Analyzing Two-Term Dominant Balances in Ordinary Differential Equations

Prasidh Chhabria, Sarah Goldfarb and Shi Le Wong

Abstract. Qualitative reasoning of mathematical and physical equations have been a key contributor to the advancement of science in the past century. In spite of the exponential development in numerical methods, the ability to simplify equations and describe them qualitatively has been essential in guiding experiments and assessing numerical results for further predictions. Various qualitative methods have been introduced over the years, including the Method of Dominant Balances. In this project, we seek to establish a numerical case for two-term dominant balances in ordinary differential equations (ODEs). We first create an algorithm to generate pseudo-random ODEs to simulate a range of scenarios. Then, we create an algorithm that will take in the symbolic expressions of the random ODEs and output the numerical solution, as well as the dominant balances in different regimes throughout the domain. The accuracy of the dominant balance solution is then evaluated against the numerical solution.

Keywords: Dominant Balance · Ordinary Differential Equations

1 Introduction

Qualitative reasoning of mathematical and physical equations have been key to the advancement of science. Prior to the development of sophisticated numerical methods, many methods for modeling scientific data were through fixed-form parametric models, or restricted model spaces [3]. Subsequently, more general qualitative methods have been developed to provide a set of tools for scientists and mathematicians to use in explaining phenomena. One groundbreaking method developed was the Order of Magnitude Reasoning [1]. In the paper, Raiman described a set of axioms to standardize the qualitative decription and understanding of the magnitudes of terms in any equation. Since then, various other qualitative methods have emerged on the principles of the Order of Magnitude Reasoning. These methods center on making sense of the relative magnitudes of terms to determine which are negligible.

In this paper, we examine one of the simplest and non-trivial methods - a two-term dominant balance. We apply them specifically to Ordinary Differential Equations, and consider the viability of the dominant balance solution vis-a-vis existing numerical solutions.

1.1 Ordinary Differential Equations

A general nth order ordinary differential equation (ODE) has the form

$$y^{(n)}(x) = F(x, y(x), y'(x), ...y^{(n-1)}(x))$$

where $y^{(k)}$ is the kth order derivative of y (i.e. $y^{(k)} = d^k y/dx^k$). ODEs are systems of derivatives and functions, thus they model dynamically changing phenomena well. As simple, well-known example is Newton's Second Law of Motion, which describes a function of the position of a particle at time t that is equal to its second derivative multiplied by a fixed mass constant. ODEs have been applied to a wide range of problems in the fields of mathematics, natural sciences and even the social sciences. Other famous ODEs include the diffusion equation, the Lotka-Volterra equations in ecological studies and the Black-Scholes equation in economics.

Since ODEs are so prevalent and applicable in our every day world, we hope to find a case that would prove the feasibility of a simple solution, such as a two-term dominant balance. This would be crucial to understanding new ODEs that emerge to explain the underlying mechanisms that guide the phenomena it describes.

1.2 Method of Dominant Balances

The Method of Dominant Balances has been well documented for various phenomena, such as acid-base reactions [2]. This method is based on the assumption that dominant behavior will prevail in a system of equations. Hence, by identifying the right dominant behavior, one can simplify solutions. This is especially useful for highly complex equations where there are no general methods to derive exact solutions. However, there has yet to be a quantitative study on the effectiveness of dominant balances, particularly the degree of effectiveness of two-term dominant balances.

A two-term dominant balance predicates on the assumption that behavior of the ODE is determined by only two dominant terms in the entire equation. For the scope of this paper, we are covering third-degree ODEs with between three to five terms.

2 Methods

To quantitatively determine the effectiveness of a two-term dominant balance for general ODEs, we will be testing the relative accuracy of two-term dominant balance solutions on a large set of pseudo-random ODEs. The overall process is split into three parts: (1) Generating of Random ODEs; (2) Solving the ODEs; and (3) Determining the accuracy of the dominant balance by comparing it to the numerical solution. The individual parts are detailed in this section.

2.1 Generating Pseudo-Random ODEs

In order to evaluate the overall effectiveness of two-term dominant balances for ODEs, a diverse set of equations had to be generated. These equations ranged from three to five terms and from first to fifth order. Each time the code was run, the order of the ODE was explicitly indicated, while the total number of terms was randomized with a uniform distribution from three to five inclusively. Note that, for this report, we generated $N = 40$ ODEs, all of which were third order. Additional ODE generation and evaluation will be completed in the coming weeks.

