Control

- Perspectiva de un ignorante - preguntas a la IA - Non Linear Model Predictive Control

Perspectiva de un ignorante - preguntas a la IA - Non Linear Model Predictive Control

¿Qué problema se resuelve en un solver de MPC no lineal?

Un **solver de MPC no lineal (Nonlinear Model Predictive Control, NMPC)** resuelve un **problema de optimización no lineal restringido**, cuyo objetivo es encontrar la **secuencia óptima de acciones de control** que lleven al sistema a un comportamiento deseado, respetando tanto la **dinámica no lineal del sistema** como las **restricciones impuestas** (por ejemplo, límites en estados o entradas).

□□ Problema que resuelve el solver NMPC:

Dado un modelo dinámico no lineal:

$\dot{x}(t) = f(x(t), u(t))$

o en forma discreta:

$x_{k+1} = f(x_k, u_k)$

y una **función de costo** a minimizar del tipo:

$J = \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)$

sujeta a:

- Dinámica del sistema: $x_{k+1} = f(x_k, u_k)$
- Restricciones en los estados: $x_k \in \mathcal{X}$
- Restricciones en las entradas: $u_k \in \mathcal{U}$

El solver NMPC resuelve el siguiente problema de optimización en cada instante de control:

Problema de optimización no lineal restringido (NLP):

$\min_{u_0, u_1, ..., u_{N-1}} \quad \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)$

sujeto a:

- $x_{k+1} = f(x_k, u_k), \quad \forall k = 0,...,N-1$
- $x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}$
- $x_0 = x_{\text{actual}}$ (condición inicial)

¿Qué hace el solver exactamente?

1. **Predice** la evolución futura del sistema usando el modelo no lineal.
2. **Optimiza** la secuencia de controles futuros en un horizonte NN.
3. **Respeta** todas las restricciones dinámicas y físicas.
4. **Ejecuta solo el primer control** de la secuencia óptima.
5. **Repite el proceso** en el siguiente instante (receding horizon).

Ejemplo de aplicación:

Control de un robot con dinámica compleja, como un dron o un vehículo autónomo, donde las ecuaciones de movimiento son no lineales, y se deben evitar obstáculos y respetar límites de velocidad o aceleración.

¿Quieres que te muestre un ejemplo numérico o código con CasADi o ACADO?

¿Y ese problema de optimización cómo se plantea y se resuelve matematicamente?

Excelente pregunta. Vamos a profundizar paso a paso en **cómo se plantea y se resuelve matemáticamente** el problema de optimización que enfrenta un **solver de MPC no lineal (NMPC)**.

---

1. Planteamiento del problema matemático (NLP)

El problema se plantea como un **problema de optimización no lineal restringido (NLP)**, donde se desea encontrar la secuencia de controles {u0,u1,...,uN−1}\{u_0, u_1, ..., u_{N-1}\} que minimiza un costo sujeto a restricciones dinámicas y físicas.

Formulación estándar (discreta):

min{xk,uk}J=∑k=0N−1ℓ(xk,uk)+ℓf(xN)\min_{\{x_k, u_k\}} \quad J = \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)

sujeto a:

1. **Dinámica del sistema** (no lineal):
   xk+1=f(xk,uk),∀k=0,...,N−1x_{k+1} = f(x_k, u_k), \quad \forall k = 0, ..., N-1
2. **Restricciones de estado y control**:
   xk∈X⊂Rn,uk∈U⊂Rmx_k \in \mathcal{X} \subset \mathbb{R}^n, \quad u_k \in \mathcal{U} \subset \mathbb{R}^m
3. **Condición inicial**:
   x0=xmedidox_0 = x_{\text{medido}}

---

2. Discretización del sistema dinámico

El modelo continuo $\dot{x}(t) = f(x(t), u(t))$ se **discretiza** en el tiempo usando algún método de integración:

- **Euler explícito**:
  $x_{k+1} = x_k + \Delta t \cdot f(x_k, u_k)$
- **Runge-Kutta** (RK4 es muy común en NMPC):
  $x_{k+1} = \Phi(x_k, u_k)$

Esto convierte el modelo en un conjunto de **igualdades algebraicas no lineales**.

