# A Benchmark Test for Parallel Algorithmic Differentiation

Bradley M. Bell

Applied Physics Laboratory
Institute of Health Metrics and Evaluation
University of Washington
bradbell@uw.edu

Peter Hepperger

Technische Universität München,
Garching bei München, Germany

hepperger@ma.tum.de

The Northwest C++ Users Group Meeting
March 3, 2012

# Outline

Introduction to AD

# Outline

# Outline

# Outline

# Outline

# Outline

# Outline

# Outline

# Outline

# Outline



IHME

# Outline

# Outline



IHME

# Outline



IHME

# Outline

# Outline

# Outline

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$
2. $u^0 = \exp(x^0)$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$
2. $u^0 = \exp(x^0)$
3. $v^0 = \exp(y^0)$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$
2. $u^0 = \exp(x^0)$
3. $v^0 = \exp(y^0)$
4. $s^0 = u^0 + v^0$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$
2. $u^0 = \exp(x^0)$
3. $v^0 = \exp(y^0)$
4. $s^0 = u^0 + v^0$
5. $z^0 = \log(s^0)$

# Zero Order Forward

$$z(x, y) = \log[\exp(x) + \exp(y)]$$

1. Input: independent variable values $(x^0, y^0)$
2. $u^0 = \exp(x^0)$
3. $v^0 = \exp(y^0)$
4. $s^0 = u^0 + v^0$
5. $z^0 = \log(s^0)$
6. Output: dependent variable values $(u^0, v^0, s^0, z^0)$

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

IHME

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$

IHME

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$
2. $u^1 = \exp(x^0)x^1$

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$
2. $u^1 = \exp(x^0)x^1$
3. $v^1 = \exp(y^0)y^1$

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$
2. $u^1 = \exp(x^0)x^1$
3. $v^1 = \exp(y^0)y^1$
4. $s^1 = u^1 + v^1$

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$
2. $u^1 = \exp(x^0)x^1$
3. $v^1 = \exp(y^0)y^1$
4. $s^1 = u^1 + v^1$
5. $z^1 = s^1/s^0$

# First Order Forward

$$z^1 = \frac{\partial z}{\partial x}(x^0, y^0)x^1 + \frac{\partial z}{\partial y}(x^0, y^0)y^1$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), (x^1, y^1)$
2. $u^1 = \exp(x^0)x^1$
3. $v^1 = \exp(y^0)y^1$
4. $s^1 = u^1 + v^1$
5. $z^1 = s^1/s^0$
6. Output: $(u^1, v^1, s^1, z^1)$

# First Order Reverse

$$(x^1, y^1) = \left[ z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0) \right]$$

IHME

# First Order Reverse

$$(x^1, y^1) = \left[ z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0) \right]$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), z^1$

# First Order Reverse

$$(x^1, y^1) = \left[z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0)\right]$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), z^1$
2. $s^1 = z^1/s^0$

# First Order Reverse

$$(x^1, y^1) = \left[z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0)\right]$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), z^1$
2. $s^1 = z^1 / s^0$
3. $u^1 = s^1$, $v^1 = s^1$

# First Order Reverse

$$(x^1, y^1) = \left[z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0)\right]$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), z^1$
2. $s^1 = z^1 / s^0$
3. $u^1 = s^1$, $v^1 = s^1$
4. $y^1 = v^1 \exp(y^0)$
5. $x^1 = u^1 \exp(x^0)$

IHME

# First Order Reverse

$$(x^1, y^1) = \left[z^1 \frac{\partial z}{\partial x}(x^0, y^0), z^1 \frac{\partial z}{\partial y}(x^0, y^0)\right]$$

1. Input: $(x^0, y^0, u^0, v^0, s^0, z^0), z^1$
2. $s^1 = z^1 / s^0$
3. $u^1 = s^1$, $v^1 = s^1$
4. $y^1 = v^1 \exp(y^0)$
5. $x^1 = u^1 \exp(x^0)$
6. Output: $(x^1, y^1)$

