PROCEEDINGS OF THE 30th European Symposium on Computer Aided Process Engineering (ESCAPE30), May 24-27, 2020, Milano, Italy

Neural Ordinary Differential Equations-based Explainable Deep Learning for Process Modeling

Michael R. Wartmann[a], B. Erik Ydstie[b]

[a]*Nouryon B.V., Digital Technology, Christian Neefestraat 2, 1077 WW Amsterdam, NL*
[b]*Carnegie Mellon University, 5000 Forbes Avenue, 15213 Pittsburgh, USA*
Corresponding author e-mail: michael.wartmann@nouryon.com

Abstract

Most recent advances in the machine learning domain pose the challenge of how to naturally integrate new data-driven methods with classical process models and control. We propose a process modeling framework enabling integration of data-driven algorithms through consistent topological properties and conservation of extensive quantities. Interconnections among process network units are represented through connectivity matrices and network graphs. The basic requirement is that the flow conditions can be expressed in terms of conic sector (passivity) conditions. Our formalism allows integration of fundamental conservation properties from topology with learned dynamic relations from data through sparse deep neural networks.

We demonstrate in a practical example of a simple inventory control system how to integrate the basic topology of a process with a neural network ordinary differential equation model. The system specific constitutive equations are left undescribed and learned by the deep neural network using the adjoint method in combination with an adaptive ODE solver from synthetic time-series data. The resulting neural network forms a state space model for use in a model predictive control algorithm.

Keywords: neural ordinary differential equations, network theory, deep learning, process networks, process modelling

1. Introduction

Modeling process systems poses the challenge of deriving a meaningful mathematical representation capturing the fundamental laws of nature while representing the systems actual real-world behavior within its given system context. Intelligent use of data in the process of modeling complex systems provides the context while model structure and general behavior is derived from first principles knowledge of the system.

A typical challenge within the context of process systems is that available data of the process is limited for the particular case at hand (Venkatasubramanian 2018). Individual systems differ significantly from each other such that data of one system cannot be simply transferred or combined with data from similar other systems. As such, the need arises to model process systems in structurally consistent ways and take maximal advantage of the knowledge derived from first principles relationships (Ydstie 2002). While machine or deep learning algorithms are suitable to extract even very complex non-linear behavior, in the context of process systems, simple relationships such as fundamental mass, energy and component balances have to be learned ab initio, i.e., the data itself has to provide the context for those relationships. The available data should be used to provide information about the dynamics of the system beyond those fundamental relationships and be extracted through statistical methods or machine learning.

To provide the most optimal starting point for any machine learning based model in the context of process systems, the structure of the material and energy flow of the process

system would need to be embedded in the model itself. In a deep learning context for example, the algorithm simply searches for the best possible fit of the given neural network structure to the input and output data provided. As a result, deep learning models have faced the challenge of explainability (Doran et al 2017), i.e., parameters and structure of the resulting neural network cannot be explained or related to the fundamental laws of nature.
A relevant component of using deep learning-based methods is therefore to understand how connections between the subunits of a process network lead to complex system behavior and how the connections can be built into a deep learning-based model, i.e., a neural network.
For the integration of data-driven methods, we provide an organizational framework for process systems using ideas from network theory in this paper.

2. Network theory and process systems

Process networks are written as a collection of interconnected sub-systems

$$\dot{x_I} = F(x_i) + \sum_{j=0, j\neq i}^{n} G(u_i, x_i, x_j), \qquad i = 0, \ldots, n \tag{1}$$

$$y_i = H(x_i) \tag{2}$$

x_i is the state of subsystem i and $x_i(0)$ is the initial condition. The function F describes the unforced motion of the system, the function G describes how the system is connected with other sub-systems, and the output function H relates the state of the system to the measurement functions y_i. The functions u_i represent the manipulated variables. The functions F, G, H are all differentiable at least once. The state of the entire network is given by the vector $x = (x_0^T, x_1^T, \ldots, x_n^T)^T$.
Subscript zero refers to the reference (exo-) system. Often, we are not interested in the dynamics of the exo-system, or more likely, it is too complex to model. The process system is modeled as the reduced system without the reference sub-system. Its state is given by the vector $x = (x_1^T, \ldots, x_n^T)^T$. The interactions with the exo-system are then established through the boundary conditions.
The network form, as illustrated in Figure 1 is convenient when we model systems with a graph structure.

