Control of Scalable Magnetic Levitation System with Deep Reinforcement Learning

Jeonghwan Kim
Seoul National University
whitealex95@snu.ac.kr

***Abstract*—A Novel Model of scalable 3D Magnetic Levitation System is proposed with its control scheme. The Control of the System is based on a Reinforcement Learning technique using Deep Deterministic Policy Gradient(DDPG) agent which succeeds in stabilizing the system with adequate energy efficiency.**

I. Introduction

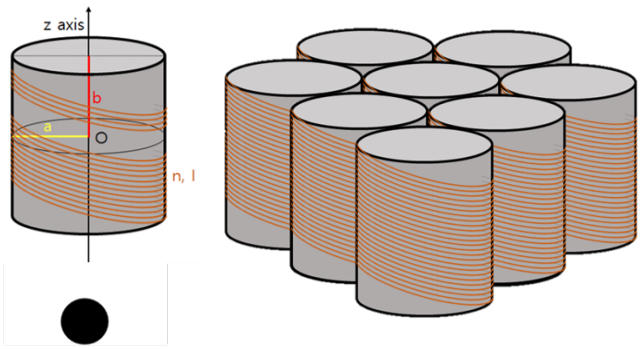
Magnetic Levitation System has come closer to our lives than any other days. Its applications are used in versatile fields of engineering, such as frictionless bearing, massive energy storing devices, and Maglev trains. Recently, computer science researchers have managed to use magnetic levitation system to realize a 3D Pixel; a tangible object in a real world, where the modification of such object is reflected to the virtual space inside a computer and the modification in the virtual space is again reflected to the object in the physical world. Such device requires a magnetically levitated object to be controlled with 3 degrees of freedom; a position in x, y, and z coordinates. zeroN, one of the most successful realization of 3D Pixel, utilizes a cylindrical solenoid and an xy-plotter to actuate a spherical magnet in a 3D space. However, due to the physical constraints, the system lacks scalability, only being capable of controlling a single magnet with limited speed.

In this paper, a novel way of realizing a scalable multi-object magnetic suspension system and its control scheme is proposed. The system is comprised of a 2D array of cylindrical electromagnets to generate spatial magnetic field and a spherical magnet to be levitated. The proposed dynamic system is highly nonlinear, unstable and difficult to linearize due to numerical error in calculating derivatives of elliptic integral terms inside the system. As a result, we control the system using the recent deep reinforcement learning techniques. As a result, the reinforcement learning agent succeeds in stabilizing the system with high robustness and with low energy consumption.

II. Backgrounds

A. Magnetic Levitation

Magnetic Levitation, or magnetic suspension is a method of levitating an object by canceling the gravitational force with force generated by external magnetic field. Although Earnshaws theorem states that paramagnetic materials cannot be levitated with stability in a static magnetic field, many work arounds such as using superconductors or diamagnetic levitation are developed. One of the most general and widely used method to achieve magnetic levitation system is use a feedback loop. One of the most famous application of such feedback loop based magnetic levitation system is a magnetic levitation train where the train wraps around the track and is lifted upward by magnetic force. In order to generate a stable feedback loop, knowledge about the system such as shape and strength of the magnetic field generated by the system are necessary.



(a) Single Electromagnet (b) Array of Electromagnets

Fig. 1: Magnetic Levitation Model with Cylindrical Solenoid. (a) is a classical example of a magnetic levitation system where the object can be controlled in only one dimension. Letters a, b, n, I each denotes the radius(m), height/2(m), windings(#/meter), and current(A). (b) is a 2-D array of cylindrical electromagnet which forms a plane parallel to the xy-plane

1) Magnetic Force of Cylindrical Electromagnet: In this section, we analyze magnetic force generated by a cylindrical solenoid drawn in Figure 1(a) According to the law of physics, the relation between magnetic field and the force generated by it is as follows.

$$dF = (dM \cdot \nabla)B = d\vec{M} \cdot \nabla\vec{B} \tag{1}$$

This implies that if we can obtain the magnetic field in space, we can calculate force from it. Luckily, with thanks to the symmetry, it is not hard to calculate the magnetic field of the points located on the axis of the cylinder.

$$\vec{B}_z = \frac{\mu_0 nI}{2}\left[\frac{z+b}{\sqrt{(z+b)^2+a^2}} - \frac{z-b}{\sqrt{(z-b)^2+a^2}}\right]\hat{z} \tag{2}$$

By calculating the z derivative of (2), force equation can be obtained:

$$\vec{F}_z = \vec{M}_z \frac{\partial B_z}{\partial z} = m_z \frac{\mu_0 n I}{2} \left[\frac{a^2}{((z+b)^2+a^2)^2} - \frac{a^2}{((z-b)^2+a^2)^2} \right] \quad (4)$$

where I is a current to the electromagnet, n is the number of rotations of coil per meter, and μ is the magnetic constant. The simplicity of the equation makes the system easy to linearize, leading to various types of control applications using magnetic levitation system.

