# Non-linear Model Predictive Control for constrained robot navigation in row crops

Trygve Utstumo*† , Therese W. Berge‡ , Jan Tommy Gravdahl*
*Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway.
‡Bioforsk - Norwegian Institute for Agricultural and Environmental Research, Plant Health and Plant Protection Division, NO-1430 Ås, Norway. †Adigo AS, NO-1405 Langhus, Norway. E-mail: trygve.utstumo@adigo.no

***Abstract*—Vehicles which operate in agricultural row crops, need to strictly follow the established wheel tracks. Errors in navigation where the robot sways of its path with one or more wheels may damage the crop plants.**

**The specific focus of this paper is on an agricultural robot operation in row cultures. The robot performs machine vision detecting weeds within the crop rows and treats the weeds by high precision drop-on-demand application of herbicide.**

**The navigation controller of the robot needs to follow the established wheel tracks and minimize the camera system offset from the seed row. The problem has been formulated as a Nonlinear Model Predictive Control (NMPC) problem with the objective of keeping the vision modules centered over the seed rows, and constraining the wheel motion to the defined wheel tracks.**

**The system and optimization problem has been implemented in Python using the Casadi framework. The implementation has been evaluated through simulations of the system, and compared with a PD controller. The NMPC approach display advantages and better performance when facing the path constraints of operating in row crops.**

## I. Introduction

An agricultural robot for weed control in row-crops is under development. The weed control is done by drop-on-demand herbicide application, where the weed is first identified by a camera system and then targeted by a drop-on-demand nozzle array. The focus of the project is on computer vision, robot integration and navigation [1].

The project share its ambition of autonomous weed control with many other research projects on robotic weed control. The review by Slaughter 2008 presents an overview of the field [2].

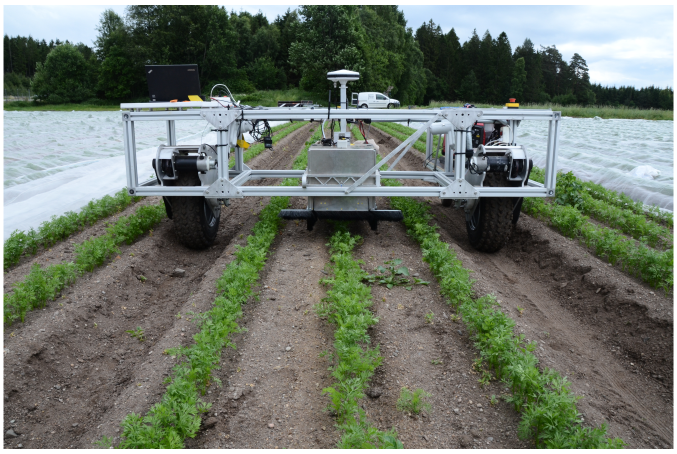
Fig. 1. The wheeled mobile robot developed for weed control in row crops.

The robotic platform has two front wheels with electro motors and two rear castor wheels, Figure 1. The robot has a monocular downward facing RGB camera primarily used for two purposes: Classification of crop and weed plants as part of the spray-on-demand system [3], and for visual odometry measurements as input to the localization filter and crop-row detection.

The visual crop-row estimate will be fused with a forward looking camera for crop-row detection. This information forms the input to the row following controller.

Crop-row following has been well explored within the field of agricultural robotics, and similar applications can be found in [4].

Application of Non-linear Model predictive control (NMPC) in agriculture has been described in [5], and [6] where an actuated trailed implement is controlled to follow field rows.

To the authors knowledge there has not been publications on NMPC applications for robotics in row crops with specific constraints on the wheels to minimize crop damage.

### A. *Minimizing crop damage*

A review of autonomous navigation in agriculture is presented in [7], where the performance of various approaches are compared.

A study of guiding principles in design of robots for agriculture revealed that the most important factor for the end users were minimizing crop damage [8].

Other controllers described in the review article [7], does not incorporate the constraints on navigation directly. This motivates our research on using an NMPC based design where the path constraints can be directly implemented in the controller, adding an additional barrier against damaging the crop.

### B. *Wheel tracks in row cultures*

The production method for most vegetables is row cultures, where the plant rows are set with a fixed intermediate distance between the wheel tracks. The centre to centre distance of the wheel tracks are typically 1.65 m to 1.80 m in European agriculture, 4.

