FPAvisual: A Tool for Visualizing the Effects of Floating-Point Finite-Precision Arithmetic

Yi Gu, Nilufer Onder, CK Shene, Chaoli Wang

Department of Computer Science

Michigan Technological University

Presented by: Nilufer Onder

Wednesday, June 18, 2014, 2:15pm-3:45pm

Indiana Convention Center, Room 127

Michigan Tech

Create the Future

Outline

- Motivation
- Background
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Create the Future

Motivation

- Help students realize how program correctness may be impacted when floating-point finite-precision arithmetic (FPA) is used

- Help instructors teach
  - Reasons for the inaccuracies caused by FPA
  - Their impact and significance in programs
  - Techniques to improve the accuracy

Outline

- Motivation
- Background
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Rounding

- Computers represent floating-point numbers using a finite number of bits
- When a number contains more digits than allowed by the hardware, it is rounded
- The rounded number is an approximation of the original number

Examples

- Example 1:
  - $123 + 2.46 = 125.46 = 125$

- Example 2:
  - $123 + 0.46 = 123.46 = 123$

Failure of the Associative Law

Calculate $0.121 \times 0.345 \times 4.32$

Order 1:

$(0.121 \times 0.345) \times 4.32$

$$= (0.041745 \times 4.21)$$
$$= 0.0417 \times 4.21$$
$$= 0.175557$$
$$= 0.176$$

Order 2:

$0.121 \times (0.345 \times 4.32)$

$$= 0.121 \times 1.45245$$
$$= 0.121 \times 1.45$$
$$= 0.17545$$
$$= 0.175$$

Cancellation

- Calculate $b^2 - 4ac$, where a = 1, b = 1.23, and c = 0.374

- $b^2 = 1.23^2 = 1.5129 = 1.51$
- $4ac = 4 \times 1 \times 0.374 = 1.496 = 1.50$
- $b^2 - 4ac = 1.51 - 1.50$ = 0.01

- Actually, $b^2 - 4ac = 1.5129 - 1.496$ = 0.0169

Outline

- Motivation
- Background
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Roots

- $ax^2 + bx + c = 0$
- $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Two problems
  - $b^2 \gg 4ac$
    - $\sqrt{b^2 - 4ac} = b$
    - One of the roots is 0
  - $b^2 \approx 4ac$
    - $\sqrt{b^2 - 4ac} = 0$
    - Two roots are equal

Avoiding Cancellation

- Remove subtraction in computing the first root r_1
  - If $b > 0$, use $-b - \sqrt{b^2 - 4ac}$
  - If $b \leq 0$, use $-b + \sqrt{b^2 - 4ac}$

- Since the product of roots is $\frac{c}{a}$, use $\frac{c}{ar_1}$ to compute the other root

