Revolution in Research

1.1 Why This Course Starts with LLMs

Learning Goal

Understand how Large Language Models fundamentally change research workflows in computational metallurgy and develop critical skills for effective LLM usage in academic work.

Welcome to what we call **Lecture 00** – the foundation lecture that comes before everything else. Before we dive into thermodynamic calculations, phase diagrams, and solidification modeling, we need to establish our toolkit: the Large Language Models that will assist us throughout this course.

Why start here?

- LLMs are now **essential research tools**, not optional add-ons
- Learning LLM interaction is itself a **critical skill** for modern researchers
- Effective LLM use requires **practice and understanding** of capabilities/limitations
- This course is designed to be **LLM-assisted from day one**

1.2 The Paradigm Shift in Computational Research

Traditional workflow:

1. Read papers manually
2. Write code from scratch
3. Debug through trial and error
4. Document results manually

LLM-assisted workflow:

1. **Collaborate** with LLM to understand papers
2. **Co-create** code with AI assistance
3. **Rapid debug** through LLM error analysis
4. **Semi-automated** documentation and reporting

Important Warning

Critical Distinction:
LLM-assisted LLM-dependent
You remain the expert. The LLM is a powerful assistant, not a replacement for understanding. Your role is to:

- Guide the LLM with domain knowledge
- Verify all outputs against established science
- Make final decisions on methodology
- Understand every piece of code you use
- Maintain academic integrity

1.3 Course Philosophy: Transparent AI Usage

This course embraces a philosophy of **transparent, ethical AI usage**:

1. **We acknowledge** when LLMs assist in our work
2. **We verify** all LLM outputs against scientific literature
3. **We understand** that LLMs can make mistakes
4. **We learn** both with and without LLM assistance
5. **We attribute** AI contributions appropriately

LLM Tip

This very document you're reading was created through collaboration with Claude AI. Every section was reviewed by Prof. Miani for technical accuracy and pedagogical appropriateness. This is the model we follow: **human expertise + AI assistance = enhanced outcomes**.

2 Claude AI: Your Primary Research Assistant

2.1 Introduction to Claude

Claude AI Feature

Claude AI is a Large Language Model developed by Anthropic, designed specifically for:

- Safe, helpful, and honest interactions
- Extended context understanding (200K+ tokens)
- Complex reasoning and analysis
- Code generation and debugging
- Document analysis and synthesis

Current Model: Claude Sonnet 4.5 (as of January 2025)
Access: `https://claude.ai`

2.1.1 Why Claude for Computational Metallurgy?

Claude excels at:

1. **Scientific reasoning**: Understanding complex thermodynamic concepts
2. **Mathematical derivations**: Step-by-step equation development
3. **Code generation**: High-quality MATLAB code with proper documentation
4. **Document analysis**: Processing research papers and extracting key information
5. **Patient teaching**: Explaining concepts at appropriate technical levels

2.2 Getting Started with Claude

2.2.1 Account Setup

Hands-On Exercise

Exercise 0.1: Create Your Claude Account
Steps:

1. Navigate to `https://claude.ai`
2. Click "Sign Up" or "Get Started"
3. Use your University of Udine email: `name.surname@uniud.it`
4. Verify your email address
5. Complete the brief tutorial

Important: Using your university email helps establish your identity as a researcher and may provide access to institutional benefits if Anthropic partners with your university in the future.

2.2.2 Free vs. Pro Accounts

Table 1: Claude Account Tiers Comparison

Feature	Free Tier	Pro Tier ($20/month)
Message limit	30-50 per day (varies)	5× more messages
Models available	Claude Sonnet 4.5	All models (Opus, Sonnet, Haiku)
Context window	Standard (200K tokens)	Extended (200K tokens)
Response priority	Standard queue	Priority access
File uploads	Up to 5 files	Up to 5 files
Projects feature	Limited	Full access
API access	No	No (separate billing)

Recommendation for this course:

- **Start with Free tier** to learn the basics
- **Consider Pro** if you hit message limits during intensive research periods
- **Share accounts** is against ToS – each student should have their own

2.3 Claude Interface Overview

2.3.1 Main Chat Interface

The Claude interface consists of:

- **Conversation area**: Where you interact with Claude
- **Input box**: Where you type your prompts (supports Shift+Enter for new lines)
- **Artifacts panel**: Right-side panel showing generated content (code, documents)
- **Conversation history**: Left sidebar with past conversations
- **Settings menu**: Access preferences, account settings

2.3.2 Key Features for Research

1. **Artifacts**:
   - Interactive code that runs in browser
   - Documents (Markdown, LaTeX)
   - Visualizations and diagrams
   - React components for interactive tools
2. **File Upload**:
   - PDF papers (up to 5 simultaneously)
   - Excel/CSV data files
   - Images (for analysis)
   - Code files (for review)
3. **Projects**:
   - Dedicated workspaces for research topics
   - Persistent context across sessions
   - Custom instructions for specialized behavior
   - Organized conversation management

2.4 Your First Conversation with Claude

Hands-On Exercise

Exercise 0.2: Introduction to Claude Interaction
Task: Have a conversation with Claude about the course topic.
Suggested Prompts:

```
Prompt 1 (Introduction):
Hi Claude! I'm a graduate student at the University of Udine
taking a course on Computational Metallurgy with Prof. Fabio
Miani. This is my first time using an LLM for academic work.
Can you explain what you can help me with in this course?
```

```
Prompt 2 (Testing Understanding):
Can you explain what the CALPHAD method is? I've heard it
mentioned but want to understand it before our next lecture.
```

```
Prompt 3 (Code Generation):
Can you write a simple MATLAB function that calculates the
Gibbs free energy of mixing for an ideal binary solution?
Include comments explaining the thermodynamic principle.
```

Observe:

- How Claude adapts its language to your level
- The structure and completeness of responses
- Whether code appears in an Artifact
- How you can follow-up with clarifying questions

2.5 Claude for Literature Review

2.5.1 Summarizing Research Papers

Claude AI Feature

Claude can analyze research papers and extract key information, saving hours of reading time. However, you should always read papers yourself – use Claude to **enhance understanding**, not replace reading.

