Symbolic Regression for Reinforcement Learning and Dynamic System Modeling

Robert Babuška

TU Delft

Research interests

- Clustering for building locally linear models
- Reinforcement learning for continuous dynamic systems
    - Neural networks, deep learning
    - Genetic programming, symbolic regression
- Applications in robotics and motion control

TU Delft

Deep reinforcement learning

+ Excellent for state representation using high-dimensional input
- Many hyper-parameters to tune
- Unpredictable and difficult to reproduce
- High computational costs

Useful to investigate other representations!

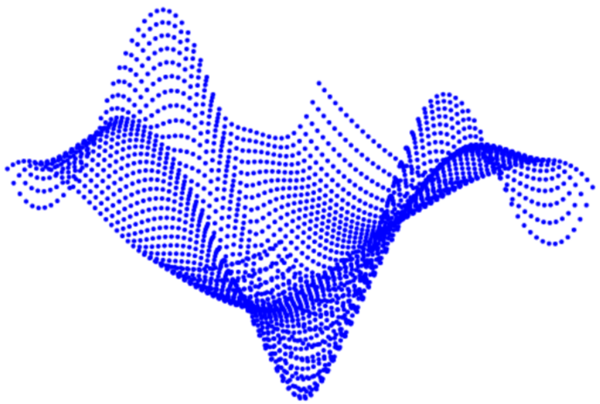
Genetic programming and symbolic regression are tools that definitely deserve more attention.

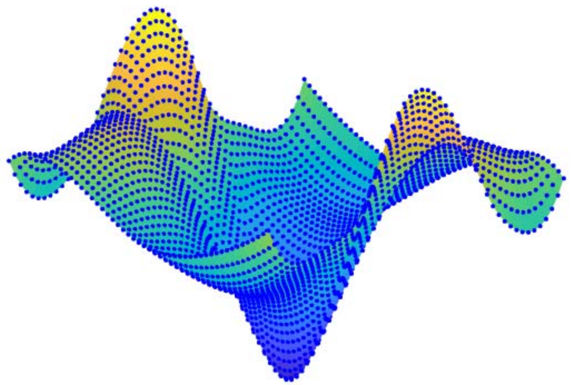
TU Delft

Genetic Programming, Symbolic Regression

TU Delft

Symbolic Regression



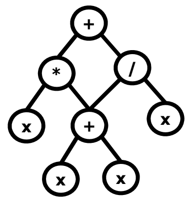


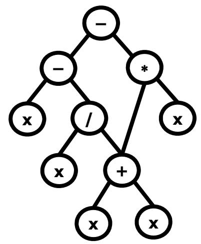
-3.141592654	-30	-23.34719731
-2.932153143	-30	-22.67195916
-2.722713633	-30	-22.07798667
-2.513274123	-30	-21.63117778
-2.303834613	-30	-21.2992009
...	...	...

f = -15.42978401 + 2.42980826 * ((x1 – (x1 * -1.49416733 + x2 * 0.51196778 + 0.00000756)) + (sqrt(power((x1 – (x1 * -1.49416733 + x2 * 0.51196778 + 0.00000756)), 2) + 1) – 1) / 2) ...

Symbolic Regression Algorithms

$$y = \sum_{j=0}^{n_f} \alpha_j F_j(x_1, \dots, x_n)$$





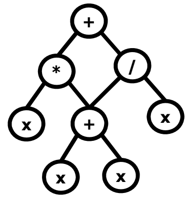
- Multiple Regression Genetic Programming [1]
- Evolutionary Feature Synthesis [2]
- Multi-Gene Genetic Programming [3]
- Single Node Genetic Programming [4, 5]

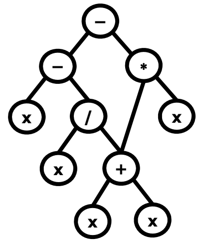
- [1] I. Arnaldo et al.: Multiple regression genetic programming (2014)
- [2] I. Arnaldo et al.: Building predictive models via feature synthesis (2015)
- [3] M. Hinchliffe et al.: Modelling chemical process systems using a multi-gene genetic programming algorithm (1996)
- [4] D. Jackson: Single node genetic programming on problems with side effects (2012)
- [5] J. Kubalík et al.: An improved Single Node Genetic Programming for symbolic regression (2015)

TU Delft

Symbolic Regression Algorithms

$$y = \sum_{j=0}^{n_f} \alpha_j F_j(x_1, \dots, x_n)$$





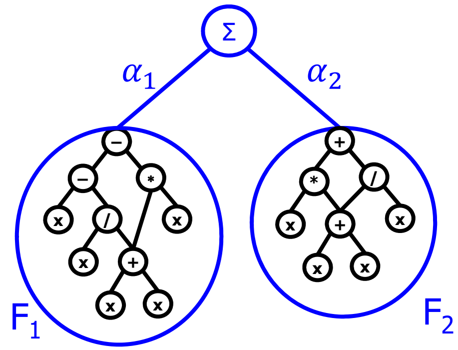
- Multiple Regression Genetic Programming [1]
- Evolutionary Feature Synthesis [2]
- **Multi-Gene Genetic Programming (MGGP)** [3]
- **Single Node Genetic Programming (SNGP)** [4, 5]

- [1] I. Arnaldo et al.: Multiple regression genetic programming (2014)
- [2] I. Arnaldo et al.: Building predictive models via feature synthesis (2015)
- [3] M. Hinchliffe et al.: Modelling chemical process systems using a multi-gene genetic programming algorithm (1996)
- [4] D. Jackson: Single node genetic programming on problems with side effects (2012)
- [5] J. Kubalík et al.: An improved Single Node Genetic Programming for symbolic regression (2015)

TU Delft

Basic SNGP

$$M = \sum_{j=0}^{n_f} \alpha_j F_j(x_1, \ldots, x_n)$$



id:	0	1	2	3	4	5	6	7	8	9	10	11
u	1	2	x_1	x_2	+	-	*	/	+	-	I_1	I_2
Succ	-	-	-	-	1,2	0,1	1,3	3,4	2,5	1,4	7	3
	consts		vars		function nodes						identity nodes	

J. Kubalík et al.: Hybrid single node genetic programming for symbolic regression (2016)

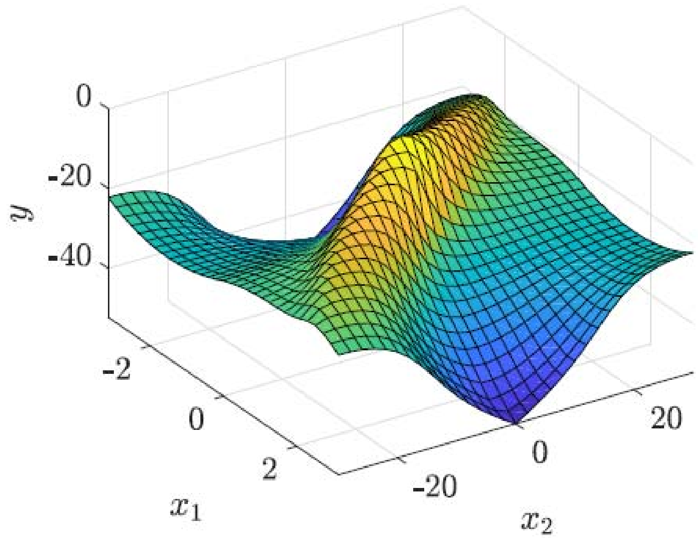
Modifications and extensions

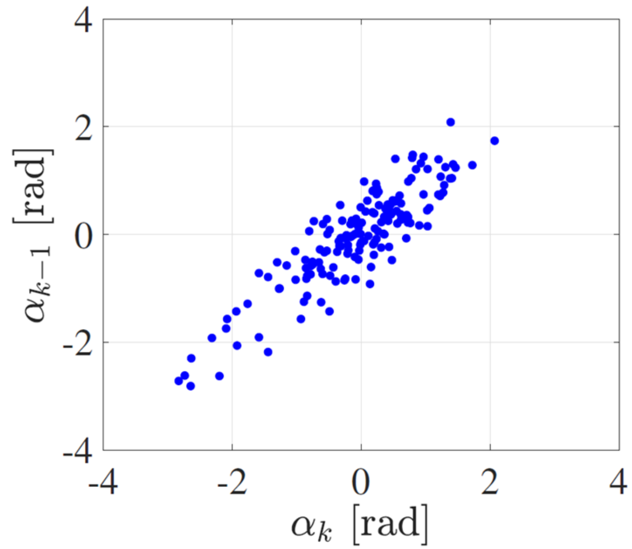
- SNGP and MGGP with affine transformation of input variables [1,2]
- MGGP: Backpropagation for model tuning and tracking dynamic data [2]
- SNGP with partitioned population [3]
- Multi-objective SNGP [4]

- [1] J. Kubalík et al.: Enhanced Symbolic Regression Through Local Variable Transformations (2017)
- [2] J. Žegklitz, P. Pošík: Symbolic Regression in Dynamic Scenarios with Gradually Changing Targets (2019)
- [3] Alibekov et al.: Symbolic Method for Deriving Policy in Reinforcement Learning (2016).
- [4] J. Kubalík et al.: Learning Accurate Robot Models via Combination of Prior Knowledge and Data (submitted, 2019)

TU Delft

Affine transformation of inputs: motivation





Extended SNGP population

Standard SNGP:

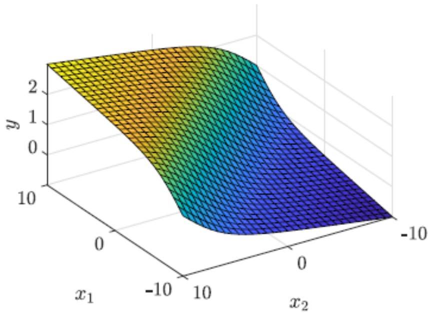
consts	vars	function nodes	identity nodes

Partitioned population and transformed inputs:

consts	head function nodes	vars	transformed vars	tail function nodes	identity nodes

TU Delft

Benefits of transformed inputs



$$f(x_1, x_2) = 0.1(0.5x_1 + 0.5x_2) + \frac{2}{1 + e^{-(0.5x_1 + 0.5x_2)}}$$

Original SNGP:

f = 1.27297628 * sigmoid(x1 + x2 – 0.0625 * x1) – 0.38266172 * (power((0.0625 * x1), 3) – (0.22340393 * ((x1 + x2) – (0.0625 * x1)))) – 2.7355E-4 * ((power(x1, 2) * x2 – x1 – (30.25 * (x1 + sigmoid(x2))))) + 0.35937439