FPAvisual

File Examples Help

Roots | Pentagon | Associative Law | Sine

$$ax^2 + bx + c = 0$$

Input

a 1.000000 b 10000.000000 c 1.000000 Calculate

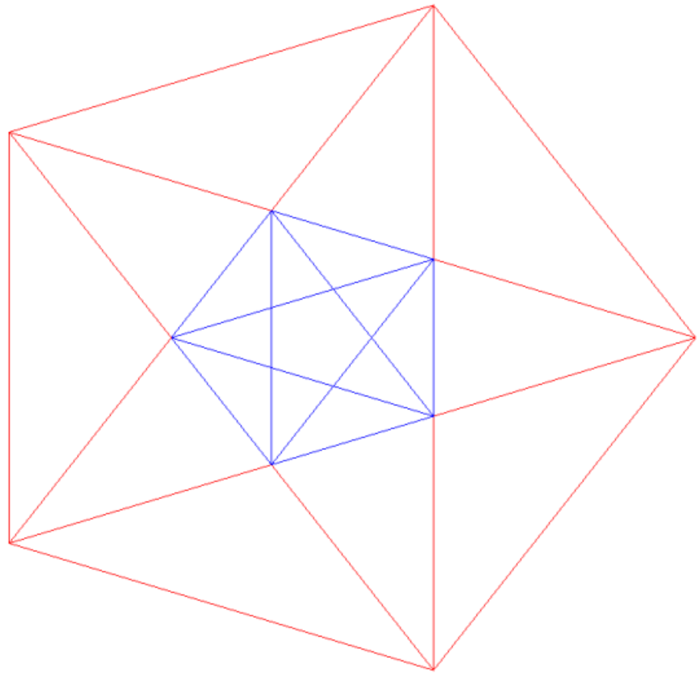
Output

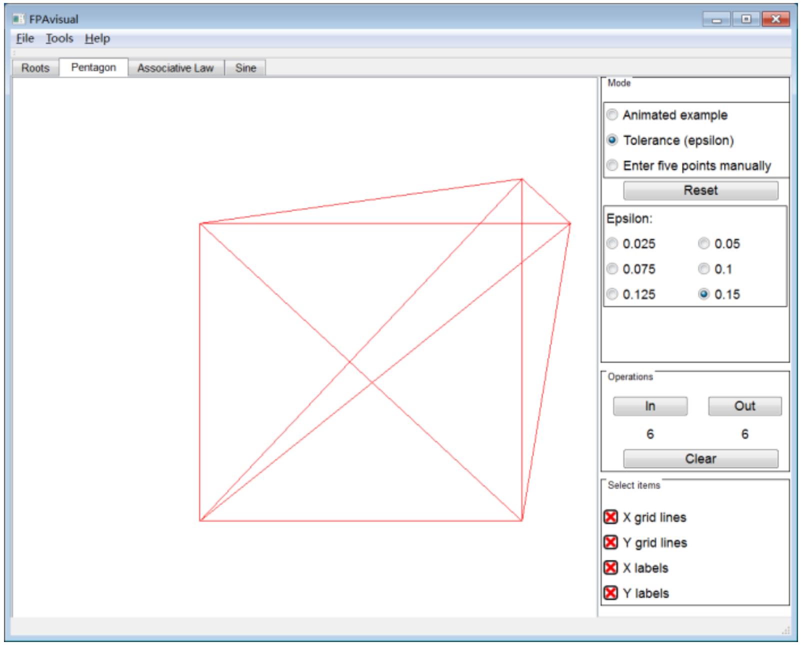
Solution	Naive Single P.	Cancellation Single P.	Naive Double P.	Naive High P.
b^2	1.000000e+008	1.000000e+008	1.000000000000e+008	1.000000000000e+008
$4ac$	4.000000e+000	4.000000e+000	4.000000000000e+000	4.000000000000e+000
$\sqrt{b^2 - 4ac}$	1.000000e+004	1.000000e+004	9.999999800000e+003	9.999999800000e+003
Large Root	0.000000e+000	-1.000000e-004	-1.000000011118e-004	-1.000000010000e-004
$ax^2 + bx + c =$	1.000000e+000	3.526213e-008	-1.117663073202e-009	6.257499694697e-084
Small Root	-1.000000e+004	-1.000000e+004	-9.999999900000e+003	-9.999999900000e+003
$ax^2 + bx + c =$	1.000000e+000	1.000000e+000	0.000000000000e+000	-1.029308202740e-075

Michigan Tech Create the Future

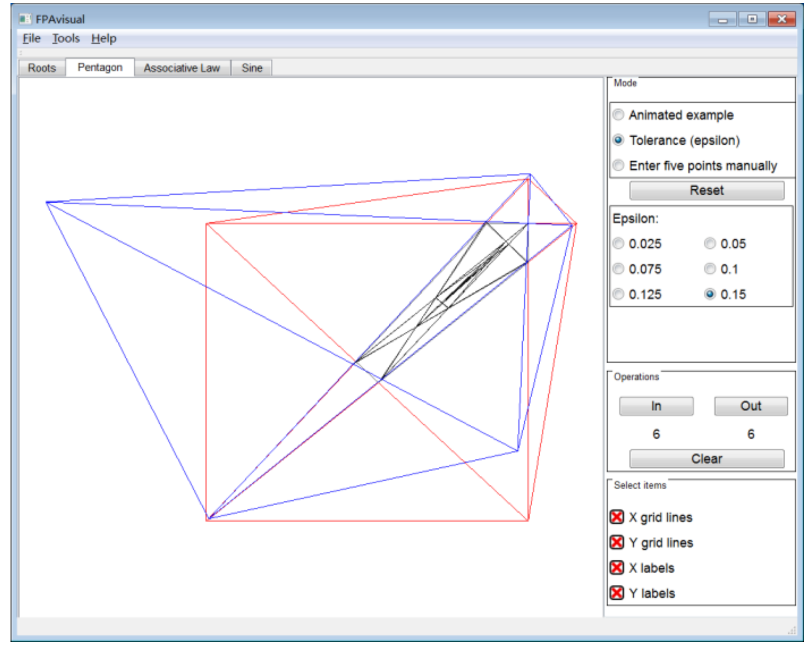
Pentagon

- Inaccuracies when calculating the intersection of two nearly parallel lines
- In operation:
  Red pentagon → Blue pentagon
- Out operation:
  Blue pentagon → Red pentagon