# AD and Global Variables

- AD by Source Code Transformation

# AD and Global Variables

- AD by Source Code Transformation
- AD by Operator Overloading

# AD and Global Variables

- AD by Source Code Transformation
- AD by Operator Overloading
- Tapeless Forward AD

# AD and Global Variables

- AD by Source Code Transformation
- AD by Operator Overloading
- Tapeless Forward AD
- Recording Operations on a Tape

# Example Use of Multi-Newton Benchmark

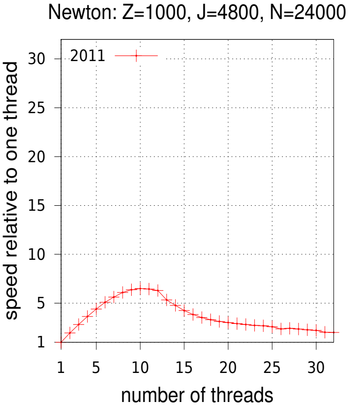
- CppAD uses a Tape with Operator Overloading

# Example Use of Multi-Newton Benchmark

- CppAD uses a Tape with Operator Overloading
- It comes with the the Multi-Newton benchmark test

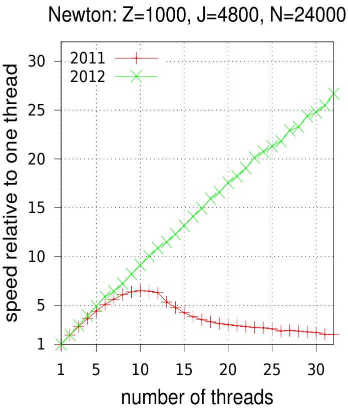
# Example Use of Multi-Newton Benchmark

- CppAD uses a Tape with Operator Overloading
- It comes with the the Multi-Newton benchmark test
- Version 2011 of CppAD on 32 Core Machine



# Example Use of Multi-Newton Benchmark

- CppAD uses a Tape with Operator Overloading
- It comes with the the Multi-Newton benchmark test
- Version 2011 & 2012.0 of CppAD on 32 Core Machine



# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.
5. If $f(x_k)f'(x_k) \geq 0$ and $x_k = a$, output $B = \emptyset$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.
5. If $f(x_k)f'(x_k) \geq 0$ and $x_k = a$, output $B = \emptyset$.
6. If $f(x_k)f'(x_k) \leq 0$ and $x_k = b$, output $B = \emptyset$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.
5. If $f(x_k)f'(x_k) \geq 0$ and $x_k = a$, output $B = \emptyset$.
6. If $f(x_k)f'(x_k) \leq 0$ and $x_k = b$, output $B = \emptyset$.
7. Set $y_k = x_k - f(x_k)/f'(x_k)$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.
5. If $f(x_k)f'(x_k) \geq 0$ and $x_k = a$, output $B = \emptyset$.
6. If $f(x_k)f'(x_k) \leq 0$ and $x_k = b$, output $B = \emptyset$.
7. Set $y_k = x_k - f(x_k)/f'(x_k)$.
8. Set $x_{k+1} = \min[b, \max(a, y_k)]$.

# Bounded Newton Method

Set $B(a, b, K, \delta, f)$ to one or no zeros in $[a, b]$ :

1. Input: bounds $[a, b]$, maximum iterations $K$, criterion $\delta$, function $f$.
2. Set $x_0 = (a + b)/2$, $k = 0$.
3. If $|f(x_k)| \leq \delta$, output $B = \{x_k\}$.
4. If $k == K$, output $B = \emptyset$.
5. If $f(x_k)f'(x_k) \geq 0$ and $x_k = a$, output $B = \emptyset$.
6. If $f(x_k)f'(x_k) \leq 0$ and $x_k = b$, output $B = \emptyset$.
7. Set $y_k = x_k - f(x_k)/f'(x_k)$.
8. Set $x_{k+1} = \min[b, \max(a, y_k)]$.
9. Set $k = k + 1$. Goto step 3

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.
2. Set sub-interval length $\gamma = (\beta - \alpha)/J$, start for first thread $s_1 = \alpha$, end for last thread $e_M = \beta$.

