U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

Extremely Low Probability of Rupture Code: 2026 Update

Christopher Nellis
U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Division of Engineering
Reactor Engineering Branch

Michael Magyar
U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Division of Engineering
Reactor Engineering Branch

Agenda

- Background
- xLPR Version 3.0
  - What is the xLPR Framework?
  - Why are we replacing the Framework?
- Project Objectives
- Project Timeline
  - Current Status
- Future Plans

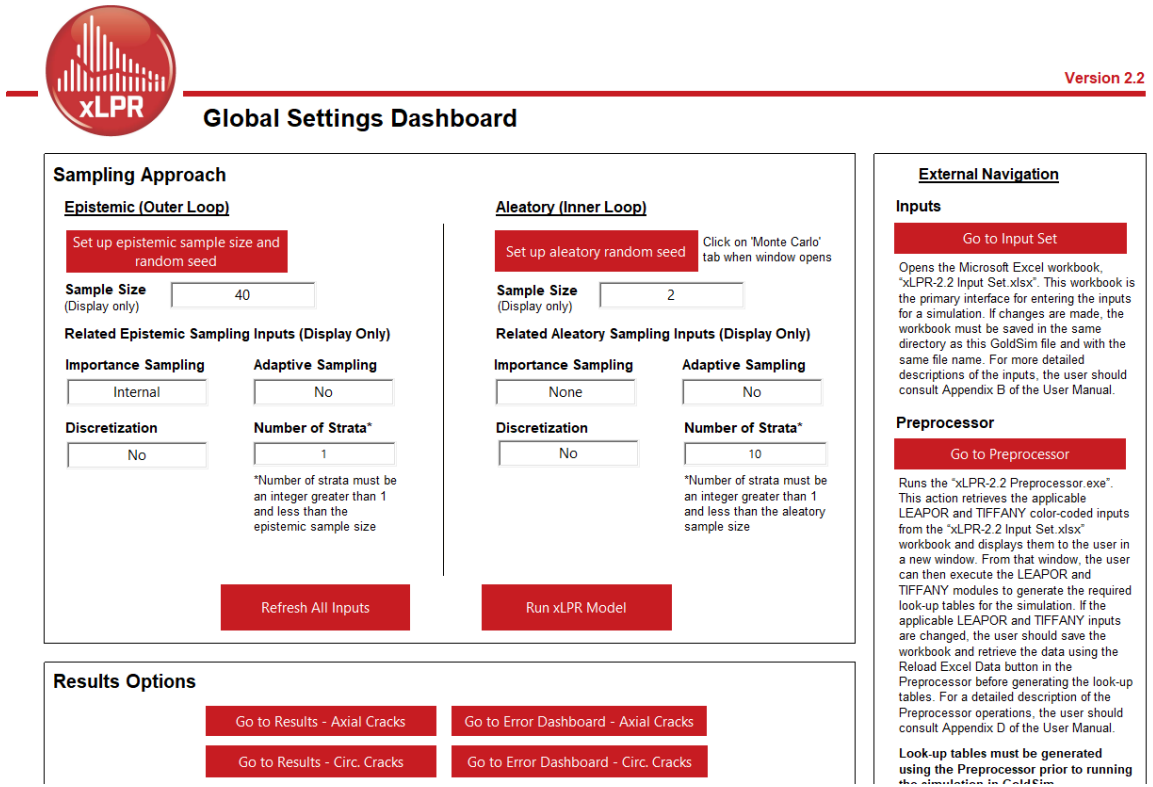
Extremely Low Probability of Rupture (xLPR) Code Overview

Probabilistic fracture mechanics (PFM) software tool for nuclear power plant piping integrity risk analysis

> Joint effort by the NRC's Office of Nuclear Regulatory Research and the Electric Power Research Institute (EPRI), now in second major version (2.4) with a third major version being developed (v3.0)

> Capable of modeling the effects of stress-corrosion cracking (SCC) and thermal fatigue

> Used by NRC and EPRI staff and contractors to risk-inform industrywide emerging piping integrity issues via probabilistic approaches



