Berechnung von Polynomnullstellen mittels Clipping

Ausarbeitung zum Praktikum Numerische Mathematik WS 2011/12

Timo Bingmann

1 Motivation und Zielsetzung

Die schnelle numerische Bestimmung der Nullstellen von Polynomen ist ein ebenso altes wie grundlegendes Problem, dessen Lösungsverfahren als Basistechniken in viele modernen rechnergestützten Anwendungen unabdingbar sind. Im Rahmen dieses Praktikums wurde das in 2007 von Michael Bartoň und Bert Jüttler [1] vorgestellte neue Verfahren zur Nullstellenberechnung mittels Clipping der Bestapproximation durch quadratische Polynome implementiert. Darüber hinaus wurde die später von Liu et alii [4] entwickelte Erweiterung dieses Verfahrens auf kubische Annäherungen ebenfalls umgesetzt und beide Algorithmen werden in dieser Ausarbeitung zusammen mit dem klassischen Bézier-Clipping verglichen.

2 Grundlagen: Bézierdarstellung und Basis-Algorithmusschema

Jedes Polynom $p : [0, 1] \to \mathbb{R}$ aus dem $(n+1)$-dimensionalen Vektorraum Π^n der Polynome n-ten Grades lässt sich folgendermaßen in der Basis der Bernstein-Polynome $(B_{i,n})_{i=0,\dots,n}$ darstellen:

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \qquad \text{wobei} \qquad B_{i,n}(t) := \binom{n}{i} t^i (1-t)^{n-i}$$

und $b_i \in \mathbb{R}$ die Bernstein-Bézier-Koeffizienten genannt werden. Die Koeffizienten repräsentieren $n+1$ Kontrollpunkte $(\frac{i}{n}, b_i)_{i=0,\dots,n}$ und bekanntermaßen enthält die konvexe Hülle dieser Kontrollpunkte die Kurve des Polynoms.

Die im Folgenden betrachteten Clipping-Algorithmen finden alle Nullstellen von p im Intervall $[0, 1]$ durch Einschränkung dieser in sukzessive kleiner werdenden Intervallen $[\alpha, \beta]$ nach folgendem Basis-Algorithmusschema.

Vorgegeben ist das Polynom p in einem Suchintervall $[\alpha, \beta]$:

1. Ist $|\alpha - \beta| < \varepsilon$, dann ist die Nullstelle mit einer Genauigkeit $\varepsilon > 0$ einschlossen. Fertig.
2. Sonst bestimmte kleinere Schnittintervalle $[\alpha'_i, \beta'_i]_{i=1,\dots,k} \subset [\alpha, \beta]$, in denen die Nullstellen liegen müssen. Hierzu werden im Folgenden verschiedene Verfahren eingesetzt werden.

3.a Gibt es keine Schnittintervalle, so gibt es keine Nullstellen. Fertig.

3.b Ist $\max_{i=1,\dots,k} |\alpha'_i - \beta'_i| \geq \frac{1}{2}|\alpha - \beta|$, dann sind die Schnittintervalle zu groß und es wird mit Intervallhalbierung weiter verfahren: suche rekursiv in $[\alpha, \frac{1}{2}(\alpha+\beta)]$ und $[\frac{1}{2}(\alpha+\beta), \beta]$.

3.c Sonst suche rekursiv in jedem der k Intervalle $[\alpha'_i, \beta'_i]_{i=1,\dots,k}$.

Am Ende jedes Rekursionzweiges kann eine Nullstelle mit vorgegebener Genauigkeit liegen, wobei noch auf Vorzeichenwechsel geprüft werden muss, um asymptotische Annäherungen an die x-Achse auszuschließen.

Die gesamte Berechnung erfolgt in der Implementierung statt auf $[\alpha, \beta]$ auf dem Einheitsintervall $[0, 1]$ und die Nullstellen werden am Ende entsprechend skaliert. Dies ist notwendig um die Bézier-Koeffizienten von p in den kleineren Intervallen schnell mit dem Algorithmus von de Casteljau interpolieren zu können [5, S.124]. Daher sind alle Plots im Anhang und in den beiliegenden Ablaufprotokolle auf dem Einheitsintervall und entsprechend andere Eingabepolynome $p : [\alpha, \beta] \to \mathbb{R}$ werden vorher durch Skalierung normiert.

