ODE Approximations in KeYmaera X

Evelyn Kuo
https://github.com/enkokoro/symbolic_ODE_approximation

December 3, 2021

Abstract

In this paper, we discuss existing tools in KeYmaera X for analyzing ordinary differential equations and propose using numerical methods to generate ODE solution approximations for a new ODE tactic for KeYmaera X. Specifically we discuss the benefits and drawbacks of the existing methods which make use of the actual solution, differential invariants, and differential ghosts. We then explore the theory of ODEs and numerical methods for ODEs and how to generate continuous approximations which end up being some predefined error tolerance away from the actual solution. We then propose a KeYmaera X tactic the work can be added in as and present the implementation for the ODE approximator.

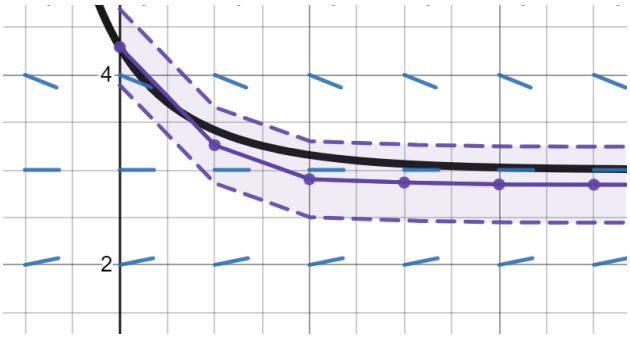

Figure 1: Discrete approximation (purple dots) and continuous approximation (purple solid line) along with actual solution (black) for a differential equation whose vector field is shown in blue. The purple area bounded by the purple dashed lines show values which are within an error tolerance of the continuous approximation.

1 Introduction

Cyber-physical systems wherein computer control interact with physical processes are a large part of our daily lives. From cars, aircraft, and robots, our lives rely on the safety of these complex systems [1]. The correctness of these systems are necessary to ensure the safety of the humans interacting with them and hence deserve proofs to justify the amount of trust we put in them.

Cyber-physical systems belong to the more general class of hybrid systems where discrete dynamics interact with continuous dynamics [1]. Differential equations, which govern continuous dynamics, are a fundamental part of analyzing physical systems as they model how physical systems evolve continuously over time. Accurate modeling of the world through differential equations ensures our provably safe controllers are not just safe within the abstractly modeled world, but also safe within the real world. Additionally, tools to analyze potentially complex differential equations are necessary so that we may prove properties about how quantities of interest to our controllers change through differential equations.

Mathematics has extensive theory regarding ordinary differential equations (ODEs), however it remains that differential equations are hard to solve in general. To combat this, mathematicians analyze differential equations using different approaches which do not require the actual solution by analyzing invariants and variants of the system. In addition, in cases which require knowledge of the actual solution, numerical methods have been developed to generate discrete approximations to the solution which are within an error bound of the actual solution and can be analyzed as a proxy for the behavior of the actual solution.

Because continuous dynamics are an integral part of hybrid systems, hybrid system theorem provers, like KeYmaera X, have tools to analyze continuous dynamics, namely ODEs. KeYmaera X has three ways of analyzing differential equations: generation of the true solution for simple differential equations; differential invariants, which analyze trends in how quantities change; and differential ghosts which make use of a ghost quantity to assist in analyzing the original differential equation. Each method has their own benefits and drawbacks which we analyze later in the paper. This paper aims to extend the existing tools in KeYmaera X for analyzing continuous dynamics by using the mathematical theory of numerical methods for ODEs.

2 Related Work

2.1 General ODE Theory and Definitions

We begin with a summary of relevant ODE definitions and theory [2].

Definition (Initial value problem (IVP))**.** *A initial value problem is a differential equation system*

$$\begin{cases} \frac{d}{dt}x(t) = f(t, x(t)) \\ x(t_0) = x_0 \end{cases} \tag{1}$$

for some $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ *and* $t_0 \in \mathbb{R}, x_0 \in \mathbb{R}^n$ *where a solution would be a function* $x : \mathbb{R} \to \mathbb{R}^n$ *which satisfies* (1)*.*

It is important to tell when solutions exist and are unique, hence we introduce another definition and an existence and uniquenss theorem for IVPs.

Definition (Lipschitz function)**.** *A function* $f(t, y)$ *is Lipschitz with respect to a norm* $\|\cdot\|$ *in* y *on a set* $D \subseteq \mathbb{R} \times \mathbb{R}^n$ *if there exists a constant* $L > 0$ *such that for* $(t, y_1), (t, y_2) \in D$*,*

$$\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\| \tag{2}$$

L *is called a Lipschitz constant for* f *on the domain* D*.*

Theorem 2.1 (Existence and Uniqueness)**.** *Let* $D = \{(t, y) \mid a \leq t \leq b \text{ and } y \in \mathbb{R}^n\}$ *and* $f(t, y)$ *continuous on* D*. If* f *satisfies a Lipschitz condition on* D *in the variable* y*, then the initial value problem*

$$y'(t) = f(t, y(t)), \quad a \leq t \leq b, \quad y(a) = \alpha \tag{3}$$

has a unique solution $y(t)$ *for* $a \leq t \leq b$*.*

In the context of proofs, we want to prove results about a system for a set of states rather than a specific state. Hence, we introduce symbolic initial value problems as well.

Definition (Symbolic initial value problem (SIVP))**.** *A symbolic value problem is essentially the same as the initial value problem (1), only now we may have symbols instead of concrete values where* x_0 *is a symbol, and the definition of* $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ *may contain symbols which we can substitute values into. A solution would be a function* $x : \mathbb{R} \to \mathbb{R}^n$ *which may depend on the symbols* x_0 *and any other symbols of* f*.*

2.2 Existing tools in KeYmaera X

KeYmaera X aims to prove the safety of a controller given any state which satisfies the preconditions. Hence, rules and tactics for dissecting ODEs are in the context of symbolic initial value problems. We discuss the advantages and disadvantages of the existing tools in KeYmaera X which are used for proving properties regarding differential equations.

