A library for artificial neural networks calculations in (not only) molecular physics

Martin Beseda

IT4Innovations national supercomputing center

November 7, 2018

Initial Motivation

Potential energy surface

- Description of the system potential energy

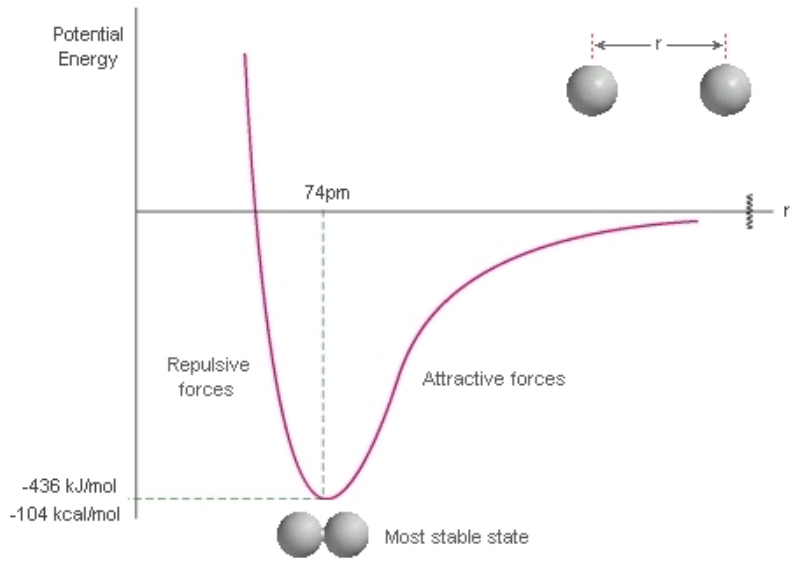


Fig. source: chem.ucalgary.ca

Potential energy surface

- Description of the system potential energy
- Grid computed by *ab initio* methods

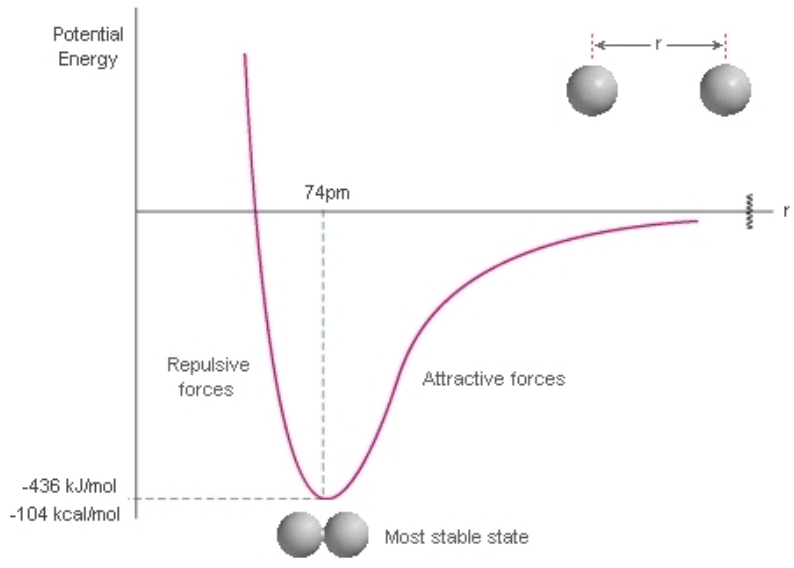


Fig. source: chem.ucalgary.ca

Potential energy surface

- Description of the system potential energy
- Grid computed by *ab initio* methods
- Analytic formulas may not be adaptive enough for complex surfaces
  - $(H_2O)_{118}$ clusters – PES function of dim. 1056

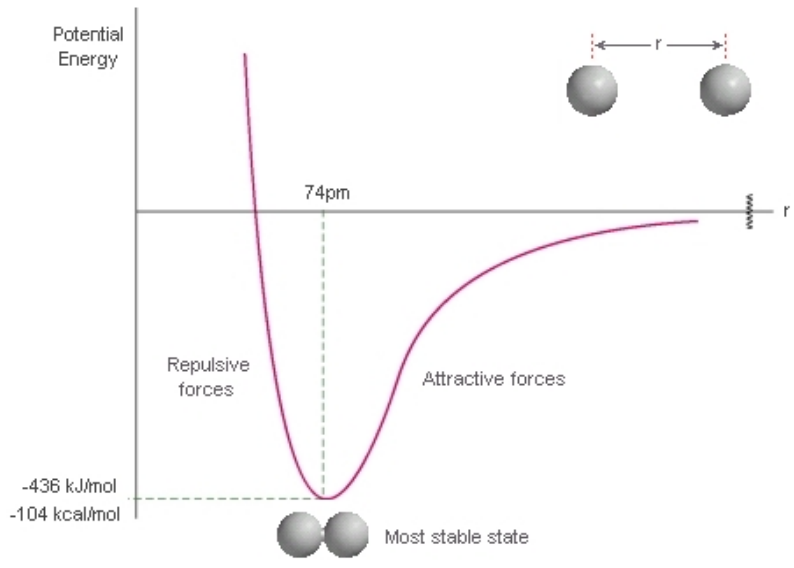


Fig. source: chem.ucalgary.ca

Potential energy surface

- Description of the system potential energy
- Grid computed by *ab initio* methods
- Analytic formulas may not be adaptive enough for complex surfaces
  - $(H_2O)_{118}$ clusters – PES function of dim. 1056

Utilization

- Interactions representation (molecular dynamics, Monte Carlo simulations)

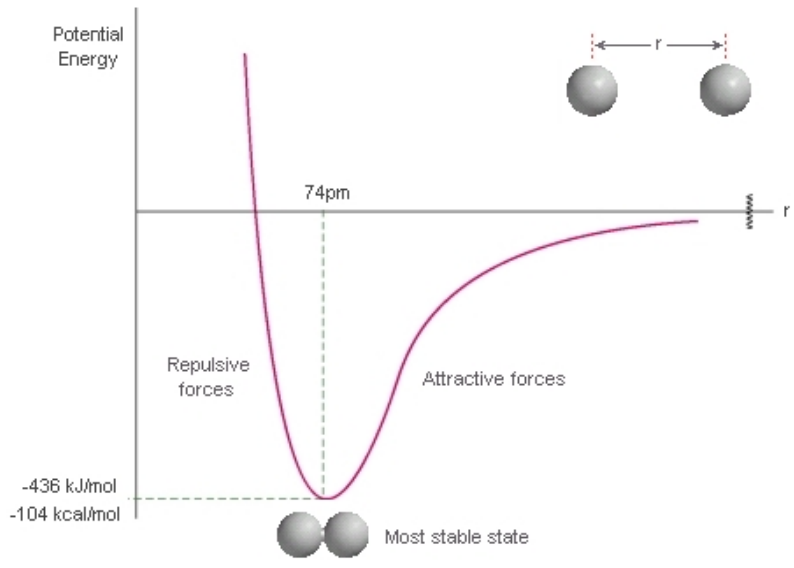


Fig. source: chem.ucalgary.ca

PES utilization example

Hamilton's equations of motion

$$\dot{\vec{r}}_K = \frac{\vec{p}_K}{m_K}$$

$$\dot{\vec{p}}_K = \vec{F}_K$$

$$\vec{F}_K = -\sum_{\alpha} |c_\alpha|^2 \nabla_K E_\alpha - \sum_{\alpha \neq \beta} c_\beta^* c_\alpha (E_\beta - E_\alpha) \vec{d}_{\beta\alpha}^{(K)}$$

m_K K-th nucleus mass

$\vec{r}_K$ K-th nucleus position

$\vec{p}_K$ K-th nucleus momentum

$\vec{F}_K$ Force acting on k-th atom

c_α Probability amplitude of the system being in α-state

$\vec{d}_{\beta\alpha}^{(K)}$ Non-adiabatic coupling elements

PES utilization example

Hamilton's equations of motion

$$\dot{\vec{r}}_K = \frac{\vec{p}_K}{m_K}$$

$$\dot{\vec{p}}_K = \vec{F}_K$$

$$\vec{F}_K = -\sum_{\alpha} |c_\alpha|^2 \nabla_K E_\alpha - \sum_{\alpha \neq \beta} c_\beta^* c_\alpha (E_\beta - E_\alpha) \vec{d}_{\beta\alpha}^{(K)}$$

m_K K-th nucleus mass

$\vec{r}_K$ K-th nucleus position

$\vec{p}_K$ K-th nucleus momentum

$\vec{F}_K$ Force acting on k-th atom

c_α Probability amplitude of the system being in α-state

$\vec{d}_{\beta\alpha}^{(K)}$ Non-adiabatic coupling elements

Schrödinger equation

$$\boxed{\hat{H}\,|\psi\rangle = \hat{E}\,|\psi\rangle}$$

- Describes the physical system with quantum effects in nuclear degrees of freedom
- Provides both the possible (quantized) energy-levels of the system and the corresponding wave functions
- Solving is difficult for high dimensional systems

Schrödinger equation

- Describes the physical system with quantum effects in nuclear degrees of freedom
- Provides both the possible (quantized) energy-levels of the system and the corresponding wave functions
- Solving is difficult for high dimensional systems

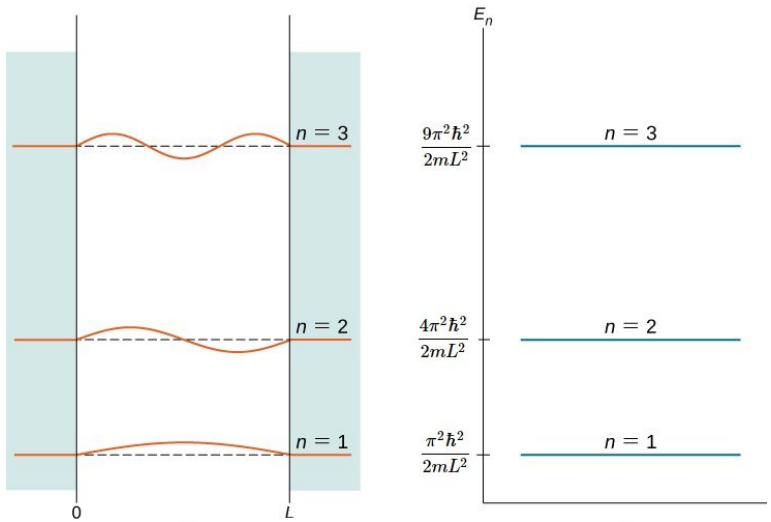


Fig. source: CC-BY 4.0; OpenStax

Schrödinger equation

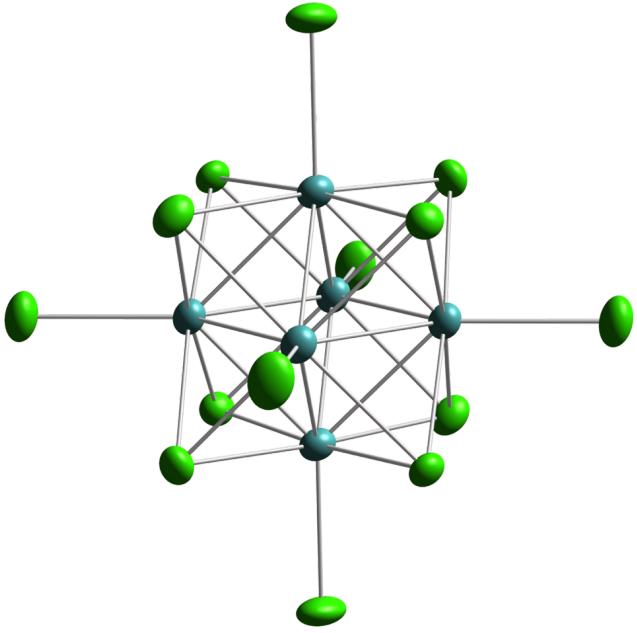
- Describes the physical system with quantum effects in nuclear degrees of freedom
- Provides both the possible (quantized) energy-levels of the system and the corresponding wave functions
- Solving is difficult for high dimensional systems

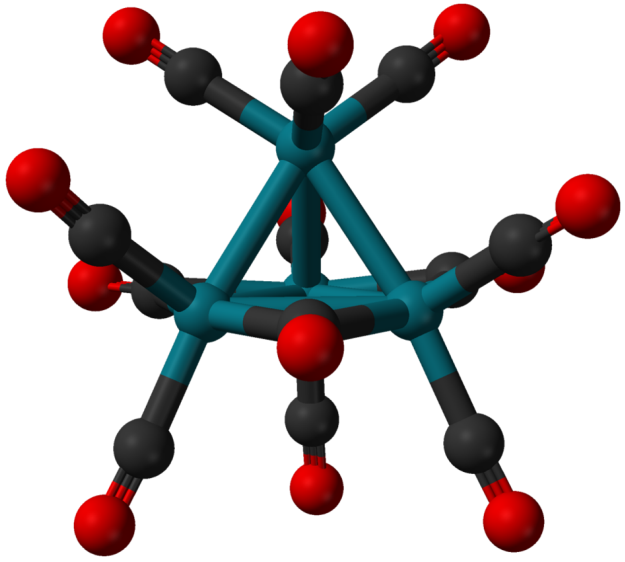
Utilization

- Computations of multi-particle systems' properties (e.g. absorption spectra)
- Preparation of initial conditions for dynamical simulations

Molecular clusters

- Large structures of molecules
- Difficult to classify "manually" (problems consisting of $\sim 10^6$ structures, divided into $\sim 10^3$ patterns)





Other applications

- PES gradients
- Transition dipole moments
- Electronic non-adiabatic couplings

New software?

- NNs applied to molecular physics are a hot research area in last years
- Large dimensionality of problems $\Rightarrow$ Need for HPC implementation
- Fast modifications to the software during research
- Improvements to contemporary optimization methods

- Utilization of other libraries (Boost toolkit...)

Literature review

Mainly framework (MF)

Major implementation work while using framework (IF)

Original code (OC)

PES representations - review

Mainly framework (MF)

- 5x *RuNNer (in development, not public)*
- 1x *ænet*

Major implementation work while using framework (IF)

- 3x Virtual Computational Chemistry Laboratory
- 1x Neural Network Toolbox (Matlab)
- 1x Theano
- 1x sckit-learn

Original code (OC)

- 4x Fortran
- 2x Matlab
- 2x Language not specified

Solving Schrödinger equation - review

Mainly framework (MF)