In the general case, the magnetic field of a cylindrical electromagnet can be written as:

$$B_\rho = \frac{\mu_0 n I}{\pi} \left[\alpha_+ C\left(k_+, 1, 1, -1\right) - \alpha_- C\left(k_-, 1, 1, -1\right) \right] \quad (5)$$

$$B_z = \frac{\mu_0 n I}{\pi} \frac{a}{a+\rho} \left[\beta_+ C\left(k_+, \gamma^2, 1, \gamma\right) - \beta_- C\left(k_-, \gamma^2, 1, \gamma\right) \right] \quad (6)$$

where $C\left(k_c, p, c, s\right)$ is Bulirsch's Generalized Complete Elliptic Integral, with the constants:

$$\alpha_\pm = \frac{a}{\sqrt{z_\pm^2 + (\rho+a)^2}} \qquad \beta_\pm = \frac{z_\pm}{\sqrt{z_\pm^2 + (\rho+a)^2}}$$

$$k_\pm = \sqrt{\frac{z_\pm^2 + (\rho-a)^2}{z_\pm^2 + (\rho+a)^2}} \qquad \gamma = \frac{a-\rho}{a+\rho}$$

2) Bulirsch's Generalized Complete Elliptic Integral: Bulirsch's generalized complete elliptic Integral, (or just simply *cel*):

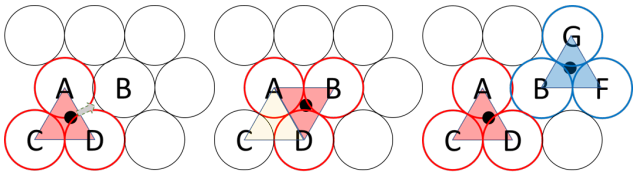
$$C\left(k_c, p, c, s\right) = \int_0^{\pi/2} \frac{c\cos^2\phi + s\sin^2\phi}{(\cos^2\phi + p\sin^2\phi)\sqrt{\cos^2\phi + k_c^2\sin^2\phi}} d\phi \quad (7)$$

is a typical form of representing complete elliptic integral that generalizes all three Legendre canonical forms (i.e. the elliptic integrals of the first, second and third kind) and the Carlson symmetric form. The relations between *cel* and other forms of elliptic integral are in the appendix.

Reinforcement Learning is a Learning frame work where a learning agent learns by interacting with the environment, observing the state and receiving rewards. At each time step t, an agent receives state value s_t from the interpretation of an environment and chooses an action a_t from the action space to interact with the environment. This results in a new state s_{t+1} and a step reward r_{t+1}. The objective of the reinforcement learning agent is to learn a policy function π that maps an action for each state so that it can maximize the return, a discounted sum of total reward the agent will receives.

In order to obtain the optimal policy, most reinforcement learning algorithms utilize action-value function which describes the expected return when the agent in state s_t takes action a_t.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}\left[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \ldots \,|s,a \right] \\ &= \mathbb{E}_{s'}\left[r + \lambda Q^\pi\left(s', a'\right) |s,a \right] \end{aligned} \quad (8)$$



(a) initial state (b) transitioned state (c) multiple levitation

Fig. 2: Solenoid Selection Rule. (a) Magnetic levitation force is generated by the three nearest solenoids. (b) The nearest solenoids are easily determined by checking which triangle the object is located at. (c) Since only three solenoids are used for each object, multiple objects can be levitated unless they are exactly adjacent to each other.

B. Reinforcement Learning

The action-value function is also noted as a Q function and the value at each state-action pair (s,a) is called the Q value. From the above equation, we can easily see that the optimum Q-value, which we denote Q^* satisfies the equation

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \lambda \max_{a'} Q^*\left(s', a'\right) |s,a \right] \quad (9)$$

and the optimal policy for state s_t is

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a) \quad (10)$$

It is known that if we repeatedly evaluate the Q value for the current policy, update the policy by using $argmax$ for each Q value, the Q value is guaranteed to converge to the optimal value. Many ways such as dynamic programming have been used to obtain the optimal Q function, but such methods are limited in that they are model based methods, meaning that we have to know the model, or the transition probability to update the function. However, Q-learning solves this problem by using temporal difference(TD) as an update rule:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right) \right] \quad (11)$$

where α is learning rate and the multiplicand of α is the temporal difference, a difference between the current value estimate and the future value estimate.

In order to adequately update the state-value function using the temporal difference, Q-learning does exploration based on ϵ-greedy method, where an agent takes greedy action with the probability of $1-\epsilon$ and takes a random action with the probability of ϵ.

The Q learning algorithm is highly useful for solving the reinforcement learning problem in a model free low dimensional, tabular spaces. However, in order to solve problems with large state and action spaces, Q-value for all state-action pair cannot be saved. Therefore, function approximators such as Neural Networks are utilized to approximate the Q value.

One of the most famous reinforcement learning algorithm Deep Q-Learning(DQN) is a q-learning algorithm which uses Deep Neural Networks as a function approximator to train an agent learn to play ATARI 2600 games while only given raw

pixel images. It enhances its performance using techniques such as Experience Replay and separate target networks, leading to the superhuman performance in many of the games it played.

