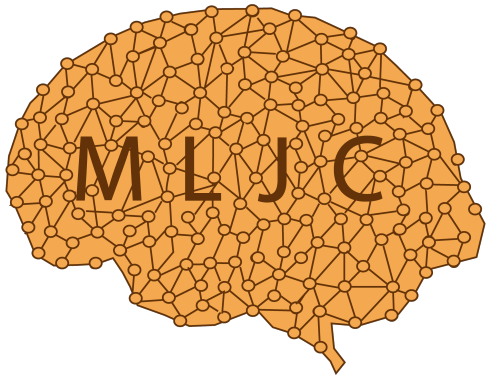
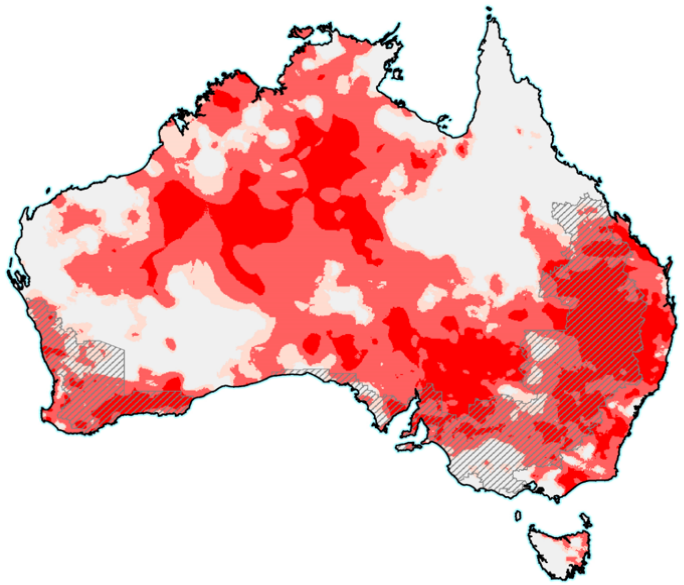
Physics-Informed Machine Learning Simulator for Wildfire Propagation

Simone Azeglio



Wildfires Spread and Real-Time Simulations



Computational Physics state-of-the-art

Weather Research & Forecasting (WRF)

It takes hours for a single run

Scientific Machine Learning techniques:

- Continuous solution (instead of discretized ones)
- Evolution forecasting (out of domain)
- Back-in-time reconstruction

Physics-Based or ML?

- **Differential equations** describe mechanisms and let the system evolve from this description $\dot{x} = f(x)$
- **Machine learning models** specify a learnable black box, which can be tuned with the right parameters in order to fit any nonlinear function

Mix the two: add scientific structure to machine learning

Physics Informed Neural Networks (PINNs)

- Curse of dimensionality?

P. Grohs et al., *arXiv:1908.10828* [math.NA], (2019)

$$P(\psi_{d,\varepsilon}) \leq c\varepsilon^{-(\mathfrak{e}+6)} d^{6[\mathfrak{d}_6+(\mathfrak{d}_1+\mathfrak{d}_2)(\theta+1)]+\mathfrak{d}_3+\mathfrak{e}\max\{\mathfrak{d}_5+\theta(\mathfrak{d}_1+\mathfrak{d}_2),\mathfrak{d}_4+\mathfrak{d}_6+2\theta(\mathfrak{d}_1+\mathfrak{d}_2)\}}$$

Some Maths behind Wildfires – Level Set Equation

Level Set Function $\Psi = \Psi(x,t)$

J.Mandel et al., *Geoscientific Model Development* (2011)

Burning Area

$$\Omega(t) = \{x : \Psi(x,t) \leq 0\}$$

Fireline

$$\Gamma(t) = \{x : \Psi(x,t) = 0\}$$

Rothermel Formula (Spread Rate)

