Tutorial

FPChecker

FPChecker:

Floating-Point Profiling Through Compiler-Instrumentation

https://fpchecker.org

Ignacio Laguna

LLNL-PRES-2012776

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Agenda

Intro & Motivation	10 min
Tool's Overview	10 min
Installation & Exercises (hands-on)	35 min
Q&A	5 min

Floating-Point is Nonintuitive & Tricky to Understand

- Representation error:

  Mathematically: `0.1 + 0.2 = 0.3`

  In floating-point: `0.30000000000000004`

- Comparing two numbers for equality is unreliable

  Say we calculate `a = 0.1 + 0.2` and `b = 0.3`

  We would expect `a == b` to be true. But it's not!

Floating-Point is Nonintuitive & Tricky to Understand (2)

- Loss of precision:

  This could evaluate to `0.0`: `(10e8 + 1e-8) - 10e8`

- Cancellation:

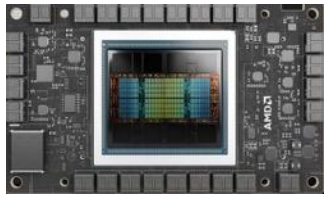
  `Suppose:  x=1.0000000000000001, y=1.0000000000000002`

  Mathematically `y-x = 0.0000000000000001`

  In floating-point: `y-x = 2.220446049250313e-16`

  Large relative error!

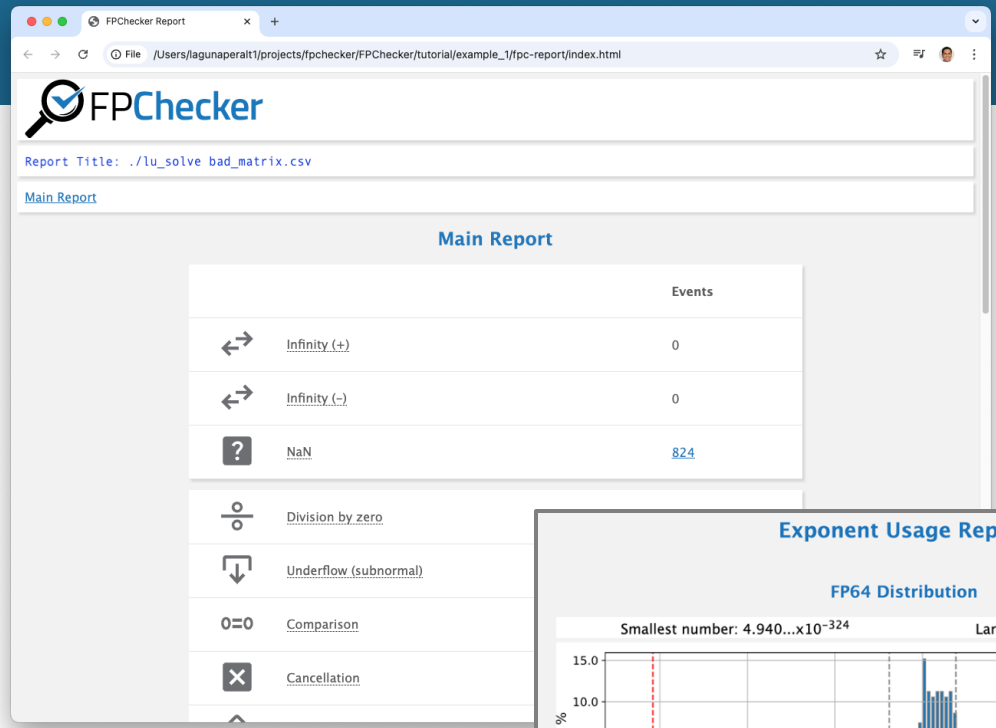
The World is Going Low-Precision

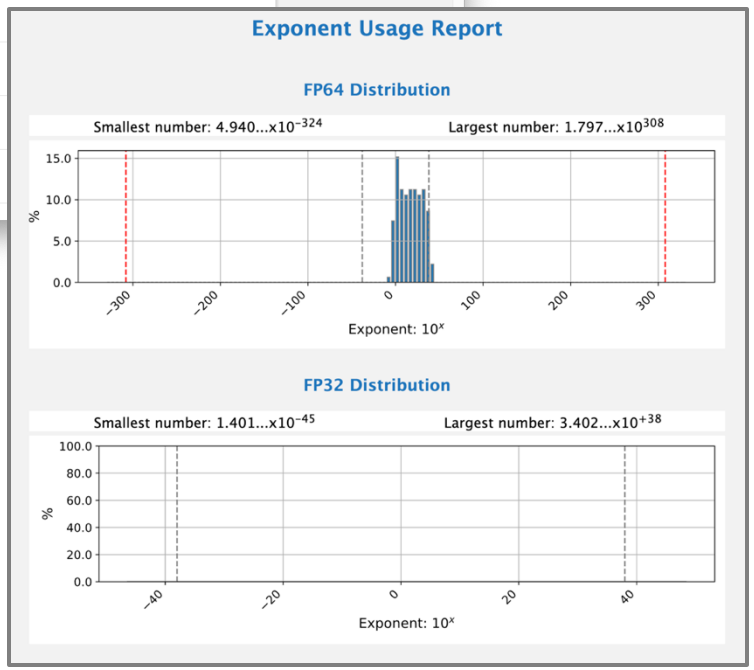


- Architectures promote lower precision formats:
  - FP16, FP8
  - Bfloat16
  - TensorFlowt-32
- We need tools to understand **mixed-precision** codes
- Important metrics to understand:
  - Dynamic range of FP32, FP16
  - Rounding error accumulation

