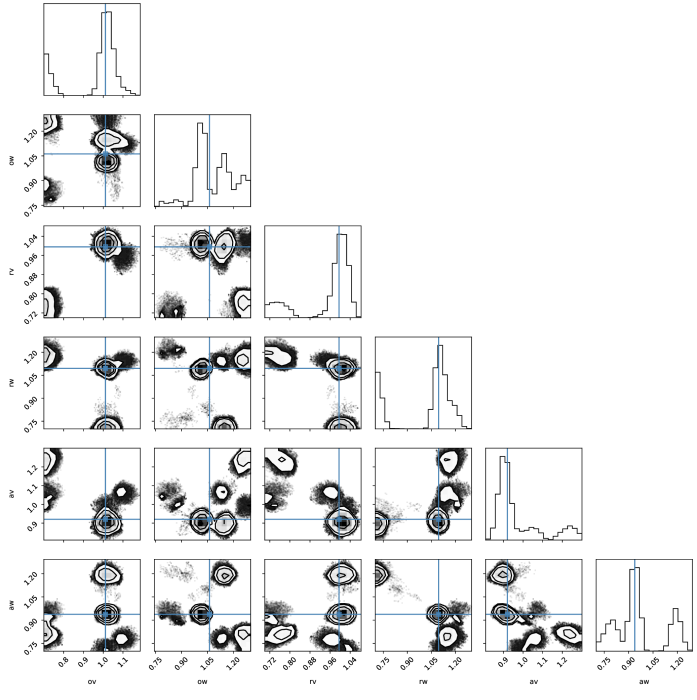
A CASE STUDY OF BAYESIAN METHODS FOR THE EVALUATION OF CROSS SECTIONS: ^{239}PU NEUTRON INDUCED REACTIONS



Los Alamos NATIONAL LABORATORY EST. 1943

LA-UR-24-33104

MATTHEW MUMPOWER

BAND Workshop

Wednesday December 18th 2024

Theoretical Division

NUCLEAR DATA IS UBIQUITOUS IN MODERN APPLICATIONS

Example: fission yields are needed for a variety of applications and cutting-edge science

Industrial applications: simulation of reactors, fuel cycles, waste management

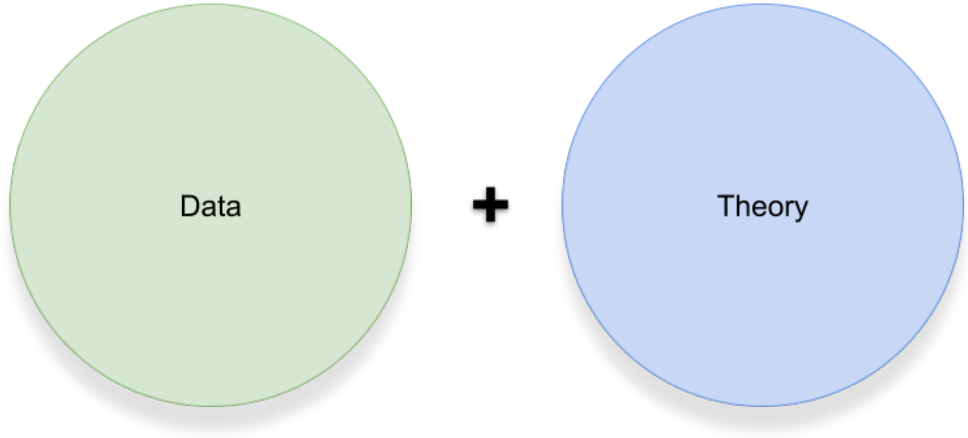
Experiments: backgrounds, isotope production with radioactive ion beams (fragmentation)

Science applications: nucleosynthesis, light curve observations

Other Applications: national security, nonproliferation, nuclear forensics

Figure: Chi-Nu detector (left), Reactor cooling towers (middle), Neutron star merger (right)

MODERN APPLICATIONS



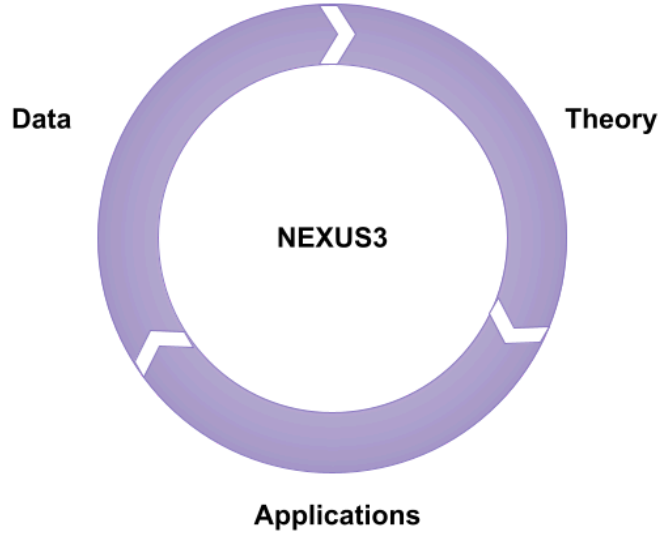
Modern applications require: a blending of data and theory (models)

Data is often times sufficiently precise, but hard to obtain and there's never enough

Theory provides great physical insight and extrapolations, but limited by assumptions and lack of precision

Evaluations: how do we go about merging information produced by both?