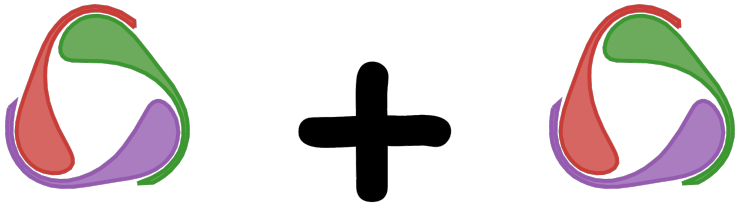
$$S = R_0(1 + \phi_W + \phi_S)$$

- R_0 Spread rate in the absence of wind
- ϕ_W Wind factor
- ϕ_S Slope factor

Level-Set Equation

$$\partial_t \Psi + S||\nabla \Psi|| = 0$$

A Novel PINN Architecture



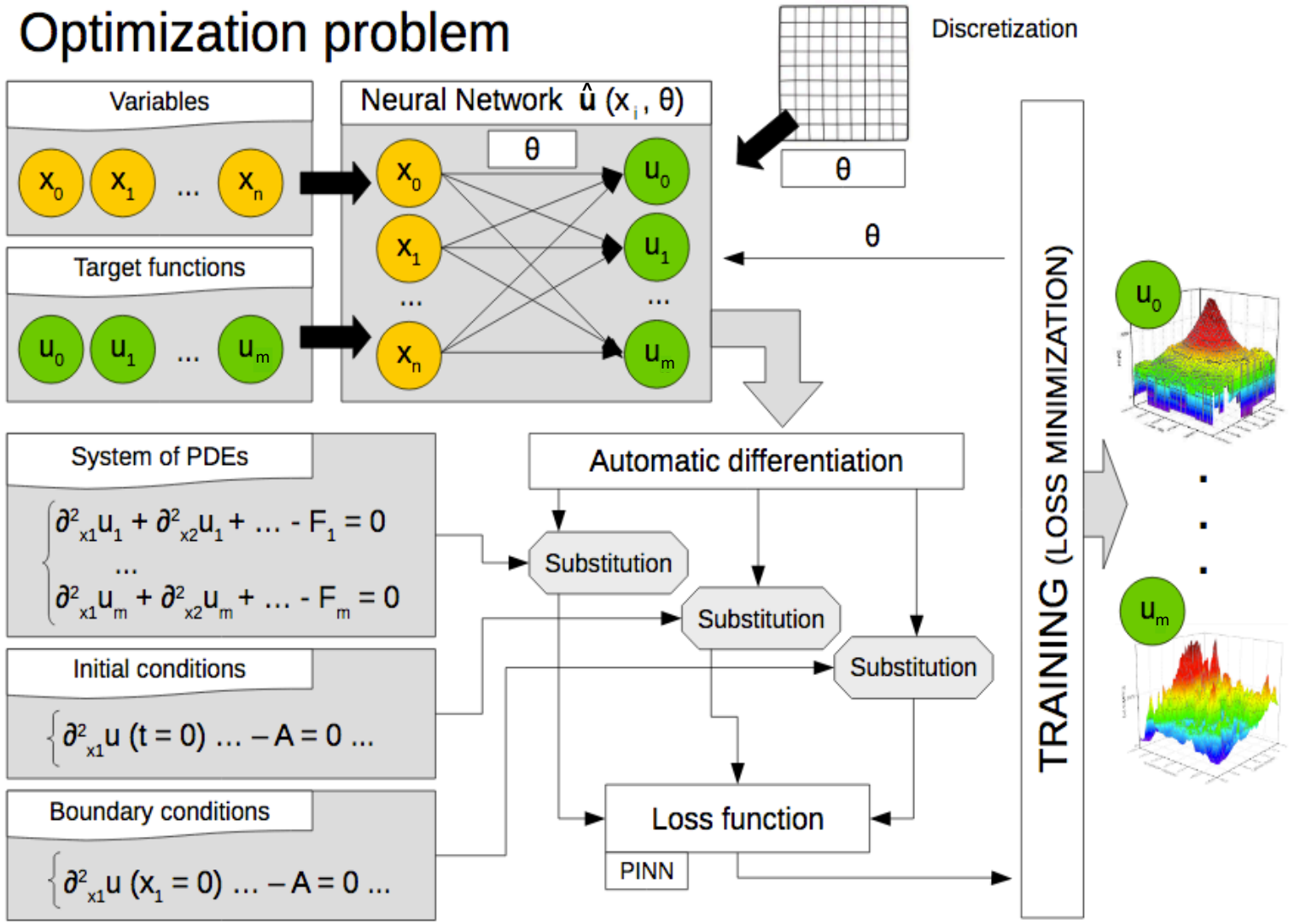
NeuralPDE.jl + **DiffEqFlux.jl**

