

CES Softwareentwicklungspraktikum

Analyse- und Entwurfsdokument



Center for Computational Engineering Science
RWTH Aachen University



Stefan Jeske, Daniel Partida, Tom Witter und Chun-Kan Chow

Matr.-Nr. 334033, 335179, 333265, 333715

email:

[stefan.jeske|daniel.partida|tom.witter|chun-kan.chow]
@rwth-aachen.de

Inhaltsverzeichnis

1	Vorwort	5
1.1	Aufgabenstellung und Struktur des Dokuments	5
1.2	Projektmanagement	5
1.3	Lob und Kritik	6
2	Analyse	7
2.1	Anforderungsanalyse	7
2.1.1	Benutzeranforderungen	7
2.1.2	Anwendungsfallanalyse	8
2.1.3	Funktionale Anforderung	17
2.1.4	Nicht Funktionale Anforderung	18
2.2	Begriffsanalyse	19
3	Entwurf	21
3.1	Klassendiagramm	21
3.2	Dynamik	21
3.2.1	Sequenzdiagramme	21
4	Benutzerdokumentation	27
4.1	Installation	27
4.1.1	Voraussetzungen und Installation der kompilierten Version	27
4.1.2	Installation der unkompilierten Version	27
4.2	Speichern und Laden	28
4.2.1	.setup-Format	28
4.2.2	.surface-Format	29
4.2.3	Einstellungen laden	31
4.2.4	Ergebnis speichern	32
4.2.5	Ergebnis laden	33
4.2.6	Mit der .surface-Datei weiterarbeiten	34
4.3	Eingaben	36
4.4	Beispielsitzung	36
4.5	Fehlersituationen	36

5	Entwicklerdokumentation	37
5.1	Codestruktur	37
5.1.1	Klasse MainWindow	37
A	Quellcode	39
A.1	Paket 1	39
A.1.1	Klasse 1.1	39
A.1.2	Klasse 1.2	39
A.2	Paket 2	39
A.3	Paket 3	39

Kapitel 1

Vorwort

1.1 Aufgabenstellung und Struktur des Dokuments

Sehr geehrte Damen und Herren,

aufgrund der Notwendigkeit zur Reduzierung von Materialverbrauch, aber auch aus dem Wunsch heraus, ästhetisch ansprechende Produkte zu produzieren, benötigt unser Unternehmen eine Simulationssoftware zur Berechnung von Minimalflächen.

Ausschlaggebend für eine gute Integration der Software in unseren Arbeitsprozess ist, dass mithilfe einer graphischen Benutzerschnittstelle Randbedingungen und numerische Parameter eingestellt, sowie erstellte Konfigurationen als Datei gespeichert und auch wieder eingelesen werden können. Außerdem soll eine Visualisierung der resultierenden Minimalfläche möglich sein.

Wir freuen uns auf eine Zusammenarbeit.
Mit freundlichen Grüßen

(Jens Deussen und Uwe Naumann)

1.2 Projektmanagement

Tom Witter:

- Implementation Diskretisierung
- Implementation User Interface
- Benutzeranforderungen

Stefan Jeske:

- Implementation Controller
- Implementation User Interface
- Kompilierung auf dem Cluster

Daniel Partida:

- Implementation Fehlermeldungen im MainWindow
- Aktivitätsdisgramme

Chun-Kan Chow:

- Implementation der Funktionalitäten (Slots, Signale)
- Implemenation Controller/MainWindow (Laden, Speichern)
- Benutzerdokumentation

Alle:

- UC

1.3 Lob und Kritik

Danke Naumann für die Organisation des Projektes und die gewonnene Praxiserfahrung. ;-)

Kapitel 2

Analyse

2.1 Anforderungsanalyse

2.1.1 Benutzeranforderungen

Die Applikation ermöglicht es dem Benutzer, für bestimmte 3-D Konstruktionen die Minimalflächen zu berechnen und graphisch darzustellen.

Die Applikation gibt nach Starten zunächst beispielhafte Einstellungen und Randwerte vor.

Der Benutzer kann per Mausklick auf der Benutzeroberfläche zwischen 3 verschiedenen Tabs wechseln.

Auf dem ersten Tab („Einstellungen“), das nach Starten des Programmes angezeigt wird, kann der Benutzer die verschiedenen numerischen Parameter (Anzahl der Stützstellen in X- und Y Richtung, Abbruchfehler, maximale Iterationsanzahl, Randfunktionen, Definitionsbereich) in den dafür zuständigen Textfeldern ändern.

In den mit „Grenzen“ markierten Textfeldern kann der Benutzer Randfunktionen setzen, indem er Funktionen eingibt, für die die Minimalflächen berechnet werden sollen. Weiterhin kann der Benutzer den Definitionsbereich der Randfunktionen einstellen in den Textfeldern markiert mit „Definitionsbereich“.

Unter „Numerische Parameter“ kann der Benutzer die Parameter für die Approximation ändern, die Fehlergrenze, die Anzahl der Stützstellen in x- und y-Richtung und die maximale Iterationszahl. Die maximale Iterationszahl ist die Anzahl an Iterationen, die die Applikation für die Approximation verwendet, falls die Fehlergrenze nicht vorher unterschritten wird.

Der Benutzer kann zu jeder Zeit mit dem Button „Quit“ das Programm beenden.

Ausserdem kann der Benutzer auch mit einem Mausklick auf „Einstellungen Speichern“ die gewählten Einstellungen in einem Dokument speichern und mit einem Klick auf „Einstellungen Laden“ kann er bereits gespeicherte Einstellungen wiederherstellen.

Wenn der Nutzer fertig mit den Einstellungen ist, kann er auf „Run“ drücken,

um die Simulation zu starten.

Die Applikation checkt die Einstellungen und zeigt im Fehlerfall eine Fehlermeldung an und ansonsten wird die Berechnung gestartet.

Geschieht dies, wechselt die Applikation automatisch auf das zweite Tab „Konsoleausgabe“. Weiterhin zeigt die Applikation die Ausgabe der Anwendung an, den Fortschritt, die Iterationszahl und den aktuellen Fehler.

Der Fortschritt ist außerdem auf einem Fortschrittsbalken zu sehen.

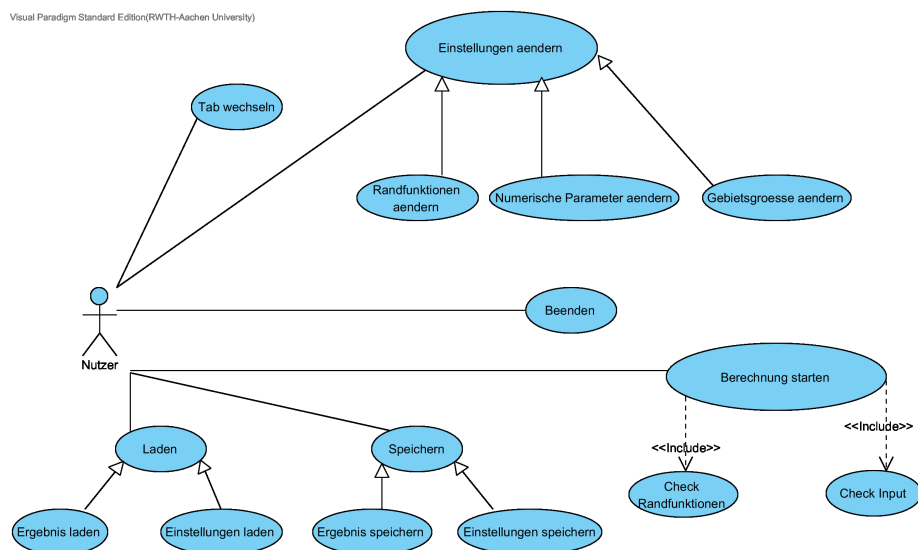
Ist die Berechnung beendet, wechselt die Applikation automatisch auf das dritte Tab „Anzeige“, wo der Benutzer die fertige 3-D Simulation auf der Zeichenfläche sehen und mithilfe der Maus beliebig drehen kann.

Der Benutzer kann dabei jederzeit die Tabs wechseln, auch bevor die Berechnung beendet ist.

Wechselt der Benutzer auf das Anzeigetab während die Berechnung läuft, dann zeigt die Applikation den derzeitigen Rechenschritt, d.h. den instationären Berechnungsverlauf an. Der Benutzer kann auch schon berechnete Minimalflächen (Ergebnisse) speichern oder laden mit einem Mausklick auf den zugehörigen Button.

2.1.2 Anwendungsfallanalyse

Visual Paradigm Standard Edition (RWTH-Aachen University)

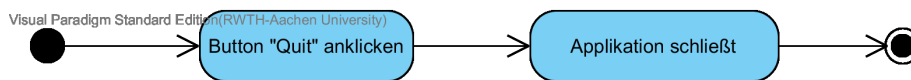


Systemanforderungen

Beenden

- *Ziel:* Der Nutzer will das Programm beenden.
- *Einordnung:* Hauptfunktion

- *Vorbedingung:* Die Applikation wurde gestartet.
- *Nachbedingung:* Das Programm ist geschlossen.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:*
- *Auslöser:* Der Nutzer möchte das Programm beenden.
- *Standardablauf:*
 1. Der Nutzer klickt auf den Button "Quit".
 2. Die Applikation schließt.



