Technische Universität München

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Computergestützte Modellierung und Simulation

Untersuchung der parametrischen Modellierung von Bogenbrücken aus Beton mit Dynamo und Revit

Bachelorthesis

für den Bachelor of Science Studiengang Bauingenieurwesen

Autor: Felix Stauch

Matrikelnummer:

1. Betreuer: Prof. Dr.-Ing. André Borrmann

2. Betreuer: Dominic Singer M.Sc.

Ausgabedatum: 01. Februar 2016

Abgabedatum: 31. Juli 2016

Abstract

This thesis is part of a series of works within the frame of the research project KBE4Infra from the Chair of Computational Modelling and Simulation at the Technical University of Munich. It is based on the work already submitted by Dominic Singer, M.Sc. The research project investigates knowledge-based engineering (KBE) solutions for infrastructure and their implementation with regard to infrastructural construction. Here, KBE is combined with parametrical design. A previous work introduced the script to design a parametric model for a beam bridge generated by a few abstract customized parameters. The script resulting from the above work is modified and further developed in this thesis.

Supported by knowledge-based engineering, a database shall be established in order to define rules for a parametric modeling of infrastructure. Thus, an economic and qualitative advantage in the planning of infrastructure will be achieved. This research investigates the possibilities of visual programming languages in the field of infrastructure.

This thesis in detail investigates the modeling of a concrete arc bridge with the help of Dynamo, a Revit AddOn used for the visual programming. The final model will be processed and rendered in Revit. Creating a model which adapts to BIM had top priority.

So as to define the variants of the arc gradients, methods for Dynamo were programmed. This enables the definition of the arc gradient along a three-dimensional terrain curve based on customized parameters. The terrain curve is analyzed and suitable gradients are defined, all based on custom user parameters.

Zusammenfassung

Die Arbeit reiht sich in das Forschungsvorhaben KBE4Infra des Lehrstuhls für Computergestützte Modellierung und Simulation an der TU München ein. Sie basiert auf der bereits geleisteten Arbeit von Dominic Singer, M.Sc. Das Forschungsprojekt widmet sich den wissensbasierten Methoden und ihrer möglichen Anwendung im Infrastrukturbau. Verknüpft werden die wissensbasierten Methoden mit parametrischer Modellierung. In einer bereits abgeschlossenen Arbeit wurde eine Balkenbrücke modelliert, welche sich aus mehreren abstrakten Parametern ableiten lässt. Das hierbei entstandene Skript wurde verändert und weiterentwickelt.

Mit Hilfe von wissensbasierten Methoden soll eine Datenbank geschaffen werden, aus welcher Regeln für die Modellierung abgeleitet werden können. Diese Regeln können dann über parametrische Modellierung zu einem Gebäudemodell überführt werden. Dadurch entsteht ein wirtschaftlicher und qualitativer Vorteil bei der Planung von Infrastrukturbauwerken. Mit dieser Arbeit wird untersucht, inwieweit sich visuelle Programmiersprachen zur Modellierung von Infrastrukturbauwerken eignen.

Diese Arbeit untersucht die Modellierung einer Bogenbrücke aus Beton in Dynamo, einem Revit AddOn zur visuellen Programmierung. Das Modell wird in Revit weiterverarbeitet und gerendert. Wichtig ist hierbei die Erzeugung eines BIM-fähigen Modells.

Zur Bestimmung der Varianten des Bogenverlaufs wurden Methoden für Dynamo programmiert. Damit ist die Bogenfindung entlang einer dreidimensionalen Geländehöhenkurve möglich. Der Verlauf des Geländes wird analysiert und geeignete Bogen- oder Doppelbogenverläufe definiert. All dies auf der Grundlage von benutzerdefinierten Parametern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BIM	Building Information Modeling
CAD	Computer Aided Design
KBE	Knowledge-based Engineering
LOD	Level-of-Development
VPL	Visuelle Programmiersprache

1 Einleitung und Problemstellung

1.1 Einleitung

Mit dem Begriff „Brücke“ verbinden wir heutzutage oft außergewöhnliche Infrastrukturbauwerke. Doch es ist ein weiter Weg von den ersten Brücken, die durch Zufall in der Natur entstanden, weil Bäume umfielen oder Wasser sich durch Steine schnitt, zu den technisch komplexen und oft sehr ästhetischen Bauwerken von heute. Aus den Balkenbrücken mit kurzen Spannweiten oder den ersten Hängebrücken aus Lianen und faserartigen Pflanzen entstanden im Laufe der Jahrhunderte bis zu kilometerlangen Bauwerke, die oft auch als repräsentatives und prestigeträchtiges Kunstbauwerk lange Täler, Flüsse oder sonstige Hindernisse überspannen. Heute sind Brücken wichtige Teile unserer modernen Infrastruktur. Sie haben einen enormen volkswirtschaftlichen Wert und sind oft auch Sinnbild für eine Epoche. So ist beispielsweise die Coalbrookdale Bridge in England ein Sinnbild für die beginnende Industrialisierung und die Hochzeit des Stahls ab dem 18. Jahrhundert.

Brückenbauwerke sind oftmals reine Ingenieurbauwerke. Somit ist relativ selten ein Architekt involviert, der ästhetische Anforderungen stellt. Dennoch zeigt uns die gebaute Realität, dass trotzdem immer wieder Bauwerke von beeindruckender Ästhetik entstehen. Der Hauptanteil der Brückenbauwerke in Deutschland sind oft kleinere Brücken. Sie überspannen Autobahnen, Gleise und kleinere Gewässer oder Täler. Oft ähneln sie sich im Erscheinungsbild sehr. Dies ist auf den hohen Grad an Standardisierung zurückzuführen. So ist die Deutsche Bahn beispielsweise im Besitz einer Regelzeichnung für die statische Berechnung kleinerer Fußgängerunterführungen. Technische Regelwerke wie Normen und Richtlinien sowie Richtzeichnungen sind daher für die Konstruktion von Brücken sehr wichtig.

Mit der Normierung von Bauteilen und Herstellungsmethoden wird viel Wissen, das aus empirischen Erkenntnissen und Erfahrungen gewonnen wurde, in Regeln zusammengefasst. Normen bilden eine sehr gute Basis für eine Berechnung als mathematisch-statisches Modell, da sie für die Mehrheit der Fälle eine Berechnungsmethode beinhalten. Dies dient dem Zweck, eine gemeinsame Basis für die Konstruktion und Ausführung zu schaffen und neue Erkenntnisse aus der Forschung für alle verfügbar zu machen. Letztendlich soll durch die Normierung ein Mehrwert für die Gesellschaft entstehen.

1.2 Problemstellung

Variantenstudien sind nicht nur in der Brückenplanung, sondern insgesamt im Baugewerbe ein kontroverses Thema, da sie sowohl viel Zeit als auch finanzielle Ressourcen benötigen. So hat beispielsweise die Deutsche Bahn bereits 2008 in einem von ihr veröffentlichten Leitfaden auf die Wichtigkeit von Variantenstudien hingewiesen:

> *„Besonders kontraproduktiv ist der Wunsch nach geringen Planungskosten, weil sich dadurch ein Variantenstudium mit der Suche nach der jeweils optimalen Lösung verbietet und unter Umständen ein Vielfaches des zunächst Ersparten später draufgelegt werden muss. Allein die Tatsache, dass der Anteil der Entwurfsplanung, in der die Weichen für das ganze Leben einschließlich der Kosten und der Gestalt einer Brücke gestellt werden, an den Gesamtplanungskosten schnöde 5 %(!) bzw. im Vergleich zu den Baukosten gerade einmal 0,8 % ausmacht, zeigt, dass es unsinnig ist, hier sparen zu wollen.“*
>
> *(DB Netze 2008), S. 11*

Mit diesen sehr deutlichen Worten wird klar, dass eine Variantenstudie gerade in der Brückenplanung unverzichtbar ist.

Die Forschung von Dominic Singer, M.Sc. am Lehrstuhl für Computergestützte Modellierung und Simulation dient als Grundlage für diese Arbeit. Mit dem Forschungsprojekt KBE4Infra wird untersucht, inwieweit KBE für Infrastrukturbauwerke in Frage kommen. Ziel ist, repetitive Planungsprozesse, welche auf die Erfahrung und Expertise des Planers zurückgreifen, mit Hilfe intelligenter Unterstützungswerkzeuge teilweise oder gar vollständig zu automatisieren. Gerade die Planung einer Brücke enthält viele sich wiederholende Arbeiten, die stark von Randbedingungen abhängen. Deshalb eignet sich die Planungsphase hervorragend für den Einsatz von KBE in Verbindung mit visuellen Programmiersprachen (Singer et al. 2016), S. 2. Eine Wissensdatenbank erfasst und speichert hierbei Ingenieurwissen, um es bei zukünftigen Aufgaben als Referenz zu nutzen. So kann sich der planende Ingenieur mit der innovativen und kreativen Planung beschäftigen. Er kontrolliert jedoch das Ergebnis und verifiziert es. Somit ersetzt er sich nicht selbst, wie man denken könnte, sondern erleichtert sich seine Arbeit.

Visuelle Programmiersprachen (VPL) eignen sich vortrefflich zur Implementierung dieses Forschungsvorhabens. Sie werden genauer in Kapitel 4 beschrieben. Auf der Basis von VPL ist es auch Laien möglich, mit wenigen Eingabeparametern ganze Bauwerke zu erstellen. Die visuelle Sprache kann dank ihrer Darstellung vom Menschen sehr viel schneller und einfacher interpretiert werden (Ritter et al. 2015), S. 2. Der bereits von einem Experten modellierte Teil ist weiterhin nachvollziehbar, da Schritt für Schritt nachvollzogen werden kann, was genau zu welchem Zeitpunkt und an welcher Stelle mit den Daten passiert. Somit ist es leichter den Überblick zu behalten und zu verstehen, wie die Parameter die Ausgabe beeinflussen. Veränderungen am Modell können in Echtzeit nachvollzogen werden. Auch Optimierungsprozesse können auf gleiche Weise durchgeführt werden. (Singer et al. 2016), S. 1. Im Rahmen dieser Arbeit wurde mit der VPL-Umgebung Dynamo gearbeitet. Dynamo wird an späterer Stelle im Detail vorgestellt.

Ziel dieser Arbeit ist es zu untersuchen, inwieweit sich die Modellierung einer Bogenbrücke mit Hilfe von visueller Programmierung umsetzen lässt. Das aus vorangegangener Forschung bereits vorhandene Skript zur Erstellung einer Balkenbrücke wird weiterentwickelt. Am Ende soll es dem Nutzer ermöglicht werden, durch die Eingabe von abstrakten Parametern mit dem Skript eine Bogenbrücke zu entwerfen. Hierfür werden die Eingangsparameter zunächst vom Planer eingegeben, beispielsweise Brückenlänge oder die Art des Überbaus. Das Skript wird von Dynamo verarbeitet und ist eine Funktion der abstrakten Eingangsparameter. Dynamo gibt als Ausgabe ein Building Information Modeling (BIM) Modell an Autodesk Revit, das dann dort weiterverarbeitet werden kann. Mit Hilfe dieses Skripts, welches vom Benutzer gesteuert wird, lässt sich schnell ein Vorentwurf visualisieren und ohne viel weiteren Aufwand eine Variantenstudie durchführen. Dieser technologische Wandel vom traditionellen zweidimensionalen Planen zum BIM-Modell, das dreidimensional am Computer erstellt wird, verändert die Baubranche momentan maßgeblich (Singer 2015), S. 2. Mit Hilfe von BIM kann die gesamte Planung bis hin zum Objektbetrieb in einem digitalen Modell abgebildet werden. Daher war es für diese Arbeit wichtig, zukunftsweisend ein BIM-Modell zu erstellen. Zum Thema BIM ist in dieser Arbeit ein eigenes Kapitel verfasst. Dort wird die zentrale Idee erläutert und die Funktionsweise beschrieben. Da BIM zwar oft als Begriff bekannt ist, jedoch oft nicht wirklich klar ist, was genau damit gemeint

ist, soll Kapitel 3 dies dem Leser kurz erläutern und den Einstieg erleichtern. Die Erläuterungen beziehen sich auf (Borrmann et al. 2015), für eine tiefere Auseinandersetzung mit BIM sehr zu empfehlen ist.

Oft entscheidet die Erfahrung des Planenden über die grobe Dimensionierung. Somit entsteht ein Prototyp, der wiederum die Basis für weitere Planungsschritte ist. Hier wird Innovation und Kreativität vom Planer verlangt, er greift für den Entwurf auf seine Erfahrung zurück. Dabei bewegt er sich im Rahmen gewisser Ausgangsbedingungen (Parameter), um eine Lösung für das Problem zu finden. Der planende Ingenieur befindet sich, wie immer, in einem Spannungsfeld aus mehreren Ansprüchen: Ästhetik, Funktionalität, Herstellung, Nachhaltigkeit und Wirtschaftlichkeit (Fischer 2013). Da allerdings der Transfer der Idee des Planers zu einem BIM-Modell grundsätzlich Schritt für Schritt „von Hand“ gemacht wird, ist eine Variantenstudie oft sehr zeitaufwendig und kostenintensiv.

Allerdings ist auch die Baubranche den stetig sinkenden Preisen für Leistungen ausgeliefert, insofern besteht ein hoher Einsparungsbedarf. Auf dem Markt herrscht ein starker Wettbewerb, dadurch werden die Preise gedrückt und oftmals am Ende der günstigste Entwurf gewählt. Dieser muss aber noch lange nicht der optimale sein. Um diesem Abwärtstrend bei Qualität und Ästhetik zu entkommen, bietet die wissensbasierte Methode mit der Hilfe von visueller Programmierung einen möglichen Ausweg.

In (Singer 2014) wird diese Problematik ebenfalls erläutert. Hierbei wird auf die auch dort zitierte Literatur (Stokes 2001) und (Verhagen et al. 2012) verwiesen. So zeigt sich, dass der Entwurf bei einer traditionellen CAD-Planung zu 20% aus Innovation / Kreativität und zu 80% aus Routine besteht. Durch die Verwendung von KBE verringert sich der Routine-Anteil, wodurch Arbeitszeit eingespart wird. Ist die erste Implementierung vollendet, so kann in sehr schneller Zeit eine Vielzahl von Varianten durchgespielt werden, da ab diesem Zeitpunkt lediglich die Parameter geändert werden müssen. Für eine detaillierte Beschreibung von KBE wird (Singer 2014) empfohlen.

Mit Hinblick auf die kürzlich geänderten Planungsvorschriften der Politikbezüglich Infrastrukturbauwerken ist es außerordentlich wichtig, den Einsatz von Building Information Modeling weiter voranzutreiben. Von staatlicher Seite wurde festgelegt, dass ab 2020 alle Infrastrukturbauwerke der öffentlichen Hand ausschließlich mit BIM zu planen sind (Bundesministerium für Verkehr und digitale Infrastruktur 15.12.2015). Mit Hilfe des parametrischen Designs einer Brücke steht im Idealfall am Ende ein BIM-

Modell, das für den gesamten Lebenszyklus der Brücke verwendet werden kann. Dies hat nicht nur planerische, sondern auch wirtschaftliche Vorteile ab dem Entwurf über den Bau, der Nutzung bis zur Sanierung oder dem Abbruch.

