Universidad Anáhuac Oaxaca

Diego Aimar Gómez de la Rosa
Ingeniería en tecnologías de la información y negocios digitales
Materia: Estática

02 de Diciembre, 2025

Índice

1. Introducción

La industria automotriz moderna, en el contexto de la Industria 4.0, exige niveles de precisión submilimétrica en sus procesos de manufactura. Los manipuladores robóticos son piezas centrales en estas líneas de producción. Sin embargo, un desafío persistente en el control de software para estos robots es la influencia de las fuerzas gravitacionales, las cuales introducen errores de posición (conocidos como *steady-state error* o çaída") si no son compensadas activamente.

Este reporte detalla el diseño metodológico y la implementación de software para un sistema de **Compensación de Gravedad Estática**. El objetivo es desarrollar un algoritmo capaz de predecir el torque exacto requerido en cada articulación para sostener una carga pesada en equilibrio, basándose en modelos físicos matemáticos, eliminando la necesidad de corrección por error y retroalimentación reactiva, lo cual optimiza el consumo energético y la seguridad operativa.

2. Marco Teórico

2.1. Planteamiento del Problema: La Celda de Manufactura

En una planta de ensamblaje automotriz real (ej. planta de transmisiones), se requiere un brazo robótico de 2 Grados de Libertad (2-DOF) en el plano vertical para realizar la tarea de *Pick-and-Place* de convertidores de par.

- **La Carga:** Un convertidor de par con una masa (m_p) de **15 kg**.
- **El Defecto:** Al extender el brazo para colocar la pieza en la carcasa de la transmisión, el software de control actual utiliza un control PID estándar. Debido a la gravedad, el brazo tiende a çolgarseünos milímetros antes de que el término integral del PID corrija la posición. Este retraso causa colisiones menores y desgaste en los engranajes.
- **El Requisito de Ingeniería:** Se necesita un módulo de software *Feed-Forward* que calcule, antes de moverse, cuánto torque debe aplicar el motor para anular el peso de la pieza de 15 kg más el peso propio del robot, garantizando que $\sum F = 0$ y $\sum \tau = 0$ en todo momento.

2.2. Solución Propuesta: Modelo Matemático

La solución consiste en implementar las ecuaciones de la estática de cuerpos rígidos mediante el método recursivo de Newton-Euler. El sistema se modela como una cadena cinemática abierta de cuerpos rígidos. Para que el sistema permanezca estático, la suma de torques (τ) generados por los motores debe ser igual y opuesta a los torques generados por la gravedad.

A) Análisis del Eslabón 2 (Muñeca/Codo): El torque en la articulación 2 (τ_2) debe soportar el momento generado por la masa del eslabón 2 (m_2) y la masa de la carga (m_p).

$$\tau_2 = g \cdot \cos(\theta_1 + \theta_2) \cdot (m_2 r_2 + m_p l_2) \tag{1}$$

Donde r_2 es la distancia al centro de masa del eslabón 2 (asumido $l_2/2$).

B) Análisis del Eslabón 1 (Hombro/Base): El torque en la articulación 1 (τ_1) debe soportar el momento de todo el sistema proyectado sobre el eje horizontal desde la base.

$$\tau_1 = g \cdot [(m_1 r_1 + m_2 l_1 + m_p l_1)\cos(\theta_1) + (m_2 r_2 + m_p l_2)\cos(\theta_1 + \theta_2)] \tag{2}$$

Esta solución matemática convierte un problema físico en una función determinista que el software puede ejecutar en microsegundos.

3. Estudio de Caso y Tablas de Cálculo

Para validar el algoritmo antes de la implementación, se realizan cálculos teóricos de tres configuraciones críticas en la línea de montaje.

Parámetros del Robot: $m_1 = 12$kg, $l_1 = 1{,}0$m, $m_2 = 8$kg, $l_2 = 0{,}8$m, $m_p = 15$kg.