Hands-On Exercise

Exercise 0.3: Paper Analysis with Claude
Task: Analyze the Cu-Zr thermodynamic assessment paper.
Method 1 - With PDF Upload:

1. Download the paper: Hsiao et al., CALPHAD 55 (2016) 77-87
2. Upload PDF to Claude
3. Use this prompt:

```
I've uploaded a paper on Cu-Zr thermodynamic assessment.
Please analyze it and provide:

1. Research Objectives
   - What problem does this paper address?
   - Why is a new assessment needed?

2. Methodology
   - What experimental data did they use?
   - What thermodynamic models were employed?
   - How was optimization performed?

3. Key Results
   - What are the optimized thermodynamic parameters?
   - How do results compare with previous assessments?
   - Which phases were successfully modeled?

4. Validation
   - How did they validate the assessment?
   - What experimental data matched well?
   - Any discrepancies noted?

5. Applications
   - How could this data be used in research?
   - What future work do they suggest?

Format as a structured summary suitable for my research notes.
```

Method 2 - Without PDF (if you don't have the paper):

```
Can you provide information about the paper "Thermodynamic
assessment of the Ag-Zr and Cu-Zr binary systems" by
Hsien-Ming Hsiao et al., published in CALPHAD Volume 55,
2016, pages 77-87?

Focus on the Cu-Zr system specifically, including methodology
and key findings.
```

Follow-up Questions to deepen understanding:

```
- What thermodynamic models did they use for the liquid phase?
- How does their assessment differ from the 2004 assessment?
- What intermetallic compounds exist in the Cu-Zr system?
- Can you explain the optimization procedure they used?
```

2.5.2 Comparative Literature Analysis

Hands-On Exercise

Exercise 0.4: Comparing Multiple Papers
Task: Compare grain refinement mechanisms across different papers.

```
I'm studying grain refinement in cast metals. Please help me
compare these two papers:

Paper 1: "Grain Refinement of Deoxidized Copper" by Balart
et al., Metallurgical and Materials Transactions A (2016)

Paper 2: "Investigation of the correlation between growth
restriction and grain size in Cu alloys" by Cziegler &
Schumacher (2017)

Create a comparison table with:
- Alloy systems studied
- Grain refinement mechanisms proposed
- Role of growth restriction factor (Q)
- Experimental methods used
- Key conclusions
- Agreement/disagreement between papers

Then explain which paper provides better guidance for
designing grain refinement experiments.
```

2.6 Claude for Concept Explanation

2.6.1 Understanding Complex Thermodynamics

Hands-On Exercise

Exercise 0.5: Deep Dive into Growth Restriction Factor
Multi-stage Conversation:

```
Stage 1 - Basic Explanation:
Explain the Growth Restriction Factor (Q) in grain refinement.
What does it represent physically?
```

```
Stage 2 - Mathematical Derivation:
Now derive the equation Q = m(k-1)C_0 from first principles,
starting with constitutional supercooling. Show each step.
```

```
Stage 3 - Physical Interpretation:
Explain what happens to grain size when:
- Q is very large
- Q is very small
- Two solutes are combined

Use the Cu-Zr system as an example.
```

```
Stage 4 - Practical Application:
Given this data for elements in copper at 1 wt%:
- Ni: Q = 67.52 K
- Zr: Q = 8.39 K
- Zn: Q = 1.03 K

Which would be most effective for grain refinement? Why?
What are the practical considerations for each?
```

Learning Outcome: Understanding how to use Claude for progressive, deep learning rather than just quick answers.

2.7 Claude for Code Generation

2.7.1 MATLAB Code with Proper Documentation

Hands-On Exercise

Exercise 0.6: Generate Growth Restriction Calculator
Comprehensive Prompt:

```
Create a MATLAB function to calculate the Growth Restriction
Factor (Q) from solidification data.

Context: I'm analyzing Scheil solidification simulations where
I have temperature (T) vs solid fraction (fs) data exported
from Pandat. The growth restriction factor is defined as the
initial slope of the solidification curve: Q = |dT/dfs| at
    f s 0

Requirements:
1. Function signature: Q = calculate_GRF(fs, T, options)
2. Inputs:
   - fs: array of solid fractions (mole/mole)
   - T: array of temperatures ( C )
   - options: struct with optional parameters
     * options.plot: boolean, whether to create plot (default
        true)
     * options.fitOrder: polynomial order (default 2)
     * options.verbose: boolean, print details (default true)

3. Processing:
   - Validate inputs (same length, monotonic fs, etc.)
   - Fit polynomial to T vs fs data
   - Calculate Q from derivative at fs=0
   - Compute R-squared for goodness of fit
   - Optionally create publication-quality plot

4. Outputs:
   - Q: growth restriction factor (K)
   - fitParams: structure with fit parameters and statistics
   - figHandle: handle to figure (if plotting enabled)

5. Code Quality:
   - Extensive comments for educational use
   - Input validation with clear error messages
   - Professional plot formatting
   - Example usage in header
   - Unit tests suggestions in comments

6. Example Data to Include:
   fs = [0, 0.004, 0.008, 0.016, 0.031, 0.062, 0.119];
   T = [1335.78, 1335.58, 1335.38, 1334.98, 1334.18, 1332.58,
      1329.38];

Make this suitable for teaching graduate students who are
learning both MATLAB and computational metallurgy.
```

Expected Outcome: A complete, well-documented MATLAB function that you

2.7.2 Improving Generated Code

LLM Tip

Never accept the first version of code. Always refine through dialogue:
Refinement Prompts:

```
- "Add error handling for edge cases like empty arrays"
- "Create a version optimized for large datasets (>10000
   points)"
- "Add export functionality to save results as Excel file"
- "Generate comprehensive unit tests"
- "Add GUI interface using GUIDE"
- "Optimize this code for speed"
- "Add parallel processing support"
```

Each refinement makes the code more robust and feature-rich.

2.8 Claude Projects for Course Organization

2.8.1 Setting Up Your MIWUS Project

Claude AI Feature

Projects are dedicated workspaces in Claude that maintain context across multiple conversations. They're perfect for course work.

Hands-On Exercise

Exercise 0.7: Create Your MIWUS Project
Steps:

1. In Claude, click "Projects" in the left sidebar
2. Click "New Project"
3. Name it: "MIWUS - Computational Metallurgy 2024/25"
4. Add project description:

```
This project is for the Mini Instant Winter University
   School
course on Computational Metallurgy taught by Prof. Fabio
   Miani
at the University of Udine.

Course focus areas:
- CALPHAD method and thermodynamic databases
- Phase diagram calculation and interpretation
- Solidification modeling (Lever rule, Scheil equation)
- Growth restriction factor in grain refinement
- Cu-Zr-Al ternary systems
- MATLAB programming for metallurgical calculations

Tools used:
- MATLAB (full license)
- Pandat (free 3-component version)
- Open Calphad
- CPDDB database

I'm a graduate student learning computational methods in
metallurgy. Please provide detailed explanations suitable
for this level, always include proper documentation in code,
and verify calculations against established literature when
possible.
```

5. Upload course materials:
   - Lecture 01 PDF
   - Key papers (Cu-Zr assessment, grain refinement papers)
   - Example datasets
6. Start conversations within this project

Benefit: Claude remembers your course context across all conversations in this project, providing more relevant and consistent assistance.

