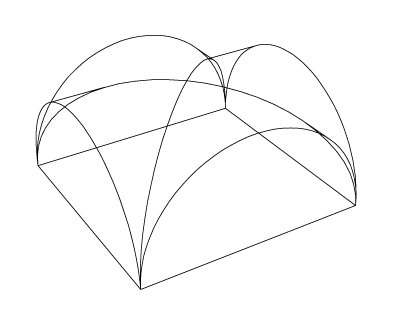
Kreuzgewölbe

Oberflächenberechnung im Detail



Merlin Worrings, Stephanie Eckert, Sophie Platen, Paul Hahn, Lars Hollenbach, Johanna Hader, Jannik Spiering

Inhalt

[2 Einleitung 2](#_Toc483216242)

[3 Ausgangsituation 2](#_Toc483216243)

[4 Ressourcen- und Ablaufplanung 2](#_Toc483216244)

[5 Pflichtenheft 2](#_Toc483216245)

[6 Protokolle 2](#_Toc483216246)

[7 Durchführung und Auftragsbearbeitung 2](#_Toc483216247)

[8 Verfahren 2](#_Toc483216248)

[9 Projektergebnisse 3](#_Toc483216249)

[10 Fazit 3](#_Toc483216250)

[11 Anhang 3](#_Toc483216251)

# Einleitung

Im Rahmen des Projektes „Kreuzgewölbe“ teilten sich die Schüler der Klasse MTS 41 selbständig in Gruppen ein. Ziel der Arbeit war es, die Fläche eines Kreuzgewölbes mit möglichst geringen Fehler zu berechnen. Als Grundlage der Berechnung dienten computergenerierte Messdaten sowie ein virtuelles Kreuzgewölbe welche auch bei der Validierung der entworfenen Algorithmen Verwendung fanden.

Der Fokus des Projektes lag in der Anwendung und Modellierung von mathematischen Modellen sowie im Projektmanagement. Die Schüler teilten das Projekt hierfür in Arbeitspakete innerhalb ihres Teams ein und arbeiteten Zielorientiert an der Umsetzung.

Die in diesem Bericht vorgestellte Gruppe wurde von Merlin Mabon Worrings geleitet. Zu den Mitgliedern des Teams zählten Stephanie Eckert, Sophie Platen, Paul Hahn, Lars Hollenbach, Johanna Harder und Jannik Spiering. Die Teammitglieder einigten sich zu Projektbeginn, die Umsetzung mithilfe der Programmiersprache C# zu realisieren. Aus Kompatibilitätsgründen mit der bereitgestellten Entwicklungsumgebung wird für die Ausführung das .NET Framework 3.5 vorausgesetzt.

# Ausgangsituation

Die Ursprüngliche Aufgabenstellung des Projektes sah vor ein reales Kreuzgewölbe zu vermessen. Bei der Auswahl der verwendeten Algorithmen spielten Bedenken bezüglich der Genauigkeit der Messdatenerfassung eine wichtige Rolle. Grund hierfür war die Tatsache das einige der vorgeschlagenen Verfahren stark von bestimmten Eigenschaften der gemessenen Punkte abhängig sind. Eine fehlerfreie Simulation mit künstlich erzeugten Werten ist mit diesen Algorithmen zwar uneingeschränkt möglich, die Verwendung von realen Messdaten wird hingegen gegeben falls aber nur unter Einschränkungen oder unzureichend unterstützt. Diese Bedenken mündeten letztendlich in der Suche einer alternativen Lösung mit hoher durch das Messverfahren bedingten Fehlertoleranz.

Kubische Splines

Lineare Regression

## Ausarbeitung eines Lösungsansatzes für die Flächenberechnung eines Kreuzgewölbes

Das Arbeiten mit abweichungsbehafteten Messdaten resultierte in der Auswahl eines Verfahrens welches sich besonders Robust gegenüber solchen Abweichungen verhält.

