Fitting Runge Kutta Coefficients using Artificial Neural Networks

Chloe Griffin

July 20, 2022

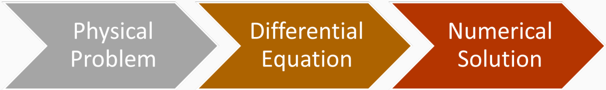
[†] KIT SCC

Introduction

Scientific Computing Motivation

Numerical Methods

- Solve challenging mathematical and physical problems using computers
- Have various applications in the sciences



Primary goal:

- Discover new Runge Kutta schemes catered to target problems or rediscover classical methods using artifical neural networks

Example: Exponential Population Growth

Pure Birth Model

- Relates
  change in population to population
  times a growth rate: $\frac{dP}{dt} = aP$
- Replace derivatives with
  simple differences: $\frac{dP}{dt} \approx \frac{P(t^{n+1})-P(t^{n})}{t^{n+1}-t^{n}}$

Create time-stepping method:

- Explicit Forward
  Euler: $P(t^{n+1}) = P(t^{n}) + \Delta t a P(t^{n})$
- Implicit Backward
  Euler: $P(t^{n+1}) = P(t^{n}) + \Delta t a P(t^{n+1})$

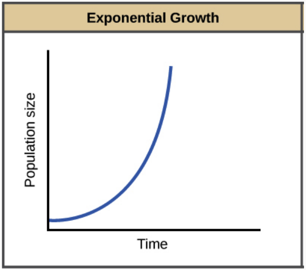


Figure 1: Exponential Population Growth (unlimited resources) [1]

We focus on fitting the coefficients of explicit Runge Kutta methods to target problems

Explicit Runge Kutta Methods

RK Structure

Consider initial value problem (IVP)

$$\frac{d\boldsymbol{u}}{dt} = f(t, \boldsymbol{u}(t)) \tag{1}$$

with an initial condition $\boldsymbol{u}(0) = \boldsymbol{u}_0$

- Approximate the continuously differentiable solution to $\boldsymbol{u}(t)$ over some time interval $[a, b]$ with time step h
- Build approximate solution iteratively with scheme

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h \sum_{i=1}^{m} c_i k_i \tag{2}$$

where, for a stage m method,

$$k_1 = f(t_n, \boldsymbol{x}_n)$$

$$k_2 = f(t_n + \alpha_2 h, \boldsymbol{x}_n + \beta_{21} k_1(t_n, x_n))$$

$$k_3 = f(t_n + \alpha_3 h, \boldsymbol{x}_n + h(\beta_{31} k_1(t_n, \boldsymbol{x}_n) + \beta_{32} k_2(t_n, \boldsymbol{x}_n)))$$

$$\vdots$$

$$k_m = f(t_n + \alpha_m h, \boldsymbol{x}_n + h \sum_{j=1}^{m-1} \beta_{mj} k_j)$$

RK Structure

- Recall: For a stage m method, k is defined as

$$k_m = f(t_n + \alpha_m h, \boldsymbol{x}_n + h \sum_{j=1}^{m-1} \beta_{mj} k_j) \tag{3}$$

- We are interested in fitting the parameters for α_i, β_{ij}, and c_i as seen in a butcher tableau

$$\begin{array}{c|cccc}
0 & & & & \\
\alpha_2 & \beta_{21} & & & \\
\alpha_3 & \beta_{31} & \beta_{32} & & \\
\vdots & \vdots & \vdots & \ddots & \\
\vdots & \vdots & \vdots & & \\
\alpha_m & \beta_{m1} & \beta_{m1} & \cdots & \beta_{mm-1} \\
\hline
c_1 & c_2 & \cdots & c_{m-1} & c_m
\end{array}$$

RK Order Conditions

- Runge Kutta designed to obtain order of accuracy
- Error is the difference between the approximation and the actual solution:

$$E(t_n) = |u_n - u(t_n)|, \tag{4}$$

where u_n is the approximate value and $u(t_n)$ is the exact value.

