E 150: Project 3

Fayyad Azhari

October 23, 2019

1 Introduction

The aim of this project is to simulate the operation of a free-form robotic 3D printer. This project is divided into 2 parts. The first part of the project involves the full model dynamics of the droplets with several point charges planted on the surface of the printer bed. The second part of the project involves the simplified free fall model with genetic algorithm. The main goal is to find the best control parameters that will generate the least error of printed pattern.

2 Background and Theory

For part 1, The free-form robotic 3D printer is modeled as a three-rod linkage where the first end of the rod is fixed. Meanwhile, the free end of the third rod has a dispenser that deposit the droplets. The coordinates in this project is such that $x - z$ is planar and the y axis is vertical. Some of the given vectors in the project are the initial angular positions, $\mathbf{\Theta_j^0}$, the constant angular velocities, $\dot{\mathbf{\Theta}}_\mathbf{j}$, the rod length,$L_j$ where $j = 1, 2, 3$. The position of the fixed end is also given to be $\mathbf{r_0}$. The dispenser position vector, r^d is given as below:

$$\mathbf{r^d} = \mathbf{r_0} + \mathbf{r^{d*}}$$

Where $\mathbf{r^{d*}} = (x_d, y_d, z_d)$

$$x_d = L_1 \cos \Theta_1 + L_2 \cos \Theta_2 + L_3 \sin \Theta_3$$

$$y_d = L_1 \sin\Theta_1 + L_2 \sin\Theta_2$$

$$z_d = L_3 \cos\Theta_3$$

We also note that $\Theta_j = \Theta_j(t)$ is a function of time where

$$\Theta_j(t) = \Theta_j^0 + \dot{\Theta}_j t$$

Next, we can find the dispenser velocity $\mathbf{v^d} = (\dot{\mathbf{x_d}}, \dot{\mathbf{y_d}}, \dot{\mathbf{z_d}})$ which is the time derivative of the dispenser position:

$$\dot{x_d} = -L_1\dot{\Theta}_1 \sin\Theta_1 - L_2\dot{\Theta}_2 \sin\Theta_2 + L_3\dot{\Theta}_3 \cos\Theta_3$$

$$\dot{y_d} = L_1\dot{\Theta}_1 \cos\Theta_1 + L_2\dot{\Theta}_2 \cos\Theta_2$$

$$\dot{z_d} = -L_3\dot{\Theta}_3 \sin\Theta_3$$

Now, using the dispenser position and velocity vector we could figure out the droplets dynamics. Each droplets is assume to start at the dispenser initially and the initial droplet velocity can be predicted by the printer arm velocity. We denote the droplets as i for $i = 1, 2, ..., N_d)$ where N_d is the number of droplets which is given to be 3000. Therefore, the initial droplet position is:

$$\mathbf{r_i^0} = \mathbf{r^d}$$

The initial droplet velocity is:

$$\mathbf{v_i^0} = \mathbf{v^d} + \mathbf{\Delta v^d}$$

where $\mathbf{\Delta v^d}$ is the relative droplet dispenser velocity. After the droplet is dispense, we can treat each of them as a lumped mass. The Newton's 2nd law can be applied:

$$m_i\ddot{\mathbf{r}}_\mathbf{i} = \mathbf{\Psi_i^{tot}} = \mathbf{F_i^{grav}} + \mathbf{F_i^{elec}} + \mathbf{F_i^{drag}}$$

where $\ddot{\mathbf{r}}_\mathbf{i}$ is the droplet acceleration. m_i is the mass of droplets calculated using the effective material properties of the density:

$$m_i = V_i\rho^*$$

$$\rho^* = (1 - v_2)\rho_1 + v_2\rho_2$$

where the density depends on the volume fraction of the different material. Using the forward Euler, we can update the formula position and velocity:

$$\mathbf{r_i}(t+\Delta t) = \mathbf{r_i}(t) + \mathbf{\Delta} t \mathbf{v_i}(t)$$

$$\mathbf{v_i}(t+\Delta t) = \mathbf{v_i}(t) + \mathbf{\Delta} t \frac{\mathbf{\Psi_i^{tot}}}{m_i}$$