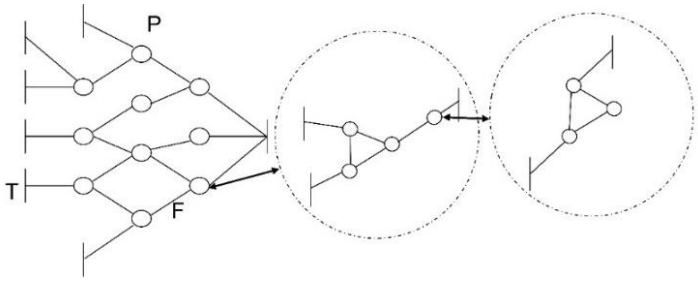

Figure 1: Graphical network representation: Topological structure of a network consisting of nodes, terminals, and flows. Nodes can contain subgraphs and give rise to a hierarchical multi-scale structure.

In such systems the interactions between the sub-systems depend on the state of the sub-system itself and the state of its immediate neighbors. Not all dynamical systems can be

decomposed in this fashion. However, many large-scale systems have sparse interconnections and they can be modeled compactly as networks of sub-systems with interconnections. It is also easy to see that many physical systems, especially those that satisfy the principle of local action, can be decomposed in the manner shown in Eq. (1).

By a graph $\mathbf{G}$ we mean a finite set $\upsilon(\mathbf{G}) = (\upsilon_1, \ldots l, \upsilon_{nP})$, whose elements are called **nodes**, together with the set $\varepsilon(\mathbf{G})$ whose elements are called **branches**. A branch is therefore an ordered pair of distinct nodes.

Definition 1. A network of nodes P_i, $i = 1,\ldots,n_p,n_p+1,\ldots,n_v$ consisting of nodes and terminals interconnected through branches E_i, $i = 1,\ldots,n_f$ with topology defined by the graph set $\mathbf{G} = (E, P)$.

The system (1) is called a *process network* if its interconnection structure is described by a directed graph and we have

(1) **1st law:** There is an inventory E (the energy) satisfying the conservation property
(2) **2nd law:** There is an inventory S (the entropy) satisfying the Clausius-Planck property

Definition 2. The $n_t \times n_f$ matrix $\mathbf{A_a}$ is called incidence matrix for the matrix elements a_{ij} being

$$a_{ij} = \begin{cases} 1, \text{if flow } j \text{ leaves node } i \\ -1, \text{if flow } j \text{ enters node } i \\ 0, \text{if flow } j \text{ is not incident with node } i \end{cases}$$

One node of the network is set as reference or datum node P_0 representing the exo-system. The $(n_t - 1) \times n_f$ matrix $\mathbf{A}$, where the row that contains the elements a_{0j} of the reference node P_0 is eliminated, is called reduced incidence matrix.

The connections between nodes through branches can be uniquely defined using the incident matrix $\mathbf{A}$. The conservation laws of Eq. 1 can now be written

$$\mathbf{AF} = \mathbf{0} \tag{3}$$

for the node-to-branch incident matrix $\mathbf{A}$, where $\mathbf{F}^T = (\frac{dZ_1}{dt}, \frac{dZ_2}{dt} \ldots, \frac{dZ_t}{dt}, f_{12}, f_{13}, \ldots, f_{n_{t-1,n_t}}, p_1, \ldots, p_t)$. The flows f_{ij} represent connections between two nodes i.e. f_{ij} connects node i to node j, p_i denotes sources or sinks. The direction of the flows is defined according to the directionality established in the graph. We now define a vector $\mathbf{W}$ so that

$$\mathbf{W} = \mathbf{A}^{\boldsymbol{T}}\mathbf{w} \tag{4}$$

where $W_{ij} = w_i - w_j$ are the potential differences across flow connections. The variables $\mathbf{w}$ are conjugate to $\mathbf{Z}$ if they are related via the Legendre transform of a convex potential like the entropy.

Eqs. 3 and 4 only reflect the topological properties of the process system. To complete a full dynamic model, we require functions relating the flow of extensive quantities $\mathbf{F}$ to the potential differences $\mathbf{W}$ and a relationship between the inventories $\mathbf{Z}$ and the potentials $\mathbf{w}$. These relationships often contain non-linearities in classical process models, e.g. kinetic relationships, pressure drop relations, thermodynamic models and chemicals potentials etc. (Wartmann and Ydstie 2009).

