Statistical Methods in Particle Physics

Selected topics 5:
Symbolic Regression

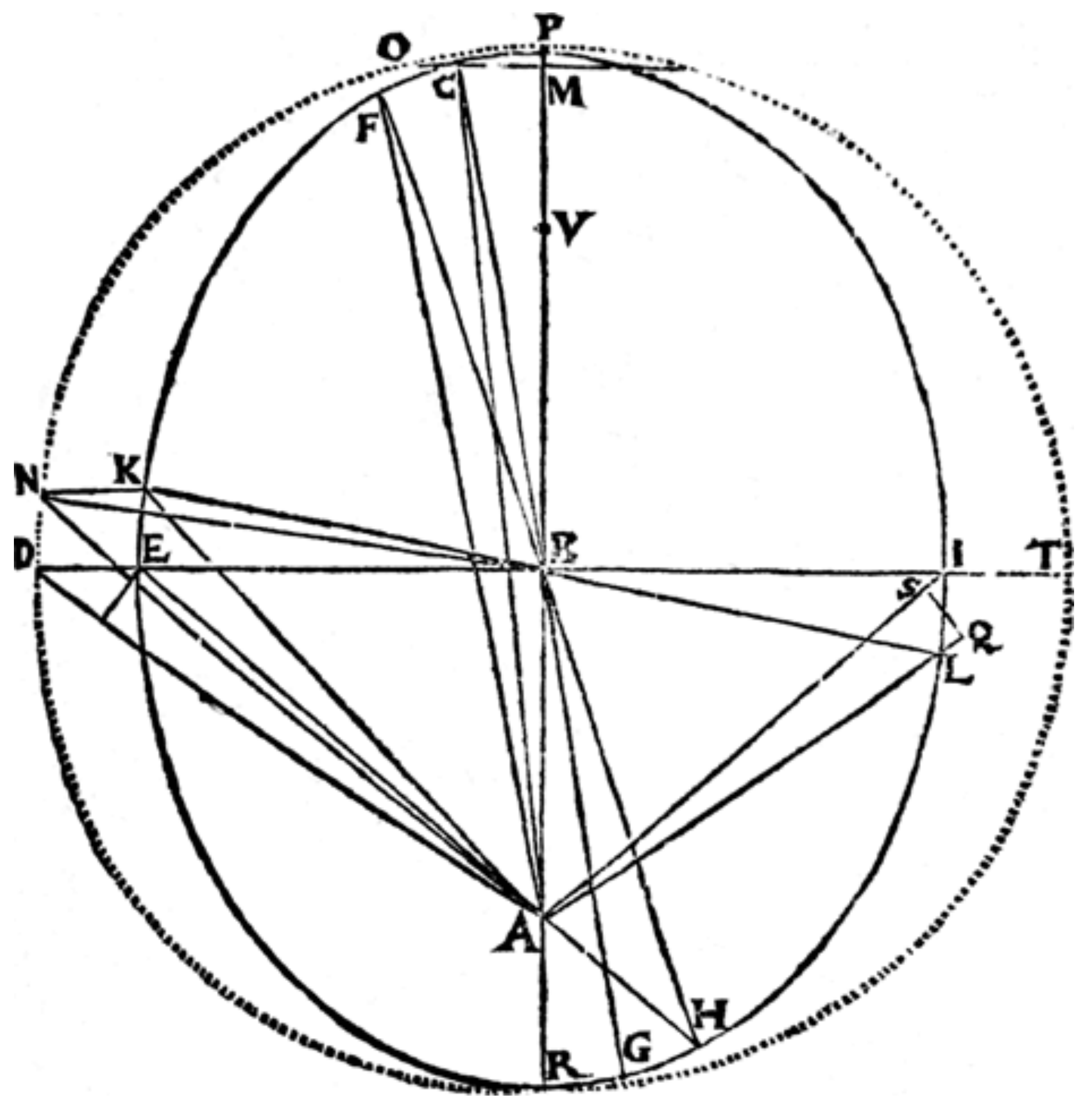
Klaus Reygers

Heidelberg University, WS 2021/22

Klaus Reygers (lectures)
Rainer Stamen, Martin Völkl (tutorials)

An early example of symbolic regression: Kepler's laws

- Johannes Kepler got access to Tycho Brahe's accurate data tables on planetary orbits
- Like many philosophers of his era, Kepler had a mystical belief that the circle was the Universe's perfect shape
- After many failed attempts to describe the data, Kepler discovered that the orbit of Mars was an ellipse



https://earthobservatory.nasa.gov/features/OrbitsHistory/page2.php

Essence of the scientific method: extracting (simple) physical laws from observation

Another example: the Rydberg formula

Wavelength of spectral lines of the hydrogen atom:

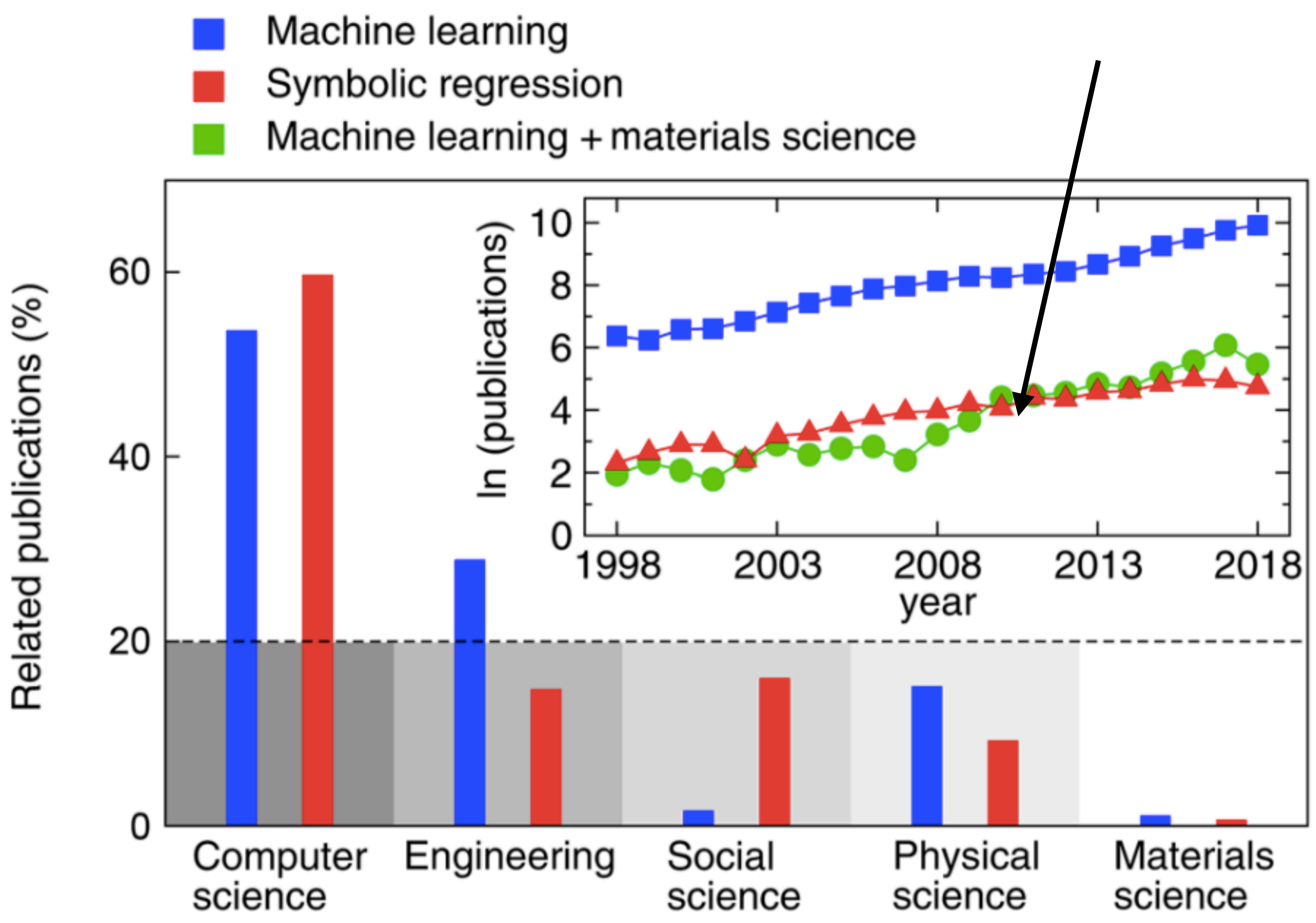
$$\frac{1}{\lambda_{\mathrm{vac}}} = R_{\mathrm{H}} \left(\frac{1}{n_1^2} - \frac{1}{n_2^2} \right)$$

Empirical formula that was guessed by Rydberg. An understanding of the formula came later.

Symbolic regression (SR)

- Simultaneously search for
  - optimal functional form and
  - optimal parameters to describe a dataset
- Comparison to Machine Learning
  - ML:
    - Predictive, but hard to interpret ("black box")
  - SR:
    - Parsimonious and yet predictive
    - Ideally gives interpretable result

SR is a relatively small field:
number of publications $n_{SR} < 0.02\, n_{ML}$



Yiqun Wang, Nicholas Wagner, James M. Rondinell,i
Symbolic regression in materials science,
https://doi.org/10.1557/mrc.2019.85

Genetic Programming (GP)

- In genetic programming one evolves a population of computer programs
- GP: developed by John Koza as a specific implementation of genetic algorithms (GAs)
- Basic algorithm

A Field Guide to Genetic Programming, http://www.gp-field-guide.org.uk/

Randomly create an *initial population* of programs from the available primitives (more on this in Section 2.2).
repeat
    Execute each program and ascertain its fitness.
    Select one or two program(s) from the population with a probability based on fitness to participate in genetic operations (Section 2.3).
    Create new individual program(s) by applying *genetic operations* with specified probabilities (Section 2.4).
until an acceptable solution is found or some other stopping condition is met (e.g., a maximum number of generations is reached).
return the best-so-far individual.

