PyDynamic tutorials Documentation

Sascha Eichstädt, Björn Ludwig

Jul 25, 2023

GETTING STARTED

PyDynamic_tutorials is a collection of tutorials based on Jupyter notebooks which are designed to simplify the handling of PyDynamic. The tutorials range from introductory examples to more in-depth use cases in the context of our work at PTB.

For the *PyDynamic_tutorials* homepage go to GitHub.

PyDynamic_tutorials are written in Python 3.

CHAPTER

ONE

PYDYNAMIC TUTORIALS

We prepared a collection of tutorials and examples to document, explain and illustrate the possibilities offered by *PyDynamic*. We will add more and more examples over time, especially those that are currently included in *PyDynamic*'s codebase subfolders examples and tutorials.

1.1 Getting started

To get going with the tutorials you can either start directly in your browser or get a local copy and experiment offline on your machine.

1.1.1 Quick start in current browser session

To start working in the notebooks directly in the browser, click .

1.1.2 Get a local copy to work offline

To get started on your local machine, follow these simple steps:

1. Clone the repository, if you haven't already
2. Set up a virtual environment for PyDynamic_tutorials
3. Install the dependencies
4. Start the Jupyter Notebook server
5. Go to localhost:8888 with your favourite browser
6. Browse the various examples in the repository, alter and execute the code right in your browser

1. Clone the repository

```
$ git clone https://github.com/PTB-M4D/PyDynamic_tutorials.git
Cloning into 'PyDynamic_tutorials'...
[...]
Receiving objects: 100% (3/3), done.
$
```

2. Set up a virtual environment

On your command line/powershell execute:

```
$ python3 -m venv PyDynamic_tutorial_venv
$
```

This will create a subfolder *PyDynamic_tutorial_venv* and prepare a fully self-contained Python environment, which we can activate in the next step and install further Python packages without polluting or disturbing your system environment.

3. Install the dependencies

First we activate the previously created environment before we then install the required dependencies in two steps, because we are utilizing `pip-tools` to ensure you get a working copy of our environments.

```
$ source PyDynamic_tutorial_venv/bin/activate
(PyDynamic_tutorial_venv) $ pip install --upgrade pip pip-tools
Collecting pip
[...]
Successfully installed click-7.1.2 pip-20.1.1 pip-tools-5.2.0 setuptools-47.1.1 six-1.15.
↪0
$ pip-sync requirements/requirements.txt
Collecting attrs==19.3.0
[...]
Installing collected packages: attrs, [...]
webencodings-0.5.1
$
```

4. Start the notebook server

Now from the environment we created previously, start up the Jupyter Notebook server.

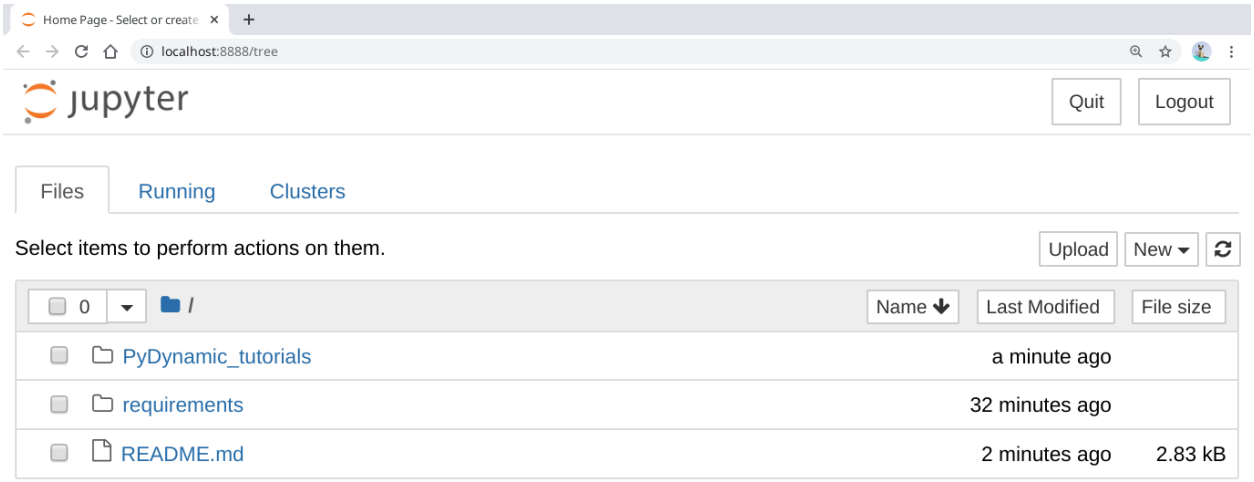
```
$ jupyter notebook
[I 13:01:24.790 NotebookApp] Serving notebooks from local directory: ~/code/PyDynamic_
↪tutorials
[I 13:01:24.790 NotebookApp] The Jupyter Notebook is running at:
[I 13:01:24.790 NotebookApp] http://localhost:8888/?
↪token=f368c552e042d48d46ff4c8a094448d0e7681790b0719215
```

5. Go to localhost:8888

Usually a browser window will have opened automatically at this point. Otherwise, just open one yourself and navigate to the printed URL in the console, in our case `http://localhost:8888/?token=f368c552e042d48d46ff4c8a094448d0e7681790b0719215`.

6. Browse the various examples

You should see something like the following:



After a click on *PyDynamic_tutorials* the source code can be edited and executed directly in the browser.

1.2 PyDynamic

The sourcecode of PyDynamic is available on GitHub. The detailed documentation of *PyDynamic*'s source code is available on pydynamic.readthedocs.io. The tutorial notebooks are all linked on these pagKes along with additional material.

CHAPTER

TWO

BASIC MEASUREMENT DATA PRE-PROCESSING

[1]:
```
%matplotlib inline
```

[2]:
```
from meas_data_preprocessing import *
```

2.1 Read data for a selected measurement scenario

[3]:
```
infos, measurement_data = read_data(meas_scenario=13)
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Downloading data from https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/
↪master/MeasuredSignals/pD-Mode%207%20MHz/pD7_MH44.DAT (30 kB)

file_sizes: 100%|| 30.5k/30.5k [00:00<00:00, 671kB/s]
Successfully downloaded file to ../datasets/pD7_MH44.DAT
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Downloading data from https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/
↪master/MeasuredSignals/pD-Mode%207%20MHz/pD7_MH44r.DAT (32 kB)

file_sizes: 100%|| 32.7k/32.7k [00:00<00:00, 637kB/s]
Successfully downloaded file to ../datasets/pD7_MH44r.DAT
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Downloading data from https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/
↪master/HydrophoneCalibrationData/MW_MH44ReIm.csv (167 kB)

file_sizes: 100%|| 171k/171k [00:00<00:00, 1.31MB/s]
Successfully downloaded file to ../datasets/MW_MH44ReIm.csv
The file ../datasets/pD7_MH44.DAT was read and it contains 2500 data points.
The time increment is 2e-09 s
```

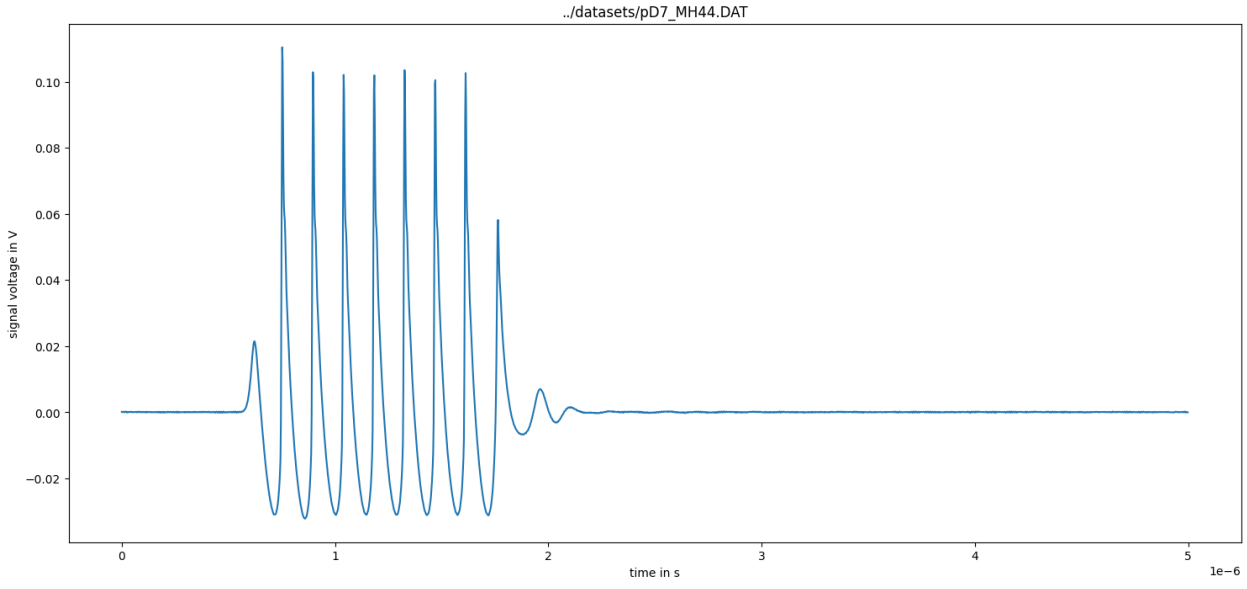
[4]:
```
# metadata for chosen measurement scenario
for key in infos.keys():
    print("%20s: %s" % (key, infos[key]))
```

```
                  i: 13
      hydrophonname: GAMPT MH44
    measurementtype: Pulse-Doppler-Mode 7 MHz
    measurementfile: ../datasets/pD7_MH44.DAT
          noisefile: ../datasets/pD7_MH44r.DAT
        hydfilename: ../datasets/MW_MH44ReIm.csv
```

[5]:
```
# available measurement data
for key in measurement_data.keys():
    print("%10s: %s" % (key, type(measurement_data[key])))
```

```
      name: <class 'str'>
   voltage: <class 'numpy.ndarray'>
      time: <class 'numpy.ndarray'>
```

[6]:
```
figure(figsize=(18, 8))
plot(measurement_data["time"], measurement_data["voltage"])
xlabel("time in s")
ylabel("signal voltage in V")
title(measurement_data["name"])
show()
```



2.2 Remove DC component

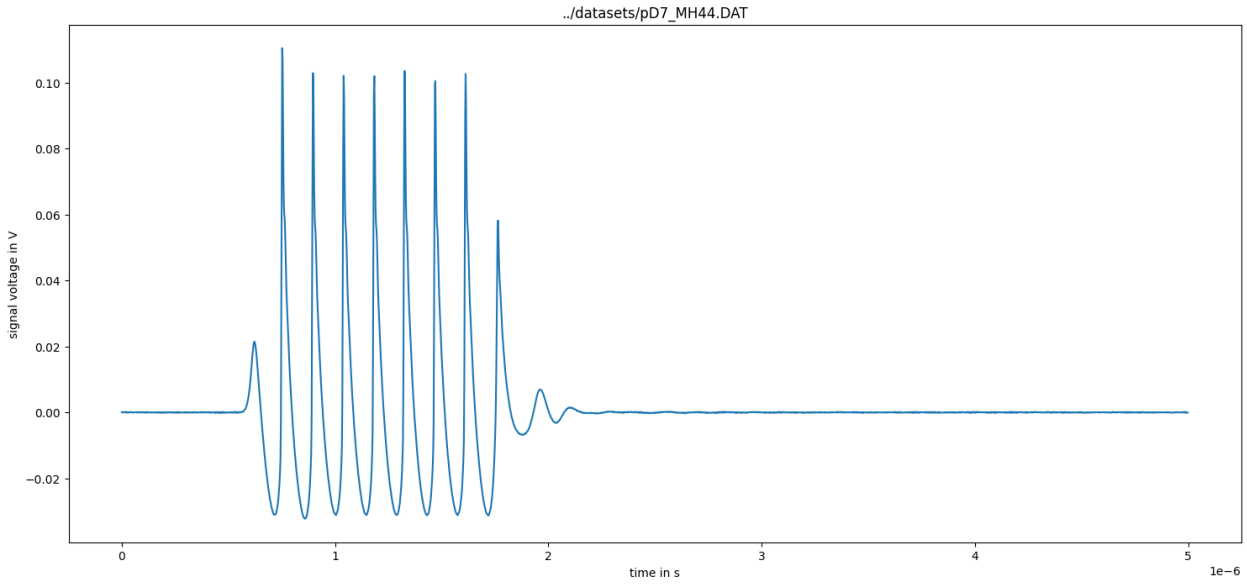
[7]:
```
measurement_data = remove_DC_component(measurement_data)
```

[8]:
```
figure(figsize=(18, 8))
plot(measurement_data["time"], measurement_data["voltage"])
xlabel("time in s")
ylabel("signal voltage in V")
```

(continues on next page)

(continued from previous page)

```
title(measurement_data["name"])
show()
```



2.3 Calculate measurement uncertainty from noise data

[9]:
```
measurement_data = uncertainty_from_noisefile(infos, measurement_data, do_plot=False)
```

```
The file "../datasets/pD7_MH44r.DAT" was read and it contains 2500 data points
```

[10]:
```
# available measurement data
for key in measurement_data.keys():
    print("%12s: %s" % (key, type(measurement_data[key])))
```

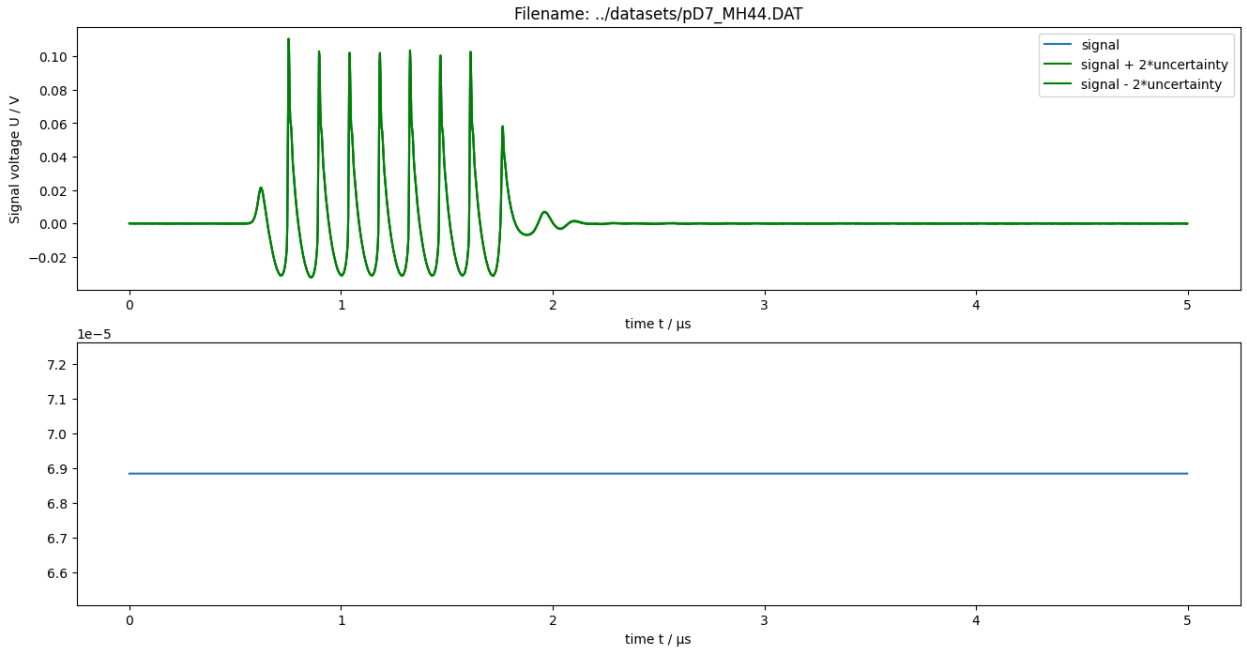
```
        name: <class 'str'>
     voltage: <class 'numpy.ndarray'>
        time: <class 'numpy.ndarray'>
 uncertainty: <class 'numpy.ndarray'>
```

[11]:
```
figure(figsize=(16, 8))
subplot(211)
plot(measurement_data["time"] / 1e-6, measurement_data["voltage"])
plot(
    measurement_data["time"] / 1e-6,
    measurement_data["voltage"] + 2 * measurement_data["uncertainty"],
    "g",
)
plot(
    measurement_data["time"] / 1e-6,
    measurement_data["voltage"] - 2 * measurement_data["uncertainty"],
    "g",
)
```

(continues on next page)

(continued from previous page)

```
legend(["signal", "signal + 2*uncertainty", "signal - 2*uncertainty"])
xlabel("time t / µs")
ylabel("Signal voltage U / V")
title("Filename: {}".format(measurement_data["name"]))

subplot(212)
plot(measurement_data["time"] / 1e-6, measurement_data["uncertainty"])
xlabel("time t / µs")
show()
```