3. Neural Ordinary Differential Equations for Process Networks

A new family of deep neural network models have been introduced by Chen et al (2018). Instead of specifying a discrete sequence of hidden layers, in Chen et al. (2018) the derivative is parameterized using a neural network. The deep neural network models allow discretization of a dynamic system in time, in which each hidden layer of the neural network represents a particular state in time. These models are especially powerful, when used in the context of time-series correlated data and allow learning differential equations from data. For process systems, dynamic models for control can be derived by applying a deep neural network if trained from e.g. operational step testing data.
Chen et al. show that for training of continuous-depth neural networks, reverse mode differentiation or backpropagation can best be carried out through an ODE solver method. The well-known adjoint method is suitable to calculate the gradients of a scalar-valued loss function L for neural network weight training when combined with a gradient decent method for backpropagation. From a process network perspective, learning the dynamic behavior of a set of time-series data is essentially equivalent to modeling the right-hand side of a differential equation system as in Eq. 1 through a neural network representation. For a template of a neural network model with trainable weights, it would be ideal to have the process system's topology already represented in the neural networks layers such that the (non-linear) relations between flows **F** and inventories **Z** and the potentials **w** are learned from data while inventory balances and their structure are pre-imposed. As shown in Fig. 3, if potentials **w** of a network are measurable inputs, then continuity of the potentials allows relating them to differentials **W** and flow variables **F** through these relations (e.g. pressure differential drives convective flow). Further, flow variables **F** in the hidden layer relate to the inventory balance and allow calculation of the inventory differences $\Delta\mathbf{Z}$.
To represent the topology of a process network in a neural network model, some of the weights in the incident matrix **A** have to be set to zero resulting in a sparsely connected neural network where no physical connection is present. In the process of training the weights, these zero weights have to be pruned in training reducing the computational effort, preserving the process topology, and resulting in potentially explainable parameters for the remaining weights.

4. An inventory system modeled by a neural ODE

A small dynamic flow and storage example shows how process networks can be modelled using neural ODE's. The network consists of two connected pipelines where each pipeline flows through a cylindrical storage tank with volume open to the atmosphere, see Fig. 2. The resulting linear classical ODE can be derived from the balances around the nodes P_1 and P_2 for Z_1 and Z_2 i.e., $dZ_{1,2}/dt = F_{1,3}\text{-}F_{2,4}$, constant terminal potentials w_{T1} and w_{T2}, the flow constitutive equations $F_i=K_i W_i$ with i=1,..,4, the potential differences W = w1 – w2 as in Eq. (4), and $w_i = C_i Z_i$

$$\begin{aligned} C_1^{-1}\frac{dw_1}{dt} &= -(K_1 - K_3)w_1 + K_1 w_{T1} + K_3 w_{T2} \\ C_2^{-1}\frac{dw_1}{dt} &= -(K_2 - K_4)w_2 + K_2 w_{T1} + K_4 w_{T2} \end{aligned} \tag{5}$$

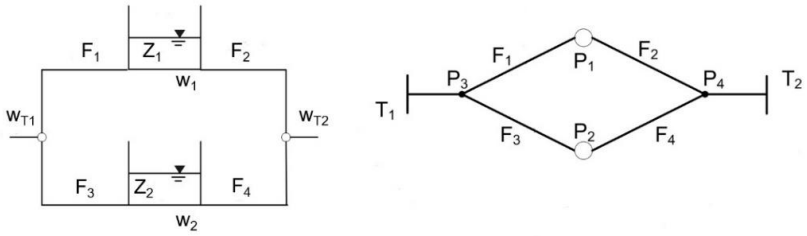

Fig. 2: Graphical network representations: Problem specific representation on the left, a generalized graph representation on the right.

Discretization using the explicit Euler method for demonstration purposes and using ReLU's as activation functions leads to Eq. 6.

$$\boldsymbol{w}^{t+1T} = \boldsymbol{w}^{tT} + \Delta t\ \mathrm{ReLU}(\mathrm{ReLU}(\boldsymbol{w}^{tT}\mathbf{A_K})\mathbf{A_C}^T) \tag{6}$$

where the connectivity matrices $\mathbf{A_K}$ and $\mathbf{A_C}$ are given as in Fig 3.

