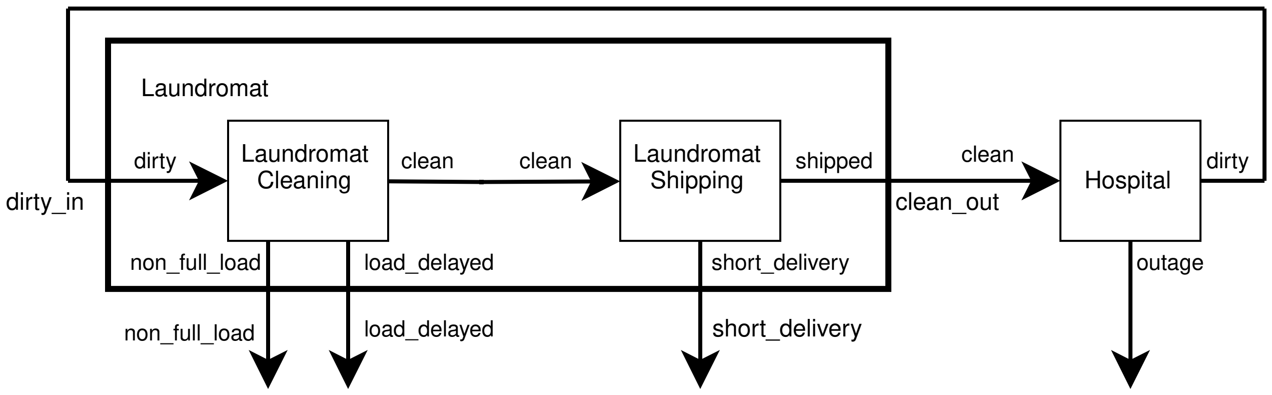
Modelling an Offsite Hospital Laundry Facility using PDEVS

Griffin Barrett
Student Number 100978435
Based on work by
Michael Lepard
Student Number: 100811047

Brief Description

This system models a laundromat working to supply a hospital with enough clean laundry to operate. This model will be able to be used to measure the average amount of unused/stockpiled clean laundry, the rate at which the hospital does not have enough laundry to meat it’s day to day needs, and the rate at which the laundromat runs loads below capacity or has to delay starting a load because it simply has nothing to wash.

For the purposes of this model, all dirty laundry takes an equal amount of effort to clean, and all delivery trucks run on scheduled. This model hospital will go through laundry at a normally distributed rate, and the laundromat will clean laundry in loads that are run only when there is laundry to wash but otherwise at a fixed scheduled.



There are 4 models in this system, three of which are atomic. Their explanations, formal definitions, and testing goals are below.

Laundry Cleaning

The laundry cleaning atomic model takes in dirty laundry, and washes loads of laundry of a specified size and at a specified rate. Cleaned laundry is then outputted.

Formal Specification:

```
Laundry_Cleaning((max_load, max_load Є N && max_load > 0),
                 (load_time, load_time Є TIME && load_time >= 0))
= {X,Y,S,δext,δint,δconf,λ,ta}

    X ={(dirty, dirty Є N && dirty >= 0)}
    Y ={(clean, clean Є N && clean >  0),
        (non_full_load, non_full_load Є N &&
                        0 < non_full_load < max_load),
        (load_delayed, load_delayed Є TIME && load_delayed > 0)}

    S ={amount_dirty Є N,
        load_size Є N,
        load_time_remaining Є TIME,
        load_delay_time Є TIME}


    δext (s, e, x) {

        for (dirty_pile in x.dirty){
            amount_dirty += dirty_pile;
        }
        if (load_size > 0){ /*We are running a load right now*/
            load_time_remaining = max(0, load_time_remaining-e);
        }else{ /*we are not running a load*/
            load_delayed_time += e;
            if (amount_dirty > 0){ /*we are starting a load*/
                load_time_remaining = load_time;
                load_size = min(amount_dirty, max_load);
                amount_dirty -= load_size;
            }
        }
    }

    δint (s) {
        if (amount_dirty > 0){ /*start a new load*/
            load_time_remaining = load_time;
            load_size = min(amount_dirty, max_load);
            amount_dirty -= load_size;
        }else{ /*wait for more laundry to arive*/
            load_size = 0;
            load_time_remaining = 0;
        }
        load_delay_time = 0;
    }

    δconf(s, e, x){
        δint (s);
        δext (s, e, x);
    }
```

```
λ(s) {
    _output_map = {clean:load_size};
    if(load_delay_time > 0){ /* This load was delayed */
        _output_map += {load_delayed: load_delayed_time};
    }
    if(load_size < max_load){ /* This load was undersized */
        _output_map += {non_full_load: max_load-load_size};
    }
    return _output_map;
}

ta(s){
    if( load_size > 0){
        return max(0, load_time_remaining);
    }else{
        return inf;
    }
}
```