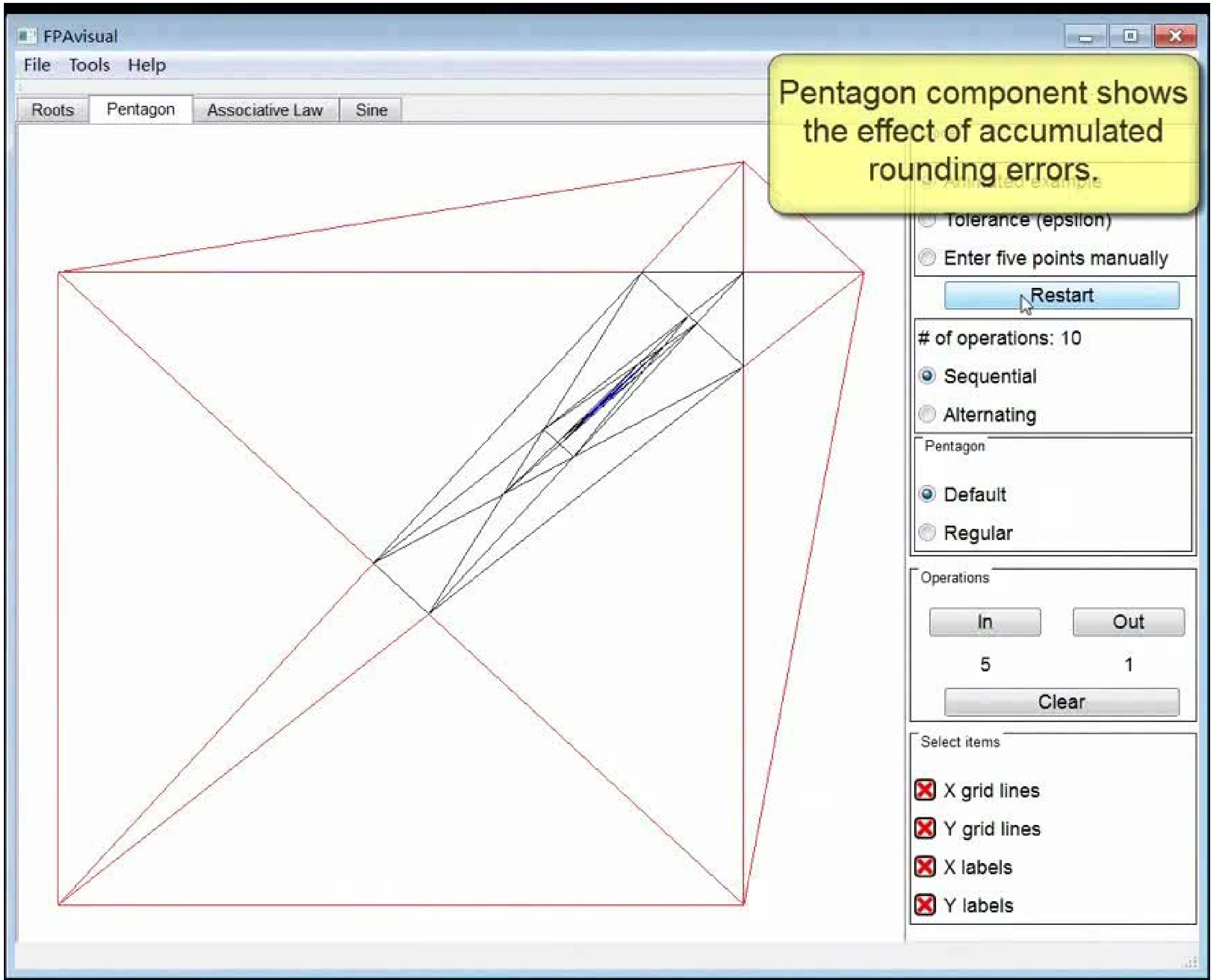


Initial
pentagon



Final
pentagon

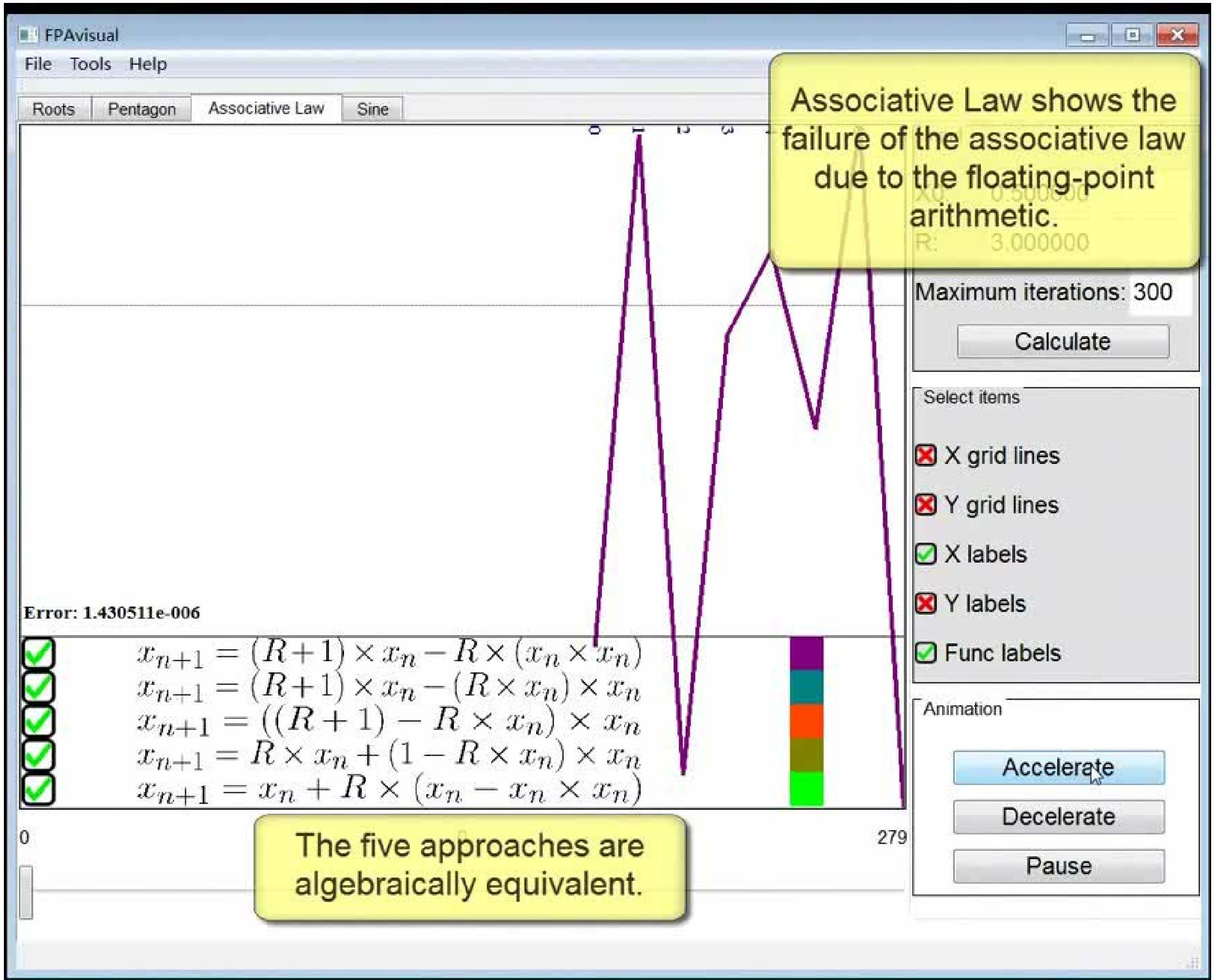
Create the Future



Michigan Tech
Create the Future

Associative Law

- Given an iterative formula
  - $X_{n+1} = (R + 1) \times X_n - R \times X_n \times X_n$
- Computing it using five orderings will generate different results
  - $X_{n+1} = (R + 1) \times X_n - R \times (X_n \times X_n)$
  - $X_{n+1} = (R + 1) \times X_n - (R \times X_n) \times X_n$
  - $X_{n+1} = ((R + 1) - R \times X_n) \times X_n$
  - $X_{n+1} = R \times X_n + (1 - R \times X_n) \times X_n$
  - $X_{n+1} = X_n + R \times (X_n - X_n \times X_n)$



Michigan Tech
Create the Future

Sine Function using Taylor Series

- $\sin(x) =$

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

- Problems:
  - If x is very large or small, x^{2n+1} may overflow or underflow when n is large
  - Overflow may occur when calculating $(2n+1)!$
  - Cancellation may occur in the summation of terms with alternating sign values

Dealing with Large *X*

- Reduce the user input x (in degrees) to [0, 90)
- Since $\sin(x) = -\sin(-x)$, if $x < 0$, we use $-\sin(|x|)$
- Since $\sin(x)$ has a period of 360, we can reduce x to [0, 360) by letting $x = x\%360$
- Since $\sin(x + 180) = -\sin(x)$, if $x \geq 180$, we may reduce x to [0, 180) by letting $x = x - 180$ and changing the sign of the computed result
- Since $\sin(180 - x) = \sin(x)$, if x is in [90, 180), we may reduce x to [0, 90) by letting $x = 180 - x$

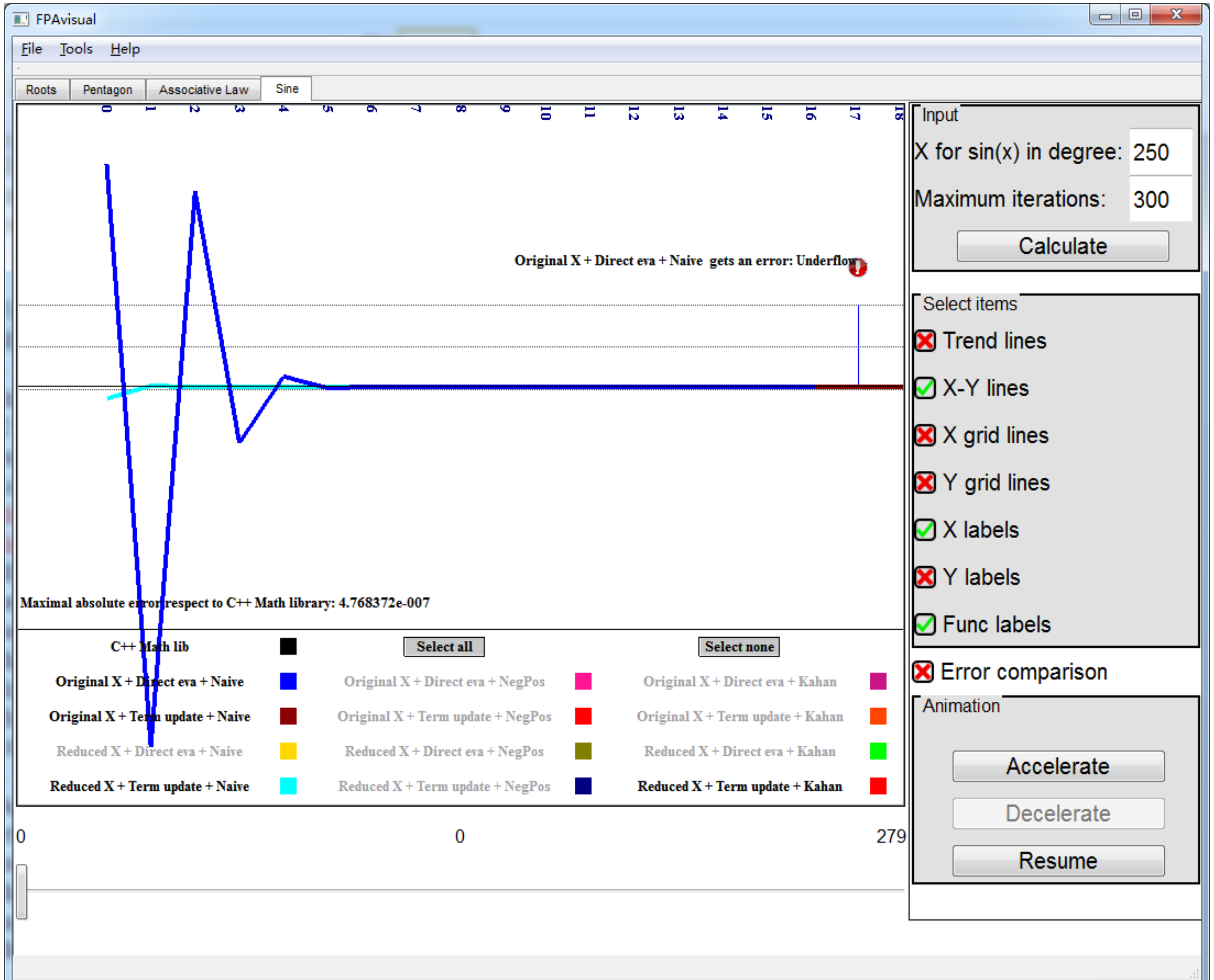
Computing the Factorial

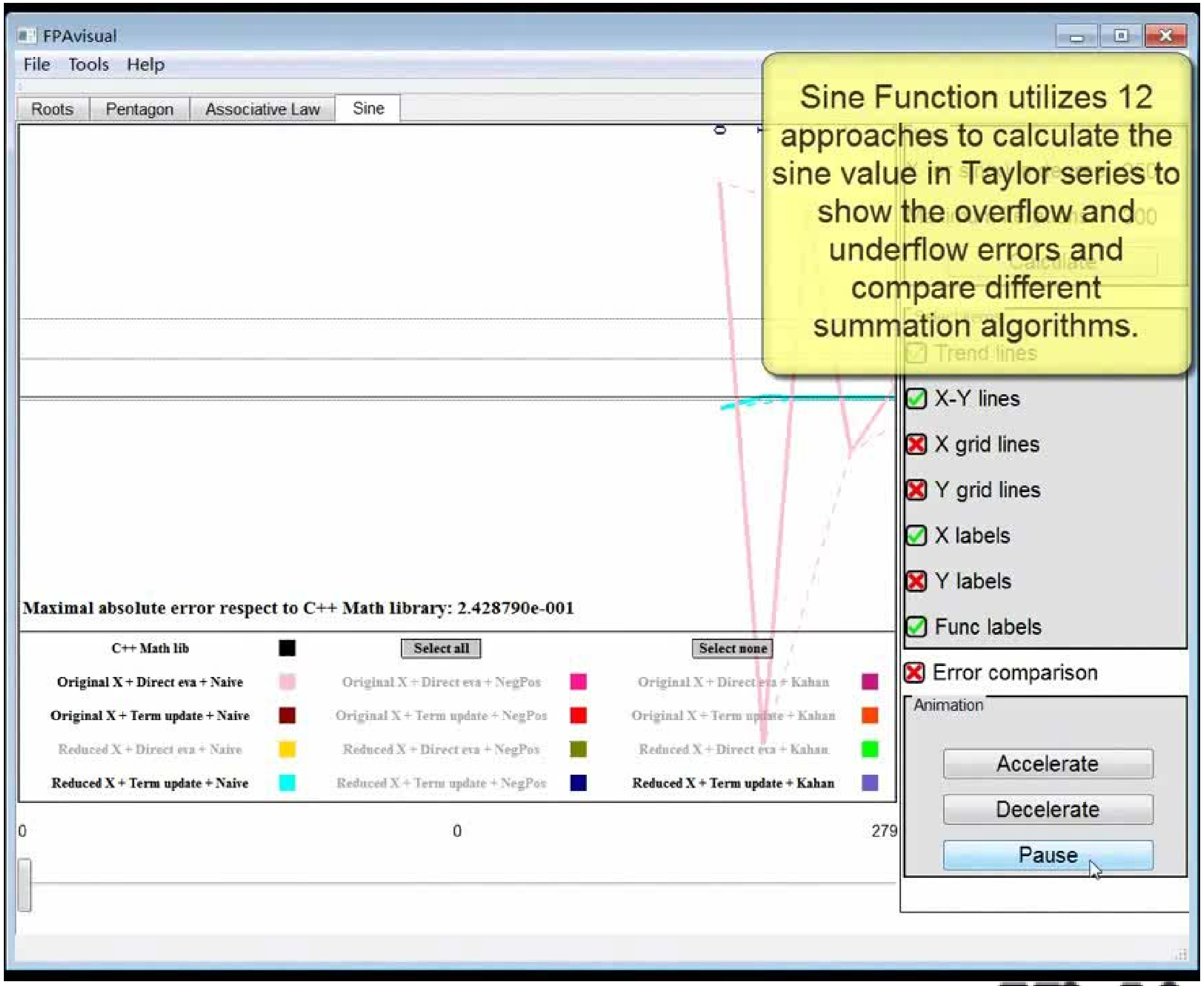
- Use a floating-point number to store the value
- Update the term from the previous one

$$\frac{x^{2n+1}}{(2n+1)!} = \frac{x^{2n-1}}{(2n-1)} \times \frac{x^2}{(2n)(2n+1)}$$

Avoiding Cancellation in Summation

- Use the positive-negative algorithm to reduce the probability of subtracting two similar values:
  - Add all positive terms
  - Add all negative terms
  - Add the above two values
- Use Kahan's summation algorithm


FPAvisual
File Tools Help
Roots Pentagon Associative Law Sine
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Original X + Direct eva + Naive gets an error: Underflow
Maximal absolute error respect to C++ Math library: 4.768372e-007
C++ Math lib
Select all
Select none
Original X + Direct eva + Naive
Original X + Direct eva + NegPos
Original X + Direct eva + Kahan
Original X + Term update + Naive
Original X + Term update + NegPos
Original X + Term update + Kahan
Reduced X + Direct eva + Naive
Reduced X + Direct eva + NegPos
Reduced X + Direct eva + Kahan
Reduced X + Term update + Naive
Reduced X + Term update + NegPos
Reduced X + Term update + Kahan
0
0
279
Input
X for sin(x) in degree: 250
Maximum iterations: 300
Calculate
Select items
Trend lines
X-Y lines
X grid lines
Y grid lines
X labels
Y labels
Func labels
Error comparison
Animation
Accelerate
Decelerate
Resume


