

Introduction to Scientific Machine Learning

Jian Tao

jtao@tamu.edu

HPRC Short Course

03/26/2021



Texas A&M Engineering
Experiment Station

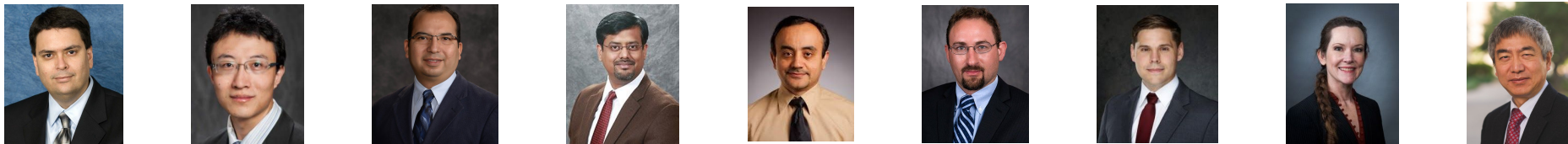


High Performance
Research Computing
DIVISION OF RESEARCH

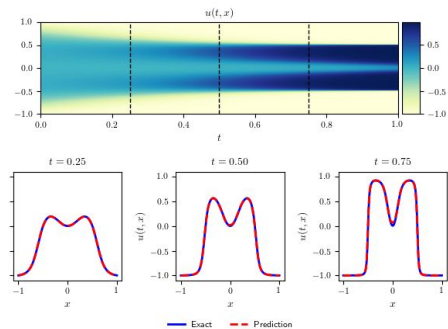


TEXAS A&M
Institute of
Data Science

LAB MEMBERS

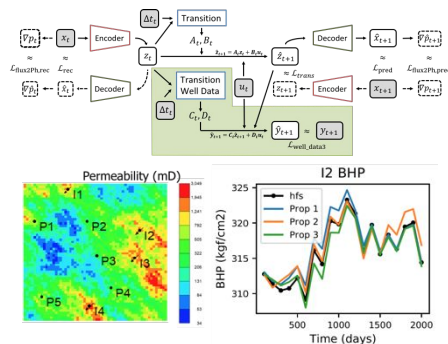


Pilot Project 1 - Microstructure Informatics



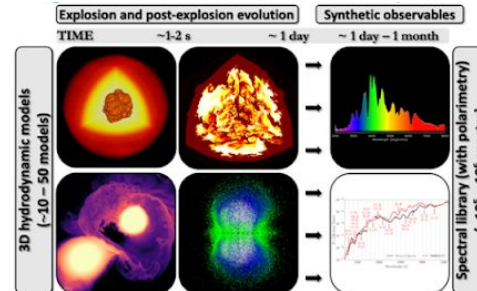
Raymundo Arroyave, Ulisses Braga-Neto, Levi McClenny, Vahid Attari

Pilot Project 2 - Reservoir Simulation



Eduardo Gildin, Ulisses Braga-Neto, Yalchin Efendiev

Pilot Project 3 - Thermonuclear Supernovae



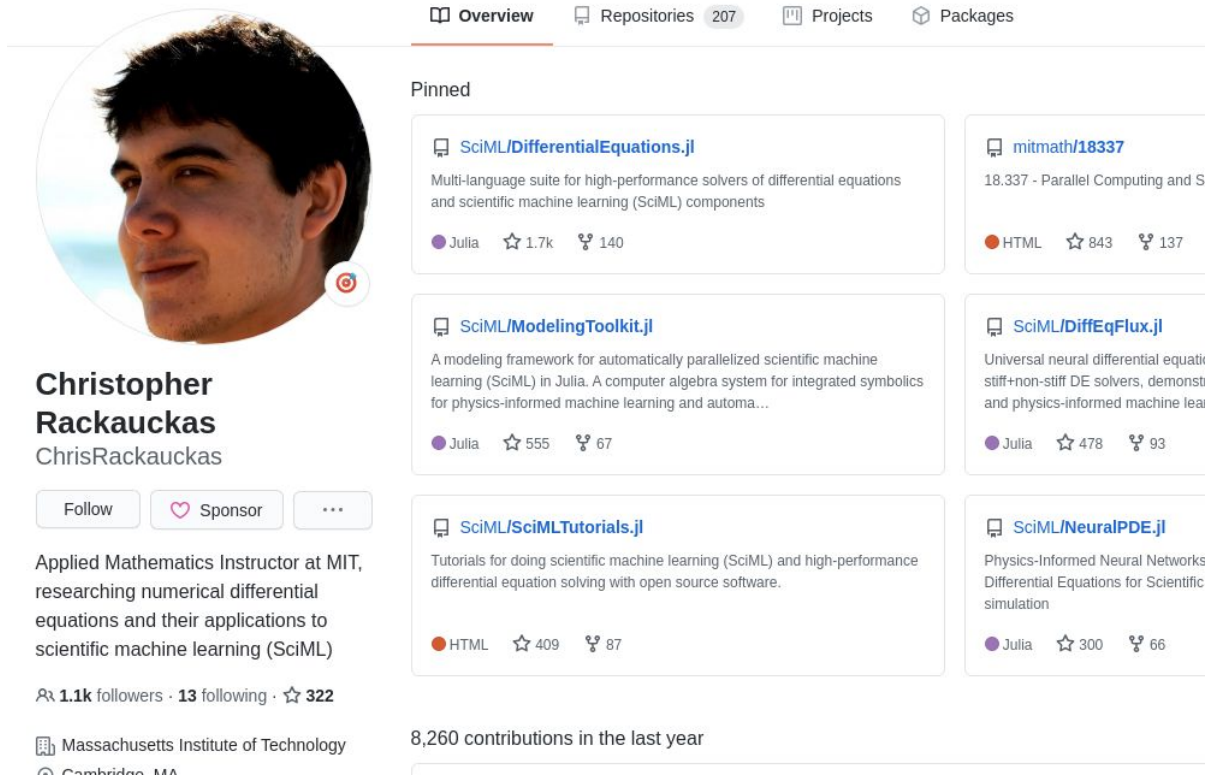
Lifan Wang, Jian Tao, Lisa Perez

TensorDiffEq is a python package built on top of Tensorflow to provide scalable and efficient PINN solvers. TensorDiffEq's primary purpose is for scalable solving of PINNs (inference) and inverse problems (discovery).

Additionally, TensorDiffEq is the only package that fully supports and implements Self-Adaptive PINN solvers and is the only Multi-GPU PINN solution suite that is fully open-source.

Levi McClenny, Ulisses Braga-Neto

Upcoming TAMIDS SciML Lab Talk (April 14)



The image shows a screenshot of a GitHub profile for Christopher Rackauckas. The profile includes a circular profile picture, a bio, and a list of pinned repositories. The bio states: "Applied Mathematics Instructor at MIT, researching numerical differential equations and their applications to scientific machine learning (SciML)". The profile has 1.1k followers and 13 following. The pinned repositories are: SciML/DifferentialEquations.jl (Multi-language suite for high-performance solvers of differential equations and scientific machine learning (SciML) components), mitmath/18337 (18.337 - Parallel Computing and S), SciML/ModelingToolkit.jl (A modeling framework for automatically parallelized scientific machine learning (SciML) in Julia. A computer algebra system for integrated symbolics for physics-informed machine learning and automa...), SciML/DiffEqFlux.jl (Universal neural differential equati stiff+non-stiff DE solvers, demonst and physics-informed machine lea), SciML/SciMLTutorials.jl (Tutorials for doing scientific machine learning (SciML) and high-performance differential equation solving with open source software.), and SciML/NeuralPDE.jl (Physics-Informed Neural Networks Differential Equations for Scientific simulation).

Christopher Rackauckas
ChrisRackauckas

Follow Sponsor ...

