Probabilistic Safety Assessment and Management (PSAM)
June 26th - July 1st, 2022
Sheraton Waikiki
Honolulu, O'ahu Hawaii USA

Integrated dynamic probabilistic safety assessments with PyCATSHOO: a new coupling approach

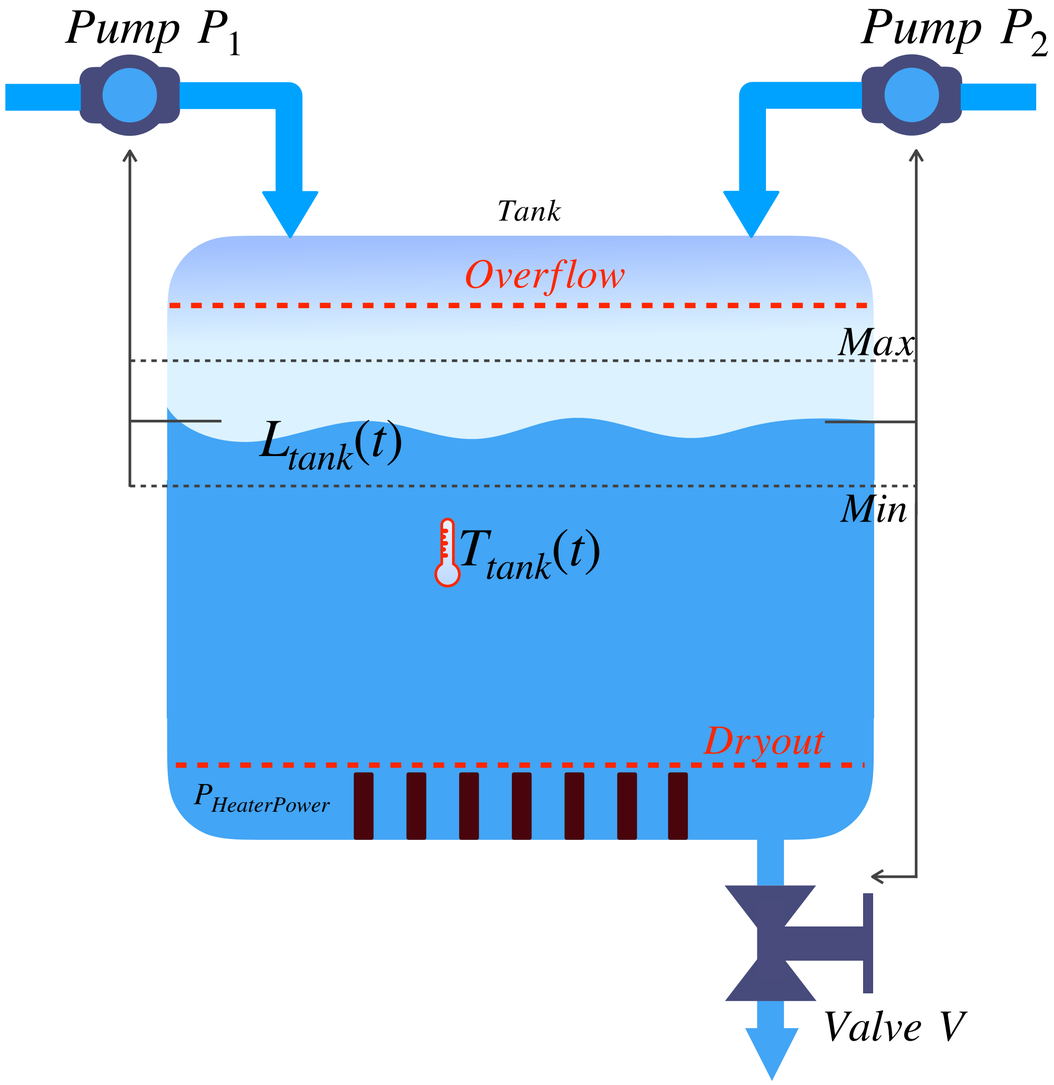
Hassane CHRAIBI (EDF)
Jean-Christophe HOUDEBINE (ARISTE)
Claudia PICOCO (EDF)
Valentin RYCHKOV (EDF)

SUMMARY

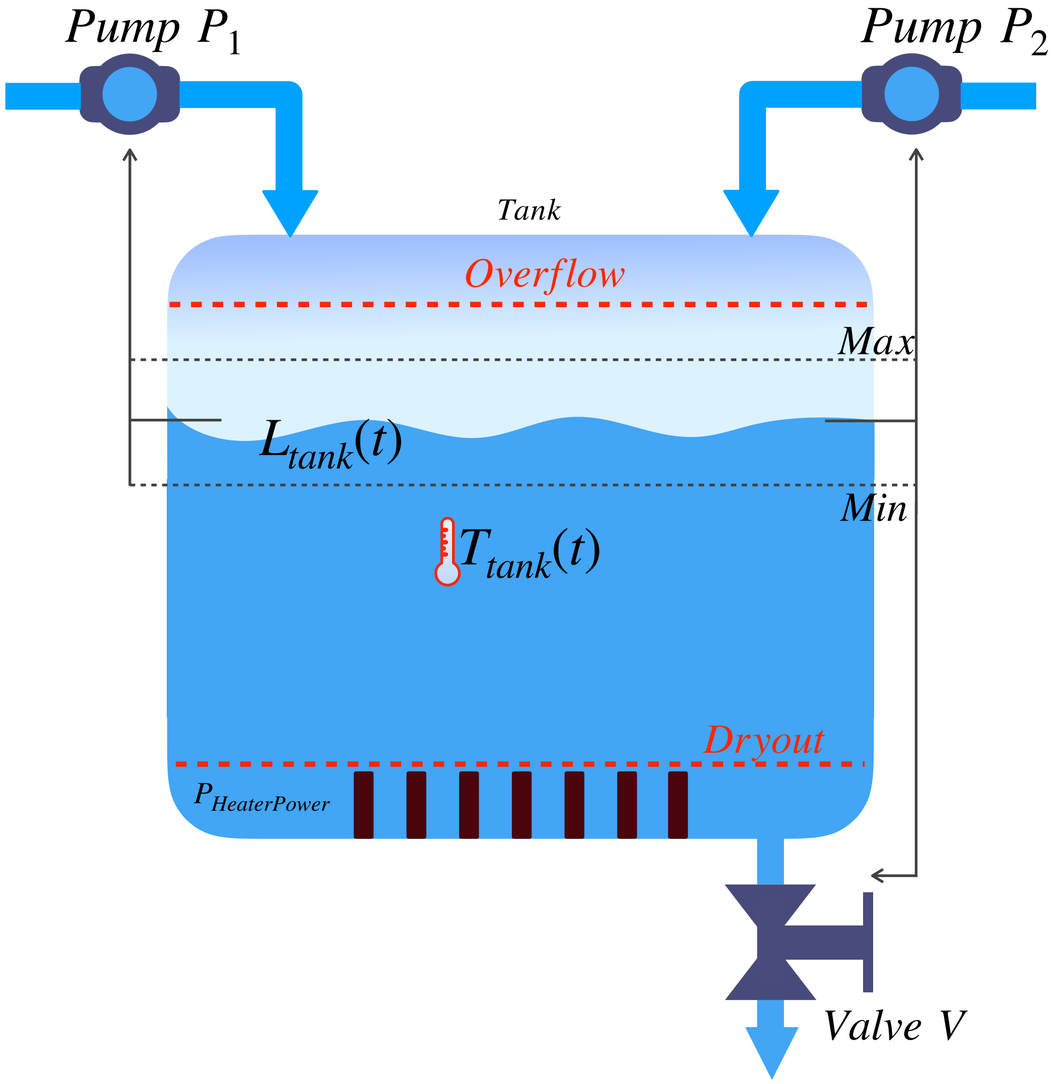
1. How to model with PyCATSHOO
2. The FMI standard
3. PyCATSHOO and external models