Algorithm 1.1: Genetic Programming

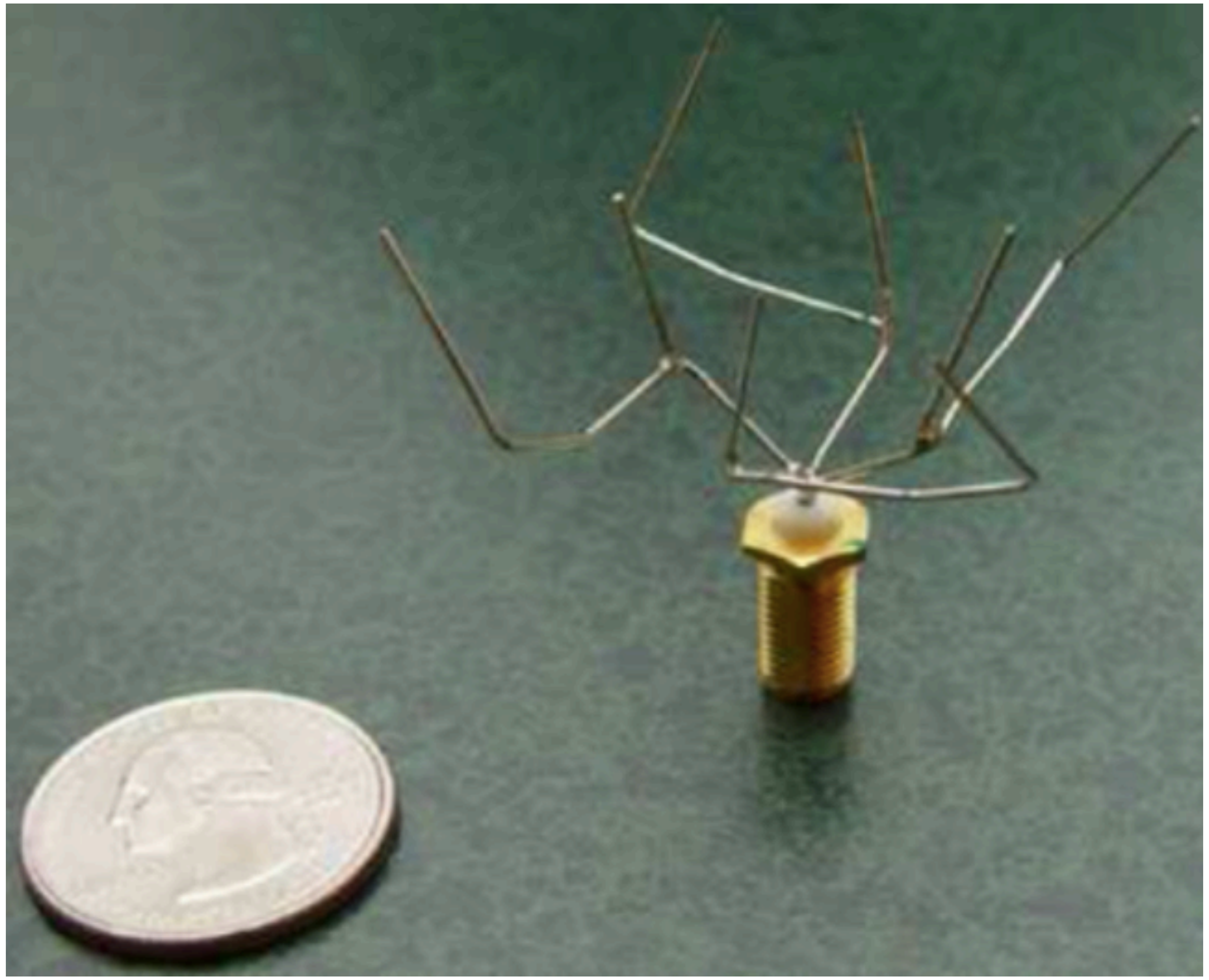
***Genetic operations*:**

Crossover:
The creation of a child program by combining randomly chosen parts from two selected parent programs.

Mutation:
The creation of a new child program by randomly altering a randomly chosen part of a selected parent program.

Genetic Programming (GP)

- Evolutionary algorithm → heuristic approach to find good programs in a vast search space
- A number of real-life applications
- Price for "human-competitive results" at the Genetic and Evolutionary Computation Conference (GECCO)
  - "An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission
  - "Automatic Quantum Computer Programming"
  - "Mate-In-N Problem in Chess"

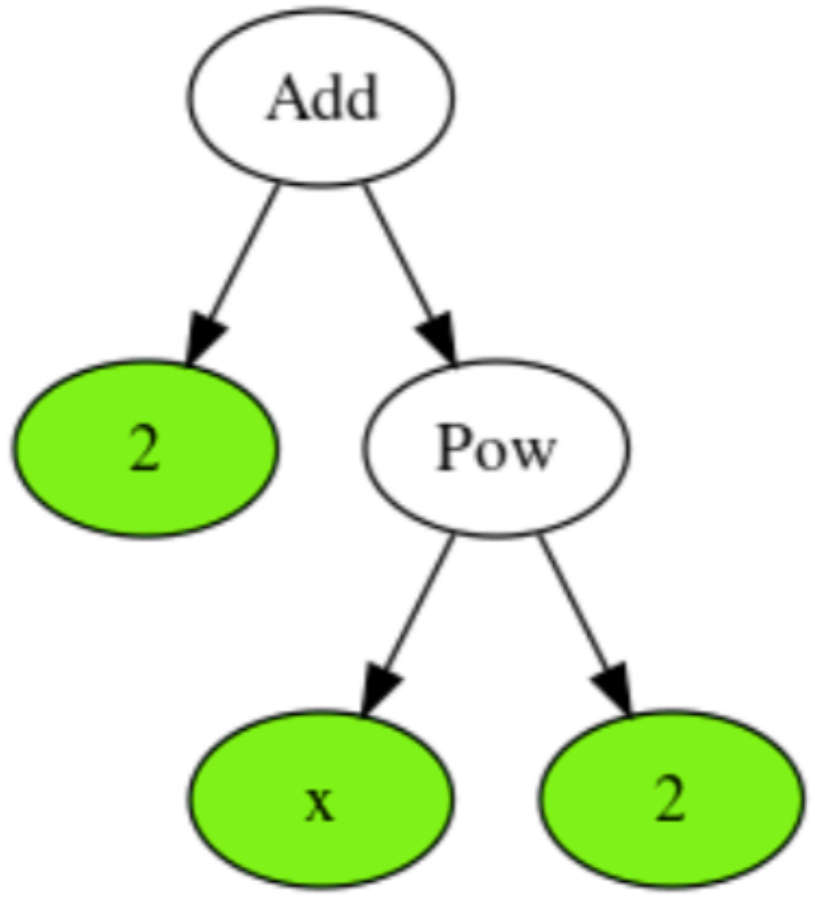


Antenna for NASA's Space Technology 5 Mission designed by an GP algorithm

A Field Guide to Genetic Programming, http://www.gp-field-guide.org.uk/

John R. Koza,
Human-competitive results produced by genetic programming,
Genet Program Evolvable Mach (2010) 11:251–284

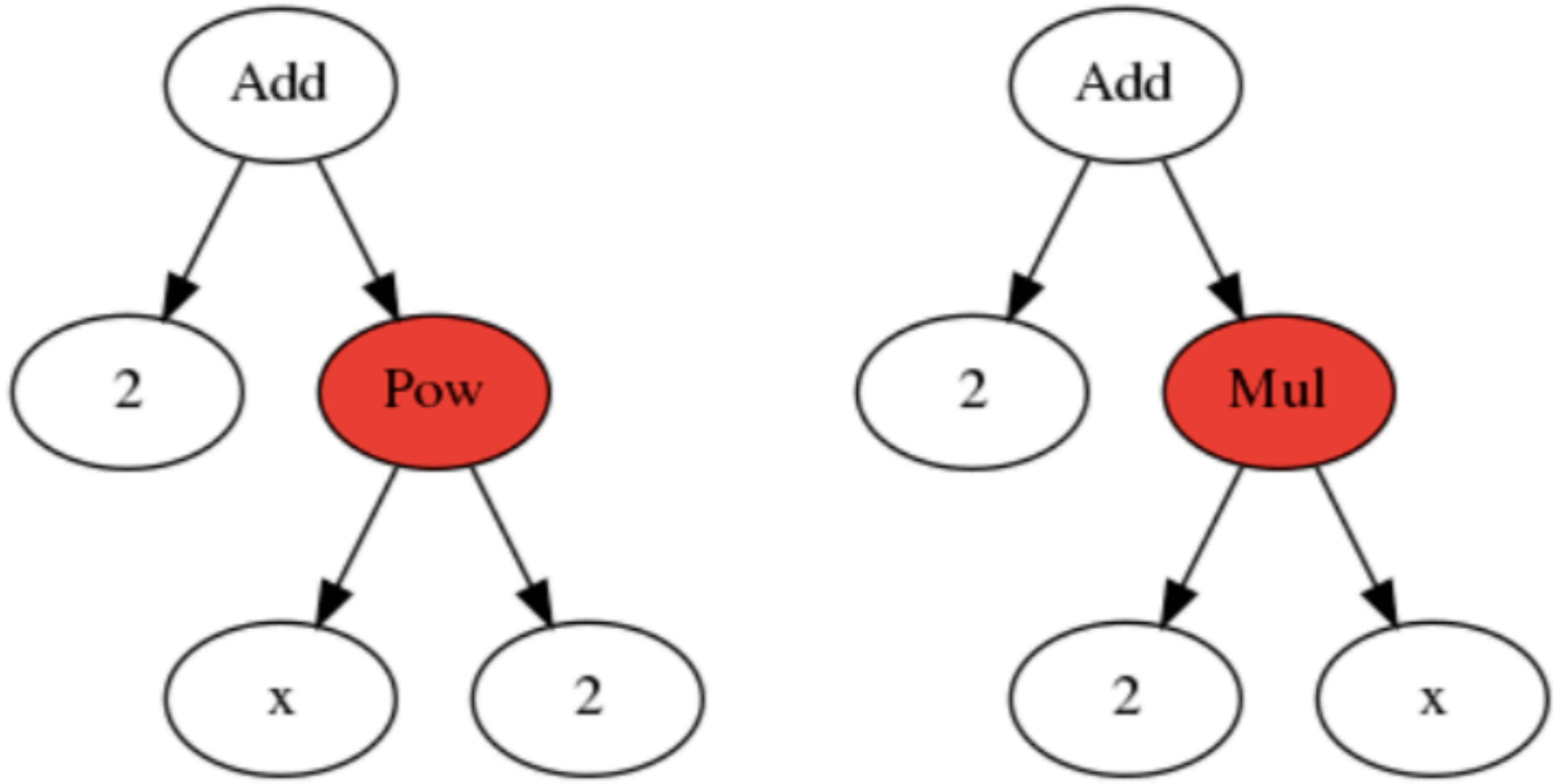
Tree representation of mathematical expressions