Applied Mathematics Instructor at MIT, researching numerical differential equations and their applications to scientific machine learning (SciML)

1.1k followers · 13 following · 322

Massachusetts Institute of Technology
Cambridge, MA

Overview Repositories 207 Projects Packages

Pinned

- SciML/DifferentialEquations.jl**
Multi-language suite for high-performance solvers of differential equations and scientific machine learning (SciML) components
Julia 1.7k 140
- mitmath/18337**
18.337 - Parallel Computing and S
HTML 843 137
- SciML/ModelingToolkit.jl**
A modeling framework for automatically parallelized scientific machine learning (SciML) in Julia. A computer algebra system for integrated symbolics for physics-informed machine learning and automa...
Julia 555 67
- SciML/DiffEqFlux.jl**
Universal neural differential equati stiff+non-stiff DE solvers, demonst and physics-informed machine lea
Julia 478 93
- SciML/SciMLTutorials.jl**
Tutorials for doing scientific machine learning (SciML) and high-performance differential equation solving with open source software.
HTML 409 87
- SciML/NeuralPDE.jl**
Physics-Informed Neural Networks Differential Equations for Scientific simulation
Julia 300 66

8,260 contributions in the last year

<https://github.com/ChrisRackauckas>

Upcoming Hackathon on Material Design with Graph Learning (April 19 - 23)



ASSOCIATE DIRECTOR

[Jian Tao](#)

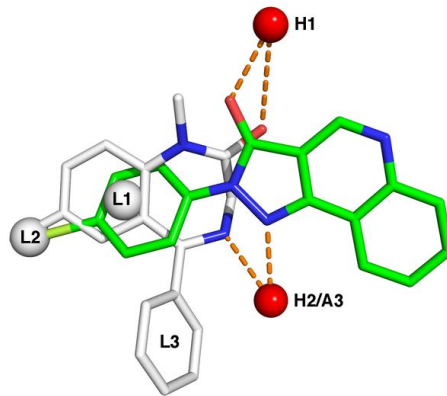
TEES Research Scientist / Computational Scientist / Adjunct Professor
Strategic Initiatives, Texas A&M Engineering Experiment Station
Texas A&M Institute of Data Science
High Performance Research Computing, Texas A&M University

Numerical analysis; workflow management; data science; HPC; scientific computation.

[Raymundo Arroyave](#)

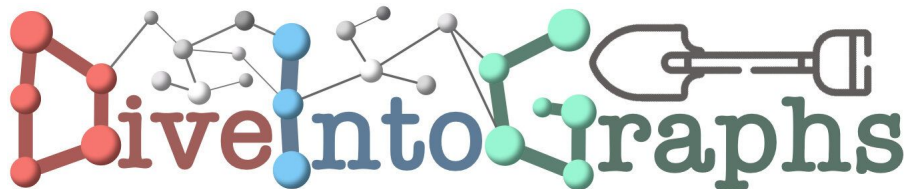
Professor
Department of Materials Science and Engineering

Materials Science, Numerical Methods



One week long Hackathon to explore potential applications of graphical learning in material design.

Please contact jtao@tamu.edu if you are interested.



<https://github.com/divelab/DIG/>



TAMIDS-SciML
DIVISION OF RESEARCH

Upcoming Tutorial on TensorDiffeq (Early May)



[Levi McClenny](#)

Research Assistant

Department of Electrical Engineering

Physics-Informed Deep Learning, Physics-Explainable AI, PINNs, Materials Science



Package Build passing Package Release passing pypi v0.1.6.7 downloads 135/month python 3.6 | 3.7 | 3.8

Efficient and Scalable Physics-Informed Deep Learning

Collocation-based PINN PDE solvers for prediction and discovery methods on top of [Tensorflow 2.X](#) for multi-worker distributed computing.

Use TensorDiffEq if you require:

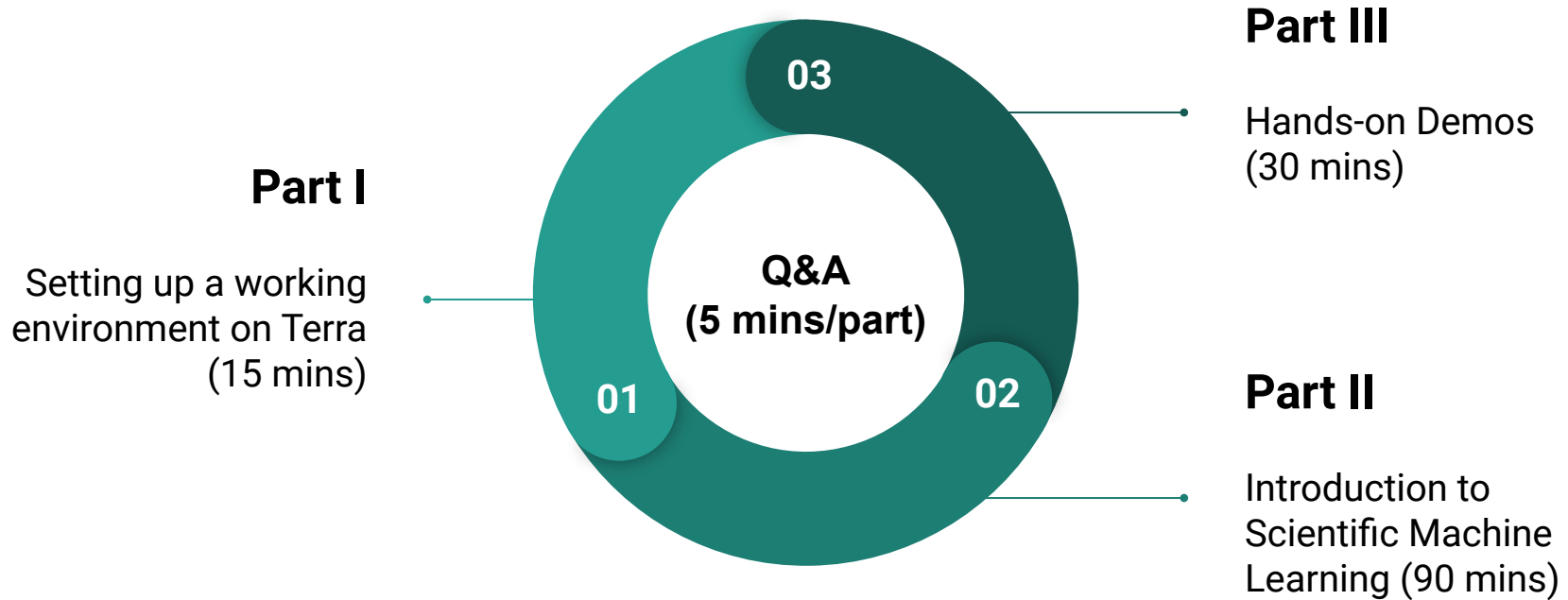
- A meshless PINN solver that can distribute over multiple workers (GPUs) for forward problems (inference) and inverse problems (discovery)
- Scalable domains - Iterated solver construction allows for N-D spatio-temporal support
 - support for N-D spatial domains with no time element is included
- Self-Adaptive Collocation methods for forward and inverse PINNs
- Intuitive user interface allowing for explicit definitions of variable domains, boundary conditions, initial conditions, and strong-form PDEs

<https://github.com/tensordiffeq/TensorDiffEq>

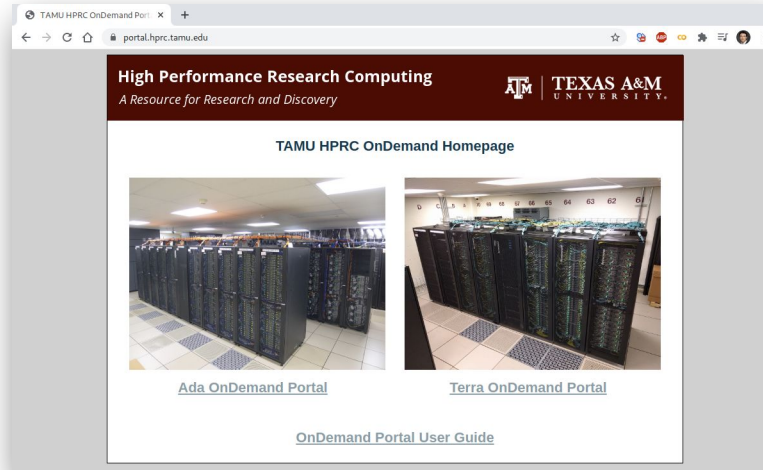


TAMIDS-SciML
DIVISION OF RESEARCH

Introduction to Scientific Machine Learning



Part I. Working Environment



HPRC Portal

Login HPRC Portal (Terra)



TAMU HPRC OnDemand Port. x +

portal.hprc.tamu.edu

High Performance Research Computing
A Resource for Research and Discovery

ATM | TEXAS A&M
UNIVERSITY.