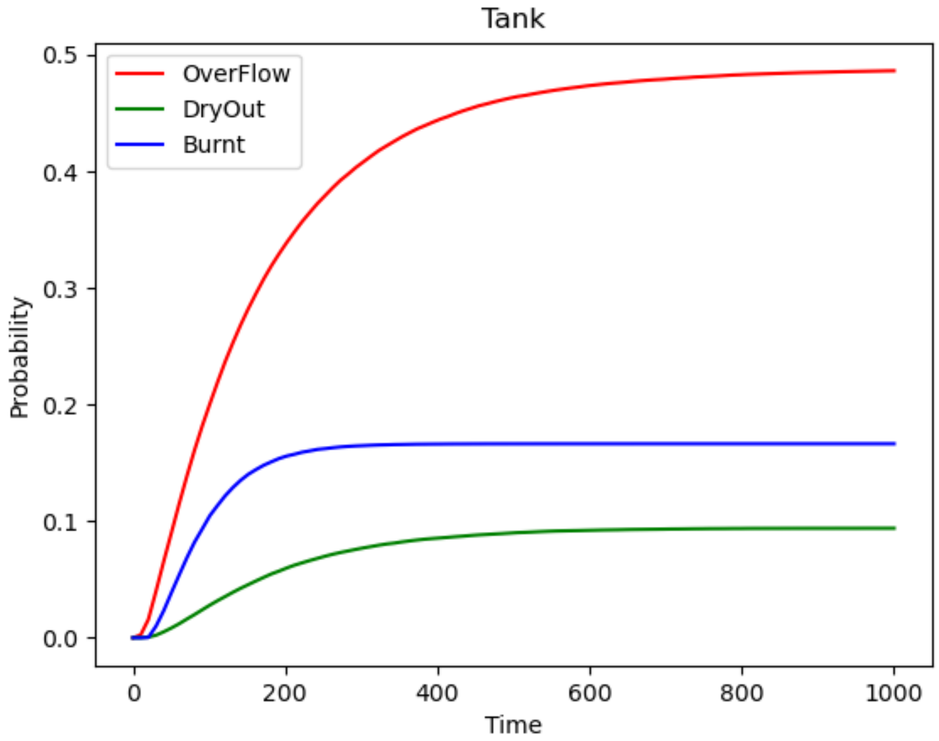
1 HOW TO MODEL WITH PyCATSHOO

HEATED TANK AND PYCATSHOO MODELING APPROACH

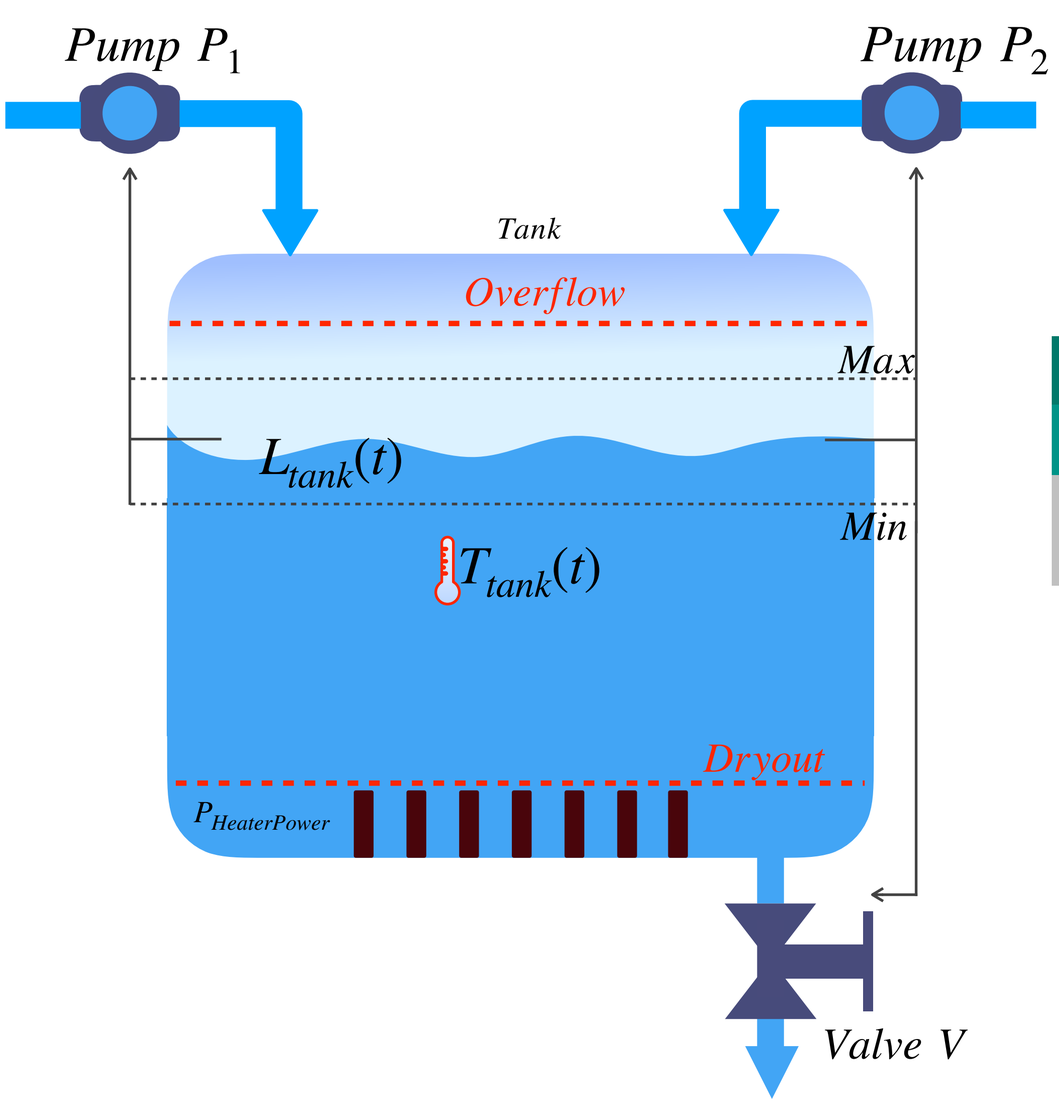


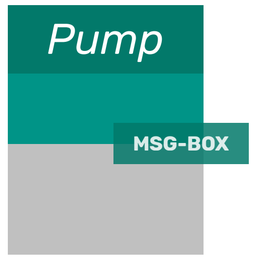
HEATED TANK AND PYCATSHOO MODELING APPROACH

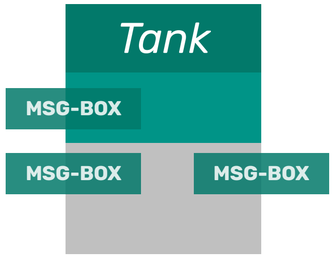


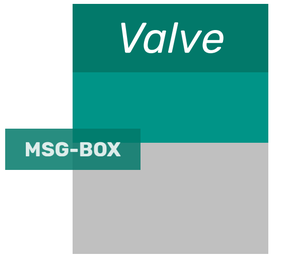


HEATED TANK AND PYCATSHOO MODELING APPROACH

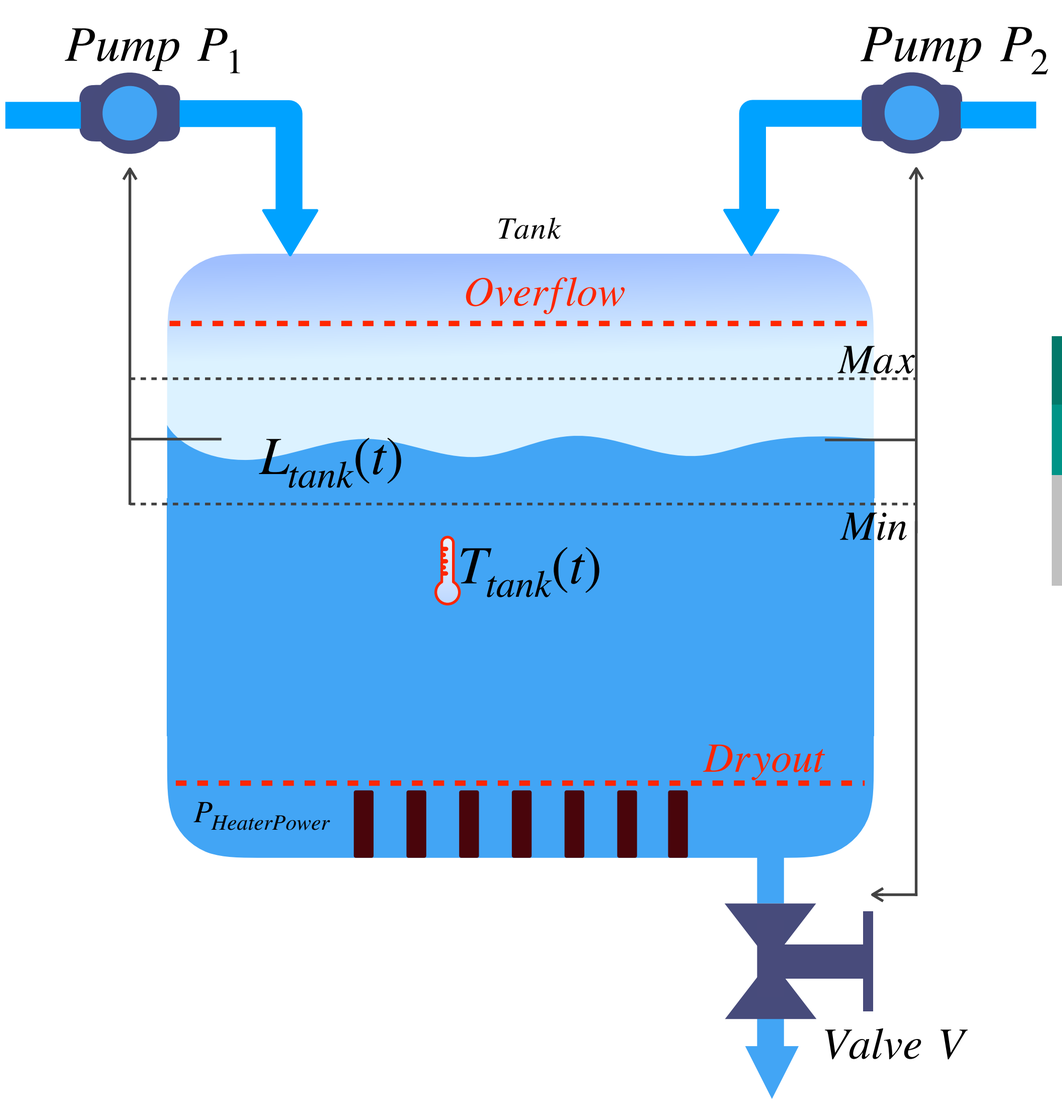


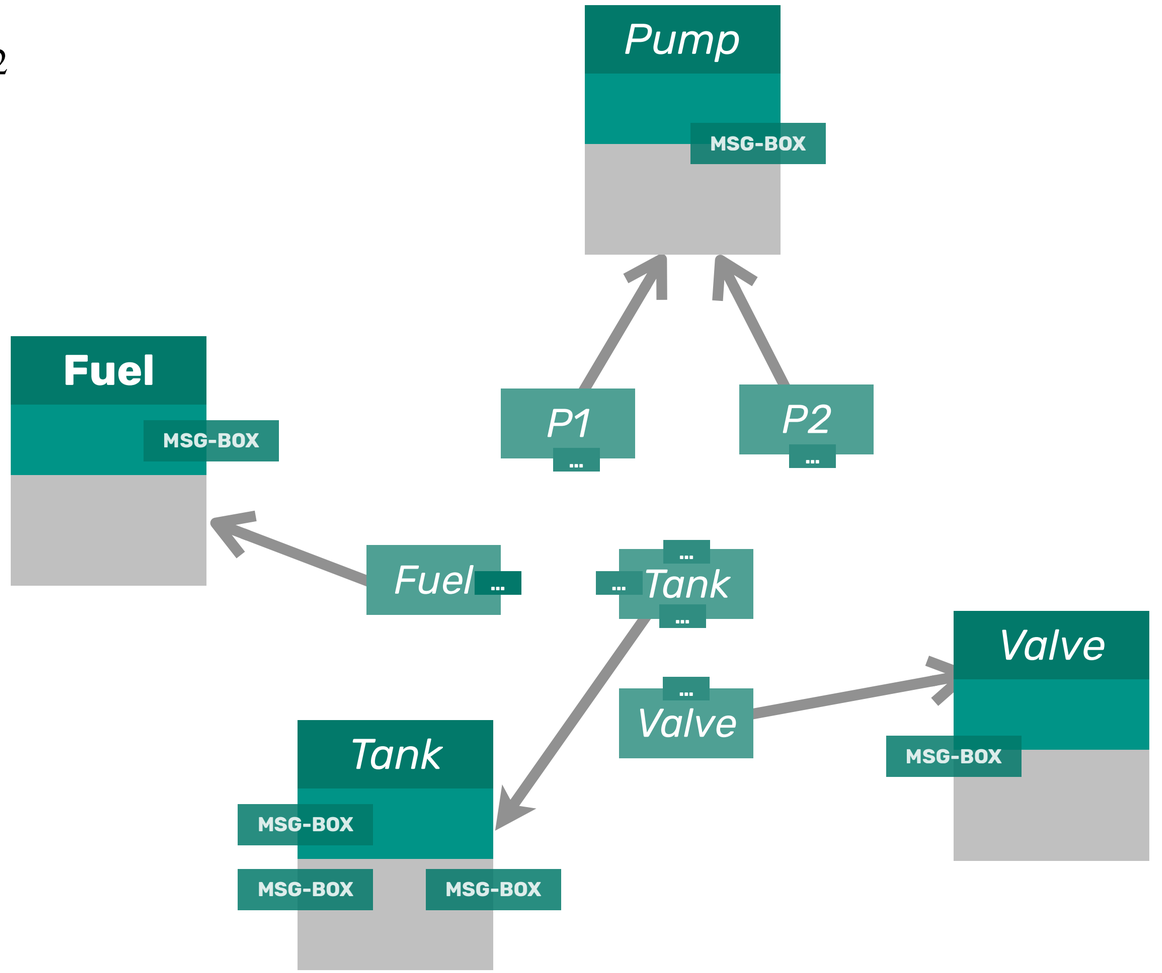




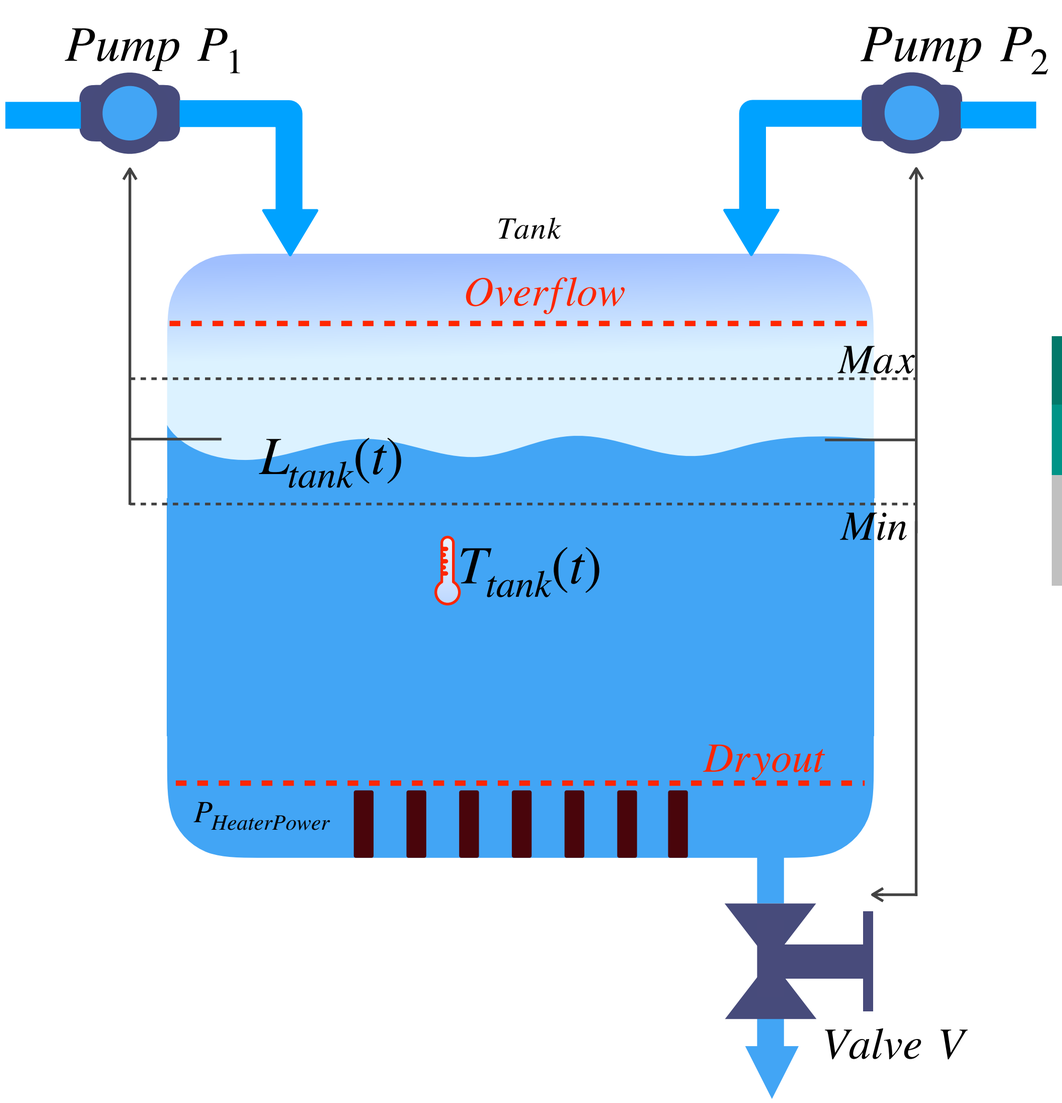


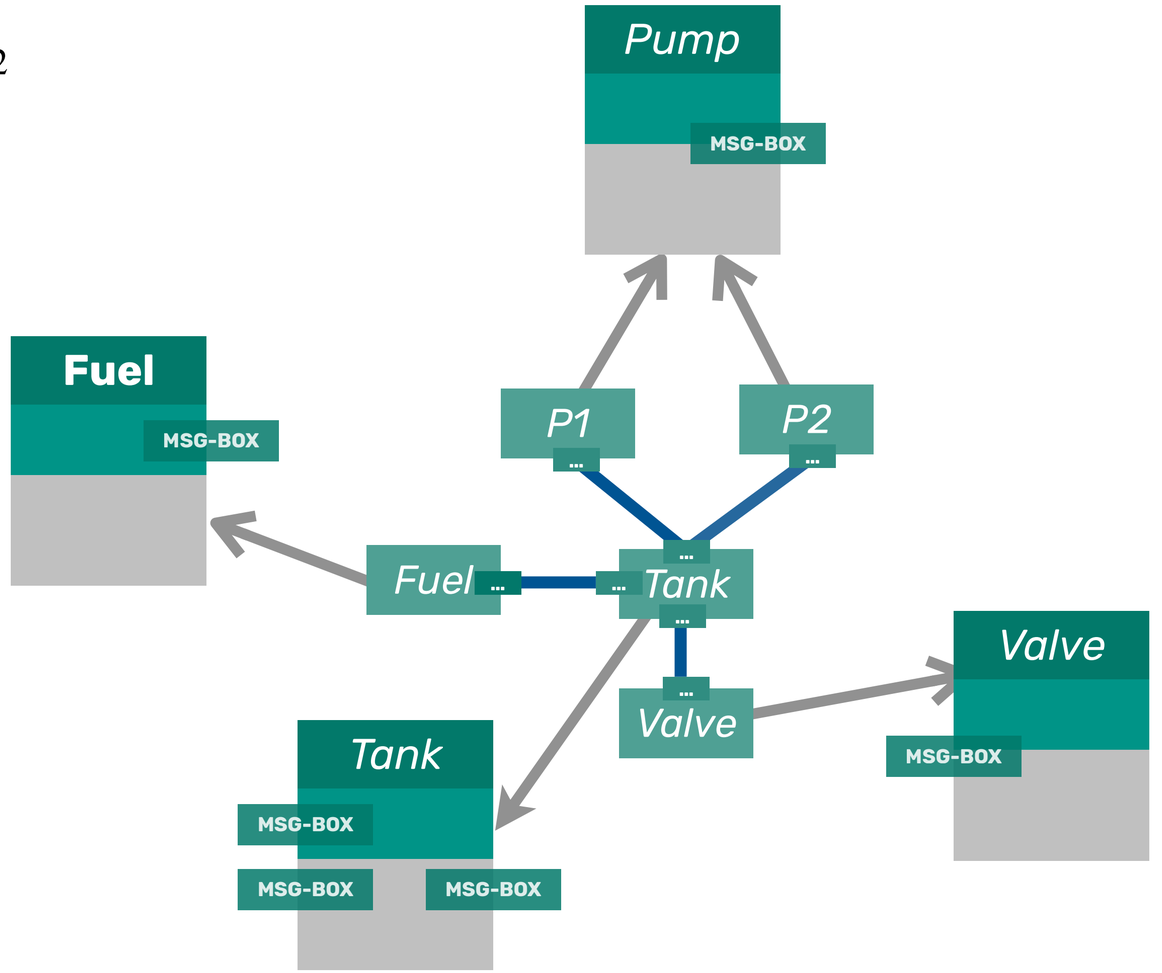
HEATED TANK AND PYCATSHOO MODELING APPROACH