Die geometrische Einschließungs-Eigenschaft der konvexen Hülle der Bézier-Koeffizienten liefert das klassische Bézier-Clipping Verfahren `BezClip` [7]: man berechnet das Schnittintervall der Hülle mit der x-Achse im 2. Schritt des Basis-Algorithmusschemas. Dies liefert ein Verfahren mit quadratischer Konvergenz bei einfachen Nullstellen [6], das in Anhang B mit den im Weiteren entwickelten Algorithmen verglichen wird.

3 Schnittintervalle zur Nullstellenbestimmung durch Gradreduktion

Eine neue Möglichkeit Schnittintervalle zu bestimmten ergeben sich durch Anwendung von Gradreduktionstechniken [1, 4]. Hierbei wird ein Polynom n-ten Grades durch quadratische Polynome approximiert, deren Nullstellen sich direkt ausrechnen lassen. Diese Methode wurde möglich durch Jüttlers [3] explizite Formulierung der dualen Basis der Bernstein-Polynome, womit sich eine Vorschrift zur direkten Berechnung der quadratischen Bestapproximation bezüglich der L_2-Norm auf $[0,1]$ in Π^n durch einfache Matrixmultiplikation mit den Bézier-Koeffizienten ableiten lässt. Aus Platzgründen wird hier auf die Entwicklung der Gradreduktionsmatrix $M_{n,k}$ in der Originalarbeit [1, Gl.(12)] und auf die Beispiele im Anhang A verwiesen. Für vorgegebene $n, k \in \mathbb{N}$ transformiert die Matrix $M_{n,k}$ die Bézier-Koeffizienten eines Polynoms $p \in \Pi^n$ in die Koeffizienten der Bestapproximation $q \in \Pi^k$. Dasselbe Verfahren kann verwendet werden, um umgekehrt eine Graderhöhungsmatrix zu generieren.

Mit diesen Techniken lässt sich folgendes Verfahren zur Bestimmung von Schnittintervallen entwickeln. Vorgegeben ist ein Polynom $p = \sum_{i=0}^{n} b_i B_{i,n} \in \Pi^n$ auf dem Einheitsintervall $[0,1]$:

1. Bestimme das quadratische Polynom $q = \sum_{i=0}^{2} c_i B_{i,2} \in \Pi^2$ mit kleinster Abweichung $\|p - q\|_2$ durch Multiplikation mit der Gradreduktionsmatrix $M_{n,2}$.
2. Berechne die Bézier-Koeffizienten $(\tilde{c}_i)_{i=0,\ldots,n}$ des quadratischen Polynoms q in Π^n durch Anwendung der Graderhöhungsmatrix $M_{2,n}$.
3. Bestimme eine Schranke für die Abweichung $\|p - q\|_2$ durch $\delta := \max_{i=0,\ldots,n} |b_i - \tilde{c}_i|$.
4. Finde zwei quadratische Polynome $M := q + \delta$ und $m := q - \delta$, die das Polynom p von oben und unten einschließen.
5. Berechne die Nullstellen der quadratischen Polynome mit der direkten Lösungsformel und bestimme hierdurch bis zu zwei Schnittintervalle, in denen Nullstellen von p liegen können.

Setzt man dieses Verfahren als 2. Schritt zur Bestimmung von Schnittintervalle in das Basis-Algorithmusschema ein, erhält man den Algorithmus `QuadClip`, der die Konvergenzgeschwindigkeit 3 bei einfachen Nullstellen und $\frac{3}{2}$ bei doppelten Nullstellen [1] aufweist.

Dasselbe Verfahren kann auch mit kubischen Polynomen durchgeführt werden [4]. Hierbei werden die Matrizen $M_{n,3}$ und $M_{3,n}$ zur Transformation und die Cardanoschen Formeln zur Nullstellenbestimmung verwendet. Den hierdurch gewonnenen Algorithmus nennen wir `CubeClip` und dieser hat Konvergenzgeschwindigkeit 4 bei einfachen und 2 bei doppelten Nullstellen [4].

4 Überblick der Implementierung

Die drei beschriebenen Verfahren `BezClip`, `QuadClip` und `CubeClip` wurden für das Praktikum in C++ implementiert. Hierbei wurde der etwas ungewöhnliche Weg gewählt, das Programm während der Berechnung wahlweise die Schritte und Zwischenergebnisse als LaTeX-Code ausgeben zu lassen. Dies ermöglicht eine komfortable Formattierung von Polynomen und Plots und in der Beilage finden sich mehrere vom Programm erzeugte, kommentierte Ablaufprotokolle.