Meanwhile the forces are given such as below:

$$\mathbf{F_i^{grav}} = (0, -m_i g, 0)$$

where g is gravity,

$$\mathbf{F_i^{elec}} = \sum_{p=1}^{N_c} \frac{q_p q_i}{4\pi\epsilon \|\mathbf{r_i} - \mathbf{r_p}\|^2} (\mathbf{r_i} - \mathbf{r_p})$$

where N_c is the number of point charge on the print bed. ϵ is the electrical permittivity. r_p is the position of the charge on the print bed. q_p is the point bed charge and q_i is the droplet charge calculated using the effective material properties of the charge:

$$q_i = V_i q^*$$

$$q^* = (1 - \upsilon_2) q_1 + \upsilon_2 q_2$$

Lastly, the drag force is:

$$\mathbf{F_i^{drag}} = \frac{1}{2} \rho_a C_{Di} \|\mathbf{v^f} - \mathbf{v_i}\| (\mathbf{v^f} - \mathbf{v_i}) A_i^D$$

where ρ_a is the surrounding medium density, v^f is the surrounding medium velocity, A_i^D is the drag reference area and C_{Di} is the drag coefficient.

$$A_i^D = \pi R^2$$

and the $C_{Di}(Re)$ is a function of Reynolds number find by:

$$Re = \frac{2R\rho_a \|\mathbf{v^f} - \mathbf{v_i}\|}{\mu_f}$$

where ν_f is the surrounding medium viscosity. C_{Di} is the piecewise function below:

$$C_{Di} = \begin{cases} \frac{24}{Re} & 0 < Re <= 1 \\ \frac{24}{Re^0.0646} & 1 < Re <= 400 \\ \frac{24}{Re^0.0646} & 1 < Re <= 400 \\ 0.5 & 400 < Re <= 3x10^5 \\ 0.000366Re^{0.4275} & 3x10^5 < Re <= 2x10^6 \\ 0.18 & Re > 2x10^6 \end{cases}$$

For part 2, we will neglect drag and electrical forces so the droplets will be in free fall with constant acceleration from gravity. The setup for the kinematics of the 3D printer and the droplets will be exactly the same as part 1.

Since we are using Genetic Algorithm, a random design parameters which act as the control parameters is created within the given ranges. The design parameters consists of the robot arm angular velocities and the relative dispenser velocity. The design strings formula is shown below:

$$\mathbf{\Lambda^i} = \dot{\Theta}_1, \dot{\Theta}_2, \dot{\Theta}_3, \mathbf{\Delta v^d}$$

The cost function used to generate the printed pattern with minimum error and as close as possible to the desired pattern is as below:

$$\Pi = \frac{\sum_{i=1}^{N_d} \left\| \mathbf{r_i^{des}} - \mathbf{r_i^{gen}} \right\|}{\sum_{i=1}^{N_d - 1} \left\| \mathbf{r_i^{des}} - \mathbf{r_{i+1}^{des}} \right\|}$$

3 Procedure and Methods

For part 1, the pseudocode starts with defining all the given constant. Then start calculating V_i and using the effective materials properties of the droplets ρ^*, q^*, m_i, q_i can be calculated. Next, the array of time and the electrical grid points is created and figure below shows the code:

Listing 1: Time array and electrical grid points

```
    % Array of time
    dt = delta_t:delta_t:T; % 1x3000

    % Grid points
    grid1 = linspace(-Lbed/2,Lbed/2,10);
    [gridx,gridy] = meshgrid(grid1);
    for r = 1:100
    rp(1:3,r) =  [gridx(r);0;gridy(r)];
    end
```

The gravitational force $\mathbf{F_i^{grav}}$ can also be calculated since it will be constant only be calculated once.

Next, 3D printer arm kinematics can be solved by calculating Θ_j and $\mathbf{r^{d*}}$. Then, find the velocities $\mathbf{v^d}$ which is the derivative of the displacement. The dispenser displacement $\mathbf{r^d}$ should be calculated using $\mathbf{r^0}$ and $\mathbf{r^{d*}}$.

