Subject: FP exception vs NaN
Posted by koldo on Mon, 26 Oct 2020 20:08:10 GMT
View Forum Message <> Reply to Message

Hi Didier

Influential people think that divisions by zero or negative square roots are admissible in a good code. I do not agree.
IMHO a NaN or an Inf is a failure in code, it should be debugged and fixed. Divisions by zero or negative sqrt... are just bugs. If your code considers these situations as normal, my friend, you are screwed. I wouldn't put money in your bank

To avoid problems with existing code that allows these situations, a new Bazaar package has to be opened.

Didier, I will open a first functional version in few days, but your support and that of other colleagues will be very acknowledged.

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 08:23:49 GMT
View Forum Message <> Reply to Message

koldo wrote on Fri, 16 October 2020 09:15I would like to capture floating point errors.
For now in case of error the program execution follows, but you can find in the doubles, things like INF or NAN.
This behaviour is very conservative for my apps, and I'd prefer in some situations to capture these situations stopping the execution.

I have tried it unsuccessfully using:
- signal(SIGFPE, ...)
- _controlfp_s()
- _set_se_translator()

Any help will be acknowledged!

PD. Somebody could say "just check that all data is adequate before using it in division, sqrt, ...". That's true, but difficult and cumbersome in some situations using uncontrolled data sources where sometimes even doing lots of "if" to check the consistency of data, that's not enough.

Ah, that explains it. So you have like 2 weeks of experience with this issue

I was all hurray 3 years ago into catching all FP exceptions and avoiding all NaNs. Not so sure today....

Just consider this: Historically, FP exceptions were default and always active, divide by zero or negative sqr always stopped the code just like dereferencing NULL. Then IEEE basically replaced this behaviour with NaNs as default. Maybe they knew something?

If you want a very simple contraexample where NaN is definitely better, look at this:

https://www.codeproject.com/Articles/1074135/Evaluating-expr
ession-with-descend-parser-and-Uplu

Mirek

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 10:09:34 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 10:58CrashHandler works only in DEBUG and is connected to TheIDE so when debugging you can capture floating point (FP) and other problems. Production code could not use CrashHandler.

Which is another stupid mistake. You expect to catch all those FP bugs during debugging? I can tell with 100% probability that you are going to miss something, because it all depends on input data. So the end result of your effort will be that in release you will be getting unexpected NaNs and the whole thing will collapse.

Quote:
Division by zero and other FP errors, are errors and have to be debugged. Just imagine two situations:
- The calculation of your bank account balance is infinite

No. If the end result of complex operation is infinite, the whole operation has to be rejected. In your application, it will be infinite, because you thought that you have fixed all FP bugs and you do not have any safeguards for unexpected inputs

Quote:
- The controller that determines the direction of your car with automatic driving sets an infinite angle

Which is exactly what can happen with your approach. Correctly written code will reject invalid angle.

Quote:I do not want my code to fall in these situations.

But you are going exactly into that trap. As I said, been there....

Mirek

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 10:16:29 GMT
View Forum Message <> Reply to Message

Yet another example, how are you going to fix this:

```
double SomeFunction(double x)
{
   return ln(x*x*x - 15*x*x + x + 20);
}
```

Now this function is used throughout your code on 50 places and you have got an exception because the argument is negative.

How are you going to fix it?

This is the main problem I have encounterd with "activate FP exception" approach and I still do not know the correct solution....

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 10:36:10 GMT
View Forum Message <> Reply to Message

Quote:Ah, that explains it. So you have like 2 weeks of experience with this issue After thirty five years programming, I think it's better late than never.
If I'd be an expert in this issue, I wouldn't ask for advice.

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 10:38:05 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 27 October 2020 11:16Yet another example, how are you going to fix this:

```
double SomeFunction(double x)
{
   return ln(x*x*x - 15*x*x + x + 20);
```

}

Now this function is used throughout your code on 50 places and you have got an exception because the argument is negative.

How are you going to fix it?

This is the main problem I have encounterd with "activate FP exception" approach and I still do not know the correct solution....

MirekIn my code I would put an ASSERT to capture the error in DEBUG. Just like Vector checks if index is negative.

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 10:40:47 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 11:38mirek wrote on Tue, 27 October 2020 11:16Yet another example, how are you going to fix this:

```
double SomeFunction(double x)
{
   return ln(x*x*x - 15*x*x + x + 20);
}
```

Now this function is used throughout your code on 50 places and you have got an exception because the argument is negative.

How are you going to fix it?

This is the main problem I have encounterd with "activate FP exception" approach and I still do not know the correct solution....

MirekIn my code I would put an ASSERT to capture the error in DEBUG. Just like Vector checks if index is negative.