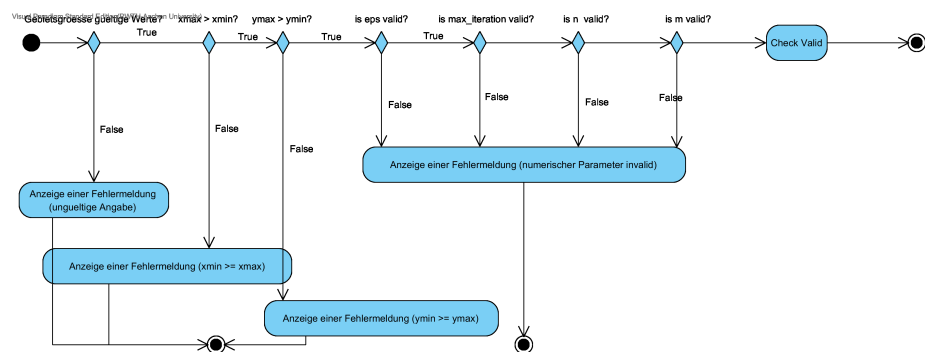
Check Input

- *Ziel:* Die Applikation will die Gebietsgröße und die numerischen Parameter überprüfen.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Der Nutzer hat auf Run angeklickt.
- *Nachbedingung:* Der Use Case „Check Randfunktionen“ wird gestartet.
- *Nachbedingung im Fehlerfall:* Die Applikation zeigt eine Fehlermeldung an und markiert die Stelle, an der sich der Fehler befindet.
- *Hauptakteure:* System
- *Nebenakteure:*
- *Auslöser:* Die Applikation möchte die Gebietsgröße und die numerischen Parameter überprüfen.
- *Standardablauf:*
 1. Die Applikation prüft, ob die eingegebenen Grenzen der Gebietsgröße gültige double Werte sind.
 2. Die Applikation prüft, ob die obere Grenze xmax größer als die untere Grenze xmin ist.
 3. Die Applikation prüft, ob die obere Grenze ymax größer als die untere Grenze ymin ist.
 4. Die Applikation prüft, ob der Abbruchfehler eps größer 0 und ein double ist.

5. Die Applikation prüft, ob die maximale Anzahl der Iterationen $max_iteration$, die Anzahl der x-Stützstellen n und y-Stützstellen m größer 0 und vom Typ `int` sind.

• *Verzweigungen:*

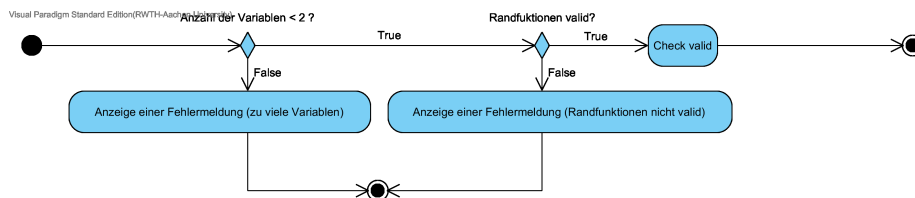
- (1a1) Die Berechnung wird abgebrochen.
- (1a2) Die Applikation zeigt eine Fehlermeldung, falls die Gebietsgröße ungültig ist. (keine Zahl eingegeben oder das Feld leer gelassen)
- (1a3) Das Feld, in dem der Fehler ist, wird markiert.
- (2a1) Die Berechnung wird abgebrochen.
- (2a2) Die Applikation zeigt eine Fehlermeldung, falls $xmin$ größer gleich $xmax$ ist.
- (2a3) Das Feld, in dem der Fehler ist, wird markiert.
- (3a1) Die Berechnung wird abgebrochen.
- (3a2) Die Applikation zeigt eine Fehlermeldung, falls $ymin$ größer gleich $ymax$ ist.
- (3a3) Das Feld, in dem der Fehler ist, wird markiert.
- (4a1) Die Berechnung wird abgebrochen.
- (4a2) Die Applikation zeigt eine Fehlermeldung, falls eps ungültig ist.
- (4a3) Das Feld „eps edit“ wird markiert.
- (5a1) Die Berechnung wird abgebrochen.
- (5a2) Die Applikation zeigt eine Fehlermeldung, falls ein numerischer Parameter ungültig ist.
- (5a3) Das Feld, in dem der Fehler ist, wird markiert.



Check Randfunktionen

- *Ziel:* Das System will die Randfunktionen überprüfen.

- *Einordnung*: Hauptfunktion
- *Vorbedingung*: Use Case Check Input wurde erfolgreich durchgeführt.
- *Nachbedingung*: Die Berechnung wird durchgeführt.
- *Nachbedingung im Fehlerfall*: Die Applikation zeigt eine Fehlermeldung an und markiert die Stelle, an der sich der Fehler befindet.
- *Hauptakteure*: System
- *Nebenakteure*:
- *Auslöser*: Der Nutzer hat auf 'run' gedrückt und UC Check Input war erfolgreich.
- *Standardablauf*:
 1. Das System prüft, ob weniger als 2 Variablen vorhanden sind.
 2. Das System prüft, ob die Randfunktionen gültig sind.
- *Verzweigungen*:
 - (1a1) Das System zeigt eine Fehlermeldung, falls mehr als 2 Variablen vorhanden sind.
 - (1a2) Die Berechnung wird abgebrochen.
 - (2a1) Das System zeigt eine Fehlermeldung, falls die Randfunktion ungültig ist.
 - (2a2) Die Berechnung wird abgebrochen.



Einstellungen laden

- *Ziel*: Der Nutzer will vorher gespeicherte Einstellungen im .setup Format laden.
- *Einordnung*: Hauptfunktion
- *Vorbedingung*: Die Applikation wurde gestartet
- *Nachbedingung*: Das System lädt eine Datei mit den gespeicherten Daten und das MainWindow wird angezeigt.
- *Nachbedingung im Fehlerfall*: Nichts wird verändert.

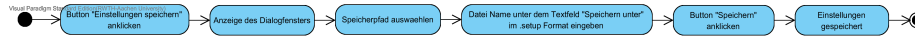
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte eine vorher eingestellte gespeicherte Einstellung im .setup Format laden.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf dem Button „Einstellungen laden“ an.
 2. Das System zeigt das Dialogfenster an.
 3. Der Nutzer sucht und wählt einen Ladepfad aus.
 4. Der Nutzer markiert die Datei im .setup Format.
 5. Der Nutzer klickt auf den Button „Öffnen“.
 6. Das System zeigt die gespeicherten Daten im Anzeige Tab an.
- *Verzweigungen:*
 - (31) Es befindet sich keine vorher gespeicherte Einstellungsdatei.



Einstellungen speichern

- *Ziel:* Der Nutzer will die eingegebenen Einstellungen speichern.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet.
- *Nachbedingung:* Das System erstellt eine .setup Datei mit den gespeicherten Daten und das MainWindow wird angezeigt.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte die eingegebenen Einstellungen speichern.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf „Einstellungen speichern“.
 2. Das System zeigt das Dialogfenster an.
 3. Der Nutzer wählt den Speicherpfad aus.
 4. Der Nutzer gibt über die Tastatur den Dateinamen ein.
 5. Der Nutzer klickt auf den Button „Speichern“.

6. Das System speichert die eingegebenen Einstellungen im angegebenen Verzeichnis als .setup Datei



Ergebnis laden

- *Ziel:* Der Nutzer will eine vorher gespeicherte Ergebnisdatei im .surface Format laden.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet
- *Nachbedingung:* Das System lädt die gespeicherten Ergebnisdaten und zeigt sie im Anzeige Tab an.
- *Nachbedingung im Fehlerfall:* Es wird nichts verändert.
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte eine vorher gespeicherte Ergebnisdatei im .surface Format laden.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf den Button „Ergebnis laden“.
 2. Das System zeigt das Dialogfenster an.
 3. Der Nutzer sucht und wählt den Ladepfad aus.
 4. Der Nutzer markiert mit der linken Maustaste die .surface Datei.
 5. Der Nutzer klickt mit der linken Maustaste auf den Button Öffnen.
 6. Das System zeigt die Ergebnisdaten im Anzeige Tab an.

- *Verzweigungen:*

(31) Es befindet sich keine vorher gespeicherte Einstellungsdatei.



Ergebnis speichern

- *Ziel:* Der Nutzer will die Ergebnisdaten speichern.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet.
- *Nachbedingung:* Das System erstellt eine .surface Datei mit den gespeicherten Daten, den numerischen Parametern und der Gebietsgröße. Das MainWindow wird angezeigt.

