【运动控制】Apollo6.0的mpc_controller解析

隐匿潮水之中
其修远路漫，而上下求索。

mpc_controller解析

目录

对Apollo 6.0的MPC模块进行解析。

【运动控制】Apollo6.0的mpc_controller解析_yanghq13的博客-CSDN博客
blog.csdn.net/weixin_44041199/article/detai

1 Init

1.1 输入

输入为控制配置表 `control_conf` ,判断是否加载成功。

```
if (!LoadControlConf(control_conf)) {
    AERROR << "failed to load control conf";
    return Status(ErrorCode::CONTROL_COMPUTE_ERROR,
                  "failed to load control_conf");
  }
```

1.2 动力学模型初始化

矩阵初始化依据车辆动力学模型，参考 `《Vehicle Dynamics and Control》` 的 `2.5 Dynamic Model in Terms of Error with Respect to Road` (P37)。

$$
\frac{d}{dt}\begin{bmatrix} lateral\_error \\ lateral\_error\_rate \\ heading\_error \\ heading\_error\_rate \\ station\_error \\ speed\_error \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{C_f+C_r}{mV_x} & \frac{C_f+C_r}{m} & \frac{C_r l_r - C_f l_f}{mV_x} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{C_r l_r - C_f l_f}{I_z V_x} & \frac{C_f l_f - C_r l_r}{I_z} & -\frac{C_r l_r^2 + C_f l_f^2}{I_z V_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} lateral\_error \\ lateral\_error\_rate \\ heading\_error \\ heading\_error\_rate \\ station\_error \\ speed\_error \end{bmatrix}
$$

$$
+ \begin{bmatrix} 0 & 0 \\ \frac{C_f}{m} & 0 \\ 0 & 0 \\ \frac{C_f l_f}{I_z} & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \delta_f \\ \Delta a_x \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{C_r l_r - C_f l_f}{mV_x} - V_x \\ 0 \\ -\frac{C_r l_r^2 + C_f l_f^2}{I_z V_x} \\ 0 \\ 0 \end{bmatrix} \dot{\Psi}_{des}
$$

$$\frac{d}{dt}x = Ax + Bu + C\dot{\Psi}_{des}$$

```
// Matrix init operations.
  matrix_a_ = Matrix::Zero(basic_state_size_, basic_state_size_);
  matrix_ad_ = Matrix::Zero(basic_state_size_, basic_state_size_);
  matrix_a_(0, 1) = 1.0;
```

```
  matrix_a_(1, 2) = (cf_ + cr_) / mass_;
  matrix_a_(2, 3) = 1.0;
  matrix_a_(3, 2) = (lf_ * cf_ - lr_ * cr_) / iz_;
  matrix_a_(4, 5) = 1.0;
  matrix_a_(5, 5) = 0.0;
  // TODO(QiL): expand the model to accommodate more combined states.

  matrix_a_coeff_ = Matrix::Zero(basic_state_size_, basic_state_size_);
  matrix_a_coeff_(1, 1) = -(cf_ + cr_) / mass_;
  matrix_a_coeff_(1, 3) = (lr_ * cr_ - lf_ * cf_) / mass_;
  matrix_a_coeff_(2, 3) = 1.0;
  matrix_a_coeff_(3, 1) = (lr_ * cr_ - lf_ * cf_) / iz_;
  matrix_a_coeff_(3, 3) = -1.0 * (lf_ * lf_ * cf_ + lr_ * lr_ * cr_) / iz_;

  matrix_b_ = Matrix::Zero(basic_state_size_, controls_);
  matrix_bd_ = Matrix::Zero(basic_state_size_, controls_);
  matrix_b_(1, 0) = cf_ / mass_;
  matrix_b_(3, 0) = lf_ * cf_ / iz_;
  matrix_b_(4, 1) = 0.0;
  matrix_b_(5, 1) = -1.0;
  matrix_bd_ = matrix_b_ * ts_;

  matrix_c_ = Matrix::Zero(basic_state_size_, 1);
  // 20210915, yanghq
  // matrix_c_(5, 0) = 1.0;
  matrix_cd_ = Matrix::Zero(basic_state_size_, 1);
```

1.3 前馈矩阵初始化

参考以下公式

$$u_{feedforward} = -kx$$

```
matrix_state_ = Matrix::Zero(basic_state_size_, 1);
  matrix_k_ = Matrix::Zero(1, basic_state_size_);
```

1.3 QP问题Q和R初始化

参考以下公式

$$\min_{x_k,u_k} J = \min_{x_k,u_k} \left[\sum_0^N (x_k - x_r)^T Q (x_k - x_r) + \sum_0^{N-1} u_k^T R u_k \right]$$

```
matrix_r_ = Matrix::Identity(controls_, controls_);

  matrix_q_ = Matrix::Zero(basic_state_size_, basic_state_size_);

  int r_param_size = control_conf->mpc_controller_conf().matrix_r_size();
  for (int i = 0; i < r_param_size; ++i) {
    matrix_r_(i, i) = control_conf->mpc_controller_conf().matrix_r(i);
  }

  int q_param_size = control_conf->mpc_controller_conf().matrix_q_size();
  if (basic_state_size_ != q_param_size) {
    const auto error_msg =
        absl::StrCat("MPC controller error: matrix_q size: ", q_param_size,
                     " in parameter file not equal to basic_state_size_: ",
```

```
                      basic_state_size_);
    AERROR << error_msg;
    return Status(ErrorCode::CONTROL_COMPUTE_ERROR, error_msg);
  }
  for (int i = 0; i < q_param_size; ++i) {
    matrix_q_(i, i) = control_conf->mpc_controller_conf().matrix_q(i);
  }

  // Update matrix_q_updated_ and matrix_r_updated_
  matrix_r_updated_ = matrix_r_;
  matrix_q_updated_ = matrix_q_;
```

1.4 滤波器初始化

```
InitializeFilters(control_conf);
```

低通滤波器，采用的是巴特沃斯双极点低通滤波器，具体可参阅 `LpfCoefficients解析` 。

```
void MPCController::InitializeFilters(const ControlConf *control_conf) {
  // Low pass filter
  std::vector<double> den(3, 0.0);
  std::vector<double> num(3, 0.0);
  common::LpfCoefficients(
      ts_, control_conf->mpc_controller_conf().cutoff_freq(), &den, &num);
  digital_filter_.set_coefficients(den, num);
  lateral_error_filter_ = common::MeanFilter(static_cast<std::uint_fast8_t>
      control_conf->mpc_controller_conf().mean_filter_window_size()));
  heading_error_filter_ = common::MeanFilter(static_cast<std::uint_fast8_t>
      control_conf->mpc_controller_conf().mean_filter_window_size()));
}
```

1.5 加载MPC增益调度器

```
LoadMPCGainScheduler(control_conf->mpc_controller_conf());
```

加载增益调度表，构造一维线性差值器 `Interpolation1D` 。

```
void MPCController::LoadMPCGainScheduler(
    const MPCControllerConf &mpc_controller_conf) {
  const auto &lat_err_gain_scheduler =
      mpc_controller_conf.lat_err_gain_scheduler();
  const auto &heading_err_gain_scheduler =
      mpc_controller_conf.heading_err_gain_scheduler();
  const auto &feedforwardterm_gain_scheduler =
      mpc_controller_conf.feedforwardterm_gain_scheduler();
  const auto &steer_weight_gain_scheduler =
      mpc_controller_conf.steer_weight_gain_scheduler();
  ADEBUG << "MPC control gain scheduler loaded";
  Interpolation1D::DataType xy1, xy2, xy3, xy4;
  for (const auto &scheduler : lat_err_gain_scheduler.scheduler()) {
    xy1.push_back(std::make_pair(scheduler.speed(), scheduler.ratio()));
  }
  for (const auto &scheduler : heading_err_gain_scheduler.scheduler()) {
    xy2.push_back(std::make_pair(scheduler.speed(), scheduler.ratio()));
  }
  for (const auto &scheduler : feedforwardterm_gain_scheduler.scheduler())
```

```
    xy3.push_back(std::make_pair(scheduler.speed(), scheduler.ratio()));
  }
  for (const auto &scheduler : steer_weight_gain_scheduler.scheduler()) {
    xy4.push_back(std::make_pair(scheduler.speed(), scheduler.ratio()));
  }

  lat_err_interpolation_.reset(new Interpolation1D);
  CHECK(lat_err_interpolation_->Init(xy1))
      << "Fail to load lateral error gain scheduler for MPC controller";

  heading_err_interpolation_.reset(new Interpolation1D);
  CHECK(heading_err_interpolation_->Init(xy2))
      << "Fail to load heading error gain scheduler for MPC controller";

  feedforwardterm_interpolation_.reset(new Interpolation1D);
  CHECK(feedforwardterm_interpolation_->Init(xy2))
      << "Fail to load feed forward term gain scheduler for MPC controller"

  steer_weight_interpolation_.reset(new Interpolation1D);
  CHECK(steer_weight_interpolation_->Init(xy2))
      << "Fail to load steer weight gain scheduler for MPC controller";
}
```

1.6 Log初始化参数

```
LogInitParameters();
```

在log里打印质量、绕z轴转动惯量、质心与前轴距离、质心与后轴距离。

```
void MPCController::LogInitParameters() {
  ADEBUG << name_ << " begin.";
  ADEBUG << "[MPCController parameters]"
         << " mass_: " << mass_ << ","
         << " iz_: " << iz_ << ","
         << " lf_: " << lf_ << ","
         << " lr_: " << lr_;
}
```