Das Programm besteht aus drei großen, aufeinander aufbauenden Modulen. Die Basiskomponente besteht aus Klassen, die Gleitkommazahlen mit verschiedener Präzision speichern und entsprechende mathematische Operationen bereitstellen. Mit diesen Klassen kann dann zwischen Rechnungen mit den Standarddatentypen `double`, `long double` oder der Gleitkomma-Bibliothek `mpfr` [8] umgeschalten werden. In allen weiteren Klassen wird eine dieser Gleitkommadarstellungen durch den jeweiligen Template-Parameter `Numeric` ausgewählt.

Aufbauend auf `Numeric` werden zwei Klassen zur Polynomdarstellung entwickelt: `PolynomialStandard` entspricht der gewohnten Monomdarstellung und `PolynomialBezier` ist eine separate Klasse für die Bézierdarstellung. Diese Klassen enthalten eine Vielzahl von später benötigten Basis-Algorithmen und Operation, darunter Umrechnungsfunktionen von Bézier- zur Monomdarstellung und zurück, den Algorithmus von de Casteljau zur Bisektion einer Bézierdarstellung

in die Koeffizienten des linken und rechten Teilintervalls [5, S.123], Jarvis' March [2, S.955] zur Berechnung der konvexen Hülle der Bézierkurve und schließlich die Formeln zur Bestimmung der Nullstellen von quadratischen und kubischen Polynomen.

Mit Hilfe dieser Polynomklassen wurden dann die drei oben vorgestellten Algorithmen zur Berechnung von Polynomnullstellen als `BezierClip` und `KClip` implementiert. Neben der Formulierung der Gradreduktionsmatrizen und den Cardanoschen Formeln bestand hierbei die Hauptschwierigkeit in der Bestimmung der Schnittintervalle von M und m mit der x-Achse: quadratische Polynome können null, eine oder zwei Nullstellen und kubische eine, zwei oder drei Nullstellen besitzen. Hierdurch ergeben sich jeweils neun verschiedene Kombinationen wie M und m die x-Achse schneiden und entsprechend unterschiedlich müssen die Nullstellen miteinander zu Intervallen gruppiert werden.

Zur Visualisierung, Fehlersuche und Vorbereitung der Geschwindigkeitsmessung wurden acht verschiedene Demos im Programm implementiert, deren Ablaufprotokolle alle in der Beilage als demo$k$.pdf mit $k = 1, \ldots, 8$ zu finden sind.

5 Vergleich der Recheneffizienz der Verfahren und numerische Instabilitäten

Zum Vergleich der Effizienz der drei Algorithmen wurde die Berechnungszeit von Test-Polynomen mit einfachen, dreifachen und zwei nahegelegenen Nullstellen in Relation zur gewünschten Genauigkeit gemessen. Anhang B enthält eine Detailanalyse der Geschwindigkeitsmessung. Eine größere Genauigkeit erfordert eine tiefere Rekursion und höhere Berechnungsdauer.

Die Berechnung der Gradreduktions- und Erhöhungsmatrizen war überraschend aufwendig wegen der auftretenden Binomialkoeffizienten. Da diese Matrizen jedoch vorberechnet werden können, wurde diese Berechnungszeit bei `QuadClip` und `CubeClip` nicht mitgezählt.

Alle Clipping-Verfahren sind aus dem Basis-Algorithmusschema entwickelt, das Bisektion statt selektiver Rekursion bei zu großen Schnittintervallen vorzieht. Enthält das Suchintervall mehrere Nullstellen, so werden diese bei allen Verfahren in der Regel durch Bisektion separiert. Erst bei einer Nullstelle im Suchintervall können die verschiedenen Eigenschaften der drei Algorithmen hervortreten. Eine weitere Untersuchung der Bisektion könnte zu besseren Verfahren für viele Nullstellen führen.