1.3 Aufbau der Arbeit

Die Arbeit ist folgendermaßen aufgebaut: Zunächst wird eine Übersicht über die Konstruktion und Planung von Bogenbrücken aus Beton gegeben. Spezifische Details und Eigenheiten werden erläutert, sodass im späteren Verlauf die Modellierung der Brücke verständlich wird.

Im Anschluss folgt eine Vorstellung der verwendeten Software. Hier wird kurz erklärt um welche Programme es sich handelt.

In Kapitel 4 wird der Einblick in die visuelle Programmierung vertieft und anhand eines Beispiels erläutert. Somit soll es dem Leser später leichter fallen, die Modellierung der Brücke nachzuvollziehen.

Der Modellierung und der damit verbundenen Schwierigkeiten widmet sich Kapitel 5. Hier wird auch der geschriebene Programmcode eingehender definiert sowie die Überlegungen zur automatisierten Bogenfindung und ihrer Umsetzung genauer erläutert. Der gesamte Code ist auf der beiliegenden Compact Disc zu finden, welche auch Teil dieser Arbeit ist.

Zum Schluss wird in einem Fazit die Arbeit diskutiert und das Potential wissensbasierter Methoden erläutert.

2 Bogenbrücken

In diesem Kapitel werden das Funktionsprinzip einer Bogenbrücke sowie die Besonderheiten bei Gründung, Querschnitt und weiteren Details beschrieben. Dies geschieht hauptsächlich auf Basis einer Literaturrecherche von (Mehlhorn und Curbach 2014) sowie (Geißler 2014). Zwar ist durchaus noch weitere Literatur zu Dimensionierung, Ästhetik und Planung von Brückenbauwerken zu finden, jedoch wurden für diese Arbeit die genannten Veröffentlichungen als ausreichend angesehen.

2.1 Der Bogen

Durch seine Form ist der Bogen eine der stabilsten Stützformen. Bei einer idealen Krümmung entspricht die Bogenlinie der Stützlinie. Man stelle sich eine Kette vor, die an bestimmten Stellen mit Gewichten behängt und an beiden Enden aufgehängt wird. Da eine Kette nur Zugkräfte übertragen kann, bildet sich aufgrund der Schwerkraft eine Durchbiegung an der biegeweichen Kette aus. Diese Figur nennt man Seillinie. Kehrt man diese Seillinie um, so erhält man das entsprechende Gegenstück: einen rein auf Druck beanspruchten Bogen. Baut man einen Bogen mit dieser Form, so kann an den Punkten, an welchen davor Gewichte hingen, eine entsprechende Kraft eingeleitet werden. Antonio Gaudi erstellte auf diese Weise die Stützlinien für die Sagrada Familia in Barcelona. Er modellierte ein Seilmodell, das er mit Sandsäcken behängte, um die Krafteinleitung zu simulieren. Nun musste er lediglich die Seillinie von den durchhängenden Seilen abzeichnen, umdrehen, und erhielt somit die ideale Stützlinie für seine Bauwerke. Daraus ergaben sich die Formen für Gewölbe, Türme und andere Überdachungen.

Doch Gaudi war durchaus nicht der erste, der Bögen als stabiles Tragwerk entdeckte. Schon früh wurden Bögen eingesetzt, um Weiten stützenfrei zu überspannen. Die römischen Aquädukte sind eindrucksvolle Beispiel hierfür. Große Spannweiten und steile Täler sind geradezu ein prädestiniertes Gelände für den Einsatz von Bögen, die so vor allem im Brückenbau neue Wege boten, um Täler und Flüsse elegant zu überbrücken.

Im Rahmen dieser Arbeit wird ausschließlich eine Bogenbrücke aus Beton betrachtet, da Beton, gerade auch Spannbeton, seine hohe Druckfestigkeit auf diese Weise be-

sonders gut ausnutzen kann. „Der Bogen kann daher als ein ideal vorgespanntes Tragwerk betrachtet werden, das alle Vorteile des Spannbetons besitzt, nicht aber dessen Nachteile (Spannkraftverlust durch Kriechen, Möglichkeit der Schädigung oder des Ausfalls von Spanngliedern).“ (Mehlhorn und Curbach 2014), S. 530. Man darf jedoch nicht vergessen das es auch Bogenbrücken aus anderen Materialien gibt. Gerade Bauwerke mit einem über der Fahrbahn angeordneten Bogen werden in der Regel in Stahlbauweise ausgeführt.

Ein Nachteil des Bogens ist, dass er kaum für in der Lage gekrümmte Fahrbahnachsen brauchbar ist. Im Idealfall verläuft die Fahrbahnachse linear. Für diese Arbeit wurde ein im Grundriss linearer Verlauf gewählt.

Generell ist zwischen zwei Bogenarten zu unterscheiden. Es gibt den „idealen“ Bogen, welcher stetig gekrümmt ist, sowie sogenannte Stabbögen (Abbildung 2.2.1). Stabbögen sind zwischen den Krafteinleitungspunkten gerade und knicken lediglich dort ab. Ein Stabbogen lässt sich bei kleiner Anzahl an Krafteinleitungspunkten bereits sehr deutlich optisch erkennen. Doch nicht nur optisch, sondern vor allem im statischen System unterscheiden sich beide Bogenarten enorm. Während der ideale Bogen, wie es Gaudi ermittelte, eine Stützlinie abbildet, die fast nur Druckkräfte überträgt, ist der Stabbogen, wie es der Name sagt, aus mehreren Stäben zusammengesetzt. Statisch gesehen repräsentiert jeder Knick ein Gelenk. Der Bogen wirkt somit lediglich aussteifend für die Fahrbahn und trägt sich nicht selbst. In dieser Arbeit wurde die Untersuchung eines idealen Bogens definiert.

2.2 Statische Systeme eines Bogens

Für einen idealen Bogen ist der Bogen als steifes System auszuführen. Zwar gibt es auch die Möglichkeit, einen Dreigelenkbogen mit einem Gelenk in der Mitte auszuführen, allerdings sind diese aber eher ungünstig für das Stabilitätsverhalten (Mehlhorn und Curbach 2014), S. 533. Bei einem steifen Bogen unterscheidet sich das statische System durch die Lagerung. Der Bogen kann beidseitig eingespannt, beidseitig gelenkig oder in einer Kombination von beidem gelagert sein. Letztere Möglichkeit ist jedoch aufgrund des schlechten Kriech-, Schwind- und Stabilitätsverhaltens nicht zu wählen (Mehlhorn und Curbach 2014), S. 533. Eingespannte Bögen sind in der Regel zu bevorzugen. Durch die Momentenbelastung resultiert eine höhere Kantenpressung im Kämpfer. Hingegen ist die Lasteinleitung im Zweigelenkbogen deutlich gleichmäßiger,

da die Momente nicht bis in den Kämpfer übertragen werden. Zweigelenkbögen eignen sich somit bei flacheren Bögen. In (Mehlhorn und Curbach 2014), S. 533 wird hierfür ein Wertebereich angegeben, der bei der Entscheidung der Wahl des statischen Systems helfen soll:

Eingespannter Bogen $\frac{f}{l} \geq \frac{1}{7}\ bis\ \frac{1}{3}$

Zweigelenkbogen $\frac{f}{l} \geq \frac{1}{10}\ bis\ \frac{1}{6}$

Hierbei steht f/l für das Pfeilverhältnis. Das Pfeilverhältnis ist die Relation der Höhe des Bogens, dem sogenannten Stich f zum Abstand der Bogenfußpunkte l (Abbildung 2.2.1). Für die Modellierung wurde dieser Wertebereich implementiert, der Algorithmus findet Bögen, die ein minimales Pfeilverhältnis übersteigen und wählt abhängig davon das statische System.

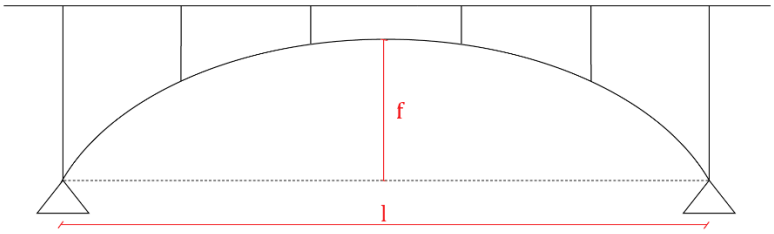

Abbildung 2.2.1: Pfeilverhältnis

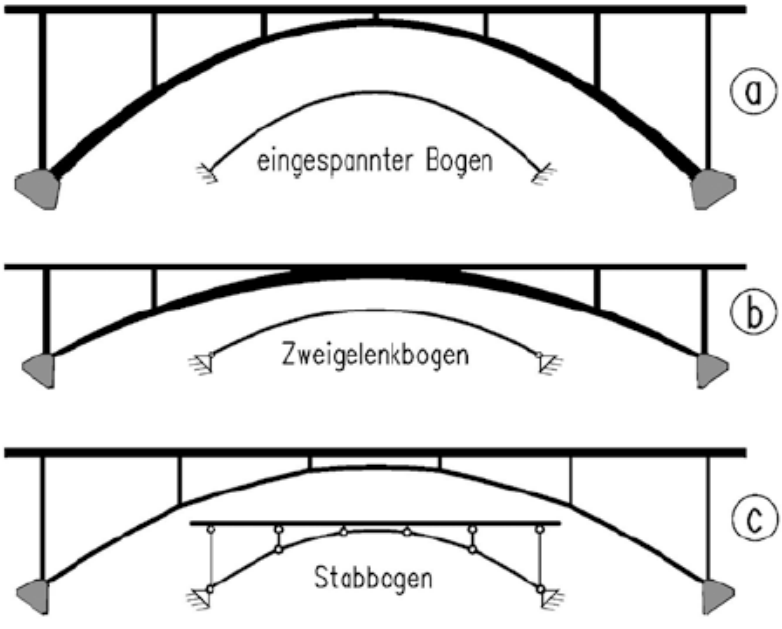

Abbildung 2.2.2: Statische Systeme von Bogenbrücken aus (Mehlhorn und Curbach 2014)

Das Pfeilverhältnis sollte nicht zu groß sein, dies ist bei Brücken mit hohen Bögen der Fall. Diese wirken unästhetisch und auch ihre Lagerung und Stabilität sind oftmals problematisch. Je nach Randbedingung kann das obere Limit variieren, daher kann man lediglich ein unteres Limit festlegen.

2.3 Gründung

Der Bogen wird auf beiden Seiten auf den Kämpfern gegründet. Diese sorgen für ein stabiles Fundament und eine gleichmäßigere Krafteinleitung in den Boden. In (Geißler 2014), S. 89 wird beschrieben, dass die Kämpfer aufgrund der hohen Kräfte eine gute Gründungsbedingung erfordern. Dies kann zum Beispiel durch Tiefgründungen oder Felsgründungen erreicht werden. Bei einem flachen Kreuzungsverhältnis von Druckkraft und Gelände müssen sehr hohe Horizontalkräfte abgetragen werden. Daher kommen die entsprechend guten Gründungsverhältnisse.

Im Gegensatz hierzu schreibt (Mehlhorn und Curbach 2014), S. 530: „Vielfach wird die Notwendigkeit eines überdurchschnittlich guten Baugrunds angenommen, allerdings nicht ganz zu Recht.“ Im weiteren Verlauf wird beschrieben, wie der Kreuzungswinkel die Gründung positiv beeinflussen kann. Die Gründung wird dadurch oft wesentlich einfacher und der Hang stabilisiert. Dies ist jedoch immer im Einzelfall zu prüfen.

Eine weitere Stabilisierung des Kämpfers bieten der Anfangs- und Endpfeiler. Durch die vertikale Krafteinleitung pressen sie den Kämpfer stärker in den Boden.

In Abbildung 2.3.1 ist die Salginatobelbrücke von Robert Maillart zu sehen. Die Brücke stützt sich regelrecht in den vertikalen Fels. Sie wurde als Dreigelenkbogen ausgeführt mit einem Gelenk in der Mitte.

Abbildung 2.3.1: Salginatobelbrücke in der Schweiz von Robert Maillart (Rama 2015)

2.4 Bogenachse

Die Bogenachse ist in der Regel nicht eine Parabel zweiter oder höherer Ordnung, sondern wird numerisch eingerechnet (Mehlhorn und Curbach 2014), S. 534. Da Brücken durch Verkehrslasten beansprucht werden, entsteht ein variables Lastbild. Deshalb ist es nicht möglich eine ideale Bogenachse zu finden, da sich je nach Lastfall auch die Bogenachse verändert (Mehlhorn und Curbach 2014), S. 531. Man kann sich nur an ein Optimum annähern. Für einen ersten Entwurf oder eine Variantenstudie ist die Annahme einer Parabel zweiter Ordnung jedoch ausreichend.

Das eigentliche Ziel jedoch ist die praktische Implementierung der Brücke in eine gegebene Landschaft. Um nun dem Nutzer möglichst viele Möglichkeiten für die Konstruktion des Bogens zu geben, kann die Bogenachse auf verschiedene Weisen bestimmt werden. Kurz zusammengefasst kann der Nutzer individuell die Kämpferstandorte bestimmen und zwischen Parabel-, Kreisbogen und eigens angegebener Bogenachse wählen. Des Weiteren kann das Programm entsprechende Bogenachsen definieren und der Nutzer wählt Varianten aus. Dieser Prozess wird in Kapitel 5 genau beschrieben.

Da Brücken auch gleichzeitig mehrere Täler überspannen können, ist es möglicherweise sinnvoll, einen Doppelbogen auszubilden. Dieser hat den Vorteil, dass sich der

mittlere Kämpfer nahezu von selbst stabilisiert. Ein Beispiel für eine solche Ausführung ist die Falkensteinbrücke in Österreich (Abbildung 2.4.1).

Abbildung 2.4.1: Falkensteinbrücke in Österreich als Beispiel für eine Doppelbogenbrücke (MAPLOGS 2010)

3 Building Information Modeling

3.1 Allgemeines

Bauprojekte zeichnen sich durch ihre hohe Komplexität aus. Verschiedene Fachplaner sind bei der Planung involviert, wodurch viele Informationsschnittstellen entstehen. Da traditionellerweise mit Bauplänen gearbeitet wird, in denen Anmerkungen und Änderungen meist mit der Hand vermerkt werden, gibt es keinen Kontrollmechanismus der sicherstellt, dass wirklich alle Informationen zusammenpassen. Darüber hinaus zeichnet jeder Fachplaner oft seine eigenen Pläne, wodurch es leicht zu Widersprüchen zwischen den jeweilig verwendeten Plänen kommen. Diese Problematik führt zu einer hohen Fehleranfälligkeit. Insbesondere Großprojekte haben in letzter Zeit damit für Schlagzeilen gesorgt: nicht nur der Flughafen Berlin ist ein Beispiel für die fehlende Planungskompetenz, Kommunikation und Zusammenarbeit. Das ursprünglich 2007 zur Eröffnung geplante Projekt ist zum aktuellen Zeitpunkt nicht nur nicht abgeschlossen, ein Eröffnungstermin wird gegenwärtig nicht einmal mehr diskutiert.

