Complexity of Real Approximation: Brent Revisited

Chee Yap

Courant Institute of Mathematical Sciences
New York University

and

Korea Institute for Advanced Study
Seoul, Korea

Joint work with Zilin Du and Vikram Sharma.

I. COMPLEXITY OF MULTIPRECISION COMPUTATION

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
    - ∗ Algebraic School (Blum-Shub-Smale, . . .
    - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications
    ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
  - ∗ Algebraic School (Blum-Shub-Smale, . . .
  - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications

  ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
  - ∗ Algebraic School (Blum-Shub-Smale, . . .
  - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications

  ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
  - ∗ Algebraic School (Blum-Shub-Smale, . . .
  - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications

  ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
  - ∗ Algebraic School (Blum-Shub-Smale, . . .
  - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications
  - ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Introduction: Current Interest in Real Computation

- Foundation of scientific and engineering computation
- Inadequacy of standard computability/complexity theory
- Two current schools of thought
  - ∗ Algebraic School (Blum-Shub-Smale, . . .
  - ∗ Analytic School (Turing (1936), Grzegorczyk (1955), Weihrauch, Ko,. . .
- Multiprecision computation ought to be part of this foundation
- Numerous applications

  ∗ Cryptography and number theory, Theorem proving, robust geometric algorithms, mathematical exploration of conjectures, etc

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:

- Standard elementary functions $(\exp, \log, \sin,$ etc) can be evaluated to n-bits in time $O(M(n)\log^{O(1)}(n))$

- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.

- Linear reducibilities among these problems:
  - Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arcsinh}) \equiv E(\mathrm{arccosh}) \equiv E(\exp) \equiv E(\log)$

- These results remain unsurpassed
  - There are various extensions, e.g., van der Hoeven on holonomic functions
  - Are most of the problems in this area essentially solved?

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:
- Standard elementary functions $(\exp, \log, \sin,$ etc$)$ can be evaluated to n-bits in time $O(M(n)\log^{O(1)}(n))$
- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.
- Linear reducibilities among these problems:
  - Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\text{arcsinh}) \equiv E(\text{arccosh}) \equiv E(\exp) \equiv E(\log)$
- These results remain unsurpassed
  - There are various extensions, e.g., van der Hoeven on holonomic functions
  - Are most of the problems in this area essentially solved?

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:

- Standard elementary functions $(\exp, \log, \sin,$ etc$)$ can be evaluated to n-bits in time $O(M(n) \log^{O(1)}(n))$

- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.

- Linear reducibilities among these problems:
  - Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arcsinh}) \equiv E(\mathrm{arccosh}) \equiv E(\exp) \equiv E(\log)$

- These results remain unsurpassed
  - There are various extensions, e.g., van der Hoeven on holonomic functions
  - Are most of the problems in this area essentially solved?

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:
- Standard elementary functions $(\exp, \log, \sin,$ etc$)$ can be evaluated to n-bits in time $O(M(n)\log^{O(1)}(n))$
- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - ∗ If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.
- Linear reducibilities among these problems:
  - ∗ Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - ∗ $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - ∗ $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arcsinh}) \equiv E(\mathrm{arccosh}) \equiv E(\exp) \equiv E(\log)$
- These results remain unsurpassed
  - ∗ There are various extensions, e.g., van der Hoeven on holonomic functions
  - ∗ Are most of the problems in this area essentially solved?

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:

- Standard elementary functions $(\exp, \log, \sin,$ etc$)$ can be evaluated to n-bits in time $O(M(n)\log^{O(1)}(n))$

- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.

- Linear reducibilities among these problems:
  - Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arcsinh}) \equiv E(\mathrm{arccosh}) \equiv E(\exp) \equiv E(\log)$

- These results remain unsurpassed
  - There are various extensions, e.g., van der Hoeven on holonomic functions
  - Are most of the problems in this area essentially solved?

Brent's Work in Complexity of Multiprecision Computation

- Remarkable series of papers by Brent over 30 years ago established:
- Standard elementary functions ($\exp, \log, \sin,$ etc) can be evaluated to n-bits in time $O(M(n)\log^{O(1)}(n))$
- Under natural conditions, zeros of $F(y)$ is equivalent to evaluating $F(y)$.
  - If $F(y)$ can be evaluated in time $O(M(n)\phi(n))$, then the inverse function $f(x)$ such that $F(f(x)) = x$ can be evaluated to n-bits in time $O(M(n)\phi(n))$.
- Linear reducibilities among these problems:
  - Multiplication Equivalence class: $M \equiv D \equiv I \equiv R \equiv S$
  - $E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv E(\arccos) \equiv E(\arctan$
  - $E(\sinh) \equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arcsinh}) \equiv E(\mathrm{arccosh}) \equiv E(\exp) \equiv E(\log)$
- These results remain unsurpassed
  - There are various extensions, e.g., van der Hoeven on holonomic functions
  - Are most of the problems in this area essentially solved?

Brent's Axioms

- Brent's multiprecision model was described in his 1976 JACM article
  - "Fast Multiple-Precision Evaluation of Elementary Functions"
  - We call them "axioms" here

- AXIOM 1: Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$.

- AXIOM 2: Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result.
  - $M(n)$ is the time to multiply two n-bit integers

- AXIOM 3: The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$.

Brent's Axioms

- Brent's multiprecision model was described in his 1976 JACM article
  - "Fast Multiple-Precision Evaluation of Elementary Functions"
  - We call them "axioms" here

- AXIOM 1: Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$.

- AXIOM 2: Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result.
  - $M(n)$ is the time to multiply two n-bit integers

- AXIOM 3: The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$.

Brent's Axioms

- Brent's multiprecision model was described in his 1976 JACM article
  - "Fast Multiple-Precision Evaluation of Elementary Functions"
  - We call them "axioms" here

- AXIOM 1: Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$.

- AXIOM 2: Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result.
  - $M(n)$ is the time to multiply two n-bit integers

- AXIOM 3: The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$.

Brent's Axioms

- Brent's multiprecision model was described in his 1976 JACM article
  - * "Fast Multiple-Precision Evaluation of Elementary Functions"
  - * We call them "axioms" here
- AXIOM 1: Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$.
- AXIOM 2: Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result.
  - * $M(n)$ is the time to multiply two n-bit integers
- AXIOM 3: The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$.

Brent's Axioms

- Brent's multiprecision model was described in his 1976 JACM article
  - "Fast Multiple-Precision Evaluation of Elementary Functions"
  - We call them "axioms" here

- AXIOM 1: Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$.

- AXIOM 2: Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result.
  - $M(n)$ is the time to multiply two n-bit integers

- AXIOM 3: The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$.