Now, the droplet dynamics can be solved starting by defining $\mathbf{r_i^0}$ and $\mathbf{v_i^0}$. Next, initialize a droplet check flag array which tells which droplets have reach the print bed to discontinue calculating it. The code to create and how to use the flag array is attached below:

Listing 2: create and use the flag array

```
    flag = true(1,length(ri0));      %flag array
    t = 0;                           %initialize time
    i = 1;
    %terminate loop when time exceed flag array is all
        mapped
    while t <= T && any(flag)
    t = t + delta_t;
    %how to use the flag array on the velocity of
        droplets
    normv = vecnorm(vf-vi0(1:3,flag));
```

Then, the time is initialize and begin the droplet time loop. Inside the loop, calculate the Reynolds number Re, the drag coefficient C_{Di} and the drag force $\mathbf{F_i^{drag}}$. After that, another loop is created to calculated the $\mathbf{F_i^{elec}}$ and the code is shown below:

Listing 3: loop to find Felec

```
    % create dummy array variable
    rib = ri0(1:3,flag);
    % finding the length of the electric loop
    flag_length = find(flag);
    %initialize Felec
    Felec = [];
    for b = 1:length(flag_length)
        rif = rib(1:3,b);
        Felec_i = ((q_p*qi)*(rif-rp))./(4*pi*E*((
           vecnorm(rif-rp)).^2));
        sumFelec = sum(Felec_i,2);
        % Append the Felec vectors into an array
        Felec(1:3,b) = sumFelec ;
    end
```

After calculating all the forces, the total forces can be calculated and using forward Euler update $\mathbf{r_i}$ and $\mathbf{v_i}$ but for only the droplets not on the print bed. Next, check whether any of the droplets if its below the print bed and the code is shown below:

Listing 4: Check the position of droplets in y axis

```
    %Check droplet if below printbed
    e2 = ri0(2,:);                  %only the y axis
    [~,col] = find(e2<0);           %finding negative y
       value
    e2(col) = 0;
    flag(col) = 0;                  %update flag array
    ri0(2,:) = e2;                  %setting y value equal
       to zero
```

Then, store the current time for the droplets that hit the bed. End the time loop and generate requested plots.

For part 2, the code is written in 2 separate files which consists of the main code and the matlab function. The pseudocode started with defining the given constants but without any electrical and drag forces information. Then, the parameters that is used in the GA is defined using $S = 100$ genetic strings per generations. Saving the top $P = 10$ parents to produce $P = 10$

children for every generations. Therefore, the number of new strings for every generations is $S - P - P = 80$. The tolerance of the cost function is set to be $\text{Tol} = 2 * 10^{-2}$ and the limit number of generation is $b < 100$.

The initial string variable is created using random function separately such as the code shown:

Listing 5: generate random strings

```
    l1= rand(S,1).* (Sw1(2) - Sw1(1)) + Sw1(1);
    l2= rand(S,1).* (Sw2(2) - Sw2(1)) + Sw2(1);
    l3= rand(S,1).* (Sw3(2) - Sw3(1)) + Sw3(1);
    l4= rand(S,1).* (SDeltav_d(2) - SDeltav_d(1)) +
        SDeltav_d(1);

    Lambda = [l1,l2,l3,l4]; %append all the string
        variable
```

Then, generation loop and the string loop which is in another matlab function is created. The same 3D printed arm equation as part 1 is applied but using the design strings created. Calculating the initial droplet displacement, $\mathbf{r_i^0}$ and velocity, $\mathbf{v_i^0}$ are also the same as part 1.

Next, the final position of the droplets in free fall on the print bed is calculated analytically. The derivation uses the constant acceleration kinematic equation:

$$\mathbf{r_i} = \mathbf{r_i^0} + \mathbf{v_i^0} t + \frac{1}{2}\mathbf{\ddot{r}_i} t^2$$

where $\ddot{r}_i$ is the gravitational acceleration, g. The final droplets position occur when $\mathbf{r_i}.e_2 = 0$, isolating the e_2 component, the time for the droplet to hit the print bed can be calculated by solving the quadratic equation:

$$t^* = \frac{-v_{i2}^0 \pm \sqrt{-v_{i2}^2 + 2r_{i2}^0 g}}{-g}$$

Eventually, the plus sign is ignored since it is not always positive. The time calculated is then inserted into e_1, e_2 components of the final position of the droplets:

$$r_{i1} = r_{i1}^0 + v_{i1}^0 t$$

$$r_{i3} = r_{i3}^0 + v_{i3}^0 t$$

Next, the cost function is calculated. The code to calculate the final position and to calculate the cost function is as below:

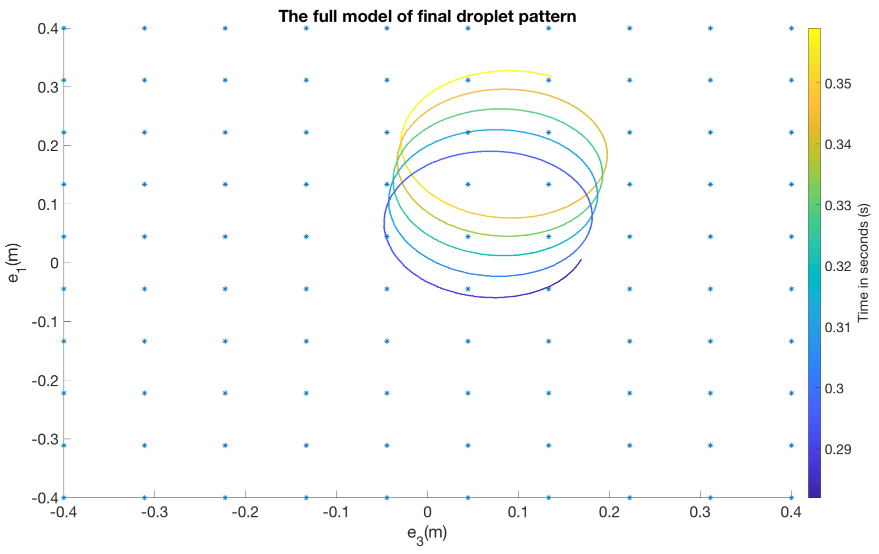
Listing 6: Generate cost function

```
t_star2 = (-vi0(2,:)-sqrt(vi0(2,:).^2+2.*ri0(2,:).*g))./(-g);
r1 = ri0(1,:) + vi0(1,:).*t_star2;  %e1 components of droplet
r3 = ri0(3,:) + vi0(3,:).*t_star2;  %e3 components of droplet
r2 = zeros(1,length(dt));           %e2 components of droplet
rfinal = [r1;r2;r3];     %Combining the position vector in array
% Calculating the cost function
pi1 = sum(vecnorm(ri-rfinal))./sum(vecnorm(diff(ri,1,2)));
```

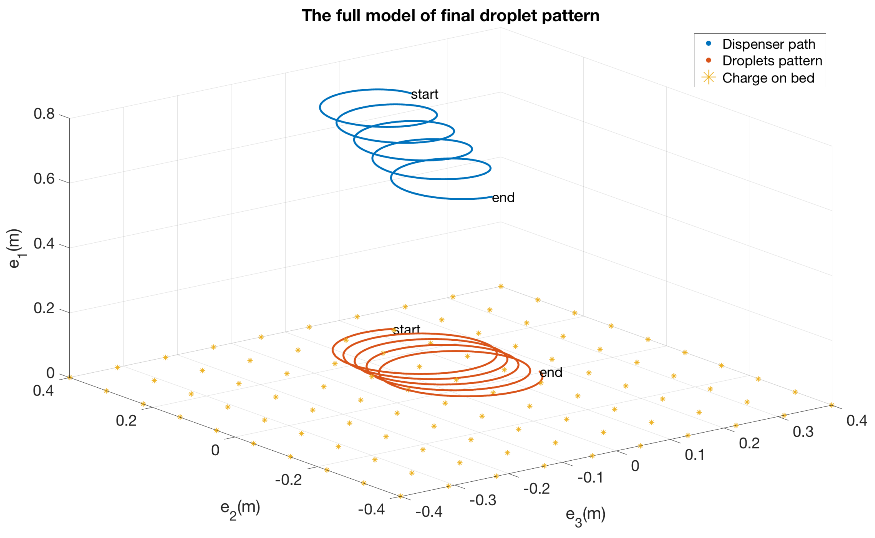
Then, the Genetic Algorithm continued by mating the parents to generate children. The cost is sorted and the generation loop is ended.

4 Results and Discussion

For part 1, the plot of the final droplet pattern in a $(z-x)$ axis with color bar showing the time is shown below:

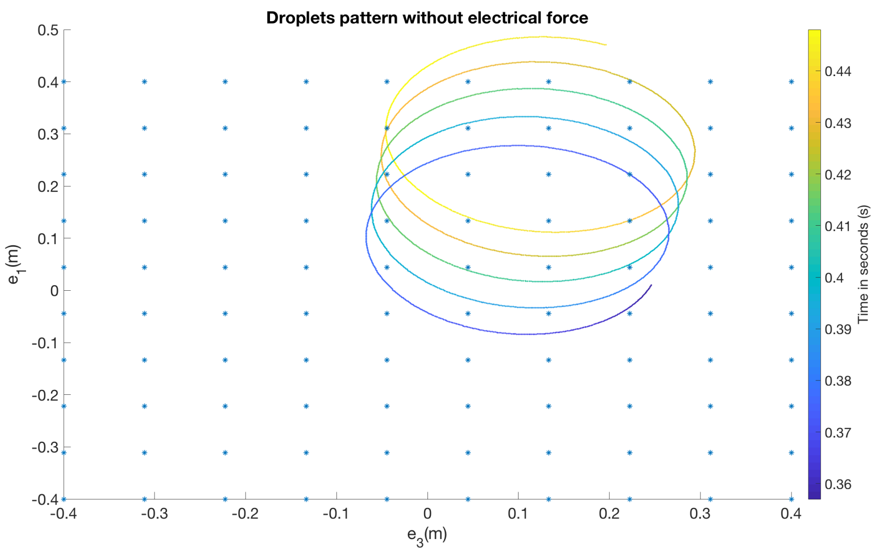


The tool path of the robot arm's dispenser end and the print pattern on the print bed is plotted in 3D:

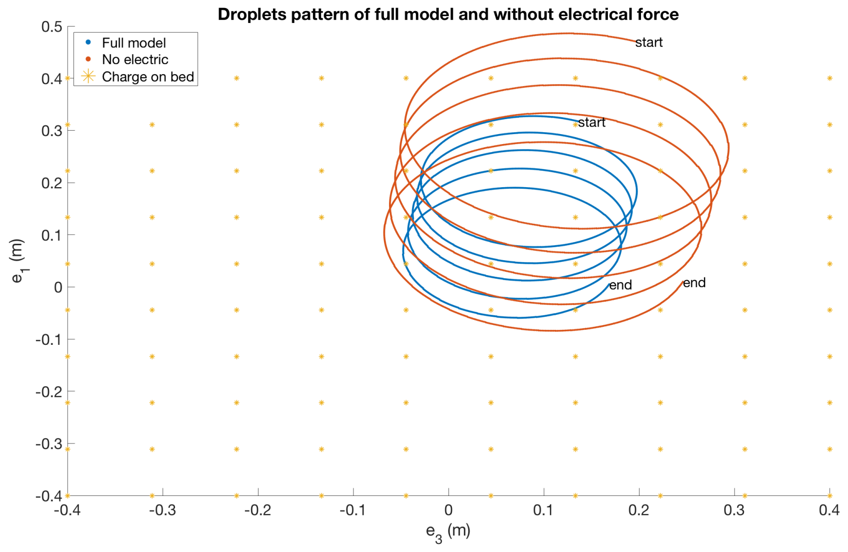


The resulting droplet pattern of the full model from the planner and the 3D seems reasonable. The time taken for the droplets to reach the print bed is really small. The shape of the final droplets configurations is similar to the shape of the dispenser path. But, the size of the printed pattern expanded compared to the dispenser path. This can actually reassemble the error in the full model. The error is caused by a lot of factors but mainly drag forces. The drag force slows down the droplets and deviate the droplets from reaching the target. The concentration of the point charge on the print bed also might effect the the droplets pattern.

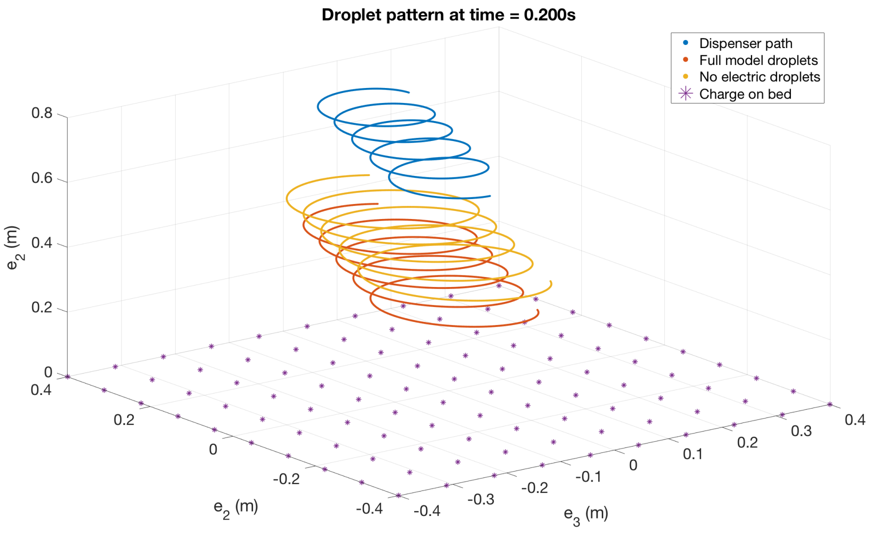
The final configurations of the droplets without any electrical forces with color bar showing the time is shown below:

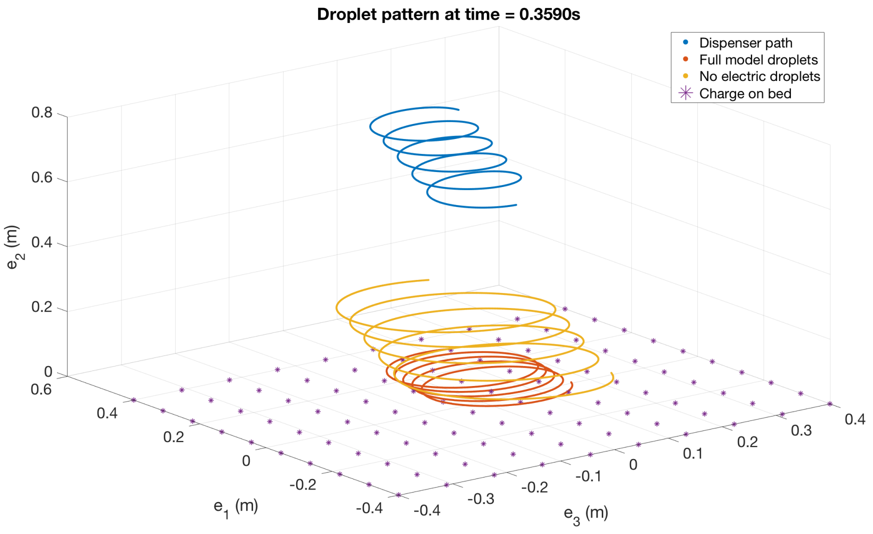


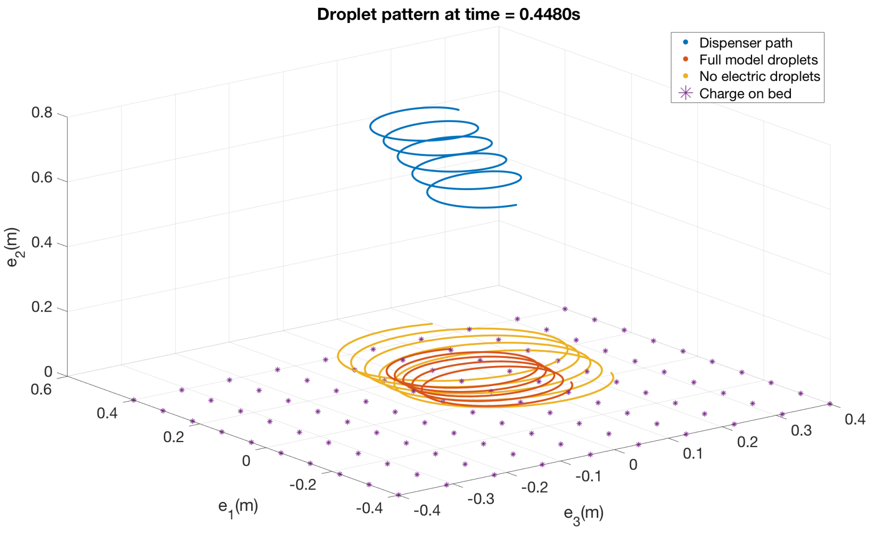
The comparison between the final configurations of the full model droplets and the droplets without electrical forces is shown below:



The animation of the the droplets as a full model and without electrical forces is shown below at different time:

