LCTp2-1L
Late-News Paper

Circuit Simulation Model of Liquid Crystal Capacitor using Reservoir Computing Approach

Makoto Watanabe[1,4], Kiyoshi Kotani[2], and Yasuhiko Jimbo[3]

watanabe@neuron.t.u-tokyo.ac.jp
[1] Graduate School of Engineering, The University of Tokyo, 153-8904, Japan
[2] Research Center for Advanced Science and Technology, The University of Tokyo, 153-8904, Japan
[3] School of Engineering, The University of Tokyo, 113-8656, Japan
[4] Silvaco Japan Co., Ltd., 220-8136, Japan
Keywords: Machine learning, Reservoir computing, Liquid crystal display, Circuit simulation, Verilog-A

ABSTRACT

Non-linear and dynamic liquid crystal capacitors have been implemented into a circuit simulator using a machine learning framework called reservoir computing handling time series data. Sufficient accuracy has been obtained at time steps determined by the circuit simulator, which is different from those in the training phase of the model.

1 Introduction

Liquid crystal displays (LCDs) are already pervasive and indispensable in our daily lives. The performance of LCDs for augmented reality (AR), virtual reality (VR), and in-vehicle applications is expected to be further improved. For this reason, new materials and cell structures are being actively developed. The ways to improve performance by driving methods have also been proposed. Liquid crystal capacitors used in circuit design have non-linear and dynamic characteristics and are generally approximated through appropriate macro models [1]. However, when the liquid crystal cell includes new phenomena, the macro model must be updated. The development of macro models is typically time-consuming, and inventions are sometimes required.

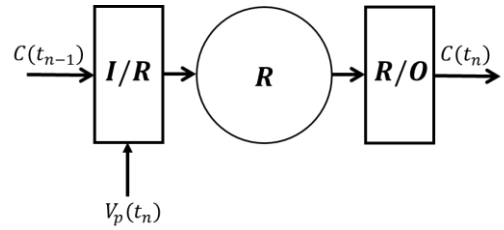
The rapid development of machine learning methods in recent years has enabled us to predict the future from big data. Before we have prototyping devices, the results of numerical calculations such as a finite differential method (FDM) can be machine-learned to create models for circuit simulations. After prototyping, measured data can be machine-learned. In the past, we achieved a speed-up of liquid crystal simulation by machine learning the transient change of the liquid crystal's director distribution using a reservoir computing approach [2]. However, this method assumed that the time step was always the same during the learning and prediction process. Therefore, it is not suitable for cases where the time step is arbitrarily determined by the circuit simulator's operation. Furthermore, generally in circuit designs, there is no need to know the distribution of directors and potentials in the cell. Therefore, there is room for further faster simulations.

In this paper, we have adopted a nonlinear dynamical equation in continuous time [3] for updating reservoir states, allowing the prediction at arbitrary time steps as determined by the circuit simulator. Furthermore, the capacitance, macroscopic information, has been directly learned and the microscopic information such as the director and the potential map has been discarded to achieve a further speed-up. We implemented the model into a commercial circuit simulator by using Verilog-A language and verified that it worked well for the display design.

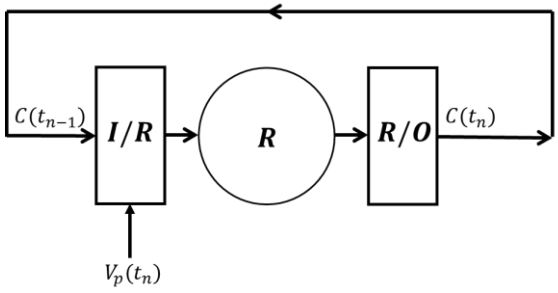
2 Methods

2.1 Reservoir Computing Model

Figure 1 shows a schematic of the reservoir computing method for a liquid crystal capacitor, which is a non-autonomous system in which a time-dependent voltage is applied from the outside [2]. I/R and R/O denote the input-to-reservoir and reservoir-to-output couplers, respectively. "**R**" denotes the reservoir which composes of D_r-dimensional vector $\boldsymbol{r}(t_n)$. In the training phase as shown in Fig. 1(a), I/R first receives the capacitance value for the previous time $C(t_{n-1})$ and an external voltage for the current time step $V_p(t_n)$ in the training data.



