Nonlinear Model Predictive Control for

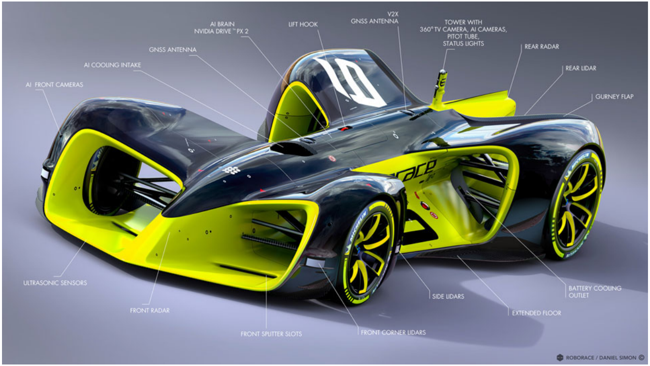
Autonomous Race Cars

Behzad Samadi

Research Group, Maplesoft, Waterloo

Autonomous Race Cars Are Here

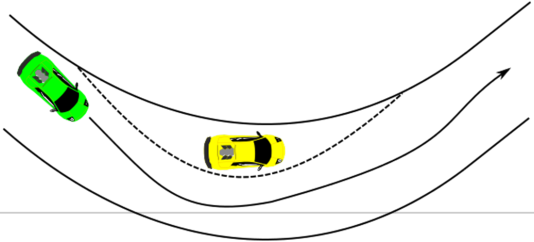
- Roborace will be a motorsport championship similar to the FIA Formula E Championship but with autonomously-driven electric race cars.



http://danielsimon.com/roborace-robocar/

Model Predictive Driver

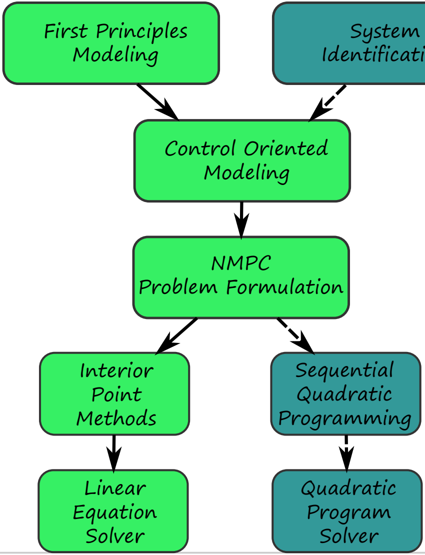
- A race driver needs to look forward! (**prediction**)
- Minimize a cost function at each time instant depending on the current situation (**closed loop optimal control**)
- Optimization constraints:
  - Vehicle's dynamic behavior
  - Limited power
  - No skidding
  - Following the road
  - Avoiding collision



Model Predictive Control

- MPC is the optimal controller in the loop:
  1. Measure/estimate the current state x_n.
  2. Solve the optimal control problem to compute u_k for $k = n, \ldots, n + N - 1$.
  3. Return u_n as the value of the control input.
  4. Update n.
  5. Goto step 1.
- MPC is implemented in real time.

NMPC: Problem and Solution



Nonlinear Model

- Consider the following nonlinear system:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ x(t_0) &= x_\circ \end{aligned}$$

where:

- $x(t)$ is the state vector
- $u(t)$ is the input vector

Optimal Control Problem

$$
\begin{aligned}
\underset{u}{\text{minimize}}\ & J(x_0, t_0) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(\tau), u(\tau))d\tau \\
\text{subject to}\ & \dot{x}(t) = f(x(t), u(t)) \\
& x(t_0) = x_\circ \\
& g_i(x(t), u(t)) = 0, \ \text{for} \ \ i = 1, \ldots, n_g \\
& h_i(x(t), u(t)) \leq 0, \ \text{for} \ \ i = 1, \ldots, n_h
\end{aligned}
$$

Discretization

- Discretize the problem into N steps from t_0 to t_f:

$$\begin{aligned}
\underset{u,\alpha}{\text{minimize}}\ & \phi_d(x_N) + \sum_{k=0}^{N-1} \left(L(x_k, u_k)\right) \\
\text{subject to}\ & x_{k+1} = f_d(x_k, u_k) \\
& x_0 = x_\circ \\
& g_i(x_k, u_k) = 0, \ \text{for} \quad i = 1, \ldots, n_g \\
& h_i(x_k, u_k) \leq 0, \ \text{for} \quad i = 1, \ldots, n_h
\end{aligned}$$

where $\Delta\tau = \frac{t_f - t_0}{N}$ and:

$$\phi_d(x_N) = \frac{\phi(x(t_f), t_f)}{\Delta\tau}$$

$$f_d(x_k, u_k) = x_k + f(x_k, u_k)\Delta\tau$$

Interior Point - Barrier Method

- Using a particular *interior-point algorithm*, the *barrier method*, the inequality constraints are converted to equality constraints:

$$
\begin{aligned}
\underset{u,\alpha}{\text{minimize}}\ & \phi_d(x_N) + \sum_{k=0}^{N-1} \left(L(x_k, u_k) - r^{\mathsf{T}} \alpha_k \right) \\
\text{subject to}\ & x_{k+1} = f_d(x_k, u_k) \\
& x_0 = x_\circ \\
& g_i(x_k, u_k) = 0, \ \text{for} \ \ i = 1, \ldots, n_g \\
& h_i(x_k, u_k) + \alpha_{ik}^2 = 0, \ \text{for} \ \ i = 1, \ldots, n_h
\end{aligned}
$$

where $\alpha_k \in \mathbb{R}^{n_h}$ is a vector slack variable and the entries of $r \in \mathbb{R}^{n_h}$ are small positive numbers.

(Boyd and Vandenberghe 2004)
(Diehl, Ferreau, and Haverbeke 2009)

Optimization Problem

$$
\begin{aligned}
\underset{u,\alpha}{\text{minimize}}\ & \phi_d(x_N) + \sum_{k=0}^{N-1} \left(L(x_k, u_k) - r^{\mathsf{T}} \alpha_k \right) \\
\text{subject to}\ & x_{k+1} = f_d(x_k, u_k) \\
& x_0 = x_\circ \\
& G(x_k, u_k, \alpha_k) = 0
\end{aligned}
$$

where:

$$
G(x_k, u_k, \alpha_k) = \begin{bmatrix} g_1(x_k, u_k) \\ \vdots \\ g_{n_g}(x_k, u_k) \\ h_1(x_k, u_k) + \alpha_{1k}^2 \\ \vdots \\ h_{n_h}(x_k, u_k) + \alpha_{n_h k}^2 \end{bmatrix}
$$

Lagrange Multipliers

- Lagrange multipliers:

$$\begin{aligned}\mathcal{L}(x,u,\alpha,\lambda,\nu) = \phi_d(x_N,N) + (x_\circ - x_0)^\mathsf{T}\lambda_0 \\ + \sum_{k=0}^{N-1}\Big(L(x_k,u_k) - r^\mathsf{T}\alpha_k \\ + (f_d(x_k,u_k) - x_{k+1})^\mathsf{T}\lambda_{k+1} \\ + G(x_k,u_k,\alpha_k)^\mathsf{T}\nu_k\Big)\end{aligned}$$

- Optimality conditions:

$$\mathcal{L}_{x_k} = 0, \mathcal{L}_{\lambda_k} = 0 \text{ for } k = 0,\ldots,N$$

$$\mathcal{L}_{\alpha_k} = 0, \mathcal{L}_{u_k} = 0, \mathcal{L}_{\nu_k} = 0 \text{ for } k = 0,\ldots,N-1$$

Hamiltonian

- $\mathcal{L}(x, u, \alpha, \lambda, \nu)$ can be rewritten as:

$$\begin{aligned}\mathcal{L}(x, u, \alpha, \lambda, \nu) =& \phi_d(x_N) + x_\circ^T \lambda_0 - x_N^T \lambda_N \\ &+ \sum_{k=0}^{N-1} \left(\mathcal{H}(x_k, u_k, \alpha_k, \lambda_{k+1}) - x_k^T \lambda_k \right)\end{aligned}$$

- Hamiltonian:

$$\begin{aligned}\mathcal{H}(x_k, u_k, \alpha_k, \lambda_{k+1}, \nu_k) =& L(x_k, u_k) - r^T \alpha_k \\ &+ f_d(x_k, u_k)^T \lambda_{k+1} + G(x_k, u_k, \alpha_k)^T \nu_k\end{aligned}$$