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results

- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f\cdot 2^{-\lfloor |f|\rfloor} \in [1,2)$
  - Represented by the (exponent/fraction) pair $\langle e, f\rangle$

- Precision of $\langle e, f\rangle$ is $\lg|f|$

- Size of $\langle e, f\rangle$ is the pair $(\lg|e|, \lg|f|)$

- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results
- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f \cdot 2^{-\lfloor |f| \rfloor} \in [1,2)$
  - Represented by the (exponent/fraction) pair $\langle e, f\rangle$
- Precision of $\langle e, f\rangle$ is $\lg |f|$
- Size of $\langle e, f\rangle$ is the pair $(\lg |e|, \lg |f|)$
- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results
- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f \cdot 2^{-\lfloor |f| \rfloor} \in [1,2)$
  - Represented by the (exponent/fraction) pair $\langle e, f\rangle$
- Precision of $\langle e, f\rangle$ is $\lg|f|$
- Size of $\langle e, f\rangle$ is the pair $(\lg|e|, \lg|f|)$
- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results
- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f \cdot 2^{-\lfloor |f| \rfloor} \in [1,2)$
    - $*$ Represented by the (exponent/fraction) pair $\langle e, f\rangle$
- Precision of $\langle e, f\rangle$ is $\lg |f|$
- Size of $\langle e, f\rangle$ is the pair $(\lg |e|, \lg |f|)$
- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results
- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f \cdot 2^{-\lfloor |f| \rfloor} \in [1, 2)$
  - Represented by the (exponent/fraction) pair $\langle e, f\rangle$
- Precision of $\langle e, f\rangle$ is $\lg|f|$
- Size of $\langle e, f\rangle$ is the pair $(\lg|e|, \lg|f|)$
- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

BigFloats or Dyadics

- Multi-precision floating point numbers (bigfloats, dyadics) are used to establish these results

- A bigfloat number has the form $2^e\langle f\rangle$ where $\langle f\rangle := f \cdot 2^{-\lfloor |f| \rfloor} \in [1, 2)$
  - $*$ Represented by the (exponent/fraction) pair $\langle e, f\rangle$

- Precision of $\langle e, f\rangle$ is $\lg |f|$

- Size of $\langle e, f\rangle$ is the pair $(\lg |e|, \lg |f|)$

- Set of dyadic numbers: $\mathbb{D} := \mathbb{Z}[\frac{1}{2}] = \{m2^n : m, n \in \mathbb{Z}\}$

Error and Accuracy

- Let $x, \widetilde{x}, \varepsilon, n \in \mathbb{R}$
- Write "$x \pm \varepsilon$" to denote *some* value of the form $x + \theta\varepsilon$ where $|\theta| \leq 1$
  - The θ variable is implicit
- Say $\widetilde{x}$ is an n-bit absolute approximation of x if $\widetilde{x} = x \pm 2^{-n}$
  - $\widetilde{x}$ is an n-bit relative approximation of x if $\widetilde{x} = x(1 \pm 2^{-n})$
  - We then say $\widetilde{x}$ has n-bits of (absolute/relative) accuracy
- Write: $[x]_n$ for $x(1 \pm 2^{-n})$, and $\langle x \rangle_n$ for $x \pm 2^{-n}$

Error and Accuracy

- Let $x, \widetilde{x}, \varepsilon, n \in \mathbb{R}$
- Write "$x \pm \varepsilon$" to denote *some* value of the form $x + \theta\varepsilon$ where $|\theta| \leq 1$
  - $*$ The θ variable is implicit
- Say $\widetilde{x}$ is an n-bit absolute approximation of x if $\widetilde{x} = x \pm 2^{-n}$
  - $*$ $\widetilde{x}$ is an n-bit relative approximation of x if $\widetilde{x} = x(1 \pm 2^{-n})$
  - $*$ We then say $\widetilde{x}$ has n-bits of (absolute/relative) accuracy
- Write: $[x]_n$ for $x(1 \pm 2^{-n})$, and $\langle x \rangle_n$ for $x \pm 2^{-n}$

Error and Accuracy

- Let $x, \widetilde{x}, \varepsilon, n \in \mathbb{R}$

- Write "$x \pm \varepsilon$" to denote *some* value of the form $x + \theta\varepsilon$ where $|\theta| \leq 1$
  * The θ variable is implicit

- Say $\widetilde{x}$ is an n-bit absolute approximation of x if $\widetilde{x} = x \pm 2^{-n}$
  * $\widetilde{x}$ is an n-bit relative approximation of x if $\widetilde{x} = x(1 \pm 2^{-n})$
  * We then say $\widetilde{x}$ has n-bits of (absolute/relative) accuracy

- Write: $[x]_n$ for $x(1 \pm 2^{-n})$, and $\langle x \rangle_n$ for $x \pm 2^{-n}$

Error and Accuracy

- Let $x, \widetilde{x}, \varepsilon, n \in \mathbb{R}$

- Write "$x \pm \varepsilon$" to denote *some* value of the form $x + \theta\varepsilon$ where $|\theta| \leq 1$
  - The θ variable is implicit

- Say $\widetilde{x}$ is an n-bit absolute approximation of x if $\widetilde{x} = x \pm 2^{-n}$
  - $\widetilde{x}$ is an n-bit relative approximation of x if $\widetilde{x} = x(1 \pm 2^{-n})$
  - We then say $\widetilde{x}$ has n-bits of (absolute/relative) accuracy

- Write: $[x]_n$ for $x(1 \pm 2^{-n})$, and $\langle x \rangle_n$ for $x \pm 2^{-n}$

II. BRENT'S COMPLEXITY MODEL

AXIOM 1: Local/Global/Uniform Complexity

- *"Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$."*

- Interpretation: real numbers $x \in [a, b]$ for fixed a, b
  - ∗ If $x = \langle e, f \rangle$, then $|e| = O(1)$
  - ∗ SO, Brent's complexity statements are about "local complexity"

- Let F be a family of real functions, $f \in F$
  - ∗ LOCAL complexity: $T_{f,x}(n)$ is time to evalute $f(x)$ to n-bits
  - ∗ GLOBAL complexity: $T_f(x, n)$ is time to evaluate $f(x)$ to n-bits
  - ∗ UNIFORM complexity: $T(f, x, n)$ is time to evaluate $f(x)$ to n-bits

AXIOM 1: Local/Global/Uniform Complexity

- *"Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$."*

- Interpretation: real numbers $x \in [a, b]$ for fixed a, b
  - $*$ If $x = \langle e, f \rangle$, then $|e| = O(1)$
  - $*$ SO, Brent's complexity statements are about "local complexity"

- Let F be a family of real functions, $f \in F$
  - $*$ LOCAL complexity: $T_{f,x}(n)$ is time to evalute $f(x)$ to n-bits
  - $*$ GLOBAL complexity: $T_f(x, n)$ is time to evaluate $f(x)$ to n-bits
  - $*$ UNIFORM complexity: $T(f, x, n)$ is time to evaluate $f(x)$ to n-bits

AXIOM 1: Local/Global/Uniform Complexity

- *"Real numbers which are not too large or small can be approximated by floating point numbers with relative error $O(2^{-n})$."*

- Interpretation: real numbers $x \in [a, b]$ for fixed a, b
  - ∗ If $x = \langle e, f \rangle$, then $|e| = O(1)$
  - ∗ SO, Brent's complexity statements are about "local complexity"