RMSE = 5.78E-2

Transformed input variables:

f = -2.6 + 0.1 * (36.0 + v1) – 2.0 * (0.5 – sigmoid(v1)) – 9.0E-8 * (sigmoid(v2 – 81.0) * 0.00195313)

v1 = 0.5 * x1 + 0.5 * x2

v2 = 0.07105142 * x1 + 0.07105142 * x2 + 4.24664016

RMSE = 6.31E-10

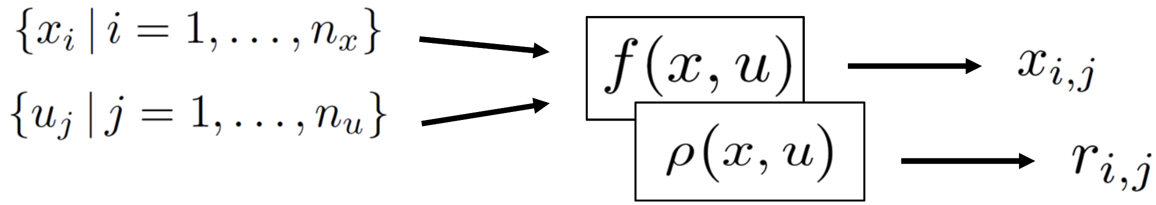
TU Delft

Solving Bellman equation via genetic programming

Solve Bellman equation by using GP

$$V(x) = \max_{u \in \mathcal{U}} \left[\rho(x,u) + \gamma V\big(f(x,u)\big) \right]$$

Generate data:



Bellman equation in terms of the data:

$$V(x_i) = \max_{j} \left[r_{i,j} + \gamma V(x_{i,j}) \right]$$

TU Delft

Direct solution of Bellman equation

$$V(x_i) = \max_j \left[r_{i,j} + \gamma V(x_{i,j}) \right]$$

Fitness function:

$$J^{\text{direct}} = \frac{1}{n_x} \sum_{i=1}^{n_x} \left[\max_j \left(r_{i,j} + \gamma \underbrace{V(x_{i,j})}_{\text{evolved}} \right) - \underbrace{V(x_i)}_{\text{evolved}} \right]^2$$

Use GP to find a symbolic representation of V

TU Delft

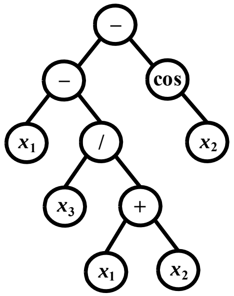
Symbolic value iteration (SVI)

Target data

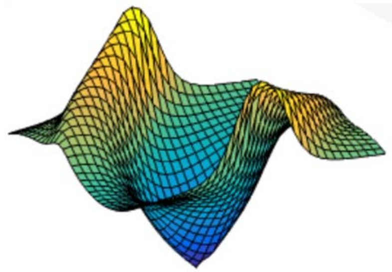
$$t_{i,\ell} = \max_j \big(r_{i,j} + \gamma V_{\ell-1}(x_{i,j}) \big)$$

Symbolic regression

$$J_\ell^{\mathrm{SVI}} = \frac{1}{n_x} \sum_{i=1}^{n_x} \Big[\underbrace{t_{i,\ell}}_{\text{target}} - \underbrace{V_\ell(x_i)}_{\text{evolved}} \Big]^2$$

