**Using Arrays**

**Background:** There are often times when we want to disaggregate a structure by groups because we are interested in understanding inequities in a system by social groups or level of need. One way to do this is to explicitly build out the structure through a series of stocks and flows as we do with aging chains, for example. However, as the number of dimensions increase, such an approach becomes cumbersome and error prone. A better way to handle these types of situations is to consider the use of arrays.

Arrays are ordered lists of values. For example, we might use an array to represent the population of several countries:

|  |  |
| --- | --- |
| Country | Population size in 2016 |
| China | 1,403,500,365 |
| India | 1,324,171,354 |
| United States | 322,179,605 |
| Brazil | 207,652,865 |
| Mexico | 127,540,423 |

The actual array or list would be (1403500365, 1324171354, 322179605, 207652865, 127540423) that correspond to China, India, United States, Brazil and Mexico respectively. This array has one dimension (country) with 5 elements that are indexed by their country name.

We also can arrays that have more than one dimension. For example, we might want to keep track of the population for each country by gender (which would have two dimensions, country and gender), and if we want to get even more specific, maybe also include age (which would have three dimensions; country, gender, and age).

To learn how to use arrays, see the online documentation for [Stella on arrays](https://www.iseesystems.com/resources/help/v1-6/default.htm#03-BuildingModels/Working_With_Arrays/Overview_Working_With_Arrays.htm). There are many powerful features in using arrays and specialized built-in functions for working with and manipulating the results from arrays, so it’s worth exploring some of the documentation and experimenting with different uses as you need.

The main benefits of using arrays include being able to make changes in one part of a structure (e.g., adding a flow) that carries through all the elements of the array without having to write equations for each element, simplifying the visual layout and presentation of a model while capturing a great amount of detail complexity, and modeling how different rules (e.g., associated with unequal power and privilege) can influence the overall dynamics of a system.

**Purpose:** Togain some exposure to the use of arrays.

**Instructions:** In the example that follows, we’re ultimately interested in modeling how two different sets of people (low needs and high needs) access and utilize services, and want to be able to talk about how people’s needs change with and without intervention. However, models using arrays are best built by starting with and getting an overall structure working that is “flat” or doesn’t use arrays. So we’ll start with a simple aging chain involving two stocks, *People with Needs* and *People in Services*. Once we are feeling reasonably confident in this structure, we will then move onto the second step and disaggregate the structure using arrays.

1. To begin, rebuild the model below using the equations listed to replicate the behavior pattern for people with needs and people in services (the completed version is Services 1.stmx).





﻿Accessing\_Services = People\_with\_Needs/ Avg\_time\_to\_access\_services {UNIFLOW}

UNITS: People/Months

DOCUMENT: People acessing services.

Average\_length\_of\_service = 6

UNITS: Months

DOCUMENT: Average length of time that someone receives services.

Avg\_time\_to\_access\_services = 3

UNITS: month

DOCUMENT: The average amount of time to access services.

Completing\_Services = People\_in\_Services/ Average\_length\_of\_service {UNIFLOW}

UNITS: People/Months

DOCUMENT: People completing/terminating services.

Onset\_of\_need = 10 {UNIFLOW}

UNITS: People/Months

DOCUMENT: Onset of needs.

People\_in\_Services(t) = People\_in\_Services(t - dt) + (Accessing\_Services - Completing\_Services) \* dt {NON-NEGATIVE}

INIT People\_in\_Services = 0

UNITS: People

DOCUMENT: People in services, which assumes that people do not dropout of services.

People\_with\_Needs(t) = People\_with\_Needs(t - dt) + (Onset\_of\_need - Accessing\_Services) \* dt {NON-NEGATIVE}

INIT People\_with\_Needs = 0

UNITS: People

DOCUMENT: People with needs that are not being met, which assumes in this model that people wait indefinitely for services to get needs met.