2.8.2 Custom Instructions for Your Project

Within your project, you can add custom instructions:

```
Custom Instructions for MIWUS Project:

1. Code Preferences:
   - Always generate MATLAB code, not Python (unless explicitly
     requested)
   - Include detailed comments suitable for learning
   - Use meaningful variable names, not single letters
   - Provide example usage with realistic metallurgy data

2. Explanation Style:
   - Start with physical intuition before mathematical details
   - Use analogies from everyday experience when helpful
   - Reference specific papers from the course materials
   - Clearly state assumptions and limitations

3. Verification:
   - Always suggest how to verify results
   - Reference established databases (CPDDB, Thermo-Calc)
   - Mention when results should be checked against literature

4. Problem-Solving Approach:
   - Break complex problems into steps
   - Explain the reasoning behind each approach
   - Offer alternative methods when appropriate
   - Highlight common pitfalls in metallurgical calculations
```

3 DeepSeek: High-Performance Alternative

3.1 Introduction to DeepSeek

DeepSeek Feature

DeepSeek is an advanced open-source LLM developed in China, known for exceptional performance in:

- Mathematical reasoning and derivations
- Code generation (especially Python and MATLAB)
- Scientific computations
- Long-form technical writing

Current Model: DeepSeek-V3 (as of January 2025)
Access: `https://chat.deepseek.com`

3.2 DeepSeek vs Claude: Strategic Usage

Table 2: When to Use Claude vs DeepSeek

Task Type	Use Claude	Use DeepSeek
Literature review	Better document analysis	Standard capability
Concept explanation	More conversational	More technical/formal
Code generation	Excellent documentation	Slightly more concise
Mathematical proofs	Step-by-step clear	Often more rigorous
Debugging	Patient explanations	Quick identification
Heavy computation	Good	Optimized algorithms
Batch processing	Limited by interface	Better for scripts
API integration	Paid API	Free tier + paid
Interactive tools	Artifacts feature	Standard output
Long documents	Projects feature	Standard memory

Recommended Strategy:

- **Primary tool**: Claude (for its interface and features)
- **Complement with DeepSeek** when:
  - You need mathematical rigor
  - You're generating complex algorithms
  - You want a second opinion on code
  - You've hit Claude's message limit

3.3 Getting Started with DeepSeek

3.3.1 Account Creation

Hands-On Exercise

Exercise 0.8: Set Up DeepSeek Account
Steps:

1. Visit `https://chat.deepseek.com`
2. Sign up with email (can use same university email)
3. Verify account
4. Explore interface

Note: DeepSeek's free tier is quite generous. Pro tier offers even more tokens but isn't necessary for this course.

3.4 DeepSeek for Mathematical Derivations

Hands-On Exercise

Exercise 0.9: Scheil Equation Derivation
Prompt for DeepSeek:

```
Derive the Scheil equation for non-equilibrium solidification:

C_L = C_0 * (1 - f_s)^(k-1)

Starting from:
1. Mass balance during solidification
2. Assumption: no diffusion in solid
3. Assumption: complete mixing in liquid
4. Definition of partition coefficient k = C_S/C_L

Provide:
- Complete mathematical derivation
- Physical interpretation of each step
- Conditions where this equation applies
- Limitations of the model
- Comparison with lever rule (equilibrium solidification)

Use rigorous mathematical notation suitable for a graduate
thermodynamics course.
```

Compare: Try the same prompt with Claude and compare the approaches. DeepSeek often provides more mathematical rigor, while Claude may emphasize physical intuition more.

3.5 DeepSeek for Algorithm Development

Hands-On Exercise

Exercise 0.10: Numerical Scheil Solver
Advanced Prompt:

```
Develop a numerical solver for the Scheil equation applied
to a Cu-5wt%Zr alloy.

Given:
- Initial composition: C_0 = 5 wt% Zr
- Partition coefficient: k = 0.12 (from liquidus/solidus
   slopes)
- Temperature range: 1200 C  to 900 C
- Liquidus equation: T_L = 1084 - 15*C_Zr (simplified)

Requirements:
1. Implement Scheil equation: C_L = C_0*(1-fs)^(k-1)
2. Calculate temperature as function of solid fraction
3. Track composition of solid and liquid phases
4. Determine final solidification temperature
5. Compare with lever rule prediction

Provide MATLAB code with:
- Vectorized calculations for efficiency
- Publication-quality plots
- Detailed comments explaining thermodynamics
- Validation against known results

Also explain the numerical method used and its accuracy.
```

Expected Outcome: High-performance, well-optimized MATLAB code with rigorous numerical approach.

3.6 Using Both LLMs Together

LLM Tip

The Two-LLM Strategy:
For complex problems, use both LLMs sequentially:
Step 1 - Claude: Generate initial approach and code

```
"Create MATLAB code to analyze solidification data..."
```

Step 2 - DeepSeek: Optimize and validate

```
"Review this MATLAB code for numerical stability and
optimization opportunities..." [paste Claude's code]
```

Step 3 - Claude: Enhance documentation and usability

```
"Add comprehensive documentation and user-friendly features
to this code..." [paste DeepSeek's optimized version]
```

Result: Code that combines Claude's clarity with DeepSeek's performance.

4 MATLAB Vibe Coding: The Modern Workflow

4.1 What is Vibe Coding?

Vibe coding is a modern development approach where you maintain creative flow by describing what you want in natural language, letting AI generate code, then iteratively refining.

