Bachelorarbeit

Simulation von Bodenbewegungsszenarien von Starkbeben

Silvio Schwarz

silvio.schwarz@uni-potsdam.de

Matrikelnummer: 743269

Erstgutachter: Prof. Dr. Frank Scherbaum

Zweitgutachter: Dr. Nicolas Kühn

Abgabetermin: 14.09.2011

Inhaltsverzeichnis

1 Einleitung und Zielstellung

Zuverlässige Aussagen zu zukünftigen Erdbeben sind eine der großen Herausforderungen in der Seismologie. Auf Seiten der Politik, der Wirtschaft aber auch des öffentlichen Lebens gibt es ein hohes Interesse an Informationen zu Ort, Zeitpunkt und Stärke eines Bebens. Doch es ist fraglich, ob der Wunsch nach der kompletten Vorhersagbarkeit jemals erfüllt werden kann. Was dahinter steckt, ist die Absicherung gegen Schäden und Ausfälle sowie der Schutz von Menschenleben. Viel wichtiger als ein Datum ist also die Frage: Welche Auswirkungen wird ein bestimmtes Erdbeben auf seine Umgebung haben? Denn obwohl sich der Zeitpunkt nicht voraussagen lässt, so gibt es doch gute Erkenntnisse, wo Erdbeben zu erwarten sind und welche Orientierung die Bruchflächen an bestehenden Verwerfungen haben. Bodenbewegungsmodelle bilden dabei eine Grundlage zur seismischen Gefährdungsanalyse. Sei es um Regionen mit einem hohen Gefährdungspotential ausfindig zu machen oder in Kombination mit Ingenieurswissenschaften potentielle Schäden zu erfassen. Diese Modelle benötigen verschiedene bruchflächenbezogene Distanzmessungen, Magnitude des Bebens und Parameter, die den Untergrund beschreiben. In dieser Arbeit wird ein Werkzeug vorgestellt, das es ermöglicht Epizentraldistanz (R_{epi}), Hypozentraldistanz (R_{hypo}) und Joyner-Boore-Distanz (R_{jb}) sowie die kürzesten Entfernung zur Bruchfläche (R_{rup}) zu berechnen (Abrahamson und Shedlock 1997). Durch leistungsstarke Computer kann eine Vielzahl an Berechnungen in kurzer Zeit erfolgen. Das ermöglicht nicht nur vergangene Erdbeben nachzuvollziehen und analysieren zu können, sondern auch Szenarien für zukünftige Ereignisse zu entwerfen und zu sehen, welche Folgen sich für die betroffene Regionen ergeben könnten. Auch können simulierte Ergebnisse mit real gemessenen Daten verglichen werden, was rückwirkend Einfluss auf die Modellwahl hat. Da der Großteil der Berechnungen auf einfachen Beziehungen der Vektoralgebra beruht, handelt es sich nicht um komplexe theoretische Betrachtungen, sondern um praktische Anwendungen, die in direktem Verhältnis zur jeweiligen Verwerfungsgeometrie stehen.

2 Umsetzung

Mathematica 8.0 von Wolfram Research liefert als Softwarepaket eine Vielzahl an Möglichkeiten effizient und problemorientiert zu arbeiten. Dabei gehen die Einsatzgebiete weit über typisch mathematisch-naturwissenschaftliche Fragestellungen hinaus. Diese Flexibilität macht Mathematica zu einem der meist genutzten Datenverarbeitungsprogramme. Somit sind die hier vorgestellten Funktionen keine speziellen Einzellösungen, sondern lassen sich auch an andere Stelle übertragen und nutzen. Mit mathSHA existiert an der Universität Potsdam bereits ein Paket für Mathematica, das eine große Anzahl an Werkzeugen für die seismische Gefährdungsanalyse bereithält. Folglich liegt es nahe, die gleiche Software zu wählen, da eine Implementierung in mathSHA angestrebt wird.

Mit RupturePlane3D und RupturePlane3DVisual (Anhang 1 und 2) wurden zwei Funktionen innerhalb Mathematicas entwickelt, die R_{epi}, R_{hypo}, R_{jb} und R_{rup} berechnen. Sie unterscheiden sich dabei nur in dem Punkt, dass RupturePlane3DVisual zusätzlich zu den Distanzen eine graphische Ausgabe der Bruchfläche liefert. Beide Funktionen benötigen folgende Eingabeparameter:

- Streichwinkel ϕ
- Fallwinkel δ
- Momentenmagnitude M_W
- Lage des Hypozentrums mit Longitude und Latitude in Dezimalgrad, Tiefe in positiven Metern
- zwei Verschiebungsparametern der Bruchfläche gegenüber dem Hypozentrum
- Lage der Station mit Longitude und Latitude in Dezimalgrad

Diese meist bekannten Parameter einer Bruchfläche bilden die Grundlage für das folgende Bruchflächenmodell und die Distanzberechnungen.

2.1 Entwicklung des Bruchflächenmodells

Zur Erstellung des Bruchflächenmodells konnte das Mathematica-Script „Styles of Faulting“ (Scherbaum et al.) als Grundlage genutzt werden. Hierbei wird durch die Eingabe von Streichen, Fallen und des Rake die jeweilige Verwerfungsart ermittelt und an 2 Halbkugeln graphisch dargestellt.

Innerhalb der Funktion wird mit kartesischen metrischen Koordinaten gearbeitet, weil sich so die Distanzen einfach über die Länge von Differenzvektoren bestimmen lassen. Die Koordinaten des Hypozentrums und der Station werden durch die Funktion XYFromLonLat aus dem mathSHA-Paket in kartesische Koordinaten umgewandelt. Dabei dient das Epizentrum als Referenzpunkt.

Die Bruchfläche wurde als rechteckiges Polygon mit den Eckpunkten (0.5, 0.5, 0), (-0.5, 0.5, 0), (-0.5,-0.5, 0) und (0.5, -0.5, 0) definiert (Abb.1).

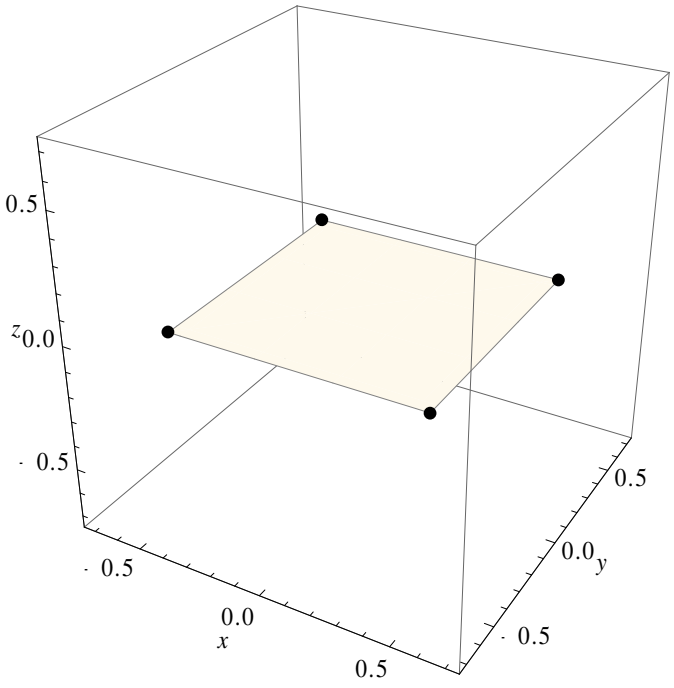

Abbildung 1: Ausgangspunkt des Bruchflächenmodells als Polygon mit den Eckpunkten (0.5, 0.5, 0), (-0.5, 0.5, 0), (-0.5,-0.5, 0), (0.5, -0.5, 0)

