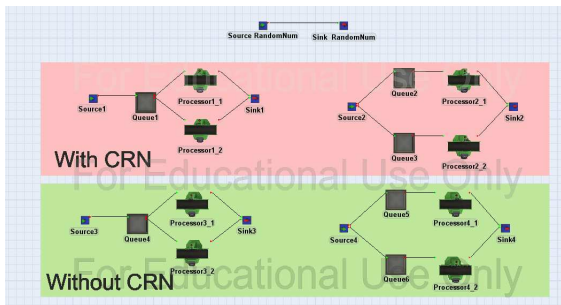


SubPrograms: “User Commands” in FlexSim

W. M. Song 桑慧敏
Tsing Hua Univ. 清華大學

2015.12.02

Motivation of Using Subprograms



- Q: How can we avoid repeating the same statements in coding. For example: We need to generate "Weibull data via inverse cdf" in all "Sources" and "Processors" **many times**. Also, we need to do "computation" in "Sink" **many times**.
- A: Via the idea of "Subprograms"

Main Program vs. Subprograms

Main Program (主程序)

Call

- Subprogram 1 (子程序 1)
- Subprogram 2 (子程序 2)

Subprogram 1

Ex. Subroutine 1:
"InverseWeibull"

Subprogram 2

Ex. Subroutine 2:
"Computation"

- Subprograms helps to build a better structure (結構) of a main program

Subprograms in FlexSim

Subprograms in “User Commands”

- **Goal:** Construct subprograms in FlexSim
- **FlexSim:** Tools → User Commands → Add
- **Name:** eg. InverseWeibull
- **Parameters:** (num a, num b, num u),
- **Return Type:** num
- **Description:** For user's reference

Return Type:

- “num” (for integer or double)
- “str” (for stream)
- “node” (for treenode)

“Subprogram” in FlexSim

- Tools → User Commands → Add
- Name: InverseWeibull

Parameters: (num a, num b, num u), Return Type: num

Subprogram in FlexSim: InverseWeibull

The image shows two windows from the FlexSim software. The left window is titled "User Commands" and shows a list of commands on the left with "InverseWeibull" selected. The main area is an "Edit" dialog for "InverseWeibull". The fields are: Name: InverseWeibull; Parameters: (num a, num b, num u); Return Type: num; Description: input value : alpha,beta,random seed; use c.d.f of weibull and uniform to generate weibull's p.d.f return value of Weibull(a,b); Example: InverseWeibull(alpha,beta,random seed); Code: Custom Code. The "Add" button at the bottom left is highlighted with a red box. The right window is titled "User Command - InverseWeibull" and shows the code for the command, which is also highlighted with a red box:

```
1 /**Custom Code*/  
2  
3 double a=parval(1);  
4 double b=parval(2);  
5 double u=parval(3);  
6  
7 return -(b*pow((log(1-u)), (1/a)));  
8  
9
```

Inter-Arrivaltime of Source

- Source1 → Source → Inter-Arrivaltime

```
Source1 - Inter-Arrival Time  
1 /**Custom Code*/  
2 treenode current = ownerobject(c);  
3  
4 double a = SourceA;  
5 double b = SourceB;  
6 double c = gettablenum("RandomNumber", getoutput(current)+1,1);  
7  
8 return InverseWeibull(a,b,c);
```

Global Table - RandomNumber

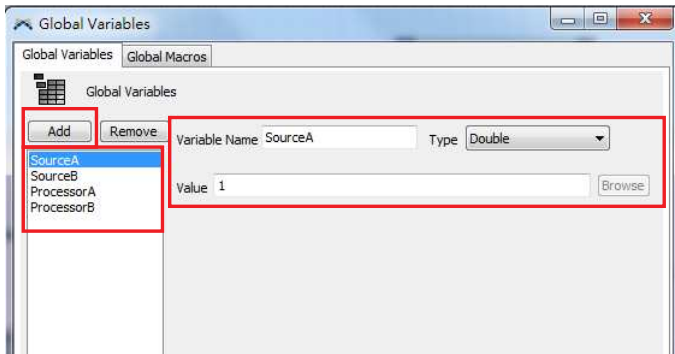
Name: RandomNumber

	Random_Source	Random_P
		0.42
Row 2		0.45
Row 3		0.67
Row 4		0.24
Row 5		0.76
Row 6		0.42
Row 7		0.23
Row 8		0.25

- Note: We use "Global Variables" in FlexSim to define value globally (see next page)

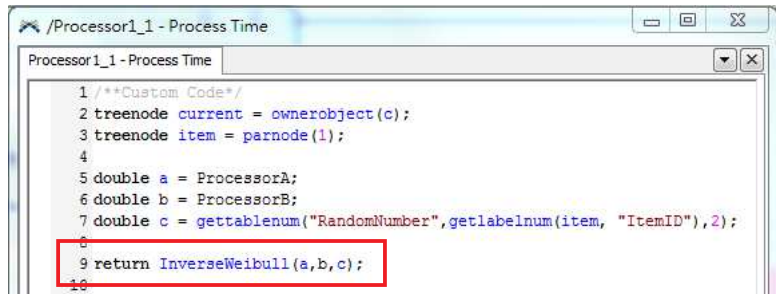
Global Variables

- Tools → Global Variables → Add
 - Variable Name: **SourceA** , Type: Double , Value=1
 - Variable Name: **SourceB** , Type: Double , Value=2
 - Variable Name: **ProcessorA** , Type: Double , Value=1
 - Variable Name: **ProcessorB** , Type: Double , Value=1



