Global optimal control using fixed grid methods for differential inclusions

Jonas Lange

Abstract

This report presents a method for global optimisation of differential inclusion initial value problems. To calculate a solution numerically a fixed grid method is used and for optimisation the temporal path is reversed. The first section comprises analysis and description of this method for the general formulation, whereas the second section shows observations and solutions of the method applied to a specific problem.

1 General formulation

A differential inclusion corresponding to a control problem with initial conditions is described by

$$\dot{x}(t) \in F(x(t)) = \{F^{ode}(x(t), u) : u \in [u_{min}, u_{max}]\}, \quad t \in [t_0, t_f], \quad x(t_0) = x_0. \tag{1}$$

In contrast to ordinary differential equations the solution to this problem is not a single trajectory, but a set of solution trajectories $S^F(x_0, [t_0, t_f])$. The elements of this set, the states, are $x(t) \in \mathbb{R}^d$ with initial values of x_0. t is the time running from the initial point t_0 to the end time t_f. $u(t)$ represents a time dependant control variable or parameter. For a known, fixed $u(\cdot)$, furtheron depicted by $\bar{u}(\cdot)$, equation (1) becomes an ordinary differential equation, see equation (2). Equation (3) shows the definition of the reachable set R^F at time t. [1]

$$\dot{x}(t) = F^{ode}(x(t), \bar{u}(t)), \quad t \in [t_0, t_f], \quad x(t_0) = x_0. \tag{2}$$

$$R^F(x_0, t) = \{x(t) : x \in S^F(x_0, [t_0, t_f])\} \tag{3}$$

1.1 Solution procedure

In order to solve (1) numerically a discritisation of time, state space and u is needed. The time is discretised into N time steps $t_0 < ... < t_N = t_f$ with length $h = \frac{T}{N}$. Using a backward [forward] Euler scheme on this time grid a solution for (2) can easily be obtained:

$$x_{k+1} = x_k + h \cdot F^{ode}(x_{k+1}, \bar{u}(t_{k+1})) \tag{4}$$

$$[\ x_{k+1} = x_k + h \cdot F^{ode}(x_k, \bar{u}(t_k))\ ] \tag{5}$$

For the discritisation of the state space a mesh in $\mathbb{R}$: $\Delta_\rho = \rho\mathbb{Z}^d$ with step size ρ is defined. The set

$$B_{\frac{\rho}{2}}(x) = \{x' \in \mathbb{R} : \|x - x'\|_\infty \leq \frac{\rho}{2}\} \tag{6}$$

is called a box with center x. Now a mapping $c_\rho : 2^{\mathbb{R}^d} \to 2^{\Delta_\rho}$, where $2^{\mathbb{R}^d}$ is the power set of $\mathbb{R}^d$, can be constructed

$$c_\rho(A) = (A + B_{\frac{\rho}{2}}(0)) \cap \Delta_\rho, \quad A \in 2^{\mathbb{R}^d} \tag{7}$$

which maps any subset of $\mathbb{R}^d$ A to a subset of Δ_ρ. Figure 1 shows a sketch of this mapping. If the condition is satisfied, that the elements in the discretised reachable set $R^F_{h,\rho}(x_0, t)$ are unique and the set is bounded, the number of elements in this set is limited. In other words the set only comprises unique elements which are also nodes of the mesh Δ_ρ.[1]

The parameter (control variable) u is also discretised into n equidistant steps $u_{min} = u_0 < ... < u_n = u_{max}$. This discretisation converts (1) into n problems of the form (2) for every time step. For clarity the single-valued steps of u are denoted by $\bar{u}$.
The overall solution procedure for one timestep consists of the steps:

1. compute the solutions of (2) for every x_k in $R^F_{h,\rho}(x_0, t = h \cdot k)$ for every u_i, $i = 0, ..., n$ (in the first timestep $R^F_{h,\rho} = \{x_0\}$),
2. apply the mapping c_ρ to these solutions and collect all the results of this mapping in a set,
3. after deleting multiple entries in this set it is equal to $R^F_{h,\rho}(x_0, t = h \cdot (k+1))$.