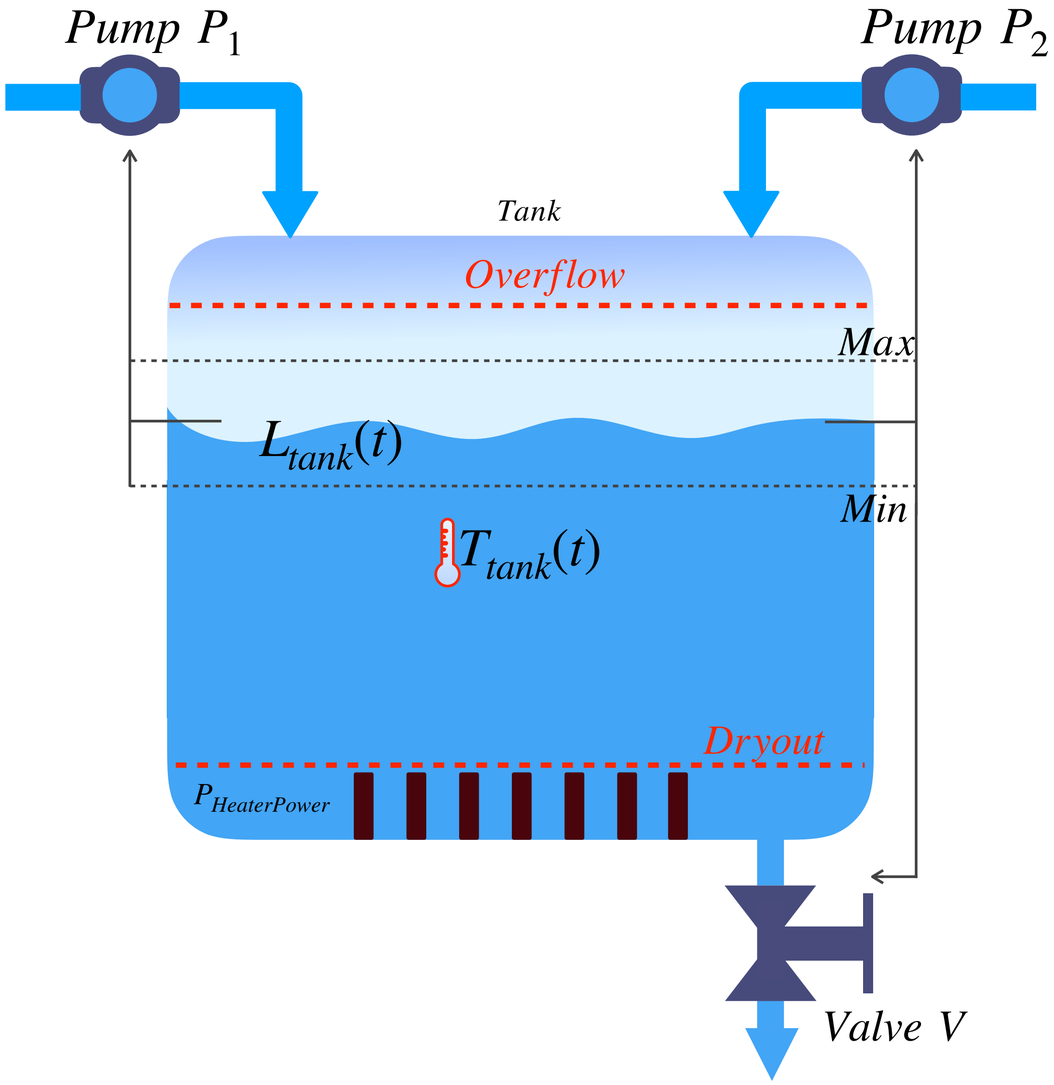


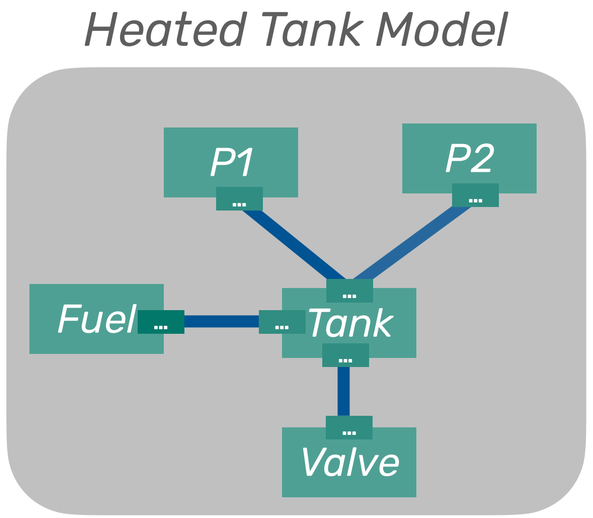
HEATED TANK AND PYCATSHOO MODELING APPROACH





HEATED TANK AND PYCATSHOO MODELING APPROACH

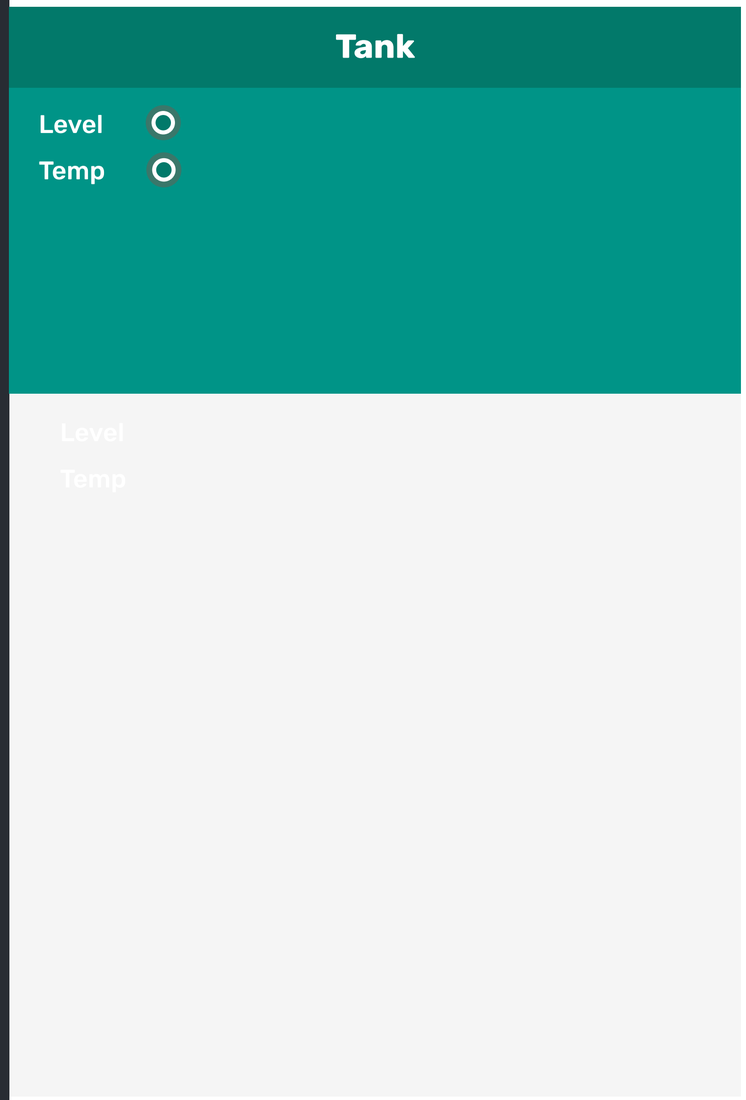




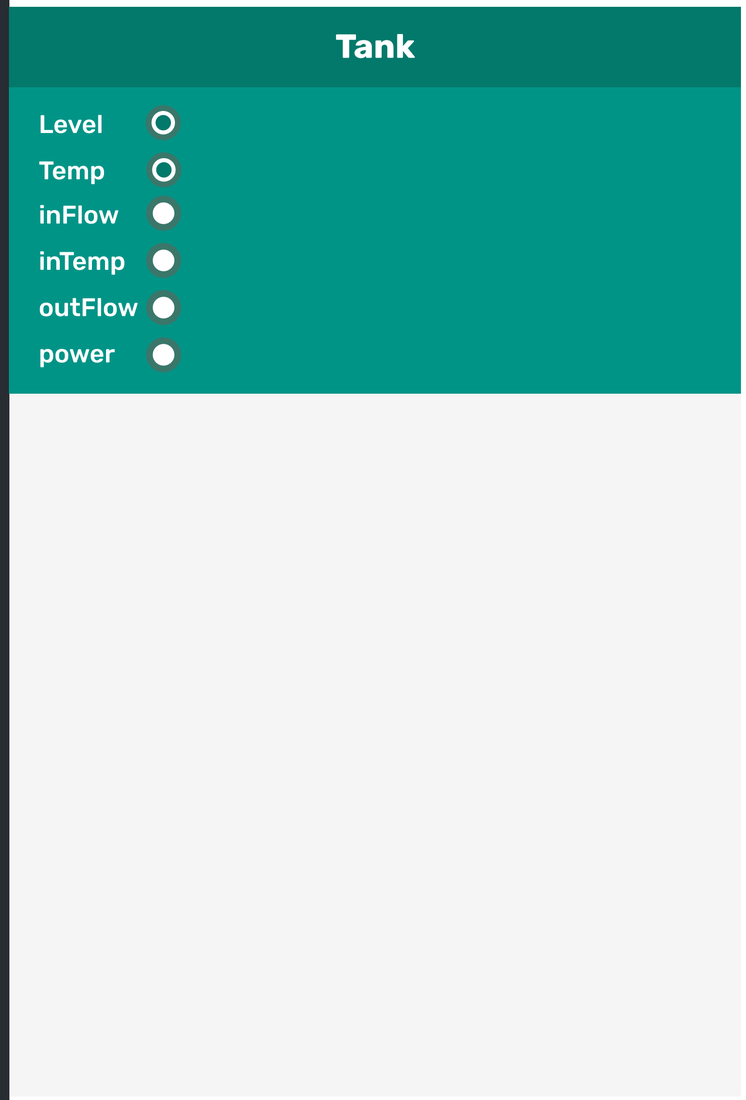
```python
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)
```

Tank

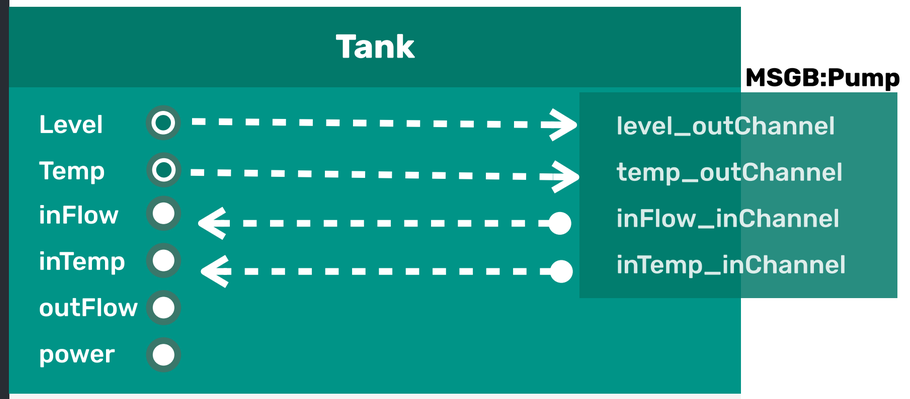
```
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.v_level           = self.addVariable("level", Pyc.TVarType.t_double, 7.)
        self.v_temperature     = self.addVariable("temp" , Pyc.TVarType.t_double, 31.)
```