KeYmaera X has three ways of analyzing differential equations: generation of the true solution for simple differential equations; differential invariants, which analyze trends in how quantities change; and differential ghosts which make use of a ghost quantity to assist in analyzing the original equation. The complete proofs using KeYmaera X tools corresponding the the following examples are included on the Github under KeYmaera X Proofs.

2.2.1 True Solution

For simple differential equation systems, KeYmaera X is able to determine the true solution to a symbolic initial value problem. For example, KeYmaera X is able to solve simple systems like

$$\{x' = a\}$$

for $a \in \mathbb{R}$ constant which has the solution $x(t) = x_0 + at$ where $x_0 \in \mathbb{R}$ is the initial value of x.

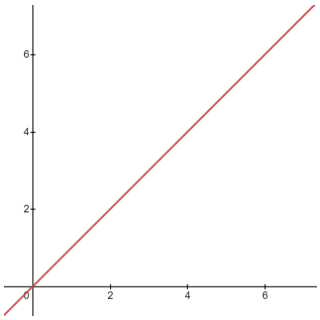

Figure 2: Solution to $\{x' = a\}$ for $a = 1$ and $x_0 = 0$.

The benefits of using the true solution to analyze differential equations is that the true solution allows you to prove everything you can prove about the differential equation, because the solution uniquely determines how the system will evolve over time. However, the method is not well-suited for the vast majority of differential equation systems, as explicit derivation of the solution is often hard or requires transcendental constants (e.g. e and π) or undecidable arithmetic which cannot be represented in KeYmaera X (e.g. trigonometric functions). For example, most students who have taken differential equations will recognize the system

$$\{x' = x\}$$

which has the solution $x(t) = x_0 e^t$, however even just writing the solution requires the use of e which is transcendental and cannot be written in KeYmaera X. Another example is the system for circular motion,

$$\{x' = -y, y' = x\}$$

whose solution requires the trigonometric functions sin and cos which we we cannot write in KeYmaera X because it has undecidable arithmetic due to the trigonometric functions.

2.2.2 Differential Invariants

Another method of analyzing differential equations is through the use of differential invariants. Differential invariants approach differential equations by looking at expressions which stay constant or always evolve in a certain way without solving for solutions to the system explicitly, which is particularly useful for more complex systems, by analyzing the differentials [1]. For example one can show without knowing the solution of the system of equations and simply using

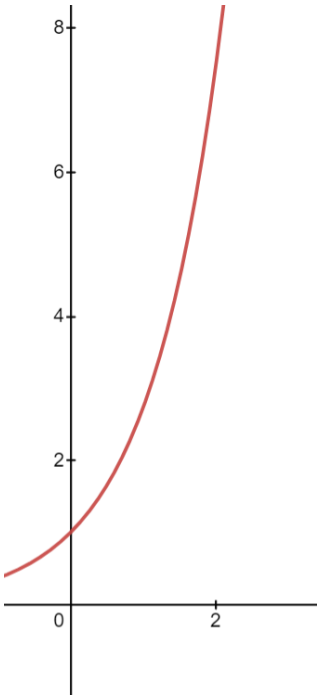

Figure 3: Solution to $\{x' = x\}$ for $x_0 = 1$.

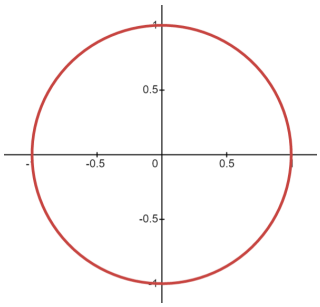

Figure 4: Solution to $\{x' = -y, y' = x\}$. Unlike the other solution plots, the x-axis is not time, but rather the x coordinate and the y-axis is the y-coordinate. As time progresses, the solution would trace along the circle.

dI that the following is valid, namely that if initially $x^2 + y^2 = r^2$ holdes, then after running the differential equation for arbitrary time, we still have $x_{new}^2 + y_{new}^2 = r^2$

$$x^2 + y^2 = r^2 \vdash [\{x' = -y, y' = x\}]x^2 + y^2 = r^2$$

because the differential of $x^2 + y^2 = r^2$ describes how the sides of the equality change with respect to one another is given by

$$(x^2 + y^2 = r^2)' \Leftrightarrow (x^2 + y^2)' = (r^2)' \Leftrightarrow 2xx' + 2yy' = 0 \Leftrightarrow -2xy + 2yx = 0.$$

This means that the quantity $x^2 + y^2$ will always remain equal to r^2 throughout the evolution of the differential equation system.

Similarly, dI also can show that things beyond equalities are preserved by differential equations. For example, it is possible with dI to show the following is also valid

$$x^3 \geq 0 \vdash [\{x' = x^6 + 42x^2 + 96\}]x^3 \geq 0$$

because the change in x^3 is given by $(x^3)' = 3x^2x'$ which is always nonnegative and hence implies that if x^3 is initially nonnegative, it will remain so.

However, dI is fairly limited in some settings where the trend in how quantities evolve go in the opposite direction of what you want to prove. For example a straightforward application of dI to attempt to prove the following (where the explicit solution would be $x(t) = x_0 e^{-t}$)

$$x > 0 \vdash [\{x' = -x\}]x > 0$$

will yield needing to show that

$$[x' := -x]x' \geq 0$$

but applying := results in needing to prove the following which we are unable to

$$-x \geq 0$$

Intuitively, differential invariants are not well-suited for proving the above property because when $x > 0$, $x' < 0$ which makes x become closer to 0 which seems to make make things "worse" because you want to prove that $x > 0$, but x is decreasing and becoming closer to the "bad" value.