The robots and vehicles operating in the field are restricted to these wheel tracks to avoid damage of the crop, as illustrated in Figure 2.

## II. Modelling and simulation

The robot can be modelled as a unicycle-like robot assuming non-slip conditions. Differentially steered robot designs are

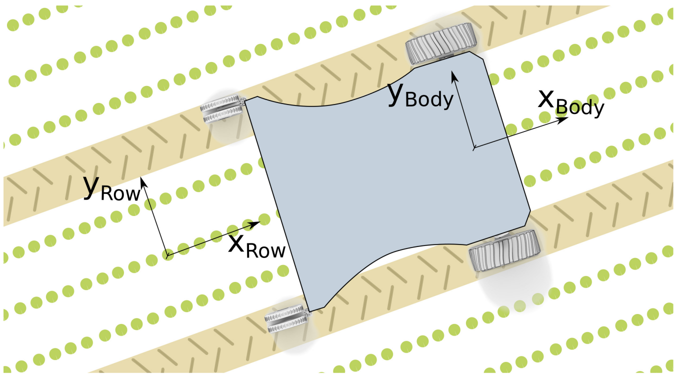

Fig. 2. The robot illustrated has a path-following algorithm seeking to maintain the camera module centered over the seed row. A sideways error will lead the controller to turn to correct the error. A small turn can easily lead the rear castor wheels to enter the sow bed and damage the crop.

very common in many applications, and numerous publications present kinematic and dynamic models for this class of robots.

Industrial motor controllers and commercial robot platforms normally provide the control inputs as linear and angular velocity set-points, not as torque or voltage set-points. A dynamic model with the motor controllers included and velocities as inputs will be advantageous when it comes to implementing the actual robot.

Such a model has been presented [9] and the formulation has been used as the basis for modelling and simulation in this paper. The robot in our project differs from the model schematic presented in [9]: In contrast to the schematic, this robot has the differential drive wheels in front, and trailing castor wheels in rear. Consistency with the schematic is maintained by describing relevant parameters with negative sign, as shown in Figure 3. The camera and spray unit has been mounted centrally at the virtual wheel axis in field experiments, which leaves the tracking point, $h$, at $a = 0$.

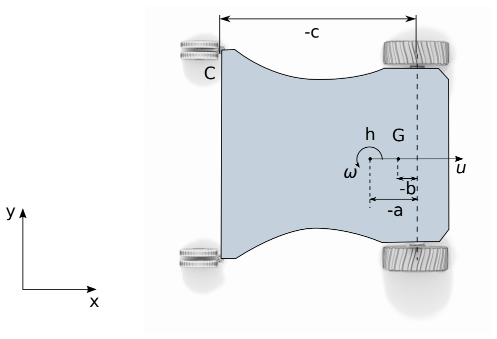

Fig. 3. The unicycle model from [9] can be applied to the robot with differential drive and passive castor wheels. The robot tracking point is located in $h$, which is the origin of the robot frame. The point $h$ is a distance $-a$ behind the virtual front wheel axis. $G$ is the center of gravity, $-b$ behind the virtual front wheel axis. The track width of the robot is $d$, and the rear castor wheels, $C$, is at a distance $-c$ from the virtual wheel axis. The robot has a forward velocity, $u$, and an angular velocity $\omega$.

The dynamic model is written as [9]:

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}) + \boldsymbol{\delta} \tag{1}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} u\cos\psi - a\omega\sin\psi \\ u\sin\psi + a\omega\cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}u\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_{ref} \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ 0 \\ \delta_u \\ \delta_\omega \end{bmatrix} \tag{2}$$

where $(x, y)$ is the position in the base frame, and $\psi$ is the robot orientation or heading, $(u, \omega)$ are the linear and angular velocities and $u_{ref}$ and $\omega_{ref}$ are the input signals of the system: linear and angular velocity.

The two vectors are identified model parameters, and parametric uncertainties. The uncertainty vector

$$\boldsymbol{\delta} = [\ \delta_x\ \delta_y\ 0\ \delta_u\ \delta_\omega\ ]^T \tag{3}$$

represent slip velocities and effects of uneven ground with its first two elements. The two last elements are functions of physical parameters as mass, inertia, wheel and tire diameters, parameters of the motors, and wheel ground interaction forces.