TAMU HPRC OnDemand Homepage



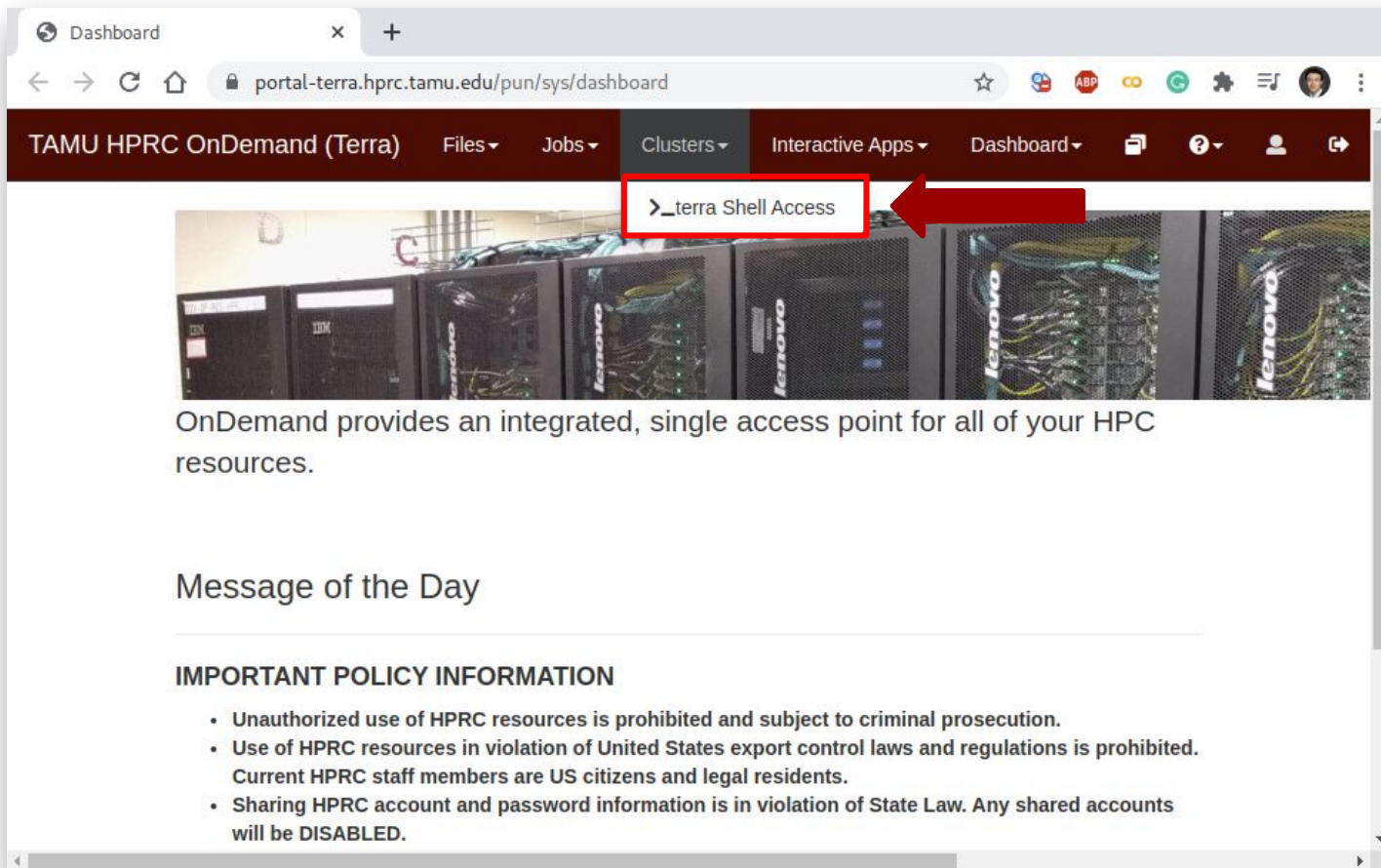
[Ada OnDemand Portal](#)

[Terra OnDemand Portal](#)

[OnDemand Portal User Guide](#)

A red arrow points to the Terra OnDemand Portal link.

Terra Shell Access - I




The screenshot shows a web browser window with the URL `portal-terra.hprc.tamu.edu/pun/sys/dashboard`. The navigation bar includes links for 'TAMU HPRC OnDemand (Terra)', 'Files', 'Jobs', 'Clusters', 'Interactive Apps', and 'Dashboard'. A red box highlights the link '>_terra Shell Access' in the 'Clusters' dropdown menu, with a red arrow pointing to it from the right. Below the navigation bar is a banner image of server racks with the 'lenovo' logo. The main content area contains the text: 'OnDemand provides an integrated, single access point for all of your HPC resources.' Below this is a section titled 'Message of the Day' and a section titled 'IMPORTANT POLICY INFORMATION' with a bulleted list of rules.

Dashboard

portal-terra.hprc.tamu.edu/pun/sys/dashboard

TAMU HPRC OnDemand (Terra) Files Jobs Clusters Interactive Apps Dashboard

>_terra Shell Access



OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.
- Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.
- Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.

Terra Shell Access - II

```
*****
This computer system and the data herein are available only for authorized
purposes by authorized users: use for any other purpose is prohibited and may
result in administrative/disciplinary actions or criminal prosecution against
the user. Usage may be subject to security testing and monitoring to ensure
compliance with the policies of Texas A&M University, Texas A&M University
System, and the State of Texas. There is no expectation of privacy on this
system except as otherwise provided by applicable privacy laws. Users should
refer to Texas A&M University Standard Administrative Procedure 29.01.03.M0.02,
Rules for Responsible Computing, for guidance on the appropriate use of Texas
A&M University information resources.
*****

Password:
Duo two-factor login for jtao

Enter a passcode or select one of the following options:

  1. Duo Push to iPhone (iOS)
  2. Duo Push to iPad (iOS)

Passcode or option (1-2): 1
```

Using Pre-installed Julia Module

Step 1. Find the module to be loaded

```
$ module spider julia
```

```
...
```

```
Julia/0.6.2-intel-2017A-Python-2.7.12-ParMETIS-4.0.3
```

```
Julia/0.6.2-intel-2017A-Python-2.7.12-METIS-5.1.0
```

```
Julia/0.6.2-intel-2017A-Python-2.7.12-noSuiteSparse
```

```
Julia/1.0.5-linux-x86_64
```

```
Julia/1.4.1-linux-x86_64
```

```
Julia/1.5.1-linux-x86_64
```

```
...
```

(You can also use the [web-based interface](#) to find software modules available on HPRC systems. For experienced users, please jump to Step 2.)

Step 2. Load the module

```
$ module load Julia/1.5.1-linux-x86_64
```

Step 3. Start Julia REPL

```
$ julia
```

```
[jtao@terra3 ~]$ module load Julia/1.5.1-linux-x86_64
[jtao@terra3 ~]$ which julia
/sw/eb/sw/Julia/1.5.1-linux-x86_64/bin/julia
[jtao@terra3 ~]$ julia
```



Documentation: <https://docs.julialang.org>

Type "?" for help, "]" for Pkg help.

Version 1.5.1 (2020-08-25)

Official <https://julialang.org/> release

```
julia> █
```

Using Your Own Julia Installation

Step 1. Find the version to be installed

Current stable release: v1.5.4 (March 11, 2021)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

Windows [help]	64-bit (GPG), 64-bit (musl) ^[1] (GPG)	Portable
macOS [help]		
Generic Linux on x86 [help]	64-bit (GPG), 64-bit (musl) ^[1] (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	
Generic FreeBSD on x86 [help]	64-bit (GPG)	
Source	Tarball (GPG)	Tarball with dependencies

Right click and
copy the link.

(You can find different versions of Julia at [Download Julia](#). The latest stable version of Julia is highly recommended.)