- As time step decreases, approximation approaches actual solution

- The order of a scheme, p, measures the convergence rate at which the error decays with respect to the step size

$$Error \approx Ch^p \tag{5}$$

Standard Finding Parameters

Suppose we want parameters to ensure 2nd order accuracy for RK2

- RK2 is given by

$$k_1 = f(t_n, \boldsymbol{u}_n)$$
$$k_2 = f(t_n + \alpha_2 h, \boldsymbol{u}_n + \beta_{21} k_1(t_n, u_n))$$
$$u_{n+1} = \boldsymbol{u}_n + h(c_1 k_1 + c_2 k_2)$$

- Taylor series expansion of u in neighborhood of t_n up to h^2 term provides the expression

$$u_{n+1} - u_n = hf(t_n, u_n) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial u}\right)\Bigg|_{(t_n,u_n)} + \mathcal{O}(h^3) \qquad (6)$$

  Expanding k_2 and substituting in original provides

$$x_{n+1} - x_n = h(c_1 + c_2)f(t_n, x_n) + h^2 c_2 \alpha_2 \frac{\partial f}{\partial t}\Bigg|_{(t_n,x_n)} + hc_2\beta_{21}\frac{\partial f}{\partial t}\Bigg|_{(t_n,x_n)} + \mathcal{O}(h^3) \qquad (7)$$

- Matching coefficients then provides order conditions [1]

Order Conditions and Classical Methods

- RK2 Order conditions:

$$c_1 + c_2 = 1,$$
$$c_2\alpha_2 = \frac{1}{2},$$
$$c_2\beta_{21} = \frac{1}{2}$$

- Three equations and four unknowns

Classic methods

- Heun's method: Let $\alpha_2 = 1$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

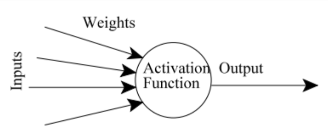
- Midpoint method: Let $\alpha_2 = 1/2$

$$\begin{array}{c|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$$

Fitting with Artificial Neural Networks

Artificial Neural Network (ANNs)

- Machine learning tool modeled after human brain
- Single node (neuron) composed of inputs,weights, mathematical activation function, and output
- Feed-forward network: input, hidden, and output layers composed of connected neurons



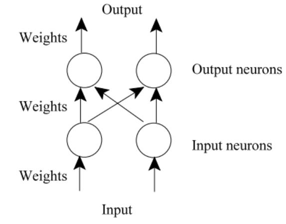


Figure 2: Example neuron with inputs, weights, activation function, and outputs (left) and example neural network (right) [2]

ANN Runge Kutta Scheme

Anastassi et al. (2014)

- Application on two-body problem with RK2
- Custom net input function and identity function for activation
- Order conditions and absolute difference between target and output for training loss function
- Fit remaining coefficients with order conditions using one parameter

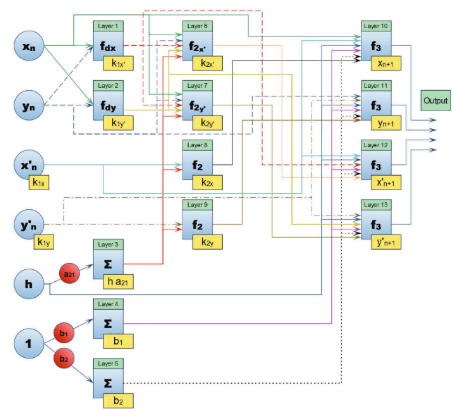


Figure 3: RK2-like neural network applied to two-body problem [3]

Guo et al. (2021)

- Similar net input function N_m using parameter weight vector $\boldsymbol{\theta}$
- Generalized RHS
- Uses regularization term with automatic differentiation at $h = 0$
- Focused applications on dynamical systems

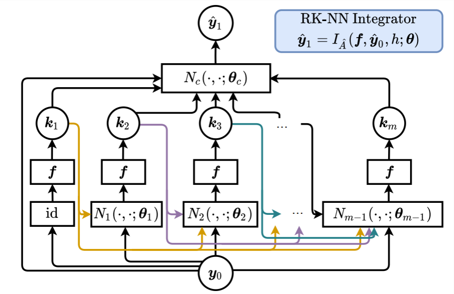


Figure 4: Generalized RK2-like neural network [4]