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.
2. Set sub-interval length $\gamma = (\beta - \alpha)/J$, start for first thread $s_1 = \alpha$, end for last thread $e_M = \beta$.
3. For $m = 1, \ldots M$, in sequential mode execute:

IHME

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.
2. Set sub-interval length $\gamma = (\beta - \alpha)/J$, start for first thread $s_1 = \alpha$, end for last thread $e_M = \beta$.
3. For $m = 1, \ldots M$, in sequential mode execute:
   3.1 Number of sub-intervals for thread $m$
   $$L_m = \begin{cases} \text{floor}(J/M) + 1 & \text{if } m \leq \mod(J, M) \\ \text{floor}(J/M) & \text{otherwise} \end{cases}$$

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.
2. Set sub-interval length $\gamma = (\beta - \alpha)/J$, start for first thread $s_1 = \alpha$, end for last thread $e_M = \beta$.
3. For $m = 1, \ldots M$, in sequential mode execute:
   3.1 Number of sub-intervals for thread $m$
   $$L_m = \begin{cases} \text{floor}(J/M) + 1 & \text{if } m \leq \mod(J, M) \\ \text{floor}(J/M) & \text{otherwise} \end{cases}$$
   3.2 Start for thread $m \geq 2$, $s_m = s_{m-1} + \gamma L_m$.

# Multi-Newton Setup

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

1. Input: bounds $[\alpha, \beta]$, maximum iterations $K$, criterion $\delta$, number of sub-intervals $J$, number of threads $M$, function $f$.
2. Set sub-interval length $\gamma = (\beta - \alpha)/J$, start for first thread $s_1 = \alpha$, end for last thread $e_M = \beta$.
3. For $m = 1, \ldots M$, in sequential mode execute:
   - 3.1 Number of sub-intervals for thread $m$
     $$L_m = \begin{cases} \text{floor}(J/M) + 1 & \text{if } m \leq \mod (J, M) \\ \text{floor}(J/M) & \text{otherwise} \end{cases}$$
   - 3.2 Start for thread $m \geq 2$, $s_m = s_{m-1} + \gamma L_m$.
   - 3.3 End for thread $m - 1 \geq 1$, $e_{m-1} = s_m$.

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:
   4.1 For $\ell = 1, \ldots, L_m$, set $a_{m,\ell} = s_m + \gamma(\ell - 1)$, and $b_{m,\ell} = a_{m,\ell} + \gamma$.

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:
   4.1 For $\ell = 1, \ldots, L_m$, set $a_{m,\ell} = s_m + \gamma(\ell - 1)$, and $b_{m,\ell} = a_{m,\ell} + \gamma$.
   4.2 Set $S_m = \cup_{\ell=1}^{L_m} B[a_{m,\ell}, b_{m,\ell}, K, \delta, f]$.

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:
   4.1 For $\ell = 1, \ldots, L_m$, set $a_{m,\ell} = s_m + \gamma(\ell - 1)$, and $b_{m,\ell} = a_{m,\ell} + \gamma$.
   4.2 Set $S_m = \cup_{\ell=1}^{L_m} B[a_{m,\ell}, b_{m,\ell}, K, \delta, f]$.
   4.3 If difference between two elements of $S_m$ is less than $\gamma/2$, remove one that has larger value for $|f(x)|$.