1.7 返回

```
ADEBUG << "[MPCController] init done!";
  return Status::OK();
```

2 ComputeControlCommand

2.1 输入

定位、底盘、规划发送轨迹、控制命令

```
const localization::LocalizationEstimate *localization,
const canbus::Chassis *chassis,
 const planning::ADCTrajectory *planning_published_trajectory,
ControlCommand *cmd
```

2.2 过程

2.2.1 拷贝轨迹和创建debug

拷贝轨迹

```
trajectory_analyzer_ =
      std::move(TrajectoryAnalyzer(planning_published_trajectory));
```

创建debug

```
SimpleMPCDebug *debug = cmd->mutable_debug()->mutable_simple_mpc_debug();
debug->Clear();
```

2.2.2 计算纵向误差

```
ComputeLongitudinalErrors(&trajectory_analyzer_, debug);
```

参数初始化

```
double s_matched = 0.0;
  double s_dot_matched = 0.0;
  double d_matched = 0.0;
  double d_dot_matched = 0.0;
```

查询位置最近点 `matched_point`

```
const auto matched_point = trajectory_analyzer->QueryMatchedPathPoint(
      VehicleStateProvider::Instance()->x(),
      VehicleStateProvider::Instance()->y());
```

在轨迹坐标系下，计算 `s_matched` , `s_dot_matched` , `d_matched` , `d_dot_matched`

```
trajectory_analyzer->ToTrajectoryFrame(
      VehicleStateProvider::Instance()->x(),
      VehicleStateProvider::Instance()->y(),
      VehicleStateProvider::Instance()->heading(),
      VehicleStateProvider::Instance()->linear_velocity(), matched_point,
      &s_matched, &s_dot_matched, &d_matched, &d_dot_matched);
```

查询时间最近点 `reference_point`

```
const double current_control_time = Clock::NowInSeconds();

  TrajectoryPoint reference_point =
      trajectory_analyzer->QueryNearestPointByAbsoluteTimInterpolation1De(
          current_control_time);
```

计算速度 `linear_v` 、加速度 `linear_a` 、航向角误差 `heading_error` 、纵向速度 `lon_speed` 、纵向加速度 `lon_acceleration` 、横向误差系数 `one_minus_kappa_lat_error`

```
const double linear_v = VehicleStateProvider::Instance()->linear_velocity()
```

```
  const double linear_a =
      VehicleStateProvider::Instance()->linear_acceleration();
  double heading_error = common::math::NormalizeAngle(
      VehicleStateProvider::Instance()->heading() - matched_point.theta());
  double lon_speed = linear_v * std::cos(heading_error);
  double lon_acceleration = linear_a * std::cos(heading_error);
  double one_minus_kappa_lat_error = 1 - reference_point.path_point().kappa
                                         linear_v * std::sin(heading_er
```

debug更新位置参考 `station_reference` 、位置反馈 `station_feedback` 、位置误差 `station_error` 、速度参考 `speed_reference` 、速度反馈 `speed_feedback` 、速度误差 `speed_error` 、加速度参考 `acceleration_reference` 、加速度反馈 `acceleration_feedback` 、加速度误差 `acceleration_error`

```
debug->set_station_reference(reference_point.path_point().s());
  debug->set_station_feedback(s_matched);
  debug->set_station_error(reference_point.path_point().s() - s_matched);
  debug->set_speed_reference(reference_point.v());
  debug->set_speed_feedback(lon_speed);
  debug->set_speed_error(reference_point.v() - s_dot_matched);
  debug->set_acceleration_reference(reference_point.a());
  debug->set_acceleration_feedback(lon_acceleration);
  debug->set_acceleration_error(reference_point.a() -
                                lon_acceleration / one_minus_kappa_lat_erro
```

debug更新加加速度参考 `jerk_reference` 、纵向加加速度反馈 `lon_jerk` 、加加速度误差 `jerk_error`

```
double jerk_reference =
      (debug->acceleration_reference() - previous_acceleration_reference_) /
      ts_;
  double lon_jerk =
      (debug->acceleration_feedback() - previous_acceleration_) / ts_;
  debug->set_jerk_reference(jerk_reference);
  debug->set_jerk_feedback(lon_jerk);
  debug->set_jerk_error(jerk_reference - lon_jerk / one_minus_kappa_lat_erro
```

上一时刻加速度参考 `previous_acceleration_reference` 和加速度反馈 `previous_acceleration_`

```
previous_acceleration_reference_ = debug->acceleration_reference();
  previous_acceleration_ = debug->acceleration_feedback();
```

2.2.3 更新状态量、矩阵和前馈

```
// Update state
UpdateState(debug);
UpdateMatrix(debug);
FeedforwardUpdate(debug);
```

