Progressor Transformation of Dynamical Systems and Application to Dynamic Optimization*

Trym Arve L. Gabrielsen[1] and Lars S. Imsland[1]

***Abstract*— Motivated by dynamic optimization of aluminium extrusion, this paper demonstrates how a progressor transformation of dynamical models can simplify the implementation of simultaneous methods for dynamic optimization. The implementation of direct collocation becomes difficult when the model exhibits known discontinuous changes, but at unknown times. If the model changes happen at known values in another progressor, the model can be transformed and implemented effortlessly. An example from the extrusion process is given, where the extrusion process model is transformed from time to extrusion length. The transformation allowed the implementation of direct collocation. In addition, the predictability of the transformed model allowed nearly a 50 percent reduction in state variables for describing the aluminium billet throughout the extrusion phase.**

I. Introduction

The motivation behind this paper comes from dynamic optimization of the extrusion cycle in the aluminium extrusion industry. Aluminium extrusion is the process of producing aluminium profiles by means of pushing heated aluminium billets through a shaped die opening. The quality of the product is highly dependent on the extrusion temperature, and accurate temperature control is therefore critical.

Aluminium extrusion is a complex time dependent process. In this paper we focus on alternative variables to time, that a system can progress in, other 'progressors', and explore advantages of using progressor transformed dynamical models in the context of optimal control. The example from the extrusion industry is presented to showcase an application of a progressor transformed dynamical model and various advantages associated with it. A natural progressor of a system is time, as all real world system state trajectories can be written as functions of time, irrespective of the control trajectory. If, for a system, a variable λ is continuous and strictly monotonically increasing in time, then the trajectory of that system can also be written as a function of λ, making it a progressor of that system. The simple way of transforming a dynamical model from time to λ then only requires the bijective mapping between them, $\lambda = \Lambda(t)$, which is shown in this paper.

A transformation of a dynamical model is not novel, and have been used in the context of optimal control for various purposes, for example in [1], [2], [3], [4], which transform a vehicle model to be progressed by the distance along a racetrack, and [5], which progresses a missile by its altitude. However, this paper, we explore the use of a progressor transformation in the context of direct collocation in particular. An obstacle may occur for simultaneous methods for dynamic optimization, such as direct collocation, when changes occur in the dynamical model at unknown times within the problem horizon;

$$\boldsymbol{f}_{dyn.model}(\cdot) = \begin{cases} \boldsymbol{f}^{p,1}(\cdot) & t \in [t_0^p, t_1^p\rangle \\ \boldsymbol{f}^{p,2}(\cdot) & t \in [t_1^p, t_2^p\rangle \\ \vdots & \\ \boldsymbol{f}^{p,h}(\cdot) & t \in [t_{h-1}^p, t_h^p\rangle \end{cases}, \tag{1}$$

where $t^p_{1,\dots,h}$ are the unknown times at which the model changes, and the superscript p denotes the parametric form. This poses a problem since for simultaneous methods, one must implement the dynamical model for the entire horizon of the problem prior to solving the problem. There are ways around this issue, by considering multiperiod [6], or multistage [7], optimization problems, or by introducing Mixed Integer Nonlinear Programming (MINLP), [8]. However, the issue can be circumvented entirely if the model changes happen to be predictable in an alternative progressor of the system. A progressor transformation is also similar to that of time scaling for free end time problems, [9], [10], and may, under the right circumstances, tie neatly into solving the minimum time problem.

The application of a progressor transformation from the aluminium extrusion industry is presented in Section IV, where a discontinuous extrusion model is transformed from time to extrusion length. By transforming the extrusion model from time to extrusion length, one may align the state trajectory discretization points with the discontinuities in the model, to obtain continuously differentiable model constraints for a direct collocation implementation. This approach also results in a significantly reduced optimization problem, due to the predictability of the model changes when progressed by extrusion length. A direct collocation problem to optimize a simplified extrusion model is implemented and solved as demonstration of the technique.