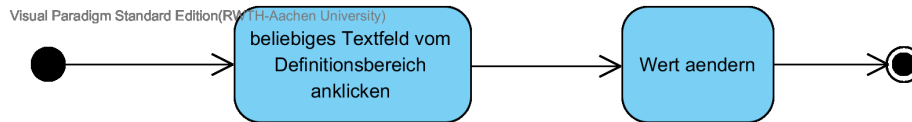
- *Nachbedingung im Fehlerfall:* Das System zeigt eine Fehlermeldung, weil noch keine Berechnung durchgeführt wurde.
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte die Ergebnisdaten speichern.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste Ergebnis speichern an.
 2. Das System prüft, ob eine Berechnung schon durchgeführt wurde.
 3. Das System zeigt das Dialogfenster an.
 4. Der Nutzer sucht und wählt einen Pfad aus.
 5. Der Nutzer gibt über die Tastatur den Dateinamen ein.
 6. Der Nutzer klickt auf den Button "Speichern".
 7. Das System speichert die Ergebnisdaten als .surface Datei
- *Verzweigungen:*
 - (21) Das System zeigt eine Fehlermeldung an, da zuvor keine Berechnung durchgeführt wurde.
 - (22) Der UC Ergebnis speichern wird beendet.



Gebietsgröße Ändern

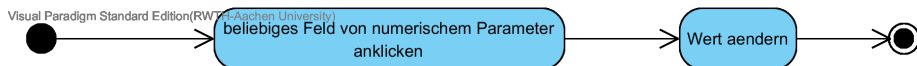
- *Ziel:* Der Nutzer will einen Wert der Gebietsgröße ändern.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet und der Tab 'Einstellungen' im MainWindow wird angezeigt.
- *Nachbedingung:* Die Gebietsgröße wurde verändert.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte einen Wert der Gebietsgröße ändern.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf beliebiges Feld der Gebietsgröße.

2. Der Nutzer gibt über die Tastatur den neuen Wert der Gebietsgröße ein.



Numerische Parameter Ändern

- *Ziel:* Der Nutzer will einen numerischen Parameter ändern.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet und der Tab 'Setting' im MainWindow wird angezeigt.
- *Nachbedingung:* Der numerische Parameter wurde verändert.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte einen numerischen Parameter ändern.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf beliebiges Feld der numerischen Parameter.
 2. Der Nutzer gibt über die Tastatur den neuen numerischen Wert ein.

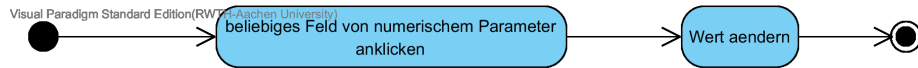


Randfunktionen Ändern

- *Ziel:* Der Nutzer will eine Randfunktion ändern.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet und der Tab 'Setting' im MainWindow wird angezeigt.
- *Nachbedingung:* Die Randfunktion wurde verändert.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte eine Randfunktion ändern.

- *Standardablauf:*

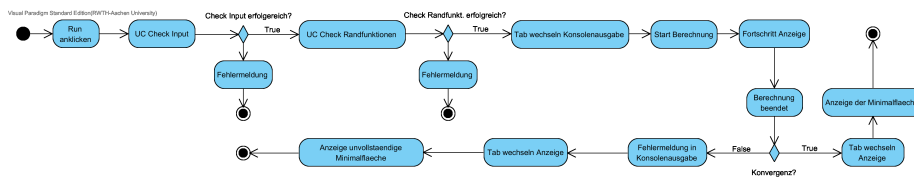
1. Der Nutzer klickt mit der linken Maustaste auf beliebiges Feld der Randfunktionen an.
2. Der Nutzer gibt über die Tastatur die neue Randfunktion ein.



run

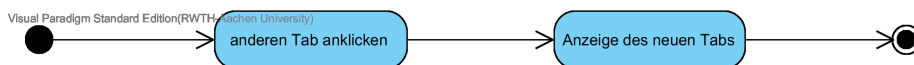
- *Ziel:* Der Nutzer will die Berechnung der Minimalfläche durchführen und anzeigen.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde gestartet, alle numerische Parameter, Randfunktionen und Gebietsgröße sind eingegeben.
- *Nachbedingung:* Die Applikation zeigt die berechneten Minimalflächen im Tab „Anzeige“ an.
- *Nachbedingung im Fehlerfall:* Es wird eine Fehlermeldung angezeigt.
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer will die Berechnung starten.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf den Button 'Run'.
 2. UC „Check Input“
 3. UC „Check Randfunktionen“
 4. Die Applikation wechselt den Tab auf „Konsolenausgabe“.
 5. Die Applikation startet mit den gegebenen Einstellungen die Berechnung der Minimalflächen.
 6. Die Applikation zeigt den Fortschritt der Berechnung auf der Benutzeroberfläche an.
 7. Die Applikation schliesst die Berechnung ab.
 8. Die Applikation wechselt den Tab auf „Anzeige“
 9. Die Applikation zeigt die berechnete Minimalfläche an.
- *Verzweigungen:*
 - (2a1) Der UC „Check Input“ ist fehlerhaft, die Applikation zeigt eine Fehlermeldung an und stoppt die Berechnung.

- (3a1) Der UC „Check Randfunktionen“ ist fehlerhaft, die Applikation zeigt eine Fehlermeldung an und stoppt die Berechnung.
- (7a1) Die Berechnung ist nicht konvergiert, die Applikation zeigt auf der „Konsolenausgabe“ eine Fehlermeldung an.
- (7a2) Die Applikation wechselt den Tab auf „Anzeige“
- (7a3) Die Applikation zeigt die nicht vollständig berechnete Minimalfläche an.



Tab wechseln

- *Ziel:* Der Nutzer will zu einem anderen Tab wechseln.
- *Einordnung:* Hauptfunktion
- *Vorbedingung:* Die Applikation wurde erfolgreich gestartet.
- *Nachbedingung:* Die Applikation zeigt den neuen Tab an.
- *Nachbedingung im Fehlerfall:*
- *Hauptakteure:* Nutzer
- *Nebenakteure:* System
- *Auslöser:* Der Nutzer möchte auf einen anderen Tab wechseln.
- *Standardablauf:*
 1. Der Nutzer klickt mit der linken Maustaste auf den gewünschten Tab.
 2. Die Applikation wechselt den Tab und zeigt den ausgewählten Tab an.



2.1.3 Funktionale Anforderung

- Anzeigen einer Benutzeroberfläche
- Speichern der Einstellungen
- Laden der Einstellungen
- Speichern der Ergebnisse

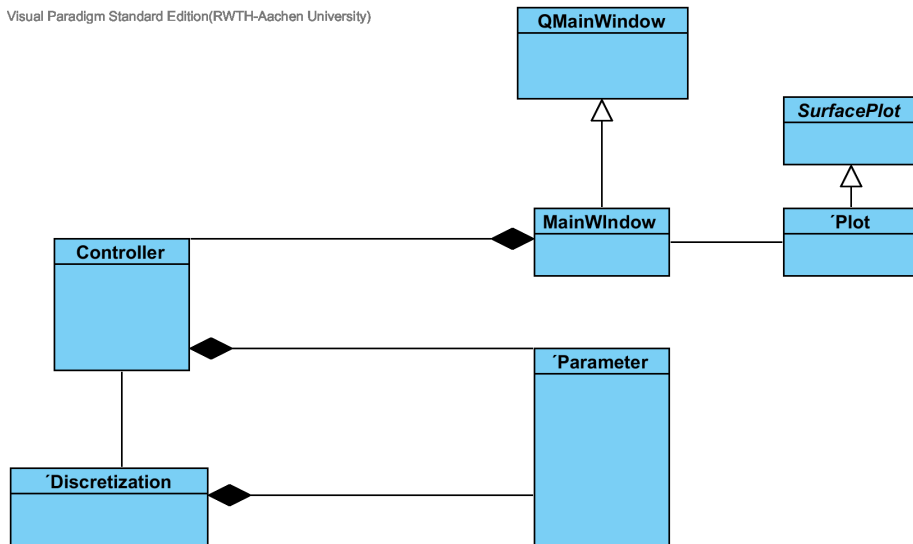
- Laden der Ergebnisse
- Möglicher Wechsel zwischen drei Tabs. Funktionen des ersten Tabs (Einstellungen):
 1. Eingabe der Anzahl der Stützstellen in X- und Y-Richtung
 2. Eingabe der Randfunktionen
 3. Eingabe des Abbruchfehlers
 4. Eingabe der Anzahl der maximalen Iterationen
 5. Änderung des Gebietes
 6. Run Button
 7. Quit Button
- Funktionen des zweiten Tabs (Konsolenausgabe):
 1. Anzeigen des Fortschrittes
 2. Anzeigen von Fehlermeldungen
- Funktionen des dritten Tabs (Anzeige):
 1. Graphische Darstellung der berechneten Minimalflächen
 2. Plot: variabler viewpoint
 3. Instationäre Anzeige des derzeitigen Ergebnisses während der Berechnung
- Test auf Gültigkeit des Gebietsbereiches
- Test auf ganze Zahl der Anzahl der Stützstellen.

2.1.4 Nicht Funktionale Anforderung

- Eingabe der Zahlen durch Tastatur
- Skalierung des Hauptfensters bei unterschiedlichen Auflösungen.
- Skalierung des Hauptfensters bei unterschiedlichen Auflösungen
- Erzeugung von korrekten Ergebnissen
- Intuitive Bedienbarkeit
- Plattformunabhängig
- Effiziente Berechnung

2.2 Begriffsanalyse

Visual Paradigm Standard Edition(RWTH-Aachen University)



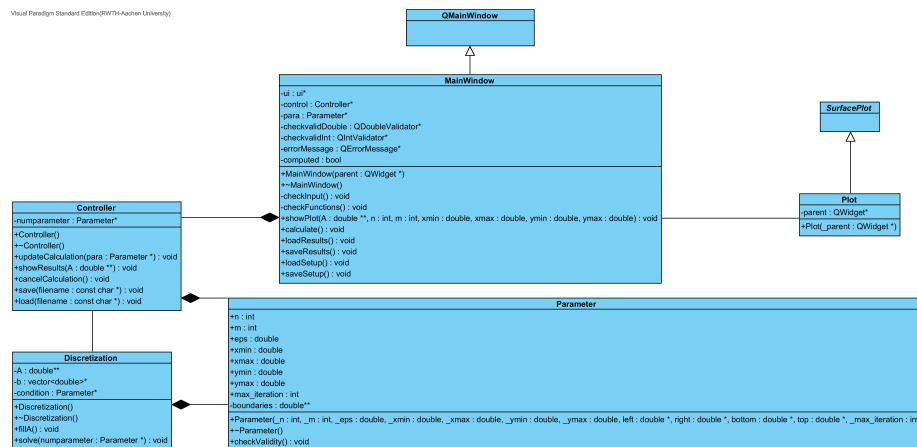
Klassenkadidaten

- Diskretisierung
- Eingabebereich
- Ausgabebereich
- Main Window
- Textfeld für Eingabe
- Button
- Radiobutton
- Dialogfenster
- Fehlermeldung(durch main implementiert)
- Berechnungsparameter
- Randwert
- Gebietsgröße

Kapitel 3

Entwurf

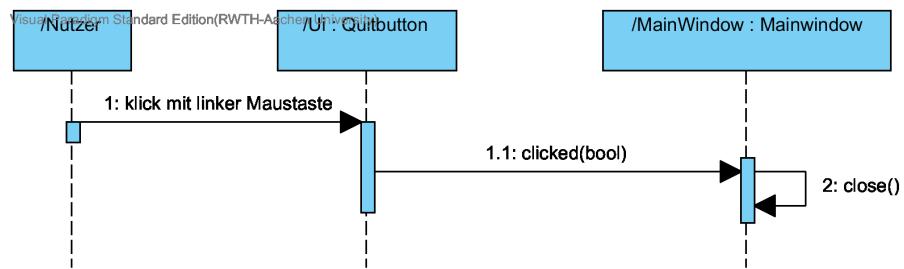
3.1 Klassendiagramm



3.2 Dynamik

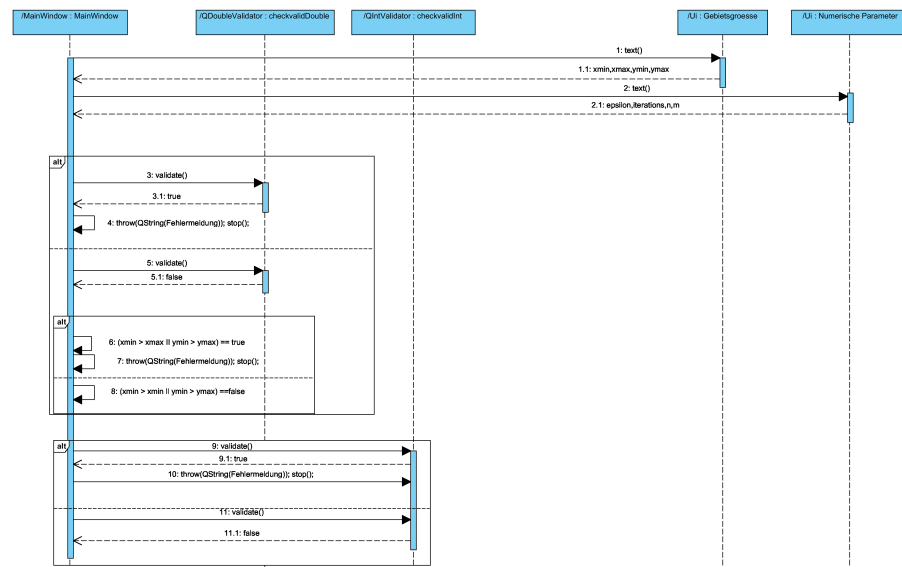
3.2.1 Sequenzdiagramme

Beenden

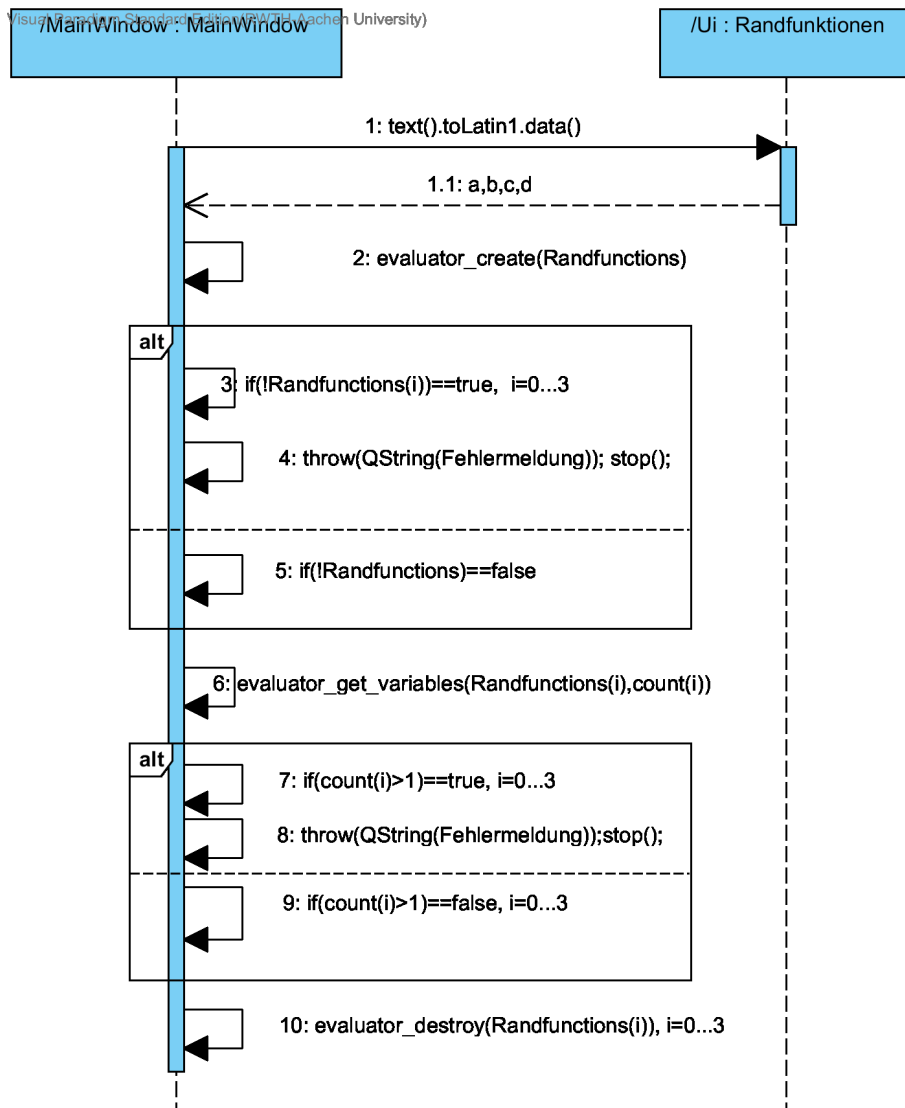


Check Input

UML Sequence Diagram illustrating the input validation process.