Traditional coding workflow:

1. Think about problem
2. Look up MATLAB syntax
3. Write code line by line
4. Debug errors
5. Repeat (often breaking flow state)

Vibe coding workflow:

1. Describe desired functionality in natural language
2. AI generates working code
3. Test immediately
4. Refine through conversation

5. Maintain creative momentum

4.2 The MATLAB + LLM Development Cycle

Rapid Development Cycle:
1. **Describe** → Tell LLM what you need
↓
2. **Generate** → LLM creates MATLAB code
↓
3. **Copy** → Paste into MATLAB
↓
4. **Run** → Execute and observe results
↓
5. **Evaluate** → Does it work?
• Yes → Continue to enhancement
• No → Copy error back to LLM
↓
6. **Refine** → Request improvements
↓
7. **Document** → Ask LLM to add documentation
↓
8. **Deploy** → Use in your research

Figure 1: The LLM-assisted MATLAB development cycle

4.3 Practical Example: Complete Workflow

Hands-On Exercise

Exercise 0.11: Full Development Cycle
Goal: Create a tool to analyze multiple solidification datasets and calculate Q for each.
Phase 1 - Initial Generation (Claude):

```
Create a MATLAB script that:

1. Reads multiple Excel files from a folder
   - Filename pattern: "CuZrAl_XAl.xlsx" where X is Al content
   - Each file has two columns: fs (solid fraction), T (
     temperature)

2. For each file:
   - Extract Al content from filename
   - Calculate growth restriction factor Q
   - Store composition and Q value

3. Generate outputs:
   - Plot Q vs Al content
   - Display table of results
   - Export summary to Excel

4. Requirements:
   - Handle missing files gracefully
   - Validate data format
   - Professional plotting style
   - Extensive comments

Provide complete, runnable code.
```

Phase 2 - Test in MATLAB:

1. Copy generated code
2. Save as `analyze_batch_solidification.m`
3. Create test data files
4. Run script
5. Note any errors

Phase 3 - Debug (if errors occur):

```
I'm getting this error when running the code:

[paste complete error message]

Here's the relevant section of code:

[paste problematic section]

The error occurs when processing file "CuZrAl_5Al.xlsx"

Can you:
```

4.4 MATLAB-Specific Tips

4.4.1 Leveraging MATLAB's Full License

You have access to all MATLAB toolboxes. Explicitly mention this when prompting:

```
I have MATLAB 2023b with full toolbox access, including:
- Optimization Toolbox
- Curve Fitting Toolbox
- Statistics and Machine Learning Toolbox
- Parallel Computing Toolbox

Please use appropriate advanced functions to optimize this
calculation. If advanced toolbox functions significantly
improve the solution, use them and explain the benefit.
```

4.4.2 GNU Octave: Free Alternative for Independent Learning

Resource

For Independent Learners Outside University of Udine
If you don't have access to a MATLAB license, **GNU Octave** is an excellent free alternative:
What is Octave?

- Free, open-source software
- MATLAB-compatible syntax (90-95% compatible)
- Available for Windows, Mac, Linux
- Active community and extensive documentation

Download: https://octave.org/download
Key Differences from MATLAB:

- Some advanced toolbox functions not available
- Plotting syntax slightly different
- Generally slower for large computations
- No GUI development tools (GUIDE)
- But: Perfect for learning and most research tasks!

When Using LLMs with Octave:

```
Important: Specify in your prompts:

"I'm using GNU Octave, not MATLAB. Please ensure the code
is Octave-compatible and avoid MATLAB-specific toolbox
functions. Suggest open-source alternatives if needed."
```

Example Octave-Specific Prompt:

```
Create an Octave script to fit quadratic curves to
solidification data. Use only core Octave functions
(no toolboxes). If optimization would benefit from
a toolbox function, explain the basic algorithm so I
can implement it manually.
```

Most code in this course works directly in Octave with minimal modification!

5 Verification and Validation of LLM Outputs

5.1 Critical Verification Principles

⚠ Important Warning

The Golden Rule of LLM-Assisted Research:
TRUST, BUT VERIFY
LLMs are powerful tools but can:

- Make mathematical errors
- Misunderstand context
- Generate plausible-sounding but incorrect code
- Confuse similar concepts
- Hallucinate references or data

Your responsibility: Verify every critical output against established sources.

5.2 Verification Checklist for Code

Hands-On Exercise

Exercise 0.12: Code Verification Protocol

When you receive MATLAB/Octave code from an LLM, verify:

1. Dimensional Analysis

```
Check with LLM:
"Review this code for dimensional consistency. Verify that
all units are correct and conversions are properly handled.
Show the units for each variable."
```

2. Limiting Cases

- Pure component limits (x = 0 or x = 1)
- Zero temperature difference
- Infinite partition coefficient

```
Ask LLM:
"What should this function return when:
- fs = 0 (start of solidification)
- fs = 1 (complete solidification)
- k = 1 (no partitioning)

Add assert statements to verify these cases."
```

3. Known Results Comparison

```
Test with published data:
"I have reference data from [paper citation]. The growth
restriction factor for Cu-1wt%Zr should be approximately
8.37 K. Run the function with this data and verify the
result matches within 5%."
```

4. Physical Reasonableness

- Temperature decreases during solidification?
- Compositions stay between 0 and 1?
- Phase fractions sum to 1?

5. Edge Cases

- Empty arrays
- Single data point
- Negative values
- Non-monotonic data

```
Request from LLM:
"Add comprehensive input validation that checks for and
handles all edge cases. Provide meaningful error messages
for each potential issue."
```

5.3 Verification Workflow Example

```
%% Verification Script for Growth Restriction Factor Calculator
% Compare LLM-generated code against known results

% Test Case 1: Cu-1wt%Zr from Cziegler & Schumacher (2017)
fprintf('Test 1: Cu-1wt%%Zr system\n');
fs_test1 = [0, 0.1, 0.2, 0.3, 0.4];
T_test1 = [1084, 1083, 1081, 1078, 1075]; % Hypothetical data
Q_test1 = calculate_GRF(fs_test1, T_test1);
Q_expected1 = 8.37; % From literature
error_pct1 = abs(Q_test1 - Q_expected1) / Q_expected1 * 100;
fprintf('  Calculated Q: %.2f K\n', Q_test1);
fprintf('  Expected Q: %.2f K\n', Q_expected1);
fprintf('  Error: %.1f%%\n', error_pct1);

if error_pct1 < 10
    fprintf('     PASS\n\n');
else
    fprintf('     FAIL - Error exceeds 10%%\n\n');
end

% Test Case 2: Limiting behavior (fs     0)
fprintf('Test 2: Limiting behavior at fs=0\n');
fs_test2 = [0, 0.001, 0.002, 0.003];
T_test2 = [1335.78, 1335.73, 1335.68, 1335.63];
Q_test2 = calculate_GRF(fs_test2, T_test2);
% At  f s 0 , Q should equal the linear coefficient
expected_slope = (T_test2(2) - T_test2(1)) / (fs_test2(2) - fs_test2(1))
    ;
fprintf('  Q from function: %.2f K\n', Q_test2);
fprintf('  Direct slope calculation: %.2f K\n', abs(expected_slope));
fprintf('  Match: %s\n\n', iif(abs(Q_test2 - abs(expected_slope)) < 1, '
      ', '    '));

% Test Case 3: Physical bounds
fprintf('Test 3: Physical reasonableness\n');
assert(Q_test1 > 0, 'Q must be positive');
assert(Q_test1 < 100, 'Q should be less than 100 K for typical systems')
    ;
fprintf('     Physical bounds satisfied\n\n');

% Test Case 4: Edge cases
fprintf('Test 4: Edge case handling\n');
try
    Q_bad = calculate_GRF([], []); % Empty arrays
    fprintf('     FAIL - Should reject empty arrays\n');
catch ME
    fprintf('     PASS - Properly rejects empty arrays\n');
end

fprintf('\n=== Verification Complete ===\n');
```

