# Forecasting with Reservoir Computing

Brian Hunt, Zhixin Lu
Michelle Girvan, Edward Ott, Jaideep Pathak
University of Maryland

We are grateful for funding from DARPA and ARO.

# Machine Learning

- Machine learning algorithms are trained with data to perform a particular task, such as classification.
- In most applications, the task is static; training data is a set of input-output pairs.
- Starting from generic input-output rules, training means choosing the parameters of these rules to optimize the difference between the actual outputs and the desired outputs.
- Reservoir computing is a type of machine learning well suited to dynamic tasks, mapping an input time series to an output time series.

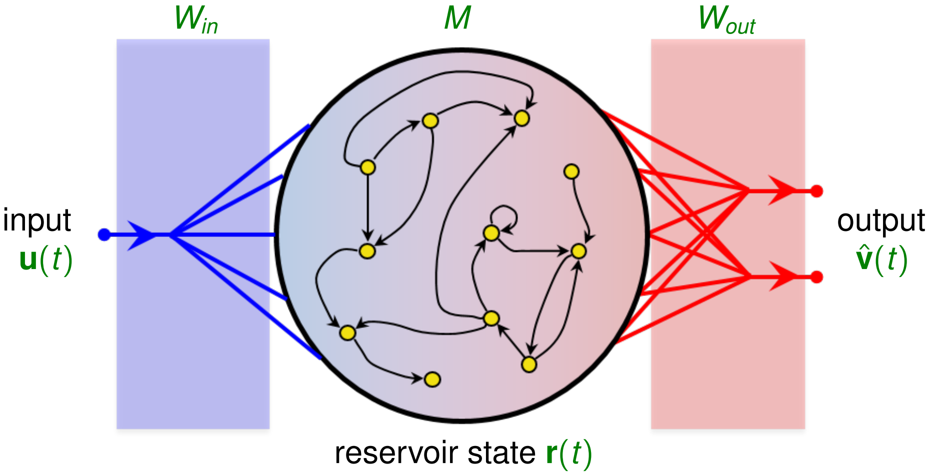
# Our Goals

- We seek to use reservoir computing to create an ad hoc model, capable of forecasting or data assimilation, based only on a finite-time sample (training data) from a dynamical system.
- We are interested in two tasks for our forecast model:
- Predict “weather”: Predict future measurements, at least in near future.
- Learn “climate”: Compute arbitrarily long times series that mimic the system that generated the data.
- We also can create an analysis model to infer unmeasured variables from measured variables.

# Reservoir Computing

- A reservoir is a recurrent neural network whose internal parameters are not adjusted to fit the data for a particular task.
- Train the reservoir by feeding it an input time series $\mathbf{u}(t)$ and fitting a linear function of the reservoir state variables $\mathbf{r}(t)$ to a desired output time series $\mathbf{v}(t)$.
- Goal: train the reservoir to estimate future values of the desired output from future values of the input.
- This approach was proposed as Echo State Networks (Jaeger 2001) and Liquid State Machines (Maass, Natschlaeger, Markram 2002); see `http://www.scholarpedia.org/article/Echo_state_network`