II. Progressor Transformation

Firstly, we define what what we mean by a "progressor" of a dynamical system;

Definition. *Let the state, $\boldsymbol{x}_A(\gamma) \in \mathbb{R}^n$, of the dynamical system A be defined on the continuous set $\gamma \in [\gamma_0, \gamma_f] \subseteq \mathbb{R}$*

*This work has been funded by the "ExtruTeC" project of the Research Council of Norway (Project Number: 313919), in partnership with Hydro Extruded Solutions AS, Cybernetica AS, SINTEF Industry, and Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, `tagabrie,lars.imsland@ntnu.no`

by the Initial Value Problem

$$\frac{d\boldsymbol{x}_A(\gamma)}{d\gamma} = \boldsymbol{f}_A(\boldsymbol{x}_A(\gamma), \boldsymbol{u}(\gamma)), \quad \boldsymbol{x}_A(\gamma_i) = \boldsymbol{x}_i, \tag{2}$$

where $\gamma_i \in [\gamma_0, \gamma_f]$, and $\boldsymbol{u}(\gamma) \in \mathbb{R}^m$ is a control trajectory. Then, any variable $\lambda(\boldsymbol{x}_A(\gamma), \boldsymbol{u}(\gamma)) \in [\lambda_0, \lambda_f] \subseteq \mathbb{R}$ is a progressor of A under control trajectory $\boldsymbol{u}(\gamma)$ iff there exists a continuously differentiable, bijective function $g : [\gamma_0, \gamma_f] \mapsto [\lambda_0, \lambda_f]$, such that

$$\lambda = g(\gamma). \tag{3}$$

Simply put, if there exists a bijective mapping between two variables, $\gamma \in [\gamma_0, \gamma_f]$ and $\lambda \in [\lambda_0, \lambda_f]$, and one of them is a progressor of a system for a given control trajectory, then so is the other. Normally, when working with dynamical systems for control purposes, the dynamical models are written with time as their progressor; $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$. However, there may exist more than one progressor of your system, such as the altitude of a rocket/missile [5], or how far along a racetrack you are in your race car, as in [1], [2], [3], [4]. Note that in both examples, it is reasonable to assume that the altitude and progress on a race track are continuous and strictly monotonically increasing in time for any reasonable input trajectory, thus making them progressors of their respective systems.

Now we derive a simple progressor transformation. A typical dynamical model is

$$\dot{\boldsymbol{x}} = \boldsymbol{f}^t(\boldsymbol{x}, \boldsymbol{u}, t), \tag{4}$$

where $\boldsymbol{x}$ describes the state of some controlled system that is progressed by time, $\boldsymbol{u}$ is the control variable, and $\boldsymbol{f}^t(\cdot)$ are the system dynamics as progressed by time. Let λ also be a progressor of $\boldsymbol{x}(\cdot)$, under the control trajectory $\boldsymbol{u}(\cdot)$. Then one can transform the time-progressed model into the λ-progressed model by using the chain rule;

$$\frac{d\boldsymbol{x}}{dt} = \frac{d\lambda}{dt}\frac{d\boldsymbol{x}}{d\lambda}. \tag{5}$$

By combining (4) and (5), one gets

$$\frac{d\boldsymbol{x}}{d\lambda} = \frac{1}{\dot{\lambda}}\boldsymbol{f}^t(\boldsymbol{x}, \boldsymbol{u}, t). \tag{6}$$

We then define the λ-progressed model as

$$\boldsymbol{f}^\lambda(\boldsymbol{x}, \boldsymbol{u}, t(\lambda)) = \frac{1}{\dot{\lambda}}\boldsymbol{f}^t(\boldsymbol{x}, \boldsymbol{u}, t), \tag{7}$$

and get

$$\frac{d\boldsymbol{x}}{d\lambda} = \boldsymbol{f}^\lambda(\boldsymbol{x}, \boldsymbol{u}, \lambda), \tag{8}$$

with the same solutions as (4), where $\boldsymbol{f}^\lambda(\cdot)$ are the system dynamics when progressed by λ.

Note that since λ is a progressor of $\boldsymbol{x}$, we are guaranteed that $\dot{\lambda} > 0$ by definition, thus avoiding division by zero. We see that if λ is the altitude of a rocket being launched, then one can obtain the altitude-progressed model by simply scaling the time progressed model by the vertical speed of the rocket.

