Math Background: Nonlinear MPC → Real-Time LTV-QP

`nav_mpc` turns a user-defined **continuous-time nonlinear MPC** problem into a **linear time-varying quadratic program (LTV-QP)** that can be updated and solved in real time with OSQP.

The key idea is:

> Nonlinear optimal control is approximated locally by a sequence of fast convex QPs whose structure stays fixed while their numerical values update online.

User-defined problem (continuous-time NMPC)

The user defines (symbolically):

- dynamics: $\dot{x}(t) = f(x(t), u(t))$
- constraints: $g(x(t), u(t)) \leq 0$
- tracking / objective error: $e(x(t), r(t))$

These expressions may be **fully nonlinear**.

A standard continuous-time NMPC problem over horizon $T = N\,\Delta t$ is:

$$
\begin{aligned}
\min_{u(\cdot)} \quad & \int_0^T \left(\tfrac{1}{2}\, e(x(t), r(t))^\top Q\, e(x(t), r(t)) + \tfrac{1}{2}\, (u(t) - u_{\text{ref}})^\top R\, (u(t) - u_{\text{ref}}) \right) dt \\
& + \tfrac{1}{2}\, e(x(T), r(T))^\top Q_N\, e(x(T), r(T)) \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \\
& g(x(t), u(t)) \leq 0, \\
& x(0) = x_{\text{meas}}.
\end{aligned}
$$

where e, f, g are nonlinear functions.

LTV-MPC and QP formulation in OSQP

To obtain a tractable real-time problem, we approximate the nonlinear system **locally** at each MPC step.

We maintain a **linearization trajectory** $\{\bar{x}_k, \bar{u}_k\}_{k=0}^{N-1}$ typically the shifted previous optimal solution. All nonlinear terms are linearized around these operating points. The resulting finite-horizon optimal control problem becomes a **linear time-varying MPC**:

$$
\begin{aligned}
\min_{x,u} \quad & (x_N - x_r)^\top Q_N (x_N - x_r) + \sum_{k=0}^{N-1} \left((x_k - x_r)^\top Q (x_k - x_r) + (u_k - u_{\text{ref}})^\top R (u_k - u_{\text{ref}}) \right) \\
\text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k + c_k, \quad k = 0, \dots, N-1 \\
& x_{\min,k} \leq x_k \leq x_{\max,k}, \quad k = 0, \dots, N \\
& u_{\min,k} \leq u_k \leq u_{\max,k}, \quad k = 0, \dots, N-1 \\
& x_0 = x_{\text{meas}}
\end{aligned}
$$

This problem is now **quadratic with linear constraints**, which means it can be written as a standard QP.

Mapping to OSQP canonical form

OSQP expects problems of the form

$$
\begin{aligned}
\min_z \quad & \tfrac{1}{2} z^\top P z + q^\top z \\
\text{s.t.} \quad & l \leq A z \leq u.
\end{aligned}
$$

We therefore stack all states and inputs into one vector

$$
z = \begin{bmatrix} x_0 \\ \vdots \\ x_N \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}, \qquad n_z = (N+1) n_x + N n_u.
$$

All MPC constraints can be written compactly as

$$
A = \begin{bmatrix} A_{\text{eq}} \\ A_{\text{ineq}} \end{bmatrix}, \quad l = \begin{bmatrix} b_{\text{eq}} \\ l_{\text{ineq}} \end{bmatrix}, \quad u = \begin{bmatrix} b_{\text{eq}} \\ u_{\text{ineq}} \end{bmatrix}.
$$

Importantly:

> **The sparsity pattern of P and A is constant: Only their numerical values change each MPC step.**

Objective function The quadratic cost has the OSQP form $\frac{1}{2} z^\top P z + q^\top z$, where the Hessian is block diagonal:

$$
P_k = \operatorname{diag}(Q_0, \dots, Q_N,\ R_0, \dots, R_{N-1}) \in \mathbb{R}^{n_z \times n_z}.
$$

```
P_k = blkdiag( Q_0, ..., Q_N, R_0, ..., R_{N-1} )

      [ Q_0    0     ...    0      |   0     ...    0   ]
      [  0    Q_1    ...    0      |   0     ...    0   ]
      [ ...   ...    ...   ...     |  ...    ...   ...  ]
      [  0     0     ...   Q_N     |   0     ...    0   ]
      --------------------------------+------------------
      [  0     0     ...    0      |  R_0    ...    0   ]
      [  0     0     ...    0      |   0     ...    0   ]
      [ ...   ...    ...   ...     |  ...    ...   ...  ]
      [  0     0     ...    0      |   0     ...  R_{N-1}]
```

The linear term arises from expanding tracking penalties:

$$(x_k - x_r)^\top Q(x_k - x_r) = x_k^\top Q x_k - 2x_r^\top Q x_k + \text{const.}$$

Thus

$$q = \begin{bmatrix} -Q_0 x_r \\ \vdots \\ -Q_N x_r \\ -R_0 u_{\text{ref}} \\ \vdots \\ -R_{N-1} u_{\text{ref}} \end{bmatrix}.$$

If the tracking error is nonlinear $e(x, r)$, we linearize it:

$$e(x_k, r_k) \approx e(\bar{x}_k, r_k) + E_k(x_k - \bar{x}_k).$$

Define

$$b_k^e = e(\bar{x}_k, r_k) - E_k \bar{x}_k.$$

Then

$$e(x_k, r_k) \approx E_k x_k + b_k^e.$$

Substituting into the quadratic cost produces a local quadratic approximation:

$$Q_k^{\text{local}} = E_k^\top Q E_k, \qquad q_k^{\text{local}} = E_k^\top Q b_k^e.$$

This is how nonlinear objectives become QP-compatible.

