

S⁴ Simulation System

外部接続マニュアル

2019 年 1 月更新

問合せ

Address 東京都新宿区信濃町 35 番地
信濃町煉瓦館 1 階

Phone 03-3358-6681

Fax 03-3358-1727

E-mail s4-support@msi.co.jp

目次

1.	はじめに.....	3
1.1.	外部接続について.....	3
2.	Python 呼び出し.....	4
2.1.	モデル作成.....	4
2.2.	Python プログラムの表示.....	11
2.3.	Python から呼び出す.....	12
3.	外部プログラム呼び出し.....	16
3.1.	Excel からの呼び出し.....	16

1. はじめに

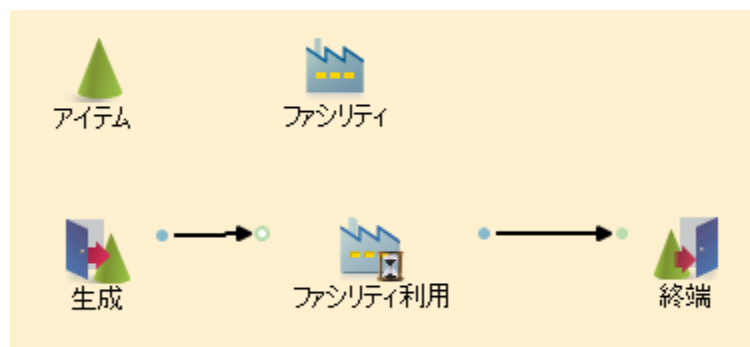
1.1. 外部接続について

S⁴ Simulation System の GUI で作成したモデルは、Python プログラムとして呼び出す事が出来ます。その為、外部ソフトから Python を実行する事で、S⁴ Simulation System を外から呼び出す事が出来ます。本マニュアルでは、その方法について記述いたします。

2. Python 呼び出し

2.1. モデル作成

まず S⁴ Simulation System を起動して、外部接続の基となるシミュレーションモデルを作成します。本マニュアルでは、下記のモデルを考えます。



生成部品からアイテムが生成され、ファシリティを利用して出て行きます。設定の詳細は下記とします。

- ・アイテムが平均 60 秒の指数分布に従って生成されます
- ・ファシリティの入力ポートのキューサイズは 10 です
- ・parameter1,parameter2 はモデルパラメータとします
- ・ファシリティの数は parameter1 です
- ・ファシリティ利用時間は平均 parameter2 秒の指数分布に従います
- ・シミュレーション時間は 100,000 秒です

シミュレーション実行後、平均待ち行列長が出力されるように、分析スクリプトを記述しておきます。

部品編集 (生成 [外部関係サンプル])

共通設定
属性設定
コード編集

説明

フローアイテムを生成します。

編集

「フローアイテム名」は、この[生成]部品が生成するフローアイテムを表します。
「初期生成時間」は、最初にフローアイテムを生成するのに要する時間を表します。
「生成時間」は、フローアイテムを生成するのに要する時間を表します。
「生成時間系列の生成」は生成時間系列の生成方法を表します。
「1回の生成数」は、1回の生成で生成されるフローアイテムの数を表します。生成数が2以上の場合はフローアイテムリストとして結合します。フローアイテムリストは分割[Split]で分割可能です。
「最大生成数」は、この[生成]部品が生成するフローアイテムの最大個数を表します。

フローアイテム名:
Item(アイテム)

初期生成時間
生成方式: 型: 値:
固定 実数 0

生成時間
生成方式: 平均(>0): 乱数の種
指数分布 60

☒ グローバル系列
☐ 独自系列 0

生成時間系列の生成:
毎回生成し直す

1回の生成数
生成方式: 型: 値:
固定 実数 1

最大生成数
☒ 無限大
☐ 指定 1

OK
キャンセル
適用

図. 生成部品

部品編集 (ファシリティ利用 [外部関係サンプル])

共通設定 属性設定 コード編集

ファシリティリスト

+ rFacility(ファシリティ) ↑ ↓

待ち受け方法:
AND

ファシリティの利用時間

生成方式: 平均(>0): 乱数の種

指数分布 param.parameter2 グローバル系列
独自系列 0

優先度

生成方式: 型: 値:

固定 実数 0.0

最大待ち時間

☒ 指定しない
☐ 最大待ち時間指定 最大待ち時間: タイムアウト時出力ポート:

0

並列処理

☐ 逐次処理
☒ 並列処理 並列数(>0):

self.param.parameter1

ファシリティ利用の記録:

なし

OK キャンセル 適用

図. ファシリティ利用(属性設定)

