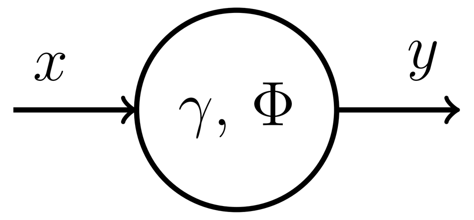
Robust learning with implicit residual networks

Viktor Reshniak, *Data Analysis and Machine Learning, Oak Ridge National Laboratory*
Clayton Webster, *University of Tennessee, Knoxville* and *Lirio LLC*

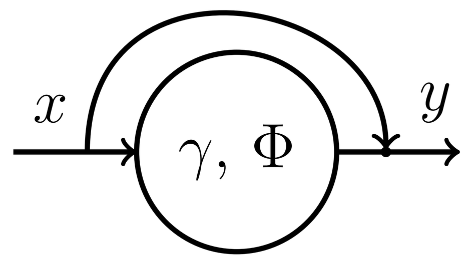
U.S. DEPARTMENT OF ENERGY

OAK RIDGE National Laboratory

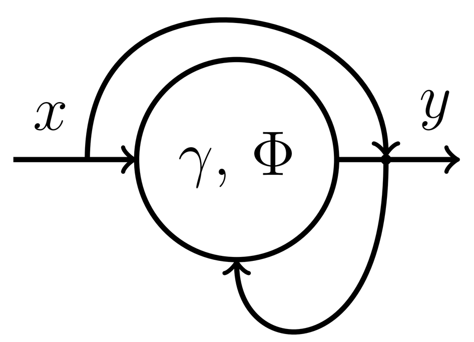
Description



feed forward layer



residual layer



implicit residual layer

In this effort, we propose a new deep architecture utilizing residual blocks inspired by implicit discretization schemes. As opposed to the standard feed-forward networks, the outputs of the proposed implicit residual blocks are defined as the fixed points of the appropriately chosen nonlinear transformations. We show that this choice leads to the improved stability of both forward and backward propagations, has a favorable impact on the generalization power and allows to control the robustness of the network with only a few hyperparameters. In addition, the proposed reformulation of ResNet does not introduce new parameters and can potentially lead to a reduction in the number of required layers due to improved forward stability. Finally, we derive the memory-efficient training algorithm, propose a stochastic regularization technique and provide numerical results in support of our findings.

[1] V. Reshniak, C. Webster, Robust learning with implicit residual networks, *arXiv:1905.10479*, 2020.

[2] `https://github.com/vreshniak/ImplicitResNet`

Vector field regularization

Spectral normalization:

$$F^{\alpha,\beta}(\gamma,x) := \frac{\alpha+\beta}{2}x + \frac{\beta-\alpha}{2}S(\vartheta)\odot F\left(\frac{\gamma}{\|\gamma\|_2},x\right)$$

where $S(\vartheta)\in(0,1)$ is the sigmoid function. All eigenvalues of the Jacobian $\frac{\partial F^{\alpha,\beta}(\gamma,x)}{\partial x}$ are located in the disc with radius $(\beta-\alpha)/2$ centered at $(\alpha+\beta)/2$.

Trajectory regularization:

$$\frac{\alpha_{div}}{dT}\sum_{t=0}^{T}{}'\left(\frac{t}{T}\right)^p \nabla\cdot F(\gamma_t,y_t) + \frac{\alpha_{TV}}{T}\sum_{t=1}^{T}\|\gamma_t-\gamma_{t-1}\|^2$$

