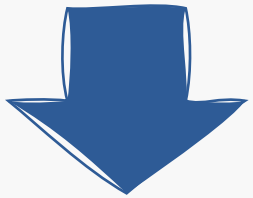
Adaptive control algorithms that provide fast and accurate vehicle control for vehicles with various characteristics

Masahiro Kubota
Team : norikenpi

Adaptive control algorithms that provide fast and accurate vehicle control for vehicles with various characteristics

- Using the given vehicle's driving data, I propose to estimate the vehicle model, which is then used to perform optimal control (model predictive control).
- The proposed method was implemented using Matlab and Simulink, resulting in more precise vehicle control compared to the default control method in Autoware.

- **High-precision vehicle control of automated vehicles is important from the perspective of safety, traffic rules, and comfort.**
  - Safety : Need to coexist with other vehicles, pedestrians and bicycles without contact.
  - Traffic rules : Need to obey stop lines, speed limits, lanes, etc.
  - Comfort : Sudden acceleration and steering operation are detrimental to comfort.



- **To make it easy for anyone to develop autonomous vehicles, algorithms are needed to achieve high-precision vehicle control for vehicles with various characteristics.**

- **MPC (Model Predictive Control) is one of the methods to achieve high-precision vehicle control.**

- **The Workflow of MPC**
  1. Predict the behavior of a system from its current state to some point in the future using a mathematical model of the object being controlled.
  2. Solve optimization problems to compute control inputs (e.g., acceleration, steering angle) to minimize a particular evaluation function based on predicted future behavior.
  3. Only the first control input of the calculated control input profile is input to the control target.
  4. Repeat 1~3.

- **To solve the optimization problem, the control target must be represented by a discrete-time state-space model such as Equation 1.**

$$\boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{w}(k) \qquad (1)$$

$\boldsymbol{x}(k)$: State vector of the system at time k
$\boldsymbol{u}(k)$: Input vector to the system at time k
$\boldsymbol{w}(k)$: Disturbance vector to the system at time k

$$\boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{w}(k) \qquad (1)$$

$\boldsymbol{x}(k)$: State vector of the system at time k
$\boldsymbol{u}(k)$: Input vector to the system at time k
$\boldsymbol{w}(k)$: Disturbance vector to the system at time k

- **Finding the optimal control input requires an accurate prediction of the future state of the controlled object.**
- **To accurately predict the future state of a control target, a highly accurate mathematical model of the control target is required.**

- **MPC is used in Autoware for lateral control of vehicles, and three different models are implemented.**
  - Bicycle kinematics model with steering 1st-order delay (default)
  - Bicycle kinematics model without steering delay
  - Bicycle dynamics model considering slip angle

- **About bicycle kinematics model without steering delay**
  - Since the simulation model used in this study has no steering input delay, the model without input delay and the control method proposed hereafter will be compared.
    The model without input delay is expressed as in Equation 2.
    (See the Autoware Universe documentation for more details.)

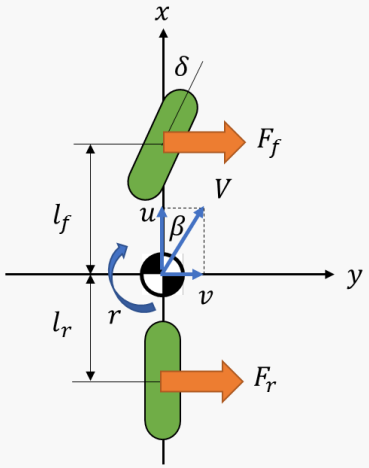


Fig. 1: Bicycle kinematics model [3]

$$\begin{bmatrix} y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & V\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{V}{L}\Delta t \end{bmatrix} \delta_k + \begin{bmatrix} 0 \\ -1 \end{bmatrix} r_{ref} \qquad (2)$$

y_k : Vehicle lateral deviation from the target position.
θ_k : Vehicle heading deviation from the target heading.
δ_k : Steering input
r_{ref} : Target yaw rate
V: Direction of travel velocity
L: Length from front wheel axle to rear wheel axle
Δt: simulation time step