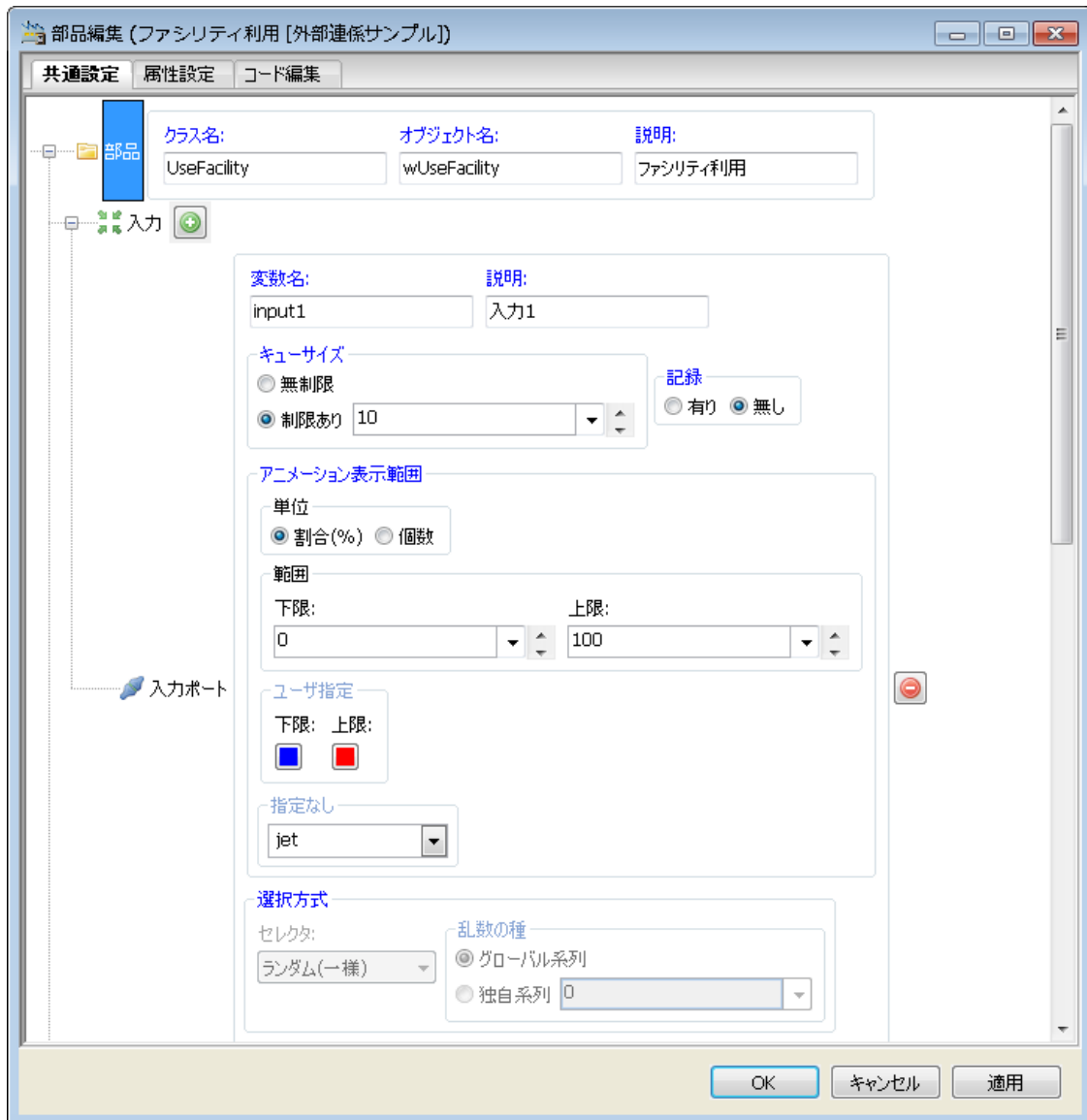


図. ファシリティ利用(キューサイズ設定)

パラメータ (外部関係サンプル)

基本設定 パラメータ 一括実行 感度分析 実験計画 最適化

実行種類:
通常実行

シミュレーション開始時間
☒ 指定無し
☐ 時刻指定 2019/01/28 0 時 0 分 0 秒 0 マイクロ秒

シミュレーション終了時間
☐ 指定無し
☐ 時刻指定 2019/01/28 0 時 0 分 0 秒 0 マイクロ秒
☒ 時間指定(秒間) 100000.0

ウォームアップ時間
☐ 指定無し
☐ 時刻指定 2019/01/28 0 時 0 分 0 秒 0 マイクロ秒
☒ 時間指定(秒間) 0.0

シミュレーション時刻フォーマット(%Y:年 %m:月 %d:日 %H:時 %M:分 %S:秒 %f:マイクロ秒):
 %Y-%m-%d %H:%M:%S.%f

乱数の種(グローバル)
☐ 設定しない
☒ 設定する 0

入力フォルダ:
 default

OK キャンセル 適用

図. パラメータ設定(シミュレーション時間)