`UpdateState` 函数，更新横向误差 `lateral_error` 、横向误差变化率 `lateral_error_rate` 、航向角误差 `heading_error` 、航向角误差变化率 `heading_error_rate` 、位置误差 `station_error` 。对应的向量如下

$$\begin{bmatrix} lateral\_error \\ lateral\_error\_rate \\ heading\_error \\ heading\_error\_rate \\ station\_error \\ speed\_error \end{bmatrix}$$

```
// State matrix update;
  matrix_state_(0, 0) = debug->lateral_error();
  matrix_state_(1, 0) = debug->lateral_error_rate();
  matrix_state_(2, 0) = debug->heading_error();
  matrix_state_(3, 0) = debug->heading_error_rate();
  matrix_state_(4, 0) = debug->station_error();
  matrix_state_(5, 0) = debug->speed_error();
```

`UpdateMatrix` 函数，更新 `matrix_a_` 、 `matrix_ad_` 、 `matrix_c_` 、 `matrix_cd_`

```
matrix_a_(1, 1) = matrix_a_(1, 1) / v;
  matrix_a_(1, 3) = matrix_a_coeff_(1, 3) / v;
  matrix_a_(3, 1) = matrix_a_coeff_(3, 1) / v;
  matrix_a_(3, 3) = matrix_a_coeff_(3, 3) / v;

  Matrix matrix_i = Matrix::Identity(matrix_a_.cols(), matrix_a_.cols());
  matrix_ad_ = (matrix_i - ts_ * 0.5 * matrix_a_).inverse() *
               (matrix_i + ts_ * 0.5 * matrix_a_);

  matrix_c_(1, 0) = (lr_ * cr_ - lf_ * cf_) / mass_ / v - v;
  matrix_c_(3, 0) = -(lf_ * lf_ * cf_ + lr_ * lr_ * cr_) / iz_ / v;
  matrix_cd_ = matrix_c_ * debug->heading_error_rate() * ts_;
```

`FeedforwardUpdate` 函数，计算 `kv` 和 `steer_angle_feedforwardterm_` ，参考公式 (`Vehicl Dynamics and Control, P57`)如下

$$K_v = \frac{l_r m}{2C_{\alpha f}(l_f + l_r)} - \frac{l_f m}{2C_{\alpha r}(l_f + l_r)}$$
$$\delta_{ff} = \frac{L}{R} + K_v a_y - k_3 \left[\frac{l_r}{R} - \frac{l_f}{2C_{\alpha r}} \frac{m V_x{}^2}{RL} \right]$$

计算转角前馈的公式有些不同，如下

$$\kappa = R^{-1}$$
$$a_y = \frac{v^2}{R} = v^2 \kappa$$
$$\delta_{ff\_1} = L\kappa + K_v v^2 \kappa$$

```
const double v = VehicleStateProvider::Instance()->linear_velocity();
  const double kv =
      lr_ * mass_ / 2 / cf_ / wheelbase_ - lf_ * mass_ / 2 / cr_ / wheelbas
  steer_angle_feedforwardterm_ = Wheel2SteerPct(
      wheelbase_ * debug->curvature() + kv * v * v * debug->curvature());
```

> 问题： K_v 的计算公式里多除了2，因为Cf和Cr赋值时已经是2倍了。

2.2.4 高速转向增益

`gain_scheduler` 参数调整q矩阵的(0, 0)和(2,2)（横向偏差和航向角偏差），前馈增益，r矩阵的（0，0）(输出转角)。

```
// Add gain scheduler for higher speed steering
  if (FLAGS_enable_gain_scheduler) {
    matrix_q_updated_(0, 0) =
        matrix_q_(0, 0) *
        lat_err_interpolation_->Interpolate(
            VehicleStateProvider::Instance()->linear_velocity());
    matrix_q_updated_(2, 2) =
        matrix_q_(2, 2) *
        heading_err_interpolation_->Interpolate(
            VehicleStateProvider::Instance()->linear_velocity());
    steer_angle_feedforwardterm_updated_ =
        steer_angle_feedforwardterm_ *
        feedforwardterm_interpolation_->Interpolate(
            VehicleStateProvider::Instance()->linear_velocity());
    matrix_r_updated_(0, 0) =
        matrix_r_(0, 0) *
        steer_weight_interpolation_->Interpolate(
            VehicleStateProvider::Instance()->linear_velocity());
  } else {
    matrix_q_updated_ = matrix_q_;
    matrix_r_updated_ = matrix_r_;
    steer_angle_feedforwardterm_updated_ = steer_angle_feedforwardterm_;
  }
```

2.2.5 debug更新q和r

```
debug->add_matrix_q_updated(matrix_q_updated_(0, 0));
  debug->add_matrix_q_updated(matrix_q_updated_(1, 1));
  debug->add_matrix_q_updated(matrix_q_updated_(2, 2));
  debug->add_matrix_q_updated(matrix_q_updated_(3, 3));

  debug->add_matrix_r_updated(matrix_r_updated_(0, 0));
  debug->add_matrix_r_updated(matrix_r_updated_(1, 1));
```

2.2.6 矩阵和参数初始化

`controls_` 为控制时域长度（代码里为2）， `horizon_` 为预测时域长度（代码里为10）

`control_gain` 和 `addition_gain` 为控制增益矩阵，对应于无约束QP问题，无约束QP问题相当于

```
Matrix control_matrix = Matrix::Zero(controls_, 1);
  std::vector<Matrix> control(horizon_, control_matrix);

  Matrix control_gain_matrix = Matrix::Zero(controls_, basic_state_size_);
  std::vector<Matrix> control_gain(horizon_, control_gain_matrix);

  Matrix addition_gain_matrix = Matrix::Zero(controls_, 1);
  std::vector<Matrix> addition_gain(horizon_, addition_gain_matrix);

  Matrix reference_state = Matrix::Zero(basic_state_size_, 1);
  std::vector<Matrix> reference(horizon_, reference_state);

  Matrix lower_bound(controls_, 1);
  lower_bound << -wheel_single_direction_max_degree_, max_deceleration_;

  Matrix upper_bound(controls_, 1);
```

```
  upper_bound << wheel_single_direction_max_degree_, max_acceleration_;

  const double max = std::numeric_limits<double>::max();
  Matrix lower_state_bound(basic_state_size_, 1);
  Matrix upper_state_bound(basic_state_size_, 1);

  // lateral_error, lateral_error_rate, heading_error, heading_error_rate
  // station_error, station_error_rate
  lower_state_bound << -1.0 * max, -1.0 * max, -1.0 * M_PI, -1.0 * max,
      -1.0 * max, -1.0 * max;
  upper_state_bound << max, max, M_PI, max, max, max;

  double mpc_start_timestamp = Clock::NowInSeconds();
  double steer_angle_feedback = 0.0;
  double acc_feedback = 0.0;
  double steer_angle_ff_compensation = 0.0;
  double unconstrained_control_diff = 0.0;
  double control_gain_truncation_ratio = 0.0;
  double unconstrained_control = 0.0;
  const double v = VehicleStateProvider::Instance()->linear_velocity();
```

2.2.7 优化求解(osqp或linear)

对于车辆误差动力学方程，有

$$x_{k+1} = Ax_k + Bu_k + C$$

状态变量x(k)，输入量u(k)，如下

$$x(k) = \begin{bmatrix} e_l(k) \\ \dot{e_l}(k) \\ e_\psi(k) \\ \dot{e_\psi}(k) \\ e_s(k) \\ \dot{e_s}(k) \end{bmatrix}, \quad u(k) = \begin{bmatrix} \delta(k) \\ a(k) \end{bmatrix}$$

状态量的约束条件为 x_{min} 和 x_{max} ，输入量的约束条件为 u_{min} 和 u_{max} 。

k 时刻的状态代价矩阵为 Q ，输入代价矩阵为 R 。

优化目标函数如下

$$\begin{aligned} &\min_{x_k, u_k} J = \min_{x_k, u_k} \left[\sum_0^N (x_k - x_r)^T Q (x_k - x_r) + \sum_0^{N-1} u_k^T R u_k \right] \\ &x_{k+1} = Ax_k + Bu_k \\ &x_{min} \le x_k \le x_{max} \\ &u_{min} \le u_k \le u_{max} \end{aligned}$$

式中， N 为预测时域 `horizon` 。

```
std::vector<double> control_cmd(controls_, 0);
  if (FLAGS_use_osqp_solver) {
    apollo::common::math::MpcOsqp mpc_osqp(
        matrix_ad_, matrix_bd_, matrix_q_updated_, matrix_r_updated_,
        matrix_state_, lower_bound, upper_bound, lower_state_bound,
        upper_state_bound, reference_state, mpc_max_iteration_, horizon_,
        mpc_eps_);
    if (!mpc_osqp.Solve(&control_cmd)) {
```

```
      AERROR << "MPC OSQP solver failed";
    } else {
      ADEBUG << "MPC OSQP problem solved! ";
      control[0](0, 0) = control_cmd.at(0);
      control[0](1, 0) = control_cmd.at(1);
    }
  } else {
    if (!common::math::SolveLinearMPC(
            matrix_ad_, matrix_bd_, matrix_cd_, matrix_q_updated_,
            matrix_r_updated_, lower_bound, upper_bound, matrix_state_,
            reference, mpc_eps_, mpc_max_iteration_, &control, &control_gai
            &addition_gain)) {
      AERROR << "MPC solver failed";
    } else {
      ADEBUG << "MPC problem solved! ";
    }
  }
```