Testing

The laundry cleaning model will be tested in the following ways:

Initialize with *amount_dirty* = 0.
Expect that it will not output anything.

Initialize with a known *max_load*, *load_time*, and *amount_dirty*.
Expect that the model output it's last load at $t = load\_time \times \left\lceil \frac{amount\_dirty}{load\_max} \right\rceil$ and that it will output $load\_max - (amount\_dirty\,(\mathbf{mod}\; load\_max))$ on port *non_full_load*.

Initialize with a known *max_load*, *load_time*, and *amount_dirty* = 0.
At some $0 \leq t_{offset} < load\_time$, send in a known non 0 amount of dirty laundry on port *dirty*.
Expect that at $t = load\_time + t_{offset}$ the model will output a load of clean laundry on port *clean* and t_{offset} on port *load_delayed*.
Expect that the model will output it's last load at $t = load\_time \times \left\lceil \frac{amount\_dirty}{load\_max} \right\rceil + t_{offset}$

Initialize with a known *amount_dirty.*
Over the lifetime of this test, input a known sequence of piles of dirty laundry
Expect that when both the sequence and this model come to rest,
$amount\_dirty + \sum dirty\_sequence = \sum outputs$

Laundry Shipping

The laundry shipping atomic model takes in loads of clean laundry and at a specified interval outputs a predetermined amount of clean laundry on port *shipped*. If an interval comes up and it does not have the predetermined amount of laundry, it will output as much as it can and will also output the amount of laundry it was short by on port *short*.

Formal Specification:

```
Laundry_Shipping(
    (shipping_amount, shipping_amount Є N && shipping_amount > 0),
    (shipping_interval,
        shipping_interval Є TIME &&
        shipping_interval > 0
    )
= {X,Y,S,δext,δint,δconf,λ,ta}

    X = { (clean, clean Є N && clean >= 0) };
    Y = { (shipped, shipped Є N && 0 <= shipped <= shipping_amount),
          (short_delivery, short_delivery Є N &&
                           0 < short_delivery <= shipping_amount) };
    S ={amount_clean Є N,
        time_until_next_shipment Є TIME};

    δext (s, e, x) {
        time_until_next_shipment
            = max(0, time_until_next_shipment-e);

        for(clean_pile in x.clean) {
            amount_clean += clean_pile;
        }
    }

    λ(s) {
        if( amount_clean < shipping_amount )
            return {shipped: amount_clean,
                    short_delivery:(shipping_amount - amount_clean)};
        else
            return {shipped: shipping_amount};
    }

    δint (s) {
        time_until_next_shipment = shipping_interval;
        amount_clean = min(0, amount_clean-shipping_amount);
    }

    δconf(s, e, x){
        δint (s);
        δext (s, e, x);
    }

    ta (s) { return time_until_next_shipment; }
```

Testing

The laundry shipping model will be tested in the following ways:

Initialize with a known *shipping_amount*, *shipping_interval*, and an *amount_clean* of 0.
Expect that at $t = shipping\_interval$ it will output 0 on port *shipped* and *shipping_amount* on port *short_delivery*.

Initialize with a known *shipping_amount*, *shipping_interval*, and an *amount_clean* such that $0 < amount\_clean < shipping\_amount$
Expect that at $t = shipping\_interval$ it will output *amount_clean* on port *shipped* and $shipping\_amount - amount\_clean$ on port *short_delivery*.