U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

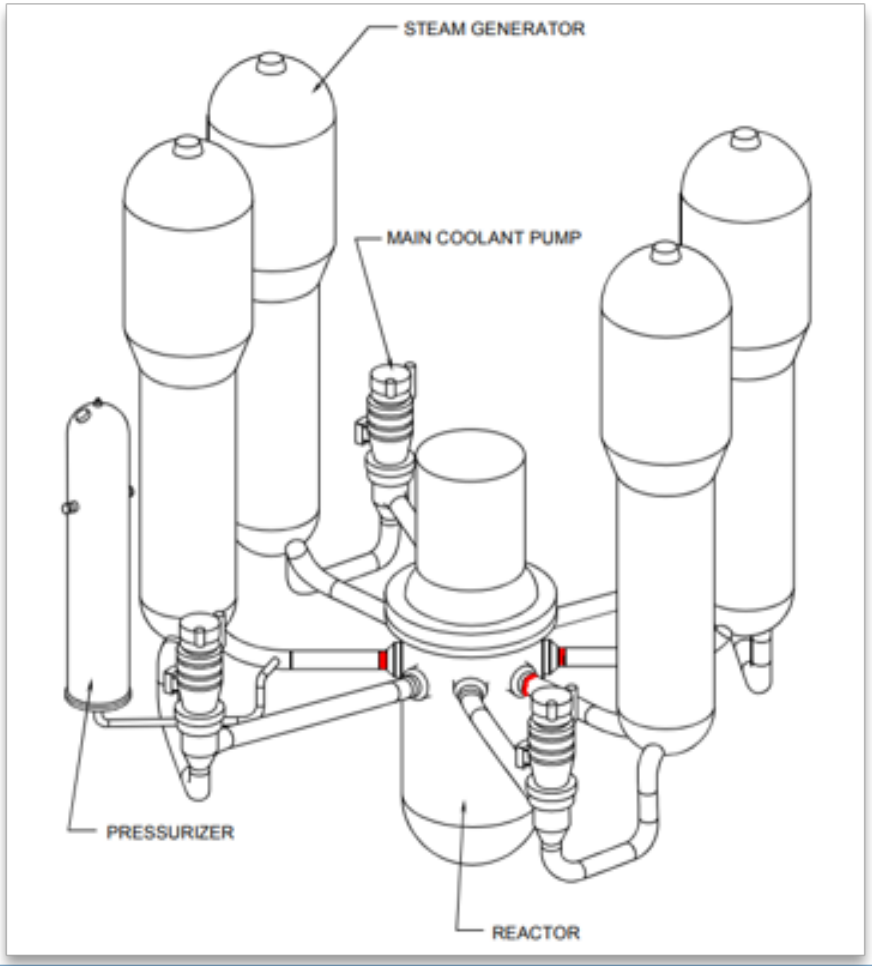
xLPR Use Cases

Latest NRC Use Cases

xLPR is used in support of NRR's LIC-504 analysis of the risk of the emergent French Stress Corrosion Cracking OE to the US PWR fleet (ML24162A131)

xLPR used to perform analyses in NRR review of Leak-before-Break (LBB) regulatory approach (ML21217A088 and ML22088A006)

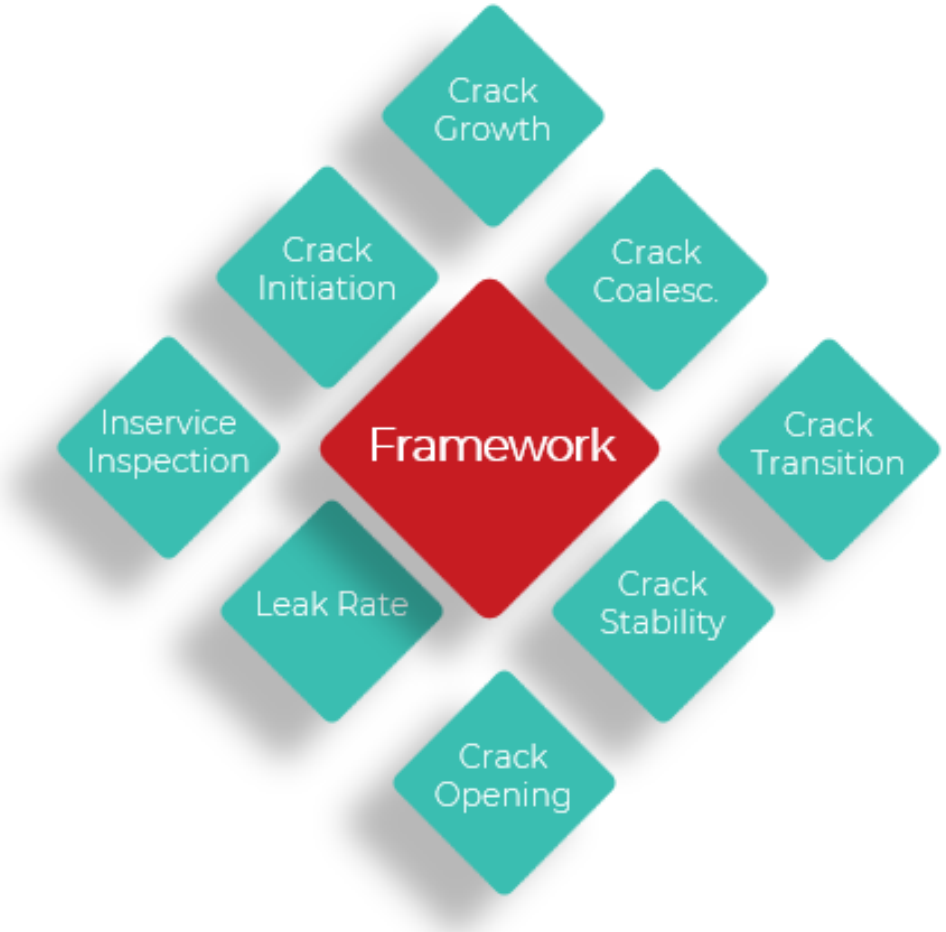
xLPR used to reassess LOCA frequencies initially calculated in NUREG-1829 (ML24193A122)



xLPR Modular Design

Framework

- Reads input parameters
- Performs sampling on probabilistic parameters