C. Rackauckas et al., *Journal of Open Research Software* (2017)

C. Rackauckas et al., *arXiv:2001.04385* [cs.LG], (2020)

Loss Function

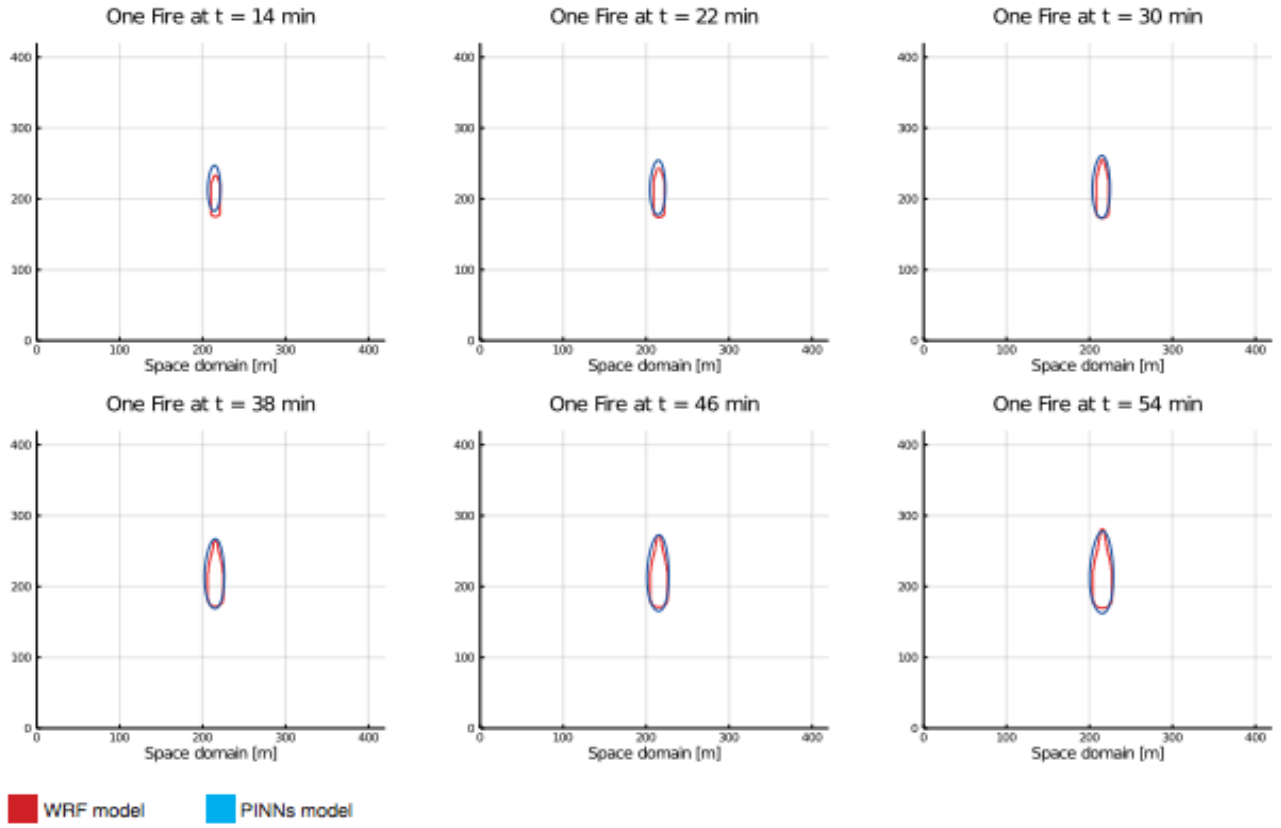
$$Loss = L^2_{PDE} + L^2_B$$



«One Fire» Simulation

Technical specs

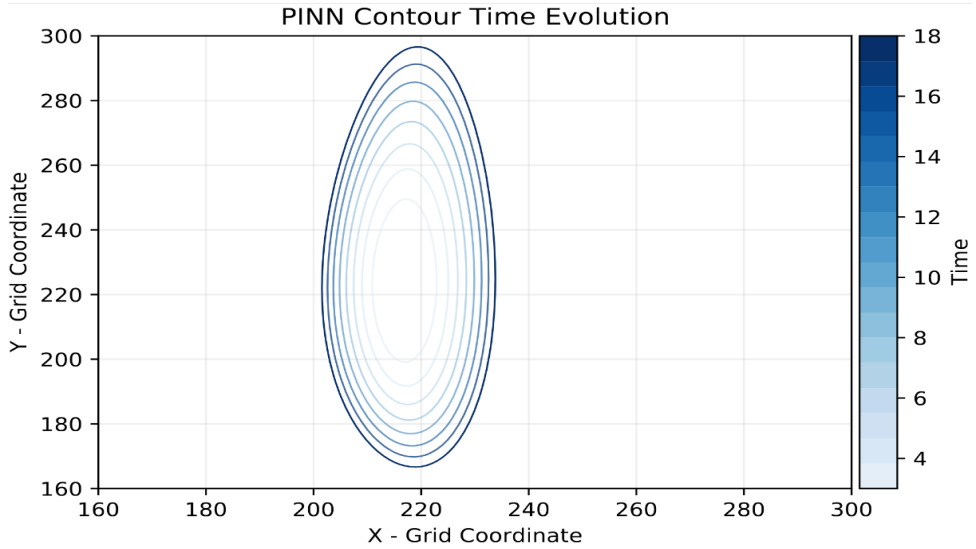
Optimization Algorithm	ADAM
Iterations	4800
Final Objective Value	6.39 e-8
Training Strategy	QuadratureTraining()
Domains	$t \in [0,10], x \in [0,10], y \in [0,10]$
Training Mesh Size	$[dt,dx,dy] = [0.17,0.02,0.02]$
Boundary Condition	$u(0,x,y,\theta) = ((5(x - 0.3))^2 + (0.15y)^2)^{\frac{1}{2}} - 0.2$
Neural Network dimensions	$3 > 16 > 1$
Training Time	647 s