Initialize with a known *shipping_amount*, *shipping_interval*, and an *amount_clean* such that $shipping\_amount \leq amount\_clean$
Expect that $t = shipping\_interval$ it will output *shipping_amount* on port *shipped* and nothing on port *short_delivery*.

Initialize with a known *amount_clean*.
Over the lifetime of this test, input a known sequence of piles of clean laundry
Expect that when both the sequence comes to rest and then the model produces output on *short_delivery*, $amount\_clean + \sum clean\_sequence = \sum shipped$

Hospital

The hospital atomic model takes in loads of clean laundry.
At a specified interval it outputs all of it's dirty laundry and then converts as much of a positive normally distributed amount of clean laundry into dirty laundry for the next shipment as it can.

Formal Specification:

```
Hospital(
    (shipping_interval,
        shipping_interval Є TIME &&
        shipping_interval > 0
    ),
    (usage_mean, usage_mean Є N),
    (usage_sd, usage_sd Є N && usage_sd >= 0))
= {X,Y,S,δext,δint,δconf,λ,ta}

    X = { (clean, clean Є N && clean >= 0) };
    Y = { (dirty, dirty Є N && dirty >= 0),
          (outage, outage Є N && outage > 0)};

    S ={amount_clean Є N,
        amount_dirty Є N,
        outage Є N,
        time_until_next_shipment Є TIME};

    δext (s, e, x) {
        time_until_next_shipment =
            max(0, time_until_next_shipment-e);
        for(clean_pile in x.clean) {
            amount_clean += clean_pile;
        }
    }

    λ(s) {
        _output_map = {dirty: amount_dirty};

        if(outage > 0){
            _output_map += {outage: outage};
        }
        return _output_map;
    }

    δint (s) {
        time_until_next_shipment = shipping_interval;
        _amount_to_dirty
            = min(0, random_normal(usage_mean , usage_sd));
        amount_dirty = min(_amount_to_dirty, amount_clean);
        outage = amount_dirty - _amount_to_dirty;
        amount_clean -= amount_dirty;
    }

    δconf(s, e, x){
        δint (s);
        δext (s, e, x);
    }
    ta(s) {
        return time_until_next_shipment;
    }
```

Testing

The hospital will be tested in the following ways.

Initialize with a known *shipping_interval*, *usage_mean*, *usage_sd*, and *amount_clean* = 0.
Expect that once every *shipping_interval* it will output an amount of *outage* that is never below *0* and is otherwise normally distributed with a mean of *usage_mean* and a standard deviation of *usage_sd*.

Initialize with a known *shipping_interval*, *usage_mean*, *usage_sd*, and an arbitrarily large *amount_clean*.
Expect that once every *shipping_interval* it will output an amount of dirty laundry that is never below 0 and is otherwise normally distributed with a mean of *usage_mean* and a standard deviation of *usage_sd*.

Initialize with a known *shipping_interval*, *usage_mean*, *usage_sd*, and *amount_clean*.
Expect that at $t = shipping\_interval$ it will output an amount of dirty laundry not greater than *amount_clean*.

Initialize with any set of known finite values, such that $usage\_sd > 0$.
Over the lifetime of the test, input any known finite sequence of finite quantities of clean laundry.
Expect that eventually $amount\_clean + \sum clean_{in} = \sum output$ and that the internal state value for the current *amount_clean* will reach 0.

Laundromat

The laundromat coupled model takes in dirty laundry to be washed, and at a specified interval attempts to output a specified amount of clean laundry. If it fails to meet that quota, it will also output how much it missed by.