- Interfaces with FORTRAN modules containing physical models
- Collects module outputs



Modular Design

- Each FORTRAN module contains the physical models behind xLPR
  - Crack Initiation, Crack Growth, Inservice Inspection, etc
- Each module developed by technical experts
- Allows flexibility of future development

Known Framework Needs

Adaptability

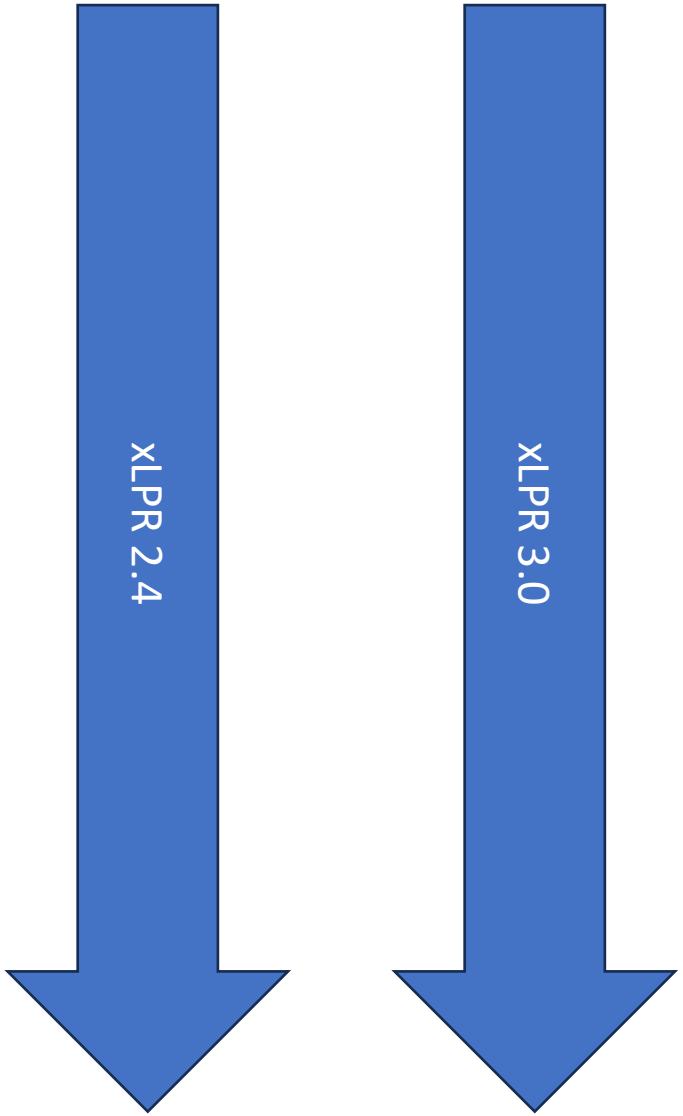
- Move away from requiring access to Commercial Off-the-Shelf (COTS) software
- Reduce the learning curve to making modifications
- Replace certain features that are proprietary and locked from users