Step 2. Download and unzip

Paste the
link here.

```
$ cd $SCRATCH
$ wget https://.../julia-1.5.4-linux-x86_64.tar.gz
$ tar zxvf julia-1.5.4-linux-x86_64.tar.gz
```

Step 3. Start Julia Shell

```
$ cd $SCRATCH/julia-1.5.4/bin; ./julia
```

```
[jtao@terral bin]$ $SCRATCH/julia-1.5.4/bin/julia
┌───┐
│   │
│   │
│   │
│   │
└───┘

Documentation: https://docs.julialang.org
Type "?" for help, "j?" for Pkg help.
Version 1.5.4 (2021-03-11)
Official https://julialang.org/ release

julia> █
```

Install Julia Packages

```
# Set Julia Depot path under $SCRATCH
$export JULIA_DEPOT_PATH=$SCRATCH/.julia

# start Julia
$cd $SCRATCH/julia-1.5.4/bin; ./julia

# type ']' to open Pkg REPL
julia>]
(@v1.5) pkg> add Plots
```

Julia - Quickstart

The julia program starts the interactive **REPL**. You will be immediately switched to the **shell mode** if you type a **semicolon**. A **question mark** will switch you to the **help mode**. The **<TAB>** key can help with autocompletion.

```
julia> versioninfo()  
julia> VERSION
```

Special symbols can be typed with the **escape symbol and <TAB>**, but they might not show properly on the web-based terminal.

```
julia> \sqrt <TAB>  
julia> for i ∈ 1:10 println(i) end #\in <TAB>
```

Part II. Introduction to Scientific Machine Learning (SciML)

**Workshop Report on Basic Research Needs for Scientific Machine Learning:
Core Technologies for Artificial Intelligence**

by Baker, Nathan, et. al.

<https://doi.org/10.2172/1478744>

**Physics-informed neural networks: A deep learning framework for solving forward
and inverse problems involving nonlinear partial differential equations.**

by Raissi, M., Perdikaris, P., & Karniadakis, G. E.

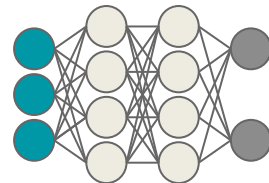
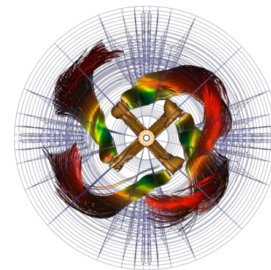
Journal of Computational Physics, 378, 686-707.

<https://doi.org/10.1016/j.jcp.2018.10.045>

SciML Scientific Machine Learning Software

by Chris Rackauckas et. al.

<https://sciml.ai/>

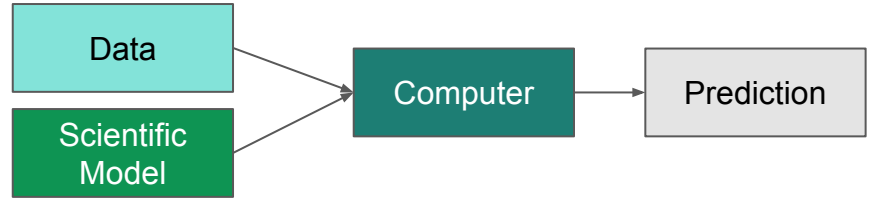


Data-Driven vs Theory-Driven

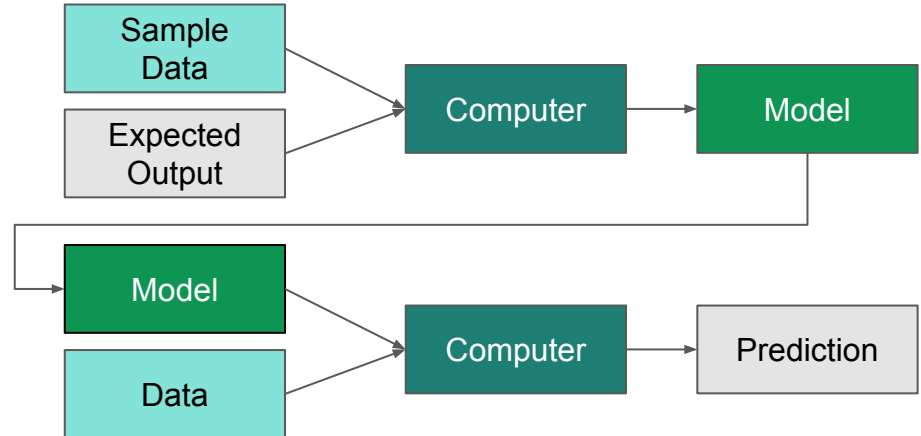
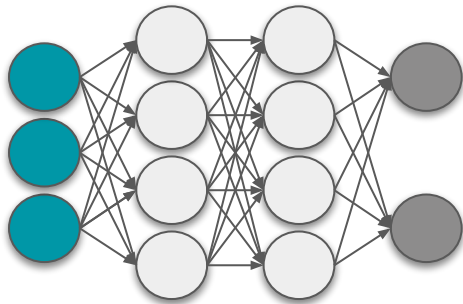
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \gamma \nabla^2 \mathbf{u} + \frac{1}{\rho} \mathbf{F}$$



Theory-Driven Modeling (Numerical Simulation)



Data-Driven Modeling (Supervised Learning)



Balance between Data and Theory

Theory-Driven Model

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \gamma \nabla^2 \mathbf{u} + \frac{1}{\rho} \mathbf{F}$$

Less Data,
More Theory,
Clean,
Unrealistic,
Explainable

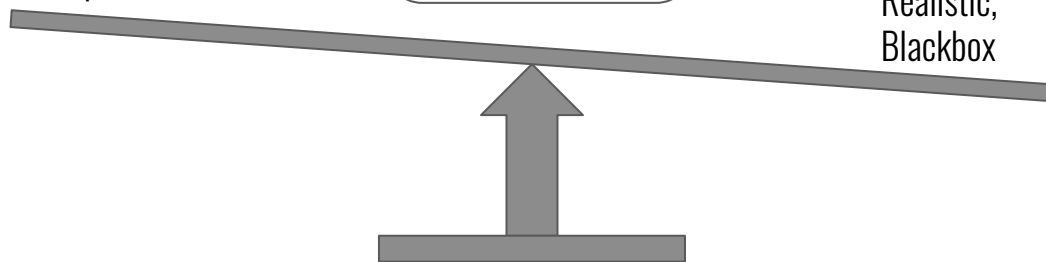
Data-Driven / ML Model

0101010101010
0001011001001



More Data,
Less Theory,
Noisy,
Realistic,
Blackbox

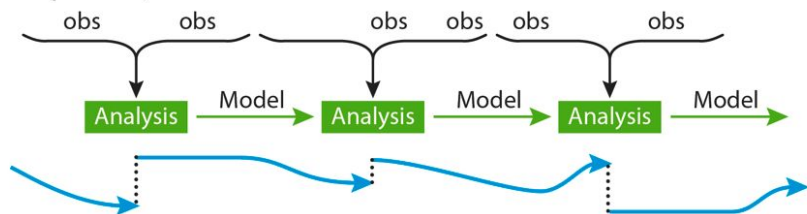
Data
Assimilation



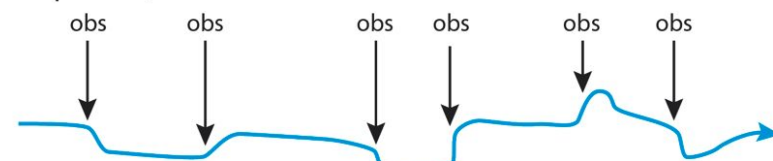
* Data assimilation is somewhere in between but not necessarily balanced.

