Optimum design

12th Optimization problem

Optimizing PID values for a transfer function

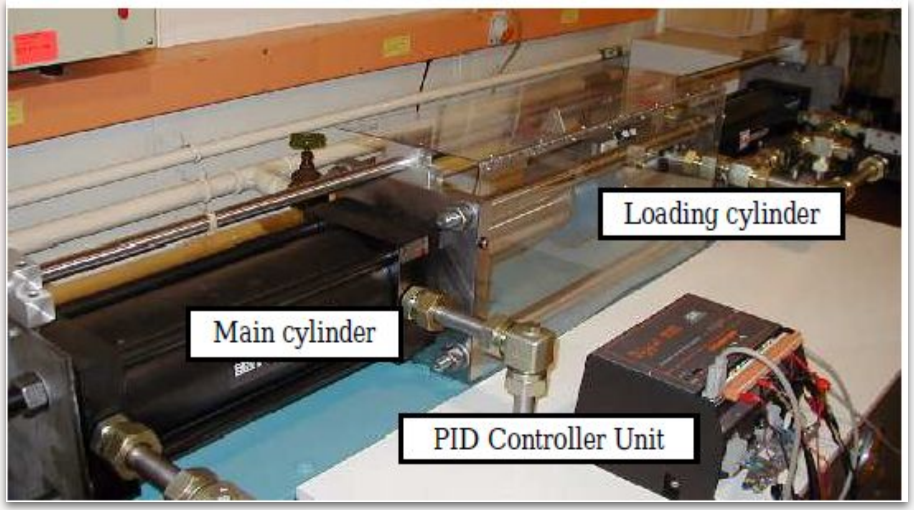
Created by:
Adham Eid 20107315
Farida Ahmed 20107458

Problem formulation

Project description

- Create a PID controller for a hydraulic position control system.
- Control a 3 way proportional valve that controls the main and loading cylinder
- Transfer function is derived from the physical system

$$G_p(s) = \frac{7.84}{3s^2 + 5.04s + 7.84}$$



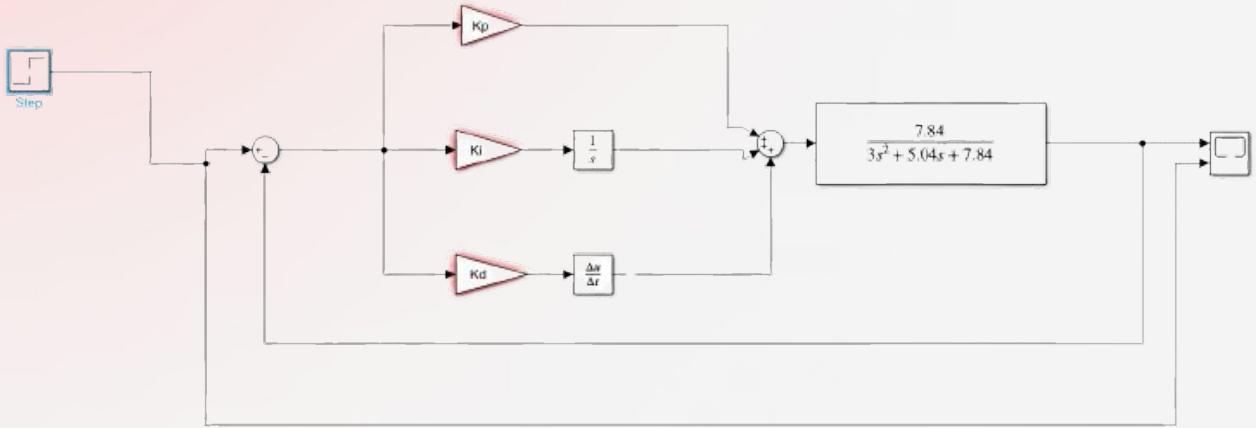
Problem formulation

Data and information collection

The following equations were used:

$$G_p(s) = \frac{7.84}{3s^2 + 5.04s + 7.84} \quad (1)$$

$$PID(s) = K_p + \frac{K_i}{s} + K_d s \quad (2)$$



Problem formulation

Design Variables

01	Kp = Proportional gain
02	Ki = Integral gain
03	Kd = Derivative component

Problem formulation

Optimization Criterion

- Any overshoot will cause damage to the system

$$f(K_p, K_i, K_d) = Minimize\ (Overshoot)$$

Problem formulation

Formulation of constraints

01

Steady-state error ≤ 5%

Justification:

- Response needs to settle near required value for the system to work correctly
- Improve repeatability

02

Settling time ≥ 1 Second

Justification:

- The system needs to reach the desired output within 1 second and cannot exceed 1.5 seconds

03

Settling time ≤ 1.5 Seconds

Justification:

- The system needs to reach the desired output within 1 second and cannot exceed 1.5 seconds

04

Rise time ≥ 0.5 Seconds

Justification:

- The system cannot be any faster than this

Method 1 (Adham)

Genetic Algorithm Optimization

Genetic Algorithm Optimization

- Tries to mimic natural selection and evolution
- Used on constrained and unconstrained problems
- Find global minimum or maximum

Genetic Algorithm

Process

Genetic Algorithm Optimization **Advantages**

- Runs effectively on high cpu core count machines
- Not affected by starting position
- Effective in searching in large solution space
- No derivative information needed

Genetic Algorithm Optimization **Limitations**

- Long convergence times
- Premature convergence (Random)
- Won't work well with complex fitness functions

Genetic Algorithm Optimization **For this problem**

Still suitable because:

- Fitness function is simple
- Small search area
- Finds a solution in an acceptable time

Genetic Algorithm

Settings

The following settings were used:

- A population size of 800
- Max generations of 300
- Elite count of 8,
- Crossover fraction of 0.8
- Mutation and Parallel processing were turned on

The following Toolbox in matlab were used:

- Global optimization toolbox
  - For genetic algorithm
- Parallel computing toolbox
  - For using parallel feature in GA

Genetic Algorithm

Results

- 4 Generations to converge
- **269.33** seconds to find the solution
  - With parallel computing enabled

Generation	Func-count	Best f(x)	Max Constraint	Stall Generations
1	42015	0.024591	0	0
2	83230	0.0244363	0	0
3	124445	0.0237628	0	0
4	165645	0	0.001	0