(a) Training phase



(b) Prediction phase

Fig. 1 Reservoir configuration in (a) training phase (b) prediction phase

Next, the value of the state in the reservoir $\boldsymbol{r}(t_n)$ is updated with the following nonlinear dynamical equations in continuous time [3].

$$\dot{\boldsymbol{r}}(t) = F(\boldsymbol{r}(t))$$
$$F\big(\boldsymbol{r}(t)\big) = -a\boldsymbol{r}(t) + a\tanh\big(\boldsymbol{W_{res}}\boldsymbol{r}(t) + \boldsymbol{W_{in}}\boldsymbol{u_{in}}(t)\big), \quad (1)$$

where "a" is the leaking rate, which can be regarded as the speed of the reservoir update, $\boldsymbol{r}(t)$ is the D_r-dimensional vector representing a reservoir $\boldsymbol{r}(t) = \big[r_1(t), r_2(t), \cdots, r_{D_r}(t)\big]^T$ and initialized as $\boldsymbol{r}(0) = \boldsymbol{0}$. $\boldsymbol{u_{in}}(t)$ is the input state vector. The elements of $\boldsymbol{W_{in}}$ are chosen randomly from a uniform distribution in $[-0.5, +0.5]$, and the elements of $\boldsymbol{W}_{res}$ are chosen randomly from a uniform distribution in $[-\sigma, +\sigma]$, where $\sigma > 0$ is adjusted so that the maximum absolute value of the eigenvalue of $\boldsymbol{W}_{res}$ is ρ. The quantity ρ is referred to as the spectral radius of the reservoir.

In our case, since we deal with discrete time series data, we discretize Eq. (1) using the fourth order Runge-Kutta method as

$$\boldsymbol{r}(t+h) = \boldsymbol{r}(t) + (\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4})/6 \quad (2)$$
$$\boldsymbol{k_1} = h\,F\big(\boldsymbol{r}(t)\big)$$
$$\boldsymbol{k_2} = h\,F(\boldsymbol{r}(t) + 0.5\boldsymbol{k_1})$$
$$\boldsymbol{k_3} = h\,F(\boldsymbol{r}(t) + 0.5\boldsymbol{k_2})$$
$$\boldsymbol{k_4} = h\,F(\boldsymbol{r}(t) + \boldsymbol{k_3})$$
$$h = \Delta t/\Delta T,$$

where h is the dimensionless time step and is the ratio of the actual time step Δt to the representative time step ΔT. The other variables, capacitance $C(t_n)$ and voltage $V_p(t_n)$ used in the learning and prediction process are also nondimensionalized and normalized as $C(t_n) \times 0.5/max|C(t_n)|$ and $V_p(t_n) \times 0.5/max|V_p(t_n)|$ so that they were distributed between −0.5 and +0.5 as required by the activation function (tanh) to promote efficient training in machine learning.

During the training time, the values of the reservoir nodes $\boldsymbol{r}(t_n)$ for each time step are stored as

$$\boldsymbol{R} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \boldsymbol{u_{in}}(t_2) & \boldsymbol{u_{in}}(t_3) & \boldsymbol{u_{in}}(t_4) & \cdots & \boldsymbol{u_{in}}(t_T) \\ \boldsymbol{r}(t_2) & \boldsymbol{r}(t_3) & \boldsymbol{r}(t_4) & \cdots & \boldsymbol{r}(t_T) \end{bmatrix}, \quad (3)$$

where t_T is the last time step in the training time, $\boldsymbol{u_{in}}(t_n)$ is the input state vector as $\boldsymbol{u_{in}}(t_n) = \begin{bmatrix} V_p(t_n) \\ C(t_{n-1}) \end{bmatrix}$.

The objective of the training process, the finding of the components of $\boldsymbol{W}_{out}$, is achieved by computing as,

$$\boldsymbol{W}_{out} = \boldsymbol{Y}\boldsymbol{R}^{\boldsymbol{T}}(\boldsymbol{R}\boldsymbol{R}^{\boldsymbol{T}} + \lambda\boldsymbol{I}), \quad (4)$$

so that the output $C(t_n)$ from R/O is asymptotic to the capacitances in the training data. In Eq. (4), $\boldsymbol{Y}$ is the training data that shifts one time step to the future as

$$\boldsymbol{Y} = [C(t_2) \quad C(t_3) \quad C(t_4) \quad \cdots \quad C(t_T)], \quad (5)$$

where λ is the regularization coefficient in Ridge regression to avoid overfitting, and $\boldsymbol{I}$ is an identity matrix.