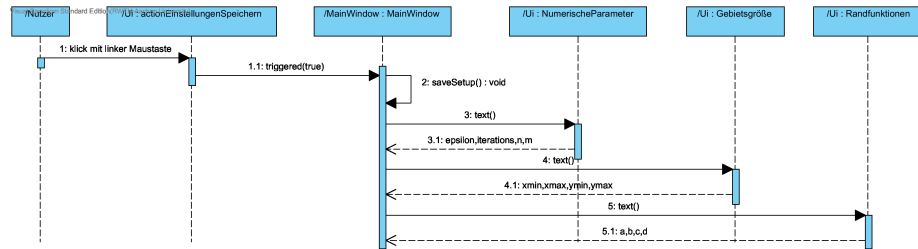
Check Randfunktionen



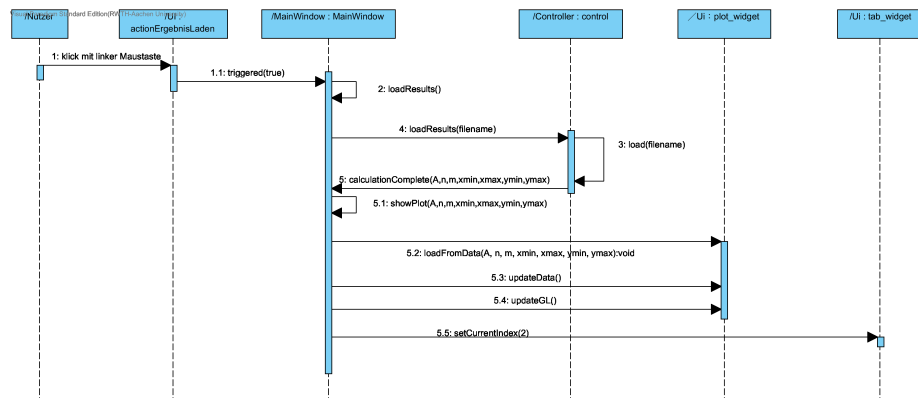
Einstellungen laden



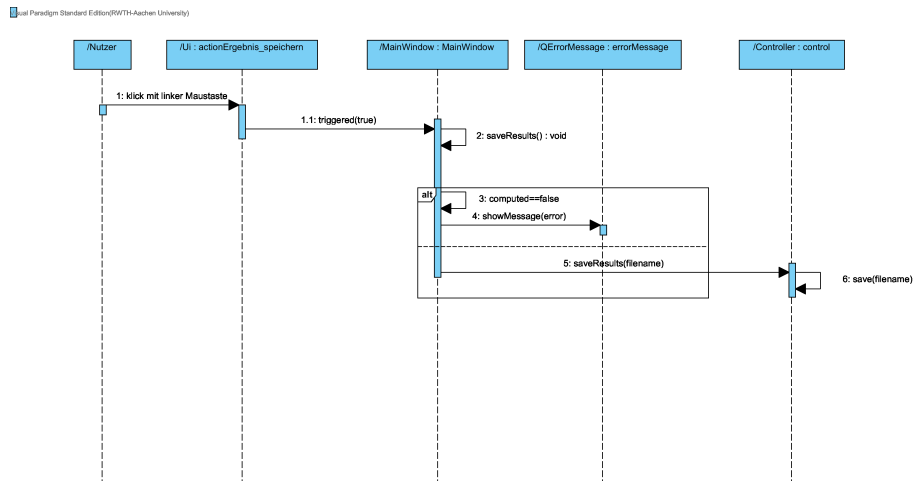
Einstellungen speichern



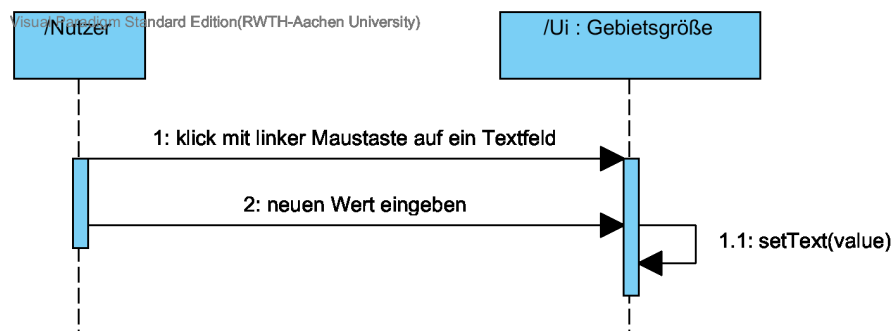
Ergebnis laden



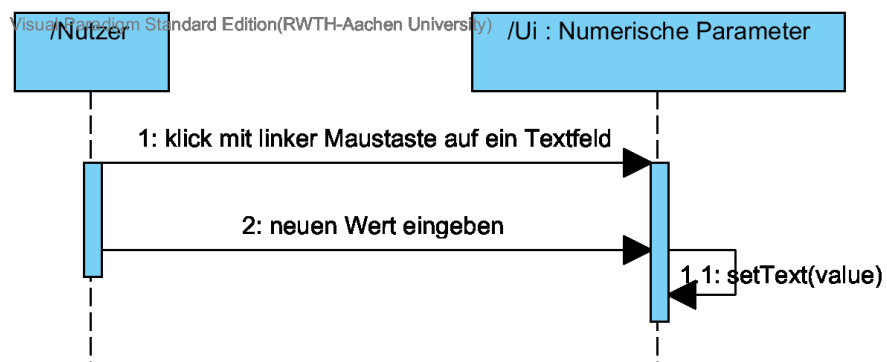
Ergebnis speichern



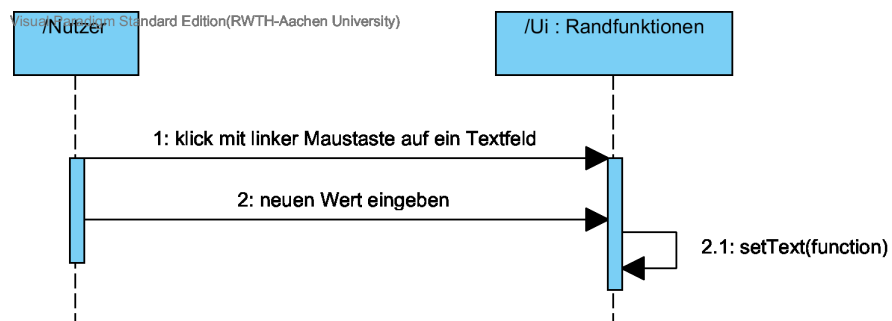
Gebietsgröße ändern



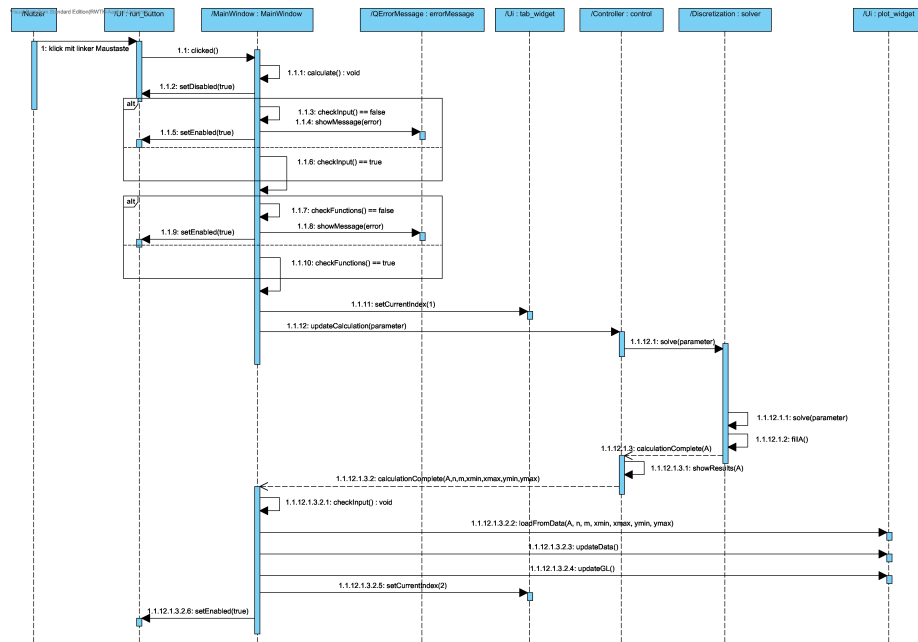
Numerische Parameter ändern



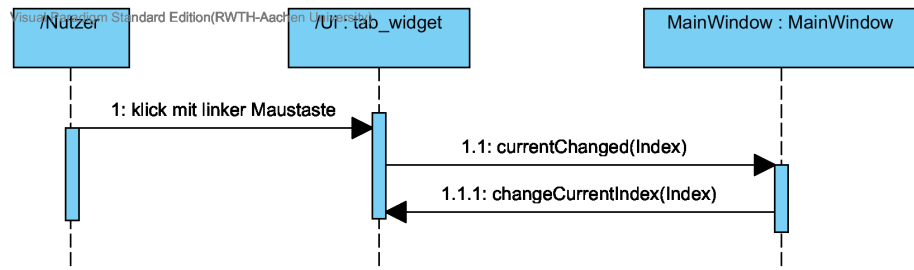
Randfunktionen ändern



Run



Tab wechseln



Kapitel 4

Benutzerdokumentation

4.1 Installation

4.1.1 Voraussetzungen und Installation der kompilierten Version

Für die Installation der Software ist ein Linux basiertes Betriebssystem notwendig. Die folgenden Schritte beschreiben den Installationsprozess für die bereits kompilierte Version. Diese Version wurde auf dem RWTH-Cluster sowie Ubuntu 14.04.3 LTS erfolgreich getestet.

