Contract: HR0011-18-C-0100
Contractor: AIMdyn, Inc., 1919 State Street, Suite 207, Santa Barbara, CA 93101

R&D STATUS REPORT

1 The date work started.

2 Description of progress during the reporting period, supported by reasons for any change in approach reported previously.

2.1 Discovering the energy eigenfunction using LSTM's Over the month of August, we continued coding, implementing, and debugging the associated software for for using Deep Learning to find Koopman principal eigenfunctions. Specifically, we continued developing the software for recurrent networks, specifically LSTMs (long-term short-term memory networks).

We obtained initial results of optimizing

$$\min_{\phi} \sum_{t=1}^{T} \dot{\mathbf{r}}_t \cdot \nabla\phi(\mathbf{r}_t), \tag{1}$$

where $\phi(\mathbf{r}_t)$ is the output of a deep networks with corresponding inputs $\mathbf{r}_t = (x_t, y_t, \dot{x}_t, \dot{y}_t)$, over T time-steps. The optimization would drive the objective function to a large negative value. However, we did not see that the model would recover an energy function. The corresponding output of the neural net was a square wave that was positive on the first half of the orbit and negative on the second half of the orbit. The energy function (an eigenfunction corresponding to $\lambda = 0$) should be constant on trajectories.

Instead, the following optimization problem was formulated:

$$\min_{\phi} \sum_{t=1}^{T} \|\dot{\mathbf{r}}_t \cdot \nabla\phi(\mathbf{r}_t)\|_2^2, \tag{2}$$

Hence looking at the sum of L_2 norms. This optimization problem forces the output function ϕ to satisfy the Koopman eigenfunction criterion

$$\mathbf{F}(\mathbf{r}) \cdot \nabla\phi(\mathbf{r}) = \lambda\phi(\mathbf{r}) = 0 \qquad (\text{since } \lambda = 0), \tag{3}$$

at every instance in time. The network was able to learn a plausible energy function $\phi(\mathbf{r})$, which is just a constant. Different constant values of the energy would correspond to different initializations of a deep network.

We are currently looking into ways of constraining the output of a deep network, so that we could discover nonhomogeneous invariants. But the first result does demonstrate that we are able to discover a meaningful solution.

2.2 Learning PDEs from data The effort this period had two directions. The first had to do with the use of neural networks in learning partial differential equations; first, partial differential equations for which we "have explicit equations", and then, more importantly, partial differential equations that are "emergent" – that is, equations that arise from, say, coupled oscillator models, and for which we do not really yet have explicit expressions. The second direction had to do with the collaboration with AIMDyN, towards the machine-learning study of planetary data for the "automated discovery" of Newtonian gravity.

2.3 Diffusion Maps/Data Mining for physical laws We have worked on extract physical laws from time series data using Diffusion Maps fro mine the data. Two approaches are being pursued:

(a) We are given a time-series of data pairs $\{(\mathbf{r}_t, \mathbf{F}_t) : t = 0, \dots, T\}$, where $\mathbf{r}_t$ is the state at time t and $\mathbf{F}_t$ is the force (right hand side of the ODE) at time t. Note $\mathbf{F}_t$ would not be normally given and would have to be determined via automatic differentiation. Using $\mathbf{F}$ directly will allow us to separate algorithmic issues coming from the data mining with those coming from automatic differentiation methods.

(b) We are only given state information $\{\mathbf{r}_t : t = 0, \dots, T\}$.