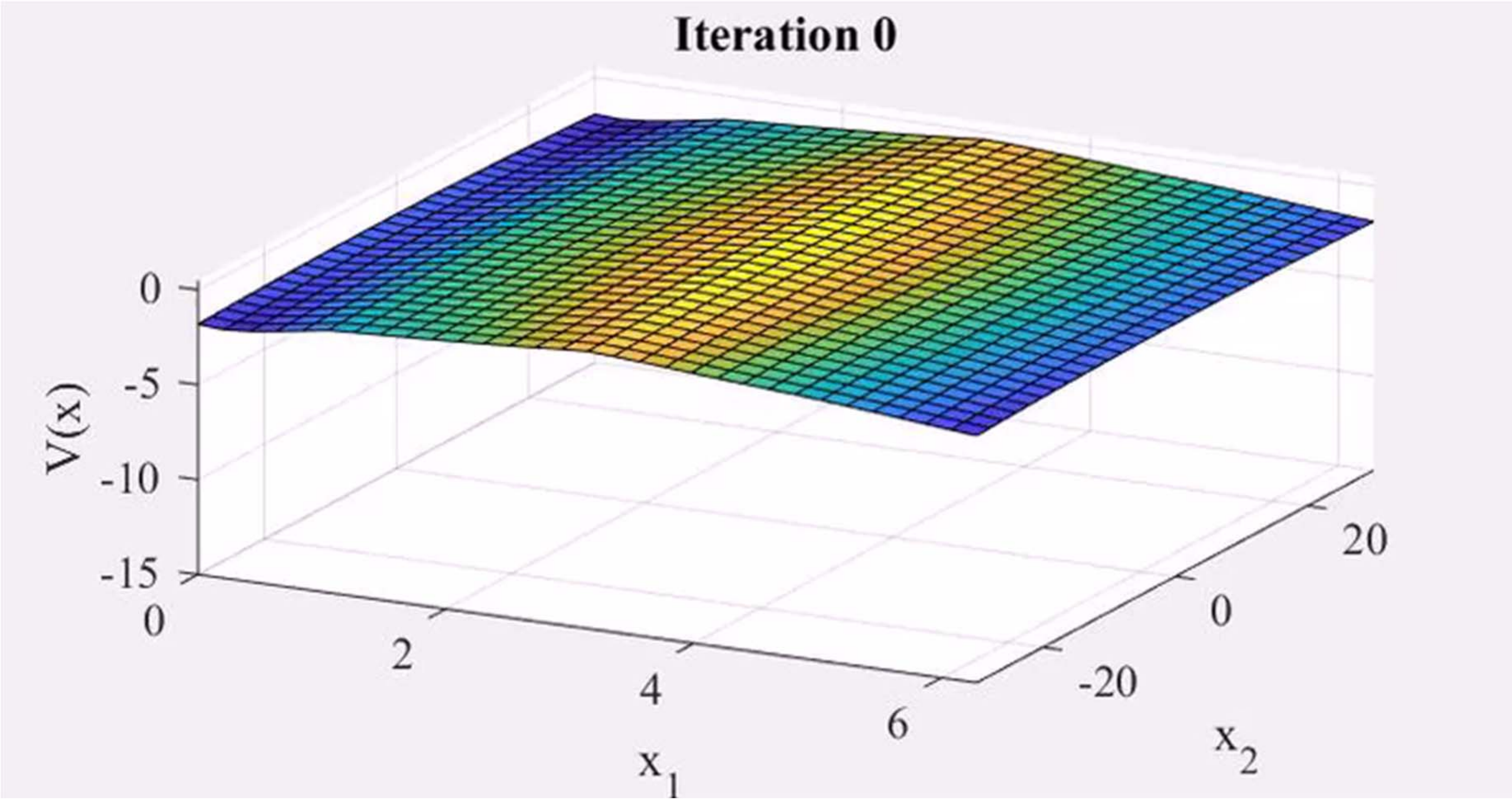


Symbolic V-function from previous iteration

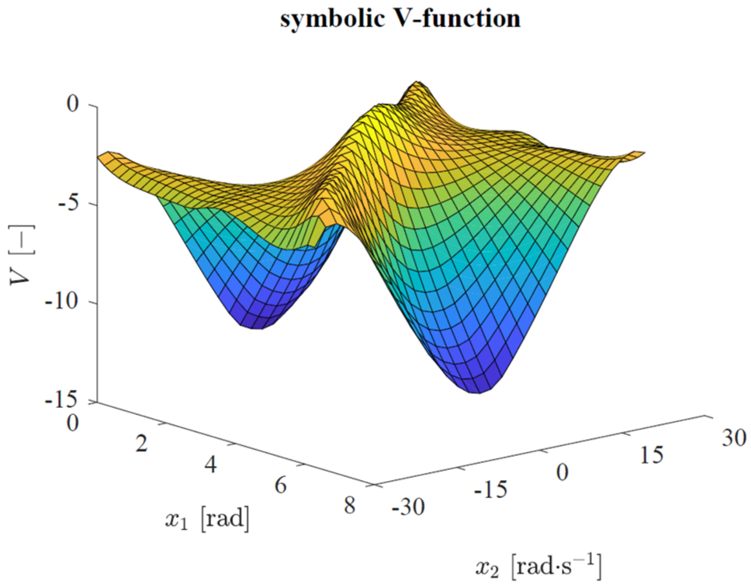


$$V_\ell(x) = 5 * x_2 - 3 * x_3 + \\ + \cos(x_1) - \sin(x_2) \ldots$$

Pendulum swing-up: symbolic value iteration



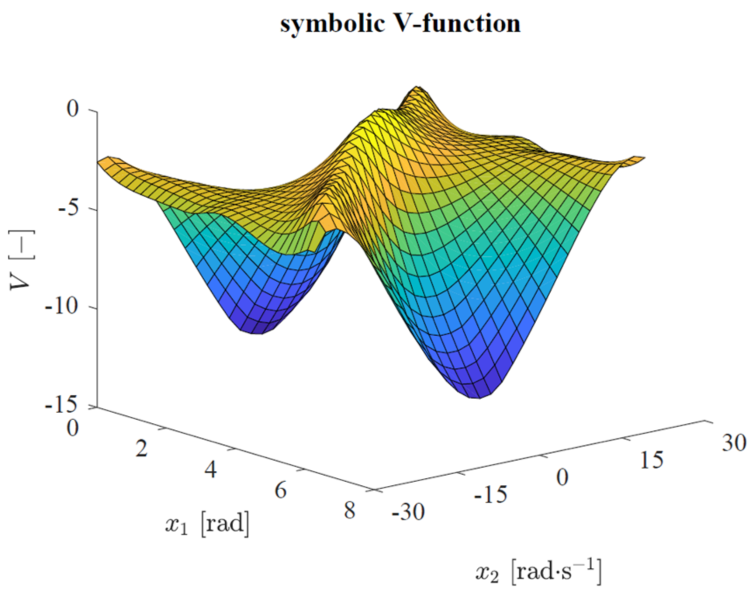
V function for 1-DOF pendulum swing-up



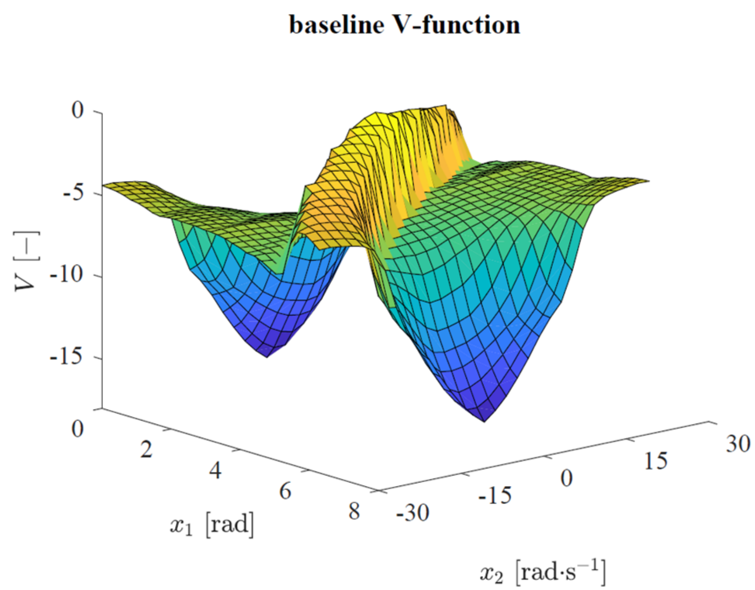
$$
\begin{aligned}
V(x) = {} & 1.7 \times 10^{-5}(10x_2 - 12x_1 + 47)(4.3 \times 10^{-2}x_2 - 3.5x_1 + 11)^3 \\
& - 7.1 \times 10^{-4}x_2 - 4.6x_1 - 8.2 \times 10^{-6}(4.3 \times 10^{-2}x_2 - 3.5x_1 \\
& + 11)^3(0.2x_1 + 0.3x_2 - 0.5)^3 - 9.8 \times 10^{-3}(0.4x_1 + 0.1x_2 - 1.1)^6 \\
& + 11(0.1x_1 - 1.5)^3 + 11((0.6x_1 + 6.3 \times 10^{-2}x_2 - 1.7)^2 + 1)^{0.5} \\