In the prediction phase as shown in Fig. 1(b), first, the same as in the training phase, I/R receives the capacitance value for the previous time $C(t_{n-1})$ and an external voltage for the current time $V_p(t_n)$. Next, the values of the state in the reservoir $\boldsymbol{r}(t_n)$ are updated as in Eq. (2), where $\boldsymbol{r}(t_n)$ is initialized as $\boldsymbol{r}(t_1) = \boldsymbol{0}$. The output $C(t_n)$ from R/O is obtained by using $\boldsymbol{W_{out}}$ as

$$C(t_n) = \boldsymbol{W_{out}}\begin{bmatrix} 1 \\ \boldsymbol{u_{in}}(t_n) \\ \boldsymbol{r}(t_n) \end{bmatrix}. \quad (6)$$

The output $C(t_n)$ is returned to the input state vector for the next time step $\boldsymbol{u_{in}}(t_{n+1})$.

2.2 Training Data

As training data, we used time-series data obtained from numerical simulations using FDM. The Fringe Field Switching (FFS) cell, currently the most popular type of cell in the market, was used as the motif for this study. The structure used to obtain the data for training is shown in Fig. 2. The boundary conditions used in the calculations were periodic boundary conditions on the left and right, the Neumann condition above the glass substrate, and the Dirichlet condition on the electrodes. The liquid crystal capacitance $\boldsymbol{C_{LC}}$ was obtained from Eq. (7) using the electrostatic energy within the domain $\boldsymbol{S}$ calculated from the internal electric field $\boldsymbol{E}$ and electric flux density $\boldsymbol{D}$. $\boldsymbol{E}$ and $\boldsymbol{D}$ were calculated by applying 1.0 V between the pixel and common electrodes after the orientations of liquid crystal were determined at each time.

$$\boldsymbol{C_{LC}} = \int_{\boldsymbol{S}} \boldsymbol{E}\cdot\boldsymbol{D}\,\boldsymbol{dS} \quad (7)$$

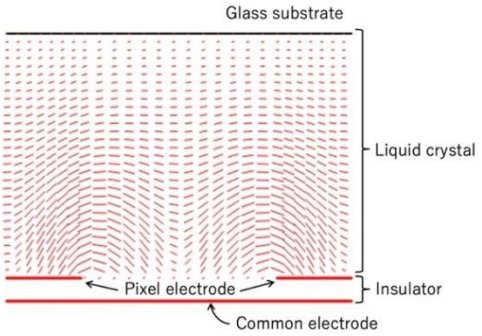


Fig. 2 Structure of a liquid crystal cell whose capacitance was calculated by FDM to obtain training data

In getting training data, the voltage for the common electrode was set to 0 V. The voltage for the pixel electrode was initially set to 3.0 V which assumed the maximum driving voltage. After that, the amplitude of voltages was changed randomly every 16.7 ms according to a uniform random number distribution from 0.5 V to 3.0 V. The polarity of voltages was reversed every 16.7 ms. The total analysis time was 8.35 s (=500 frames) and the input data to the machine learning model were sampled from the data obtained by FDM every 500 μs.

2.3 Implementation to Circuit Simulator

We implemented our machine learning model into a commercial simulator SmartSpice [4] with Verilog-A language (Version 2.3.1 [5]).

2.4 Index for Model Evaluation

For the evaluation index of our model, we used a root mean square error normalized with the average of reference data shown as

$$RMS\ error\ (\%) = \frac{\sqrt{\sum_{k=1}^{m} \frac{(C_k^{ML} - C_k^{FDM})^2}{m}}}{\sum_{k=1}^{m} \frac{C_k^{FDM}}{m}}, \qquad (8)$$

where m indicates the total number of data, C_k^{ML} and C_k^{FDM} are the capacitance from machine learning and FDM, respectively. Since the reservoir computing approach uses a probability distribution, we calculated 10 times with different random number seeds to check the robustness of the model. RMS errors are shown with error bars indicating "average ± standard deviation."

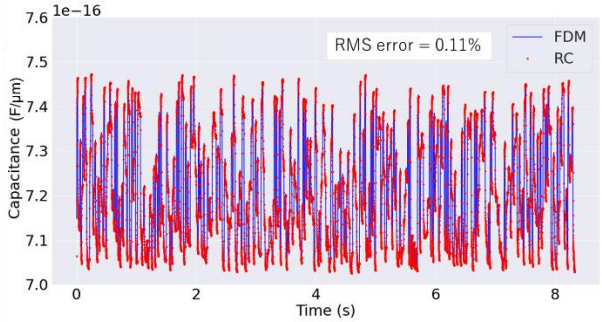
3 Results

3.1 Model Evaluation

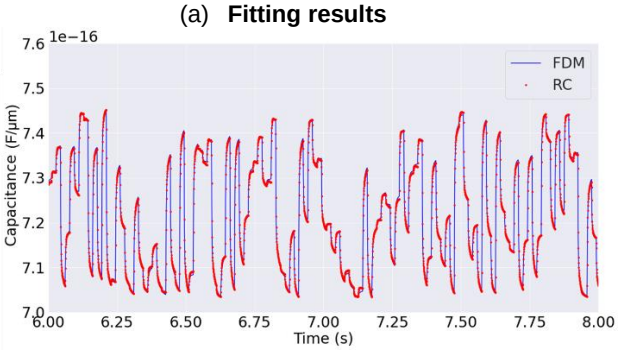
Here we show the abilities of our machine learning model developed by the reservoir computing approach.