# Reservoir for Training and Inference



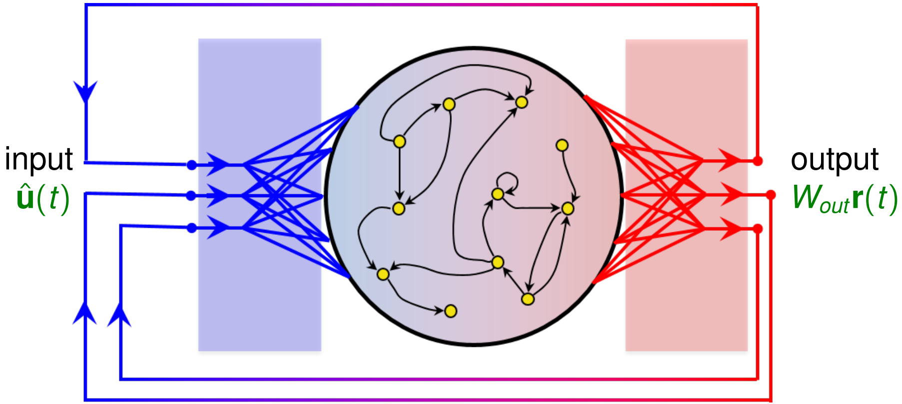

Matrices $W_{in}$ and $M$ are chosen randomly in advance

# Continuous-Time Jaeger ESN

- Listen: $\beta d\mathbf{r}/dt = -\mathbf{r}(t) + \tanh[M\mathbf{r}(t) + W_{in}\mathbf{u}(t)]$
- For training, we choose $\beta$ to match the input time scale and we listen for $-\tau \leq t \leq T$.
- Here $\tau$ is a transient time, and $T$ is the training time period.
- Fit: Find the matrix $W_{out}$ such that $\hat{\mathbf{v}}(t) = W_{out}\mathbf{r}(t)$ least-squares minimizes the residuals $\hat{\mathbf{v}}(t) - \mathbf{v}(t)$ for $0 \leq t \leq T$.
- To train a forecast model, we let $\mathbf{v}(t) = \mathbf{u}(t)$.

# Forecast Model for $t > T$

Predict: $\beta\, d\mathbf{r}/dt = -\mathbf{r}(t) + \tanh[M\mathbf{r}(t) + W_{in}W_{out}\mathbf{r}(t)]$



# Example: Lorenz System

- We generated and tried to predict a trajectory of the Lorenz system:

$$\frac{dx}{dt} = 10(y-x), \quad \frac{dy}{dt} = x(28-z)-y, \quad \frac{dz}{dt} = xy+8z/3.$$

- Listening input: $\mathbf{u}(t) = [x(t), y(t), z(t)]^T$ for forecasting; for inference, $\mathbf{u}(t) = x(t)$ and $\mathbf{v}(t) = [y(t), z(t)]^T$.
- After a transient time $\tau = 100$, we train the reservoir for time $T = 60$.
- Technical note: To perform the fit to $z(t)$, we squared some of the coordinates of $\mathbf{r}(t)$ before performing linear regression.

# Reservoir Parameters

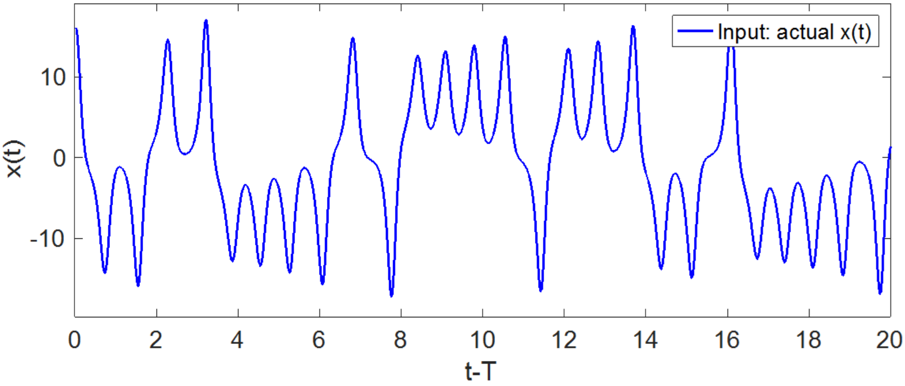
- We use a reservoir of 2000 nodes, using random (Erdös-Rényi) connections with an average degree of 40; thus, $M$ is a sparse matrix (2% nonzero entries).
- Connection strengths (nonzero elements of $M$) are chosen from a uniform distribution on $[-1, 1]$; then $M$ is rescaled so that the magnitude of its largest eigenvalue is 0.9.
- Each row of $W_{in}$ has one nonzero element, chosen from a uniform distribution on $[-1, 1]$, and scaled by a parameter $w = 0.11$.