Um in Zukunft Probleme dieser Art zu vermeiden und auch bei Großprojekten die Kontrolle zu behalten, benötigt man eine Lösung, die alle Beteiligten gleichzeitig an demselben Modell arbeiten lässt. Damit ist jeder im Projekt Involvierte in Echtzeit auf dem aktuellen Stand und sieht, an welcher Stelle andere Gewerke Änderungen eingebracht haben. Fehler können so frühzeitig erkannt und vermieden werden. Ein weiterer Vorteil eines solchen Modells ist, dass es im Laufe der Planungsphase stetig mit – im Vergleich zu einem handschriftlichen Plan – übersichtlich strukturierten, jederzeit abrufbaren Informationen gefüllt wird. So gehen diese nicht verloren und können nach Fertigstellung für den weiteren Lebenszyklus des Bauwerks verwendet werden. Der Objektbetrieb und die Instandhaltung verwenden einfach das im Entwurf erstellte Modell weiter.

3.1.1 Begriffsdefinition

Genau dieser Ansatz ist die Grundidee des Building Information Modeling (BIM). BIM ist für den digitalen Entwurf im Bauwesen konzipiert, soll die Kommunikation verbessern, die Zusammenarbeit erleichtern und wird bereits ab der Entwurfsphase für den gesamten Lebenszyklus eines Bauwerkes genutzt (Abbildung 3.1.1). Im digitalen Modell kann der Bauprozess nicht nur simuliert, sondern später auch überwacht werden. BIM ist das digitale, dreidimensionale Abbild eines Bauwerks, beinhaltet jedoch weitaus mehr als lediglich die dreidimensionale Geometrie. Mit Hilfe sogenannter semantischer Informationen werden im Modell Materialeigenschaften, Bauteilbezeichnungen und Beziehungen untereinander gespeichert. Die Semantik eines BIM-Modells bezeichnet die Beziehung der Objekte zueinander. So hängt beispielsweise die Höhe einer zu platzierenden Wand von der definierten Geschosshöhe ab. Wenn sich zu einem späteren Zeitpunkt die Höhe aller Wände eines Stockwerkes ändert, muss lediglich die Geschosshöhe verändert werden, da alle Wände mit diesem Parameter verknüpft sind.

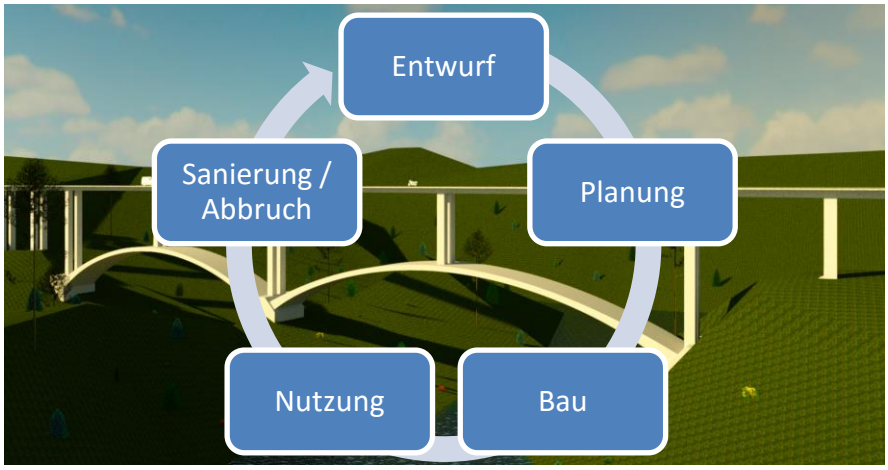

Abbildung 3.1.1: Lebenszyklus eines Bauwerks

BIM ist objektorientiert, das heißt jedes Bauteil wird platziert und ist mit seinen Eigenschaften in einer zum Modell gehörenden Datenbank verknüpft. Jedes Bauteil ist mit einer Identifikationsnummer belegt und kann im Laufe seines Lebenszyklus mit weiteren Informationen versehen werden. Materiallisten oder Volumina können aus dieser Datenbank des BIM-Modells abgeleitet werden und sind für den Nutzer mit nur wenigen Klicks verfügbar.

3.2 Verwendete Software

3.2.1 Autodesk Revit 2017

Abbildung 3.2.2: Logo Revit 2017

Revit ist eine BIM-Softwareanwendung aus dem Hause Autodesk. Autodesk ist ein weltweiter Anbieter von CAD- und Planungs-Software. Im Unterschied zu Autodesks bekanntem Programm AutoCAD arbeitet Revit objektorientiert. Mit Revit stieg das Unternehmen 1998 in den USA in die BIM-Software ein. Seit dem Jahr 2004 ist Revit auch in Deutschland erhältlich. Diese Software vereint mehrere Komponenten. So bietet sie beispielsweise Werkzeuge für Architekten, Tragwerksplaner sowie Gebäudetechniker, wodurch nun alle Disziplinen schlussendlich mit ein und demselben Modell arbeiten können. Im Rahmen dieser Arbeit wurde die aktuelle Version Revit 2017 verwendet.

In Revit gibt es die Möglichkeit, sogenannte Familien zu erstellen. Als Familie versteht Revit ein vorgefertigtes Objekt, das als Instanz im Projekt platziert werden kann. Eine Familie hängt von Parametern ab. Fenster und Wand sind einfache Beispiele für Familien in Revit: Einerseits kann ein Parameter die Beziehung zu einem anderen Objekt beschreiben, beispielsweise die Position eines Fensters in einer Wand, andererseits beschreibt ein Parameter auch die Abmessungen eines solchen Fensters (beziehungsweise der Wand). Bevor allerdings eine Familie platziert wird, muss sie durch ihren Ort, ihre Beziehung zu anderen Objekten sowie die dazugehörigen Parameter definiert werden. Deutlich wird dies an einem Stahlträger: Der Querschnitt ist bereits vordefiniert, Länge und Einbauort müssen aber noch durch den Benutzer definiert werden. Die Familie der Stahlträger beinhaltet mehrere Typen (zum Beispiel HEA, IPE, …). Typen können für Familien existieren, sind aber nicht notwendig.

Das Programm bietet zwar eine gewisse Breite von Familien, diese sind jedoch nicht immer ausreichend. So waren für die Umsetzung der Bogenbrücke beispielsweise keine geeigneten Familien vorhanden, um deren Geometrie korrekt darzustellen. Wie mit diesem Problem umgegangen wurde, wird in Kapitel 5 beschrieben.

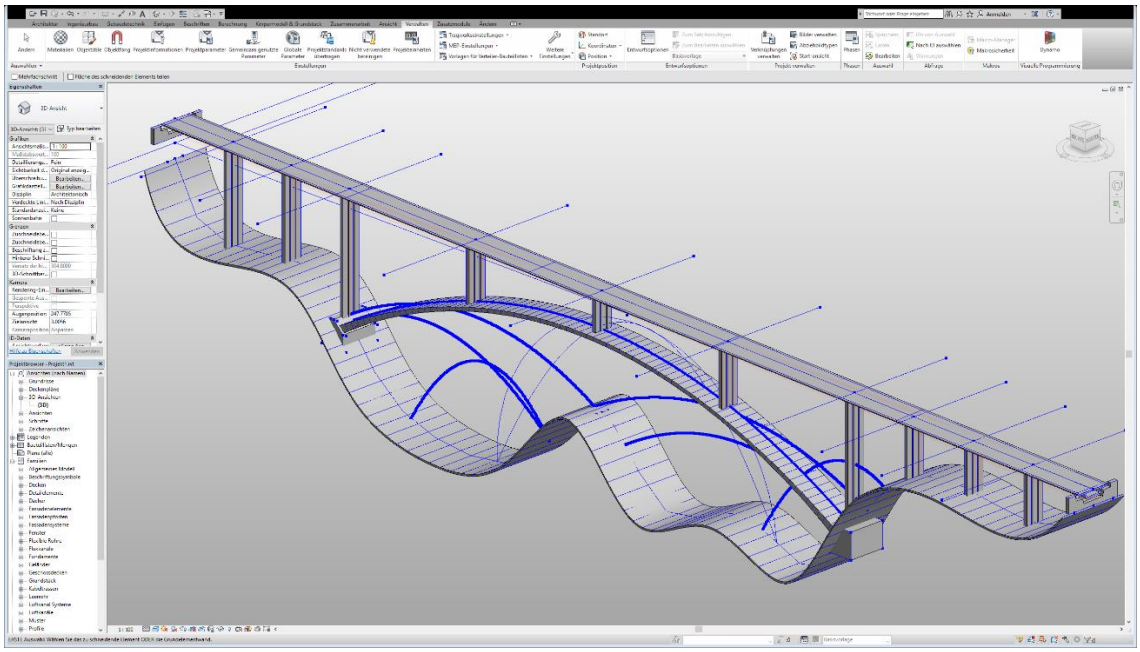

Abbildung 3.2.3: Bildschirmausschnitt aus Revit 2017

3.2.2 Dynamo 1.0

Dynamo ist ein Open Source Programm und wird von der Firma Autodesk und anderen Unterstützern kontinuierlich weiterentwickelt.

Abbildung 3.2.3: Logo Dynamo

Dynamo ist frei verfügbar und kann als Erweiterung für Revit heruntergeladen oder – wenn man nicht über eine gültige Revitlizenz verfügt - als Autodesk Dynamo Studio gekauft werden. Für diese Arbeit wurde mit Revit und Dynamo 1.0 gearbeitet. Das Programm kann als Erweiterung ausgeführt werden oder aber als eigenständige Instanz, jedoch fehlen dann alle Knoten zur Verknüpfung mit Revit.

Dynamo bietet neben einer visuellen Programmierebene auch eine eigene graphische Visualisierung des Outputs. In der Standardversion sind zunächst Methoden zur Bearbeitung von Listen, mathematische Operatoren, geometrische Erzeugung und Manipulation sowie weitere nützliche Methoden vorhanden. Über die große Gemeinschaft von Nutzern können weitere Funktionen geteilt werden.

Schaut man sich die Beschreibung von Dynamo im offiziellen Handbuch an, so steht dort: „Dynamo is, quite literally, what you make it.“ (Autodesk 2015). Durch die Vielzahl

möglicher Erweiterungen hat Dynamo tatsächlich ein sehr großes Potential. Mit Hilfe von C# können eigene Klassenbibliotheken erstellt und als .dll-Datei in Dynamo importiert werden. Dadurch sind der Kreativität und den Möglichkeiten keine Grenzen gesetzt. Alle Methoden, die in Dynamo verfügbar sind, lassen sich auch in der C#-Programmierung verwenden. Für diese Arbeit wurden die Knoten mit Microsofts Visual Studio 2015 programmiert und erstellt.

Eine weitere nützliche Funktionalität sind die benutzerdefinierten Knoten, die *CustomNodes*. Eine Kombination von Knoten kann gespeichert und als einzelner Knoten eingefügt werden und ist damit beliebig oft wiederverwendbar. So können mehrere Knoten gruppiert und als übergeordneter Knoten definiert werden. Dies ist vergleichbar mit einer Funktion in der klassischen objektorientierten Programmierung. Im Gegensatz zu den Dynamo-Projektdateien, die als .dyn-Datei gespeichert werden, werden *CustomNodes* als .dyf-Datei gespeichert. Sie können damit auch in anderen Projekten wiederverwendet werden. Werden sie jedoch an einer Stelle geändert, so ändert sich der Knoten in allen Projekten, in denen er verwendet wird.

Während der Modellierung der Brücke haben sich die *CustomNodes* als nützlich erwiesen, allerdings sollten nicht zu viele Funktionen in einem *CustomNode* zusammengefasst werden. Dadurch würden unübersichtliche und große Knoten entstehen. Daten, die innerhalb dieses überdimensionalen Knotens verarbeitet werden, sind nicht nachvollziehbar und können bei auftretenden Problemen zu sehr viel Arbeit führen. Dieses Problem ist auch in der klassischen objektorientierten Programmierung bekannt. Hier gilt die Faustregel: Eine Aufgabe, eine Funktion. Für kleinere, oft auftretende Methoden (Algorithmen) empfiehlt es sich jedoch einen benutzerdefinierten Knoten zu erstellen.

Ein weiterer Knotentyp sind die *CodeBlock*s. Sie werden in Dynamo erstellt und können dort mit Programmcode gefüllt werden. So ist es auch ohne eigene C#-Klassenbibliothek möglich, textbasiert Objekte zu erstellen, If-Anweisungen oder sonstige Operationen auszuführen. Dies wird in Kapitel 4 ebenfalls genauer beschrieben.

In Abbildung 3.2.4 ist ein Bildschirmausschnitt der Benutzeroberfläche von Dynamo abgebildet. Man sieht im großen zentralen Fenster die verknüpften Knoten und im Hintergrund die graphische Darstellung. Zwischen Graphik und Knoten kann rechts oben umgeschaltet werden.

Dynamo besitzt, wie viele andere Programme zur visuellen Programmierung, eine zweidimensionale Zeichenfläche. In dieser Zeichenfläche können Knoten aus der Bibliothek platziert werden. Ein Knoten ist eine Methode, welche in der Regel auf der einen Seite die Eingangswerte über Verknüpfungspunkte darstellt und auf der anderen Seite die Ausgabe zur weiteren Verknüpfung bereitstellt. Der Benutzer kann diese Knoten miteinander verbinden und visualisiert so den Informationsfluss. Diese visuelle Verknüpfung von Informationen erleichtert das Verständnis und Ausgaben können infolgedessen schneller interpretiert werden. Die schwarzen Linien zwischen den Knoten bezeichnen eine Verknüpfung dieser Information. Parameter werden hier übergeben.

In der linken Spalte ist die Sammlung der Knoten. Es gibt mehrere Kategorien, zum Beispiel Core, Geometry, Office, … Diese Kategorien können durch C#-Klassenbibliotheken oder Pakete aus der online Datenbank erweitert werden. Wird Dynamo über Revit geöffnet, so erscheint auch die Kategorie Revit, welche Methoden beinhaltet, die Revit Application Programming Interfaces (API) nutzen. Die Revit APIs dienen unter anderem dem Erstellen, Platzieren, Modifizieren oder Auswählen von Objekten und Familien in Revit. Beim Stand von Version 1.0 ist die Zusammenarbeit von Revit und Dynamo aber noch nicht sehr ausgereift. Diese Problematik wird am Ende des Kapitels 5 näher beschrieben.

Im nachstehenden Bildschirmausschnitt ist die Kategorie Geometry geöffnet und in einem Unterpunkt wurde Solid ausgewählt. Fährt man nun mit der Maus über eine der Methoden zum Unterpunkt Solid, so erscheint jeweils eine kurze Beschreibung mit Visualisierung. Damit ist es auch für unerfahrene Nutzer leicht verständlich, was genau die Methode macht, welche Parameter sie benötigt und was als Ausgabe übergeben wird.

