FPEAnalysis System: User's Manual

An Independent Reading and Research Project (Fall 2015)
under Professor David Bindel (bindel@cs.cornell.edu)
Cornell University

Humam A Alwassel (haa54@cornell.edu)

Revision 1, updated December 21, 2015

This document is the user's manual for FPEAnanlysis system. It explains how to install and run FPEAnalysis, details the syntax for input files, and shows how to interpret the analysis report. This document is non-technical in nature and is aimed toward readers with an introductory computer science background. Please refer to the "Developer's Manual" if you want to understand how FPEAnalysis works or wish to develop the system further.

1 What is FPEAnalysis System?

Floating point error analysis of simple expressions is a standard topic in introductory scientific computing courses. Unfortunately, such analyses can be tedious and error prone. **FPEAnalysis** (**F**loating **P**oint **E**rror **Analysis**) is a system that automates this analysis process using the $(1 + \delta)$ model. FPEAnalysis takes in a program and values ranges for each of its variables. The system identifies and reports any critical subexpressions that result in a high relative error in the output. Users can use FPEAnalysis to identify numerical instabilities in their program, and make the necessary rewrites to avoid such instabilities.

2 Installation and Usage

The source code is written in OCaml. You need OCaml version 4.02.1 or higher installed on your machine. Code is available on GitHub on https://github.com/HumamAlwassel/FPEAnalysis. To install FPEAnalysis, make sure you have git installed on your machine, and then run the following commands in your terminal:

```
$ git clone git@github.com:HumamAlwassel/FPEAnalysis.git
$ cd FPEAnalysis/src/
$ make
```

To run the system on an input file, use

```
$ ./FPEAnalysis <filename>
```

Use the option [-v] to run the system in verbose mode,

```
$ ./FPEAnalysis -v <filename>
```

3 Input File Syntax

The syntax for the input files to FPEAnalysis is

```
(* comment comment. *)
{var1: val1, var2: val2, ... }
{subexpr1: val1, subexpr2: val2, ...}
program body
```

where,

- Comments are specified inside $(*\ *)$
- The list inside the first curly braces is for specifying the values (or value ranges) of each variable in the program body
- The list inside the second curly braces is for specifying any conditions on the values of subexpressions in the program body
- The program body can include expression with any of the following:
  - Real numbers: in decimal or scientific notation (such as 1.42e-4)
  - Special numbers: "machine_epsilon" and "inf"
  - Variables: an array of alphanumeric characters and underscores, but must start with a letter
  - Arithmetic operations: addition $(+)$, subtraction and negation $(-)$, multiplication $(*)$, and division $(/)$
  - Basic mathematical functions: square root $(sqrt)$, natural log (log), exponentiation (exp), and trigonometric functions $(sin,\ cos,\ tan,$ and $cot)$

Supplying the conditions for subexpression is not necessary, and the second set of curly braces can be left empty. The conditions are only meant to help guide the error analysis system. If you know that the value of a subexpression must fall within some boundaries, then add it to the conditions list. The value of a variable or a conditioned subexpression can be either a real number or a *closed* interval. The interval ends can include $\pm inf$ (use '[' and ']' to specify intervals even when using $\pm inf$).

An example of an input file is

```
(* quadratic formula *)
{a: 1, b: [100.3, 150.42], c: [10e-4, 10e-2]}
{b*b-4*a*c: [0, inf]}
(-b + sqrt(b * b - 4 * a * c)) / (2 * a)
```

See the directory FPEAnalysis/examples/ for more examples.

4 Interpreting Output

FPEAnalysis outputs a report that gives the worst case relative error of the program body (with a warning if it's high), and a list of any critical subexpressions, if any. When FPEAnalysis is run in verbose mode, it returns detailed analysis report about each subexpression of the program body.

Consider the two examples:

```
(* example 1: cancellation error example *)
{x: [1e-10, 1e-8]}
{}
sqrt(1 + x) - sqrt(1 - x) + 42 * x
```

and

```
(* example 2: stable rewrite of example 1 *)
{x: [1e-10, 1e-8]}
{}
(2 * x) / (sqrt(1 + x) + sqrt(1 - x)) + 42 * x
```

The two examples are mathematically equivalent, but the first one is numerically instable for small values of x, while the second is stable under the same values of x. The output of example 1 is

```
$ ./FPEAnalysis example1.txt

Variables bindings: {x: [1e-10, 1e-08]}
Conditions: {}
Program Expression: ((sqrt((1 + x)) - sqrt((1 - x))) + (42 * x))
************** ANALYSIS REPORT **************
Worst case relative error = 1.54914884879e-07 [HIGH RELATIVE ERROR]
1 critical subexpression was found. Consider rewriting the following:
  (sqrt((1 + x)) - sqrt((1 - x)))
```

The output of example 2 is

```
$ ./FPEAnalysis example2.txt

Variables bindings: {x: [1e-10, 1e-08]}
Conditions: {}
Program Expression: (((2 * x) / (sqrt((1 + x)) + sqrt((1 - x)))) + (42 * x))
************** ANALYSIS REPORT **************
Worst case relative error = 4.62162608124e-14
No critical subexpressions were found
```