Die Darstellung wird im Anschluss durch verschiedene Transformationsfunktionen bearbeitet, sodass der Verwerfungsgeometrie mit Streichen und Fallen, der Lage des Hypozentrums im Raum und der Ausdehnung der Bruchfläche mit Länge und Breite entsprochen wird. Dabei wird nicht das Polygon als Ganzes verändert, sondern nur die Eckpunkte transformiert. Die somit neugewonnenen Eckpunkte (*transpoints*)

dienen dann als Stützpunkte, um die gewünschte Bruchfläche als Polygon darzustellen. Mittels *RotationTransform* wird eine Transformationsmatrix erstellt, die von den jeweiligen Werten des Streichens und Fallens abhängt. Des Weiteren werden so auch neue bruchflächenbezogene Einheitsvektoren (*sdv* – Einheitsvektor in Streichrichtung, *ddv* – Einheitsvektor in Fallrichtung) berechnet. *TranslationTransform* erfüllt dann die Aufgabe, das Hypozentrum an den richtigen geographischen Ort im Raum zu bewegen. Das entspricht einer einfachen Addition der Eckpunkte mit den Hypozentrumskoordinaten. Weil die Eckpunkte bei der Definition des Ursprungspolygons (Abb.1) symmetrisch um den Koordinatenursprung liegen, befindet sich das Hypozentrum so im Mittelpunkt des Polygons. Da dies nicht immer der Realität entspricht wird in den *TranslationTransform – Befehl* noch ein Term eingefügt, der es ermöglicht die Bruchfläche gegenüber dem Hypozentrum entlang der zwei Bruchflächeneinheitsvektoren zu verschieben. Die letzte Aufgabe, um die Bruchfläche gemäß ihrer Natur darzustellen, ist die richtige Skalierung der Länge und Breite. Länge bezeichnet dabei die horizontale Ausdehnung und Breite die Ausdehnung entlang des Fallens der Bruchfläche. Dazu werden die von Wells und Coppersmith (1994) gefundenen Beziehungen zwischen Länge und Breite einer Bruchfläche und der Momentenmagnitude benutzt*:*

$$\log_{10}(\mathrm{RDL}) = -2.44 + 0.59\, M_w$$

$$\log_{10}(\mathrm{RW}) = -1.01 + 0.32\, M_w$$

RDL bezeichnet dabei die Länge der Bruchfläche unterhalb der Erdoberfläche. RW ist die Breite der Bruchfläche entlang des Fallens.

ScalingTransform gibt dann die entsprechende Transformationsfunktion, bei der die Eckpunkte mit den Werten von Länge und Breite skalar multipliziert werden. Da diese Transformation auf alle Ausgangspunkte angewendet wird, ist es wichtig diese mit Werten von 0.5 beziehungsweise -0.5 zu definieren, sonst würde die resultierende Bruchfläche doppelte Länge und Breite aufweisen.

Diese drei Transformationsfunktionen werden als Variable *trans* gespeichert und auf die Ausgangspunkte angewendet. Somit erhält man die transformierten Punkte

transpoints. Für die Betrachtung der Joyner-Boore-Distanz ist später noch die Oberflächenprojektion der Bruchfläche notwendig. Dazu werden die z-Werte der transformierten Punkte auf 0 gesetzt und bilden als *transpoints2* die Eckpunkte des Polygons der Oberflächenprojektion der Bruchfläche. Aus den Koordinaten für das Hypozentrum wird auf gleichem Weg die Lage des Epizentrums ermittelt. Zur anschaulicheren Darstellung wurde die in „Styles of Faulting“ (Scherbaum et al.) existierende Gradskala so verändert, dass sich Radius und Lage immer anhand der Ausdehnung des Bruchflächenpolygons und der Position des Hypozentrums definieren (Abb. 2). Zusätzliche wurden noch zwei Fehlermeldungen implementiert, die mit dem Ergebnis der Berechnungen ausgegeben werden. Einmal darauf hingewiesen, falls sich die Bruchfläche über der Erdoberfläche befindet. Die zweite Fehlermeldung erscheint, falls das Hypozentrum außerhalb der Bruchfläche liegt.

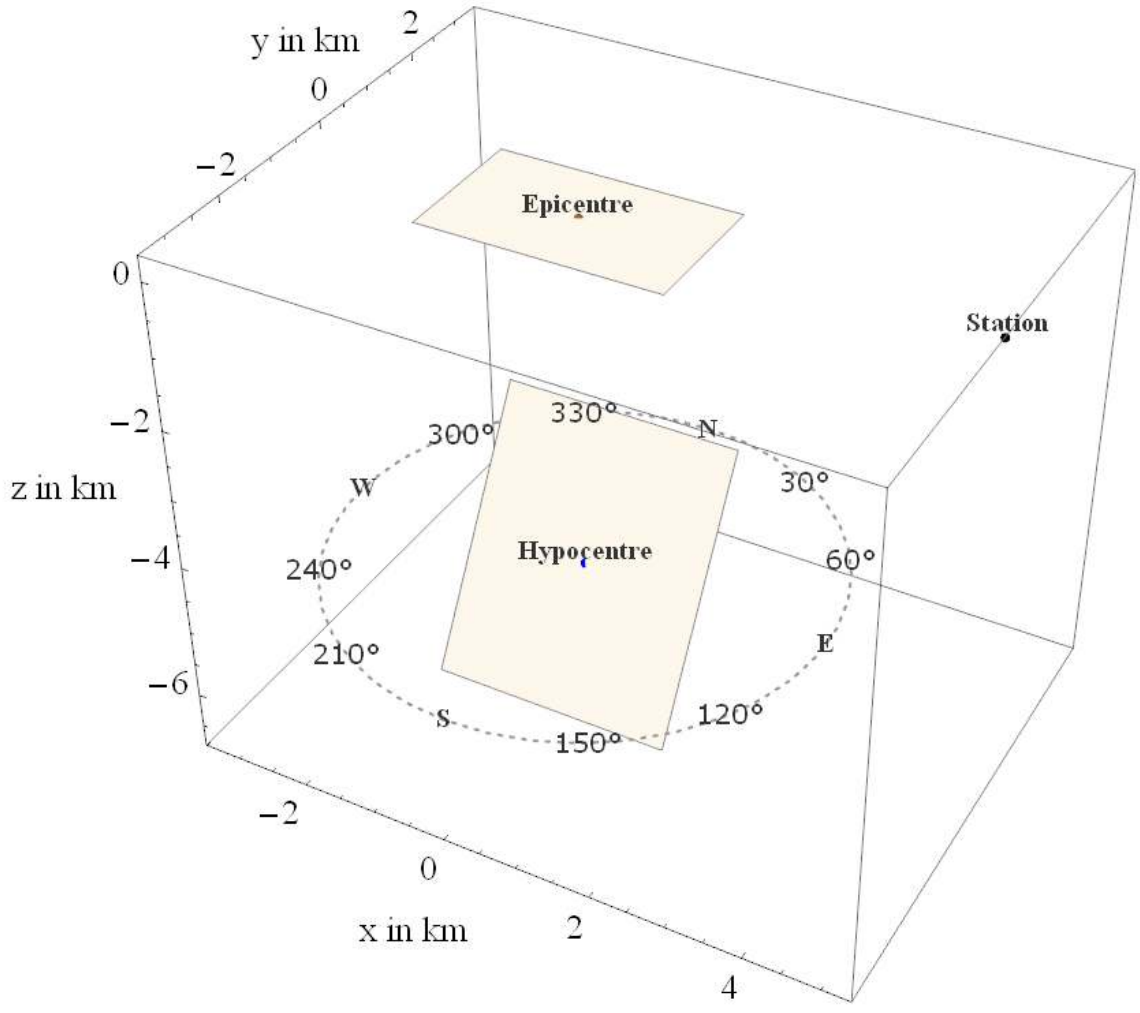

Abbildung 2: Komplettes Bruchflächenmodell mit Epizentrum, Hypozentrum und Stationspunkt. Des Weiteren ist die Oberflächenprojektion der Bruchfläche dargestellt. Ein Kreis mit Gradeinteilung sowie die Achsenbeschriftung in Kilometern sollen die räumliche Vorstellung erleichtern.