Middle Ground - Data Assimilation

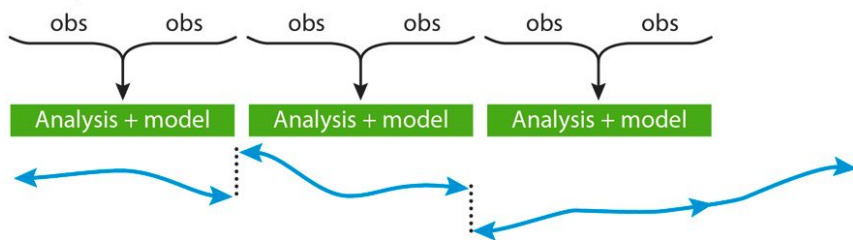
Sequential, intermittent assimilation



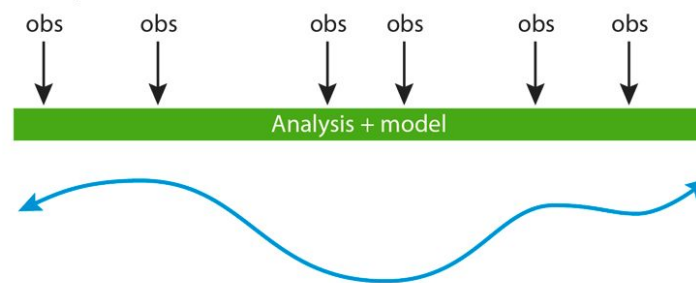
Sequential, continuous assimilation



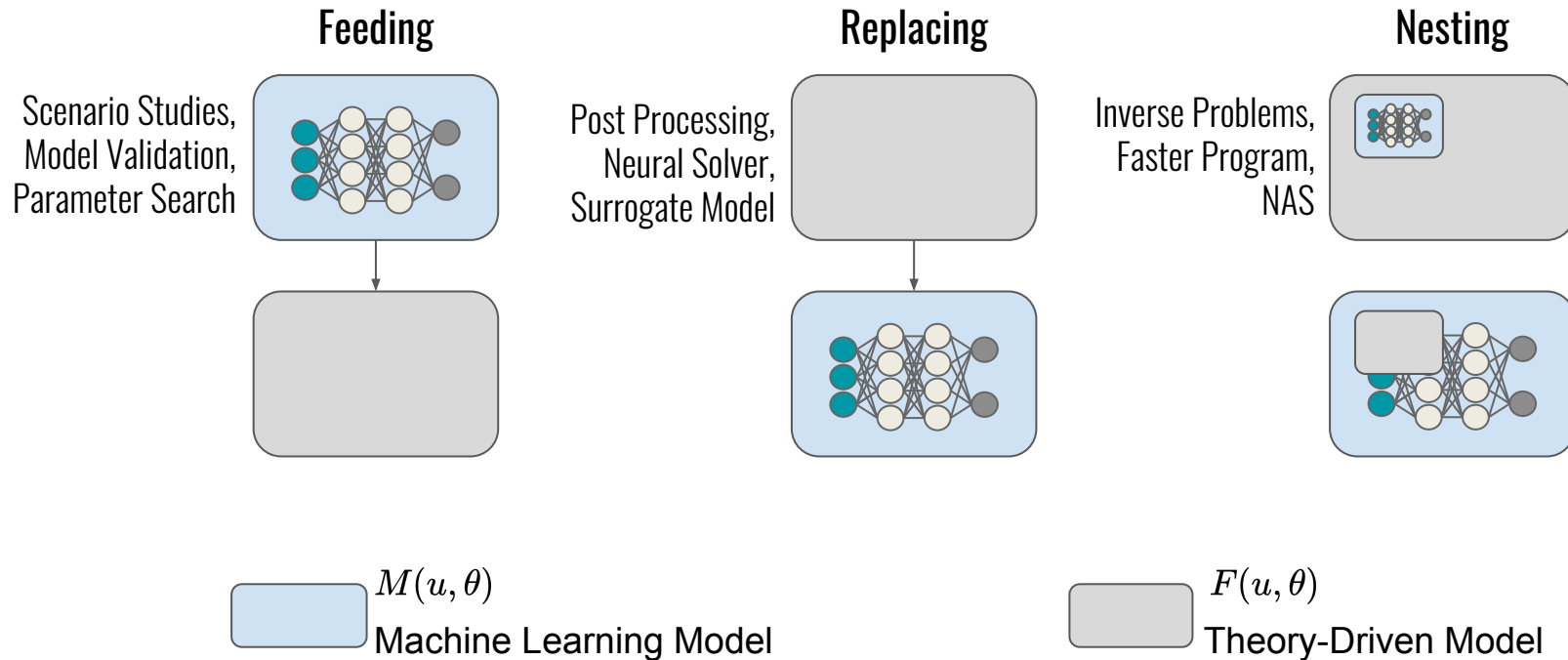
Non-sequential, intermittent assimilation



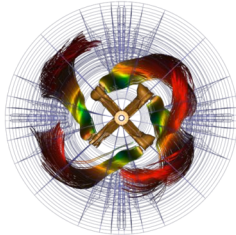
Non-sequential, continuous assimilation



SciML - Best of the Two Worlds

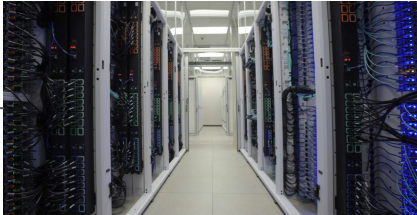
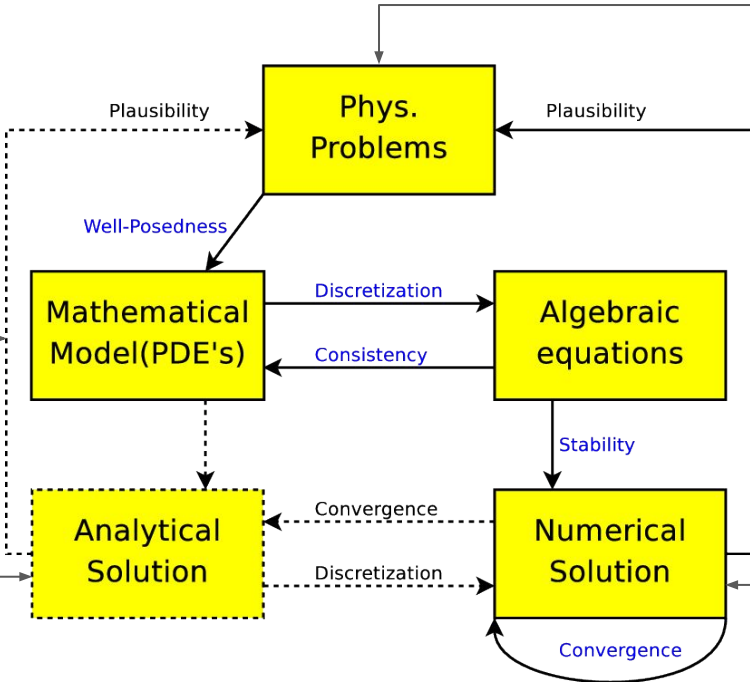
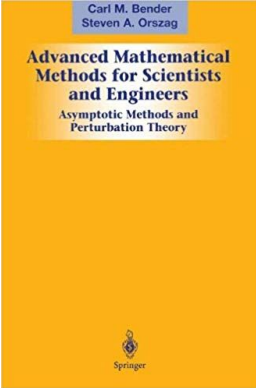


Theory / Numerical Modeling



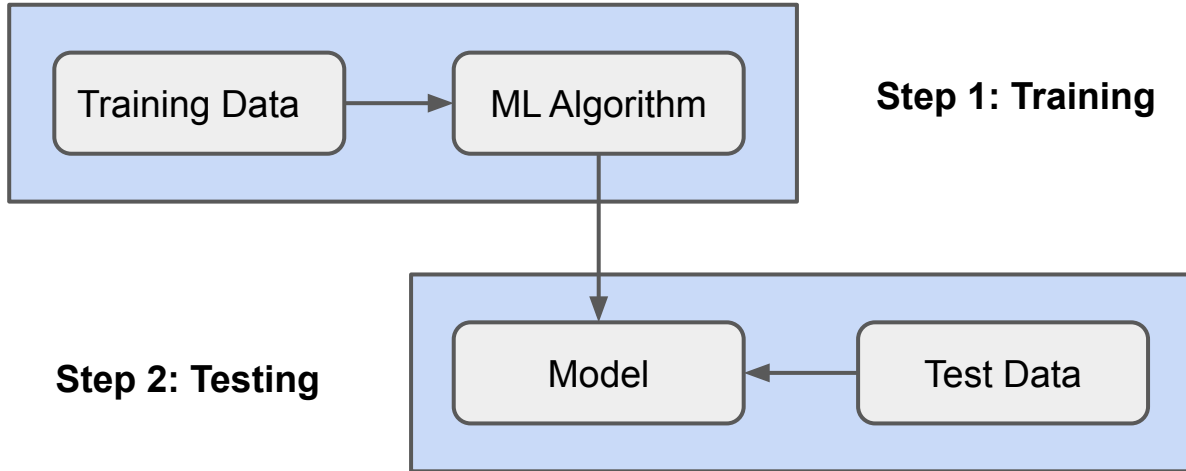
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{(u_{i+1}^n - 2u_i^n + u_{i-1}^n)}{\Delta x^2}$$