Als für die meisten praktischen Zwecke am Besten geeignete Algorithmus stellt sich `QuadClip` heraus. Das aufwendigere Verfahren `CubeClip` erreicht schneller eine höhere Genauigkeit, insbesondere bei mehrfachen Nullstellen, benötigt jedoch durch die komplexere Nullstellenbestimmung für kubische Polynome eine höhere Gleitkommazahlen-Präzision und es treten öfter numerische Instabilitäten auf. Der Rechenaufwand von `BezClip` ist mit `QuadClip` vergleichbar, da die Berechnung der konvexe Hülle und das Lösen quadratischer Gleichungen ähnlich aufwendig sind. `QuadClip` erreicht aber eine höhere Konvergenzgeschwindigkeit und ist daher effizienter.

Literatur

[1] Michael Bartoň und Bert Jüttler. „Computing roots of polynomials by quadratic clipping“. In: *Computer Aided Geometric Design* 24.3 (2007), S. 125–141. ISSN: 0167-8396.

[2] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest und Charles E. Leiserson. *Introduction to Algorithms*. 2nd Edition. McGraw-Hill Higher Education, 2001. ISBN: 0-262-03293-7.

[3] Bert Jüttler. „The dual basis functions for the Bernstein polynomials“. In: *Advances in Computational Mathematics* 8 (4 1998), S. 345–352. ISSN: 1019-7168.

[4] Ligang Liu, Lei Zhang, Binbin Lin und Guojin Wang. „Fast approach for computing roots of polynomials using cubic clipping“. In: *Computer Aided Geometric Design* 26.5 (2009), S. 547–559. ISSN: 0167-8396.

[5] Franz Locher. *Numerische Mathematik I*. Studienbrief zu Kurs 1271. FernUniversität in Hagen, 1998.

[6] Christian Schulz. „Bézier clipping is quadratically convergent“. In: *Computer Aided Geometric Design* 26.1 (2009), S. 61–74. ISSN: 0167-8396.

[7] T.W. Sederberg und T. Nishita. „Curve intersection using Bézier clipping“. In: *Computer-Aided Design* 22.9 (1990), S. 538–549. ISSN: 0010-4485.

[8] *The GNU MPFR Library*. URL: http://www.mpfr.org/.

A Gradreduktion- und Erhöhungsmatrizen für Grad 5 mit Beispielen

Dieser Abschnitt zeigt beispielhaft die Funktionsweise der Gradreduktion und Wiedererhöhung eines Polynom vom Grad 5. Sei etwa das folgende Polynom $p \in \Pi^5$ in Monom- und Bézierdarstellung gegeben:

$$\begin{aligned} p &= 25X^5 - 35X^4 - 15X^3 + 40X^2 - 15X + 1 \\ &= 1B_{5,5}(X) - 0B_{4,5}(X) + 2.5B_{3,5}(X) - 1B_{2,5}(X) - 2B_{1,5}(X) + 1B_{0,5}(X) \end{aligned}$$

Mit der von Bartoň und Jüttler [1] entwickelten Gradreduktionstechnik lassen sich mit Hilfe folgender Matrizen das Polynom p durch Polynome vom Grad 4 bis 0 annähern. Die Approximationen sind bezüglich der L_2-Norm auf $[0,1]$ bestmöglich und lassen sich durch schlichte Matrixmultiplikation mit den Bézier-Koeffizienten ausrechnen.

$$M_{5,4} = \begin{pmatrix} \frac{251}{252} & \frac{-113}{504} & \frac{1}{12} & \frac{-13}{504} & \frac{1}{252} \\ \frac{5}{252} & \frac{565}{504} & \frac{-5}{12} & \frac{65}{504} & \frac{-5}{252} \\ \frac{-5}{126} & \frac{65}{252} & \frac{5}{6} & \frac{-65}{252} & \frac{5}{126} \\ \frac{5}{126} & \frac{-65}{252} & \frac{5}{6} & \frac{65}{252} & \frac{-5}{126} \\ \frac{-5}{252} & \frac{65}{504} & \frac{-5}{12} & \frac{565}{504} & \frac{5}{252} \\ \frac{1}{252} & \frac{-13}{504} & \frac{1}{12} & \frac{-113}{504} & \frac{251}{252} \end{pmatrix} \quad M_{5,3} = \begin{pmatrix} \frac{121}{126} & \frac{-3}{7} & \frac{1}{6} & \frac{-2}{63} \\ \frac{8}{63} & \frac{37}{42} & \frac{-3}{7} & \frac{11}{126} \\ \frac{-1}{9} & \frac{16}{21} & \frac{1}{21} & \frac{-2}{63} \\ \frac{-2}{63} & \frac{1}{21} & \frac{16}{21} & \frac{-1}{9} \\ \frac{11}{126} & \frac{-3}{7} & \frac{37}{42} & \frac{8}{63} \\ \frac{-2}{63} & \frac{1}{6} & \frac{-3}{7} & \frac{121}{126} \end{pmatrix} \quad M_{5,2} = \begin{pmatrix} \frac{23}{28} & \frac{-3}{7} & \frac{3}{28} \\ \frac{9}{28} & \frac{2}{7} & \frac{-3}{28} \\ 0 & \frac{9}{14} & \frac{-1}{7} \\ \frac{-1}{7} & \frac{9}{14} & 0 \\ \frac{-3}{28} & \frac{2}{7} & \frac{9}{28} \\ \frac{3}{28} & \frac{-3}{7} & \frac{23}{28} \end{pmatrix}$$