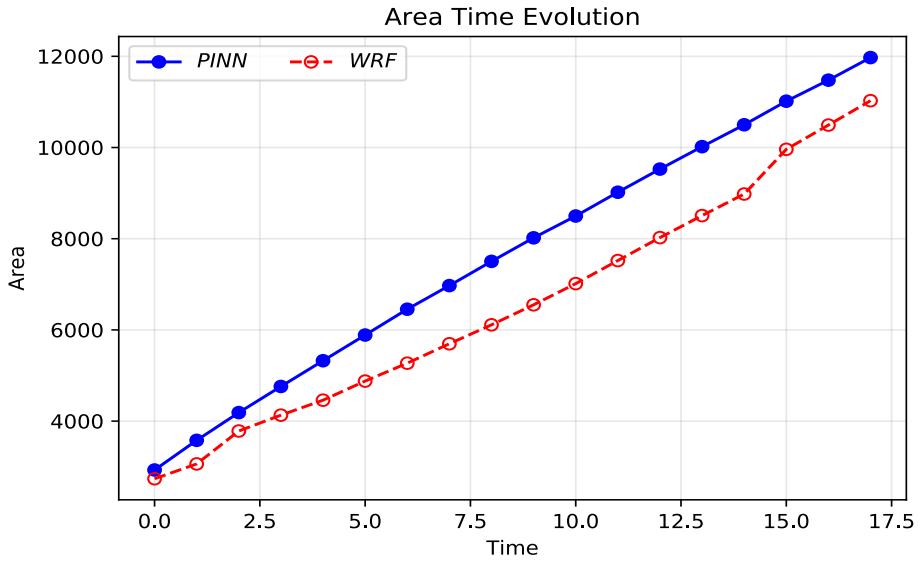


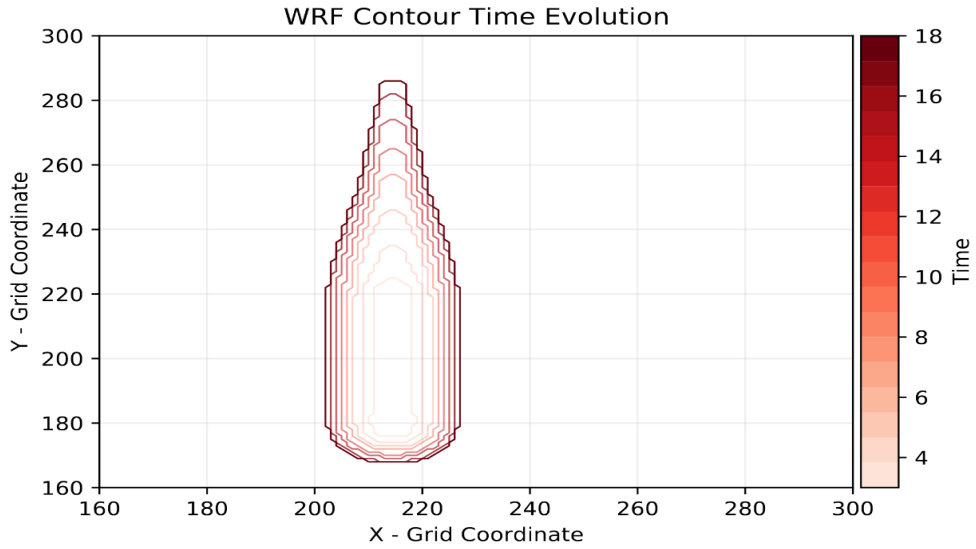
Hausdorff Distance

$$d_H(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\}$$

«One Fire» Simulation







Conclusions

Physics-Informed Machine Learning Simulator for Wildfire Propagation

GitHub https://github.com/MachineLearningJournalClub

Luca Bottero*, Francesco Calisto*, Giovanni Graziano*, Valerio Pagliarino*, Martina Scauda†, Sara Tiengo* and Simone Azeglio*

*Student, Department of Physics, University of Turin, †Student, Department of Mathematics, University of Turin
E-mails: luca.bottero192, francesco.calisto, giovanni.graziano136, valerio.pagliarino, martina.scauda, sara.tiengo, simone.azeglio@edu.unito.it

arXiv:2012.06825

Appendix – Why Differential Equations vs Why ML?

❑ **Differential equations are great**

- They understand the structure
- They extrapolate far beyond data
- They are interpretable

❑ **Differential equations are limiting**

- They require that you know the mechanism
- ML models can give a predictive model directly from data

➢ **Universal Approximation Theorem**

- Neural Networks are fancy Taylor Series: good for computing, bad for analysis

➢ **Neural Networks overcome the "Curse of Dimensionality"**

- They work well in high dimensions

➢ **Pros of NN:** They can approximate anything

➢ **Cons of NN:** They can approximate anything!

Appendix – Solving ODEs with Neural Networks

E.Lagaris et al., *arXiv: physics/9705023* (1997)

System of ODEs $u' = f(u,t) \qquad t \in [0,1] \qquad u(0) = u_0$

NN approximation $NN(t) \approx u(t) \longrightarrow NN'(t) = f(NN(t),t) \quad \forall t$

Loss Function $L(p) = \sum_i \left(\frac{dNN(t_i)}{dt} - f(NN(t_i),t_i) \right)^2 \xrightarrow{\text{minimization}} \frac{dNN(t_i)}{dt} \approx f(NN(t_i),t_i)$

Handling initial condition $L(p) = (NN(0) - u_0)^2 + \sum_i \left(\frac{dNN(t_i)}{dt} - f(NN(t_i),t_i) \right)^2$