- Let F be a family of real functions, $f \in F$
  - ∗ LOCAL complexity: $T_{f,x}(n)$ is time to evalute $f(x)$ to n-bits
  - ∗ GLOBAL complexity: $T_f(x, n)$ is time to evaluate $f(x)$ to n-bits
  - ∗ UNIFORM complexity: $T(f, x, n)$ is time to evaluate $f(x)$ to n-bits

EXAMPLE: Uniform Evaluation of Polynomials

- Let $F = \mathbb{D}[X]$
  - $f \in F$ where $f = \sum_{i=0}^{d} a_i X^i$
  - and $-L < \lg |a_i| < L$
  - Let $T(d, L, L_x, n)$ be worst case time to evaluate $f(x)$ to absolute n-bits, where $-L_x < \lg |x| < L_x$

- LEMMA [SDY'05]:
  - $T(d, L, L_x, n) = O(dM(n + L + dL_x))$

- Local complexity is $T(n) = O(M(n))$, when f, x are fixed
  - Global complexity is exponential in $\lg L_x$, as x varies
  - Uniform complexity is exponential in $\lg L$, as f also varies
  - Question: what is the optimal uniform complexity for evaluating polynomials?

- In general, the uniform and global complexity for most families are currently open
  - Brent's genius is to realize that the situation is much cleaner under local complexity

EXAMPLE: Uniform Evaluation of Polynomials

- Let $F = \mathbb{D}[X]$
  - $f \in F$ where $f = \sum_{i=0}^{d} a_i X^i$
  - and $-L < \lg |a_i| < L$
  - Let $T(d, L, L_x, n)$ be worst case time to evaluate $f(x)$ to absolute n-bits, where $-L_x < \lg |x| < L_x$
- LEMMA [SDY'05]:
  - $T(d, L, L_x, n) = O(dM(n + L + dL_x))$
- Local complexity is $T(n) = O(M(n))$, when f, x are fixed
  - Global complexity is exponential in $\lg L_x$, as x varies
  - Uniform complexity is exponential in $\lg L$, as f also varies
  - Question: what is the optimal uniform complexity for evaluating polynomials?
- In general, the uniform and global complexity for most families are currently open
  - Brent's genius is to realize that the situation is much cleaner under local complexity

EXAMPLE: Uniform Evaluation of Polynomials

- Let $F = \mathbb{D}[X]$
  - $f \in F$ where $f = \sum_{i=0}^{d} a_i X^i$
  - and $-L < \lg |a_i| < L$
  - Let $T(d, L, L_x, n)$ be worst case time to evaluate $f(x)$ to absolute n-bits, where $-L_x < \lg |x| < L_x$

- LEMMA [SDY'05]:
  - $T(d, L, L_x, n) = O(dM(n + L + dL_x))$

- Local complexity is $T(n) = O(M(n))$, when f, x are fixed
  - Global complexity is exponential in $\lg L_x$, as x varies
  - Uniform complexity is exponential in $\lg L$, as f also varies
  - Question: what is the optimal uniform complexity for evaluating polynomials?

- In general, the uniform and global complexity for most families are currently open
  - Brent's genius is to realize that the situation is much cleaner under local complexity

EXAMPLE: Uniform Evaluation of Polynomials

- Let $F = \mathbb{D}[X]$
  - $*$ $f \in F$ where $f = \sum_{i=0}^{d} a_i X^i$
  - $*$ and $-L < \lg |a_i| < L$
  - $*$ Let $T(d, L, L_x, n)$ be worst case time to evaluate $f(x)$ to absolute n-bits, where $-L_x < \lg |x| < L_x$

- LEMMA [SDY'05]:
  - $*$ $T(d, L, L_x, n) = O(dM(n + L + dL_x))$

- Local complexity is $T(n) = O(M(n))$, when f, x are fixed
  - $*$ Global complexity is exponential in $\lg L_x$, as x varies
  - $*$ Uniform complexity is exponential in $\lg L$, as f also varies
  - $*$ Question: what is the optimal uniform complexity for evaluating polynomials?

- In general, the uniform and global complexity for most families are currently open
  - $*$ Brent's genius is to realize that the situation is much cleaner under local complexity

EXAMPLE: Uniform Evaluation of Hypergeometric Functions

- Let F be the family of hypergeometric functions ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x)$

  * a's and b's are rational numbers with ℓ-bit numerator and denominators
  * x has total size m (i.e., $m \geq s + p$ where size of x is (s, p)

- THEOREM [DY'05, D'06]:

  * The uniform complexity of evaluating hypergeometric functions to absolute n-bits is

$$O(K^2 M(n + (q+1)K \lg K + Km))$$

where $K = 4^m \left(n + 2^{4(q+1)(2(q+1)^2\ell + m)}\right)$

  * So, local complexity is $O(M(n))$
  * and uniform complexity is single exponential in ℓ, m, q.

- The uniform procedure requires nontrivial estimates based on the hypergeometric parameters

  * It is open whether there is a uniform procedure to evaluate hypergeometric functions to relative n-bits

EXAMPLE: Uniform Evaluation of Hypergeometric Functions

- Let F be the family of hypergeometric functions ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x)$
  - $*$ a's and b's are rational numbers with ℓ-bit numerator and denominators
  - $*$ x has total size m (i.e., $m \geq s + p$ where size of x is (s, p)
- THEOREM [DY'05, D'06]:
  $*$ The uniform complexity of evaluating hypergeometric functions to absolute n-bits is
  $$O(K^2 M(n + (q+1)K \lg K + Km))$$
  where $K = 4^m \left(n + 2^{4(q+1)(2(q+1)^2 \ell + m)}\right)$
  - $*$ So, local complexity is $O(M(n))$
  - $*$ and uniform complexity is single exponential in ℓ, m, q.
- The uniform procedure requires nontrivial estimates based on the hypergeometric parameters
  $*$ It is open whether there is a uniform procedure to evaluate hypergeometric functions to relative n-bits

EXAMPLE: Uniform Evaluation of Hypergeometric Functions

- Let F be the family of hypergeometric functions ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x)$

    * a's and b's are rational numbers with ℓ-bit numerator and denominators
    * x has total size m (i.e., $m \geq s + p$ where size of x is (s, p)

- THEOREM [DY'05, D'06]:
    * The uniform complexity of evaluating hypergeometric functions to absolute n-bits is

$$O(K^2 M(n + (q+1)K \lg K + Km))$$

where $K = 4^m \left(n + 2^{4(q+1)(2(q+1)^2\ell + m)}\right)$

    * So, local complexity is $O(M(n))$
    * and uniform complexity is single exponential in ℓ, m, q.

- The uniform procedure requires nontrivial estimates based on the hypergeometric parameters
    * It is open whether there is a uniform procedure to evaluate hypergeometric functions to relative n-bits

EXAMPLE: Uniform Evaluation of Hypergeometric Functions

- Let F be the family of hypergeometric functions ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x)$

  * a's and b's are rational numbers with ℓ-bit numerator and denominators
  * x has total size m (i.e., $m \geq s + p$ where size of x is (s, p)

- THEOREM [DY'05, D'06]:

  * The uniform complexity of evaluating hypergeometric functions to absolute n-bits is

$$O(K^2 M(n + (q+1)K \lg K + Km))$$

where $K = 4^m \left(n + 2^{4(q+1)(2(q+1)^2\ell+m)}\right)$

  * So, local complexity is $O(M(n))$
  * and uniform complexity is single exponential in ℓ, m, q.