& + 8.7 \times 10^{-6}((10x_2 - 12x_1 + 47)^2(4.3 \times 10^{-2}x_2 - 3.5x_1 + 11)^6 + 1)^{0.5} \\
& + 0.3((1.1x_1 + 0.4x_2 - 3.3)^2 + 1)^{0.5} + (3.9 \times 10^{-3}(4.3 \times 10^{-2}x_2 \\
& - 3.5x_1 + 11)^2(0.2x_1 + 0.3x_2 - 0.5)^2 + 1)^{0.5} + 6.5 \times 10^{-5}((1.2x_1 \\
& + 14x_2 - 10)^2(9.1 \times 10^{-2}x_2 - 2.9x_1 + 0.5((9.1 \times 10^{-2}x_2 - 2.9x_1 \\
& + 8.3)^2 + 1)^{0.5} + 7.8)^2 + 1)^{0.5} - 5.5 \times 10^{-2}(4.3 \times 10^{-2}x_2 \\
& - 3.5x_1 + 11)(0.2x_1 + 0.3x_2 - 0.5) - 1.7((3.6x_1 + 0.4x_2 - 11)^2 + 1)^{0.5} \\
& - 2((x_1 - 3.1)^2 + 1)^{0.5} - 1.3 \times 10^{-4}(1.2x_1 + 14x_2 - 10)(9.1 \times 10^{-2}x_2 \\
& - 2.9x_1 + 0.5((9.1 \times 10^{-2}x_2 - 2.9x_1 + 8.3)^2 + 1)^{0.5} + 7.8) + 23\ .
\end{aligned}
$$