Formal Specification:

```
Laundromat(
     (max_load, max_load Є N && max_load > 0),
     (load_time, load_time Є TIME && load_time >= 0),
     (shipping_amount, shipping_amount Є N && shipping_amount > 0),
     (shipping_interval, shipping_interval Є TIME &&
                        shipping_interval > 0))
= {X, Y, D, IC, EIC, EOC}

     X = { (dirty_in, dirty_in Є N && dirty_in >= 0) }

     Y = { (clean_out, clean_out Є N && clean_out >= 0),
           (non_full_load, non_full_load Є N &&
                           0< non_full_load < max_load),
           (load_delayed, load_delayed Є TIME && load_delayed > 0),
           (short_delivery, short_delivery Є N &&
            0 < short_delivery <= shipping_amount) }

     D ={cleaning : Laundry_Cleaning(max_load, load_time),
         shipping :
             Laundry_Shipping(shipping_amount, shipping_interval)};

     IC  = { ((cleaning, clean), (shipping, clean)) };
     EIC = { (dirty_in, (cleaning, dirty) };
     EOC = {
             ((cleaning, non_full_load), non_full_load),
             ((cleaning, load_delayed), load_delayed),
             ((shipping, short_delivery), short_delivery),
             ((shipping, shipped), clean_out)
           };
```

Testing

The laundromat will be tested in the following ways

Initialize with known values.
Do not input any dirty laundry.
Expect that it will not output anything on *clean_out*, and that at the interval specified by *shipping_interval* it will output *shipping_amount* on *short_delivery*.

Initialize with any valid inputs.
Over the lifetime of the test, input any positive known finite sequence of piles of dirty laundry
After the sequence has come to rest, eventually
Laundromat:Cleaning:amount_dirty == 0
Laundromat:Cleaning:load_size == 0
Laundromat:Shipping:amount_clean == 0
and that $\sum dirty_{\mathrm{in}} = \sum clean\_out$

Additionally, expect that any outputs on any of
Laundromat:Cleaning:non_full_load
Laundromat:Cleaning:load_delayed
Laundromat:Shipping:short_delivery
will be forwarded out on the equivalently named ports.

Simulation

For this analysis, 60k simulations were collected, with values in the following ranges. Each were simulated for 1 week.

Model	Constant or State element	Value or range of values
Laundry_Cleaning	max_load	[80%, 200%] of $\text{Laundry\_Shipping}_{\text{shipping\_amount}} \times \frac{\text{Laundry\_Cleaning}_{\text{load\_time}}}{\text{Laundry\_Shipping}_{\text{shipping\_interval}}}$
Laundry_Cleaning	load_time	1m
Laundry_Shipping	shipping_amount	[80%, 200%] of $\text{Hospital}_{\text{usage\_mean}} \times \frac{\text{Laundry\_Shipping}_{\text{shipping\_interval}}}{\text{Hospital}_{\text{shipping\_interval}}}$
Laundry_Shipping	shipping_interval	1h
Hospital	shipping_interval	1.5h
Hospital	amount_clean	5000
Hospital	usage_mean	[500, 900]
Hospital	usage_sd	[0, 1000]

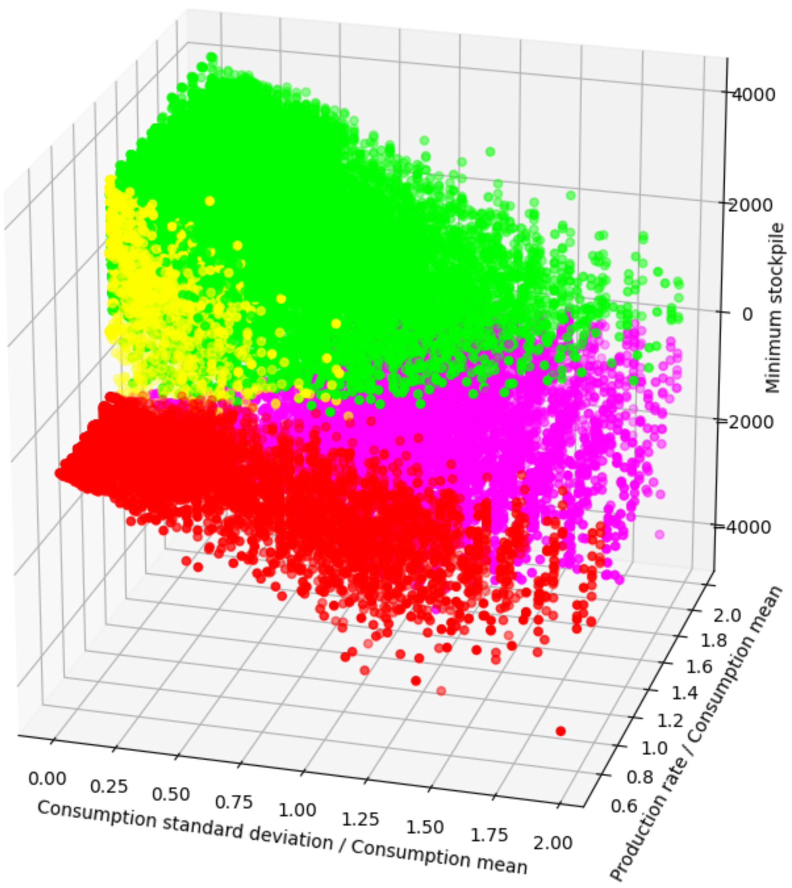
In the plot below, each point represents 1 run, with x, y, z, and colour determined in the following ways.