In general dI, differential invariants struggle with trying to prove lower bounds for expressions which are decreasing or upper bounds for expressions which are increasing or more generally proving things where the direction of change seems to make it harder to fulfil the desired proof property.

2.2.3 Differential Ghosts

Lastly, KeYmaera X allows analyzing differential equations using differential ghosts. The idea of differential ghosts is to make use of a ghost quantity which may depend on the quantity of interest and analyzing the system of the ghost and quantity of interest jointly to derive conclusions about the original quantity [1]. Back to the previous example

$$x > 0 \vdash [\{x' = -x\}]x > 0$$

we can make use of a ghost $y' = y/2$ and instead analyze the system

$$xy^2 = 1 \vdash [\{x' = -x, y' = y/2\}]xy^2 = 1$$

because $x > 0 \Leftrightarrow \exists y \quad xy^2 = 1$.

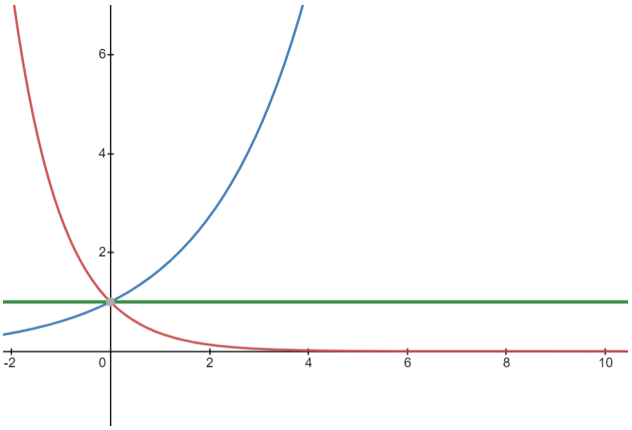

Figure 5: The solution to $\{x' = -x\}$ for $x0 = 1$ is shown in red. The solution to $\{y' = y/2\}$ for $y0 = 1$ is shown in blue. The quantity which we analyze, xy^2 is shown in green. Even though the two quantities by themselves are quite complicated, when we consider the quantity xy^2, it is constant and therefore easy to analyze.

It is counter-intuitive that adding to the system would make the system easier to analyze, however the differential of $xy^2 = 1$ and the strategic choice of the ghost y simplify the analysis of a changing quantity to a constant quantity whose differential is 0 [1]

$$\vdash [x' := -x][y' := y/2](xy^2 = 1)' \Leftrightarrow \quad \vdash [x' := -x][y' := y/2]x'y^2 + x2yy' = 0$$

Again, differential ghosts do not require knowledge of the true solution and we are able to do more things than with just differential invariants. However, there are limitations of what kinds of ghosts we can construct as we need to ensure that the ghost's solution is at least as long as the solution to the quantity of interest which is guaranteed by the assumption that

$$y' = a(x)y + b(x)$$

for some functions of a, b of x [1].

For example,it is fairly common to measure how some sort of energy changes over time. Consider the system

$$\{x' = f(t,x), i' = x^2\}$$

where $i = \int_0^t x^2(t)dt$ (where $i_0 = 0$ is a measure of total energy in a system for a certain period of time. Suppose we wanted to prove that for some upper bound T on time, i never surpasses a critical energy threshold $C > 0$. Hence, we want to show

$$i_0 < C \vdash [\{x' = f(t,x), i' = x^2\}]i < C$$

If we apply dI naively, we end up needing to show

$$\vdash [x' := f(t,x)][i' := x^2]i' \leq 0 \Leftrightarrow x^2 \leq 0$$

which is only true when $x = 0$ which will very likely not be true all the time. We can also try to apply dG. Note that $i < C$ is equivalent to

$$0 < (C - i) \Leftrightarrow \exists y, (C-i)y^2 = 1 \Leftrightarrow \exists y, (C-i)y^2 > 0$$

then, we can try to evaluate the differential of $(C-i)y^2$ to see what kind of differential ghost we may need.

$$P' = ((C-i)y^2)' = -i'y^2 + (C-i)2yy' = -x^2y^2 + C2yy' - i2yy'$$

Note that we can show using dI or the fundamental theorem of Calculus that $i \geq 0$. If we want $P' \geq 0$, we will need y' such that

$$-x^2y + 2Cy' - 2iy' \geq 0$$

If we assume that $y' = \alpha y$ for some α, the above reduces to finding an α such that

$$-x^2 + 2C\alpha - 2i\alpha \geq 0$$

note that $\alpha = \frac{x^2}{2(C-i)}$ or $y' = \frac{x^2}{2(C-i)}y$ works, however, how can we justify that we are not dividing by 0? This would require justifying that $i \neq C$, which we could show if we already knew that $i < C$ forever, but that was precisely what we were trying to show. Hence, differential ghosts are limited in the fact that we need to make sure the ghosts survive for long enough.

2.3 Numerical Methods

Due to the complex nature of mathematices, we often are not able to find exact solutions. Hence, numerical methods have been developed to generate numerical approximations to solutions for problems of interest. We will go through a brief survey of popular numerical methods for solving the initial value problem (1) [2]. For the numerical methods presented here, we assume $t_0 = 0$ and a pre-specified timestep $h > 0$ and will define a sequence $\{y^i\}_{i \geq 0}$ where y^i approximates $x(t_i)$ where $t_i := ih$.

