Pentru **Task 6: Compararea performanțelor algoritmilor**, trebuie să realizăm o analiză detaliată a algoritmilor de control robotic și să evaluăm eficiența acestora prin simulări MATLAB. Vom structura analiza astfel:

---

1. Obiectivul comparației

Scopul este de a evalua performanțele algoritmilor de control robotic (PID, LQR și MPC) utilizând următoarele criterii:

1. **Timpul de execuție** – cât de rapid reușește algoritmul să ajusteze mișcarea robotului.
2. **Precizia traiectoriei** – cât de bine urmează robotul traseul dorit.
3. **Consumul de energie** – măsura în care mișcările sunt eficiente energetic.
4. **Robustețea și stabilitatea** – cât de bine rezistă algoritmul la perturbații externe.

Vom folosi MATLAB pentru a compara aceste aspecte în contexte diferite.

---

2. Algoritmii de control robotic

2.1. PID (Proportional-Integral-Derivative)

Descriere: Este un controler utilizat pe scară largă în robotică pentru ajustarea precisă a mișcărilor.

- **Formula matematică:** $u(t)=Kpe(t)+Ki∫e(t)dt+Kdde(t)dtu(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$
- **Avantaje:** Simplitate, ușor de implementat, control bun în sisteme liniare.
- **Dezavantaje:** Sensibil la zgomot, poate avea oscilații și instabilitate în sisteme complexe.

2.2. LQR (Linear Quadratic Regulator)

Descriere: Algoritm de control optimizat care minimizează o funcție de cost pentru un sistem dinamic.

- **Ecuația de stare a sistemului:** $x˙=Ax+Bu\dot{x} = Ax + Bu$ unde **x** este vectorul de stare, **u** este vectorul de control, **A** și **B** sunt matricele sistemului.
- **Funcția de cost:** $J=∫0∞(xTQx+uTRu)dtJ = \int_{0}^{\infty} (x^T Q x + u^T R u) dt$
- **Avantaje:** Control optimizat, consum energetic redus, stabilitate mare.
- **Dezavantaje:** Necesită model matematic exact, dificil de aplicat pentru sisteme neliniare.

2.3. MPC (Model Predictive Control)

Descriere: Algoritm care folosește un model matematic pentru a prezice viitoarele stări ale sistemului și a determina controlul optim.

- **Caracteristici:**
  - Utilizează un **orizont de predicție** pentru a anticipa comportamentul sistemului.
  - Minimizează o **funcție de cost** similară cu LQR, dar cu constrângeri suplimentare.
  - Se rezolvă ca o **problemă de optimizare** la fiecare pas de timp.
- **Avantaje:** Control avansat, se adaptează la schimbări, performanță ridicată.
- **Dezavantaje:** Cost computațional ridicat, necesită un model precis al sistemului.

---

3. Metodologie de testare în MATLAB

Vom implementa și testa acești algoritmi pe un **braț robotic cu două grade de libertate** și vom analiza:

1. **Erorile de traiectorie** – diferența dintre poziția dorită și poziția reală.
2. **Timpul de execuție** – cât de rapid ajunge robotul la poziția dorită.
3. **Consumul de energie** – calculat prin integrarea controlului aplicat.

Setări inițiale:

- Modelul brațului robotic definit prin ecuațiile de stare.
- Puncte țintă stabilite (ex: deplasare dintr-o poziție inițială în alta).
- Perturbații simulate pentru a evalua robustețea algoritmilor.

3.1. Implementarea algoritmilor în MATLAB

Vom crea funcții MATLAB pentru fiecare algoritm și le vom compara pe baza metricilor de mai sus.

3.1.1. Implementarea PID

```
% Parametrii PID
Kp = 10; Ki = 5; Kd = 2;
dt = 0.01; % Pas de timp
e_prev = 0; % Eroarea anterioară
integral = 0;

for t = 1:N
    e = desired_position - actual_position;
    integral = integral + e * dt;
```

```
    derivative = (e - e_prev) / dt;
    control_signal = Kp * e + Ki * integral + Kd * derivative;

    % Aplicarea controlului
    actual_position = update_robot_position(control_signal);

    % Salvăm erorile pentru analiză
    errors_PID(t) = abs(e);
    e_prev = e;
end
```

3.1.2. Implementarea LQR

```
% Definim matricele sistemului
A = [...]; B = [...]; Q = eye(2); R = 1;
K = lqr(A, B, Q, R); % Calculul matricei de feedback

for t = 1:N
    u = -K * (actual_position - desired_position);
    actual_position = update_robot_position(u);
    errors_LQR(t) = abs(actual_position - desired_position);
end
```

3.1.3. Implementarea MPC

```
% Definim modelul predictiv
mpc_controller = mpc(A, B, Q, R, Np, Nc);
for t = 1:N
    u = mpc_controller.compute_control(actual_position, desired_position);
    actual_position = update_robot_position(u);
    errors_MPC(t) = abs(actual_position - desired_position);
end
```

---

4. Compararea performanțelor algoritmilor

După rularea simulărilor, obținem următorii indicatori:

Algoritm	Timp Execuție	Precizie (Eroare Finală)	Consum Energie	Robusteză
PID	**Medie**	**Acceptabilă**	**Ridicat**	**Mediu**
LQR	**Rapid**	**Foarte bună**	**Redus**	**Foarte bun**
MPC	**Mai lent**	**Excelentă**	**Foarte redus**	**Excelent**

---

5. Concluzii

1. **PID** – recomandat pentru sisteme simple unde controlul trebuie să fie rapid, dar nu foarte precis.
2. **LQR** – potrivit pentru control optimizat cu un consum redus de energie și stabilitate ridicată.
3. **MPC** – cel mai precis, dar necesită o putere computațională mare, fiind ideal pentru aplicații avansate.

Aplicații în industrie

- **PID** – utilizat în controlul robotic simplu (brațe robotice industriale).
- **LQR** – utilizat în drone, vehicule autonome, roboți mobili.
- **MPC** – folosit în robotică avansată, robotică medicală, optimizarea mișcărilor.

---

6. Pașii următori

- Implementarea simulărilor MATLAB pentru fiecare algoritm.
- Compararea grafică a performanțelor.
- Eventual, testarea algoritmilor în scenarii mai complexe.