Genetic Algorithm

Results

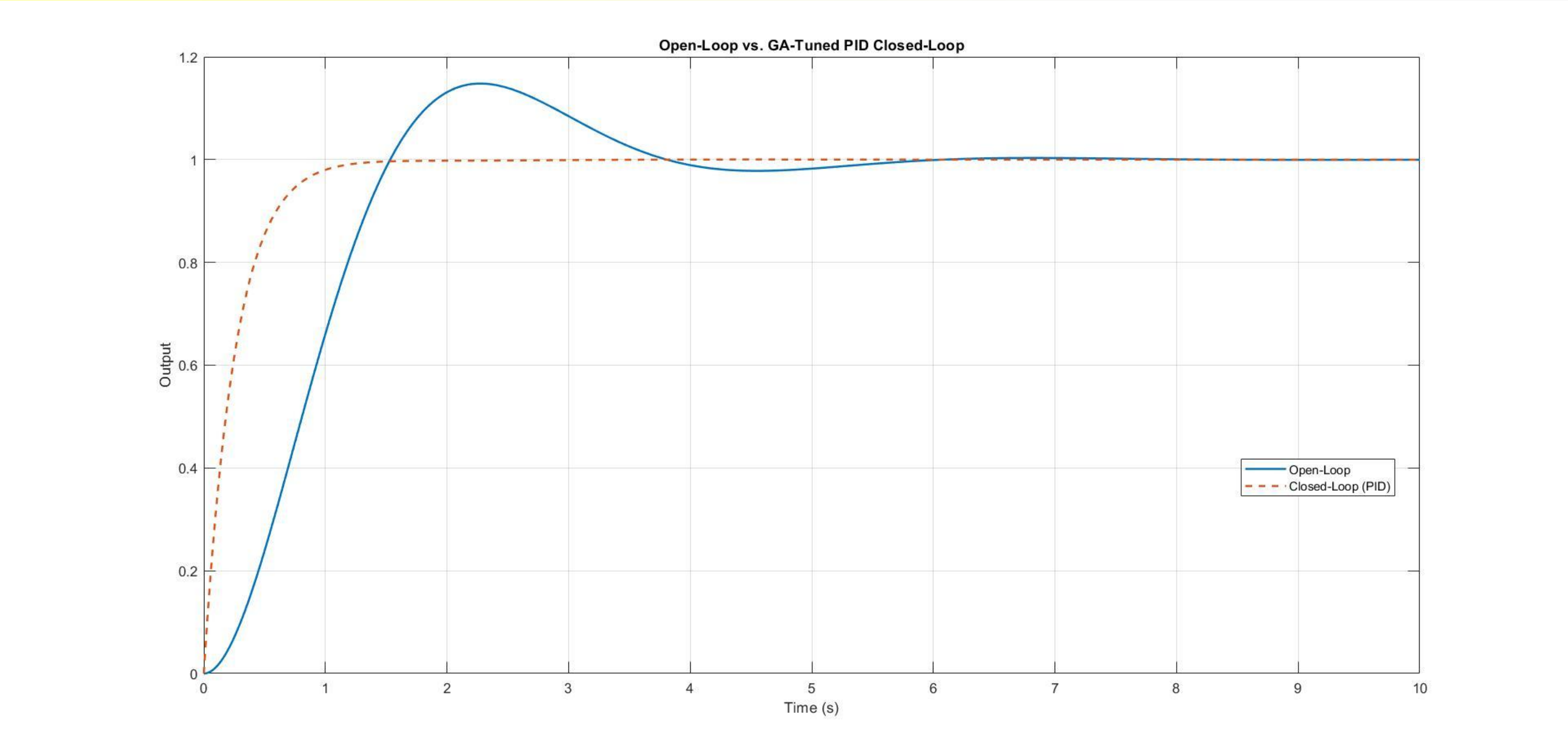
The solution from the genetic algorithm is:

- Kp = 2.4956
- Ki = 3.8180
- Kd = 1.4552

Parameters	Open-Loop(no PID)	Closed-Loop (with PID)
Rise time	1.0383 s	0.581 s
Settling time	4.8291 s	0.9990 s
Overshoot	14.79%	0.00%
Steady-State error	0.50%	0.00%

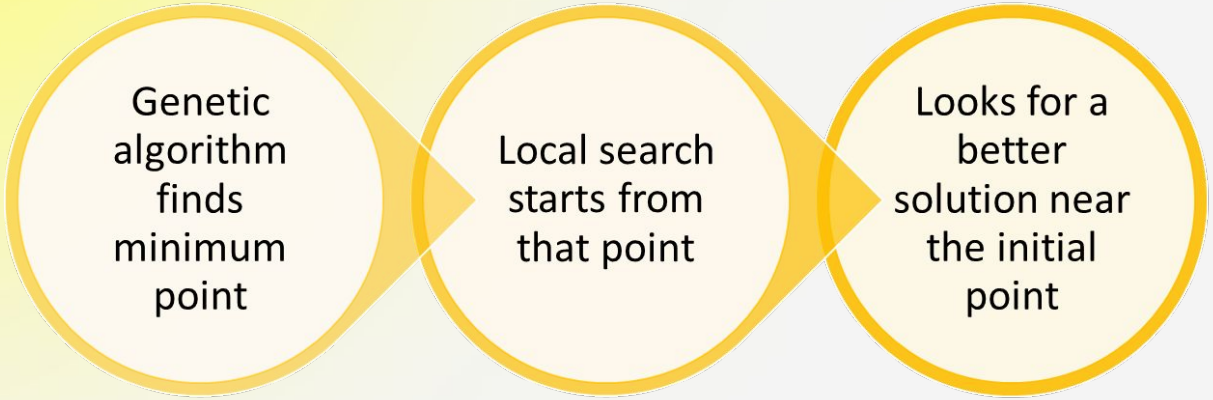
Genetic Algorithm

Results



Hybrid

Genetic Algorithm + Local search



Hybrid

Results

Parameters	Open-Loop(no PID)	Closed-Loop (with PID) (GA)	Closed-Loop (with PID) (Hybrid)
Rise time	1.0383 s	0.6309 s	0.6310 s
Settling time	4.8291 s	0.9990s	1.0000 s
Overshoot	14.79%	0.00%	0.00%
Steady-State error	0.50%	0.00%	0.00%