using Hutchinson trace estimator

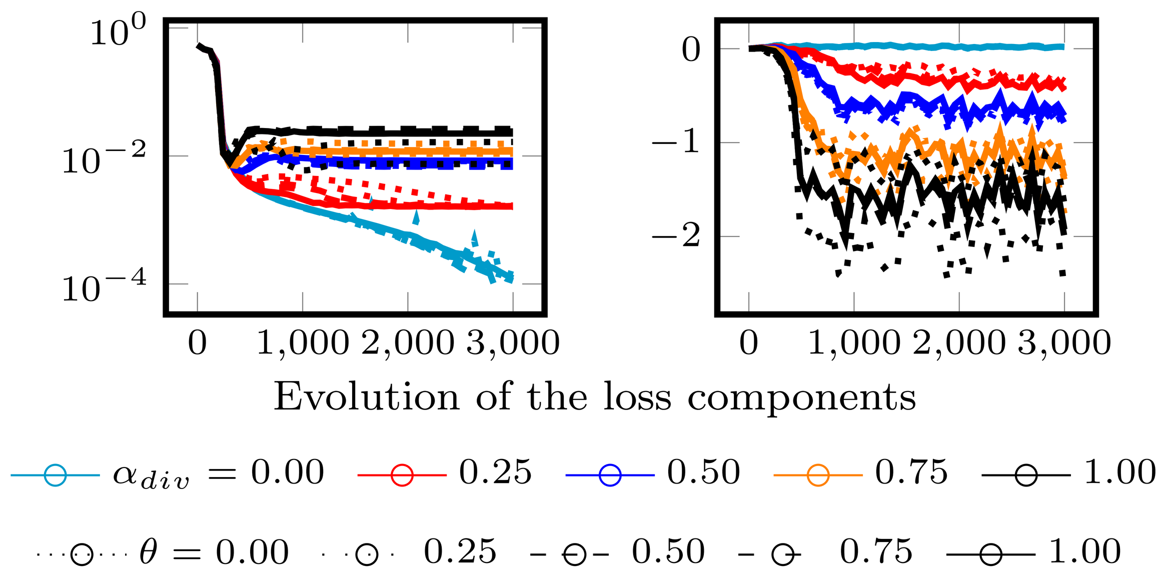
$$\nabla\cdot F(\gamma_t,y_t) = \mathrm{E}_{z\sim\mathcal{N}(0,1)}\left(z^T\frac{\partial F(\gamma_t,y_t)}{\partial y_t}z\right)$$

Results

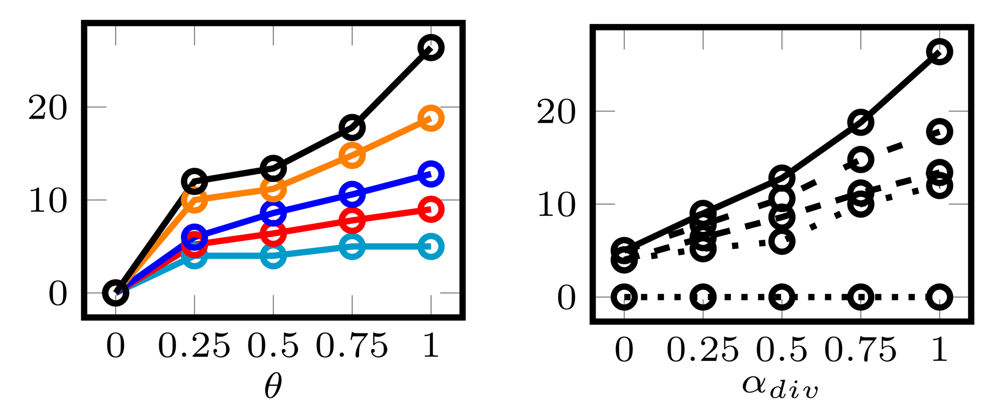
Example 1. Regression

Network: $T=5$ residual layers and GeLU MLP with 3 hidden layers of width 10 without normalization