2.2.7.1 osqp

osqp二次规划的标准形式如下

$$\min_x \frac{1}{2}x^T P x + q^T x$$
$$l \le A_c x \le u$$

上述方程的决策变量 x ，由状态变量和输入构成，维度为 `horizon+1+control` ，如下

$$x = \begin{bmatrix} x(k) \\ x(k+1) \\ \vdots \\ x(k+N) \\ u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

式中， N 为预测步长。

`Hessian` 矩阵P的形式如下(`CalculateKernel`)

$$P = diag(Q, Q, \ldots, Q, R, \ldots, R)$$

`Gradient` 向量q的形式如下(`CalculateGradient`)

> 问题：向量q计算时， x_r 基本为零。osqp的形式只有 $\frac{1}{2}x^T P x$ 起作用，基本就退化为 $\sum_0^N (x_k - x_r)^T Q (x_k - x_r) + \sum_0^{N-1} u_k^T R u_k$ 。
> 对于每个部分的误差，实际上是 $error = Ax_k + Bu_k - x_{k+1}$ ，但 $Ax_k + Bu_k - x_{k+1} = -C$ ，这个误差不会趋近于零。

$$q = \begin{bmatrix} -Qx_r \\ -Qx_r \\ \vdots \\ -Qx_r \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

`Equality Constraint` 矩阵A的形式如下(`CalculateEqualityConstraint`)

$$A_c = \begin{bmatrix} E_x & E_u \\ IE_x & IE_u \end{bmatrix}$$

$$E_x = \begin{bmatrix} -I & 0 & 0 & \dots & 0 \\ A & -I & 0 & \dots & 0 \\ 0 & A & -I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -I \end{bmatrix}, E_u = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ 0 & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B \end{bmatrix}$$

$$IE_x = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, IE_u = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix}$$

`Constraint` 向量l和u的形式如下(`CalculateConstraintVectors`)

$$l = \begin{bmatrix} -x_0 \\ 0 \\ \vdots \\ 0 \\ x_{min} \\ \vdots \\ x_{min} \\ u_{min} \\ \vdots \\ u_{min} \end{bmatrix}, u = \begin{bmatrix} -x_0 \\ 0 \\ \vdots \\ 0 \\ x_{max} \\ \vdots \\ x_{max} \\ u_{max} \\ \vdots \\ u_{max} \end{bmatrix}$$

对于 `MpcOsqp` 对象， `matrix_a` 为系统动力学矩阵， `matrix_b` 为控制矩阵， `matrix_q` 为状态量的代价矩阵， `matrix_r` 为控制量的代价矩阵， `matrix_initial_x` 为初始状态量， `matrix_u_lower` 为控制下限， `matrix_u_upper` 为控制上限， `matrix_x_lower` 为状态量下限， `matrix_x_upper` 为状态量上限， `matrix_x_ref` 为参考状态量， `max_iter` 为最大迭代次数， `horizon` 为预测时域， `eps_abs` 为容忍度。

`state_dim` 为状态量维度， `control_dim` 为控制量维度, `num_param` 为未知。。。

```
MpcOsqp::MpcOsqp(const Eigen::MatrixXd &matrix_a,
                 const Eigen::MatrixXd &matrix_b,
                 const Eigen::MatrixXd &matrix_q,
                 const Eigen::MatrixXd &matrix_r,
                 const Eigen::MatrixXd &matrix_initial_x,
                 const Eigen::MatrixXd &matrix_u_lower,
                 const Eigen::MatrixXd &matrix_u_upper,
                 const Eigen::MatrixXd &matrix_x_lower,
                 const Eigen::MatrixXd &matrix_x_upper,
                 const Eigen::MatrixXd &matrix_x_ref, const int max_iter,
                 const int horizon, const double eps_abs)
    : matrix_a_(matrix_a),
      matrix_b_(matrix_b),
      matrix_q_(matrix_q),
      matrix_r_(matrix_r),
      matrix_initial_x_(matrix_initial_x),
```

```
      matrix_u_lower_(matrix_u_lower),
      matrix_u_upper_(matrix_u_upper),
      matrix_x_lower_(matrix_x_lower),
      matrix_x_upper_(matrix_x_upper),
      matrix_x_ref_(matrix_x_ref),
      max_iteration_(max_iter),
      horizon_(horizon),
      eps_abs_(eps_abs) {
  state_dim_ = matrix_b.rows();
  control_dim_ = matrix_b.cols();
  ADEBUG << "state_dim" << state_dim_;
  ADEBUG << "control_dim_" << control_dim_;
  num_param_ = state_dim_ * (horizon_ + 1) + control_dim_ * horizon_;
}
```