Approach (a): Consider data from a single planet. If we do Diffusion Maps for the PAIRS of data $(\mathbf{r}_t, \mathbf{F}_t)_{t=0}^{T}$ we will discover "immediately" that the data are one-dimensional, with some coordinate θ. This will be some "graph" curve in $(\mathbf{r}, \mathbf{F})$-space. We will be able to write the state and force as functions of the diffusion coordinate θ:

$$\mathbf{r}(\theta) \text{ and } \mathbf{F}(\theta). \tag{4}$$

We will compare obtaining the law if the following ways:

1. Train a neural net with $\mathbf{r}$ as input and $\mathbf{F}$ as output.
2. Do semi-supervied learning. For a new point $\hat{\mathbf{r}}$, use the nearby pairs $(\mathbf{r}(\theta), \mathbf{F}(\theta))$, where "nearby" is measured using the diffusion coordinate θ, and get the force $\hat{\mathbf{F}}$.
3. Using a set of basis functions to fit the "graph" of the data.

A note pertaining to (iii). Given a physical law, there are an infinite number of representations (change of variables) that describe the same physics. The familiar inverse-square law is the most parsimonious. The choice of basis in (iii) used to fit the data determines the parsimony of the representation. The question of parsimony drives a particular choice of basis and will, at this stage, be determined by human intervention/intuition.

If we allow for trajectories from multiple planets as the training set, diffusion maps will find a two-dimensional surface. The first direction will be given by the previously determined phase coordinate θ. The second principal diffusion coordinate (one NOT given by a harmonic of the diffusion eigenfunction corresponding to θ) should correspond to a function dependent on the masses of the planets. We conjecture that these diffusion coordinates will be factorable, in the sense that one varies on the level sets of the other. This will be tested. To get the law, we can train a neural net having two inputs, say $\mathbf{r}$ and m, and one output $\mathbf{F}$.

Approach (b): This situation by-passes the need for the force or any issues with automatic differentiation. We are given just the state state $\{\mathbf{r}_t : t = 0, \dots, T\}$. Many approach learn the time 1 map between the states $\Phi^1(\mathbf{r}_t) = r_{t+1}$. This map, however, has different asymptotic properties than the continuous function it is trying to approximate. This approach can be termed "*a discrete approximation of a continuous object*". Our approach can be considered the dual of the previous one: *we learn a continuous function whose discretization gives back the data.* Of course, there is freedom in the choice of discretization. We approach this using neural networks templated off a numerical integration scheme: say Runge-Kutta 4(5): letting $t_i = t_0 + ih$, the RK45 for solving the IVP

$$\begin{aligned} y' &= f(t, y) \\ y(t_0) &= \alpha \end{aligned} \tag{5}$$

is

$$\begin{aligned}
Y_0 &= \alpha \\
k_1 &= h\,f(t_i, Y_i) \\
k_2 &= h\,f(t_i + \frac{h}{2}, Y_i + \frac{k_1}{2}) \\
k_3 &= h\,f(t_i + \frac{h}{2}, Y_i + \frac{k_2}{2}) \\
k_4 &= h\,f(t_i + h, Y_i + k_3) \\
Y_{i+1} &= Y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned} \tag{6}$$

We use a deep net to approximate f and take *copies* of it and connect them into the Runge-Kutta scheme. The deep net learns the continuous f and its Runge-Kutta discretization returns the data $\{\mathbf{r}_t\}_{t=0}^{T}$. Figure 1 gives a schematic of this approach.

3 Planned activities and milestones for the next reporting period.

1. Continue exploring constraining the network so that we could discover nonhomogeneous invariants using deep learning.
2. Begin numerical testing of approaches (a) and (b) to "discover" Newtonian gravity through data mining. we will use neural nets, diffusion maps and semi supervised learning, and we hope to
3. Begin exploring how things "start to fail" when we approach limits where general relativity becomes important.

4 Description of any major items of experimental or special equipment purchased or constructed during the reporting period.

N/A

5 Notification of any changes in key personnel associated with the contract during the reporting period.

N/A

6 Summary of substantive information derived from noteworthy trips, meetings, and special conferences held in connection with the contract during the reporting period.

N/A

7 Summary of all problems or areas of concern.

N/A

8 Related accomplishments since last report.

We have been able to show that we can "learn" partial differential equations like the viscous Burgers from data; and that we can use the laws learned (through deep nets) to successfully simulate these PDEs.