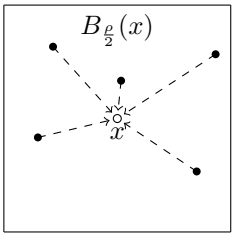

Figure 1: Box $B(x_1)$, the dashed lines represent the mapping $c_\rho(A)$, the filled dots are in A, the unfilled dot is in Δ_ρ

1.2 Optimisation

A global minimisation [maximisation] problem is given if in addition to satifying (1) x^* from (8) is wanted. x^* is a global minimum [maximum] and f is called the objective function.

$$f(x(t_f)) \geq f(x^*(t_f)) \quad [f(x(t_f)) \leq f(x^*(t_f))] \quad \forall\, x(t_f) \in R^F(x_0, t_f) \tag{8}$$

In practice $u^*(t)$, which leads to the optimum, is also wanted. Such problems are referred to as optimal control problems. The advantage of the method described in section 1.1 is that once the reachable set is computed the optimisation reduces to a comparison of a finite number of function values.
All possible discretised states at $t = t_f$ are elements of the set $R^F_{h,\rho}(x_0, t_f)$, so the global minimum $x^*(t_f)$ can be expressed as

$$x^*(t_f) = \underset{x \in R^F_{h,\rho}(x_0, t_f)}{\arg\min} f(x). \tag{9}$$

By saving two additional pieces of information per element of $R^F_{h,\rho}(x_0, t = k \cdot h)$ a hierachical tree structure as depicted in figure 2 can be created. Necessary are the value of u_{k+1} and which element of the set $R^F_{h,\rho}(x_0, t = (k-1) \cdot h)$, the predecessors, was used to calculate it. Since the predecessors of all elements in the sets $S^F_{h,\rho}(x_0, t_i), i = 1, ..., N$ are now known it is possible to reverse the path to the minimum. During this reversal the values of $u^*(t_i)$ can also be collected. This reversal is shown in figure 2 by the dotted line.

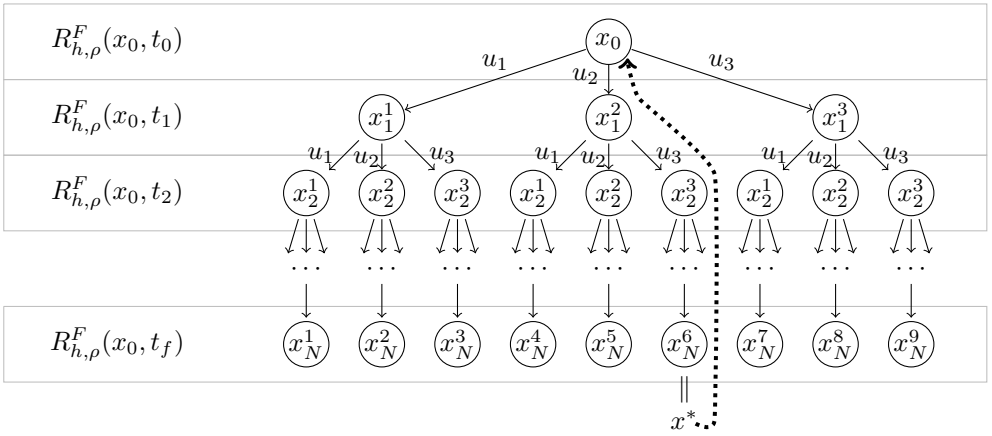

Figure 2: Skematic of tree and reversal, here $n = 3$
(not shown: different u_i might result in the same x)

1.3 Semi-implicit Euler scheme

J. Rieger [2] suggests another scheme for temporal discretisation, the semi-implicit Euler scheme. The differential inclusion (1) is split into two additive parts. The first part, which is dependent on u, is denoted by the set $M(x) \subset \mathbb{R}^d$. The second, independent part is denoted by the vector $m(x) \in \mathbb{R}^d$. Using this $F(x, u)$ can be expressed as

$$F(x) = M(x) + m(x). \tag{10}$$

After applying discretisation of the time derivative equation (11) emerges. The important aspect of this scheme is, that m is calculated implicitly, where M is using the x-value of the previous timestep.

$$\begin{aligned} z_{k+1} &= x_k + h \overbrace{m(z_{k+1})}^{implicit} \\ x_{k+1} &\in z_{k+1} + h \underbrace{M(x_k)}_{explicit} \end{aligned} \tag{11}$$

This approach has the advantage that the costly implicit part only needs to be calculated once per timestep. The part, which has to be calculated n times each timestep due to the discretisation of u, is cheaper, because it is explicit.
To compare this scheme to the standard forward and backward Euler schemes the so called Hopf bifurcation problem is solved. This problem is a differential inclusion, which reads:

$$\begin{aligned} \dot{x} &\in \underbrace{u \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{M(x)} + \underbrace{\begin{pmatrix} -x_2 - x_1(x_1^2 + x_2^2) \\ x_1 - x_2(x_1^2 + x_2^2) \end{pmatrix}}_{m(x)} \\ u &\in [0.3, 0.6] \\ x(0) &= (0.2, -0.2) \end{aligned} \tag{12}$$