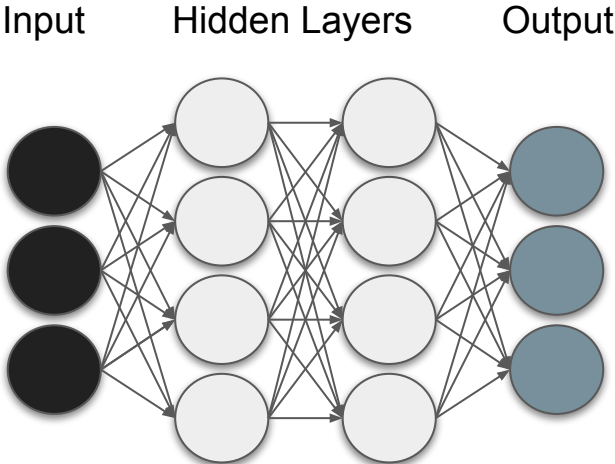
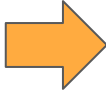
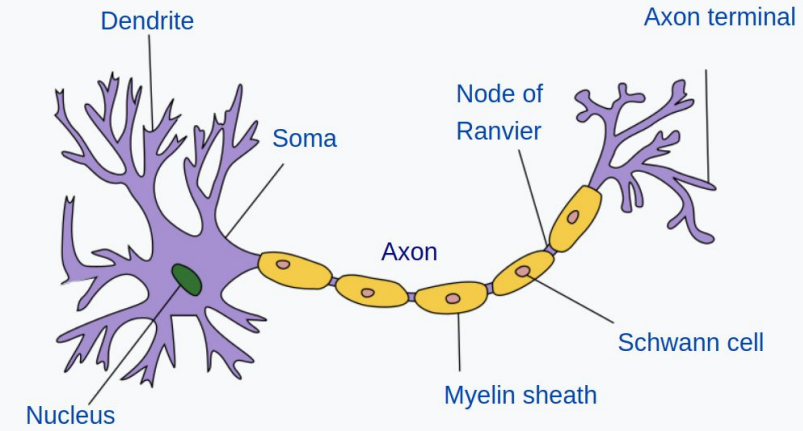


Data-Driven Model - Supervised Learning

When both input variables - X and output variables - Y are known, one can approximate the mapping function from X to Y .



Artificial Neural Network



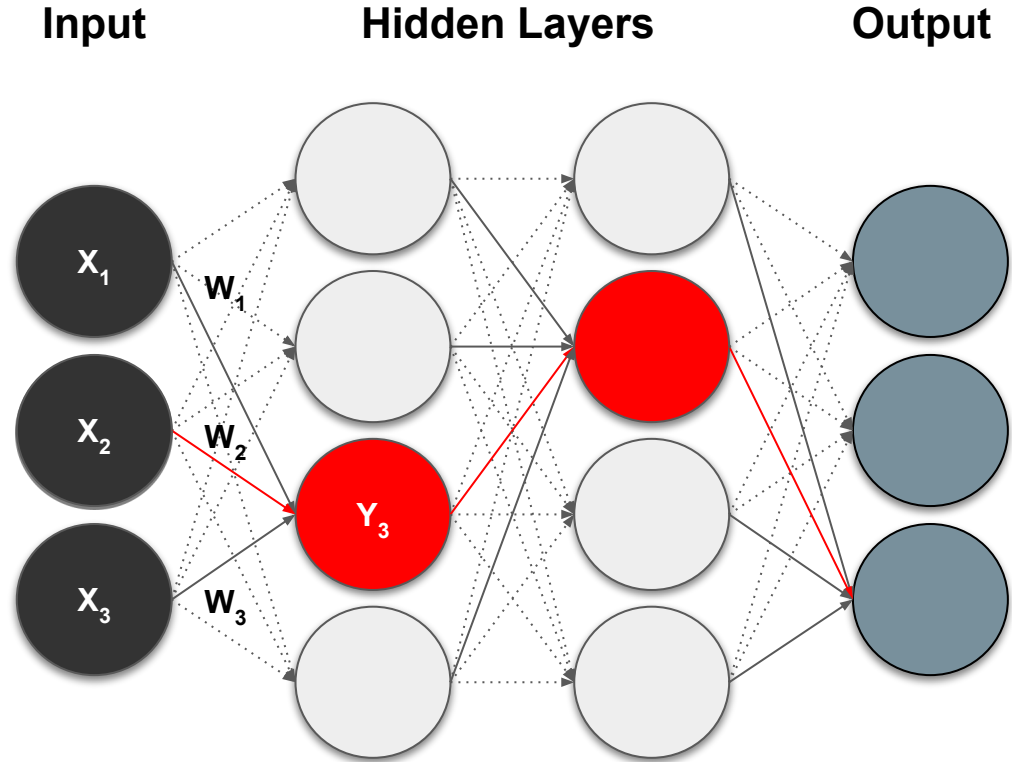
(Image Credit: Wikipedia)

Supervised Deep Learning with Neural Networks

From one layer to the next

$$Y_j = f\left(\sum_i W_i X_i + b_i\right)$$

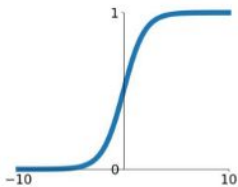
f is the activation function,
 W_i is the weight, and b_i is
the bias.



Activation Functions

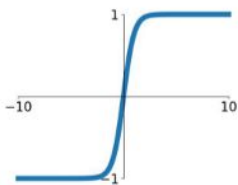
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



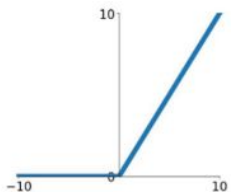
tanh

$$\tanh(x)$$



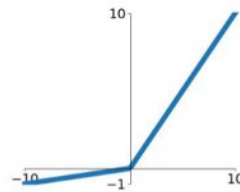
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

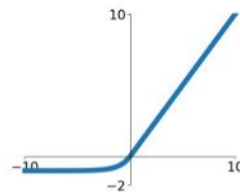


Image Credit: towardsdatascience.com

Training - Minimizing the Loss

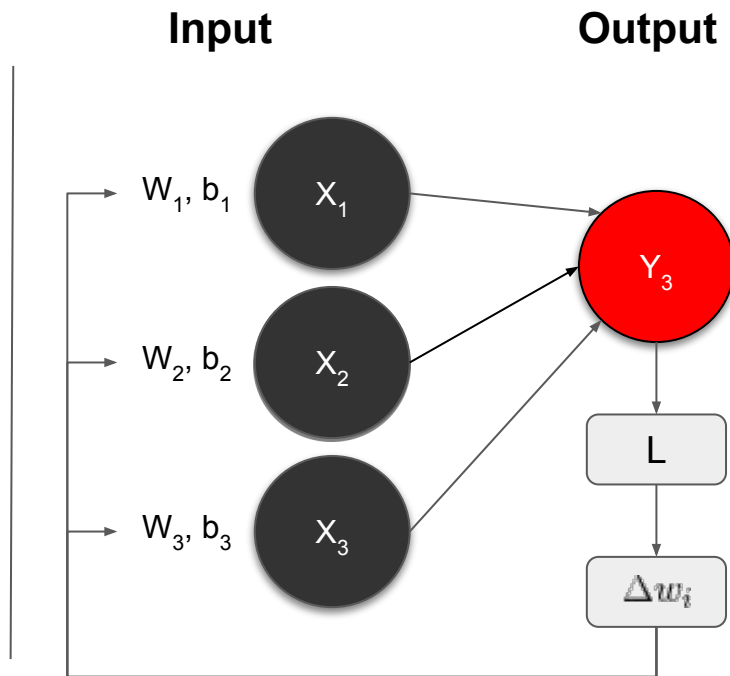
The loss function with regard to weights and biases can be defined as

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{2} \sum_i (\mathbf{Y}(\mathbf{X}, \mathbf{w}, \mathbf{b}) - \mathbf{Y}'(\mathbf{X}, \mathbf{w}, \mathbf{b}))^2$$

The weight update is computed by moving a step to the opposite direction of the cost gradient.

$$\Delta w_i = -\alpha \frac{\partial L}{\partial w_i}$$

Iterate until L stops decreasing.