2.2 Distanzberechnungen

Da die Punkte auch als Ortsvektoren aufgefasst werden können, lassen sich Epizentral- und Hypozentraldistanz einfach bestimmen, in dem die Differenz zwischen Station und jeweiligen Punkt für Epizentrum beziehungsweise Hypozentrum gebildet wird. Anschließend berechnet die Funktion *Norm* die Euklidische Länge dieses Differenzvektors, was der gesuchten Distanz entspricht.

Für R_{rup} und R_{jb} müssen erst die korrespondierenden Punkte auf der Bruchfläche beziehungsweise auf der Oberflächenprojektion der Bruchfläche gefunden werden. Dabei ist der Algorithmus, der verwendet wurde, um dieser Aufgabe innerhalb von Mathematica gerecht zu werden, für beide Distanzberechnungen ähnlich.

Für $R_{rup,}$ muss zuerst der Stationspunkt auf die Ebene, in der das Bruchflächenpolygon liegt, projiziert werden. Mit *sdv* und *ddv* sind schon die beiden Vektoren bekannt, die diese Ebene aufspannen. Die Ebenengleichung lässt sich somit leicht durch das Kreuzprodukt dieser Vektoren, das den Normalenvektor $\vec{n}$ der Ebene liefert, ausdrücken: $\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$. Mit $n_x\, x + n_y\, y + n_z\, z = d$ erhält man eine Ebenenschar, bei der sich die einzelnen Ebenen nur durch ihren Abstand zum Ursprung d unterscheiden. Da sich das Hypozentrum innerhalb der gesuchten Ebene befindet, lässt sich durch Einsetzen dieser Koordinaten in die Gleichung der Ebenenschar d ermitteln und somit die gesuchte Bruchflächenebene definieren.

Der kürzeste Abstand von der Station zu der Bruchflächenebene muss senkrecht auf der Ebene stehen. Es lässt sich also eine Gerade ausgehend vom Stationspunkt (*stationcoord*) mit Richtung des Normalenvektors $\vec{n}$ der Ebene bilden, die diese Bedingung erfüllt:

$$stationcoord + h\,\vec{n}.$$

Setzt man diese Gerade in die Ebenengleichung ein, so findet sich ein bestimmter Wert für h, der die Gleichung eindeutig löst. Dieses h wird anschließend wieder in die Geradengleichung eingesetzt, was den Projektionspunkt (*projectionpoint*) der

Station auf die Ebene des Bruchflächenpolygons liefert. Die weitere Vorgehensweise lehnt sich dabei an Kaklamanos et al. (2010) an, bei der neun verschiedene Fälle der Lage einer Station in Bezug auf die Oberflächenprojektion der Bruchfläche unterschieden werden. Diese Unterscheidung wird genutzt, um festzustellen, ob *projectionpoint* bereits im Bruchflächenpolygon liegt und somit den kürzesten Abstand zur Bruchfläche darstellt oder ob *projectionpoint* noch auf den Rand des Polygons gesetzt werden muss.

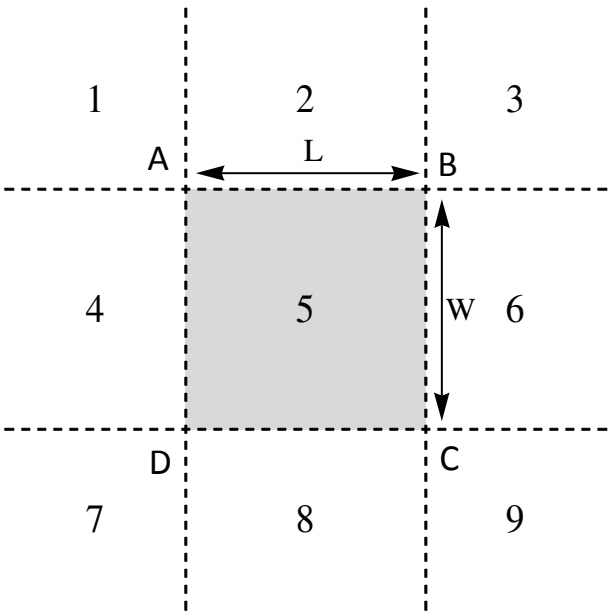

Abbildung 3: Unterscheidung von neun verschiedenen Lagen eines Punktes in der Bruchflächenebene in Bezug auf das Bruchflächenpolygon (grau). L und W stehen dabei für die Untergrundlänge der Bruchfläche beziehungsweise der Breite der Bruchfläche entlang ihres Fallens. Die Punkte A, B, C und D bezeichnen dabei die vier in *transpoints* bekannten Eckpunkte.

Um zu bestimmen, in welcher der neun Zonen *projectionpoint* liegt, wird der Differenzvektor zwischen *projectionpoint* und dem Mittelpunkt des Polygons gebildet. Anschließend wird dieser Differenzvektor mittels Skalarprodukt jeweils auf die beiden Flächeneinheitsvektoren *sdv* und *ddv* projiziert. Die so entstandenen Größen *check1* und *check2* geben an, wie weit sich *projectionpoint* vom Mittelpunkt des Polygons in *sdv*- beziehungsweise in *ddv*-Richtung befindet. Um die Zonen zu unterscheiden, werden *check1* und *check2* mit Länge und Breite des Bruchflächenpolygons verglichen (Tab. 1).

Zone	Unterscheidungskriterien
1	check1 ≤ - 0.5 L check2 ≥ 0.5 W
2	- 0.5 L < check1 < 0.5 L check2 > 0.5 w
3	check1 ≥ - 0.5 L check2 ≥ 0.5 W
4	check1 ≤ - 0.5 L - 0.5 W < check2 < 0.5 W
5	- 0.5 L ≤ check1 ≤ 0.5 L - 0.5 W ≤ check2 ≤ 0.5 W
6	check1 > 0.5 L - 0.5 W < check2 < 0.5 W
7	check1 ≤ - 0.5 L check2 ≤ - 0.5 W
8	- 0.5 L < check1 < 0.5 L check2 < -0.5 W
9	check1 ≤ - 0.5 L check2 ≥ 0.5 W

Tabelle 1: Unterscheidungskriterien für die neun verschiedenen Fälle, die ein Punkt in der Bruchflächenebene annehmen kann. Verglichen werden dabei die Projektionen des Ortsvektors von *projectionpoint* in Bezug auf den Mittelpunkt des Bruchflächenpolygons auf die Flächeneinheitsvektoren *sdv* und *ddv* mit der Länge und Breite des Polygons.

Für Zone 5 entspricht *projectionpoint* schon dem gesuchten nahesten Punkt von der Station zur Bruchfläche. Für die Zonen 1, 3, 7 und 9 wird der jeweilige Eckpunkt des Polygons, A, B, C oder D (Abb. 3), genommen. Dieser ist in *transpoints* bekannt. Für die Zonen 2, 4, 6 und 8 ist noch ein weiterer Schritt notwendig. Entsprechend Tabelle 2 werden Gleichungssysteme aufgestellt. Diese bilden jeweils den Schnittpunkt einer Gerade vom Eckpunkt des Polygons entlang eines Flächeneinheitsvektors (*sdv* oder *ddv*) mit einer Gerade von *projectionpoint* entlang des anderen Flächeneinheitsvektors. Dieser Schnittpunkt liegt somit auf dem Rand des Bruchflächenpolygons und ist

dann der gesuchte kürzeste Abstand zur Bruchfläche. Als Schwierigkeit stellte sich heraus, dass diese Gleichungssysteme mit zwei Variablen und drei Gleichungen überbestimmt sind. Das führte, trotz der hohen Genauigkeit von Mathematica, zu Fällen, in denen eine Zeile des Gleichungssystems nicht als Vielfaches einer anderen dargestellt werden konnte. Um dem entgegen zu wirken, wurde eine dritte Variable eingeführt, die aber für die Berechnung des Punktes keine Rolle spielt.