COMPUTATIONAL NUCLEAR DATA: THE NEXUS FRAMEWORK



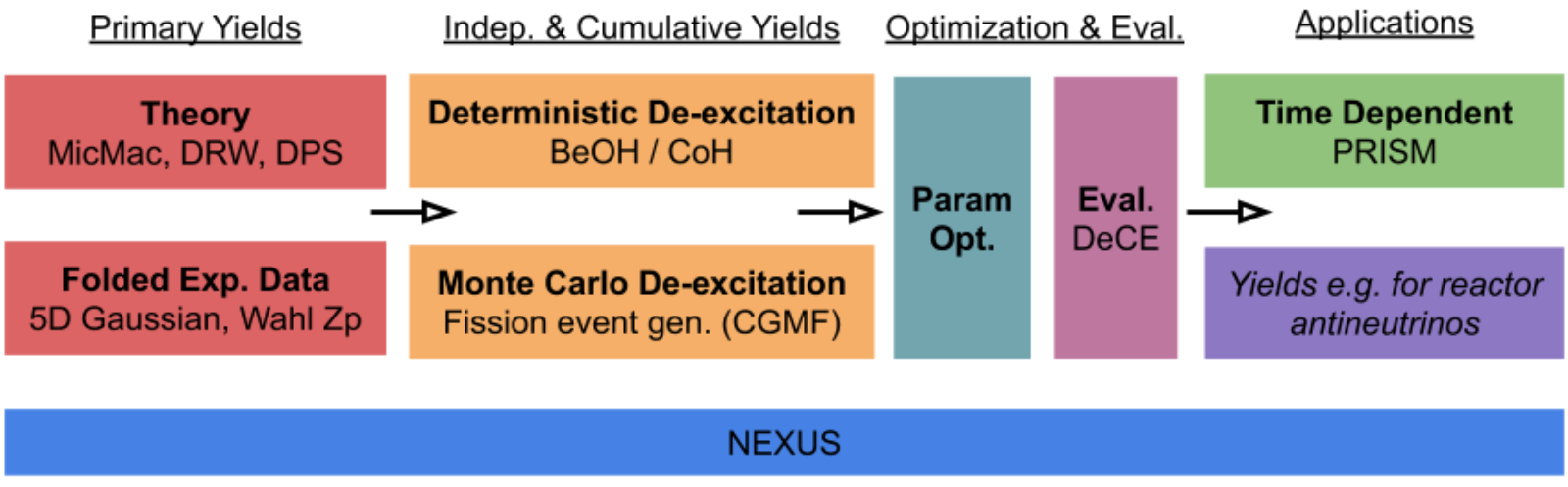
We have built a Python3 platform to handle this complex task which provides:

Access to nuclear data (experiment, evaluation, etc.) and metadata

Access to theoretical models (CoH, BeOH, eFRLDM, DRW, etc.)

Seamless integration between data, codes and applications (marshalling); straightforward data release

EXAMPLE EVALUATION WORKFLOW: FISSION YIELDS



Our current workflow combines many distinct codes and data

NEXUS provides

Code structures and marshalling that allow theory, data and applications to seemlessly communicate

A LITTLE ABOUT CROSS SECTION EVALUATIONS

In the `fast' ($E_n \gtrsim 0.1$ MeV) energy region the statistical Hauser-Feshbach (HF) theory may be used

HF codes consist of a patchwork of models, and associated parameters

Evaluations tune 50+ model parameters

Optical Model	Nuclear levels	Fission	Other
Potential depths	Discrete levels	Barrier heights and curvatures (inner / outer)	Gamma strength functions; for each Lorentzian (E,W,S)
Radius	Nuclear level density	Transmission coefficients	Collective excitations
Diffuseness	States in the continuum	Class I and Class II states	
Deformation	State density		

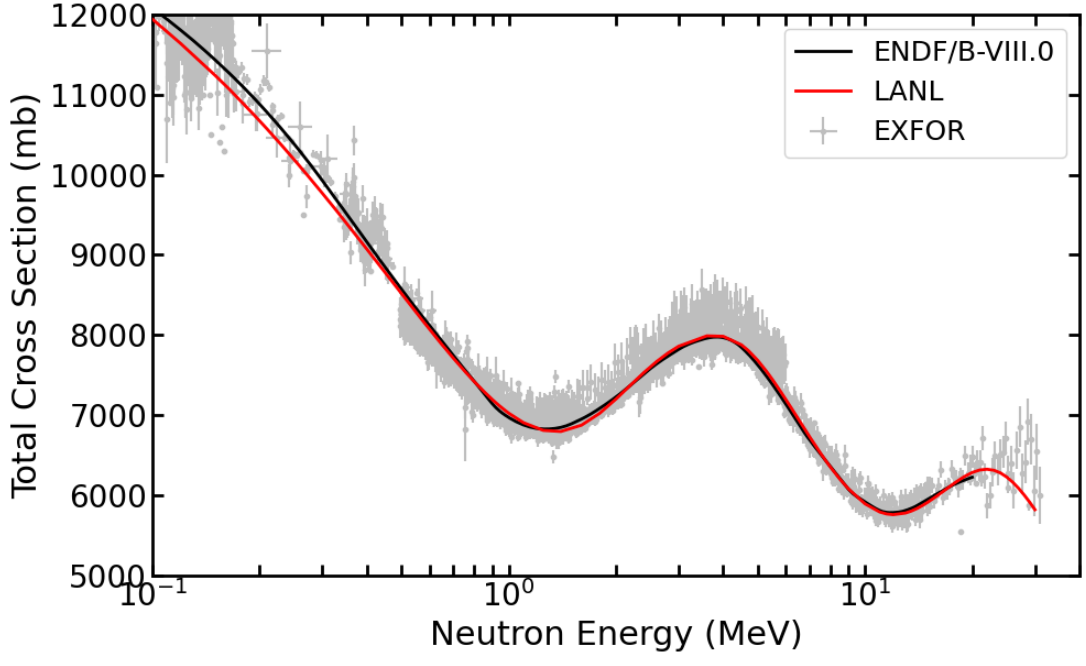
At LANL we use the publicly available 'CoH' code

Because of patchwork nature, order of fitting model parameters matters...