- The uniform procedure requires nontrivial estimates based on the hypergeometric parameters

  * It is open whether there is a uniform procedure to evaluate hypergeometric functions to relative n-bits

AXIOM 2: Weak versus Strong Mode of Computation

- *"Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result."*

- At issue: input numbers can have their own precision m, independent of output precision n

- Interpretation : $T_M(L,m,n) = O(M(n)), \quad T_A(L,m,n) = O(M(n))$
  - $T_M(L,m,n)$ is the time to multiply inputs of size (L,m) to relative n-bits
  - $T_A(L,m,n)$ is the time to add inputs of size (L,m) to relative n-bits

- But addition can have catastrophic cancellation
  - E.g., Let $x = 3 \cdot 2^{-m-1} = \langle -m, 3 \rangle = +0.\underbrace{0\cdots0}_{m-1}11$
  - and $y = -2^{-m} = \langle -m, -1 \rangle = -0.\underbrace{0\cdots0}_{m-1}01.$
  - Time to compute $[x+y]_n$ is $\Omega(m)$ for any $n \geq 1$

- WEAK Mode of Floating Point Computation
  - i.e., Generalized IEEE standard of floating point arithmetic

AXIOM 2: Weak versus Strong Mode of Computation

- *"Floating-point addition and multiplication can be performed in* $O(M(n))$ *operations, with relative error* $O(2^{-n})$ *in the result."*

- At issue: input numbers can have their own precision m, independent of output precision n

- Interpretation : $T_M(L, m, n) = O(M(n)), \quad T_A(L, m, n) = O(M(n))$
  - $T_M(L, m, n)$ is the time to multiply inputs of size (L, m) to relative n-bits
  - $T_A(L, m, n)$ is the time to add inputs of size (L, m) to relative n-bits

- But addition can have catastrophic cancellation
  - E.g., Let $x = 3 \cdot 2^{-m-1} = \langle -m, 3 \rangle = +0.\underbrace{0 \cdots 0}_{m-1}11$
  - and $y = -2^{-m} = \langle -m, -1 \rangle = -0.\underbrace{0 \cdots 0}_{m-1}01.$
  - Time to compute $[x + y]_n$ is $\Omega(m)$ for any $n \geq 1$

- WEAK Mode of Floating Point Computation
  - i.e., Generalized IEEE standard of floating point arithmetic

AXIOM 2: Weak versus Strong Mode of Computation

- *"Floating-point addition and multiplication can be performed in* $O(M(n))$ *operations, with relative error* $O(2^{-n})$ *in the result."*

- At issue: input numbers can have their own precision m, independent of output precision n

- Interpretation : $T_M(L, m, n) = O(M(n)), \quad T_A(L, m, n) = O(M(n))$
  - $T_M(L, m, n)$ is the time to multiply inputs of size (L, m) to relative n-bits
  - $T_A(L, m, n)$ is the time to add inputs of size (L, m) to relative n-bits

- But addition can have catastrophic cancellation
  - E.g., Let $x = 3 \cdot 2^{-m-1} = \langle -m, 3 \rangle = +0.\underbrace{0 \cdots 0}_{m-1}11$
  - and $y = -2^{-m} = \langle -m, -1 \rangle = -0.\underbrace{0 \cdots 0}_{m-1}01.$
  - Time to compute $[x + y]_n$ is $\Omega(m)$ for any $n \geq 1$

- WEAK Mode of Floating Point Computation
  - i.e., Generalized IEEE standard of floating point arithmetic

AXIOM 2: Weak versus Strong Mode of Computation

- *"Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with relative error $O(2^{-n})$ in the result."*
- At issue: input numbers can have their own precision m, independent of output precision n
- Interpretation : $T_M(L,m,n) = O(M(n)), \quad T_A(L,m,n) = O(M(n))$
  - $T_M(L,m,n)$ is the time to multiply inputs of size (L,m) to relative n-bits
  - $T_A(L,m,n)$ is the time to add inputs of size (L,m) to relative n-bits
- But addition can have catastrophic cancellation
  - E.g., Let $x = 3 \cdot 2^{-m-1} = \langle -m, 3 \rangle = +0.\underbrace{0 \cdots 0}_{m-1}11$
  - and $y = -2^{-m} = \langle -m, -1 \rangle = -0.\underbrace{0 \cdots 0}_{m-1}01.$
  - Time to compute $[x+y]_n$ is $\Omega(m)$ for any $n \geq 1$
- WEAK Mode of Floating Point Computation
  - i.e., Generalized IEEE standard of floating point arithmetic

* Given an algorithm A in ideal arithmetic, let A_θ be implementation of each operation using precision θ
* Thus, $T_A(L, m, n) = O(M(n))$ holds only in the WEAK Mode

- STRONG Mode of Floating Point Computation
  * Algorithms actively modify the precision of its operations during computation
  * E.g., in Brent's self-adjusting Newton methods

∗ Given an algorithm A in ideal arithmetic, let A_θ be implementation of each operation using precision θ

∗ Thus, $T_A(L, m, n) = O(M(n))$ holds only in the WEAK Mode

- STRONG Mode of Floating Point Computation
  - ∗ Algorithms actively modify the precision of its operations during computation
  - ∗ E.g., in Brent's self-adjusting Newton methods

* Given an algorithm A in ideal arithmetic, let A_θ be implementation of each operation using precision θ
* Thus, $T_A(L, m, n) = O(M(n))$ holds only in the WEAK Mode

- STRONG Mode of Floating Point Computation
  * Algorithms actively modify the precision of its operations during computation
  * E.g., in Brent's self-adjusting Newton methods

AXIOM 3: Pointer Machines versus Turing Machines

- *"The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$."*

- Interpretation: let $B(L, m, n)$ be the time to compute $[x]_n$ given any bigfloat x of size (L, m).
  - The axiom says $B(L, m, n) = O(M(n))$

- Brent's ultimate computational model is the (multitape) Turing machine
  - Thus $M(n) = O(n \lg n \lg \lg n)$ (Strassen-Schönhage)
  - Note that $B(L, m, n) = O(M(n) + L)$ on a Turing machine, and since $L = O(1)$, Axiom 3 holds

- If we consider more general classes of real computation, involving matrices
  - It is no longer obvious that $B(L, m, n) = O(M(n))$ can be simultaneously achieved for all the numbers in the matrix

AXIOM 3: Pointer Machines versus Turing Machines

- *"The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$."*

- Interpretation: let $B(L, m, n)$ be the time to compute $[x]_n$ given any bigfloat x of size (L, m).
  - The axiom says $B(L, m, n) = O(M(n))$