2.3.1 Forward Euler

$$\begin{cases} y^0 = x_0 \\ y^{i+1} = y^i + hf(t_i, y^i) \end{cases} \tag{FE}$$

2.3.2 Runge-Kutta

These define a family of functions of the form

$$\begin{cases} K_1 & = f(t_i, y^i) \\ K_2 & = f(t_i + \alpha_2 h, y^i + h\beta_{2,1} K_1) \\ & \vdots \\ K_N & = f(t_i + \alpha_N h, y^i + h(\beta_{N,1} K_1 + \ldots + \beta_{N,N-1} K_{N-1} \\ y^{i+1} & = y^i + h \sum_{l=1}^{N} a_l K_l \end{cases} \quad \text{(RK General)}$$

of which the popular ones are Modified Euler,

$$\begin{cases} K_1 & = f(t_i, y^i) \\ K_2 & = f(t_i + h, y^i + hK_1) \\ y^{i+1} & = y^i + \frac{h}{2}(K_1 + K_2) \end{cases} \quad \text{(Modified)}$$

Midpoint method,

$$\begin{cases} K_1 & = f(t_i, y^i) \\ K_2 & = f(t_i + \frac{h}{2}, y^i + \frac{h}{2} K_1) \\ y^{i+1} & = y^i + hK_2 \end{cases} \quad \text{(Midpoint)}$$

and Runge-Kutta 4.

$$\begin{cases} K_1 & = f(t_i, y^i) \\ K_2 & = f(t_i + \frac{h}{2}, y^i + \frac{h}{2} K_1) \\ K_3 & = f(t_i + \frac{h}{2}, y^i + \frac{h}{2} K_2) \\ K_4 & = f(t_i + h, y^i + hK_3) \\ y^{i+1} & = y^i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{cases} \quad \text{(RK4)}$$

2.3.3 Other Numerical Methods

The methods listed above are explicit methods where the calculation of y^i only depends on y^j for $j < i$. More specifically, the above are so-called one-step methods as y^i only depends on y^{i-1}. There are other methods where y^i depends on multiple y^j for $j < i$ which are called multi-step methods. There also exist other methods which are called implicit methods where y^i is defined as a function of itself and involves solving a fixed point problem. There are also methods where the timestep h is adaptive and may vary throughout the computation. However, as these methods are also more complex, we do not discuss them for this paper and for the sake of simplicity, only consider explicit, one-step methods.

2.3.4 Errors and symbols

Because numerical methods generate approximations, there are errors in the approximations. There are typically two methods of analyzing error for numerical methods: local truncation error (the error from running the numerical method for one time step) and global error (the accumulated error from running the numerical method for a time horizon). In addition, the numerical methods rely on concrete values for the initial value and a fully defined function f, however in order to make numerical methods work for proofs regarding hybrid systems, we need to convert the numerical methods which generate approximations to the solution to the initial value problem to generate approximations to solutions to the symbolic value problem.

2.4 Image Computation for Hybrid Systems

The general form for a proof for a hybrid system is where one assumes some preconditions of the system, specifies a discrete controller interacting with continuous dynamics, and aims to

show a postcondition which contains safety guarantees of the system.

$$\text{precondition} \rightarrow [\text{discrete controller}; \{\text{continuous dynamics}\}]\text{postcondition} \tag{4}$$

The discrete controller and continuous dynamics can be represented as a hybrid automaton A which includes the state of variables, a directed graph representing discrete transitions and flows representing continuous evolution [3]. We may then consider the set of states Y which obey the precondition and the post-image of the set for automaton A, in order to restructure the problem from (4) in terms of finding properties of the following set

$$Post_A(Y) = \{\text{states reachable from } Y \text{ through the transitions of automaton A}\} \tag{5}$$

As in [3], define for $\epsilon \geq 0$

$$U_\epsilon(Y) := \{x \in \mathbb{R}^n \mid d(x, Y) = \inf_{y \in Y} \|x - y\| \leq \epsilon\}$$

Then, note that $Y \subseteq U_\epsilon(Y)$ because for any $y \in Y$, $d(y, Y) = 0$. Then, through monotonicity, if for some $\epsilon > 0$, $U_\epsilon(Post_A(Y))$ is safe, $Post_A(Y)$ is safe [3]. Note however, that if for some $\epsilon > 0$, $U_\epsilon(Post_A(Y))$ is not safe, this does not mean that $Post_A(Y)$ is unsafe as well, because it might be the case that the approximation tolerance ϵ is too coarse [3]. Previous work in [3] has constructed the AMC algorithm which can be used to adjust for the approximation tolerance ϵ needed to prove the desired property.

3 Motivation

Each of the tools in KeYmaera X have their advantages and disadvantages as well as the situations where they may be used. However, the existing tools are still a bit lacking when we need to prove things which are hard to prove without knowledge of the actual solution. Hence, in these cases, we can look to numerical methods in ODEs to yield approximate solutions which are close enough to the solution to have the similar behavior necessary to prove the desired property. Because KeYmaera X is deals with the symbolic initial value problem, we must adapt existing numerical methods to account for symbolic variables and must consider how we may need to alter our desired property to account for approximation errors.

4 Theory

4.1 Approximation errors with image computation for hybrid systems

Suppose for some continuous dynamics, the actual solution is given by $x(t)$ and suppose for some $\epsilon > 0$ we have a continuous approximation $\tilde{x}(t)$ such that $\forall 0 \leq t \leq T$, $\|x(t) - \tilde{x}(t)\| < \epsilon$. Then if we can generate A_ϵ, a hybrid automaton which is the same as the hybrid automaton A except that instead of the original continuous dynamics, it has a ϵ-approximation of the continuous dynamics for some pre-defined $\epsilon > 0$,

$$Post_A(Y) \subseteq U_\epsilon(Post_{A_\epsilon}(Y))$$

then by monotonicity, we can conclude that $Post_A(Y)$ is safe if we can show $U_\epsilon(Post_{A_\epsilon}(Y))$, the states where variables are within ϵ of an ϵ accurate approximation.