Performance

- Currently there is limited parallelization capacity
  - 4 cores for free GoldSim version
  - 10 cores for subscription GoldSim
- Memory issues with large sample sizes
- 32-bit COTS application limits improvements to performance

xLPR 3.0 Project Timeline

- Framework replacement project started August 2024
  - Contracted with Information Systems Laboratory
- Concurrent with development of xLPR V2.4
- Framework replacement used xLPR 2.3 as a baseline for conversion

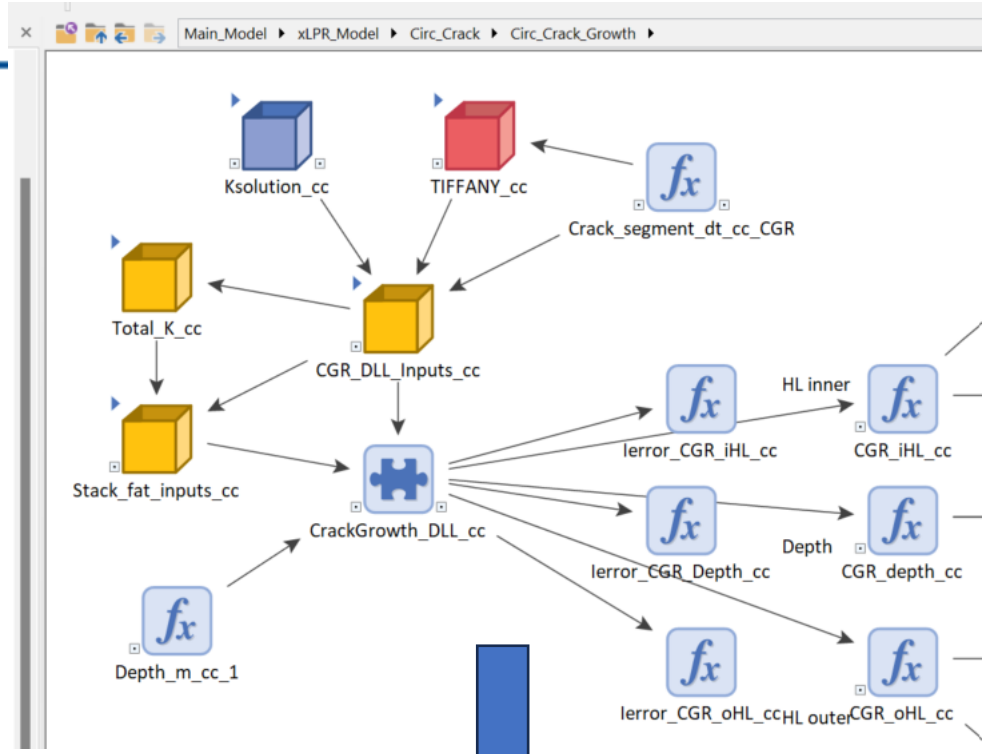


Language Selection

- Go has a syntax similar to C++ and Java and is simpler to learn
- Go eliminates aspects of C++ that are problematic and emphasizes features that improve maintainability
- The Go language is designed to limit dependencies that make code hard to modify
- Go was designed to simplify parallel computing and make threads more efficient
- Go compiles to an executable that incorporates Fortran routines and simplifies distribution

Framework Conversion

- 4000+ unique elements in the GoldSim framework need to be converted into Go code
- Create new interfaces between the Go Framework and the compiled FORTRAN DLLs
- Confirming consistency of outputs from DLLs compared to xLPR 2.3