Listing 1: Verification script template

5.4 Validating Thermodynamic Calculations

Hands-On Exercise

Exercise 0.13: Cross-Platform Validation
Multi-Tool Verification Strategy:
Step 1: Generate calculation with Claude/DeepSeek

```
"Calculate the solidification range for Cu-5wt%Zr using
the Scheil equation with k=0.12, starting temperature
1084 C , and liquidus slope -15 K/wt%."
```

Step 2: Compare with Pandat

- Run same composition in Pandat
- Export Scheil solidification curve
- Compare temperature ranges

Step 3: Check against CPDDB

- Look up Cu-Zr phase diagram
- Verify liquidus/solidus temperatures
- Check intermetallic phases

Step 4: Literature cross-reference

- Find published Cu-Zr data
- Compare with your calculations
- Document any discrepancies

Acceptance Criteria:

- Temperature agreement within ± 5°C
- Phase identification matches
- Trends consistent across methods

If discrepancies found:

```
Return to LLM with:
"I compared your calculation with Pandat and found a 10 C
difference in solidus temperature. Here's the data:
- Your calculation: 920 C
- Pandat result: 930 C
- Database used: PanCu2018

Can you:
1. Identify potential sources of difference
2. Verify the calculation method
3. Suggest which result is more reliable
4. Explain what might cause this discrepancy"
```

6 Debugging with LLM Assistance

6.1 The LLM Debugging Workflow

Efficient Debugging Protocol:
1. Error Occurs
→ Don't panic! MATLAB/Octave errors are expected
2. Collect Information

- Full error message
- Line number where error occurred
- Input values that caused error
- Relevant code section

3. Paste to LLM with context:

```
"I'm getting this error in my solidification analysis:

[Full error message]

Here's the code section:
[Lines X-Y from your script]

The error occurs when processing file with:
- fs range: [0, 0.5]
- T range: [1335, 1280]

What's causing this and how do I fix it?"
```

4. Implement Fix
→ LLM provides corrected code + explanation
5. Test Fix
→ Run with original problematic input
6. If Still Failing
→ Return to LLM with new error message

Figure 2: LLM-assisted debugging workflow

6.2 Common MATLAB Errors and LLM Solutions

Hands-On Exercise

Exercise 0.14: Debugging Practice
Scenario: Intentionally create and fix errors
Error Type 1: Dimension Mismatch

```
% Buggy code
fs = [0, 0.1, 0.2, 0.3, 0.4]';  % Column vector
T = [1335, 1330, 1325, 1320, 1315];  % Row vector
plot(fs, T);  % May cause issues in some operations
```

Paste to LLM:

```
"Getting dimension mismatch errors. Here's my code:
[paste above]
How do I ensure consistent dimensions and avoid this issue?"
```

Error Type 2: Division by Zero

```
% Buggy code
Q = abs(coeff(2));  % What if coeff(2) is undefined?
```

Error Type 3: File Not Found

```
% Buggy code
data = readtable('CuZrAl_5Al.xlsx');  % File might not exist
```

For each error:

1. Run the buggy code
2. Capture complete error message
3. Paste to Claude/DeepSeek with context
4. Implement suggested fix
5. Verify fix works

Learning Outcome: Develop efficient debugging skills using LLMs.

6.3 Advanced Debugging: Performance Issues

```
Prompt for Performance Debugging:

"This code is very slow when processing 50 files with 500
data points each. It takes about 5 minutes to complete.

[Paste code]

Can you:
1. Profile the code and identify bottlenecks
2. Suggest specific optimizations
```

```
3. Provide optimized version
4. Estimate expected speedup

I have MATLAB with Parallel Computing Toolbox available."
```

7 Best Practices and Ethical Guidelines

7.1 Academic Integrity in LLM-Assisted Work

⚠ Important Warning

University of Udine Policy on AI Usage
Always check your institution's current AI usage policy. General principles:
Acceptable Use:

- Learning and understanding concepts
- Debugging and improving code
- Generating initial drafts for refinement
- Exploring alternative approaches
- Literature summarization (with verification)

Requires Attribution:

- Substantial code generated by LLMs
- Text or analysis significantly shaped by LLMs
- Novel approaches suggested by LLMs

Not Acceptable:

- Submitting LLM output without understanding
- Claiming LLM work as entirely your own
- Using LLMs during closed-book exams (unless permitted)
- Bypassing learning objectives through LLM shortcuts

7.2 Proper Attribution Examples

7.2.1 In Code Comments

```
function Q = calculate_GRF(fs, T)
% CALCULATE_GRF Calculate Growth Restriction Factor
%
```

```
% Author: [Your Name]
% Course: MIWUS Computational Metallurgy 2024/25
% Date: January 2026
%
% Code Development: Initial implementation generated with
% assistance from Claude AI (Anthropic) and subsequently
% modified and verified against Cziegler & Schumacher (2017).
% All thermodynamic principles and validation performed by author.
%
% Algorithm verified against:
% - Pandat simulation results
% - Published data in CALPHAD journal
% - Hand calculations for limiting cases

[... rest of code ...]
```

Listing 2: Proper code attribution

7.2.2 In Research Reports

Example Acknowledgment Section:

"The MATLAB scripts for solidification analysis were developed with assistance from Claude AI (Anthropic, Claude Sonnet 4.5) and DeepSeek-V3 for initial code generation and optimization. All code was subsequently verified against established thermodynamic principles, validated using Pandat software, and tested with published experimental data. The author takes full responsibility for the scientific accuracy and interpretation of all results."

7.2.3 In LinkedIn/Public Sharing

LinkedIn Post Template

Sharing Your LLM-Assisted Work:
"Excited to share my computational metallurgy analysis of Cu-Zr-Al solidification!
This project demonstrates modern research workflows: Claude AI for rapid MATLAB prototyping DeepSeek for algorithm optimization Validation against Pandat & published data Full thermodynamic verification
Key finding: Growth restriction factor varies non-linearly with Al content in Cu-Zr-Al system...
[Continue with your findings]
#ComputationalMaterials #AI assisted Research #MIWUS #UniversityOfUdine
Code and methodology available upon request. All LLM assistance properly documented and verified."