This problem is chosen, because an explicit analytical solution can be obtained, if it is transformed into polar coordinates. The two equations decouple, see (13). The solution is a line segment since

the angle φ is independant of u. This line segment ranges from $r(t, u = 0.3)$ to $r(t, u = 0.6)$.

$$\begin{aligned} \dot{r} &\in r(u - r^2) \\ \dot{\varphi} &= 1 \quad \Rightarrow \varphi = t + \varphi_0 \\ u &\in [0.3, 0.6] \\ (r_0, \varphi_0) &= (\sqrt{0.08}, -\frac{\pi}{4}) \end{aligned} \tag{13}$$

Figure 3 shows a comparison of the solution to the Hopf bifurcation for different points in time applying the schemes (5), (4) and (11). Occuring implicit equations are solved by the Newton-Raphson method using analytical derivatives. For comparison the solution to (13) is also plotted, which is obtained by executing Matlabs *ode45* function. All solutions are computed with $h = 0.1$ and $\rho = 0.01$.
Clearly the semi-implicit solution is a lot worse than the other two. This might be due to the fact, that the argument of m in (11) is z_{k+1}, so x is approximated by z. It is obvious that the error of this approxmimation gets larger, the bigger the ratio M/m.

1.4 Error estimates

Beyn and Rieger show that the error for the explicit Euler scheme can be approximated by

$$dist(S^F_{h,\rho}(x_0, [t_0, t_f]), S^F(x_0, [t_0, t_f])) \leq (e^{Lt_f} - 1) \left(\frac{1}{2L}\rho + \frac{h}{2}\rho + Ph \right), \tag{14}$$

if $F : \mathbb{R} \to \mathcal{CC}(\mathbb{R}^d)$ is Lipschitz continuous with respect to the symmetric Hausdorff distance, Lipschitz constant $L > 0$ and $P > 0$ such that $F(x) \subset B_P(0)$ for all x in $\mathbb{R}^d$. Here *dist* is the unsymmtric Hausdorff distance. [1]
Although this estimate and its derivation helps to understand the underlying mathematical principles it is not practical, because the estimate is very coarse.