T. Kawano EPJA 57 16 (2021) • https://github.com/toshihikokawano/coh3

CROSS SECTION EVALUATION

We have more recently used NEXUS for the evaluation of ^{239}Pu neutron-induced cross sections (fast energy range)



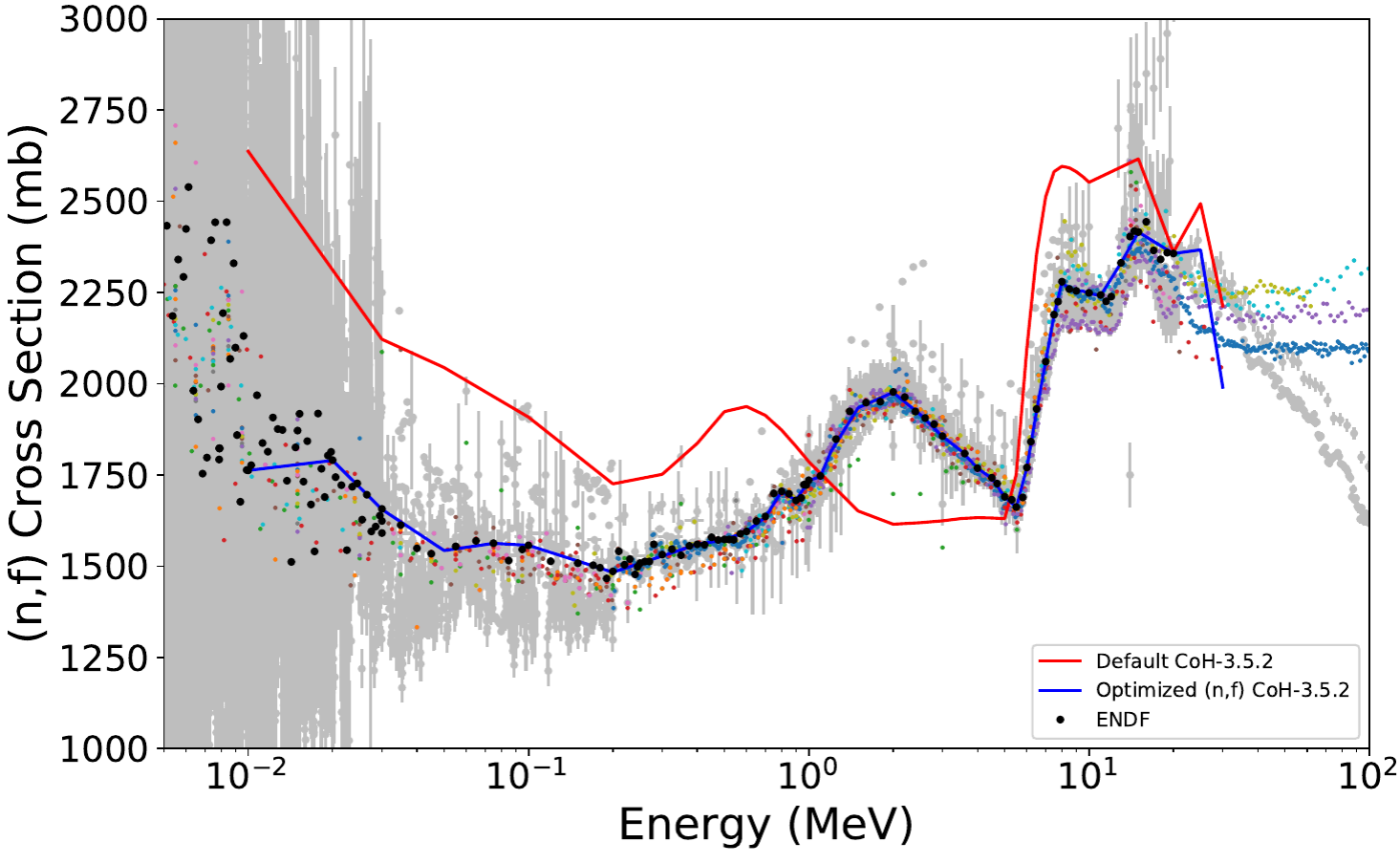
Total neutron-induced cross section

Modeled with CoH (Kawano) using 15 experimental datasets

Soukhovitskii (2005) optical model [$\beta \sim 0.21$]; coupled channels with 7 levels

LANL report [**Mumpower** *et al.* LA-UR-22-24121]