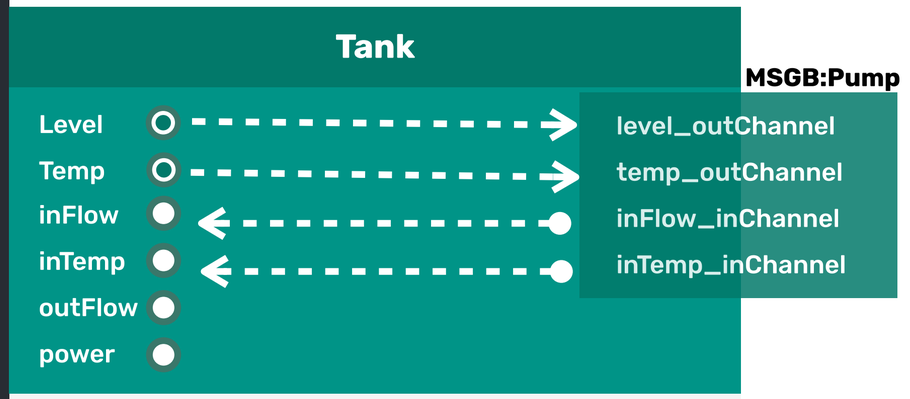
```
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.v_level          = self.addVariable("level", Pyc.TVarType.t_double, 7.)
        self.v_temperature    = self.addVariable("temp" , Pyc.TVarType.t_double, 31.)
        ……
        self.r_inFlow         = self.addReference("inFlow")
        self.r_temperature    = self.addReference("inTemp")
        self.r_outFlow        = self.addReference("outFlow")
        self.r_power          = self.addReference("power")
```



```
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.v_level          = self.addVariable("level", Pyc.TVarType.t_double, 7.)
        self.v_temperature    = self.addVariable("temp" , Pyc.TVarType.t_double, 31.)
        ……
        self.r_inFlow         = self.addReference("inFlow")
        self.r_temperature    = self.addReference("inTemp")
        self.r_outFlow        = self.addReference("outFlow")
        self.r_power          = self.addReference("power")
        ……
        self.addMessageBox("Pump")
        self.addMessageBoxExport("Pump", self.v_level      , "level_outChannel")
        self.addMessageBoxExport("Pump", self.v_temperature, "temp_outChannel")
        self.addMessageBoxImport("Pump", self.r_inFlow     , "inFlow_inChannel")
        self.addMessageBoxImport("Pump", self.r_temperature, "inTemp_inChannel")
```



```
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.v_level          = self.addVariable("level", Pyc.TVarType.t_double, 7.)
        self.v_temperature    = self.addVariable("temp" , Pyc.TVarType.t_double, 31.)
        ……
        self.r_inFlow         = self.addReference("inFlow")
        self.r_temperature    = self.addReference("inTemp")
        self.r_outFlow        = self.addReference("outFlow")
        self.r_power          = self.addReference("power")
        ……
        self.addMessageBox("Pump")
        self.addMessageBoxExport("Pump", self.v_level      , "level_outChannel")
        self.addMessageBoxExport("Pump", self.v_temperature, "temp_outChannel")
        self.addMessageBoxImport("Pump", self.r_inFlow     , "inFlow_inChannel")
        self.addMessageBoxImport("Pump", self.r_temperature, "inTemp_inChannel")
        ……
        pdmp = self.addPDMPManager ("pdmpManager")
        self.addPDMPODEVariable     ("pdmpManager", self.v_level)
        self.addPDMPODEVariable     ("pdmpManager", self.v_temperature)
        self.addPDMPEquationMethod ("pdmpManager", "odeMethod", self.odeMethod)
```

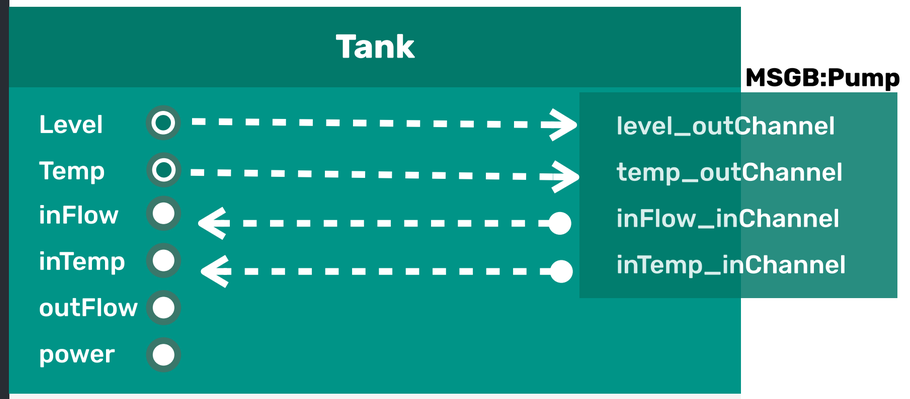


PDMP: pdmpManage

$$\frac{dLevel}{dt} = \frac{\sum_i inFlow[i] - \sum_i outFlow[i]}{area}$$

$$\frac{dTemp}{dt} = \frac{\sum_i inFlow[i] \,.\, (intTemp[i] - Temp)}{Level \,.\, area}$$

```
class Tank(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.v_level          = self.addVariable("level", Pyc.TVarType.t_double, 7.)
        self.v_temperature    = self.addVariable("temp" , Pyc.TVarType.t_double, 31.)
        ……
        self.r_inFlow         = self.addReference("inFlow")
        self.r_temperature    = self.addReference("inTemp")
        self.r_outFlow        = self.addReference("outFlow")
        self.r_power          = self.addReference("power")
        ……
        self.addMessageBox("Pump")
        self.addMessageBoxExport("Pump", self.v_level      , "level_outChannel")
        self.addMessageBoxExport("Pump", self.v_temperature, "temp_outChannel")
        self.addMessageBoxImport("Pump", self.r_inFlow     , "inFlow_inChannel")
        self.addMessageBoxImport("Pump", self.r_temperature, "inTemp_inChannel")
        ……
        pdmp = self.addPDMPManager ("pdmpManager")
        self.addPDMPODEVariable     ("pdmpManager", self.v_level)
        self.addPDMPODEVariable     ("pdmpManager", self.v_temperature)
        self.addPDMPEquationMethod ("pdmpManager", "odeMethod", self.odeMethod)
        ……
    def odeMethod(self):
        iFlow    = self.r_inFlow.sumValue()
        oFlow    = self.r_outFlow.sumValue()
        self.v_level.setDvdtODE((iFlow - oFlow) / self.v_area.dValue())

        sumiFiT = 0
        for i in range(self.r_inFlow.nbCnx()):
            sumiFiT = sumiFiT + self.r_inFlow.dValue(i) * \
                      (self.r_temperature.dValue(i) - self.v_temperature.value())

        self.v_temperature.setDvdtODE(
              (sumiFiT + self.r_power.dValue(0)) / \
              (self.v_level.dValue()* self.v_area.dValue())
                                      )
```

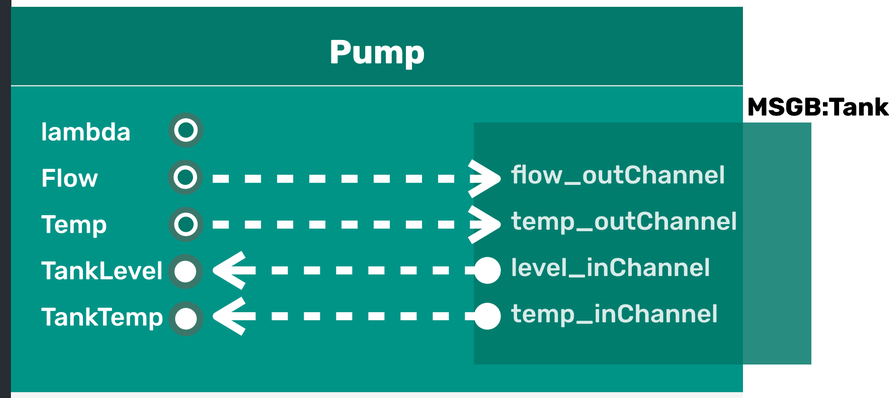


PDMP: pdmpManage

$$\frac{dLevel}{dt} = \frac{\sum_i inFlow[i] - \sum_i outFlow[i]}{area}$$

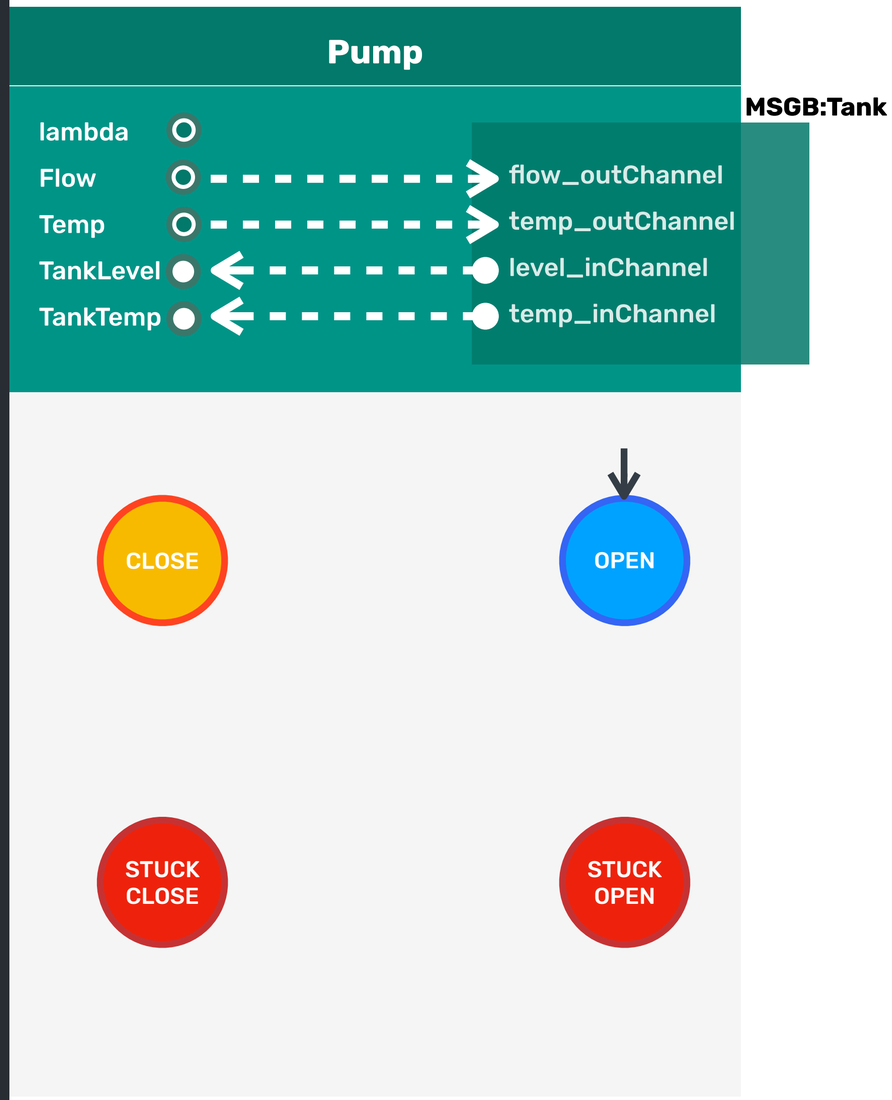
$$\frac{dTemp}{dt} = \frac{\sum_i inFlow[i] \,.\, (intTemp[i] - Temp)}{Level \,.\, area}$$

```
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)
```



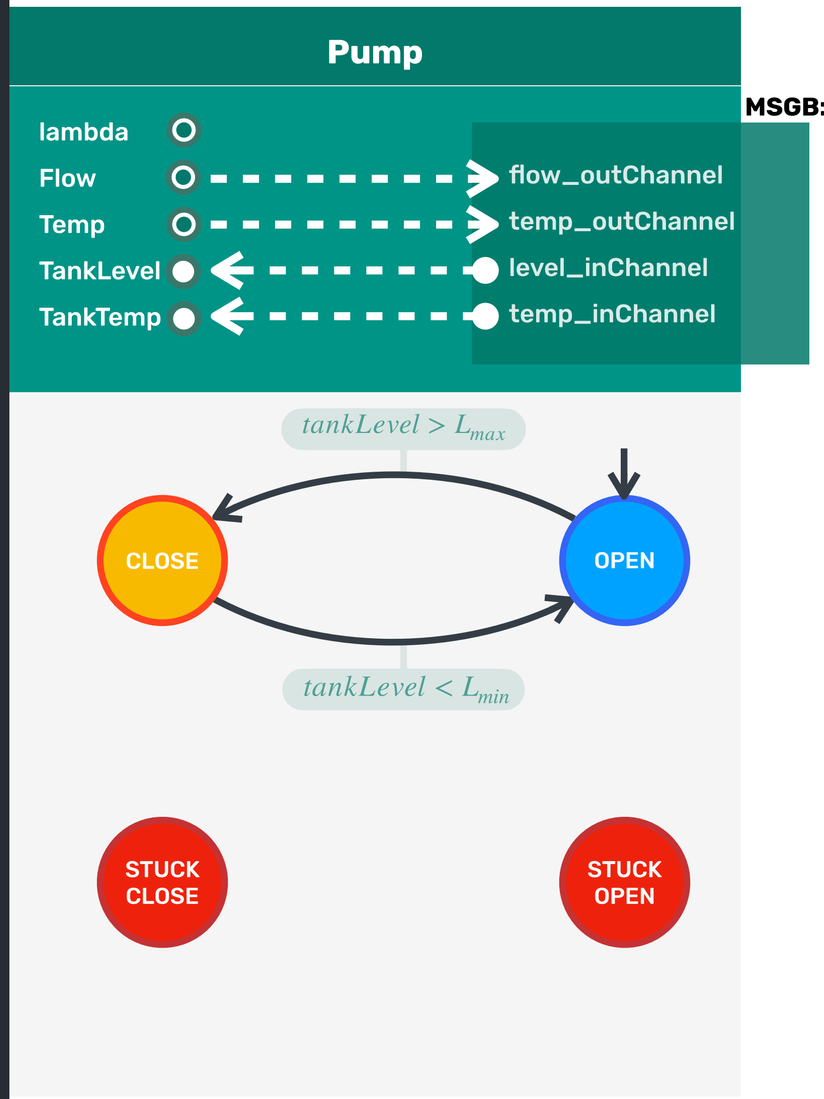
```
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)
        ……
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN"  , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN"  , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)
```

```
        ……
    def updateFlow(self):
        if self.openState.isActive() or self.stuckOpenState.isActive()
            self.v_flow.setValue(self.v_nominalFlow.dValue())
        else
            self.v_flow.setValue(0)
```