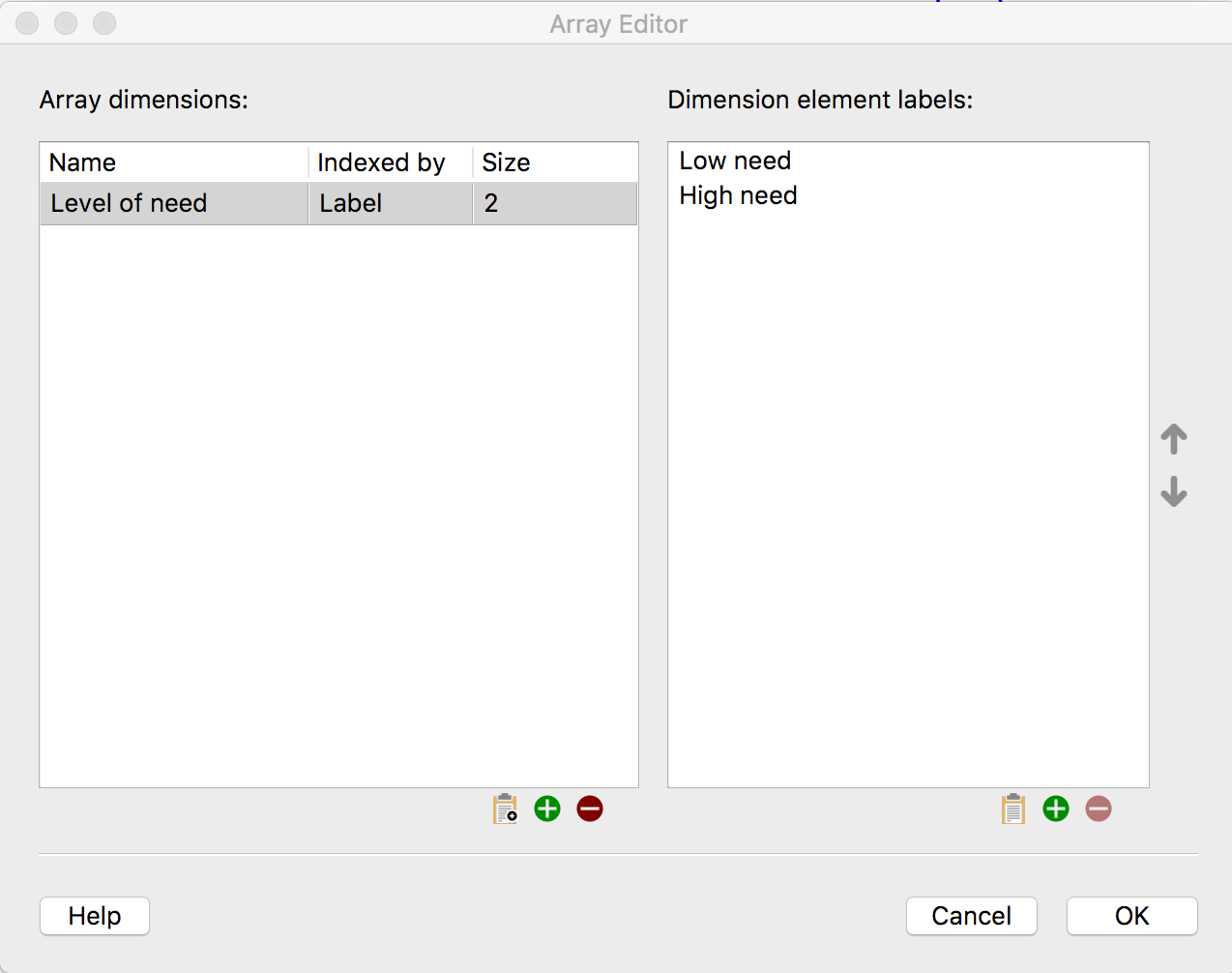
{ The model has 7 (7) variables (array expansion in parens).

In root model and 0 additional modules with 0 sectors.

Stocks: 2 (2) Flows: 3 (3) Converters: 2 (2)

Constants: 3 (3) Equations: 2 (2) Graphicals: 0 (0)

}

1. Having built, tested, debugged, and developed some basic confidence in the flat model without arrays, we are now ready to disaggregate the structure by low and high needs using arrays. To do this, you’ll first need to define the array dimension “level of need” by going to the array editor (accessible from the main menu, MODEL | ARRAY EDITOR. Create the “Level of need” dimension so you have the following:   
     
   
2. Now you can array the stocks using this dimension, and modify the equations to match the equations below. Note that as you do this, some equations use the same equation for all elements of the array (e.g., Completing Services) while others have unique equations for each element (e.g., Change in Level of Needs).   
     
   You’ll also notice that we’re using bi-flows with stocks to manage transitions between level of needs. We need to do this because as people’s needs change, they leave one category (e.g., low need) and enter another category (e.g., high need). When we do this, we need to remove the people transitioning out and add the people transitioning into a category. If you’ve entered everything correctly, you should have graphs for the stocks similar to what is shown below.