More efficiently $g(t) = u_0 + tNN(t) \longrightarrow L(p) = \sum_i \left(\frac{dg(t_i)}{dt} - f(g(t_i),t_i) \right)^2$

Appendix – Solving ODEs with Neural Networks

```julia
using Flux , Statistics, Plots
```

```julia
NN_ODE = Chain(x -> [x], # Take a scalar as input and transform it into an array
           Dense(1,16,tanh),#input layer
           Dense(16,1), #output layer
           first) # Take first value, i.e. return a scalar
```

```
Chain(#147, Dense(1, 16, tanh), Dense(16,
```

```julia
# initial condition u_0 = 1. + all the prior info is HERE
g(t) = 1f0 + t*NN_ODE(t)
```

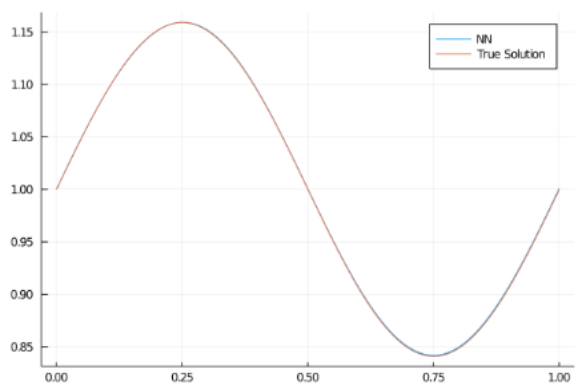
```
g
```

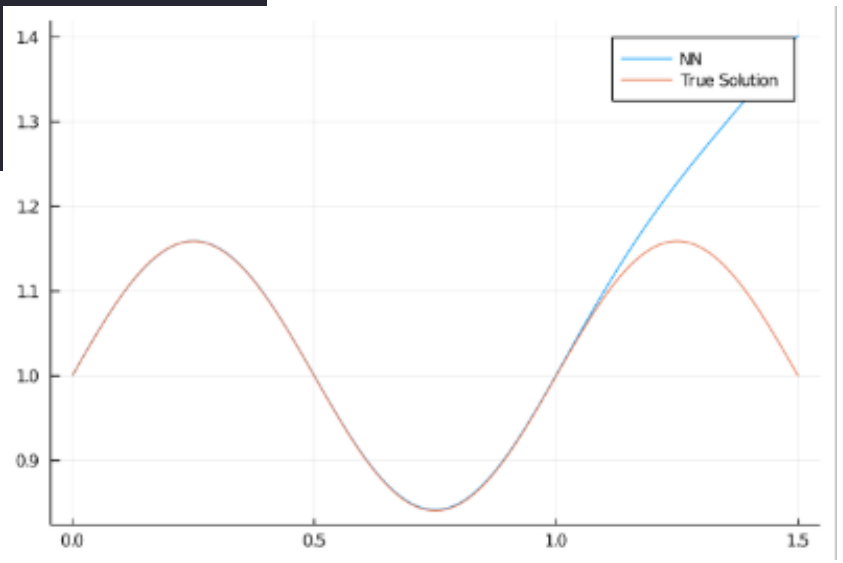
```julia
# small increment
ϵ = sqrt(eps(Float32))
```

```
0.000345…
```

```julia
# g(t+ϵ)-g(t))/ϵ --> difference quotient, derivative definition as a limit
loss() = mean(abs2(((g(t+ϵ)-g(t))/ϵ) - cos(2π*t)) for t in 0:1f-2:1f0)
```