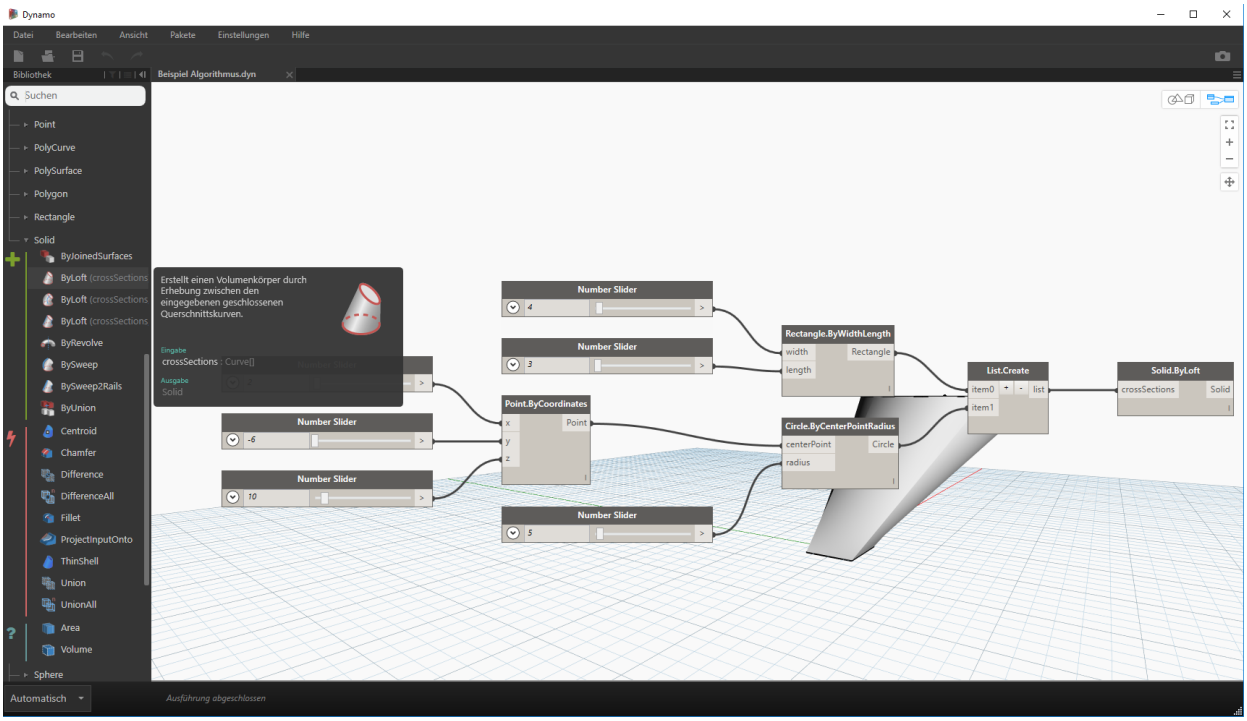

Abbildung 3.2.5: Bildschirmausschnitt aus Dynamo 1.0

Der Import der erstellten Geometrie aus Dynamo nach Revit wird für diese Arbeit mit Familien erledigt. Jedes Bauteil wird als neue allgemeine Familie angelegt und einmal am entsprechenden Ort in Revit platziert. Dies resultiert aus einer Problematik beim Import der Geometrien, auf welche in Kapitel 5 genauer eingegangen wird.

3.2.3 Visual Studio 2015

Abbildung 3.2.5: Logo Visual Studio

Visual Studio wird von Microsoft herausgegeben und erschien das erste Mal in der Version 97 im Jahr 1997. Es ist keine BIM-Software, wird aber zur Übersicht in diesem Teil mitbeschrieben. Visual Studio ist eine integrierte Entwicklungsumgebung für verschiedene Programmiersprachen, darunter C, C++, C# und Visual Basic .NET. Außerdem lassen sich mit HTML, JavaScript und CSS Webanwendungen entwickeln. Die Programme können beispielsweise als Anwendungen für das .NET-Framework, aber auch für Android, iOS und Windows Phone entwickelt werden.

Visual Studio bietet einige hilfreiche Funktionen. So werden einzelne Schlüsselwörter in verschiedenen Farben dargestellt, der Code wird auf Fehler überprüft und automatisch formatiert. Für diese Arbeit wurde Visual Studio 2015 verwendet.

4 Visuelle Programmiersprachen

Im Gegensatz zur textbasierten Programmierung bieten visuelle Programmiersprachen (VPL) den Vorteil, dass das klassische Schreiben von Programmcode in textueller Form, welches oftmals die Hürde für Anfänger darstellt, komplett wegfallen kann. Allein durch die reine visuelle Verknüpfung von Methoden ist es möglich, ein gegebenes Problem zu lösen. Der Nutzer sieht, welche Ein- und Ausgabeparameter durch eine Methode verarbeitet werden und verknüpft diese.

4.1 Algorithmen

Ein Algorithmus beschreibt einen Rechenvorgang, der Variablen schrittweise umformt. Dies geschieht nach einem bestimmten Schema (Duden - Algorithmus). Ein Algorithmus führt nacheinander eine endliche Abfolge von Operationen aus, wobei Eingangsdaten in Ausgangsdaten verarbeitet werden. Bei der klassischen textbasierten Programmierung werden die Algorithmen als Code geschrieben. Der Benutzer muss die Sprache und ihre Eigenschaften kennen, um zu einem Ergebnis zu kommen.

Im Gegensatz dazu ist eine VPL sehr intuitiv und leicht zu bedienen. Der Benutzer verknüpft graphisch die Ein- und Ausgangsparameter, um sein Problem zu lösen. Der Benutzer muss zwar auch die Knoten und ihre Funktionalitäten kennen, er benötigt jedoch keine Kenntnis in textbasierten Programmiersprachen. Beim Erstellen kann aufbauend vorgegangen und immer wieder ein Zwischenergebnis angezeigt werden. Um diesen Unterschied zu verdeutlichen, zeigt Abbildung 4.1.1 den Programmcode in C# und Abbildung 4.1.2 die Umsetzung in Dynamo als eine VPL. Erstellt wird ein 3D-Körper durch einen sogenannten Loft zwischen zwei Rechtecken. Dafür benötigt man mindestens zwei geschlossene Querschnitte.

```
public static Solid CreateSolidFromRectangleToCircle(double rect_width, double
rect_length, double circ_x,
    double circ_y, double circ_z, double circ_radius)
{
    var crosssections = new List<Curve>();
    crosssections.Add(Circle.ByCenterPointRadius(Point.ByCoordinates(circ_x, circ_y,
circ_z), circ_radius));
    crosssections.Add(Rectangle.ByWidthLength(rect_width, rect_length));
    return Solid.ByLoft(crosssections);

}
```

Abbildung 4.1.1: C#-Code zum Erstellen von Loft zwischen Rechteck und Kreis

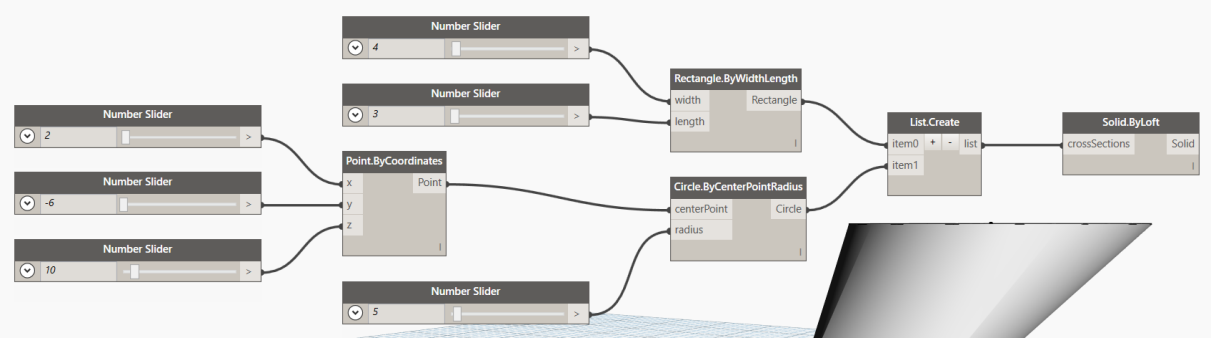

Abbildung 4.1.2: Bildschirmausschnitt aus Dynamo 1.0: Erstellen von Loft zwischen Rechteck und Kreis

Für den Laien ist die visuelle Verknüpfung der einzelnen Knoten deutlich einfacher und verständlicher, auch wenn der Code signifikant kürzer und für den erfahrenen Nutzer auch schnell zu programmieren ist. Dynamo richtet sich auch an Anwender, die noch keine Erfahrung mit Programmieren haben.

4.2 Visuelle Programmierung in Dynamo

Um die visuelle Programmierung anschaulicher zu gestalten, wird nachfolgend eine Oberfläche mit Dynamo erstellt. Diese Oberfläche soll sich aus mehreren Linien ableiten. Begonnen wird zunächst mit einem Punkt. Die Methode *Point.ByCoordinates* fordert als Eingangsparameter die Koordinaten x, y und z. Diese können mit einem sogenannten *NumberSlider* erstellt werden. Anschließend werden double-Werte mit den Eingangsparametern der Methode verbunden.

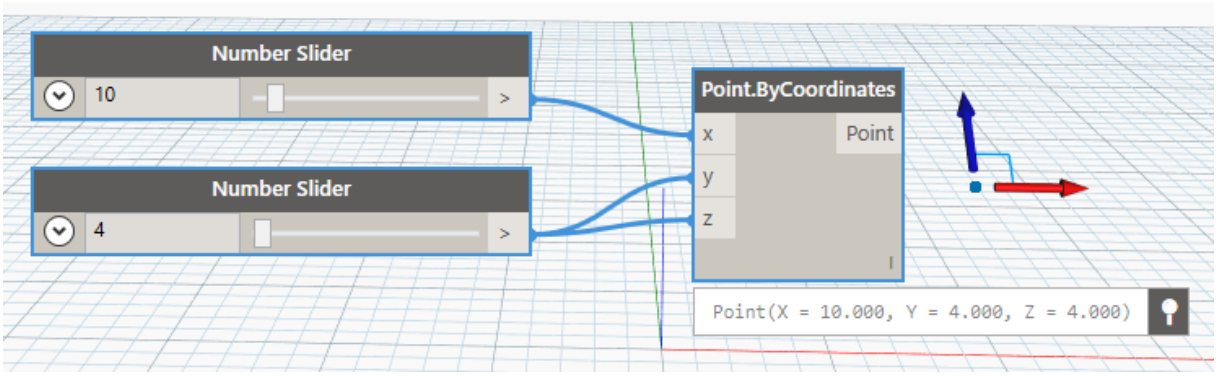

Abbildung 4.2.1: Bildschirmausschnitt aus Dynamo 1.0: Erstellen eines Punktes

Es entsteht ein Punkt als Ausgabe der Werte der beiden *NumberSlider*. Diese Eingangsparameter lassen sich nach Belieben ändern. Mit dem Knoten *Line.ByStartPointEndPoint* wird eine Linie erstellt. Hierfür werden Start- und Endpunkt als Eingangsparameter verlangt und eine oder mehrere Linien ausgegeben. Der Endpunkt wird analog zum Startpunkt erzeugt, allerdings mit anderen Eingangsparametern.

Ein weiterer Vorteil bei Dynamo ist die Verarbeitung von Listen. Eine Methode kann auch eine Liste von Eingangsparametern erhalten und erstellt damit mehrere Objekte. So kann man der Methode *Point.ByCoordinates* eine Liste von double-Werten übergeben und erhält dann eine Liste von Punkten. Werden mehrere Eingangsparameter als Listen übergeben, dann ist der Nutzer in der Lage, die Vergitterung zu wählen. Er kann bestimmen, in welcher Art die Werte verarbeitet werden.

Als Beispiel hierfür soll ein Punkt aus mehreren double-Werte-Listen erstellt werden. Die Liste für die x-Werte hat eine Länge von 10, die der y-Werte ebenfalls. Es könnten als kürzeste Vergitterung eine Liste von Punkten mit jeweils dem gleichen Index von x- und y-Wert erstellt werden, oder als Kreuzprodukt jede Kombination von x- und y-

Wert in den Punkten abgebildet werden. Welche Vergitterung ein Knoten anwendet wird rechts unten im Knoten dargestellt. Im Beispiel wird nur die kürzeste Vergitterung verwendet, welche durch einen vertikalen Strich dargestellt wird.

Mit den erstellten Punkten kann im nächsten Schritt eine Liste von Linien erstellt werden. Der benutzte *CodeBlock* dient dazu mehrere Werte zu generieren. Dynamo kann Wertbereiche zwischen Limits generieren. Für eine genauere Beschreibung der dazu zu verwendenden Syntax wird auf (Autodesk 2015) verwiesen.

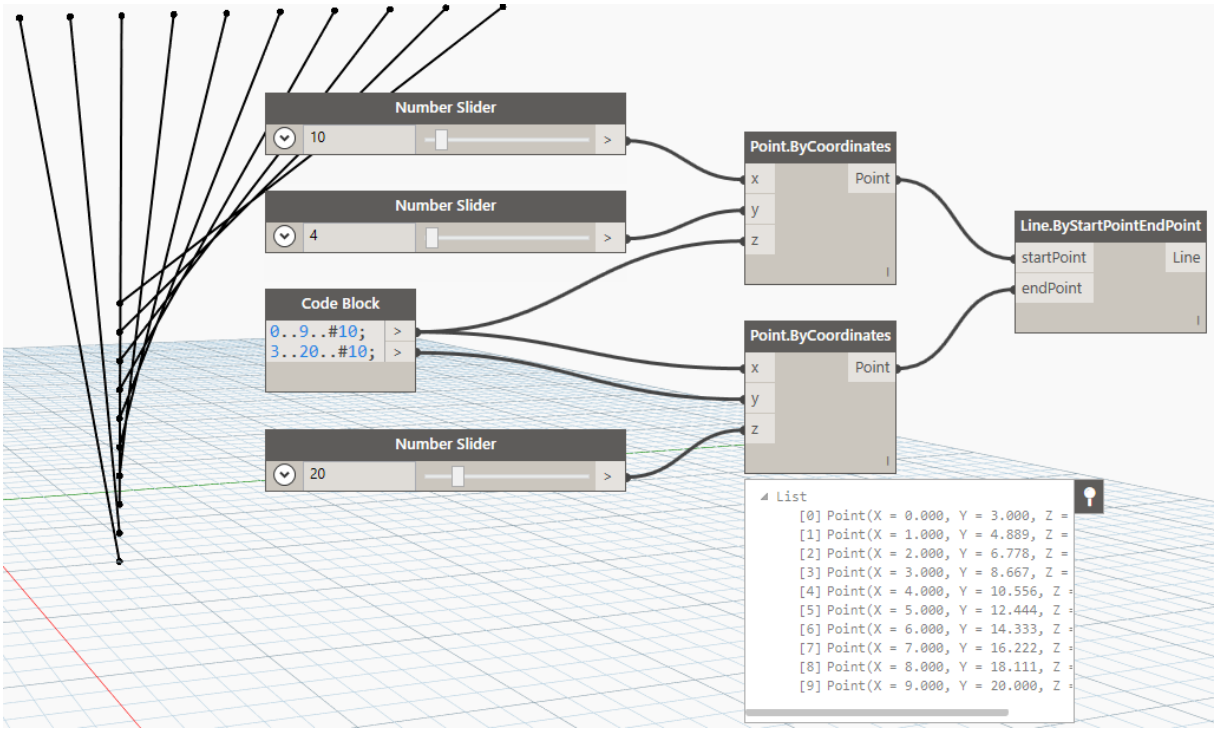

Abbildung 4.2.2: Bildschirmausschnitt aus Dynamo 1.0: Erstellen von mehreren Linien aufgrund von double-Listen

Nachdem so mehrere Linien erzeugt wurden, kann nun die Oberfläche daraus erstellt werden. Mit der Methode *Surface.ByLoft* kann man eine Oberfläche ableiten; die Linien dienen hier als Querschnitte. Das Ergebnis ist in Abbildung 4.2.3 zu sehen:

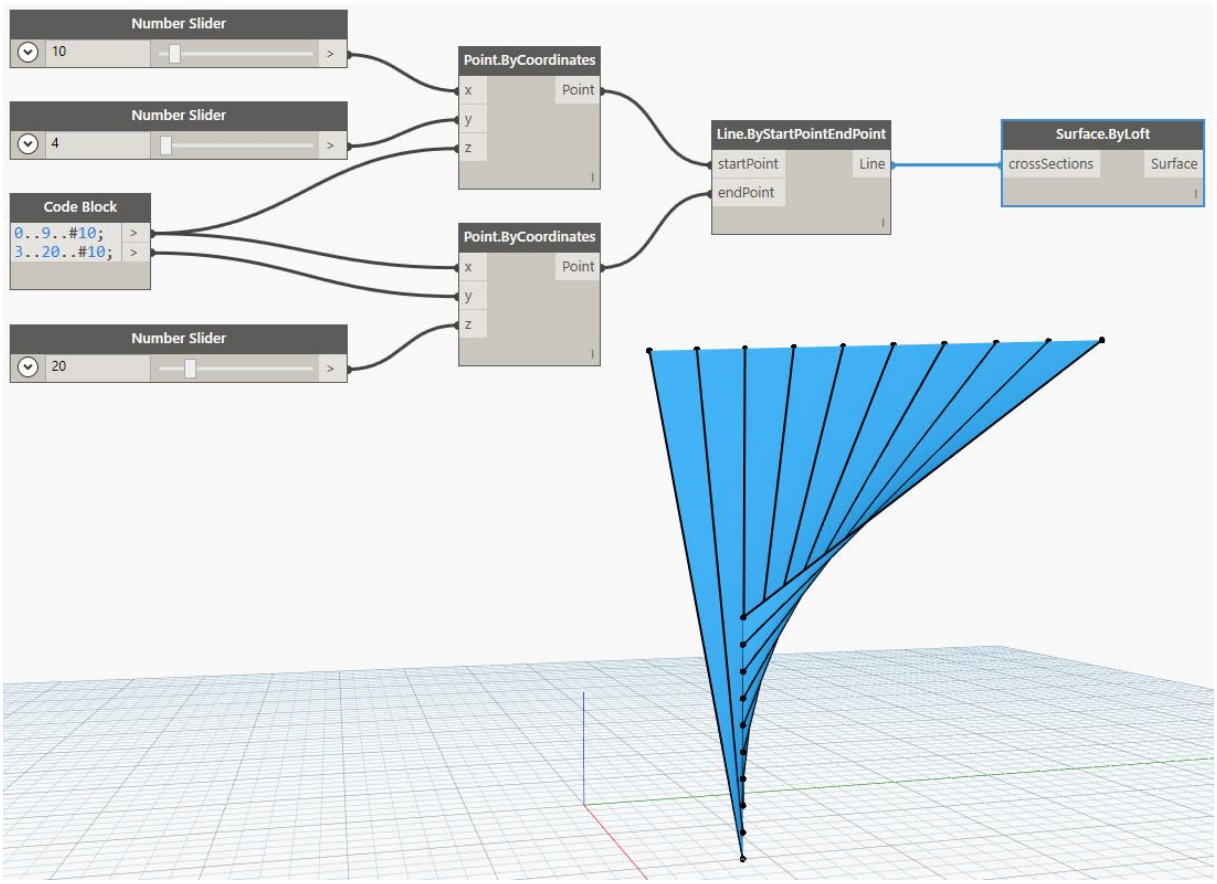

Abbildung 4.2.3: Bildschirmausschnitt aus Dynamo 1.0: Oberfläche, die sich aus mehreren Linien ableitet

Wie bereits gezeigt, lassen sich in Dynamo Objekte auf mehrere Arten erstellen. Durch Verknüpfen der Ein- und Ausgangsparameter der verschiedenen Methoden können Zusammenhänge erstellt werden. Ein sehr nützliches Werkzeug sind die sogenannten *CodeBlock*s. Hier kann der Benutzer Codes eingeben, die von eigens definierten Eingangsparametern abhängen. So kann man beispielsweise den ersten Punkt im vorherigen Beispiel auch textbasiert erstellen. Dies ist sehr ähnlich dem Programmcode in C#. Im *CodeBlock* können auch If-Anweisungen ausgeführt, Rechenoperationen und viele weitere Methoden programmiert werden.

5 Implementierung einer Bogenbrücke

5.1 Grundlagen

Zur Lösung eines definierten Problems werden in der Regel Eingangsparameter mit Hilfe von Formeln in Ausgangsparameter konvertiert. So kann zum Beispiel im Massivbau aus den Eingangsparametern der Belastung, der Abmessungen, Lagerung, etc. die benötigte Bewehrung berechnet werden sowie deren Anordnung. Mit einer entsprechenden Formel kann dann - in Abhängigkeit von mehreren Parametern - das Problem gelöst werden. Man spricht hierbei von einem statischen Modell. Die Eingangsparameter sind bekannt und konstant. Die Gleichung hängt von einer komplexen Funktion ab, aus welcher sich die Ausgangsparameter ergeben. Ein solches Skript zur Erstellung von digitalen Bauwerksmodellen ist bereits vorhanden. Es wurde von Dominic Singer, M.Sc. erstellt Auf dieses Skript wurde zurückgegriffen. Das Skript leitet eine Brücke aus wenigen Parametern ab, welche sich über ein ebenfalls aus Parametern erstelltes Tal erstreckt. Die Brücke ist eine klassische Balkenbrücke, definiert durch die abstrakten Parameter des Benutzers. Nachstehend eine Auswahl von Parametern, um einen Überblick zu geben:

- Brückenlänge
- Talgeometrie
- Anzahl der Pfeiler
- Breite der Fahrbahn
- Überbautyp
- Pfeilertyp

Man sieht sehr schnell, dass dies abstrakte Parameter sind, die auch für einen Laien einfach zu bestimmen sind.

Zur Implementierung wurde das oben erwähnte Skript weiterentwickelt. Ziel war es, das bereits bestehende Skript zur Erstellung von digitalen Bauwerksmodellen um die Möglichkeit einer Bogenbrücke zu erweitern. Diese sollte parallel zur bereits erstellten Brücke konstruiert werden, sodass der Nutzer durch einen einzigen Eingangsparameter zwischen Balken- und Bogenbrücke wählen kann. Einige Funktionen aus dem originalen Skript sind nicht mehr vorhanden, da für diese Arbeit das Skript modifiziert werden musste. Eine Zusammenführung beider Modelle kann jedoch schnell umgesetzt werden.

Tabelle 5.1.1 zeigt alle für das Erstellen der Brücke notwendigen Eingangsparameter, ihren Datentyp, sowie eine kurze Beschreibung der sie beeinflussenden Eigenschaften.

Tabelle 5.1.1: Eingangsparameter für die Erstellung der Bogenbrücke

Bauteil	Parametername	Beschreibung	Datentyp
Geländekurve	d1 bis d12	Höhe der Geländepunkte zum bilden Geländekurve	double
Brückenachse	dstart	Höhe über Gelände am Anfang	double
	dend	Höhe über Gelände am Ende	double
	z	Vertikale Krümmung in den Drittelspunkten	double
	y	Horizontale Krümmung in den Drittelspunkten	double
	x	Länge der Brücke	double
Bogen	CurveTyp	0=Arc, 1=Parabel, 2=Input eigene Kurve aus CSV, 3= Algorithmus zur optimalen Talsuche	int
	DistStart	Abstand zum Anfang in Metern für den Kämpfer des Bogens	double
	DistEnd	Abstand zum End in Metern für den Kämpfer des Bogens	double
	OffsetVertical	Abstand von Oberkante des Überbaus zum Bogen (nur bei manueller Kämpferstandortwahl)	
	AngleCrit	kritischer Winkel für Knickfindung	double
	DistCrit	kritischer Abstand für Punktfindung	double
	ToleranceHeight	erlaubte Höhendifferenz zwischen Kämpfern	double
	CombinationChoice	Auswahl des Bogens, welcher durch die automatische Bogenfindung gesucht wird	int
	MinSpan	Mindest Spannweite des Bogens	double
	MaxSpan	Maximale Spannweite des Bogens	double
	ProfileArc	0 = Vollquerschnitt, 1 = Hohlquerschnitt	int
	VoutedArc	boolean Auswahl; true = ja, false = nein	bool
	ThicknessStart	Falls nicht variabel ist dies die Querschnittshöhe für den Bogen	double
	ThicknessMiddle	Zweite Querschnittshöhe, ob die höhere Stelle in der Mitte oder an den Rändern ist hängt vom stat. System ab	double
	MinRiseSpanRatio	mindest Pfeilverhältnis um einen Bogen zu Erstellen (flachster Bogen)	double
	DistBottomSurf	Abstand von Unterkante des Überbaus zum Bogen	double
Pfeiler	NumberOfPillarsTrans	Anzahl der Pfeiler pro Stütze	int
	MinSpanArc	Mindest Spannweite zwischen Pfeilern auf dem Bogen	double
	MinSpanBefore	Mindest Spannweite zwischen Pfeilern vor dem Bogen	double
	MinSpanAfter	Mindest Spannweite zwischen Pfeilern nach dem Bogen	double
	PillarType	Rechteckige (0) oder runde (1) Pfeiler	int
Widerlager	width	Breite des Widerlagers	double
Gründung	SubstructureType	Art des Unterbaus (0 für Stützen und 1 für Bogen). Wird erst wichtig wenn mehrere Brückenarten implementiert sind.	int
Überbau	StartCrossSlope	Querneigung der Fahrbahn am Anfang der Brücke	double
	EndCrossSlope	Querneigung der Fahrbahn am Ende der Brücke	double
	SuperstructureType	0 = Massives Rechteck, 1 = Plattenbalken, 2 = Hohlkasten	int
	NumberOfWorkplanes	Anzahl an verwendeten Arbeitsebene entlang der Achse	int
	Height	Höhe des Überbaus	double
	Width	Weite des Überbaus	double
	VoutePercentage	Voutung in längsrichtung des Überbaus	double

5.2 Implementierung des Bogens

In einem ersten Schritt wurde der Bogen anhand von benutzerdefinierten Parametern bestimmt. Der Benutzer gibt den Abstand der Kämpferstandorte zum Start und zum Ende an, danach wählt er zwischen einem klassischen Kreisbogen, einer Parabel oder einer benutzerdefinierten Kurve. Da dem Benutzer möglichst viele Freiheiten gegeben werden sollten und eine Brücke aufgrund der wechselnden Verkehrslast nie eine ideale Stützlinie für alle Lastfälle hat, war es wichtig, auch eine benutzerdefinierte Kurve einlesen zu können. Oftmals wird die endgültige Bogenkurve numerisch berechnet und ist nicht durch eine einfache quadratische Parabel zu lösen.

Hat man die Bogenkurve definiert, kann man entlang dieser die Querschnitte anordnen und so beispielsweise zwischen Voll- und Kastenquerschnitt unterscheiden. Anschließend wird anhand dieser Querschnitte der Bogen in Revit als Familie und in Dynamo als sogenannter *Solid* erstellt. *Solid*s sind geometrische Objekte, die einen dreidimensionalen Körper darstellen.

In der Literatur zum Bau von Bogenbrücken aus Beton finden sich einige Regeln zur Wahl des Querschnitts. Legt man eine dieser Quellen als Grundlage für die Konstruktionsregeln fest, so kann man hieraus alle Parameter ableiten, da diese oft voneinander abhängen (z.B. „statisches System hängt von Pfeilverhältnis ab“, …). Für einige Parameter wurde dies auf Grundlage von (Mehlhorn und Curbach 2014) implementiert. Entsprechende Implementierung werden bei ihrer Verwendung beschrieben, sie beziehen sich nicht nur auf den Querschnitt des Bogens.

5.3 Intelligente Bogensuche

Da das Skript am Ende auch von Nicht-Experten genutzt werden kann, ist die automatische Bogenfindung ein wichtiger Teil. Hierfür ist es notwendig, einen Algorithmus zu schreiben der in der Lage ist, die Geländekurve zu analysieren und geeignete Stellen zu finden. Im Anschluss sollen aus diesen Hoch-, Tief- und Knickpunkten den Kriterien entsprechende Täler und Bögen konstruiert werden.

Damit man die Kämpferstandorte bestimmen kann, muss die Geländekurve zunächst in einer Liste von Punkten abgebildet werden. Dafür werden Punkte im Abstand von einem Meter entlang der Geländekurve bestimmt. Die Distanz wird dabei nicht entlang der Kurve, sondern in der Projektion auf die x-y-Ebene gemessen. Hierfür wurde der

Knoten *BridgeClass.SchnittpunktAlle1* in Visual Studio programmiert. Wie der Name bereits andeutet, wird die Geländekurve entlang der Geraden zwischen ihrem Start- und Endpunkt mit mehreren vertikalen Ebenen geschnitten. Die Ausgabe ist eine Liste von Punkten, welche alle auf der eingegebenen Kurve liegen und den Sehnenlängenabstand von eins haben. Der Parameter wurde mit eins gewählt (eins entspricht 1 Meter), da die Eingabe aus dem Geländemodell womöglich minimale Kuppen und Senkungen aufweist, welche für die Bogenanalyse einer Brücke irrelevant sind.

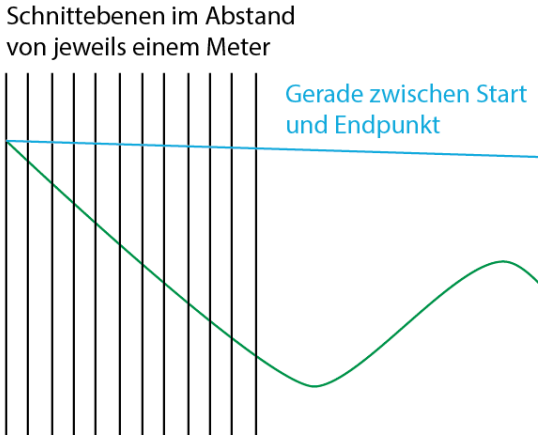

Abbildung 5.3.1: Visualisierung der Verschneidung von Schnittebene und Geländekurve

5.3.1 Hoch-, Tief- und Knickpunkte

Die zuvor erstellte Punkteliste wird anschließend weiterverwendet. Um die kritischen Punkte zu finden, wurde der Knoten *BridgeClass.FindPOI* in Visual Studio programmiert. Er benötigt die Punkte entlang der Geländekurve [points], den kritischen Winkel [anglecrit] sowie die kritische Distanz [distcrit] als Eingangsparameter.

Zunächst werden alle Hoch- und Tiefpunkte entlang der Geländekurve in einer Liste gespeichert. Sind Vor- und Nachgänger entweder beide höher oder tiefer als der mittlere Punkt, dann wird der mittlere Punkt in der Liste hinzugefügt. Diese Iteration läuft einmal über die gesamte Liste der Eingangspunkte.