Although DQN was able to solve high dimensional problems, it is not yet able to solve problems with continuous action spaces. To work around this problem, Deep Deterministic Policy Gradient(DDPG) algorithm utilizes an actor critic way to solve the continuous action space problem. Unlike DQN, an actor learns the policy directly, while a critic is used for evaluating the policy function estimated by the actor in a similar way as DQN. Also, due to its deterministic policy and the continuous nature, in order to do exploration, DDPG adds noise generated by Ornstein-Uhlenbeck random process, instead of using ϵ-greedy like Q-learning methods.

III. Model

In order to control a 3 dimensional location of a spherical magnet, an actuator that can generate magnetic forces to all directions(x,y,z) is needed. In the previous approach zeroN, the z-axis of a spherical magnet is controlled by a single electromagnet and the x-axis and y-axis are controlled by mechanically moving the electromagnet in the x,y direction using an xy-plotter.

Since the x,y position of the levitated object is controlled by moving a stable point, there is limit on the speed that the object can be moved. Also, due to the use of an xy-plotter for mechanically positioning the cylindrical magnet, only a single object can be controlled.

In this paper, we change the method so that a scalable 3d magnetic levitation is available by using multiple solenoids. In order for the system to be scalable, the magnet's position should be controlled only by electromagnetic force. To achieve this, 2D array of solenoids are positioned parallel to the xy-plane like in figure 1(b). Since each of the solenoid pulls the object towards its center, by superpositioning the forces generated by the array of solenoids, the plane will be able to generate the pure z-axis force, canceling out xy-directional force element.

During the activation of the system, it is intractable to activate all solenoids in the array. Therefore, only three solenoids that are nearest to the levitating object are activated for the control of one levitated object. Figure 2 illustrates such situation. Since the levitated object, illustrated as a black circle, is inside the red triangle$\triangle$ACD, the solenoids selected for the levitational force become the solenoids A,C, and D. Figure 2(b) illustrates a situation when the levitated object is moved to a different triangular region $\triangle$ADB, solenoids A,D,B being the three closest solenoids. As a result, solenoid C gets deactivated and the solenoid B will join the generation of magnetic levitation force. Figure 2(c) shows how multiple objects can be levitated and controlled. Unless objects are too close to each other, control of multiple levitation can be done almost independently given the fact that the magnetic field's intensity decreases rapidly in the radial direction.

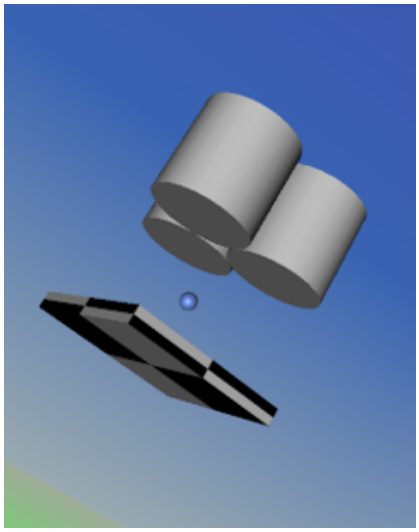


Fig. 3: 3D Modeled 3D Magnetic levitation system for simulation in MATLAB/Simulink

In order to calculate the exact magnetic force of an ideal electromagnet, equation (1) can be used by calculating the jacobian of a magnetic field:

$$\nabla \vec{B} = \begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_x}{\partial y} & \frac{\partial B_x}{\partial z} \\ \frac{\partial B_y}{\partial x} & \frac{\partial B_y}{\partial y} & \frac{\partial B_y}{\partial z} \\ \frac{\partial B_z}{\partial x} & \frac{\partial B_z}{\partial y} & \frac{\partial B_z}{\partial z} \end{bmatrix}$$

However, unlike the Legendre canonical forms, differentiation *cel* does not directly lead to a *cel*-like function. Therefore, the *cel* function in each of the elements is transformed to the Legendre canonical forms:

$$C\left(k_c,1,1,-1\right) = \frac{k_c^2+1}{k_c^2-1}\Pi\left(\sqrt{1-k_c^2}\right) - \frac{2}{k_c^2-1}E\left(\sqrt{1-k_c^2}\right)$$

$$C\left(k_c,\gamma^2,1,\gamma\right) = \frac{\gamma}{\gamma+1}\Pi\left(1-\gamma^2,\sqrt{1-k_c^2}\right) + \frac{1}{1+\gamma}K\left(\sqrt{1-k_c^2}\right)$$

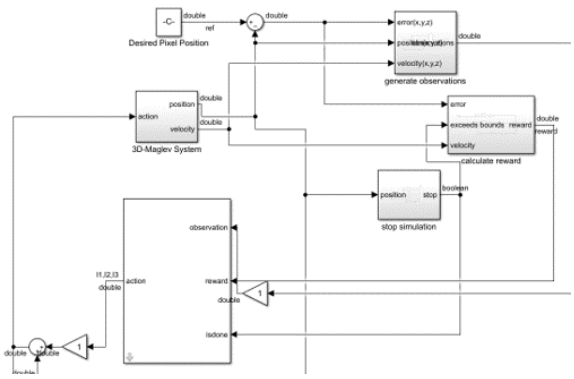
resulting in a modified representation of the magnetic field:

$$B_\rho = \frac{\mu_0 nI}{\pi}\left[\frac{\alpha_+}{k_+^2-1}\left\{\left(k_+^2+1\right)K\left(\sqrt{1-k_+^2}\right) - 2E\left(\sqrt{1-k_+^2}\right)\right\}\right.$$
$$\left. -\frac{\alpha_-}{k_-^2-1}\left\{\left(k_-^2+1\right)K\left(\sqrt{1-k_-^2}\right) - 2E\left(\sqrt{1-k_-^2}\right)\right\}\right.$$