FPAvisual
File Tools Help
Roots
Pentagon
Associative Law
Sine
Sine Function utilizes 12 approaches to calculate the sine value in Taylor series to show the overflow and underflow errors and compare different summation algorithms.
Trend lines
X-Y lines
X grid lines
Y grid lines
X labels
Y labels
Func labels
Error comparison
Animation
Accelerate
Decelerate
Pause
Maximal absolute error respect to C++ Math library: 2.428790e-001
C++ Math lib
Select all
Select none
Original X + Direct eva + Naive
Original X + Direct eva + NegPos
Original X + Direct eva + Kahan
Original X + Term update + Naive
Original X + Term update + NegPos
Original X + Term update + Kahan
Reduced X + Direct eva + Naive
Reduced X + Direct eva + NegPos
Reduced X + Direct eva + Kahan
Reduced X + Term update + Naive
Reduced X + Term update + NegPos
Reduced X + Term update + Kahan
0
0
279

Outline

- Motivation
- Basic knowledge
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Usefulness

	Fall		Spring	
	μ	σ	μ	σ
The "Roots" component helped me understand the effects of floating-point errors.	3.9	0.7	4.0	0.9
The "Roots" component helped me understand how to compute the roots of a quadratic equation more accurately.	3.6	0.8	3.8	0.8
The "Pentagon" component helped me understand the effects of floating-point errors.	3.9	1.1	3.8	1.0
The "Pentagon" component helped me understand that calculating the intersection points of two almost parallel lines can lead to noticeable errors.	3.9	0.9	4.1	0.9
The "Associative Law" component helped me understand how executing floating-point operations in different orders affects the computed results.	4.2	0.7	4.0	1.0
The "Associative Law" component helped me understand that there are no general techniques to detect and correct the errors coming from the failure of the associative law.	4.2	0.7	3.9	1.1
The "Sine" component helped me understand the effects of floating-point errors.	3.5	1.0	3.5	1.3
The "Sine" component helped me compare the effects of reducing X to the [0,90] range, using the term update method, and using Kahan's summation algorithm.	3.3	1.1	3.6	1.0
FPAvisual was a useful complement to the material presented in class.	3.8	0.6	3.9	1.0

Usability

	Fall		Spring	
	μ	σ	μ	σ
The example inputs provided in the "Roots" component helped me to see what kind of input values cause noticeable floating-point errors.	3.8	0.8	4.1	0.8
In the "Roots" component, seeing the results of computations in different colors helped me notice the differences between the approaches.	4.0	0.9	4.1	1.0
The animated examples in the "Pentagon" component helped me compare the results of in-out operations for differently shaped pentagons.	3.9	0.9	3.9	1.0
Being able to select pentagons for comparison was useful for me to see the accumulated floating-point errors.	3.8	0.9	3.8	1.0
The animations in the "Associative Law" component were useful for me to gain an impression of the effect of floating-point errors.	4.1	0.7	3.9	1.1
The color encoding in the "Associative Law" component was useful for me to track the trend of the five computations.	4.3	0.7	4.0	0.8
The animations in the "Sine" component helped me track the trend of different approaches.	3.7	0.9	3.8	1.0
Overall, I'm satisfied with the color encoding.	4.2	0.6	4.2	0.9
The freedom of manual input was useful to select inputs that cause noticeable floating-point errors.	4.0	0.7	4.3	0.8

Outline

- Motivation
- Basic knowledge
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Michigan Tech
Create the Future

Conclusion

- Instructors are able to present the effects of different types of floating-point errors:
  one-time, accumulated, unexpected errors
- FPAvisual software complements the lectures by helping students see various methods to reduce errors:
  domain specific and domain independent techniques
- The evaluation results suggest that FPAvisual is a useful complement to class teaching:
  flexible, allows exploration, can fit into most courses

Outline

- Motivation
- Basic knowledge
  - Rounding
  - Cancellation
- FPAvisual software
  - Roots
  - Pentagon
  - Associative law
  - Sine function
- Evaluation
- Conclusion
- Future work

Future Work

- Visualize what the errors are and where they occur
- Make the Sine Function component more understandable by distinguishing between the 12 approaches
- Add detailed explanation text for the components
- Develop a MacOS version
- Expand the type and number of the examples in the program
- Conduct a summative assessment of the software

Michigan Tech
Create the Future

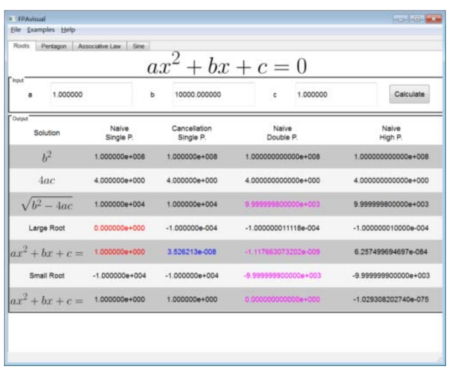
Thank you!