2.2.7.2 linear

待补充

2.2.8 转角和加速度反馈

输出参数为前轮转角 `steer_angle_feedback` 和加速度增量 `acc_feedback`，如下

$$U = \begin{bmatrix} \delta_f \\ \Delta a_x \end{bmatrix}$$

```
steer_angle_feedback = Wheel2SteerPct(control[0](0, 0));
  acc_feedback = control[0](1, 0);
```

2.2.9 前馈补偿

控制时序 `control_gain`，参考时序 `addition_gain`，参考公式如下

$$\delta_{ff} = \frac{L}{R} + K_v a_y - k_3 \left[\frac{l_r}{R} - \frac{l_f}{2C_{\alpha r}} \frac{m{V_x}^2}{Rl} \right]$$

代码里用的公式进行了一些改动，如下

$$\delta_{ff\_2} = -k_3 \kappa \left[l_r - \frac{l_f}{2C_{\alpha r}} \frac{m{V_x}^2}{L} \right]$$

但实际计算的公式多了一项，如下

$$\delta_{ff\_2} = -k_3 \kappa \left[l_r - \frac{l_f}{2C_{\alpha r}} \frac{m{V_x}^2}{L} \right] - v\kappa \cdot k_{addition}$$

这里的 k_3 是求解无约束规划问题的黎卡提方程得到的，`addition_gain` 不知道是什么。

> 为什么 k_3 用的是黎卡提方程的解?

```
for (int i = 0; i < basic_state_size_; ++i) {
    unconstrained_control += control_gain[0](0, i) * matrix_state_(i, 0);
  }
  unconstrained_control += addition_gain[0](0, 0) * v * debug->curvature();
  if (enable_mpc_feedforward_compensation_) {
    unconstrained_control_diff =
        Wheel2SteerPct(control[0](0, 0) - unconstrained_control);
```

```
    if (fabs(unconstrained_control_diff) <= unconstrained_control_diff_limi
      steer_angle_ff_compensation =
          Wheel2SteerPct(debug->curvature() *
                         (control_gain[0](0, 2) *
                              (lr_ - lf_ / cr_ * mass_ * v * v / wheelbase_
                          addition_gain[0](0, 0) * v));
    } else {
      control_gain_truncation_ratio = control[0](0, 0) / unconstrained_cont
      steer_angle_ff_compensation =
          Wheel2SteerPct(debug->curvature() *
                         (control_gain[0](0, 2) *
                              (lr_ - lf_ / cr_ * mass_ * v * v / wheelbase_
                          addition_gain[0](0, 0) * v) *
                         control_gain_truncation_ratio);
    }
  } else {
    steer_angle_ff_compensation = 0.0;
  }
```

2.2.10 限制和输出转角

`steer_angle` 由三部分组成，分别是转角反馈、转角前馈1和转角前馈2。

```
// TODO(QiL): evaluate whether need to add spline smoothing after the resul
  double steer_angle = steer_angle_feedback +
                       steer_angle_feedforwardterm_updated_ +
                       steer_angle_ff_compensation;
  if (FLAGS_set_steer_limit) {
    const double steer_limit =
        std::atan(max_lat_acc_ * wheelbase_ /
                  (VehicleStateProvider::Instance()->linear_velocity() *
                   VehicleStateProvider::Instance()->linear_velocity())) *
        steer_ratio_ * 180 / M_PI / steer_single_direction_max_degree_ * 10(

    // Clamp the steer angle with steer limitations at current speed
    double steer_angle_limited =
        common::math::Clamp(steer_angle, -steer_limit, steer_limit);
    steer_angle_limited = digital_filter_.Filter(steer_angle_limited);
    steer_angle = steer_angle_limited;
    debug->set_steer_angle_limited(steer_angle_limited);
  }
  steer_angle = digital_filter_.Filter(steer_angle);
  // Clamp the steer angle to -100.0 to 100.0
  steer_angle = common::math::Clamp(steer_angle, -100.0, 100.0);
  cmd->set_steering_target(steer_angle);
```

2.2.11 限制和输出加速度

`acceleration_cmd` 由两部分组成，分别是加速度反馈 `acc_feedback` 和加速度参考 `acceleration_reference` 。