Dynamics (equality constraints) Using the **linearization trajectory** $\{\bar{x}_k, \bar{u}_k\}_{k=0}^{N-1}$ we discretize and linearize the dynamics around $(\bar{x}_k, \bar{u}_k)$:

$$x_{k+1} \approx A_k\, x_k + B_k\, u_k + c_k,$$

where A_k, B_k, c_k are produced from exact symbolic Jacobians and a (2nd-order) Taylor discretization:

$$A_k = A_d(\bar{x}_k, \bar{u}_k), \quad B_k = B_d(\bar{x}_k, \bar{u}_k), \quad c_k = c_d(\bar{x}_k, \bar{u}_k).$$

Stacked into OSQP equalities, we write the residual form:

$$x_0 = x_{\text{meas}}, \qquad x_{k+1} - A_k x_k - B_k u_k = c_k, \quad k = 0, \dots, N-1.$$

This yields

$$A_{\text{eq}} z = b_{\text{eq}}, \qquad b_{\text{eq}} = \begin{bmatrix} x_{\text{meas}} \\ c_0 \\ \vdots \\ c_{N-1} \end{bmatrix} \in \mathbb{R}^{(N+1)n_x}.$$

The matrix $A_{\text{eq}} \in \mathbb{R}^{(N+1)n_x \times n_z}$ has the standard MPC band structure with constant sparsity:

```
A_eq =
[   I      0      0      ...    0 |  0     0   ...     0   ]
[ -A_0     I      0      ...    0 |-B_0    0   ...     0   ]
[   0    -A_1     I      ...    0 |  0   -B_1  ...     0   ]
[  ...                          | ...                    ]
[   0      0     ...  -A_{N-1}  I |  0     0   ... -B_{N-1}]
```

In OSQP bound form, we enforce equalities by setting:

$$l_{\text{eq}} = u_{\text{eq}} = b_{\text{eq}}.$$

Inequalities Nonlinear inequality constraints $g(x_k, u_k) \le 0$ are linearized per stage:

$$G_k^x x_k + G_k^u u_k \le b_k, \qquad b_k := -\Big(g(\bar{x}_k, \bar{u}_k) - G_k^x \bar{x}_k - G_k^u \bar{u}_k\Big).$$

These constraints are stacked into a sparse block $A_g z \le b$. In OSQP bound form, such inequalities use $-\infty$ as the lower bound.

Nonlinear inequality constraints $g(x_k, u_k) \le 0$ are linearized as:

$$g(x_k, u_k) \approx g(\bar{x}_k, \bar{u}_k) + G_k^x(x_k - \bar{x}_k) + G_k^u(u_k - \bar{u}_k) \leq 0$$

Rearranging yields:

$$G_k^x\, x_k + G_k^u\, u_k \leq b_k, \qquad b_k := -\Big(g(\bar{x}_k, \bar{u}_k) - G_k^x \bar{x}_k - G_k^u \bar{u}_k\Big).$$

All inequalities are stacked into $A_{\text{ineq}} z \leq u_{\text{ineq}}$ and represented in OSQP bound form using (l, u) with $l = -\infty$ on inequality rows.

Offline vs online in `nav_mpc` (how it stays fast)

Offline (`core/mpc2qp/qp_offline.py`) - Build the **fixed sparsity pattern** of P and A once (CSC format). - Precompute **index maps** pointing to exact memory locations of time-varying entries (e.g., where $-A_k$, $-B_k$, G_k^x, G_k^u live inside `A.data`). - Generate compiled stage kernels via SymPy Jacobians + `autowrap(..., backend="cython")`: - $A_k(\bar{x}_k, \bar{u}_k)$, $B_k(\bar{x}_k, \bar{u}_k)$, $c_k(\bar{x}_k, \bar{u}_k)$ - $G_k^x(\bar{x}_k, \bar{u}_k)$, $G_k^u(\bar{x}_k, \bar{u}_k)$, $g(\bar{x}_k, \bar{u}_k)$ - $e(\bar{x}_k, r_k)$, $E_k(\bar{x}_k, r_k)$

Online (`core/mpc2qp/qp_online.py`) - Shift the previous solution to get new linearization points: $\bar{x}_k \leftarrow x_{k+1}^*$, $\bar{u}_k \leftarrow u_{k+1}^*$ (with terminal extrapolation). - Evaluate compiled kernels stage-wise to obtain $\{A_k, B_k, c_k, G_k^x, G_k^u, b_k\}$. - Overwrite **only** time-varying values in `A.data`, `l`, `u`, and (optionally) `P.data`, `q` using the precomputed index maps (no sparsity changes, no reallocations). - Call OSQP `prob.update(Px=..., q=..., Ax=..., l=..., u=...)` and solve.

This design leverages the fact that OSQP requires a **constant sparsity pattern**. `nav_mpc` keeps structure fixed and performs fast in-place numeric updates each MPC step.

Final interpretation

At runtime each MPC iteration performs:

1. Shift previous solution → new linearization points
2. Evaluate Jacobians → obtain A_k, B_k, c_k, G_k
3. Update numeric entries of P, q, A, l, u
4. Solve QP with OSQP

Because sparsity is fixed → updates are extremely fast.
Because problem is quadratic → solution is extremely fast.