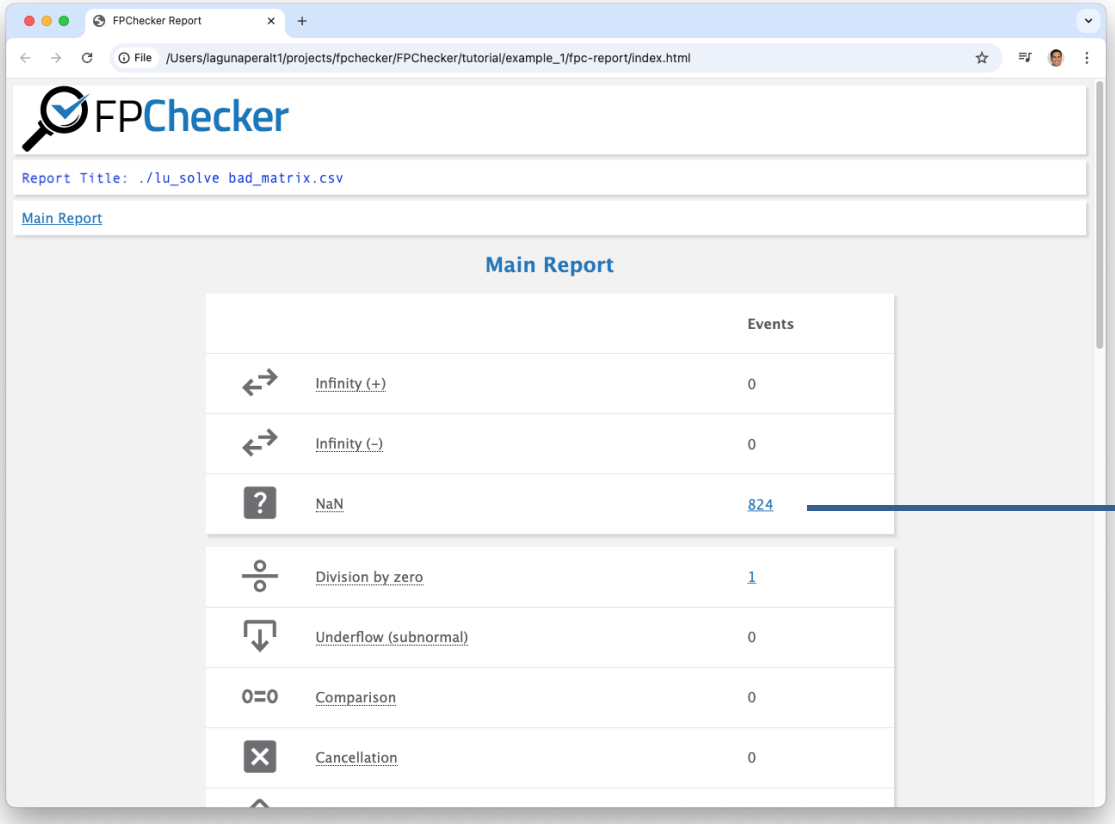
FPChecker Overview

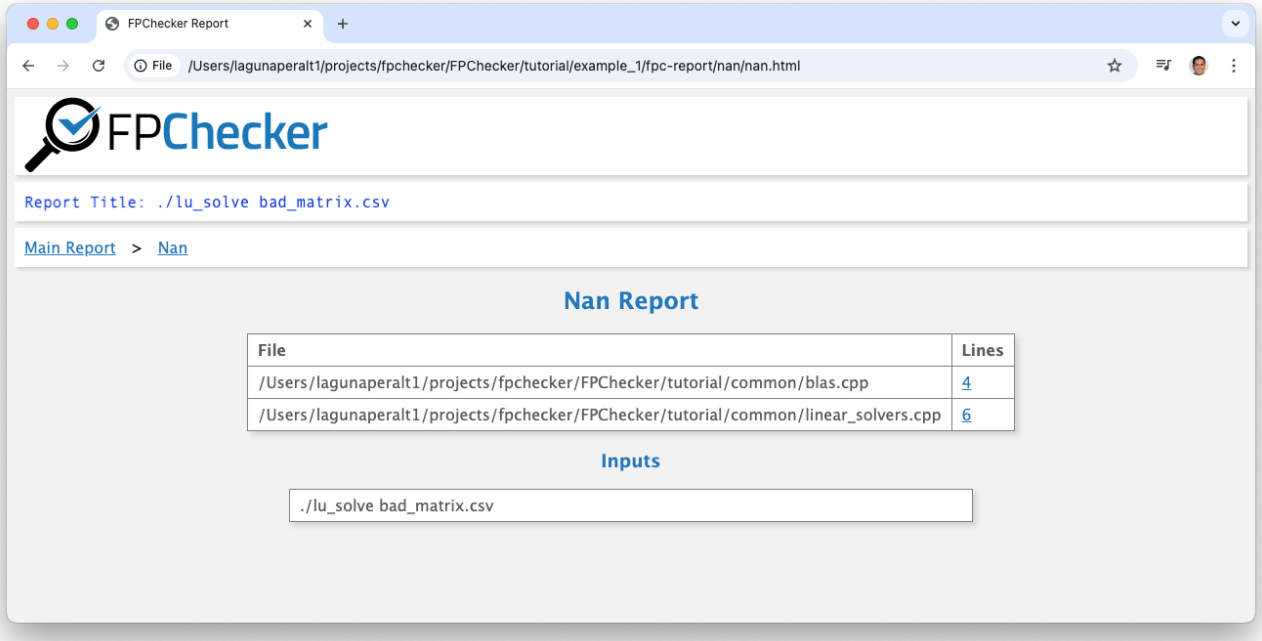
- Detects floating-point exceptions
  - NaN, Infinity
- Shows impacted *lines of code*
- Shows other "code smells"
  - Cancellations, underflows
- Analyzes dynamic range
  - Is FP32 or FP64 enough?
- Works by compiler instrumentation
  - Relies on Clang/LLVM
- Documentation: https://fpchecker.org/



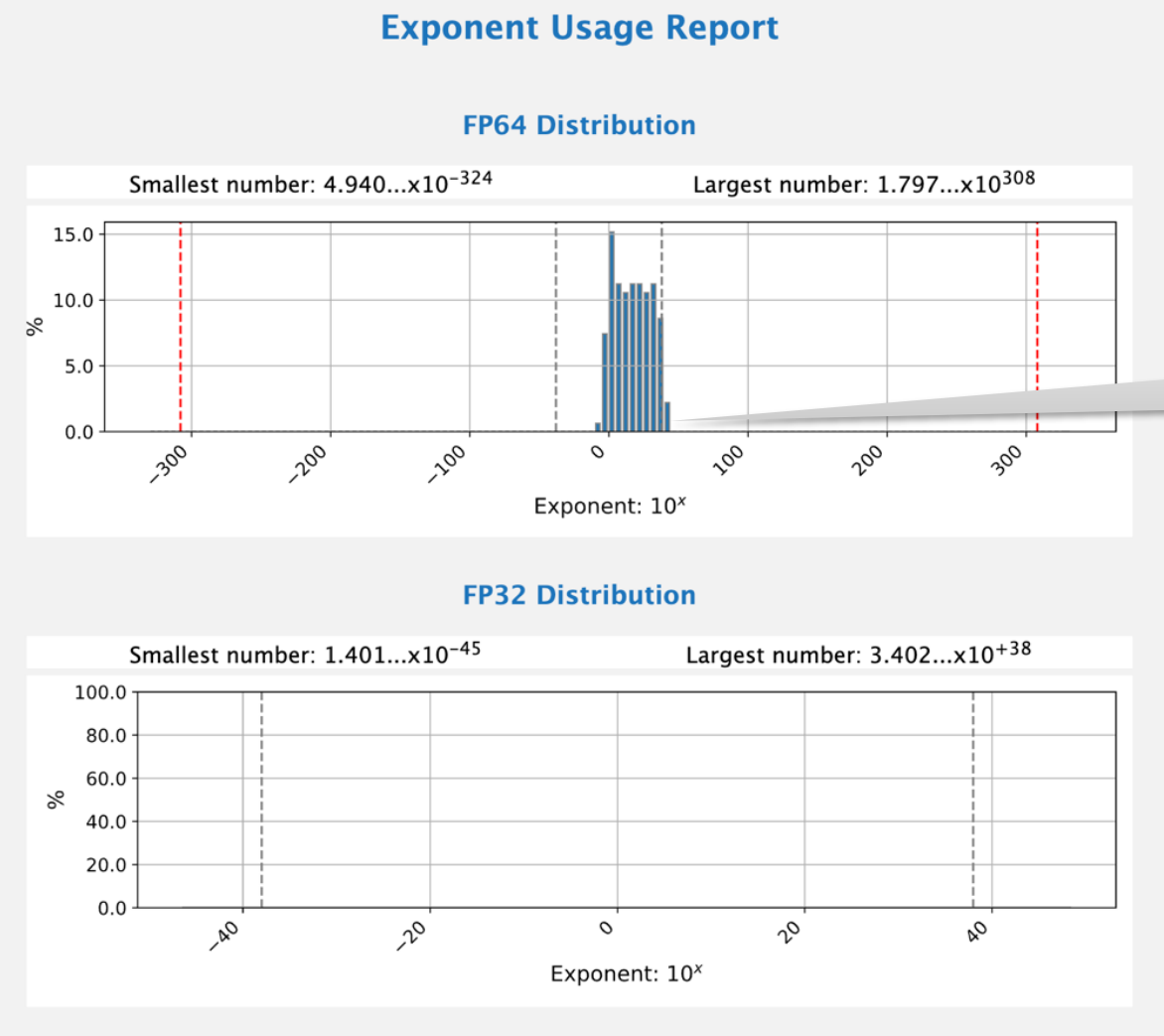


Report Example





Exponent Usage (Dynamic Range)



