# Efficient determination of zero-crossings in noisy real-life time series

Marat S. Mukhametzhanov

# Background

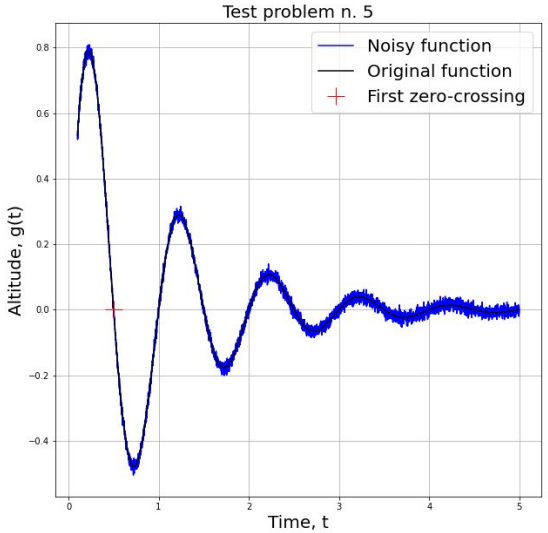
- A difficult nonlinear time series is considered with the objective function *g(t, x(t))*, which depends on time t and additional variables $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, each of them depends on t as well.
- The system variables *x(t)* can be unknown: each concrete value *x(t)* at time *t* can be obtained after a numerical simulation/solution to ODEs/getting the data from sensors, etc.
- So, the function *g(t,x)* is:
  - Black-box (i.e., its analytical representation is unknown to a user)
  - Hard to evaluate (each its evaluation is an expensive operation in computational and/or temporal resources)
  - Noisy (e.g., the values of *x(t)* can arrive with a small random error)
- When *g(t,x)* becomes equal to zero, there can occur a discrete event:
  - The simulated/studied unit can crush/explode (e.g., in case of sensors on a physical object).
  - A discrete interaction can occur (e.g., changing the system's state and/or variables, as in hybrid systems).
- It is important to predict if there will occur a zero-crossing of *g(t,x(t))* at the time steps $t_{k+1}, t_{k+2}, \ldots, t_{k+M}$ following to the current time $t_k$ before the actual data on these time steps arrive.

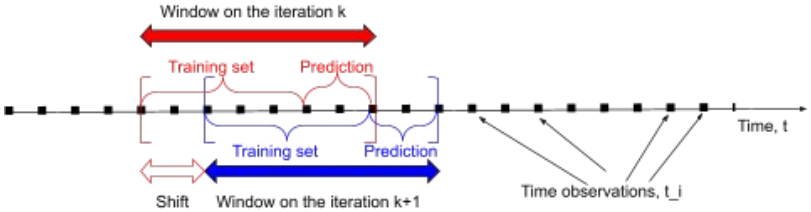
# Problem statement

The first zero-crossing of *g(t)* is determined, $t \in [t_0, T],\ g(t_0) > 0.$

**The main process:**

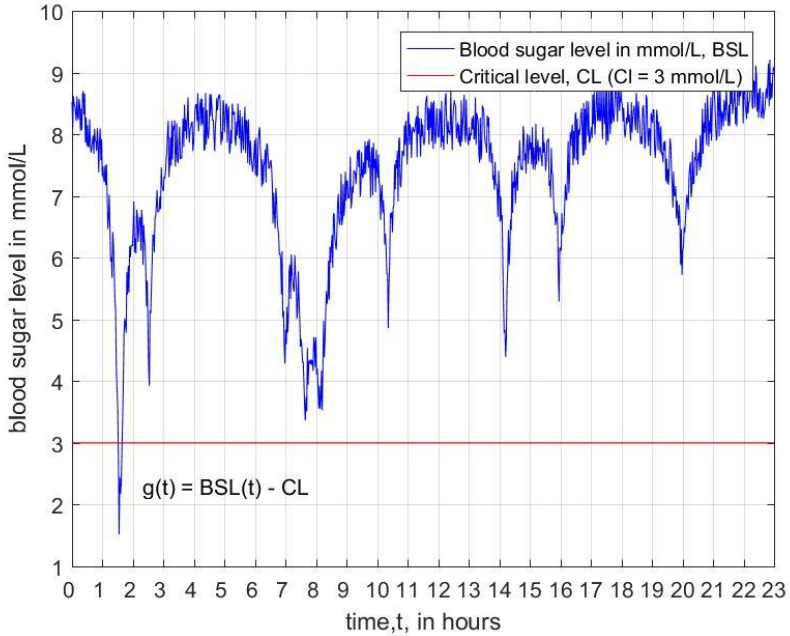
1. Get the first N observations (N=1000 was used): $(t_0, g(t_0)), (t_1, g(t_1)), \ldots, (t_{N-1}, g(t_{N-1})).$
   The system should be "stable" at the initial time: $g(t_0) > 0$
2. Predict M values of the time series (M is adaptively estimated during the search) using the last N observations: $(t_N, \hat{g}(t_N)), (t_{N+1}, \hat{g}(t_{N+1})), \ldots, (t_{N+M-1}, \hat{g}(t_{N+M-1})).$
3. Check the zero-crossings among the predicted observations.
4. Get new data: $(t_N, g(t_N)), (t_{N+1}, g(t_{N+1})), \ldots, (t_{N+M-1}, g(t_{N+M-1})).$
5. Repeat the process taking the last N observations to train the models