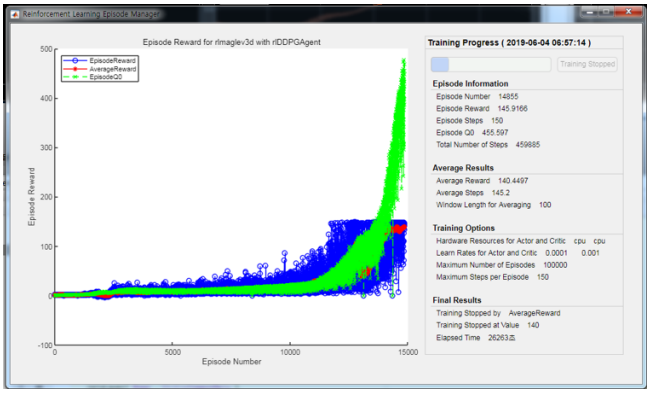
Using the derivatives of the canonical forms

$$\frac{\partial\Pi(n,k)}{\partial n} = \frac{1}{2(k^2-n)(n-1)}\left(E(k) + \frac{k^2-n}{n}\Pi(k) + \frac{n^2-k^2}{n}\Pi(n,k)\right)$$
$$\frac{\partial\Pi(n,k)}{\partial k} = \frac{k}{n-k^2}\left(\frac{E(k)}{k^2-1} + \Pi(n,k)\right)$$

and the partial derivatives of the coefficients $\alpha_\pm$, $\beta_\pm$, $k_\pm$, and γ in radial and axial directions,



(a) Magnetic Levitation System created in a simulink environment



(b) training status of a DDPG Agent

Fig. 4: (a) Magnetic Levitation System was simulated in a simulink simulator. (b) Training statistics of the agent. In only 7 hours in a single CPU, the agent learns how to stabilize in a highly unstable 3D Magnetic Levitation System

$$\frac{\partial a_\pm}{\partial \rho} = \frac{-a(a+\rho)}{\left(z_\pm^2+(a+\rho)^2\right)^{3/2}}, \frac{\partial \alpha_\pm}{\partial z} = \frac{-az_\pm}{\left(z_\pm^2+(a+\rho)^2\right)^{3/2}},$$
$$\frac{\partial \beta_\pm}{\partial \rho} = \frac{-z_\pm(a+\rho)}{\left(z_\pm^2+(a+\rho)^2\right)^{3/2}}, \frac{\partial \beta_\pm}{\partial z} = \frac{(a+\rho)^2-z_\pm^2}{\left(z_\pm^2+(a+\rho)^2\right)^{3/2}},$$
$$\frac{\partial k_\pm}{\partial \rho} = \frac{-2a\left(z_\pm^2+a^2-\rho^2\right)}{\sqrt{\left(z_\pm^2+(a+\rho)^2\right)\left(z_\pm^2+(a-\rho)^2\right)}},$$
$$\frac{\partial k_\pm}{\partial z} = \frac{4az_\pm\rho}{\sqrt{\left(z_\pm^2+(a+\rho)^2\right)\left(z_\pm^2+(a-\rho)^2\right)}},$$
$$\frac{\partial \gamma}{\partial \rho} = \frac{-2a}{(a+\rho)^2}, \frac{\partial \gamma}{\partial z} = 0$$

With the above result, the jacobian of the magnetic field can be derived in the cylindrical coordinates. It is than converted to cartesian coordinates using the equations below.

$$\frac{\partial B_x}{\partial x} = \frac{\partial B_\rho}{\partial x}\frac{x}{\rho} + B_\rho\frac{y^2}{\rho^3} = \frac{\partial B_\rho}{\partial \rho}\left(\frac{x}{\rho}\right)^2 + B_\rho\frac{y^2}{\rho^3}$$
$$\frac{\partial B_x}{\partial y} = \frac{\partial B_\rho}{\partial y}\frac{x}{\rho} - B_\rho\frac{xy}{\rho^3} = \frac{\partial B_\rho}{\partial \rho}\frac{xy}{\rho^2} - B_\rho\frac{xy}{\rho^3}$$
$$\frac{\partial B_x}{\partial z} = \frac{\partial B_\rho}{\partial z}\frac{xy}{\rho^2} - B_\rho\frac{xy}{\rho^3}$$
$$\frac{\partial B_y}{\partial x} = \frac{\partial B_\rho}{\partial x}\frac{y}{\rho} - B_\rho\frac{xy}{\rho^3} = \frac{\partial B_\rho}{\partial \rho}\frac{xy}{\rho^2} - B_\rho\frac{xy}{\rho^3}$$

With the jacobian of the magnetic field derived, the magnetic force the magnet receives can be calculated using the equation (1). Since the main focus of the magnetic levitation system is to cancel the gravitational force of the object, most of the magnetic forces generated are in the +z direction. As a result, the dipole axis of the levitated object lines up with the z-axis, leading to the simplified calculation of magnetic force.

