École Polytechnique

BACHELOR THESIS IN COMPUTER SCIENCE

Reparametrizing ODE models by scaling

Author:

Milos Oundjian, École Polytechnique

Advisor:

Gleb Pogudin, LIX

Academic year 2023/2024

Abstract

The goal of this paper is to elaborate an algorithm to simplify differential equations through scaling. The algorithm has been detailed in the paper “Scaling Invariants and Symmetry Reduction of Dynamical Systems” by Evelyne Hubert and George Labahn.

Contents

1 Introduction

Differential equations are a fundamental part of many natural sciences. The fields of Biology and Chemistry use differential models for many things. However, computationally, differential equations are not very easy to solve. In 2013, Evelyne Hubert and George Labahn hypothesized (and maybe implemented I'm not sure I couldn't find their code) an algorithm that would allow simplification of differential equations in order to make them easier to solve.

In their paper "Scaling Invariants and Symmetry Reductions of Dynamical Systems" [1], they discuss the theory behind the algorithm and how it can be implemented.

The main overarching goal of this thesis is to implement an algorithm using scaling invariants and symmetry reduction of dynamical systems to improve the efficiency of solving ODEs computationally. What this means is that given an ODE, say for example the predator-prey model given by the following set of equations:

$$\frac{\mathrm{d}n}{\mathrm{d}t} = n(r(1-\frac{n}{K}-k\frac{p}{n+d})),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = sp(1-h\frac{p}{n})$$

In order to do this, I implemented in Julia, the necessary functions, such as getting the HNF multipliers in both row and column form. Most of the references for this. These include the functions [1]

Furthermore, I was (hopefully) able to integrate this algorithm that I had implemented into a more overarching Julia library 'StructuralIdentifiability.jl'

Notes from Week 1

The goal of this week was to get familiar with the algorithm I will be working on. To do this I read through the main paper I will be working on (http://hal.inria.fr/hal-00668882). My first goal was to understand what Hermitian Normal Forms (HNFs) were. So I read up on them. HNFs fall into two categories, row and column, and this paper requires the use of both. The first issue I encountered was that the library I was using (specifically the Nemo Julia package) did not have an implementation for the column HNF. So I had to come up with my own implementation of it. While at first I looked at the algorithm (Cohen's algorithm from A Course in Computational Algebraic Number Theory), I realized that from the function could be implemented directly using the current 'hnf' and 'hnf_with_transform' function that already existed.

2 Hermite Normal Forms and Hermite Multipliers

2.1 Hermite Normal Forms

Hermite Normal Form can be thought of as the integer matrix equivalent of reduced row echelon form.

It is important to note that for the purpose of this algorithm we need to consider two types of Hermite Normal Forms. By default Hermite Normal form

2.2 Hermite Multipliers

In order to go from a given matrix A to its corresponding unique Hermite Normal Form, a series of row/column operations are done. One can preserve these row operations in a separate matrix commonly called the Hermite Transform. For the purpose of this paper, and based on [1], we will use the term Hermite Multiplier, which we give the same meaning. The result is that we have a matrix V such that

$$A \cdot V = H.$$

Or when we work with HNF column form,

$$V \cdot A = H$$

One of the forms that is used for the algorithm is Hermite Normal Forms and Hermite Multipliers.

Hermite Multipliers, or also known as hermite transforms are...

The Nemo library for Julia provides a function for calculating the HNF but it only works for the HNF row form and not the column form. Hence I created a function that using the algorithm for calculating the HNF row form can calculate the HNF column form. I also used the function that calculated the HNF with transform and it also worked for that. Allowing me to get the HNF transform for the column form as well. Also the initial algorithm did get the normal hermite normal form transform so I also implemented that.

2.3 Algorithmically determining HNF and Hermite Multipliers

There has been work done in algorithmically determining HNFs and Hermite Multipliers

2.4 Practical Implementation

There exists open source libraries that implement the algorithms to calculate the HNF form of a given matrix. For the purpose of this algorithm I have chosen to use the impl

3 Scalings

In order to reduce ODEs through scaling we must define what a scaling is. A scaling is...

4 Methodology

In this section I will explain with an example the full methodology of the algorithm. The example I use is the the Schackenberg Model for a simple chemical reaction with limit cycle as shown in example 7.4 of [1]. The equation for this model is as follows.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a - kx + hx^2y$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = b - hx^2y$$

I explain the process to work on this from now on.

Step 1: Form the function F with the Laurent polynomials of the differential equations.

5 References

[1] Evelyne Hubert and George Labahn. Scaling invariants and symmetry reduction of dynamical systems. *Foundations of Computational Mathematics*, 13(4):479–516, Aug 2013.

A Appendix