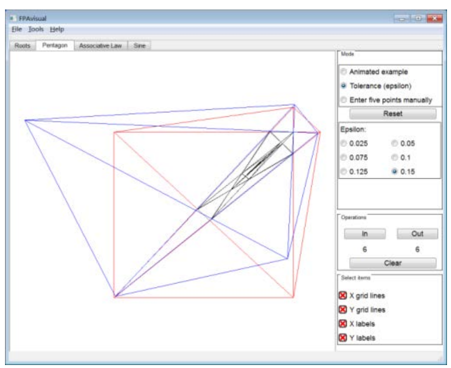
FPAvisual: A Tool for Visualizing the Effects of Floating-Point Finite-Precision Arithmetic

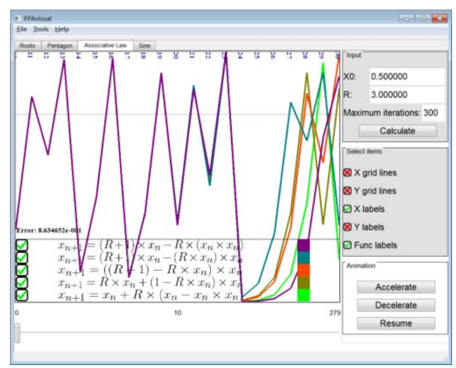
Yi Gu, Nilufer Onder, CK Shene, Chaoli Wang