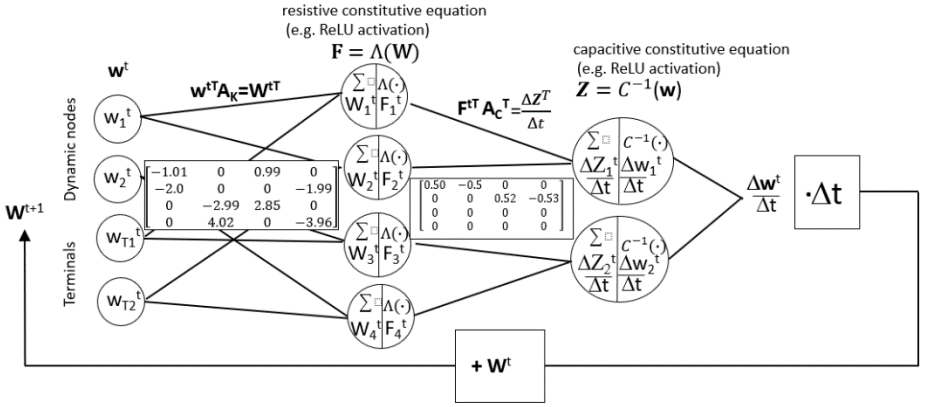

Figure 3: Neural network representation of the process network example, with potentials w in the input layer, flows F at the hidden layer, and potential differences per time step at the output layer. Final weights after training as given in the connectivity matrices are equivalent to the differential equation system from which synthetic data was derived.

The Neural ODE system was trained in Google Co-Lab using cloud GPU's based on synthetic dynamic data with added Gaussian noise (5%) from the corresponding ODE of Eqs. 5 with parameters K_1=1, K_2=2, K_3=3, K_4=4, C_1=C_2=2, and w_{T1}=4, w_{T2}=0. A Pytorch Python-based implementation was chosen with a time discretization using the explicit Euler method and time steps of Δt = 0.02 s. The process topology of the neural ODE was generated by pruning weights of non-incident edges, i.e., setting the corresponding weights to zero in every forward pass iteration during training. An adaptive deep learning stochastic gradient descent algorithm was applied for backpropagation to train the non-zero weights of the neural network. Training batches were selected through randomly choosing time periods within the synthetic data and 1000 training iterations carried out. The training run took T=17 sec and the neural network model matched the original data of the synthetic model where weights match with the original ODE (see Fig. 4) for the final w_1-w_2 time trajectory versus training data.

The resulting neural network simulates the dynamic system even if initial conditions are chosen differently from the original training data, i.e., extrapolates versus the dynamic trajectory of the original data.

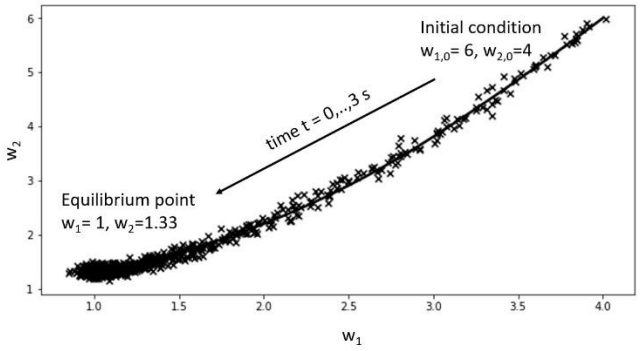

Figure 4: Time trajectory of dynamic potentials w_1-w_2 of inventory system after training the neural network. Synthetic time series data (marks) vs. neural network model (solid line).

5. Conclusions and Outlook

Using a graph theory based neural network representation for a process network allows using the topology of the system to be incorporated into the neural network through the incident matrices between edges and nodes. A sparsely connected neural network with meaningful topology allows preservation of e.g. inventory balances and a consistent potential field. Relationships between extensive quantities such as flows and inventories to intensive quantities, i.e., potentials are learned from data through training the remaining non-zero weights in the neural network through pruning. By applying classical ODE solvers and discretizing in time via e.g. the explicit Euler method in combination with adjoint equations allows determining gradients through backpropagation with a deep learning stochastic gradient descent method. Neural ODE models for process systems and networks have the potential to be used for model predictive control for which weight parameters are explainable and can be understood and updated when new data becomes available or the fundamental process structure is changed.

6. References

R.T.Q. Chen,Y. Rubanova, J. Bettencourt, and D. Duvenaud, 2018, Neural ordinary differential equations. *ArXiv 1806.07366*.

D. Doran, S. Schulz, and T. R. Besold. 2017, What does explainable AI really mean? a new conceptualization of perspectives. *ArXiv 1710.00794*.

M. R. Wartmann and B.E. Ydstie, 2009, Network-based analysis of stability, optimality of process networks. *Proceedings of ADCHEM*, 197–202.

B.E. Ydstie, 2002, New vistas for process control: Integrating physics and communication networks. *AICHE*, 48, 422–426.

V. Venkatasubramanian, 2019, The Promise of Artificial Intelligence in Chemical Engineering: Is It Here, Finally? *AICHE*, 65, 466-478.