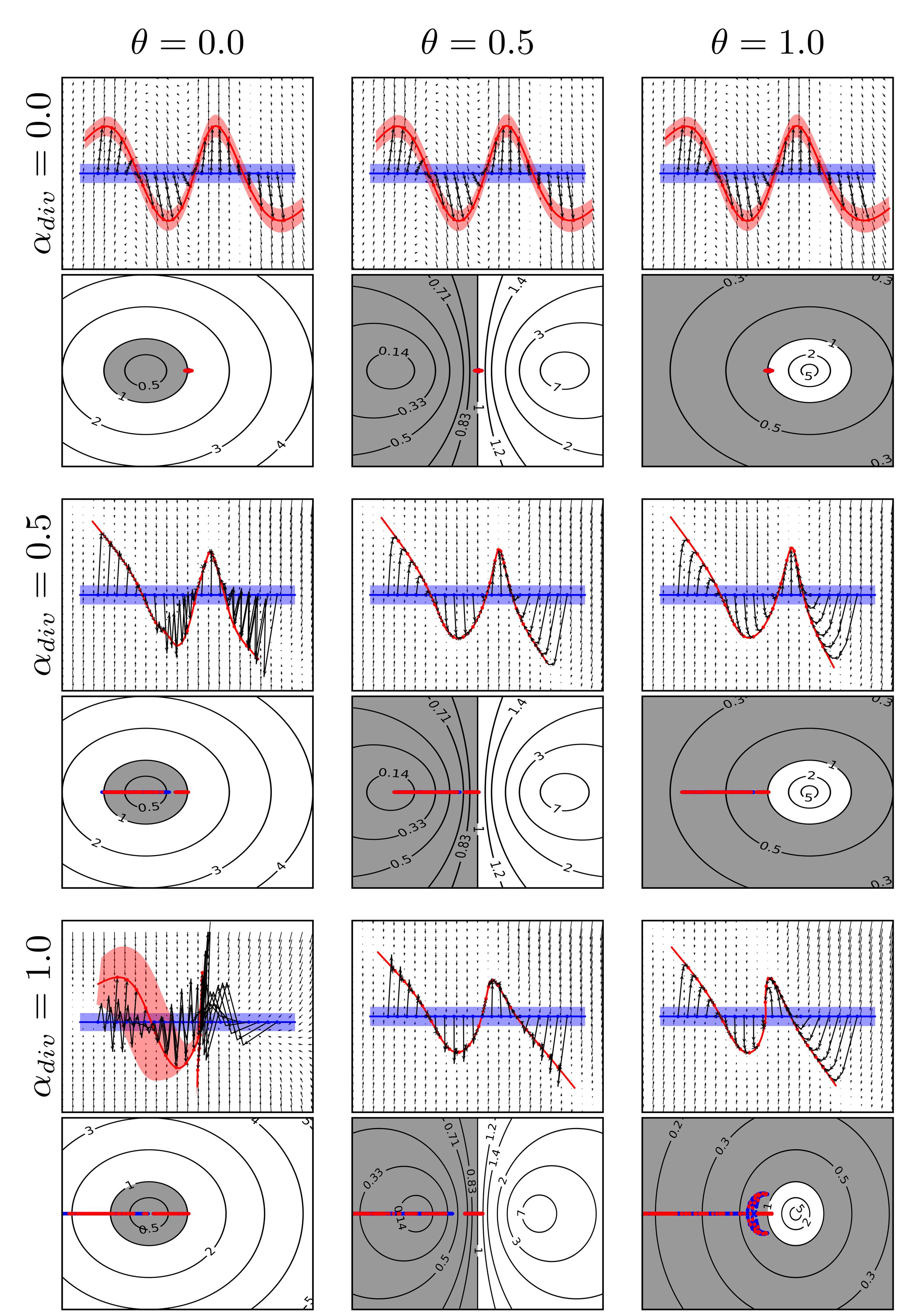
Loss: $\frac{1}{N}\sum_{i=1}^{N}\left\|g(y_T^i)-f(x^i)\right\|^2 + \frac{\alpha_{div}}{dT}\sum_{t=0}^{T}{}'\left(\frac{t}{T}\right)^2\nabla\cdot F(\gamma,y_t^i)$