Pontryagin's Maximum Principle

	Optimality Conditions
$\mathcal{L}_{\lambda_{k+1}} = 0$	$x^\star_{k+1} = f_d(x^\star_k, u^\star_k)$
$\mathcal{L}_{\lambda_0} = 0$	$x^\star_0 = x_\circ$
$\mathcal{L}_{x_k} = 0$	$\lambda^\star_k = \mathcal{H}_x(x^\star_k, u^\star_k, \alpha^\star_k, \lambda^\star_{k+1}, \nu^\star_k)$
$\mathcal{L}_{x_N} = 0$	$\lambda^\star_N = \frac{\partial}{\partial x_N}\phi_d(x^\star_N)$
$\mathcal{L}_{u_k} = 0$	$\mathcal{H}_u(x^\star_k, u^\star_k, \alpha^\star_k, \lambda^\star_{k+1}, \nu^\star_k) = 0$
$\mathcal{L}_{\alpha_k} = 0$	$\mathcal{H}_\alpha(x^\star_k, u^\star_k, \alpha^\star_k, \lambda^\star_{k+1}, \nu^\star_k) = 0$
$\mathcal{L}_{\nu_k} = 0$	$G(x^\star_k, u^\star_k, \alpha^\star_k) = 0$

for $k = 0, \ldots, N-1$ where $\star$ denote the optimal solution

cGMRES Method: Compute Optimality Conditions

- Step 1: Compute x_k and λ_k as functions of u_k, α_k and ν_k, given the following equations:

$$
\begin{aligned}
x_{k+1} &= f_d(x_k, u_k) \\
x_0 &= x_n \\
\lambda_k &= \mathcal{H}_x(x_k, u_k, \alpha_k, \lambda_{k+1}, \nu_k) \\
\lambda_N &= \frac{\partial}{\partial x_N}\phi_d(x_N)
\end{aligned}
$$

(Ohtsuka 2004)

cGMRES Method: Compute Optimality Conditions

- Step 2: For

$$U = [u_0^\mathsf{T}, \ldots, u_{N-1}^\mathsf{T}, \alpha_0^\mathsf{T}, \ldots, \alpha_{N-1}^\mathsf{T}, \nu_0^\mathsf{T}, \ldots, \nu_{N-1}^\mathsf{T}]^\mathsf{T}$$

solve the equation $F(x_n, U) = 0$, where:

$$F(x_n, U) = \begin{bmatrix} \mathcal{H}_u(x_0, u_0, \alpha_0, \lambda_1, \nu_0) \\ \mathcal{H}_\alpha(x_0, u_0, \alpha_0, \lambda_1, \nu_0) \\ G(x_0, u_0, \alpha_0) \\ \vdots \\ \mathcal{H}_u(x_{N-1}, u_{N-1}, \alpha_{N-1}, \lambda_N, \nu_{N-1}) \\ \mathcal{H}_\alpha(x_{N-1}, u_{N-1}, \alpha_{N-1}, \lambda_N, \nu_{N-1}) \\ G(x_{N-1}, u_{N-1}, \alpha_{N-1}) \end{bmatrix}$$

(Ohtsuka 2004)

cGMRES Method: Solver

- *Continuation method*: Instead of solving $F(x, U) = 0$, find U such that:

$$\dot{F}(x, U) = A_s F(x, U)$$

  where A_s is a matrix with negative eigenvalues.

- Now, we have:

$$F_x \dot{x} + F_U \dot{U} = A_s F(x, U)$$

- *GMRES*: To compute $\dot{U}$ using the following equation, which is linear in $\dot{U}$, we use the generalized minimum residual (GMRES) algorithm.

$$F_U \dot{U} = A_s F(x, U) - F_x f(x, u)$$

- To compute U at any given time, we need to have an initial value for U and then use the above $\dot{U}$ to update it.

(Ohtsuka 2004)

cGMRES Method: Solver

- *Numerical approximation*:

$$F_U \dot{U} \simeq \frac{F(x + hf(x,u), U + h\dot{U}) - F(x + hf(x,u), U)}{h}$$

$$F_x f(x,u) \simeq \frac{F(x + hf(x,u), U) - F(x,U)}{h}$$

(Ohtsuka 2004)

Maple Implementation

- *Problem formulation*:

$$A(x, U, \dot{U}) = \frac{F(x, U + h\dot{U}) - F(x, U)}{h}$$

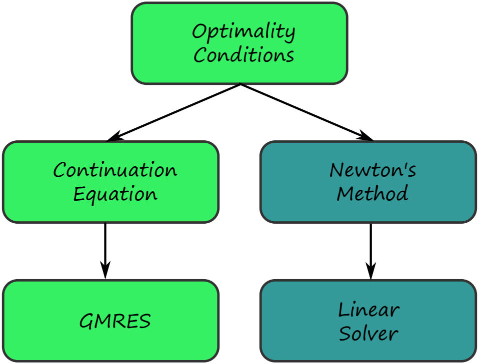
$$b(x, U) = A_s F(x, U) - \frac{F(x + hf(x, u), U) - F(x, U)}{h}$$

- $b(x, U)$ is only called once per each step at the beginning and $A(x, U, \dot{U})$ is called several times by the GMRES solver.