89 parameters

V-function for 1-DOF pendulum swing-up



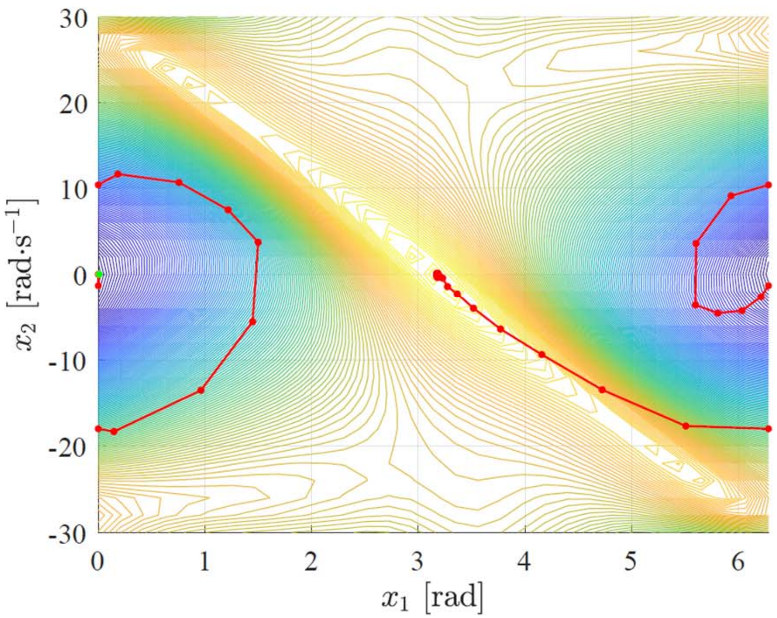
89 parameters



961 parameters

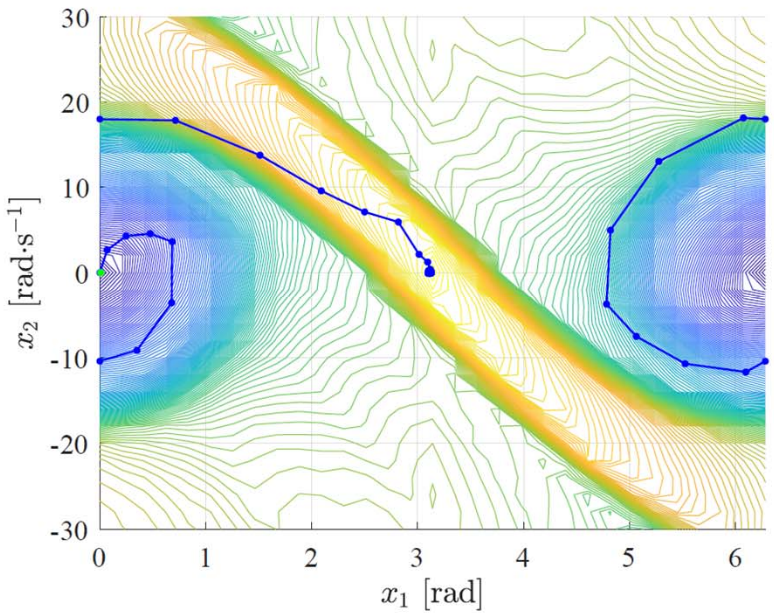
V-function for 1-DOF pendulum swing-up

Symbolic V-function



Smooth swing-up trajectory

Baseline V-function

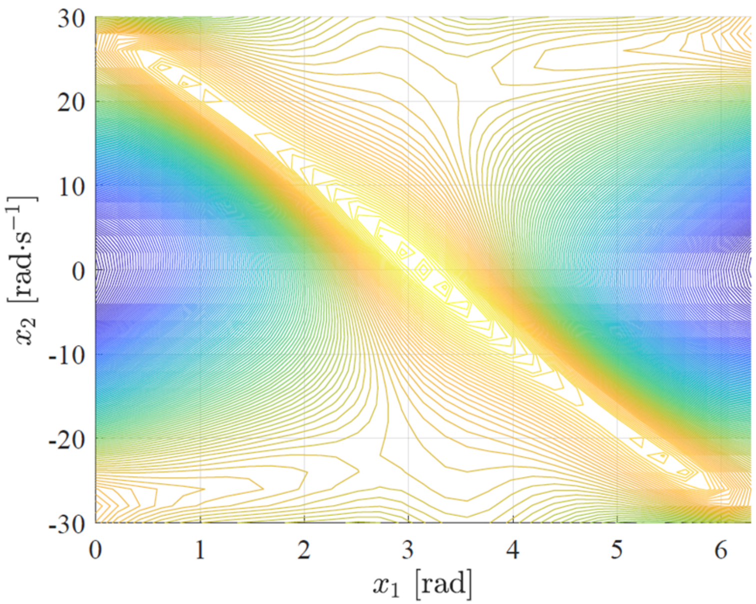


Less smooth trajectory

TU Delft

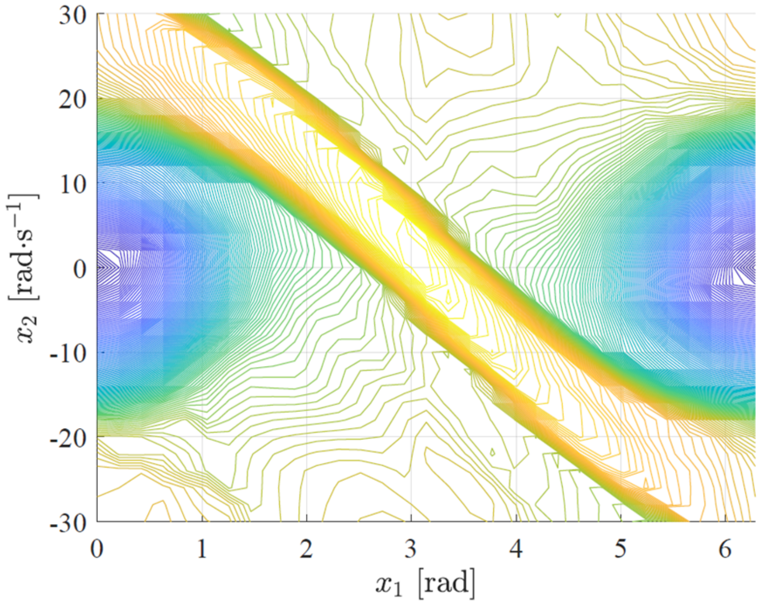
Comparison with a neural network

Symbolic V-function



89 parameters

Neural network V-function

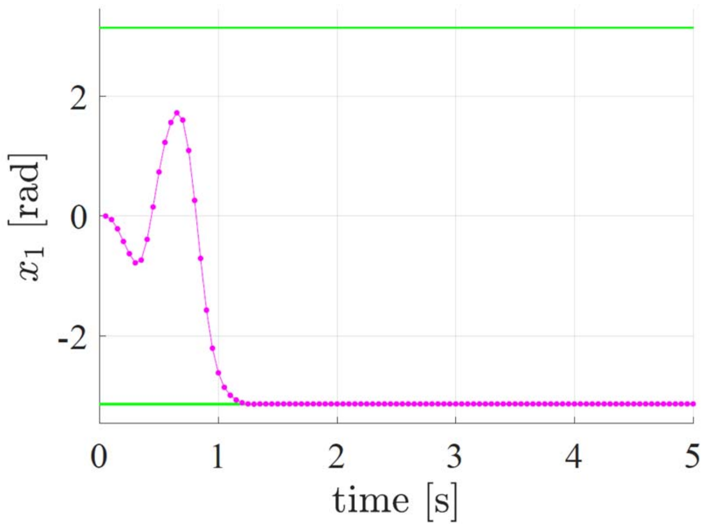


201 parameters

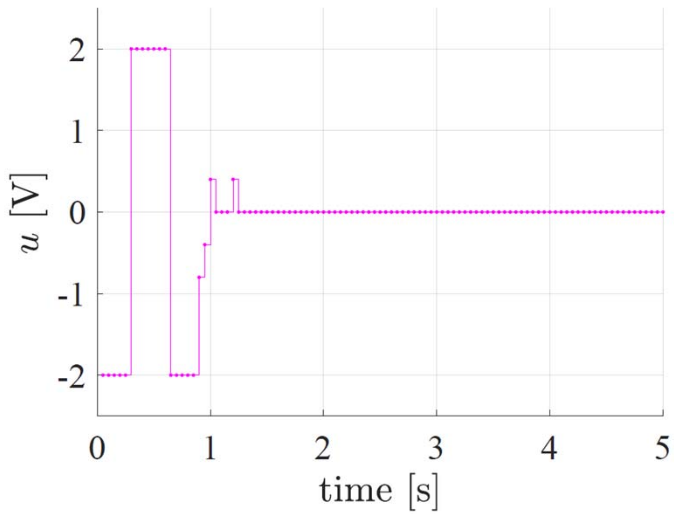
TU Delft

Swing-up experiment on the real system

Pendulum angle



Control action



Performance very close to theoretically optimal bang-bang control

TU Delft

Conclusions on symbolic value functions

- Compact and typically very smooth V-functions. Analytic, can be plugged in other algorithms.