$$M_{5,1} = \begin{pmatrix} \frac{11}{21} & \frac{-4}{21} \\ \frac{8}{21} & \frac{-1}{21} \\ \frac{5}{21} & \frac{2}{21} \\ \frac{2}{21} & \frac{5}{21} \\ \frac{-1}{21} & \frac{8}{21} \\ \frac{-4}{21} & \frac{11}{21} \end{pmatrix} \quad M_{5,0} = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \end{pmatrix} \quad M_{4,5} = \begin{pmatrix} 1 & \frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & \frac{4}{5} & \frac{2}{5} & 0 & 0 & 0 \\ 0 & 0 & \frac{3}{5} & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{5} & 1 \end{pmatrix} \quad M_{3,5} = \begin{pmatrix} 1 & \frac{2}{5} & \frac{1}{10} & 0 & 0 & 0 \\ 0 & \frac{3}{5} & \frac{3}{5} & \frac{3}{10} & 0 & 0 \\ 0 & 0 & \frac{3}{10} & \frac{3}{5} & \frac{3}{5} & 0 \\ 0 & 0 & 0 & \frac{1}{10} & \frac{2}{5} & 1 \end{pmatrix}$$

$$M_{2,5} = \begin{pmatrix} 1 & \frac{3}{5} & \frac{3}{10} & \frac{1}{10} & 0 & 0 \\ 0 & \frac{2}{5} & \frac{3}{5} & \frac{3}{5} & \frac{2}{5} & 0 \\ 0 & 0 & \frac{1}{10} & \frac{3}{10} & \frac{3}{5} & 1 \end{pmatrix} \quad M_{1,5} = \begin{pmatrix} 1 & \frac{4}{5} & \frac{3}{5} & \frac{2}{5} & \frac{1}{5} & 0 \\ 0 & \frac{1}{5} & \frac{2}{5} & \frac{3}{5} & \frac{4}{5} & 1 \end{pmatrix} \quad M_{0,5} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Die zum Praktikum implementierten Funktionen zur Polynomreduktion haben folgende gerundete Bestapproximationen $q_4 \in \Pi^4, \ldots, q_0 \in \Pi^0$ zu obigem p berechnet, welche in Abbildung 1 als Diagramme gezeichnet sind. Die übrigen Matrizen $M_{4,5}$ bis $M_{0,5}$ werden im Algorithmus verwendet, um die Bézierdarstellung wieder auf 5 Koeffizienten zu erhöhen.

$$\begin{aligned} q_4 &= 27.5X^4 - 70.5556X^3 + 60.8333X^2 - 17.9762X + 1.09921 \\ q_3 &= -15.5556X^3 + 25.4762X^2 - 10.119X + 0.706349 \\ q_2 &= 2.14286X^2 - 0.785714X - 0.0714286 \\ q_1 &= 1.35714X - 0.428571 \\ q_0 &= 0.25 \end{aligned}$$