X = $$\frac{\text{Hospital}_{\text{usage\_sd}}}{\text{Hospital}_{\text{usage\_mean}}}$$

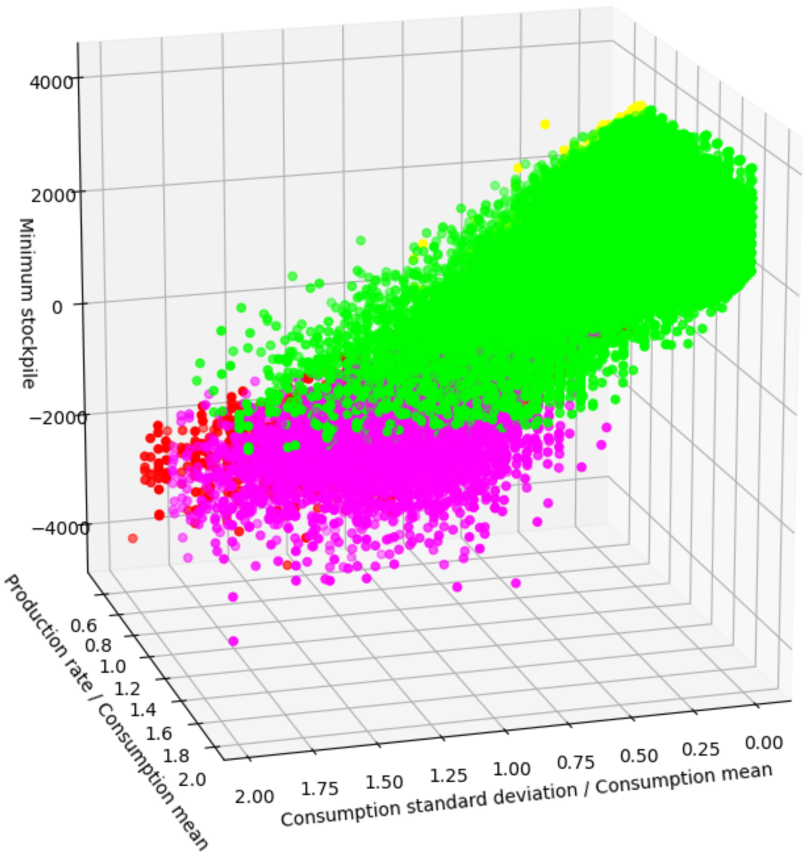
Y = $$\left. min\left(\frac{\text{Laundry\_Cleaning}_{\text{max\_load}}}{\text{Laundry\_Cleaning}_{\text{load\_time}}}, \frac{\text{Laundry\_Shipping}_{\text{shipping\_amount}}}{\text{Laundry\_Shipping}_{\text{shipping\_interval}}}\right) \middle/ \text{Hospital}_{\text{usage\_mean}} \right.$$