- Near optimal control performance, outperforms other approximators (basis functions, DNN).
- High computational costs, comparable to NN.
- So far tested on systems with a small number of state variables.

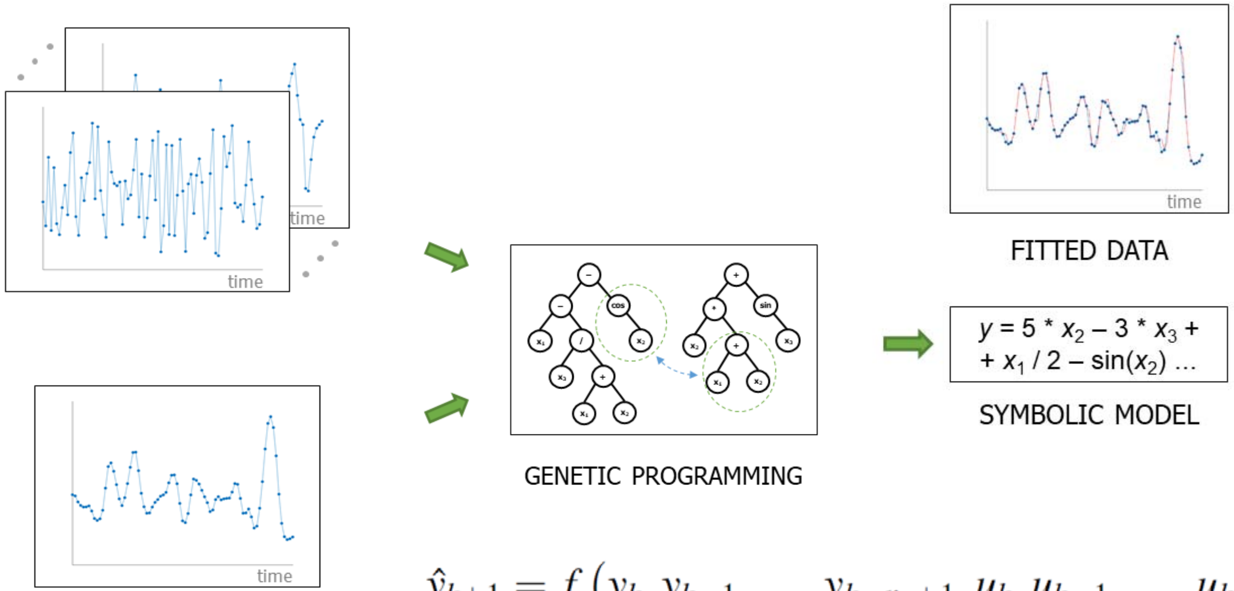
Challenges:
Direct solution, high-dimensional state spaces, convergence guarantees, model-free variant.

TU Delft

Genetic programming for building dynamic models

TU Delft

Symbolic regression for modeling dynamic systems



$$\underbrace{\hat{y}_{k+1}}_{\text{Predicted output}} = f\left(\underbrace{y_k, y_{k-1}, \ldots, y_{k-n_y+1}}_{\text{Past outputs}}, \underbrace{u_k, u_{k-1}, \ldots, u_{k-n_u+1}}_{\text{Past inputs}}\right)$$

Nonlinear autoregressive with exogenous input model (NARX)

TU Delft

Challenges of model building for dynamic systems

- Use short data sequences
- Consistent models of multi-variable systems
- Include prior knowledge
- Automatically select data for updating models
- Model accuracy – complexity tradeoff