$2 + x^2$

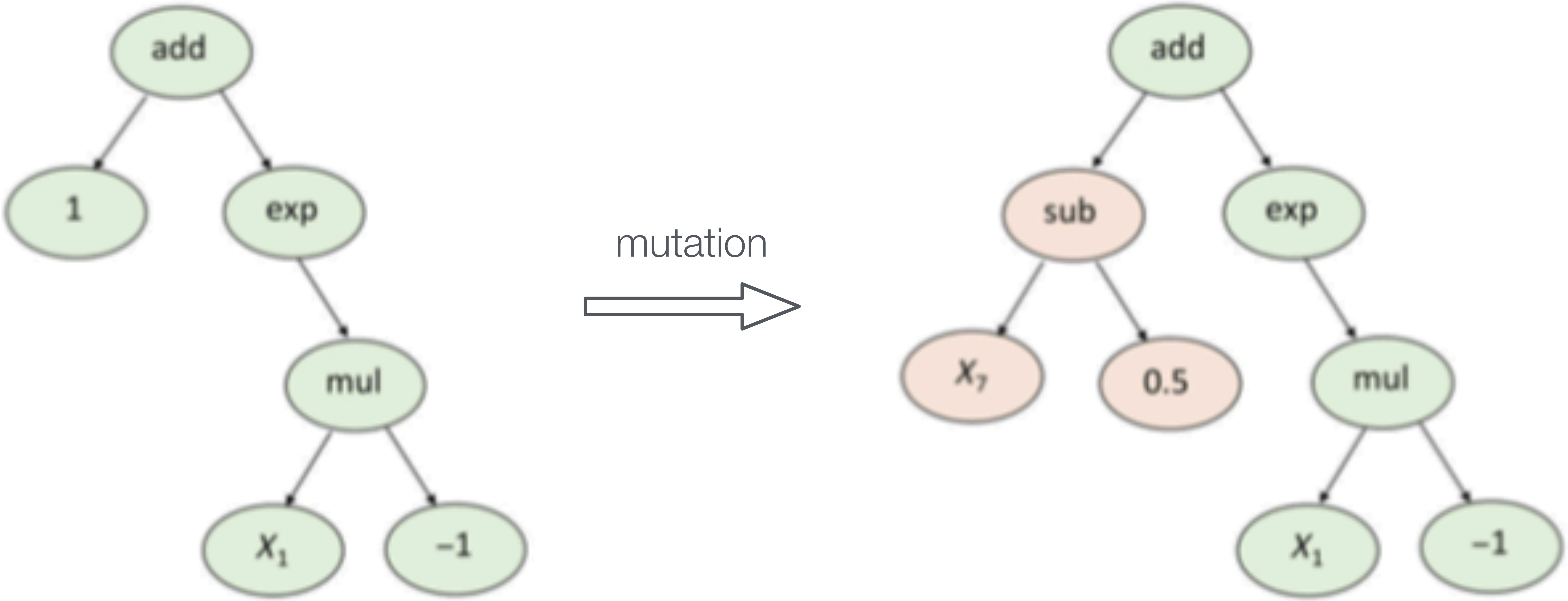
- Symbolic regression:
  one of the earliest applications of GP
- Program = mathematical expression
- Easy to apply mutation and cross-over in tree representation

Point mutation:



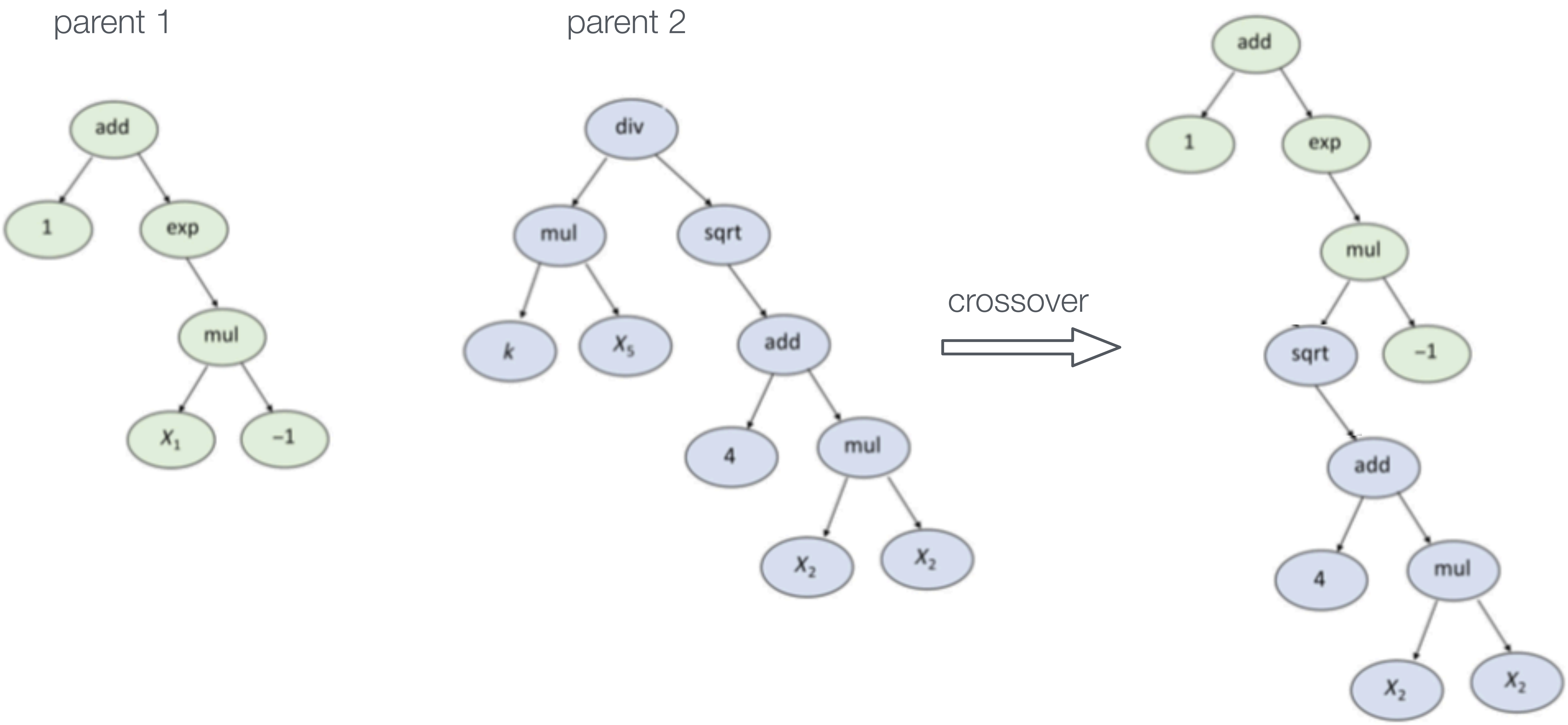
(a) $x^2 + 2$ (b) $2x + 2$

Mutation



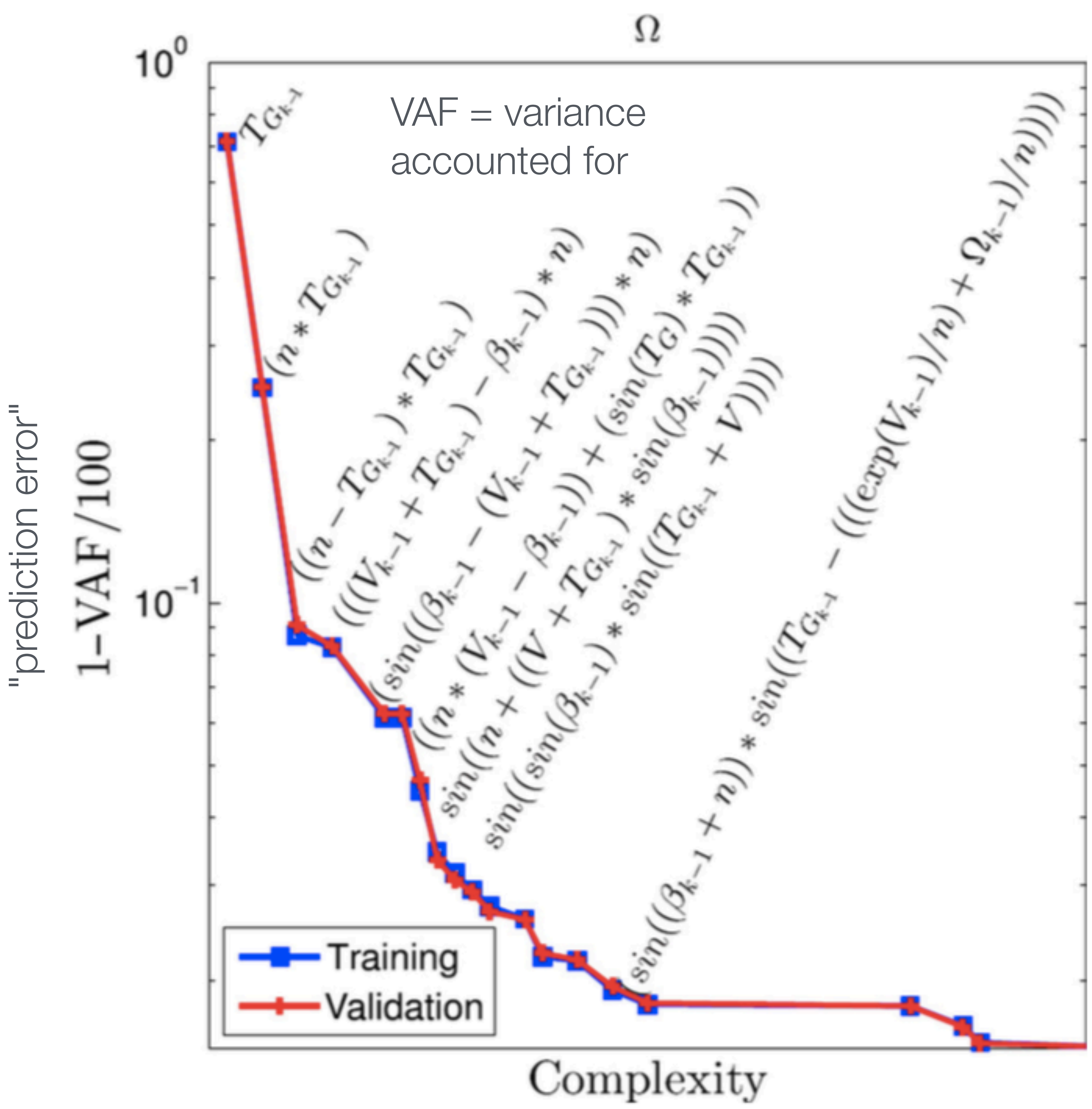
Yiqun Wang, Nicholas Wagner, James M. Rondinelli
https://doi.org/10.1557/mrc.2019.85

Crossover



Yiqun Wang, Nicholas Wagner, James M. Rondinelli
https://doi.org/10.1557/mrc.2019.85

Pareto front



For each function complexity there is an optimum solution. These optimum solutions form a set, the Pareto front.