Out of range for FP32

LLVM Instrumentation Process

1. Use `clang++-fpchecker` wrapper in your Makefile
   - Or add instrumentation pass to your CFLAGS and/or CXXFLAGS
2. Enable `FPC_INSTRUMENT` environment variable when compiling

```
$ FPC_INSTRUMENT=1 make
```

3. Run executable
4. Create report

Two Classes of Env Variables: *Compile-time & Run-time*

✓ Covered in the Tutorial

Compile-time Variables

Variable	Type	Description
✓ FPC_INTRUMENT	Compile-time	Instruments the application
✓ FPC_ANNOTATED	Compile-time	Indicates that the program is annotated

Run-time Variables

Variable	Type	Description
✓ FPC_EXPONENT_USAGE	Run-time	Profiles exponent usage for FP32/FP64
FPC_TRAP_INFINITY_POS	Run-time	Program exits when Infinity positive is found
FPC_TRAP_INFINITY_NEG	Run-time	Program exits when Infinity negative is found
FPC_TRAP_NAN	Run-time	Program exits when NaN is found
FPC_TRAP_DIVISION_ZERO	Run-time	Program exits when division-by-zero is found
FPC_TRAP_CANCELLATION	Run-time	Program exits when cancellation is found
FPC_TRAP_COMPARISON	Run-time	Program exits when Comparison is found
FPC_TRAP_UNDERFLOW	Run-time	Program exits when underflow is found
FPC_TRAP_LATENT_INF_POS	Run-time	Program exits when Latent Infinity positive is found
FPC_TRAP_LATENT_INF_NEG	Run-time	Program exits when Latent Infinity negative is found
FPC_TRAP_LATENT_UNDERFLOW	Run-time	Program exits when Latent Underflow is found

Software Requirements

- Linux or Mac OS
  - Windows not supported
- LLVM/Clang 19
- Cmake
- Python 3.12
- Matplotlib
- Optional for parallel code:
  - MPI
  - OpenMP

Installation Process

1. Install **Conda** (this allow us to install LLVM easily)
2. Install FPChecker

How to install Anaconda on Mac or Linux

For Mac, watch this YouTube video:

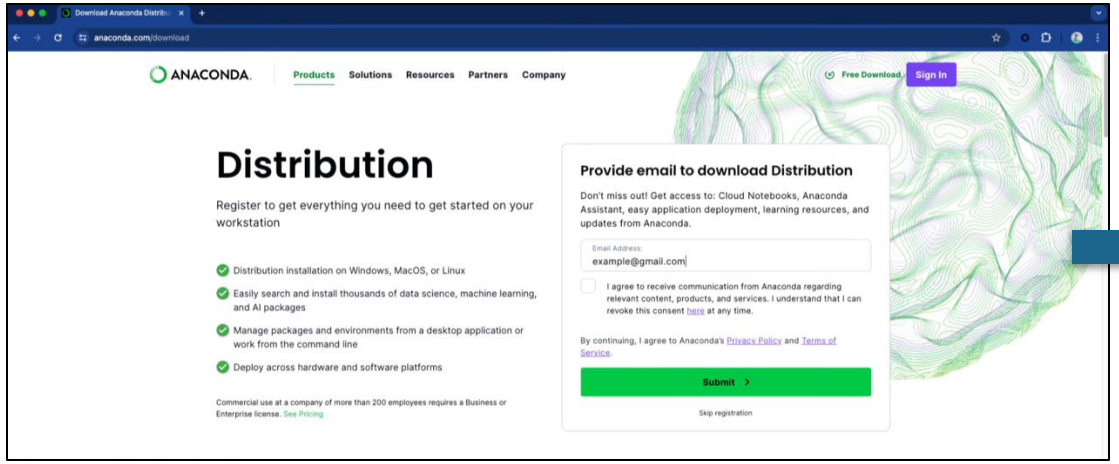
https://youtu.be/DNu8pQOYRGg

For Linux:

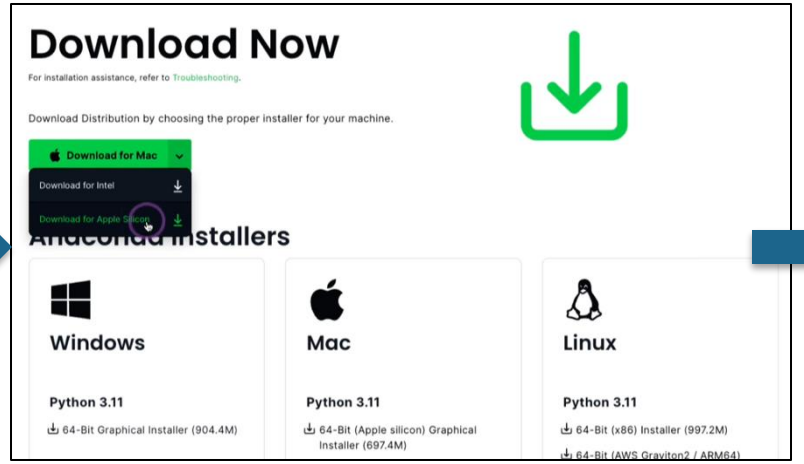
https://docs.conda.io/projects/conda/en/stable/user-guide/install/linux.html

Steps to Install Conda in Mac

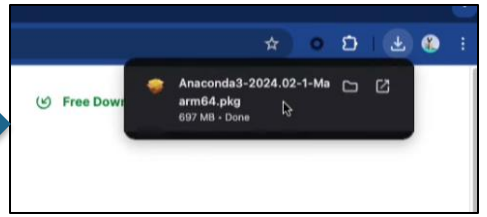
Got https://www.anaconda.com/ -> Download