Zone	**Gleichungssystem**
2	$A + var11\ sdv = projectionpoint + (var1 + var12)\ ddv$
4	$A + var21\ ddv = projectionpoint + (var2 + \ var22)\ sdv$
6	$B + var31\ ddv = projectionpoint + (var3 + \ var32)\ sdv$
8	$D + var41\ sdv = projectionpoint + (var4 + var12)\ ddv$

Tabelle 2: Gleichungssysteme zur Ermittlung des Punktes auf dem Rand des Bruchflächenpolygons mit der kürzesten Entfernung zu *projectionpoint*.

In der Betrachtung von R_{jb} liegen Stationspunkt und das Polygon der Oberflächenprojektion schon in derselben Ebene bei $z = 0$. Eine Projektion entfällt somit. Einzig die Breite der Oberflächenprojektion muss durch $W_{jb} = W \cos\delta$ berechnet werden und die beiden Flächeneinheitsvektoren, die zur Projektion benötigt werden, werden durch Differenzbildung der Eckpunkte und Normierung auf die Länge 1 bestimmt. Ansonsten wird dasselbe Verfahren wie für die Bestimmung von $R_{rup.}$ Die Gleichungssysteme haben dabei wegen $z = 0$ nur zwei Gleichungen und zwei Variablen und sind somit nicht überbestimmt.

Abbildung 4 zeigt das komplette Bruchflächenmodell mit allen für die Distanzberechnung notwendigen Punkten.

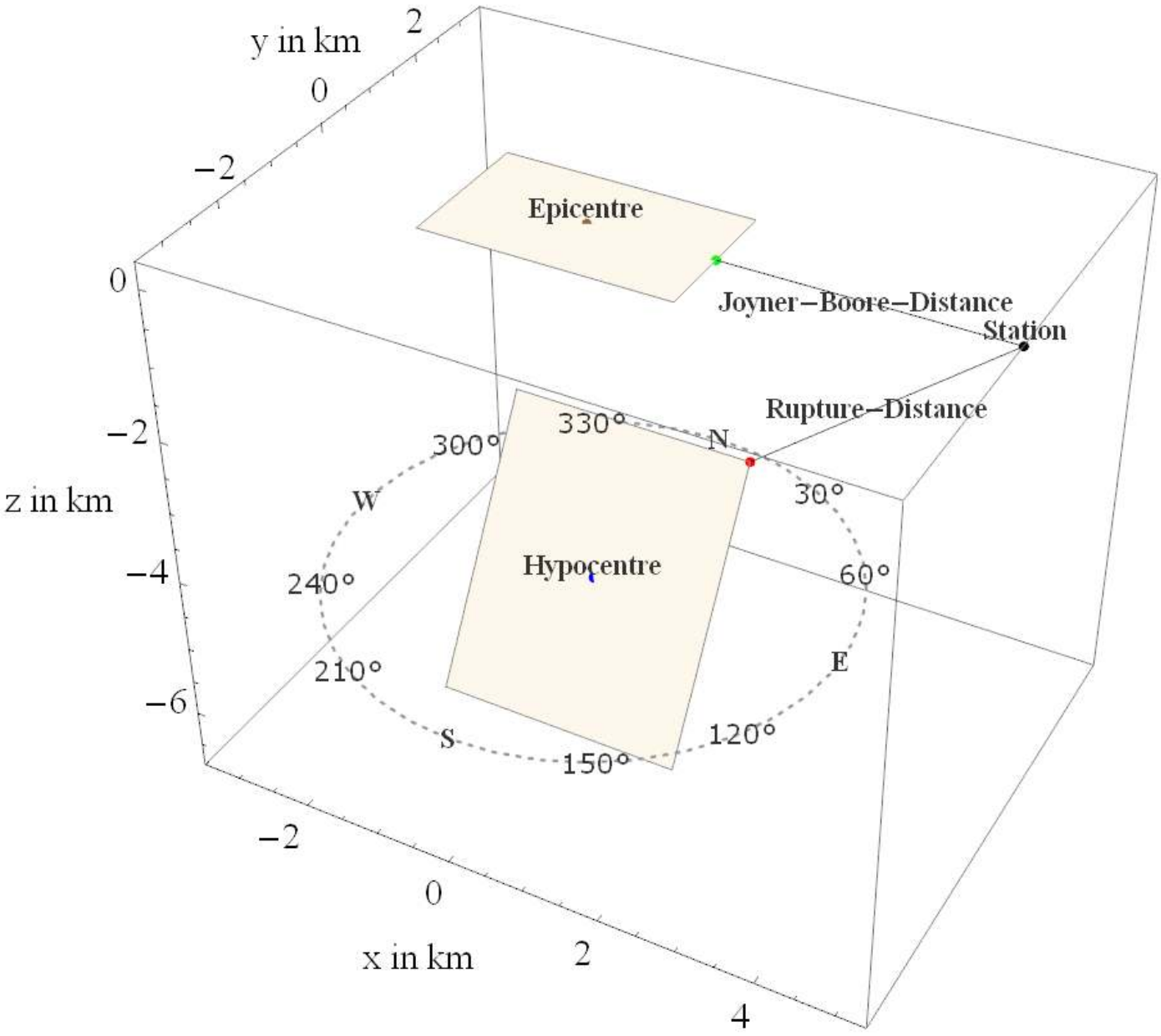

Abbildung 4: Komplettes Bruchflächenmodell mit Epizentrum, Hypozentrum und Stationspunkt. Des Weiteren ist die Oberflächenprojektion der Bruchfläche dargestellt. Der grüne Punkt stellt den nächsten Punkt der Oberflächenprojektion zur Station dar, der rote Punkt den nächsten Punkt zur Bruchfläche selbst.

2.3 Anwendung am Beispiel Roermond 1992

Das Roermond-Erdbeben vom 13. April 1992 in den Niederlanden war mit einer Magnitude von $M_{WA} = 5.9$ (Scherbaum 1994) eines der stärksten Erdbeben in Mitteleuropa. Somit spielt es eine wichtige Rolle für die seismische Gefährdungsanalyse in dieser Region. Das Epizentrum befand sich dabei im Roertal-Graben; einer Region, die momentan von Extension geprägt ist (Braunmiller et al. 1994). Trotz umfangreicher Studien gibt es noch große Unsicherheiten in den Herdparametern, vor allem für die Tiefe.