In general, there is a trade-off between model complexity and the accuracy of the prediction

Selecting a relatively simple function avoids over-fitting

W. La Cava, K. Danai, L. Spector, P. Fleming, A. Wright, and M. Lackner:
Automatic identification of wind turbine models using evolutionary multi-objective optimization.
Renew. Energy 87, 892–902 (2016)

Challenges of symbolic regression using genetic programming

- A highly complex problem
  - functions = string or tree of symbols → number of strings/trees grows exponentially with length
  - symbolic regression probably a NP-hard problem
- Not obvious that genetic algorithms are better than brute-force searches
- Non-deterministic optimization (heuristic approach)
  - Descendent generations can perform worse than their parents
  - No guarantee to find a useful expression
- Preservation of good components of the equation
  - Good equation components do not guarantee high fitness of the full expression
- No effective way to find numerical values for constants in standard GPSR
- Bloat = program growth without (significant) return in terms of fitness

Some symbolic regression tools/libraries

http://geneticprogramming.com/software/

- PySR
  - https://github.com/MilesCranmer/PySR
  - “the goal of this package is to have an open-source symbolic regression tool as efficient as eureqa, while also exposing a configurable python interface.”
- gplearn
  - scikit-learn inspired and compatible API
  - https://gplearn.readthedocs.io/en/stable/
- DEAP
  - https://deap.readthedocs.io/en/master/examples/gp_symbreg.html
- FFX: Fast Function Extraction
  - https://github.com/natekupp/ffx
  - Fast, scalable, deterministic
  - Cannot handle error bars
- Mathematica
  - FindFormula
  - Cannot handle error bars
- Eureqa
  - https://www.nutonian.com
  - Comercial
  - Free version of available upon request for non-profit academic research (?)
  - Used in "The first analytical expression to estimate photometric redshifts suggested by a machine" (arXiv:1308.4145)
- HeuristicLab
  - https://dev.heuristiclab.com
  - Only Windows

Example from physics: Double Pendulum

Michael Schmidt; Hod Lipson (2009), "Distilling free-form natural laws from experimental data", Science. 324 (5923): 81–85

Non-trivial conservation law found through GPSR

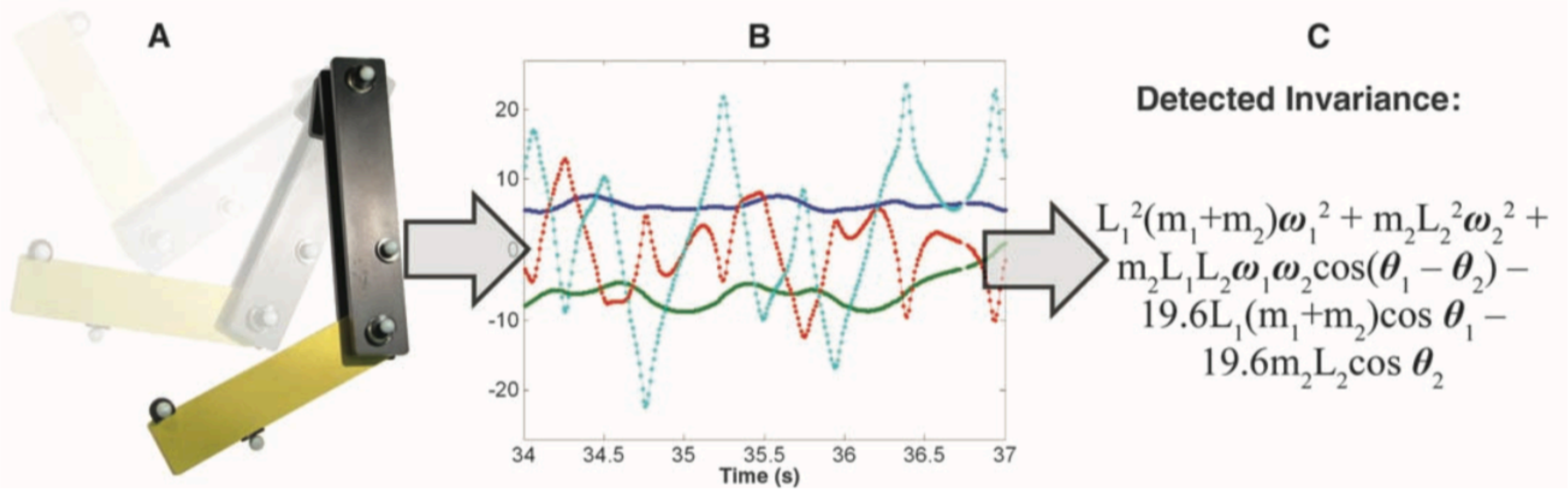


Fig. 1. Mining physical systems. We captured the angles and angular velocities of a chaotic double-pendulum (**A**) over time using motion tracking (**B**), then we automatically searched for equations that describe a single natural law relating these variables. Without any prior knowledge about physics or geometry, the algorithm found the conservation law (**C**), which turns out to be the double pendulum's Hamiltonian. Actual pendulum, data, and results are shown.

Code now commercially available as "Eureqa"

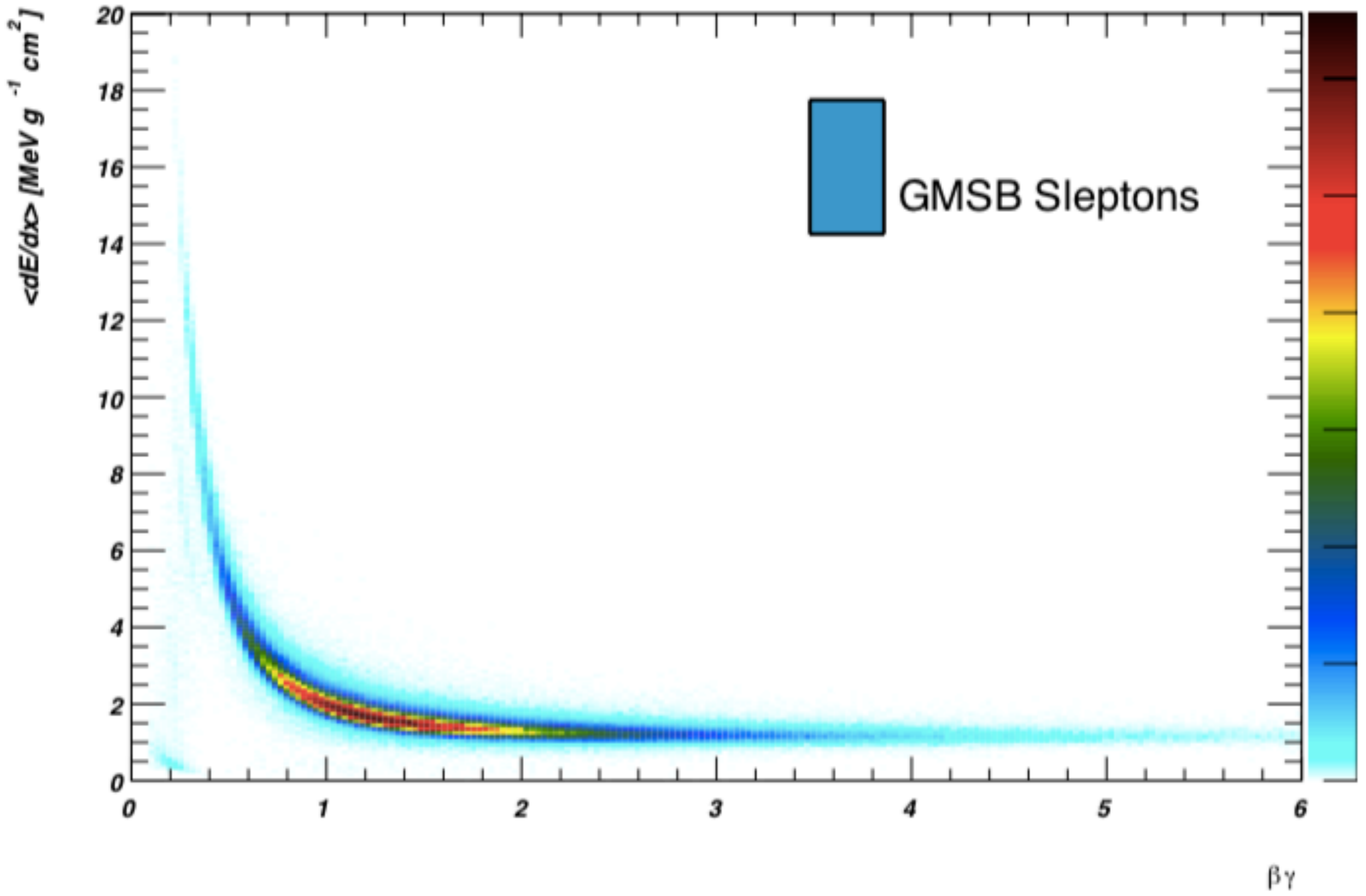
An example from particle physics

- Search for exotic long-lived particles with ATLAS
- Goal: parameterize dE/dx *vs.* βγ **including** detector effects for a hypothetical particle
- Result using Eureqa:

$$\left\langle \frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle (\beta\gamma) = \frac{\beta\gamma + \frac{1.1751}{\beta\gamma} - 0.2306}{0.8924\,\beta\gamma + 0.0797}$$

