GLIMSO

(General Language to Integrate Modeling, Simulation and Optimization)

Hasperué, W. and J. Rabinovich
2014

GLIMSO was developed by Dr. Waldo Hasperué and Dr. Jorge Rabinovich, at the CEPAVE research center (Center for the Study of Parasites and Vectors), of the National University of La Plata, La Plata, Argentina., for their own use in scientific research.

The following constitute a brief description of the main commands as well as some clarification of the language usage.

1. General comment on the use of parentheses and brackets:

- The use of parentheses is to make reference to elements of matrices and vectors.
- The use of square brackets is to make reference to values at simulation time.

2. Section "Parameters"

p = 2

p = condazar (u; 0; 1;> = 0.1; 4; 5)

Choice of a uniform random number between 0 and 1. If the number chosen at random is greater than or equal to 0.1 the parameter takes the value 4, otherwise it takes the value 5. This parameter is assigned a random value in each loop of the simulation time.

p = condazar (u; 0; 1;> = 0.1; 4;> 0; 5; ...; 6)

Choice of a uniform random number between 0 and 1. If the number chosen at random is greater than or equal to 0.1 the parameter takes the value 4, otherwise if it is greater than 0 it takes the value 5, otherwise it takes the value value 6. You can have as many condition-value pairs as needed. This parameter is assigned a random value in each loop. simulation time.

p = dn (20; 5)

Choice of a random number with normal distribution with mean 20 and standard deviation 5. This parameter is assigned a random value in each loop of the simulation time) sj (11) = 0.42

Declaration of a vector of 11 elements all initialized to 0.42.

fecs (8) = [3 10 0.74 0 0 0 9.32 -0.12324]

Declaration of a vector of 8 elements each initialized with a different value, the values can be separated by space, tab or comma and could even be written in vertical one below the other).

3. Section "Variables"

a = 3

av (32) = 0

Declaration of a vector of 32 elements all initialized to 0)

You can also use the function dim to declare vectors and matrices

PPNAs = dim (8, [1, 2, 3, 4, 5, 6, 7, 8]) #Pivot value of State intervals

4. Section "Restrictions"

CHIINM => = 0	The variable must always be positive
att => = 0; <= 1	The variable must always be between 0 and 1

5. Section "Initialization"

Initializes the values of the variables (temporary or not). It only runs once before starting the simulation.

INMI = 1	
AV (1) = 3	AV is either a parameter declared as a vector or a variable declared as a vector in the "Variable" section
AV (6:32) = 7	Initializes positions 6 to 32 with the value 7) (AV is either a parameter declared as a vector or a variable declared as a vector in the "Variable" section

6. Section "Model"

In this section is the simulation "Loop" with all its equations and commands.

7. Section "Finalization"

This section only runs once at the end of the simulation.

8. Differential equations

Differential equations begin with dt

dt a = a * b * c

9. Conditionals

General structure:

Conditional, ", if condition, else default value"

a = b + c, if b> c, else 3

Whether meets the condition the equation is evaluated, otherwise, the variable is assigned the value by default)

NOTE: If an inverse case condition (else) is applied to a differential equation, the else assignment is made on the variable.

Example of multiple conditions:

a = b + b, if c> b and f <> k and (o> = 98 or w = q + 3), else 0

10. Use of variables

variable [tx]

It refers to the value of the variable 'x' steps back in the simulation. 'x' can be a number, a variable or a parameter. The value must always be an integer

variable [0]

It refers to the initial value of the variable (time 0)

a(3)

`` refers to the third position of vector a

a(j + 4-i)

refers to the position j + 4-i of vector a, j + 4-i should result in an integer, otherwise an exception is raised and the simulation ends immediately

a(4)[t-2]

refers to position 4 of vector a, but 2 steps back in the simulation)

a(j-1)[0]

refers to the initial value (time 0) of position j-1 of vector a)

11. *Use of field data*

To access data stored as field data:
```
b = field_data ("Set.Variable")
```

If a project has a field data set called "Rains" and in this set there is a variable called "LaPlata" then to make use of the data from that variable:
```
LluviasEnLaPlata = field_data("Lluvias.LaPlata")
```

12. *IF condition*

```
IF condition {
        statements
}
else {
        statements
}
```

The use of the else is optional. If used it should go immediately on the line under the stopcock