7.3 Recognizing LLM Limitations

LLM Tip

What LLMs Do Well:

- Generate boilerplate code quickly
- Explain concepts clearly
- Debug common errors
- Suggest alternative approaches
- Format and document code
- Translate between programming languages

What LLMs Struggle With:

- Novel research requiring creativity
- Highly domain-specific thermodynamic databases
- Cutting-edge methods published after training cutoff
- Complex multi-step numerical optimizations
- Verifying their own outputs
- Understanding your specific experimental setup

Strategy: Use LLMs for their strengths, apply human expertise for their weaknesses.

8 Course Integration and Workflow

8.1 MIWUS Course Structure

Resource

Mini Instant Winter University School (MIWUS)
Focus: Computational Metallurgy
Instructor: Prof. Fabio Miani
Institution: University of Udine, Italy
Course Components:

1. **Lecture 00** (this lecture): LLM Tools (Claude, DeepSeek, MATLAB/Octave)
2. **Lecture 01**: Introduction to Computational Metallurgy
   - CALPHAD method
   - Thermodynamic databases
   - Contemporary research topics
3. **Subsequent Lectures**:
   - Phase diagram calculations
   - Solidification modeling
   - Growth restriction and grain refinement
   - Ternary systems analysis
   - Advanced applications

Related Course: **MIFUS** (Mini Instant Fall University School) - Steel Decarbonization

8.2 Integrated LLM Workflow for MIWUS

Weekly Research Workflow with LLM Integration:
Monday - Literature Review

- Upload papers to Claude Project
- Request summaries and comparisons
- Identify key equations and methods

Tuesday/Wednesday - Code Development

- Describe needed functionality to LLM
- Generate initial MATLAB/Octave code
- Test and debug with LLM assistance
- Verify against known results

Thursday - Analysis

- Run calculations on real data
- Use LLM for interpretation assistance
- Cross-check with Pandat/OpenCalphad

Friday - Documentation

- LLM-assisted report drafting
- Code documentation enhancement
- Prepare visualizations
- Final verification and attribution

Throughout Week:

- Ask LLMs to explain unclear concepts
- Request derivations of key equations
- Get debugging assistance
- Explore alternative methodologies

Figure 3: Integrated weekly workflow for MIWUS course

8.3 Preparing for MIFUS: Steel Decarbonization

Resource

Upcoming Course: MIFUS - Steel Decarbonization
LLM Applications for Steel Decarbonization:

1. **Process Modeling**:

```
"Develop MATLAB code to model carbon emissions from
blast furnace vs electric arc furnace steelmaking.
Include energy balances and CO2 calculations."
```

2. **Alternative Technologies**:

```
"Explain hydrogen direct reduction (H-DR) technology
for ironmaking. Compare thermodynamics with traditional
carbon-based reduction."
```

3. **Life Cycle Analysis**:

```
"Create a simplified LCA model for steel production
routes. Calculate total carbon footprint including
raw materials, energy, and transportation."
```

4. **Economic Analysis**:

```
"Develop cost model comparing conventional vs green
steel production. Include carbon pricing scenarios."
```

Recommended Preparation:

- Set up separate Claude Project for MIFUS
- Familiarize yourself with molten oxide electrolysis (Prof. Sadoway's lecture)
- Practice LLM-assisted literature review on decarbonization technologies
- Prepare MATLAB/Octave environment for process calculations

9 Student Assignments and Evaluation

9.1 Assignment 1: LLM Tool Mastery

Hands-On Exercise

Assignment 0.1: Demonstrate LLM Proficiency
Due Date: [To be announced]
Objective: Demonstrate effective use of Claude and/or DeepSeek for metallurgical research.
Tasks:
Part A: Literature Analysis (30%)

1. Select one paper from the course reading list
2. Use Claude to generate comprehensive summary
3. Verify key facts against the original paper
4. Identify any errors or misinterpretations by the LLM
5. Write 1-page reflection on LLM accuracy and usefulness

Part B: Code Development (40%)

1. Choose a thermodynamic calculation relevant to the course
2. Use LLM to generate MATLAB/Octave code
3. Document your prompt engineering process
4. Verify results against at least one independent source
5. Submit:
   - Initial prompt
   - Generated code with your modifications
   - Verification results
   - Attribution statement

Part C: Debugging Exercise (20%)

1. Take the provided buggy code
2. Use LLM to identify and fix errors
3. Document the debugging conversation
4. Explain what you learned about the error

Part D: Reflection (10%)

- What worked well with LLM assistance?
- What were the limitations?
- How will you integrate LLMs in your research workflow?
- Ethical considerations you identified

Submission Format:

9.2 Assignment 2: Integrated Research Project

Hands-On Exercise

Assignment 0.2: LLM-Assisted Research Mini-Project
Objective: Complete a small research task using full LLM-assisted workflow.
Project Options:
Option 1: Cu-Zr-Al Ternary Analysis

- Calculate solidification paths for 5 compositions
- Determine growth restriction factors
- Predict grain refinement effectiveness
- Compare with published data

Option 2: Database Comparison

- Compare CPDDB vs Pandat for a binary system
- Analyze discrepancies in phase boundaries
- Investigate sources of differences
- Recommend which to use for specific applications

Option 3: Custom Tool Development

- Develop comprehensive analysis tool
- Create GUI interface (MATLAB) or interactive plots (Octave)
- Include validation suite
- Write user documentation

Requirements:

1. Use both Claude and DeepSeek at different stages
2. Document all LLM assistance with screenshots
3. Verify all results independently
4. Provide proper attribution
5. Include reflection on LLM effectiveness

Deliverables:

- Technical report (10-15 pages)
- Fully documented code
- Verification data and plots
- LLM interaction log
- LinkedIn-ready summary post (optional, for bonus points)

9.3 Grading Rubric for LLM-Assisted Work

Table 3: Evaluation Criteria for LLM-Assisted Assignments

Criterion	Description	Weight
Technical Accuracy	Results verified against literature/software	30%
Prompt Engineering	Quality and effectiveness of LLM prompts	15%
Code Quality	Documentation, readability, robustness	20%
Understanding	Demonstrated comprehension of methods	20%
Attribution	Proper credit to LLM assistance	5%
Critical Thinking	Analysis of LLM limitations and verification	10%

Note: Simply pasting LLM output without understanding or verification will receive low marks. The goal is to demonstrate **effective collaboration** with AI tools, not dependency on them.

