Factorio Model, Version 3

Jaiden King 2/7/2025

This script shows how to use the Factorio Model. The model will balance the whole system from mining drills to assembling machines.

The model inputs are: desired production rates, resource scope, process scope.

The model outputs are: computed production rates, suggested machine/process combinations, belt requirement info.

```
% Initialize the system
clear all
close all
clc
```

These external files populate the global variables used to run the model. They define all the resources, machines, and processes that exist in the game.

```
resources
machines
processes
```

The first thing the user does is define the production rates that they want to solve for. This is just a list of numbers, but the `new_rate_vector` function helps make the syntax more readable by letting you use item names. It's not uncommon to have multiple versions of the function call commented out as reference or for quick access.

```
% Set the desired output rate
b = new_rate_vector({"electronic_circuit", 15});
%b = new_rate_vector({"automation_science_pack", 10});
%b = new_rate_vector({"automation_science_pack", 10; "electronic_circuit", 100});
```

Sometimes there are multiple solutions to a system due to there being some processes that produce the same resource. Sometimes a resource may be produced or consumed that you don't care about balancing (you just want to treat it as a free variable). In these scenarios, you can reduce the scope of the problem with the following methods.

```
% Limit the resource scope
resource_scope = ["copper_cable" "electronic_circuit"]';
[resource_indices,~] = item_names_to_indices(resource_scope);
% Limit the process scope
process_scope = ["copper_cable_crafting" "circuit_crafting"]';
[process_indices,~] = process_names_to_indices(process_scope);

% Grab the subsets
A_ = A(resource_indices,process_indices);
b_ = b(resource_indices);
```

After doing all the work to set up the problem in the form of $A\mathbf{x} = \mathbf{b}$, solving it is a single operation.

```
% Solve the reduced system
x_ = A_\b_;
```

The result we have now is a solution to the reduced-scope system which has rows removed for the resources we chose to ignore. We now have to add back the rows to bring it back to the proper vector space.

```
x = zeros(size(A,2),1);
for i=1:length(process_indices)
    x(process_indices(i)) = x_(i);
end
```

We now compute the final production rate vector $A\mathbf{x}$. This will include the resources that were previously ignored during the solve. It's always good to do this to double check the output rather than assuming it resulted in the rates you asked for. We also compute the gross rates by supressing all the negative entries of the A matrix. These values would match what you see in the in-game produciton stats.

```
Ax=A*x;
Ax_gross = gross_rates(A,x);
```

So far $\mathbf{x}$ has been used to represent how many copies of each process were running, assuming it was all done with crafting speed 1. However, many processes can't run on a machine with crafting speed 1. The `assign_default_machines_to_unit_processes` function converts the $\mathbf{x}$ vector into a matrix which represents processes implemented on specific machines. This can then be used as a guide for what to build in-game.

```
x2 = assign_default_machines_to_unit_processes(x);
```

The final report summarizes our work.

```
display_results;
```

```
 --- Model Input ---

Target Production Rates:
electronic_circuit: 15

Resource Scope:
    "copper_cable"
    "electronic_circuit"


Process Scope:
    "copper_cable_crafting"
    "circuit_crafting"


 --- Model Results ---

Net Production Rates:
iron_plate: -15
```

```
copper_plate: -22.5
electronic_circuit: 15

Gross Production Rates:
copper_cable: 45
electronic_circuit: 15

Belt Requirements (Yellow Belt):
copper_cable: 3
electronic_circuit: 1
Note: Belt requirements depend on specific implementation details.

Unit Process Analysis:
Non-zero UNIT processes:
copper_cable_crafting    11.25
circuit_crafting    7.5

Construction Suggestion:
copper_cable_crafting    assembling_machine_3    9
circuit_crafting    assembling_machine_3    6
```

Misc Todo:

Function to make the machine matrix (`x2`) from string-string-int like the `new_rate_vector` function.