Im Folgenden soll kurz anhand dieses Erdbebens die Verwendung der Funktion RupturePlane3D gezeigt werden. Dafür werden die ermittelten Distanzen in einem Intensitätsmodell verwendet und mit realen Intensitäten verglichen. Die dafür nötigen Parameter lauten wie folgt:

Latitude	51,17° N
Longitude	5.93° E
Tiefe	18 km
Strike	138°
Dip	58°
Momentenmagnitude	$M_W = 5.3$

Tabelle 3: Parameter zum Erdbebenherd, der Orientierung der Bruchfläche und Momentenmagnitude, die als Eingabewerte für RupturePlane3D dienen (Braunmiller et al. 1994)

Für 2806 Stationen sind deren geographische Lage sowie die entsprechenden gemessenen Intensitäten bekannt (Scherbaum 2011). Damit die Funktion RupturePlane3D dieselben Erdbebenparameter mit unterschiedlichen Stationslokationen verwenden kann, muss eine Liste von Eingabewerten erstellt werden. Dazu wird durch

$$\text{sourceparam} = \text{Table}[\{138, 58, 5.3, 5.93, 51.17, 18000, 0, 0\}, \{2806\}];$$

eine Liste mit den Erdbebenherdparametern geschaffen, die dieselbe Länge hat, wie die Anzahl der Stationen. Anschließend liefert

$$\text{input} = \text{Partition}[\text{Flatten}[\text{Table}[\{\text{sourceparam}[[i]], \text{lon}[[i]], \text{lat}[[i]]\}, \{i, 2806\}]], 10]$$

die komplette Eingabeliste für RupturePlane3D mit den vorher nach Mathematica importierten Werten für Longitude (*lon*) und Latitude (*lat*) der verschiedenen Stationen. Mit *Map* wird dann RupturePlane3D auf diese Liste angewandt und die entsprechenden Distanzen berechnet.

Die Werte für R_{rup} und $M_W = 5.3$ dienten nachfolgend als Eingabeparameter für die Funktion *AtkinsonAndWald2007MMICENA* (Atkinson, G. M., und D. Wald 2007), die innerhalb von mathSHA vorliegt und die jeweiligen Intensitäten berechnet. Abbildung 5 zeigt die Differenzen zwischen den gemessenen und den berechneten Intensitäten in Abhängigkeit von $R_{rup.}$ Der Mittelwert der Abweichungen liegt bei -0.08840. Berechnete und reale Intensitäten unterscheiden sich im Mittel also kaum voneinander. Erkennbar sind verschieden Äste, die aus jeweils zwei linearen Teilen bestehen und einen Abstand zueinander von etwa 1 haben. Außerdem scheint es einen Trend mit zunehmender Entfernung zu geben. Jedoch liegt eine umfassende Analyse außerhalb des Rahmens dieser Arbeit.

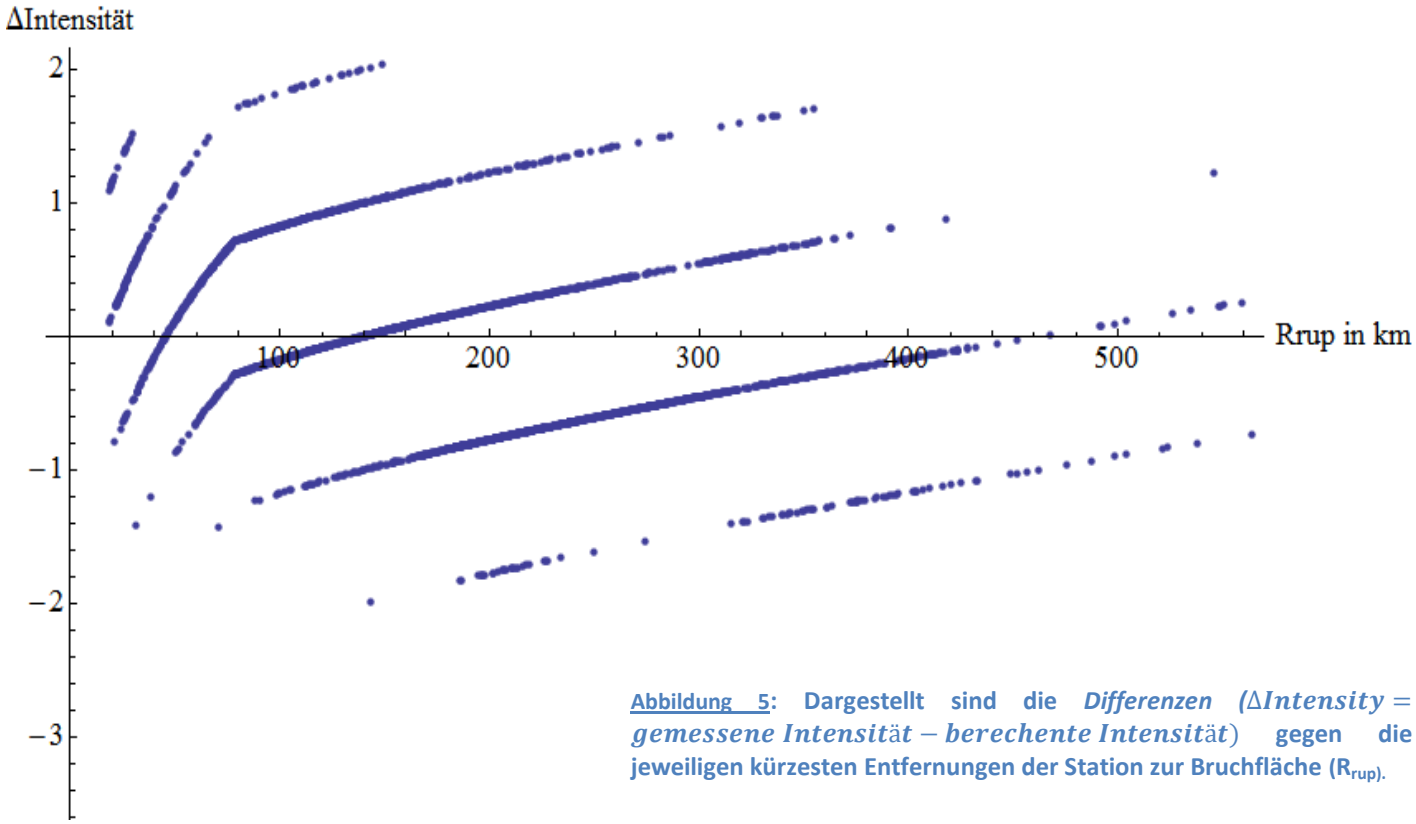

Abbildung 5: Dargestellt sind die *Differenzen* ($\Delta Intensity = gemessene\ Intensität - berechente\ Intensität$) gegen die jeweiligen kürzesten Entfernungen der Station zur Bruchfläche (R_{rup}).

3 Ausblick

Die beiden Funktionen RupturePlane3D und RupturePlane3DVisual für Mathematica bieten ein solides Werkzeug, um die bruchflächenbezogenen Distanzen R_{epi}, R_{hypo}, R_{jb} und R_{rup} zu berechnen. Dabei beruhen die zur Berechnung benutzen Beziehungen auf einfachen Sachverhalten der Vektoralgebra.

Dennoch gibt es Möglichkeiten der Verbesserung. So liefern Blaser et al. (2010) neue bessere Skalierungsbeziehungen zwischen Länge und Breite einer Bruchfläche und der Momentenmagnitude als Wells und Coppersmith (1994) und beziehen dabei auch Subduktionszonen mit ein. Um diese Beziehungen zu implementieren, müsste

allerdings noch eine Unterscheidung in der Art der Verwerfung getroffen werden, die bei Wells und Coppersmith (1994) nicht notwendig ist. Auch ist das Bruchflächenmodell auf eine einzige rechteckige Bruchfläche beschränkt und es werden noch keine Unsicherheiten, sei es bei den Beziehungen zur Ausdehnung oder den Eingabeparametern zu Lage und Orientierung der Bruchfläche, berücksichtigt. Diese spielen jedoch für die seismische Gefährdungsanalyse eine wichtige Rolle. Des Weiteren ist es erstrebenswert, dass die Überführung in kartesische Koordinaten innerhalb von RupturePlane3D mittels Mathematica-Funktionen vollzogen werden kann, damit diese beiden Werkzeuge der Distanzberechnung auch für Benutzer ohne das mathSHA-Paket nutzbar werden. Auch lässt sich die Geschwindigkeit beispielsweise durch Kompilierung noch erhöhen.