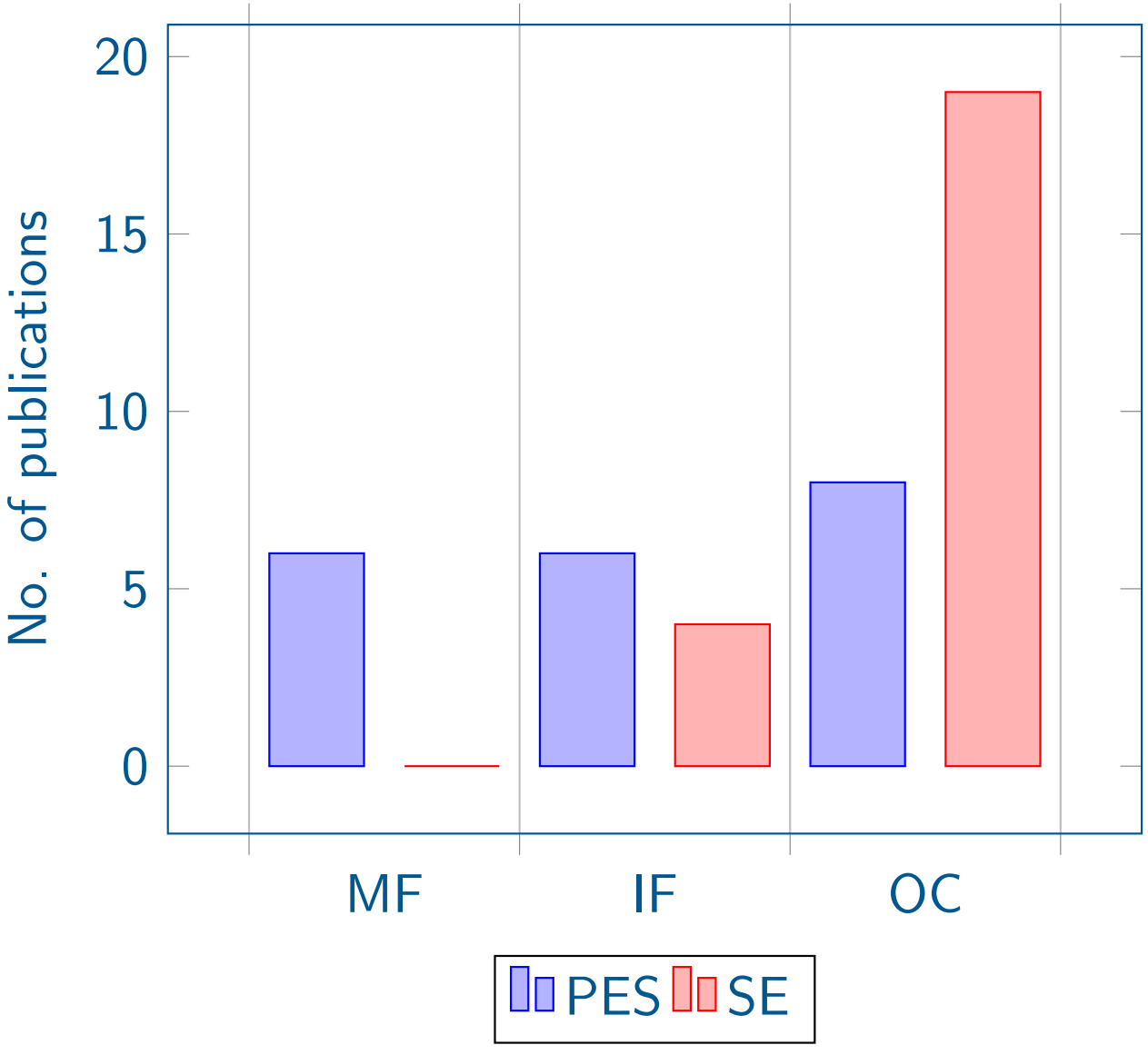
Major implementation work while using framework (IF)

- 1x Neural Network Toolbox (Matlab)
- 1x TensorFlow
- 1x Merlin
- 1x Merlin/MCL

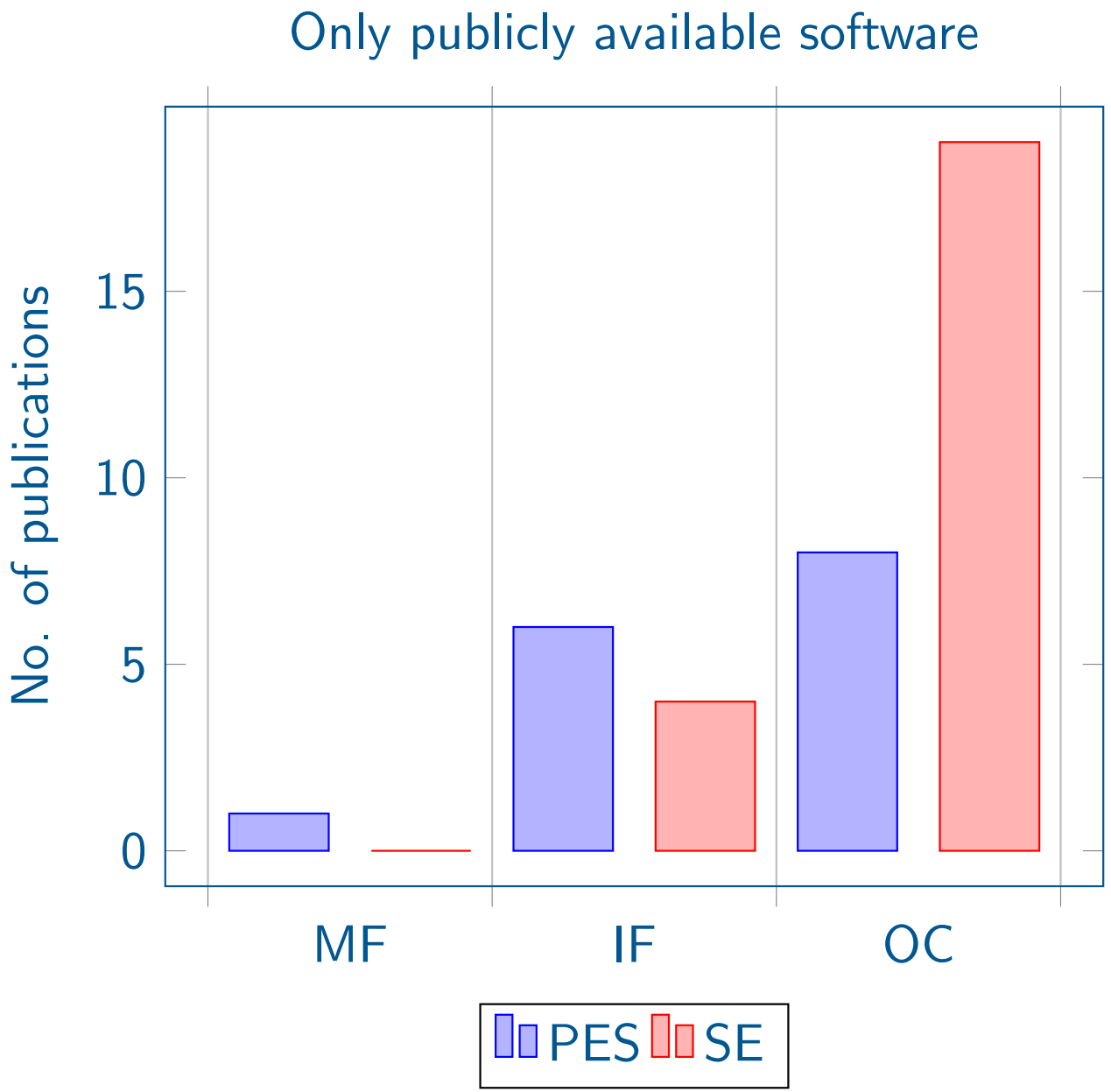
Original code (OC)

- 9x Language not specified
- 3x C++
- 4x Fortran
- 2x Matlab
- 1x Wolfram Mathematica