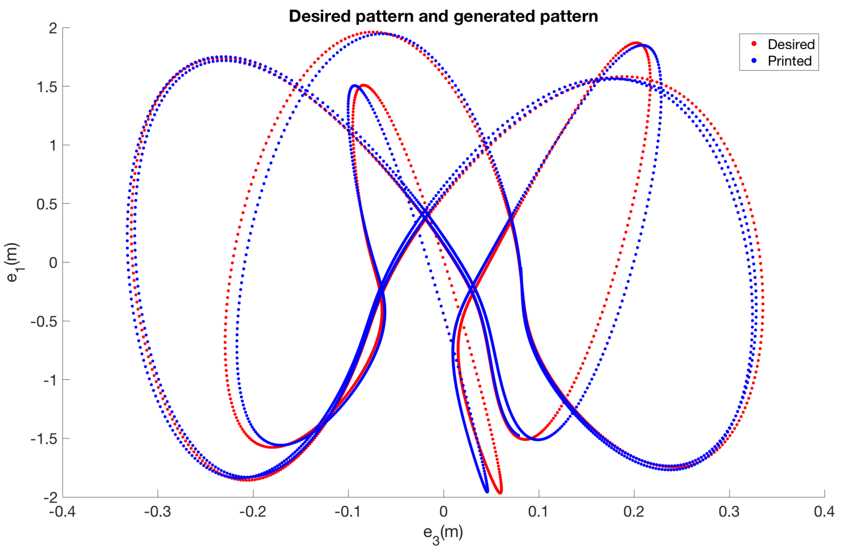




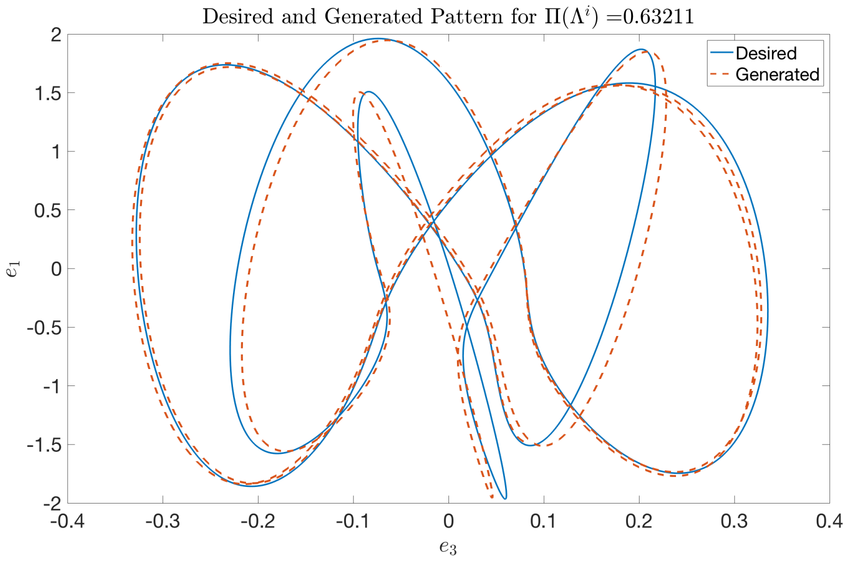


When the electrical charge is set to be zero, the droplets seems to still follow the shape of the dispenser. But, the time taken for the droplets to reach the print bed increases. This can be seen in the animation when all the full model droplets reach the print bed, the droplets with no electric are still floating. The final droplets configuration also expended larger than the full model compared to the dispenser path. This is caused by the drag force acting on the droplets. The electrical charged substrate on the print bed attract the droplets charge causing the droplets to fall faster and expanded less from the dispenser path therefore reducing the amount of error when printing.

For part 2, the final desired and generated droplets pattern from the genetic algorithm is plotted in $(z - x)$ axis which is the top-down view shown below:

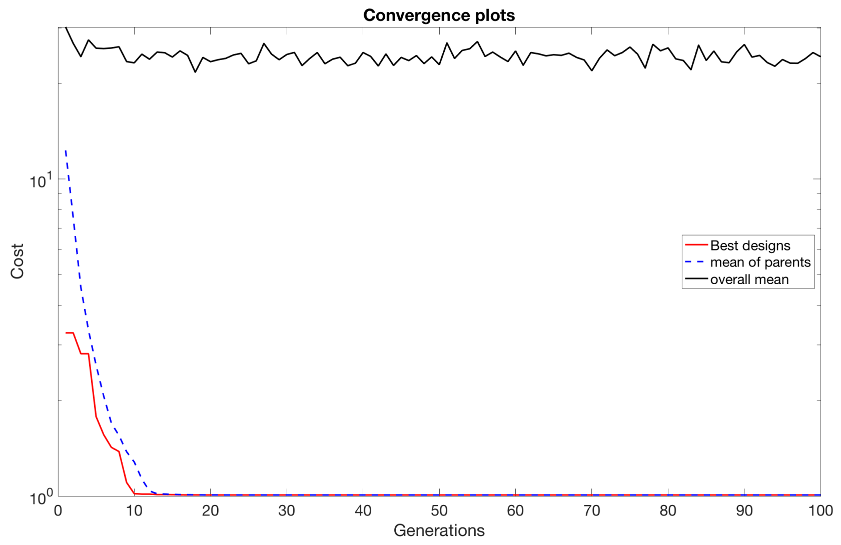


The generated droplets that is obtained is also compared with the provided helper function comparepattern.p shown below:



The printed droplets from the design string of the genetic algorithm is doing well in tracking the desired pattern. The error from the generated final droplets configuration is small since the cost function is just $\Lambda = 0.63211$.

The figure below shows the convergence plot from the genetic algorithm showing the cost of the best parents, mean cost of the parents and mean cost of the whole population for each generations:



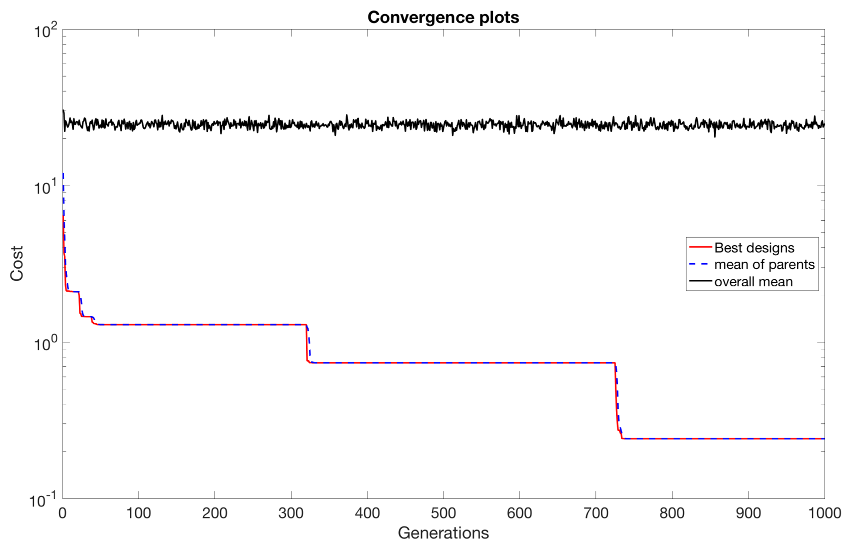
The four best performing design and its cost is reported in the table below:

Design	Λ_1	Λ_2	Λ_3	Λ_4	Π
1	15.7064	15.7120	6.2519	-3.1924	0.6321
2	15.7064	15.7120	6.2519	-3.1924	0.6321
3	15.7064	15.7120	6.2519	-3.1924	0.6321
4	15.7064	15.7120	6.2519	-3.1924	0.6321

Table 1: Best 4 Designs

The design string created by the genetic algorithm is different from part 1 which is $\lambda = [0.2, -0.2, 10, -1.2]$. It is because the search range for the values is different than part 1. The only design sting that is negative is λ_4 which is the relative droplet velocity that add the negative velocity out the droplets. It is negative in y axis to push the droplets faster towards the print bed.

Based on the convergence plot and the table of the best 4 designs, it is observed that the cost value is constant after a few generations. From the table, it is clearly shown that design parameters is also constant for the four best designs. The genetic algorithm seems to be stuck at some minimum point. This might be caused by the breeding technique and the number of generations run. When running 1000 generations, the convergence plot is shown below:



The convergence plot shows that the higher the number of generations the lower the cost value.

5 Conclusion

The main goal which is to find the best control parameters that will generate the least error of printed pattern is achieved. The operation of a of a free-form robotic 3D printer is also successfully simulated. Using the simulation of the full model dynamics the cost function is minimized using the genetic algorithm and the optimal solution is computed.