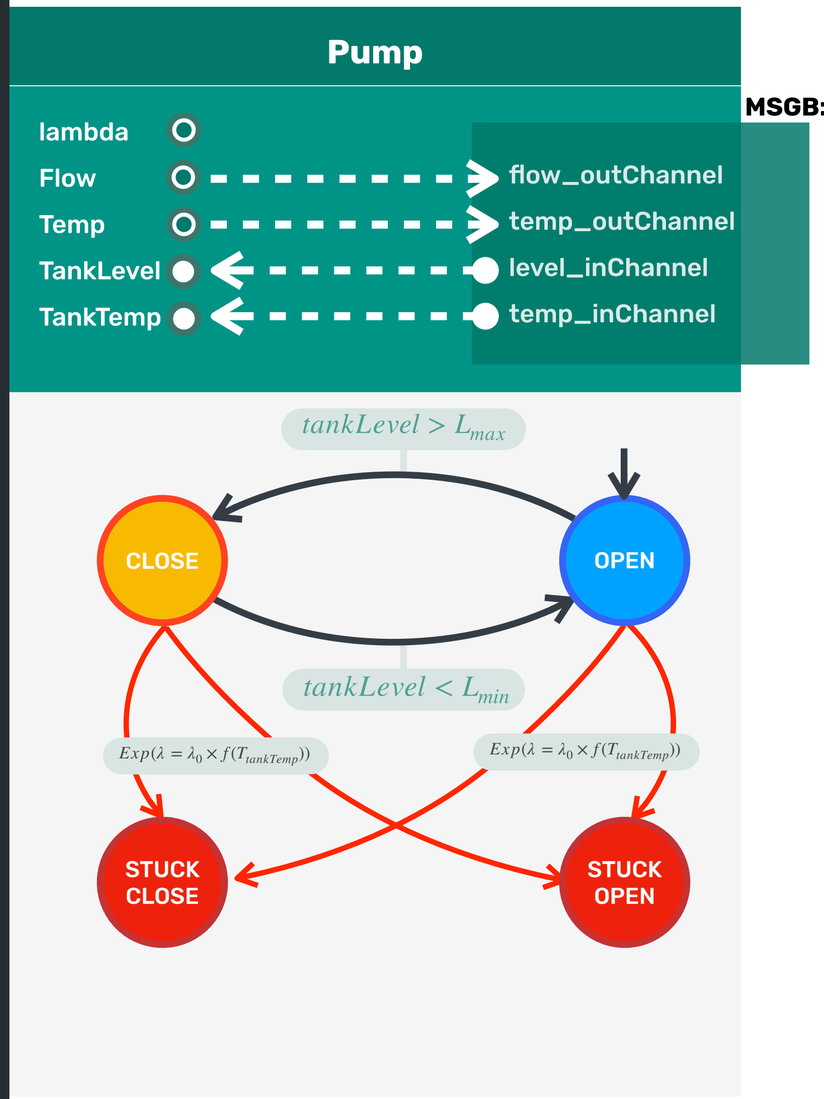
```
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ......
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN" , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN" , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)

        ......
        transO2C = self.openState.addTransition("OPEN_to_CLOSE")
        transO2C.setCondition(lambda : self.r_tankLevel.dValue(0) > \
                                                    self.v_levelMax.dValue())
        transO2C.addTarget(self.closeState, Pyc.TTransType.trans)
```

```
    ......
    def updateFlow(self):
        if self.openState.isActive() or self.stuckOpenState.isActive()
            self.v_flow.setValue(self.v_nominalFlow.dValue())
        else
            self.v_flow.setValue(0)
```



```python
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN" , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN" , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)

        ……
        transO2C = self.openState.addTransition("OPEN_to_CLOSE")
        transO2C.setCondition(lambda : self.r_tankLevel.dValue(0) > \
                                       self.v_levelMax.dValue())
        transO2C.addTarget(self.closeState, Pyc.TTransType.trans)

        ……
        trans = self.openState.addTransition("OPEN_to_STUCKOPEN")
        trans.setDistLaw(Pyc.TLawType.expo, self.v_lambda)
        trans.addTarget(self.stuckOpenState, Pyc.TTransType.fault)
        trans.setModifiable(Pyc.TModificationMode.continuous_modification)
```

```python
    ……
def updateFlow(self):
    if self.openState.isActive() or self.stuckOpenState.isActive()
        self.v_flow.setValue(self.v_nominalFlow.dValue())
    else
        self.v_flow.setValue(0)
```

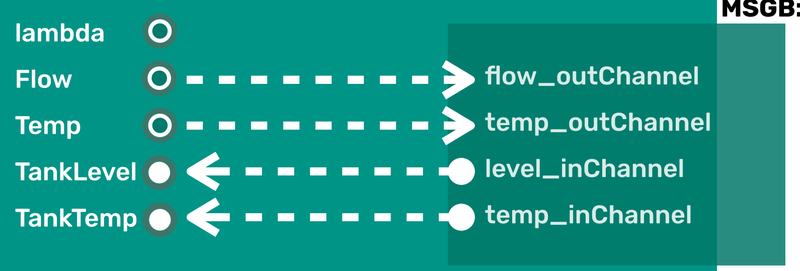


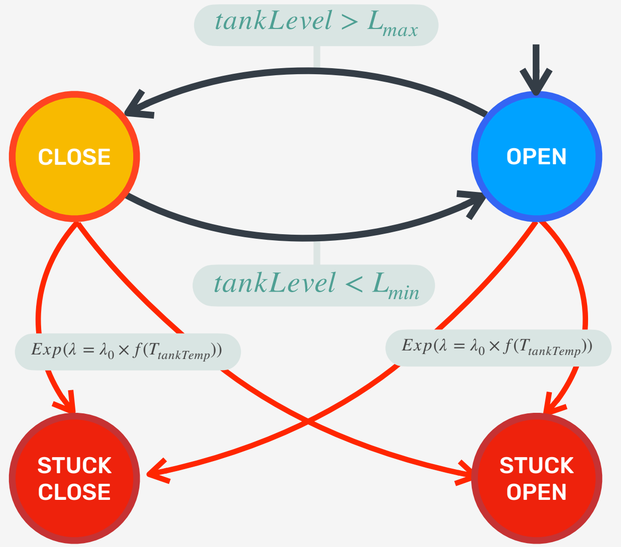
```
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN" , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN" , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)

        ……
        transO2C = self.openState.addTransition("OPEN_to_CLOSE")
        transO2C.setCondition(lambda : self.r_tankLevel.dValue(0) > self.v_levelMax.dValue())
        transO2C.addTarget(self.closeState, Pyc.TTransType.trans)

        ……
        trans = self.openState.addTransition("OPEN_to_STUCKOPEN")
        trans.setDistLaw(Pyc.TLawType.expo, self.v_lambda)
        trans.addTarget(self.stuckOpenState, Pyc.TTransType.fault)
        trans.setModifiable(Pyc.TModificationMode.continuous_modification)

        ……
        pdmp = self.addPDMPManager  (“pdmpManager")
        self.addPDMPExplicitVariable("pdmpManager", self.v_lambda)
        self.addPDMPEquationMethod  ("pdmpManager", "equationMethod",
                                                    self.equationMethod)
```

```
    ……
def updateFlow(self):
    if self.openState.isActive() or self.stuckOpenState.isActive()
        self.v_flow.setValue(self.v_nominalFlow.dValue())
    else
        self.v_flow.setValue(0)
```

Pump

MSGB:Tank





PDMP : pdmpManage

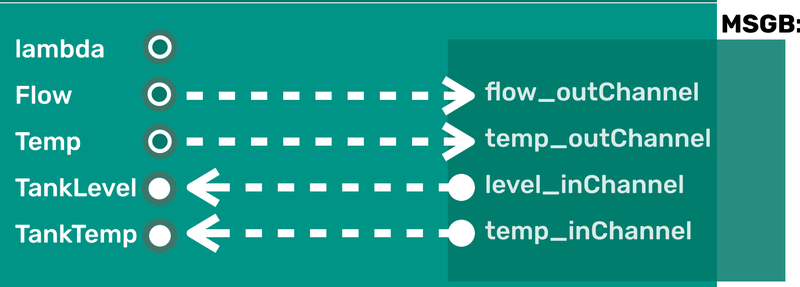
$$\lambda = \lambda_0 \times (b_1 \times e^{b_c \times (T_{tank} - 20)} + b_2 \times e^{-b_d \times (T_{tank} - 20)})$$

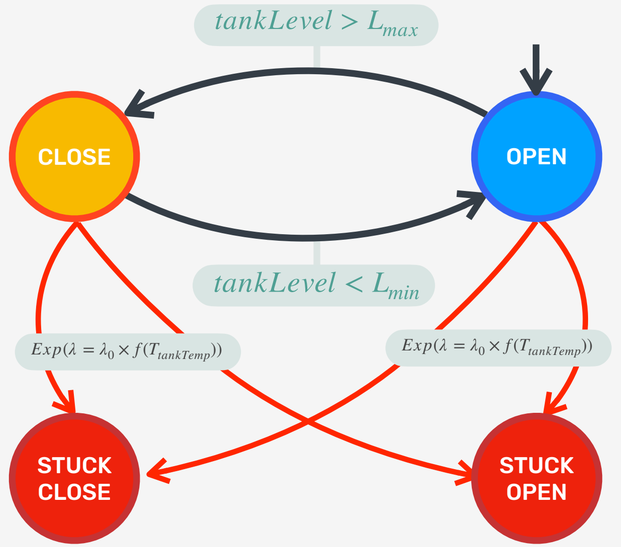
```
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN" , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN" , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)

        ……
        transO2C = self.openState.addTransition("OPEN_to_CLOSE")
        transO2C.setCondition(lambda : self.r_tankLevel.dValue(0) > self.v_levelMax.dValue())
        transO2C.addTarget(self.closeState, Pyc.TTransType.trans)

        ……
        trans = self.openState.addTransition("OPEN_to_STUCKOPEN")
        trans.setDistLaw(Pyc.TLawType.expo, self.v_lambda)
        trans.addTarget(self.stuckOpenState, Pyc.TTransType.fault)
        trans.setModifiable(Pyc.TModificationMode.continuous_modification)

        ……
        pdmp = self.addPDMPManager  (“pdmpManager")
        self.addPDMPExplicitVariable("pdmpManager", self.v_lambda)
        self.addPDMPEquationMethod  ("pdmpManager", "equationMethod",
                                                    self.equationMethod)

        pdmp.addWatchedTransition   (transO2C)
```

```
    ……
def updateFlow(self):
    if self.openState.isActive() or self.stuckOpenState.isActive()
        self.v_flow.setValue(self.v_nominalFlow.dValue())
    else
        self.v_flow.setValue(0)
```

Pump

MSGB:Tank





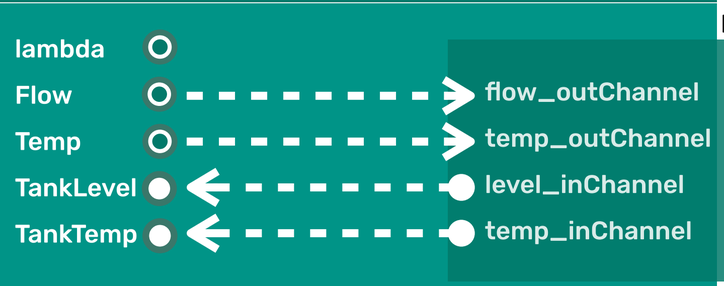
PDMP : pdmpManage

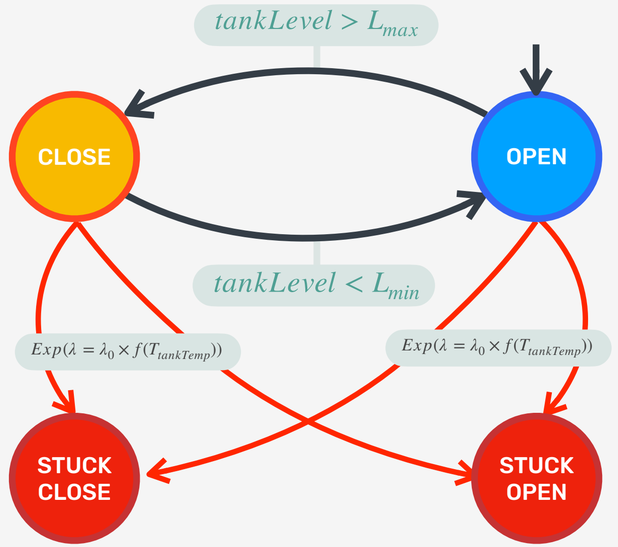
$$\lambda = \lambda_0 \times (b_1 \times e^{b_c \times (T_{tank} - 20)} + b_2 \times e^{-b_d \times (T_{tank} - 20)})$$

```python
class Pump(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.automaton       = self.addAutomaton("FuncAutomation")
        self.openState       = self.addState("FuncAutomation", "OPEN" , 0)
        self.closeState      = self.addState("FuncAutomation", "CLOSE", 1)
        self.stuckOpenState  = self.addState("FuncAutomation", "STUCKOPEN" , 2)
        self.stuckCloseState = self.addState("FuncAutomation", "STUCKCLOSE", 3)
        self.automaton.setInitState(self.openState)
        self.automaton.addSensitiveMethod("updateFlow", self.updateFlow)

        ……
        transO2C = self.openState.addTransition("OPEN_to_CLOSE")
        transO2C.setCondition(lambda : self.r_tankLevel.dValue(0) > self.v_levelMax.dValue())
        transO2C.addTarget(self.closeState, Pyc.TTransType.trans)

        ……
        trans = self.openState.addTransition("OPEN_to_STUCKOPEN")
        trans.setDistLaw(Pyc.TLawType.expo, self.v_lambda)
        trans.addTarget(self.stuckOpenState, Pyc.TTransType.fault)
        trans.setModifiable(Pyc.TModificationMode.continuous_modification)

        ……
        pdmp = self.addPDMPManager  (“pdmpManager")
        self.addPDMPExplicitVariable("pdmpManager", self.v_lambda)
        self.addPDMPEquationMethod  ("pdmpManager", "equationMethod",
                                                    self.equationMethod)
        pdmp.addWatchedTransition   (transO2C)

        ……
    def equationMethod(self):
        b1 = 3.0295, b2 = 0.7578, bc = 0.05756, bd = 0.2301
       self.v_lambda.setDValue(self.v_lambda0.value()*\
          (b1 * np.exp(+ bc * (self.r_tankTemperature.value(0) - 20)) +
           b2 * np.exp(- bd * (self.r_tankTemperature.value(0) - 20))  ) / (b1 + b2)
        ……
    def updateFlow(self):
        if self.openState.isActive() or self.stuckOpenState.isActive()
            self.v_flow.setValue(self.v_nominalFlow.dValue())
        else
            self.v_flow.setValue(0)
```

Pump

MSGB:Tank





PDMP : pdmpManage

$$\lambda = \lambda_0 \times (b_1 \times e^{b_c \times (T_{tank} - 20)} + b_2 \times e^{-b_d \times (T_{tank} - 20)})$$