4.2 Discrete approximations to continuous approximations vs. actual solutions

We analyze how to generate continuous approximations from discrete approximation using linear interpolation and how the continuous approximation relates to actual solutions. If we have an actual solution $x(t)$ on the time $[0, T]$, we can consider the discrete approximation where time is discretized: $t_i = ih$ for $i = 0, ..., N$ (for simplicity, we consider the case where $N = T/h \in \mathbb{N}$ i.e. h divides T cleanly) and timestep h and the discrete approximation is given as a sequence

$\{\overline{x}_i\}_{i=0,...,N}$. Suppose $\{\overline{x}_i\}_{i=0,...,N}$ is such that for all $i = 0, ..., N$, $|x(t_i) - \overline{x}_i| < \epsilon_a$ where ϵ_a is an upper bound on the approximation error. For time $t \in [t_i, t_{i+1}]$ for some $i = 0, ..., N-1$, we can take

$$\overline{x}(t) = \overline{x}_i(1 - \frac{t - t_i}{t_{i+1} - t_i}) + \overline{x}_{i+1}\frac{t - t_i}{t_{i+1} - t_i}$$

(weighted average of the neighboring timesteps' approximated values) as an approximation at time t. Then, we may ask ourselves, what can we say about $|x(t) - \overline{x}(t)|$ for $t \in [0, T]$?

Theorem 4.1 (Approximation error for linear interpolation of discrete approximations)**.** *Let $\overline{x}(t)$ be the function on $[0, T]$ generated as a linear interpolation of $\{\overline{x}_i\}_{i=0,...,N}$ which is a discrete approximation of $x(t)$ such that for $t_i := ih$ where $h = T/N$*

$$\forall 0 \leq i \leq N, \quad |\overline{x}_i - x(t_i)| < \epsilon_a.$$

Also suppose $x(t)$ is Lipschitz on $[0, T]$ with constant L, then

$$\forall 0 \leq t \leq T, \quad |\overline{x}(t) - x(t)| < \epsilon_a + Lh$$

Proof. If $x(t)$ is Lipschitz on $[0, T]$ with constant L, then

$$|\overline{x}_i - x(t)| \leq |\overline{x}_i - x(t_i)| + |x(t_i) - x(t)| < \epsilon_a + L|t_i - t|$$

Then for $t \in [t_i, t_{i+1}]$

$$\begin{aligned}
|\overline{x}(t) - x(t)| &\leq \left|\overline{x}_i(1 - \frac{t - t_i}{t_{i+1} - t_i}) + \overline{x}_{i+1}\frac{t - t_i}{t_{i+1} - t_i} - (1 - \frac{t - t_i}{t_{i+1} - t_i})x(t) - \frac{t - t_i}{t_{i+1} - t_i}x(t)\right| \\
&\leq (1 - \frac{t - t_i}{t_{i+1} - t_i})|\overline{x}_i - x(t)| + \frac{t - t_i}{t_{i+1} - t_i}|\overline{x}_{i+1} - x(t)| \\
&< (1 - \frac{t - t_i}{t_{i+1} - t_i})(\epsilon_a + L|t_i - t|) + \frac{t - t_i}{t_{i+1} - t_i}(\epsilon_a + L|t_{i+1} - t|) \\
&\leq \epsilon_a + L|t_{i+1} - t_i| \\
&\leq \epsilon_a + Lh
\end{aligned}$$

□

Hence, to summarize, if the solution is Lipschitz on $[0, T]$, we have a bound on the difference between the approximated solution and the actual solution at all time points within $[0, T]$ which is a function of the approximation error on the discrete time points and the number of discrete time points taken and a Lipschitz constant. However, this may seem a bit fishy since we need niceness assumptions on the solution but we don't want the knowledge of the explicit solution to be necessary! However, due to the fundamental theorem of calculus, we have that for $t_1, t_2 \in [0, T]$

$$|x(t_1) - x(t_2)| = \left|\int_{t_2}^{t_1} x'(t)\right| \leq |t_1 - t_2| \max_{t \in [t_1, t_2]} |x'(t)|$$

Hence a bound on $\max_{t \in [t_1, t_2]} |x'(t)|$ (in essence a maximum derivative bound) will be a Lipschitz constant for x which can be found by analyzing just $x'(t) = f(t, x)$ instead of $x(t)$ directly.

4.3 Finding timestep h for some ϵ tolerance for error

Define a generic one step update rule for $h > 0$, $t \in \mathbb{R}$ and $x \in \mathbb{R}^n$

$$\hat{x} = x + h\Phi(t, x|h) \tag{update}$$

Then, for a particular update rule, we can define a property regarding update rules.

Definition (Stability of update rule)**.** *Consider an update rule given by (update). The update rule is stable with constant $C \geq 0$ in the domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$ if for $t \in \mathbb{R}$ and $y, z \in \mathbb{R}^n$ such that $(t, y), (t, z) \in D$,*

$$|\hat{y} - \hat{z}| \leq |y - z|C$$

Then, we can analyze some numerical methods for the above property.

Lemma (Forward euler has stability with constant $1 + hL$)**.** *Suppose $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz with constant L in y on $D \subseteq \mathbb{R} \times \mathbb{R}^n$. Then, forward Euler is stable on D with constant $C = 1 + hL$.*

Proof. The update rule for forward Euler is given by

$$\hat{x} = x + hf(t, x)$$

Then, for $t \in \mathbb{R}$ such that $(t, y), (t, z) \in D$,

$$\begin{aligned} |\hat{y} - \hat{z}| &= |(y + hf(t, y)) - (z + hf(t, z))| \\ &\leq |y - z| + h|f(t, y) - f(t, z)| \\ &\leq |y - z| + hL|y - z| \\ &= |y - z|(1 + hL) \end{aligned}$$

□

Lemma (Modified euler has stability with constant $1 + hL + \frac{h^2L^2}{2}$)**.** *Suppose $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz with constant L in y on $D \subseteq \mathbb{R} \times \mathbb{R}^n$. Then, modified Euler is stable on D with constant $C = 1 + hL + \frac{h^2L^2}{2}$.*

Proof. The update rule for modified Euler is given by

$$\begin{cases} K1(x) = f(t, x) \\ K2(x) = f(t + h, x + h * K1(x)) \\ \hat{x} = x + \frac{h}{2}(K1(x) + K2(x)) \end{cases}$$