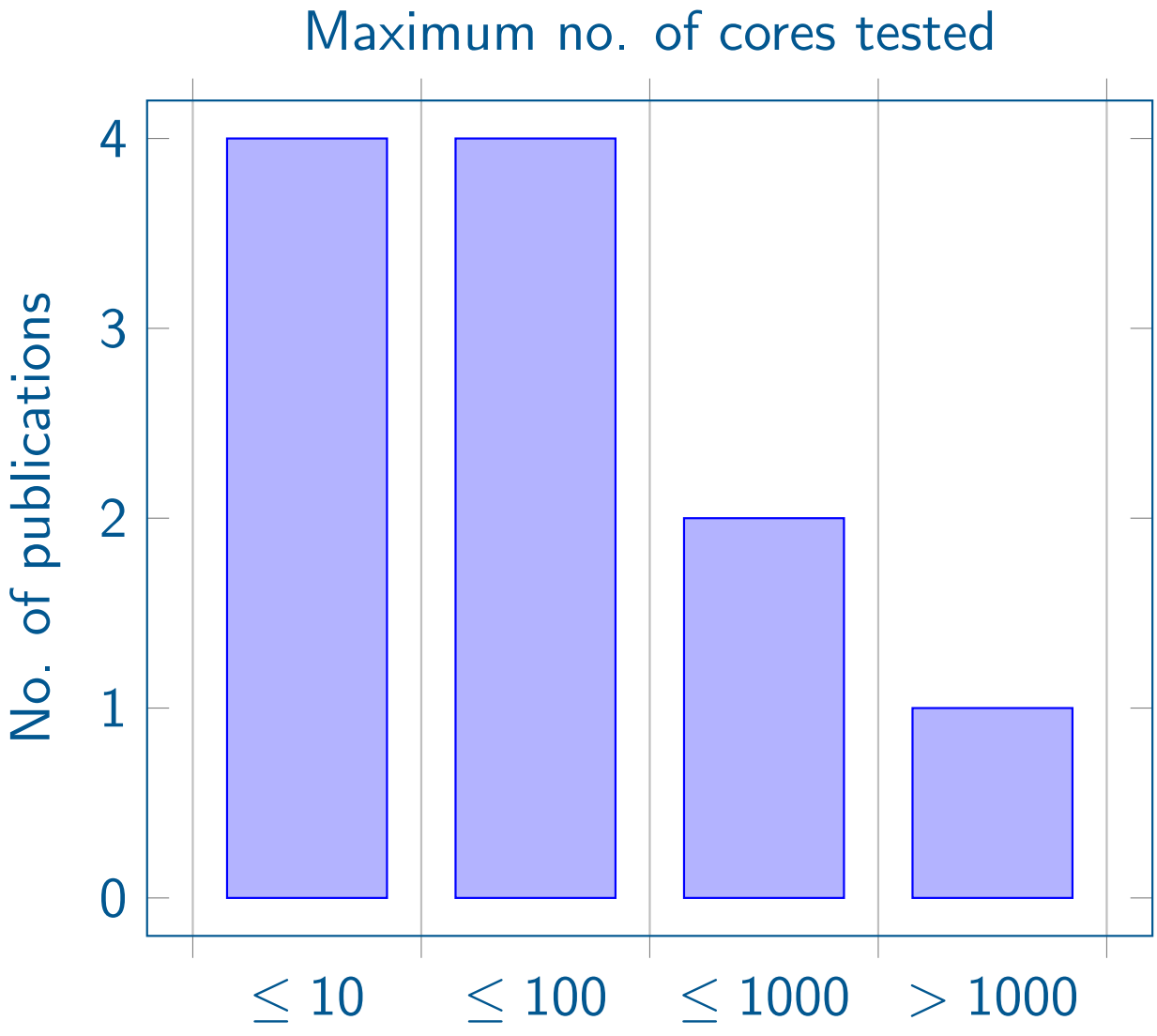
Literature review



Literature review



Literature review



lib4neuro library

Current capabilities

- Training of generally structured (incomplete, no layers) feed-forward networks
- Training of sets of neural networks with shared weights
- Using functions of "atomic" weights as new connection weights

- API for solving differential equations

ODE example[1]

$$\ddot{y}(t) + 4\dot{y}(t) + 4y(t) = 0, \quad t \in [0, 4]$$
$$y(0) = 1$$
$$\dot{y}(0) = 1$$

Analytical solution

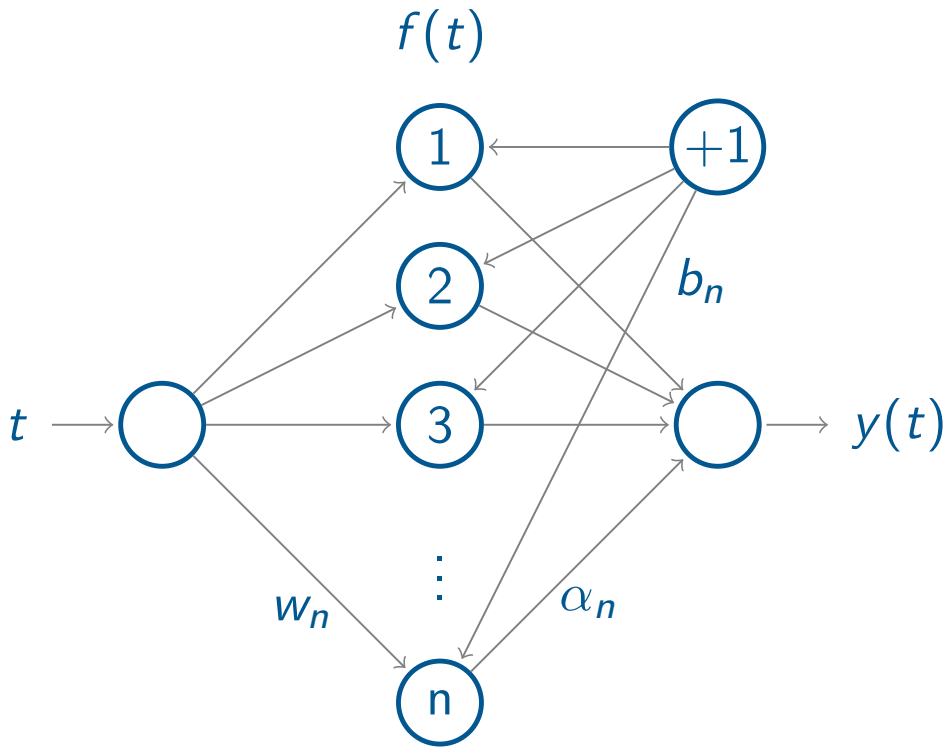
$$\boxed{y(t) = e^{-2t}(3t + 1)}$$

[1] Aarts, Lucie P., and Peter Van Der Veer. "Neural network method for solving partial differential equations." Neural Processing Letters 14.3 (2001): 261-271.

ODE example

Neural network solution

$$y(t) = \sum_{i=1}^{n} \alpha_i \overbrace{\left(1 + e^{-tw_i + b_i}\right)^{-1}}^{f(t)}$$

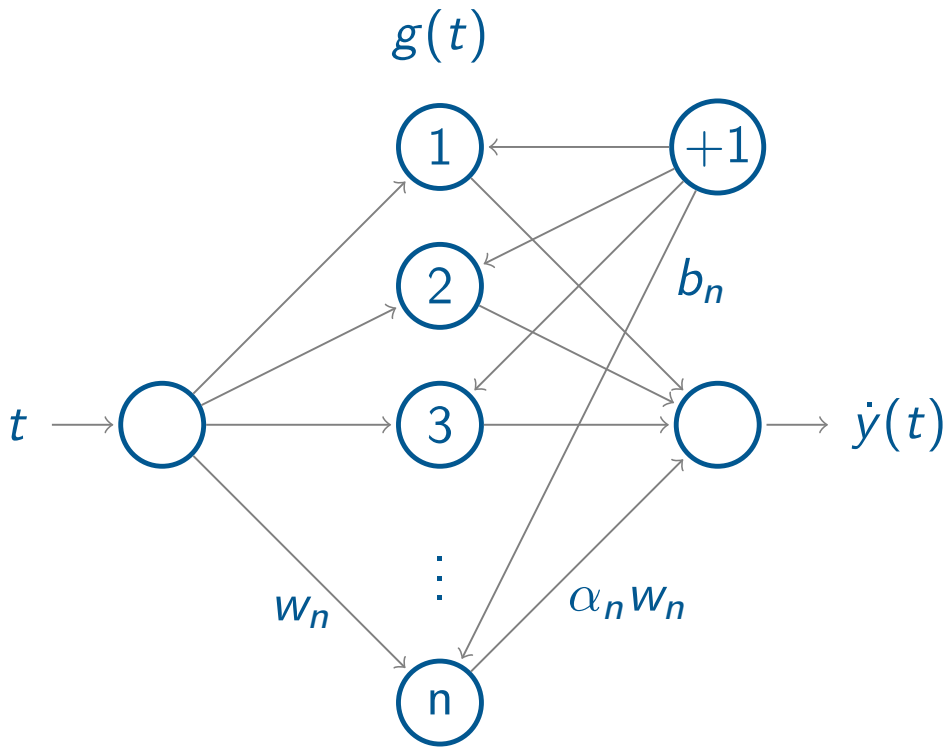


ODE example

Neural network solution

$$y(t) = \sum_{i=1}^{n} \alpha_i \overbrace{(1 + e^{-tw_i + b_i})^{-1}}^{f(t)}$$

$$\dot{y}(t) = \sum_{i=1}^{n} \alpha_i \overbrace{\frac{d}{dt} f_i(t)}^{w_i g_i(t)}$$