Yeah, so your program now does not work in debug for particular input data. What is the next step?

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 16:18:17 GMT
View Forum Message <> Reply to Message

Quote:Yeah, so your program now does not work in debug for particular input data. What is the next step?

The next step when TheIDE debugger breaks is to solve the problem

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 17:52:37 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 17:18Quote:Yeah, so your program now does not work in debug for particular input data. What is the next step?

The next step when TheIDE debugger breaks is to solve the problem

But how? There is obviously a combination of input data that leads to FP exceptions. How are you going to solve it?

If you wanted to check the inputs before processing them, you would have to trace back all calculations back to inputs and do other calculations just make sure this particucal input dataset does not cause the problem.

That is the part I never solved and that makes me appreciate the idea of using NaN for what it was indended too.

Mirek

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 18:19:41 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 17:18Quote:Yeah, so your program now does not work in debug for particular input data. What is the next step?

The next step when TheIDE debugger breaks is to solve the problem

Or if you want more practical example, take

examples/FnGraph

then activate fp exceptions, start the debugger and enter ln(x). Then tell me how you fixed the code....

Then try to imagine all the fixes you will need to make to avoid all FP exceptions and I can guarantee you two things:

a) you have probably missed something
b) the code will be about 2 times longer and much harder to maintain

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 18:25:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 27 October 2020 19:19koldo wrote on Tue, 27 October 2020 17:18Quote:Yeah, so your program now does not work in debug for particular input data. What is the next step?

The next step when TheIDE debugger breaks is to solve the problem

Or if you want more practical example, take

examples/FnGraph

then activate fp exceptions, start the debugger and enter ln(x). Then tell me how you fixed the code....

Then try to imagine all the fixes you will need to make to avoid all FP exceptions and I can guarantee you two things:

a) you have probably missed something
b) the code will be about 2 times longer and much harder to maintain

Mirek

Sorry Mirek

What you say is not conceivable for me. In my world a negative square root is the same as a Vector v(2); v[-1] = 2;
The only explanation is that we work in very different computing environments.

Subject: Re: Capture division by zero
Posted by Klugier on Tue, 27 Oct 2020 19:35:52 GMT
View Forum Message <> Reply to Message

Hello,

I slowly stop understanding what this thread is about? I think it is perfectly fine to validate input each time it can be wrong - this is the secure coding principles. So, in example bring by kold when i (-1) "v(2); v[i] = 2;", we should always check and handling appropriate. The same is true for divided by zero problem.

The only problem I have is that the CrashHandler is forced by using SysInfo package on Debug mode. For me the class should not be there in nay cases. If you would like to write such debug code just do it in your application not in the library. This is my postulate. If this code will help you find errors in your application that's fine, but use it only there.

EDIT: I saw that crash handler was removed from SysInfo. Thanks!

Klugier

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 19:47:09 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 27 October 2020 20:35Hello,
I slowly stop understanding what this thread is about?

I am trying to discuss relative merits of using FP exceptions vs NaNs. Mostly because this is still a hot topic for me. Please forget about CrashHandler, that problem is solved (thanks), this is important and interesting topic.

In one way I am just checking that my latest mind model of situation is correct...

Quote:
I think it is perfectly fine to validate input each time it can be wrong - this is the secure coding principles.

It is perfectly fine when possible. Usually it is not or it is too complex to rely on.

Mirek

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 19:50:30 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 19:25
What you say is not conceivable for me. In my world a negative square root is the same as a

Vector v(2); v[-1] = 2;

Cool, then please fix example/FnGraph.

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Wed, 28 Oct 2020 08:24:37 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 27 October 2020 20:50koldo wrote on Tue, 27 October 2020 19:25
What you say is not conceivable for me. In my world a negative square root is the same as a
Vector v(2); v[-1] = 2;

Cool, then please fix example/FnGraph.

Mirek

In this case my old Casio calculator prints an error message.

Edited: Just checked with my daughter's CASIO Graph 35+E. 1/0 prints "Math ERROR. Press EXIT". Really updated, this version even comes with Python.

Subject: Re: Capture division by zero
Posted by koldo on Wed, 28 Oct 2020 08:29:11 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 27 October 2020 20:35Hello,

I slowly stop understanding what this thread is about? I think it is perfectly fine to validate input each time it can be wrong - this is the secure coding principles. So, in example bring by kold when i (-1) "v(2); v[i] = 2;", we should always check and handling appropriate. The same is true for divided by zero problem.

The only problem I have is that the CrashHandler is forced by using SysInfo package on Debug mode. For me the class should not be there in nay cases. If you would like to write such debug code just do it in your application not in the library. This is my postulate. If this code will help you find errors in your application that's fine, but use it only there.