- **The bicycle kinematics model is only an approximate model, and there are factors that are not considered.**
- **Elements for accurately modeling vehicles.**
  - Vehicles with three or more wheels
  - Pedal input delay, dynamics time constant, dead zone
  - Insufficient control cycles
  - Black box software intervention
  - Air resistance
- **More accurate mathematical models can be built by taking the above into account, but this takes a lot of time and money.**

- **Modeling vehicles using driving data has the potential to quickly and accurately model a wide variety of vehicles.**
- **In this study, we consider building a vehicle lateral control model from driving data.**
- **Modeling Method**
  1. The relationship between the inputs (steering angle) and the outputs to it (lateral and attitude deviation) is viewed as an ARX model, and its parameters are estimated using the least-squares method.
     - In this case, we considered the ARX model as in Equation 3 and estimated the parameters.

$$y(k) + a_1 y(k-1) = b_1 u(k-1) \quad (3)$$

  2. Convert the ARX model to a discrete-time state-space model.

$$y(k+1) = -a_1 y(k) + b_1 u(k) \quad (4)$$

- **A vehicle motion simulator that reproduces the vehicle with high accuracy is needed to verify the performance of MPC using the model identified by the system from the operation data.**
  - Although AWSIM can be easily connected to Autoware, it is not suitable for the evaluation of this proposal because it approximates the vehicle to a bicycle model and simulates its motion.
- **In this case, the Vehicle Body 3DOF Dual Track in matlab's Vehicle Dynamics Blockset was used.**
- **Vehicle Body 3DOF Dual Track considers the following factors**
  - 4-wheel vehicle dynamics
  - Air resistance
  - Lateral Corner Stiffness and Relaxation Dynamics
  - Tire Slip

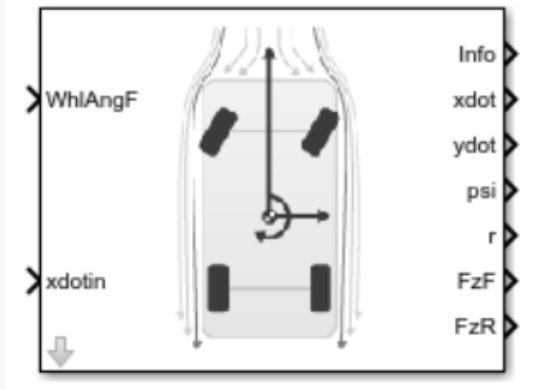


Fig. 2: Vehicle Body 3DOF Dual Track [1]

- **Driving data and reference trajectory**
  - The speed in the direction of vehicle travel is constant at 3 m/s.
  - Generates driving data set containing data for 110 seconds every 0.1 second (1100 data sets in total)
  - Representative vehicle parameters
    (For the rest, please refer to the simulink model uploaded on github.)
    - Longitudinal distance from center of gravity to front axle : 1.4m
    - Longitudinal distance from center of gravity to rear axle : 1.6m
    - Truck width : 1.4m
    - Weight : 2000kg

- **System identification**
  - In the bicycle kinematics model, a linear approximation was made by performing an appropriate coordinate transformation.
  - Therefore, this time, 10 consecutive driving data sets were extracted to create a driving data set, and a coordinate transformation was performed with the first driving data as the origin coordinate.
  - Finally, 110 driving data sets containing 10 driving data were created.
- **Evaluation of vehicle control performance**
  - Comparison of tracking errors between MPC using a bicycle kinematics model and MPC using a model identified by the system from driving data

- **The parameters of the ARX model were estimated and transformed into a discrete-time state-space model based on the driving data with trajectories as shown in Fig. 3, resulting in the following discrete-time state-space model.**