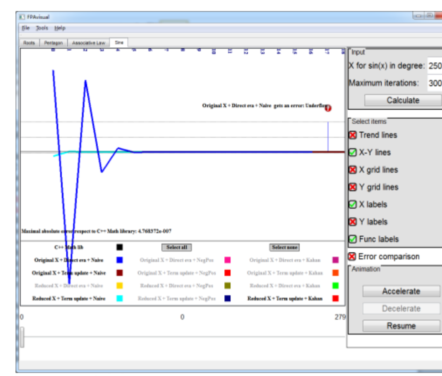
Department of Computer Science

Michigan Technological University

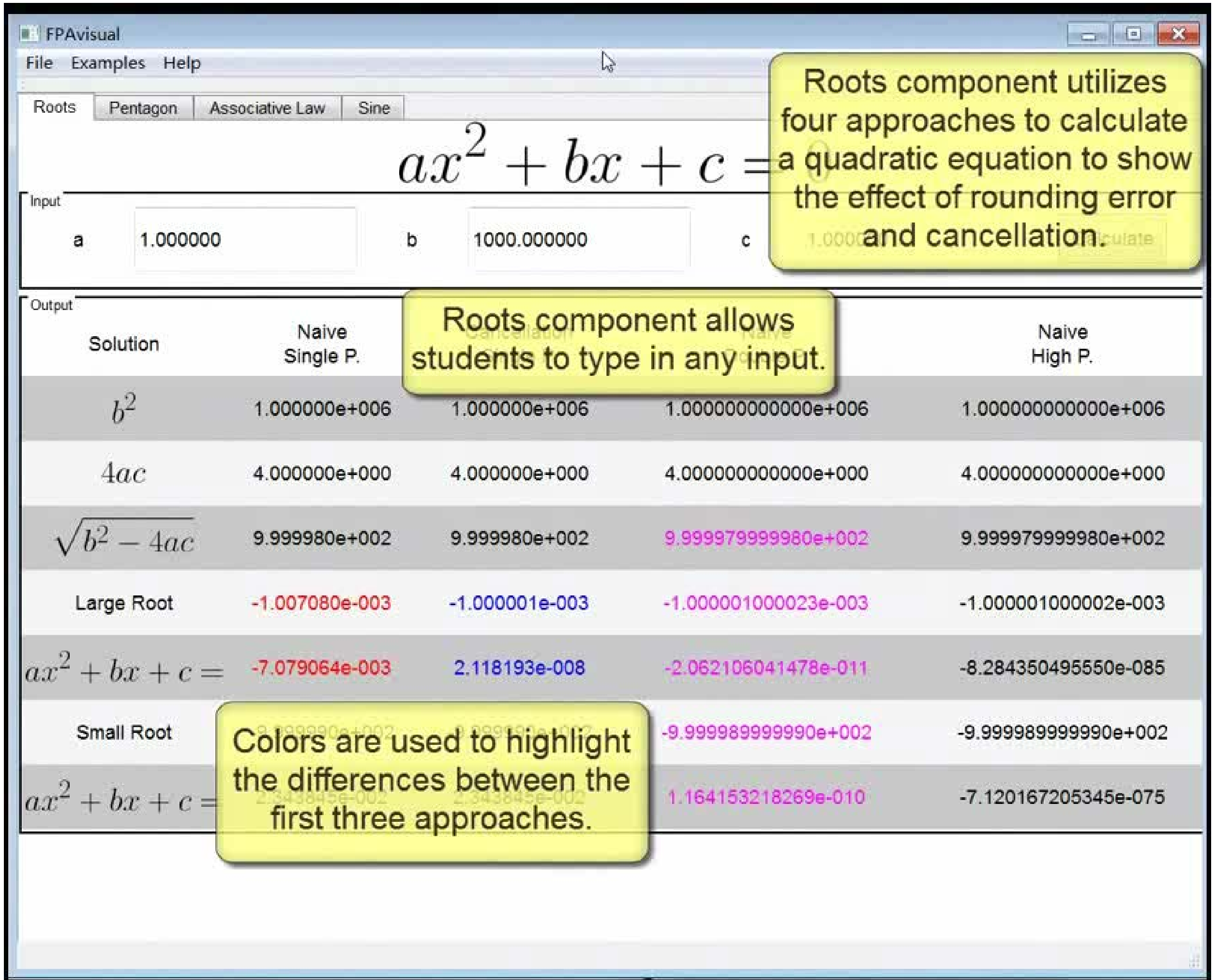


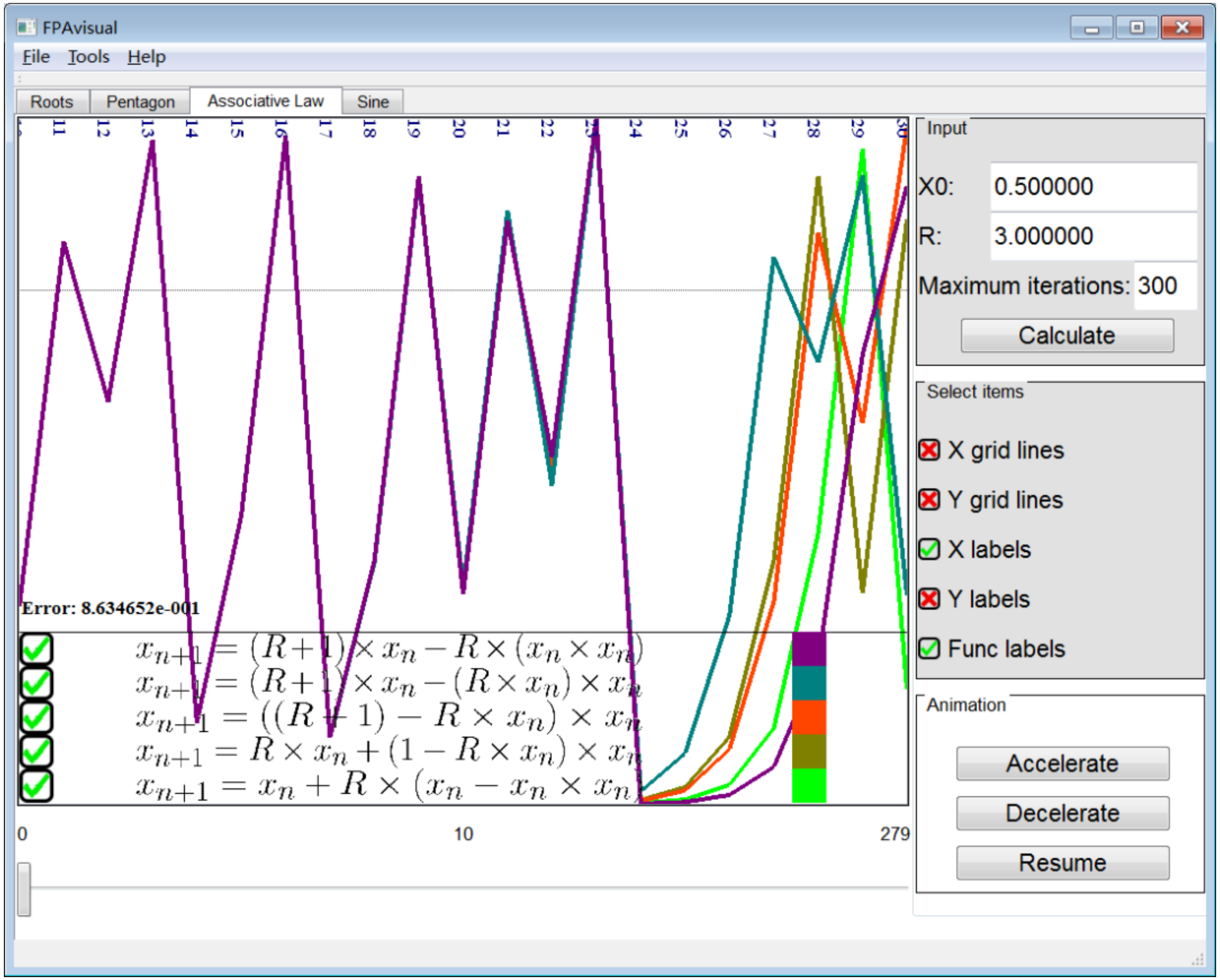






Create the Future


FPAvisual
File Examples Help
Roots Pentagon Associative Law Sine
ax^2 + bx + c =
Input
a 1.000000
b 1000.000000
c
Roots component utilizes four approaches to calculate a quadratic equation to show the effect of rounding error and cancellation.
Output
Solution
Naive Single P.
Naive High P.
Roots component allows students to type in any input.
b^2 1.000000e+006 1.000000e+006 1.000000000000e+006 1.000000000000e+006
4ac 4.000000e+000 4.000000e+000 4.000000000000e+000 4.000000000000e+000
√(b^2 − 4ac) 9.999980e+002 9.999980e+002 9.999979999980e+002 9.999979999980e+002
Large Root -1.007080e-003 -1.000001e-003 -1.000001000023e-003 -1.000001000002e-003
ax^2 + bx + c = -7.079064e-003 2.118193e-008 -2.062106041478e-011 -8.284350495550e-085
Small Root -9.999989999990e+002 -9.999989999990e+002
ax^2 + bx + c = 1.164153218269e-010 -7.120167205345e-075
Colors are used to highlight the differences between the first three approaches.


FPAvisual
File Tools Help
Roots
Pentagon
Associative Law
Sine
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Error: 8.634652e-001
$x_{n+1} = (R+1) \times x_n - R \times (x_n \times x_n)$
$x_{n+1} = (R+1) \times x_n - (R \times x_n) \times x_n$
$x_{n+1} = ((R+1) - R \times x_n) \times x_n$
$x_{n+1} = R \times x_n + (1 - R \times x_n) \times x_n$
$x_{n+1} = x_n + R \times (x_n - x_n \times x_n)$
0
10
279
Input
X0: 0.500000
R: 3.000000
Maximum iterations: 300
Calculate
Select items
X grid lines
Y grid lines
X labels
Y labels
Func labels
Animation
Accelerate
Decelerate
Resume