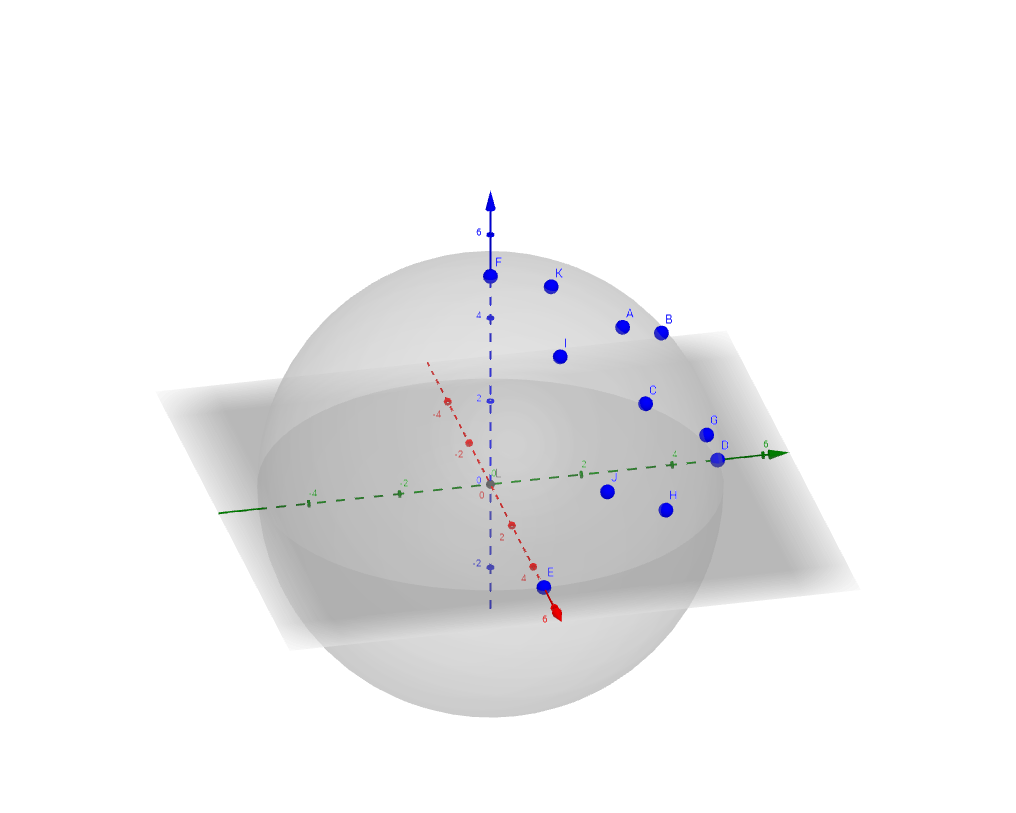
# Ressourcen- und Ablaufplanung

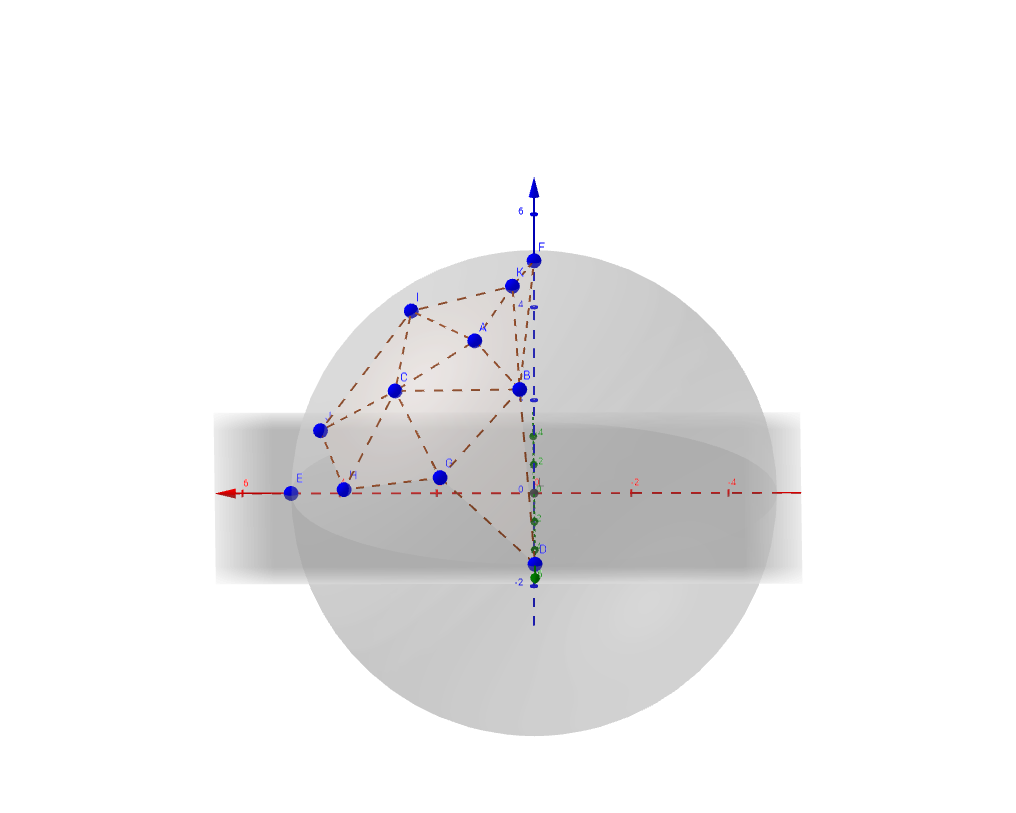
# Pflichtenheft

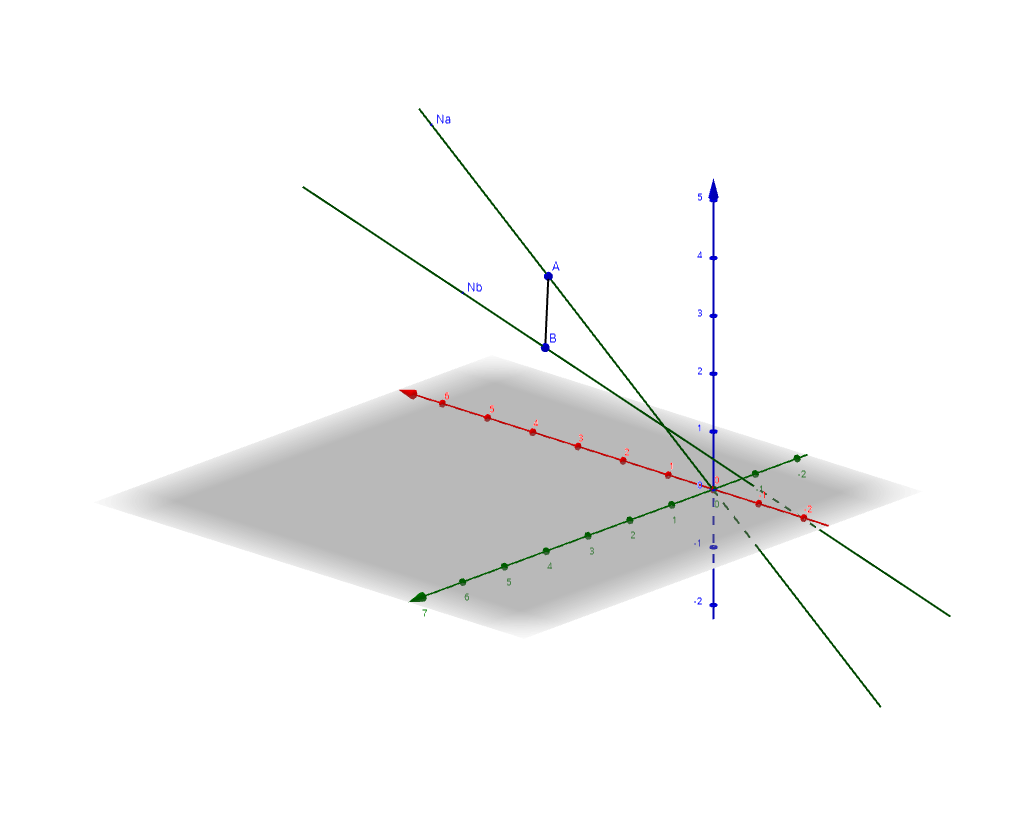
# Protokolle

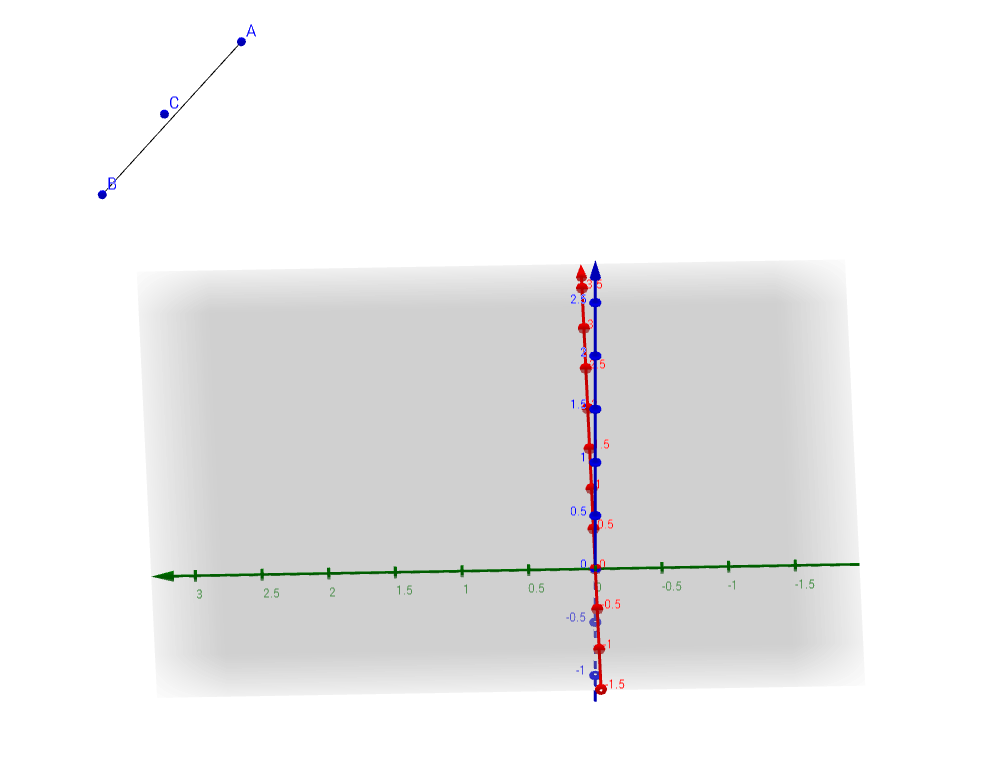
# Durchführung und Auftragsbearbeitung

# Verfahren









1. Zufällig angeordnete Messpunkte innerhalb der zu messenden Fläche und weitere Messpunkte entlang der Flächengrenzen sammeln.
2. Netz aus Dreiecken aus den Messpunkten generieren.
3. Für jedes Dreieck die Normale der Ebene berechnen, die durch dessen drei Ecken definiert wird.
   * Dabei ist wichtig, dass alle Normalen auf dieselbe Seite der Fläche zeigen
4. Für jede Dreiecks-Ecke (Vertex) aus den Normalvektoren aller daran anliegenden Dreiecke den Durchschnittsnormalvektor ausrechnen.
   * Die Normalvektoren der einzelnen Dreiecke müssen gewichtet werden, dabei ist die Länge der an die Ecke anliegenden Kanten des Dreiecks ausschlaggebend.