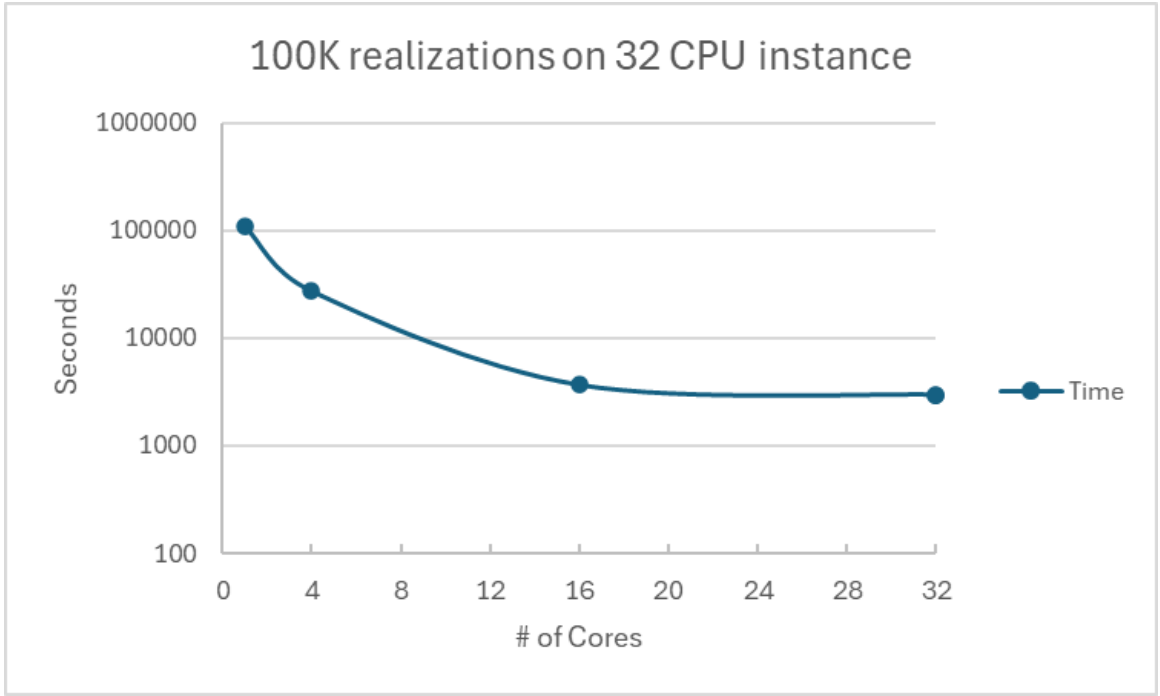



```
// GOLDSIM: CrackGrowth_DLL_cc; Pos 1309
//
// A [fc.FortranFnWrapper] that computes the crack growth of all initiated cracks.
type CrackGrowthCCcalc struct {
    name            string
    calculationType fc.FortranFnType
}

func (c CrackGrowthCCcalc) Name() string {
    return c.name
}

// Returns the [fc.FortranFnType] of this [fc.FortranFnWrapper]
func (c CrackGrowthCCcalc) FortranFnType() fc.FortranFnType {
    return c.calculationType
}

// Runs the growth subroutine from the CGR Fortran90 file.
func (c CrackGrowthCCcalc) DoStep(currentParams pc.ParamCollector) {
    status := C.int(0)
    outputs := make([]C.double, 211)
    inputArray := c.getInputs(currentParams)
    C.growth(&status, &inputArray[0], &outputs[0])
    c.storeOutputs(outputs, currentParams)
}
```

Computational Performance

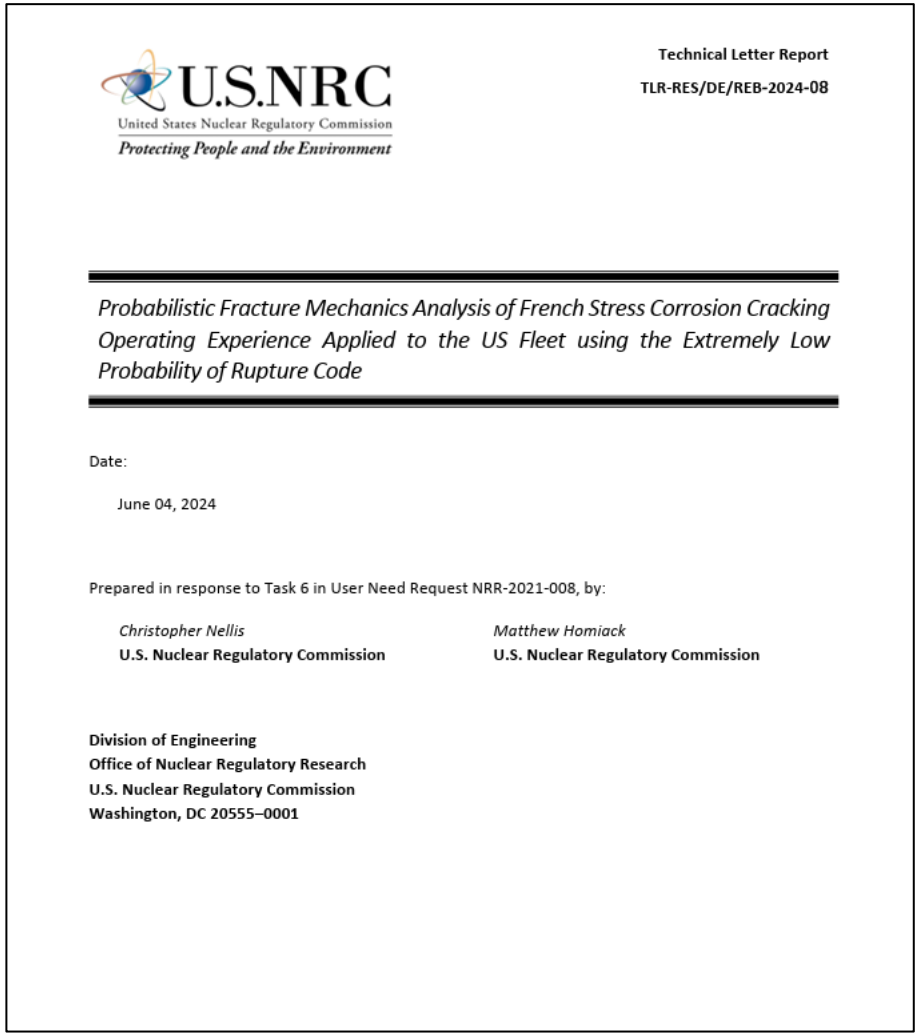
- xLPR 3.0 on single processor is ~2-3x faster than xLPR 2.3

- Parallelization does substantially reduce the time to completion.



Memory Usage Overview

- Took example simulation from prior NRC work on SCC in US PWR fleet
- xLPR v2.3: 100K realization simulation with initial flaw crack
- Memory issues required:
  - Reducing savepoints to only 4 20-year timepoints
  - Broken into ten 10K realization simulations
  - 8 hours each when utilizing GoldSim DPS



U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

Technical Letter Report
TLR-RES/DE/REB-2024-08

Probabilistic Fracture Mechanics Analysis of French Stress Corrosion Cracking Operating Experience Applied to the US Fleet using the Extremely Low Probability of Rupture Code

Date:

June 04, 2024

Prepared in response to Task 6 in User Need Request NRR-2021-008, by:

Christopher Nellis
U.S. Nuclear Regulatory Commission

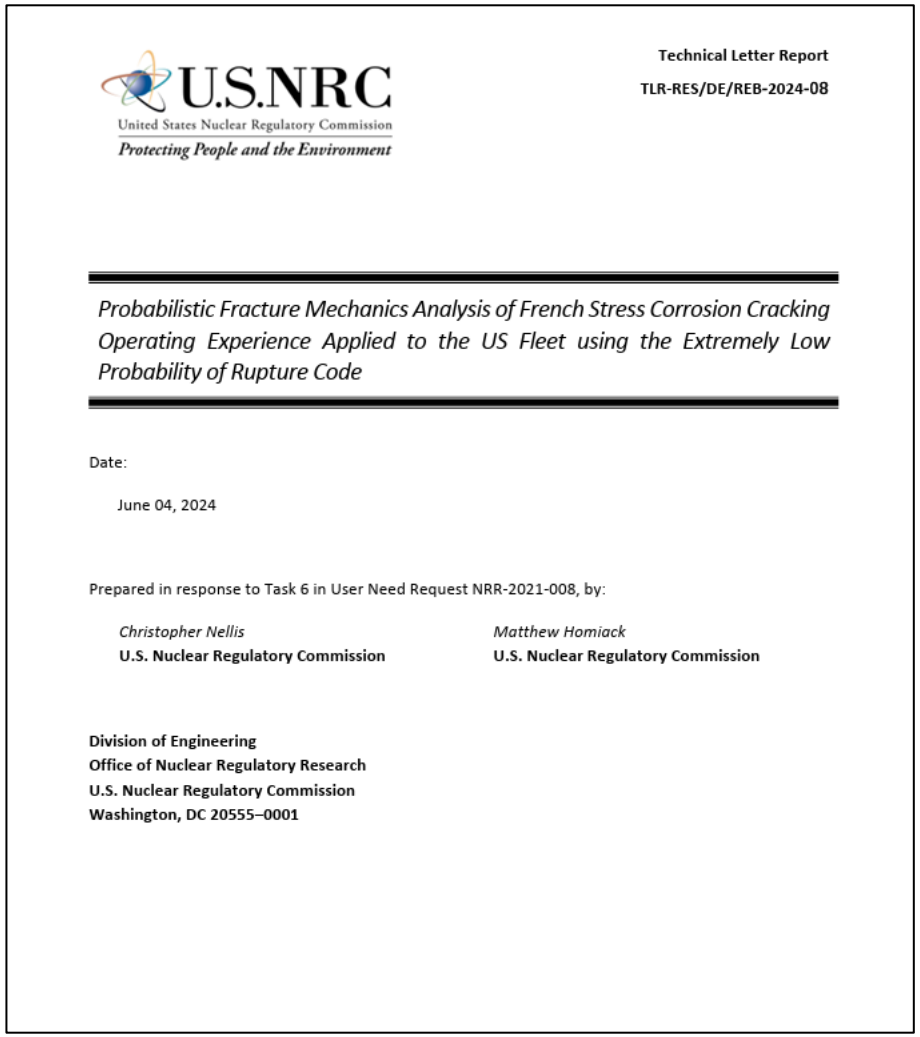
Matthew Homiack
U.S. Nuclear Regulatory Commission

Division of Engineering
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001

U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

Memory Usage Overview

- Completed in <8 hours in xLPR 3.0 with 16 threads
- Full linear time histories of important outputs
- 13 GB of output data in xLPR 3.0
  - ~20GB of output data from GoldSim 2.3



U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

Technical Letter Report
TLR-RES/DE/REB-2024-08

Probabilistic Fracture Mechanics Analysis of French Stress Corrosion Cracking Operating Experience Applied to the US Fleet using the Extremely Low Probability of Rupture Code

Date:

June 04, 2024

Prepared in response to Task 6 in User Need Request NRR-2021-008, by:

Christopher Nellis
U.S. Nuclear Regulatory Commission

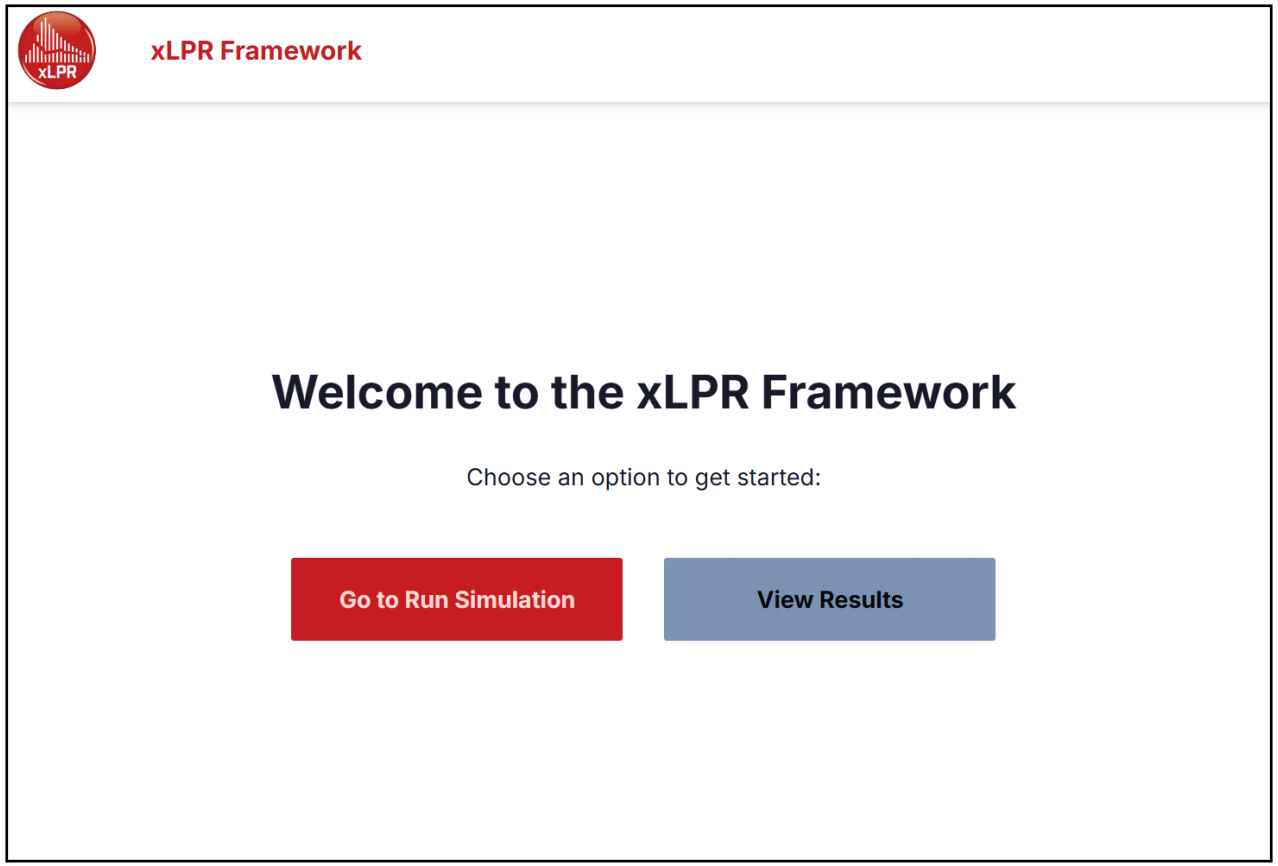
Matthew Homiack
U.S. Nuclear Regulatory Commission