2.4 Calculate spectrum of measured data

[12]:
```
measurement_data = calculate_spectrum(measurement_data, do_plot=False)
```

[13]:
```
# available measurement data
for key in measurement_data.keys():
    print("%12s: %s" % (key, type(measurement_data[key])))
```

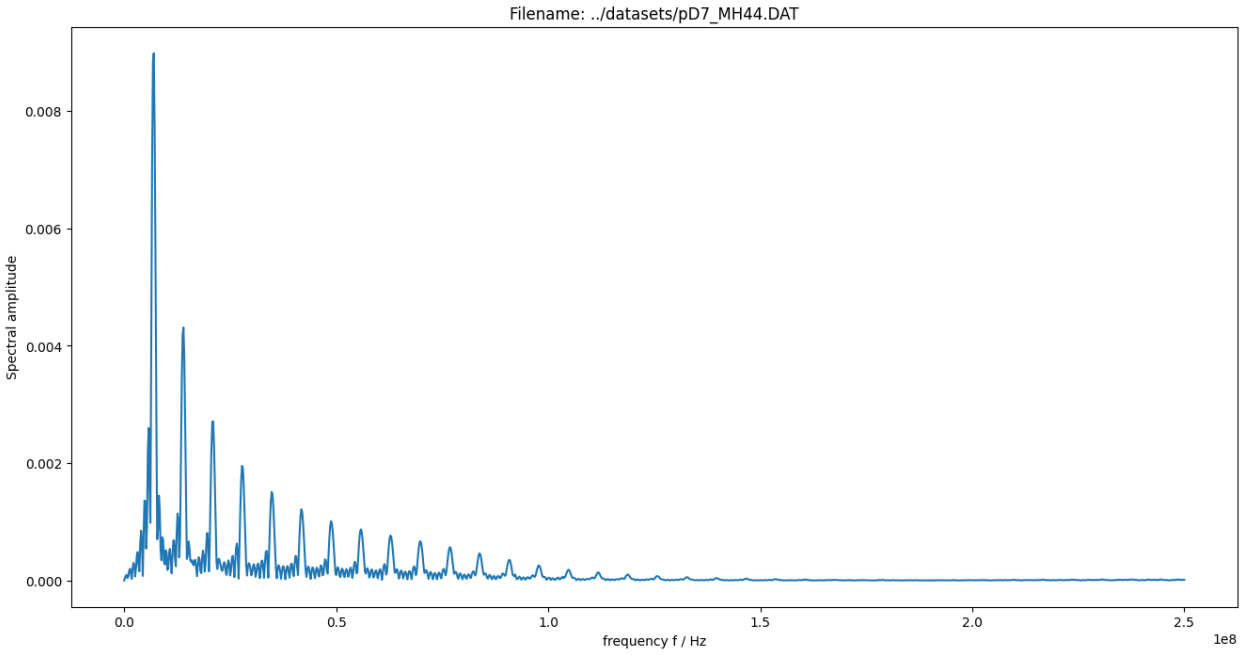
```
        name: <class 'str'>
     voltage: <class 'numpy.ndarray'>
        time: <class 'numpy.ndarray'>
 uncertainty: <class 'numpy.ndarray'>
   frequency: <class 'numpy.ndarray'>
    spectrum: <class 'numpy.ndarray'>
     varspec: <class 'numpy.ndarray'>
```

[14]:
```
figure(figsize=(16, 8))
plot(realpart(measurement_data["frequency"]), amplitude(measurement_data["spectrum"]))
```

(continues on next page)

(continued from previous page)

```
xlabel("frequency f / Hz")
ylabel("Spectral amplitude")
title("Filename: {}".format(measurement_data["name"]))
show()
```



CHAPTER

THREE

PREPARATION OF CALIBRATION DATA

[1]:
```
%matplotlib inline
```

[2]:
```
from meas_data_preprocessing import *
from hydrophone_data_preprocessing import *
```

3.1 Read calibration data for selected measurement scenario

[3]:
```
infos, hyd_data = read_calib_data(meas_scenario=13, do_plot=False)
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
```

[4]:
```
# metadata for chosen measurement scenario
for key in infos.keys():
    print("%20s: %s" % (key, infos[key]))
```

```
                   i: 13
      hydrophonname: GAMPT MH44
    measurementtype: Pulse-Doppler-Mode 7 MHz
    measurementfile: ../datasets/pD7_MH44.DAT
          noisefile: ../datasets/pD7_MH44r.DAT
        hydfilename: ../datasets/MW_MH44ReIm.csv
```

[5]:
```
# available measurement data
for key in hyd_data.keys():
    print("%10s: %s" % (key, type(hyd_data[key])))
```

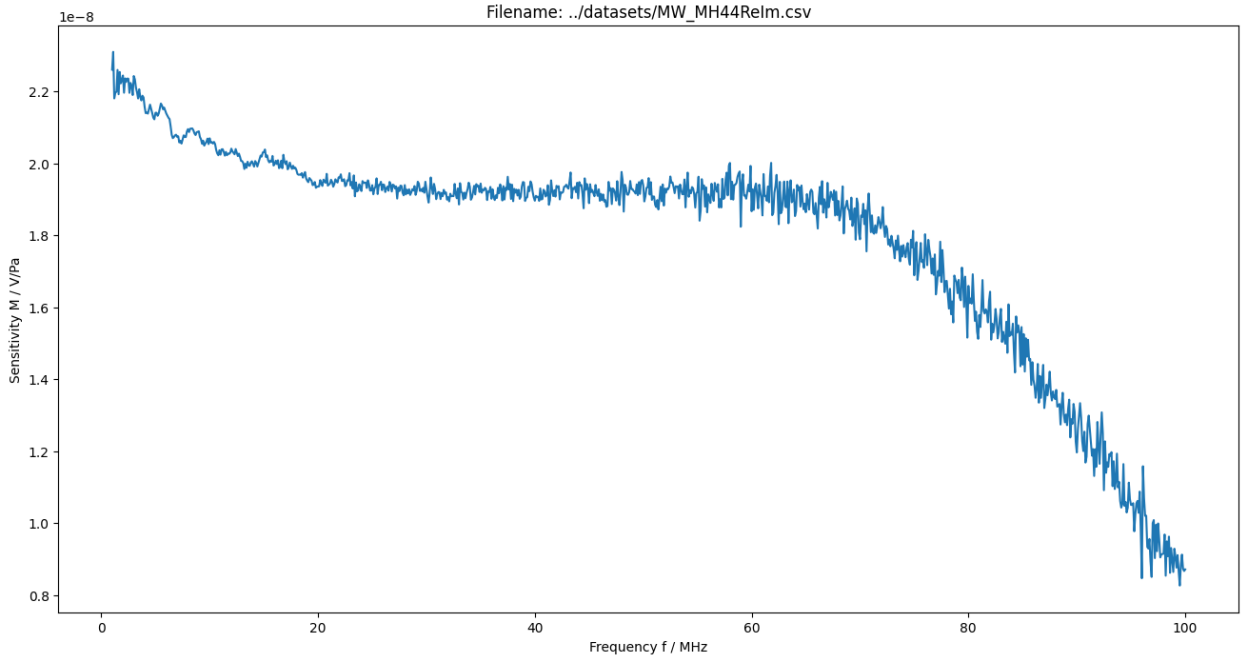
```
      name: <class 'str'>
 frequency: <class 'numpy.ndarray'>
      real: <class 'numpy.ndarray'>
      imag: <class 'numpy.ndarray'>
   varreal: <class 'numpy.ndarray'>
   varimag: <class 'numpy.ndarray'>
       cov: <class 'numpy.ndarray'>
```

3.1.1 Reduce frequency range

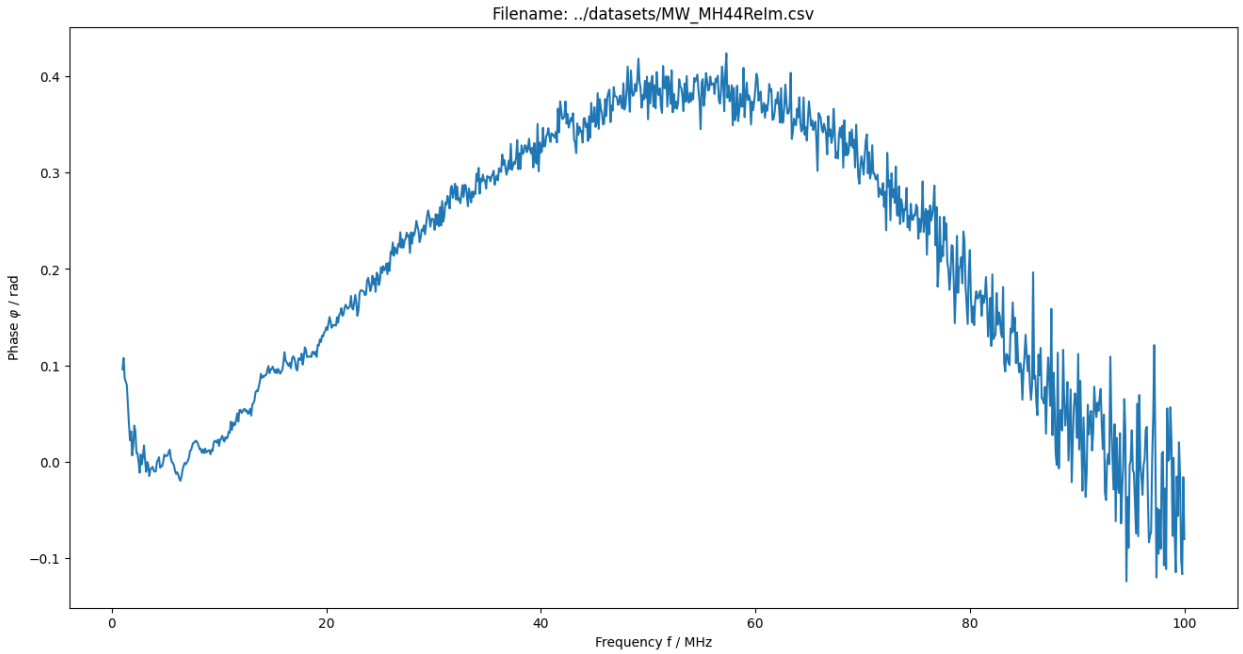
[6]:
```
hyd_data = reduce_freq_range(hyd_data, fmin=1e6, fmax=100e6)
```

3.1.2 Plot amplitude and phase data

[7]:
```
figure(figsize=(16, 8))
plot(
    hyd_data["frequency"] / 1e6, np.sqrt(hyd_data["real"] ** 2 + hyd_data["imag"] ** 2)
)
xlabel("Frequency f / MHz")
ylabel("Sensitivity M / V/Pa")
title("Filename: {}".format(hyd_data["name"]))
show()
```



[8]:
```
figure(figsize=(16, 8))
plot(hyd_data["frequency"] / 1e6, np.arctan2(hyd_data["imag"], hyd_data["real"]))
xlabel("Frequency f / MHz")
ylabel(r"Phase $\varphi$ / rad")
title("Filename: {}".format(hyd_data["name"]))
show()
```



CHAPTER

FOUR

INTERPOLATION AND EXTRAPOLATION OF CALIBRATION DATA

[1]:
```
%matplotlib inline
```

[2]:
```
from meas_data_preprocessing import *
from hydrophone_data_preprocessing import *
```

[3]:
```
from PyDynamic.uncertainty.interpolate import interp1d_unc
```

4.1 Read measured data and calibration data from file

[4]:
```
meas_scenario = 13
infos, measurement_data = read_data(meas_scenario=meas_scenario)
_, hyd_data = read_calib_data(meas_scenario=meas_scenario, do_plot=False)
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
The file ../datasets/pD7_MH44.DAT was read and it contains 2500 data points.
The time increment is 2e-09 s
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
```

(continues on next page)

(continued from previous page)

```
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
```

[5]:
```
# metadata for chosen measurement scenario
for key in infos.keys():
    print("%20s: %s" % (key, infos[key]))
```

```
                   i: 13
      hydrophonname: GAMPT MH44
    measurementtype: Pulse-Doppler-Mode 7 MHz
    measurementfile: ../datasets/pD7_MH44.DAT
          noisefile: ../datasets/pD7_MH44r.DAT
        hydfilename: ../datasets/MW_MH44ReIm.csv
```

4.2 Perform basic pre-processing

[6]:
```
# remove DC component in measured data
measurement_data = remove_DC_component(measurement_data)
```

[7]:
```
# reduce frequency range of calibration data
hyd_data = reduce_freq_range(hyd_data, fmin=1e6, fmax=100e6)
```

4.3 Align spectral data of calibration and measured data

[8]:
```
measurement_data = uncertainty_from_noisefile(
    infos, measurement_data, do_plot=False, verbose=False
)
measurement_data = calculate_spectrum(measurement_data, do_plot=False)
fmeas = measurement_data["frequency"].round()
N = len(fmeas) // 2
```

4.3.1 Interpolation of real part

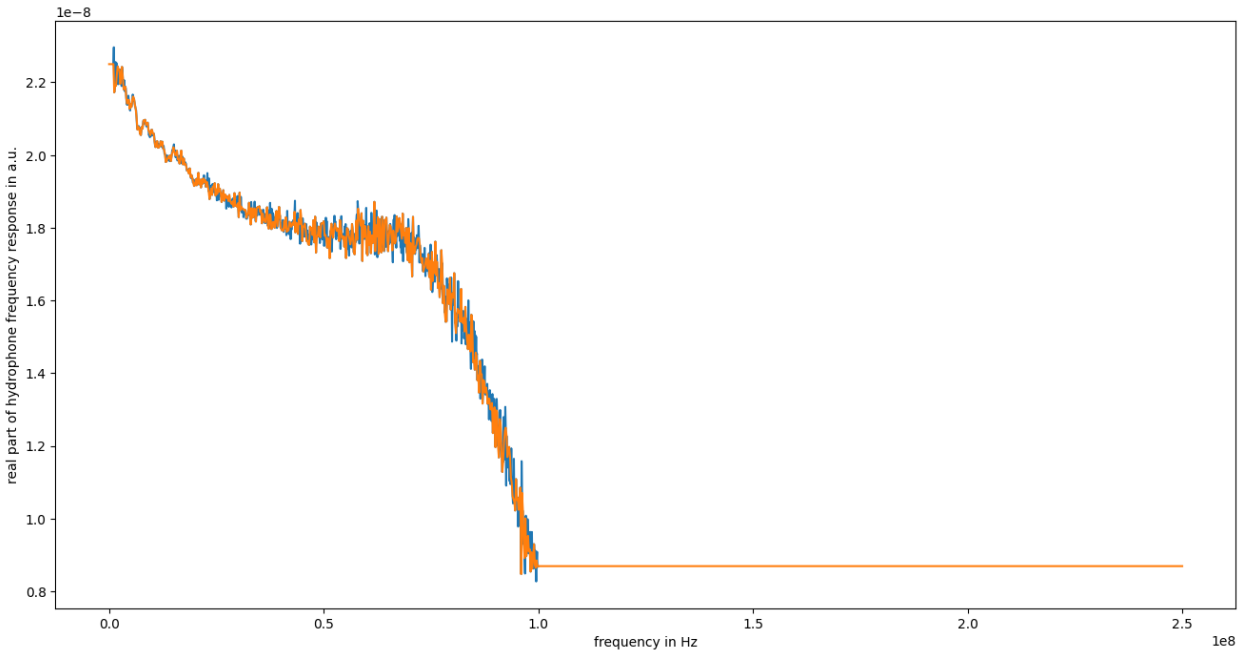
[9]:
```
hyd_interp = dict([])
(
    hyd_interp["frequency"],
    hyd_interp["real"],
    hyd_interp["varreal"],
    Creal,
```