10 Advanced Topics and Future Directions

10.1 API Integration for Automation

LLM Tip

For Advanced Users: Both Claude and DeepSeek offer API access for automation.
Use Cases:

- Batch processing of calculations with LLM verification
- Automated report generation
- Interactive research notebooks
- Custom research tools with built-in LLM assistance

Example Application:

```
# Python script to automate solidification analysis
import anthropic  # Claude API
# or: import openai  # DeepSeek compatible API

# Process multiple compositions automatically
for composition in composition_list:
    # Calculate with MATLAB/Octave
    result = run_solidification_analysis(composition)

    # Verify with LLM
    verification = claude.verify_thermodynamic_result(result)

    # Generate interpretation
    interpretation = claude.interpret_metallurgy(result)

    # Compile report
    generate_report(result, verification, interpretation)
```

Learning Resources:

- Claude API: https://docs.anthropic.com/
- DeepSeek API: https://platform.deepseek.com/

10.2 Emerging LLM Capabilities

Trends to Watch:

1. **Multimodal LLMs**: Analyzing metallographic images directly
2. **Extended Context**: Entire textbooks and databases as context
3. **Specialized Models**: Domain-specific LLMs for materials science

4. **Interactive Simulations**: Real-time thermodynamic calculations
5. **Collaborative Research**: Multiple LLMs working together

10.3 Resources for Continued Learning

Resource

Recommended Resources:
LLM Usage:

- Anthropic Documentation: https://docs.anthropic.com/
- DeepSeek Documentation: https://platform.deepseek.com/docs
- Prompt Engineering Guide: https://www.promptingguide.ai/

MATLAB/Octave:

- MATLAB Documentation: https://www.mathworks.com/help/
- GNU Octave Manual: https://docs.octave.org/
- MATLAB Central File Exchange: https://www.mathworks.com/matlabcentral/

Computational Metallurgy:

- CALPHAD Journal: https://www.journals.elsevier.com/calphad
- OpenCalphad: http://www.opencalphad.com/
- NIMS CPDDB: https://cpddb.nims.go.jp/
- Prof. Miani's Resources: http://www.gotrawama.eu/

LinkedIn Learning:

- Follow #ComputationalMetallurgy
- Connect with researchers in the field
- Share your LLM-assisted projects (with proper attribution)

11 Conclusion and Next Steps

11.1 Key Takeaways from Lecture 00

You are now equipped to:

1. **Access and use** Claude AI and DeepSeek effectively

2. **Generate high-quality code** for metallurgical calculations
3. **Verify and validate** LLM outputs rigorously
4. **Debug efficiently** using LLM assistance
5. **Maintain academic integrity** with proper attribution
6. **Use GNU Octave** as a free MATLAB alternative
7. **Integrate LLMs** into your research workflow

11.2 Preparing for Lecture 01

Hands-On Exercise

Pre-Lecture 01 Tasks:

1. Set Up Your Environment

- Claude account created and tested
- DeepSeek account created
- MATLAB/Octave installed and working
- MIWUS Project in Claude configured

2. Explore Course Resources

- Download Lecture 01 materials
- Watch Prof. Bhadeshia's YouTube lecture
- Register at CPDDB (NIMS)
- Download Pandat free version

3. Practice with LLMs

```
Ask Claude to explain:
"What is the CALPHAD method? Explain it to a graduate
student who understands thermodynamics but hasn't done
computational metallurgy. Include historical context and
modern applications."
```

4. Test MATLAB/Octave

```
% Simple test script
fs = linspace(0, 1, 100);
T = 1335 - 50*fs.^2;
plot(fs, T);
xlabel('Solid Fraction');
ylabel('Temperature ( C )');
title('Test Solidification Curve');
grid on;
```

If this works, you're ready for Lecture 01!

11.3 Philosophy for the Course Ahead

The MIWUS Approach

Human-AI Collaboration in Research:
This course represents a new paradigm in engineering education where:

- **Technology amplifies expertise**, not replaces it
- **Verification is paramount** - trust but always verify
- **Understanding comes first** - tools second
- **Transparency matters** - acknowledge all assistance
- **Efficiency enables depth** - use saved time for deeper learning

As Prof. Miani emphasizes: *"LLMs are powerful assistants that allow us to focus on the science, not syntax. But never forget - YOU are the metallurgist, not the AI."*
For the journey ahead:

1. Use these tools boldly but wisely
2. Question everything, verify rigorously
3. Share your learning with proper attribution
4. Help build the community of LLM-assisted researchers

Remember: This course itself was co-created with Claude AI, demonstrating the very principles we teach. The synergy between human expertise and AI capability is the future of computational research.

11.4 Connect and Share

Join the MIWUS Community

Prof. Fabio Miani invites you to connect:
LinkedIn: Share your progress, insights, and projects

- Use hashtags: #MIWUS #ComputationalMetallurgy #LLMAssistedResearch
- Tag: @FabioMiani @UniversityOfUdine
- Show your work (with proper LLM attribution!)

Course Updates:

- Additional resources and tutorials
- Guest lectures and industry insights
- Research opportunities
- Upcoming MIFUS (Steel Decarbonization) announcements

Community Guidelines:

- Share generously, attribute properly
- Ask questions openly
- Collaborate respectfully
- Celebrate each other's progress

11.5 Looking Ahead: MIFUS Preview

Resource

Upcoming: MIFUS - Steel Decarbonization
When: [To be announced - few weeks]
Focus: Applying computational methods to steel industry sustainability
LLM Applications You'll Use:

- Process simulation and optimization
- Carbon footprint modeling
- Economic scenario analysis
- Technology comparison studies
- Policy impact assessment

Why This Matters: Steel production accounts for 7-9% of global CO emissions. Computational tools, enhanced by LLMs, will be critical for:

- Designing decarbonization strategies
- Optimizing new processes (H-DR, electrolysis)
- Economic feasibility analysis
- Supporting policy decisions

Preparation: The skills you develop in MIWUS will directly transfer to MIFUS. Both courses share the same LLM-assisted methodology.

11.6 Final Words

Welcome to the future of computational metallurgy education!

You are among the first students to experience a fully LLM-integrated course. What you learn here about human-AI collaboration will serve you throughout your career, regardless of which specific tools become dominant.

The key insights:

- **Adaptability**: Tools change, but principles of verification and critical thinking remain
- **Efficiency**: LLMs free you to focus on science, not syntax
- **Responsibility**: With powerful tools comes responsibility for accuracy and attribution
- **Community**: Share your learning to accelerate collective progress

You are ready to begin your computational metallurgy journey!

See you in Lecture 01, where we'll dive into the CALPHAD method, thermodynamic databases, and start calculating real phase diagrams - all with your new LLM assistants by your side.

Buono studio! Good studies!