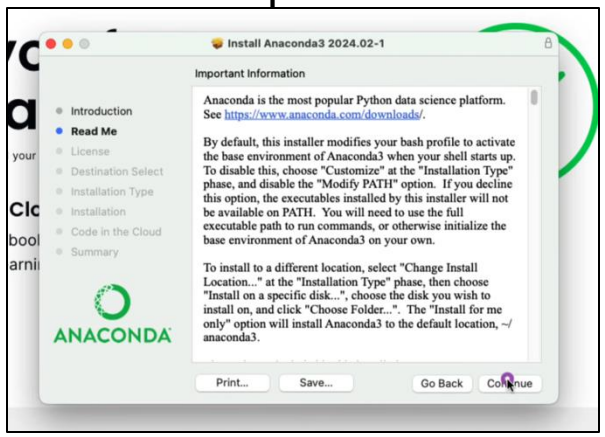
Download



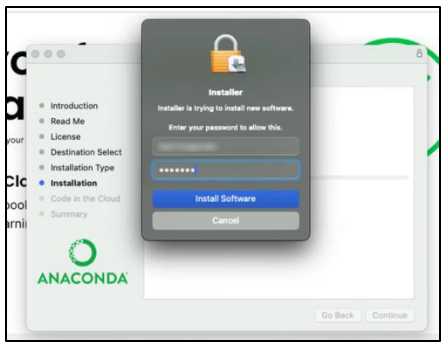
Open Installer



Follow the steps



Provide password if asked



In the terminal, you should see the word **base**:

```
(base) [user@system] $
```

Get FPChecker

This tutorial is based on release v0.5

```
   # We will install all in /tmp
$  cd /tmp
$  mkdir tutorial
$  cd tutorial

   # Clone FPChecker
$  git clone https://github.com/LLNL/FPChecker.git
$  cd FPChecker
```

Install FPChecker Dependencies with Conda (Option 1)
Manual Installation

```
   # Create a conda env
$  conda create --name tutorial_env
$  conda activate tutorial_env

   # Install dependencies: cmake, LLVM, clang++, python
$  conda install cmake
$  conda install llvmdev=19.1.7 -c conda-forge
$  conda install clangxx=19.1.7 -c conda-forge
$  conda install python=3.12.9

   # Install matplotlib. Not required to build FPChecker, but needed for reports
$  pip3 install matplotlib

   # MPI for some examples, but not required to build for FPChecker
$  conda install openmpi=5.0.7 -c conda-forge
```

Install FPChecker Dependencies with Conda (Option 2)
With environment file for Mac (ARM)

```
   # Create a conda env and dependencies
$  cd tutorial
$  conda create --name tutorial_env --file conda_environment.txt -c conda-forge
$  conda activate tutorial_env
```

- The `conda_environment.txt` file provides the packages for Mac (ARM 64-bit)
- This installs all the dependencies:
  - LLVM 19
  - Clang++ 19
  - Cmake

Install FPChecker

```
$ cd ..
$ mkdir build
$ cd build/
$ cmake -DCMAKE_INSTALL_PREFIX=../../install ..
$ make && make install

  # Export installation path
$ export PATH=/tmp/tutorial/install/bin:$PATH
```

```
# If the right python3 is not found, set the DPython3_ROOT_DIR
$ cmake -DCMAKE_INSTALL_PREFIX=../../install -DPython3_ROOT_DIR=/opt/anaconda3 ..
```

Tutorial Exercises

Tutorial Examples

	Topics	Example Program
1	NaN and Infinity exceptions	Linear system (Ax=b) solver with LU decomposition + partial pivoting
2	Exponent usage – from FP64 and FP32 precision	Finite differences + 1D Reaction-Diffusion PDE
3	Controlling slowdown with code annotations	Finite Elements + 2D Heat Conduction PDE solver
4	Analyzing parallel code: MPI/OpenMP	MPI-based Parallel Heat PDE solver

First, build the *common* library
It will be used in all examples

- Common library:
  - BLAS operations
  - Linear solvers (LU, CG)
  - Matrix & vector printing
  - Other functionalities

```
   # Compile library
$  cd /tmp/tutorial/FPChecker/tutorial/common/
$  FPC_INSTRUMENT=1 make
```

Example 1:
NaN & Infinity exception in linear solver

- Location:

  `tutorial/example_1/lu_solve.cpp`

- Program description
  - Linear solver $Ax = b$
  - Solves $Ax = 1$
  - LU decomposition: $PA = LU$
  - Partial pivoting
  - Solve by forward/backward substitution

FPChecker Use Case

- Use an ill-condition problem (matrix)
- Produces NaN and Infinity
  - U factor ends up with zero diagonal
  - Division by zero
- FPChecker locates the exceptions

Example 1: Script (Good Matrix)

```
  # Compile and run problem
$ FPC_INSTRUMENT=1 make
$ ./lu_solve matrix.csv

  # List trace files (there is one)
$ ls -l .fpc_logs/

  # Create report
$ fpc-create-report -t "./lu_solve matrix.csv"
$ open fpc-report/index.html

  # Clear logs and remove report
$ fpc-create-report -rc
```