Tabla 1: Cálculo de Torques Requeridos (Validación Teórica)

Caso	**Configuración**	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\tau_2}$ (Nm)	$\boldsymbol{\tau_1}$ (Nm)
A	Extensión Máxima	0°	0°	**149.11**	**480.69**
B	Retracción Parcial	45°	−45°	**149.11**	**234.39**
C	Posición Vertical	90°	0°	**0.00**	**0.00**

4. Implementación del Algoritmo (C++)

A continuación se presenta el código fuente diseñado para ser integrado en el controlador del robot.

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <iomanip>

const double GRAVEDAD = 9.81;
const double PI = 3.14159265358979323846;

struct ParametrosRobot {
```

```
    double m1, l1;
    double m2, l2;
};

struct ResultadoTorque {
    double torque_hombro;
    double torque_codo;
};

class ControladorRobot {
private:
    ParametrosRobot params;

    double aRadianes(double grados) {
        return grados * PI / 180.0;
    }

public:
    ControladorRobot(ParametrosRobot p) : params(p) {}

    ResultadoTorque calcularTorqueEstatico(double carga_kg, double
    theta1_deg, double theta2_deg) {

        double q1 = aRadianes(theta1_deg);
        double q2 = aRadianes(theta2_deg);
        double q12 = q1 + q2;

        double r1 = params.l1 / 2.0;
        double r2 = params.l2 / 2.0;

        double momento_masa2 = params.m2 * r2;
        double momento_carga = carga_kg * params.l2;
        double t2 = GRAVEDAD * std::cos(q12) * (momento_masa2 +
    momento_carga);

        double termino_link1 = (params.m1 * r1 + params.m2 * params.l1 +
    carga_kg * params.l1) * std::cos(q1);
        double termino_link2 = (params.m2 * r2 + carga_kg * params.l2) *
    std::cos(q12);

        double t1 = GRAVEDAD * (termino_link1 + termino_link2);

        return {t1, t2};
    }
};

int main() {
    ParametrosRobot robotIndustrial = {12.0, 1.0, 8.0, 0.8};
    ControladorRobot controlador(robotIndustrial);
    double carga_convertidor = 15.0;

    std::vector<std::pair<double, double>> escenarios = {
        {0.0, 0.0}, {45.0, -45.0}, {90.0, 0.0}
    };
```

```
60
61      std::cout << "--- REPORTE DE TORQUES ---" << std::endl;
62      std::cout << std::fixed << std::setprecision(2);
63
64      int caso = 0;
65      for (auto config : escenarios) {
66          ResultadoTorque res = controlador.calcularTorqueEstatico(
    carga_convertidor, config.first, config.second);
67
68          std::cout << "Caso " << (char)('A' + caso++) << ": "
69                    << "Hombro = " << res.torque_hombro << " Nm | "
70                    << "Codo = " << res.torque_codo << " Nm" << std::endl;
71      }
72      return 0;
73  }
```

Listing 1: Código de Control de Estática en C++

5. Conclusión

El análisis y desarrollo presentados en este reporte demuestran la simbiosis crítica entre la física estática y la ingeniería de software. Se ha evidenciado que el problema de la çaída.[en] brazos robóticos industriales no se resuelve únicamente con fuerza bruta mecánica, sino con inteligencia algorítmica.

La implementación del algoritmo de compensación de gravedad permite predecir con exactitud la fuerza necesaria para mantener el equilibrio estático. Esto tiene tres impactos fundamentales en la industria: **1) Eficiencia Energética**, ya que el motor solo consume la corriente exacta necesaria; **2) Precisión**, al eliminar el error de estado estable causado por la gravedad; y **3) Seguridad**, asegurando que la carga no colapse si el lazo de control PID falla momentáneamente.

6. Referencias Bibliográficas

Referencias

[1] Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). *Robot Modeling and Control* (2nd Edition). Wiley.

[2] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2010). *Robotics: Modelling, Planning and Control*. Springer Science & Business Media.

[3] Craig, J. J. (2017). *Introduction to Robotics: Mechanics and Control* (4th Edition). Pearson.

[4] Corke, P. (2017). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer.