Numerical Experiments for Validating Prediction Algorithms Report

Draft 1

Jin Whan Bae and Gwendolyn Chee

2017-09-23

Introduction

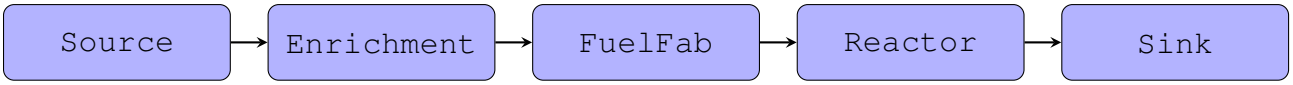
The Demand-Driven Cycamore Archetype project (NEUP-FY16-10512) aims to develop CYCLUS in-situ demand-driven deployment capabilities through non-optimizing, deterministic-optimizing and stochastic-optimizing prediction algorithms.

These prediction models are being developed by the University of South Carolina. In this report, we discuss numerical experiments for testing the non-optimizing, deterministic and stochastic optimizing methods.

1 Once through Nuclear Fuel Cycle

This section evaluates the required tests for each method assuming a once-through fuel cycle.

Figure 1: Flow Chart of Once through Nuclear Fuel Cycle



2 Input File Format

We assume the module to be an `INSTITUTION`, since it governs deployment and decommission of facilities.

The user would only have to define the reactor deployment (either directly or indirectly, see below) and the remaining fuel facilities , both front end and back end, would be 'connected' and

deployed. If an input file does not have the necessary 'connections' for the reactor to receive fuel, it will throw an error.

We suggest an example schema: First, the institution would define a prototype, defined in the input file, to be deployed. There can be multiple reactors.

```
<institution>
        <name>NO_ddd</name>
         <config><NO_ddd/></config>
        <reactor_list>
          <val>lwr</val>
          <val>sfr</val>
          <val>mox_lwr</val>
        </reactor_list>
```

Listing 1: One-reactor fleet institution input schema

[Something About Transition Scenario Capabilities]

Then deployment of the reactors is defined, and the user is given the option to manually define the deployment(DeployInst):

```
        <deployment>
        <type>manual</type>

          <build_times>
            <val>1</val>
            <val>10</val>
            <val>20</val>
            <val>40</val>
          </build_times>
          <n_build>
            <val>3</val>
            <val>3</val>
            <val>3</val>
            <val>3</val>
          </n_build>
          <lifetimes>
            <val>960</val>
            <val>960</val>
            <val>960</val>
            <val>960</val>
          </lifetimes>
        </deployment>

    </institution>
```

Listing 2: Reactor deployment input schema

or have the institution deploy reactors according to power demand (GrowthRegion):

```
    <deployment>
    <type>growth</type>

    <growth>
      <piecewise_function>
            <piece>
              <start>0</start>
              <function>
                <type>linear</type>
                <params>1  2</params>
              </function>
            </piece>
      </piecewise_function>
    </growth>

    </deployment>

  </institution>
```

Listing 3: Reactor deployment input schema

This can be advanced to model transition scenarios, where a the power demand function is defined separately for each reactor:

```
    <institution>
    <name>NO_ddd</name>
     <config><NO_ddd/></config>
    <reactor_list>
      <val>lwr</val>
      <val>sfr</val>
      <val>mox_lwr</val>
    </reactor_list>

    <deployment>
    <type>growth</type>

    <growth>
      <piecewise_function>
            <piece>
              <reactor>lwr</reactor>
```

```
            <start>0</start>
            <function>
              <type>linear</type>
              <params>1</params>
            </function>
          </piece>
          <piece>
            <reactor>lwr</reactor>
            <start>80</start>
            <function>
              <type>linear</type>
              <params>-1</params>
          <piece>
            <reactor>sfr</reactor>
            <start>80</start>
            <function>
              <type>linear</type>
              <params>0.87</params>
            </function>
          </piece>
          <piece>
            <reactor>lwr_mox</reactor>
            <start>80</start>
            <function>
              <type>linear</type>
              <params>0.13</params>
            </function>
          </piece>
      </piecewise_function>
    </growth>

    </deployment>

  </institution>
```

Listing 4: Reactor deployment input schema for EG30

2.1 Non-optimizing prediction method

USE SENTENCES

Conditions to satisfy:

1. Do all the `Reactors` run at full capacity (not lacking fuel)?

```
TEST(ReactorTests, DDDeploy_NO) {
    [Example input with the following attributes:]
        [int simdur = 20;]
        [Defines reactor with zero refueling cycle and operation cycle of
            1 month]
        [Defines fuel cycle facilities parameters]
        [Defines Reactor Deploy Scheme / Power Demand]
        [Increasing Fuel Demand with Time]
    [Run test]
    [Test if Reactor has no zero values in output Timeseriespower]
}
```

Listing 5: Test to see all reactors run without lack of fuel

2. Is the predicted fuel demand within a specific uncertainty of the analytic solution?
3. Is the output of the `Fuelfab` within a specific range (more than?) of the input required by the `Reactors` (calculated by the analytic solution) for all of them to run for each time step?