IV. Simulation

Using the mathematical model derived from the previous section, we created a simulator for the 3d magnetic leviation system via matlab and simulink. The simulation dealt with 3 electromagnets, restricting the xy region of control to a triangular space and not taking account of the solenoid selection rule. The radius of the modeled ideal solenoid is 3cm, height is 6cm, and the solenoid is coiled 100 times per meter. The position of the three magnets are (0,0,0), (0.03,0.03$\sqrt{3}$,0), and (0.06,0,0).

Limiting the x,y axis to the center of the three triangles, the system succeeded in remaining stable using a PID controller, where the same current was applied to all three electromagnets. However, due to numerical unstability that appears in the subtraction of elliptic integral inside the force calculation formula, the system could not linearize accurately, making it hard to apply classical control techniques for general location. As a work around, the Deep Deterministic Policy Gradient(DDPG), a well known algorithm in the reinforcement learning framework was implemented to control the system.

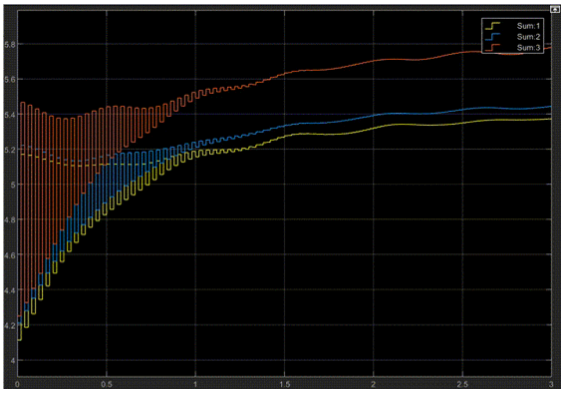
V. Control via Reinforcement Learning

In order to transfer the 3d magnetic levitation system to the reinforcement learning framework, reward function generator and failure detector was added to the system. The objective of this learning problem was to stay levitated and reach an arbitrary target location with stability. In order to reinforce the agent to stay levitated without falling, the agent was given bonus for staying in a safe region while being punished for being far from target location. The norm of the velocity of the object was used as a penalty in order to decrease the oscillation that occurs frequently in feedback systems. The resulting reward function becomes:

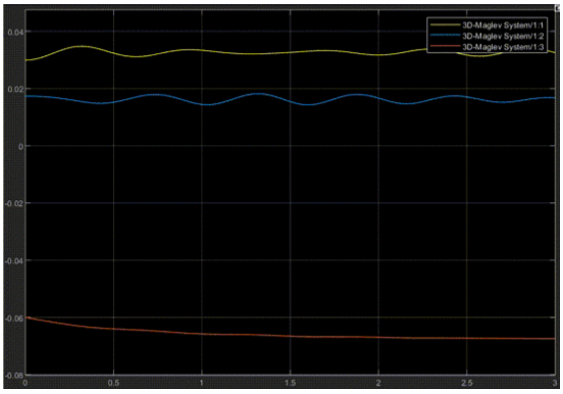
$$r_t = is\_alive - 200error_t^2 - 0.5v_t^2$$

where $error_t$ is the error in location between target location and current location, and v_t is the velocity of the object. The agent is considered *not_alive* if the position of the agent went across the triangular prism created by the three axes or if the magnet fell below 10 centimeters.

The DDPG Agent received 9 dimensional vector of position, velocity, and error(a distance between the target and the agent). Receiving such state information, the agent generates an action signal, a 3 dimensional vector that indicates the current applied to each of the three electromagnets. The agent has two networks, an actor network which learns the policy mapping from



(a) Current Needed for Levitation



(b) Position of the Levitated Object

Fig. 5: (a) DDPG Agent learns a policy that uses far less current than the maximum current available. It's yellow, blue, red line stands for the currents of solenoid 1, 2, and 3 (b) The DDPG agent succeeds in both levitating and stabilizing the magnetic object. Its yellow, blue, red line stands for x,y,z position of the levitated object.

state to action and a critic network that tends to approximate the Q-value. Both actor and critic networks are comprised of fully connected layers. Critic Network recieves both action and state, and each of them go through two different fully connected layer of sizes 50 and 25. The output of the two layers are than added up and passes one more Fully connected layer with ReLU activation. The actor network passes through a relative small network which is consisted of only one hidden layer of size 9 with tanh activation. The training was done on MATLAB's Reinforcement Learning Toolbox. Learning was set to end when the agent learns to stay levitated for 3 seconds and received low position/velocity penalty. The Learning took only y hours of training in a synchronous CPU system, which ran less than 15000 episodes. The learning curve is shown as in the Figure 4(b).

Performance of the trained agent is plotted in Figure 5(a) and (b). Figure 5 is an experiment where the object is placed near (0.03, 0.01*$\sqrt{(3)}$,-0.06) + small gaussian error, and the target is also placed at (0.03, 0.01*$\sqrt{(3)}$,-0.06) + small gaussian error. Figure 5(a) shows that even though the agent could use up to 20 A of current, the agent only managed to use significantly less amount of current. Which was very different from the result of the PID controller which used up to 80A of current when the limit was not set.

Figure 5(b)'s yellow, blue, red line stands for x,y,z position of the levitated object. It is interesting how smoothly the levitated object moves as time passes. One of the most satisfsying fact is that the agent managed to stabilize both the x-directional and the y-directional errors, which the controller created in the previous section could not accomplish.

VI. Discussion and Future Work

A newly designed scalable 3d magnetic levitation system was successfully controlled in the simulation environment by using the reinforcement learning techniques. The controller learned from the DDPG agent succeeded in controlling the system with adequate amount of current which is bearable to affordable electronic devices. Also, it is be expected that the controller's performance can get better by using bigger networks for the agent's actor network, which was formally only a layer deep.

Since the newly modeled system can move freely in a 3D physical space without making any contact with the external environment, the system will be useful for representing multiple pixel in a digital system to the physical system. It will also be useful in ultra clean environments where small disturbance of the environment might result in disastrous results.

Through this model, it is shown that the reinforcement learning agent can take over the place where classical control used to take place in. With the rapid advances in the field of reinforcement learning, more places of the classical control may be replaced due generality of the reinforcement learning algorithms

References

[1] BERRY, Michael Victor; GEIM, Andre Konstantin. Of flying frogs and levitrons. European Journal of Physics, 1997, 18.4: 307.

[2] CRAIN, C. D.; BROWN, M. D.; CORTNER, A. H. Design and initial calibration of a magnetic suspension system for wind tunnel models. ARNOLD ENGINEERING DEVELOPMENT CENTER ARNOLD AFB TN, 1965.

[3] LEE, Jinha; POST, Rehmi; ISHII, Hiroshi. ZeroN: mid-air tangible interaction enabled by computer controlled magnetic levitation. In: Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011. p. 327-336.

[4] DERBY, Norman; OLBERT, Stanislaw. Cylindrical magnets and ideal solenoids. American Journal of Physics, 2010, 78.3: 229-235.

[5] TEUKOLSKY, Saul A., et al. Numerical recipes in C. SMR, 1992, 693.1: 59-70.

[6] JENKINS, A. W.; PARKER, Hermon M. Electromagnetic support arrangement with threedimensional control. I. theoretical. Journal of Applied Physics, 1959, 30.4: S238-S239.

[7] CRAIN, C. D.; BROWN, M. D.; CORTNER, A. H. Design and initial calibration of a magnetic suspension system for wind tunnel models. ARNOLD ENGINEERING DEVELOPMENT CENTER ARNOLD AFB TN, 1965.

[8] LILLICRAP, Timothy P., et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

APPENDIX

A. Relation between cel and Legendre Form

Elliptic Integral is a function that can be written as

$$f(x) = \int_c^x R(t, \sqrt{P(t)})dt$$

where c is a constant, R is a rational function, P is a polynomial with degree 3 or 4 without repeated root. It is known that every elliptic integral can be reduced to three Legendre canonical forms and integral over rational functions. The first, second, and third kind of Legendre canonical form of incomplete elliptic integral are

$$F(\phi, k) = \int_0^\phi \frac{1}{\sqrt{1 - k^2 \sin^2(t)}} dt$$

$$E(\phi, k) = \int_0^\phi \sqrt{1 - k^2 \sin^2(t)} dt$$

$$\Pi(\phi, n, k) = \int_0^\phi \frac{1}{\left(1 - n \sin^2(t)\right)\sqrt{1 - k^2 \sin^2(t)}} dt$$

We call the elliptic integrals complete when the ϕ becomes $\pi/2$. In this case, the complete elliptic integral of first, second, and third kind become:

$$K(k) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1 - k^2 \sin^2\theta}} = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}$$

$$E(k) = \int_0^{\frac{\pi}{2}} \sqrt{1 - k^2 \sin^2\theta} d\theta = \int_0^1 \frac{\sqrt{1-k^2t^2}}{\sqrt{1-t^2}} dt$$

$$\Pi(n, k) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\left(1 - n \sin^2\theta\right)\sqrt{1 - k^2 \sin^2\theta}}$$

Since the elliptic integrals cannot be expressed in elementary functions, it has to be calculated using numerical methods. In general, Carlson's symmetric forms are used to calculate the incomplete elliptical integrals.

$$R_F(x,y,z) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}}$$
$$R_J(x,y,z,p) = \frac{3}{2}\int_0^\infty \frac{dt}{(t+p)\sqrt{(t+x)(t+y)(t+z)}}$$
$$R_C(x,y) = R_F(x,y,y) = \frac{1}{2}\int_0^\infty \frac{dt}{(t+y)\sqrt{(t+x)}}$$
$$R_D(x,y,z) = R_J(x,y,z,z) = \frac{3}{2}\int_0^\infty \frac{dt}{(t+z)\sqrt{(t+x)(t+y)(t+z)}}$$