III. Application to Dynamic Optimization

A major category of dynamic optimization methods are direct transcription methods, particularly for complex processes such as metal extrusion. In this paper, we focus on direct collocation. When transcribing your system into discrete time points of the system state and control inputs, one typically assigns a specific point in time, t_k, for each discrete time point, implying a time step Δt_k, between each consecutive discrete time points $[t_k, t_{k+1}]$. In direct collocation, this approach yields a symbolic integration over the horizon, N, typically on the form [6]:

$$\min_{\boldsymbol{x}_{1,\dots,N}, \boldsymbol{u}_{0,\dots,N-1}, \mathcal{C}_{0,\dots,N-1}} J(\cdot) \tag{9a}$$

s.t.

$$\boldsymbol{c}_x(\boldsymbol{x}) \geq 0 \quad \forall k \in \mathbb{N}_{1,N} \tag{9b}$$

$$\boldsymbol{c}_u(\boldsymbol{u}) \geq 0 \quad \forall k \in \mathbb{N}_{0,N-1} \tag{9c}$$

$$\begin{bmatrix} \boldsymbol{p}(\tau_1, \mathcal{C}_k)' - \Delta t_k \cdot \boldsymbol{f}^t_{k,1}(\boldsymbol{p}(\tau_1, \mathcal{C}_k), \boldsymbol{u}_k) \\ \boldsymbol{p}(\tau_2, \mathcal{C}_k)' - \Delta t_k \cdot \boldsymbol{f}^t_{k,2}(\boldsymbol{p}(\tau_2, \mathcal{C}_k), \boldsymbol{u}_k) \\ \vdots \\ \boldsymbol{p}(\tau_d, \mathcal{C}_k)' - \Delta t_k \cdot \boldsymbol{f}^t_{k,d}(\boldsymbol{p}(\tau_d, \mathcal{C}_k), \boldsymbol{u}_k) \end{bmatrix} = \boldsymbol{0} \quad \forall k \in \mathbb{N}_{0,N-1} \tag{9d}$$

$$\boldsymbol{p}(0, \mathcal{C}_k) = \boldsymbol{x}_k \quad \forall k \in \mathbb{N}_{0,N-1} \tag{9e}$$

$$\boldsymbol{p}(1, \mathcal{C}_k) = \boldsymbol{x}_{k+1} \quad \forall k \in \mathbb{N}_{0,N-1}, \tag{9f}$$

where (9a)-(9c) are general objective and trajectory constraints, $\boldsymbol{p}(\cdot)$ is the collocation polynomial, $\tau_i \in [0,1]$ are the collocation points, $\mathcal{C}_k$ are the collocation polynomial coefficients, Δt_k is the size of the discretization interval $[t_k, t_{k+1}]$, $\boldsymbol{f}^t_{k,i}$ are the time progressed dynamics at time $t_k + \tau_i \Delta t_k$, $\boldsymbol{u}_k$ are the control inputs at time t_k, and $\mathbb{N}_{a,b} = \{a, a+1, \dots, b\}$. It is not possible to implement (9) if the model is on the parametric form in (1), and $t^p_{1,\dots,h}$ are the unknown. However, if one could deliberately place the discrete time points $t_{1,\dots,N}$ of the transcribed trajectories $\boldsymbol{x}_{1,\cdot,N}$ and $\boldsymbol{u}_{1,\cdot,N-1}$ at the transition times $t^p_{0,\dots,h}$ of the parametric system model, with $N \geq h$, then one could implement (9d) with

$$\boldsymbol{f}^t_{k,i} = \boldsymbol{f}^{p,k} \quad \forall (k,i), \tag{10}$$

thus only integrating between transitions.

If the model changes happen to be known in a variable λ,

$$\boldsymbol{f}_{dyn.model}(\cdot) = \begin{cases} \boldsymbol{f}^{p,1}(\cdot) & \lambda \in [\lambda^p_0, \lambda^p_1\rangle \\ \boldsymbol{f}^{p,2}(\cdot) & \lambda \in [\lambda^p_1, \lambda^p_2\rangle \\ \vdots & \\ \boldsymbol{f}^{p,h}(\cdot) & \lambda \in [\lambda^p_{h-1}, \lambda^p_h\rangle \end{cases}, \tag{11}$$

where $\lambda^p_{1,\cdots,h}$ are known, and λ happens to be a progressor of the system, then a progressor transformation allows one to transcribe the model in λ rather than time, and implement the model using the same idea as in (10). One can choose the discrete progressor points, $\lambda_{1,\cdots,N}$, to match that of $\lambda^p_{1,\cdots,h}$, such that the model $\boldsymbol{f}_{k,i}^{\lambda} = \boldsymbol{f}_{k,i}/\dot{\lambda}$, is known in every time interval. To this end, (9d)-(9f) can be rewritten as