Then, for $t \in \mathbb{R}$ such that $(t, y), (t, z), (t + h, y + h * K1(y)), (t + h, z + h * K1(z)) \in D$,

$$\begin{aligned} |K1(y) - K1(z)| &= |f(t, y) - f(t, z)| \leq L|y - z| \\ |K2(y) - K2(z)| &= |f(t + h, y + h * K1(y)) - f(t + h, y + h * K1(z))| \\ &\leq L|(y + h * K1(y)) - (z + h * K1(z))| \\ &\leq L|y - z| + hL|K1(y) - K1(z)| \\ &\leq L|y - z| + hL^2|y - z| = |y - z|(L + hL^2) \\ |\hat{y} - \hat{z}| &= |(y + \frac{h}{2}(K1(y) + K2(y))) - (z + \frac{h}{2}(K1(z) + K2(z)))| \\ &\leq |y - z| + \frac{h}{2}|K1(y) - K1(z)| + \frac{h}{2}|K2(y) - K2(z)| \\ &\leq |y - z| + \frac{hL}{2}|y - z| + \frac{hL}{2}|y - z| + \frac{h^2L^2}{2}|y - z| \\ &= |y - z|(1 + hL + \frac{h^2L^2}{2}) \end{aligned}$$

□

Lemma (Midpoint method has stability with constant $1+hL+\frac{h^2L^2}{2}$)**.** *Suppose $f : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz with constant L in y on $D \subseteq \mathbb{R} \times \mathbb{R}^n$. Then, midpoint method is stable on D with constant $C = 1 + hL + \frac{h^2L^2}{2}$.*

Proof. The update rule for midpoint method is given by

$$\begin{cases} K1(x) = f(t, x) \\ K2(x) = f(t + \frac{h}{2}, x + \frac{h}{2} * K1(x)) \\ \hat{x} = x + h * K2(x) \end{cases}$$

Then, for $t \in \mathbb{R}$ such that $(t,y),(t,z),(t+\frac{h}{2},y+\frac{h}{2}*K1(y)),(t+\frac{h}{2},z+\frac{h}{2}*K1(z)) \in D$,

$$\begin{aligned}
|K1(y)-K1(z)| &= |f(t,y)-f(t,z)| \leq L|y-z| \\
|K2(y)-K2(z)| &= |f(t+\frac{h}{2},y+\frac{h}{2}*K1(y))-f(t+\frac{h}{2},y+\frac{h}{2}*K1(z))| \\
&\leq L|(y+\frac{h}{2}*K1(y))-(z+\frac{h}{2}*K1(z))| \\
&\leq L|y-z|+\frac{hL}{2}|K1(y)-K1(z)| \\
&\leq L|y-z|+\frac{hL^2}{2}|y-z| = |y-z|(L+\frac{hL^2}{2}) \\
|\hat{y}-\hat{z}| &= |(y+h*K2(y))-(z+h*K2(z))| \\
&\leq |y-z|+h|K2(y)-K2(z)| \\
&\leq |y-z|+|y-z|(hL+\frac{h^2L^2}{2}) \\
&= |y-z|(1+hL+\frac{h^2L^2}{2})
\end{aligned}$$

□

In addition, we can define an version of truncation error for a special case of the above update rule

$$y_{i+1} = y(t_i) + h\Phi(t_i, y(t_i)|h)$$

by

$$\tau_i := |y_{i+1} - y(t_{i+1})|$$

Note that because the local truncation error $\hat{\tau}_i = \frac{\tau_i}{h}$ can be found easily in mathematical literature [2][4][5] for the common numerical methods and in addition to having big O of the truncation error, can be computed in terms of maximums of various higher order derivatives of $f(t,x)$ for $x' = f(t,x(t))$, we will omit discussion of the truncation error for each method here.

Let us define a generic one step method in terms of (update)

$$\begin{cases} \overline{y}_{i+1} = \overline{y}_i + h\Phi(t_i, \overline{y}_i|h) \\ \overline{y}_0 = y(t_0) \end{cases} \qquad \text{(Generic One Step Method)}$$

which generates the sequence $\{\overline{y}_i\}$ which is the finite approximation of $y(t)$.

Then, we can analyze how the error between the finite approximation varies in terms of the timestep size.

Theorem 4.2. *Suppose $\{\overline{y}_i\}_{0\leq i\leq N}$ is a finite approximation of $y(t)$, the solution to $y'(t) = f(t,y), \quad y(0) = y_0$ for $t \in [0,T]$ as given by (Generic One Step Method) with timestep $h > 0$ such that $N = \frac{T}{h}$. If*

1. *f is Lipschitz with constant L on $D \subseteq \mathbb{R} \times \mathbb{R}^n$*
2. *the one step method is stable with constant C on D*
3. *$\tau > 0$ is such that $\forall i$, $|y_{i+1} - y(t_{i+1})| = |y(t_i) + h\Phi(t_i, y(t_i)|h) - y(t_{i+1})| < \tau$ (i.e. a bound on the truncation error)*

then, for $0 \leq i \leq N$

$$E_i := |\overline{y}_i - y(t_i)| \leq \tau\frac{C^i - 1}{C-1} \leq \tau\frac{C^{T/h} - 1}{C-1}$$

in essence, we have a global bound on the error between the finite approximation and the actual solution dependent on the truncation error, T, h and C.