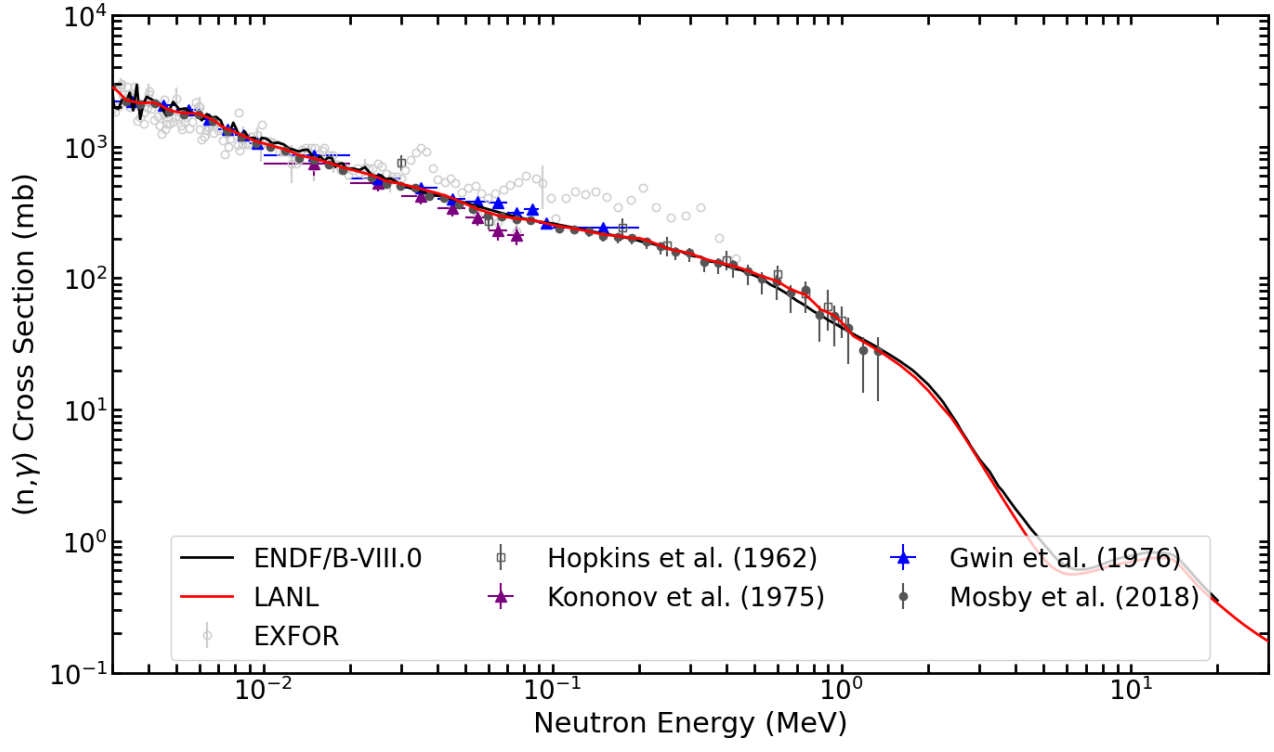
^{239}Pu (n,f) CROSS SECTION



Default output (red line) after fit becomes blue line

LANL report [**Mumpower** *et al.* LA-UR-22-24121]

^{239}Pu (n,γ) CROSS SECTION

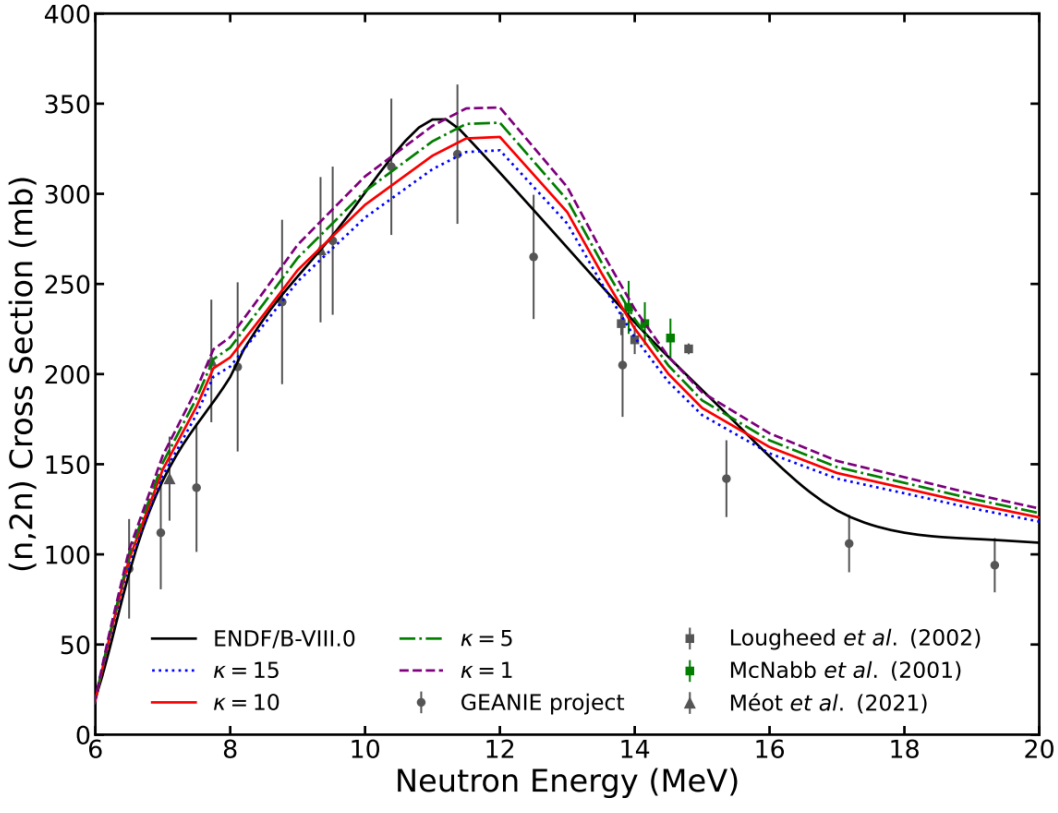


Rapid incorporation of new high-precision data, e.g. Mosby *et al.* (2018)

As well as new model developments (M1 enhancement)