1. Entpacken des Archivs `minimalflaechen_projekt_gruppe2.zip` in den gewünschten Installationsort.
2. Wechseln in das Verzeichnis `minimalflaechen_projekt_build`.
3. Ausführen des Skripts `minimalflaechen.sh` öffnet das Programm.

Hinweis: Um das Skript ausführen zu können müssen gegebenenfalls Rechte für dieses mithilfe des Befehls `chmod -x ./minimalflaechen.sh` gesetzt werden.

4.1.2 Installation der unkompilierten Version

Die unkompilierte Version des Programms findet sich in dem Verzeichnis `minimalflaechen_projekt_source`. Öffnen und konfigurieren des Projekts erfordert das Programm Qt Creator sowie die Installation der QT Bibliotheken. Zusätzlich dazu müssen die Bibliotheken `qwtplot3d` und `matheval` kompiliert und in den Ordner `libs` in dem Projektverzeichnis kopiert werden. Der Source-Code der Bibliotheken befindet sich in dem Verzeichnis `ThirdParty`. Die Kompilierung dieser Bibliotheken ist unterschiedlich für verschiedene Betriebssysteme und erfordert ggf. weitere Bibliotheken. Bei Fragen bitte an `stefan.jeske@rwth-aachen.de` wenden.

wohlstrukturierte und gut lesbare Dokumentation basierend auf den Anwendungsfällen

4.2 Speichern und Laden

Die Applikation verfügt über eine Speicher- und Lade Funktionalität. Die Einstellungen (Randfunktionen, numerische Parameter und Gebietsgröße) kann man in einem **.setup** Format und die berechnete Minimalfläche kann man in einem **.surface** Format speichern und laden

4.2.1 .setup-Format

Das Speichern und Laden der Einstellungen arbeiten mit Dateien im **.setup**-Format. Die Datei ist so aufgebaut, dass sich in jeder Zeile Informationen zu den eingegebenen Werten/Funktionen befinden, welche mit einem Semikolon am Ende abgetrennt werden. Dabei spielt die Reihenfolge der Informationen eine wichtige Rolle. Die Informationen von der ersten bis zur letzten Zeile:

1. epsilon - Genauigkeit der Lösung
2. iterations - Maximale Iterationsschritte
3. n - Anzahl der Stützstellen in x-Richtung
4. m - Anzahl der Stützstellen in y-Richtung
5. xmin - untere x-Grenze
6. xmax - obere x-Grenze
7. ymin - untere y-Grenze
8. ymax - obere y-Grenze
9. a - Randfunktion auf der linken Seite des Gebietes
10. b - Randfunktion auf der oberen Seite des Gebietes
11. c - Randfunktion auf der rechten Seite des Gebietes
12. d - Randfunktion auf der unteren Seite des Gebietes

```

1 1e-09;
2 20;
3 30;
4 30;
5 -0.5;
6 0.5;
7 -0.5;
8 0.5;
9 1;
10 cos(x);
11 1;
12 x;
13

```

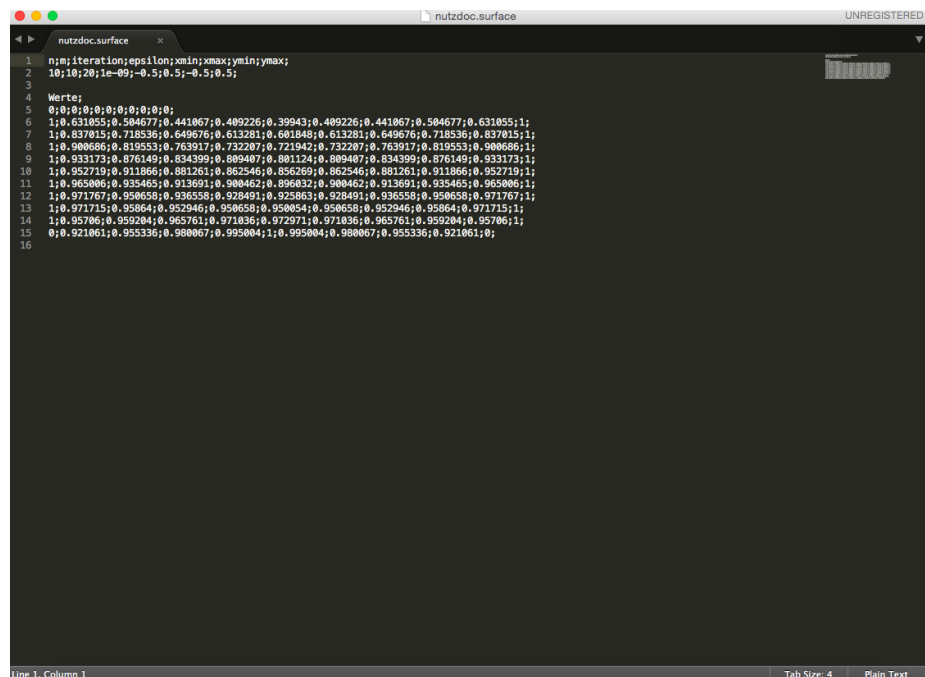
4.2.2 .surface-Format

Das Speichern und Laden der Einstellungen arbeiten mit Dateien im **.surface**-Format. Eine **.surface**-Datei ist in zwei Abschnitten aufgebaut. Im ersten Abschnitt enthält die Datei ein paar grundlegende Informationen zu den Eingabedaten der Berechnung:

1. n - Anzahl der Stützstellen in x-Richtung
2. m - Anzahl der Stützstellen in y-Richtung
3. iterations - Maximale Iterationsschritte
4. epsilon - Genauigkeit der Lösung
5. xmin - untere x-Grenze
6. xmax - obere x-Grenze
7. ymin - untere y-Grenze
8. ymax - obere y-Grenze

Der zweite Abschnitt enthält die berechnete Minimalfläche inklusive den ausgewerteten Rändern. Jeder Wert wird mit einem Semikolon getrennt und insgesamt stellen die Zahlen die Einträge von der Ergebnismatrix dar. (Lösung von $Ax=b$, mit x als eine Matrix transformiert)

Hinweis: Die Lösung der Minimalfläche ist so gespeichert, dass man die Daten mit Programmen wie Excel oder Matlab weiterarbeiten kann. Die Semikolons dienen zur Trennung zwischen den einzelnen Daten, wie sie bei Excel z.B. mit unterschiedlichen Zellen realisiert werden.