$$\begin{bmatrix} \dot{\lambda}\boldsymbol{p}(\tau_1, \mathcal{C}_k)' - \Delta\lambda_k \boldsymbol{f}_{k,1}^t(\boldsymbol{p}(\tau_1, \mathcal{C}_k), \boldsymbol{u}_k) \\ \dot{\lambda}\boldsymbol{p}(\tau_2, \mathcal{C}_k)' - \Delta\lambda_k \boldsymbol{f}_{k,2}^t(\boldsymbol{p}(\tau_2, \mathcal{C}_k), \boldsymbol{u}_k) \\ \vdots \\ \dot{\lambda}\boldsymbol{p}(\tau_d, \mathcal{C}_k)' - \Delta\lambda_k \boldsymbol{f}_{k,d}^t(\boldsymbol{p}(\tau_d, \mathcal{C}_k), \boldsymbol{u}_k) \end{bmatrix} = \boldsymbol{0} \quad \forall k \in \mathbb{N}_{0,N-1} \tag{12a}$$

$$\boldsymbol{p}(0, \mathcal{C}_k) = \boldsymbol{T}_k \quad \forall k \in \mathbb{N}_{0,N-1} \tag{12b}$$

$$\boldsymbol{p}(1, \mathcal{C}_k) = \boldsymbol{T}_{k+1} \quad \forall k \in \mathbb{N}_{0,N-1}, \tag{12c}$$

where k now denotes discrete time points in λ. The collocation equations are now implementable as continuously differentiable constraints in an NLP as long as the scaling variable $\dot{\lambda}$ is known. Note that $\dot{\lambda}$ should the be evaluated at the respective progressor point, k, for each equation.

In some cases, the scaling variable, $\dot{\lambda}$, is a control variable or have a bijective mapping to a control variable. From the race car example, we see that the scaling, that is; the speed of the car, has a bijective mapping to the gas pedal position (simplified by ignoring transmission, free rolling, etc.), which can be seen as a control variable. In such cases, the scaling variable is known for 'free', as it is a decision variable, and the NLP can easily be implemented.

By noticing the similarity of this scaling technique to that of the "free end time" technique described in [9], we can simultaneously solve the minimum time problem. For simplicity, we assume that the scaling, $\dot{\lambda}$, is a piecewise constant control variable;

$$\dot{\lambda}(t) = \phi_k, \quad t \in [t_k, t_{k+1}] \quad \forall k \in \mathbb{N}_{0,N-1}. \tag{13}$$

The minimum time problem is then solved by rewriting (9a) as

$$\min_{\boldsymbol{x}_{1,\dots,N}, \boldsymbol{u}_{0,\dots,N-1}, \mathcal{C}_{0,\dots,N-1}} \left(t_f = \sum_{k=0}^{N-1} \Delta t_k = \sum_{k=0}^{N-1} \frac{\Delta\lambda_k}{\phi_k} \right), \tag{14}$$

where t_f is the final time of the state trajectory. Note that ϕ_k is a part of the decision variable $\boldsymbol{u}_k$, and that the step in lambda, $\Delta\lambda_k$, is known.

IV. Example from Aluminium Extrusion

Now we introduce an example from the extrusion industry, where the aforementioned advantages of a progressor transformation become clear. A typical extrusion process consists of

- a metal cylinder/'billet' that will be extruded,
- a container in which the cylinder is placed,
- a 'die' through which the metal is extruded and shaped,
- and a piston/'ram' that pushes on the billet from behind, forcing it through the die.

A model that describes the heat in the billet-container-die system is typically progressed by time [11], [12], and are described by Partial Differential Equations (PDEs). Consider the form

$$\begin{aligned} \frac{dT(\bar{r},\bar{x};t)}{dt} = \; & -v(\bar{r},\bar{x};t)\frac{\partial T(\bar{r},\bar{x};t)}{\partial \bar{x}} \\ & + \alpha\left(\frac{1}{\bar{r}}\frac{\partial}{\partial \bar{r}}\left(\bar{r}\frac{\partial T(\bar{r},\bar{x};t)}{\partial \bar{r}}\right) + \frac{\partial^2 T(\bar{r},\bar{x};t)}{\partial \bar{x}^2}\right) \\ & + \tilde{\Phi}(T;v) + \tilde{\Psi}(T;v), \end{aligned} \tag{15}$$