# Use case 1: Blood Sugar Levels (BSL) of a diabetic patient

Blood sugar level (BSL) smaller than the critical value CL (e.g., 3 mmol/L) can harm a patient: zero-crossings of g(t) = BSL(t) - CL should be predicted.




Another possibility:
Blood sugar level (BSL) smaller than CL1 (e.g., 3 mmol/L) or higher than CL2 (e.g., 30 mmol/L) can harm a patient: zero-crossings of g(t) = (BSL(t) - CL1)*(CL2-BSL(t)) should be predicted.

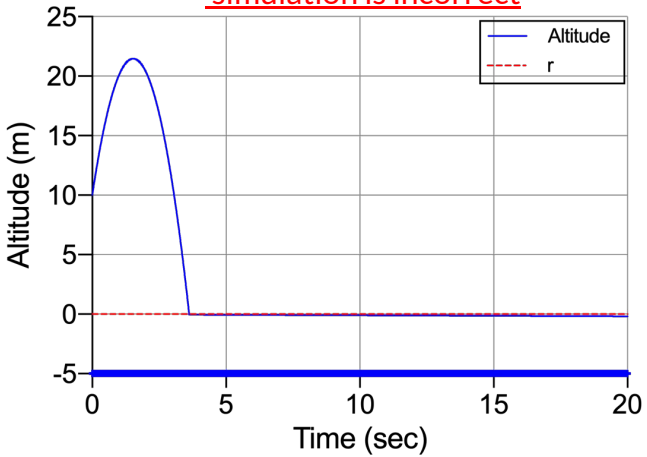
# Use case 2: simulation of hybrid systems

Hybrid systems: continuous dynamics (e.g., ordinary differential equations (ODEs)) + discrete interactions

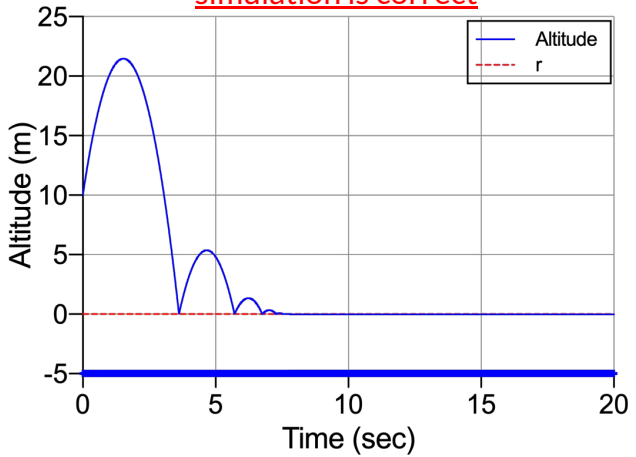
Example: Bouncing ball system (simplified):

- Continuous dynamics: H - altitude of a ball, V - velocity of a ball, g = 9.81m/s^2 - gravitational acceleration, $\begin{cases} \dot{H} = V \\ \dot{V} = -g, \end{cases}$
- Discrete interactions: $\text{if } H(t) = r, \text{ then } V(t) := -c \cdot V(t-0), c = const, 0 < c < 1$
- The "ground" level r can be constant, time-variable or even can depend on the system's variables x(t) = (H(t),V(t)).
- The zero-crossing function: *g(t,H(t)) = H(t)-r.*

The first zero-crossing was determined with large errors: simulation is incorrect