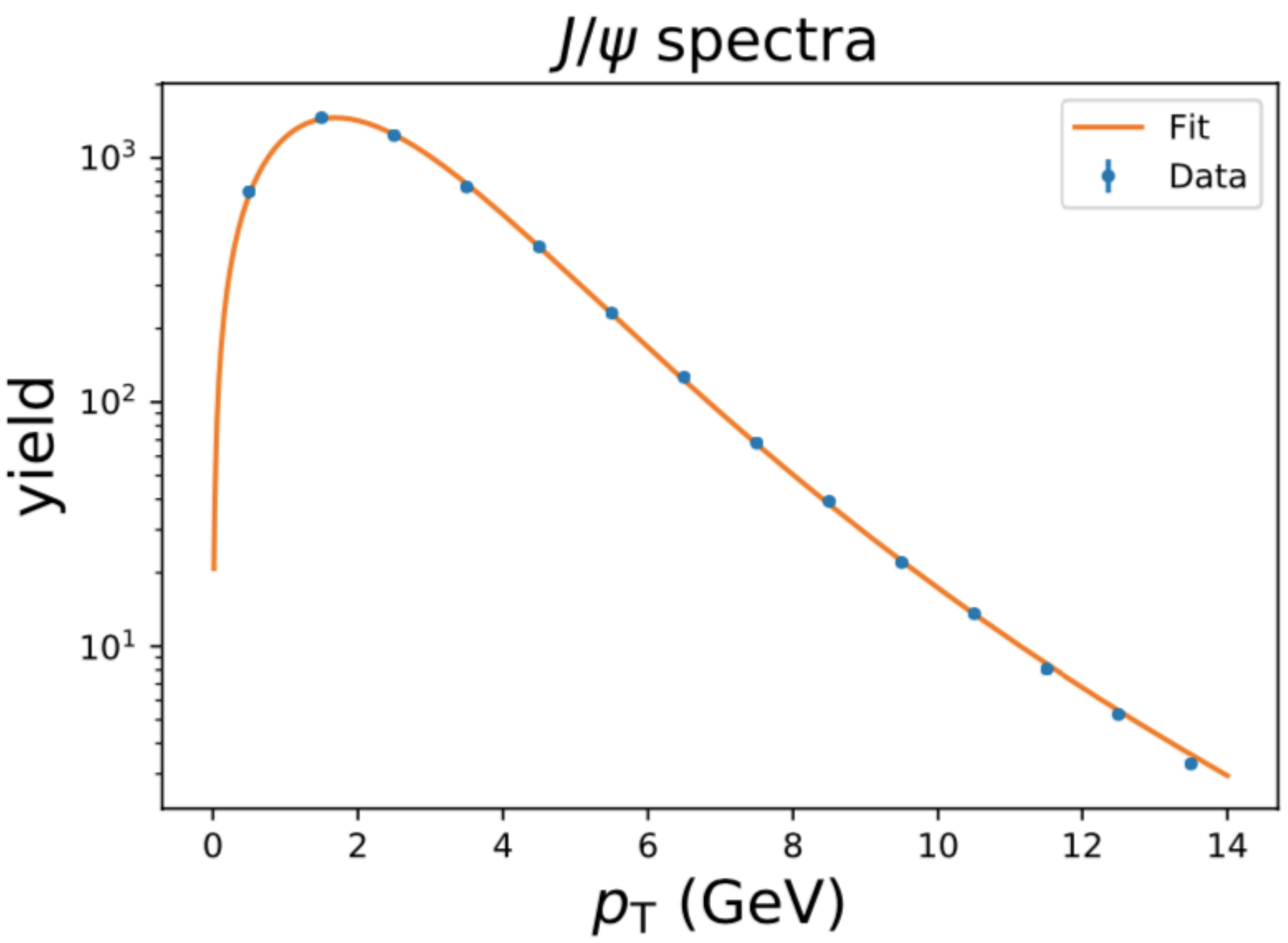
$$\Rightarrow \left\langle \frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle (\beta\gamma) = \frac{a + b\,\beta\gamma + (\beta\gamma)^2}{\beta\gamma\,(c + d\,\beta\gamma)}$$

Joergensen, Morten Dam,
Exotic Long-Lived Particles
CERN-THESIS-2014-021



Size:	Fitness:	Equation:
15	0.0135497	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = (1.175/\beta\gamma + \beta\gamma - 0.231)/(0.080 + 0.892\beta\gamma)$
14	0.0138698	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.184 + (1.376 - 0.554\beta\gamma)/\beta\gamma^{1.785}$
19	0.0135428	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = (\beta\gamma^{1.006} + 1.180/\beta\gamma - 0.211)/(0.079 + 0.905\beta\gamma)$
13	0.0142499	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.102 + 0.906\beta\gamma^{-0.290\beta\gamma - 1.947}$
11	0.0147863	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.072 + 1.156(0.099 + \beta\gamma)^{-}2.385$
9	0.0210256	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.057 + (0.030 + \beta\gamma)^{-2.132}$
8	0.0257238	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.040 + 0.970/\beta\gamma^2$
7	0.0276055	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.027 + \beta\gamma^{-}1.968$
6	0.163132	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = \exp(0.763/\beta\gamma)$
4	0.427812	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 2.508/\beta\gamma$
1	1.01597	$\mathrm{d}E/\mathrm{d}x(\beta\gamma) = 1.535$

An further example from particle physics



- Custom GPSR code (python)
- Can handle error bars
- Fitness evaluation: Numerical constants optimized by standard fitting (Levenberg-Marquardt or Minuit)

Function Set:	add, mul, exp, $x \mapsto x^{-1}$, $x \mapsto x^k$
Terminal Set:	x_0, c_0
Initial Population:	ramped half-and-half, depth = random(3,4,5), sizelimit = 40
Algorithm:	age_fitness with default settings[3]

Result: $$c_0 c_1^{k_1} p_{\mathrm{T}}^{-k_0} \left(1 + \frac{p_{\mathrm{T}}^2}{c_1}\right)^{k_1}$$

Similar in structure to well-known form

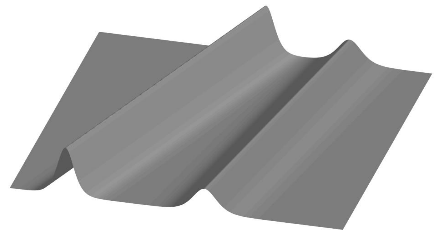
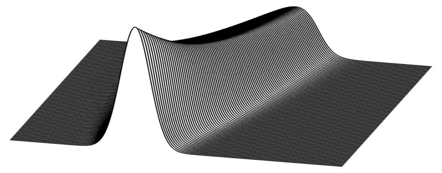
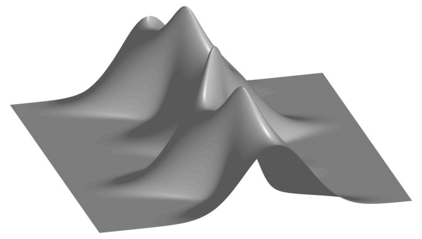
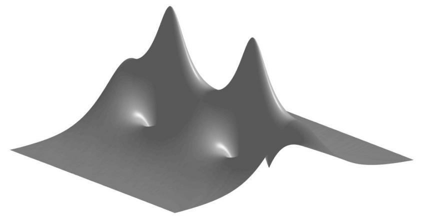
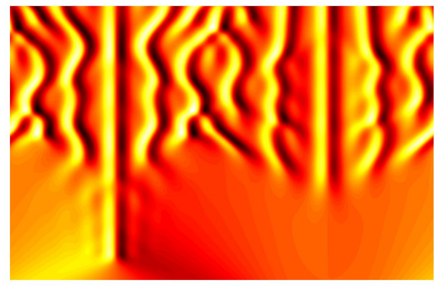
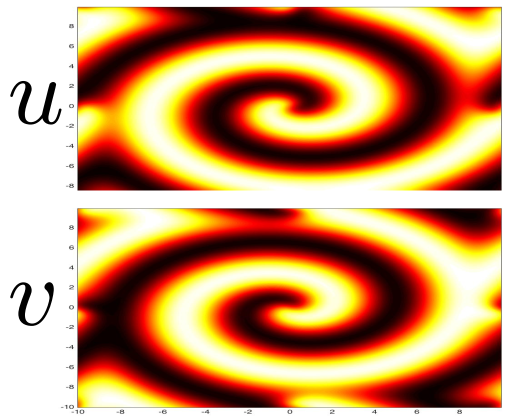
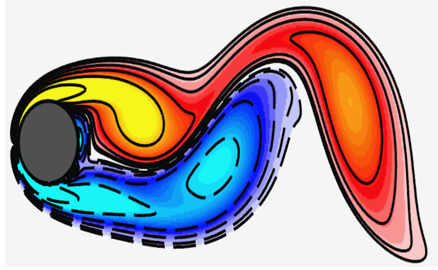
$$f(p_{\mathrm{T}}) = C_0 \frac{p_{\mathrm{T}}}{(1 + (p_{\mathrm{T}}/p_0)^2)^n}$$

David Korbany,
Symbolic Regression in Heavy-Ion Physics, Bachelorarbeit, 2019

Finding partial differential equations (PDE) from data

- In addition to searching for a function $y = f(x_1, \ldots, x_n)$ one can search for the PDE governing a physical system
- Rudy et al., Sci. Adv. 2017:
  - Deterministic SR algorithm
  - Terms of the governing PDE taken from a large library of potential candidate functions

Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz,
Data-driven discovery of partial differential equations, arXiv:1609.06401, Sci. Adv. 2017