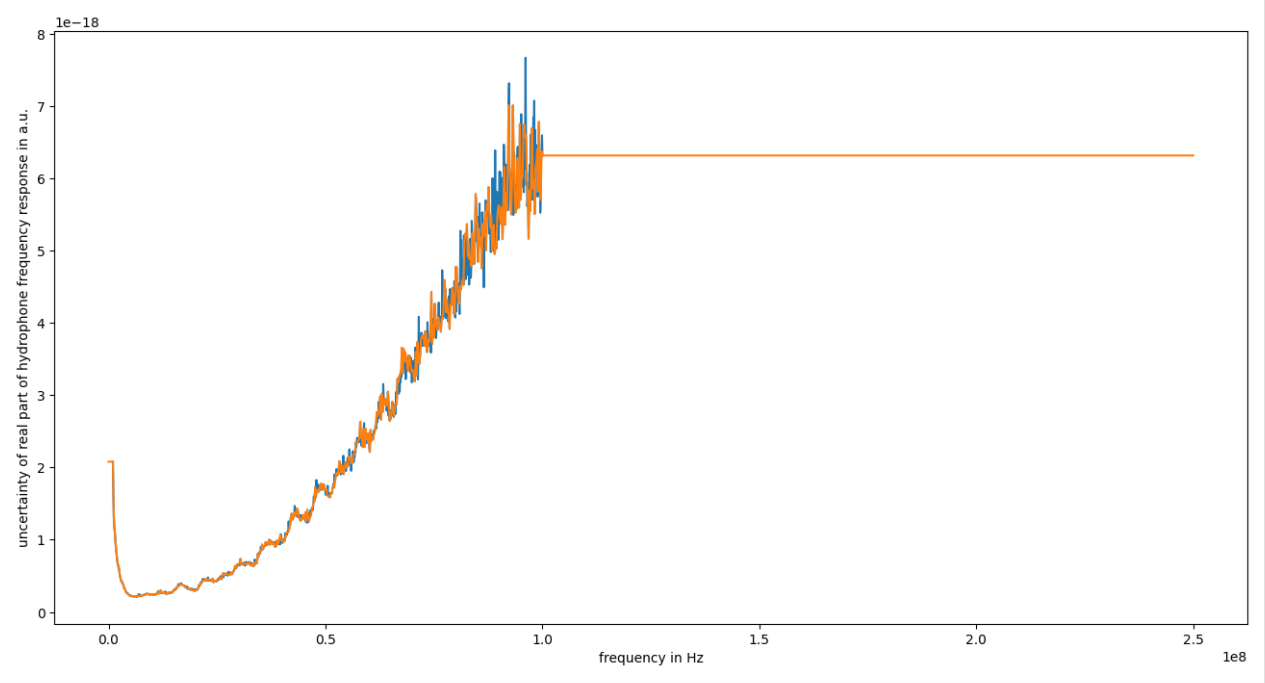
(continues on next page)

(continued from previous page)

```
) = interp1d_unc(
    fmeas[:N],
    hyd_data["frequency"],
    hyd_data["real"],
    hyd_data["varreal"],
    bounds_error=False,
    fill_value="extrapolate",
    fill_unc="extrapolate",
    returnC=True,
)
```

[10]:
```
figure(figsize=(16, 8))
plot(hyd_data["frequency"], hyd_data["real"])
plot(hyd_interp["frequency"], hyd_interp["real"])
xlabel("frequency in Hz")
ylabel("real part of hydrophone frequency response in a.u.")

figure(figsize=(16, 8))
plot(hyd_data["frequency"], hyd_data["varreal"])
plot(hyd_interp["frequency"], hyd_interp["varreal"])
xlabel("frequency in Hz")
ylabel("uncertainty of real part of hydrophone frequency response in a.u.")
show()
```





4.3.2 Interpolation of imaginary part

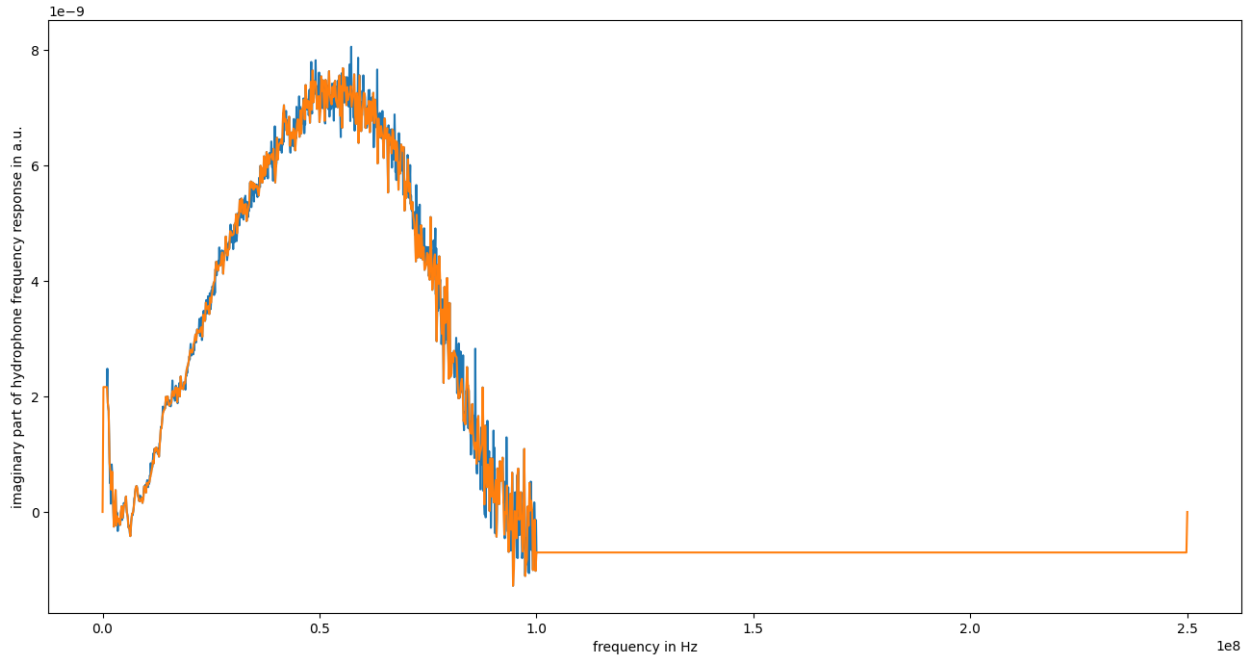
[11]:
```
(
    hyd_interp["frequency"],
    hyd_interp["imag"],
    hyd_interp["varimag"],
    Cimag,
) = interp1d_unc(
    fmeas[:N],
    hyd_data["frequency"],
    hyd_data["imag"],
    hyd_data["varimag"],
    bounds_error=False,
    fill_value="extrapolate",
    fill_unc="extrapolate",
    returnC=True,
)
# adjustment of end points
hyd_interp["imag"][0] = 0  # Must be 0 by definition
hyd_interp["imag"][-1] = 0
hyd_interp["varimag"][0] = 0  # Must be 0 by definition
hyd_interp["varimag"][-1] = 0
```

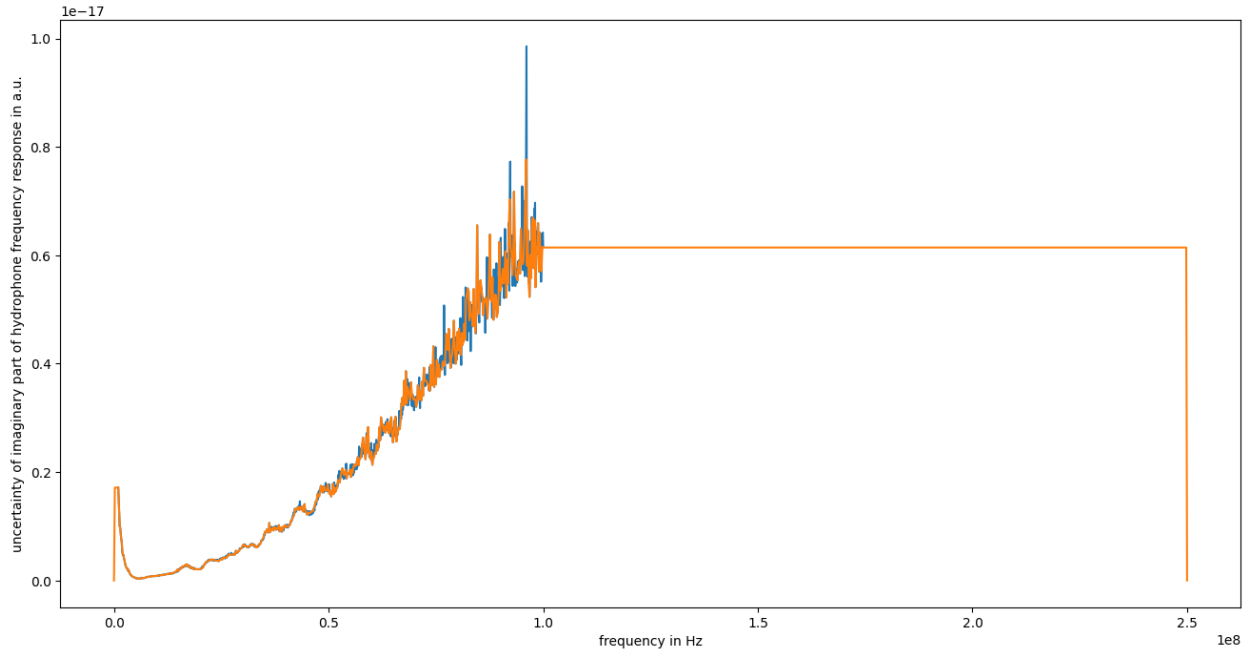
[12]:
```
figure(figsize=(16, 8))
plot(hyd_data["frequency"], hyd_data["imag"])
plot(hyd_interp["frequency"], hyd_interp["imag"])
xlabel("frequency in Hz")
ylabel("imaginary part of hydrophone frequency response in a.u.")
```

(continues on next page)

(continued from previous page)

```
figure(figsize=(16, 8))
plot(hyd_data["frequency"], hyd_data["varimag"])
plot(hyd_interp["frequency"], hyd_interp["varimag"])
xlabel("frequency in Hz")
ylabel("uncertainty of imaginary part of hydrophone frequency response in a.u.")
show()
```



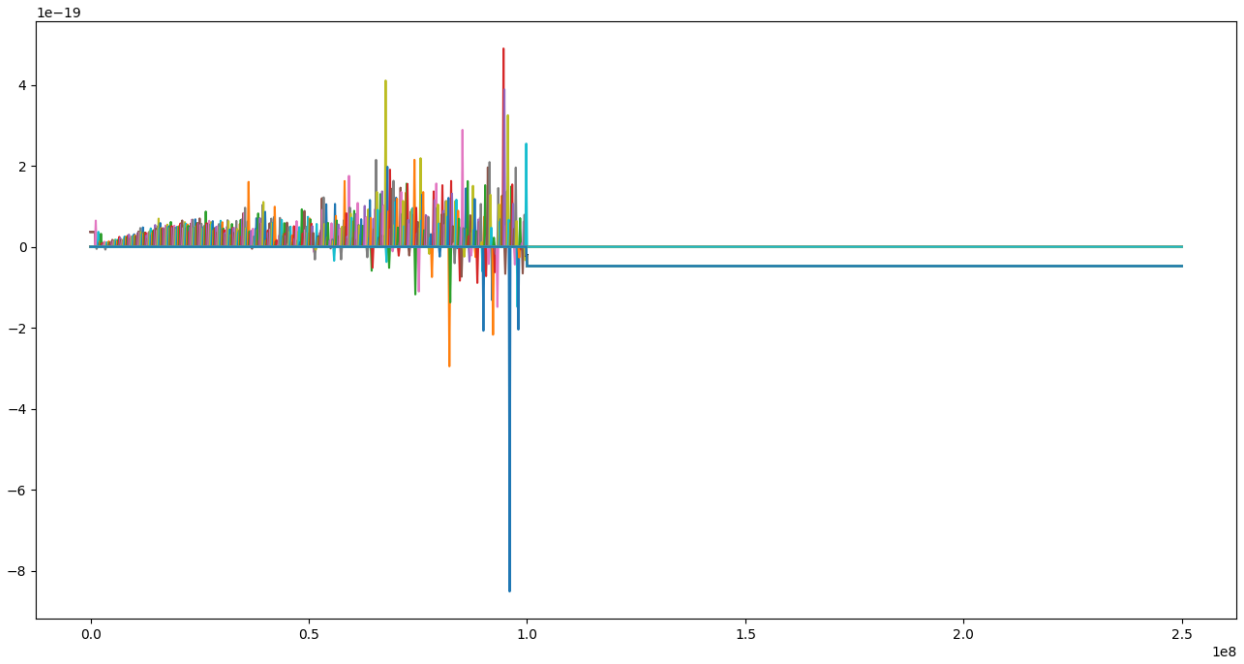


4.3.3 Calculation of mixed uncertainties at new frequencies

$$U_{r_{interp},i_{interp}} = C_r U_{r,i} C_i^T$$

[13]:
```
hyd_interp["cov"] = (Creal.dot(np.diag(hyd_data["cov"]))).dot(Cimag.T)
```

[14]:
```
figure(figsize=(16, 8))
plot(hyd_interp["frequency"], hyd_interp["cov"])
show()
```



CHAPTER

FIVE

CALCULATION OF IMPULSE RESPONSE OF HYDROPHONE

[1]:
```
%matplotlib inline
```

[2]:
```
from meas_data_preprocessing import *
from hydrophone_data_preprocessing import *
from PyDynamic.uncertainty.propagate_DFT import GUM_iDFT
```

5.1 Load calibration data

[3]:
```
meas_scenario = 13
infos, measurement_data = read_data(meas_scenario=meas_scenario)
_, hyd_data = read_calib_data(meas_scenario=meas_scenario, do_plot=False)
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
The file ../datasets/pD7_MH44.DAT was read and it contains 2500 data points.
The time increment is 2e-09 s
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
```

(continues on next page)

(continued from previous page)

```
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
```

5.2 Align calibration data with measurement data

[4]:
```
# reduce frequency range of calibration data
hyd_data = reduce_freq_range(hyd_data, fmin=1e6, fmax=100e6)
```

[5]:
```
measurement_data = uncertainty_from_noisefile(
    infos, measurement_data, do_plot=False, verbose=False
)
measurement_data = calculate_spectrum(measurement_data, do_plot=False)
fmeas = measurement_data["frequency"].round()
```

[6]:
```
hyd_interp = interpolate_hyd(hyd_data, fmeas)
```

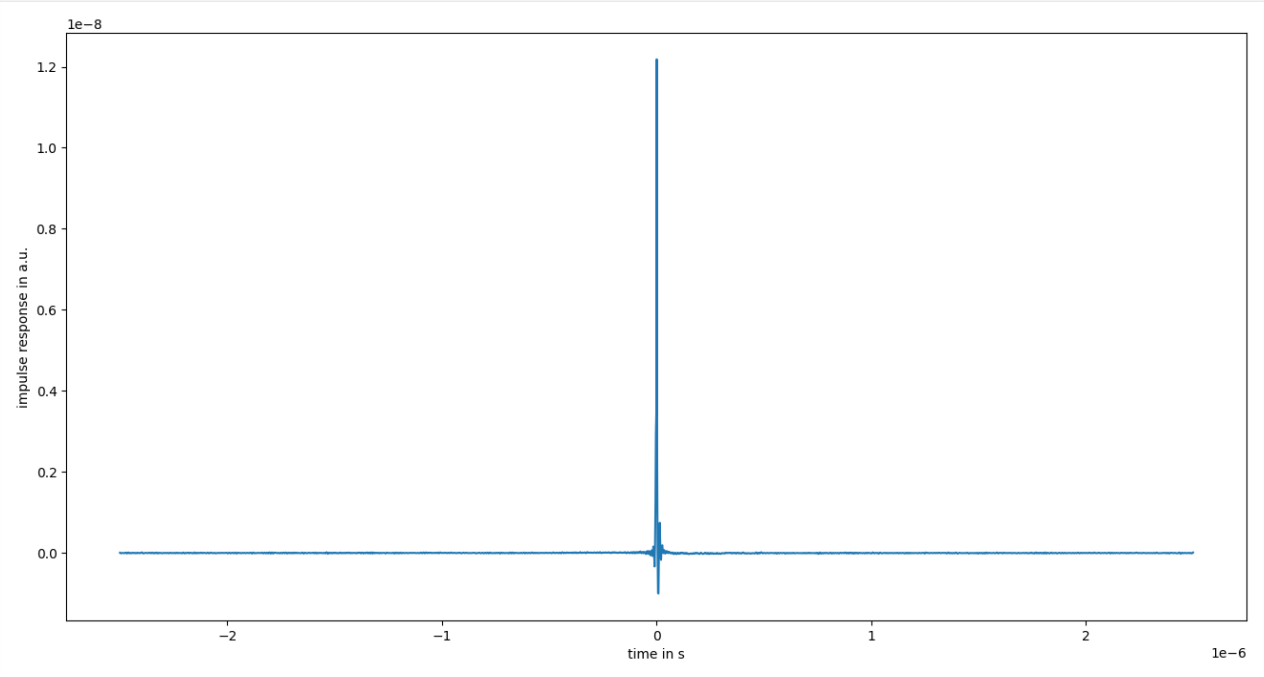
5.3 Transform to time domain to calculate impulse response

[7]:
```
H_RI = np.r_[hyd_interp["real"], hyd_interp["imag"]]
U_HRI = np.r_[
    np.c_[np.diag(hyd_interp["varreal"]), hyd_interp["cov"]],
    np.c_[hyd_interp["cov"], np.diag(hyd_interp["varimag"])],
]
```

[8]:
```
# application of inverse Fourier transform
imp, Uimp = GUM_iDFT(H_RI, U_HRI)
```

[9]:
```
# centralisation of impulse response
dt = 1 / (hyd_interp["frequency"][1] - hyd_interp["frequency"][0])
c_time = np.linspace(-dt / 2, dt / 2, np.size(imp))
c_imp = np.fft.fftshift(imp)
```