Proof. Note that

$$E_0 := |\overline{y}_0 - y(t_0)| = |y(t_0) - y(t_0)| = 0$$

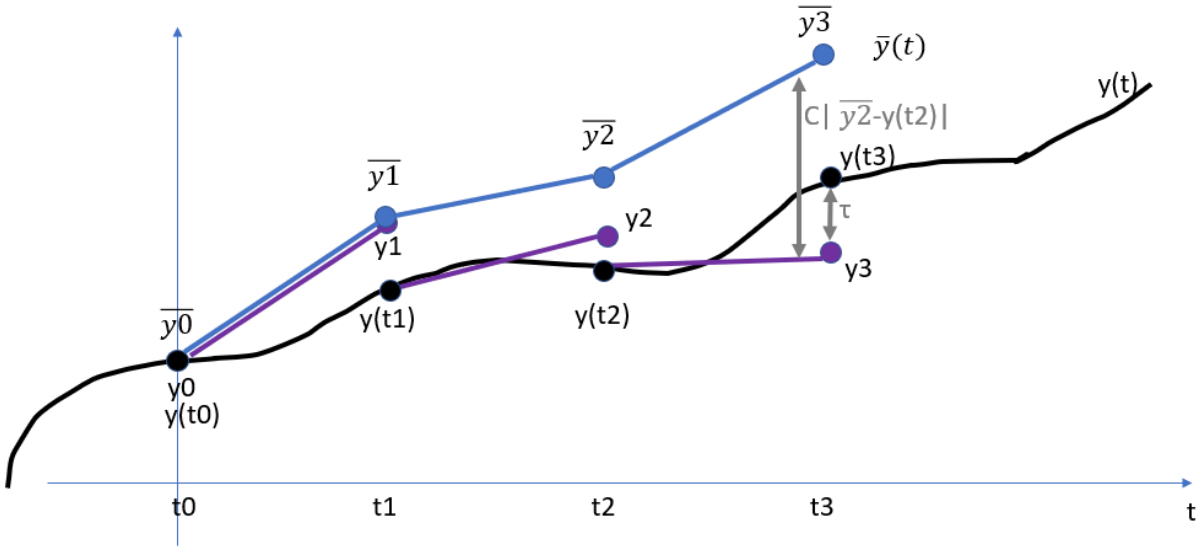

Figure 6: The proof idea is to use triangle inequality to estimate $E_{i+1} := |\overline{y}_{i+1} - y(t_{i+1})| \leq |\overline{y}_{i+1} - y_{i+1}| + |y_{i+1} - y(t_{i+1})|$, the former able to be bounded by CE_i because of the stability and the latter being the truncation error and hence bounded by τ.

Then, for $i \geq 0$, using assumption 2 and assumption 3

$$\begin{aligned} E_{i+1} &= |\overline{y}_{i+1} - y(t_{i+1})| \\ &\leq |\overline{y}_{i+1} - [y(t_i) + h\Phi(t_i, y(t_i)|h)]| + |[y(t_i) + h\Phi(t_i, y(t_i)|h)] - y(t_{i+1})| \\ &\leq |[\overline{y}_i + h\Phi(t_i, \overline{y}_i|h)] - [y(t_i) + h\Phi(t_i, y(t_i)|h)]| + \tau \\ &\leq |\overline{y}_i - y(t_i)|C + \tau = E_i C + \tau \end{aligned}$$

Then, by induction,

$$E_i \leq E_0 C^i + \tau \sum_{j=0}^{i-1} C^j = \tau \sum_{j=0}^{i-1} C^j = \tau \frac{C^i - 1}{C - 1} \leq \tau \frac{C^N - 1}{C - 1} = \tau \frac{C^{T/h} - 1}{C - 1}$$

□

Using Theorem 4.1 and 4.2, we may derive the following

Theorem 4.3 (Timestep h to have approximation error ϵ)**.** *Consider the (symbolic) initial value problem*

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(0) = y_0 \end{cases}$$

for f Lipschitz with constant L on the domain D. Then given $T > 0$, a maximum time bound, and ϵ a desired approximation error bound, a numerical method with truncation error τ which is stable with constant C, we can derive a formula for the timestep h which will yield a continuous approximation within ϵ of the solution $y(t)$.

Proof. From theorem 4.2,

$$\epsilon_a = \tau \frac{C^{T/h} - 1}{C - 1}$$

is a bound on the error between the discrete numerical approximation using timestep h and the actual solution on the time interval $[0, T]$. Then, from theorem 4.1 we can use linear interpolation

to convert the discrete approximation to a continuous approximation where the new bound on the error between the continuous approximation and the actual solution is given by

$$\epsilon_a + Lh = \tau \frac{C^{T/h} - 1}{C - 1} + Lh$$

Then, finding the timestep h amounts to solving the equation

$$\tau \frac{C^{T/h} - 1}{C - 1} + Lh = \epsilon$$

□

Using the above and our previous work on finding the stability constant C and truncation error τ of various numerical methods, we may now generate a timestep h given an input approximation error bound.

4.4 KeYmaera X Rule

The above sections have outlined how to generate a ϵ approximate continuous approximation to the solution to an initial value problem. We now discuss how it may be added as a proof rule in KeYmaera X.

```
\exists e > 0 \forall t (0 <= t <= T & |x-x_approx(x0,t,e)| < e -> P(x))
------------------------------------------------------------------------ [dApprox]
[t:=0; x0 := x; {t'=1, x'=f(t,x) & t <= T}] P(x)
```

where `x_approx(x0,t,e)` is the continuous approximation for $\{x' = f(t, x)\}$ for $0 \leq t \leq T$ with approximation error bounded by `e`. The proof rule is sound because if there exists such an `e`, then then by construction of the continuous approximation, $\forall t, \quad 0 \leq t \leq T$, $|x(t) - x_{approx}(x0, t, e)| < e$ and hence $\forall t, \quad 0 \leq t \leq T$, $P(x(t))$ holds.

5 Implementation

5.1 Overview

For this project, I have implemented an ODE approximation module which functions separately from KeYmaera X. A flowchart of the implementation is provided and the codebase can be found on Github. The idea is that KeYmaera X will provide an ODE $\{x' = f(t, x) \& t \leq T\}$ for some maximum time T and a desired error tolerance ϵ to the ODE approximation module which will output a continuous approximation to the solution back to KeYmaera X.