# Results for Inference and Forecasting

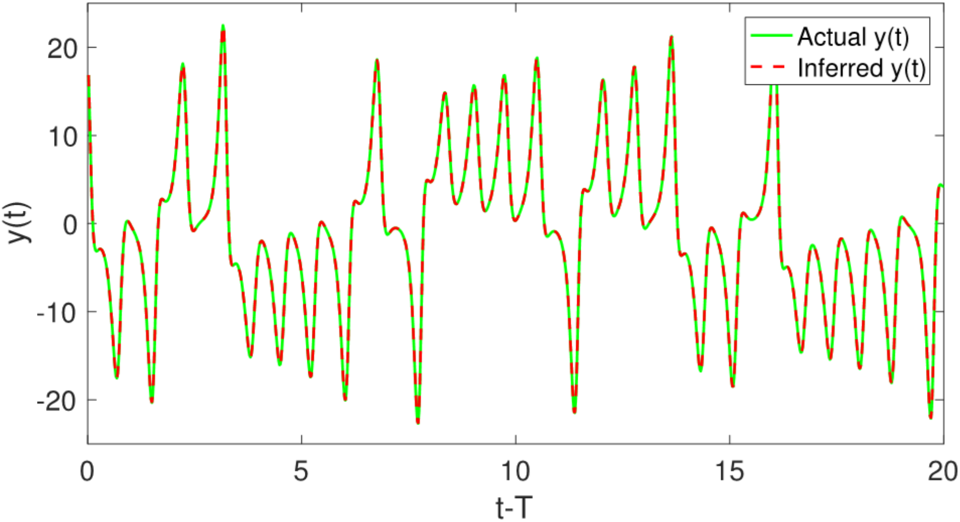
- For our analysis/inference model, we use $x(t)$ as input during both training and operation. We train the reservoir to output approximations to $y(t)$ and $z(t)$. Results shown on next three slides.

- For our forecast model, we input $x(t)$, $y(t)$, and $z(t)$ during training. We train the model output to approximate the input. After training, the model runs autonomously with feedback replacing input. Results shown after next three slides.