[10]:
```
figure(figsize=(16, 8))
plot(c_time, c_imp)
xlabel("time in s")
ylabel("impulse response in a.u.")
show()
```


1e−8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
impulse response in a.u.
−2
−1
0
1
2
time in s
1e−6

CHAPTER

SIX

DECONVOLUTION IN THE FREQUENCY DOMAIN

[1]:
```
%matplotlib inline
```

[2]:
```
from meas_data_preprocessing import *
from hydrophone_data_preprocessing import *
from PyDynamic.uncertainty.propagate_DFT import DFT_deconv, GUM_iDFT
```

6.1 Load calibration data

[3]:
```
meas_scenario = 13
infos, measurement_data = read_data(meas_scenario=meas_scenario)
_, hyd_data = read_calib_data(meas_scenario=meas_scenario, do_plot=False)
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
The file ../datasets/pD7_MH44.DAT was read and it contains 2500 data points.
The time increment is 2e-09 s
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
```

(continues on next page)

(continued from previous page)

```
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
```

[4]:
```
# metadata for chosen measurement scenario
for key in infos.keys():
    print("%20s: %s" % (key, infos[key]))
```

```
                   i: 13
      hydrophonname: GAMPT MH44
     measurementtype: Pulse-Doppler-Mode 7 MHz
     measurementfile: ../datasets/pD7_MH44.DAT
           noisefile: ../datasets/pD7_MH44r.DAT
         hydfilename: ../datasets/MW_MH44ReIm.csv
```

6.2 Pre-process measurement data

[5]:
```
# remove DC component
measurement_data = remove_DC_component(measurement_data)
```

[6]:
```
# Calculate measurement uncertainty from noise data
measurement_data = uncertainty_from_noisefile(infos, measurement_data, do_plot=False)
```

```
The file "../datasets/pD7_MH44r.DAT" was read and it contains 2500 data points
```

[7]:
```
# calculate spectrum
measurement_data = calculate_spectrum(measurement_data, do_plot=False)
```

[8]:
```
# available measurement data
for key in measurement_data.keys():
    print("%12s: %s" % (key, type(measurement_data[key])))
```

```
        name: <class 'str'>
     voltage: <class 'numpy.ndarray'>
        time: <class 'numpy.ndarray'>
 uncertainty: <class 'numpy.ndarray'>
   frequency: <class 'numpy.ndarray'>
    spectrum: <class 'numpy.ndarray'>
     varspec: <class 'numpy.ndarray'>
```

6.3 Pre-process calibration data

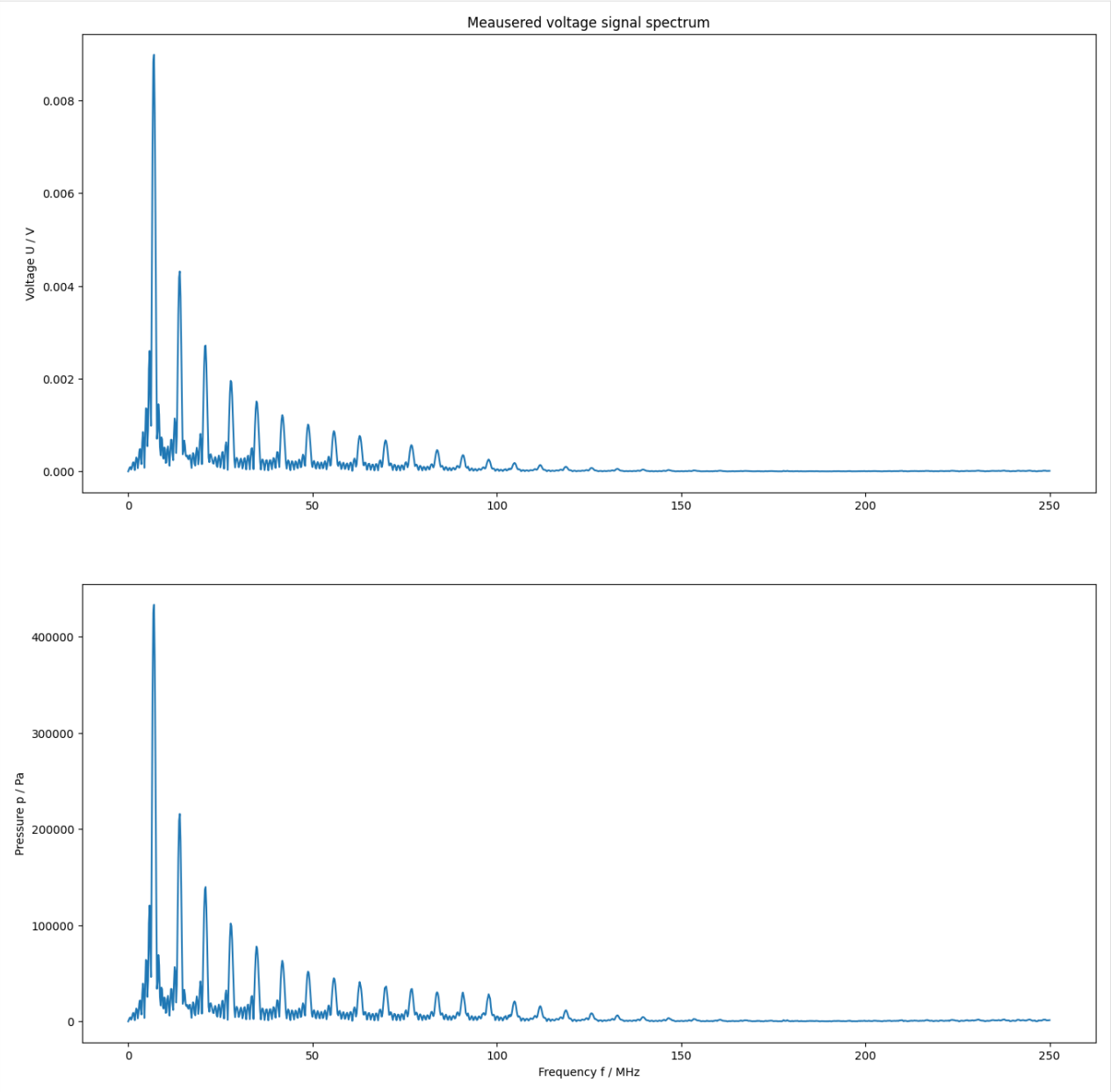
[9]:
```
# reduce frequency range of calibration data
hyd_data = reduce_freq_range(hyd_data, fmin=1e6, fmax=100e6)
```

[10]:
```
# align spectrum of hydrophone frequency response with spectrum of measurement
fmeas = measurement_data["frequency"].round()
hyd_interp = interpolate_hyd(hyd_data, fmeas)
```

6.4 Deconvolution in the frequency domain

[11]:
```
# prepare matrix-vector notation for DFT_deconv
H_RI = np.r_[hyd_interp["real"], hyd_interp["imag"]]
U_HRI = np.r_[
    np.c_[np.diag(hyd_interp["varreal"]), hyd_interp["cov"]],
    np.c_[hyd_interp["cov"], np.diag(hyd_interp["varimag"])],
]

# application of DFT_deconv
deconv = {"frequency": measurement_data["frequency"]}
deconv["P"], deconv["U_P"] = DFT_deconv(
    H_RI, measurement_data["spectrum"], U_HRI, measurement_data["varspec"]
)
```

[12]:
```
f = measurement_data["frequency"]
N = len(f) // 2
figure(figsize=(16, 16))
subplot(2, 1, 1)
plot(f[:N] / 1e6, amplitude(measurement_data["spectrum"]))
title("Meausered voltage signal spectrum")
ylabel("Voltage U / V")

subplot(2, 1, 2)
plot(f[:N] / 1e6, amplitude(deconv["P"]))
xlabel("Frequency f / MHz")
ylabel("Pressure p / Pa")
show()
```



6.5 Transformation to the time domain

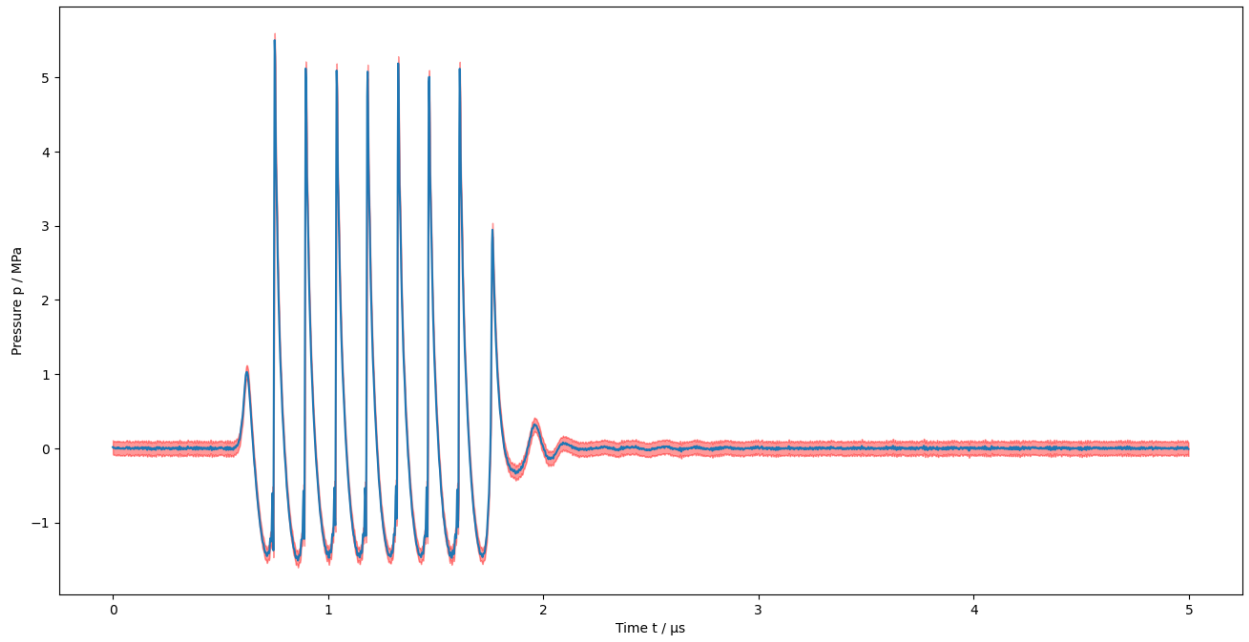
[13]:
```
deconvtime = {"t": measurement_data["time"]}
deconvtime["p"], deconvtime["Up"] = GUM_iDFT(deconv["P"], deconv["U_P"])

# correct for normalisation
deconvtime["p"] = deconvtime["p"] / 2 * np.size(deconvtime["t"])
deconvtime["Up"] = deconvtime["Up"] / 4 * np.size(deconvtime["t"]) ** 2
```

[14]:
```
figure(figsize=(16, 8))
plot(deconvtime["t"] / 1e-6, deconvtime["p"] / 1e6)
```

(continues on next page)

(continued from previous page)

```
fill_between(
    deconvtime["t"] / 1e-6,
    deconvtime["p"] / 1e6 - 2 * np.sqrt(np.diag(deconvtime["Up"])) / 1e6,
    deconvtime["p"] / 1e6 + 2 * np.sqrt(np.diag(deconvtime["Up"])) / 1e6,
    color="red",
    alpha=0.4,
)
xlabel("Time t / µs")
ylabel("Pressure p / MPa")
show()
```



CHAPTER

SEVEN

REGULARIZED DECONVOLUTION

[1]:
```
%matplotlib inline
```

[2]:
```
import matplotlib.pyplot as plt

from meas_data_preprocessing import *
from hydrophone_data_preprocessing import *
from PyDynamic.uncertainty.propagate_DFT import DFT_deconv, GUM_iDFT, DFT_multiply
from helper_methods import *
from regularization_bias import *
```

7.1 Pre-process measurement data

[3]:
```
meas_scenario = 13
infos, measurement_data = read_data(meas_scenario=meas_scenario)
_, hyd_data = read_calib_data(meas_scenario=meas_scenario, do_plot=False)

# metadata for chosen measurement scenario
for key in infos.keys():
    print("%20s: %s" % (key, infos[key]))
```

```
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
The file ../datasets/pD7_MH44.DAT was read and it contains 2500 data points.
The time increment is 2e-09 s
Checking if file ../datasets/pD7_MH44.DAT is already present or download it from https://
```

(continues on next page)

(continued from previous page)

```
↪raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/pD7_MH44r.DAT is already present or download it from https:/
↪/raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/MeasuredSignals/pD-
↪Mode%207%20MHz/pD7_MH44r.DAT otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
Checking if file ../datasets/MW_MH44ReIm.csv is already present or download it from␣
↪https://raw.githubusercontent.com/Ma-Weber/Tutorial-Deconvolution/master/
↪HydrophoneCalibrationData/MW_MH44ReIm.csv otherwise:
Replace is False and data exists, so doing nothing. Use replace=True to re-download the␣
↪data.
                  i: 13
       hydrophonname: GAMPT MH44
     measurementtype: Pulse-Doppler-Mode 7 MHz
     measurementfile: ../datasets/pD7_MH44.DAT
           noisefile: ../datasets/pD7_MH44r.DAT
         hydfilename: ../datasets/MW_MH44ReIm.csv
```

[4]:
```
# remove DC component
measurement_data = remove_DC_component(measurement_data)

# Calculate measurement uncertainty from noise data
measurement_data = uncertainty_from_noisefile(infos, measurement_data, do_plot=False)

# calculate spectrum
measurement_data = calculate_spectrum(measurement_data, do_plot=False)

# available measurement data
for key in measurement_data.keys():
    print("%12s: %s" % (key, type(measurement_data[key])))
```

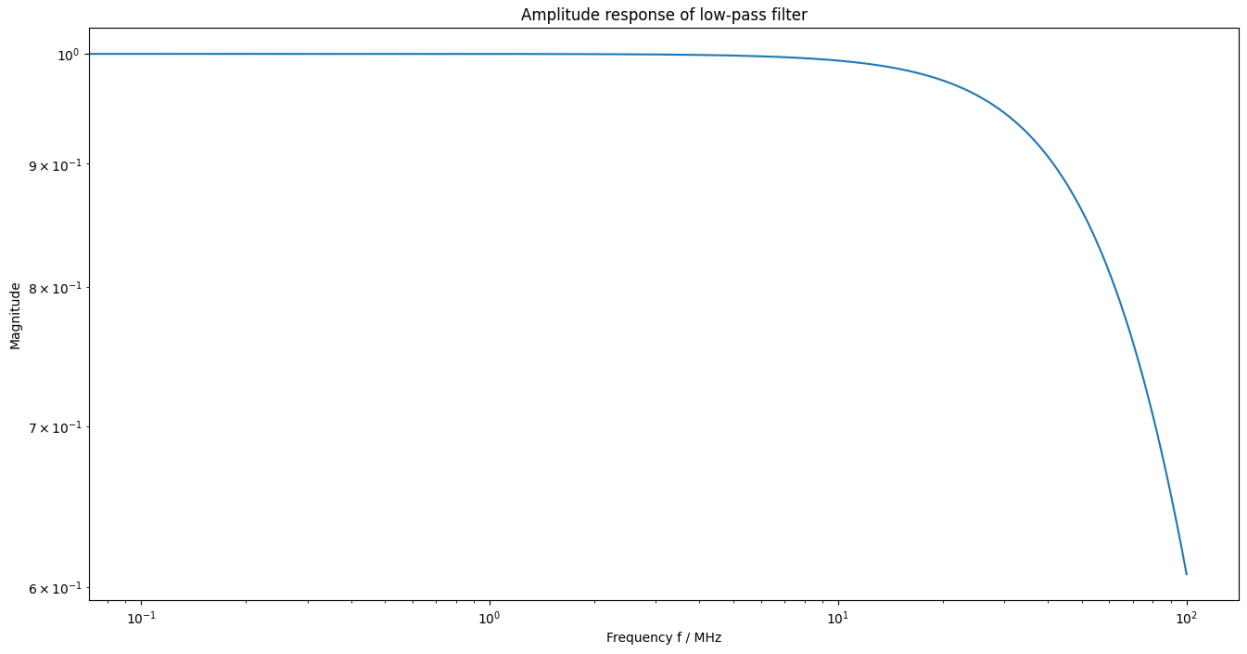
```
The file "../datasets/pD7_MH44r.DAT" was read and it contains 2500 data points
        name: <class 'str'>
     voltage: <class 'numpy.ndarray'>
        time: <class 'numpy.ndarray'>
 uncertainty: <class 'numpy.ndarray'>
   frequency: <class 'numpy.ndarray'>
    spectrum: <class 'numpy.ndarray'>
     varspec: <class 'numpy.ndarray'>
```

7.2 Pre-process calibration data

[5]:
```
# reduce frequency range of calibration data
hyd_data = reduce_freq_range(hyd_data, fmin=1e6, fmax=100e6)

# align spectrum of hydrophone frequency response with spectrum of measurement
fmeas = measurement_data["frequency"].round()
hyd_interp = interpolate_hyd(hyd_data, fmeas)
```