Anschließend wird jeweils ein Vektor mit dem Vorgängerpunkt und einer mit dem Nachgängerpunkt in der Liste gebildet. Es wird der Winkel zwischen beiden Vektoren berechnet und mit dem kritischen Winkel verglichen. Der kritische Winkel definiert, ab wann eine Krümmungsänderung als Knick angesehen wird, da es Situationen gibt, in denen das Gelände zwar an einem Punkt einen Knick macht, der als Kämpferstandort in Frage kommt, jedoch kein Hoch- oder Tiefpunkt ist. Ein Beispiel für einen solchen Punkt sind beide Punkte in Abbildung 5.3.2 oder Punkt 6 in Abbildung 5.3.5.

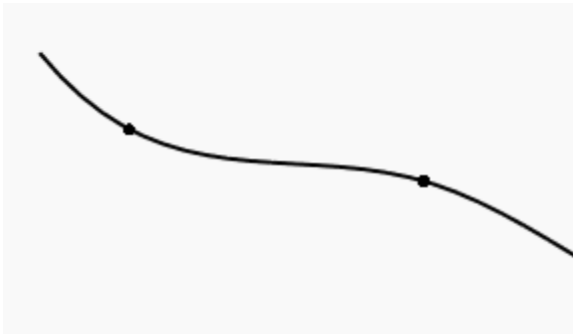

Abbildung 5.3.2: Bildschirmausschnitt aus Dynamo 1.0

Da man nicht auf die mathematische Lösung zurückgreifen kann, mittels derer man durch Nullsetzen der zweiten Ableitung der Funktion Hoch- und Tiefpunkte entlang einer Kurve zu finden, wurde die Lösung mit dem kritischen Winkel für das Skript implementiert. Der kritische Winkel ist leider eine nicht optimale Lösung des Problems. Die Praxis hat ergeben, dass es sinnvoll ist, einen kritischen Winkel von 1 festzulegen. Da die Methode des kritischen Winkels sehr oft mehrere Punkte entlang einer Krümmung findet und eine exakte Annäherung an den kritischen Winkel sehr schwierig ist,

wurde in einem zweiten Schritt in einer weiteren do-while-Schleife, jeweils der Mittelpunkt von einer Serie von Punkten ausgewählt.

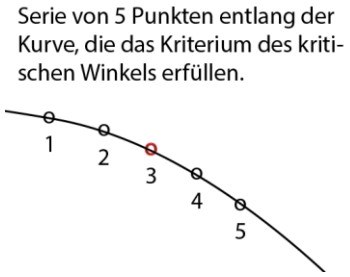

Abbildung 5.3.3: 1. Schritt: Bestimmen der Punkte, 2. Schritt: Auswählen des Mittelpunktes (rot)

Hierfür wird zunächst der Abstand zum nächsten Punkt in der im ersten Schritt erhaltenen Liste gemessen. Ist dieser kleiner als 2m, wird eine temporäre Liste erstellt und alle folgenden Punkte im Abstand von unter 2m werden hinzugefügt. Am Ende wird aus der temporären Liste der Punkt in der Mitte ausgewählt. Sollte nur ein Punkt an einem Knick gefunden werden, wird dieser direkt weiterverarbeitet.

Des Weiteren muss geprüft werden ob der Knick nicht zu nah an einem Hoch oder Tiefpunkt liegt. Der Eingangsparameter der kritischen Distanz legt diesen Abstand fest. Hier hat sich ein Abstand von 20m zu benachbarten Punkten bewährt. Dieser kann jedoch vom Benutzer frei verändert werden, sollte das Programm unpassende Ergebnisse liefern.

Zum Schluss werden noch Start- und Endpunkt hinzugefügt und alle Punkte mit Hilfe des Bubble-Sort Algorithmus entlang der Achse geordnet. Der Bubble-Sort-Algorithmus sortiert eine Liste von Elementen durch zwei gegenläufige Schleifen. Die daraus erstellte endgültige Liste wird dann in Dynamo zurückgeben.

```
Finden der Hoch-, Tief- und Knickpunkte

START
Input: Liste der Punkte entlang der Geländekurve, kritischer Winkel, kritische
Distanz

WHILE i < Länge der Liste der Punkte der Geländekurve – 2
       IF Vor- und Nachgänger höher als Mittelpunkt
              Mittelpunkt zur Ausgabeliste hinzufügen
       ELSE Vor- und Nachgänger tiefer als Mittelpunkt
              Mittelpunkt zur Ausgabeliste hinzufügen

WHILE i < Länge der Liste der Punkte der Geländekurve – 2
       Tangente bilden zwischen Punkt 0 und 1 sowie 1 und 2
       Winkel zwischen beiden Tangenten berechnen
       IF Winkel > kritischer Winkel
              Mittelpunkt zu Liste der Knickpunkte hinzufügen

WHILE i < Länge der Liste der Knickpunkte - 1
       Abstand von Punkt i zu Punkt i+1 berechnen
       IF Abstand < 2m
              Punkt i zu neuer temporärer Liste hinzufügen
              WHILE Abstand von Punkt i+1 zu Punkt i+2 < 2m
                     Punkte zu temporärer Liste hinzufügen
              IF zwei Punkte in temporärer Liste
                     zweiter Punkt wird ausgewählt
                     IF Abstand von zweitem Punkt zu jedem Punkt in der
                        Ausgabeliste > kritische Distanz
                            zweiter Punkt in die Ausgabeliste
              ELSE mehr als zwei Punkte in temporärer Liste
                     Mittelpunkt von allen Punkten wird ausgewählt
                     IF Abstand Mittelpunkt zu jedem Punkt in der
                        Ausgabeliste > kritische Distanz
                            Mittelpunkt in die Ausgabeliste
       ELSE Abstand > 2m
              IF Abstand von Punkt i zu jedem Punkt in der
                 Ausgabeliste > kritische Distanz
                     Punkt i in die Ausgabeliste

Hinzufügen von erstem und letztem Punkt der Geländekurve

Sortieren der Punkte in der Ausgabeliste nach Ort auf der Geländekurve mit Bubble-
Sort Algorithmus

Rückgabe der Ausgabeliste
END
```

Abbildung 5.3.4: Pseudocode zur Findung der Hoch-, Tief- und Knickpunkte entlang der Geländekurve

5.3.2 Bestimmung der Kämpferstandorte

Mit den im vorherigen Knoten beschriebenen Punkten werden in einem zweiten Knoten die Kämpferstandorte ermittelt und als Kombinationen zurückgegeben. In Abbildung 5.3.4 findet sich ein Überblick. Der Algorithmus *BridgeClass.FindStartEndSegment* liefert die Kombinationen. Die Form der Bögen wird im Knoten *BridgeClass.FindArc* bestimmt und danach wird mit *BridgeClass.SortParabolasByCombinationToNurbsCurve* dem Benutzer die Möglichkeit gegeben, eine Kombination auszuwählen. Kombination kann in diesem Fall ein einzelnes Tal mit entsprechendem Bogen sein oder, wie Tal 5 in Abbildung 5.3.5, eine Kombination von zwei Tälern, in deren Mitte auf dem Hochpunkt ein Kämpfer für Bögen über beide Täler steht.

Die Möglichkeit dem Benutzer ein Tal zur Wahl zu stellen, erfordert mindestens eine Iteration, da die möglichen Kombinationen erst nach dem ersten Durchlauf zu sehen sind. Hier besteht Optimierungsbedarf, jedoch würde dies den Umfang dieser Arbeit übersteigen. Des Weiteren ist der Benutzer in der Regel planender Ingenieur, das heißt er kann mit seinem Fachwissen aus den Vorschlägen einen passenden aussuchen und erhält dann mit einem weiteren Durchlauf das gewünschte Modell.

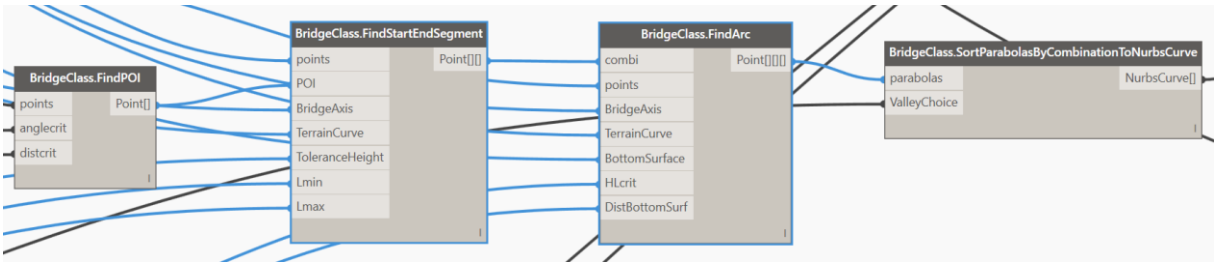

Abbildung 5.3.5: Bildschirmausschnitt aus Dynamo 1.0 mit den Knoten zur Ermittlung der Hoch-, Tief- und Knickpunkte, der Start- und Endsegmente und Auswahl der Kombination

Der Knoten *BridgeClass.FindStartEndSegment* benötigt die folgenden Eingabeparameter:

Tabelle 5.3.1: Eingangsparameter für den Knoten BridgeClass.FindStartEndSegment

Parametername	**Beschreibung**
points	Punkte entlang der Geländekurve
POI	Hoch-, Tief- und Knickpunkte
BridgeAxis	Brückenachse
TerrainCurve	Geländekurve
ToleranceHeight	Toleranzwert der Differenz zwischen Start- und Endpunkt
Lmin]	Minimale Bogenlänge
Lmax	Maximale Bogenlänge

Alle in blau gedruckten Eingabeparameter sind Werte, die der Benutzer nach seinem Belieben festlegen kann und sollte. Die anderen Werte leiten sich aus dem Modell ab.

Zu Beginn werden die Täler an sich definiert. Dazu werden die Hoch-, Tief- und Knickpunkte (POI) untersucht. Sind Vor- und Nachgängerpunkt höher als der Mittelpunkt, ist es logischerweise ein Tal. Anschließend wird überprüft, ob das Tal den Kriterien des Benutzers entspricht. Der Abstand der beiden Hochpunkte muss größer als Lmin und kleiner als Lmax sein. Stimmen alle diese Kriterien überein, wird mit dem gefundenen Tal eine neue Liste erstellt. Das Tal bekommt überdies eine Identifikationsnummer. Diese ist wichtig, um später verschiedene Täler unterscheiden zu können.

Nachdem ein Tal gefunden ist, überprüft der Algorithmus erneut, ob Vor- oder Nachgänger nochmals höher als die schon vorhandenen Hochpunkte sind. Am besten lässt sich dies an einem Beispiel erklären. Das Beispiel bezieht sich auf Abbildung 5.3.5. Tal 3 wird in erster Instanz gefunden. Die Punkte 6 und 8 sind beide höher als 7, und der Mindest- und Maximalabstand werden ebenfalls eingehalten. Nun wird überprüft, ob der Vorgänger oder Nachgänger nochmals höher sind. Punkt 5 ist höher als Punkt 6. Daraus folgt, dass es eine weitere Kombination eines Tals aus den Punkten 5, 7 und 8 gibt. Dieses Tal wird als Tal 4 gespeichert. Es bekommt dieselbe Identifikationsnummer wie Tal 3 zugeordnet.

Wenn der Algorithmus den letzten Punkt erreicht hat, beginnt er die Tiefpunkte zu löschen. Dadurch ergeben sich neue Täler. Im Beispiel ist dies in der 1. Iteration die rote Linie und in der 2. Iteration die orange Linie. Mit diesen künstlichen Geländekurven

wird wieder analog verfahren. Es ergeben sich zunächst die Täler 5 und 6. Nach der 2. Iteration ergibt sich noch Tal 7.

Da es auch möglich ist einen Doppelbogen zu konstruieren, wird in einem weiteren Schritt nach Tälern mit unterschiedlichen Identifikationsnummern und gleichen Punkten gesucht. Da alle Täler, die sich aus einer Situation wie Tal 3 und 4 ergeben, identische Identifikationsnummern haben, kann es nicht zu Doppelbögen über ein gleiches Tal kommen, da Tal 3 und 4 mit Punkt 8 einen gemeinsamen Punkt hätten.

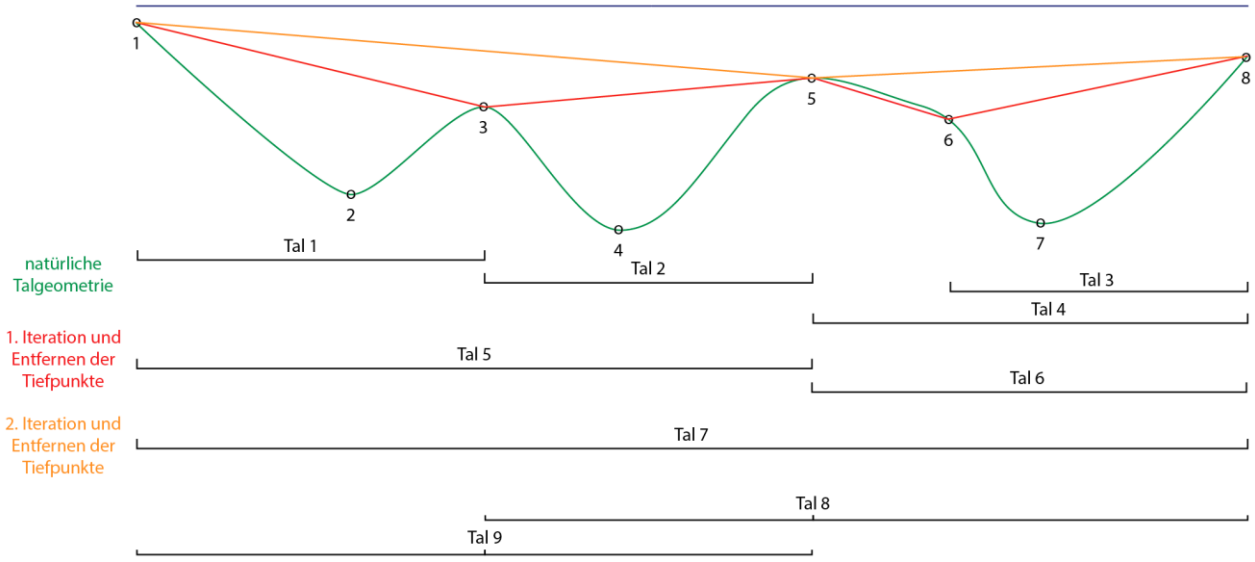

Abbildung 5.3.6: Visualisierung des Algorithmus zur Ermittlung der Täler (die rote und orange Linie stellen das fiktive Tal dar, welches der Algorithmus durch das Löschen der Tiefpunkte findet)

Findet der Algorithmus ein Tal mit entweder gleichem Start und Endpunkt oder umgekehrt, so sucht er direkt nach den Kämpferstandorten. Als erster Kämpferstandort wird der Mittelpunkt gewählt. Im Beispiel wären dies für Tal 8 der Punkt 5 und für Tal 9 der Punkt 3. Das Programm sucht nun mit Hilfe einer Geraden rechts und links nach Schnittpunkten (Abbildung 5.3.6). Die Schnittpunkte werden als Kämpferstandorte zusammen mit dem Mittelpunkt in einer Liste gespeichert.