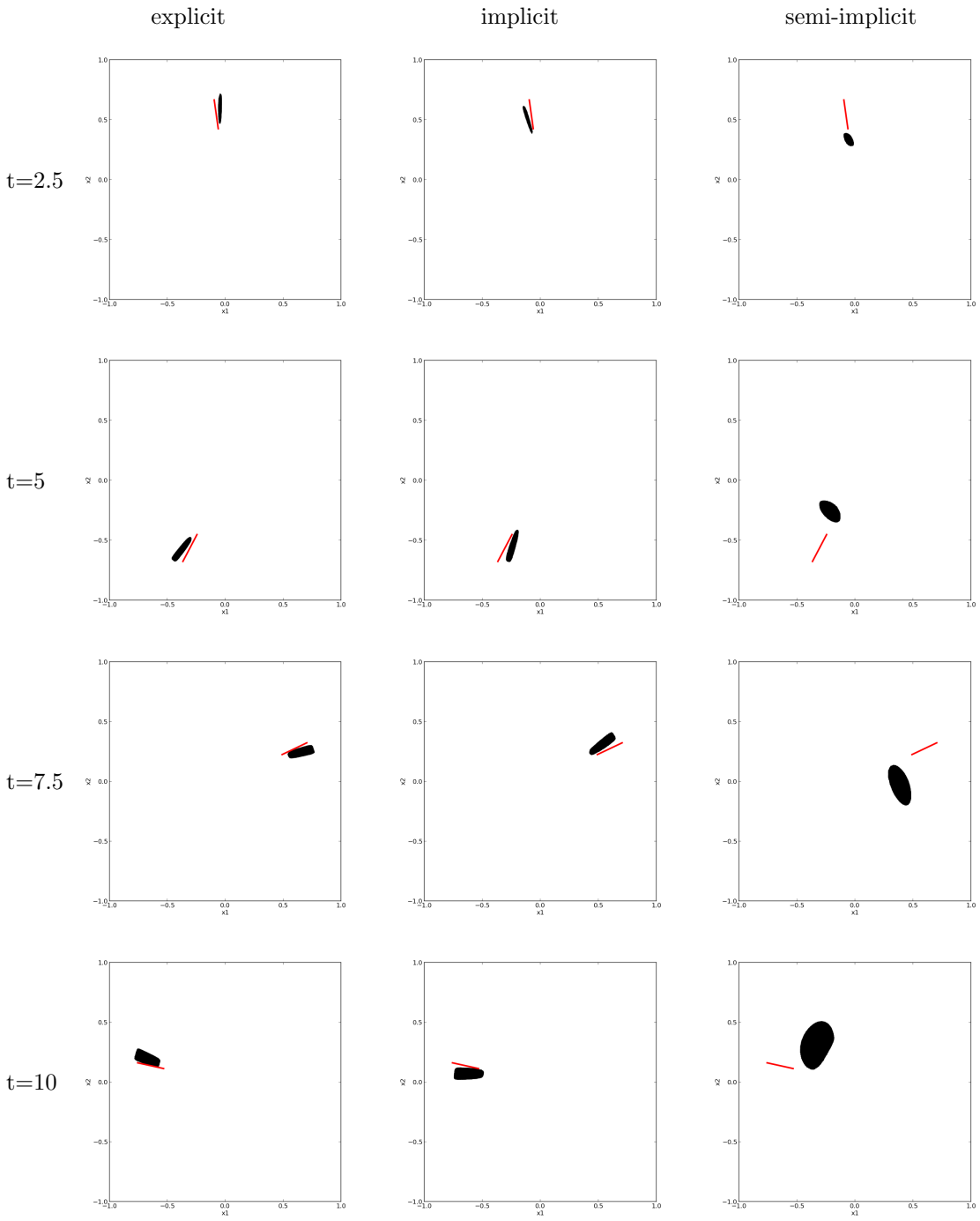

Figure 3: Comparison of different time-discretisation schemes
(The Black set is the computed solution, the red set is the solution of (13))

2 Catalytic reaction problem

The problem solved in this section describes the reactions in an isothermal tubular reactor. The educt is entering the reactor on the inflow side, reacts inside the reactor and leaves together with the products on the outflow side of the reactor. This reactor is modelled as a plug flow reactor, which means the fluid velocity is assumed to be uniform over a cross-section A_c, the flow is in a steady state and the density of the fluid is constant. Assuming incompressibility makes most sense if the fluids are liquids.

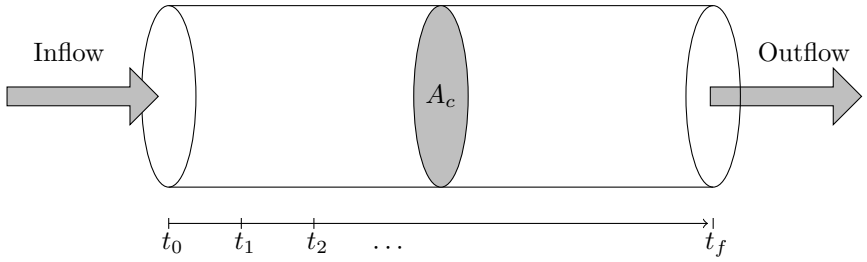

Figure 4: Sketch of the catalyst blend reactor

Figure 4 sketches this type of reactor. It is important to remark, that the variable t does not represent the time in this problem, but the spatial dimension in direction of the reactor axis. The reactions can be described by the chemical equation

$$A \underset{2}{\overset{1}{\rightleftharpoons}} B \xrightarrow{3} C. \tag{15}$$

The reactions taking place in the reactor tube are catalysed. Two catalysts are used, one for reaction 1, the other for reaction 2. The fraction of these catalysts determines the velocity of the reaction between A and B. The wanted control variable u corresponds to this fraction. At the points t_i $i = 1, ..., N$ this fraction can be regulated. The problem to be solved is how to adjust the fractions of the catalysts at t_i to get a maximal amount of the product at the outflow t_f of the reactor tube. Equation (16) shows the mathematical formulation of this setting.

$$\begin{aligned}
\dot{x}_A(t) &= u(t) \cdot (k_2 x_B(t) - k_1 x_A(t)) \\
\dot{x}_B(t) &= -u(t) \cdot (k_2 x_B(t) - k_1 x_A(t)) - (1 - u(t)) \cdot k_3 x_B(t) \\
x_C(t) &= 1 - x_A(t) - x_B(t) \\
t &\in [t_0, t_f] \\
u(t) &\in [u_{min}, u_{max}] \\
f &= x_C(t_f)
\end{aligned} \tag{16}$$

In (16) $x_A(t)$ and $x_B(t)$ are the unknown mole fractures of the educts A and B at position t, where t is the distance from the inflow. k_1, k_2 and k_3 are the velocity constants of reactions 1, 2 and 3 in (15). In all following solutions to this problem the values $k_1 = k_3 = 1$, $k_2 = 10$, $t_0 = 0$ and $t_f = 1$ are used. As stated before the control variable u represents the mole fraction of the catalyst, therefore $u_{min} = 0$ and $u_{max} = 1$.
In the simulated problem it is assumed that the inflow only consists of educt A. So the last piece of information needed to solve this problem, the initial value, has to be

$$\begin{aligned}
x_A(t_0) &= 1 \\
x_B(t_0) &= 0.
\end{aligned} \tag{17}$$

The mathematical formulation of this problem (16) is a differential inclusion with an optimality criterion, so it can be solved with the methods presented in section 1.1.[3]
In the following sections a solution to this problem will be shown and discussed.