7.3 Set low-pass filter to suppress noise

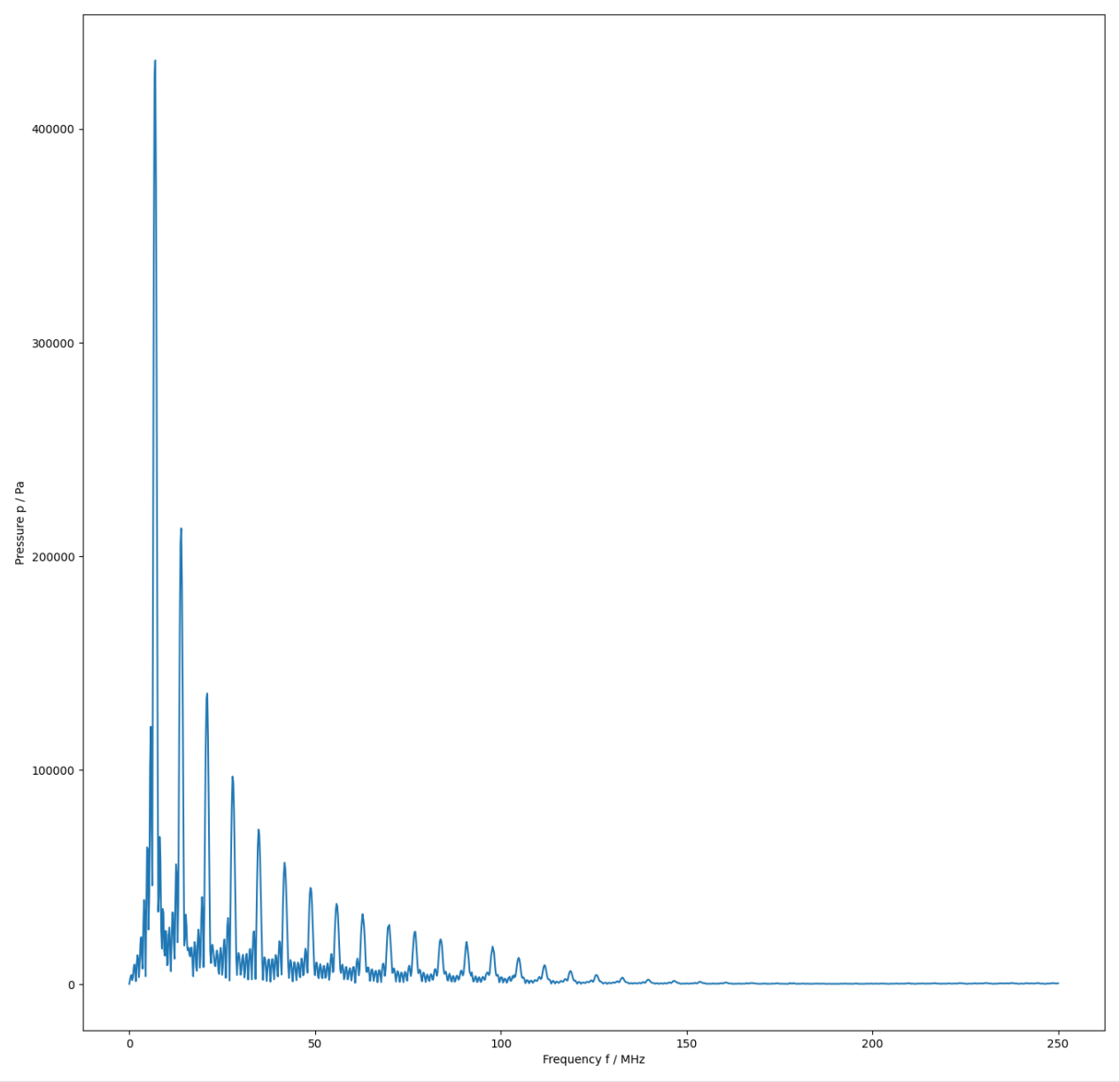
[6]:
```
fc = 80e6  # cut of frequency (Hz) #default is 80 MHz

H_lowpass = lambda f: 1 / (1 + 1j * f / (fc * 1.555)) ** 2

fpl = np.linspace(0, 1e8, 1000)
figure(figsize=(16, 8))
loglog(fpl * 1e-6, np.abs(H_lowpass(fpl)))
plt.title("Amplitude response of low-pass filter")
plt.xlabel("Frequency f / MHz")
plt.ylabel("Magnitude")
show()
```



7.4 Deconvolution and low-pass filtering

[7]:
```
# prepare matrix-vector notation for DFT_deconv
H_RI = np.r_[hyd_interp["real"], hyd_interp["imag"]]
U_HRI = np.r_[
    np.c_[np.diag(hyd_interp["varreal"]), hyd_interp["cov"]],
    np.c_[hyd_interp["cov"], np.diag(hyd_interp["varimag"])],
]

# application of DFT_deconv
deconv = {"frequency": measurement_data["frequency"]}
deconv["P"], deconv["U_P"] = DFT_deconv(
    H_RI, measurement_data["spectrum"], U_HRI, measurement_data["varspec"]
)

# application of low-pass filter
N = len(deconv["frequency"]) // 2
H1 = H_lowpass(deconv["frequency"][:N])
Hl_RI = np.r_[np.real(H1), np.imag(H1)]
deconv["P"], deconv["U_P"] = DFT_multiply(deconv["P"], Hl_RI, deconv["U_P"])
```

[8]:
```
f = measurement_data["frequency"]
N = len(f) // 2
figure(figsize=(16, 16))
plot(f[:N] / 1e6, amplitude(deconv["P"]))
xlabel("Frequency f / MHz")
ylabel("Pressure p / Pa")
show()
```



7.5 Estimation of regularization error

[9]:
```
# calculate the average working frequency
f_awf = calc_awf(measurement_data["frequency"], measurement_data["spectrum"])
```

```
searching for f2 in interval [6.8,20] MHz
determined f1: 6.68571 MHz
determined f2: 19.5559 MHz
resulting f_awf = 13.12079550035726 MHz
```

[10]:
```
def calculate_freq_points(fH, f, H, X, Fs, candidates, verbose=True):
    def get_candidates(f, S, number=40):
```

(continues on next page)

(continued from previous page)

```
        # Largest local maxima of np.abs(S)
        inds = dsp.argrelmax(np.abs(S))[0]
        inds2 = inds[np.argsort(np.abs(S[inds]))[::-1]]
        return f[inds2[:number]], inds2[:number], np.abs(S[inds2[:number]])

    def get_closest(freqs, f_localmax):
        # Closest local maximum to selected frequency
        cfreqs = freqs.copy()
        for k in range(len(freqs)):
            indf = np.argmin(np.abs(f_localmax - freqs[k]))
            cfreqs[k] = f_localmax[indf]
        return cfreqs

    Ts = 1 / Fs
    f = np.fft.rfftfreq(Nf, Ts)

    Xh = X / H

    f_local = get_candidates(fH, np.abs(Xh))[0]
    frequencies = get_closest(candidates, f_local)

    return frequencies
```

[11]:
```
# calculate center frequency candidates as multiples of fawf
multiples = [1, 3, 8]
fvals = [mult * f_awf for mult in multiples]
```

[12]:
```
H = amplitude(np.r_[hyd_interp["real"], hyd_interp["imag"]])
M = amplitude(np.r_[measurement_data["spectrum"]])
Ts = measurement_data["time"][1] - measurement_data["time"][0]
Fs = 1 / Ts
N = len(measurement_data["frequency"]) // 2
# center_frequencies = calculate_freq_points(measurement_data["frequency"][:N],
#                                            H, N, Ts, Fs, fvals)

# figure(figsize=(16,8))
# plot(f, amplitude(measurement_data["spectrum"]))
```

[13]:
```
# center_frequencies
```

[14]:
```
fvals
```

[14]:
```
[13120795.50035726, 39362386.50107178, 104966364.00285809]
```

CHAPTER

EIGHT

INTERPOLATION WITH PYDYNAMIC.UNCERTAINTY.INTERPOLATE.INTERP1D_UNC

In this series of notebooks we illustrate the use of our method `interp1d_unc`, which is very much inspired by Scipy's *interp1d* method and therefore closely aligned with its signature and corresponding capabilities. The main features are:

- interpolation of measurement values and associated uncertainties
- available interpolation methods are *linear, cubic, next, nearest, previous* (aligned with *scipy.interpolate.interp1d*)
- extrapolation of measurement values and associated uncertainties based on the values at the original data's bounds or by custom input
- returning sensitivity coefficients
- performance oriented optional parameters
- `copy`
- `assume_sorted`

Comprehensive details about the parameters meanings, you find on pydynamic.readthedocs.io.

8.1 Content

The examples proceed according to the following scheme.

8.1.1 01 Basic measurement data pre-processing.ipynb

- Set up the appropriate execution and plotting environment.
- Download an example data set of real sensor recordings from Zenodo.org.
- Visualize the relevant part of the contained data.

8.1.2 02 Basic interpolation.ipynb

- Conduct a simple interpolation.

8.1.3 03 Basic extrapolation.ipynb

- Conduct simple interpolation and extrapolation outside the original bounds.
- Demonstrate returning the sensitivity coefficients

8.2 Setup the Python environment

[1]:
```
import holoviews as hv
import numpy as np
import h5py
import pickle
from download import download
```

8.3 Setup plotting environment and labels

[2]:
```
# Set one of the available plotting backends ('plotly', 'bokeh', 'maplotlib').
hv.extension("bokeh")

# Define labels and units for plots.
timestamp_labels = hv.Dimension("relative measurement time", unit="s")
measurement_labels = hv.Dimension("Primary Nominal Current", unit="A")
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

8.4 Download sample data

This step of course is only necessary once, but it checks on execution, if the expected file is present. Thus you can safely execute it without producing unnecessary network traffic or waiting times. We use a sample data set of real measured motor current data which you can find on Zenodo:

The data actually used in this tutorial is contained in the axis 3 file.

In case you are having trouble downloading the data via the following code, please download it manually from the provided URL und store it as *PyDynamic_tutorials/datasets/axis3_2kHz.h5*.

[3]:
```
# Set URL and extract filename.
url = "https://zenodo.org/record/3929385/files/axis3_2kHz.h5"
filename = "".join(("../datasets/", url.split("/")[-1]))

path = download(url, filename, replace=False, verbose=True)
```

```
Downloading data from https://zenodo.org/record/3929385/files/axis3_2kHz.h5 (1.03 GB)

file_sizes: 100%|| 1.11G/1.11G [07:56<00:00, 2.32MB/s]
Successfully downloaded file to ../datasets/axis3_2kHz.h5
```

8.5 Unpack and prepare data to visualize sample set

We extract only a small excerpt of the data, which can be used well for demonstration purposes. According to the datasets documentation on Zenodo, we have to convert the measurement values to receive SI units as stated in the included PDF file.

[4]:
```
# Read the h5-file.
hf = h5py.File(path, "r")

# Set well-suited range of sensor measurements.
start = 2
n_nodes = 4
end = start + n_nodes

# Extract well-suited sensor measurements.
data_points_measured = hf["Sensor_Data"][8, start:end, 0]

# Free memory.
hf.close()

# Convert extraction into SI units.
data_points_measured_si = (
    ((data_points_measured * 8.76e-5) + 1.36e-2) * 5.299641744 * 2.0
)
data_points_measured_si
```

[4]:
```
array([2.47691988, 4.1246116 , 3.95322654, 1.41385091])
```

[5]:
```
# Determine measurement uncertainties from data sheet stating 1.5% of measured value.
data_points_measured_si_unc = data_points_measured_si * 0.015
data_points_measured_si_unc
```

[5]:
```
array([0.0371538 , 0.06186917, 0.0592984 , 0.02120776])
```

[6]:
```
# Construct relative time stamps from the beginning of the measurement based on the
# known frequency of 2 kHz in milliseconds.
t = np.linspace(start=start * 2e-3, stop=(end - 1) * 2e-3, num=n_nodes)
t
```

[6]:
```
array([0.004, 0.006, 0.008, 0.01 ])
```

8.6 Visualize the original data points

[7]:
```
original_curve = hv.Curve(
    (t, data_points_measured_si),
    timestamp_labels,
    measurement_labels,
    label="measurements",
)

original_uncertainties_plot = hv.Spread(
    (t, data_points_measured_si, data_points_measured_si_unc),
    vdims=[measurement_labels, "Associated Uncertainty"],
    kdims=timestamp_labels,
    label="uncertainties",
)

original_plot = (original_uncertainties_plot * original_curve).opts(
    title="Measured values and associated uncertainties",
)

original_plot.opts(width=600, height=800, legend_position="top_right")
```

[7]:
```
:Overlay
   .Spread.Uncertainties :Spread   [relative measurement time]   (Primary Nominal␣
↪Current,Associated Uncertainty)
   .Curve.Measurements   :Curve   [relative measurement time]   (Primary Nominal Current)
```

8.7 Store results to be used in later lessons

[8]:
```
np.save("data_points", data_points_measured_si)
np.save("data_points_unc", data_points_measured_si_unc)
np.save("time_stamps", t)
with open("original_plot.p", "wb") as f:
    pickle.dump(original_plot, f)
```

CHAPTER

NINE

BASIC INTERPOLATION WITH PYDYNAMIC.UNCERTAINTY.INTERP1D_UNC

This is the second notebook in the series to illustrate the use of our method `interp1d_unc`. We will *conduct a simple interpolation* and *return the sensitivity coeffients*.

9.1 Preparation

First we setup our Python and plotting environment and collect all previously extracted measurement data and their visualization.

9.1.1 Setup the Python environment

[1]:
```
import warnings

warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")

import holoviews as hv
import numpy as np
import pickle

from PyDynamic.uncertainty import interp1d_unc
```

9.1.2 Setup plotting environment and labels

[2]:
```
# Set one of the available plotting backends ('plotly', 'bokeh', 'maplotlib').
hv.extension("bokeh")

# Define labels and units for plots.
timestamp_labels = hv.Dimension(("Time", "relative measurement time"), unit="s")
measurement_labels = hv.Dimension(("Current", "Primary Nominal Current"), unit="A")
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

(continued from previous page)

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

9.1.3 Load results from previous lessons

[3]:
```
try:
    y = np.load("data_points.npy")
    uy = np.load("data_points_unc.npy")
    x = np.load("time_stamps.npy")
    with open("original_plot.p", "rb") as f:
        original_plot = pickle.load(f)
except FileNotFoundError:
    print(
        "The measurement data is not available. Please execute all steps in '01 "
        "Basic measurement data pre-processing' before you proceed."
    )
```

9.2 Setup interpolation timestamps

[4]:
```
# Setup list of vectors of interpolation timestamps with increasing number.
x_news = [
    np.linspace(start=x.min(), stop=x.max(), num=n_nodes)
    for n_nodes in np.arange(start=2, stop=20, step=3)
]
x_news
```

[4]:
```
[array([0.004, 0.01 ]),
 array([0.004 , 0.0055, 0.007 , 0.0085, 0.01  ]),
 array([0.004     , 0.00485714, 0.00571429, 0.00657143, 0.00742857,
        0.00828571, 0.00914286, 0.01      ]),
 array([0.004 , 0.0046, 0.0052, 0.0058, 0.0064, 0.007 , 0.0076, 0.0082,
        0.0088, 0.0094, 0.01  ]),
 array([0.004     , 0.00446154, 0.00492308, 0.00538462, 0.00584615,
        0.00630769, 0.00676923, 0.00723077, 0.00769231, 0.00815385,
        0.00861538, 0.00907692, 0.00953846, 0.01      ]),
 array([0.004   , 0.004375, 0.00475 , 0.005125, 0.0055  , 0.005875,
        0.00625 , 0.006625, 0.007   , 0.007375, 0.00775 , 0.008125,
        0.0085  , 0.008875, 0.00925 , 0.009625, 0.01    ])]
```

9.3 Conduct and visualize interpolation

To conduct the simplest interpolation, we specify new interpolation nodes and the original timestamps or frequencies, measurement values and uncertainties. This results by default in a linear interpolation. The available interpolation methods can be chosen in the dropdown menu in the following plot. Those can be specified using the parameter `kind`.