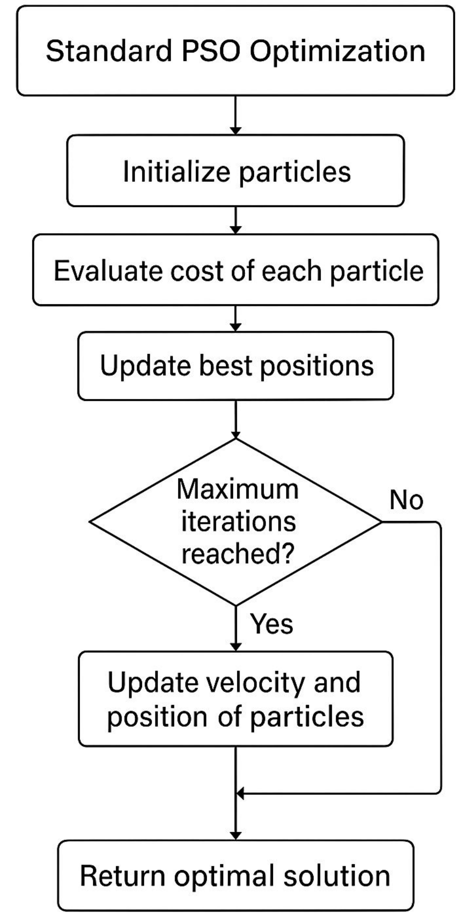
Method 2 (Farida)

Particle swarm Optimization

Particle swarm Optimization

- group of particles moving through a solution space to find the optimum of an objective function.
- balances exploration and exploitation,
- Useful for solving nonlinear, multidimensional optimization problems

Particle swarm Optimization



Particle swarm Optimization Advantages

- Simple to implement
- Fewer parameters to adjust compared to algorithms like Genetic Algorithms
- Requires only a few control parameters
- Often converges faster than other optimization algorithms
- Applicable to a wide range of optimization problems
- Supports parallelization, enhancing computation speed for large problems

Particle swarm Optimization limitations

- stuck in local minima, especially in complex or multimodal search spaces
- Performance depends heavily on tuning of inertia weight and learning factors
- Poor parameter settings may cause instability or poor optimization results
- Scalability issues can reduce performance as problem dimensionality increases
- Lack of diversity may limit exploration and reduce ability to find global optima
- No guaranteed convergence, especially without mechanisms to escape local optima

Particle swarm Optimization
For this problem

- Handles nonlinear and complex systems
- No need for mathematical linearization or model derivatives
- Derivative-free, ideal for black-box system optimization
- Easily customizable for multi-objective optimization
- Global optimization capability, better performance in noisy systems
- Effective for systems with time delays
- Does not rely on system time constants or frequency responses
- Flexible and efficient for optimizing hydraulic systems

Particle swarm Optimization

Settings

The following settings were used:

- Number of particles of 30
- Max iterations of 100
- Inertia weight that decays by 0.99 per iteration
- Cognitive coefficient (self-confidence) of 1.5
- Social coefficient (swarm confidence) of 1.5

The following Toolbox in matlab were used:

- Global optimization toolbox
  - For Particle swarm Optimization

Particle swarm Optimization

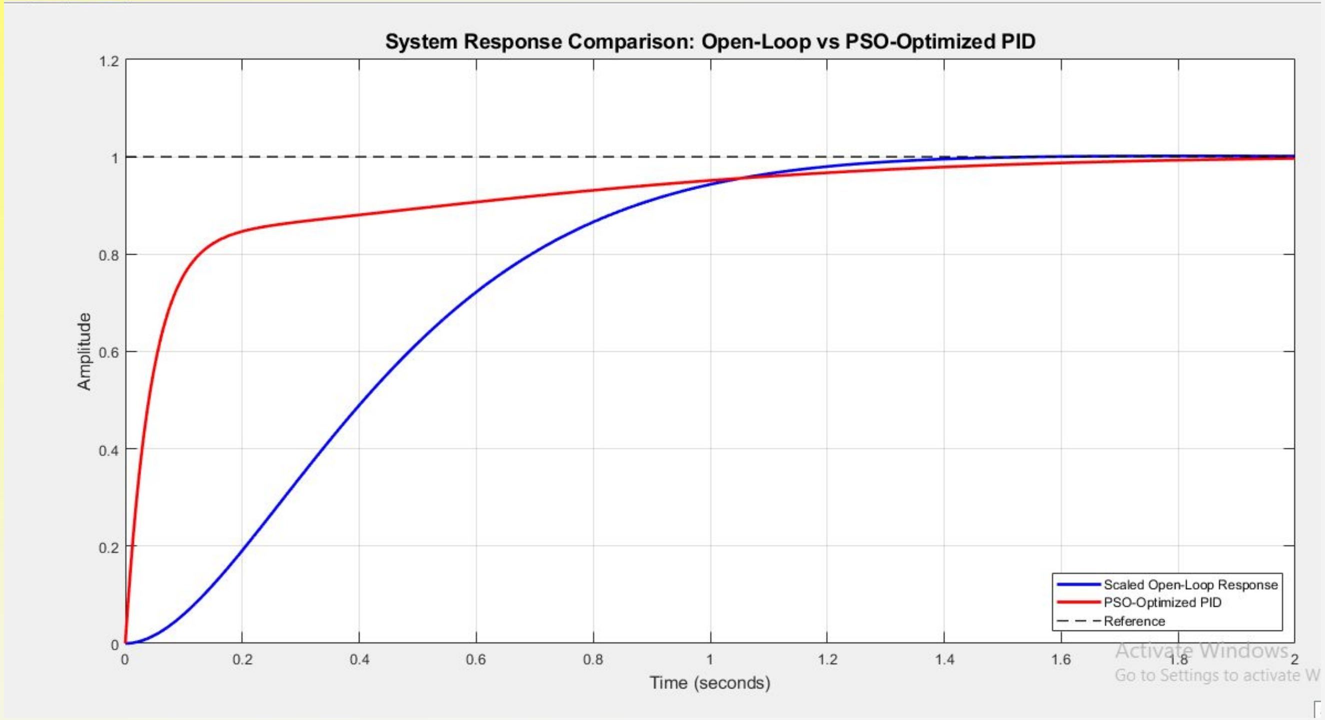
The output for the PSO optimization was as follows:

- Kp = 1.124
- Ki = 1.572,
- Kd = 0.318

parameters	Open loop (no PID)	Standard PSO
Rise time:	1.0407s	0.5404s
Settling time:	4.9542s	1.4189s
Overshoot:	14.58%	0.00%
Steady state error (ess):	0.18%	0.34%

Particle swarm Optimization

Results



Particle swarm Optimization + Simulated Annealing Method

Settings

The following settings were used:

- Number of particles of 30
- Max iterations of 100
- Inertia weight that decays by 0.99 per iteration
- Cognitive coefficient (self-confidence) of 1.5
- Social coefficient (swarm confidence) of 1.5
- Cooling rate of 0.95
- the Number of SA iterations per PSO iteration 5

The following Toolbox in matlab were used:

- Global optimization toolbox
  - For Particle swarm Optimization

Particle swarm Optimization + Simulated Annealing Method

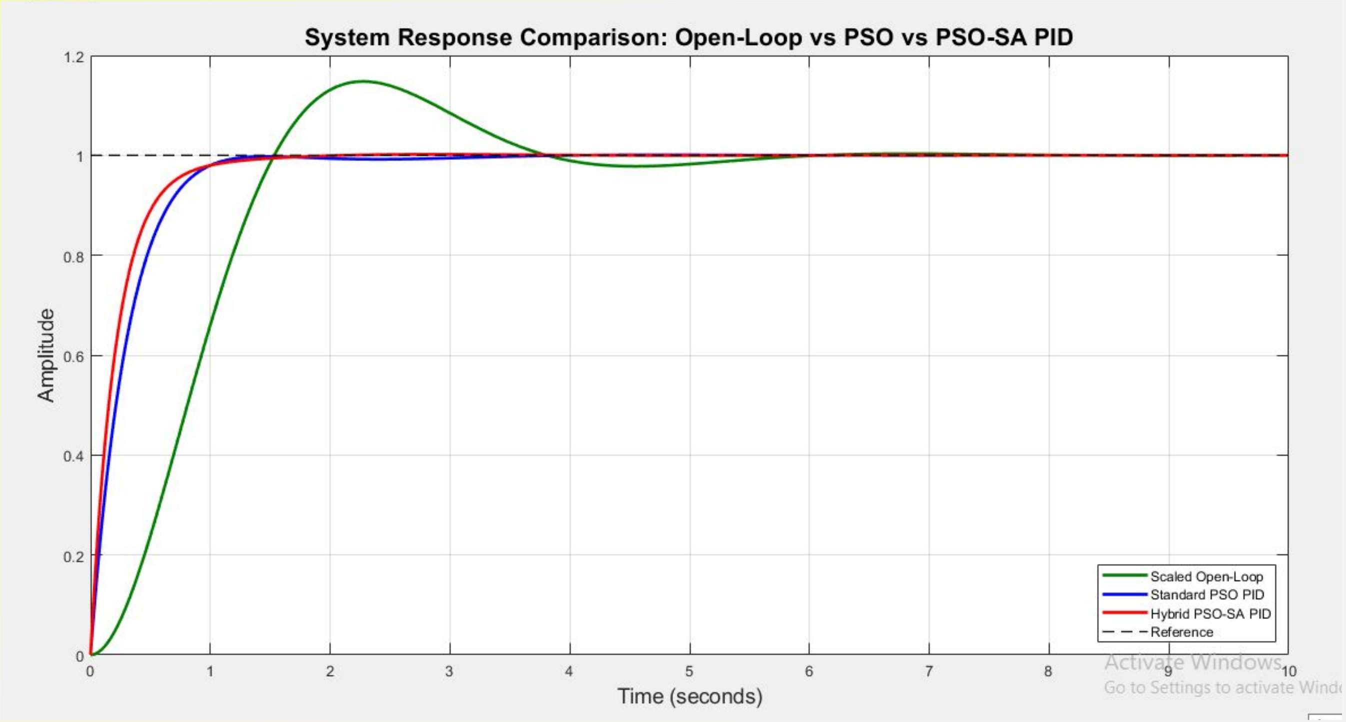
The output for the PSO -SA optimization was as follows:

- Kp = 2.8309,
- Ki = 4.3907
- Kd = 1.7615

parameters	Open loop (no PID)	Standard PSO	Hybrid PSO-SA
Rise time:	1.0407s	0.6190s	0.5003s
Settling time:	4.9542s	1.0000s	1.0006s
Overshoot:	14.58%	0.00%	0.00%
Steady state error (ess):	0.18%	0.25%	0.34%

Particle swarm Optimization + Simulated Annealing Method

Results



Method comparison

Results

- GA is slightly better (No overshoot)
- PSO + SA is better than PSO alone

PID values	GA	GA+LS	PSO	PSO+SA
Kp	2.4956	2.2386	1.124	2.8309
Ki	3.818	3.2746	1.572	4.3907
Kd	1.4552	1.1735	0.318	1.7615

Parameters	Open-Loop(no PID)	Closed-Loop (with PID) (GA)	Closed-Loop (with PID) (GA+LS)	Closed-Loop (with PID) (PSO)	Closed-Loop (with PID) (PSO+SA)
Rise time	1.0383 s	0.6309 s	0.6310 s	0.5404 s	0.5003 s
Settling time	4.8291 s	0.9990s	1.0000 s	1.4189 s	1.0006 s
Overshoot	14.79%	0.00%	0.00%	0.00%	0.00%
Steady-State error	0.50%	0.00%	0.00%	0.34%	0.34%

Github link

Thank you