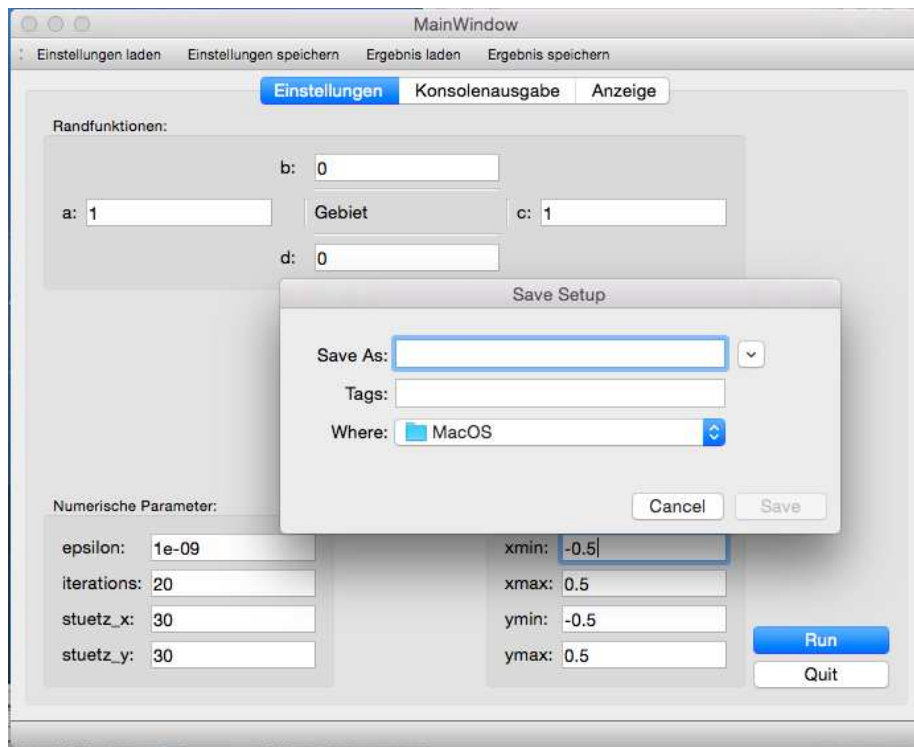


```

1 n; iteration; epsilon; xmin; xmax; ymin; ymax;
2 10; 10; 20; 1e-09; -0.5; 0.5; -0.5; 0.5;
3
4 Werte;
5 0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
6 1;0.631855;0.594677;0.441867;0.489226;0.39943;0.489226;0.441867;0.594677;0.631855;1;
7 1;0.837815;0.718536;0.649676;0.613281;0.601848;0.613281;0.649676;0.718536;0.837815;1;
8 1;0.980686;0.819553;0.763917;0.732287;0.721942;0.732287;0.763917;0.819553;0.980686;1;
9 1;0.933173;0.876149;0.834399;0.809407;0.881124;0.809407;0.834399;0.876149;0.933173;1;
10 1;0.952719;0.911866;0.881261;0.862546;0.856269;0.862546;0.881261;0.911866;0.952719;1;
11 1;0.965086;0.935465;0.913691;0.900462;0.896832;0.900462;0.913691;0.935465;0.965086;1;
12 1;0.971767;0.950658;0.936558;0.928491;0.925863;0.928491;0.936558;0.950658;0.971767;1;
13 1;0.971715;0.95864;0.952946;0.950658;0.950854;0.950658;0.952946;0.95864;0.971715;1;
14 1;0.95786;0.959284;0.965761;0.971836;0.972971;0.971836;0.965761;0.959284;0.95786;1;
15 0;0.921861;0.955336;0.980067;0.995004;1;0.995004;0.980067;0.955336;0.921861;0;
16

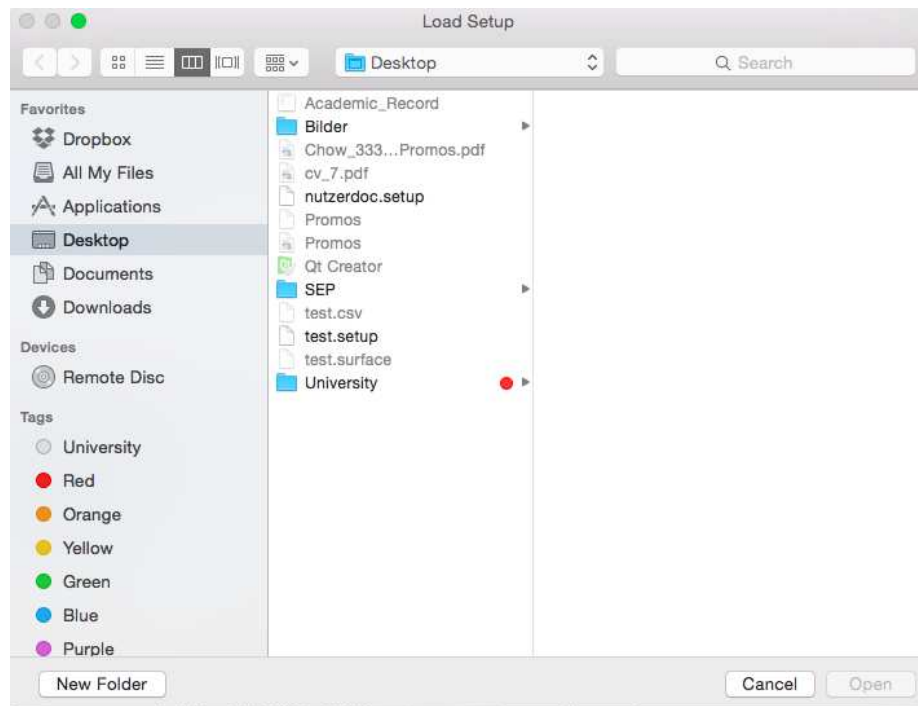
```

Nachdem der Nutzer den Button “Einstellungen speichern” geklickt hat, erscheint ein Dialogfenster, wo der Nutzer den Speicherpfad, sowie den Dateinamen eingeben kann. Nach der Betätigung des Save Buttons wird dann die Datei im **filename.setup** erstellt.



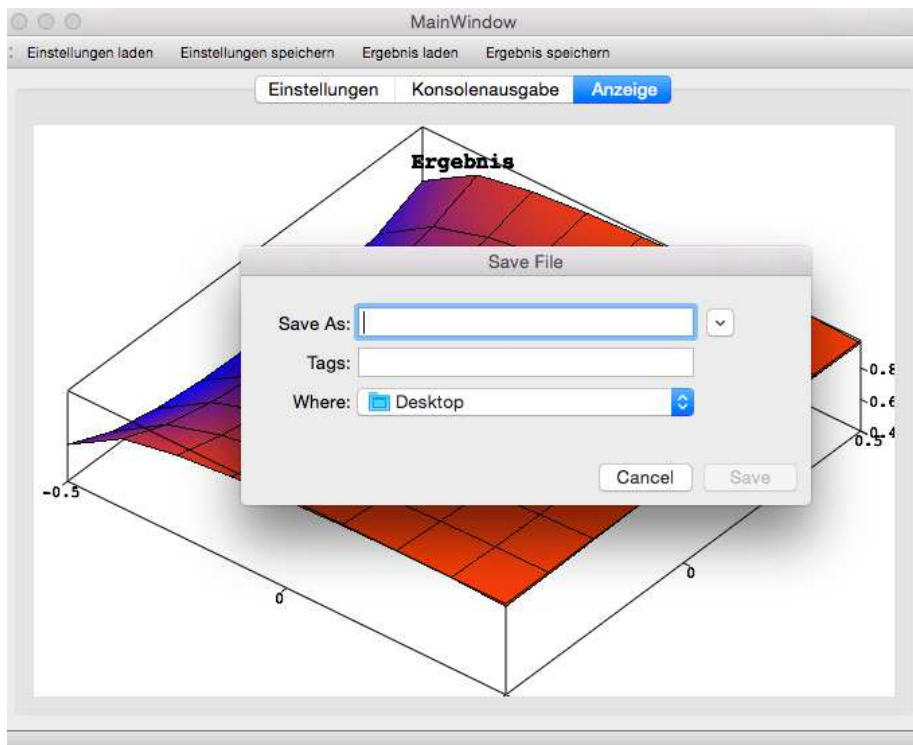
4.2.3 Einstellungen laden

Nachdem der Nutzer den Button “Einstellungen laden” geklickt hat, erscheint ein Dialogfenster, wo der Nutzer das Verzeichnis suchen kann, wo sich die Datei im **.setup**-Format befindet. Durch anschließendes Klicken auf den Open-Button werden die Randfunktionen, numerischen Parameter sowie die Gebietsgröße im Tab Einstellungen aktualisiert.



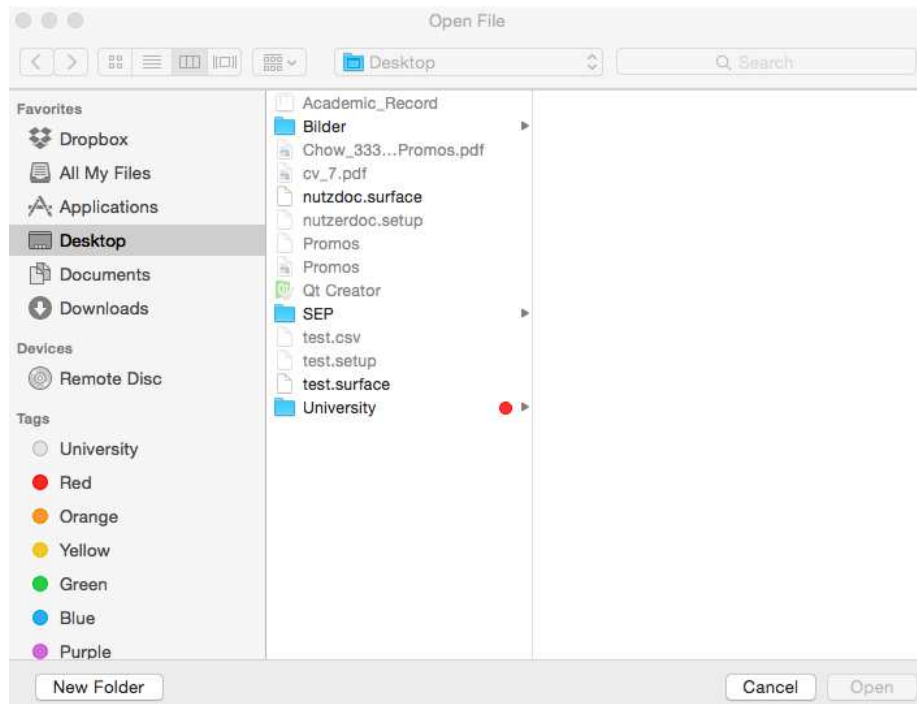
4.2.4 Ergebnis speichern

Nachdem der Nutzer den Button “Ergebnis speichern” geklickt hat, erscheint ein Dialogfenster, wo der Nutzer den Speicherpfad, sowie den Dateinamen eingeben kann. Nach der Betätigung des Savebuttons wird dann die Datei im **filename.surface** erstellt. Hinweis: Speicherung eines Ergebnisses erst nach der Berechnung möglich.