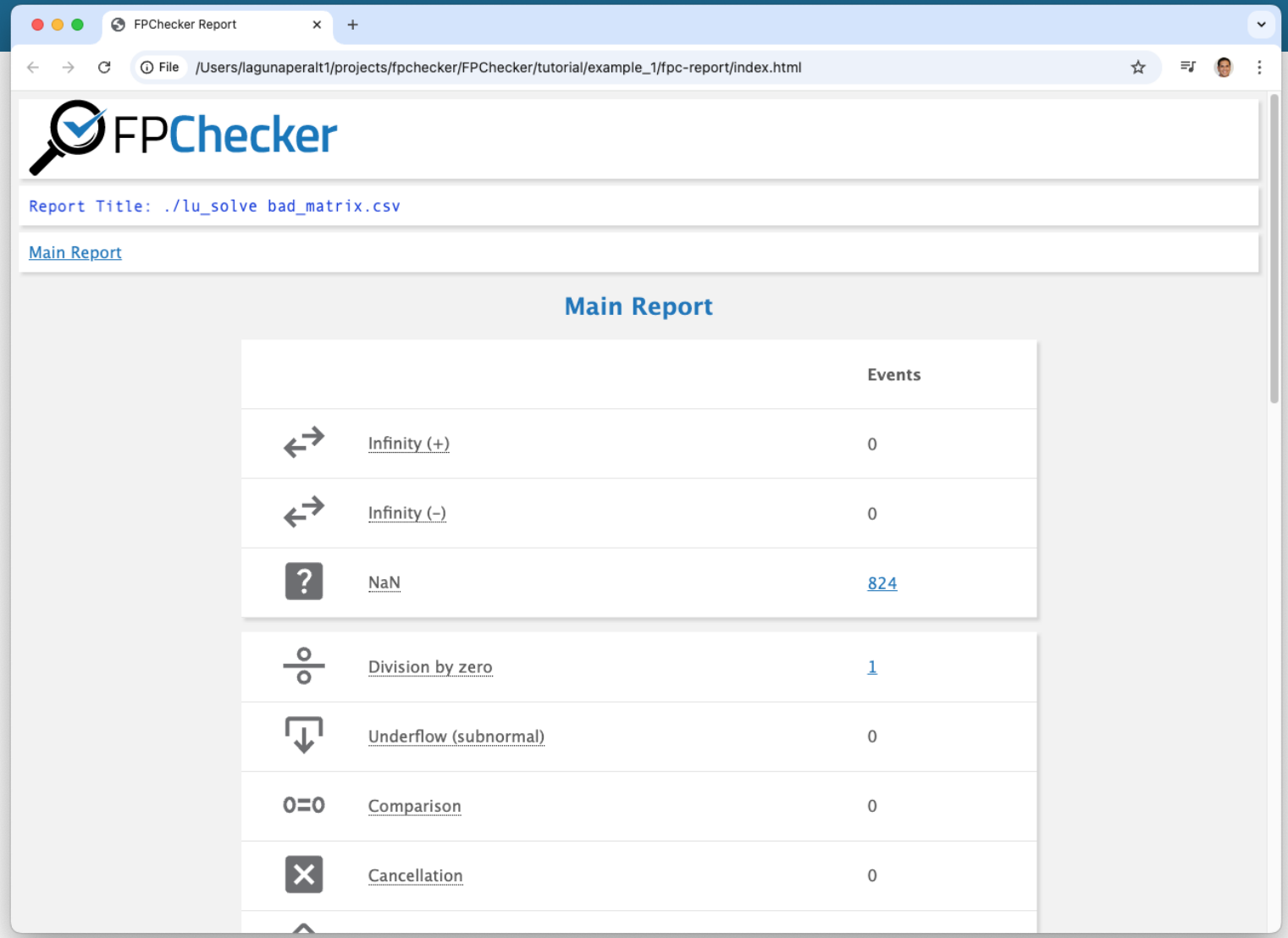
Nothing interesting in the report

Example 1: Script (Bad Matrix)

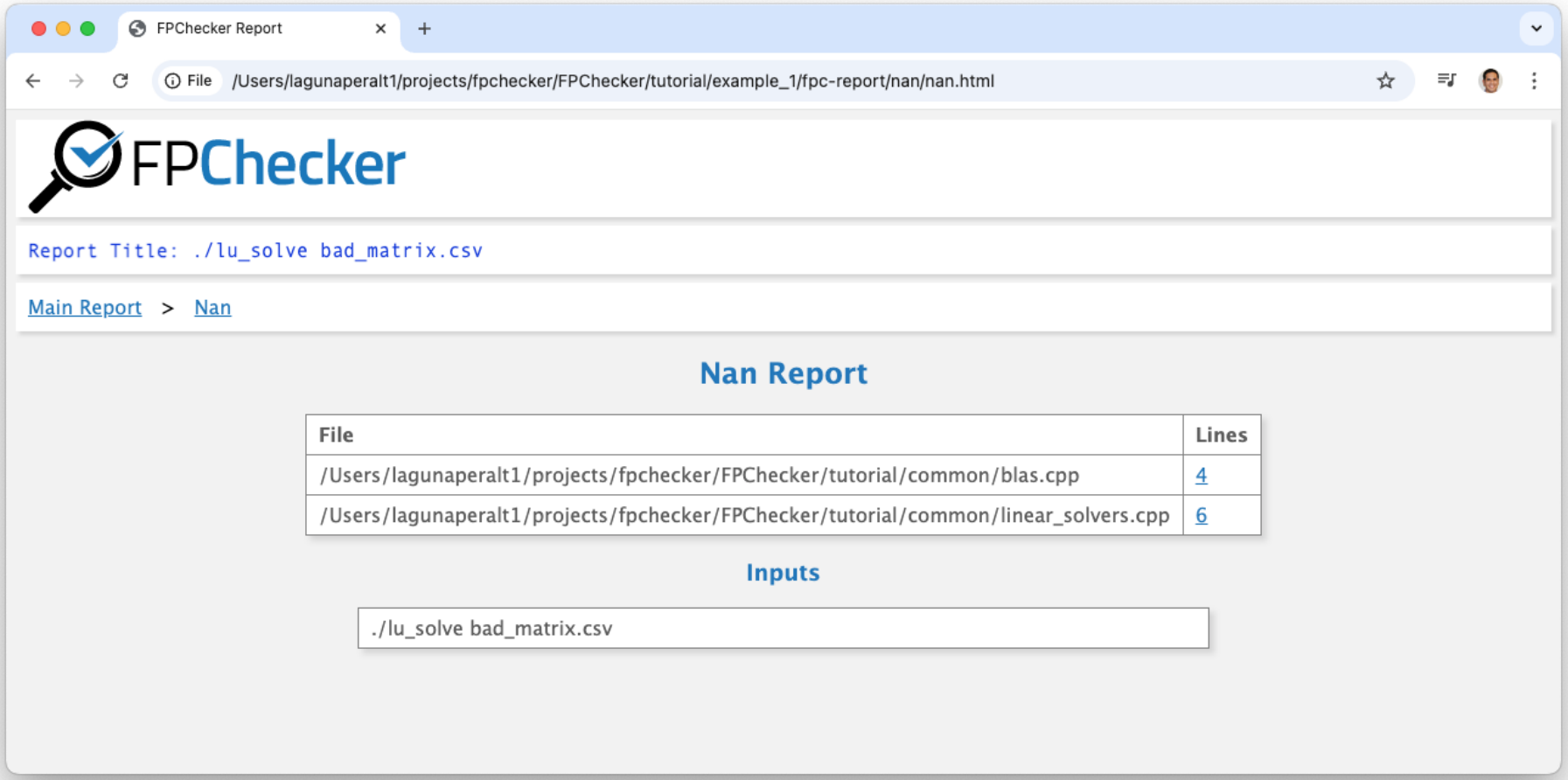
```
  # Run problem
$ ./lu_solve bad_matrix.csv

  # Create report
$ fpc-create-report -t "./lu_solve bad_matrix.csv"
$ open fpc-report/index.html
```

- NaN
- Division by zero

Report of Example 1



Report of Example 1



Tutorial Examples

	Topics	Example Program
1	NaN and Infinity exceptions	Linear system (Ax=b) solver with LU decomposition + partial pivoting
2	Exponent usage – from FP64 and FP32 precision	Finite differences + 1D Reaction-Diffusion PDE
3	Controlling slowdown with code annotations	Finite Elements + 2D Heat Conduction PDE solver
4	Analyzing parallel code: MPI/OpenMP	MPI-based Parallel Heat PDE solver

Example 2:
Exponent Usage on FP64-to-FP32 porting

- Location:

  `tutorial/example_2/reaction_diffusion.cpp`

- Program description
  - 1D linear **reaction-diffusion equation**
  - PDE: $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + \lambda u$
  - Explicit finite difference method
    - Forward Euler in time, Central Difference in space
  - Large λ provides positive feedback
    - Over time, it leads to **exponential growth**

- Code parameters:

$$D = 0.01$$

$$\lambda = 25$$

FPChecker Use Case

- Run simulation in FP64 and FP32
- Vizualize exponent usage
- In FP32, values are ”out-of-range”
  - Produce exceptions