We can also integrate feedback from integral benchmarks when appropriate

^{239}Pu (n,2n) CROSS SECTION

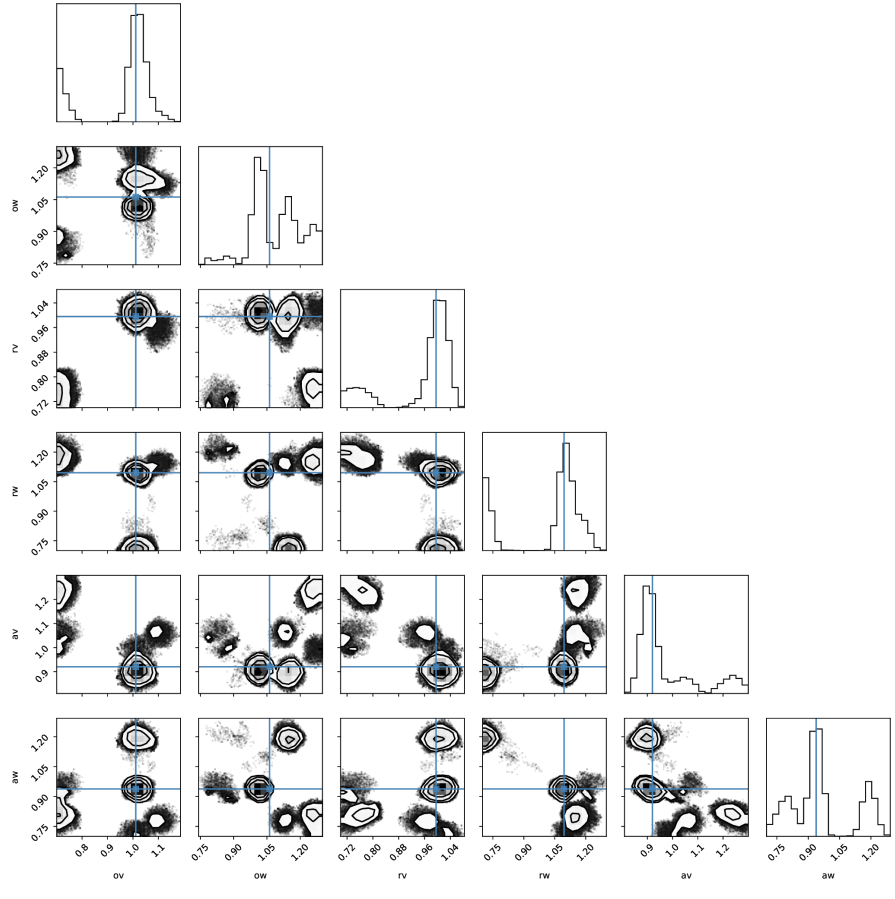


Model: new collective enhancement allows for simultaneous description of (n,f) and (n,2n) channels [difficult to describe in past versions of CoH]

Tight integration of model and experimental data leads to the updates relative to ENDF/B-VIII.0

Mumpower *et al.* PRC 107 034606 (2024)

BAYESIAN OPTIMIZATION

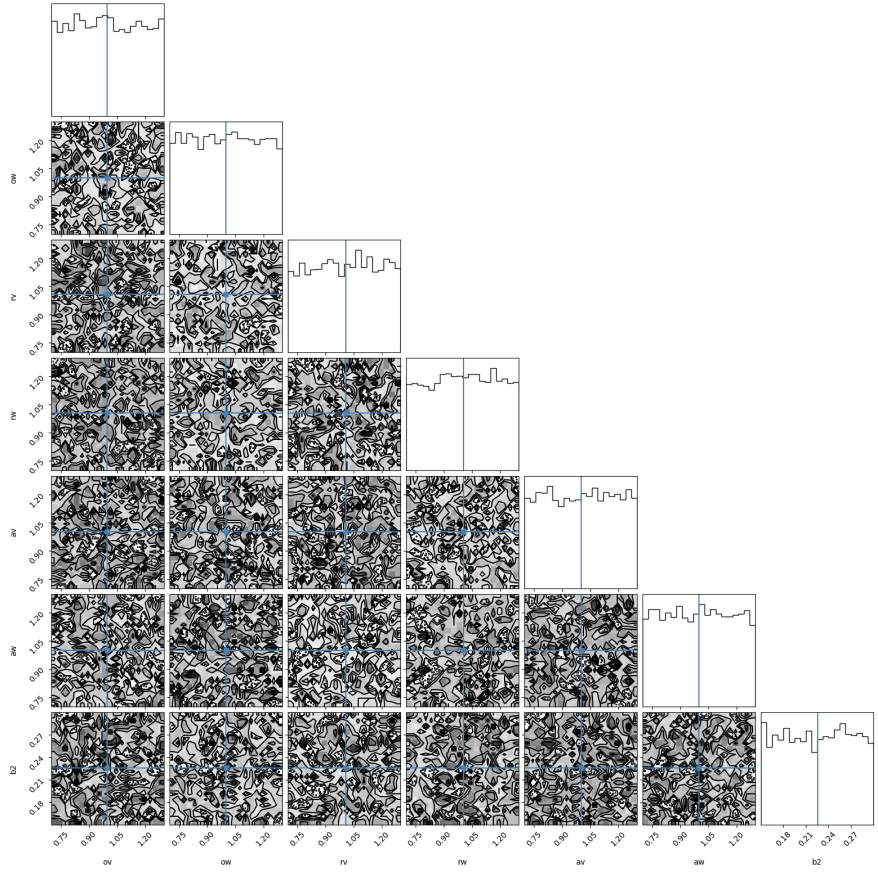


Bayesian fit of optical model parameters (potential depths, radii and diffuseness)

We find excellent performance of algorithm on par with past ENDF evaluation efforts

LANL report [Mumpower et al. LA-UR-22-24121]

FAILED OPTIMIZATION



Attempting to fit model parameters out of order can cause huge headaches...

LANL report [Mumpower et al. LA-UR-22-24121]

BAYESIAN HYPOTHESIS TESTING

New Bayesian approach developed - change some HF parameters at the same time

Allows the support of hypothesis testing and weighting of individual datasets

Equal weighting:

$$\ln[P(\boldsymbol{D}|\boldsymbol{\theta}, H_A)] = -\frac{1}{2}\sum_{k=1}^{N} n_k \ln(2\pi) + \ln(|\boldsymbol{V}_k|) + \chi_k^2 \ .$$

Individual weighting:

$$\ln[P(\boldsymbol{D}|\boldsymbol{\theta}, H_B)] = \sum_{k=1}^{N} \ln[\frac{2\Gamma(n_k/2+1)}{\pi^{n_k/2}}] - \ln(|\boldsymbol{V}_k|) - (n_k/2+1)\ln(\chi_k^2+2) \ .$$