where $T(\bar{r},\bar{x},t)$ is the temperature at point $(\bar{r},\bar{x})$ at time t, $v(\cdot)$ is the axial velocity of the metal, α is the diffusivity of the metal, and $\tilde{\Phi}(\cdot)$ and $\tilde{\Psi}(\cdot)$ are heat generation terms due to viscous dissipation and area reduction respectively. Due to the complexity of the extrusion process, (15) is typically discretized by finite difference schemes [13] into ordinary differential equations (ODEs);

$$\frac{d\boldsymbol{T}(\bar{r},\bar{x},t)}{dt} = \boldsymbol{f}^{t,ext}(\boldsymbol{T}(\bar{r},\bar{x},t), v_{ram}(t), L(t)), \tag{16}$$

where $\boldsymbol{f}^{t,ext}(\cdot)$ is the time progressed extrusion model, the ram speed v_{ram} is the control input, and $L(t)$ is the extrusion length.

In the extrusion industry, it is not only common, but mandatory, that the ram speed is always positive. That is, the ram never stops nor goes backwards, due to safety concerns. We then have: $v_{ram} = \dot{L} > 0 \implies L(t)$ is strictly monotonically increasing. Neither can the ram speed jump, as this would require infinitely large forces in the system, thus L is continuously differentiable. Therefore, L is a progressor of the extrusion process under all feasible control trajectories, and one can write the dynamical model in terms of extrusion length by simply scaling the model by the ram speed, as seen from (6);

$$\frac{d\boldsymbol{T}(\bar{r},\bar{x},L)}{dL} = \frac{1}{v_{ram}(L)} f^{ext}(\boldsymbol{T}(\bar{r},\bar{x},L), v_{ram}(L), L). \tag{17}$$

A version of the extrusion model is used as an example, containing the billet and control volumes for the downstream aluminium, and some surrounding steel components. The discretization of (15) into ODEs is done according to a spatial discretization of the billet and the other modelled components into control volumes. These control volumes are fixed in space. During extrusion, the whole billet moves towards the die, such that aluminium is gradually leaving the control volumes from the other end. This effect is illustrated in Fig. 1, along with the partitioning of the billet into cells. The part of a control volume that is occupied by aluminium is referred to as a 'cell'. As the cells gradually decrease in size, the model gradually changes, implying an extrusion length dependent model; $f^{ext}(\cdot\ , L)$. Not only does the model undergo continuous change in L, as the backmost control volumes become completely empty, and the cells 'die', the

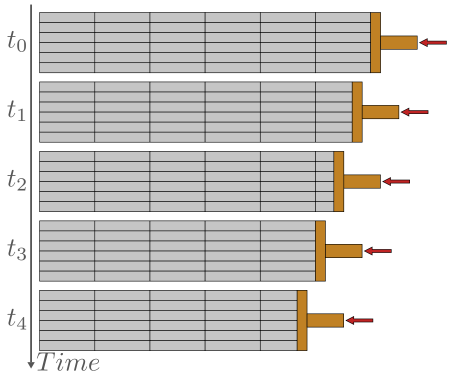


Fig. 1: Illustration of cells 'dying' during the extrusion process. The billet is shown in gray, and the ram is represented by the orange piston, pushing the billet in the direction indicated by the red arrow. In the first three time instances, the billet has a constant number of cells, and by time t_3, the number of cells have decreased, causing a discontinuous change in the model equations.

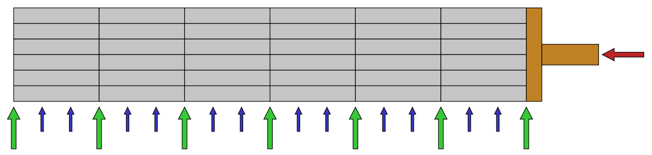


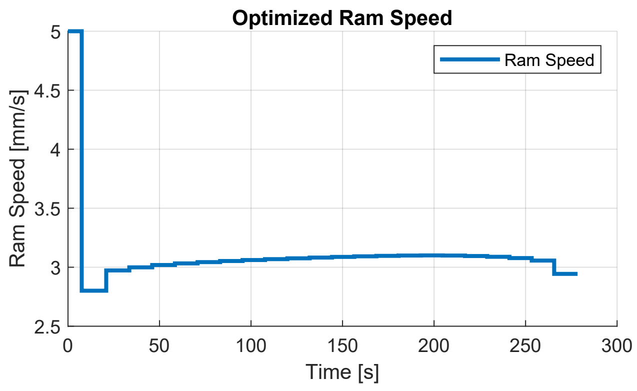
Fig. 2: Depiction of the discretization for the state and input trajectories. Vertical arrows indicate the extrusion length assigned to the transcribed state and control variables, where the extrusion length dimension, L, is horizontal from right to left. The green arrows are where the discretization points align with the discontinuities in the extrusion model.