EDIT: I saw that crash handler was removed from SysInfo. Thanks!

Klugier

CrashHandler capture of FP errors was activated only in DEBUG mode, like U++ Vector bounds calculation: no overhead added to RELEASE.
U++ Vector bounds check is forced by default in DEBUG. It seemed fair to do the same with other errors.

Subject: Re: Capture division by zero
Posted by mirek on Wed, 28 Oct 2020 12:51:14 GMT
View Forum Message <> Reply to Message

koldo wrote on Wed, 28 October 2020 09:24mirek wrote on Tue, 27 October 2020 20:50koldo wrote on Tue, 27 October 2020 19:25
What you say is not conceivable for me. In my world a negative square root is the same as a Vector v(2); v[-1] = 2;

Cool, then please fix example/FnGraph.

Mirek

In this case my old Casio calculator prints an error message.

But so does FnGraph, that is the point (well, actually, it does not print the error, just does not render the graph in areas where the function is undefined). But if you activate FP exception, it will crash on it.

Quote:
Edited: Just checked with my daughter's CASIO Graph 35+E. 1/0 prints "Math ERROR. Press EXIT". Really updated, this version even comes with Python.

Sure, because it gets NaN and interprets correctly as error...

Mirek

Subject: Re: Capture division by zero
Posted by mirek on Wed, 28 Oct 2020 13:04:47 GMT
View Forum Message <> Reply to Message

I am also pretty interested what will you do when you get exception in eigen. I am not that advanced in math, but I am pretty sure that e.g. matrix inversion can produce a lot of divide by zero for degenerate matrix and I am also pretty sure that eigen follows NaN model like every other

mature library (if for nothing else, then for performance reasons).

Are you going to compute determinant before each call to inversion?

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Thu, 29 Oct 2020 13:19:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 28 October 2020 14:04I am also pretty interested what will you do when you get exception in eigen. I am not that advanced in math, but I am pretty sure that e.g. matrix inversion can produce a lot of divide by zero for degenerate matrix and I am also pretty sure that eigen follows NaN model like every other mature library (if for nothing else, then for performance reasons).

Are you going to compute determinant before each call to inversion?

Mirek
I am well aware of operations that could be bad conditioned . I use a lot matrix inversion.
In fact STEM4U includes some tools to handle infinite precision numbers (well, but as expected, very inefficiently)
If you ask me if I prefer to search for NaNs after every matrix algebra operation, or catching an exception, I will choose the second.
And remember that U++ is not for people who are content to following STL or BOOST, but is for people who wants more performance and simplicity of use. There are other libraries more compliant but, unfortunately for them, they are worse than U++. And you are the biggest culprit of it .

Subject: Re: Capture division by zero
Posted by mirek on Mon, 02 Nov 2020 09:31:01 GMT
View Forum Message <> Reply to Message

koldo wrote on Thu, 29 October 2020 14:19mirek wrote on Wed, 28 October 2020 14:04I am also pretty interested what will you do when you get exception in eigen. I am not that advanced in math, but I am pretty sure that e.g. matrix inversion can produce a lot of divide by zero for degenerate matrix and I am also pretty sure that eigen follows NaN model like every other mature library (if for nothing else, then for performance reasons).

Are you going to compute determinant before each call to inversion?

Mirek
I am well aware of operations that could be bad conditioned . I use a lot matrix inversion.
In fact STEM4U includes some tools to handle infinite precision numbers (well, but as expected, very inefficiently)

If you ask me if I prefer to search for NaNs after every matrix algebra operation, or catching an exception, I will choose the second.

Now actually catching the exception as part of program logic in release mode (as oppposed to just stopping on NaN in debug), that is completely different thing! I could agree that would be a good solution, except for one nasty tiny detail: you cannot officially catch FP exceptions in C++ (see e.g. https://stackoverflow.com/questions/38221471/how-to-convert- sigfpe-into-a-c-exception). It might be possible on specific platforms / compilers, but I would think twice before making your code reliant on it

All you can realistically do in the handler is to set some per-thread and/or global "fp-error" flag. That frankly does sound quite similar to NaN...

All that said, activating FP exception might have its uses, e.g. when you are getting NaN out of calculation for unknown reasons and you want to track it down.

I just do not think that it is the only valid way. And in my humble experience, insisting that your code never produces NaNs creates much more mess than it solves.

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Mon, 02 Nov 2020 17:07:35 GMT
View Forum Message <> Reply to Message

Yes, AFAIK it is not possible to catch FP errors in RELEASE mode, at least without hurting performance.
Because of this I wanted to handle them only in DEBUG mode.