2.1 Solution to the differential inclusion

The solution procedure from section 1.1 is now apllied to the problem described in the previous section. Since equation (16) is linear in x no iterative solver has to be used for the implicit solution. After time discretisation a $2x2$ linear system emerges. This linear system can be rearranged to (18). Using a Newton solver would not change the calculation time essentially. Because of the linearity it would converge in one iteration.

$$\begin{aligned} x_{B,k+1} &= \left(\frac{x_{B,k}}{1+10hu+h-u} + \frac{x_{A,k}hu}{(1+hu)(1+10hu+h-u)} \right) \\ &\quad \cdot \left(\frac{(1+hu)(1+10hu+h-u)}{1-10h^2u^2} \right) \\ x_{A,k+1} &= x_{B,k+1} \frac{10hu}{1+hu} + \frac{x_{A,k}}{1+hu} \end{aligned} \tag{18}$$

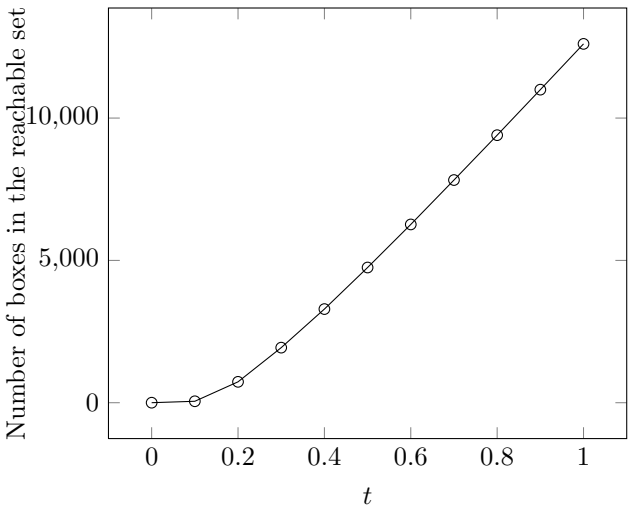

Figure 5: y-axis:$|S^F_{h,\rho}(x_0,t)|$, x-axis: t_i

Figure 5 shows the number of elements in the reachable set for the steps in t with $\rho = 0.0005$. After $t \approx 0.3$ the set grows linearly in t.
Figure 7 shows the reachable set for different t. Here x_A is denoted by x_1 and x_B by x_2. Again $\rho = 0.0005$ and $h = 0.1$ was used.
Furthermore it can be observed, that the fraction of x_B rapidly increases at the beginning. Since the reachable set is close to the line segment between $(0,1)$ and $(1,0)$ after the first timestep, it can be assumed that x_A is converted to x_B. After this large increase of x_B at the beginning the reachable set starts to spread in negative x_A direction, meaning that more of the product x_C is produced, because the sum of all fractions is always equal to 1.
Figure 6 shows the size of the reachable set at $t = 1$, calculated by $|S^F| \cdot \rho^2$ for the variation of ρ. The value for $\rho \to 0$ is unknown, but since the area depends quadratic on ρ the parabolic decrease of the area for decreasing ρ is plausible. From this figure it can be assumed, that the value for $\rho \to 0$ is somewhere close to 2.7.