Evolution of the loss components



Nonlinear iterations per residual layer



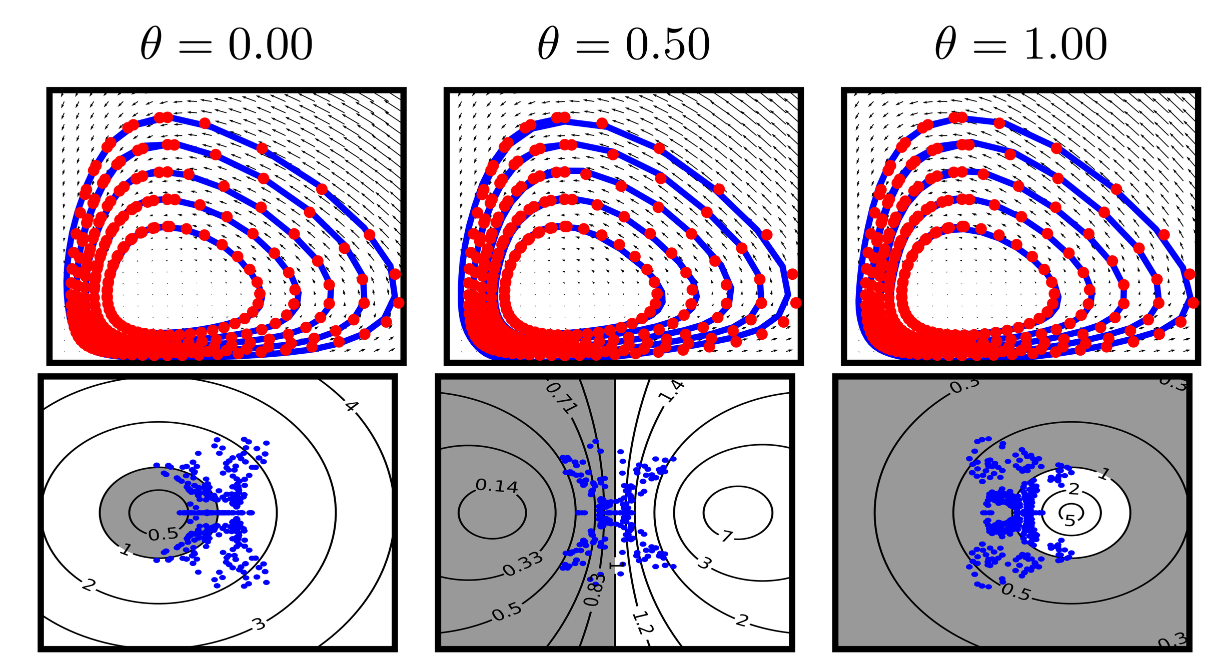
Vector fields, stability regions and spectrum along trajectories

Example 2. Periodic ODE

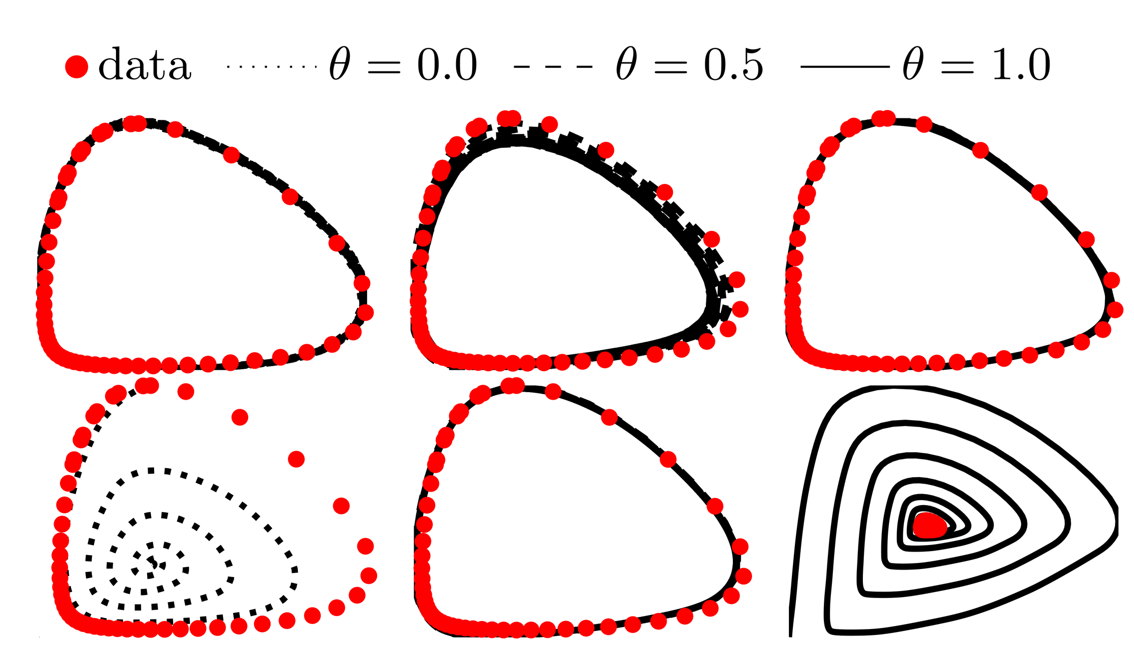
$$\dot z_1 = \frac{2}{3}z_1 - \frac{4}{3}z_1z_2, \qquad \dot z_2 = z_1z_2 - z_2$$