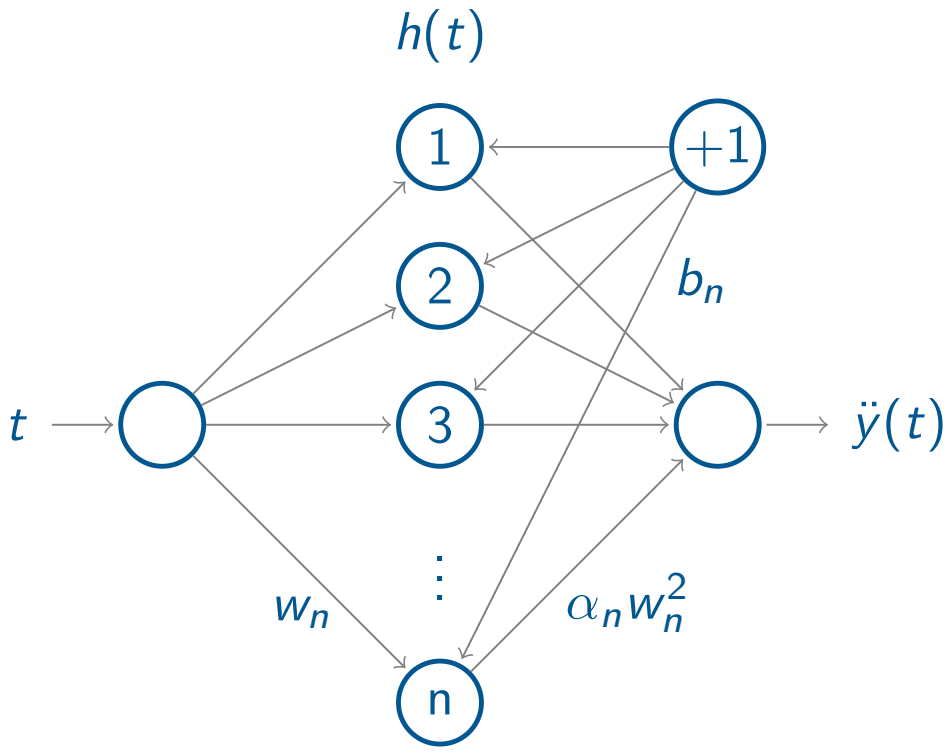
ODE example

Neural network solution

$$y(t) = \sum_{i=1}^{n} \alpha_i \overbrace{(1 + e^{-tw_i + b_i})^{-1}}^{f(t)}$$

$$\dot{y}(t) = \sum_{i=1}^{n} \alpha_i \overbrace{\frac{d}{dt} f_i(t)}^{w_i g_i(t)}$$

$$\ddot{y}(t) = \sum_{i=1}^{n} \alpha_i w_i \underbrace{\frac{d}{dt} g_i(t)}_{w_i h_i(t)}$$



ODE example

Neural network solution

$$y(t) = \sum_{i=1}^{n} \alpha_i \overbrace{(1 + e^{-tw_i + b_i})^{-1}}^{f(t)}$$

$$\dot{y}(t) = \sum_{i=1}^{n} \alpha_i \overbrace{\frac{d}{dt} f_i(t)}^{w_i g_i(t)}$$

$$\ddot{y}(t) = \sum_{i=1}^{n} \alpha_i w_i \underbrace{\frac{d}{dt} g_i(t)}_{w_i h_i(t)}$$

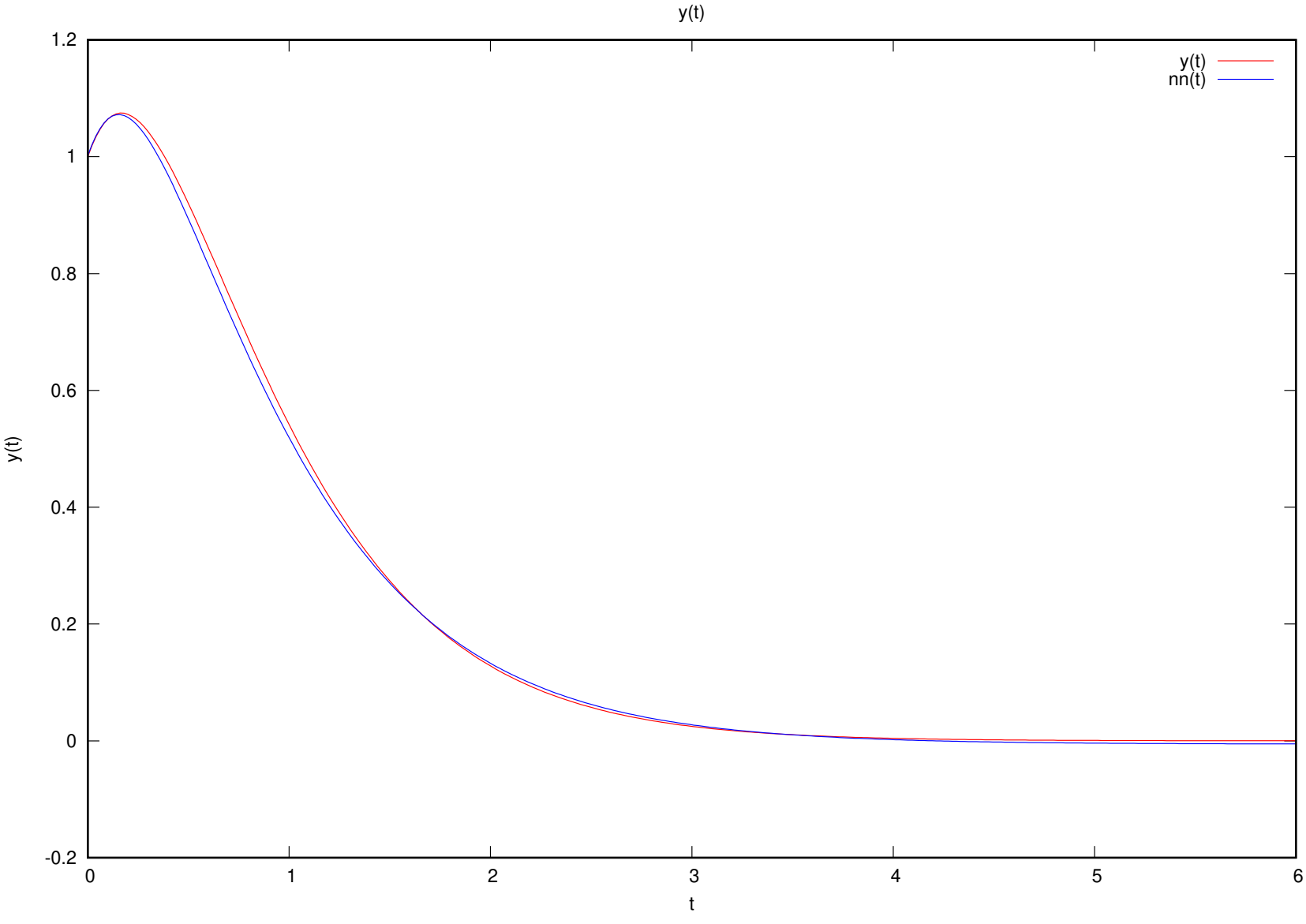
$$e_1 = \frac{1}{|S|} \sum_{t \in S \subset [0,4]} (\ddot{y}(t) + 4\dot{y}(t) + 4y(t))^2$$