5.2 ODE Approximation

5.2.1 timestep calculation

Using theorem 4.3 and the stability constant C and truncation error τ for the selected numerical method, the module computes a timestep h which will yield approximations which are within the specified error tolerance ϵ of the actual solution.

5.2.2 symbolic numerical methods

Using the ODE parameters and the timestep h, the module uses numpy and sympy to compute a symbolic discrete approximation to the actual solution.

5.2.3 symbolic linear interpolation

Using sympy, the module then computes a continuous approximation using linear interpolation of the discrete approximation which is within ϵ of the actual solution.

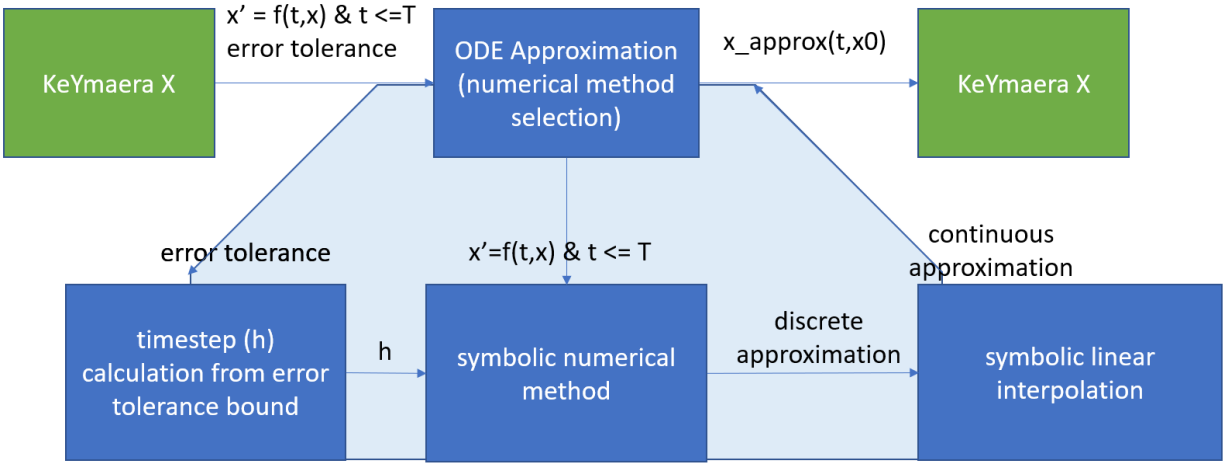

Figure 7: Implementation flowchart

5.3 Further Work

Currently, the implementation does not allow for symbols within the function $f(t, x)$ aside from the symbol for the initial value $x0$, however this can be added into the implementation relatively easily in the future. The current implementation creates a python function representing the continuous approximation, however the goal would be to port over the function into KeYmaera X. To do so, we would require a tactic which can generate a piecewise linear function in KeYmaera X. Further work can also look into other types of numerical methods beyond the one-step methods and other interpolation methods.

6 Discussion

Throught the project, I explored how to utilize numerical methods used in mathematics for tackling ODEs as an additional tactic for analyzing ODEs because I felt that the existing tactics were sometimes lacking and that it would be cool to use what mathematicians use when they need to know about solutions to differential equations which are hard to explicitly solve.

The initial part of the project focused on analyzing the existing methods in KeYmaera X to summarize their respective benefits and drawbacks. In particular, the method of using actual solutions is very useful because the actual solution makes it significantly easier to prove properties of the system because of the total knowledge of how the system evolves, however it is unsuitable for more complex differential equations because it is hard to write the actual solution in closed form in decidable arithmetic for the majority of differential equation systems. Differential invariants were another method which used trends in how quantities evolved to deduce aspects of the system. While differential invariants did not require knowledge of the actual solution and hence were applicable to any differential equation system, some proof capabilities were lost. For example differential invariants are not able to prove when quantities stay above a certain threshold despite the quantity decreasing over time. Differential ghosts are another method of analyzing differential equations which make use of a ghost variable which we analyze together with the original variable to draw conclusions of the original variable. Differential ghosts also had the benefit of not requiring knowledge of the actual solution, however its proof capabilities were limited by the need to ensure that ghost quantities survive for at least as long as the original differential equation.

The second part of the project consisted of the mathematical theory behind constructing continuous approximations of an actual solution. It required figuring out an optimal timestep h dependent on the properties of the system, how tight of an approximation is required, and which numerical method we aim to use. We then were able to make use of existing numerical methods to generate discrete approximations to the actual solution, for which we used linear interpolation

to create continuous approximations of the actual solution. The difficult part was figuring out how to compute a timestep h which will yield at the very end a continuous approximation which is within a pre-defined error tolerance of the actual solution and figuring out how to compute a bound on the global error for the numerical methods as many existing literature cover how to compute the local truncation error for the methods, but not the global error. Because the local truncation error is related to the global truncation error, I was initially planning to use the local truncation error as a proxy for the global error, however I realized that using the local truncation error instead of the actual global error might end up affecting the soundness of the proof tactic and so decided to spend the extra time on proving a bound for the global error.

The last part consisted of the actual implementation of the ODE approximation module, which took all the ideas from the second part of the project and translated them into code. I was initially daunted by the task of converting numerical methods which worked on the initial value problem to work on symbolic value problem, however was able to find pre-existing packages for symbolic computation which were of immense help.

During the project proposal stage, I had wanted to add in a case study where I showed an explicit example which uses ODE approximation to analyze, however due to the time spent on the second part of the project figuring out how to compute the timestep h, I was not able to work through a complete example.

References

[1] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham, 2018.

[2] Richard L. Burden, Douglas J. Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, Boston, 1978.

[3] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *HSCC*, volume 4416 of *LNCS*, pages 473–486. Springer, 2007.

[4] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition, 2008.

[5] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.