If there is sufficient data individual weighting not necessary; this is the case with most reaction channels

However, some very precise datasets may prefer individual weighting

This is the so called hyperparameter optimization method

Hobson *et al.* MNRAS 335 2 (2002)

DATASET WEIGHTING

The dataset weighting can be approximated at the optimal parameter fit ($\widehat{\theta}$) of the cross section using

$$\alpha(\widehat{\theta}) \approx \frac{n_k}{\chi_k^2}$$

In the case of (n,2n) for ^{239}Pu, we have:

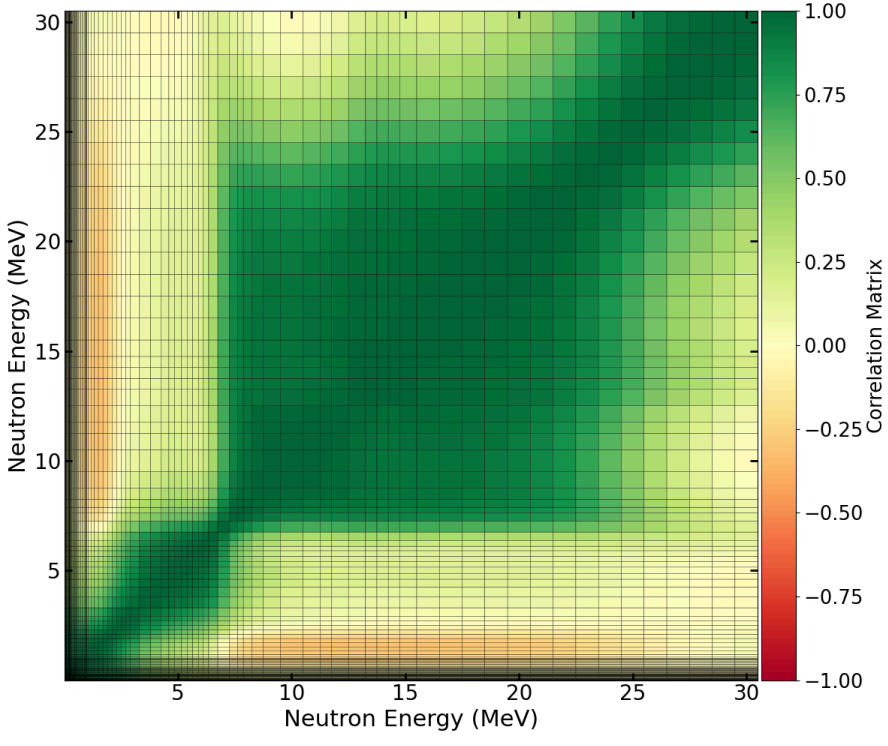
1. $\alpha_{\text{Bernstein}} \sim 0.499$
2. $\alpha_{\text{Meot}} \sim 0.531$
3. $\alpha_{\text{Lougheed}} \sim 0.595$

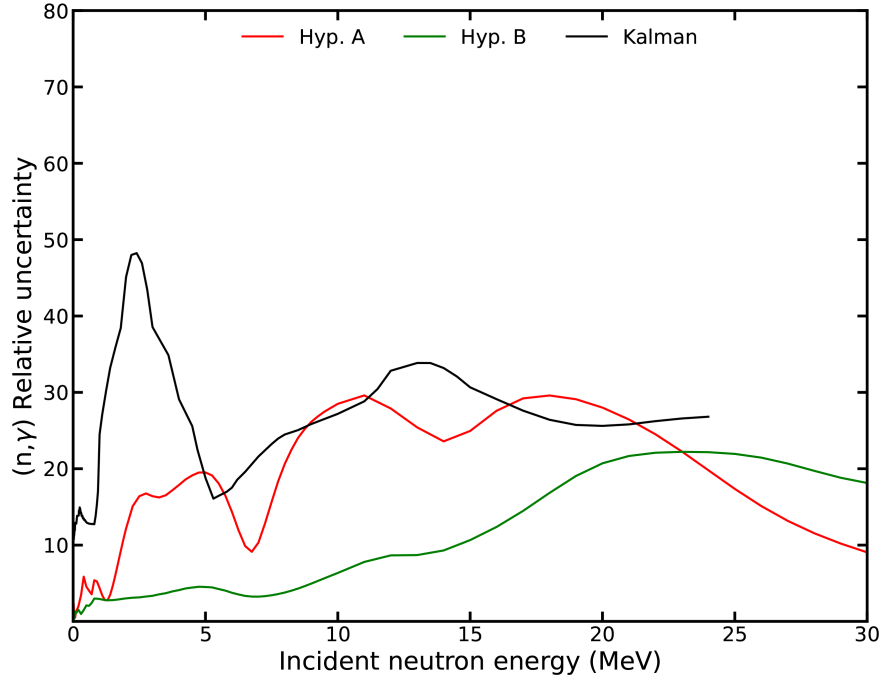
We can order the datasets in terms of their value to the fit

In the case of (n,2n) Lougheed is most influential, followed by Meot and Bernstein

Mumpower *et al.* in prep. (2025)

TOWARDS SELF-CONSISTENT COVARIANCES





Correlation matrix for the (n,γ) cross section after data fit

In contrast to the KALMAN filter (one parameter change at a time)

Covariances are more self-consistent; sample = all parameters changing

Deviations (non-linear dependencies) are found between KALMAN and this approach (right panel)

Mumpower *et al.* in prep. (2025)

BAND COLLABORATION INSIGHT

We want to employ this technology for many nuclei - not just one!

How can we achieve this goal?

Current algorithm is computationally expensive requiring HPC usage (coded by myself - not optimal!)