- Brent's ultimate computational model is the (multitape) Turing machine
  - Thus $M(n) = O(n \lg n \lg \lg n)$ (Strassen-Schönhage)
  - Note that $B(L, m, n) = O(M(n) + L)$ on a Turing machine, and since $L = O(1)$, Axiom 3 holds

- If we consider more general classes of real computation, involving matrices
  - It is no longer obvious that $B(L, m, n) = O(M(n))$ can be simultaneously achieved for all the numbers in the matrix

AXIOM 3: Pointer Machines versus Turing Machines

- *"The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$."*

- Interpretation: let $B(L, m, n)$ be the time to compute $[x]_n$ given any bigfloat x of size (L, m).
  - * The axiom says $B(L, m, n) = O(M(n))$

- Brent's ultimate computational model is the (multitape) Turing machine
  - * Thus $M(n) = O(n \lg n \lg \lg n)$ (Strassen-Schönhage)
  - * Note that $B(L, m, n) = O(M(n) + L)$ on a Turing machine, and since $L = O(1)$, Axiom 3 holds

- If we consider more general classes of real computation, involving matrices
  - * It is no longer obvious that $B(L, m, n) = O(M(n))$ can be simultaneously achieved for all the numbers in the matrix

AXIOM 3: Pointer Machines versus Turing Machines

- *"The precision n is a variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating point number with precision m, for any positive $m < n$."*

- Interpretation: let $B(L, m, n)$ be the time to compute $[x]_n$ given any bigfloat x of size (L, m).
  * The axiom says $B(L, m, n) = O(M(n))$

- Brent's ultimate computational model is the (multitape) Turing machine
  * Thus $M(n) = O(n \lg n \lg \lg n)$ (Strassen-Schönhage)
  * Note that $B(L, m, n) = O(M(n) + L)$ on a Turing machine, and since $L = O(1)$, Axiom 3 holds

- If we consider more general classes of real computation, involving matrices
  * It is no longer obvious that $B(L, m, n) = O(M(n))$ can be simultaneously achieved for all the numbers in the matrix

Pointer Machines

- To preserve Axiom 3 in the more general setting, we propose to use Schöhage's elegant and flexible model of Pointer Machines
  - $M(n) = O(n)$ in this model (Schönhage)
  - Much nicer that $O(n \lg n \lg \ln n)$!

- LEMMA (cf. [SDY'05]) Assume the Pointer machine model
  - Give k-vectors U and V whose entries are floating point numbers of size (L, m), we can
  - (1) Truncate $[U]_n$ in time $O(kM(n))$
  - (2) Approximate $[U + V]_n$ in time $O(kM(n))$
  - (3) Approximate $[U \odot V]_n$ in time $O(kM(n))$ where $\odot$ means componentwise multiplication

- This result is unlikely to hold in Turing machines
  - We need this kind of bounds in our complexity statements

Pointer Machines

- To preserve Axiom 3 in the more general setting, we propose to use Schöhage's elegant and flexible model of Pointer Machines
  - $M(n) = O(n)$ in this model (Schönhage)
  - Much nicer that $O(n \lg n \lg \ln n)$!

- LEMMA (cf. [SDY'05]) Assume the Pointer machine model
  - Give k-vectors U and V whose entries are floating point numbers of size (L, m), we can
  - (1) Truncate $[U]_n$ in time $O(kM(n))$
  - (2) Approximate $[U + V]_n$ in time $O(kM(n))$
  - (3) Approximate $[U \odot V]_n$ in time $O(kM(n))$ where $\odot$ means componentwise multiplication

- This result is unlikely to hold in Turing machines
  - We need this kind of bounds in our complexity statements

Pointer Machines

- To preserve Axiom 3 in the more general setting, we propose to use Schöhage's elegant and flexible model of Pointer Machines
  - ∗ $M(n) = O(n)$ in this model (Schönhage)
  - ∗ Much nicer that $O(n \lg n \lg \ln n)$!

- LEMMA (cf. [SDY'05]) Assume the Pointer machine model

  ∗ Give k-vectors U and V whose entries are floating point numbers of size (L, m), we can

  - ∗ (1) Truncate $[U]_n$ in time $O(kM(n))$
  - ∗ (2) Approximate $[U + V]_n$ in time $O(kM(n))$

  ∗ (3) Approximate $[U \odot V]_n$ in time $O(kM(n))$ where $\odot$ means componentwise multiplication

- This result is unlikely to hold in Turing machines
  - ∗ We need this kind of bounds in our complexity statements

Pointer Machines

- To preserve Axiom 3 in the more general setting, we propose to use Schöhage's elegant and flexible model of Pointer Machines
  - ∗ $M(n) = O(n)$ in this model (Schönhage)
  - ∗ Much nicer that $O(n \lg n \lg \ln n)$!

- LEMMA (cf. [SDY'05]) Assume the Pointer machine model

  ∗ Give k-vectors U and V whose entries are floating point numbers of size (L, m), we can

  - ∗ (1) Truncate $[U]_n$ in time $O(kM(n))$
  - ∗ (2) Approximate $[U + V]_n$ in time $O(kM(n))$
  - ∗ (3) Approximate $[U \odot V]_n$ in time $O(kM(n))$ where $\odot$ means componentwise multiplication

- This result is unlikely to hold in Turing machines
  - ∗ We need this kind of bounds in our complexity statements

III. FURTHER ISSUES IN MULTIPRECISION COMPUTATION (CASE STUDY)

Motivation: Guaranteed Accuracy Computation

- Nonrobustness is a widespread problem in geometric computation
  - ∗ Geometry is about discrete relations: Is a point on a line?
  - ∗ Any error on such decision is a "qualitative error", causing programs to crash
- In the last decade, the "Exact Geometric Computation" (EGC) approach has proven to be the most successful solution to nonrobustness
  - ∗ Current EGC libraries include LEDA, CGAL and Core Library
  - ∗ They all depend on guaranteed accuracy computation
- "Guaranteed accuracy computation" here means:
  - ∗ the requirement of a priori guarantees on error bounds
  - ∗ Cf. Interval analysis gives a posteriri guarantees on error

Motivation: Guaranteed Accuracy Computation

- Nonrobustness is a widespread problem in geometric computation
  - Geometry is about discrete relations: Is a point on a line?
  - Any error on such decision is a "qualitative error", causing programs to crash
- In the last decade, the "Exact Geometric Computation" (EGC) approach has proven to be the most successful solution to nonrobustness
  - Current EGC libraries include LEDA, CGAL and Core Library
  - They all depend on guaranteed accuracy computation
- "Guaranteed accuracy computation" here means:
  - the requirement of a priori guarantees on error bounds
  - Cf. Interval analysis gives a posteriri guarantees on error