model experiences a discontinuous change;

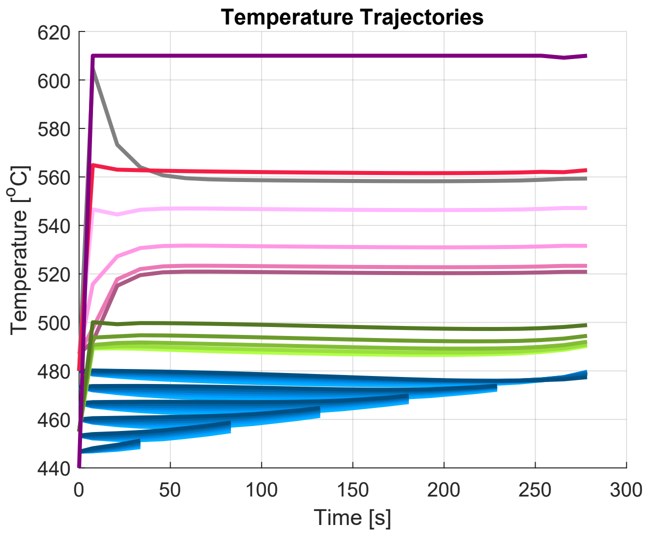
$$\boldsymbol{f}^{ext}(\cdot\,,L) = \begin{cases} \boldsymbol{f}^{p,1,ext}(\cdot\,,L) & L \in [L_0^p, L_1^p\rangle \\ \boldsymbol{f}^{p,2,ext}(\cdot\,,L) & L \in [L_1^p, L_2^p\rangle \\ \vdots & \\ \boldsymbol{f}^{p,nx,ext}(\cdot\,,L) & L \in [L_{nx-1}^p, L_{nx}^p\rangle \end{cases}, \tag{18}$$

By using the extrusion length progressed model (17), (18), the dynamics can be discretized such that the transcribed trajectories align with the discontinuities in L, as depicted in Fig. 2. Equation (12) can then implemented without the need to keep track of the extrusion length, and, most prominently, without handling the discontinuities in the model.

This technique is used in an implementation of an NLP that optimizes the ram speed with respect to extrusion time, using CasADi [14] and the IPOPT algorithm [15], the solution of which is shown in Fig. 3. The resulting trajectories are not verified by experimentation, though they resemble trajectories previously recorded in industry. In addition, they correspond with what is expected based on basic understanding and intuition of heat diffusion and metal extrusion. Nevertheless, the convergence of the NLP shows



(a) Optimal Ram Speed.



(b) Temperatures: ■-peak, ■-port, ■-feeder, ■-billet, ■-plate, ■-die. Darker graphs represent cells with higher radial coordinate

Fig. 3: The state and input trajectories as a solution to a dynamic optimization problem, optimizing the extrusion process with respect to total extrusion time using direct collocation with the progressor transformation approach.

the effectiveness of the technique, as the same problem without a progressor transformation would result in a complex Mixed Integer Nonlinear Program (MINLP), a multi-stage approach, or some approximation technique.

To acquire the times at which the discrete time points of the trajectories occur, one simply has to re-scale using the ram speed, v_{ram};

$$t_k = \sum_{i=0}^{k-1} \frac{\Delta L_i}{v_{ram,i}}. \tag{19}$$

Aside form the sheer ability to implement the discontinuous model as an NLP and effectively solve the control problem, there is another major benefit to the progressor transformation technique. In a formulation based on a time progressed model, one needs to account for all cells at every discrete time point, at least the worst case, since one does not know at what times the various cells die. By transcribing in extrusion length, in which the cells are also defined, one