Training Phase

Figure 3 shows the results of verifying how well the reservoir computing model ("RC") can reproduce the training data ("FDM"). The results from the reservoir computing model and the FDM model almost overlapped, indicating that the fitting accuracy was good enough. The hyperparameters that composed the reservoir computing framework were $D_r = 20$, $\rho = 0.4$, $a = 0.2$, $\lambda = 1.0 \times 10^{-4}$. These values of hyperparameters were obtained by trial and error, which gave a relatively good accuracy within a reasonable time.



(a) **Fitting results**



(b) Magnification from 6 s to 8 s

Fig. 3 Reproducibility of training data with a trained W_{out}

Prediction Phase

We evaluated the prediction ability of our model with the weights of outputs W_{out} obtained in the training phase. W_{in}, W_{res}, and hyperparameters were the same as those given in the training phase. A waveform had a polarity reversing every 16.7 ms and an amplitude randomly determined from 0.5 V to 3.0 V same as in the training phase. The test time was 1.67 s (=100 frames). The random number seeds for generating external voltages were set to different values from the ones for the training data. Figure 4 shows the prediction accuracy when the time step (Δt in Eq. (2)) during the prediction phase was changed. We found that the discrepancy from FDM was small enough not to cause problems in design works. Both the average and standard deviation of RMS error were the smallest in the case of 500 μs for the time step which was the same as in the training phase. As representatives, the prediction results for the time steps of 500 μs and 500 ns are shown in Fig. 5. These results indicated that the reservoir computing model can be a highly accurate predictor.

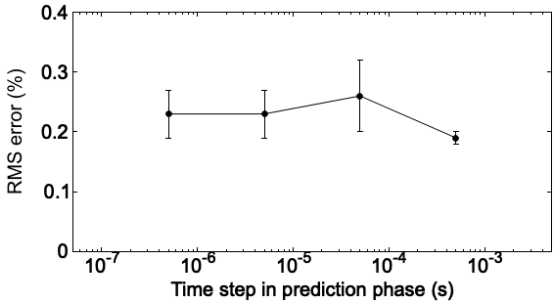
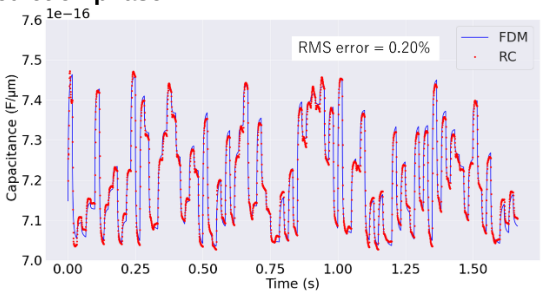
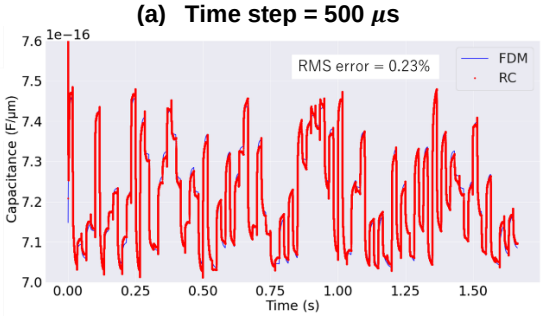