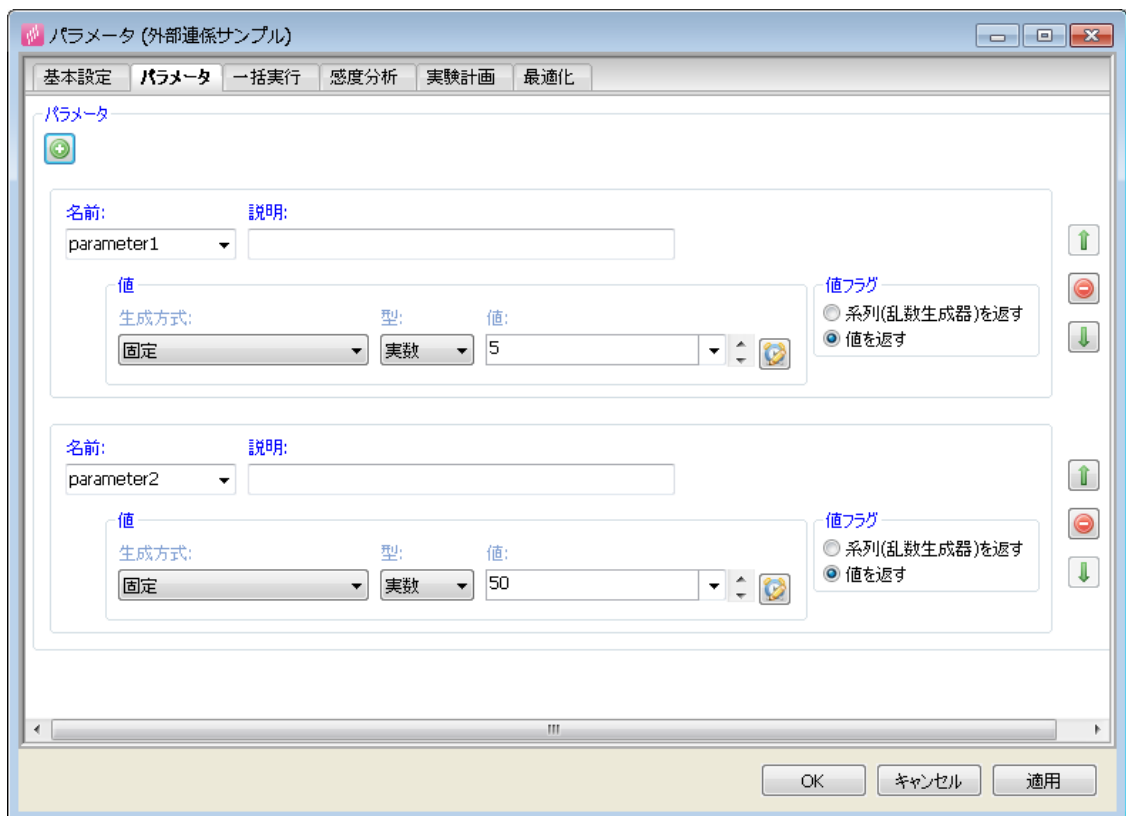


図. パラメータ設定(モデルパラメータ)

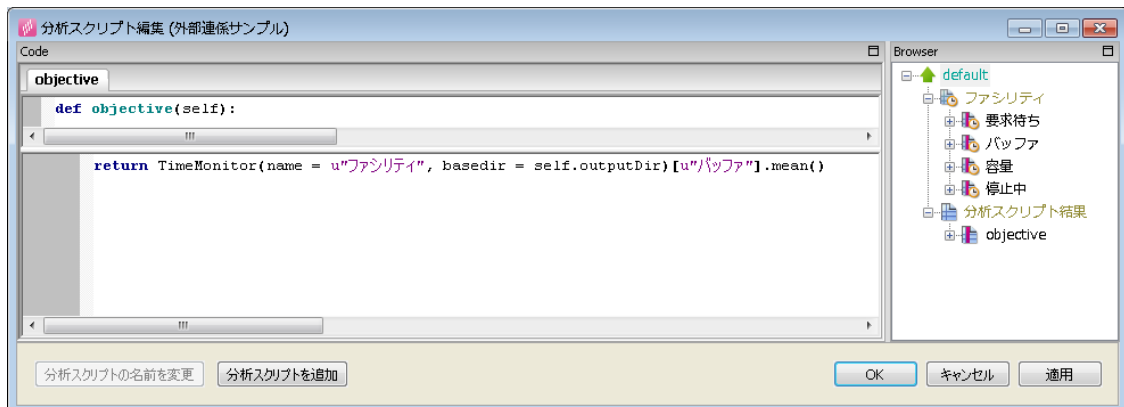


図. 分析スクリプト設定

2.2. Python プログラムの表示

シミュレーションモデルが作成できたら、モデルの Python プログラムを表示します。Python プログラムを表示するには、ツールバーの「C」をクリックします。