﻿Accessing\_Services[Level\_of\_need] = People\_with\_Needs/ Avg\_time\_to\_access\_services {UNIFLOW}

UNITS: People/Months

DOCUMENT: People accessing services, where the assumption is that people are accessing services within the same average time and independent of actual need.

Average\_length\_of\_service = 6

UNITS: Months

DOCUMENT: The average length of service.

Avg\_time\_to\_access\_services = 3

UNITS: month

DOCUMENT: Average time to access services.

Change\_in\_Level\_of\_Needs[Low\_need] = People\_with\_Needs[High\_need]\*Fraction\_rate\_of\_decreasing\_needs-People\_with\_Needs[Low\_need]\*Risk\_of\_increasing\_needs

UNITS: People/Months

DOCUMENT: This represents a change in needs by having people with needs at one level leave that level and enter another level, which is influenced by the risk of needs increasing from one category to another and the fractional rate of needs decrasing from one level to another.

Change\_in\_Level\_of\_Needs[High\_need] = People\_with\_Needs[Low\_need]\*Risk\_of\_increasing\_needs-People\_with\_Needs[High\_need]\*Fraction\_rate\_of\_decreasing\_needs

UNITS: People/Months

DOCUMENT: This represents a change in needs by having people with needs at one level leave that level and enter another level, which is influenced by the risk of needs increasing from one category to another and the fractional rate of needs decrasing from one level to another.

UNITS: People/Months

DOCUMENT: This represents a change in needs by having people with needs at one level leave that level and enter another level, which is influenced by the risk of needs increasing from one category to another and the fractional rate of needs decrasing from one level to another.

Completing\_Services[Level\_of\_need] = People\_in\_Services/ Average\_length\_of\_service {UNIFLOW}

UNITS: People/Months

DOCUMENT: People completing services.

Effect\_size = .25

UNITS: Dimensionless/Months

DOCUMENT: The effect size of the intervention, which is represented as a fractional rate of transitioning from higher needs to lower needs.

Fraction\_rate\_of\_decreasing\_needs = .025

UNITS: Dimensionless/month

DOCUMENT: Fractional rate that needs are decreasing naturally (i.e., without formal interventions) from one category to another category.

Onset\_of\_need[Level\_of\_need] = IF Level\_of\_need=1 THEN 10 ELSE 20 {UNIFLOW}

UNITS: People/Months

DOCUMENT: Onset of needs.

People\_in\_Services[Level\_of\_need](t) = People\_in\_Services[Level\_of\_need](t - dt) + (Accessing\_Services[Level\_of\_need] + Treatment[Level\_of\_need] - Completing\_Services[Level\_of\_need]) \* dt {NON-NEGATIVE}

INIT People\_in\_Services[Level\_of\_need] = 0

UNITS: People

DOCUMENT: People in services receiving treatment.

People\_with\_Needs[Level\_of\_need](t) = People\_with\_Needs[Level\_of\_need](t - dt) + (Onset\_of\_need[Level\_of\_need] + Change\_in\_Level\_of\_Needs[Level\_of\_need] - Accessing\_Services[Level\_of\_need]) \* dt {NON-NEGATIVE}

INIT People\_with\_Needs[Level\_of\_need] = 0

UNITS: People

DOCUMENT: People with needs by level of need.

Risk\_of\_increasing\_needs = .1

UNITS: Dimensionless/months

DOCUMENT: This is the risk that needs will increase without services.

Treatment[Low\_need] = People\_in\_Services[High\_need]\*Effect\_size

UNITS: People/Months

DOCUMENT: Treatment is assumed to be generally beneficial and lowering the level of need an individual might have.

Treatment[High\_need] = -People\_in\_Services[High\_need]\*Effect\_size

UNITS: People/Months

DOCUMENT: Treatment is assumed to be generally beneficial and lowering the level of need an individual might have.

UNITS: People/Months

DOCUMENT: Treatment is assumed to be generally beneficial and lowering the level of need an individual might have.

{ The model has 12 (19) variables (array expansion in parens).

In root model and 0 additional modules with 0 sectors.

Stocks: 2 (4) Flows: 5 (10) Converters: 5 (5)

Constants: 5 (5) Equations: 5 (10) Graphicals: 0 (0)

}