The parameters in the vector $\boldsymbol{\theta}$ are functions of the robots physical parameters, such as its mass $m$, inertia $I_Z$ about $G$, the electrical resistance $R_a$ of the DC motors with motor constant $k_a$, the friction coefficient $B_e$, reduction gear inertia $I_e$, radius of the wheel $r$, nominal radius of the tire $R_t$, and the distances $b$ and $d$. The model assumes a PD motor control loop with gains $k_{PT} > 0$, $k_{PR}$, $k_{DT}$ and $k_{DR}$. The equations for $\boldsymbol{\theta}$ were presented in [9], and methods for online parameter identification and an adaptive controller has been presented [10]. The parameter equations are reproduced here for reference:

$$\begin{aligned}
\theta_1 &= \left(\frac{R_a}{k_a}(mR_t r + 2I_e) + 2r_k DT\right)(2rk_{PT}) \\
\theta_2 &= \left(\frac{R_a}{k_a}(I_e d^2 + 2R_t r(I_z + mb^2)) + 2rdk_{DR}\right)/(2rdk_{PR}) \\
\theta_3 &= \frac{R_a}{k_a} mbR_t/(2k_{PT}) \\
\theta_4 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right)(rk_{PT}) + 1 \\
\theta_5 &= \frac{R_a}{k_a} mbR_t/(dk_{PR}) \\
\theta_6 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right)d/(2rk_{PR}) + 1
\end{aligned} \tag{4}$$

For the simulations in this paper, we have used a set of parameters identified from indoor experiments with a robot comparable to our:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} 0.19 \\ 0.14 \\ 0.02 \\ 1.00 \\ 0.16 \\ 1.00 \end{bmatrix} \tag{5}$$

The disturbances in the $\boldsymbol{\delta}$ have been left at zero for the simulations shown here, while it does provide a possible input for perturbing the system in simulation.

The model has been implemented in Python using the Casadi framework [11] and a Runge Kutta 4 integrator scheme for time simulation.

For this implementation, the robot is assumed to navigate relative to a local coordinate frame aligned with the crop row, as illustrated in Figure 2.

### A. *Reference PD controller*

A PD controller has been implemented for reference and comparison with the NMPC controller. The controller has been tuned to stay within the constraints when operating with a tracking error less than half the allowed region. That is $y \in [-\frac{\tau_w}{2}, \frac{\tau_w}{2}]$.

The controller has been implemented as a P controller driving both the sideways tracking error and the heading to zero. For small heading angles, the time-derivative of $y$ approximates to the heading $\psi$, and the controller can be though of as a PD controller. The velocity reference is constant.

$$v_{ref} = 0.3\mathrm{m\,s}^{-1} \tag{6}$$

$$\omega_{refPD} = -k_p y - k_d \psi \tag{7}$$

where the tuning constants has been set to:

$$k_p = 0.70 \quad k_d = 0.49 \tag{8}$$

## III. Formulating the optimal control problem

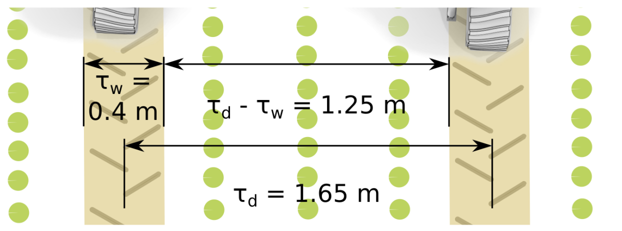

Fig. 4. The feasible wheel track area where a robot can operate without damaging the crop, based upon experience from field test during spring 2014.

The objective for the robot is to maintain a constant velocity while centring the camera systems over each crop row. At the same time, the rear castor wheels should be constrained to the wheel tracks, not to damage the crop.