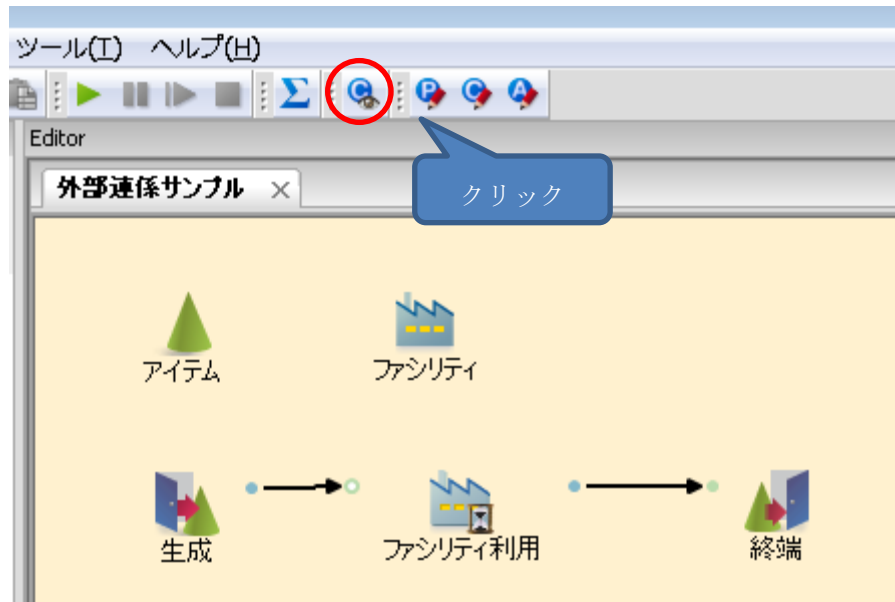
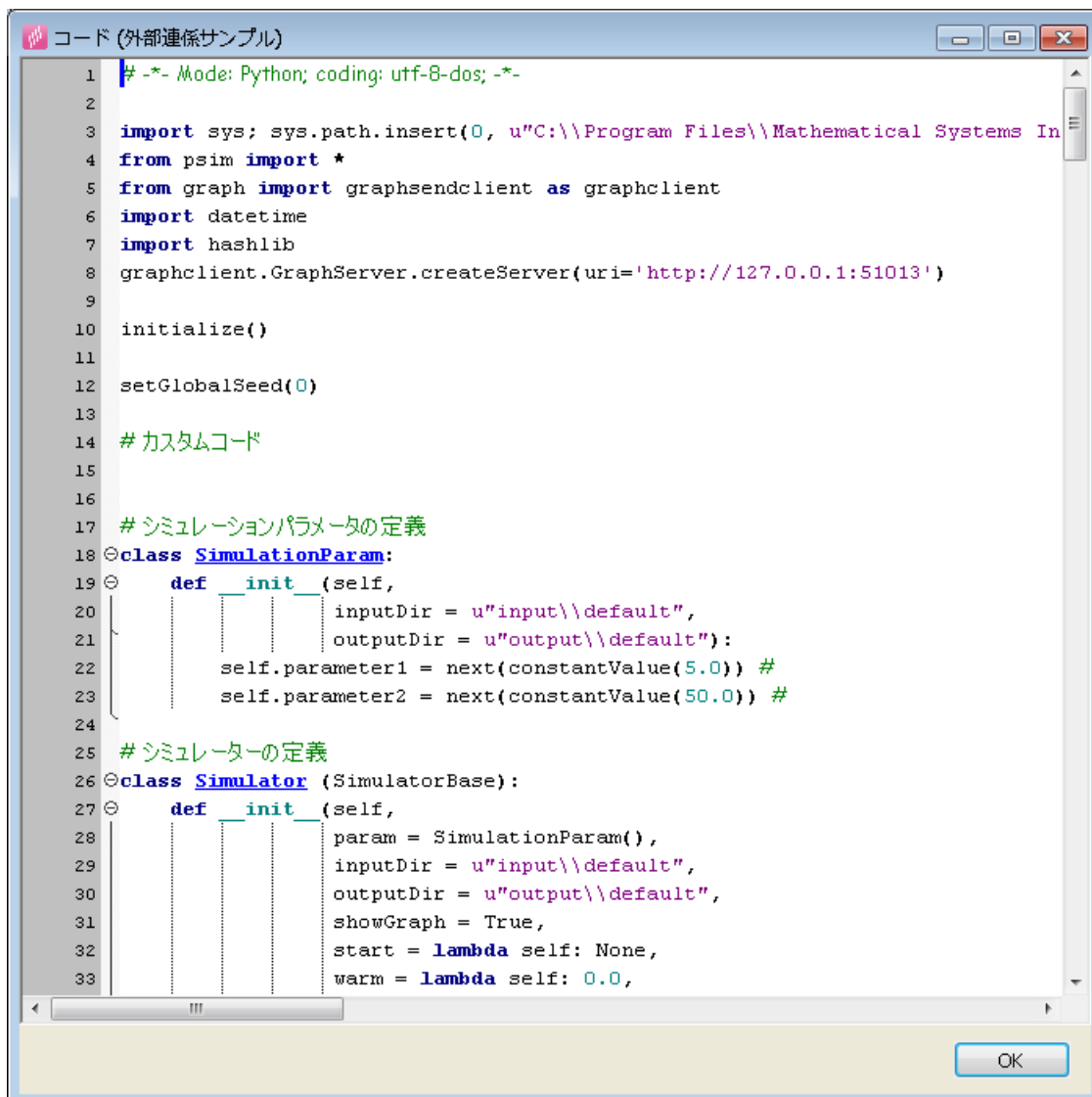