PDE	Form
 KdV	$u_t + 6uu_x + u_{xxx} = 0$
 Burgers	$u_t + uu_x - \epsilon u_{xx} = 0$
 Schrödinger	$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0$
 NLS	$iu_t + \frac{1}{2}u_{xx} + \lvert u\rvert^2 u = 0$
 KS	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$
 Reaction Diffusion	$u_t = 0.1\nabla^2 u + \lambda(A)u - \omega(A)v$ $v_t = 0.1\nabla^2 v + \omega(A)u + \lambda(A)v$ $A^2 = u^2 + v^2, \omega = -\beta A^2, \lambda = 1 - A^2$
 Navier Stokes	$\omega_t + (\mathbf{u} \cdot \nabla)\omega = \frac{1}{Re}\nabla^2\omega$

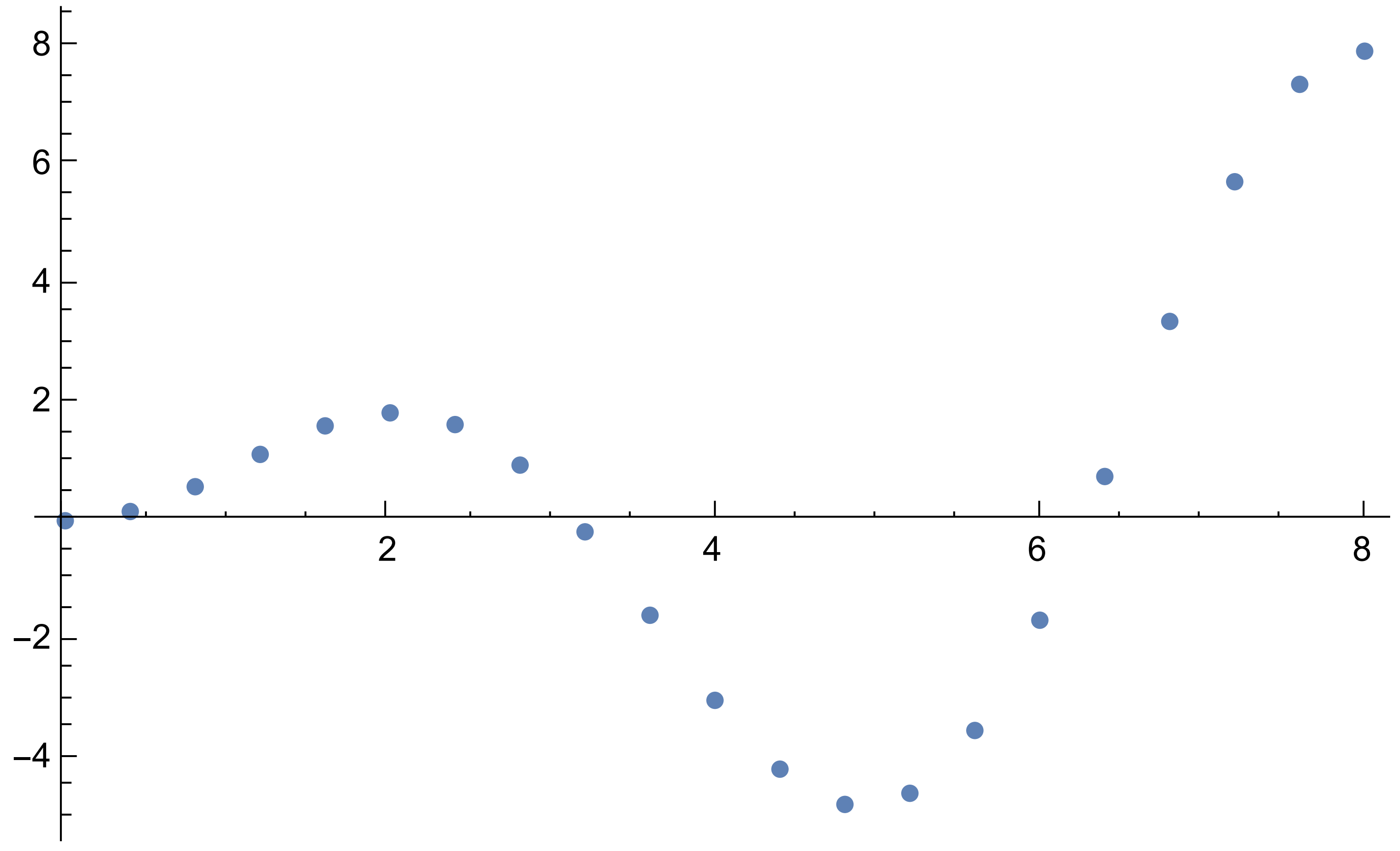
Mathematica: FindFormula

- Sometimes gives desired result
- Could show many rather simple examples where is does not
- User cannot define loss function or error bars
- Not clear how FindFormula works

```
In[51]:= data = Table[{x, N[x Sin[x]]}, {x, 0, 8, .4}];

In[52]:= p1 = ListPlot[data]
```

Out[52]=



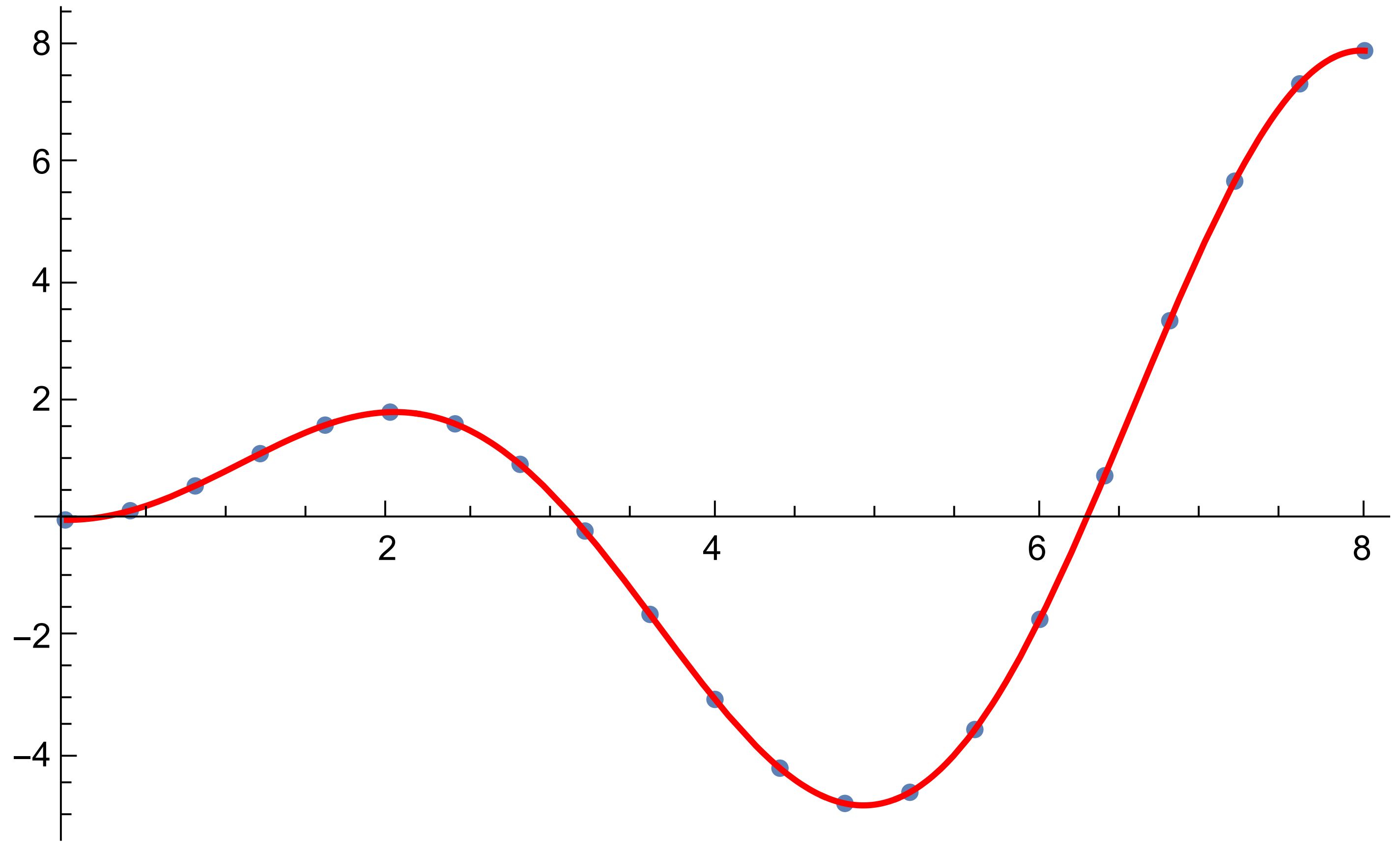

```
In[53]:= f = FindFormula[data, x]

Out[53]= x Sin[x]

In[57]:= p2 = Plot[f, {x, 0.01, 8.}, PlotStyle → Red];

In[58]:= Show[p1, p2]
```