Deep Neural Network as a Universal Approximator

Universal Approximation Theorem

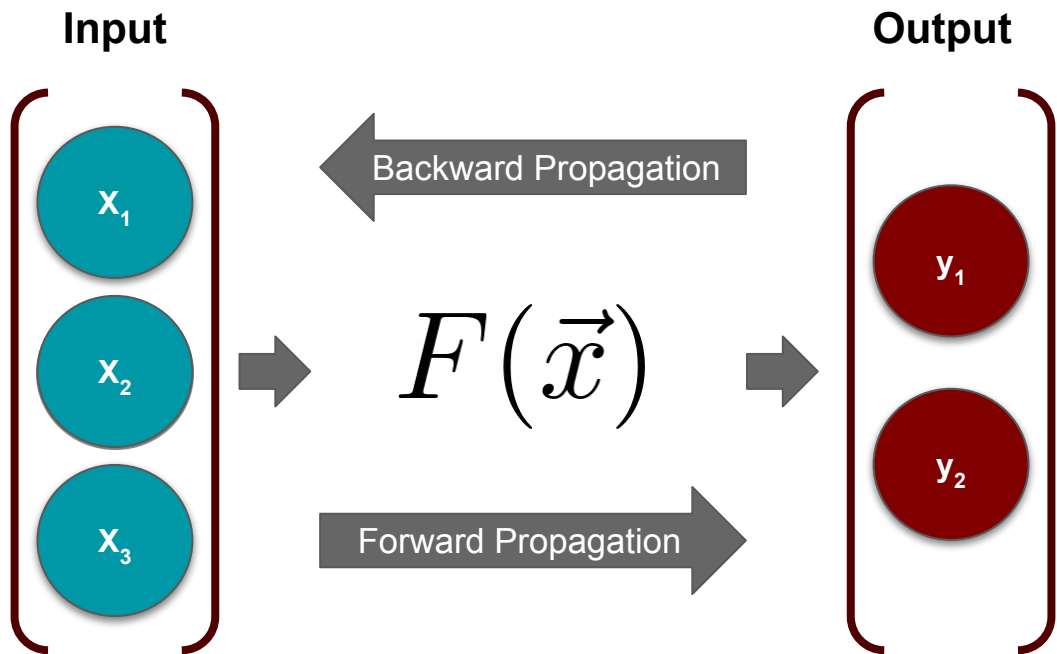
(Cybenko, 1989)

Universal approximation theorems imply that neural networks can represent a wide variety of **functions**.

Pinkus Theorem

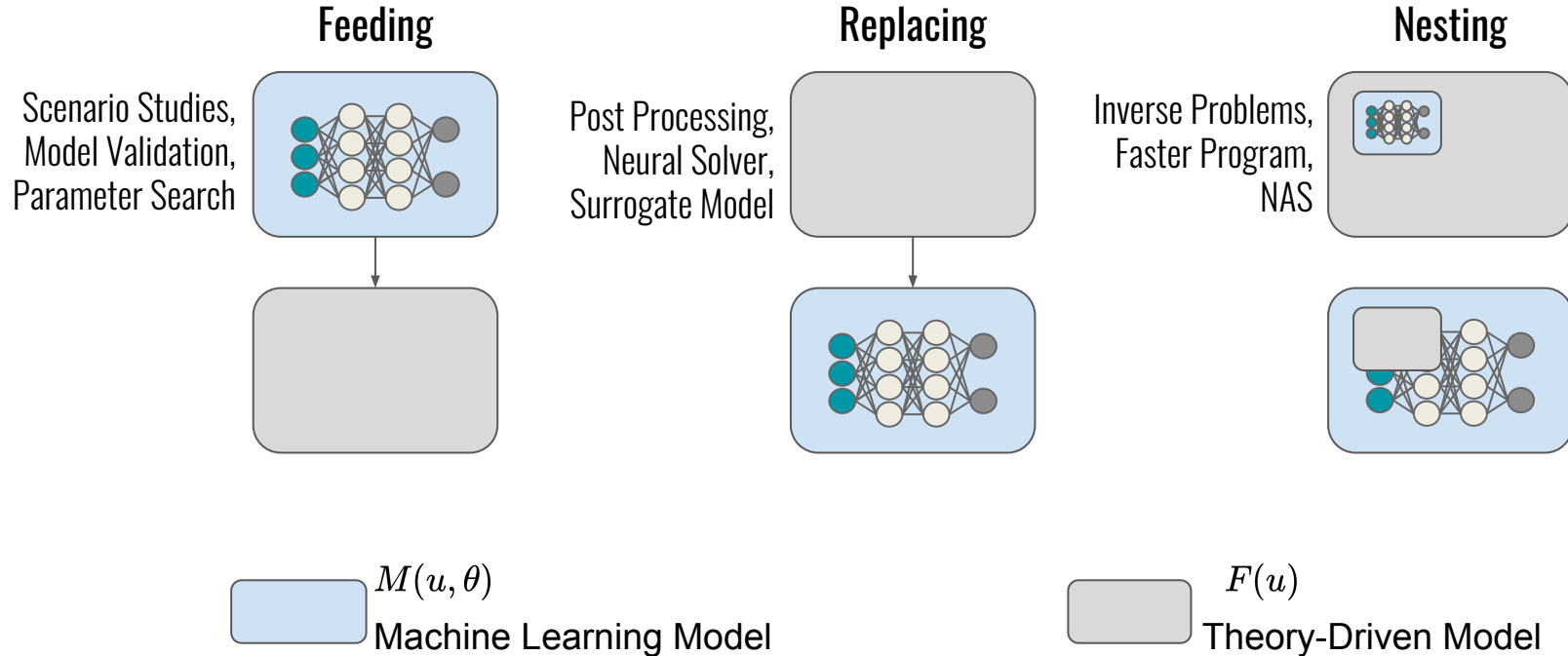
(Pinkus, 1999)

Pinkus theorems imply that neural networks can represent **directives of a function** simultaneously.

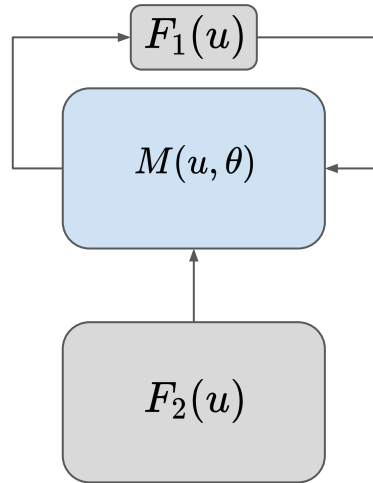


- **Training:** given **input** and **output**, find best-fit F
- **Inference:** given **input** and F , predict **output**

SciML - Best of the Two Worlds



Sample Workflow of a SciML Application



Nested Function

$$Y(u, \theta) = F_2(M(F_1(M(u, \theta)), \theta))$$

Loss Function

$$L(\theta) = \frac{1}{2} \sum_i (Y(u, \theta) - Y'(u, \theta))^2 + L_{I/BC/\dots}(\theta)$$

$M(u, \theta)$ Machine Learning Model

$F(u)$ Theory-Driven Model

Gradient Descent

$$\Delta\theta = -\alpha \frac{\partial L}{\partial \theta}$$

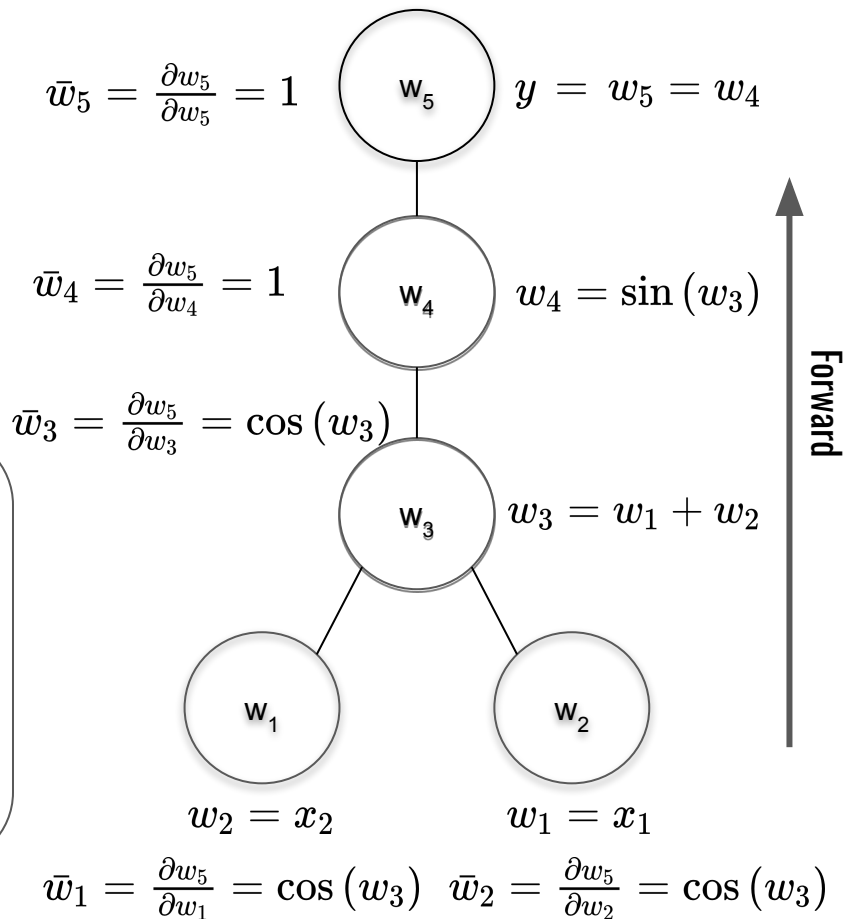
Automatic Differentiation

An autodiff system converts the program into a sequence of primitive operations to compute derivatives.