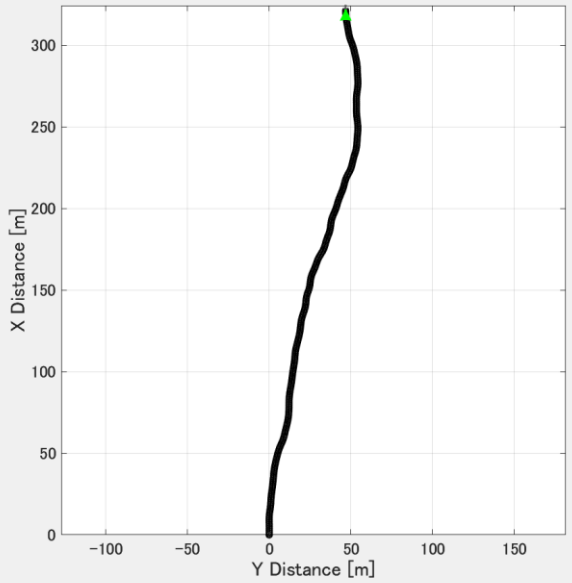


Fig. 3: Trajectory of driving data

$$\begin{bmatrix} y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0.3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \delta_k + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} r_{ref} \quad (5)$$

Discrete-time state-space model of the bicycle kinematics model.

$$\begin{bmatrix} y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1.027 & 0.8819 \\ 0.0006119 & 0.9975 \end{bmatrix} \begin{bmatrix} y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} 0.6952 \\ 0.003328 \end{bmatrix} \delta_k + \begin{bmatrix} -0.3238 \\ 0.8923 \end{bmatrix} r_{ref} \quad (6)$$

Discrete-time state-space model of the model with system identification from driving data.

- **Fig. 5 and Fig. 6 show the tracking error when MPC is performed to follow a reference trajectory as shown in Fig. 4 using a bicycle kinematics model and a model identified by the system based on driving data.**