図. Python プログラムの表示



```
コード (外部関係サンプル)
1  -*- Mode: Python; coding: utf-8-dos; -*-
2
3  import sys; sys.path.insert(0, u"C:\\Program Files\\Mathematical Systems Inc
4  from psim import *
5  from graph import graphsendclient as graphclient
6  import datetime
7  import hashlib
8  graphclient.GraphServer.createServer(uri='http://127.0.0.1:51013')
9
10 initialize()
11
12 setGlobalSeed(0)
13
14 # カスタムコード
15
16
17 # シミュレーションパラメータの定義
18 class SimulationParam:
19     def __init__(self,
20                 inputDir = u"input\\default",
21                 outputDir = u"output\\default"):
22         self.parameter1 = next(constantValue(5.0)) #
23         self.parameter2 = next(constantValue(50.0)) #
24
25 # シミュレータの定義
26 class Simulator (SimulatorBase):
27     def __init__(self,
28                 param = SimulationParam(),
29                 inputDir = u"input\\default",
30                 outputDir = u"output\\default",
31                 showGraph = True,
32                 start = lambda self: None,
33                 warm = lambda self: 0.0,
```

図. 表示されたプログラム

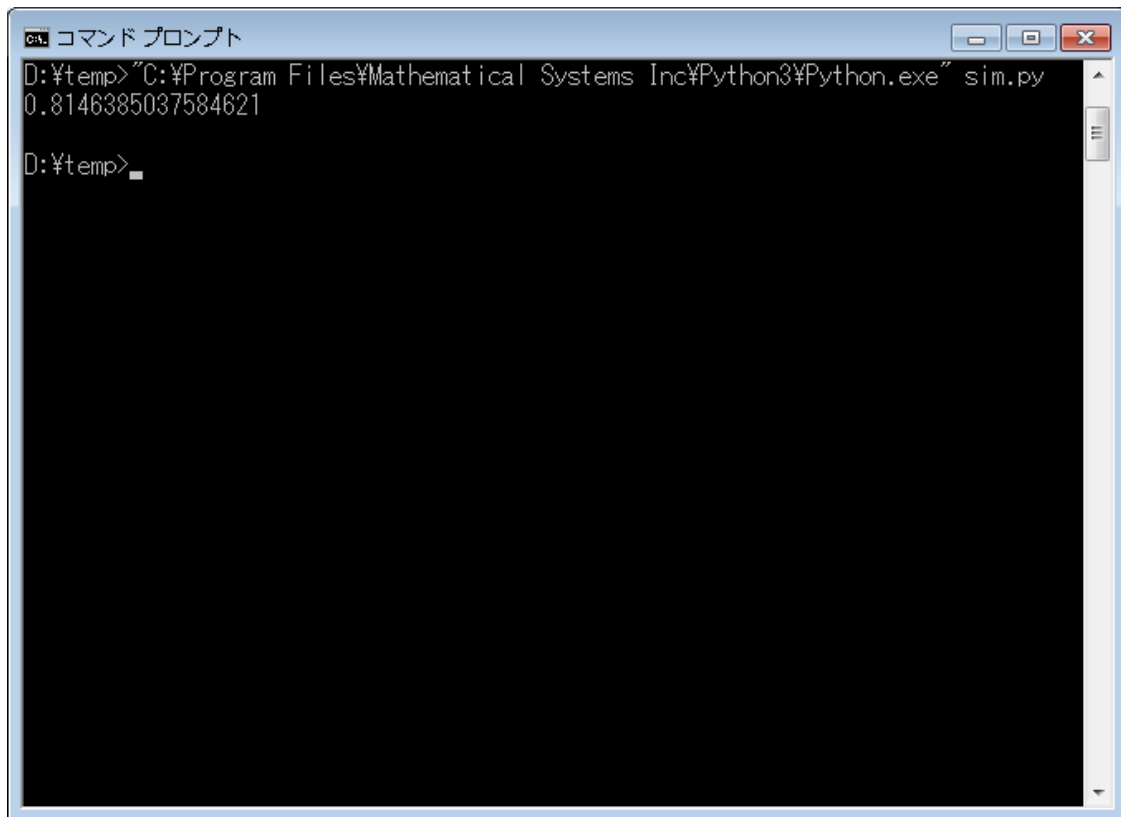
2.3. Python から呼び出す

2.2 で表示されたプログラムを Python から呼び出します。まず、表示されたプログラムをメモ帳などにコピー&ペーストし、ファイルに保存します。ここでは、sim.py と名前を付けておきます。dos コマンドから、次のコマンドを実行し、Python からシミュレーションを実行します。

```
"C:\Program Files\Mathematical Systems Inc\Python3\Python.exe" sim.py
```