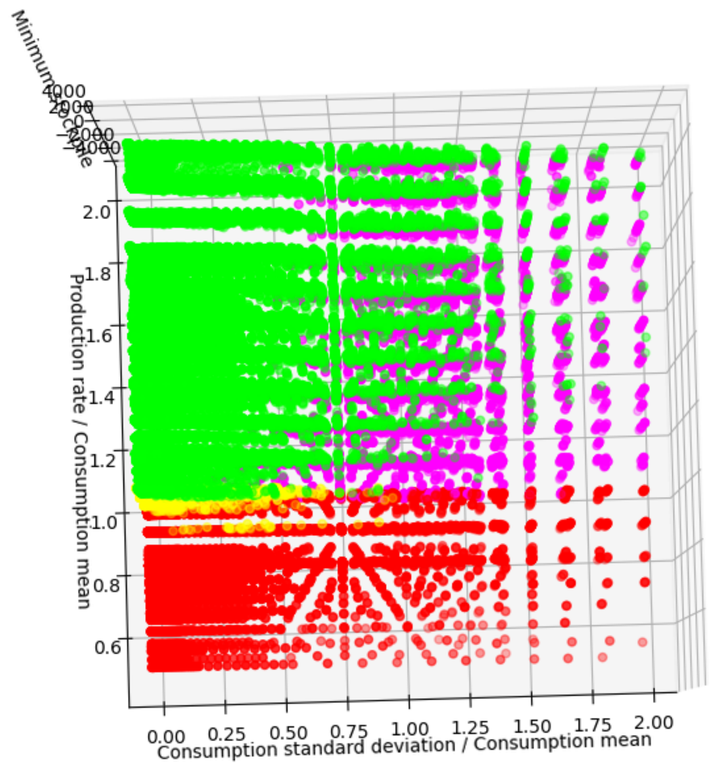
Z = $$\begin{cases} -max\left(Hospital_{outage}\right), & if\ max\left(Hospital_{outage}\right) > 0 \\ min\left(Hospital_{\text{amount\_clean}}\right), & if\ max\left(Hospital_{outage}\right) = 0 \end{cases}$$

Colour = $$\begin{cases} \text{Red}, & if\left(Y < 1 \&\& Z < 0\right) \\ \text{Yellow}, & if\left(Y < 1 \&\& Z \geq 0\right) \\ \text{Purple}, & if\left(Y \geq 1 \&\& Z < 0\right) \\ \text{Green}, & if\left(Y \geq 1 \&\& Z \geq 0\right) \end{cases}$$



4000
2000
0
−2000
−4000
Minimum stockpile
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Consumption standard deviation / Consumption mean
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
Production rate / Consumption mean


Minimum stockpile
4000
2000
0
−2000
−4000
Production rate / Consumption mean
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
Consumption standard deviation / Consumption mean
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00



Observations

There is a drop-off around y=1. This point represents the crossover point of the laundromat simply not being able to keep up with the hospital’s mean usage. This drop-off is sharp, despite the assumption for the value of Y that the rate of cleaning is bottlenecked by ether the cleaning or the shipping atomic models. The simulation models some of the interactions between these two atomics, but such interactions did not seem to play a large role in the effective cleaning rate of the laundromat. Perhaps in future models the laundromat could be simplified further.

There are very few yellow points, and they are all very close to y = 1. Each of these points represents a run where the laundromat cannot keep up with the average consumption of laundry by the hospital, but the hospital did not run out of laundry within the first week. I would expect that if the simulation time was extended beyond 1 week, we would see fewer and fewer yellow points. Eventually there would be none at all.

As standard deviation goes up, the hospital's initial stockpile of laundry is taxed more and more. Even with a stockpile of 5000, and a cleaning rate of 2x the mean rate of consumption, the hospital can still have occasional outages. This is to be expected with any unbounded usage distribution. On any shipment cycle, there is a non-zero chance that the hospital goes through it's entire stockpile in one go. It is expected that if the duration of the simulation were extended, all of the green points with a standard deviation >0 would eventually be replaced by purple ones. This would not be true if the usage distribution were bounded somehow. Perhaps there is a limit to the number of patients that the hospital can handle, and a limit to the worst case of laundry consumption by one patient within a given shipment cycle. That level of granularity was outside the scope of this model, and may be the subject of further review.

Results

This model does not seem to pass validation. In the real world, hospitals that are stocked with many cleaning cycles worth of laundry do not often have outages, but this model seems to suggest that such outages should be quite common. This speaks to poor modelling of the rate at witch laundry is consumed. This should be the focus of future work in this area.