To ensure that the order of the randomly generated ODE resulted in the order that was intended, exactly one of the terms was predetermined to include the nth derivative of y. The rest of the terms were randomly assigned to x, y, y', etc. None of these terms could be present in the equation as the highest order of its own term more than twice.

Once the number of recurrences of each x, y, y', ..., $y^{(n)}$ were determined, each term was recursively (up to 3 times) sent through a function that applied operations such as addition/subtraction and/or multiplication/division and passed the term into a mathematical function. The possible functions included power, exponential, logarithmic, and trigonometric functions.

2.2 Solving Symbolic ODEs

Numerical Solving of Overall ODE To determine the ground truth solution for the ODE that has been generated, we have elected to numerically solve all the random ODEs that have been generated. We used the `solve_ivp` function from `Python`'s `scipy` library for greater stability. In order to take in the symbolic expressions generated into the previous step and input it directly into `solve_ivp`, we first had to rearrange the terms in the ODE so that it is written in the form $y''' = F(x, y, y', y'')$. Then, we used the `lambdify` function from the `sympy` library to convert the equation into a function that takes in x, y, y' and y'' as inputs, and outputs y', y'' and y'''. Then, we used this function as the output of the function plugged into `solve_ivp`.

For consistency, we fixed the time steps so that the algorithm will always solve and output solutions at every 0.1 step in the domain. We randomly generated initial conditions each time, where the values of y, y' and y'' will range between -1 and 1.

Solving Symbolic Dominant Balances We would like to generate dominant balances across the whole domain to compare with the numerical solution. To determine the two terms that would form the dominant balance, we first have to compute the absolute sizes of each of the terms throughout the domain. Using the numerical solutions from the earlier step, we input each term as an equation and used `lambdify` to convert it into a function that takes in x, y, y', y'' and outputs the value of the term. Then, we stored the absolute values of all the terms at every time step. Finally, we iterate over the whole domain to determine the top two largest terms at each time step, and store the indices of the values to derive the two dominant terms. Then, we iterate through the array of indices to determine regimes where the dominant balance is constant. For example, if the dominant terms are the first and second term for all time steps, then there will only be one regime. The segmentation into different regimes allows us to solve for the dominant balance at each regime instead of at each individual time step.

Next, for each pair of dominant terms at each regime, we first checked for the highest order derivative in the pair. Then, we would rearrange the variables in the two terms to form an equation $y^{(n)} = F(x, y, ..., y^{(n-1)})$, where n is the highest order derivative in the two terms. This gives us a new, simplified ODE. Again, we convert it into a function using `lambdify` and use `solve_ivp` to derive the numerical solution for this simplified ODE. At every regime, we will solve the ODE numerically and use the value at the last time step as the initial values for the dominant balance in the next regime.

2.3 Verifying Degree of Accuracy

We used a measure of Mean Relative Error (MRE) defined as:

Let y be the true solution and y_d be the dominant balance solution. Let there be n points in the regime.

$$\text{Mean Relative Error (MRE) in regime} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|y_i - y_{di}|}{|y_i|}\right)$$

Note the absolute value in place of a squared metric, which prevents the error being underestimated across all percentage values (i.e. less than 1).

3 Results

In this early stage study, we generated $N = 40$ random ordinary differential equations of the third order as per the procedures outlined in (2.1). We then solved the ODEs numerically using 'scipy' until a divergence event as outlined in (2.2). Finally, we proceeded to find two-term dominant balances throughout the relevant domain and solve those to yield a continuous solution. Finally, as in (2.3), we assessed the absolute-value variant of the Mean Relative Error for our dominant balance solutions.

Results are summarized below. (See Appendix for full results).

Number of regimes	**Mean length to divergence in timesteps** ($\Delta = 0.1$)	**Number of ODEs**	**Average MRE (%)**
1	24.86	7	9.031
2	29	13	16.871
3	16.56	9	24.290
4+	37.91	11	35.560

Average mean squared error is seen to systematically increase with the number of regimes. (The case of 4+ regimes has been grouped as one case, given that there are not enough cases to accurately draw conclusions about each individual number of regimes in that range).