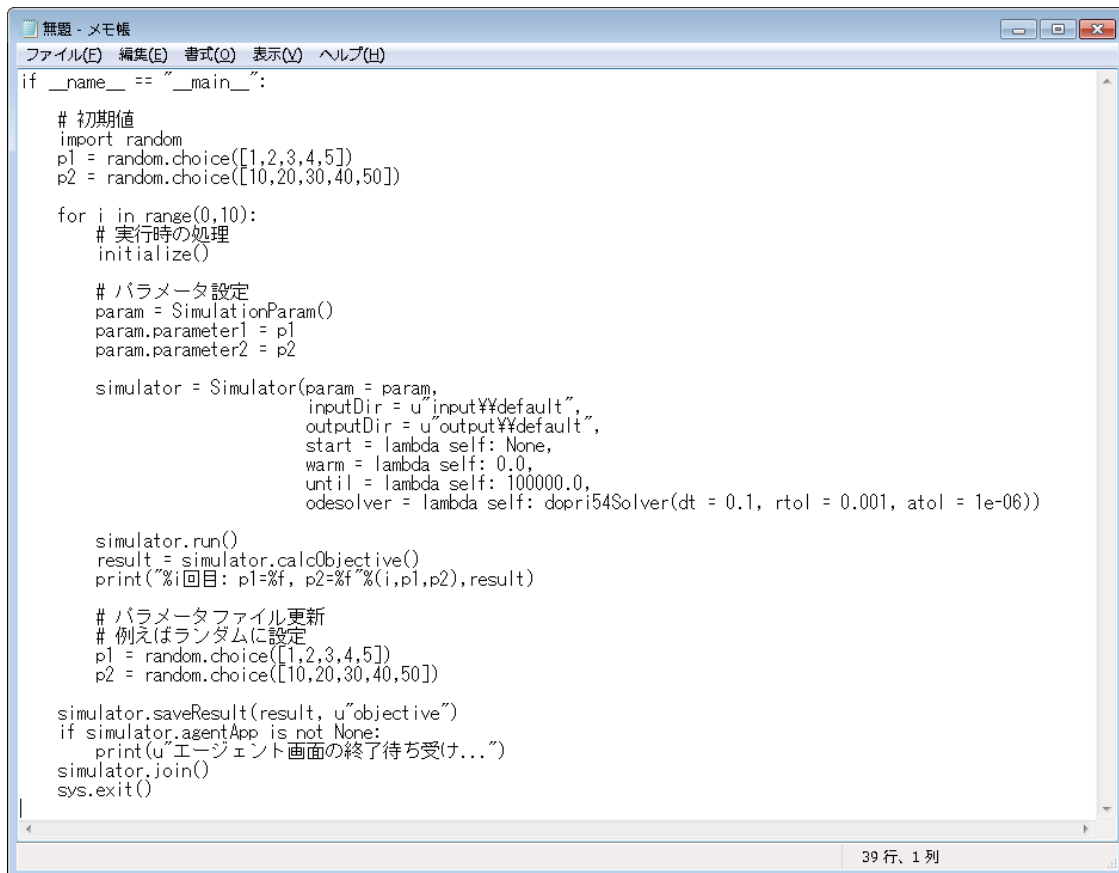
「C:\Program Files\Mathematical Systems Inc」は S⁴ Simulation System がインストール

ールされているフォルダです。シミュレーションが終了すると、コマンドプロンプトにシミュレーション結果（平均待ち行列長）が出力されます。



```
コマンド プロンプト
D:¥temp>"C:¥Program Files¥Mathematical Systems Inc¥Python3¥Python.exe" sim.py
0.8146385037584621
D:¥temp>_
```