Division of Engineering
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001

U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

v3.0 vs v2.0 Capabilities

- Faster operation in serial
- Much easier parallelization
  - More cores available
- Ability to collect sampled values from realizations with ruptures
- Ability to run a specific realization
- Improved memory performance
- In-progress results dashboard

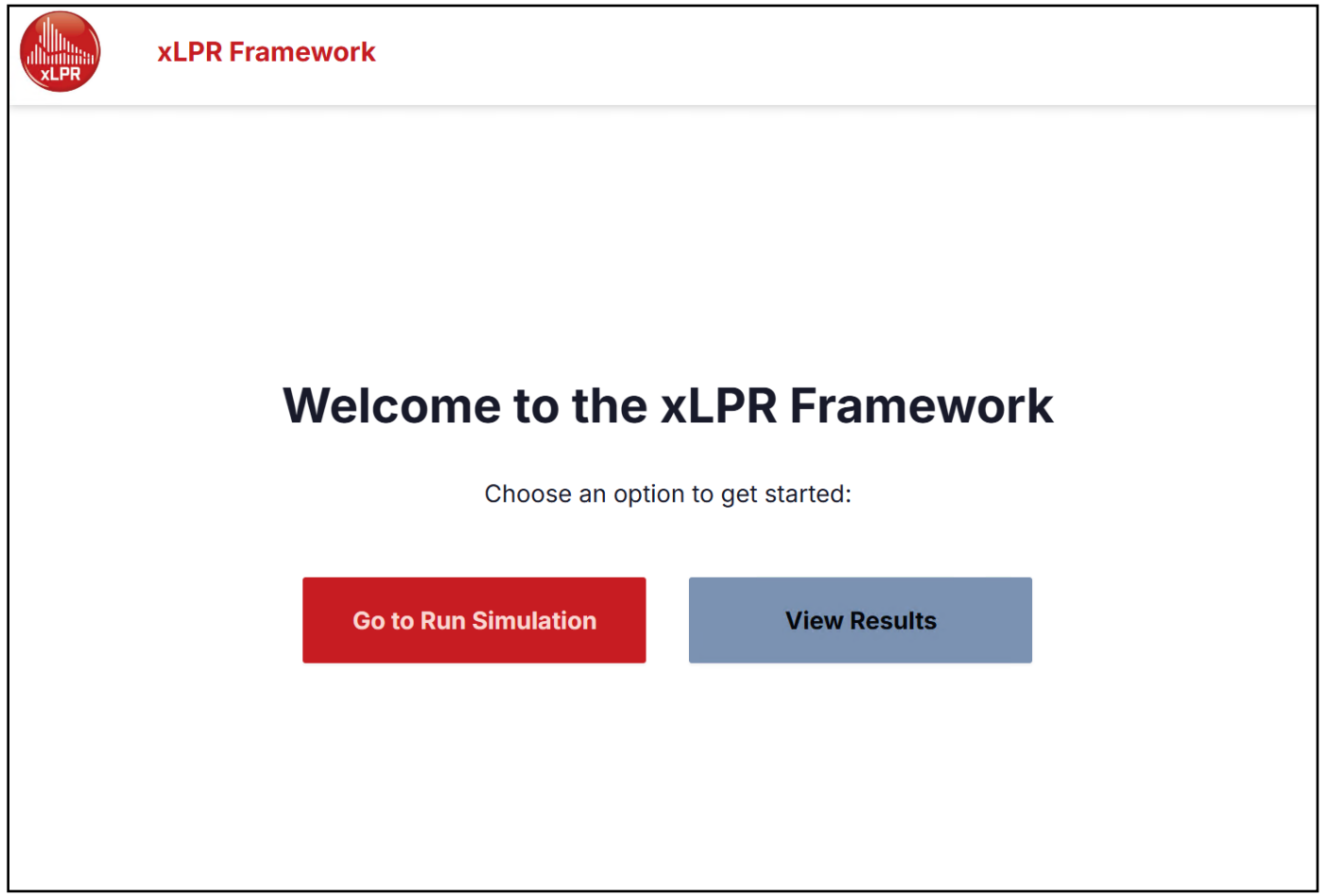


Summary

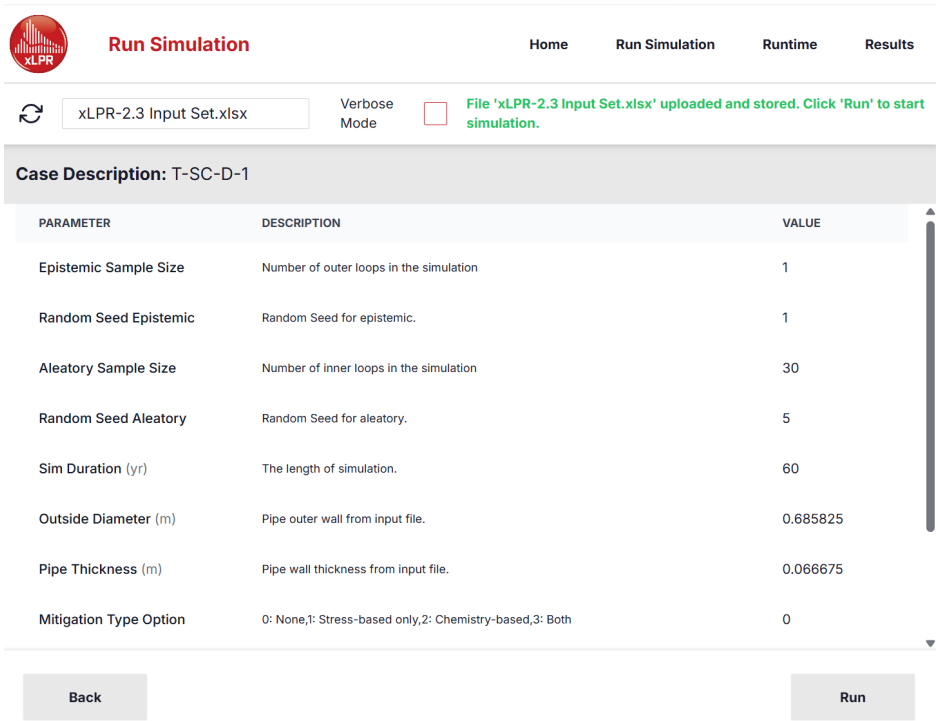
xLPR 3.0 anticipated for release in early Fall 2026

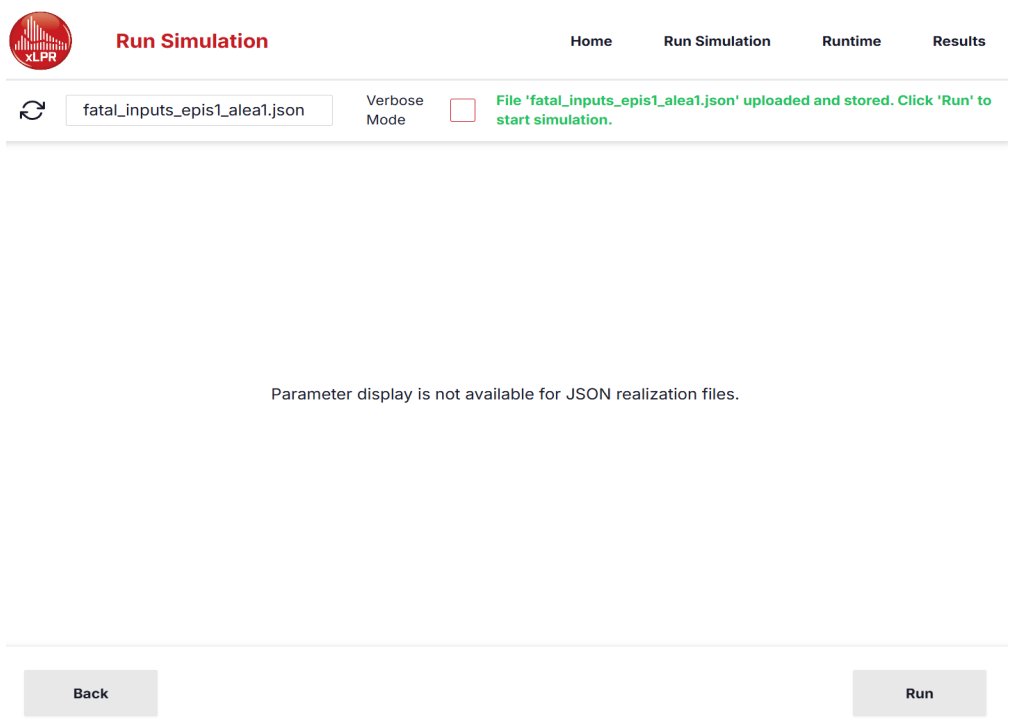
xlpr@nrc.gov or xlpr@epri.com for further information