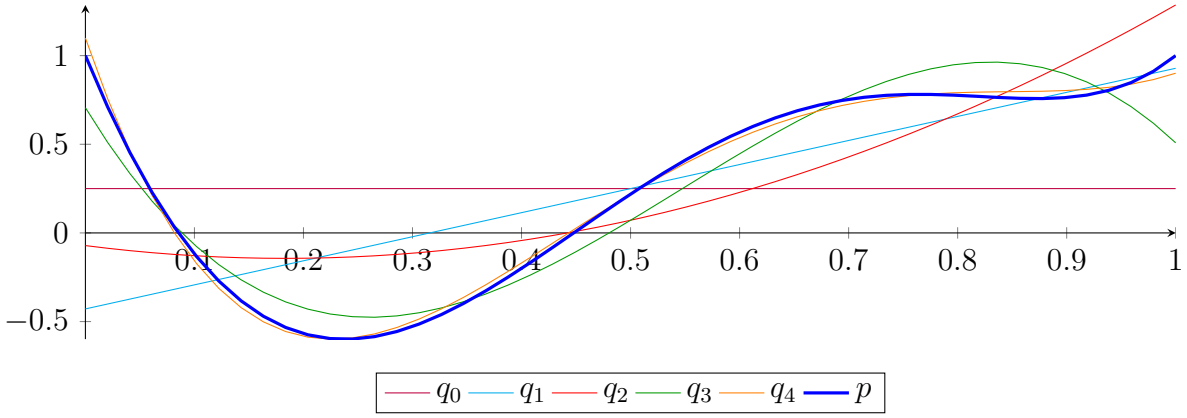

Abbildung 1: Plot der Bestapproximationen zu p in Π^4 bis Π^0

B Detailanalyse der Verfahren: Rechenzeit in Relation zur Genauigkeit

Im Folgenden wird eine detaillierte Analyse der Rechenzeit von Beispielpolynomen [1, 4] relativ zur erzielten Genauigkeit durchgeführt um die Effizienz der drei Algorithmen zu vergleichen. Hierbei werden drei Gruppen von Polynomen verwendet: f_2, f_4, f_8, f_{16} sind Polynome entsprechenden Grades mit einer einfachen Nullstelle bei $\frac{1}{3}$, g_4, g_8, g_{16} besitzen eine dreifache Nullstelle bei $\frac{1}{3}$ und h_2, h_4, h_8, h_{16} haben zwei nahegelegene Nullstellen. Die Polynome werden vom Programm automatisch aus Nullstellen-Darstellung in Monom-Darstellung umgerechnet.

$$
\begin{aligned}
f_2 &:= (t-\tfrac{1}{3})(3-t) \\
f_4 &:= (t-\tfrac{1}{3})(2-t)(t+5)^2 \\
f_8 &:= (t-\tfrac{1}{3})(2-t)^3(t+5)^4 \\
f_{16} &:= (t-\tfrac{1}{3})(2-t)^5(t+5)^{10}
\end{aligned}
\qquad
\begin{aligned}
g_4 &:= (t-\tfrac{1}{3})^3(t-5) \\
g_8 &:= (t-\tfrac{1}{3})^3(2+t)^3(t-5)^2 \\
g_{16} &:= (t-\tfrac{1}{3})^3(2+t)^2(t-5)^7(t+7)^4
\end{aligned}
$$

$$
\begin{aligned}
h_2 &:= (t-0.56)(t-0.57) \\
h_4 &:= (t-0.4)(t-0.40000001)(t+1)(2-t) \\
h_8 &:= (t-0.50000002)(t-0.50000003)(t+5)^3(t+7)^3 \\
h_{16} &:= (t-0.30000008)(t-0.30000009)(6-t)^7(t+5)^6(t+7)
\end{aligned}
$$

Um die Konvergenzgeschwindigkeit hervorzuheben sind die Polynome f_i und g_i so gewählt, dass im untersuchten Intervall $[0, 1]$ nur eine Nullstelle liegt. Alle Geschwindigkeitsmessungen wurden auf einem Intel i7 Prozessor mit 3.07 GHz, 8 MB L3 Cache und 6 GB RAM durchgeführt und das Programm wurde mit `gcc` 4.5.3 in der Optimierungsstufe `-O2` kompiliert. In folgendem Experiment wurde ausschließlich die Gleitkomma-Bibliothek `mpfr` mit 1024-bit Präzision eingesetzt, um etwaige numerische Instabilitäten zu reduzieren und die reine Recheneffizienz der Algorithmen bewerten zu können. Die erzielte Genauigkeit ε wurde mit 10^{-2}, 10^{-4}, 10^{-8} bis 10^{-128} in jedem Schritt quadriert. Die angegebenen Rechenzeiten sind Mittelwerte über eine große Zahl von Iterationen dieselben Rechnung, dabei wurden immer insgesamt mindestens eine