sim.py は Python プログラムですので、コードを自由に編集する事ができます。例えば、parameter1 と parameter2 の値をランダムに変えながら、実行するプログラムは main を次のように書き換えることで実現できます(sim_iterative.py)。



```
if __name__ == "__main__":

    # 初期値
    import random
    p1 = random.choice([1,2,3,4,5])
    p2 = random.choice([10,20,30,40,50])

    for i in range(0,10):
        # 実行時の処理
        initialize()

        # パラメータ設定
        param = SimulationParam()
        param.parameter1 = p1
        param.parameter2 = p2

        simulator = Simulator(param = param,
                              inputDir = u"input¥¥default",
                              outputDir = u"output¥¥default",
                              start = lambda self: None,
                              warm = lambda self: 0.0,
                              until = lambda self: 100000.0,
                              odesolver = lambda self: dopri54Solver(dt = 0.1, rtol = 0.001, atol = 1e-06))

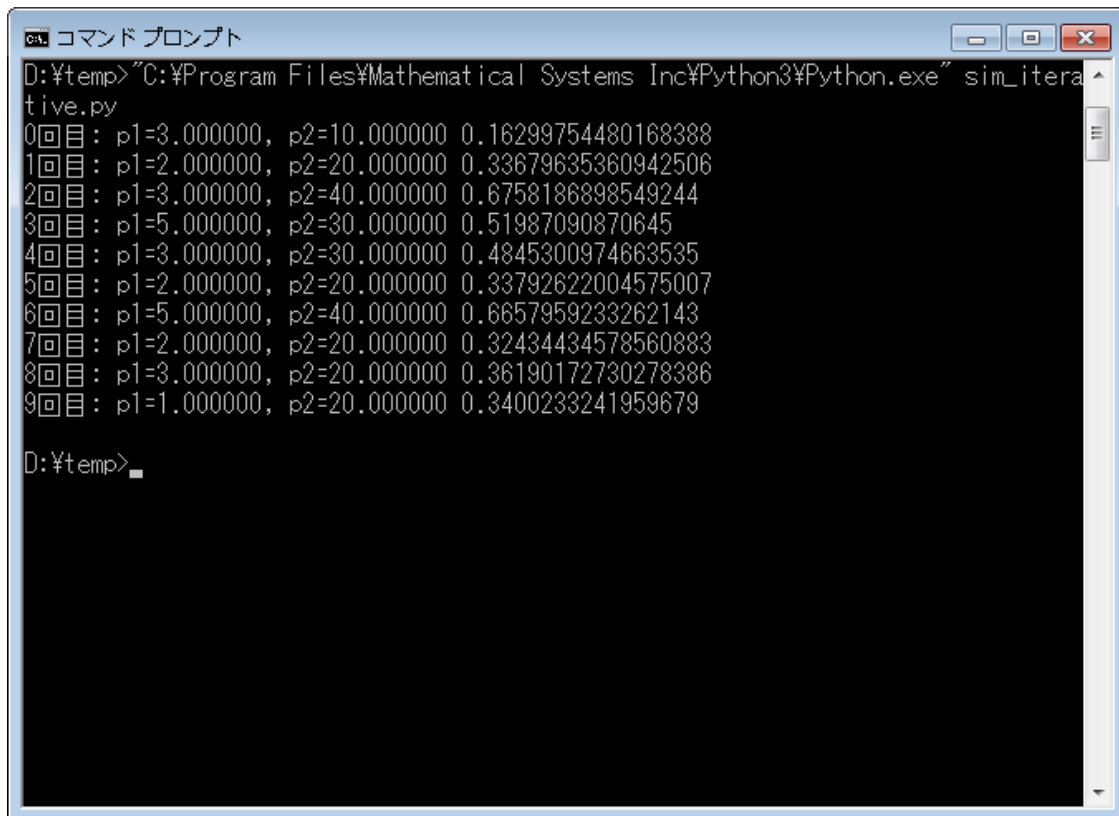
        simulator.run()
        result = simulator.calcObjective()
        print("%i回目: p1=%f, p2=%f"%(i,p1,p2),result)

        # パラメータファイル更新
        # 例えばランダムに設定
        p1 = random.choice([1,2,3,4,5])
        p2 = random.choice([10,20,30,40,50])

    simulator.saveResult(result, u"objective")
    if simulator.agentApp is not None:
        print(u"エージェント画面の終了待ち受け...")
    simulator.join()
    sys.exit()
```

39 行、1 列

図. パラメータを変更して実行



```
コマンド プロンプト
D:\temp>"C:\Program Files\Mathematical Systems Inc\Python3\Python.exe" sim_iterative.py
0回目: p1=3.000000, p2=10.000000 0.16299754480168388
1回目: p1=2.000000, p2=20.000000 0.33679635360942506
2回目: p1=3.000000, p2=40.000000 0.6758186898549244
3回目: p1=5.000000, p2=30.000000 0.51987090870645
4回目: p1=3.000000, p2=30.000000 0.4845300974663535
5回目: p1=2.000000, p2=20.000000 0.33792622004575007
6回目: p1=5.000000, p2=40.000000 0.6657959233262143
7回目: p1=2.000000, p2=20.000000 0.32434434578560883
8回目: p1=3.000000, p2=20.000000 0.36190172730278386
9回目: p1=1.000000, p2=20.000000 0.3400233241959679
D:\temp>
```

図. 繰り返し実行結果

3. 外部プログラム呼び出し

3.1. Excel からの呼び出し

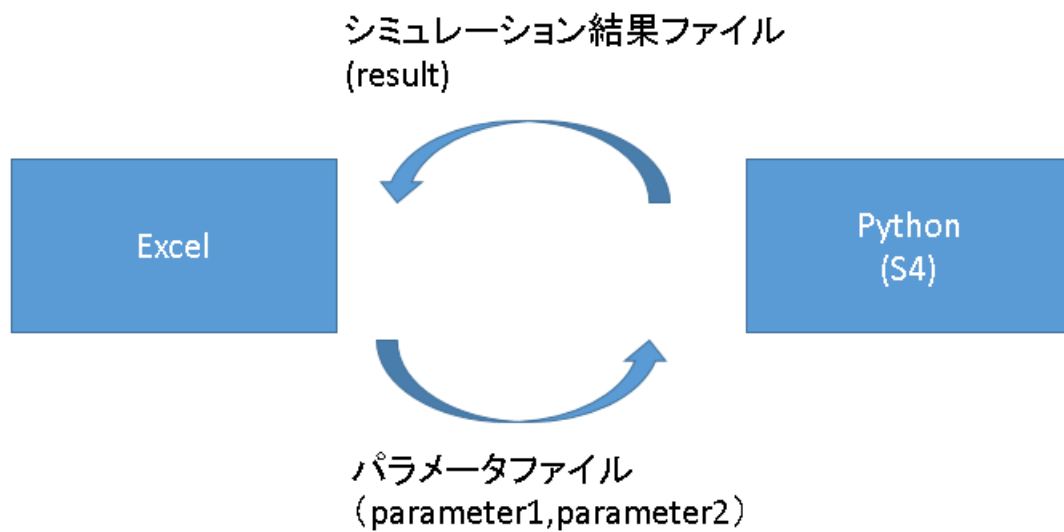
2 で作成したプログラムは Python プログラムですので、Python を呼び出せる外部プログラムから呼び出す事が可能です。ここでは ExcelVBA からの呼び出しを紹介しますが、VBA だけでなく、C やその他の言語も同様な方法で呼び出す事ができます。