The position of the rear castor wheels can be expressed in vector notation in the BODY frame, $\mathbf{w}^B$, and rotated into the ROW frame, $\mathbf{w}^R$, to find the wheels' positions. The calculation for the left rear castor wheel becomes:

$$\mathbf{w}_l^B = \begin{bmatrix} c \\ \frac{d}{2} \end{bmatrix} \tag{9}$$

$$\mathbf{w}_l^R = \mathbf{R}(\psi)\mathbf{w}_l^B + \begin{bmatrix} x \\ y \end{bmatrix}^R \tag{10}$$

$$= \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} c \\ \frac{d}{2} \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}^R \tag{11}$$

$$= \begin{bmatrix} c\cos(\psi) + \frac{d}{2}\sin(\psi) + x \\ -c\sin(\psi) + \frac{d}{2}\cos(\psi) + y \end{bmatrix} \tag{12}$$

Only the y-component is of interest in the ROW frame, and the constraint for the left castor wheel can be written with respect to the distance between two tracks $\tau_d$ and the width of each wheel track $\tau_w$, illustrated in Figure 4, as:

$$\frac{-\tau_d - \tau_w}{2} \leq -c\sin(\psi) + \frac{d}{2}\cos(\psi) + y \leq \frac{\tau_d - \tau_w}{2} \tag{13}$$

If the robot track width and the row track width are equal, $\tau_d = d$, the constraint will be symmetric for the left and right wheel, for small deviations of y, and it will be sufficient to consider one wheel constraint.

The quadratic cost function describes the robots deviation from the row centre line, and deviation from the reference velocity:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \int_{t_0}^{T} (y - y_{row})^2 + (u - u_{setpoint})^2 dt \qquad (14) \\ \text{subject to} \quad & \dot{\mathbf{x}} \;=\; f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}) \\ & \frac{-\tau_d - \tau_w}{2} \leq c\sin(\psi) + \frac{d}{2}\cos(\psi) + y \leq \frac{\tau_d - \tau_w}{2} \end{aligned}$$

### A. *Inverse proportional limit on the feasible control space*

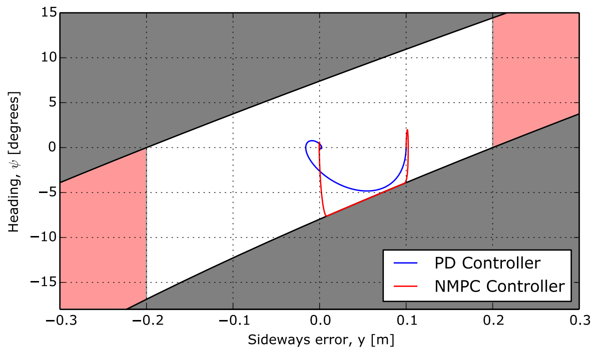

Fig. 5. The allowed range of steering angles $\psi$ as a function of the tracking error. The regions marked in red, are outside the track width. If the robot enters the red region the front wheels are outside the wheel tracks, and the robot will diverge from the row. The NMPC controller with constraints can only be used within the white region. Two trajectories starting with a sideways error of 0.1 meter are shown in the plot, one with the NMPC controller and one with the PD controller. Note that the NMPC controller closely follows the constraint in steering angle.

The kinematics of the system leaves us in a special case when we apply the constraint to the rear castor wheel. An analogy of this scenario is driving a forklift alongside a wall. As the forklift approaches the wall, it will be increasingly impossible to turn away from the wall, as its rear end needs to swing out to turn.

The robot and the wheel tracks scenario leave us with the same situation. The allowed and stable region of steering, or vehicle angles, are shown in Figure 5.

A typical path following algorithm using a PID-type controller, would proportionally increase its control input as the error increases. The feasible control region for this problem leaves us with a set of controls which does not fit well with a proportional control strategy: As the tracking error increase, say to 0.1 m, the maximum vehicle angle to counter the error is reduced by a factor of two. This limitation on the feasible state space can be taken as an argument for implementing a NMPC controller to utilize the limited control space optimally.

## IV. Implementation

The system has been implemented in Python using the Casadi framework [11]. The chosen method for solving the optimization problem (14), is a direct multiple shooting method.

The infinite horizon problem is reformulated to a finite discretized nonlinear problem. The time horizon has been limited to 5 seconds, and the control input has been discretized to N = 20 steps.

The second step is to parameterize the system of differential algebraic equations (DAE) by *multiple shooting*. We also exploit the quadratic form of the cost function, by using the Gauss-Newton method to solve the sequential quadratic program (SQP). The implementation follow the details given in [12]. A more advanced implementation applied to automobile collision avoidance, follows the same implementation approach [13].