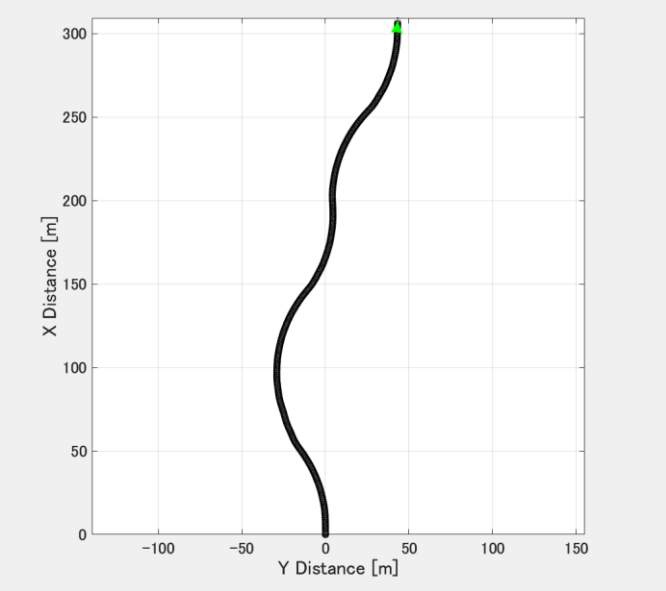


Fig. 4: Reference trajectory

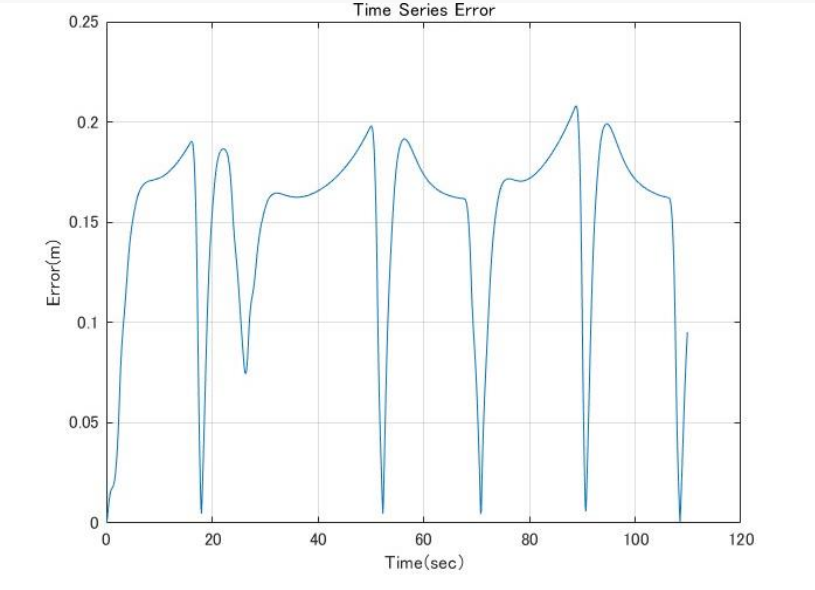


Fig. 5: MPC tracking error using a bicycle kinematics model.
RMSE=15.23cm

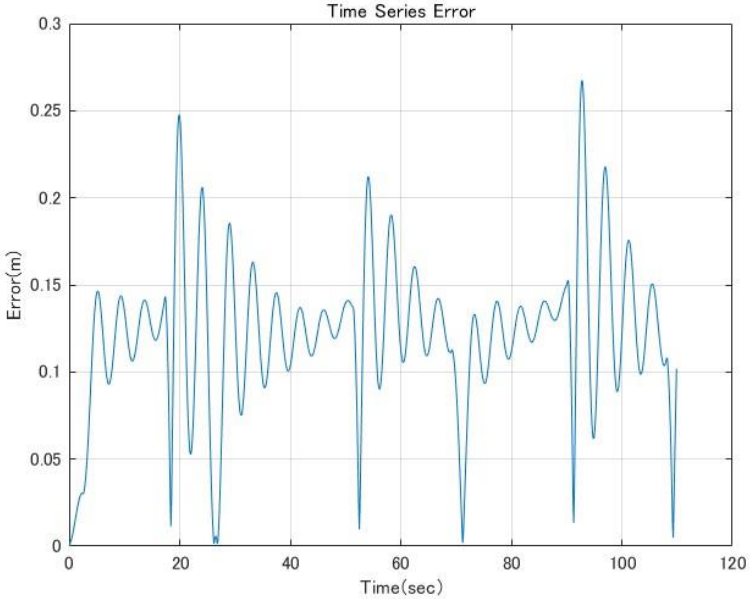


Fig. 6: MPC tracking error using a model with system identification from driving data.
RMSE=12.15cm

The vehicle control was more accurate using a model with system identification from driving data.

Possible methods for more precise vehicle control

- **Non-linear modeling of the vehicle with neural nets, etc., and sequential linearization to perform MPC.**
- **System identification is performed by increasing the order of states so that more complex states can be represented.**
- **The simulink model and matlab code constructed this time can also be used for these purposes.**

Conclusion

- MPC using a model that was system-identified from driving data allowed for more accurate vehicle control than MPC using a bicycle kinematics model.
- The simulator with the simulink model and matlab code constructed in this study can be used to evaluate other system identification and control methods.
- The simulink model and matlab code used in this study have been uploaded to the following github repository.
  https://github.com/norikenpi/adaptive_mpc

- **Matlab and simulink references**

[1] Simulation of longitudinal and lateral vehicle motion with Vehicle Dynamics Blockset
https://jp.mathworks.com/videos/simulating-longitudinal-and-lateral-vehicle-dyanamics-1664547380990.html
[2] Design adaptive MPC in a black box
https://www.youtube.com/watch?v=XP3lHpqBG9M&list=PLPQVx3HzGQWKnCOSnOxFfR5231Hesddos&index=6
[3] MPC Implementation Example
https://jp.mathworks.com/matlabcentral/fileexchange/77879-mpc-implementation-example

- **Autoware references**

[4] About MPC implemented in Autoware
https://autowarefoundation.github.io/autoware.universe/main/control/mpc_lateral_controller/model_predictive_control_algorithm/