Process Time of Processor

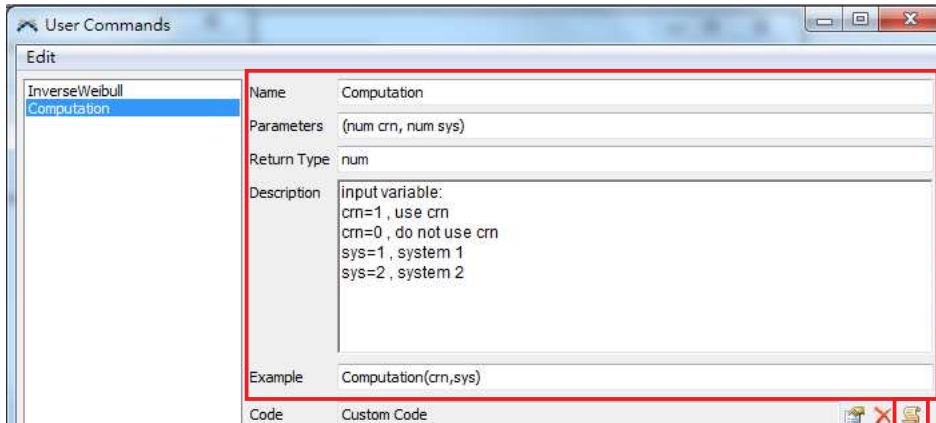
- Processor1_1 → Processor → Process Time



```
1 /**Custom Code*/
2 treenode current = ownerobject(c);
3 treenode item = parnode(1);
4
5 double a = ProcessorA;
6 double b = ProcessorB;
7 double c = gettablenum("RandomNumber",getlabelnum(item, "ItemID"),2);
8
9 return InverseWeibull(a,b,c);
10
```


User Commands: Computation

- Tools → User Commands → Add
- Name: **Computation** , Parameters: (num crn, num sys), Return Type: num



Computation (crn, sys)

```
User Command - Computation
Computation
1 /**Custom Code*/
2 int crn=parval(1);
3 int sys=parval(2);
4 double Sum=0;
5 double SumW=0;
6 double SumWQ1=0;
7 double SumWQ2=0;
8 string WQt;
9 int EnterCol;
10 int LeaveCol;
11 int WqCol;
12
13 if(crn==1)
14 {
15     WQt="WQtableCRN";
16 }
17 else
18 {
19     crn=2;
20     WQt="WQtable";
```

```
34
35     settablenum("IndexCheck",crn,1,gettablenum("IndexCheck",
36 //calculate WQ
37 for(int k=1 ;k<=gettablenum("InitialValues",1,1)+gettabl
38 {
39     settablenum(WQt,k,WqCol,gettablenum(WQt,k,LeaveCol)-
40 }
41 //calculate average WQ
42 for(int k=gettablenum("InitialValues",2,1)+1 ;k<=gettabl
43 {
44     SumWQ1 = SumWQ1 + gettablenum(WQt,k,WqCol);
45 }
46 settablenum("WQ",crn,sys,SumWQ1/gettablenum("InitialValu
47 for(int k=gettablenum("InitialValues",3,1)+1 ;k<=gettabl
48 {
49     SumWQ2 = SumWQ2 + gettablenum(WQt,k,WqCol);
50 }
51 settablenum("WQ",crn,sys+2,SumWQ2/gettablenum("InitialVa
52
53 if(gettablenum("IndexCheck",crn,1)==2)
54 {
55     //calculate differ
56     for(int k=1 ;k<=gettablenum("InitialValues",1,1)+get
```

Computation (crn, sys) - With CRN

```
/Sink1 - OnEntry  
Sink1 - OnEntry  
1 /**Custom Code*/  
2 treenode item = parnode(1);  
3 treenode current = ownerobject(c);  
4 int port = parval(2);  
5  
6 if (getinput(current) == gettablenum("InitialValues",1,1)+gettablenum("InitialValues",3,1))  
7 {  
8     Computation(1,1);  
9 }
```

```
/Sink2 - OnEntry  
Sink2 - OnEntry  
1 /**Custom Code*/  
2 treenode item = parnode(1);  
3 treenode current = ownerobject(c);  
4 int port = parval(2);  
5 if (getinput(current) == gettablenum("InitialValues",1,1)+gettablenum("InitialValues",3,1))  
6 {  
7     Computation(1,2);  
8 }
```

Computation (crn, sys) - Without CRN

```
/Sink3 - OnEntry  
Sink3 - OnEntry  
1 /**Custom Code*/  
2 treenode item = parnode(1);  
3 treenode current = ownerobject(c);  
4 int port = parval(2);  
5  
6 if (getinput(current) == gettablenum("InitialValues",1,1)+gettablenum("InitialValues",3,1))  
7 {  
8   Computation(0,1);  
9 }
```

```
/Sink4 - OnEntry  
Sink4 - OnEntry  
1 /**Custom Code*/  
2 treenode item = parnode(1);  
3 treenode current = ownerobject(c);  
4 int port = parval(2);  
5  
6 if (getinput(current) == gettablenum("InitialValues",1,1)+gettablenum("InitialValues",3,1))  
7 {  
8   Computation(0,2);  
9 }
```