## Numerisches Lösen - Bisektion

1. **Vorgehensmodell**

Als Vorgehensmodell habe ich Kanban gewählt, da es eine klare visuelle Darstellung des Projektfortschritts ermöglicht und gleichzeitig hohe Flexibilität bietet. Das es besonders einfach zu implementieren ist kann es besonders in diesem kleinen Umfang schnell eingebaut werden.

1. **Programmier-Paradigma (Strukturiert, Objektorientiert)**

Als Programmier-Paradigma habe ich mich für das Objektorientierte Programmierung entschieden um den Code möglichst übersichtlich, lesbar und modularisierbar zu gestallten.

1. **Zeitmanagement**

|  |  |
| --- | --- |
| Erste Berechnungen in Excel, um Bisektion zu verstehen (Aufgabe 1 & 2) | 3 Stunden |
| PAP zu dem Bisektionsproblem erstellen (Aufgabe 3) | 1 Stunde |
| Planungsarbeiten (Aufgabe 4) | 2 Stunden |
| Programmierung sowie Tests in Python (Aufgabe 5) | 2.5 Stunden |
| Alternative Lösungsmöglichkeit (Aufgabe 6) | 3 Stunden |
| Diagramme mittels matplotlib (Aufgabe 7) | 1.5 Stunden |
| Polynom Berechnung (Aufgabe 8) | 1.5 Stunden |
| Elektrische Leitung Berechnung (Aufgabe 9) | 5 Stunden |
| **Summe** | **19.5 Stunden** |

1. **Work Breakdown Structure (WBS)**
   1. Proben und Verständnis
      1. Aufgabenstellung verstehen
      2. Handschriftliche Berechnungen
      3. Excel Berechnungen
   2. Planung
      1. Vorgehensmodell
      2. Programmier-Paradigma
      3. Zeitmanagement
      4. Work Breakdown Structure
      5. Gantt-Diagramm
      6. Ressourcenmanagement
      7. Risikomanagement
      8. Veröffentlichungsprüfung
   3. Implementierung
      1. Programmierung in Python (Bisektion)
      2. Programmierung in Python (Regula falsi oder Newton-Raphson)
      3. Matplotlib Programmierung
      4. Polynom Programmierung in Python
      5. Elektrische Leitungen Programmierung in Python
   4. Tests, Kontrollen
      1. Test und Vergleich Bisektion und Regula falsi oder Newton-Raphson
      2. Test und Kontrolle Polynom Berechnung
      3. Test und Kontrolle elektrische Leitungen Berechnung
   5. Finale Abgabe
      1. Zusammenfassung der Informationen
      2. Formatierung der einzelnen Dateien
      3. Abgabe des gesamt Packets
2. **Gantt-Diagramm**

| **Aufgabe** | **W1** | **W1** | | **W2** | **W2** | **W3** | **W3** | **W4** | **W4** | **W5** | | **W5** | **W6** | **W6** | **W7** | **W7** | **W8** | **W8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Proben und Verständnis** |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Aufgabenstellung verstehen |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Handschriftliche Berechnungen |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Excel Berechnungen |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **Planung** |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Vorgehensmodell |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Programmier-Paradigma |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Zeitmanagement |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Work Breakdown Structure |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Gantt-Diagramm |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Ressourcenmanagement |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Risikomanagement |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Veröffentlichungsprüfung |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **Implementierung** |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Programmierung in Python (Bisektion) |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Programmierung in Python (Regula falsi oder Newton-Raphson) |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Matplotlib Programmierung |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Polynom Programmierung in Python |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Elektrische Leitungen Programmierung |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **Tests, Kontrollen** |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Test und Vergleich Bisektion und Regula falsi oder Newton-Raphson |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Test und Kontrolle Polynom Berechnung |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Test und Kontrolle elektrische Leitungen |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **Finale Abgabe** |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Zusammenfassung der Informationen |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Formatierung der einzelnen Dateien |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Abgabe des Gesamtpakets |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **IST** | | | **SOLL** | | | | | | | | **Osterferien/Exkursion** | | | | | | | |

1. **Ressourcenmanagement**
   1. Personal:
      1. 1 Entwickler (Ich)
   2. Hardware
      1. Laptop
      2. Ladegerät
      3. Maus
      4. Papier
      5. Stift
      6. Taschenrechner
   3. Software
      1. Python
      2. Matplotlib
      3. Word
      4. Excel
      5. Google
2. **Risikomanagement**
   1. Unterschätzer Aufwand -> Pufferzeit einplanen
   2. Speicherverlust -> Cloud-Speicher / GitHub-Repo
   3. Hardwareprobleme -> Ersatzhardware und Cloud-Speicher
   4. Fehlerhafter Code -> Stack Overflow / Google / Professoren bei Fragen
3. **Veröffentlichungsprüfung**
   1. Matplotlib: Open-Source-Bibliothek -> Mit Python unter PSF-Lizenz
   2. Visual Studio Code (VSC): Open-Source-Editor unter MIT-Lizenz
      1. Verbreitung: Matplotlib und Python können frei verwendet, verändert und weitervergeben werden, solange die Lizenzbedingungen der Software eingehalten werden. Da VSC nicht mitgeliefert wird fällt die Lizenz hierbei weg.