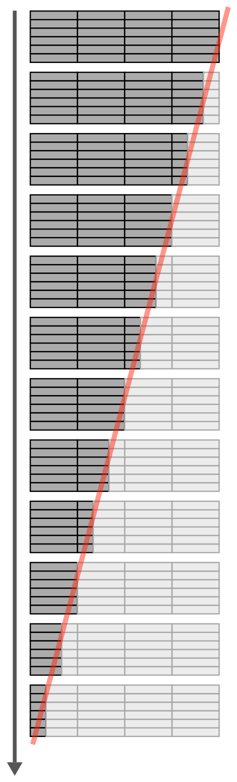


Fig. 4: A visualization of the reduction in decision variables for the billet cells in the NLP implementation, due to discretizing in extrusion length rather than time. Dark cells represent active cells. The arrow indicates increasing extrusion length.

knows exactly what cells are active at every discrete time point, allowing one to only assign state variables for the active cells. Using a discretization such as shown in Fig. 2, the number of state decision variables necessary in the NLP is reduced by

$$\frac{w_{L,billet}}{w_{t,billet}} = \frac{\frac{1}{2}(1+d)n_r(n_x+1)N}{(1+d)n_r n_x N} = \frac{1}{2} + \frac{1}{2n_x}, \qquad (20)$$

where $w_{L,billet}$ and $w_{t,billet}$ are the total number of active cells across every discrete time point when discretized in L and time respectively, n_r and n_x are the number of cells in the billet in the radial and axial dimensions respectively, and N is the number of discrete time points. The reduction of decision variables is visualized in Fig. 4. We see that the number of variables necessary to describe the billet states is nearly halved when using the technique covered in this paper.

V. Conclusion

In the beginning of this paper we defined the term "progressor" for a dynamical system, followed by a description of how to transform a dynamical model to be progressed by a different variable. When implementing direct collocation on a dynamic model that is dependent on alternative progressor of the system, performing such a progressor transformation on the model allows integration of the model without the need to keep track of the alternative progressor. In addition, we see that one can implement a model that is discontinuous at unknown times as an NLP, if its discontinuities occur at known times in another progressor of the system, and the scaling variable for that particular progressor transformation is known. That way, one avoids the use of cumbersome and slow MINLPs or other methods of circumventing the problem. We take note of the fact that this is only possible when the bijective mapping between the two progressors is known. That is, the scaling variable, $\dot{\lambda}(t)$, is available.

From the aluminium extrusion example, we saw that the predictability of the model changes that arose from transforming the model from time to extrusion length, allowed us to align the discretization points of the state trajectory with the discontinuities of the model. It also allowed us to predict what cells were active at what discretization points, which nearly cut the necessary number of decision variables for the billet in half.

By employing the technique presented in this paper, a direct collocation NLP of the extrusion process is implementable, allowing optimization over a long horizon. This opens up the possibility of optimizing open loop trajectories over the entire extrusion cycle, and predict optimal references for the billet preheating phase. Reference trajectories for a hierarchical control scheme, as in [1], may also be produced in this way, for the extrusion phase. Such a control scheme may contribute to an increase in production overall.

References

[1] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2397–2403.

[2] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7559–7565, 2014.

[3] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager, "Optimal control of formula 1 race cars in a vdrift based virtual environment," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 907–11 912, 2011.

[4] F. Kehrle, "Optimal control of vehicles in driving simulators," Ph.D. dissertation, Diploma thesis, Uni Heidelberg, 2010.

[5] X. Liu, Z. Shen, and P. Lu, "Exact convex relaxation for optimal flight of aerodynamically controlled missiles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1881–1892, 2016.

[6] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[7] S. Gros and M. Diehl, "Numerical optimal control (draft)," 2020.

[8] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[9] J. Andersson, "A general-purpose software framework for dynamic optimization (een algemene softwareomgeving voor dynamische optimalisatie)," 2013.

[10] H. Hong, A. Maity, and F. Holzapfel, "Free final-time constrained sequential quadratic programming–based flight vehicle guidance," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 1, pp. 181–189, 2021.

[11] C. F. Cuéllar Matamoros, "Modeling and control for the isothermal extrusion of aluminium," Ph.D. dissertation, ETH Zurich, 1999.

[12] M. N. Ã-zisik, M. N. Özışık, and M. N. Özısık, *Heat conduction*. John Wiley & Sons, 1993.

[13] J. C. Strikwerda, *Finite difference schemes and partial differential equations*. SIAM, 2004.

[14] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[15] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, pp. 25–57, 2006.