Out[58]=

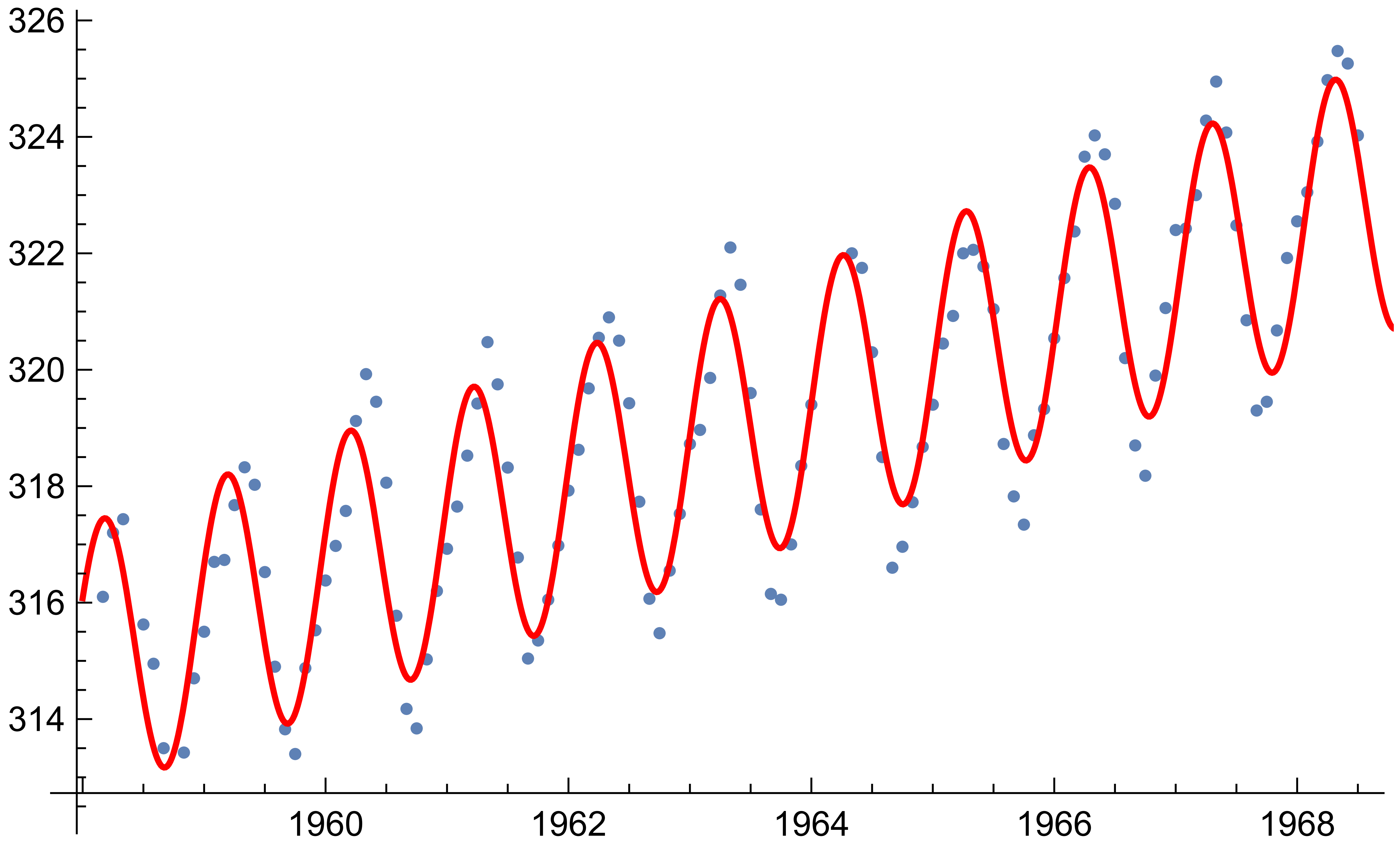


FindFormula: Mauna Loa CO_2 concentration data

Mauna Loa C02 data: https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_co2.html

SpecificityGoal:
controls complexity of the fit function

17]=



In[114]:=
```
f = FindFormula[data, x, 3, SpecificityGoal → 2]
```

Out[114]=
```
{-1145.7 + 0.745995 x + 2.38897 Sin[6.2 x],
 -1149.47 + 0.747925 x, -1149.44 + 0.747925 x}
```

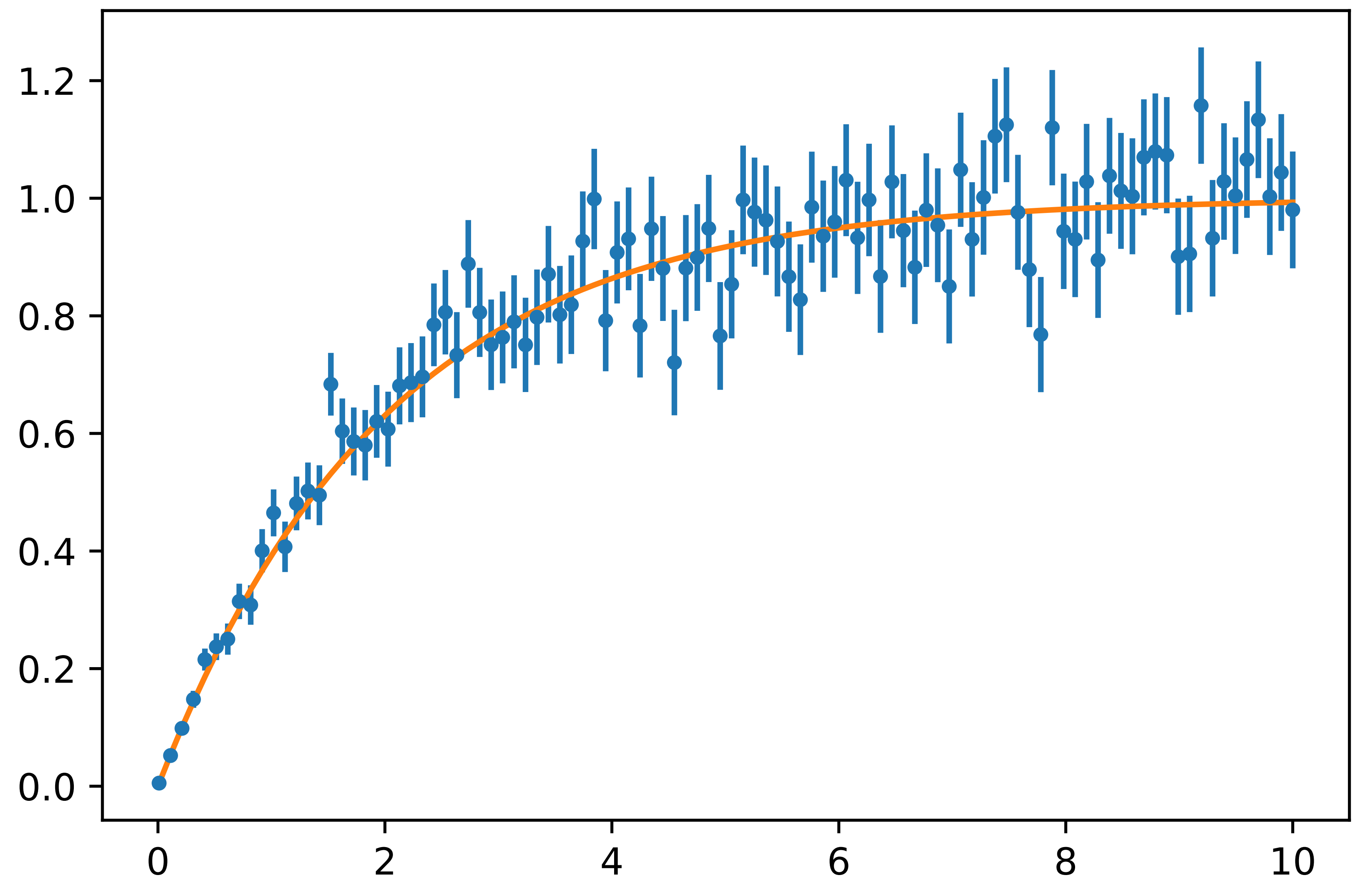
In[115]:=
```
g = FindFormula[data, x, 3, SpecificityGoal → 4]
```

Out[115]=
```
{-1140.64 + 0.743427 x + 2.32643 Sin[6.2 x],
 -1184.54 + 0.765759 Abs[x] + 0.746585 Cos[x (-0.0246137 + Cot[x])] -
  0.00747384 Csc[x], -1184.54 + 0.765759 Abs[x] +
  0.746585 Cos[x (-0.0246137 + Cot[Abs[x]])] - 0.00747384 Csc[x]}
```

gplearn example (1)

```
# define data
npoints = 100
x = np.linspace(0.01, 10, npoints)
y_smooth = 1 - np.exp(-x/2)

# smear data points (10% relative error)
mean, std = 0., 1.
s = np.random.normal(mean, std, npoints)
y_err = 0.1 * y_smooth * np.ones(npoints)
y = y_smooth + y_err * s
```




```
# define loss function
def _chi2(y, y_pred, w):
    """Calculate relative difference"""
    pulls = w * (y - y_pred) / y
    return np.sum(pulls * pulls)

chi2 = make_fitness(_chi2, greater_is_better=False)
```

```
func_set = ('add', 'sub', 'mul', 'div', 'neg', pexp)

est_gp = SymbolicRegressor(population_size=500,
                           generations=30,
                           metric=chi2,
                           const_range=(0.0, 1.0),
                           init_depth = (2,4),
                           stopping_criteria=0.01,
                           p_crossover=0.05,
                           p_subtree_mutation=0.05,
                           p_hoist_mutation=0.05,
                           p_point_mutation=0.3,
                           parsimony_coefficient=1,
                           random_state=0,
                           function_set=func_set,
                           verbose=1)
```

gplearn example (2)