`FLAGS_steer_angle_rate` 默认为100。

```
debug->set_acceleration_cmd_closeloop(acc_feedback);

  double acceleration_cmd = acc_feedback + debug->acceleration_reference();
  // TODO(QiL): add pitch angle feed forward to accommodate for 3D control
```

```
if ((planning_published_trajectory->trajectory_type() ==
      apollo::planning::ADCTrajectory::NORMAL) &&
     (std::fabs(debug->acceleration_reference()) <=
          max_acceleration_when_stopped_ &&
      std::fabs(debug->speed_reference()) <= max_abs_speed_when_stopped_))
  acceleration_cmd =
      (chassis->gear_location() == canbus::Chassis::GEAR_REVERSE)
          ? std::max(acceleration_cmd, -standstill_acceleration_)
          : std::min(acceleration_cmd, standstill_acceleration_);
  ADEBUG << "Stop location reached";
  debug->set_is_full_stop(true);
}
// TODO(Yu): study the necessity of path_remain and add it to MPC if need

debug->set_acceleration_cmd(acceleration_cmd);
double calibration_value = 0.0;
if (FLAGS_use_preview_speed_for_table) {
  calibration_value = control_interpolation_->Interpolate(
      std::make_pair(debug->speed_reference(), acceleration_cmd));
} else {
  calibration_value = control_interpolation_->Interpolate(std::make_pair(
      VehicleStateProvider::Instance()->linear_velocity(), acceleration_c
}

debug->set_calibration_value(calibration_value);

double throttle_cmd = 0.0;
double brake_cmd = 0.0;
if (calibration_value >= 0) {
  throttle_cmd = std::max(calibration_value, throttle_lowerbound_);
  brake_cmd = 0.0;
} else {
  throttle_cmd = 0.0;
  brake_cmd = std::max(-calibration_value, brake_lowerbound_);
}

cmd->set_steering_rate(FLAGS_steer_angle_rate);
// if the car is driven by acceleration, disgard the cmd->throttle and br
cmd->set_throttle(throttle_cmd);
cmd->set_brake(brake_cmd);
cmd->set_acceleration(acceleration_cmd);
```

2.2.12 debug更新计算数据

```
debug->set_heading(VehicleStateProvider::Instance()->heading());
  debug->set_steering_position(chassis->steering_percentage());
  debug->set_steer_angle(steer_angle);
  debug->set_steer_angle_feedforward(steer_angle_feedforwardterm_updated_);
  debug->set_steer_angle_feedforward_compensation(steer_angle_ff_compensati
  debug->set_steer_unconstrained_control_diff(unconstrained_control_diff);
  debug->set_steer_angle_feedback(steer_angle_feedback);
  debug->set_steering_position(chassis->steering_percentage());
```

2.2.13 输出挡位

若 `速度小于停车最大平均车速` 或 `挡位处于规划挡位` 或 `档位处于空挡` ，则设置挡位为规划挡位；否则设置挡位为底盘所处挡位。

```
if (std::fabs(VehicleStateProvider::Instance()->linear_velocity()) <=
          vehicle_param_.max_abs_speed_when_stopped() ||
      chassis->gear_location() == planning_published_trajectory->gear() ||
      chassis->gear_location() == canbus::Chassis::GEAR_NEUTRAL) {
    cmd->set_gear_location(planning_published_trajectory->gear());
  } else {
    cmd->set_gear_location(chassis->gear_location());
  }
```

2.2.14 debug更新chassis

```
ProcessLogs(debug, chassis);
```

2.3 返回

```
return Status::OK();
```

3 结语

Mpc的模块解析写的比较仓促，有一个地方仍然没有弄清楚（`addition_gain`），欢迎大家批评指正

编辑于 2021-09-15 16:31

模型预测控制 Matrix 矩阵

文章被以下专栏收录

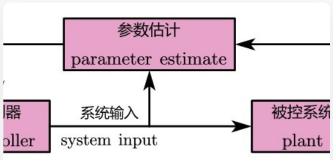
自动驾驶
主要为学习自动驾驶技术的朋友们分享一些见解

推荐阅读

MPC-GPS：模型预测控制指导策略搜索

原文链接： Zhang T, Kahn G, Levine S, et al. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search[C]//2016 IEEE international conference ...

企鹅 发表于深度学习与...



Certainty-equivalence based adaptive control

皮皮夏 发表于控制算法手...

AUTOSAR架构MBD基于模型设计

在AUTODSAR架构下，使用simulink基于模型设计有两种开发方式。第一种是自下而上，即先在AUTOSAR工具中配置好SWC信息，将配置信息导出arxml文件，再import进Simulink中进行设计。

非有为青年

基于车辆运动学的模型预测控制MPC（一） 非线性优化

模型预测控制本质上是一种基于优化控制算法，通过对车辆的运动状态进行预测，然后优化目标，求解有限时间内的开环优化问题，并将预测时间周期的前部分控制量作用于被控对象。 在上一篇文章...

羽哥

5 条评论

切换为时间排序

写下你的评论...

花衬衫 2021-09-22

我就一直没有搞懂 mpc 动力学方程最后一行的意思，没想到6.0把期望航向角变化率那项给删了，哈哈。

赞

隐匿潮水之中 (作者) 回复 花衬衫 2021-09-22

哈哈 这个动力学方程一直很坑，主要是Apollo没有写参考的是什么论文

赞

隐匿潮水之中 (作者) 回复 花衬衫 2021-09-22

航向角变化率那一项就是错的，5.5没修正，6.0改了

赞

展开其他 2 条回复