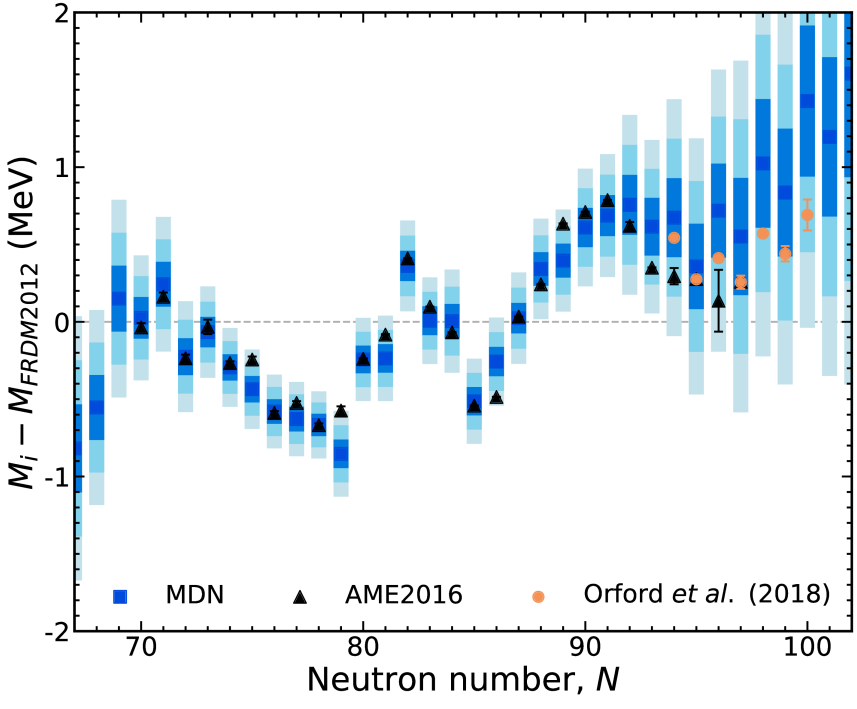
Are there better ways to evaluate the Likelihood in high dimensional spaces?

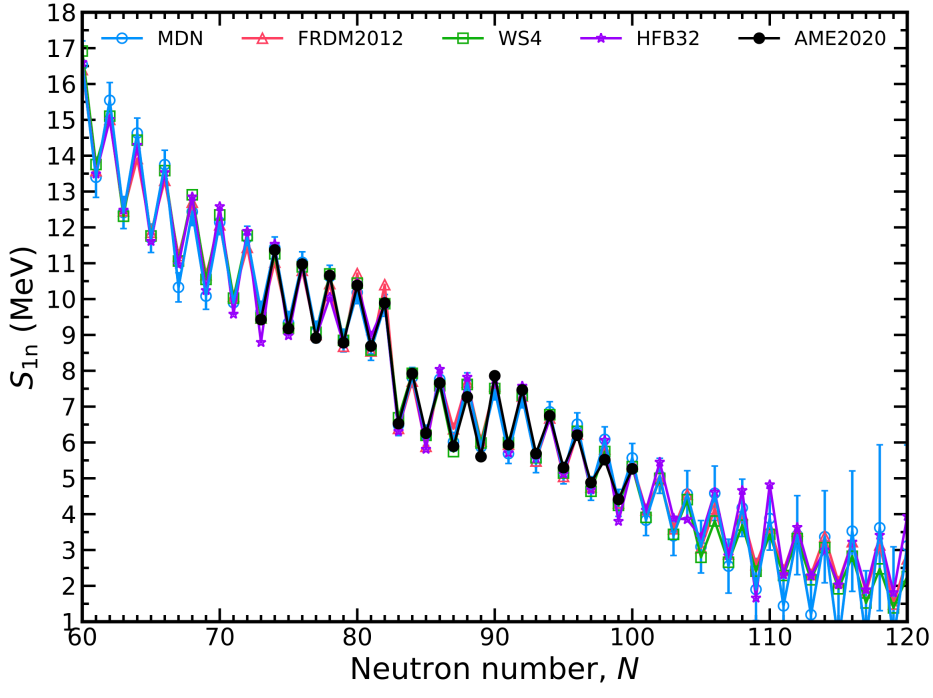
Perhaps some sort of emulator, or robust algorithm could be employed to aid both optimization and covariance creation

Ideas welcome!...

OVERLAP WITH OPEN SOURCE TOOLS (SOON!)

REDUCED ORDER MODELING OF ATOMIC NUCLEI (ROMAN)





ML methods to predict nuclear structure properties across chart of nuclides with quantified uncertainties

Example: Mixture Density Network (MDN) constrained ground state masses by both data and physics

Extrapolation is possible, but uncertainties overwhelm away from measured data

Lovell ***et al.*** PRC 106 014305 (2022) • **Mumpower** *et al.* PRCL 106 021301 (2022) • **Mumpower** *et al.* Fr. in Phys. 1198572 (2023)