Danksagung

Ich möchte mich bei der Universität Potsdam sowie dem Land Brandenburg bedanken, die mir das Studium der Geowissenschaften in Potsdam ermöglichen. Ein besonderer Dank gilt meinen beiden Gutachtern Prof. Dr. Frank Scherbaum und Dr. Nicolas Kühn, die mich bei der Arbeit konstruktiv unterstützt und eher auf meine eigene Kreativität vertraut haben als mir einen vorgefertigten Lösungsweg vorzugeben.

Quellen

Atkinson, G. M. und D. Wald (2007): "Did You Feel It?" Intensity Data: A Surprisingly Good Measure of Earthquake Ground Motion. Seismol. Res. Lett., May/June 2007, Vol. 78, No. 3, pp. 362-368.

Abrahamson, N. A. und K. M. Shedlock (1997): Overview. Seismol. Res. Lett., January/February 1997, Vol. 68, No. 1, pp. 9-23.

Blaser, L., Krüger, F., Ohrnberger, M. und F. Scherbaum (2010): Scaling Relations of Earthquake Source Parameter Estimates with Special Focus on Subduction Environment. Bull. Seismol. Soc. Am., December 2010, Vol. 100, No. 6, pp. 2914–2926.

Braunmiller, J., Dahm, T. und K.-P. Bonjer (1994): Source mechanism of the 1992 Roermond earthquake from surface- wave inversion of regional data. Geophys. J. Int., (1994) 116, pp. 663-672.

Kaklamanos, J., Eeri, S.M., Baise, L.G. Eeri, M. und D.M. Boore (2010): Estimating Unknown Input Parameters when Implementing the NGA Ground-Motion Prediction Equations in Engineering Practice. Office of Geohazards Research at Tufts University, http://geohazards.cee.tufts.edu/Members/jkakla01/publications/kaklamanosEQS_NGAinputParameters.pdf.

Scherbaum, F. (1994): Modelling the Roermond earthquake of 1992 April 13 by stochastic simulation of its high-frequency strong ground motion. Geophys. J. Int. (1994) 119, pp. 31-43.

Scherbaum, F. (2011): Roermond-Intensities.doc (persönliche Kommunikation).

Scherbaum, F., Kühn, N. und B. Zimmermann: Styles of Faulting (Mathematica Skript), http://demonstrations.wolfram.com/StylesOfFaulting/

Wells, D. L. und K. J. Coppersmith (1994): New empirical relationships among magnitude, rupture length, rupture width, rupture area, and surface displacement. Bull. Seismol. Soc. Am., Vol. 84, pp. 974–1002.

Anhang

Anhang 1: RupturePlane3D

```
Needs["mathSHA`"]
RupturePlane3D[{phis_,delta_, mw_,lon_,lat_, dep_,a_,b_,statlon_,
statlat_}]:= Module[{e1,e2,e3,r1,r2,strikedir,allrotations, trans, coord,
size, hypo, station,planepoints, rhypo,repi,hypocoord,
planeequation,vs,vd,wjb, sdv,ddv,stationcoord, middlecoord,transpoints,
transpoints2,planenormal,projectionline,d,
sol,projectionpoint,projectionpoint2,projectionpoint3,mp,
check1,check2,check3,check4,solrup1,solrup2,solrup3,solrup4,
rrup,ms,soljb1,soljb2,soljb3,soljb4,rjb,w,l,flag1,flag2},


planepoints:={{-0.5,-0.5,0},{-0.5,0.5,0},{0.5,0.5,0},{0.5,-0.5,0}};
(*hypo centre location*)
hypocoord:=Append[XYFromLonLat[{lon, lat}, RefPoint → {lon,lat}]*1000, -
dep];
(*station location*)
stationcoord:=Append[XYFromLonLat[{statlon, statlat},UTMZone →
UTMZoneGet[{statlon, statlat}], RefPoint → {lon,lat}]*1000,0];
epicoord:= hypocoord {1,1,0};

e1:={1,0,0};
e2:={0,1,0};
e3:={0,0,1};

(*Define rotation into strike direction*)
r1:=RotationTransform[-N[phis Degree],e3];
(*strike direction*)
strikedir=r1[e2]//Chop;
(*Define rotation around strike direction to accomodate dip*)
r2:=RotationTransform[N[delta Degree],strikedir];
(*combine all rotations into single transform*)
allrotations=Composition[r2,r1];
```

```
w:=10 ^(-1.01 + 0.32 mw);
l:= 10 ^(-2.44 + 0.59 mw);
(*Define Length and Width of the rupture-area using the Wells&Coppersmith
relations*)

sdv:=allrotations[e2] //Chop;(*strike vector direction*)
ddv:=- allrotations[e1]//Chop; (*dip vector direction*)
(*plane position and translation along strike and dip direction vetors*)
middlecoord:=hypocoord +(a sdv + b ddv);
coord := TranslationTransform[Evaluate[((a sdv + b ddv) + hypocoord)]];
(*size of the rupture plane*)
size:=ScalingTransform[{w,l,0}];
(*combine all transformation into single one*)
trans=Composition[coord,allrotations ,size];
transpoints= (trans[#])&/@(Join[planepoints]);(*transformed points*)
transpoints2= transpoints {{1,1,0},{1,1,0},{1,1,0},{1,1,0}};(*surface
projection of transformed points*)

flag1=Text[" "];
flag2=Text[" "];
If[Positive[transpoints[[2,3]]], flag1=Text["Rupturearea lies above earth's
surface"]];

If[transpoints[[2,3]]  <  hypocoord[[3]], flag2=Text["Hypocentre is not
inside the ruptureearea"]];
If[transpoints[[3,3]] >   hypocoord[[3]], flag2=Text["Hypocentre is not
inside the ruptureearea"]] ;

(*Errormessages*)

(*Distancecalculations*)


rhypo:= Norm[stationcoord - hypocoord];
repi:= Norm[stationcoord - epicoord];

(*RRUP*)
planenormal:= sdv × ddv; (*Normalvector*)
d:= planenormal . middlecoord; (*Distance to the origin*)
(*planeequation:=planenormal . {x,y,z} -d; (*Equation of the
ruptureplane*)*)
projectionline := stationcoord + h planenormal;(*Line from station
perpendicular to the ruptureplane*)
planeequation := planenormal . projectionline -d; (*Intersection of line
and plane*)
sol :=Solve[planeequation == 0, h];(*Parameter of the line equation to
determine the intersectionpoint*)

projectionpoint:= stationcoord +  sol[[1,1,2]] * planenormal;(*Projection
of the station onto the ruptureplane*)

mp:= projectionpoint - middlecoord;
check1:= mp.sdv;
check2:= mp.ddv;


projectionpoint2 = projectionpoint;
Which[check2≥ 0.5w,
Which[check1 ≤ -0.5l,
projectionpoint2 = transpoints[[1]],
Abs[check1] <0.5l ,
```

```
{solrup1= Solve[transpoints[[1]]   + var11 sdv == projectionpoint  +
(var1+var21 ddv),{var11,var21,var1}] //Chop;
projectionpoint2 =  transpoints[[1]] + solrup1[[1,1,2]] sdv;},
check1≥ 0.5l,
projectionpoint2 = transpoints[[2]]],

Abs[check2]<0.5 w,
Which[check1<-0.5l,
{solrup3= Solve[transpoints[[4]]  - var13 ddv == projectionpoint
+(var3+var23 sdv),{var13,var23,var3}] //Chop;
projectionpoint2= transpoints[[4]] - solrup3[[1,1,2]] ddv;},
check1>0.5l,
{solrup4= Solve[transpoints[[2]]  - var14 ddv == projectionpoint  -
(var4+var24 sdv),{var14,var24,var4}] //Chop;
projectionpoint2= transpoints[[2]] - solrup4[[1,1,2]] ddv;}],

check2≤ -0.5 w,
Which[check1≤ -0.5l,
projectionpoint2 = transpoints[[4]],
Abs[check1]<0.5l,
{solrup2= Solve[transpoints[[3]]  - var12 sdv == projectionpoint
+(var22+var2 ddv),{var12,var22,var2}] //Chop;
projectionpoint2= transpoints[[3]] - solrup2[[1,1,2]] sdv;},
check1≥ 0.5l,
projectionpoint2 = transpoints[[3]]]];

rrup:= Norm[projectionpoint2 - stationcoord];



(*RJB*)

vs:=  (transpoints2[[2]] - transpoints2[[1]] )/ Norm[transpoints2[[1]] -
transpoints2[[2]]] //Chop;
vd :=  (transpoints2[[1]] - transpoints2[[4]] )/ Norm[transpoints2[[1]] -
transpoints2[[4]]] //Chop;

wjb := Norm[transpoints2[[1]] - transpoints2[[4]]];
ms := stationcoord - (middlecoord {1,1,0});
check3:= ms.vs;
check4:= ms.vd;


projectionpoint3 = stationcoord;
Which[check4 ≥0.5wjb ,
Which[check3 ≤-0.5l,
projectionpoint3 = transpoints2[[1]],
Abs[check3] <0.5l ,
{soljb1 = Solve[transpoints2[[1]] + var31 vs == stationcoord -var41 vd,
{var31, var41}] //Chop;
projectionpoint3 = transpoints2[[1]] + soljb1[[1,1,2]] vs;},
check3≥0.5l,
projectionpoint3= transpoints2[[2]]],

Abs[check4] <0.5 wjb,
Which[check3 < -0.5l,
{soljb3 = Solve[transpoints2[[1]] - var33 vd == stationcoord +var43 vs,
{var33, var43}] //Chop;
projectionpoint3  = transpoints2[[1]] - soljb3[[1,1,2]] vd; },
check3>0.5l,
```

```
{soljb4 = Solve[transpoints2[[2]] - var34 vd == stationcoord - var44 vs,
{var34, var44}] //Chop;
projectionpoint3 = transpoints2[[2]] - soljb4[[1,1,2]] vd;}],

check4≤-0.5 wjb,
Which[check3 ≤ -0.5l,
projectionpoint3 = transpoints2[[4]],
Abs[check3] < 0.5l,
{soljb2 = Solve[transpoints2[[4]] + var32 vs == stationcoord +var42 vd,
{var32, var42}] //Chop;
projectionpoint3 = transpoints2[[4]] + soljb2[[1,1,2]] vs;},
check3≥0.5l,
projectionpoint3 = transpoints2[[3]]]];


rjb:= Norm[stationcoord - projectionpoint3];

Return[{rhypo,repi,rrup,rjb,mw,flag1,flag2}];]
```