2 THE FMI STANDARD

FMI OVERVIEW

- **Origin**

  MODELISAR (2008-2011) an ITEA2 project →FMI

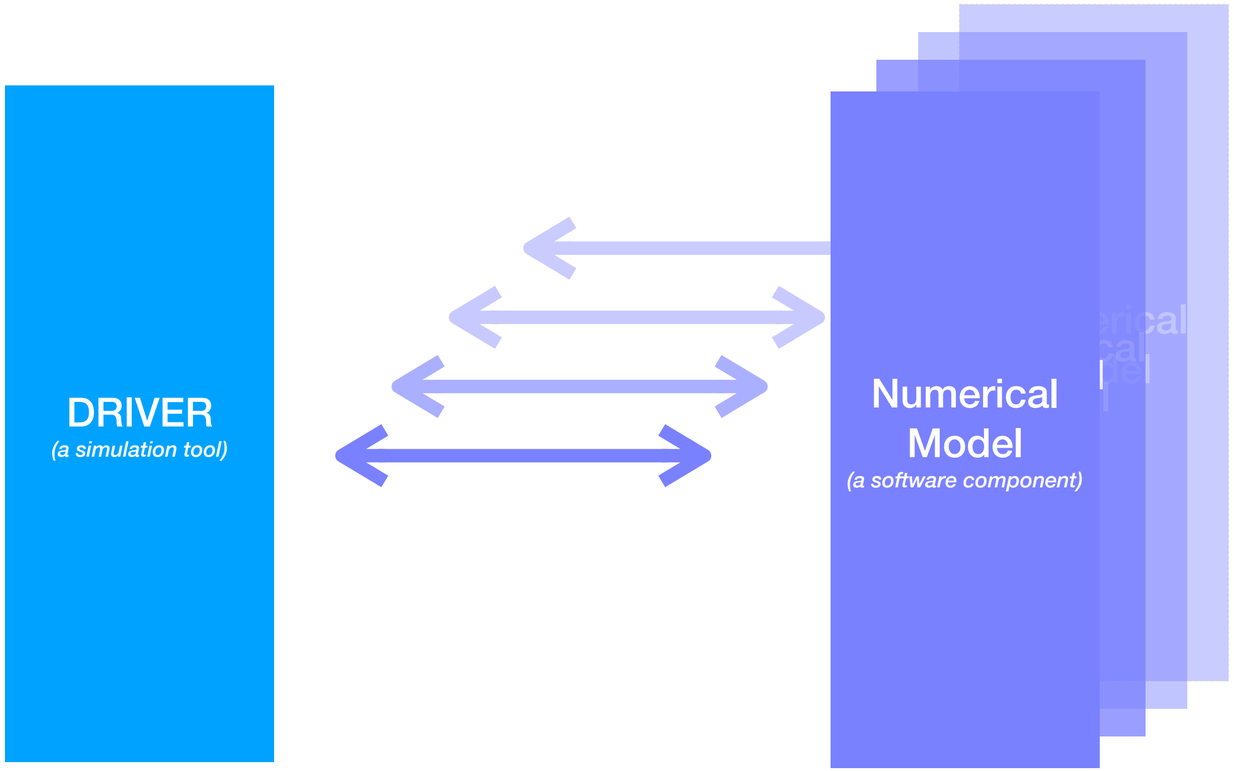
- **Motivation**

  The Problem of Interoperability between software components from different suppliers and different simulation stools

- **Current status**

  FMI is a permanent project of Modelica association

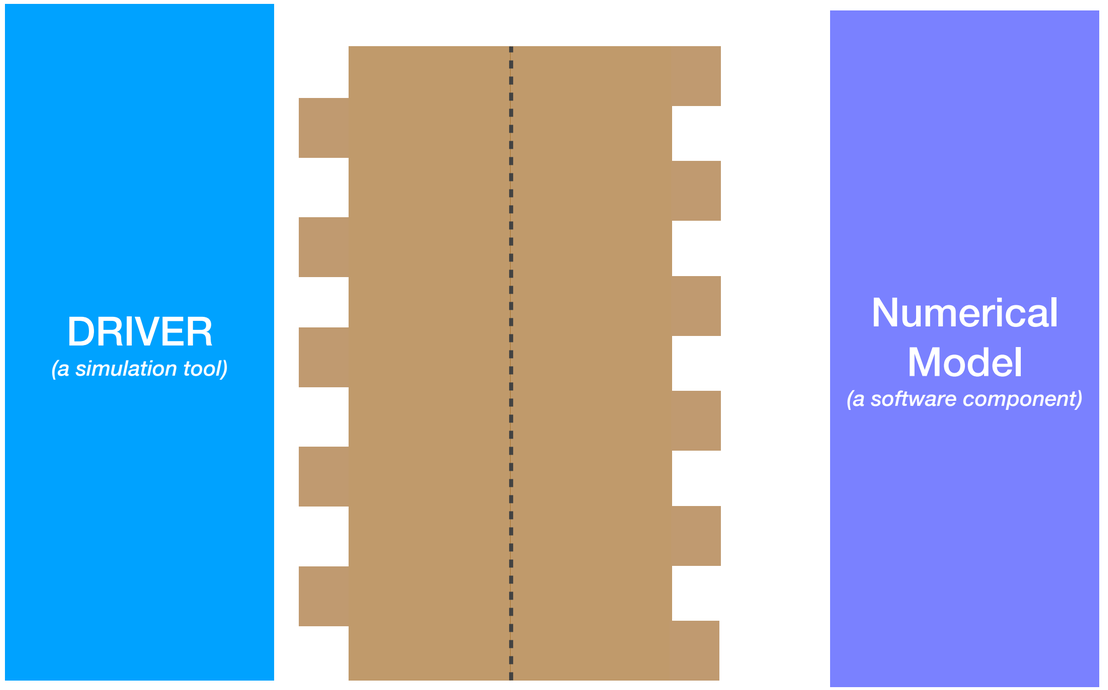
COUPLING IN HYBRID DYNAMIC PROBABILISTIC ASSESSMENT



Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

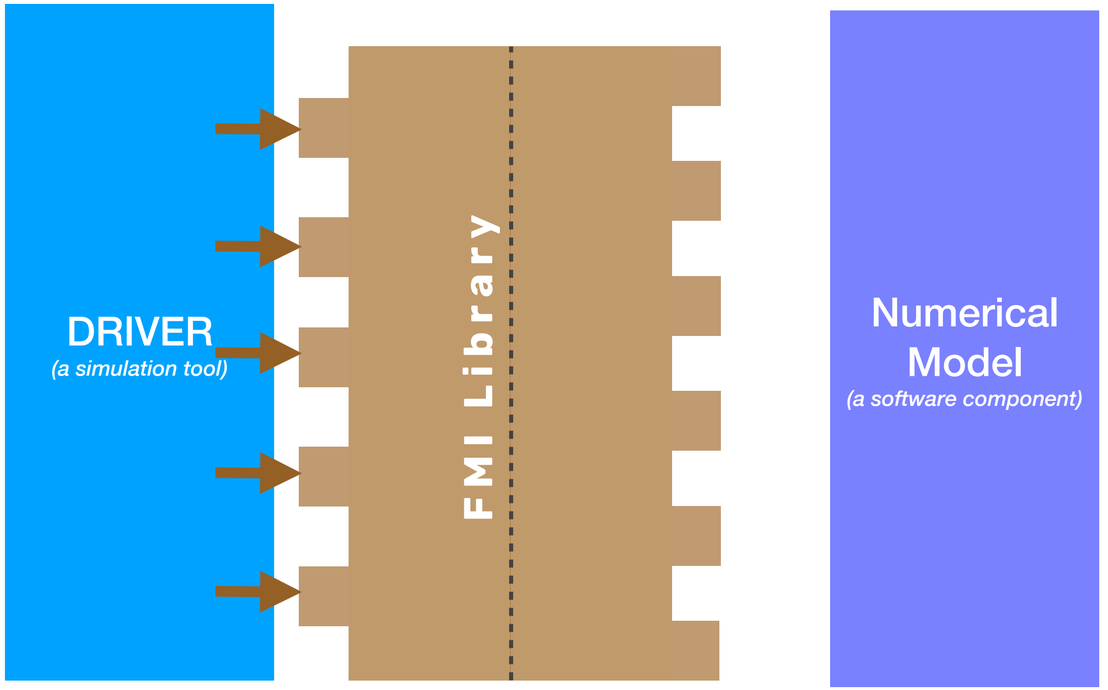
FMI CO-SIMULATION COUPLING APPROACH



Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

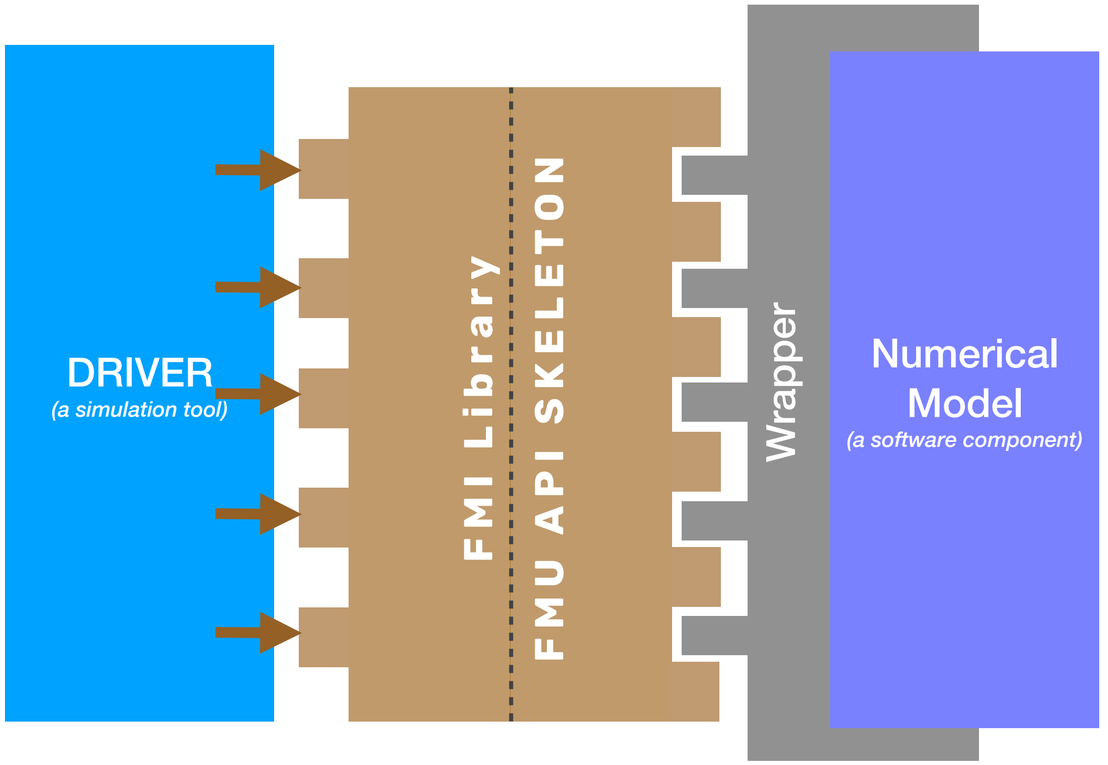
FMI CO-SIMULATION COUPLING APPROACH



Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

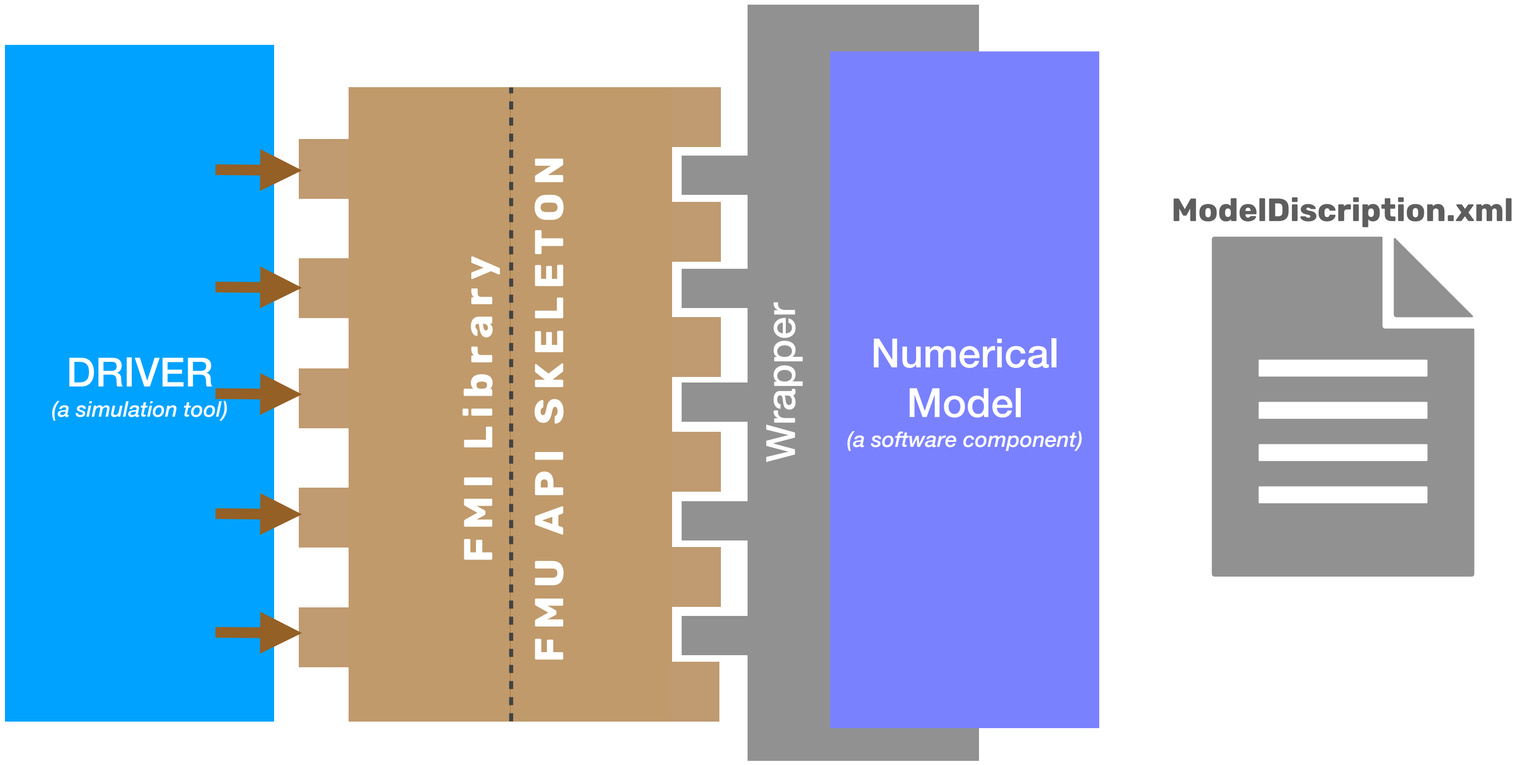
FMI CO-SIMULATION COUPLING APPROACH



Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

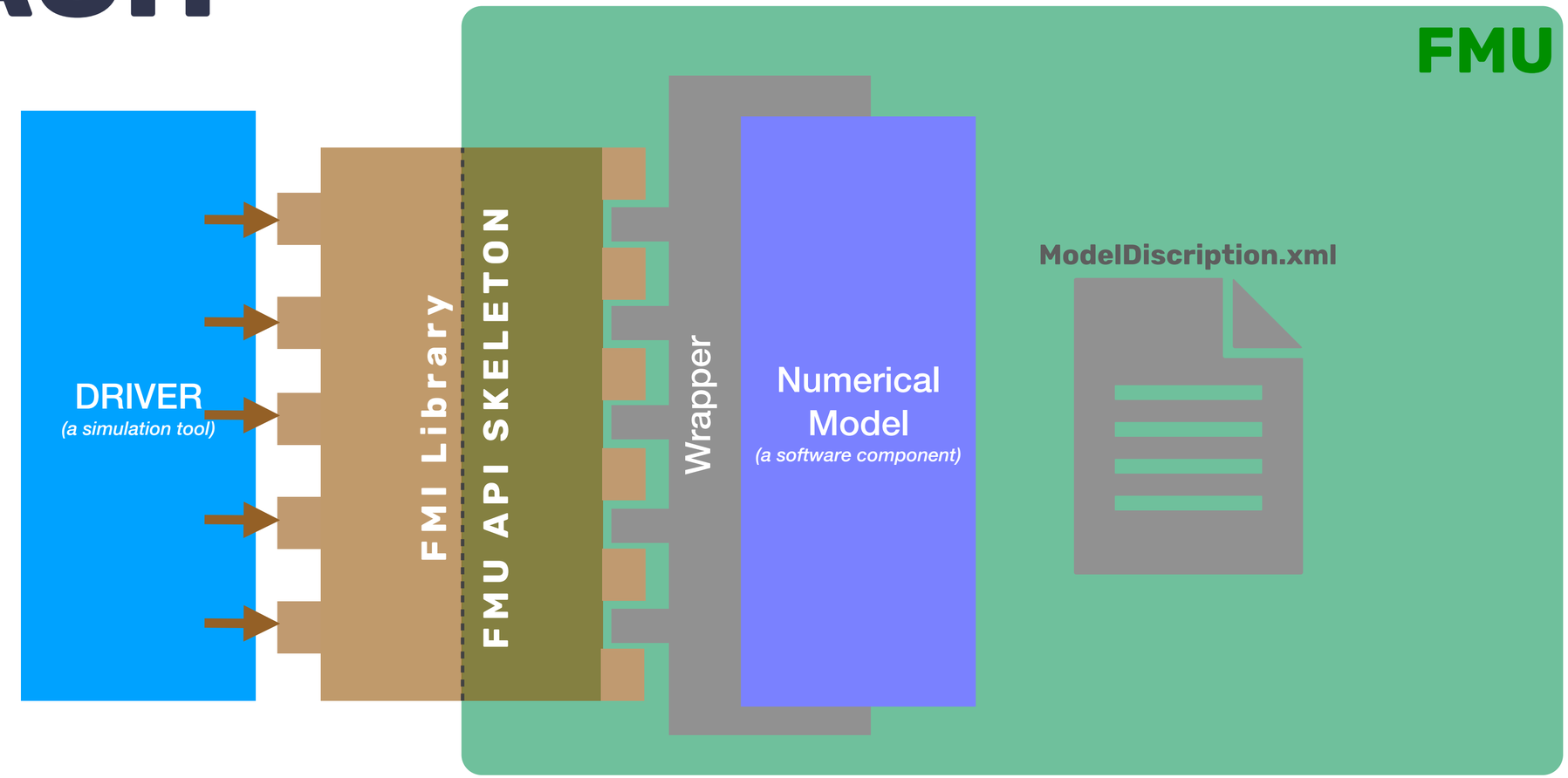
FMI CO-SIMULATION COUPLING APPROACH



Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

FMI CO-SIMULATION COUPLING APPROACH

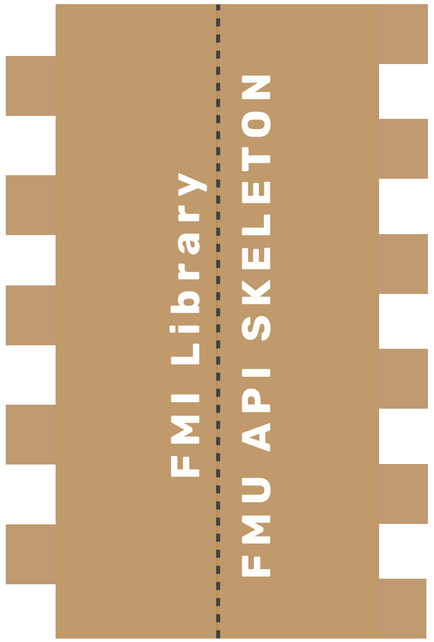


Model of
Discrete/Stochastic Behavior

Model of
Continuous/Deterministic Physics

FMI CO-SIMULATION COUPLING APPROACH

```
……
fmi2_import_get_variable_name

……

fmi2_import_get_real
fmi2_import_get_integer
……
fmi2_import_set_real
fmi2_import_set_integer
……
fmi2_import_get_fmu_state
fmi2_import_set_fmu_state
……
fmi2_import_do_step
……
```



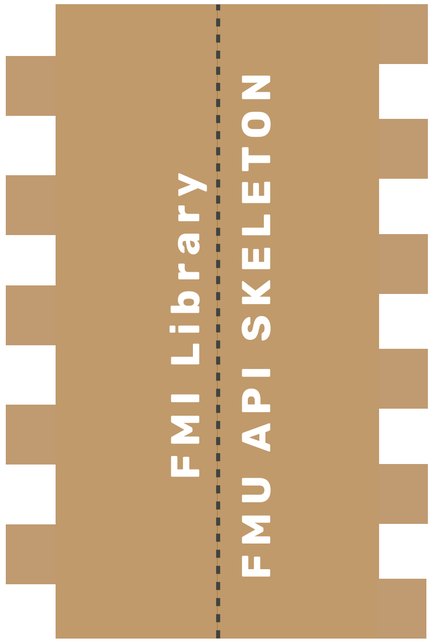

```
……

fmi2GetReal
fmi2GetInteger
……
fmi2SetReal
fmi2SetInteger
……
fmi2GetFMUstate
fmi2SetFMUstate
……
fmi2DoStep
……
```

FMI Library

FMU API Skeleton

FMI CO-SIMULATION COUPLING APPROACH

```
……
fmi2_import_get_variable_name

……

fmi2_import_get_real
fmi2_import_get_integer
……
fmi2_import_set_real
fmi2_import_set_integer
……
fmi2_import_get_fmu_state
fmi2_import_set_fmu_state
……
fmi2_import_do_step
……
```




```
……

fmi2GetReal
fmi2GetInteger
……
fmi2SetReal
fmi2SetInteger
……
fmi2GetFMUstate
fmi2SetFMUstate
……
fmi2DoStep
……
```

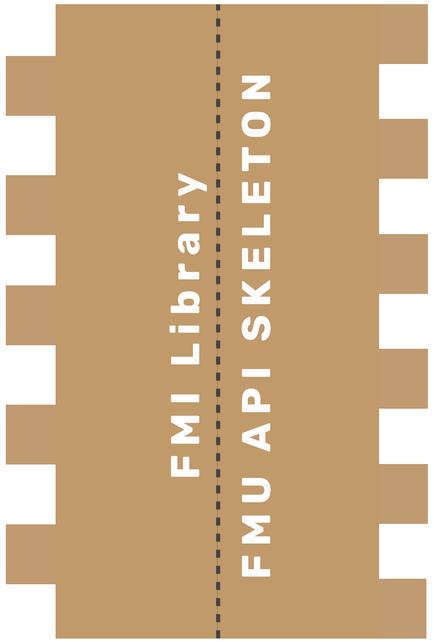
FMI Library

FMU API Skeleton

3 PYCATSHOO AND EXTERNAL MODELS

FMI CO-SIMULATION COUPLING APPROACH

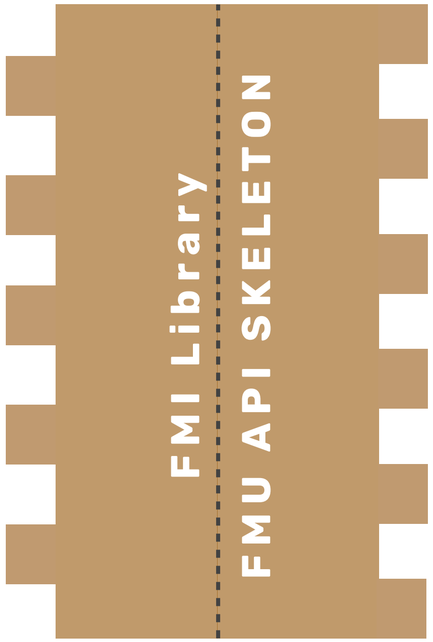
```
……
fmi2_import_get_variable_name

……

fmi2_import_get_real
fmi2_import_get_integer
……
fmi2_import_set_real
fmi2_import_set_integer
……
fmi2_import_get_fmu_state
fmi2_import_set_fmu_state
……
fmi2_import_do_step
……
```




```
……

fmi2GetReal
fmi2GetInteger
……
fmi2SetReal
fmi2SetInteger
……
fmi2GetFMUstate
fmi2SetFMUstate
……
fmi2DoStep
……
```

FMI CO-SIMULATION COUPLING APPROACH

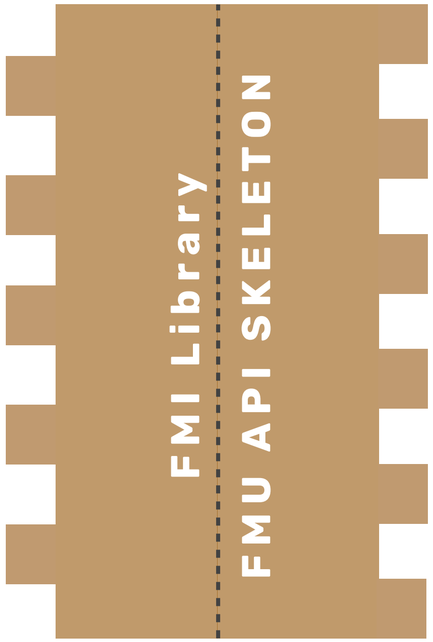
```
……
fmi2_import_get_variable_name

……

fmi2_import_get_real
fmi2_import_get_integer
……
fmi2_import_set_real
fmi2_import_set_integer
……
fmi2_import_get_fmu_state
fmi2_import_set_fmu_state
……
fmi2_import_do_step
……
```




```
……

fmi2GetReal
fmi2GetInteger
……
fmi2SetReal
fmi2SetInteger
……
fmi2GetFMUstate
fmi2SetFMUstate
……
fmi2DoStep
……
```

FMI CO-SIMULATION COUPLING APPROACH WITH PYCATSHOO

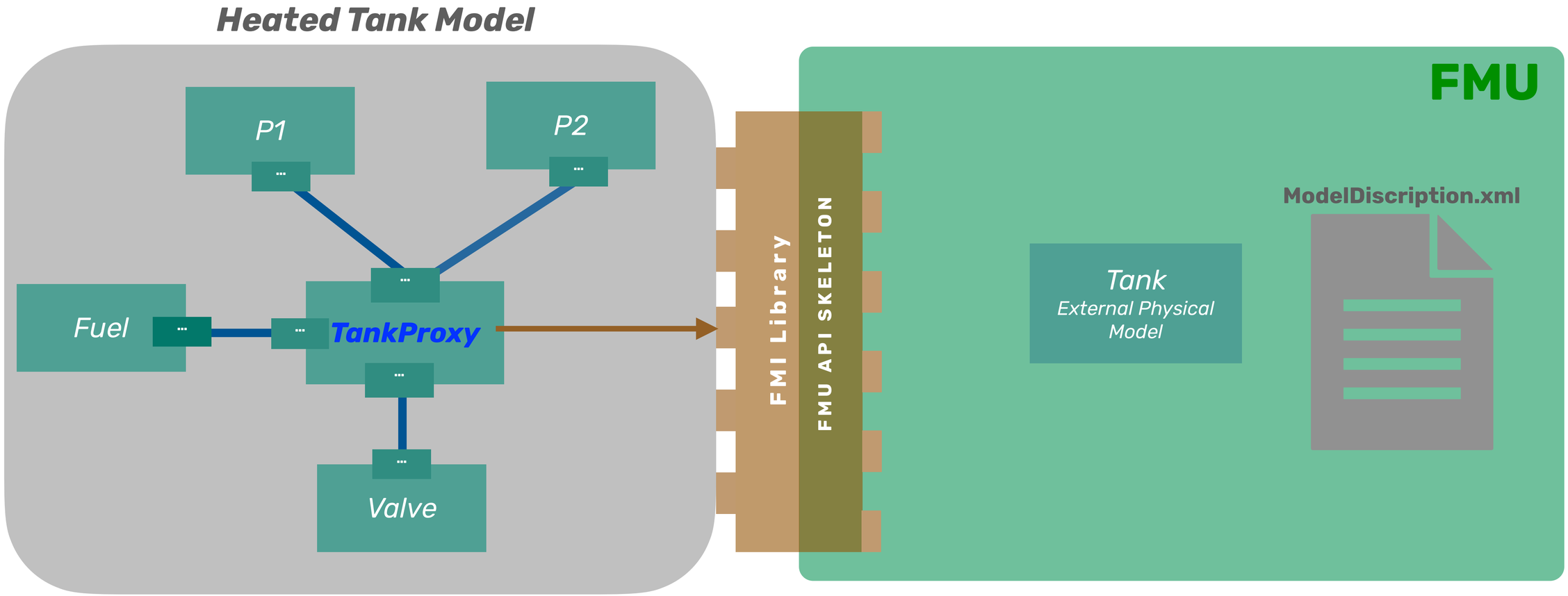
1. Load an FMU
2. Bind FMU Variables to PyCATSHOO Variables



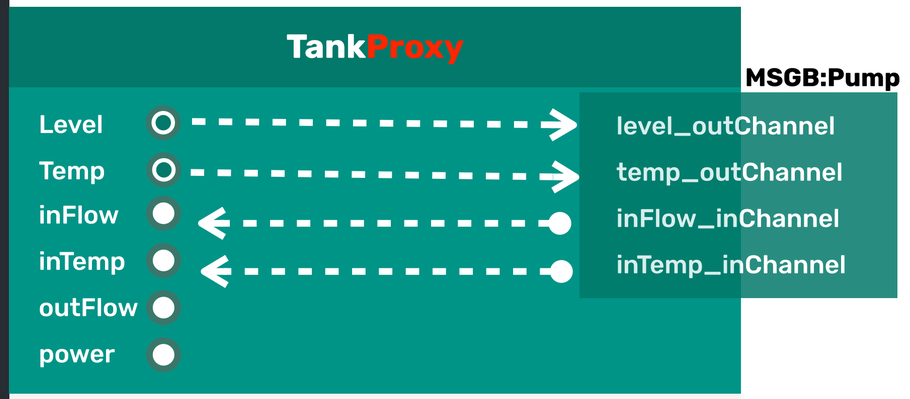
……

fmi2GetReal
fmi2GetInteger
……
fmi2SetReal
fmi2SetInteger
……
fmi2GetFMUstate
fmi2SetFMUstate
……
fmi2DoStep
……

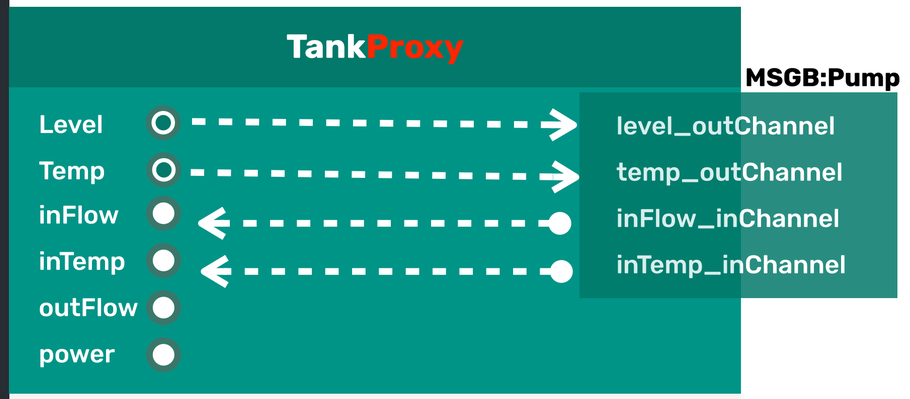
FMI CO-SIMULATION COUPLING APPROACH WITH PYCATSHOO



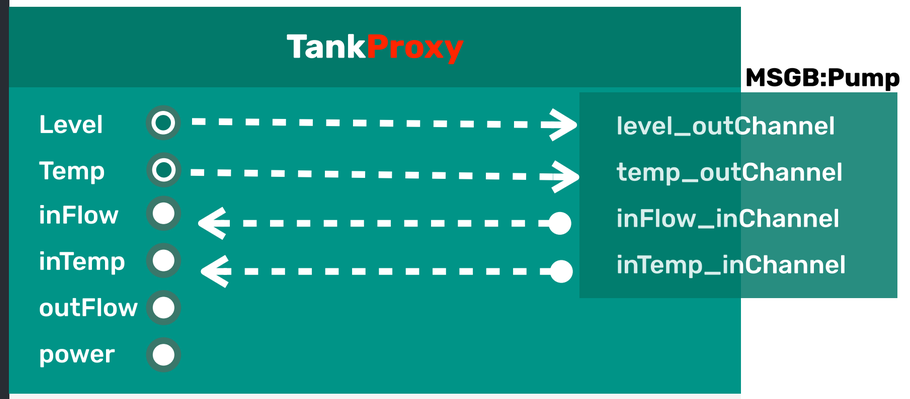
```
class TankProxy(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)
```



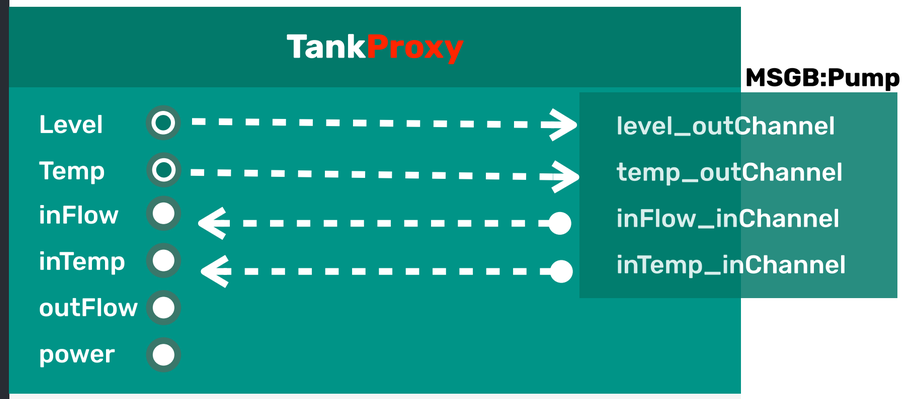
```
class TankProxy(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.fmu = self.system().addFMU(“MyFMIFolder/heatedTank.fmu”, "FMU")
```



```
class TankProxy(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.fmu = self.system().addFMU(“MyFMIFolder/heatedTank.fmu”, "FMU")

        ……
        self.v_temperature = self.fmu.variable(“FMUTank.Temperature")
        self.v_level       = self.fmu.variable("FMUTank.Level")
        self.v_fmuPower    = self.fmu.variable('FMUTank.Power')
        ……
        ……

        ……
```



```
class TankProxy(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.fmu = self.system().addFMU(“MyFMIFolder/heatedTank.fmu”, "FMU")

        ……
        self.v_temperature = self.fmu.variable(“FMUTank.Temperature")
        self.v_level       = self.fmu.variable("FMUTank.Level")
        self.v_fmuPower    = self.fmu.variable('FMUTank.Power')
        ……
        ……

        ……


        pdmp = self.addPDMPManager  ("pdmpManager")

        pdmp.addFMU( self.system().FMU(“FMU"))

        self.addPDMPBeginMethod("pdmpManager", "beginMethod",  self.beginMethod)
```

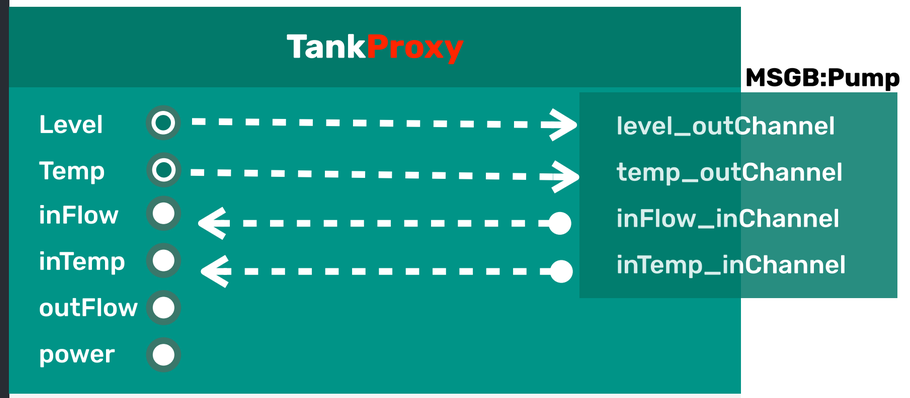


PDMP: pdmpManage

Relies On FMU for equation solving

Ensures FMU Variables initialisation

```
class TankProxy(Pyc.CComponent):
    def __init__(self, name):
        Pyc.CComponent.__init__(self, name)

        ……
        self.fmu = self.system().addFMU(“MyFMIFolder/heatedTank.fmu”, "FMU")

        ……
        self.v_temperature = self.fmu.variable(“FMUTank.Temperature")
        self.v_level       = self.fmu.variable("FMUTank.Level")
        self.v_fmuPower    = self.fmu.variable('FMUTank.Power')
        ……
        ……

        ……


        pdmp = self.addPDMPManager  ("pdmpManager")

        pdmp.addFMU( self.system().FMU(“FMU"))

        self.addPDMPBeginMethod("pdmpManager", "beginMethod",  self.beginMethod)


    def beginMethod(self):

        ……

        self.v_fmuPower.setDValue(self.r_power.sumValue());

        ……
```

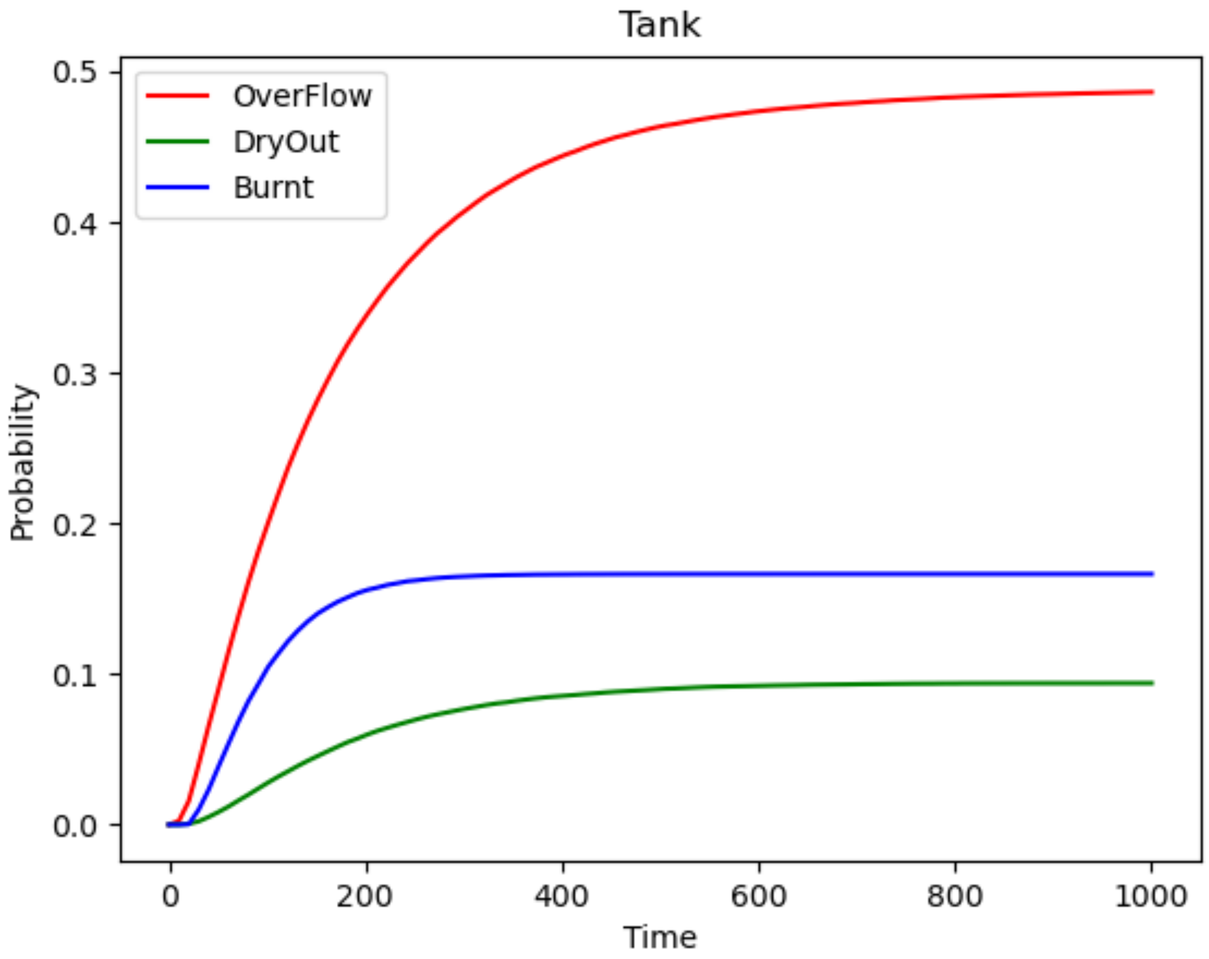


PDMP: pdmpManage

Relies On FMU for equation solving

Ensures FMU Variables initialisation

FMI CO-SIMULATION WITH PYCATSHOO: OUTPUTS



FMI CO-SIMULATION WITH PYCATSHOO: OUTPUTS

COMPUTATION TIMES FOR DIFFERENT NUMBERS OF SIMULATED SEQUENCES *(10 threads on six-core Intel® i7-8750H / 32GB)*				
Number of sequences	**10^3**	**10^4**	**10^5**	**10^6**
Full PyCATSHOO Model	1.2 s	2.20 s	16.4 s	180 s

FMI CO-SIMULATION WITH PYCATSHOO: OUTPUTS

COMPUTATION TIMES FOR DIFFERENT NUMBERS OF SIMULATED SEQUENCES *(10 threads on six-core Intel® i7-8750H / 32GB)*				
Number of sequences	**10^3**	**10^4**	**10^5**	**10^6**
Full PyCATSHOO Model	1.2 s	2.20 s	16.4 s	180 s
PyCATSHOO using external FMU	1.7 s	7.7 s	69 s	727 s

4 CONCLUSIONS AND OUTLOOK

CONCLUSIONS AND OUTLOOK

- **FMI standard in PyCATSHOO**
  - ✓ *PyCATSHOO can now use external physical models that conform to FMI*
  - ✓ *FMI API complexity taken care of by PyCATSHOO*

- **What next?**
  - ✓ *Application to an industrial-sized study: HVAC?*
  - ✓ *Exploring the potential benefits of the FMI Model Exchange approach*

PyCATSHOO.org

hassane.chraibi@edf.fr