Network: $T=50$ residual layers and ReLU MLP with 4 hidden layers of width 20 without normalization

$$\text{Loss: } \frac{1}{50N}\sum_{i=1}^{N}\sum_{j=1}^{50}\left\|y_j^i - z^i(0.2j)\right\|^2$$



(Top) Learned vector fields and trajectories on the time interval $t\in[0,10]$. (Bottom) Eigenvalues of the vector fields along these trajectories.



(Top) Single trajectory generated by three trained implicit residual networks on the time interval $t\in[0,200]$; (Bottom) continuous-time trajectory generated by the learned vector fields of these residual networks on the same time interval.

Example 3. MNIST classification

Network: ResNet-18 with 8 input channels and $F^{-3,1}$

Noise intensity	Accuracy				
	$\theta=0$	0.25	0.50	0.75	1.00
0.00	100.0	100.0	100.0	100.0	100.0
0.10	98.1	100.0	99.9	99.9	100.0
0.20	90.9	96.7	97.1	97.9	98.1
0.30	78.5	87.3	90.3	91.9	94.2
0.40	63.1	75.1	77.0	78.9	85.1
0.50	50.3	63.2	63.6	65.4	73.4

Classifiaction accuracy for data corrupted with Gaussian noise.

Implicit layer: $y = x + \Phi(\gamma,x,y)$

Block of implicit layers for $t=1,\dots,T$:

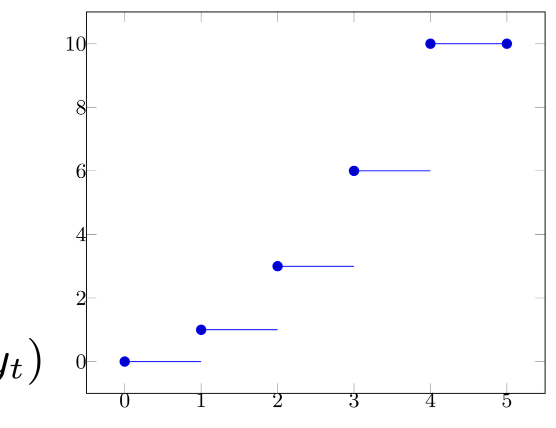
$$y_t = y_{t-1} + \Phi(\gamma_t, y_{t-1}, y_t),$$
$$y_0 = x.$$

Nonlinear maps $\Phi(\gamma,y_{t-1},y_t)$:

$(1-\theta)F(\gamma_{t-1},y_{t-1}) + \theta F(\gamma_t,y_t)$

or

$F(\gamma_{t-1},(1-\theta)y_{t-1}+\theta y_t)$

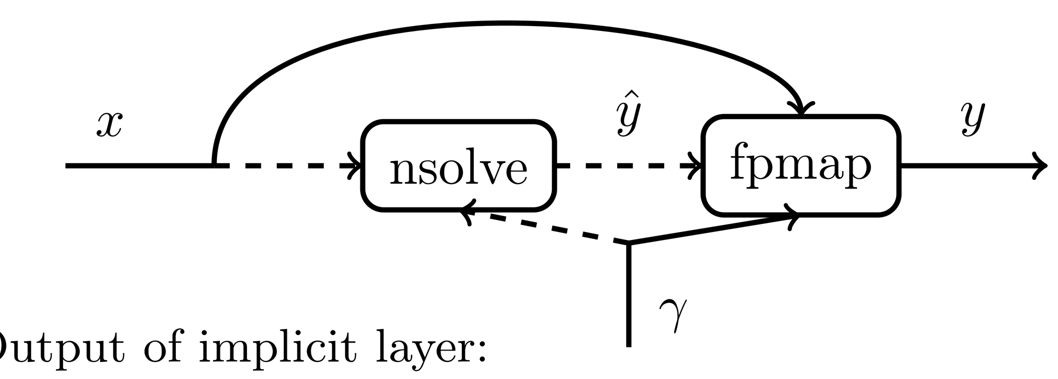


γ is cádlág function

Derivatives of the nonlinear maps:

$\Phi(\gamma,x,y)$	$(1-\theta)F(\gamma,x)+\theta F(\gamma,y)$	$F(\gamma,z)$, $z=(1-\theta)x+\theta y$
$\frac{\partial\Phi(\gamma,x,y)}{\partial x}$	$(1-\theta)\frac{\partial F(\gamma,x)}{\partial x}$	$(1-\theta)\frac{\partial F(\gamma,z)}{\partial z}$
$\frac{\partial\Phi(\gamma,x,y)}{\partial y}$	$\theta\frac{\partial F(\gamma,y)}{\partial y}$	$\theta\frac{\partial F(\gamma,z)}{\partial z}$
$\frac{\partial\Phi(\gamma,x,y)}{\partial\gamma}$	$(1-\theta)\frac{\partial F(\gamma,x)}{\partial\gamma}+\theta\frac{\partial F(\gamma,y)}{\partial\gamma}$	$\frac{\partial F(\gamma,z)}{\partial\gamma}$

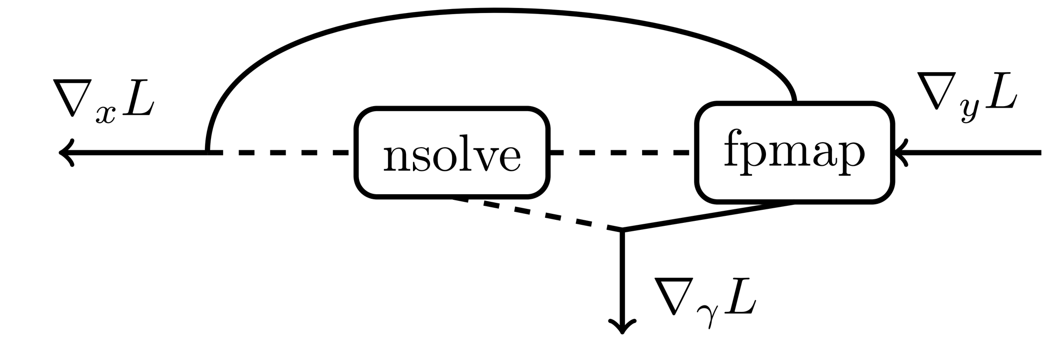
Forward propagation:



Output of implicit layer:

$$y \leftarrow \arg\min_z \|z - x - \Phi(\gamma,x,z)\|^2$$

Backward propagation:



$$\frac{\partial y}{\partial x} = \left(I - \frac{\partial\Phi(\gamma,x,y)}{\partial y}\right)^{-1}\left(I + \frac{\partial\Phi(\gamma,x,y)}{\partial x}\right)$$

The backpropagation formulas follow immediately

$$\left(I - \frac{\partial\Phi(\gamma,x,y)}{\partial y}\right)^T\overline{\nabla_y L} = \nabla_y L$$

$$\nabla_x L = \left(I + \frac{\partial\Phi(\gamma,x,y)}{\partial x}\right)^T\overline{\nabla_y L}$$

$$\nabla_\gamma L = \frac{\partial\Phi(\gamma,x,y)}{\partial\gamma}^T\overline{\nabla_y L}$$