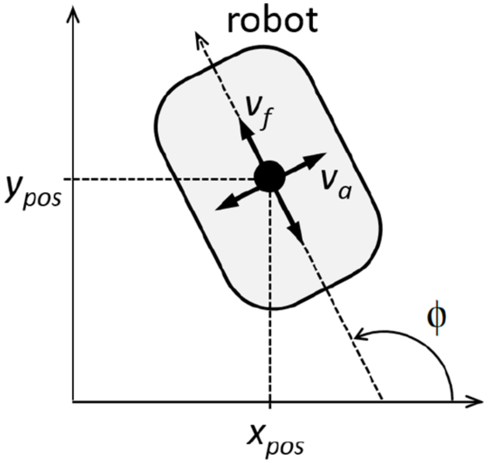
TU Delft

Challenges of model building for dynamic systems

- Use short data sequences
- **Consistent models of multi-variable systems**
- **Include prior knowledge**
- Automatically select data for updating models
- **Model accuracy – complexity tradeoff**

TU Delft

Mobile robot experiments



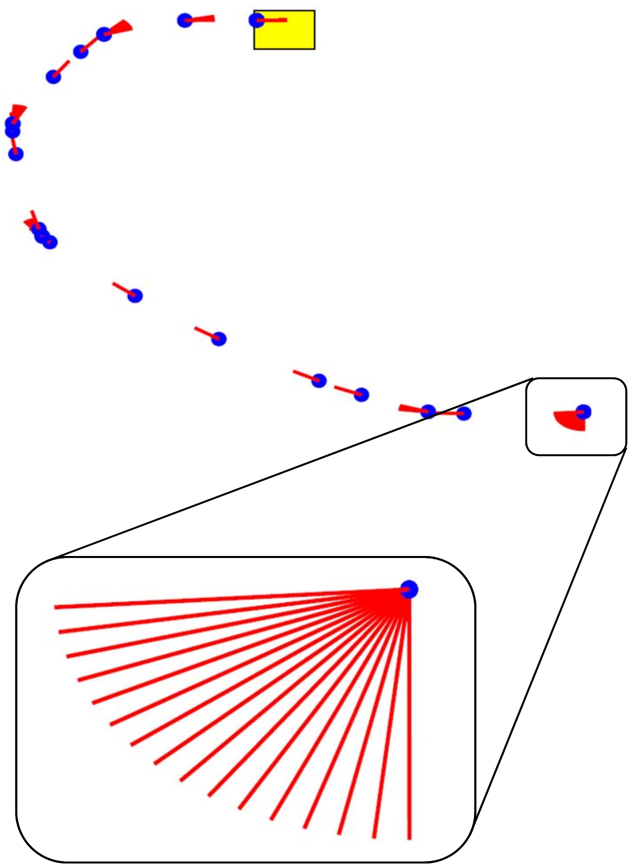
Mechanistic model:

$$\dot{x}_{pos} = v_f \cos(\phi)$$
$$\dot{y}_{pos} = v_f \sin(\phi)$$
$$\dot{\phi} = v_a$$

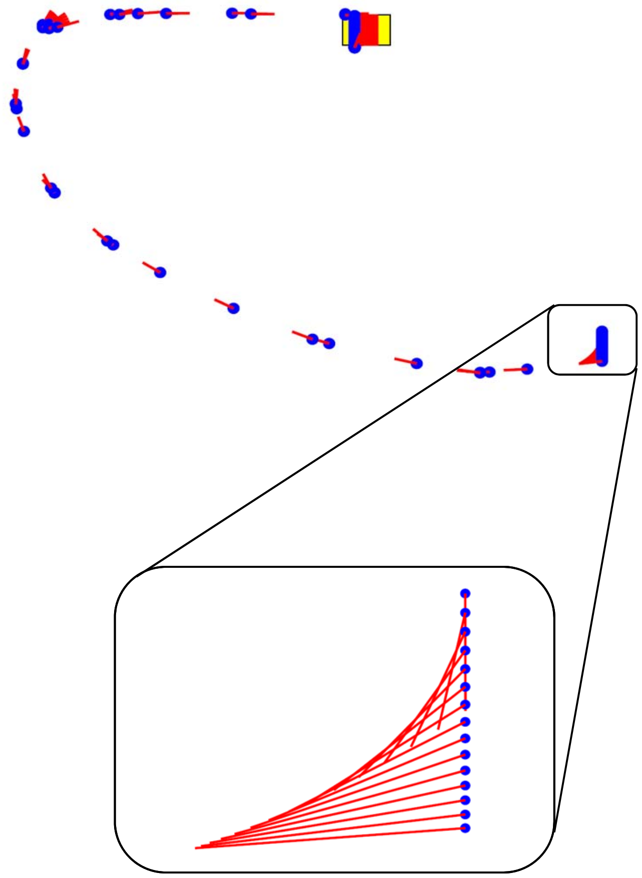
- Mechanistic model correctly represents the physics, but is inaccurate as a prediction model (actuator nonlinearities).
- Data-driven model constructed via symbolic regression is accurate, but does not necessarily respect the physical constraints.

TU Delft

Motion planning with mechanistic model



Motion planning with data-driven model



Solution: include prior knowledge

Generate synthetic data representing physical constraints, use MO GP

Examples:

- Equilibrium under zero input

$$x_0 = f(x_0, 0)$$

- Non-holonomic constraint (robot cannot move sideways)

$$y_{pos} = f_y([x_{pos}, y_{pos}, \phi]^T, [v_f, 0])$$

TUDelft

Conclusions on symbolic model construction

- Accurate and compact models from small data sets
- Model structure can be constrained to a specific model class

Challenges:
Effective incorporation of prior knowledge, computational costs, multi-dimensional models.

TU Delft