$$e_2 = (y(0) - 1)^2$$

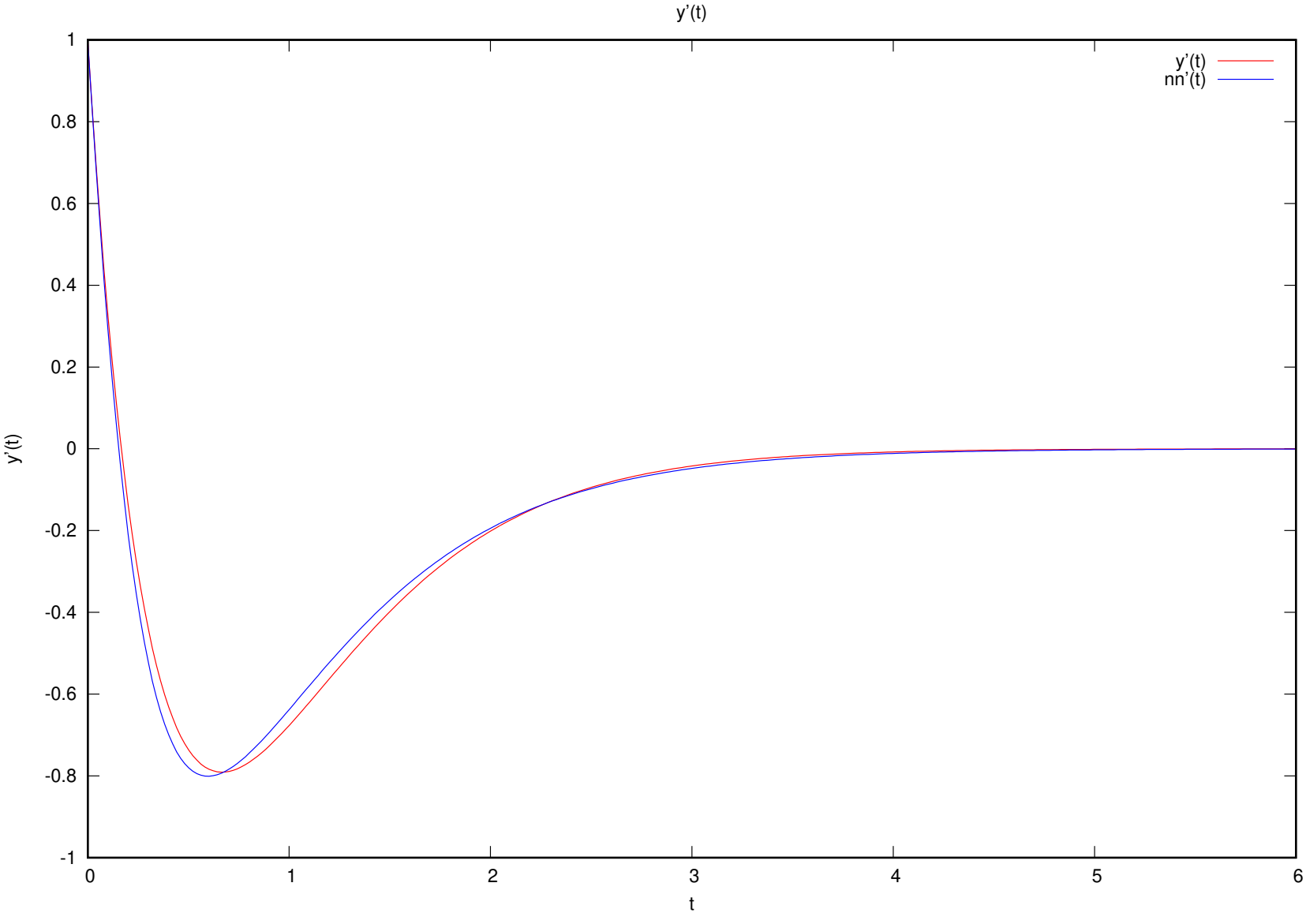
$$e_3 = (\dot{y}(0) - 1)^2$$

$$\boxed{e = e_1 + e_2 + e_3}$$

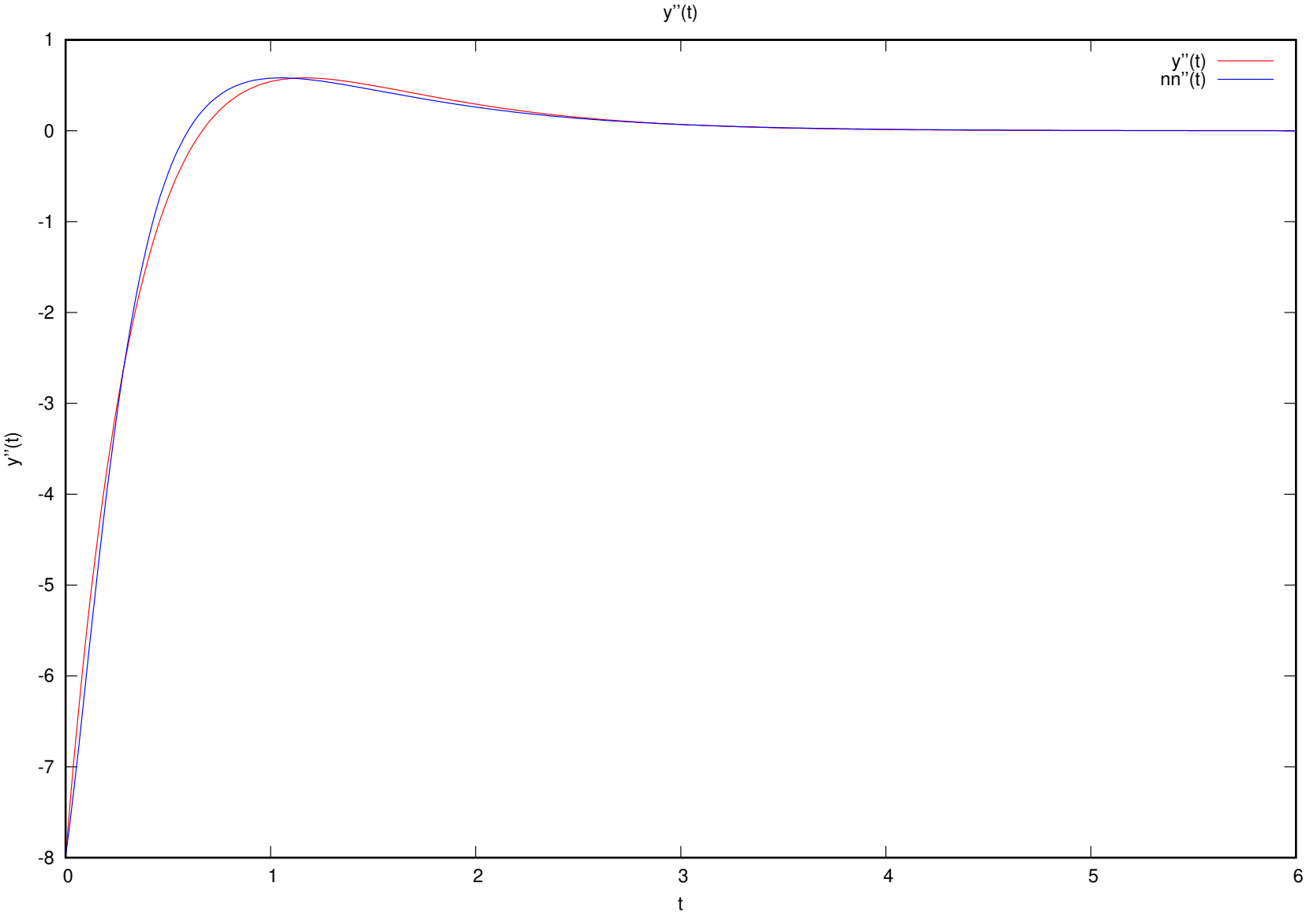
ODE example



ODE example



ODE example



PDE example[1]