Demonstration



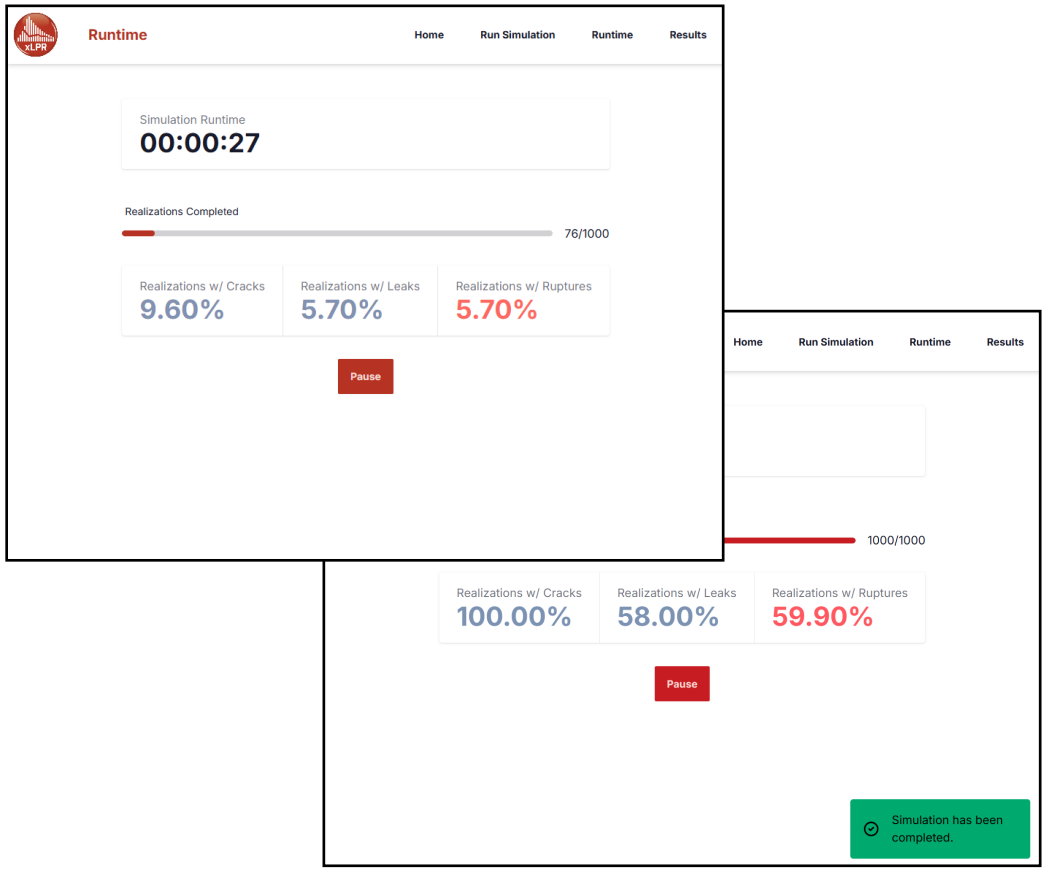
Graphical User Interface





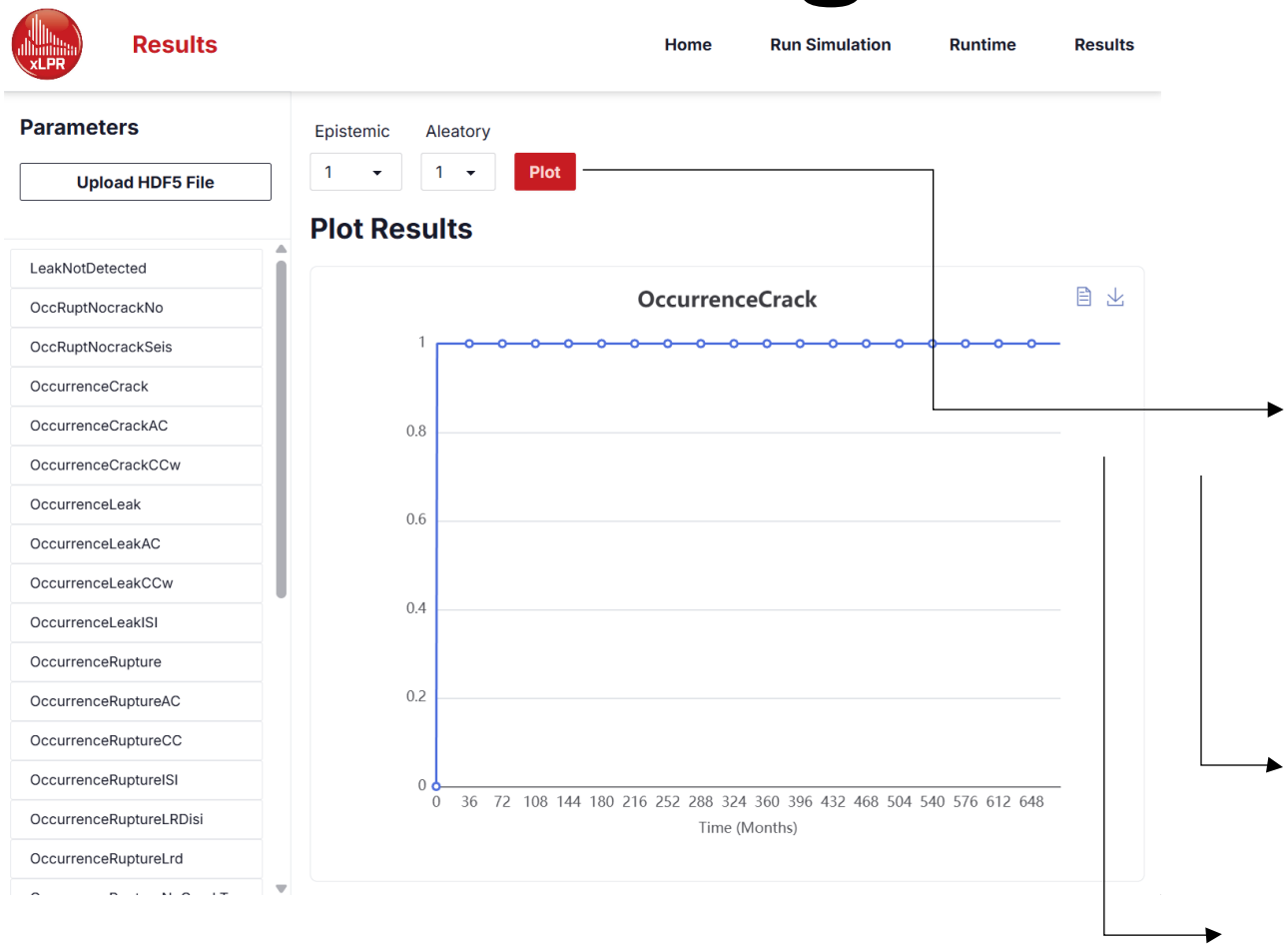
- The user can select to upload either an Excel input file or a JSON realization file.
- JSON realization files that are created for realizations that fail from previous simulation runs (found in the ‘fatal’ folder).
  - Parameter display is not available for these file types since only one selected realization will run
- Case description and selected parameter details from an input sheet will be shown once the file is uploaded. The user can review this information before choosing to run the simulation

Runtime View



- Once the user selects to run the simulation, they will navigate to the Runtime page
- This page tracks real time data from the simulation:
  - Total simulation runtime
  - Number of realizations completed
  - Total percentage of realizations with cracks, leaks, and ruptures
- Pause button is used to pause or resume the simulation
- User can navigate to other areas of the application while simulation is running and return to runtime page to view statistics.
  - The simulation completed notification will display on all pages when the simulation has finished.

Results Page



- Once the simulation has been completed, a HDF5-formatted results file will be loaded on the results page for plotting.
  - This will load the available parameters from the HDF5 file into a list for selection.
  - Once the user selects a parameter from the list, the plot will be displayed in the plot results area
- A specific realization is selected using the selection list at the top of the Plot Results area.
- The plot that is being displayed can be downloaded in PNG format using the download button in the top right corner
- A table view of the plot can also be selected in the same area