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:
   4.1 For $\ell = 1, \ldots, L_m$, set $a_{m,\ell} = s_m + \gamma(\ell - 1)$, and $b_{m,\ell} = a_{m,\ell} + \gamma$.
   4.2 Set $S_m = \cup_{\ell=1}^{L_m} B[a_{m,\ell}, b_{m,\ell}, K, \delta, f]$.
   4.3 If difference between two elements of $S_m$ is less than $\gamma/2$, remove one that has larger value for $|f(x)|$.
5. Set $S = \cup_{m=1}^{M} S_m$.

# Multi-Newton Continued

Set $S(\alpha, \beta, K, \delta, J, M, f)$ to multiple zeros in $[\alpha, \beta]$ :

4. For $m = 1, \ldots, M$, in parallel mode execute:
   4.1 For $\ell = 1, \ldots, L_m$, set $a_{m,\ell} = s_m + \gamma(\ell - 1)$, and $b_{m,\ell} = a_{m,\ell} + \gamma$.
   4.2 Set $S_m = \cup_{\ell=1}^{L_m} B[a_{m,\ell}, b_{m,\ell}, K, \delta, f]$.
   4.3 If difference between two elements of $S_m$ is less than $\gamma/2$, remove one that has larger value for $|f(x)|$.
5. Set $S = \cup_{m=1}^{M} S_m$.
6. If difference between two elements of $S$ is less than $\gamma/2$, remove one that has larger value for $|f(x)|$.
7. Output $S$.

# Threading System Interface

- $m = \mathrm{thread\_num}()$
  identifies current thread, $0 \leq m \leq M - 1$.

# Threading System Interface

- $m = \mathrm{thread\_num}()$
  identifies current thread, $0 \leq m \leq M - 1$.
- $b = \mathrm{in\_parallel}()$
  is true if in parallel execution mode, otherwise may be false.

# Threading System Interface

- $m = \mathrm{thread\_num}()$
  identifies current thread, $0 \leq m \leq M - 1$.
- $b = \mathrm{in\_parallel}()$
  is true if in parallel execution mode, otherwise may be false.
- $ok = \mathrm{team\_create}(num\_threads)$
  creates a team of *num_threads* threads.

# Threading System Interface

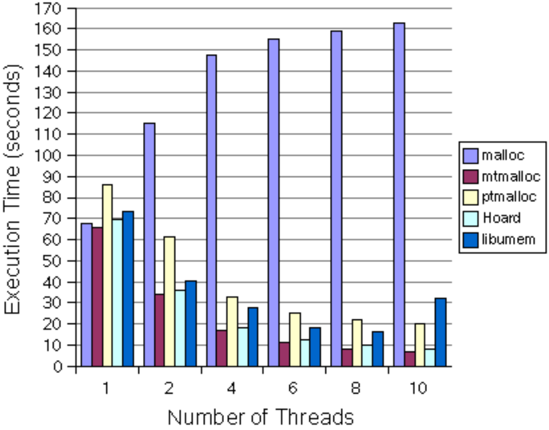
- $m = \mathrm{thread\_num}()$
  identifies current thread, $0 \leq m \leq M - 1$.
- $b = \mathrm{in\_parallel}()$
  is true if in parallel execution mode, otherwise may be false.
- $ok = \mathrm{team\_create}(num\_threads)$
  creates a team of *num_threads* threads.
- $ok = \mathrm{team\_work}(worker)$
  Each call runs *num_threads* versions of *worker* with corresponding $\mathrm{thread\_num}()$ between 0 and $M - 1$.

# Threading System Interface

- $m = \mathrm{thread\_num}()$
  identifies current thread, $0 \leq m \leq M - 1$.
- $b = \mathrm{in\_parallel}()$
  is true if in parallel execution mode, otherwise may be false.
- $ok = \mathrm{team\_create}(num\_threads)$
  creates a team of *num_threads* threads.
- $ok = \mathrm{team\_work}(worker)$
  Each call runs *num_threads* versions of *worker* with corresponding $\mathrm{thread\_num}()$ between 0 and $M - 1$.
- $ok = \mathrm{team\_destroy}()$
  terminates all but the master thread; i.e., $m = 0$.