Fig. 4 RMS errors as a function of time step in the prediction phase



(a) Time step = 500 μs

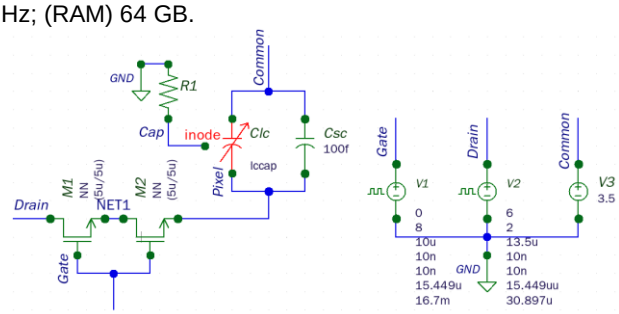


(b) Time step = 500 ns

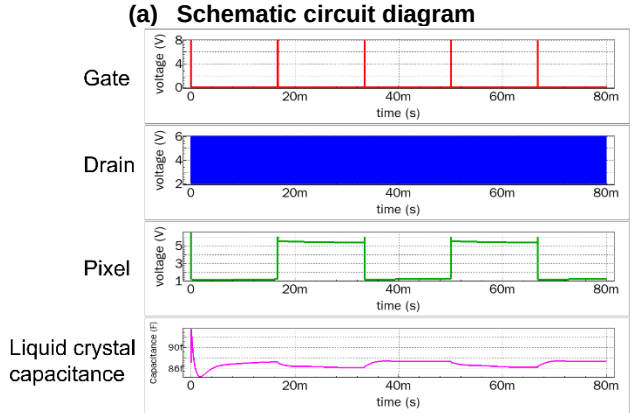
Fig. 5 Prediction ability with a trained reservoir computing model

3.2 Circuit Simulation

The electrical behavior of LCD pixels was simulated using the reservoir computing model. The component highlighted in red in the schematic in Fig. 6(a) is the machine learning based capacitor developed in this study. The third terminal, "inode," was added to monitor the ever-changing capacitance values on the circuit simulator. The maximum time step that can be set in most commercial circuit simulators was set to 500 μs, which is the same time step in the training phase. This makes it sufficient to assume that the time step given by the circuit simulator to the LCD capacitor is only the interpolated time steps investigated in the previous section. Fig. 6(b) shows the results of a circuit simulation. It shows that the value of the liquid crystal capacitance changes according to the voltage between the pixel node and the common node. The voltage of the common node was 3.5 V (DC). The time required for this calculation was 10.51 s. When the macro model was replaced with the built-in static capacity model, the time required was 10.46 s, indicating that the machine learning based model has little effect on the calculation time. The specifications of the machine we used were as follows: (CPU) AMD Ryzen Threadripper 1950x, 1888.625 MHz; (RAM) 64 GB.



(a) Schematic circuit diagram



(b) Circuit simulation result

Fig. 6 Circuit simulation result using liquid crystal capacitor developed by the machine learning method

4 Discussion

Figure 4 shows that the high prediction accuracy was obtained with good reproducibility when the prediction was made at the same time step as that used during the training phase. This could be due to the inclusion of errors caused by interpolation using the Runge-Kutta method when forecasting at other time steps. However, even when interpolation was used, sufficient accuracy was ensured, including the variation caused by the randomness in the framework of the reservoir computing.

As seen in Fig. 5(b), the accuracy was low at the beginning of the prediction process. In the result of the circuit simulation shown in Fig. 6(b), unexpected behavior was observed at the beginning of the simulation. This is because the value of the reservoir was artificially set to zero at the initial time and its influence remains for the time being. When using this model in practical use, the results for some time after the simulation starts should be ignored to obtain reliable results.

5 Conclusions

We have developed a machine learning model of nonlinear and dynamic liquid crystal capacitors using a reservoir computer approach. Since the voltages and time steps given by circuit simulators are arbitrary, we investigated the response of the model to these variables and found it to be sufficiently accurate for implementation in circuit simulators. We have implemented the liquid crystal capacitance model into a circuit simulator using Verilog-A language and showed that it can be used in practical design. In this study, numerical simulation results by FDM were used as training data for machine learning, but measurement results can also be used. This feature is useful in that it allows circuit simulations that reflect device behavior even when the internal physics model is not yet well understood. In principle, this approach can be widely applied to other fields than liquid crystal cells.

6 Acknowledgments

This paper was supported in part by KAKENHI (grant 18H04122, 22K19785) and Tateisi Science and Technology Foundation, and also supported through the activities of VDEC, The University of Tokyo, in collaboration with SILVACO Japan Co., Ltd.

References

[1] M. Watanabe, K. Ishihara, T. Tsuruma, Y. Iguchi, Y. Nakajima, and Y. Maki, “Macro-modeling of Liquid Crystal Cell with VerilogA,” 2007 IEEE Int. Behavioral Modeling and Simulation Workshop, 2007, p. 132.

[2] M. Watanabe, K. Kotani, and Y. Jimbo, “High-speed liquid crystal display simulation using parallel reservoir computing approach,” JJAP (2022).

[3] A. Griffith, A. Pomerance, and D. J. Gauthier, “Forecasting chaotic systems with very low connectivity reservoir computers,” Chaos 29(12), 123108 (2019).

[4] SmartSpice User Manual (Silvaco, Inc. 2022).

[5] Verilog-AMS Language Reference Manual (Accellera, 2009).