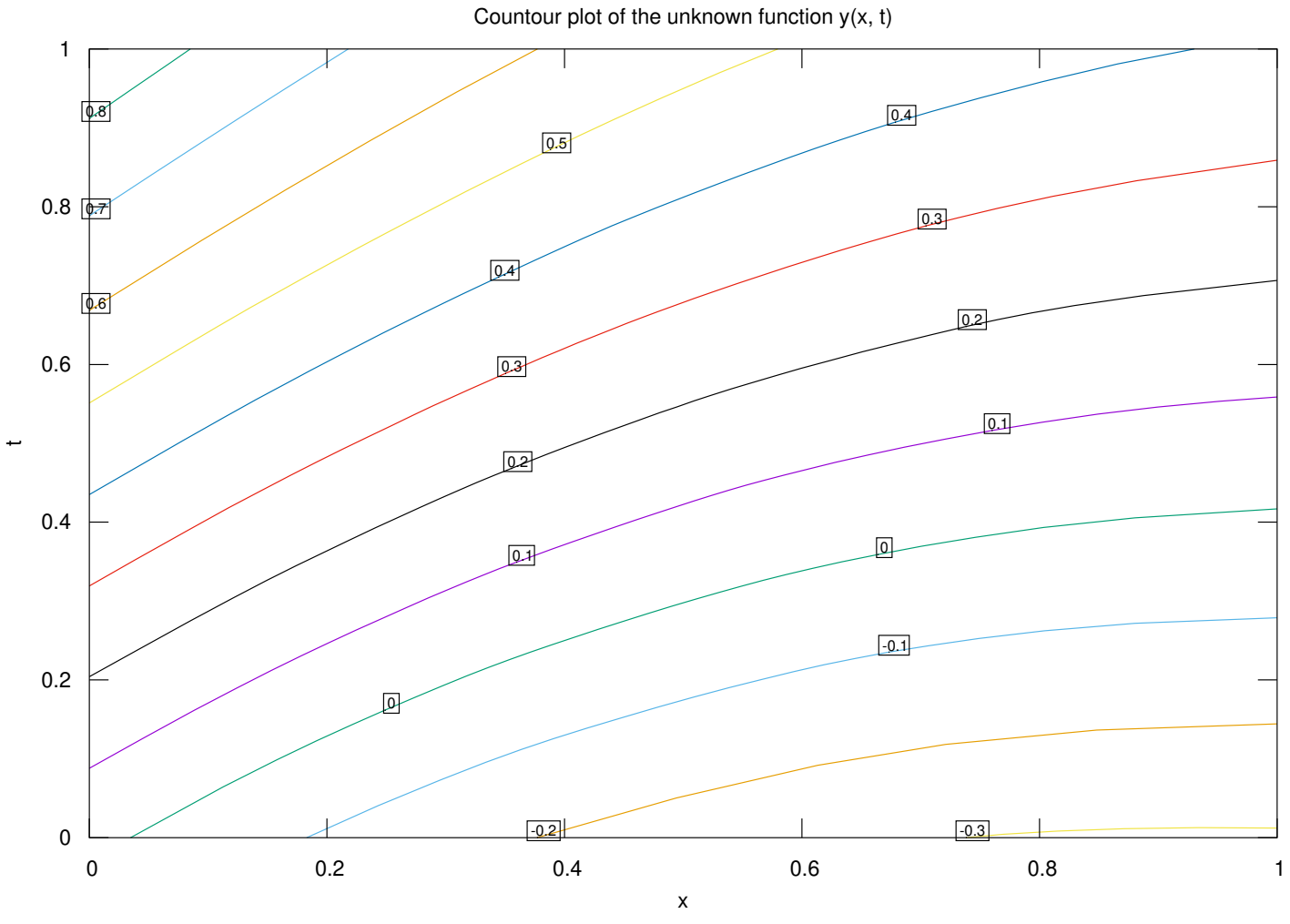
$$\frac{d^2}{dx^2}y(x,t) - \frac{d}{dt}y(x,t) = 0, \quad (x,t) \in [0,1] \times [0,1]$$

$$y(0,t) = \sin(t)$$

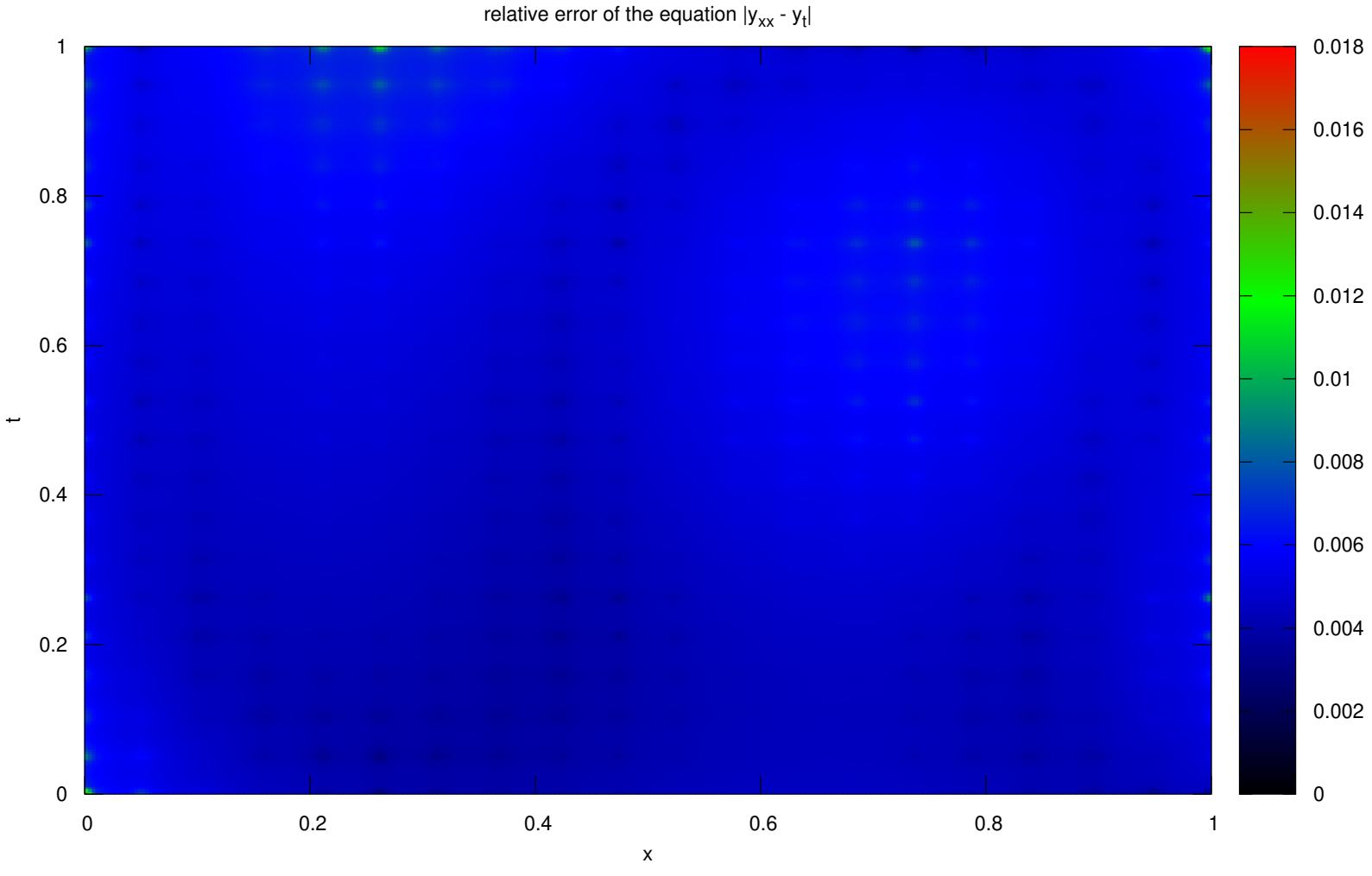
$$y(x,0) = e^{-x\sqrt{0.5}} \cdot \sin(-x\sqrt{0.5})$$

[1] Aarts, Lucie P., and Peter Van Der Veer. "Neural network method for solving partial differential equations." Neural Processing Letters 14.3 (2001): 261-271.