# Motivation

The following plot is taken from
A comparison of memory allocators in multiprocessors:



# CppAD Allocation Versions

**Capacity** Let $c_1 = 128$ and for $i = 2, \ldots, I$, $c_i = 3 \cdot \mathrm{floor}[(c_{i-1} + 1)/2]$.

# CppAD Allocation Versions

**Capacity** Let $c_1 = 128$ and for $i = 2, \ldots, I$, $c_i = 3 \cdot \mathrm{floor}[(c_{i-1} + 1)/2]$.

**2011** Uses system memory allocation with capacity modification above.

# CppAD Allocation Versions

**Capacity** Let $c_1 = 128$ and for $i = 2, \ldots, I$,
$c_i = 3 \cdot \mathrm{floor}[(c_{i-1} + 1)/2]$.

**2011** Uses system memory allocation with capacity modification above.

**2012.0** For each thread $m$ and each capacity $i$, a singly linked list of available memory is retained with root $A_{m*I+i}$. In addition, counter vectors for the number of bytes inuse $u_m$ and available $a_m$ for corresponding thread.

# CppAD Allocation Versions

**Capacity** Let $c_1 = 128$ and for $i = 2, \ldots, I$,
$c_i = 3 \cdot \mathrm{floor}[(c_{i-1} + 1)/2]$.

**2011** Uses system memory allocation with capacity modification above.

**2012.0** For each thread $m$ and each capacity $i$, a singly linked list of available memory is retained with root $A_{m*I+i}$. In addition, counter vectors for the number of bytes inuse $u_m$ and available $a_m$ for corresponding thread.

**2012.1** For each thread $m$, a separate structure $S_m$ is allocated for the inuse counter, available counter, and vector of available roots for that thread.

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N}\sum_{n=1}^{N} \sin(x)$$

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N} \sum_{n=1}^{N} \sin(x)$$

$N$ Work load per function evaluation.

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N} \sum_{n=1}^{N} \sin(x)$$

$N$ Work load per function evaluation.

$Z$ Number of zeros $Z = 1000$.

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N}\sum_{n=1}^{N}\sin(x)$$

$N$ Work load per function evaluation.

$Z$ Number of zeros $Z = 1000$.

$\alpha$ Lower search limit $\alpha = 0$.

IHME

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N}\sum_{n=1}^{N}\sin(x)$$

$N$ Work load per function evaluation.

$Z$ Number of zeros $Z = 1000$.

$\alpha$ Lower search limit $\alpha = 0$.

$\beta$ Upper search limit $\beta = (Z-1)\pi$.

IHME

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N}\sum_{n=1}^{N}\sin(x)$$

$N$ Work load per function evaluation.

$Z$ Number of zeros $Z = 1000$.

$\alpha$ Lower search limit $\alpha = 0$.

$\beta$ Upper search limit $\beta = (Z - 1)\pi$.

$\delta$ Convergence criterion: $\delta = \beta \cdot 10^2 \cdot \varepsilon$.

IHME

# Benchmark Parameters Choices

$f(x)$ Function we are finding zeros for:

$$f(x) = \frac{1}{N} \sum_{n=1}^{N} \sin(x)$$

$N$ Work load per function evaluation.

$Z$ Number of zeros $Z = 1000$.

$\alpha$ Lower search limit $\alpha = 0$.