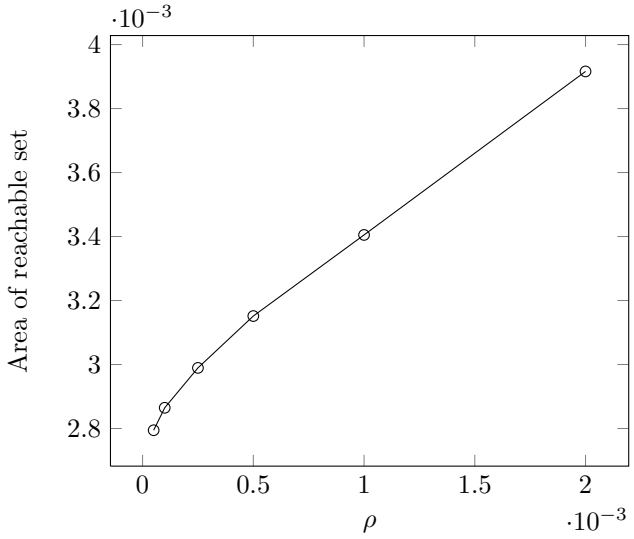

Figure 6: Area of the reachable set at $t = 1$ for different ρ

2.2 Optimisation

Li and Lui show that the catalytic reaction optimal control problem can be analytically solved by employing Pontryagin's maximum principle. They compute the solution and prove that it is globally optimal. The optimal value for the control variable is shown in figure 9 by the green line, the analytical optimum is approximately 0.0480557.[3]
Figure 8 shows the reachable set at $t = 1$ for $\rho = 0.0005$ and the isolines of the objective function, which increases towards zero. The yellow dot and line are the calculated optimum and the path towards it over t. The green dot and line show the same for the analytical solution. The red line shows a solution generated by Matlab, which uses the calculated function for u, but no discretisation of the state space and a much finer discretisation of t. So the red and the yellow line are comparable and the difference between these two lines is caused by the discretisation error.
Figure 9 shows a comparison of the analytical (green) and numerical (red) solution for the catalytic reaction optimal control problem. It is important to notice here that the analytical solution is not realizable in reality because it assumes that the fraction of the catalyst u can be controlled at every point along t. This is usually not the case, the catalyst can only be regulated at some points t_i. Therefore the analytical solution might give a better optimum but is not realizable. Although in this sense the problems solved analytically and numerically are different the solutions look similar.
Figure 10 shows a comparison of the distance between the red and the yellow dot of figure 8 for different values of ρ. It can be observed, that the error gets bigger for smaller ρ between $\rho = 0.5 \cdot 10^{-3}$ and $\rho = 0.05 \cdot 10^{-3}$. This happens because the discritisation error of ρ and t partly cancel each other out, for smaller h this behaviour should disappear.