Motivation: Guaranteed Accuracy Computation

- Nonrobustness is a widespread problem in geometric computation
  - ∗ Geometry is about discrete relations: Is a point on a line?
  - ∗ Any error on such decision is a "qualitative error", causing programs to crash
- In the last decade, the "Exact Geometric Computation" (EGC) approach has proven to be the most successful solution to nonrobustness
  - ∗ Current EGC libraries include LEDA, CGAL and Core Library
  - ∗ They all depend on guaranteed accuracy computation
- "Guaranteed accuracy computation" here means:
  - ∗ the requirement of *a priori* guarantees on error bounds
  - ∗ Cf. Interval analysis gives *a posteriri* guarantees on error

Motivation: Guaranteed Accuracy Computation

- Nonrobustness is a widespread problem in geometric computation
  - Geometry is about discrete relations: Is a point on a line?
  - Any error on such decision is a "qualitative error", causing programs to crash
- In the last decade, the "Exact Geometric Computation" (EGC) approach has proven to be the most successful solution to nonrobustness
  - Current EGC libraries include LEDA, CGAL and Core Library
  - They all depend on guaranteed accuracy computation
- "Guaranteed accuracy computation" here means:
  - the requirement of a priori guarantees on error bounds
  - Cf. Interval analysis gives a posteriri guarantees on error

Implications for Multiprecision Computation

- The guaranteed accuracy “mode” of computation imposes strong requirements
  - * (1) We cannot use asymptotic error analysis
  - * (2) Our algorithms must explicitly control the error in each operations
  - * (3) We need to decide Zero

- We illustrate with the problem of Newton iteration

Implications for Multiprecision Computation

- The guaranteed accuracy "mode" of computation imposes strong requirements
  - (1) We cannot use asymptotic error analysis
  - (2) Our algorithms must explicitly control the error in each operations
  - (3) We need to decide Zero
- We illustrate with the problem of Newton iteration

Implications for Multiprecision Computation

- The guaranteed accuracy "mode" of computation imposes strong requirements
  - (1) We cannot use asymptotic error analysis
  - (2) Our algorithms must explicitly control the error in each operations
  - (3) We need to decide Zero
- We illustrate with the problem of Newton iteration

Approximate Zeros

- Fix $f : \mathbb{R} \longrightarrow \mathbb{R}$, a smooth function.

- Given $z_0 \in \mathbb{R}$, construct the Newton iteration sequence
  - $z_{i+1} = N(z_i)$, where $N(z) = z - f(z)/f'(z)$
  - Assume $z_i \longrightarrow z^*$.

- DEFINITION (Smale) z_0 is an approximate zero
  - if it converges quadratically:
  - i.e., $|z_i - z^*| \leq 2^{1-2^i}|z_0 - z^*|$ for all $i \geq 0$

- POINT ESTIMATE THEOREM (Smale, et al)
  - If $\alpha(z_0) < 3 - 2\sqrt{2} \sim 0.17$, then z_0 is an approximate zero.

- $\gamma(z) := \max_{k \geq 2} \left| \frac{f^{(k)}}{k! f'} \right|^{1/(k-1)}$
  - $\beta(z) := \left| \frac{f(z)}{f'(z)} \right|$
  - $\alpha(z) := \beta(z)\gamma(z)$
  - So, lower bounds for $\alpha(z)$ are effectively computable

Approximate Zeros

- Fix $f : \mathbb{R} \to \mathbb{R}$, a smooth function.
- Given $z_0 \in \mathbb{R}$, construct the Newton iteration sequence
  - $z_{i+1} = N(z_i)$, where $N(z) = z - f(z)/f'(z)$
  - Assume $z_i \to z^*$.
- DEFINITION (Smale) z_0 is an approximate zero
  - if it converges quadratically:
  - i.e., $|z_i - z^*| \le 2^{1-2^i}|z_0 - z^*|$ for all $i \ge 0$
- POINT ESTIMATE THEOREM (Smale, et al)
  - If $\alpha(z_0) < 3 - 2\sqrt{2} \sim 0.17$, then z_0 is an approximate zero.
- $\gamma(z) := \max_{k\ge 2} \left|\frac{f^{(k)}}{k!f'}\right|^{1/(k-1)}$
  - $\beta(z) := \left|\frac{f(z)}{f'(z)}\right|$
  - $\alpha(z) := \beta(z)\gamma(z)$
  - So, lower bounds for $\alpha(z)$ are effectively computable

Approximate Zeros

- Fix $f : \mathbb{R} \to \mathbb{R}$, a smooth function.

- Given $z_0 \in \mathbb{R}$, construct the Newton iteration sequence
  - $z_{i+1} = N(z_i)$, where $N(z) = z - f(z)/f'(z)$
  - Assume $z_i \to z^*$.

- DEFINITION (Smale) z_0 is an approximate zero
  - if it converges quadratically:
  - i.e., $|z_i - z^*| \le 2^{1-2^i}|z_0 - z^*|$ for all $i \ge 0$

- POINT ESTIMATE THEOREM (Smale, et al)
  - If $\alpha(z_0) < 3 - 2\sqrt{2} \sim 0.17$, then z_0 is an approximate zero.

- $\gamma(z) := \max_{k\ge 2} \left|\frac{f^{(k)}}{k!f'}\right|^{1/(k-1)}$
  - $\beta(z) := \left|\frac{f(z)}{f'(z)}\right|$
  - $\alpha(z) := \beta(z)\gamma(z)$
  - So, lower bounds for $\alpha(z)$ are effectively computable

Approximate Zeros

- Fix $f : \mathbb{R} \to \mathbb{R}$, a smooth function.
- Given $z_0 \in \mathbb{R}$, construct the Newton iteration sequence
  - $z_{i+1} = N(z_i)$, where $N(z) = z - f(z)/f'(z)$
  - Assume $z_i \to z^*$.
- DEFINITION (Smale) z_0 is an approximate zero
  - if it converges quadratically:
  - i.e., $|z_i - z^*| \le 2^{1-2^i}|z_0 - z^*|$ for all $i \ge 0$
- POINT ESTIMATE THEOREM (Smale, et al)
  - If $\alpha(z_0) < 3 - 2\sqrt{2} \sim 0.17$, then z_0 is an approximate zero.
- $\gamma(z) := \max_{k\ge 2} \left|\frac{f^{(k)}}{k!f'}\right|^{1/(k-1)}$
  - $\beta(z) := \left|\frac{f(z)}{f'(z)}\right|$
  - $\alpha(z) := \beta(z)\gamma(z)$
  - So, lower bounds for $\alpha(z)$ are effectively computable

Approximate Zeros

- Fix $f : \mathbb{R} \rightarrow \mathbb{R}$, a smooth function.

- Given $z_0 \in \mathbb{R}$, construct the Newton iteration sequence
  - $z_{i+1} = N(z_i)$, where $N(z) = z - f(z)/f'(z)$
  - Assume $z_i \rightarrow z^*$.

- DEFINITION (Smale) z_0 is an approximate zero
  - if it converges quadratically:
  - i.e., $|z_i - z^*| \leq 2^{1-2^i}|z_0 - z^*|$ for all $i \geq 0$

- POINT ESTIMATE THEOREM (Smale, et al)
  - If $\alpha(z_0) < 3 - 2\sqrt{2} \sim 0.17$, then z_0 is an approximate zero.