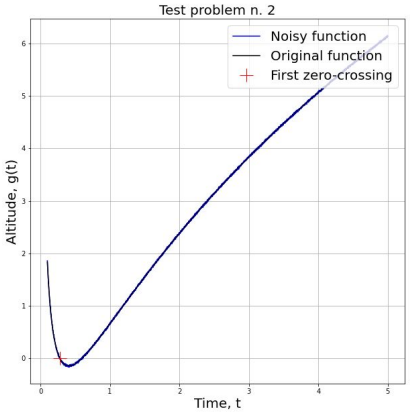

The first zero-crossing was determined well: simulation is correct

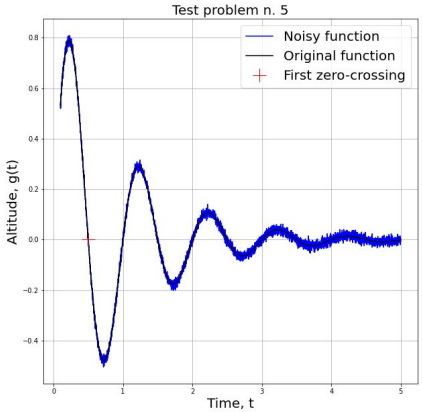


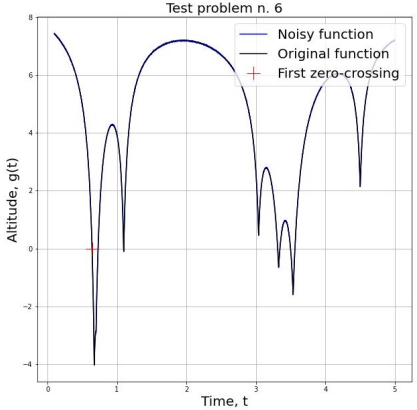
# Interest

- Efficient zero-crossing determination *"on-the-fly"* is very important for vast groups of stakeholders: from **pharmaceutical companies** (e.g., preventing critical levels of sugar in blood for diabetic patients) to **Space Agencies** (e.g., in simulating difficult hybrid systems like landing the rockets).
- Methodologies presented in this study are of general-purpose and can be easily applied to any problem, where an efficient determination of zero-crossings is required.
- Only the Data section of the Jupyter notebooks related to this study should be changed for different problems.
- The main parameters of the system are fixed automatically adapting to a concrete problem.

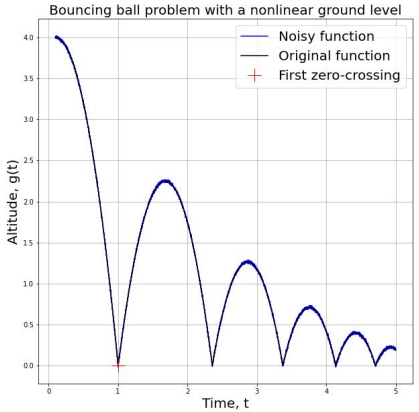
# Data and Exploration

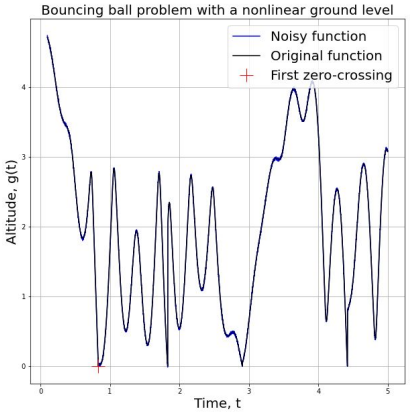



- The first 10 test problems from Casado, Garcia, Sergeyev (2003) have been used for evaluating the models (examples of functions number 2, 5 and 6 are presented).




- Bouncing ball hybrid system with two different case studies have been also implemented: with a constant ground and difficult nonlinear ground level.
- A random gaussian noise has been added to all function evaluations in order to complicate the problems.

# Solution: interactive Jupyter notebooks

- ***[Zero-crossings_in_time_series].import_libraries.python.ipynb***
  Optional notebook for importing of all necessary libraries once.
- ***[Zero-crossings_in_time_series].data_exploration.python.ipynb***
  Data exploration. Contains the implementations of all test problems, the function for Lipschitz constant estimation and all initial exploration plots.
- ***[Zero-crossings_in_time_series].etl.python.ipynb***
  Extract-Transform-Load. Contains the functions for generating/getting new time observations and the objective function evaluations. Can be connected to an external dataset. All function values are checked to be numeric.
- ***[Zero-crossings_in_time_series].feature_eng.python.ipynb***
  Feature engineering. Contains the function `prepare_features', which transforms the vector t into polynomial features and then scales the obtained features into [0,1]. The result is divided into training set and prediction set.
- ***[Zero-crossings_in_time_series].model_def.python.ipynb***
  Model definition. Compiles the selected model with the predefined parameters.
- ***[Zero-crossings_in_time_series].model_train.python.ipynb***
  Model training. Contains the functions required for model fitting and prediction. Contains the function `find_degrees`, which returns the optimal number of degrees for polynomial features.
- ***[Zero-crossings_in_time_series].model_evaluate.python.ipynb***
  Model evaluation. Contains the function for checking zero-crossings and computation of the relative accuracy (if the real value t* is provided).
- ***[Zero-crossings_in_time_series].model_deployment.python.ipynb***
  Model deployment. An example of the application of the project. Contains the loading of all notebooks defined above and the function for the full simulation cycle including all steps defined above.