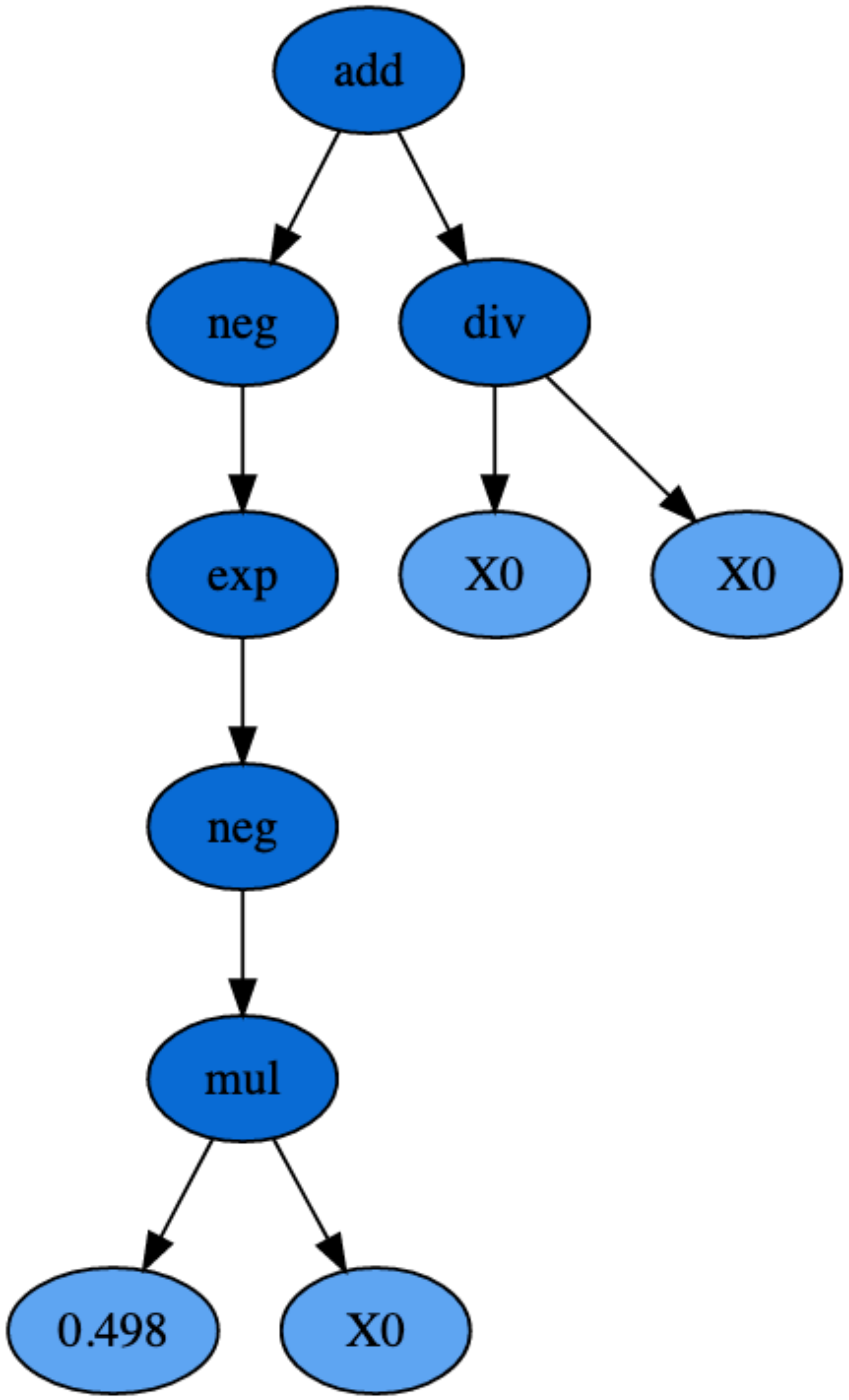
```
X_train = x.reshape(-1,1)
y_train = y
est_gp.fit(X_train, y_train)
```

```
    |    Population Average    |             Best Individual              |
---- ------------------------- ---------------------------------------- ----------
 Gen   Length          Fitness   Length          Fitness      OOB Fitness  Time Left
   0    11.54      4.55051e+16       12          9.92597              N/A     21.49s
   1    11.44      2.36876e+07       12          9.92597              N/A     17.49s
   2    10.96      3.94285e+07       12          9.92597              N/A     13.83s
   3    11.39      3.68436e+07       12          9.92597              N/A     12.45s
   4     9.24      2.59533e+07       11          4.18918              N/A     11.47s
   5     9.00      3.98642e+07       11          4.18918              N/A     12.36s
   6     9.94      2.22681e+07       11          0.81926              N/A     11.14s
   7    10.84      2.12555e+07       11         0.707691              N/A     10.72s
```

```
print(est_gp._program)
```

```
add(neg(exp(neg(mul(0.498, X0)))), div(X0, X0))
```

```
dot_data = est_gp._program.export_graphviz()
graph = graphviz.Source(dot_data)
graph
```



Nice, but requires a lot of tuning of hyper parameters …

PySR

```
import numpy as np
from pysr import pysr, best

# Dataset
x = np.arange(0., 8., 0.4)
y = x * np.sin(x)

# Learn equations
equations = pysr(
    x,
    y,
    niterations=5,
    binary_operators=["+", "*"],
    unary_operators=[
        "cos",
        "exp",
        “sin",
       "inv(x) = 1/x", # Define your own operator!)
    ]
)

print(best(equations))
```

```
Hall of Fame:
-----------------------------------------
Complexity  Loss       Score      Equation
1           9.902e+00  -0.000e+00  0.07158296
2           6.582e+00  4.084e-01  sin(x0)
4           2.005e-13  1.556e+01  (x0 * sin(x0))

==============================
Press 'q' and then <enter> to stop execution early.
x0*sin(x0)
```

AI Feynman: a Physics-Inspired Method for Symbolic Regression

- Additional search heuristics
  - Dimensional analysis
  - Symmetry
    - check for translational, rotational or scaling symmetry of the function
  - Compositionality
    - function f is a composition of a small set of elementary functions, each typically taking no more than two arguments
- Equations from Feynman lectures
  - Eureqa: discovered 68%
  - AI Feynman algorithm: 100%
- Impressive? Not obvious how much prejudice was put into the algorithm by knowing the answer

Examples of formulas found by AI Feynman, but not by Eureqa

Rutherford
$$A = \left(\frac{Z_1 Z_2 \alpha \hbar c}{4E \sin^2(\frac{\theta}{2})} \right)^2$$

Compton scattering
$$U = \frac{E}{1 + \frac{E}{mc^2}(1 - \cos\theta)}$$

Klein-Nishina formula
$$A = \frac{\pi \alpha^2 \hbar^2}{m^2 c^2} \left(\frac{\omega_0}{\omega}\right)^2 \left[\frac{\omega_0}{\omega} + \frac{\omega}{\omega_0} - \sin^2\theta \right]$$

A recent paper:
Deep Symbolic Regression for Recurrent Sequences

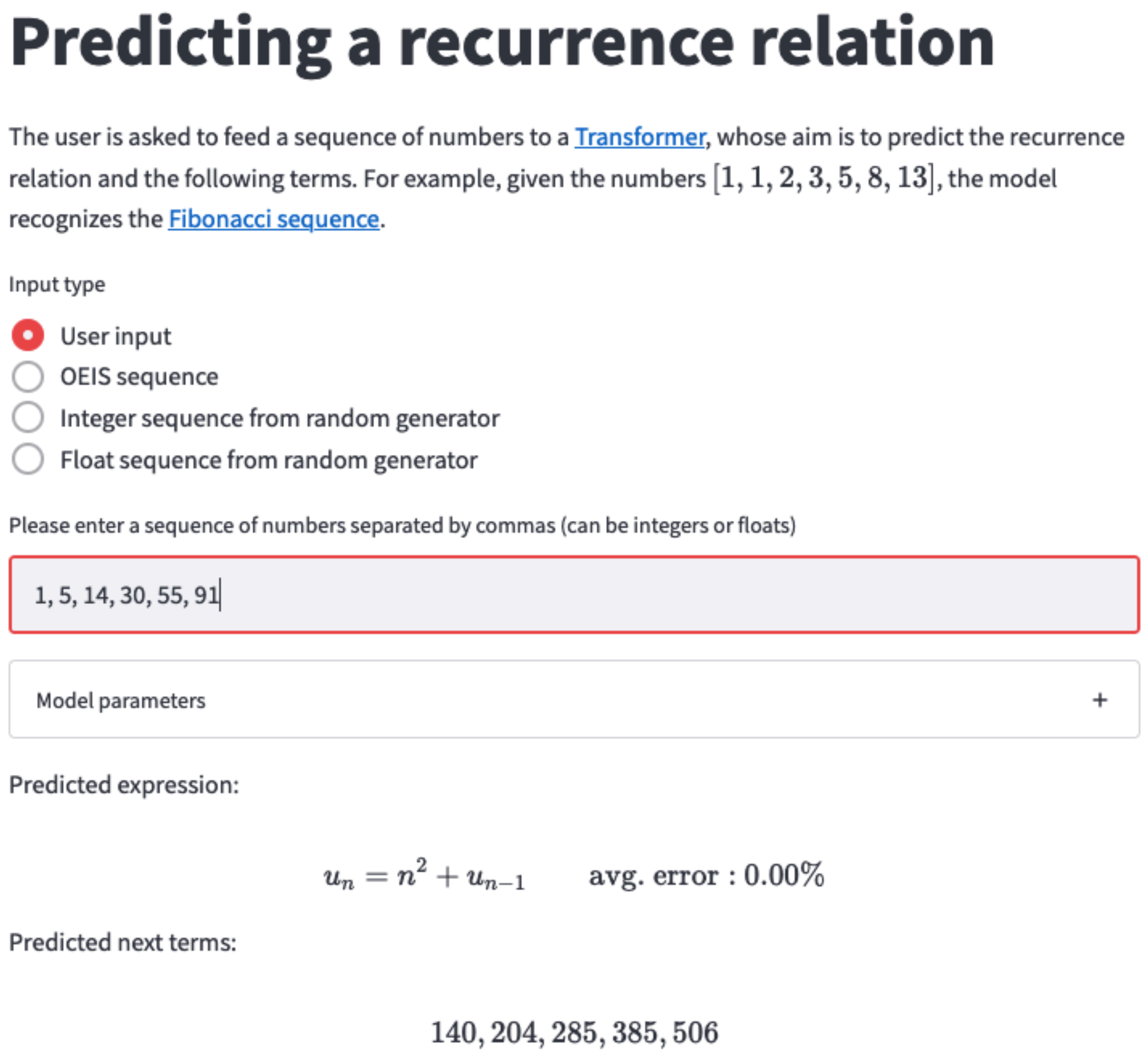
arXiv:2201.04600

Algorithms tries to figure out rules underlying a sequence of numbers.

Example: sum of squares

1. $1^2 = 1$
2. $1^2 + 2^2 = 5$
3. $1^2 + 2^2 + 3^2 = 14$
4. $1^2 + 2^2 + 3^2 + 4^2 = 30$
5. $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$
6. $1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 = 91$

http://recur-env.eba-rm3fchmn.us-east-2.elasticbeanstalk.com/



Summary: Genetic programming-based symbolic regression

- Provides (in some cases) relatively simple and interpretable parameterization of data
- Heuristic search: no guarantee that a useful result is found
- Ultimate goal: automated discovery of physical laws

From arXiv:1905.11481:

> "We look forward to the day when, for the first time in the history of physics, a computer, just like Kepler, discovers a useful and hitherto unknown physics formula through symbolic regression!"

References

- Silviu-Marian Udrescu, Max Tegmark, AI Feynman: a Physics-Inspired Method for Symbolic Regression, arXiv:1905.1148
- Yiqun Wang, Nicholas Wagner, James M. Rondinelli, Symbolic regression in materials science, MRS Communications (2019), 9, 793–805
- David Korbany, Symbolic Regression in Heavy-Ion Physics, Bachelorarbeit, 2019
- Michael Kommenda, Local Optimization and Complexity Control for Symbolic Regression, doctoral thesis, 2018