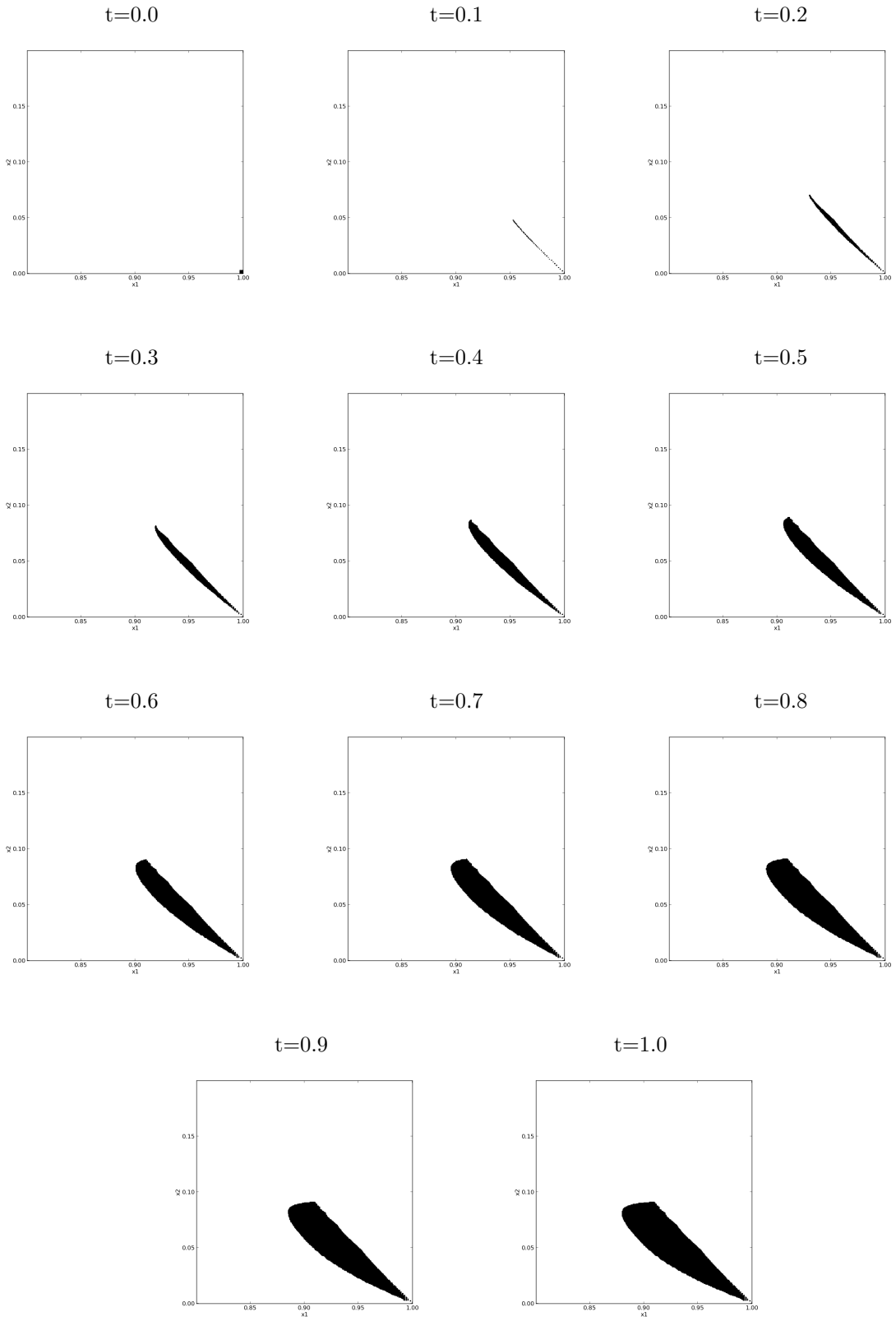

Figure 7: Solution to (16) with $\rho = 0.0005$ and $h = 0.1$

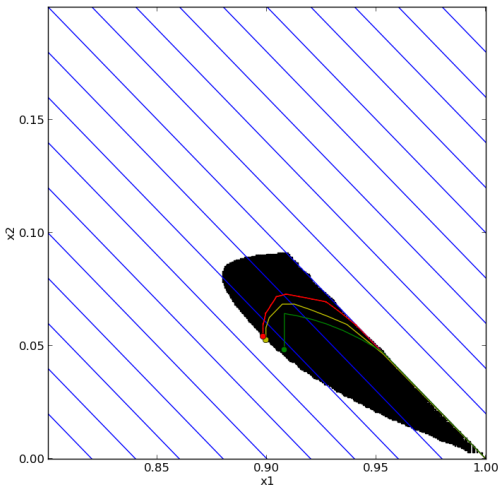

Figure 8: Reachable set at $t = 1$ with calculated optimum (yellow), analytical optimum (green) and Matlab solution (red)

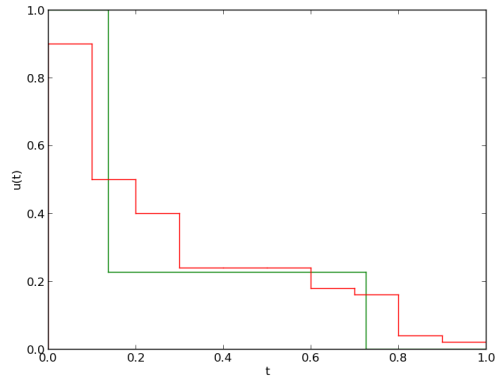

Figure 9: Analytical (green) and numerical (red,$\rho = 0.00005$) optimal control solutions

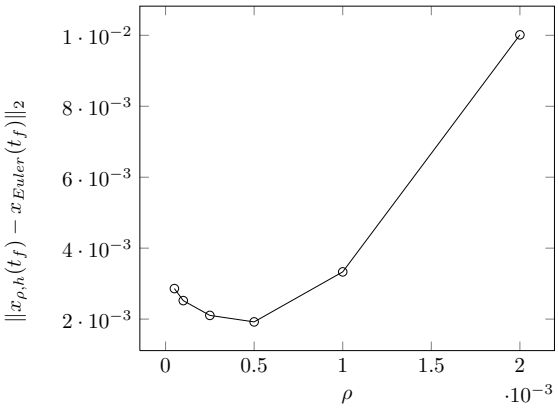

Figure 10: Comparison between a Matlab solution with different discretisation and the fixed grid solution

3 Program structure

The program is written in C++ and is designed to be easily extendable to solve other problems. Therefore the problem-specific routines are encapsulated in a template-like class called “Problem”. The constructor of this class needs an identifier, so it is clear which kind of problem has to be solved. “Problem” has an element whose type is another class called “Domain”. In this class all functions are gathered, which implement functionalities concerning the space discretisation, so different discretisations for the same problem are also easy to implement. One of the most important features of this class is the management of the mapping (7). Throughout the whole program there is a clear distinction between states and the boxes defined in (6). Recall: the states are the center points of these boxes. The states are depicted by the class “State”, the boxes are depicted by their center point in a class “Point”. For the solution of the differential inclusion there are two vectors as elements of the “Problem”-class. These vectors save the “Points” of the reachable set at the current and the previous timestep. Since the optimisation needs the reachable sets of all timesteps another class called “History” executes the task to save all these sets and perform optimisation. Figure 11 shows the file folder structure of the program.