- Is a new `Fuelfab` deployed when the input required by the reactors exceeds the output of current `Fuelfab`?

```
TEST(ReactorTests, DDDeploy_NO) {
    [Example input with the following attributes:]
        [int simdur = 20;]
        [Defines reactor with zero refueling cycle and operation
            cycle of 1 month]
        [Defines fuel cycle facilities parameters]
        [Defines Reactor Deploy Scheme / Power Demand]
        [Increasing Fuel Demand with Time]
    [Run test]
    [Test if Fuelfab is deployed in the beginning]
    [Test if Fuelfab is deployed later in the simulation (have
        analytic solution)]
}
```

Listing 6: Test demand-driven deployment of fuel cycle facility

- Is a `Fuelfab` decommissioned when the input required by the reactors falls behind the output of current `Fuelfab` facilities?

4. Is the output of the `Enrichment` within a specific range of the input required by the `Fuelfab` (calculated by the analytic solution) for each time step?

- Is a new `Enrichment` deployed when the input required by the `Fuelfab` exceeds the output of current `Enrichment` facilities?
- Is a `Enrichment` decommissioned when the input required by the `Fuelfab` falls behind the output of current `Enrichment`?

5. Is the output of `Source` within a specific range of the input required by the `Enrichment` (calculated by the analytic solution) for each time step?
   - Does the `Source` output increase when the input required by the `Enrichment` exceeds the output of current `Source`?
   - Does the `Source` output decrease when the input required by the `Enrichment` falls behind the output of current `Source`?

However, the user may not have his or her cycle set to the generic fuel cycle, and may have variations so that there is no `Fuelfab` or `Enrichment`, but the fuel is directly supplied by the `Source`. This brings the need for the algorithm to recognize the supply chain the supply fuel to the reactor, and deploy necessary facilities accordingly.

2.2 Deterministic-Optimizing/Stochastic prediction method

Conditions for test to satisfy:

1. Do all the `Reactors` run at full capacity (not lacking fuel)?

```
1  TEST(ReactorTests, DDDeploy_DO) {
2      [Example input with the following attributes:]
3          [int simdur = 20;]
4          [Defines reactor with zero refueling cycle and operation cycle of
               1 month]
5          [Defines fuel cycle facilities parameters]
6          [Defines Reactor Deploy Scheme / Power Demand]
7          [Increasing Fuel Demand with Time]
8      [Run test]
9      [Test if Reactor has no zero values in output Timeseriespower]
10 }
```

Listing 7: Test to see all reactors run without lack of fuel

2. Is the objective function optimized?

3. Is the constraint followed?

4. Do the related fuel cycle facilities get deployed upon demand?

```
TEST(ReactorTests, DDDeploy_NO) {
    [Example input with the following attributes:]
        [int simdur = 20;]
        [Defines reactor with zero refueling cycle and operation cycle of
            1 month]
        [Defines fuel cycle facilities parameters]
        [Defines Reactor Deploy Scheme / Power Demand]
        [Increasing Fuel Demand with Time]
    [Run test]
    [Test if fuel facility is deployed in the beginning]
    [Test if fuel facility is deployed later in the simulation (have
        analytic solution)]
}
```

Listing 8: Test demand-driven deployment of fuel cycle facility

5. Do the related fuel cycle facilities exit upon demand decrease?

```
TEST(ReactorTests, DDDeploy_NO) {
    [Example input with the following attributes:]
        [int simdur = 20;]
        [Defines reactor with zero refueling cycle and operation cycle of
            1 month]
        [Defines fuel cycle facilities parameters]
        [Defines Reactor Deploy Scheme / Power Demand]
        [Decreasing Fuel Demand with Time]
    [Run test]
    [Test if fuel facility is deployed in the beginning]
    [Test if fuel facility exits later in the simulation (have analytic
        solution)]
}
```

Listing 9: Test demand-driven exit of fuel cycle facility

3 Advanced Fuel Cycles

Advanced fuel cycles denote the fuel cycles where reprocessing of spent fuel takes place, to create fuel for advanced reactors.