[5]:
```
# Create plots of interpolated values and uncertainties for the increasing number of
# interpolation nodes.
interpolation_dict = {}
for x_new in x_news:
    for i_kind in ("linear", "nearest", "next", "previous", "cubic"):
        # Conduct the actual interpolation for the current set of interpolation nodes␣
↪and kind.
        x_new, y_new, uy_new = interp1d_unc(x_new=x_new, x=x, y=y, uy=uy, kind=i_kind)
        print(x_new, y_new, uy_new)

        # Create plot of the interpolated values.
        curve_interp = hv.Curve(
            (x_new, y_new),
            timestamp_labels,
            measurement_labels,
            label="interpolated values",
        )

        # Create plot of the interpolated uncertainties.
        interp_uncertainties = hv.Spread(
            (x_new, y_new, uy_new),
            vdims=[measurement_labels, "Associated Uncertainty"],
            kdims=timestamp_labels,
            label="interpolated uncertainties",
        )

        interpolation_dict[x_new.size, i_kind] = curve_interp * interp_uncertainties
```

```
[0.004 0.01 ] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.004 0.01 ] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.004 0.01 ] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.004 0.01 ] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.004 0.01 ] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.004  0.0055 0.007  0.0085 0.01  ] [2.47691988 3.71268867 4.03891907 3.31838263 1.
↪41385091] [0.0371538  0.0473224  0.04284885 0.04478872 0.02120776]
[0.004  0.0055 0.007  0.0085 0.01  ] [2.47691988 4.1246116  4.1246116  3.95322654 1.
↪41385091] [0.0371538  0.06186917 0.06186917 0.0592984  0.02120776]
[0.004  0.0055 0.007  0.0085 0.01  ] [2.47691988 4.1246116  3.95322654 1.41385091 1.
↪41385091] [0.0371538  0.06186917 0.0592984  0.02120776 0.02120776]
[0.004  0.0055 0.007  0.0085 0.01  ] [2.47691988 2.47691988 4.1246116  3.95322654 1.
↪41385091] [0.0371538  0.0371538  0.06186917 0.0592984  0.02120776]
[0.004  0.0055 0.007  0.0085 0.01  ] [2.47691988 3.86178517 4.30061078 3.5618237  1.
↪41385091] [0.0371538  0.06658834 0.04827905 0.06395319 0.02120776]
[0.004      0.00485714 0.00571429 0.00657143 0.00742857 0.00828571
 0.00914286 0.01      ] [2.47691988 3.18307347 3.88922707 4.07564444 4.0021937  3.
↪59045859
 2.50215475 1.41385091] [0.0371538  0.03396776 0.05329567 0.04732865 0.04589666 0.
```

(continues on next page)

(continued from previous page)

```
↪05091741
 0.02815519 0.02120776]
[0.004      0.00485714 0.00571429 0.00657143 0.00742857 0.00828571
 0.00914286 0.01      ] [2.47691988 2.47691988 4.1246116  4.1246116  3.95322654 3.
↪95322654
 1.41385091 1.41385091] [0.0371538  0.0371538  0.06186917 0.06186917 0.0592984  0.0592984
 0.02120776 0.02120776]
[0.004      0.00485714 0.00571429 0.00657143 0.00742857 0.00828571
 0.00914286 0.01      ] [2.47691988 4.1246116  4.1246116  3.95322654 3.95322654 1.
↪41385091
 1.41385091 1.41385091] [0.0371538  0.06186917 0.06186917 0.0592984  0.0592984  0.
↪02120776
 0.02120776 0.02120776]
[0.004      0.00485714 0.00571429 0.00657143 0.00742857 0.00828571
 0.00914286 0.01      ] [2.47691988 2.47691988 2.47691988 4.1246116  4.1246116  3.
↪95322654
 3.95322654 1.41385091] [0.0371538  0.0371538  0.0371538  0.06186917 0.06186917 0.0592984
 0.0592984  0.02120776]
[0.004      0.00485714 0.00571429 0.00657143 0.00742857 0.00828571
 0.00914286 0.01      ] [2.47691988 3.37061027 3.98779646 4.2852695  4.21982042 3.
↪74824026
 2.82732008 1.41385091] [0.0371538  0.0585222  0.06543768 0.05221132 0.05062119 0.
↪06278262
 0.05557142 0.02120776]
[0.004  0.0046 0.0052 0.0058 0.0064 0.007  0.0076 0.0082 0.0088 0.0094
 0.01   ] [2.47691988 2.9712274  3.46553491 3.95984243 4.09033459 4.03891907
 3.98750355 3.69928898 2.93747629 2.1756636  1.41385091] [0.0371538  0.03195152 0.
↪03998588 0.05580607 0.05089637 0.04284885
 0.04902595 0.05341068 0.03657637 0.02317011 0.02120776]
[0.004  0.0046 0.0052 0.0058 0.0064 0.007  0.0076 0.0082 0.0088 0.0094
 0.01   ] [2.47691988 2.47691988 4.1246116  4.1246116  4.1246116  4.1246116
 3.95322654 3.95322654 3.95322654 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪06186917 0.06186917 0.06186917 0.06186917
 0.0592984  0.0592984  0.0592984  0.02120776 0.02120776]
[0.004  0.0046 0.0052 0.0058 0.0064 0.007  0.0076 0.0082 0.0088 0.0094
 0.01   ] [2.47691988 4.1246116  4.1246116  4.1246116  3.95322654 3.95322654
 3.95322654 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.06186917 0.
↪06186917 0.06186917 0.0592984  0.0592984
 0.0592984  0.02120776 0.02120776 0.02120776 0.02120776]
[0.004  0.0046 0.0052 0.0058 0.0064 0.007  0.0076 0.0082 0.0088 0.0094
 0.01   ] [2.47691988 2.47691988 2.47691988 2.47691988 4.1246116  4.1246116
 4.1246116  3.95322654 3.95322654 3.95322654 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.06186917 0.06186917
 0.06186917 0.0592984  0.0592984  0.0592984  0.02120776]
[0.004  0.0046 0.0052 0.0058 0.0064 0.007  0.0076 0.0082 0.0088 0.0094
 0.01   ] [2.47691988 3.12957009 3.65308495 4.0326438  4.25342597 4.30061078
 4.15937756 3.81490563 3.25237433 2.45696298 1.41385091] [0.0371538  0.04994546 0.
↪0652044  0.06457347 0.05500119 0.04827905
 0.05297116 0.06192576 0.06258734 0.04591678 0.02120776]
[0.004      0.00446154 0.00492308 0.00538462 0.00584615 0.00630769
 0.00676923 0.00723077 0.00769231 0.00815385 0.00861538 0.00907692
 0.00953846 0.01      ] [2.47691988 2.85715643 3.23739298 3.61762953 3.99786608 4.
```

(continues on next page)

(continued from previous page)

```
↪09824467
 4.05869427 4.01914387 3.97959347 3.75788996 3.17188019 2.58587043
 1.99986067 1.41385091] [0.0371538  0.03194769 0.0348658  0.04433185 0.05718147 0.
↪05313978
 0.04438177 0.04356442 0.05107041 0.05476129 0.04156812 0.02965537
 0.02129306 0.02120776]
[0.004      0.00446154 0.00492308 0.00538462 0.00584615 0.00630769
 0.00676923 0.00723077 0.00769231 0.00815385 0.00861538 0.00907692
 0.00953846 0.01      ] [2.47691988 2.47691988 2.47691988 4.1246116  4.1246116  4.1246116
 4.1246116  3.95322654 3.95322654 3.95322654 3.95322654 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.06186917 0.06186917 0.
↪06186917
 0.06186917 0.0592984  0.0592984  0.0592984  0.0592984  0.02120776
 0.02120776 0.02120776]
[0.004      0.00446154 0.00492308 0.00538462 0.00584615 0.00630769
 0.00676923 0.00723077 0.00769231 0.00815385 0.00861538 0.00907692
 0.00953846 0.01      ] [2.47691988 4.1246116  4.1246116  4.1246116  4.1246116  3.
↪95322654
 3.95322654 3.95322654 3.95322654 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.06186917 0.06186917 0.06186917 0.06186917 0.0592984
 0.0592984  0.0592984  0.0592984  0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776]
[0.004      0.00446154 0.00492308 0.00538462 0.00584615 0.00630769
 0.00676923 0.00723077 0.00769231 0.00815385 0.00861538 0.00907692
 0.00953846 0.01      ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 4.1246116
 4.1246116  4.1246116  4.1246116  3.95322654 3.95322654 3.95322654
 3.95322654 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.
↪06186917
 0.06186917 0.06186917 0.06186917 0.0592984  0.0592984  0.0592984
 0.0592984  0.02120776]
[0.004      0.00446154 0.00492308 0.00538462 0.00584615 0.00630769
 0.00676923 0.00723077 0.00769231 0.00815385 0.00861538 0.00907692
 0.00953846 0.01      ] [2.47691988 2.98988055 3.42845353 3.78589296 4.05545297 4.2303877
 4.30395127 4.26939781 4.11998146 3.84895634 3.4495766  2.91509636
 2.23876975 1.41385091] [0.0371538  0.04451507 0.06025142 0.06650832 0.06402752 0.
↪05662064
 0.04970399 0.04878094 0.0544091  0.06138933 0.06389044 0.05742552
 0.0391948  0.02120776]
[0.004    0.004375 0.00475  0.005125 0.0055   0.005875 0.00625  0.006625
 0.007    0.007375 0.00775  0.008125 0.0085   0.008875 0.00925  0.009625
 0.01    ] [2.47691988 2.78586208 3.09480427 3.40374647 3.71268867 4.02163087
 4.10318847 4.07105377 4.03891907 4.00678437 3.97464967 3.79451557
 3.31838263 2.8422497  2.36611677 1.88998384 1.41385091] [0.0371538  0.03233966 0.
↪03282536 0.03841037 0.0473224  0.05804881
 0.05464062 0.04639633 0.04284885 0.04511994 0.05245928 0.05560805
 0.04478872 0.03462178 0.02588766 0.02050702 0.02120776]
[0.004    0.004375 0.00475  0.005125 0.0055   0.005875 0.00625  0.006625
 0.007    0.007375 0.00775  0.008125 0.0085   0.008875 0.00925  0.009625
 0.01    ] [2.47691988 2.47691988 2.47691988 4.1246116  4.1246116  4.1246116
 4.1246116  4.1246116  4.1246116  3.95322654 3.95322654 3.95322654
 3.95322654 3.95322654 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.06186917 0.06186917 0.06186917
```

(continues on next page)

(continued from previous page)

```
 0.06186917 0.06186917 0.06186917 0.0592984  0.0592984  0.0592984
 0.0592984  0.0592984  0.02120776 0.02120776 0.02120776]
[0.004    0.004375 0.00475  0.005125 0.0055   0.005875 0.00625  0.006625
 0.007    0.007375 0.00775  0.008125 0.0085   0.008875 0.00925  0.009625
 0.01    ] [2.47691988 4.1246116  4.1246116  4.1246116  4.1246116  4.1246116
 3.95322654 3.95322654 3.95322654 3.95322654 3.95322654 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.06186917 0.
↪06186917 0.06186917 0.06186917 0.06186917
 0.0592984  0.0592984  0.0592984  0.0592984  0.0592984  0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776]
[0.004    0.004375 0.00475  0.005125 0.0055   0.005875 0.00625  0.006625
 0.007    0.007375 0.00775  0.008125 0.0085   0.008875 0.00925  0.009625
 0.01    ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.47691988
 4.1246116  4.1246116  4.1246116  4.1246116  4.1246116  3.95322654
 3.95322654 3.95322654 3.95322654 3.95322654 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.0371538  0.0371538
 0.06186917 0.06186917 0.06186917 0.06186917 0.06186917 0.0592984
 0.0592984  0.0592984  0.0592984  0.0592984  0.02120776]
[0.004    0.004375 0.00475  0.005125 0.0055   0.005875 0.00625  0.006625
 0.007    0.007375 0.00775  0.008125 0.0085   0.008875 0.00925  0.009625
 0.01    ] [2.47691988 2.89916327 3.27313417 3.59521425 3.86178517 4.06922862
 4.21392625 4.29225975 4.30061078 4.23536102 4.09289214 3.86958581
 3.5618237  3.16598748 2.67845883 2.09561941 1.41385091] [0.0371538  0.04116281 0.
↪05527826 0.06422911 0.06658834 0.06366031
 0.05764441 0.051439   0.04827905 0.05001373 0.05533815 0.06103035
 0.06395319 0.06159038 0.05201839 0.03458366 0.02120776]
```

[6]:
```
# Visualize the interpolation result along-side the original data.
(
    original_plot
    * hv.HoloMap(interpolation_dict, kdims=["number of nodes", "kind of interpolation"])
).opts(width=600, height=800, legend_position="bottom")
```

[6]:
```
:HoloMap   [number of nodes,kind of interpolation]
   :Overlay
      .Spread.Uncertainties              :Spread   [relative measurement time]  ␣
↪(Primary Nominal Current,Associated Uncertainty)
      .Curve.Measurements                :Curve   [relative measurement time]   (Primary␣
↪Nominal Current)
      .Curve.Interpolated_values         :Curve   [Time]   (Current)
      .Spread.Interpolated_uncertainties :Spread   [Time]   (Current,Associated␣
↪Uncertainty)
```

9.4 Return sensitivity coefficients

For (almost) all interpolation or extrapolation runs, the sensitivity coefficients can be returned by setting `returnC=True`. For details refer to the return value's documentation.

[7]:
```
# Conduct linear interpolation for the last set of interpolation nodes and display c.
interp1d_unc(x_new=x_news[-1], x=x, y=y, uy=uy, returnC=True)[-1]
```

[7]:
```
array([[ 1.    ,  0.    ,  0.    ,  0.    ],
       [ 0.8125,  0.1875,  0.    ,  0.    ],
       [ 0.625 ,  0.375 ,  0.    ,  0.    ],
       [ 0.4375,  0.5625,  0.    ,  0.    ],
       [ 0.25  ,  0.75  ,  0.    ,  0.    ],
       [ 0.0625,  0.9375,  0.    ,  0.    ],
       [ 0.    ,  0.875 ,  0.125 ,  0.    ],
       [ 0.    ,  0.6875,  0.3125,  0.    ],
       [ 0.    ,  0.5   ,  0.5   ,  0.    ],
       [ 0.    ,  0.3125,  0.6875,  0.    ],
       [ 0.    ,  0.125 ,  0.875 ,  0.    ],
       [ 0.    ,  0.    ,  0.9375,  0.0625],
       [ 0.    ,  0.    ,  0.75  ,  0.25  ],
       [ 0.    ,  0.    ,  0.5625,  0.4375],
       [ 0.    ,  0.    ,  0.375 ,  0.625 ],
       [ 0.    ,  0.    ,  0.1875,  0.8125],
       [ 0.    ,  0.    , -0.    ,  1.    ]])
```

CHAPTER

TEN

BASIC EXTRAPOLATION WITH PYDYNAMIC.UNCERTAINTY.INTERPOLATE.INTERP1D_UNC

This is the third notebook in the series to illustrate the use of our method `interp1d_unc`. We will conduct a simple interpolation with constant extrapolation outside the original data's bounds.

10.1 Preparation

First we setup our Python and plotting environment and collect all previously extracted measurement data and their visualization.