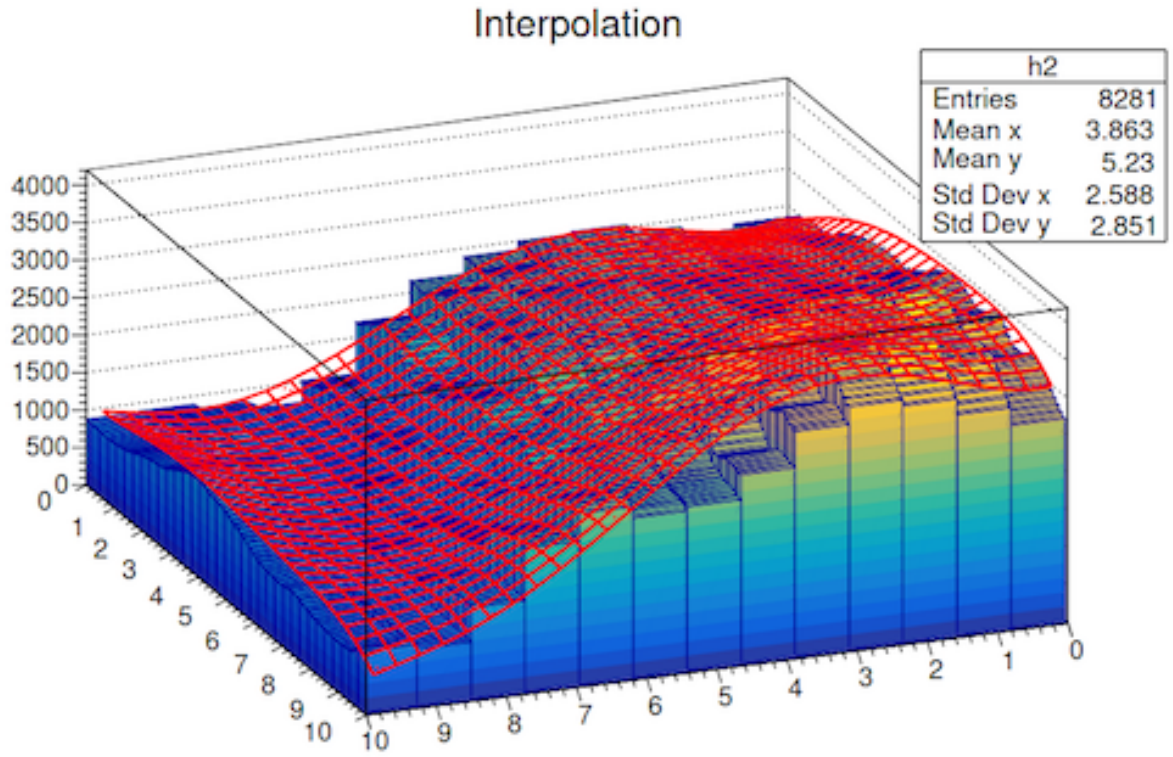
```
loss
```





Appendix – Level Set Implementation

- **90 d** — •ProjectX 2020: EVALUATING THE FEASIBILITY OF A PINNS-BASED ARCHITECTURE
- **40 d** — •IMPLEMENTING INPUT OF DISCRETIZED DATA (ARRAYS) IN NEURALPDE.JL
- **30 d** — •FIXING STABILITY FOR LARGE DOMAINS
- **60 d** — •FIXING BUGS IN THE CURRENT INTERFACE FOR GPUS
•UPGRADING MULTITHREADING
- **15 d** — •ENABLING THE MACROS OF MODELINKTOOLKIT.JL FOR DEFINITION OF NUMERICAL PROBLEM
- **200 d** — •IMPLEMENTATION OF COUPLED WRF + SFIRE PHYSICS USING THE NEW ARCHITECTURE
- **60 d** — •TEST PHASE 'A'
- **45 d** — •IMPLEMENTATION / INTERFACING FOR NON-GLOBAL MODULES
- **30 d** — •NESTING AND SUBDOMAINS
- **40 d** — •TEST PHASE 'B'
- **80 d** — •WRAPPING AND EMBEDDING INSIDE THE WRF ARCHITECTURE OR DESIGN OF A NEW FRONTEND
- **180 d** — •TEST PHASE 'C'
- **α** — •ALPHA RELEASE



Appendix - Limitations

Limitations

- It can't run on GPUs or multi-CPUs due to library limitations
- Usage of a variable fuel map as a matrix is not yet supported
- Stability analysis on discretization strategies and degree of freedom of the NN has to be made
- How to deal with multiple ignition points?!
- The technique hasn't been extended on different architectures: CNNs, RNNs
- Generalization capabilities aren't still 100% clear