- Simple test differential equation

$$\frac{du}{dt} = \frac{3u}{t} \tag{8}$$

- Exact solution

$$u(t) = t^3 \tag{9}$$

Training process

- Random uniform distribution of initial conditions over target time interval $[1, 2]$
- Send random distribution initial time vector forward one step for training
- Python TensorFlow with stochastic gradient descent

Figure 5: Python TensorFlow [5]

Project Aims

- Combine method approaches and apply to various target problems
- Understand the importance of order conditions within the loss function used for training
- Uncover the sources of errors with and without order conditions
- Evaluate the performance of methods using various time steps for training
- Determine the correlation between training performance and model performance

Results

Adjusted Fit with Order Conditions

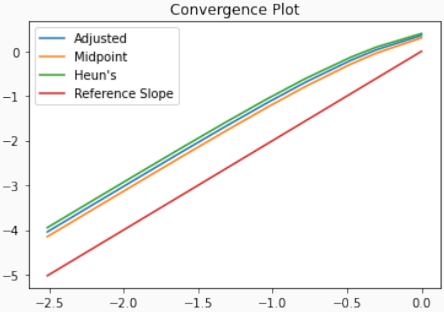
- Tried approach described by Anastassi et al. [3]
- Trained with loss function

$$(p[1]+p[2]-1)^2+(p[2]p[0]-0.5)^2+\sum(u(t)-u_t)^2 \qquad (10)$$

- Training time step $(1/2)^{10}$
- Type of Method
  - Pure: Use $p[0], p[1]$, and $p[2]$ from training
  - Adjusted: Use $p[0]$ from training and solve for $p[1]$ and $p[2]$ using order conditions

Adjusted Fit with Order Conditions Results

Sum of Squares Error Adjusted Scheme with Order Conditions			
Time Step	Adjusted Scheme	Heun's Method	Midpoint Method
$(1/2)^5$	6.0394e-04	9.4259e-04	3.7665e-04
$(1/2)^6$	7.6762e-05	1.2055e-04	4.7627e-05
$(1/2)^7$	9.6734e-06	1.5237e-05	5.9859e-06
$(1/2)^8$	1.2142e-06	1.9151e-06	7.5021e-07
$(1/2)^9$	1.5217e-07	2.4004e-07	9.3898e-08
$(1/2)^{10}$	1.9064e-08	3.0045-08	1.1744e-08



Evaluating Importance of Loss Function

- Tested the results of a control parameter (c) for a convex combination of order conditions with training time step 0.001

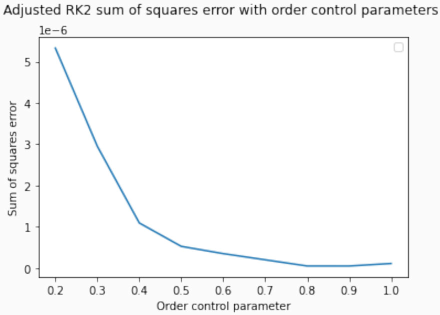
$$c((p[1]+p[2]-1)^2+(p[2]p[0]-0.5)^2))+(1-c)\sum(u(t)-u_t)^2 \tag{11}$$

- As c approaches
  - 1: fully controlled by order conditions
  - 0: fully controlled by accuracy after step

PURE (WITHOUT FORCED ORDER CONDITIONS) - RK2

- Pure sum of squares error from RK2 with order control parameter (c) and testing time step (h)
- Reduced error as c approaches 0 and reliance on order condition increases

RK2 Sum of Squares Error with Order Control (Pure)											
$h \mid c$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$(1/2)^7$	1.10E-01	7.31E-05	7.24E-06	9.77E-06	8.96E-06	8.67E-06	9.80E-06	1.05E-05	9.32E-06	1.10E-05	9.71E-06
$(1/2)^8$	4.15E-02	1.79E-04	4.00E-07	6.63E-07	5.99E-07	6.98E-07	6.98E-07	1.05E-06	9.03E-07	1.22E-06	1.22E-06
$(1/2)^9$	9.00E-02	3.87E-04	1.88E-06	9.86E-07	2.84E-07	1.33E-07	1.20E-08	9.91E-08	2.28E-08	9.66E-08	1.87E-07
$(1/2)^{10}$	5.57E-04	7.93 E-04	5.33E-06	2.95E-06	1.09E-06	5.23E-07	3.47E-07	1.97E-07	4.74E-08	4.71E-08	1.08E-07



Results from Changing Training Time Step (RK2)

Training Timestep Overview

- Tested the results of training on time step values ($h = (1/2)^n$) for $n = 1, 2...20$
- Tested with order constraint parameter $c = 0$ without order conditions

Results

- Errors grew exponentially and the scheme was unstable for training step sizes 0.5, 0.25, 0.125, and 0.0625 for RK2 also 0.03175 for RK4
- Worse performance with larger and smaller training values
- May be an optimal time step for training

Pure (Without forced order conditions)

- Sum of squares error from RK2 with training time step (h_{tr}) and testing time step (h_{te})

RK2 Sum of Squares Error with Training Time Steps (Pure)						
$h_{te}\|h_{tr}$	$(1/2)^5$	$(1/2)^6$	$(1/2)^7$	$(1/2)^8$	$(1/2)^9$	$(1/2)^{10}$
$(1/2)^5$	4.05E-05	0.12106	0.247548	0.308532	0.320471	0.556302
$(1/2)^6$	0.00115	0.00690	0.070263	0.123242	0.145987	0.266812
$(1/2)^7$	0.01451	0.03112	0.031123	0.034886	0.05812	0.119192
$(1/2)^8$	0.05604	0.20775	0.018431	0.001993	0.016412	0.0459
$(1/2)^9$	0.14957	0.62575	0.121894	0.00885	0.000941	0.011967
$(1/2)^{10}$	0.34326	1.4942	0.366257	0.059081	0.004214	0.000859

Loss Function Analysis

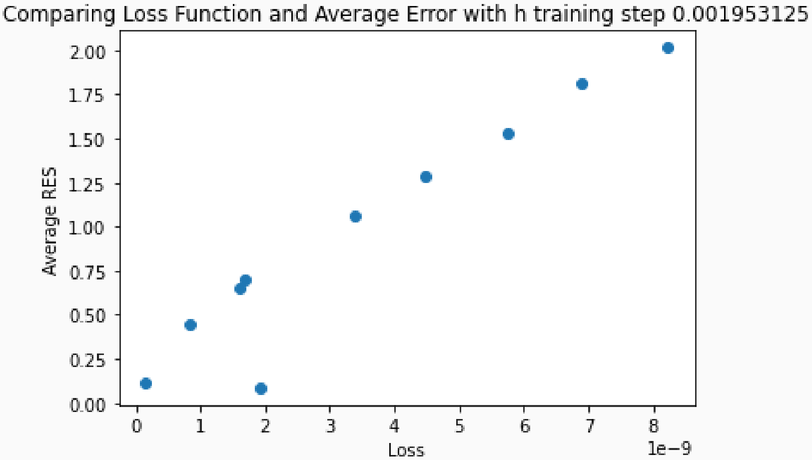


Figure 6: Plot of loss function on x-axis and average RES on right for 10 runs with same training value

Future Work

1. Implement RK4 order conditions and compare with RK2 results
2. Implement regularization term with automatic differentiation at $h = 0$ as described by Guo et al.
3. Try other optimization strategies than stochastic gradient descent
4. Apply to other test problems and generalize across function families
5. Aim to achieve higher order methods without order conditions

References

- Runge-Kutta Methods.
- (PDF) Artificial Neural Networks for Beginners.
  URL https://www.researchgate.net/publication/1956697_Artificial_Neural_Networks_for_Beginners
- A. A. Anastassi, Constructing Runge-Kutta Methods with the Use of Artificial Neural Networks , Tech. rep.
  URL http://link.springer.com/article/10.1007%2Fs00521-
- Y. Guo, F. Dietrich, T. Bertalan, D. T. Doncevic, M. Dahmen, I. G. Kevrekidis, Q. Li, Personalized Algorithm Generation: A Case Study in Meta-Learning ODE Integrators (5 2021).
  URL http://arxiv.org/abs/2105.01303
- TensorBoard — TensorFlow.
  URL https://www.tensorflow.org/tensorboard

Questions?