	A	B	C	D	E
1	parameter1	parameter2	result		
2	1	10			
3	2	20			
4	3	30			
5	4	40			
6	5	50			
7					
8					
9					

実行

図.Excel からの呼び出し例

ここでは、例として実行ボタンを押すと、parameter1、parameter2 のセルに入力されている値でシミュレーションし、結果(平均待ち行列長)を result に表示するプログラムを考えます。この際、Python プログラムとの parameter と result のやり取りは、ファイル経由で行います。Excel ではセルに入力されている値を、逐一パラメータファイルに出力します。Python では Excel が出力したパラメータファイルを読み込み、シミュレーションを実行します。その際、Python プログラムで、シミュレーション結果をファイルに出力するようにしておきます。Excel はシミュレーション結果のファイルを読み込み、result に表示します。



このとき、呼び出す Python プログラムは、2 章で作成したプログラムで `main` を以下のように変えているプログラムを呼び出します(`sim_excel.py`)。



```
if __name__ == "__main__":
    # パラメーターファイルへのパス
    parameterFile = u"D:¥¥temp¥¥prm.csv"
    resultFile = u"D:¥¥temp¥¥result.csv"

    def readPrm(fname):
        f = open(fname, "r")
        data = []
        for l in f:
            ls=l.split(",")
            data[ls[0].strip()]=ls[1].strip()
        f.close()
        return data

    def writeResult(fname,p):
        f = open(fname, "w")
        f.write("%f¥n" % p)
        f.close()

    # 初期値
    data = readPrm(parameterFile)
    param = SimulationParam()
    param.parameter1 = data["parameter1"]
    param.parameter2 = data["parameter2"]

    # 実行時の処理
    simulator = Simulator(param = param,
        inputDir = u"input¥¥default",
        outputDir = u"output¥¥default",
        start = lambda self: None,
        warm = lambda self: 0.0,
        until = lambda self: 100000.0,
        odesolver = lambda self: dopri54Solver(dt = 0.1, rtol = 0.001, atol = 1e-06))

    simulator.run()
    result = simulator.calcObjective()
    print(result)

    writeResult(resultFile,result)

    simulator.saveResult(result, u"objective")
    if simulator.agentApp is not None:
        print(u"エージェント画面の終了待ち受け...")
    simulator.join()
    sys.exit()
```

2 行, 1 列

Excel VBA では、次のような VBA プログラムを記述します(excelvba.xlsm)。Shell 関数を用いて Python を実行しています。

```

Sub simRun(pgFile As String)
    Const PROCESS_ALL_ACCESS As Long = &H1F0FFF
    Const INFINITE As Long = &HFFFF
    Dim TaskId As Long
    Dim hProc As Long

    TaskId = Shell("C:\Program Files\Mathematical Systems Inc\Python3\Python.exe " & pgFile, vbMinimizedFocus)
    hProc = OpenProcess(PROCESS_ALL_ACCESS, 0, TaskId)
    If hProc <> 0 Then
        Call WaitForSingleObject(hProc, INFINITE)
        CloseHandle hProc
    End If
End Sub

Sub prog1()
    prmFile = "D:\temp\prm.csv"
    resFile = "D:\temp\result.csv"
    pgFile = "D:\temp\sim_excel.py"

    Dim i As Long
    For i = 2 To 6
        p1 = Cells(i, 1)
        p2 = Cells(i, 2)

        Open prmFile For Output As #1
        Print #1, "parameter1" & "," & p1
        Print #1, "parameter2" & "," & p2
        Close #1

        simRun (pgFile)

        Open resFile For Input As #2
        Input #2, res
        Cells(i, 3) = res
        Close #2
    Next i
End Sub

```

図.VBA マクロ

```

Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal dwMilliseconds As Long) As Long
Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long
Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

```

図.標準モジュール

	A	B	C	D	E
1	parameter1	parameter2	result		
2	1	10	0.159082		
3	2	20	0.299085		
4	3	30	0.424221		
5	4	40	0.522924		
6	5	50	0.600271		
7					
8					
9					

実行

図.実行結果

実行ボタンを押すと結果が上図のように表示されます。