Anhang 2: RupturePlane3DVisual

```
RupturePlane3DVisual[phis_, delta_,mw_,lon_,lat_, dep_,a_,b_,statlon_,
statlat_]:= Module[{e1,e2,e3,r1,r2,r3,strikedir,allrotations, trans, coord,
size, hypo, station, eo, plane,planepoints,plane2, rhypo,repi,hypocoord,
planeequation,vs,vd,wjb, sdv,ddv,stationcoord, middlecoord,transpoints,
transpoints2,planenormal,planeequation2,projectionline,d,
sol,param,projectionpoint,projectionpoint2,projectionpoint3,mp,check1,check
2,check3,check4,solrup1,solrup2,solrup3,solrup4,paramrup1,paramrup2,paramru
p3,paramrup4,pointyrup1,pointyrup2,pointyrup3,pointyrup4,rrup,ms,soljb1,sol
jb2,soljb3,soljb4,paramjb1,paramjb2,paramjb3,paramjb4,pointyjb1,pointyjb2,p
ointyjb3,pointyjb4,rjb,rupjb,epipoint,rupline,jbline,degreescale,circle,tex
tStyle,north,south,west,east,text,middlepoint,plotColor,w,l,epicoord,flag1,
flag2},


planepoints:={{-0.5,-0.5,0},{-0.5,0.5,0},{0.5,0.5,0},{0.5,-0.5,0}};
(*hypo centre location*)
hypocoord:=Append[XYFromLonLat[{lon, lat}, RefPoint → {lon,lat}]*1000, -
dep];
(*station location*)
stationcoord:=Append[XYFromLonLat[{statlon, statlat},UTMZone →
UTMZoneGet[{statlon, statlat}], RefPoint → {lon,lat}]*1000,0];
epicoord:= hypocoord {1,1,0};

e1:={1,0,0};
e2:={0,1,0};
e3:={0,0,1};

(*Define rotation into strike direction*)
r1:=RotationTransform[-N[phis Degree],e3];
(*strike direction*)
strikedir=r1[e2]//Chop;
(*Define rotation around strike direction to accomodate dip*)
r2:=RotationTransform[N[delta Degree],strikedir];
(*combine all rotations into single transform*)
allrotations=Composition[r2,r1];
```

```
w:=10 ^(-1.01 + 0.32 mw);
l:= 10 ^(-2.44 + 0.59 mw);
(*Define Length and Width of the rupture-area using the Wells&Coppersmith
relations*)

sdv:=allrotations[e2] //Chop;(*strike vector direction*)
ddv:=- allrotations[e1]//Chop; (*dip vector direction*)
(*plane position and translation along strike and dip direction vetors*)
middlecoord:=hypocoord +(a sdv + b ddv);
coord := TranslationTransform[Evaluate[((a sdv + b ddv) + hypocoord)]];
(*size of the rupture plane*)
size:=ScalingTransform[{w,l,0}];
(*combine all transformation into single one*)
trans=Composition[coord,allrotations ,size];
transpoints= (trans[#])&/@(Join[planepoints]);(*transformed points*)
transpoints2= transpoints {{1,1,0},{1,1,0},{1,1,0},{1,1,0}};(*surface
projection of transformed points*)

flag1=Text[" "];
flag2=Text[" "];
If[Positive[transpoints[[2,3]]], flag1=Text["Ruptureareа lies above earth's
surface"]];

If[transpoints[[2,3]]  <  hypocoord[[3]], flag2=Text["Hypocentre is not
inside the ruptureareа"]];
If[transpoints[[3,3]] >   hypocoord[[3]], flag2=Text["Hypocentre is not
inside the ruptureareа"]] ;

(*Errormessages*)

(*Distancecalculations*)


rhypo:= Norm[stationcoord - hypocoord];
repi:= Norm[stationcoord - epicoord];

(*RRUP*)
planenormal:= sdv × ddv; (*Normalvector*)
d:= planenormal . middlecoord; (*Distance to the origin*)
(*planeequation:=planenormal . {x,y,z} -d; (*Equation of the
ruptureplane*)*)
projectionline := stationcoord + h planenormal;(*Line from station
perpendicular to the ruptureplane*)
planeequation := planenormal . projectionline -d; (*Intersection of line
and plane*)
sol :=Solve[planeequation == 0, h];(*Parameter of the line equation to
determine the intersectionpoint*)

projectionpoint:= stationcoord +  sol[[1,1,2]] * planenormal;(*Projection
of the station onto the ruptureplane*)

mp:= projectionpoint - middlecoord;
check1:= mp.sdv;
check2:= mp.ddv;


projectionpoint2 = projectionpoint;
Which[check2≥ 0.5w,
Which[check1 ≤ -0.5l,
projectionpoint2 = transpoints[[1]],
Abs[check1] <0.5l ,
```

```
{solrup1= Solve[transpoints[[1]]   + var11 sdv == projectionpoint  +
(var1+var21 ddv),{var11,var21,var1}] //Chop;
projectionpoint2 =  transpoints[[1]] + solrup1[[1,1,2]] sdv;},
check1≥ 0.5l,
projectionpoint2 = transpoints[[2]]],

Abs[check2]<0.5 w,
Which[check1<-0.5l,
{solrup3= Solve[transpoints[[4]]  - var13 ddv == projectionpoint
+(var3+var23 sdv),{var13,var23,var3}] //Chop;
projectionpoint2= transpoints[[4]] - solrup3[[1,1,2]] ddv;},
check1>0.5l,
{solrup4= Solve[transpoints[[2]]  - var14 ddv == projectionpoint  -
(var4+var24 sdv),{var14,var24,var4}] //Chop;
projectionpoint2= transpoints[[2]] - solrup4[[1,1,2]] ddv;}],

check2≤ -0.5 w,
Which[check1≤ -0.5l,
projectionpoint2 = transpoints[[4]],
Abs[check1]<0.5l,
{solrup2= Solve[transpoints[[3]]  - var12 sdv == projectionpoint
+(var22+var2 ddv),{var12,var22,var2}] //Chop;
