Global and target analysis of time-resolved fluorescence data

From a single experiment on Photosystem I

Reference

DOI:

Imports

In [ ]:
```
# imports
from pyglotaran_extras.io import setup_case_study

from glotaran.io import load_dataset, load_model, load_parameters, save_r
from glotaran.optimization.optimize import optimize
from glotaran.project.scheme import Scheme
```

In [ ]:
```
# extra import for plotting
import matplotlib.pyplot as plt
from pyglotaran_extras.plotting.plot_overview import plot_overview
from pyglotaran_extras.plotting.style import PlotStyle
```

Location of data files

Note that for the sake of reproduciblity we seperate the model used for global analysis from the model used for target analysis.

We will use the same data files for both analyses.

Note: when comparing the results with previous results, we only use at the target analysis.

In [ ]:
```
# paths

GA_MODEL_PATH = "models/model.yaml"
GA_PARAMETERS_PATH = "models/parameters.yaml"

results_folder, script_folder = setup_case_study(
    output_folder_name="pyglotaran_examples_results",
)

data_path = script_folder.joinpath("data/data.ascii")
model_path = script_folder.joinpath(GA_MODEL_PATH)
parameter_path = script_folder.joinpath(GA_PARAMETERS_PATH)
```

Loading in data files

In [ ]:
```
# load the data
experiment_data = {"dataset1": load_dataset(data_path)}
```

```
# load the (model) scheme (library)
model = load_model(model_path, format_name="yml")
# load the parameters
parameters = load_parameters(parameter_path)
# attach the data to the scheme
scheme = Scheme(
    model,
    parameters,
    experiment_data,
    maximum_number_function_evaluations=6,  # 6 for TRF, 46 for LM
    # optimization_method="Levenberg-Marquardt", #lm needs nfev=46
)

scheme.validate()
```

Optimizing the global fit

In [ ]:
```
# Run the optimizer
result = optimize(scheme, verbose=True)
```

In [ ]:
```
result  # shows result as markdown formatted table
```

Inspection of results

In [ ]:
```
# Inspect the results
res = result.data["dataset1"]

res
```

Plotting of results

This requires pyglotaran-extras

In [ ]:
```
# Plot the results

# %% Set subsequent plots to the glotaran style
plot_style = PlotStyle()
plt.rc("axes", prop_cycle=plot_style.cycler)

# %%
fig, _ = plot_overview(res, linlog=False)
# note species concentration plot still needs work to match styles betwee

# %%
figure_output_path = results_folder / f"plot_overview_{results_folder.nam
fig.savefig(str(figure_output_path), bbox_inches="tight")
print(results_folder)
```

Target analysis

In [ ]:
```
TA_MODEL_PATH = "models/model-target.yaml"
TA_PARAMETERS_PATH = "models/parameters-target.yaml"

results_folder, script_folder = setup_case_study(
```

```
    output_folder_name="pyglotaran_examples_results",
)
results_folder / "study_fluorescence"

data_path = script_folder.joinpath("data/data.ascii")
model_path = script_folder.joinpath(TA_MODEL_PATH)
parameter_path = script_folder.joinpath(TA_PARAMETERS_PATH)

experiment_data = {"dataset1": load_dataset(data_path)}
model = load_model(model_path, format_name="yml")
parameters = load_parameters(parameter_path)
scheme = Scheme(
    model,
    parameters,
    experiment_data,
    maximum_number_function_evaluations=6,  # 6 for TRF, 46 for LM
    # optimization_method="Levenberg-Marquardt", #lm needs nfev=46
)

result = optimize(scheme, verbose=True)
```

In [ ]:
```
result
```

In [ ]:
```
save_result(result, results_folder / "result.yml", allow_overwrite=True)
```

In [ ]:
```
res = result.data["dataset1"]
```

In [ ]:
```
# Plot the results

# %% Set subsequent plots to the glotaran style
plot_style = PlotStyle()
plt.rc("axes", prop_cycle=plot_style.cycler)

# %%
fig, _ = plot_overview(res, linlog=False)
# note species concentration plot still needs work to match styles betwee

# %%
figure_output_path = results_folder / f"plot_overview_{results_folder.nam
fig.savefig(str(figure_output_path), bbox_inches="tight")
print(results_folder)
```