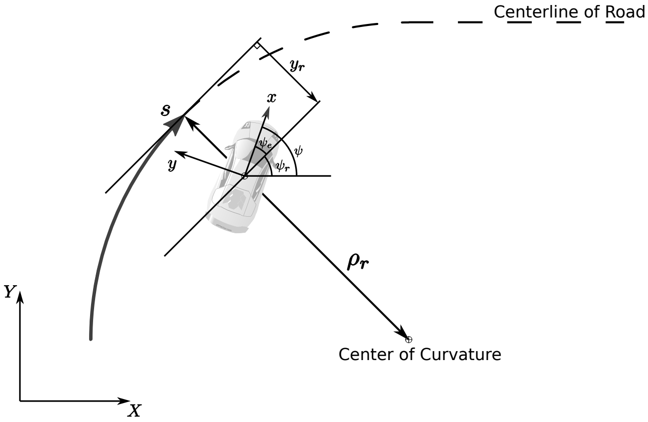
Maple Implementation

- Maple procedures are generated for A, b and optimized.
- The GMRES solver is also implemented in Maple.
- C code is then generated automatically.
- The C code is then used to simulate the closed loop system in Maplesim.

Interior Point Methods



Application: Autonomous Race Car



$$\dot{s} = \frac{\rho_r}{\rho_r - y_r}\left(v_x \cos(\psi_e) - v_y \sin(\psi_e)\right)$$

Application: Autonomous Race Car

Approximate Vehicle Model

$$\begin{aligned}
\dot{X} &= v\cos(\psi + C_1\delta)\\
\dot{Y} &= v\sin(\psi + C_1\delta)\\
\dot{\psi} &= vC_2\delta\\
\dot{v} &= (C_{m_1} - C_{m_2}v)F_{x_r} - C_{r_2}v^2 - C_{r_0} - (v\delta)^2 C_2 C_1
\end{aligned}$$

(Verschueren et al. 2014)

Application: Autonomous Race Car

Key Equation

$$\frac{dz}{ds} = \frac{\dot{z}}{\dot{s}}$$

Velocity on the Centerline

$$\dot{s} = \frac{1}{1 - \kappa_r y_r}\left(v_x \cos(\psi_e) - v_y \sin(\psi_e)\right)$$

where:

$$\kappa_r = \frac{1}{\rho}$$

Application: Autonomous Race Car

Spatial model

$$
\begin{aligned}
\frac{dy_r}{ds} &= \frac{1}{\dot{s}}\left(v\sin(\psi) + vC_1\delta\cos(\psi)\right)\\
\frac{d\psi_e}{ds} &= \frac{\dot{\psi}}{\dot{s}} - \kappa_r\\
\frac{dv}{ds} &= \frac{\dot{v}}{\dot{s}}\\
\frac{dt}{ds} &= \frac{1}{\dot{s}}
\end{aligned}
$$

Application: Autonomous Race Car

Spatial model

- State variables:

$$x = [y_r, \psi_r, v, t]$$

- Control inputs:

$$u_c = [\delta, F_{x_r}]$$

- External input:

$$u_e = [\kappa_r]$$

Application: Autonomous Race Car

Optimal Control Problem

$$\underset{u_c}{\text{minimize}}\ J(x_0, s_0) = \phi(x(s_f)) + \int_{s_0}^{s_f} L(x(\sigma), u_c(\sigma))d\sigma$$

Time Optimal Problem

$$\begin{aligned} \phi(x(s_f)) &= t(s_f) \\ L(x(\sigma), u_c(\sigma)) &= 0 \end{aligned}$$

Application: Autonomous Race Car

Issues

- Trade-off: time-optimality vs tracking and collision avoidance
- Robustness

References

Boyd, S.P., and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge Univ Pr.

Diehl, Moritz, Hans Joachim Ferreau, and Niels Haverbeke. 2009. “Efficient Numerical Methods for Nonlinear Mpc and Moving Horizon Estimation.” In *Nonlinear Model Predictive Control*, 391–417. Springer.

Ohtsuka, Toshiyuki. 2004. “A Continuation/Gmres Method for Fast Computation of Nonlinear Receding Horizon Control.” *Automatica* 40 (4). Elsevier: 563–74.

Verschueren, Robin, Stijn De Bruyne, Mario Zanon, Janick V Frasch, and Moritz Diehl. 2014. “Towards Time-Optimal Race Car Driving Using Nonlinear Mpc in Real-Time.” In *Decision and Control (Cdc), 2014 Ieee 53rd Annual Conference on*, 2505–10. IEEE.

Maplesoft™

Mathematics • Modeling • Simulation

A Cybernet Group Company