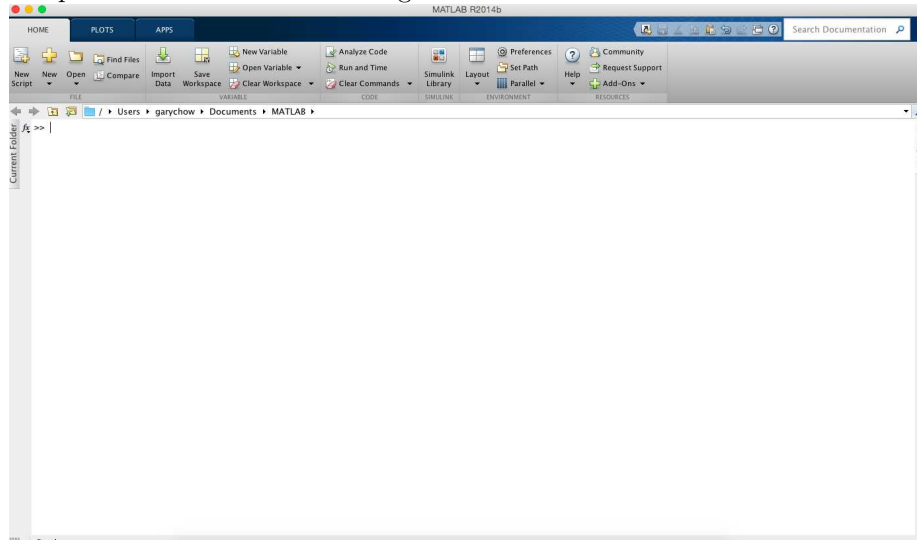
4.2.5 Ergebnis laden

Nachdem der Nutzer den Button “Ergebnis laden” geklickt hat, erscheint ein Dialogfenster, wo der Nutzer das Verzeichnis suchen kann, wo sich die Datei im **.surface**-Format befindet. Durch anschließendes Klicken auf den Open-Button werden die Ergebnisse geladen und die Minimalfläche angezeigt.

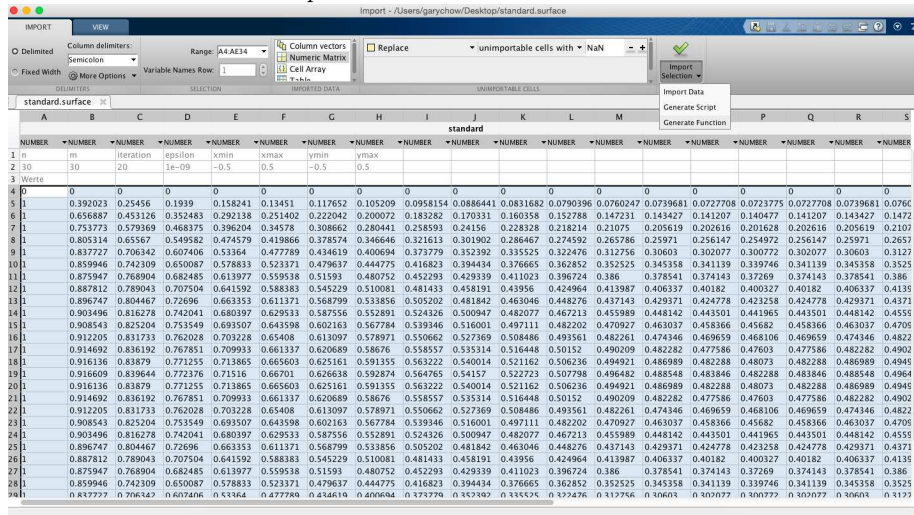


4.2.6 Mit der .surface-Datei weiterarbeiten

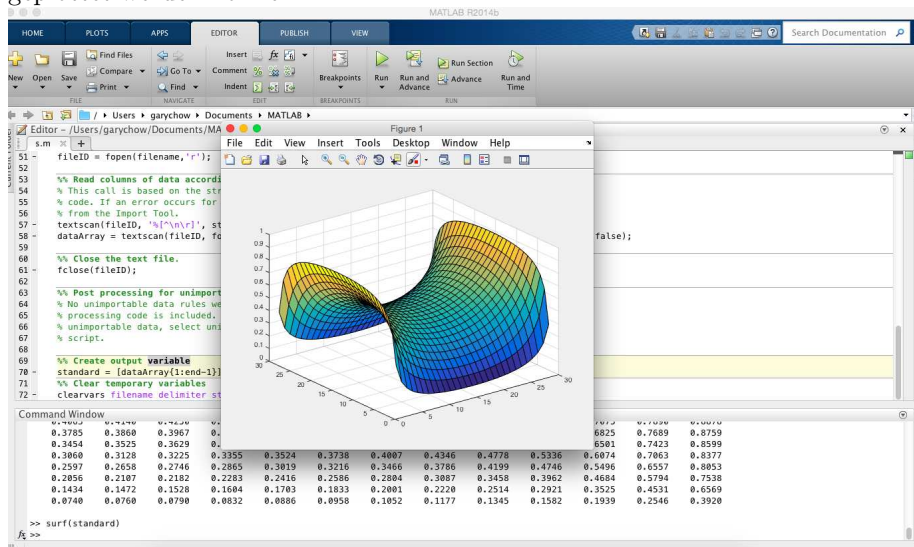
In diesem Abschnitt wird eine kurz Einführung gegeben, wie man mit einer .surface-Datei in üblichen Datenanalyseprogrammen weiterarbeiten kann. Als Beispiel wird hierfür **MATLAB** genommen.



Nachdem man auf *Import Data* geklickt hat, muss man bei den Dateientypen **All files** auswählen, damit die .surface-Dateien zur Auswahl angezeigt werden kann. In unserem Fall importieren wir die Daten aus der Datei *standard.surface*. Nachdem man die zu ausgewerteten Daten markiert hat und die Importeinstellung *Numeric Matrix* ausgewählt ist, kann man unter *Import Selection* die Unterfunktion *Generate Script* anklicken.



Nun werden die Daten in Matlab geladen, welche z.B. mit dem Befehl `surf(filename)` geplottet werden können.



4.3 Eingaben

Das Programm besitzt zwar eingebaute Erkennung richtiger Eingabewerte, jedoch nicht Erkennung sinnvoller Eingabewerte, d.h. wie sich die Rechenzeit in Abhängigkeit der eingegebenen Parameter verhält. Im Folgenden sind also Wertebereiche beschrieben, bei denen das Programm nicht länger als eine Minute Laufzeit auf einem durchschnittlichen Laptop hatte.

- $\epsilon \in [10^{-9}, 1]$
- $iterations \in [10, 20]$
- $stuetz_x, stuetz_y \in [10, 60]$

Am größten beeinflusst wird die Laufzeit des Programms durch die gewählte Anzahl der Stützstellen.

Die Wahl der Randfunktionen ist relativ beliebig und gut geschützt, d.h. durch Fehlersituationen abgefangen. Es werden alle

4.4 Beispielsitzung

4.5 Fehlersituationen

Das Programm gibt Fehlermeldungen aus. Diese können nur entstehen, wenn der Definitionsbereich verändert wird. Es wird eine Fehlermeldung ausgegeben, wenn ein Feld leer ist, einen Buchstaben enthält, die Grenzen außerhalb eines vordefinierten Intervalls sind oder die Anzahl der Nachkommastellen zu groß ist. Es sind alle anderen Ausnahmen und mögliche Fehlersituationen so behandelt, dass diese nicht angezeigt werden müssen und sich der Benutzer komplett mit der Bedienung des Programms beschäftigen kann. Sollten dennoch Fehler auftreten bitten wir diese an stefan.jeske@rwth-aachen.de zu melden.

Kapitel 5

Entwicklerdokumentation

5.1 Codestruktur

Der Code besteht aus mehreren Klassen. Die Quelldateien befinden sich alle im gleichen Verzeichnis. Siehe Kapitel 4 für Installation. Im Folgenden wird die Codestruktur im Groben beschrieben. Für eine detaillierte Dokumentation siehe Abschnitt 2 in diesem Kapitel.

5.1.1 Klasse MainWindow

Die Klasse `MainWindow` ist für die graphische Darstellung der Benutzeroberfläche zuständig. Der Quellcode findet sich in Sektion A.1.1 und in den Dateien `mainwindow.h` und `mainwindow.cpp`.

wohlstrukturierte und gut lesbare Dokumentation der Software aus Entwicklersicht; zahlreiche Referenzen nach Kapitel 3 und in den Quellcode in Kapitel A

Die Signatur der Funktion `foo` finden Sie in Zeile 3 des Quellcodes A.1 in Sektion A.1.1.

Anhang A

Quellcode

einfach referenzierbare Version des Quelltexts

A.1 Paket 1

A.1.1 Klasse 1.1

Listing A.1: Dokumentierter Quellcode in `klasse1.1.hpp`

```
1 class foo {  
2     public :  
3         int bar(const float& f);  
4 }
```

A.1.2 Klasse 1.2

A.2 Paket 2

A.3 Paket 3