10.1.1 Setup the Python environment

[1]:
```
import holoviews as hv
import numpy as np
import pickle

from PyDynamic.uncertainty import interp1d_unc
```

10.1.2 Setup plotting environment and labels

[2]:
```
# Set one of the available plotting backends ('plotly', 'bokeh', 'maplotlib').
hv.extension("bokeh")

# Define labels and units for plots.
timestamp_labels = hv.Dimension(("Time", "relative measurement time"), unit="s")
measurement_labels = hv.Dimension(("Current", "Primary Nominal Current"), unit="A")
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

10.1.3 Load results from previous lessons

[3]:
```
try:
    y = np.load("data_points.npy")
    uy = np.load("data_points_unc.npy")
    x = np.load("time_stamps.npy")
    with open("original_plot.p", "rb") as f:
        original_plot = pickle.load(f)
except FileNotFoundError:
    print(
        "The measurement data is not available. Please execute all steps in '01 "
        "Basic measurement data pre-processing' before you proceed."
    )
```

10.2 Setup extra- and interpolation timestamps

We extend the time axis of the original values at both the upper and lower interval boundaries.

[4]:
```
# Setup list of vectors of extra- and interpolation timestamps with increasing number.
x_news = [
    np.linspace(start=x.min() - 4e-3, stop=x.max() + 4e-3, num=n_nodes)
    for n_nodes in np.arange(start=2, stop=40, step=3)
]
x_news
```

[4]:
```
[array([0.   , 0.014]),
 array([0.    , 0.0035, 0.007 , 0.0105, 0.014 ]),
 array([0.   , 0.002, 0.004, 0.006, 0.008, 0.01 , 0.012, 0.014]),
 array([0.    , 0.0014, 0.0028, 0.0042, 0.0056, 0.007 , 0.0084, 0.0098,
        0.0112, 0.0126, 0.014 ]),
 array([0.        , 0.00107692, 0.00215385, 0.00323077, 0.00430769,
        0.00538462, 0.00646154, 0.00753846, 0.00861538, 0.00969231,
        0.01076923, 0.01184615, 0.01292308, 0.014     ]),
 array([0.      , 0.000875, 0.00175 , 0.002625, 0.0035  , 0.004375,
        0.00525 , 0.006125, 0.007   , 0.007875, 0.00875 , 0.009625,
        0.0105  , 0.011375, 0.01225 , 0.013125, 0.014   ]),
 array([0.        , 0.00073684, 0.00147368, 0.00221053, 0.00294737,
        0.00368421, 0.00442105, 0.00515789, 0.00589474, 0.00663158,
        0.00736842, 0.00810526, 0.00884211, 0.00957895, 0.01031579,
        0.01105263, 0.01178947, 0.01252632, 0.01326316, 0.014     ]),
 array([0.        , 0.00063636, 0.00127273, 0.00190909, 0.00254545,
        0.00318182, 0.00381818, 0.00445455, 0.00509091, 0.00572727,
        0.00636364, 0.007     , 0.00763636, 0.00827273, 0.00890909,
        0.00954545, 0.01018182, 0.01081818, 0.01145455, 0.01209091,
        0.01272727, 0.01336364, 0.014     ]),
 array([0.     , 0.00056, 0.00112, 0.00168, 0.00224, 0.0028 , 0.00336,
        0.00392, 0.00448, 0.00504, 0.0056 , 0.00616, 0.00672, 0.00728,
        0.00784, 0.0084 , 0.00896, 0.00952, 0.01008, 0.01064, 0.0112 ,
        0.01176, 0.01232, 0.01288, 0.01344, 0.014  ]),
 array([0.    , 0.0005, 0.001 , 0.0015, 0.002 , 0.0025, 0.003 , 0.0035,
        0.004 , 0.0045, 0.005 , 0.0055, 0.006 , 0.0065, 0.007 , 0.0075,
```

(continues on next page)

(continued from previous page)

```
       0.008 , 0.0085, 0.009 , 0.0095, 0.01  , 0.0105, 0.011 , 0.0115,
       0.012 , 0.0125, 0.013 , 0.0135, 0.014 ]),
 array([0.        , 0.00045161, 0.00090323, 0.00135484, 0.00180645,
       0.00225806, 0.00270968, 0.00316129, 0.0036129 , 0.00406452,
       0.00451613, 0.00496774, 0.00541935, 0.00587097, 0.00632258,
       0.00677419, 0.00722581, 0.00767742, 0.00812903, 0.00858065,
       0.00903226, 0.00948387, 0.00993548, 0.0103871 , 0.01083871,
       0.01129032, 0.01174194, 0.01219355, 0.01264516, 0.01309677,
       0.01354839, 0.014     ]),
 array([0.        , 0.00041176, 0.00082353, 0.00123529, 0.00164706,
       0.00205882, 0.00247059, 0.00288235, 0.00329412, 0.00370588,
       0.00411765, 0.00452941, 0.00494118, 0.00535294, 0.00576471,
       0.00617647, 0.00658824, 0.007     , 0.00741176, 0.00782353,
       0.00823529, 0.00864706, 0.00905882, 0.00947059, 0.00988235,
       0.01029412, 0.01070588, 0.01111765, 0.01152941, 0.01194118,
       0.01235294, 0.01276471, 0.01317647, 0.01358824, 0.014     ]),
 array([0.        , 0.00037838, 0.00075676, 0.00113514, 0.00151351,
       0.00189189, 0.00227027, 0.00264865, 0.00302703, 0.00340541,
       0.00378378, 0.00416216, 0.00454054, 0.00491892, 0.0052973 ,
       0.00567568, 0.00605405, 0.00643243, 0.00681081, 0.00718919,
       0.00756757, 0.00794595, 0.00832432, 0.0087027 , 0.00908108,
       0.00945946, 0.00983784, 0.01021622, 0.01059459, 0.01097297,
       0.01135135, 0.01172973, 0.01210811, 0.01248649, 0.01286486,
       0.01324324, 0.01362162, 0.014     ])]
```

10.3 Conduct and visualize extrapolation

There are several options, how to determine values in the extrapolation range. Those options are available for the extrapolation of measurement values as well as associated uncertainties independently. The provided parameters are `fill_value` and `fill_unc`. All of those methods require the parameter `bounds_error=False` to explicitly express the desire to extrapolate. Otherwise a *ValueError* will be thrown. In the following we demonstrate the different methods:

- *Extrapolate based on original data*
- *Extrapolate based on additional input*

10.3.1 Extrapolate based on original data

To conduct the simplest extrapolation, we choose *extrapolate* as both `fill_value` and `fill_unc` to fill the extrapolation range with the next adjacent original values. This results in a constant extrapolation based on the original values' at the interval bounds.

[5]:
```
# Create plots of extra- and interpolated values and uncertainties for the increasing
# number of nodes.
interpolation_dict = {}
for x_new in x_news:
    # Conduct the actual extra- and interpolation for the current set of nodes.
    x_new, y_new, uy_new = interp1d_unc(
        x_new=x_new,
```

(continues on next page)

(continued from previous page)

```
        x=x,
        y=y,
        uy=uy,
        kind="cubic",
        bounds_error=False,
        fill_value="extrapolate",
        fill_unc="extrapolate",
    )
    print(x_new, y_new, uy_new)

    # Create plot of the extra- and interpolated values.
    curve_interp = hv.Curve(
        (x_new, y_new),
        timestamp_labels,
        measurement_labels,
        label="interpolated values",
    )

    # Create plot of the extra- and interpolated uncertainties.
    interp_uncertainties = hv.Spread(
        (x_new, y_new, uy_new),
        vdims=[measurement_labels, "Associated Uncertainty"],
        kdims=timestamp_labels,
        label="interpolated uncertainties",
    )

    interpolation_dict[x_new.size] = curve_interp * interp_uncertainties
```

```
[0.    0.014] [2.47691988 1.41385091] [0.0371538  0.02120776]
[0.     0.0035 0.007  0.0105 0.014 ] [2.47691988 2.47691988 4.30061078 1.41385091 1.
↪41385091] [0.0371538  0.0371538  0.04827905 0.02120776 0.02120776]
[0.    0.002 0.004 0.006 0.008 0.01  0.012 0.014] [2.47691988 2.47691988 2.47691988 4.
↪1246116  3.95322654 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.06186917 0.0592984  0.
↪02120776
 0.02120776 0.02120776]
[0.     0.0014 0.0028 0.0042 0.0056 0.007  0.0084 0.0098 0.0112 0.0126
 0.014 ] [2.47691988 2.47691988 2.47691988 2.70790346 3.92303416 4.30061078
 3.6523559  1.78999209 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.03596855 0.06624638 0.04827905
 0.06359732 0.02530684 0.02120776 0.02120776 0.02120776]
[0.         0.00107692 0.00215385 0.00323077 0.00430769 0.00538462
 0.00646154 0.00753846 0.00861538 0.00969231 0.01076923 0.01184615
 0.01292308 0.014     ] [2.47691988 2.47691988 2.47691988 2.47691988 2.82682583 3.
↪78589296
 4.26650548 4.18296586 3.4495766  1.98064018 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.03879058 0.
↪06650832
 0.05395581 0.05206813 0.06389044 0.03090299 0.02120776 0.02120776
 0.02120776 0.02120776]
[0.       0.000875 0.00175  0.002625 0.0035   0.004375 0.00525  0.006125
 0.007    0.007875 0.00875  0.009625 0.0105   0.011375 0.01225  0.013125
```

(continues on next page)

(continued from previous page)

```
 0.014   ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.89916327
 3.69041759 4.17288882 4.30061078 4.02761732 3.30794225 2.09561941
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.0371538  0.04116281
 0.06570834 0.05982792 0.04827905 0.05736188 0.06309699 0.03458366
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776]
[0.         0.00073684 0.00147368 0.00221053 0.00294737 0.00368421
 0.00442105 0.00515789 0.00589474 0.00663158 0.00736842 0.00810526
 0.00884211 0.00957895 0.01031579 0.01105263 0.01178947 0.01252632
 0.01326316 0.014     ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.
↪47691988
 2.94776256 3.6208712  4.07844023 4.29301995 4.23716065 3.88341267
 3.20432629 2.17245183 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.0371538
 0.04292271 0.06468969 0.06339827 0.05134839 0.04994423 0.06077502
 0.0620627  0.03706643 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776]
[0.         0.00063636 0.00127273 0.00190909 0.00254545 0.00318182
 0.00381818 0.00445455 0.00509091 0.00572727 0.00636364 0.007
 0.00763636 0.00827273 0.00890909 0.00954545 0.01018182 0.01081818
 0.01145455 0.01209091 0.01272727 0.01336364 0.014     ] [2.47691988 2.47691988 2.
↪47691988 2.47691988 2.47691988 2.47691988
 2.47691988 2.98264639 3.56817364 3.99479809 4.24483779 4.30061078
 4.14443512 3.75862884 3.12551001 2.22739665 1.41385091 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.04423834 0.06369699 0.06532009 0.05563395 0.04827905
 0.0535279  0.06266537 0.06104178 0.03883156 0.02120776 0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776]
[0.      0.00056 0.00112 0.00168 0.00224 0.0028  0.00336 0.00392 0.00448
 0.00504 0.0056  0.00616 0.00672 0.00728 0.00784 0.0084  0.00896 0.00952
 0.01008 0.01064 0.0112  0.01176 0.01232 0.01288 0.01344 0.014  ] [2.47691988 2.47691988␣
↪2.47691988 2.47691988 2.47691988 2.47691988
 2.47691988 2.47691988 3.00889661 3.52694483 3.92303416 4.18511483
 4.3011371  4.2590512  4.04680739 3.6523559  3.06364699 2.26863088
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.0371538  0.04524727 0.06279743 0.06624638 0.05922602
 0.05023311 0.04912985 0.05679877 0.06359732 0.06010796 0.04014406
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776]
[0.     0.0005 0.001  0.0015 0.002  0.0025 0.003  0.0035 0.004  0.0045
 0.005  0.0055 0.006  0.0065 0.007  0.0075 0.008  0.0085 0.009  0.0095
 0.01   0.0105 0.011  0.0115 0.012  0.0125 0.013  0.0135 0.014 ] [2.47691988 2.47691988␣
↪2.47691988 2.47691988 2.47691988 2.47691988
 2.47691988 2.47691988 2.47691988 3.02936253 3.49384322 3.86178517
 4.1246116  4.27374573 4.30061078 4.19662998 3.95322654 3.5618237
 3.01384466 2.30071266 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.0371538  0.0371538  0.04604122 0.0620018  0.06658834
 0.06186917 0.05332441 0.04827905 0.05153456 0.0592984  0.06395319
```

(continues on next page)

(continued from previous page)

```
 0.05927581 0.04115559 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776]
[0.         0.00045161 0.00090323 0.00135484 0.00180645 0.00225806
 0.00270968 0.00316129 0.0036129  0.00406452 0.00451613 0.00496774
 0.00541935 0.00587097 0.00632258 0.00677419 0.00722581 0.00767742
 0.00812903 0.00858065 0.00903226 0.00948387 0.00993548 0.0103871
 0.01083871 0.01129032 0.01174194 0.01219355 0.01264516 0.01309677
 0.01354839 0.014     ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.
↪47691988
 2.47691988 2.47691988 2.47691988 2.55284495 3.04576482 3.46669813
 3.80932493 4.06732527 4.23437922 4.30416683 4.27036816 4.12666327
 3.8667322  3.48425502 2.97291179 2.32638255 1.53834737 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.0371538  0.0371538  0.03558371 0.04668041 0.06130304
 0.06658881 0.06371279 0.056357   0.04965463 0.0487502  0.0541722
 0.06108156 0.0639662  0.05854034 0.04195782 0.02094216 0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776]
[0.         0.00041176 0.00082353 0.00123529 0.00164706 0.00205882
 0.00247059 0.00288235 0.00329412 0.00370588 0.00411765 0.00452941
 0.00494118 0.00535294 0.00576471 0.00617647 0.00658824 0.007
 0.00741176 0.00782353 0.00823529 0.00864706 0.00905882 0.00947059
 0.00988235 0.01029412 0.01070588 0.01111765 0.01152941 0.01194118
 0.01235294 0.01276471 0.01317647 0.01358824 0.014     ] [2.47691988 2.47691988 2.
↪47691988 2.47691988 2.47691988 2.47691988
 2.47691988 2.47691988 2.47691988 2.47691988 2.61436603 3.05920366
 3.44404325 3.76409454 4.01456722 4.19067101 4.28761563 4.30061078
 4.22486619 4.05559155 3.7879966  3.41729104 2.93868457 2.34738693
 1.63860781 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091] [0.0371538  0.0371538  0.
↪0371538  0.0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.0371538  0.0371538  0.0371538  0.03519329 0.04720517
 0.06068931 0.06639095 0.06495398 0.0589396  0.05196268 0.04827905
 0.05042291 0.05653217 0.06230174 0.06377662 0.05789089 0.04260903
 0.02203892 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776]
[0.         0.00037838 0.00075676 0.00113514 0.00151351 0.00189189
 0.00227027 0.00264865 0.00302703 0.00340541 0.00378378 0.00416216
 0.00454054 0.00491892 0.0052973  0.00567568 0.00605405 0.00643243
 0.00681081 0.00718919 0.00756757 0.00794595 0.00832432 0.0087027
 0.00908108 0.00945946 0.00983784 0.01021622 0.01059459 0.01097297
 0.01135135 0.01172973 0.01210811 0.01248649 0.01286486 0.01324324
 0.01362162 0.014     ] [2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.
↪47691988
 2.47691988 2.47691988 2.47691988 2.47691988 2.47691988 2.66520347
 3.07041516 3.42485447 3.7248044  3.96654794 4.14636809 4.26054784
 4.30537019 4.27711814 4.17207468 3.9865228  3.71674551 3.35902579
 2.90964664 2.36489106 1.72104205 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091 1.41385091
 1.41385091 1.41385091] [0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.0371538
 0.0371538  0.0371538  0.0371538  0.0371538  0.0371538  0.0354263
```

(continues on next page)

(continued from previous page)

```
 0.04764322 0.06014857 0.06608022 0.06575736 0.06101105 0.05444555
 0.04931408 0.04854929 0.05248837 0.05847941 0.06310264 0.06346951
 0.05731598 0.04314785 0.02362907 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776 0.02120776
 0.02120776 0.02120776]
```

[6]:
```
# Visualize the interpolation result along-side the original data.
(original_plot * hv.HoloMap(interpolation_dict, kdims="number of nodes")).opts(
    width=600, height=320, legend_position="bottom"
)
```

[6]:
```
:HoloMap   [number of nodes]
   :Overlay
      .Spread.Uncertainties              :Spread   [relative measurement time]  ␣
↪(Primary Nominal Current,Associated Uncertainty)
      .Curve.Measurements                :Curve   [relative measurement time]   (Primary␣
↪Nominal Current)
      .Curve.Interpolated_values         :Curve   [Time]   (Current)
      .Spread.Interpolated_uncertainties :Spread   [Time]   (Current,Associated␣
↪Uncertainty)
```

10.3.2 Extrapolate based on additional input

Besides using the values at the interpolation interval bounds, it is possible to specify different constant values for the extrapolation range. This can be done via a single float for both ranges or via a two-tuple. In the latter case the first value will be used for the lower range.

[7]:
```
# Create plots of extra- and interpolated values and uncertainties for the increasing
# number of nodes.
interpolation_dict = {}
for x_new in x_news:
    # Conduct the actual extra- and interpolation for the current set of nodes.
    x_new, y_new, uy_new = interp1d_unc(
        x_new=x_new,
        x=x,
        y=y,
        uy=uy,
        kind="cubic",
        bounds_error=False,
        fill_value=2,
        fill_unc=[0.1, 0.4],
    )
    print(x_new, y_new, uy_new)

    # Create plot of the extra- and interpolated values.
    curve_interp = hv.Curve(
        (x_new, y_new),
        timestamp_labels,
        measurement_labels,
        label="interpolated values",
```

(continues on next page)

(continued from previous page)

```
    )