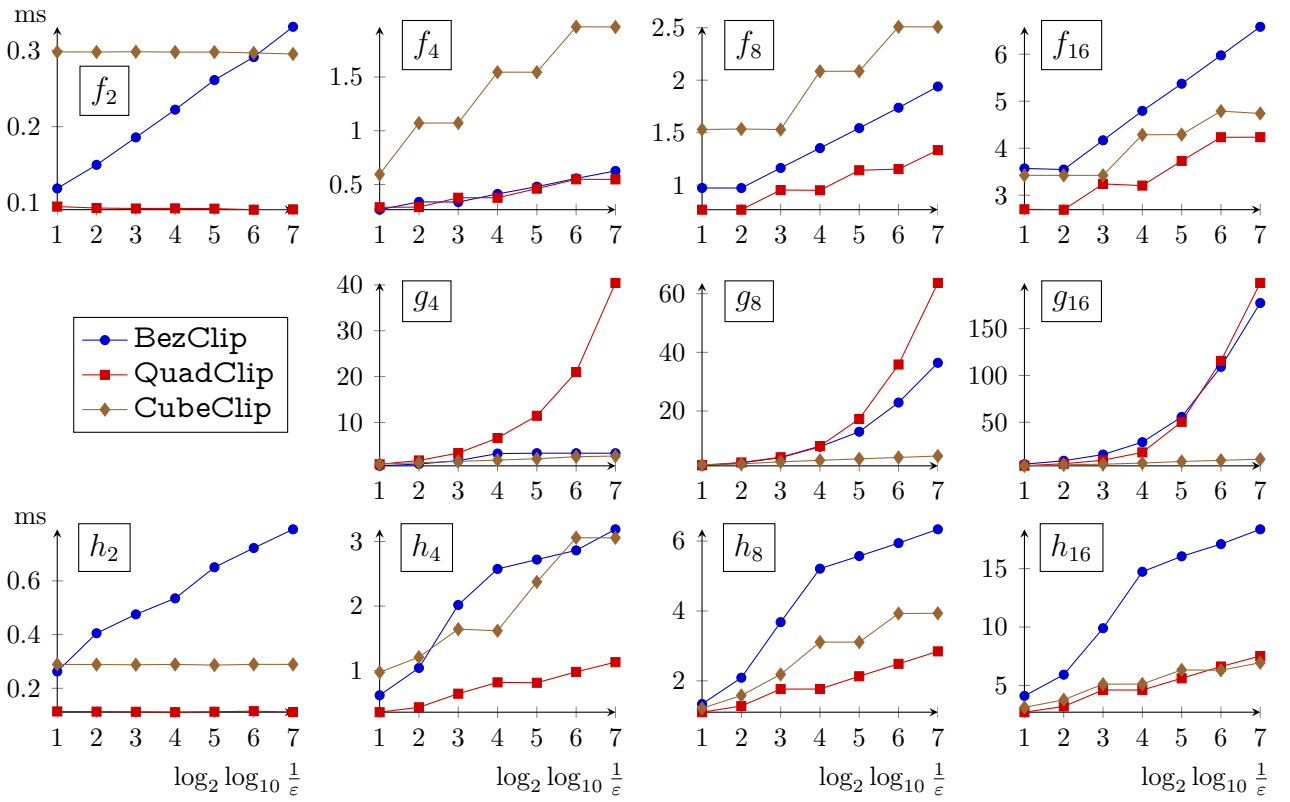

Abbildung 2: Rechenzeit in ms (10^{-3}) für alle Test-Polynome mit `mpfr` (1024-bit Genauigkeit)

Sekunde iteriert oder mindestens 1 000 Wiederholungen durchgerechnet. Ein genaues Ablaufprotokoll der Rechnungen des Geschwindigkeitsvergleichs sind als demo6, demo7 und demo8 in der Beilage zu finden.

Abbildung 2 zeigt die Berechnungszeit für die Nullstellen der Test-Polynome und Tabelle 1 listet die Tiefe der Rekursion, in der die gewünschte Intervallbreite erreicht wurde. Aus den Ablaufprotokollen geht hervor, dass wegen numerischer Instabilität in der tiefsten Rekursionsebene die Nullstelle leider nicht immer zuverlässig erkannt werden konnte. Diese Ungenauigkeit ist schwer vorherzusehen und tritt bei den Nullstellenformeln für quadratische und kubische Polynome auf. Selbst eine weitere Erhöhung der Präzision von `mpfr` hat diese Instabilität nicht verbessert. Trotzdem kann die Recheneffizienz der Algorithmen verglichen werden, da die Instabilität stets genau bei Erreichen der Genauigkeit ε in der maximalen Rekursionstiefe auftritt. Die Graphen in Abbildung 2 zeigen zum Teil einen stufenartigen Verlauf, da die in einem Rekursionsschritt gewonnene Intervallverkleinerung oft auch für die nächsthöhere Genauigkeitsstufe hinreichend ist.