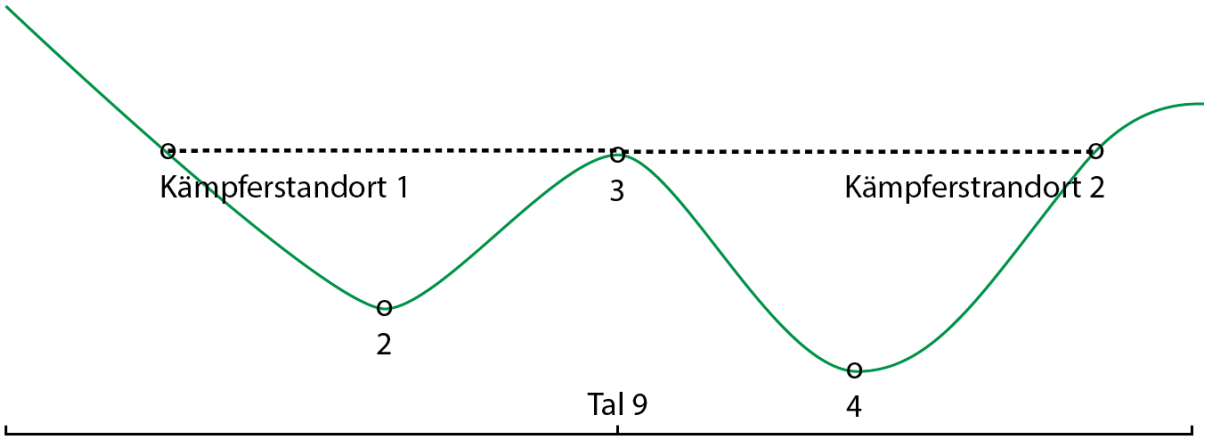

Abbildung 5.3.7: Visualisierung des Algorithmus bei Doppelbögen

Wurden alle Kombinationen mit einem Doppelbogen bearbeitet, fährt der Algorithmus mit den zuvor gefundenen Einzeltälern fort. Auf Grund der hohen Kraft entlang der Stützlinie ist es sinnvoll, möglichst senkrecht in den Boden zu gründen. Dies führt sowohl zu einer Stabilisierung des Bodens als auch zu kleineren horizontalen Kräften, die verankert werden müssen. Um bei der späteren Bogenoptimierung möglichst ideale Verhältnisse vorzufinden, wird nach der steilsten Stelle innerhalb der ersten Hälfte zwischen Hoch- und Tiefpunkt eines Tals gesucht. Hierbei wird das Raster der Vektoren zwischen zwei Punkten auf 5m erhöht. Dadurch ist gesichert, dass kleine Unebenheiten beim Vermessen des Geländes keine Fehler erzeugen können und der grobe Geländeverlauf untersucht wird.

Im Beispiel von Tal 1 in Abbildung 5.3.7 liegen die Kämpferstandorte auf unterschiedlichen Höhen. Diese Differenz kann durch den Eingabeparameter *ToleranceHeight* beschränkt werden. Eine zu große Differenz kann sowohl unästhetisch wirken als auch zu Problemen bei der Bogenfindung führen, da der Mittelpunkt bei größerer Differenz weiter vom höchsten Punkt entfernt liegt. Bei der Bildung der Parabel durch drei Punkte ist der Mittelpunkt demnach nicht der höchste Punkt. Es muss also eine weitere Überprüfung durchgeführt werden, ob der Bogen an einem Punkt höher ist als der erlaubte Grenzwert. Ist dies der Fall, ist er ebenfalls ungültig. Wird *ToleranceHeight* mit 0 angesetzt, müssen sich beide Kämpferstandorte auf derselben Höhe befinden.

Sollten die Punkte nicht das Kriterium der Höhendifferenz erfüllen, so wird vom kleineren Punkt aus eine Gerade gebildet und mit dem Gelände der Schnittpunkt gesucht.

Hier müssen wieder die Kriterien Lmin und Lmax für die Länge des Bogens eingehalten werden.

Sind die steilsten Stellen gefunden, werden die beiden Standorte der Kämpfer ebenfalls in einer Liste gespeichert. Diese Ausgabe, bestehend aus Doppel- und Einzelbogentälern, wird für die Findung der idealen Bögen benutzt.

```
Finden der Kämpferstandorte für den Bogen

START
Input: Liste der Punkte entlang der Geländekurve (PG), Liste der Hoch-, Tief- und
Knickpunkte (POI), Brückenachse, Terrainachse, maximale Höhendifferenz der Kämp-
ferstandorte, maximale und minimale Bogenlänge

FOR i=0 TO Länge von POI – 2, i++
       IF Vor- und Nachgänger höher als Mittelpunkt
              IF Tallänge innerhalb von minimaler und maximaler Bogenlänge
                     Füge Tal zur Talliste hinzu

              //Überprüfung ob Nachbarpunkte des Tals nochmals höher sind
              FOR j=i TO 1, j—-
                     IF Punkt vor dem Tal höher als Randpunkte des Tals
                            Füge Tal zur Talliste hinzu
              FOR j=i+2; TO Länge von POI – 1, j++
                     IF Punkt nach dem Tal höher als Randpunkte des Tals
                            Füge Tal zur Talliste hinzu

//Tiefpunkt löschen um neue Talvarianten zu erhalten (größere Täler)
WHILE Tiefpunkte gelöscht werden können
       FOR i=0 TO Länge von POI – 2, i++
              IF Vor- und Nachgänger höher als Mittelpunkt
                     Mittelpunkt (Tiefpunkt) löschen
       //Tiefpunkt ist aus POI Liste gelöscht, es ergeben sich größere Täler
       FOR i=0 TO Länge von POI -2, i++
              IF Vor- und Nachgänger höher als Mittelpunkt
                     IF Tallänge innerhalb von minimaler und maximaler Bogenlänge
                            Füge Tal zur Talliste hinzu

              //Überprüfung ob Nachbarpunkte des Tals nochmals höher sind
              FOR j=i TO 1, j—-
                     IF Punkt vor dem Tal höher als Randpunkte des Tals
                            Füge Tal zur Talliste hinzu
              FOR j=i+2; TO Länge von POI – 1, j++
                     IF Punkt nach dem Tal höher als Randpunkte des Tals
                            Füge Tal zur Talliste hinzu
```

```
//Überprüfen ob Randpunkte der Täler in der Talliste identisch sind
IF Randpunkte identisch mit anderem Tal in der Talliste
       Erstelle Gerade von identischem Punkt aus in beide Richtungen und finde
       in einem Abstand größer als der minimalen Bogenlänge einen Schnitt-
       punkt mit der Geländekurve

       Füge Kombination aus erstem Schnittpunkt, identischem Punkt und zweitem
       Schnittpunkt zur Ausgabeliste hinzu

FOR i=0 TO Länge der Talliste, i++
       Erstelle entlang der Geländekurve Tangenten zwischen Punkten im
       Abstand von 5m

       Berechne Winkel zur x-y-Ebene der Tangenten und erstelle Liste mit Winkeln

       FOR Ort des Startpunkts des Tals TO Ort des Mittelpunkts des Tals
              Finde steilsten Winkel
              IF steilster Winkel innerhalb der ersten Hälfte der Strecke
                 von Start- zu Mittelpunkt
                     Anfangskämpferstandort gefunden
       FOR Ort des Mittelpunkts des Tals TO Ort des Endpunkts des Tals
              Finde steilsten Winkel
              IF steilster Winkel innerhalb der ersten Hälfte der Strecke
                 von Mittel- zu Endpunkt
                     Endkämpferstandort gefunden

       IF Höhendifferenz zwischen Anfangs- und Endkämpferstandort
          > maximale Höhendifferenz der Kämpferstandorte
              IF Anfangskämpferstandort höher als Endkämpferstandort
                     Erstelle Gerade von Anfangskämpferstandort in Richtung
                     Endkämpferstandort

                     Schnittpunkt mit Geländekurve ist neuer Endkämpferstandort

              ELSE IF Endkämpferstandort höher als Anfangskämpferstandort
                     Erstelle Gerade von Endkämpferstandort in Richtung
                     Anfangskämpferstandort

                     Schnittpunkt mit Geländekurve ist neuer Anfangskämpferstand-
                     ort

       Füge Kombination aus Anfangs- und Endkämpferstandort zur Ausgabeliste hinzu

Rückgabe der Ausgabeliste
END
```

Abbildung 5.3.8: Pseudocode zur Findung der Kämpferstandorte

5.3.3 Bestimmung der Bogenform

Wie bereits beschrieben ist ein Kreuzungswinkel von Bogenachse und Geländekurve von 90 Grad ideal. Da aus dem Bogen fast ausschließlich Normalkräfte übertragen werden, können diese Kräfte leicht in den Boden abgeleitet werden. Jede Kombination besteht aus einer Unterliste, welche die Kämpferstandorte als Punkte führt. Mit der Ausgabe von *BridgeClass.FindStartEndSegment* wird im nächsten Schritt ein geeigneter Standort für die Kämpfer gesucht. Hierfür wurde der Knoten *BridgeClass.FindArc* programmiert. Er benötigt folgende Eingabeparameter:

Tabelle 5.3.2: Eingangsparameter für den Knoten BridgeClass.FindArc

Parametername	**Beschreibung**
points	Punkte entlang der Geländekurve
POI	Hoch-, Tief- und Knickpunkte
BridgeAxis	Brückenachse
TerrainCurve	Geländekurve
BottomSurface	untere Fläche des Überbaus
HLcrit	kritisches Pfeilverhältnis
DistBottomSurf	Abstand von Fahrbahnoberfläche zum höchsten Punkt des Bogens

Blaue Schrift steht wieder für benutzerdefinierte Eingaben.

Mit Hilfe der Funktion *ScheitelpunktParabel* kann eine quadratische Parabel durch drei Punkte definiert werden. Start- und Endpunkt der Parabel sind bereits bekannt, es sind die Kämpferstandorte. Der mittlere Punkt soll in der Mitte von beiden Punkten liegen. Die Höhe wird iterativ ermittelt. Hier kommt das Pfeilverhältnis ins Spiel. Der Benutzer hat angegeben, ab welchem Mindestpfeilverhältnis er einen Bogen erzeugen lassen will. In Kapitel 2 wurde die Abhängigkeit von Pfeilverhältnis und statischem System erläutert. Der Algorithmus erkennt diese Abhängigkeit. So wird der Querschnitt, wenn er variabel konstruiert werden soll, je nach statischem System zum Kämpfer hin dünner oder dicker. Der erste Mittelpunkt hat folgende Koordinaten: x-Wert = der Mittelwert aus beiden Kämpferstandorten, y-Wert = ebenfalls der Mittelwert aus beiden Kämpferstandorten und z-Wert = der Mittelwert der Höhe von beiden Kämpferstandorten.

Aus diesen drei Punkten wird die Parabel gebildet. Jetzt müssen zwei Kriterien erfüllt oder teilweise erfüllt sein. Als erstes muss das Pfeilverhältnis das definierte Mindestpfeilverhältnis übersteigen. Zweitens muss entweder der rechte oder der linke Kreuzungswinkel mit dem Gelände im Bereich zwischen 80 und 100 Grad sein. Somit wird

ein möglichst orthogonaler Schnitt mit dem Gelände garantiert. Wird eines dieser Kriterien nicht erfüllt, wird die z-Koordinate des Mittelpunkts um einen Meter nach oben gesetzt. Abbildung 5.3.7 veranschaulicht das Vorgehen. Diese Iteration wird so lange durchgeführt, bis entweder ein passender Bogen gefunden oder die Obergrenze erreicht wurde. Die Obergrenze ist die Höhe der Brückenachse an der Stelle des Mittelpunkts, reduziert um den Wert *DistBottomSurf*, der die Höhe des Überbaus und weiteren Abstands des Bogens zur Fahrbahn beschreibt. Wird die Obergrenze erreicht, bricht die Iteration ab und in diesem Tal kann kein Bogen nach den benutzerdefinierten Kriterien gefunden werden.

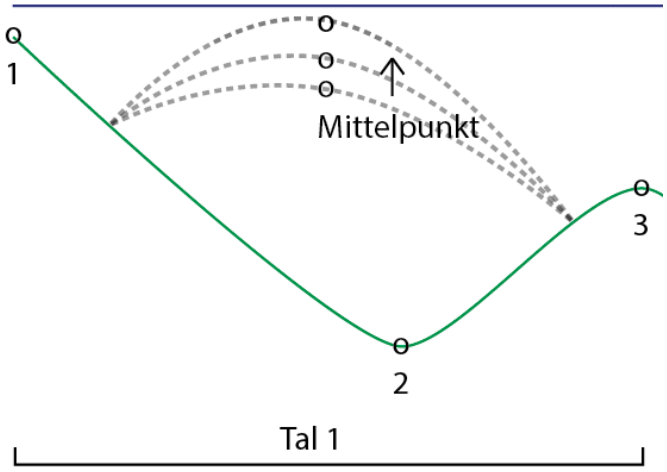

Abbildung 5.3.9: Visualisierung der Bogenfindung

Wurden alle möglichen Bögen gefunden, wird diese Ausgabe im letzten Knoten wieder weiterverarbeitet. Er dient lediglich dazu, eine Variante auszuwählen. Wie bereits erwähnt, wird der Anwender zu einer Iteration gezwungen. Im ersten Durchlauf werden alle Möglichkeiten visualisiert. Danach kann der Anwender hieraus eine passende auswählen und ändert so das Modell in Revit.

Der Knoten hierzu heißt: *BridgeClass.SortParabolasByCombinationToNurbsCurve*. Diese Implementierung ist allerdings noch sehr verbesserungswürdig. Da Dynamo zunächst alles rechnet und dann die Varianten visualisiert, ist eine vorherige Auswahl nicht möglich. Außerdem wird die Kombination anhand einer Nummer gewählt. Es lässt sich im Vorfeld nur schwer erkennen, welche Kombinationsnummer zu welcher Kurve gehört. Wird eine Kombinationsnummer ausgewählt, die nicht existiert, so gibt

der Knoten eine leere Liste zurück und erzeugt einen Fehler im folgenden Programmverlauf. Leider war es im zeitlichen Rahmen der Arbeit nicht möglich, hier eine bessere Lösung zu finden.