However, it is known that Bulirsch's generalized elliptic integral, *cel*, works well in the complete case. The complete elliptic integrals of the First, Second, and Third kinds can be represented using *cel* by

$$C\left(k_c, p, c, s\right) = \int_0^{\pi/2} \frac{c\cos^2\phi + s\sin^2\phi}{(\cos^2\phi + p\sin^2\phi)\sqrt{\cos^2\phi + k_c^2\sin^2\phi}} d\phi$$
$$K(k) = C\left(\sqrt{1-k^2}, 1, 1, 1\right)$$
$$E(k) = C\left(\sqrt{1-k^2}, 1, 1, 1-k^2\right)$$
$$\Pi(n, k) = C\left(\sqrt{1-k^2}, n+1, 1, 1\right)$$

In cases where the derivatives of *cel* needs to be calculated, its derivative does not fit to the form of *cel* directly. As a work around, the derivative can be represented using the derivatives of the Legendre canonical forms

$$\frac{\partial K(k)}{\partial k} = \frac{E(k)}{k(1-k^2)} - \frac{K(k)}{k}$$
$$\frac{\partial E(k)}{\partial k} = \frac{E(k)-K(k)}{k}$$
$$\frac{\partial \Pi(n,k)}{\partial n} = \frac{1}{2(k^2-n)(n-1)}\left(E(k) + \frac{k^2-n}{n}K(k) + \frac{n^2-k^2}{n}\Pi(n,k)\right)$$
$$\frac{\partial \Pi(n,k)}{\partial k} = \frac{k}{n-k^2}\left(\frac{E(k)}{k^2-1} + \Pi(n,k)\right)$$

B. Derivatives of Magnetic Field in Cylindrical Coordinates

Using the above facts and the derivations in the section 3, the derivatives of the magnetic field of the cylindrical solenoid in cylindrical coordinates can be written as follows:

$$\begin{aligned}
\frac{\partial B_\rho}{\partial \rho} =& \frac{\mu_0 nI}{\pi}\left[\frac{\partial \alpha_+}{\partial \rho}\left(\frac{k_+^2+1}{k_+^2-1}K\left(\sqrt{1-k_+^2}\right) - \frac{2}{k_+^2-1}E\left(\sqrt{1-k_+^2}\right)\right) - \frac{\partial k_+}{\partial \rho}\frac{4\alpha_+}{(k_+^2-1)^2}K\left(\sqrt{1-k_+^2}\right)\right.\\
&+ \frac{\partial k_+}{\partial \rho}\frac{\alpha_+}{(k_+^2-1)^2}\left(E\left(\sqrt{1-k_+^2}\right)\left(\frac{1}{k_+}+3k_+\right) - K\left(\sqrt{1-k_+^2}\right)(3-k_+^2)k_+\right)\\
&- \frac{\partial \alpha_-}{\partial \rho}\left(\frac{k_-^2+1}{k_-^2-1}K\left(\sqrt{1-k_-^2}\right) - \frac{2}{k_-^2-1}E\left(\sqrt{1-k_-^2}\right)\right) + \frac{\partial k_-}{\partial \rho}\frac{4\alpha_-}{(k_-^2-1)^2}K\left(\sqrt{1-k_-^2}\right)\\
&\left.- \frac{\partial k_-}{\partial \rho}\frac{\alpha_-}{(k_-^2-1)^2}\left(E\left(\sqrt{1-k_-^2}\right)\left(\frac{1}{k_-}+3k_-\right) - K\left(\sqrt{1-k_-^2}\right)(3-k_-^2)k_-\right)\right.
\end{aligned}$$

$$\begin{aligned}
\frac{\partial B_\rho}{\partial z} =& \frac{\mu_0 nI}{\pi}\left[\frac{\partial \alpha_+}{\partial z}\left(\frac{k_+^2+1}{k_+^2-1}K\left(\sqrt{1-k_+^2}\right) - \frac{2}{k_+^2-1}E\left(\sqrt{1-k_+^2}\right)\right) - \frac{\partial k_+}{\partial z}\frac{4\alpha_+}{(k_+^2-1)^2}K\left(\sqrt{1-k_+^2}\right)\right.\\
&+ \frac{\partial k_+}{\partial z}\frac{\alpha_+}{(k_+^2-1)^2}\left(E\left(\sqrt{1-k_+^2}\right)\left(\frac{1}{k_+}+3k_+\right) - K\left(\sqrt{1-k_+^2}\right)(3-k_+^2)k_+\right)\\
&- \frac{\partial \alpha_-}{\partial z}\left(\frac{k_-^2+1}{k_-^2-1}K\left(\sqrt{1-k_-^2}\right) - \frac{2}{k_-^2-1}E\left(\sqrt{1-k_-^2}\right) + \frac{\partial k_-}{\partial z}\frac{4\alpha_-}{(k_-^2-1)^2}K\left(\sqrt{1-k_-^2}\right)\right.\\
&\left.- \frac{\partial k_-}{\partial z}\frac{\alpha_-}{(k_-^2-1)^2}\left(E\left(\sqrt{1-k_-^2}\right)\left(\frac{1}{k_-}+3k_-\right) - K\left(\sqrt{1-k_-^2}\right)(3-k_-^2)k_-\right)\right.
\end{aligned}$$