Betrachtet man die Ergebnisse für die Polynome f_i mit einfachen Nullstellen, so erkennt man `QuadClip` als den Algorithmus mit kürzester Berechnungsdauer. Die Anzahl notwendiger Rekursionen ist zwar bei `CubeClip` oft geringer als bei `QuadClip`, doch die elementare Nullstellenberechnung mit den Cardanoschen Formeln ist wesentlich aufwendiger.

Dieser Aufwand zahlt sich jedoch bei den Polynomen g_i mit dreifachen Nullstellen entsprechend aus. Die höhere Konvergenzgeschwindigkeit von `CubeClip` macht diesen Algorithmus bei solchen Polynomen zur besseren Wahl.

Bei den Polynomen mit nahegelegenen Nullstellen traten verschiedene Instabilitätseffekte auf. So hat `CubeClip` bei dem quadratischen Polynom h_2 die dreifache Nullstelle nicht gefunden, da für die im ersten Schritt berechneten approximierenden kubischen Polynome bereits keine Nullstellen mehr gefunden wurden. Derselbe Effekt trat auch bei `QuadClip` auf, wenn auch seltener. Trat diese Instabilität bei der ersten Approximation nicht auf, so wurden bei entsprechend hoher Zielgenauigkeit zuverlässig die beiden Nullstellen getrennt.

In der Beilage demo6 wurden die Polynome f_i auch mit den Standarddatentypen `double` und `long double` durchgerechnet und die Algorithmen lieferten richtige Ergebnisse bis zur Genauigkeit 10^{-16}, ab der die Datentypen nicht mehr hinreichend große Präzision bereitstellten. Dabei ist für alle f_i die Rekursionstiefe von `QuadClip` und `CubeClip` höchstens 5 beziehungsweise 4. Auf Grund der schnelleren Berechnungsschritte ist daher `QuadClip` insgesamt auch für die Standarddatentypen der beste Algorithmus und liefert auch bei $\varepsilon > 10^{-16}$ noch zuverlässig die gesuchten Nullstellen.

ε	10^{-2}			10^{-4}			10^{-8}			10^{-16}			10^{-32}			10^{-64}			10^{-128}		
Alg	B	Q	C	B	Q	C	B	Q	C	B	Q	C	B	Q	C	B	Q	C	B	Q	C
f_2	2	1	1	3	1	1	4	1	1	5	1	1	6	1	1	7	1	1	8	1	1
f_4	2	2	1	3	2	2	3	3	2	4	3	3	5	4	3	6	5	4	7	5	4
f_8	3	2	2	3	2	2	4	3	2	5	3	3	6	4	3	7	4	4	8	5	4
f_{16}	3	2	2	3	2	2	4	3	2	5	3	3	6	4	3	7	5	4	8	5	4
g_4	7	7	5	14	14	7	27	27	9	54	54	11	55	103	13	55	200	16	55	396	17
g_8	7	7	5	14	14	6	27	27	9	54	54	11	81	94	13	134	174	15	205	276	17
g_{16}	6	7	3	12	14	5	23	27	6	45	54	8	90	94	11	179	174	13	293	277	15
h_2	7	1	1	9	1	1	10	1	1	11	1	1	12	1	1	13	1	1	14	1	1
h_4	7	3	3	13	4	4	22	6	5	26	7	5	27	7	6	28	8	7	30	9	7
h_8	5	4	2	9	5	3	18	7	4	22	7	5	23	8	5	24	9	6	25	10	6
h_{16}	4	2	2	7	3	3	14	5	4	18	5	4	19	6	5	20	7	5	21	8	6

Tabelle 1: Maximale Rekursionstiefe für alle Test-Polynome mit `mpfr` (1024-bit Genauigkeit), hierbei stehen `B` für `BezClip`, `Q` für `QuadClip` und `C` für `CubeClip`.