---

3. Reescritura como NLP (con todas las variables)

Una vez discretizado, se agrupan todas las variables:

- Variables de decisión:
  $z = \{x_1, ..., x_N, u_0, ..., u_{N-1}\}$
- Función objetivo:
  $J(z) = \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)$
- Restricciones de igualdad:
  $h(z) = x_{k+1} - f(x_k, u_k) = 0$
- Restricciones de desigualdad:
  $g(z) \leq 0 \quad \text{(por ejemplo, límites en } x_k, u_k \text{)}$

Entonces, el problema se convierte en:

$$\begin{aligned} \min_{z} & \quad J(z) \\ \text{sujeto a} & \quad h(z) = 0 \\ & \quad g(z) \leq 0 \end{aligned}$$

---

4. Resolución del problema (técnicas)

Este NLP se resuelve con métodos de **optimización numérica no lineal**, como:

Métodos de punto interior (IPM)

- Utilizados por **IPOPT**, **CasADi**, **FORCES Pro**.
- Introducen penalizaciones logarítmicas para desigualdades.
- Convierten el problema a un problema KKT (Karush-Kuhn-Tucker).
- Iterativamente resuelven sistemas lineales para actualizar zz.

SQP (Sequential Quadratic Programming

- Aproximan el NLP por una **QP (programación cuadrática)** en cada iteración.
- Usan linealización de restricciones y cuadratización del costo.
- ACADO y algunas implementaciones en C usan este método.

---

5. Condiciones de optimalidad (KKT)

Para el óptimo $z*z^*$, se deben cumplir las **condiciones KKT**:

1. **Gradiente nulo del lagrangiano**:
   $\nabla_z L(z*, \lambda*, \mu*) = 0$
2. **Restricciones primales**:
   $h(z*) = 0, g(z*) \leq 0$
3. **Multiplicadores duales**:
   $\mu* \geq 0, \mu_i * g_i(z*) = 0 \quad \text{(complementariedad)}$

□□ Receding Horizon (horizonte móvil)

Una vez resuelto el problema:

1. Se **aplica solo el primer control óptimo** u0∗u_0^*.
2. Se **mide** el nuevo estado del sistema.
3. Se **replantea y resuelve** el problema nuevamente con el nuevo x0x_0.

□□ Ejemplo ilustrativo (simplificado)

Supongamos un sistema no lineal:

xk+1=xk+Δt(xk2+uk)x_{k+1} = x_k + \Delta t (x_k^2 + u_k)

Costo:

J=∑k=0N−1(xk2+uk2)J = \sum_{k=0}^{N-1} (x_k^2 + u_k^2)

Restricciones:

- −1≤uk≤1-1 \leq u_k \leq 1
- x0=1x_0 = 1

Esto se discretiza, se plantea como NLP, y se resuelve numéricamente (por ejemplo, usando IPOPT o CasADi).

¿Te gustaría que resolviera este ejemplo con código para verlo en práctica? ¿O prefieres explorar un sistema realista como el péndulo invertido o un robot móvil?

¿Que son igualdades algebraicas no lineales y como se resuelven?¿Que metodos numericos existen?

Muy buena pregunta. Vamos por partes para entender **qué son las igualdades algebraicas no lineales**, **cómo se resuelven**, y **qué métodos numéricos existen**.

□ ¿Qué son igualdades algebraicas no lineales?

Son **ecuaciones en las que intervienen variables relacionadas a través de funciones no lineales**, y **no contienen derivadas** (a diferencia de ecuaciones diferenciales).

□□ Definición formal:

Una **igualdad algebraica no lineal** tiene la forma:

f(x1,x2,...,xn)=0f(x_1, x_2, ..., x_n) = 0
donde:

- ff es **no lineal** (ej. contiene términos como x2,sin(x),ex,x·yx^2, \sin(x), e^x, x \cdot y, etc.)
- x1,...,xnx_1, ..., x_n son las variables desconocidas a resolver.