4 Discussion

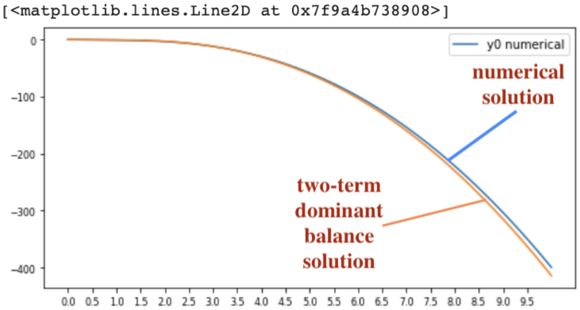
The present preliminary study is limited first and foremost by $N = 40$ equations. This is due to several constraints encountered in generating and solving random ordinary differential equations numerically. Some of these include

- Numerical instability brought about by composite functions in the terms.
- "Division by zero" or "zoo" complex errors raised by numerical solver `solve_ivp`. Efforts were made to avoid division by zero errors (such as by replacing $\sin(x)$ with $\sin(x+1)$ for a positive x domain, however it is still likely that functions with roots at a positive finite x will land up in the denominator of a term in the randomly generated ODE, and these roots will be encountered in the domain of integration of `solve_ivp`.
- The timescale of integration is occasionally thrown off during the multi-part numerical dominant balance integration procedure such that the two series of timesteps (numerical vs. dominant balance) do not match up in length.

```
terms are [cos(x + 1)/10 -exp(x)*log(x + 1)/8 y0 sin(y1 + 1) 10*x**9*exp(y2) y3]
init condition is  [-0.16225677 -0.43678139 -0.75822685]
[-0.16225677 -0.43678139 -0.75822685]
t_stop is  10.0
T step is  0.1
[['start index' 'end index' {'term1', 'term2'}]
 [0 1 {3.0, 5.0}]
 [1 4 {2.0, 3.0}]
 [4 7 {2.0, 5.0}]
 [7 9 {2.0, 4.0}]
 [9 21 {4.0, 5.0}]
 [21 60 {2.0, 4.0}]
 [60 100 {1.0, 4.0}]]
[-0.16225677 -0.43678139 -0.75822685]
[-0.20981229296569182]
[-0.41293868 -0.57432166  0.65686314]
[-0.55363803 -0.35632665]
[-0.55363803 -0.35632665]
[-0.63377128 -0.51580515 -1.8104076 ]
[-3.572794   -5.62602151]
[-103.34226303  -49.36428562]
[<matplotlib.lines.Line2D at 0x7f9a4b738908>]
```

Final t-value (t = 10, timesteps = 100)

Terms of ODE




```
[521] round(100*verifier(soln[1,],regimes_soln,0,False),3)
```

(MRE)

6.309

```
len(regimes)-1
```

Number of regimes

7

Fig. 1: Format of output.

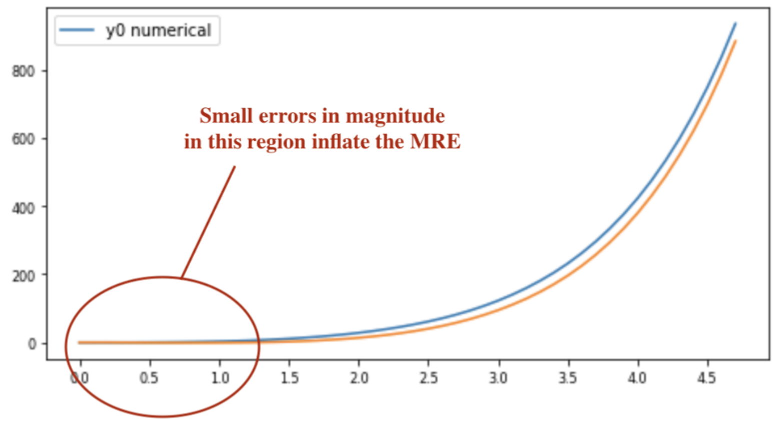
Immediate future improvements include, hence:

- Simplifying the ODE generation process to include less functional composition and more numerical checks in some capacity.
- Passing in the time scale from the results output of the overall numerical solver as the time scale for the dominant balances so that the dominant balances will be computed at the same time steps.
- Investigating which types of functions or equations most often cause issues with integration or dominant balance solutions (i.e. via numerical instability, early divergence, "zoo" errors, or division by zero).

Furthermore, we discuss the limitations as shown by the non-trivial mean relative error (MRE), well above the target of approximately 5% set out by the authors. Some limitations in the use of the dominant balance regimes method and assessment by MRE in this context include:

- The number of regimes is often surprisingly high, particularly in cases where one would analytically expect fewer regimes. This is because we try to classify every point in the domain to a regime, whereas the difference in magnitude of terms may not be significant in a particular sub-domain, so a two-term dominant balance may not suffice.
- In such cases, three or more terms may be comparable in magnitude so making a two-term dominant balance is an oversimplification. There are cases in which adding additional regimes, hence, does not considerably reduce MRE.
- Numerical problems raised earlier, such as complex "zoo" errors or roots / singularities (causing division by zero) mean that rearranging the equation to solve for the highest order term is sometimes impossible, or that solving for the highest order term gives many equations (i.e. the relationship is not one to one, such as for a polynomial term in y''').
- The MRE metric for relative error magnifies very errors of minor magnitude when the solution itself is very small (i.e. order of 10^{-3}. Hence, some dominant balances that perform very well have overestimated MREs since the true numerical solution is very close to 0. See the following image, for instance.

```
T step is  0.1
[['start index' 'end index' {'term1', 'term2'}]
 [0 1 {0.0, 4.0}]
 [1 2 {2.0, 4.0}]
 [2 3 {0.0, 2.0}]
 [3 47 {0.0, 4.0}]]
[-0.06578861  0.17177929 -0.96100207]
[-0.05170651  0.12738826  0.09070711]
[-0.03722720128347608]
[-0.26557108 -2.29266836 -0.18125208]
[<matplotlib.lines.Line2D at 0x7f9a4c519470>]
```




```
[538] round(100*verifier(soln[1,],regimes_soln,0,False),3)
```

MRE:

77.827

Altogether, the random ODE generation must be revised with respect to function composition and numerical stability, while the MRE metric must be reassessed.

5 Appendix

Presented below are the full results of the $N = 40$ random ordinary differential equations generated and solved numerically and via two-term dominant balance.

Sorted #	MRE %	Equation Terms	# of regimes	Length of t
1	0.021	[10*sin(x + 1) -10*y0**8 -y1**4*log(x + 1) log(y1 + 1) -4*log(y2 + 1) y3]	1	1
2	0.029	[-x/6 -8*y0**10 -10*exp(y1) - 6 y3]	1	5
3	3.494	[x sin(x + 1)*tan(y0 + 1)/6 cos(y0 + 1) 6*y1*tan(y0 + 1) cos(y2 + 1) y3]	1	18
4	19.819	[tan(y0 + 1) exp(x)*tan(y1 + 1) y2**2 + 4 y3]	1	7
5	6.948	[cos(x + 1) - 4 sin(y2 + 1) y3]	1	100
6	7	[x**2 -6*x*y1 y3]	1	32
7	25.906	[x**2 - 2 -6*exp(y1) log(y2 + 1) y3]	1	11
8	11.691	[6*x**4 4*tan(y0 + 1) y1**8 exp(y2) y3]	2	9
9	1.806	[8*log(y0 + 1) + 4 -1/(4*y0**6*y1**3) -(tan(y2 + 1) + 6)*exp(y0)/6 y3]	2	5
10	2.336	[-4*sin(x + 1)*tan(x + 1) -sin(y0 + 1)/8 exp(y2) y3]	2	5
11	30.405	[-y1/2 exp(y2) y3]	2	96
12	1.714	[(x + 1)**(-6) -4*y0 - 8 -2*cos(1 - 10/y1**8) y2**5 y3]	2	100
13	21.678	[exp(x)*cos(x + 1)/2 y1**10 + 4 10*exp(y2) y3]	2	8
14	0.898	[-exp(x)*log(x + 1)/4 x x*y2**0.111111111111111 y3]	2	8
15	6.396	[sin(2*x + 2)/16 tan(y0 + 1) -x*(y1**6 + 80) y2 y3]	2	6
16	0.01	[-8*log(x + 1)*log(y0 + 1) tan(y2 + 1) - 6 y3]	2	100
17	139.016	[-x**3/4 tan(exp(x) + 1) -x**8*sin(y0 + 1)/8 6*y1**6 -12*y0*tan(y2 + 1) y3]	2	13
18	0.993	[cos(x + 1) log(y0 + 1) y3]	2	8
19	0.902	[cos(x + 1) -6*x*(exp(y0) + 4) log(y1 + 1) + 2 -16*x**9*exp(y2) y3]	2	6
20	1.474	[-4*x**9*tan(x + 1) y0**7 -log(y0 + 1)/2 + 2 y1 12*y2 2*y0*sin(y2 + 1)/3 y3]	2	13
21	7.06	[10*x**4*(log(x + 1) + 10) x*y0**8/4 6*exp(y1) + 2 y2 y3]	3	12
22	3.088	[-x**6*tan(x + 1)/10 sin(y0 + 1) y1**7 4*y1**3*exp(y2) y3]	3	12
23	13.05	[x**7*(16*tan(x + 1) - 8) -6*x**5*y1 x**4*(cos(y2 + 1) + 2) -4*tan(y2 + 1) y3]	3	20
24	9.517	[6*x**7 -y0*cos(x + 1)/6 -10*x**9*y1 4*y0**4*exp(y2) y3]	3	17
25	108.067	[sin(y0 + 1) y1**4 y3]	3	23
26	10.843	[-6*x*cos(x + 1) x*y0 log(y0 + 1)*tan(x + 1) y1*cos(y0 + 1)/10 y2**2 y3]	3	15
27	1.833	[-262144*x**63 2*cos(y1 + 1) 10*y2 y3]	3	11
28	39.951	[-log(x + 1)*sin(x + 1)/10 x*cos(y0 + 1)/4 x**10*log(y0 + 1)/8 y1 (y2 + 6)*sin(x	3	16
29	25.204	[-x**3/10 x y1**8 sin(y2 + 1) y3]	3	23
30	13.292	[tan(x + 1) -sin(y0 + 1)/8 y1**9*sin(x + 1) (y2 + 1)*exp(8) -10*y1**8*y2 y3]	4	21
31	8.847	[sin(x + 1)*tan(x + 1)/5 -10*x**0.125*y0**8 y0**3*y1**9 32*sin(y2 + 1)*tan(y0	4	7
32	80.292	[x**0.142857142857143 y1 log(y2 + 1)*sin(y0 + 1) y3]	4	44
33	70.541	[5*x**18 -10*tan(y0 + 1) -10*(tan(y1 + 1) + 8)*tan(x + 1) 6*y1 y0/y2**9 y3]	5	19
34	52.635	[-10*exp(x) log(y1 + 1) -4*sin(x + 1)*tan(y1 + 1) -6*cos(10*y2**9 - 1) y3]	5	47
35	1.79	[-sin(x + 1)/4 cos(y0 + 1) -6*cos(y1 + 1) - 4 4*y1**7*sin(y0 + 1) -8*exp(y2)/y0*	6	6
36	41.77	[-4*(x**9)**1.0 + 10 tan(y1 + 1) + 10 y1 + 2 y3]	6	28
37	14.005	[-16*exp(x)*sin(x + 1) tan(y0 + 1) 2*exp(y1) -2*tan(y2 + 1) y3]	6	8
38	28.49	[4*x*cos(x + 1) -8/y1**8 4*x**7*y2**3 y3]	7	100
39	6.309	[cos(x + 1)/10 -exp(x)*log(x + 1)/8 y0 sin(y1 + 1) 10*x**9*exp(y2) y3]	7	100
40	73.186	[x**3 log(y0 + 1) -6*y0**2 -y1**5*cos(x + 1)/8 -2*sin(y2 + 1) y3]	8	37

6 Acknowledgments

We thank Michael P. Brenner at Harvard University for his guidance and support.

References

1. Raiman, O.: Order of Magnitude Reasoning. Artificial Intelligence. (1991). https://doi.org/10.1016/0004-3702(91)90107-U
2. Raiman, O. and Williams, B.C.: Caricatures: Generating Models of Dominant Behavior. Proceedings 6th International Workshop on Qualitative Reasoning. (1992) https://doi.org/10.1.1.88.156
3. Schmidt, M. and Lipson, H.: Distilling Free Form Natural Laws from Experimental Data. Science, 324 (2009). https://doi.org/10.1126/science.1165893