Example 2: Script

FP64 Version

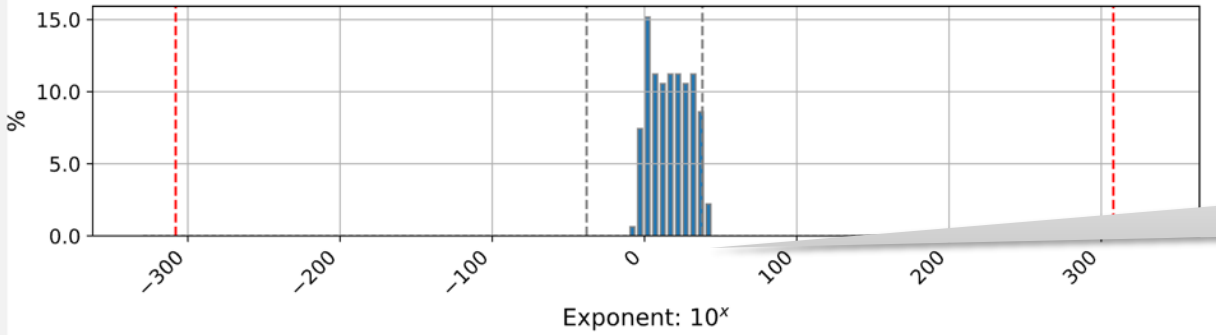
```
   # Compile and run problem
$  FPC_INSTRUMENT=1 make
$  FPC_EXPONENT_USAGE=1 ./reaction_diffusion

   # List trace files (there are two)
$  ls -l .fpc_logs/

   # Create report
$  fpc-create-report
$  open fpc-report/index.html

   # Clear logs and remove report
$  fpc-create-report -rc
```

Report (Example 1, FP64)

Exponent Usage Report

FP64 Distribution

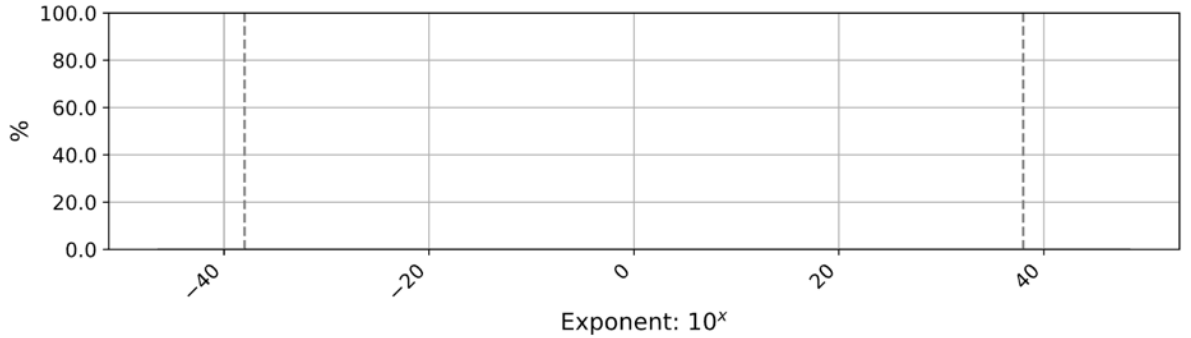
Smallest number: $4.940...x10^{-324}$ Largest number: $1.797...x10^{308}$



Out of range for FP32

FP32 Distribution

Smallest number: $1.401...x10^{-45}$ Largest number: $3.402...x10^{+38}$



Example 2: Script

First Step:

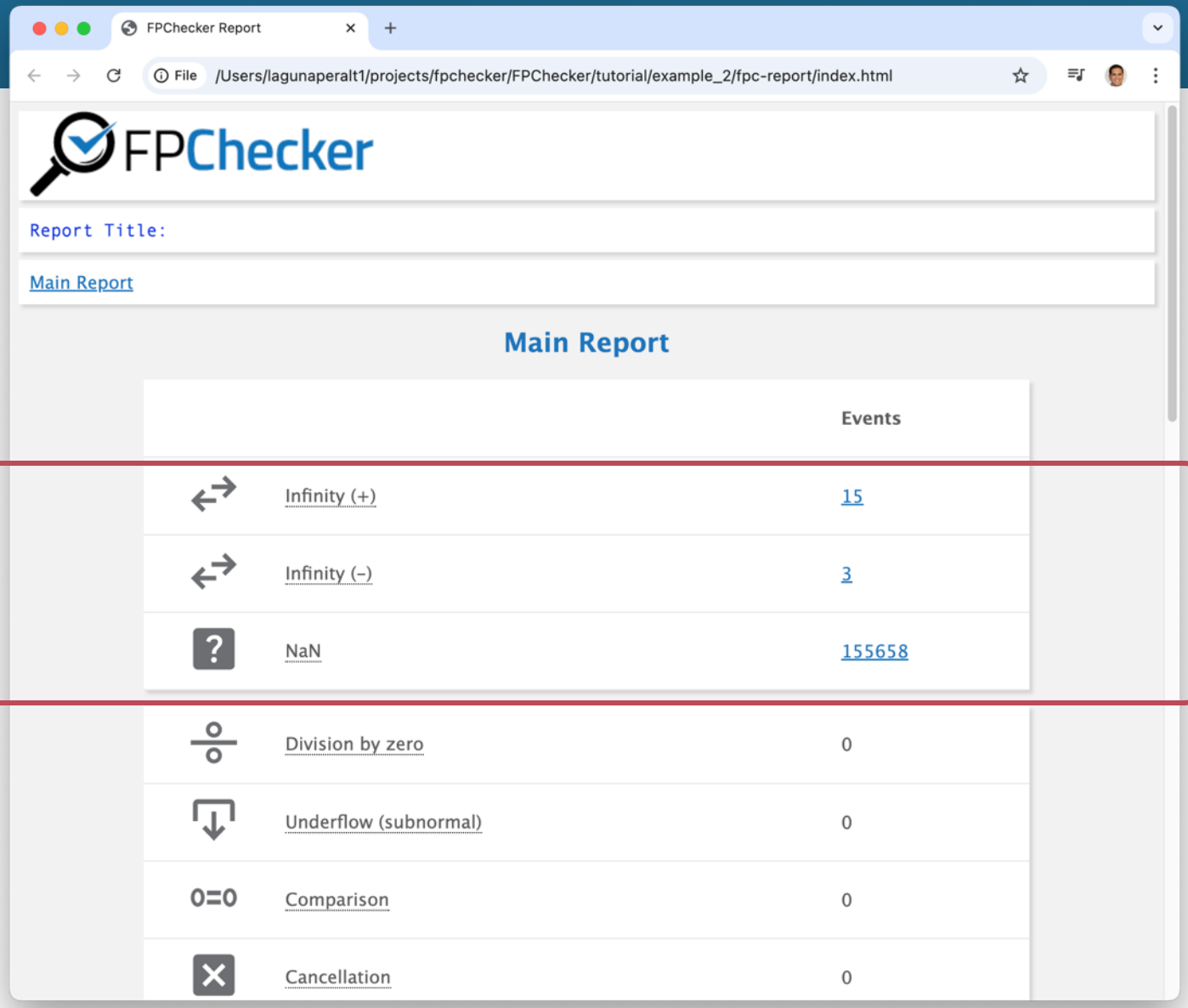
- Open `reaction_diffusion.cpp`
- Modify lines 7-8 to use FP32

```
typedef double Real_t;
// typedef float Real_t;
```

FP32 Version

```
   # Compile and run problem
$  FPC_INSTRUMENT=1 make
$  FPC_EXPONENT_USAGE=1 ./reaction_diffusion

   # List traces files (there are two)
$  ls -l .fpc_logs/

   # Create report
$  fpc-create-report
$  open fpc-report/index.html

   # Clear logs and remove report
$  fpc-create-report -rc
```