Examples:

```
If a = 4 {                    # An if without else
        b = 4
}

If (a> = 2) and (a <= 5) {    # An if with else
        b = 4
}
else {
        b = 3
}
```

Nested if example
```
If (a> = 2) and (a <= 5) {    # This if has an else
        If b = 4 {            # This if does NOT have an else
                c = 3
        }
}
else {
```

```
                if b = 3 {                    # This if has an else
                        d = 3
                }
                else {
                        d = 2
                }
        }

if Z> 0 {
....
 a = 3
....
}

ELSE {
....
A = 8
....
}
```

13. Additional format of conditionals

```
    if "Condition" {
        a = 1
        b = 2
        ...
        c = 3
        }
   else  {
        a = 10
        b = 20
        ...
        c = 30
        }
```

Only if the "Condition" is TRUE

Only if the "Condition" is FALSE

14. The “For” loop

```
        For j = 1: 5 {                  (A loop from 1 to 5)
                A (j + 1) = A (j) / w (j)
        }
```

For j = 6: 3: -1 {
Equations
}
	(A loop from 6 to 3 with step -1)

For j = s: t: p {
Equations
}
	(A sat bow with step p; s, typ can be variables or parameters, as long as they are integers, otherwise an exception is raised and the simulation ends immediately)

15. *Comment, #*

#this is a comment
# a = b + c this equation does not run
a = b + c #this equation if executed

16. *Arithmetic operators*

+	(Sum)
-	(Subtraction)
*	(Multiplication)
/	(Division)
^	(Power)

17. *Logical operators*

>	
<	
> =	
<=	
=	
<>	(different)
and	
or	
not	

18. Functions

LN(a)	(Returns the natural logarithm of a)
LOG10(a)	(Returns the decimal logarithm of a)
EXP(a)	(Returns the exponential of a)
SIN(a)	(Returns the sine of a)
Cos(a)	(Returns the cosine of a)
SQR(a)	(Returns the square root of a)
ABS(a)	(Returns the absolute value of a)
Mod(a, b)	(Returns the remainder of division a / b)
ModT(b)	(Calculates the following expression: Mod (T-1, b) +1. Returns a number between 1 and b depending on the T (time) according to the following table:

T	Outcome
1	1
2	2
3	3
...	
b	b
b + 1	1
b + 2	2
...	
2 * b	b
2 * b + 1	1
2 * b + 2	2
...	
...	
n * b	b
n * b + 1	1
n * b + 2	2
...	

Ex:

J = ModT (4)

T	J
1	1
2	2
3	3
4	4
5	1
6	2
...	

NOTE: This function is useful to use as an index of vectors in a particular range as a function of simulation time.

Sumar(v)	(v is a vector, returns the sum of the elements of vector v)
Estbas(v)	(v is a vector, returns the sum of the elements of vector v)
Round(v)	(Returns the number v rounded)
Trunc(v)	(Returns the truncated number v)
Runi(a, b)	(Returns a uniform distribution random number between a and b)
Rnormal(a, b)	(Returns a random number with normal distribution with mean a and standcard deviation b)
Fun_Tasa_Intrinseca(m)	(m is a matrix, returns the r0 of the matrix)
Fun_Tasa_De_Reemplazo(m)	(m is a matrix, returns the R0 of the matrix)
Fun_Tasa_Finita(m)	(m is a matrix, returns the lambda of the matrix)
Fun_Tiempo_Genecional(m)	(m is a matrix, returns the T of the matrix)
LHP(m)	(m is an array, returns multiple population parameters): 'r0_intrinseca', 'Lambda', 'R0_replace', 'T', 'Mu1', 'Ro_damping', 'Life expectancy', 'Life_Expectation_Variance', 'Hope_of_life_of_stadium_i', 'Age_first_production', 'Var_Mu1', 'Average_duration_development_estadio_i', 'Var_Media_duration_development_estadio_i', 'Fundamental_array', 'Variance_lambda', 'SAD', 'Productive_value', 'Sensitivity', 'Elasticity'

19. MODEL

The following is an example of indirect ways of doing "conditional jumps" based on a series of individual conditionals:

a = w + q, if v> 0, else s * t
b = w + 2, if v> 0, else s * tr
c = w + q, if v> 0, else s * t + 3
d = w + q, if v> 0, else s * t-8
e = w + q, if v> 0, else s * t / 4
f = w + q, if v> 0, else s * t ^ 7