   * Je länger die Kanten, desto weniger wichtig der Normalvektor des Dreiecks.
5. Jede Ecke enthält nun die Information über die eigene Position und die Normale einer Tangentialebene an diesem Punkt, die an der gemessenen Oberfläche anliegt. Mithilfe dieser Informationen wird nun über jeder Kante eine Funktion gebildet, die der Oberfläche möglichst nahekommen soll.
   * Die Funktion sollte eine möglichst gleichmäßige Steigung haben um nah an die runde gemessene Oberfläche heranzukommen (was nicht immer perfekt machbar ist).
   * Möglich (und einfach) ist Kubische Funktion, gibt es bessere Möglichkeiten?
   * Schritt für Schritt zur Kubischen Funktion:

* Aus Informationen in den Punkten A und B, die die Kante definieren, die nötigen Informationen für eine Funktion über der Kante bilden:
  1. (Normale in A) x AB für das Lot
  2. (Lot von Na und AB) x Na für Steigungsvektor über der Kante in A
  3. Das gleiche nochmal für die Steigung von B über der Kante
* Parametrisieren der Daten in XY-Ebene um einfaches Verfahren für Kubische Funktion zu erhalten:
  1. Man wählt für die parametrisierte Funktion f(t) folgendes: f (0) =A=0 und f (1) =B=0
  2. Der Vektor Va verläuft zwischen der Kante AB und dem Steigungsvektor in A über der Kante. Er berechnet sich Va = Normalisiert ((Lot von Na und AB) x AB)
  3. Für die Parametrisierung der Steigung in A über der Kante werden zwei Geradengleichungen gebildet. G1 = A + r\*(Steigung in A über Kante) und G2 = B + s\*Va. G1 und G2 werden geschnitten. Für den Schnittpunkt ist bekannt, dass er bei t=1 liegt und weil Va normalisiert ist, ist s = f(1). Die parametrisierte Steigung von A über der Kante ist s.
     1. Nach Umstellung der Gleichung für den Schnittpunkt ergibt sich die Formel:

s=(B2\*d1-A2\*d1-B1\*d2+A1\*d2)/(Va1\*d2-Va2\*d1)

d = Steigung in A über der Kante

z.B. d2 = Zweite Komponente des Vektors d

* 1. Das gleiche für die Steigung von B über der Kante, allerdings muss hier –s genommen werden, damit eine negative Steigung entsteht.
  2. Nun sind f‘ (0) = g und f‘ (1) = h auch bekannt.
  3. Wenn g und h beide 0 sind, wird der Mittelpunkt von AB genommen, statt den Extrempunkt zu berechnen.
* Mit vier Informationen über f(t) lassen sich a,b,c und d in f(t)=at^3+bt^2+ct+d berechnen.
  1. Wir wissen, dass f (0) = 0 und f (1) = 0 und f´(0) = g und f´(1) = h. Daraus folgt für unsere kubische Funktion f(t) dass a = h+g

b = -h-2g

c = g

d = 0

also f(t)= (h+g)t^3+(-h-2g)t^2+gt+0, 0 < t < 1

1. Man nimmt dem höchsten Punkt (der Extremwert der obenstehenden kubischen Funktion) auf der Funktion jeder Kante und bildet aus diesen drei Punkten ein neues Dreieck, das sich nun innerhalb des vorherigen Dreiecks befindet. Da die Punkte auf dem Rand des Dreiecks liegen, wird das Dreieck in vier kleinere Dreiecke geteilt.
   * m = (2(-h-2g)) /(3(h+g))

* Nullstellen: t1/2= (-m/2) +- sqrt((m/2)^2 – g/(3(h+g)))
* Extrempunkt: P(t,f(t))
  + Ein Punkt im Raum(t,y) kann zurückparametrisiert werden in den Raum(x,y,z)
  1. Dafür P (X, Y, Z) = A + t \* (B-A) + y \* Normalisiert(Va\*(1-t)+Vb\*t) rechnen. Va verläuft zwischen der Kante AB und dem Steigungsvektor in A über der Kante. Er berechnet sich Va = Normalisiert ((Lot von Na und AB) x AB).
     1. A und B sind die Eckpunkte des Dreiecks im Raum(x,y,z)
* Wenn ein Wendepunkt im Intervall 0 < t < 1 auf f(t) vorliegt, müsste die Kante durch zwei neue Kanten ersetzt werden, die zu diesem Wendepunkt laufen. Die zwei an die Kante anliegenden Dreiecke werden zu vier Dreiecken geteilt. Die anliegenden Normalen müssen neu berechnet werden (Schritt 4 für die vier alten und den neuen Punkt).

1. Man beginnt erneut bei Schritt 3.
   * Die Normalen müssen auch für die Vertices berechnet werden, für die es vorher bereits eine Normale gab, da nun nur noch Teildreiecke der bisherigen anliegenden Dreiecke an den Vertices anliegen.
2. Wiederholen, bis die Teildreiecke so klein sind, dass es keinen signifikanten Unterschied mehr macht, ob man ihre Fläche als Ebene betrachtet.
3. Alle Flächen der winzigen Teildreiecke berechnen und dann aufsummieren.

Das Verfahren löst das Problem, dass es bei einer einfachen Interpolation der Werte gegeben hätte, da ein Punkt in der Mitte des Dreiecks, der einen höheren Abstand zur Dreiecksebene hat, als die Kantenfunktionen nun auch korrekt berechnet werden kann.

# Diagramme

# Projektergebnisse

## Programm

# Fazit

# Quellenangaben

<https://de.wikipedia.org/wiki/Delaunay-Triangulierung> (ein paar Algorithmen für 2D)

<http://www.research.ibm.com/vistechnology/pdf/bpa_tvcg.pdf> (Ballpivoting für 3D)

<https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2002/tr-2002-02.pdf> (Diskussion versch. Algorithmen für 3D)

# Anhang