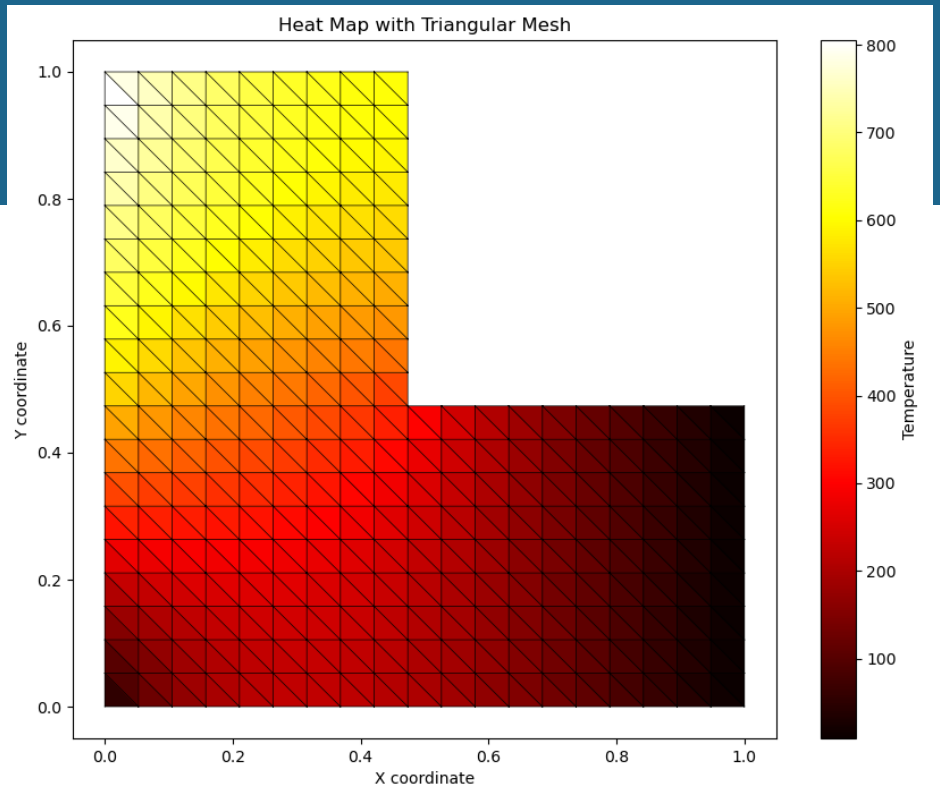
Report (Example 2, FP32)



Tutorial Examples

	Topics	Example Program
1	NaN and Infinity exceptions	Linear system (Ax=b) solver with LU decomposition + partial pivoting
2	Exponent usage – from FP64 and FP32 precision	Finite differences + 1D Reaction-Diffusion PDE
3	Controlling slowdown with code annotations	Finite Elements + 2D Heat Conduction PDE solver
4	Analyzing parallel code: MPI/OpenMP	MPI-based Parallel Heat PDE solver

Example 3: Annotations to Control Slowdown



- Location:

  `tutorial/example_3/heat_PDE_finite_elements.cpp`

- Program description
  - **2D Heat conduction equation** with a source term
    - Steady state
  - PDE: $\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = s(x, y)$
  - Finite elements method
    - Triangular shapes
  - $T(x, y)$: temperature distribution
  - Domain: L shape
  - Source $s(x, y) = 0$
  - Boundary conditions: temp applied on the sides

FPChecker Use Case

- Slowdown can be high (for a large problem)
- Reduce slowdown by annotating the code

Example 3: Script

- Run small problem (10 nodes)
- Should take less than 1 second

```
   # Compile and run problem
$  FPC_INSTRUMENT=1 make
$  time FPC_EXPONENT_USAGE=1
$  ./heat_PDE_finite_elements 10

   # Optional: Vizualize heat map
$  python3 plot.py
```

- Run a larger problem (50 nodes)
- It takes about 20 seconds in my laptop

```
   # Compile and run problem
$  time FPC_EXPONENT_USAGE=1 ./heat_PDE_finite_elements 50
```

Code Annotations

- Let's annotate the matrix-multiply function
  - That is: we only instrument and analyze that function
  - Should reduce overhead significantly

- Location:

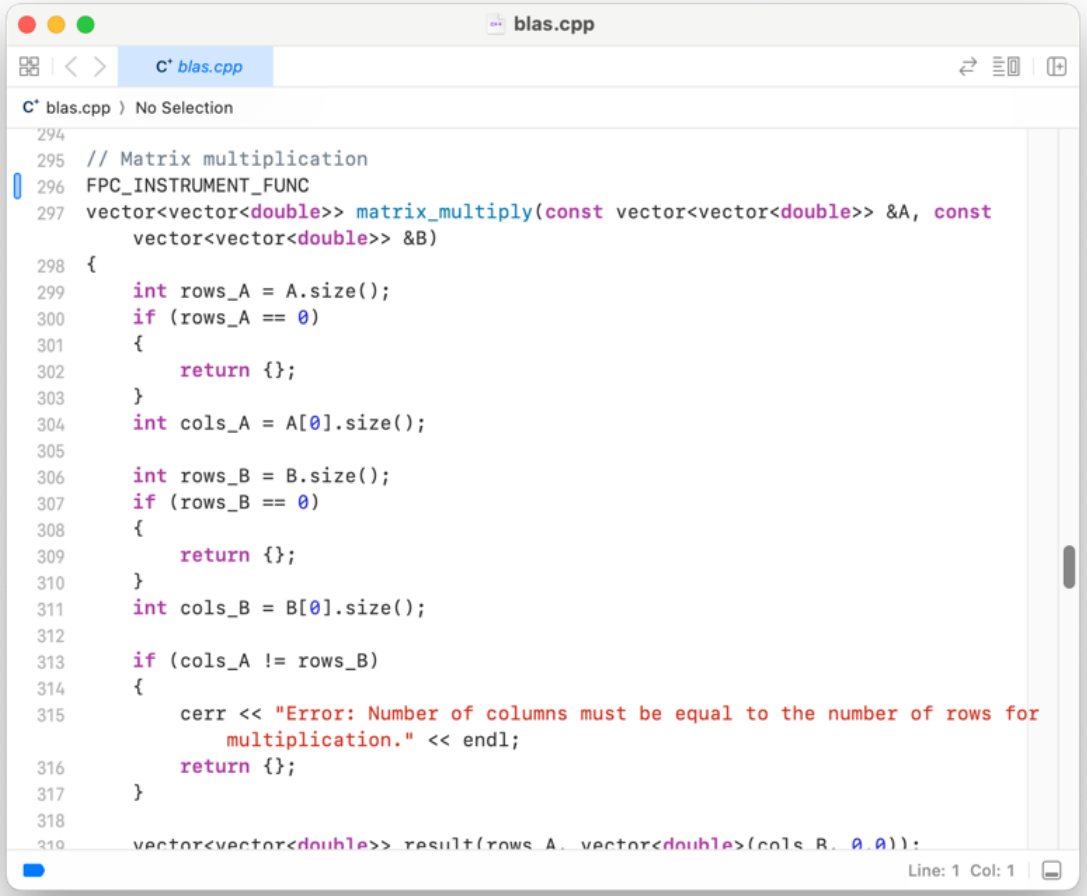
  `tutorial/common/blas.cpp`

- Search for:

  `vector<vector<double>> matrix_multiply(…`

- Add (or uncomment):

  `FPC_INSTRUMENT_FUNC`



Example 3: Script

Recompile the common library

```
$  cd ../common/
$  make clean
$  FPC_INSTRUMENT=1 FPC_ANNOTATED=1 make

$  cd ../example_3/
$  FPC_INSTRUMENT=1 FPC_ANNOTATED=1 make
$  time FPC_EXPONENT_USAGE=1 ./heat_PDE_finite_elements 50
   ...
   ...
   real    0m1.049s
   user    0m0.729s
   sys     0m0.071s
```

Lower run time

Tutorial Examples

	Topics	Example Program
1	NaN and Infinity exceptions	Linear system (Ax=b) solver with LU decomposition + partial pivoting
2	Exponent usage – from FP64 and FP32 precision	Finite differences + 1D Reaction-Diffusion PDE
3	Controlling slowdown with code annotations	Finite Elements + 2D Heat Conduction PDE solver
4	Analyzing parallel code: MPI/OpenMP	MPI-based Parallel Heat PDE solver

Example 4: Analyzing MPI code

- Location:

  `tutorial/example_4/heat_mpi.cpp`

- Program description
  - 1D heat equation
  - PDE: $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$
  - Explicit finite difference method
- MPI domain decomposition:
  - 1D spatial grid divided into NPROC contiguous segments
  - NPROC: number of MPI processes
  - Each process computes temperature in its segment

FPChecker Use Case

- Generates traces for MPI programs
- Combine traces into a single report

Example 4: Script

```
   # Show Makefile uses CXX = mpic++-fpchecker
   # Compile and run problem
   # OMPI_CXX indicates to Open MPI which conda compiler to use
$  OMPI_CXX=clang++ FPC_INSTRUMENT=1 make
$  FPC_EXPONENT_USAGE=1 mpiexec -n 4 ./heat_mpi

   # List trace files (there are 4)
$  ls .fpc_logs/
   fpc_king01_95250.json fpc_king01_95251.json
   fpc_king01_95252.json fpc_king01_95253.json

   # Create report
$  fpc-create-report
$  open fpc-report/index.html
```

Showing the Configuration of the Installation

```
$ fpchecker-show
=====================================
         FPChecker Configuration
=====================================

Installation path: /tmp/tutorial/install

Add this to CFLAGS and/or CXXFLAGS:
-g -include /tmp/tutorial/install/src/Runtime_cpu.h -fpass-plugin=/tmp/tutorial/install/lib/libfpchecker_cpu.dylib

Wrappers are located here:
/tmp/tutorial/install/bin/clang-fpchecker
/tmp/tutorial/install/bin/clang++-fpchecker
/tmp/tutorial/install/bin/mpicc-fpchecker
/tmp/tutorial/install/bin/mpicxx-fpchecker
```

Conda Commands:

- Removing an environment:
  `conda env remove --name <environment_name>`
- Adding an environment:
  `conda env --name <environment_name>`

Demo FPChecker in Some Packages

Package		Build Model
SuperLU	Linear solver	C, cmake
Hypre	Linear solver	C, cmake, MPI
LAMMPS	Molecular dynamics	C, C++, cmake
FFTW	Fourier transform	C, autotools

Example: SuperLU

```
$  git clone git@github.com:xiaoyeli/superlu.git
$  git clone https://github.com/xiaoyeli/superlu.git
$  cd superlu
$  git checkout 4ef39075e029927e8c959b22c8e7052dcb40c995
$  mkdir build
$  cd build
$  CC=clang-fpchecker cmake -DCMAKE_INSTALL_PREFIX=./install -Denable_internal_blaslib=ON ..
$  FPC_INSTRUMENT=1 make -j
   make install
```

- Configure to build internal BLAS
  - There seems to be an error with fpchecker-clang and Cmake finding BLAS in Mac

Example: FFTW

```
$  wget https://www.fftw.org/fftw-3.3.10.tar.gz
$  tar -zxvf fftw-3.3.10.tar.gz
$  cd fftw-3.3.10
$  CC=clang-fpchecker ./configure --disable-fortran --prefix=/tmp/tutorial/examples/fftw/fftw-
   3.3.10/fftw-install
$  FPC_INSTRUMENT=1 make –j
$  make install
```

- See example at:
  - FPChecker/tutorial/other_examples/fftw:
    - fftw_test.c

Example: LAMMPS

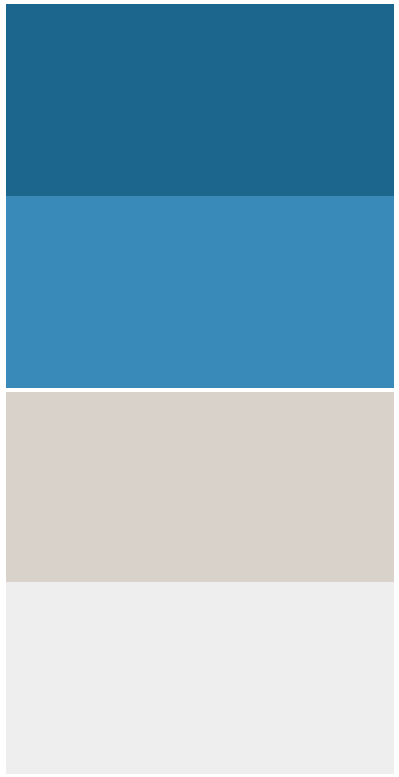
```
$  wget https://github.com/lammps/lammps/archive/refs/tags/stable_29Aug2024_update2.tar.gz
$  tar -xvf stable_29Aug2024_update2.tar.gz
$  cd lammps-stable_29Aug2024_update2/
$  cd cmake/
$  mkdir build
$  cd build
$  CC=clang CXX=clang++ cmake \
   -DCMAKE_CXX_FLAGS="-include /tmp/tutorial/install/src/Runtime_cpu.h -fpass-
   plugin=/tmp/tutorial/install/lib/libfpchecker_cpu.dylib -g" \
   -DBUILD_MPI=OFF -DBUILD_OMP=OFF -DBUILD_FORTRAN=OFF \
   -DBUILD_SHARED_LIBS=OFF -DENABLE_TESTING=OFF ..
$  FPC_INSTRUMENT=1 make -j
$  ./lmp -in ../../examples/melt/in.melt
```

Example: Hypre

```
$  git clone git@github.com:hypre-space/hypre.git
$  cd hypre
$  git checkout be52325a3ed8923fb93af348b1262ecfe44ab5d2
$  cd src
$  mkdir build
$  cd build
$  CC=clang cmake -DHYPRE_ENABLE_MPI=ON \
   -DHYPRE_WITH_EXTRA_CFLAGS="-include /tmp/tutorial/install/src/Runtime_cpu.h -fpass-
   plugin=/tmp/tutorial/install/lib/libfpchecker_cpu.dylib -g" ..
$  FPC_INTRUMENT=1 make –j
$  make install
```

- HYPRE is installed in:
  - /tmp/tutorial/examples/hypre/src/hypre
- Test in tutorial/other_examples/hypre:
  - To compile example:
    FPC_INSTRUMENT=1 OMPI_CC=clang make
    HYPRE_MATRIX=matrix.csv ./hypre_test 0

Color Palette



Contrast red