RIPLPY: RIPL-4 PYTHON INTERFACE

```python
def in_sections(x: Nuclide | Element) -> list:
    """Check if the given Nuclide (or Element) is in various sections of the RIPL database. Returns a list of sections where True.
       This method may only be used once riplpy.load() has been issued.

       Input:
          x [Nuclide | Element]: a Nuclide or Element object that may be in the RIPL database

       Output:
          [list]: A list of section names (as a string) x is in; empty otherwise

       Examples:
          (1) Return the section names that the input is in::
             $ import riplpy
             $ riplpy.load() # This will take some time
             $ n = riplpy.Nuclide(z=46, a=119)
             $ riplpy.in_sections(n) # ['densities', 'gamma', 'levels', 'masses']
    """

    _sections = []
    # Loop over each section of RIPL
    for section in sections:
        # Check the databases in this section
        for db in eval(f"{section}.db.__all__"):
            if x in eval(f"{section}.db.{db}.data"):
                _sections.append(section)
                break
    return _sections
```

```
Nuclide(z=24, a=49): Entry(n=Nuclide(z=24, a=49), flag=2, Mexp=-45.33, Err=0.002, Mth=-46.29, Emic=1.4, beta2=0.216, beta3=0.0, beta4=0.067, beta6=0.002),
Nuclide(z=24, a=50): Entry(n=Nuclide(z=24, a=50), flag=2, Mexp=-50.26, Err=0.001, Mth=-50.6, Emic=1.76, beta2=0.0, beta3=-0.024, beta4=0.0, beta6=0.0),
Nuclide(z=24, a=51): Entry(n=Nuclide(z=24, a=51), flag=2, Mexp=-51.449, Err=0.001, Mth=-52.76, Emic=0.36, beta2=0.107, beta3=0.0, beta4=0.004, beta6=0.0),
Nuclide(z=24, a=52): Entry(n=Nuclide(z=24, a=52), flag=2, Mexp=-55.417, Err=0.001, Mth=-56.35, Emic=0.12, beta2=0.0, beta3=-0.023, beta4=0.0, beta6=0.0),
Nuclide(z=24, a=53): Entry(n=Nuclide(z=24, a=53), flag=2, Mexp=-55.285, Err=0.001, Mth=-55.99, Emic=0.07, beta2=0.0, beta3=0.0, beta4=0.0, beta6=0.0),
Nuclide(z=24, a=54): Entry(n=Nuclide(z=24, a=54), flag=2, Mexp=-56.933, Err=0.001, Mth=-57.47, Emic=0.74, beta2=0.18, beta3=0.0, beta4=0.045, beta6=0.0),
Nuclide(z=24, a=55): Entry(n=Nuclide(z=24, a=55), flag=2, Mexp=-55.107, Err=0.001, Mth=-55.16, Emic=1.57, beta2=0.189, beta3=0.0, beta4=0.029, beta6=0.001),
Nuclide(z=24, a=56): Entry(n=Nuclide(z=24, a=56), flag=2, Mexp=-55.281, Err=0.002, Mth=-55.8, Emic=1.98, beta2=0.189, beta3=0.0, beta4=0.022, beta6=0.001),
Nuclide(z=24, a=57): Entry(n=Nuclide(z=24, a=57), flag=2, Mexp=-52.524, Err=0.002, Mth=-52.64, Emic=2.67, beta2=0.199, beta3=0.0, beta4=-0.019, beta6=-0.001),
Nuclide(z=24, a=58): Entry(n=Nuclide(z=24, a=58), flag=2, Mexp=-51.835, Err=0.203, Mth=-52.48, Emic=2.9, beta2=0.199, beta3=0.0, beta4=-0.026, beta6=-0.002),
Nuclide(z=24, a=59): Entry(n=Nuclide(z=24, a=59), flag=2, Mexp=-47.891, Err=0.244, Mth=-48.68, Emic=3.34, beta2=0.172, beta3=0.0, beta4=-0.038, beta6=-0.002),
Nuclide(z=24, a=60): Entry(n=Nuclide(z=24, a=60), flag=2, Mexp=-46.504, Err=0.213, Mth=-47.91, Emic=3.25, beta2=0.181, beta3=0.0, beta4=-0.021, beta6=-0.001),
Nuclide(z=24, a=61): Entry(n=Nuclide(z=24, a=61), flag=2, Mexp=-42.181, Err=0.255, Mth=-42.7, Emic=4.28, beta2=0.32, beta3=0.0, beta4=0.044, beta6=0.032),
Nuclide(z=24, a=62): Entry(n=Nuclide(z=24, a=62), flag=2, Mexp=-40.415, Err=0.337, Mth=-41.18, Emic=4.11, beta2=0.329, beta3=0.0, beta4=0.047, beta6=0.018),
Nuclide(z=24, a=63): Entry(n=Nuclide(z=24, a=63), flag=1, Mexp=-35.527, Err=0.298, Mth=-36.03, Emic=4.33, beta2=0.329, beta3=0.0, beta4=0.048, beta6=0.014),
Nuclide(z=24, a=64): Entry(n=Nuclide(z=24, a=64), flag=1, Mexp=-33.152, Err=0.401, Mth=-34.95, Emic=2.93, beta2=0.018, beta3=0.0, beta4=0.0, beta6=0.001),
Nuclide(z=24, a=65): Entry(n=Nuclide(z=24, a=65), flag=1, Mexp=-27.796, Err=0.503, Mth=-28.74, Emic=3.51, beta2=0.053, beta3=0.0, beta4=0.017, beta6=0.012),
Nuclide(z=24, a=66): Entry(n=Nuclide(z=24, a=66), flag=1, Mexp=-24.796, Err=0.596, Mth=-26.33, Emic=2.73, beta2=0.053, beta3=0.0, beta4=0.009, beta6=0.001),
Nuclide(z=24, a=67): Entry(n=Nuclide(z=24, a=67), flag=1, Mexp=-19.049, Err=0.699, Mth=-20.5, Emic=2.29, beta2=0.108, beta3=0.0, beta4=0.053, beta6=0.02),
```

Robust Python API provides access to many datasets for nuclear structure / reaction calculations

SPECIAL THANKS TO

My collaborators at LANL

M. Chadwick, N. Gibson, M. Herman, T. Kawano, N. Kleedtke, A. E. Lovell, A. Mohan, D. Neudecker, H. Sasaki, I. Stetcu, and P. Talou

SUMMARY

Modern applications require a seamless integration of data and modeling

We're entering an era of computational nuclear data:

experimental data ▲ theoretical models ▲ nuclear evaluations

New evaluations can be constructed with robust Bayesian techniques (e.g. ^{239}Pu)

Open tools are essential for rapidly incorporated into applications to accelerate the pace of scientific discovery

This in turn leads to a tighter collaboration between the data, modeling and experimental communities

Results / Data / Papers @ MatthewMumpower.com