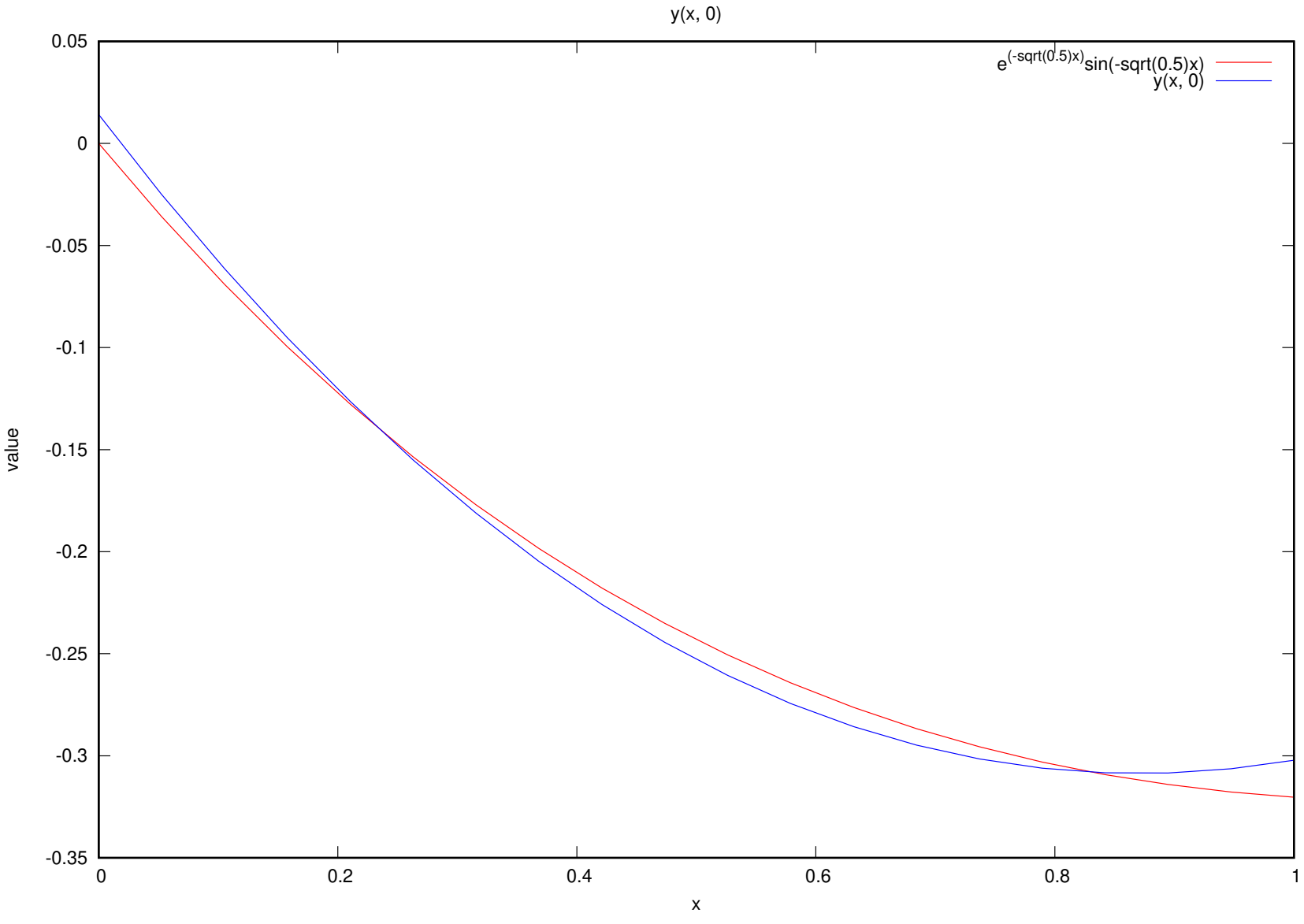
PDE example


Countour plot of the unknown function y(x, t)
t
x
0
0.2
0.4
0.6
0.8
1
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3

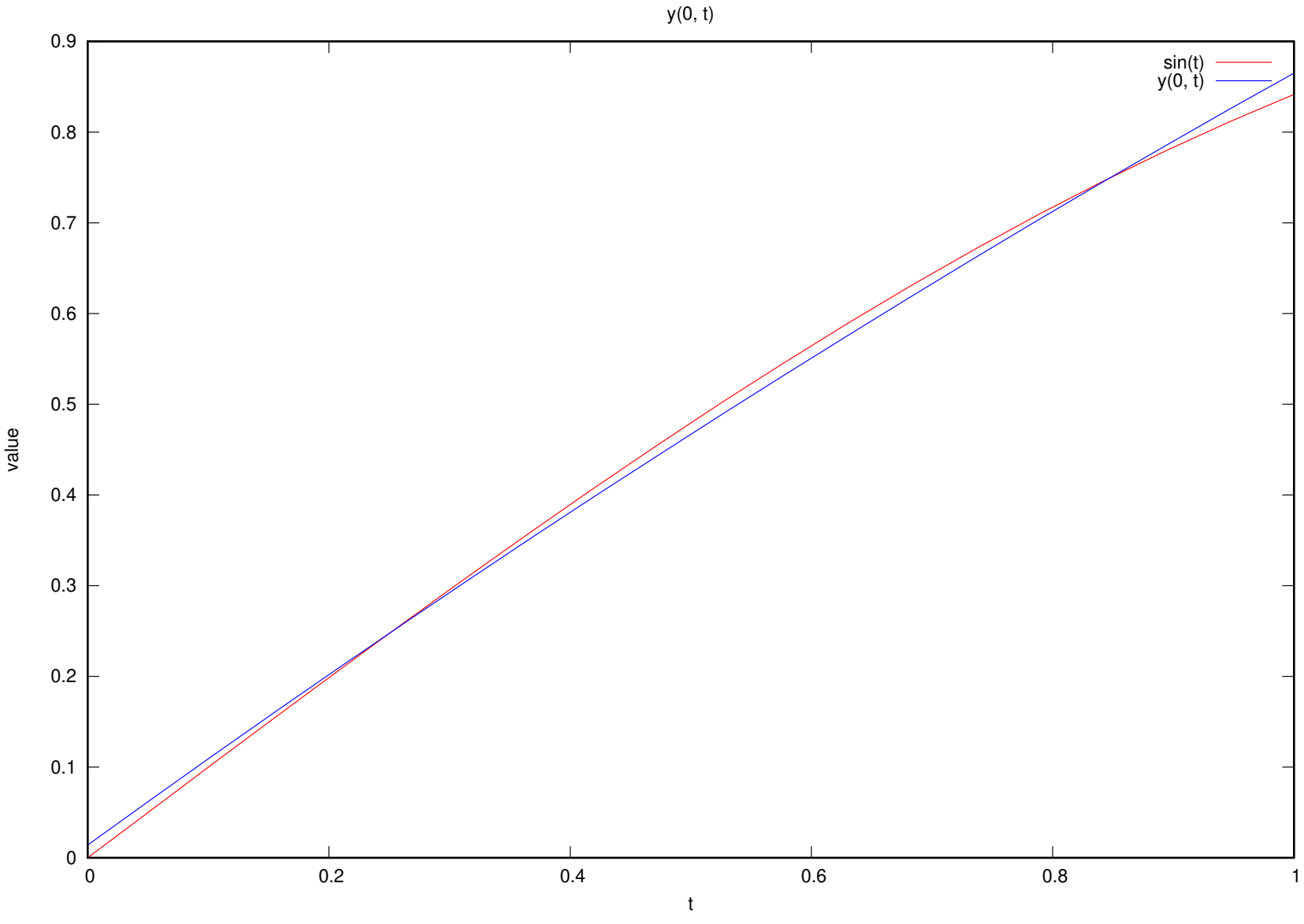
PDE example


relative error of the equation |y_xx - y_t|
t
x
0
0.2
0.4
0.6
0.8
1
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018

PDE example


y(x, 0)
e^(-sqrt(0.5)x)sin(-sqrt(0.5)x)
y(x, 0)
value
x
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
0
0.2
0.4
0.6
0.8
1

PDE example



Current challenges

- HPC implementation (MPI & DASH)
- Development of efficient training methods based on stochastic global optimization approaches

- APIs for specific library applications (PESs representations, Structure classification...)

- Fortran API implementation
- R API implementation

Many thanks to

Martin Mrovec

Michal Kravčenko

David Vojtek

Stanislav Paláček

Libuše Horáčková

Rajko Ćosić

for collaboration on this presentation.

With special thanks also to René Kalus for valuable remarks.

git@code.it4i.cz:moldyn/lib4neuro.git