projectionpoint2= transpoints[[3]] - solrup2[[1,1,2]] sdv;},
check1≥ 0.5l,
projectionpoint2 = transpoints[[3]]]];

rrup:= Norm[projectionpoint2 - stationcoord];



(*RJB*)

vs:=  (transpoints2[[2]] - transpoints2[[1]] )/ Norm[transpoints2[[1]] -
transpoints2[[2]]] //Chop;
vd :=  (transpoints2[[1]] - transpoints2[[4]] )/ Norm[transpoints2[[1]] -
transpoints2[[4]]] //Chop;

wjb := Norm[transpoints2[[1]] - transpoints2[[4]]];
ms := stationcoord - (middlecoord {1,1,0});
check3:= ms.vs;
check4:= ms.vd;


projectionpoint3 = stationcoord;
Which[check4 ≥0.5wjb ,
Which[check3 ≤-0.5l,
projectionpoint3 = transpoints2[[1]],
Abs[check3] <0.5l ,
{soljb1 = Solve[transpoints2[[1]] + var31 vs == stationcoord -var41 vd,
{var31, var41}] //Chop;
projectionpoint3 = transpoints2[[1]] + soljb1[[1,1,2]] vs;},
check3≥0.5l,
projectionpoint3= transpoints2[[2]]],

Abs[check4] <0.5 wjb,
Which[check3 < -0.5l,
{soljb3 = Solve[transpoints2[[1]] - var33 vd == stationcoord +var43 vs,
{var33, var43}] //Chop;
projectionpoint3  = transpoints2[[1]] - soljb3[[1,1,2]] vd; },
check3>0.5l,
```

```
{soljb4 = Solve[transpoints2[[2]] - var34 vd == stationcoord - var44 vs,
{var34, var44}] //Chop;
projectionpoint3 = transpoints2[[2]] - soljb4[[1,1,2]] vd;}],

check4≤-0.5 wjb,
Which[check3 ≤ -0.5l,
projectionpoint3 = transpoints2[[4]],
Abs[check3] < 0.5l,
{soljb2 = Solve[transpoints2[[4]] + var32 vs == stationcoord +var42 vd,
{var32, var42}] //Chop;
projectionpoint3 = transpoints2[[4]] + soljb2[[1,1,2]] vs;},
check3≥0.5l,
projectionpoint3 = transpoints2[[3]]]];


rjb:= Norm[stationcoord - projectionpoint3];


  (*Plotting Part*)

  degreescale=Graphics3D[Table[Text[Style[ToString[30
i]<>"°",GrayLevel[0.2],"Label",18], {(l +0.1)Sin[30 °
i]+hypocoord[[1]]/1000,(l +0.1)Cos[30 ° i]+hypocoord[[2]]/1000,(-
dep)/1000}],{i,{1,2,4,5,7,8,10,11}}]];

  circle=ParametricPlot3D[ { (l +0.1)  Sin[x]+hypocoord[[1]]/1000, (l +0.1)
Cos[x]+hypocoord[[1]]/1000,(- dep)/1000 },{x,0,2
π},PlotStyle→{GrayLevel[0.6],Dashed,AbsoluteThickness[2]},PlotPoints→20,Box
Ratios→Automatic,Boxed→False,Axes→True];


  textStyle={GrayLevel[0.2],Bold,18};
  north=Graphics3D[Text[Style["N",textStyle],{0.+hypocoord[[1]]/1000,(l
+0.1)+hypocoord[[2]]/1000,(- dep)/1000}]];
  east=Graphics3D[Text[Style["E",textStyle],{(l
+0.1+hypocoord[[1]]/1000),0+hypocoord[[2]]/1000,(- dep)/1000}]];
  west=Graphics3D[Text[Style["W",textStyle],{-(l
+0.1)+hypocoord[[1]]/1000,0.+hypocoord[[2]]/1000,(- dep)/1000}]];
  south=Graphics3D[Text[Style["S",textStyle],{0.+hypocoord[[1]]/1000,-(l
+0.1)+hypocoord[[2]]/1000,(- dep)/1000}]];

  plotColor=RGBColor[0.996,0.973,0.925];

  hypo:= Graphics3D[{Text[Style["Hypocentre",textStyle], hypocoord
+{0,0,0.2}],PointSize[Large],Blue,Point[hypocoord]}];
  station:= Graphics3D[{Text[Style["Station",textStyle], stationcoord
+{0,0,0.2}],PointSize[Large],Black,Point[stationcoord]}];


  epipoint:= Graphics3D[{Text[Style["Epicentre",textStyle], (epicoord
+{0,0,0.2})],PointSize[Large],Brown,Point[epicoord]}];
  rupjb:=Graphics3D[{PointSize[Large],Green,Point[projectionpoint3],Red,
Point[projectionpoint2]}];
  jbline:=stationcoord +0.5( projectionpoint3-stationcoord);
  rupline:= stationcoord + 0.5( projectionpoint2-stationcoord);

  text:= Graphics3D[{Text[Style["Joyner-Boore-Distance",textStyle],
jbline],Line[{stationcoord,projectionpoint3}],Text[Style["Rupture-
Distance",textStyle], rupline],Line[{stationcoord,projectionpoint2}]}];
  plane
:=Graphics3D[{Lighting→{},EdgeForm[GrayLevel[0.5]],Glow[plotColor],Polygon[
transpoints],Glow[plotColor],Polygon[transpoints2]}];
```

```
  title = Text[Style["Distances",textStyle]];
  distancetext = Text[Style[Row[{
       Style["Hypocentral Distance",Italic],"= ",NumberForm[rhypo,{5,1}],
       "m ",Style["Epicentral Distance",Italic],"=
",NumberForm[repi,{5,1}],
       "m ",Style["Rupture Distance",Italic],"= ",NumberForm[rrup,{5,1}],
       "m ",Style["Joyner-Boore-Distance",Italic],"=
",NumberForm[rjb,{5,1}],
       "m "}],GrayLevel[0.2],12]];

  Show[plane,circle,degreescale,north,east,west,south,
epipoint,hypo,station,text,rupjb,
   Axes→True, AxesLabel → {"x in km","y in km","z in
km"},LabelStyle→Directive[Black,Large],PlotLabel→Column[{title,distancetext
}],PlotRange->All,DisplayFunction->Identity
   ]
```

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine weiteren Hilfsmittel oder Quellen als die angegebenen benutzt habe.

Potsdam, dem 8. September 2011

Silvio Schwarz