□□ Ejemplos:

- x2+y2−1=0x^2 + y^2 - 1 = 0 → ecuación de un círculo.
- sin(x)+x·y−3=0\sin(x) + x \cdot y - 3 = 0
- ex+ln(y)−5=0e^x + \ln(y) - 5 = 0

En contextos como el **NMPC**, estas igualdades suelen representar:

- Las **restricciones de dinámica discretizada**:
  Ej: xk+1=f(xk,uk)⇒xk+1−f(xk,uk)=0x_{k+1} = f(x_k, u_k) \Rightarrow x_{k+1} - f(x_k, u_k) = 0

□ ¿Cómo se resuelven?

El objetivo es **encontrar los valores de las variables** que satisfacen simultáneamente todas las ecuaciones:

Encontrar x∈Rn tal que fi(x)=0,i=1,...,m\text{Encontrar } x \in \mathbb{R}^n \text{ tal que } f_i(x) = 0, \quad i = 1, ..., m

Esto se llama un **sistema de ecuaciones no lineales**.

🧠 Métodos numéricos para resolverlas

Dado que la mayoría de estas ecuaciones **no tienen solución analítica**, usamos métodos numéricos **iterativos**.

🔁 1. Método de Newton-Raphson

Es el más clásico y usado.

Idea básica (1 variable):

xk+1=xk−f(xk)f′(xk)x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}

Multivariable (sistema no lineal):

xk+1=xk−Jf(xk)−1f(xk)x_{k+1} = x_k - J_f(x_k)^{-1} f(x_k)

- Jf(xk)J_f(x_k): jacobiano de ff evaluado en xkx_k
- Se resuelve un sistema lineal en cada iteración:
  Jf(xk)⋅δx=−f(xk),xk+1=xk+δxJ_f(x_k) \cdot \delta x = -f(x_k), \quad x_{k+1} = x_k + \delta x

✅ **Ventajas**:

- Rápida convergencia **(cuadrática)** cerca de la solución.

⚠️ **Desventajas**:

- Necesita una buena estimación inicial.
- Necesita el **jacobiano** (matriz de derivadas), que puede ser costoso de calcular.

---

2. Métodos de Newton modificados o cuasi-Newton

Usan una aproximación del Jacobiano:

- **Broyden's method**
- **DFP, BFGS (usados en optimización)**

Estos reducen el coste computacional cuando JfJ_f es difícil de obtener.

---

3. Métodos de punto fijo (fixed-point iteration)

Si puedes reescribir la ecuación como:

x=g(x)x = g(x)
entonces iteras:

xk+1=g(xk)x_{k+1} = g(x_k)
 Simple, perd**ento** y depende de que gg sea contractivo.

---

4. Métodos de homotopía o continuación

Resuelven una familia de problemas que conecta uno fácil con el problema original.

H(x,λ)=(1−λ)·f0(x)+λ·f(x)H(x, \lambda) = (1 - \lambda) \cdot f_0(x) + \lambda \cdot f(x)
Se empieza con λ=0\lambda = 0, y se avanza hasta λ=1\lambda = 1.

 Útiles cuando el sistema tiene múltiples soluciones o es difícil de resolver directamente.

□ En práctica: ¿cómo se aplican en MPC?

En MPC no lineal:

- Las **igualdades no lineales** que surgen de la dinámica se **incluyen como restricciones** en el problema de optimización.
- El **solver del NLP** (como IPOPT, SNOPT, etc.) internamente resuelve estos sistemas con **Newton-Raphson** o variantes (ej. SQP o IPM), cada vez que necesita evaluar la factibilidad o buscar una dirección de descenso.

□□ Ejemplo breve (Newton en 2 variables)

Queremos resolver:

{x2+y−37=0x−y2−5=0\begin{cases} x^2 + y - 37 = 0 \\ x - y^2 - 5 = 0 \end{cases}

Pasos:

1. Definir f(x,y)=[x2+y−37x−y2−5]f(x, y) = \begin{bmatrix} x^2 + y - 37 \\ x - y^2 - 5 \end{bmatrix}
2. Calcular su Jacobiano:
   J=[2x11−2y]J = \begin{bmatrix} 2x & 1 \\ 1 & -2y \end{bmatrix}
3. Iterar usando Newton hasta que ‖f(x,y)‖\|f(x, y)\| sea pequeño.

¿Te gustaría ver esto resuelto paso a paso con código (Python/NumPy) o con una herramienta como **CasADi** para usarlo en MPC?