- $\gamma(z) := \max_{k \geq 2} \left| \frac{f^{(k)}}{k! f'} \right|^{1/(k-1)}$
  - $\beta(z) := \left| \frac{f(z)}{f'(z)} \right|$
  - $\alpha(z) := \beta(z)\gamma(z)$
  - So, lower bounds for $\alpha(z)$ are effectively computable

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
  - Even if exact computation is possible, we may prefer approximation
- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i + C}$
  - Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)
- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
  - if for all $C \geq -\lg |\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically
- THEOREM [SDY'05]
  - If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero
- Cf. Malajovich (1994) – weak model

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
  - ∗ Even if exact computation is possible, we may prefer approximation

- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i+C}$
  - ∗ Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)

- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
  - ∗ if for all $C \geq -\lg|\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically

- THEOREM [SDY'05]
  - ∗ If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero

- Cf. Malajovich (1994) – weak model

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
    - $*$ Even if exact computation is possible, we may prefer approximation

- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i+C}$
    - $*$ Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)

- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
    - $*$ if for all $C \geq -\lg|\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically

- THEOREM [SDY'05]
    - $*$ If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero

- Cf. Malajovich (1994) – weak model

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
  - $*$ Even if exact computation is possible, we may prefer approximation

- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i+C}$
  - $*$ Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)

- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
  - $*$ if for all $C \geq -\lg|\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically

- THEOREM [SDY'05]
  - $*$ If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero

- Cf. Malajovich (1994) – weak model

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
  - Even if exact computation is possible, we may prefer approximation
- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i+C}$
  - Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)
- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
  - if for all $C \geq -\lg|\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically
- THEOREM [SDY'05]
  - If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero
- Cf. Malajovich (1994) – weak model

Robust Approximate Zeros

- Problem: $N(f)$ must be approximated
  - $*$ Even if exact computation is possible, we may prefer approximation
- Let $N_{i,C}(z) := \langle N(z) \rangle_{2^i+C}$
  - $*$ Starting from $\widetilde{z}_0$, let $\widetilde{z}_i = N_{i,C}(\widetilde{z}_{i-1})$ define the robust Newton sequence (relative to C)
- DEFINITION: $\widetilde{z}_0$ is a robust approximate zero
  - $*$ if for all $C \geq -\lg|\widetilde{z}_0 - z^*|$, the robust sequence relative to C converges quadratically
- THEOREM [SDY'05]
  - $*$ If $\alpha(\widetilde{z}_0) < 0.02$, the $\widetilde{z}_0$ is a robust approximate zero
- Cf. Malajovich (1994) – weak model

How to Implement Robust Newton Iteration

- TWO PROBLEMS
  - (C) How to estimate C?
  - (N) How to evaluate $N_{i,C}(z)$?

- (SOLUTION C) Let n_0 be the first n such that $\langle N(z_0)\rangle_n > 2^{-n+1}$
  - LEMMA: It suffices to choose C to be $n_0 + 2$. Moreover, this choice is no larger than $-\lg|z_0 - z^*| + 5$.

- (SOLUTION N) LEMMA:
  - To compute $N_{i,C}(z)$, it suffices to compute
  - (a) $f(z)$ to absolute $(K + 2^{i+1} + 4 + C)$-bits
  - (b) $f'(z)$ to absolute $(K' + 2^i + 3 + C)$-bits
  - (c) the division to relative $(K'' + 2^i + 1 + C)$-bits
  - where $K \geq -\lg|f'(z)|$, $K' \geq -\lg|f'(z_0)|\gamma(z)$, $K'' \geq 3 - \lg\gamma(z)$

How to Implement Robust Newton Iteration

- TWO PROBLEMS
  - (C) How to estimate C?
  - (N) How to evaluate $N_{i,C}(z)$?

- (SOLUTION C) Let n_0 be the first n such that $\langle N(z_0)\rangle_n > 2^{-n+1}$
  - LEMMA: It suffices to choose C to be $n_0 + 2$. Moreover, this choice is no larger than $-\lg|z_0 - z^*| + 5$.

- (SOLUTION N) LEMMA:
  - To compute $N_{i,C}(z)$, it suffices to compute
  - (a) $f(z)$ to absolute $(K + 2^{i+1} + 4 + C)$-bits
  - (b) $f'(z)$ to absolute $(K' + 2^i + 3 + C)$-bits
  - (c) the division to relative $(K'' + 2^i + 1 + C)$-bits
  - where $K \geq -\lg|f'(z)|$, $K' \geq -\lg|f'(z_0)|\gamma(z)$, $K'' \geq 3 - \lg\gamma(z)$

How to Implement Robust Newton Iteration

- TWO PROBLEMS
  - (C) How to estimate C?
  - (N) How to evaluate $N_{i,C}(z)$?

- (SOLUTION C) Let n_0 be the first n such that $\langle N(z_0)\rangle_n > 2^{-n+1}$
  - LEMMA: It suffices to choose C to be $n_0 + 2$. Moreover, this choice is no larger than $-\lg|z_0 - z^*| + 5$.

- (SOLUTION N) LEMMA:
  - To compute $N_{i,C}(z)$, it suffices to compute
  - (a) $f(z)$ to absolute $(K + 2^{i+1} + 4 + C)$-bits
  - (b) $f'(z)$ to absolute $(K' + 2^i + 3 + C)$-bits
  - (c) the division to relative $(K'' + 2^i + 1 + C)$-bits
  - where $K \geq -\lg|f'(z)|$, $K' \geq -\lg|f'(z_0)|\gamma(z)$, $K'' \geq 3 - \lg\gamma(z)$

How to Implement Robust Newton Iteration

- TWO PROBLEMS
  - (C) How to estimate C?
  - (N) How to evaluate $N_{i,C}(z)$?

- (SOLUTION C) Let n_0 be the first n such that $\langle N(z_0)\rangle_n > 2^{-n+1}$

  ∗ LEMMA: It suffices to choose C to be $n_0 + 2$. Moreover, this choice is no larger than $-\lg|z_0 - z^*| + 5$.

- (SOLUTION N) LEMMA:
  - To compute $N_{i,C}(z)$, it suffices to compute
  - (a) $f(z)$ to absolute $(K + 2^{i+1} + 4 + C)$-bits
  - (b) $f'(z)$ to absolute $(K' + 2^i + 3 + C)$-bits
  - (c) the division to relative $(K'' + 2^i + 1 + C)$-bits
  - where $K \geq -\lg|f'(z)|$, $K' \geq -\lg|f'(z_0)|\gamma(z)$, $K'' \geq 3 - \lg\gamma(z)$

UPSHOT: Uniform Complexity for Approximating Real Zeros

- Assume $f(X) \in \mathbb{R}[X]$ is square-free
  - Let $f(X) = \sum_{i=0}^{d} a_i X^i$, where $-L < \lg |a_i| < L$
  - Assume that we can compute a bigfloat approximation $[a_i]_n$ in time $B(n)$

- FOR SIMPLICITY, assume $L \geq \lg d$.

  ∗ PROBLEM: given a robust approximate zero z_0 with associated zero z^*, to approximate $\langle z^* \rangle_n$

- THEOREM [SDY'05]:
  - Assume $\Delta \geq -\lg |f(z_0)|$.
  - Then we can compute $\langle z^* \rangle_n$ in time

$$O[dM(n) + dM(\Delta) + d\lg(dL)M(dL)] +$$
$$O[d\lg(n+L)B(n+dL) + d\lg(dL+\Delta)B(dL+\Delta)]$$

- COROLLARY (Brent):
  - The local complexity of finding zeros of $f(X)$ is $O(M(n))$

UPSHOT: Uniform Complexity for Approximating Real Zeros

- Assume $f(X) \in \mathbb{R}[X]$ is square-free
  - $*$ Let $f(X) = \sum_{i=0}^{d} a_i X^i$, where $-L < \lg |a_i| < L$
  - $*$ Assume that we can compute a bigfloat approximation $[a_i]_n$ in time $B(n)$
- FOR SIMPLICITY, assume $L \geq \lg d$.
  - $*$ PROBLEM: given a robust approximate zero z_0 with associated zero z^*, to approximate $\langle z^* \rangle_n$
- THEOREM [SDY'05]:
  - $*$ Assume $\Delta \geq -\lg |f(z_0)|$.
  - $*$ Then we can compute $\langle z^* \rangle_n$ in time

$$O[dM(n) + dM(\Delta) + d\lg(dL)M(dL)] + \\ O[d\lg(n+L)B(n+dL) + d\lg(dL+\Delta)B(dL+\Delta)]$$

- COROLLARY (Brent):
  - $*$ The local complexity of finding zeros of $f(X)$ is $O(M(n))$

UPSHOT: Uniform Complexity for Approximating Real Zeros

- Assume $f(X) \in \mathbb{R}[X]$ is square-free
  - $*$ Let $f(X) = \sum_{i=0}^{d} a_i X^i$, where $-L < \lg |a_i| < L$
  - $*$ Assume that we can compute a bigfloat approximation $[a_i]_n$ in time $B(n)$

- FOR SIMPLICITY, assume $L \geq \lg d$.

  $*$ PROBLEM: given a robust approximate zero z_0 with associated zero z^*, to approximate $\langle z^* \rangle_n$

- THEOREM [SDY'05]:
  - $*$ Assume $\Delta \geq -\lg |f(z_0)|$.
  - $*$ Then we can compute $\langle z^* \rangle_n$ in time

$$O[dM(n) + dM(\Delta) + d\lg(dL)M(dL)] +$$
$$O[d\lg(n+L)B(n+dL) + d\lg(dL+\Delta)B(dL+\Delta)]$$

- COROLLARY (Brent):
  - $*$ The local complexity of finding zeros of $f(X)$ is $O(M(n))$

UPSHOT: Uniform Complexity for Approximating Real Zeros

- Assume $f(X) \in \mathbb{R}[X]$ is square-free
  - Let $f(X) = \sum_{i=0}^{d} a_i X^i$, where $-L < \lg |a_i| < L$
  - Assume that we can compute a bigfloat approximation $[a_i]_n$ in time $B(n)$

- FOR SIMPLICITY, assume $L \geq \lg d$.
  - PROBLEM: given a robust approximate zero z_0 with associated zero z^*, to approximate $\langle z^* \rangle_n$

- THEOREM [SDY'05]:
  - Assume $\Delta \geq -\lg |f(z_0)|$.
  - Then we can compute $\langle z^* \rangle_n$ in time

$$O[dM(n) + dM(\Delta) + d\lg(dL)M(dL)] +$$
$$O[d\lg(n+L)B(n+dL) + d\lg(dL+\Delta)B(dL+\Delta)]$$

- COROLLARY (Brent):
  - The local complexity of finding zeros of $f(X)$ is $O(M(n))$

UPSHOT: Uniform Complexity for Approximating Real Zeros

- Assume $f(X) \in \mathbb{R}[X]$ is square-free
  - $*$ Let $f(X) = \sum_{i=0}^{d} a_i X^i$, where $-L < \lg |a_i| < L$
  - $*$ Assume that we can compute a bigfloat approximation $[a_i]_n$ in time $B(n)$

- FOR SIMPLICITY, assume $L \geq \lg d$.

  $*$ PROBLEM: given a robust approximate zero z_0 with associated zero z^*, to approximate $\langle z^* \rangle_n$

- THEOREM [SDY'05]:
  - $*$ Assume $\Delta \geq -\lg |f(z_0)|$.
  - $*$ Then we can compute $\langle z^* \rangle_n$ in time

$$
\begin{aligned}
&O[dM(n) + dM(\Delta) + d\lg(dL)M(dL)] + \\
&\quad O[d\lg(n+L)B(n+dL) + d\lg(dL+\Delta)B(dL+\Delta)]
\end{aligned}
$$

- COROLLARY (Brent):
  - $*$ The local complexity of finding zeros of $f(X)$ is $O(M(n))$

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark
- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)
- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?
- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms
- Specific Open Problem:
    * What is the uniform complexity of polynomial evaluation?

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark
- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)
- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?
- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms
- Specific Open Problem:
  - ∗ What is the uniform complexity of polynomial evaluation?

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark

- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)

- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?

- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms

- Specific Open Problem:
  - ∗ What is the uniform complexity of polynomial evaluation?

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark
- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)
- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?
- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms
- Specific Open Problem:
  - ∗ What is the uniform complexity of polynomial evaluation?

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark
- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)
- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?
- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms
- Specific Open Problem:
  - What is the uniform complexity of polynomial evaluation?

CONCLUSION, OPEN PROBLEMS

- Brent's work on complexity of multiprecision computation 30 years ago remains a landmark
- Most of his results have withstood the test of time, and are suspected optimal (but would require major breakthrough in complexity theory to show)
- But the situation is completely open when we extend his fundamental framework to global and uniform complexity Could we find examples of tradeoffs among the different parameters?
- Guaranteed precision computation enforces a stronger standard in design and error analysis of multiprecision algorithms
- Specific Open Problem:
  - ∗ What is the uniform complexity of polynomial evaluation?

END OF TALK

Thanks for Listening!

- Papers cited in this talk:
  - [SDY'05]: "Robust Approximate Zeros", V.Sharma, Z.Du, C.Yap, ESA 2005
  - [DY'05]: "Uniform Complexity of Approximating Hypergeometric Functions with Absolute Error", Z.Du, C.Yap, ASCM 2005
  - [D'06]: "Algebraic and Transcendental Computation Made Easy: Theory and Implementation in Core Library", Ph.D.Thesis, New York University, May 2006
  - [Y'06]: "Theory of Real Computation according to EGC", To appear, special issue of LNCS based on Dagstuhl Seminar on 'Reliable Implementation of Real Number Algorithms: Theory and Practice'

"A rapacious monster lurks within every computer, and it dines exclusively on accurate digits."
– B.D. McCullough (2000)