$$\begin{aligned}
\frac{\partial B_z}{\partial \rho} =& -\frac{\mu_0 nI}{\pi}\frac{a}{(a+\rho)^2}\left[\frac{\beta_+}{1+\gamma}\left\{\gamma\Pi\left(1-\gamma^2, \sqrt{1-k_+^2}\right) + K\left(\sqrt{1-k_+^2}\right)\right\} - \frac{\beta_-}{1+\gamma}\left\{\gamma\Pi\left(1-\gamma^2, \sqrt{1-k_-^2}\right) + K\left(\sqrt{1-k_-^2}\right)\right\}\right]\\
&+ \frac{\mu_0 nI}{\pi}\frac{a}{a+\rho}\left[\frac{\partial \beta_+}{\partial \rho}\frac{1}{1+\gamma}\left\{\gamma\Pi\left(1-\gamma^2, \sqrt{1-k_+^2}\right) + K\left(\sqrt{1-k_+^2}\right)\right\} + \frac{\partial \gamma}{\partial \rho}\frac{\beta_+}{(1+\gamma)^2}\left\{\begin{array}{l}\Pi\left(1-\gamma^2, \sqrt{1-k_+^2}\right)\left(1+\frac{\gamma^4-2\gamma^2+k_+^2}{(\gamma^2-k_+^2)(1-\gamma)}\right)\\ +K\left(\sqrt{1-k_+^2}\right)\frac{\gamma}{1-\gamma} + E\left(\sqrt{1-k_+^2}\right)\frac{1+\gamma}{\gamma^2-k_+^2}\end{array}\right\}\right.\\
&- \frac{\partial k_+}{\partial \rho}\frac{k_+\beta_+}{1+\gamma}\left\{\frac{\gamma}{k_+^2-\gamma^2}\Pi\left(1-\gamma^2, \sqrt{1-k_+^2}\right) + \frac{1}{k_+^2-1}K\left(\sqrt{1-k_+^2}\right) + \frac{1}{k_+^2}\left(\frac{1}{1-k_+^2} - \frac{1}{k_+^2-\gamma^2}\right)E\left(\sqrt{1-k_+^2}\right)\right\}\\
&- \frac{\partial \beta_-}{\partial \rho}\frac{1}{1+\gamma}\left\{\gamma\Pi\left(1-\gamma^2, \sqrt{1-k_-^2}\right) + K\left(\sqrt{1-k_-^2}\right)\right\} - \frac{\partial \gamma}{\partial \rho}\frac{\beta_-}{(1+\gamma)^2}\left\{\begin{array}{l}\Pi\left(1-\gamma^2, \sqrt{1-k_-^2}\right)\left(1+\frac{\gamma^4-2\gamma^2+k_-^2}{(\gamma^2-k_-^2)(1-\gamma)}\right)\\ +K\left(\sqrt{1-k_-^2}\right)\frac{\gamma}{1-\gamma} + E\left(\sqrt{1-k_-^2}\right)\frac{1+\gamma}{\gamma^2-k_-^2}\end{array}\right\}\\
&+ \frac{\partial k_-}{\partial \rho}\frac{k_-\beta_-}{1+\gamma}\left\{\frac{\gamma}{k_-^2-\gamma^2}\Pi\left(1-\gamma^2, \sqrt{1-k_-^2}\right) + \frac{1}{k_-^2-1}\Pi\left(\sqrt{1-k_-^2}\right) + \frac{1}{k_-^2}\left(\frac{1}{1-k_-^2} - \frac{1}{k_-^2-\gamma^2}\right)E\left(\sqrt{1-k_-^2}\right)\right\}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial B_\rho}{\partial z} =& \frac{\mu_0 nI}{\pi}\left[\frac{\partial \alpha_+}{\partial z}\left(\frac{k_+^2+1}{k_+^2-1}K\left(\sqrt{1-k_+^2}\right) - \frac{2}{k_+^2-1}E\left(\sqrt{1-k_+^2}\right)\right) - \frac{\partial k_+}{\partial z}\frac{4\alpha_+}{(k_+^2-1)^2}K\left(\sqrt{1-k_+^2}\right)\right.\\
&+ \frac{\partial k_+}{\partial z}\frac{\alpha_+}{(k_+^2-1)^2}\left(E\left(\sqrt{1-k_+^2}\right)\left(\frac{1}{k_+}+3k_+\right) - K\left(\sqrt{1-k_+^2}\right)(3-k_+^2)k_+\right)\\
&- \frac{\partial \alpha_-}{\partial z}\left(\frac{k_-^2+1}{k_-^2-1}K\left(\sqrt{1-k_-^2}\right) - \frac{2}{k_-^2-1}E\left(\sqrt{1-k_-^2}\right) + \frac{\partial k_-}{\partial z}\frac{4\alpha_-}{(k_-^2-1)^2}K\left(\sqrt{1-k_-^2}\right)\right.\\
&\left.- \frac{\partial k_-}{\partial z}\frac{\alpha_-}{(k_-^2-1)^2}\left(E\left(\sqrt{1-k_-^2}\right)\left(\frac{1}{k_-}+3k_-\right) - K\left(\sqrt{1-k_-^2}\right)(3-k_-^2)k_-\right)\right.
\end{aligned}$$