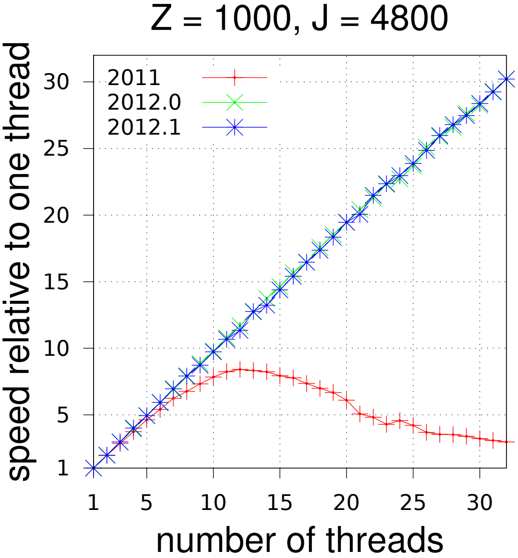
$\beta$ Upper search limit $\beta = (Z - 1)\pi$.

$\delta$ Convergence criterion: $\delta = \beta \cdot 10^2 \cdot \varepsilon$.

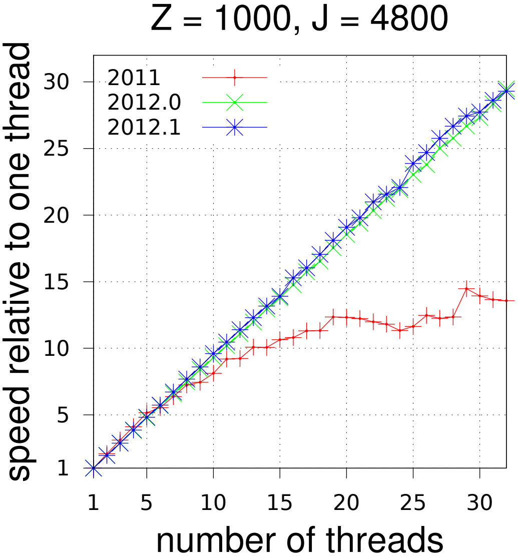
$J$ Number of bounded Newton sub-intervals $J = 4800$.

IHME

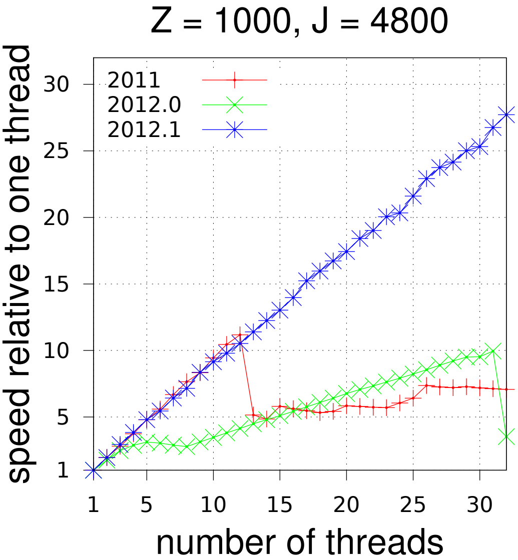
# 32 cores, $N = 10000$



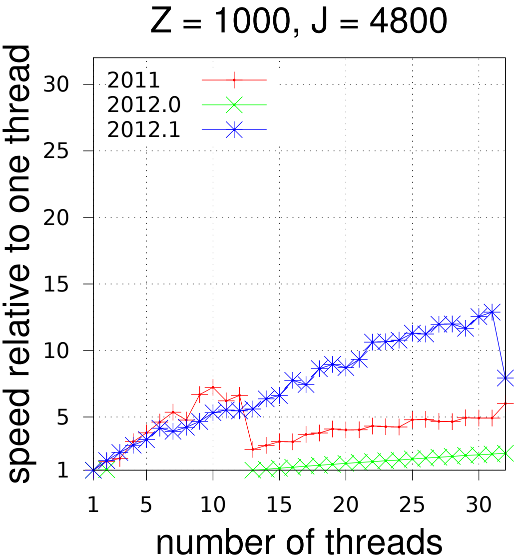
# 32 cores, N = 1000



## 32 cores, N = 100



# 32 cores, N = 10



Z = 1000, J = 4800
2011
2012.0
2012.1
speed relative to one thread
number of threads
1
5
10
15
20
25
30