Bogenfindung

```
START
Input: Liste der Kombinationen der Kämperstandorte (KK), Liste der Punkte entlang der
Geländekurve (PG), Brückenachse, Geländekurve, untere Fläche des Oberbaus, minimales
Pfeilverhältnis, Abstand zur unteren Fläche des Oberbaus

FOR i=0 TO Länge der Liste KK, i++
       IF kein Doppelbogen
              Bestimme minimale Höhe des Mittelpunkts durch Berechnung des
              Mittelwerts der Höhe der Kämpferstandort + 1

              Bestimme maximale Höhe des Mittelpunkts durch Berechnung der
              Höhe der unteren Fläche des Oberbaus am Mittelpunkt – Abstand zur
              unteren Fläche des Oberbaus

              FOR q=minimale Höhe des Mittelpunkts TO maximale Höhe des Mittelpunkts,
                  q=+0.5
                     Erstelle Parabel durch beide Kämpferstandorte und Mittelpunkt
                     mit der Höhe q anhand der Formel zur Berechnung einer Parabel
                     durch drei Punkte

                     Berechne den Schnittwinkel mit dem Gelände auf beiden Seiten

                     IF Schnittwinkel auf einer Seite im Bereich zwischen 80 und 100
                        Grad und Pfeilverhältnis > minimales Pfeilverhältnis und der
                        Bogen schneidet die Geländekurve nur in den Kämpferstandorten

                            Füge Bogen zur Ausgabeliste hinzu

       ELSE IF Doppelbogen
              Bestimme minimale Höhe des Mittelpunkts für beide Bögen durch Berechnung
              des Mittelwerts der Höhe der Kämpferstandort + 1

              Bestimme maximale Höhe des Mittelpunkts für beide Bögen durch Berechnung
              der Höhe der unteren Fläche des Oberbaus am Mittelpunkt – Abstand zur
              unteren Fläche des Oberbaus
```

```
            FOR q=minimale Höhe des Mittelpunkts des ersten Bogens TO maximale
                Höhe des Mittelpunkts des ersten Bogens, q=+0.5

                   Erstelle Parabel durch beide Kämpferstandorte und Mittelpunkt mit
                   der Höhe q anhand der Formel zur Berechnung einer Parabel durch
                   drei Punkte jeweils für beide Bögen

                   Berechne Schnittwinkel mit dem Gelände auf beiden Seiten und
                   zwischen den beiden Bögen am mittleren Kämpferstandort

                   IF Schnittwinkel eines Bogens zum Gelände im Bereich zwischen 80
                   und 100 Grad und das Pfeilverhältnis dieses Bogens > minimales
                   Pfeilverhältnis und dieser Bogen die Geländekurve nur in den
                   Kämpferstandorten schneidet

                          Berechnen des anderen Bogens bis er ebenfalls die
                          Kriterien erfüllt

                          Füge beide Bögen zur Ausgabeliste hinzu

Rückgabe der Ausgabeliste
END
```

Abbildung 5.3.10: Pseudocode zur Bogenfindung

5.4 Auftretende Probleme beim Import von Dynamo nach Revit

Da Revit und Dynamo unabhängig voneinander entwickelt wurden und Autodesk erst später in die Entwicklung einstieg, sind die Programme noch nicht perfekt aufeinander abgestimmt. Dies macht sich besonders beim Import der erstellten Objekte bemerkbar. Wird Dynamo über Revit gestartet, erscheint die Kategorie Revit ebenfalls in Dynamo. Sie beinhaltet Knoten zum Import, zur Auswahl von Objekten in Revit und weitere Knoten, welche die Revit API nutzen. Mit dem Knoten *ImportInstance* können Körper aus Dynamo (Solids) nach Revit importiert werden. Problematisch hierbei ist, dass die Geometrie von Revit nicht korrekt erkannt wird. So fehlen Informationen über das Volumen sowie zusätzliche Attribute, wie zum Beispiel die Zuweisung eines Materials. Gerade für die weitere Einbindung als BIM ist es daher wichtig, diese Attribute verändern zu können.

Es gibt ebenfalls die Möglichkeit, Objekte als sogenannte *DirectShapes* zu importieren. Doch auch dieser Lösungsansatz bringt keine befriedigenden Ergebnisse. Daher wird er nicht weiter beschrieben.

Für diese Arbeit wurde auf eine andere Methode zum Import zurückgegriffen. Hierfür werden mit Hilfe der Querschnitte, sogenannte *LoftInfos* erstellt. Darin sind neben den zu verbindenden Querschnitten auch Material und Name enthalten. In Revit werden die Querschnitte dann zu einem Körper extrudiert. Dies geschieht jeweils von einem zum nächsten Querschnitt, wodurch auch der Bogen mit Hilfe von mehreren Querschnitten gut erstellt werden kann. Mit dieser Methode wird derselbe Modellierungsschritt aus Dynamo (*Solid.ByLoft*) in Revit durchgeführt. Gesteuert wird er aus Dynamo heraus. Diese Methode wurde am Lehrstuhl für Computergestützte Modellierung und Simulation der TU München entwickelt.

In der Abbildung 5.4.1 sind die Unterschiede zwischen den importierten Objekten zu sehen. Rechts ist ein Objekt, das mit Hilfe von *ImportInstance* importiert wurde und links eines, welches mit Hilfe von *LoftInfos* importiert wurde.

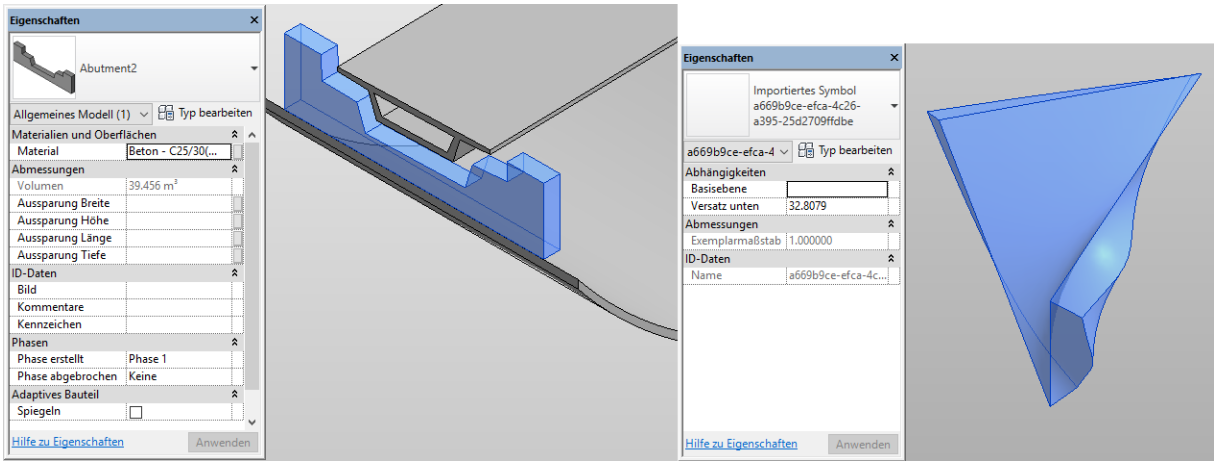

Abbildung 5.4.1: Links Import mit *LoftInfos*, rechts Import mit *ImportInstance*

Aufgrund dieser Problematik beim Import der Objekte werden die effektiven Körper erst in Revit erzeugt. Dynamo liefert lediglich die Querschnitte, aus denen sich die Körper ableiten. Sämtliche Verschneidungen müssen vom Nutzer anschließend in Revit durchgeführt werden. Auch hier ist noch großes Verbesserungspotential, um eine nutzerfreundlichere und schnellere Lösung zu finden.

5.5 Ergebnis und Ausblick

Am Ende erhält man in Revit ein BIM-Modell der Brücke. Dadurch ist es möglich, weitere Untersuchungen an diesem Modell zu machen. Beispielsweise könnten FEM-Berechnungen am selben Modell durchgeführt werden.

Zur besseren Visualisierung des Tals wurde ebenfalls die Geländekurve importiert. In Zukunft sollte das gesamte Gelände aus geodätischen Vermessungen vorhanden sein. Somit ist es ein Leichtes, eine fotorealistische Visualisierung der Brücke zu erstellen. So kann die ästethische Wirkung der Brücke ebenfalls sehr schnell untersucht werden. Ansätze hierzu finden sich ebenfalls in (Singer 2014), S. 80.

In Abbildung 5.5.1 und 5.5.2 sind Renderings verschiedener Varianten aus Revit zu sehen. Es sind lediglich Materialien für die Oberflächendarstellung festgelegt worden. Das sogenannte Level-of-Development (LOD) ist beim erstellten Modell noch sehr gering. So sind technische Ausstattungen, Fahrbahnmarkierungen und weitere Details wie beispielsweise Geländer noch nicht dargestellt. LOD ist eine Definition des American Institute of Architects. Mit LOD werden zwischen den verschiedenen Stufen der Entwicklung eines digitalen Modells unterschieden. Sie reichen von LOD 100, der Darstellung als Symbol, bis LOD 500, wo alle Details, Abmessungen und Eigenschaften vorhanden sind (American Institute of Architects 2013). Dies kann jedoch jederzeit weiterentwickelt werden. So könnte man genormte Regelquerschnitte oder Standardquerschnitte der Deutschen Bahn implementieren und so zu einem kompletten Modell kommen. Dies übersteigt den Rahmen dieser Arbeit, jedoch ist es nicht unmöglich.

Abbildung 5.5.1: Gerendertes Beispiel einer einfachen Bogenbrücke

Abbildung 5.5.2: Gerenderte Variante mit Doppelbogen einer Brücke

Abbildung 5.5.3: Gerenderte Variante mit langem Einfachbogen derselben Brücke

6 Fazit

Im Rahmen dieser Arbeit wurde die Eignung der parametrischen Modellierung für Bogenbrücken untersucht. Mit den verwendeten Programmen ist es definitiv möglich, ein sehr gutes Ergebnis zu erzielen. Auch wenn beim Import derzeit noch Probleme bestehen, so darf man doch sicher davon ausgehen, dass parametrische Modellierung in Zukunft eine sehr große Rolle spielen wird. Wenn man bedenkt, dass Dynamo sich aktuell überhaupt erst in Version 1.0 befindet, kann man zuversichtlich davon ausgehen, dass die derzeit bestehenden Probleme noch verbessert werden.

Auch am Skript selbst gibt es noch etliche Verbesserungsmöglichkeiten: So könnte man beispielsweise Regelquerschnitte aus der DIN-Norm oder der Deutschen Bahn modellieren, damit der Nutzer aus ihnen auswählen kann. Auch Geländer, technische Einrichtung und Details der Verbindung von Bogen und Fahrbahn sind eindeutig noch ausbaufähig. Diese Erweiterungen sind zwar zeitintensiv, am Ende kompensieren sie jedoch den Aufwand unbedingt. Momentan ist auch die Berechnungszeit mit einer Dauer von mehreren Minuten leider noch relativ hoch. Auch hier ebenfalls noch eindeutigen Optimierungsbedarf in der Performance des Skripts.

Das eingangs dieser Arbeit aufgeführte Zitat der Deutschen Bahn definiert klar eine unabdingbare Notwendigkeit, Variantenstudien durchzuführen. Variantenstudien dienen der Findung einer optimalen technischen Lösung für den Bau einer jeden Brücke. Mit den erzeugten Varianten, die als BIM-Modell zur Verfügung stehen und sich aus individuellen Parametern ableiten lassen, kann man weitere virtuelle Optimierungswerkzeuge und Analysesoftware einsetzen. Beispielsweise könnte ohne großen Aufwand eine FEM-Berechnung an selbigem Modell durchgeführt werden.

Wie oben bereits erwähnt, könnte parametrische Modellierung vor allem in Verbindung mit KBE in Zukunft eine große Rolle in der Bauwirtschaft spielen. Mit Hilfe von künstlicher Intelligenz entstünde nicht nur ein qualitativer, sondern auch ein wirtschaftlicher Vorteil. Das Finden einer optimalen Lösung wird durch Variantenstudien erleichtert und dank präziserer, Gewerke übergreifender Planung können über die gesamte Lebensdauer des Bauwerks Kosten gespart werden. Mittels der Nutzung von BIM-Modellen und KBE kann das Baugewerbe von den sich weiterentwickelnden computertechnischen Entwicklungen des 21. Jahrhunderts profitieren.

Literaturverzeichnis

American Institute of Architects (2013): Document G202. Project Building Information Modeling Protocol Form.

Autodesk (2015): The Dynamo Primer. For Dynamo v9.0. Hg. v. Autodesk. Online verfügbar unter http://dynamoprimer.com/index.html, zuletzt geprüft am 07.07.2016.

Borrmann, André; König, Markus; Koch, Christian; Beetz, Jakob (Hg.) (2015): Building Information Modeling. Technologische Grundlagen und industrielle Praxis. Wiesbaden: Springer Vieweg (VDI-Buch). Online verfügbar unter http://dx.doi.org/10.1007/978-3-658-05606-3.

Bundesministerium für Verkehr und digitale Infrastruktur (15.12.2015): Building Information Modeling (BIM) wird bis 2020 stufenweise eingeführt. Dobrindt: Großprojekte durch Digitalisierung optimieren. Online verfügbar unter https://www.bmvi.de/SharedDocs/DE/Pressemitteilungen/2015/152-dobrindt-stufenplan-bim.html, zuletzt geprüft am 21.07.2016.

DB Netze (2008): Leitfaden Gestalten von Eisenbahnbrücken. Unter Mitarbeit von Jörg Schlaich, Thomas Fackler, Matthias Weißbach, Victor Schmitt, Christian Ommert, Steffen Marx, Ludolf Krontal.

Duden - Algorithmus: Algorithmus Definition. Hg. v. Duden. Online verfügbar unter http://www.duden.de/rechtschreibung/Algorithmus, zuletzt geprüft am 25.04.2016.

Fischer, Oliver (2013): Vorlesungsskript Massivbrücken Technische Universität München.

Geißler, Karsten (2014): Handbuch Brückenbau. Entwurf, Konstruktion, Berechnung, Bewertung und Ertüchtigung. Berlin, Germany: Ernst & Sohn. Online verfügbar unter http://onlinelibrary.wiley.com/book/10.1002/9783433603437.

MAPLOGS (2010): Falkensteinbrücke. Online verfügbar unter http://elevation.maplogs.com/poi/gemeinde_flattach_austria.13696.html, zuletzt geprüft am 08.07.2016.

Mehlhorn, Gerhard; Curbach, Manfred (Hg.) (2014): Handbuch Brücken. Entwerfen, Konstruieren, Berechnen, Bauen und Erhalten. 3. Aufl. Wiesbaden: Springer Vieweg. Online verfügbar unter http://dx.doi.org/10.1007/978-3-658-03342-2.

Rama (2015): Salginatobelbrücke. WikimediaCommons. Online verfügbar unter https://upload.wikimedia.org/wikipedia/commons/d/d1/Salginato-bel_Bridge_mg_4074.jpg, zuletzt geprüft am 08.07.2016.

Ritter, F.; Preidel, C.; Singer, D. (2015): Visuelle Programmiersprachen im Bauwesen - Stand der Technik und aktuelle Entwicklungen. In: Proceedings of the 27th Forum Bauinformatik. Aachen, Germany.

Singer, D. (2015): Einsatz wissensbasierter Methoden in frühen Phasen des Brückenentwurfs. In: Proceedings of the 27th Forum Bauinformatik. Aachen, Germany.

Singer, Dominic (2014): Entwicklung eines Prototyps für den Einsatz von Knowledge-based Engineering in frühen Phasen des Brückenentwurfs: Technische Universität München.

Singer, Dominic; Bügler, Maximilian; Borrmann, André (2016): Knowledge based Bridge Engineering - Artificial Intelligence meets Building Information Modeling. In: Proc. of the EG-ICE Workshop on Intelligent Computing in Engineering. Krakow, Poland.

Stokes, M. (2001): Managing engineering knowledge. MOKA: methodology for knowledge based engineering applications. London: Professional Engineering Pub.

Verhagen, W.; Bermell-Garcia, P.; van Dijk, R.; Curran, R. (2012): A critical review of Knowledge-Based Engineering: An identification of research challenges. In: *Advanced Engineering Informatics* (26), S. 5–15. DOI: 10.1016/j.aei.2011.06.004.

Anhang A

Auf der beigefügten CD befindet sich folgender Inhalt:

- Der schriftliche Teil der Arbeit als Worddokument
- Die Daten des Projektes
- Die gerenderten 3D-Ansichten
- Der gesamte Programmcode
- Das Skript der Brücke in Dynamo

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 22. August 2016

Felix Stauch