Prof. Fabio Miani
University of Udine
January 6, 2026

Final Note on This Document:
This lecture material was created through collaborative interaction between Prof. Fabio Miani and Claude AI (Anthropic, Sonnet 4.5) on January 6, 2026. It demonstrates the very principles it teaches: human expertise guiding AI capability to create educational content that neither could produce as effectively alone.
The content has been reviewed for technical accuracy, pedagogical appropriateness, and alignment with University of Udine academic standards.
Document Status: Draft for review and refinement
Next Steps: Faculty review, student feedback, iterative improvement
This is education in the age of AI - transparent, collaborative, and continuously evolving.

A Quick Reference Guide

A.1 Essential Prompts for Computational Metallurgy

```
I'm studying [specific topic] for my computational
metallurgy course.

Paper: [Full citation]

Please provide:
1. Main objectives and motivation
2. Methodology (be specific about models used)
3. Key results and findings
4. Validation methods
5. Limitations noted by authors
6. How this relates to [your research question]

Format as structured notes for my research journal.
```

Listing 3: Literature Review Prompt Template

```
Create [MATLAB/Octave] code for [specific task].

Context: [Your research situation]
Background: [Relevant metallurgy concepts]

Requirements:
1. [Specific functionality 1]
2. [Specific functionality 2]
3. Input validation with clear error messages
4. Publication-quality plotting
5. Extensive educational comments

Include:
- Example usage with realistic data
- Unit tests for verification
- Documentation header

I have [MATLAB with toolboxes / GNU Octave].
Make code suitable for graduate-level learning.
```

Listing 4: Code Generation Prompt Template

```
I'm getting this error in my [MATLAB/Octave] code:

ERROR MESSAGE:
[Paste complete error message]

CODE SECTION:
[Paste relevant lines with line numbers]
```

```
CONTEXT:
- Purpose: [What the code should do]
- Input data: [Describe input that caused error]
- Expected output: [What you expected]

Please:
1. Explain what's causing the error
2. Provide corrected code
3. Explain how to prevent this in future
4. Suggest any additional error checking
```

Listing 5: Debugging Prompt Template

A.2 Useful MATLAB/Octave Snippets

```
function [fs, T] = import_solidification_data(filename)
    % Import and validate solidification data from Excel

    % Check file exists
    if ~isfile(filename)
        error('File not found: %s', filename);
    end

    % Import data
    try
        data = readtable(filename);
        fs = data{:, 1};  % First column
        T = data{:, 2};   % Second column
    catch ME
        error('Error reading file: %s', ME.message);
    end

    % Validate data
    if length(fs) ~= length(T)
        error('fs and T must have same length');
    end

    if any(isnan(fs)) || any(isnan(T))
        warning('Data contains NaN values');
        % Remove NaN rows
        valid = ~isnan(fs) & ~isnan(T);
        fs = fs(valid);
        T = T(valid);
    end

    % Check physical bounds
    if any(fs < 0) || any(fs > 1)
        error('Solid fraction must be between 0 and 1');
    end

    fprintf('Successfully imported %d data points\n', length(fs));
end
```

Listing 6: Data import from Excel with error handling

```
function format_publication_plot(fig_handle)
    % Apply professional formatting to plots

    if nargin < 1
        fig_handle = gcf;
    end

    figure(fig_handle);

    % Font settings
    set(gca, 'FontSize', 12);
    set(gca, 'FontName', 'Arial');

    % Line width
    set(findall(gca, 'Type', 'Line'), 'LineWidth', 1.5);

    % Grid
    grid on;
    set(gca, 'GridLineStyle', '-');
    set(gca, 'GridAlpha', 0.3);

    % Box
    box on;

    % Tick direction
    set(gca, 'TickDir', 'out');

    % Figure size for publication
    set(fig_handle, 'Units', 'inches');
    set(fig_handle, 'Position', [1, 1, 6, 4.5]);

    % Background
    set(gca, 'Color', 'white');
    set(fig_handle, 'Color', 'white');
end
```

Listing 7: Professional plot formatting

A.3 Octave-Specific Compatibility Notes

Table 4: MATLAB vs Octave: Key Differences

Feature	MATLAB	Octave
Table import	`readtable()`	Use `csvread()` or `dlmread()`
String arrays	Full support	Use cell arrays instead
App Designer	Available	Not available
Live Scripts	.mlx files	Use .m scripts only
Toolbox functions	All available	Many missing, use alternatives
Plotting	Advanced features	Basic, but sufficient
Performance	Optimized	Slower for large data

When requesting Octave-compatible code:

```
"Generate Octave-compatible code (not MATLAB-specific).
Avoid: readtable, string arrays, toolbox functions.
Use: csvread, cell arrays, basic functions only."
```

B Troubleshooting Common Issues

B.1 LLM Access Issues

Problem: Cannot access Claude or DeepSeek
Solutions:

- Check university network restrictions
- Try different browser
- Use VPN if geo-restricted
- Contact IT support for institutional access

Problem: Hit message limits
Solutions:

- Switch between Claude and DeepSeek
- Wait for limit reset (usually 24 hours)
- Consider Pro account for heavy usage
- Use more efficient prompts (combine multiple questions)

B.2 MATLAB/Octave Issues

Problem: Code works in MATLAB but not Octave
Solution: Ask LLM to convert:

```
"This MATLAB code doesn't work in Octave:
[paste code]
Please convert to Octave-compatible version using only
core functions."
```

Problem: Slow performance with large datasets
Solution: Request optimization:

```
"This code is slow for large datasets. Please optimize:
[paste code]
Focus on vectorization and avoiding loops."
```

B.3 Data Analysis Issues

Problem: Results don't match published data
Troubleshooting Steps:

1. Verify units are consistent
2. Check if different databases were used
3. Confirm same compositions
4. Ask LLM: "Why might my results differ from [paper]?"

C Additional Resources

C.1 Video Tutorials

Create your own learning path by asking LLMs to explain videos:

```
"I watched Prof. Bhadeshia's lecture on [topic] at
[YouTube URL]. Can you:
1. Summarize the key concepts
2. Explain the equations he derived
3. Provide MATLAB code to implement his examples
4. Suggest related papers to read"
```

C.2 Recommended Reading Order

1. This lecture (Lecture 00) - LLM tools
2. Lecture 01 - Introduction to Computational Metallurgy
3. CALPHAD method papers (with Claude summaries)
4. Specific system studies (Cu-Zr, etc.)
5. Advanced applications

C.3 Contact Information

Prof. Fabio Miani
Department of Polytechnic Engineering and Architecture
University of Udine
Italy

Course Website: http://www.gotrawama.eu/

Questions: Use course forum or office hours

End of Lecture 00

Next: Lecture 01 - Introduction to Computational Metallurgy