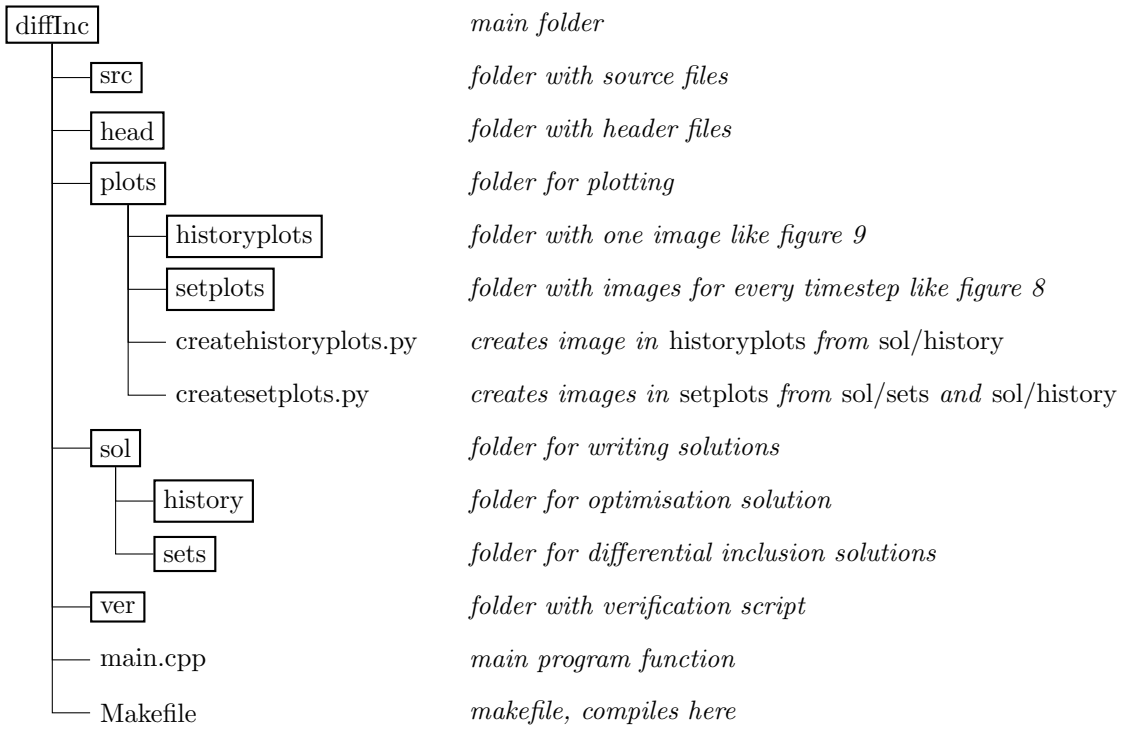

Figure 11: Folder and file structure of the Program

The solutions are written into files. Two types of files are used, the first ones are the reachable sets at specific timesteps, the other type is the solution of the optimisation. This second type is a file inluding the states, which are part of the optimal path and their corresponding value of the control variable.
To visualise and compare solutions to the analytic solution (if it is known) three python scripts are used. “createsetsplots.py” creates figures like figure 8. To create this figure both solutions from the differential inclusion and the optimisation are required. This script also calculates the optimal path for the analytical solution. “createhistoryplots.py” creates figures like figure 9, this script only needs to plot the values in the second type of file. The third script is called “verification.py”, it compares the optimal solution with the solution of an implicit Euler using the analytical optimal control values.

4 Conclusion

This report shows an approach to solve differential inclusion optimal control problems and its application to the catalytic reaction problem. It describes the implementation and shows solutions of numerical experiments. These solutions compare well with the analytical solutions and only exhibit an acceptable numerical error. With the implemented program numerical solutions with sufficient accurancy can be optained in reasonable time.
The next step to speed up the calculation on multi-processor systems would be to parallelise the program. Since the states only depend on their predecessors, the calculation can be split up on multiple processors easily. The difficult part of parallelisation might be to merge the solutions of different processors, because the elements of the reachable set must stay unique.
The computational time of the program largely depends on the discretisation of the state variables. It might also be interesting to analyse whether this approach is still suitable if there are more than two state variables.

References

[1] W.-J. Beyn and J. Rieger, *Numerical fixed grid methods for differential inclusions*, Fakultät für Mathematik, Universität Bielefeld, Bielefeld, Germany, Springer-Verlag, 2007.

[2] J. Rieger, *Semi-implicit Euler schemes for ordinary differential inclusions*, Siam J. Numer. Anal. Vol. 52, No. 2, pp. 895-914, Society for industrial and Applied Mathematics, 2014.

[3] G. Li, X. Liu, *Comments on "Optimal Use of Mixed Catalysts for Two Successive Chemical Reactions*, J Optim theory Appl (2015), Springer Science+Buisiness Media New York, 2014.