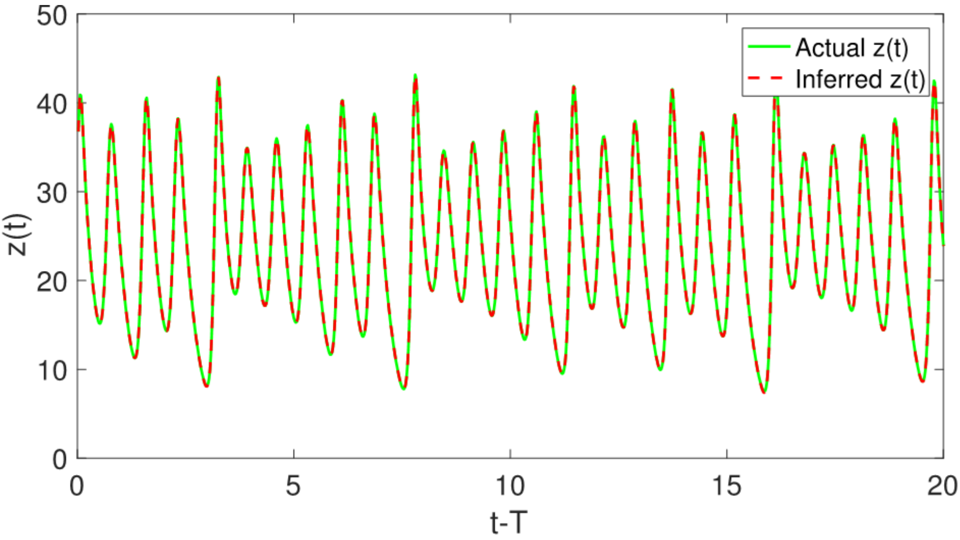
# Reservoir input $x(t)$ for $t > T$



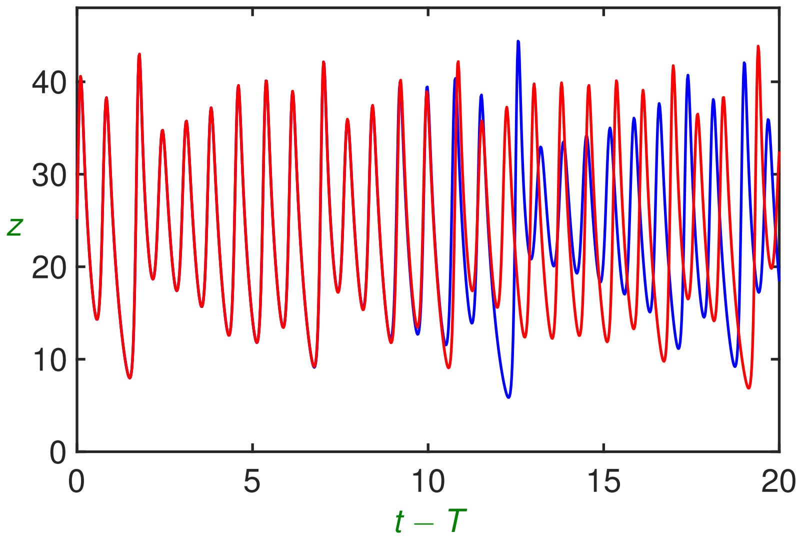
# Inferred $y(t)$ for $t > T$



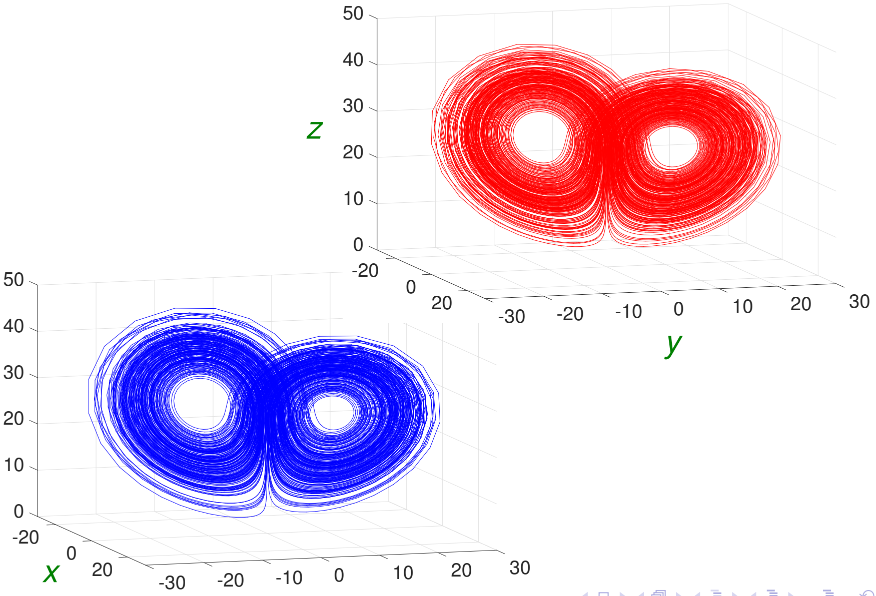
# Inferred $z(t)$ for $t > T$



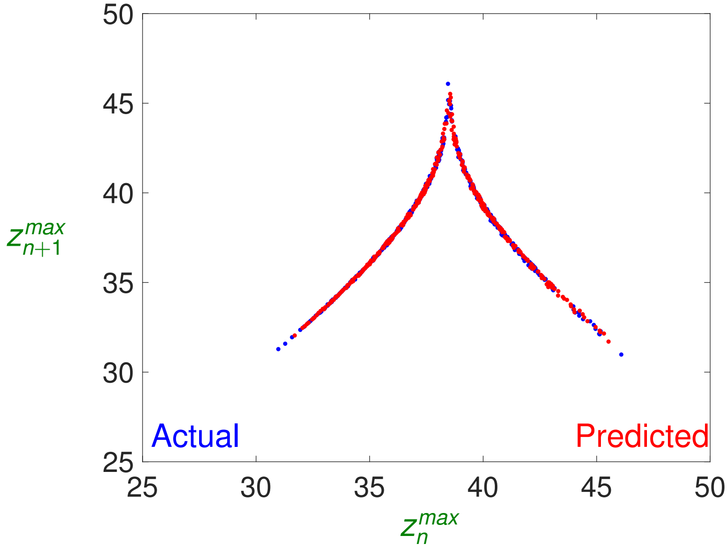
# Actual and Predicted $z(t)$



# Actual and Predicted Attractors



# Poincaré Section



# Concluding Remarks

- Reservoir forecasting is simple to implement and capable of learning the dynamics behind a low-dimensional chaotic time series from a modest amount of data.
- The size of the reservoir must be large compared to the dimensionality of the dynamics to be modeled; however, large reservoirs can be implemented in hardware (e.g., FPGAs).
- We are working on performing the tasks described in this talk for high-dimensional, spatially extended systems using a collection of local reservoirs running in parallel.