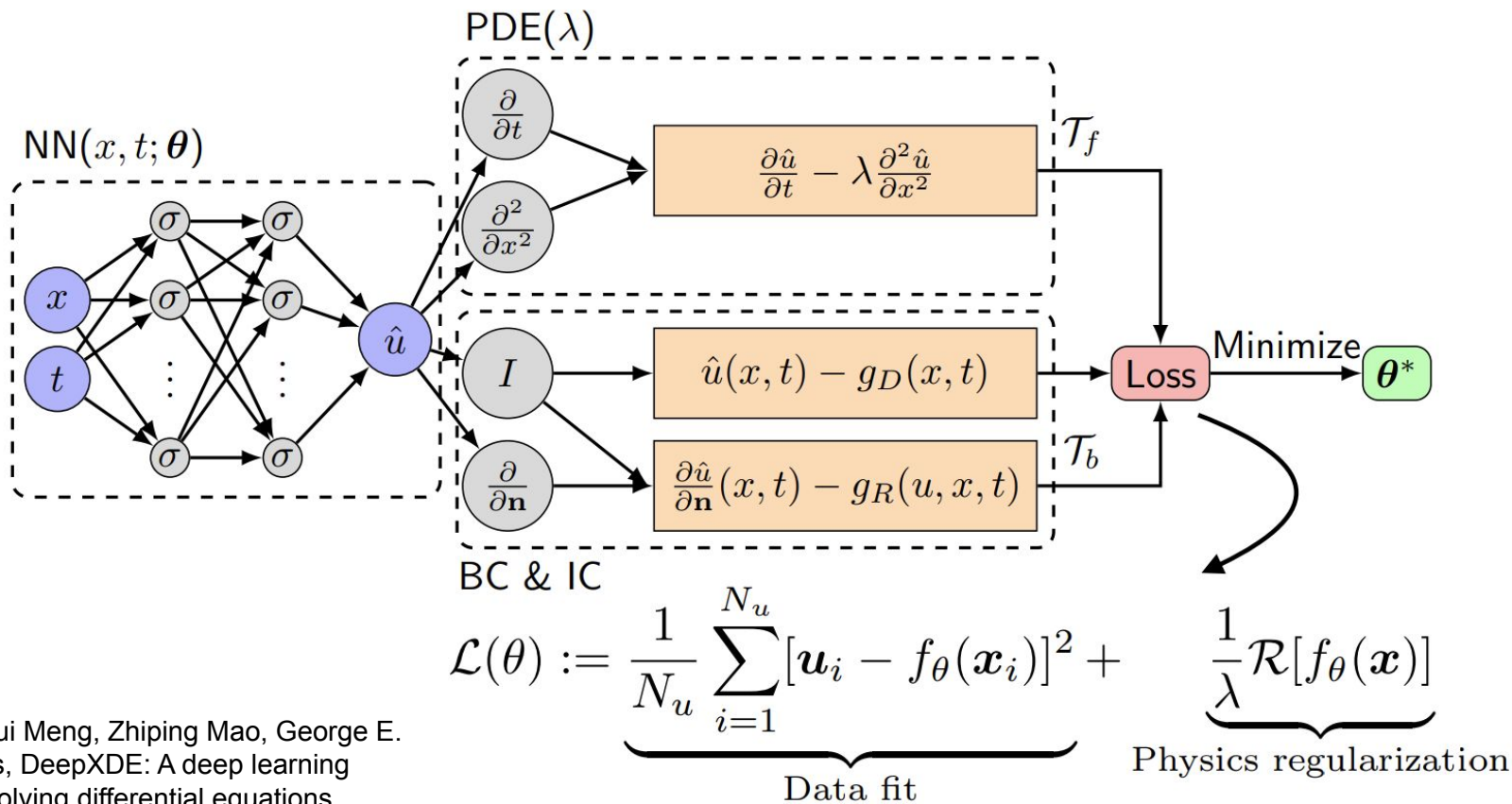
Given $y(x_1, x_2) = \sin(x_1 + x_2)$

Numerical:
(Finite Difference) $\frac{\Delta y}{\Delta x_1} = \frac{\sin(x_1+h+x_2) - \sin(x_1+x_2)}{h}$

Symbolic: $y'(x_1, x_2) = \cos(x_1 + x_2)$

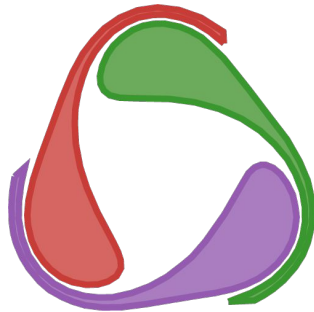


Physics-Informed Neural Networks (PINNs)



Hands-on Session

Getting Started with NeuralPDE.jl



Install Julia Packages

```
# Set Julia Depot path under $SCRATCH
$export JULIA_DEPOT_PATH=$SCRATCH/.julia

# start Julia
$cd $SCRATCH/julia-1.5.4/bin; ./julia

# type ']' to open Pkg REPL
julia>]
(@v1.5) pkg> add Plots, NeuralPDE, Flux, ModelingToolkit, GalacticOptim, Optim,
DiffEqFlux

# type 'Backspace' to get back to REPL and paste the code directly into the shell.
julia> using NeuralPDE, Flux, ModelingToolkit, GalacticOptim, Optim, DiffEqFlux
```


ODE with a 3rd-Order Derivative

$$\frac{\partial^3 u(x)}{\partial x^3} = \cos(\pi x)$$
$$u(0) = 0$$
$$u(1) = \cos(\pi)$$
$$\frac{\partial u(0)}{\partial x} = 1$$
$$x \in [0, 1]$$

```
using UnicodePlots
```

```
analytic_sol_func(x) =  
( $\pi$ *x*(-x+( $\pi$ ^2)*(2*x-3)+1)-sin( $\pi$ *x))/( $\pi$ ^3)
```

```
dx = 0.05  
xs =  
[domain.domain.lower:dx/10:domain.domain.upper  
for domain in domains][1]  
u_real = [analytic_sol_func(x) for x in xs]  
u_predict = [first(phi(x, res.minimizer))  
for x in xs]
```

```
x_plot = collect(xs)  
plot(x_plot, u_real, title = "real")  
plot!(x_plot, u_predict, title = "predict")
```

```
using NeuralPDE, Flux, ModelingToolkit, GalacticOptim, Optim, DiffEqFlux
```

```
@parameters x  
@variables u(..)
```

```
Dxxx = Differential(x)^3  
Dx = Differential(x)  
# ODE  
eq = Dxxx(u(x)) ~ cos( $\pi$ *x)
```

```
# Initial and boundary conditions  
bcs = [u(0.) ~ 0.0,  
       u(1.) ~ cos( $\pi$ ),  
       Dx(u(1.)) ~ 1.0]
```

```
# Space and time domains  
domains = [x  $\in$  IntervalDomain(0.0,1.0)]
```

```
# Neural network  
chain = FastChain(FastDense(1,8,Flux. $\sigma$ ),FastDense(8,1))
```

```
discretization = PhysicsInformedNN(chain, QuasiRandomTraining(20))  
pde_system = PDESystem(eq,bcs,domains,[x],[u])  
prob = discretize(pde_system,discretization)
```

```
cb = function (p,l)  
    println("Current loss is:  $\$$ l")  
    return false  
end
```

```
res = GalacticOptim.solve(prob, ADAM(0.01); cb = cb, maxiters=2000)  
phi = discretization.phi
```