    # Create plot of the extra- and interpolated uncertainties.
    interp_uncertainties = hv.Spread(
        (x_new, y_new, uy_new),
        vdims=[measurement_labels, "Associated Uncertainty"],
        kdims=timestamp_labels,
        label="interpolated uncertainties",
    )

    interpolation_dict[x_new.size] = curve_interp * interp_uncertainties
```

```
[0.    0.014] [2. 2.] [0.1 0.4]
[0.     0.0035 0.007  0.0105 0.014 ] [2.         2.         4.30061078 2.         2.
 ] [0.1        0.1        0.04827905 0.4        0.4       ]
[0.    0.002 0.004 0.006 0.008 0.01  0.012 0.014] [2.         2.         2.47691988 4.
1246116  3.95322654 1.41385091
 2.         2.        ] [0.1        0.1        0.0371538  0.06186917 0.0592984  0.
02120776
 0.4        0.4       ]
[0.     0.0014 0.0028 0.0042 0.0056 0.007  0.0084 0.0098 0.0112 0.0126
 0.014 ] [2.         2.         2.         2.70790346 3.92303416 4.30061078
 3.6523559  1.78999209 2.         2.         2.        ] [0.1        0.1        0.1
  0.03596855 0.06624638 0.04827905
 0.06359732 0.02530684 0.4        0.4        0.4       ]
[0.         0.00107692 0.00215385 0.00323077 0.00430769 0.00538462
 0.00646154 0.00753846 0.00861538 0.00969231 0.01076923 0.01184615
 0.01292308 0.014     ] [2.         2.         2.         2.         2.82682583 3.
78589296
 4.26650548 4.18296586 3.4495766  1.98064018 2.         2.
 2.         2.        ] [0.1        0.1        0.1        0.1        0.03879058 0.
06650832
 0.05395581 0.05206813 0.06389044 0.03090299 0.4        0.4
 0.4        0.4       ]
[0.       0.000875 0.00175  0.002625 0.0035   0.004375 0.00525  0.006125
 0.007    0.007875 0.00875  0.009625 0.0105   0.011375 0.01225  0.013125
 0.014   ] [2.         2.         2.         2.         2.         2.89916327
 3.69041759 4.17288882 4.30061078 4.02761732 3.30794225 2.09561941
 2.         2.         2.         2.         2.        ] [0.1        0.1        0.1
  0.1        0.1        0.04116281
 0.06570834 0.05982792 0.04827905 0.05736188 0.06309699 0.03458366
 0.4        0.4        0.4        0.4        0.4       ]
[0.         0.00073684 0.00147368 0.00221053 0.00294737 0.00368421
 0.00442105 0.00515789 0.00589474 0.00663158 0.00736842 0.00810526
 0.00884211 0.00957895 0.01031579 0.01105263 0.01178947 0.01252632
 0.01326316 0.014     ] [2.         2.         2.         2.         2.         2.
 2.94776256 3.6208712  4.07844023 4.29301995 4.23716065 3.88341267
 3.20432629 2.17245183 2.         2.         2.         2.
 2.         2.        ] [0.1        0.1        0.1        0.1        0.1        0.1
 0.04292271 0.06468969 0.06339827 0.05134839 0.04994423 0.06077502
 0.0620627  0.03706643 0.4        0.4        0.4        0.4
 0.4        0.4       ]
```

(continues on next page)

(continued from previous page)

```
[0.         0.00063636 0.00127273 0.00190909 0.00254545 0.00318182
 0.00381818 0.00445455 0.00509091 0.00572727 0.00636364 0.007
 0.00763636 0.00827273 0.00890909 0.00954545 0.01018182 0.01081818
 0.01145455 0.01209091 0.01272727 0.01336364 0.014     ] [2.         2.         2.
↪  2.         2.         2.
 2.         2.98264639 3.56817364 3.99479809 4.24483779 4.30061078
 4.14443512 3.75862884 3.12551001 2.22739665 2.         2.
 2.         2.         2.         2.         2.        ] [0.1        0.1        0.1
↪  0.1        0.1        0.1
 0.1        0.04423834 0.06369699 0.06532009 0.05563395 0.04827905
 0.0535279  0.06266537 0.06104178 0.03883156 0.4        0.4
 0.4        0.4        0.4        0.4        0.4       ]
[0.      0.00056 0.00112 0.00168 0.00224 0.0028  0.00336 0.00392 0.00448
 0.00504 0.0056  0.00616 0.00672 0.00728 0.00784 0.0084  0.00896 0.00952
 0.01008 0.01064 0.0112  0.01176 0.01232 0.01288 0.01344 0.014  ] [2.         2.
↪2.         2.         2.         2.
 2.         2.         3.00889661 3.52694483 3.92303416 4.18511483
 4.3011371  4.2590512  4.04680739 3.6523559  3.06364699 2.26863088
 2.         2.         2.         2.         2.         2.
 2.         2.        ] [0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.04524727 0.06279743 0.06624638 0.05922602
 0.05023311 0.04912985 0.05679877 0.06359732 0.06010796 0.04014406
 0.4        0.4        0.4        0.4        0.4        0.4
 0.4        0.4       ]
[0.     0.0005 0.001  0.0015 0.002  0.0025 0.003  0.0035 0.004  0.0045
 0.005  0.0055 0.006  0.0065 0.007  0.0075 0.008  0.0085 0.009  0.0095
 0.01   0.0105 0.011  0.0115 0.012  0.0125 0.013  0.0135 0.014 ] [2.         2.
↪2.         2.         2.         2.
 2.         2.         2.47691988 3.02936253 3.49384322 3.86178517
 4.1246116  4.27374573 4.30061078 4.19662998 3.95322654 3.5618237
 3.01384466 2.30071266 1.41385091 2.         2.         2.
 2.         2.         2.         2.         2.        ] [0.1        0.1        0.1
↪  0.1        0.1        0.1
 0.1        0.1        0.0371538  0.04604122 0.0620018  0.06658834
 0.06186917 0.05332441 0.04827905 0.05153456 0.0592984  0.06395319
 0.05927581 0.04115559 0.02120776 0.4        0.4        0.4
 0.4        0.4        0.4        0.4        0.4       ]
[0.         0.00045161 0.00090323 0.00135484 0.00180645 0.00225806
 0.00270968 0.00316129 0.0036129  0.00406452 0.00451613 0.00496774
 0.00541935 0.00587097 0.00632258 0.00677419 0.00722581 0.00767742
 0.00812903 0.00858065 0.00903226 0.00948387 0.00993548 0.0103871
 0.01083871 0.01129032 0.01174194 0.01219355 0.01264516 0.01309677
 0.01354839 0.014     ] [2.         2.         2.         2.         2.         2.
 2.         2.         2.         2.55284495 3.04576482 3.46669813
 3.80932493 4.06732527 4.23437922 4.30416683 4.27036816 4.12666327
 3.8667322  3.48425502 2.97291179 2.32638255 1.53834737 2.
 2.         2.         2.         2.         2.         2.
 2.         2.        ] [0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.1        0.03558371 0.04668041 0.06130304
 0.06658881 0.06371279 0.056357   0.04965463 0.0487502  0.0541722
 0.06108156 0.0639662  0.05854034 0.04195782 0.02094216 0.4
 0.4        0.4        0.4        0.4        0.4        0.4
```

(continues on next page)

(continued from previous page)

```
 0.4        0.4       ]
[0.         0.00041176 0.00082353 0.00123529 0.00164706 0.00205882
 0.00247059 0.00288235 0.00329412 0.00370588 0.00411765 0.00452941
 0.00494118 0.00535294 0.00576471 0.00617647 0.00658824 0.007
 0.00741176 0.00782353 0.00823529 0.00864706 0.00905882 0.00947059
 0.00988235 0.01029412 0.01070588 0.01111765 0.01152941 0.01194118
 0.01235294 0.01276471 0.01317647 0.01358824 0.014     ] [2.         2.         2.
↪  2.         2.         2.
 2.         2.         2.         2.         2.61436603 3.05920366
 3.44404325 3.76409454 4.01456722 4.19067101 4.28761563 4.30061078
 4.22486619 4.05559155 3.7879966  3.41729104 2.93868457 2.34738693
 1.63860781 2.         2.         2.         2.         2.
 2.         2.         2.         2.         2.        ] [0.1        0.1        0.1
↪  0.1        0.1        0.1
 0.1        0.1        0.1        0.1        0.03519329 0.04720517
 0.06068931 0.06639095 0.06495398 0.0589396  0.05196268 0.04827905
 0.05042291 0.05653217 0.06230174 0.06377662 0.05789089 0.04260903
 0.02203892 0.4        0.4        0.4        0.4        0.4
 0.4        0.4        0.4        0.4        0.4       ]
[0.         0.00037838 0.00075676 0.00113514 0.00151351 0.00189189
 0.00227027 0.00264865 0.00302703 0.00340541 0.00378378 0.00416216
 0.00454054 0.00491892 0.0052973  0.00567568 0.00605405 0.00643243
 0.00681081 0.00718919 0.00756757 0.00794595 0.00832432 0.0087027
 0.00908108 0.00945946 0.00983784 0.01021622 0.01059459 0.01097297
 0.01135135 0.01172973 0.01210811 0.01248649 0.01286486 0.01324324
 0.01362162 0.014     ] [2.         2.         2.         2.         2.         2.
 2.         2.         2.         2.         2.         2.66520347
 3.07041516 3.42485447 3.7248044  3.96654794 4.14636809 4.26054784
 4.30537019 4.27711814 4.17207468 3.9865228  3.71674551 3.35902579
 2.90964664 2.36489106 1.72104205 2.         2.         2.
 2.         2.         2.         2.         2.         2.
 2.         2.        ] [0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.0354263
 0.04764322 0.06014857 0.06608022 0.06575736 0.06101105 0.05444555
 0.04931408 0.04854929 0.05248837 0.05847941 0.06310264 0.06346951
 0.05731598 0.04314785 0.02362907 0.4        0.4        0.4
 0.4        0.4        0.4        0.4        0.4        0.4
 0.4        0.4       ]
```

[8]:
```
# Visualize the interpolation result along-side the original data.
(original_plot * hv.HoloMap(interpolation_dict, kdims="number of nodes")).opts(
    width=600, height=350, legend_position="bottom"
)
```

[8]:
```
:HoloMap   [number of nodes]
   :Overlay
      .Spread.Uncertainties              :Spread   [relative measurement time]
↪(Primary Nominal Current,Associated Uncertainty)
      .Curve.Measurements                :Curve    [relative measurement time]   (Primary
↪Nominal Current)
      .Curve.Interpolated_values         :Curve    [Time]   (Current)
      .Spread.Interpolated_uncertainties :Spread   [Time]   (Current,Associated
```

(continues on next page)

(continued from previous page)

```
↪Uncertainty)
```

CHAPTER

ELEVEN

CUBIC SPLINE INTERPOLATION WITH PYDYNAMIC.UNCERTAINTY.INTERPOLATE.INTERP1D_UNC

Interpolate a non-equidistant sine signal using cubic / bspline method with uncertainty propagation. Comparing the resulting uncertainties to a Monte-Carlo experiment yields good overlap.

[1]:
```
import numpy as np
import plotly.graph_objs as go
from PyDynamic.uncertainty.interpolate import interp1d_unc
from scipy.interpolate import interp1d
```

11.1 Create non-equidistant sine-signal

[2]:
```
n_nodes = 10
t = np.cumsum(range(1, n_nodes))
x = np.sin(t)
ux = np.full_like(x, 0.2)
t, x, ux
```

[2]:
```
(array([ 1,  3,  6, 10, 15, 21, 28, 36, 45]),
 array([ 0.84147098,  0.14112001, -0.2794155 , -0.54402111,  0.65028784,
         0.83665564,  0.27090579, -0.99177885,  0.85090352]),
 array([0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2]))
```

11.2 Interpolate with PyDynamic

[3]:
```
ti = np.linspace(np.min(t), np.max(t), 60)
ti, xi, uxi = interp1d_unc(ti, t, x, ux, kind="cubic")
ti, xi, uxi
```

[3]:
```
(array([ 1.        ,  1.74576271,  2.49152542,  3.23728814,  3.98305085,
         4.72881356,  5.47457627,  6.22033898,  6.96610169,  7.71186441,
         8.45762712,  9.20338983,  9.94915254, 10.69491525, 11.44067797,
        12.18644068, 12.93220339, 13.6779661 , 14.42372881, 15.16949153,
        15.91525424, 16.66101695, 17.40677966, 18.15254237, 18.89830508,
        19.6440678 , 20.38983051, 21.13559322, 21.88135593, 22.62711864,
        23.37288136, 24.11864407, 24.86440678, 25.61016949, 26.3559322 ,
        27.10169492, 27.84745763, 28.59322034, 29.33898305, 30.08474576,
```

(continues on next page)

(continued from previous page)

```
       30.83050847, 31.57627119, 32.3220339 , 33.06779661, 33.81355932,
       34.55932203, 35.30508475, 36.05084746, 36.79661017, 37.54237288,
       38.28813559, 39.03389831, 39.77966102, 40.52542373, 41.27118644,
       42.01694915, 42.76271186, 43.50847458, 44.25423729, 45.        ]),
 array([ 0.84147098,  0.5134559 ,  0.26887907,  0.09049061, -0.03895932,
       -0.13672062, -0.22004314, -0.30603185, -0.40015754, -0.48792672,
       -0.55288113, -0.57856251, -0.54851261, -0.45094022, -0.29697194,
       -0.10475754,  0.10755139,  0.32180327,  0.51984651,  0.68357697,
        0.80217466,  0.87988176,  0.92280604,  0.93705524,  0.92873714,
        0.90395947,  0.86883   ,  0.82944871,  0.78981465,  0.74899325,
        0.70534253,  0.65722049,  0.60298513,  0.54099446,  0.4696065 ,
        0.38717924,  0.29207069,  0.18306475,  0.06213925, -0.0672781 ,
       -0.20175235, -0.33784857, -0.4721318 , -0.6011671 , -0.72151951,
       -0.8297541 , -0.92243592, -0.99613001, -1.04740144, -1.07281526,
       -1.06893652, -1.03233027, -0.95956157, -0.84719547, -0.69179702,
       -0.48993128, -0.23816331,  0.06694186,  0.42881915,  0.85090352]),
 array([0.2       , 0.16837861, 0.19741966, 0.1962444 , 0.17596306,
       0.16866971, 0.18622166, 0.20285793, 0.19817887, 0.1825655 ,
       0.17473489, 0.18389472, 0.1992225 , 0.20369264, 0.19571168,
       0.18332587, 0.17637735, 0.1804066 , 0.19195061, 0.20145338,
       0.20182147, 0.19446415, 0.18421328, 0.1767848 , 0.17639785,
       0.18326204, 0.19338735, 0.20091735, 0.20172299, 0.19643529,
       0.18769373, 0.17895702, 0.17368016, 0.17410604, 0.1801646 ,
       0.1894922 , 0.1985253 , 0.20383972, 0.20425977, 0.20029238,
       0.1931453 , 0.18453319, 0.17654847, 0.17142144, 0.17103588,
       0.17629981, 0.18680117, 0.20105082, 0.21705456, 0.23277969,
       0.24637648, 0.25624082, 0.26102733, 0.25968413, 0.25156834,
       0.23674281, 0.21668996, 0.19594729, 0.18494793, 0.2       ]))
```

11.3 Interpolate with Monte Carlo

[4]:
```
X_mc = []
for i in range(2000):
    interp_x = interp1d(t, x + ux * np.random.randn(len(x)), kind="cubic")
    xm = interp_x(ti)
    X_mc.append(xm)
x_mc = np.mean(X_mc, axis=0)
ux_mc = np.std(X_mc, axis=0);
```

11.4 Compare results

11.4.1 Visual

[5]:
```
# NBVAL_IGNORE_OUTPUT
# interpolated signal
fig = go.Figure(
    [
```

(continues on next page)

(continued from previous page)

```
        go.Scatter(
            name="PyDynamic interpolation",
            x=ti,
            y=xi,
            mode="lines",
            marker_color="rgba(168, 68, 68, 0.8)",
        ),
        go.Scatter(
            hoverinfo="x",
            x=ti,
            y=xi + uxi,
            mode="lines",
            line=dict(width=0),
            showlegend=False,
        ),
        go.Scatter(
            hoverinfo="x",
            x=ti,
            y=xi - uxi,
            line=dict(width=0),
            mode="lines",
            fillcolor="rgba(168, 68, 68, 0.3)",
            fill="tonexty",
            showlegend=False,
        ),
        go.Scatter(
            name="Monte Carlo interpolation",
            x=ti,
            y=x_mc,
            mode="lines",
            marker_color="rgba(68, 68, 168, 0.8)",
        ),
        go.Scatter(
            hoverinfo="x",
            x=ti,
            y=x_mc + ux_mc,
            mode="lines",
            line=dict(width=0),
            showlegend=False,
        ),
        go.Scatter(
            hoverinfo="x",
            x=ti,
            y=x_mc - ux_mc,
            line=dict(width=0),
            mode="lines",
            fillcolor="rgba(68, 68, 168, 0.3)",
            fill="tonexty",
            showlegend=False,
        ),
        go.Scatter(
            name="original",
```

(continues on next page)

(continued from previous page)

```
            x=t,
            y=x,
            mode="lines",
            marker_color="rgba(68, 168, 68, 0.8)",
        ),
        go.Scatter(
            hoverinfo="x",
            x=t,
            y=x + ux,
            mode="lines",
            line=dict(width=0),
            showlegend=False,
        ),
        go.Scatter(
            hoverinfo="x",
            x=t,
            y=x - ux,
            line=dict(width=0),
            mode="lines",
            fillcolor="rgba(68, 168, 68, 0.3)",
            fill="tonexty",
            showlegend=False,
        ),
    ]
)
fig.update_layout(
    yaxis_title="Amplitude (a.u.)",
    xaxis_title="Time (a.u.)",
    title="Comparison between MonteCarlo simulation and PyDynamic interpolation",
    hovermode="x",
)
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

11.4.2 Numerical

[6]:
```
tolerance = 1e-1
```

[7]:
```
np.allclose(xi, x_mc, atol=tolerance, rtol=tolerance)
```

[7]:
```
True
```

[8]:
```
np.allclose(uxi, ux_mc, atol=tolerance, rtol=tolerance)
```

[8]:
```
True
```

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search