The QP problem is solved at each iteration by using the IPOPT library [14]. The QP problem is initialized with the last state at every iteration as an aid to the QP solver.

The system has been simulated without disturbances, $\boldsymbol{\delta} = \mathbf{0}$, from various perturbed initial conditions to investigate the system behaviour. The target velocity has been set to $u_{setpoint} = 0.3\,\mathrm{m\,s^{-1}}$ and the crop row is at the origin of the coordinate system, $y_{row} = 0$

## V. Results

The NMPC controller has tested by starting the system in several different initial conditions. Figure 7 show the system recovering from a small tracking error, with the NMPC controller and the reference PD controller. Note the increasing curve of $\psi$ as the robot gains increasingly more headroom to navigate. Figure 8 show the states of the NMPC controller in the time domain. This scenario is also illustrated in the time domain in Figure 8 and with respect to the constraints in Figure 5.

The NMPC controller use two iterations to converge to the optimal trajectory, and steers the robot in the opposite direction with the first control input, as shown in Figure 6. Figure 9 and 10 show the system from an initial condition close to the boundary constraint. The recovery of the robot is significantly slower, and the amplitude of $\psi$ is limited by the path constraint.

In addition to the displayed figures, the NMPC controller has been initialized in several infeasible initial conditions. The trajectory then diverge from the desired trajectory for as long as the path constraint can be met. These scenarios break the assumption of small y deviations, which the symmetry assumption of the path constraint relies on.

## VI. Discussion

Looking at Figure 7 it is interesting to see the increasing correction in heading, as the robot approach the desired trajectory. This is the opposite of behaviour of a PID based controller, as illustrated by the reference PD controller. The NMPC controller maintains the rear castor wheels on the path constraint, until the target is reached. The PD controller is not able to converge as quickly without violating the constraints.

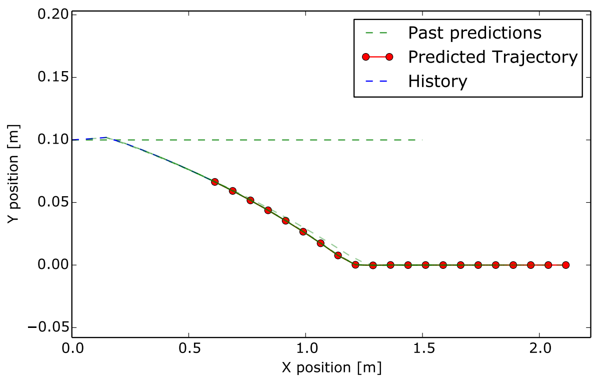

Fig. 6. With the system initialized at $[x, y] = [0\ 0.1]$ the NMPC controller use the first to iterations to converge to the optimal solution. The first prediction can be seen as the green dotted line, and the current prediction is shown as the red dots.

Figures 9 and 10 illustrate the characteristic of the constrained controller: If the tracking error is sufficiently large, the controller is left with close to no headroom for navigation, and the convergence is slow. If the front wheels are at, or outside the boundary the solution will diverge. The reference PD controller is outside its intended region of operation, and it violates the constraint in heading angle.

For operation outside the feasible region, the constraint should be reformulated without the symmetry assumption for the castor wheel constraints. This assumption relies on small deviations in angle and lateral position, and will be increasingly inaccurate outside the wheel tracks.

Another control strategy should be implemented to handle operation outside the feasible region. Some alternative solutions may be:

- Drop the velocity reference from the cost function, and add a quadratic term in the heading, $\psi$. This will allow the robot to reverse back into the wheel tracks and correct it's heading and position.
- Drop or expand the path constraint to allow the robot to quickly get back to the path, and accept damage to the crop for a short section.
- Switch to a different type of controller with desired dynamics and let that bring the robot back into the feasible region.

In a real field implementation these strategies need to be evaluated with practical considerations in mind.

The implementation of the inequality constraints are relatively straight forward, within the multiple shooting method. One can easily imagine applying such a strategy to vehicles and robots with more complex kinematics or environments, an example pointing in that direction is presented in [5].

The oscillations in $\omega_{ref}$ as the system reaches the trajectory is caused by the NMPC controller exploiting its knowledge of the motor controller and system dynamics to maximize the system response. This behaviour may be problematic when faced with inaccuracies in the estimated parameters $\boldsymbol{\theta}$. For example: If the robot is significantly lighter than estimated; this may lead to oscillations in the control. A term to dampen control inputs can be considered in the cost function.

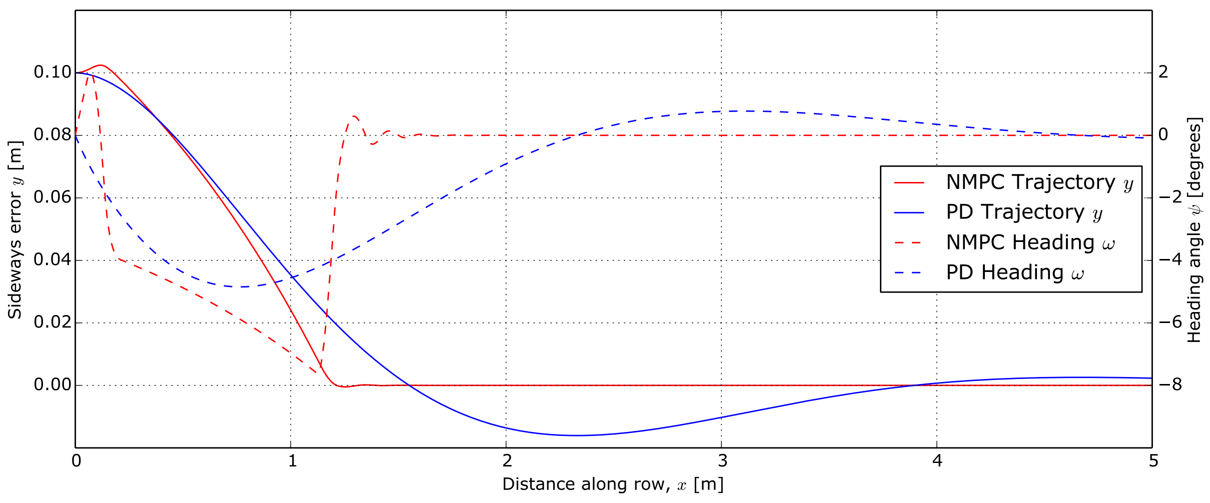

Fig. 7. A comparison of the reference PD controller with the NMPC controller, with sideways error and heading angle as the robot moves along the row. The robot was initialized at $[x, y] = [0\ 0.1]$. The NMPC controller follow the constraint in heading angle, and converge faster than the PD controller. The same trajectories are also illustrated in figure 5.

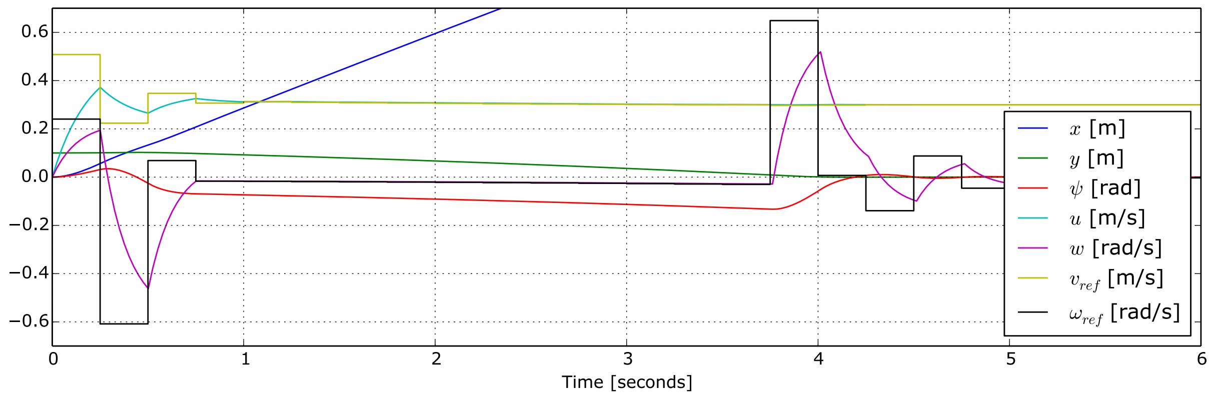

Fig. 8. The robot states and NMPC control inputs shown in the time domain, when initialized at $[x, y] = [0\ 0.1]$. After the robot has accelerated, the rear castor wheel follow the edge of the constraint until the y target is reached just before $t = 4s$. Note that the robot heading, $\psi$, increase in amplitude, as more headroom for navigation is available. When the target is reached it quickly steers the heading back to zero. The NMPC controller oscillates here to maximize the response from the motor controllers.

The computational performance of the algorithm has not been systematically evaluated, but the run-time is consistently below 50 ms per iteration. Further optimizations may be implemented for real-time applications, and there exists code generation tools within the Casadi framework, which may be useful, [12].

## VII. Conclusion

A crop row following controller has been formulated with special focus on constraining the motion of the trailing castor wheels to the wheel tracks. The implementation uses Nonlinear Model Predictive Control (NMPC) with a direct multiple shooting method, and a Gauss-Newton quadratic objective.

The implementation is flexible with regards to expressing the constraints and it can be suitable for real-time implementations. The controller needs to be expanded to operate on a global frame with an arbitrary model of the crop row, and the implementation needs to be verified in experiments.

The kinematic limitations of a trailed castor wheel with path constraints has been investigated. The limited range of feasible control inputs can be an argument for applying constrained model based control, such as this NMPC application, over other control methods. An NMPC approach will better utilize the available control room, and in row crops the NMPC controller can provide safety against damaging the crop.

## Acknowledgment

The authors would like to thank the Casadi open source community and Joel Andersson and Greg Horn for their useful input through the Casadi online user forum. The authors would also like to thank Lars Imsland for his valuable input on the NMPC implementation.

## References

[1] T. Utstumo and J. T. Gravdahl, “Implementation and comparison of attitude estimation methods for agricultural robotics,” in *Agricontrol*, vol. 4, no. 1. IFAC, 2013, pp. 52–57.

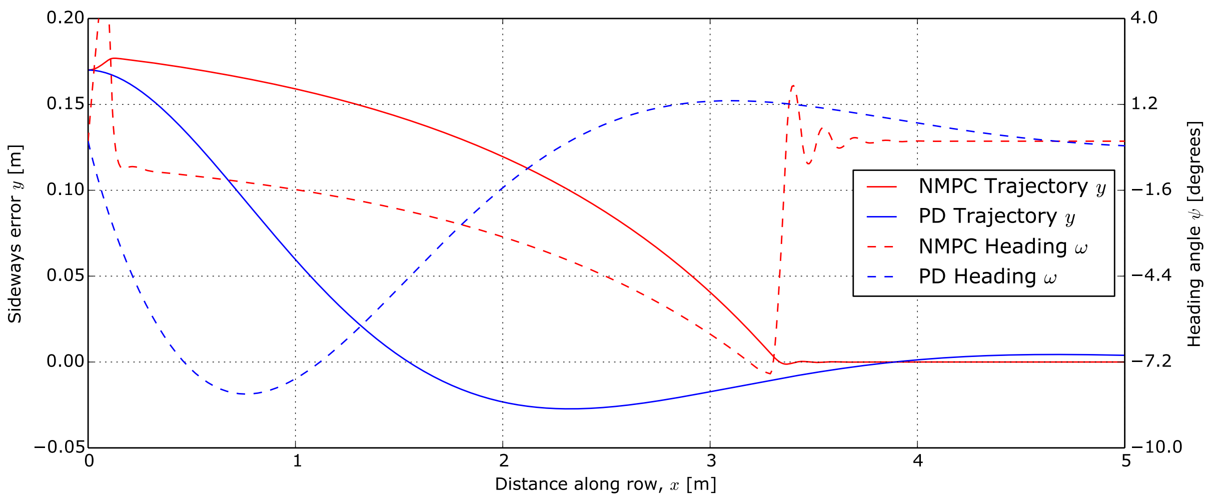

Fig. 9. The position xy plot for the robot, when initialized at $[x, y] = [0\ 0.17]$. This initial condition is close to the boundary, and the space available for navigation is very tight. The controller is slow to reach it's target. The reference PD controller is now far outside its intended region of operation and it goes far outside the constraints. In an actual application, this would result in damage to the neighbouring crop row.

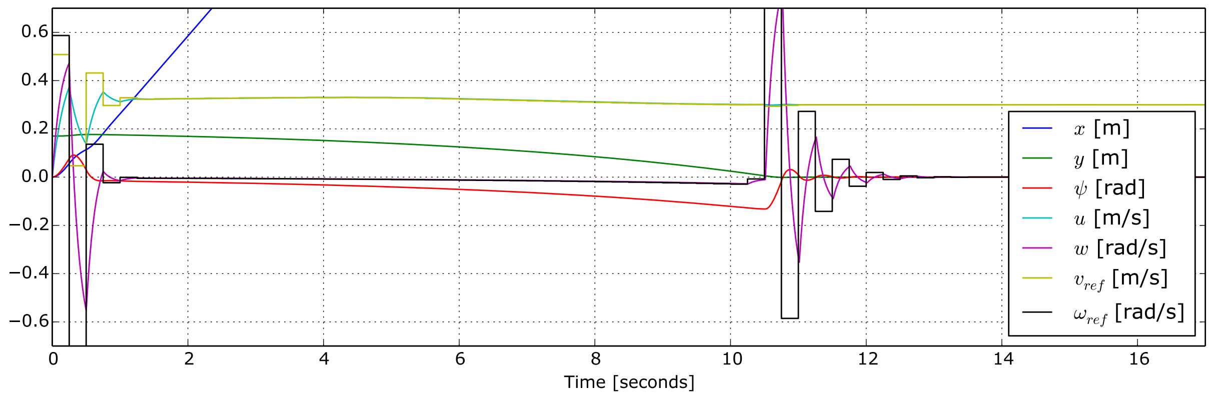

Fig. 10. The robot is initialized at $[x, y] = [0\ 0.17]$, close to the constraint. Note especially the limited amplitude of $\psi$ in the first seconds of this trajectory. As before the NMPC controller use two iterations to converge to the optimal solution, where it steers the robot in the opposite direction.

[2] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, 2008.

[3] F. Urdal, T. Utstumo, S. A. Ellingsen, J. K. Vatne, and T. Gravdahl, "Design and control of precision drop-on-demand herbicide application in agricultural robotics," in *The 13th international Conference on Control, Automation, Robotics and Vision, Singapore*. IEEE, 2014.

[4] F. Dong, W. Heinemann, and R. Kasper, "Development of a row guidance system for an autonomous robot for white asparagus harvesting," *Computers and Electronics in Agriculture*, vol. 79, no. 2, pp. 216–225, 2011.

[5] T. Kraus, H. J. Ferreau, E. Kayacan, H. Ramon, J. De Baerdemaeker, M. Diehl, and W. Saeys, "Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles," *Computers and Electronics in Agriculture*, vol. 98, pp. 25–33, 2013.

[6] J. Backman, T. Oksanen, and A. Visala, "Navigation system for agricultural machines: Nonlinear model predictive path tracking," *Computers and Electronics in Agriculture*, vol. 82, pp. 32–43, 2012.

[7] H. Mousazadeh, "A technical review on navigation systems of agricultural autonomous off-road vehicles," *Journal of Terramechanics*, vol. 50, no. 3, pp. 211–232, 2013.

[8] C. Sørensen, R. N. Jørgensen, J. Maagaard, K. K. Bertelsen, L. Dalgaard, and M. Nørremark, "Conceptual and user-centric design guidelines for a plant nursing robot," *Biosystems Engineering*, vol. 105, no. 1, pp. 119–129, 2010.

[9] D. La Cruz *et al.*, "Dynamic modeling and centralized formation control of mobile robots," in *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*, 2006, pp. 3880–3885.

[10] F. N. Martins, W. C. Celeste, R. Carelli, M. Sarcinelli-Filho, and T. F. Bastos-Filho, "An adaptive dynamic controller for autonomous mobile robot trajectory tracking," *Control Engineering Practice*, vol. 16, no. 11, pp. 1354–1363, 2008.

[11] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[12] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[13] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *Control Conference (ECC), 2013 European*. IEEE, 2013, pp. 4136–4141.

[14] 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, no. 1, pp. 25–57, 2006.