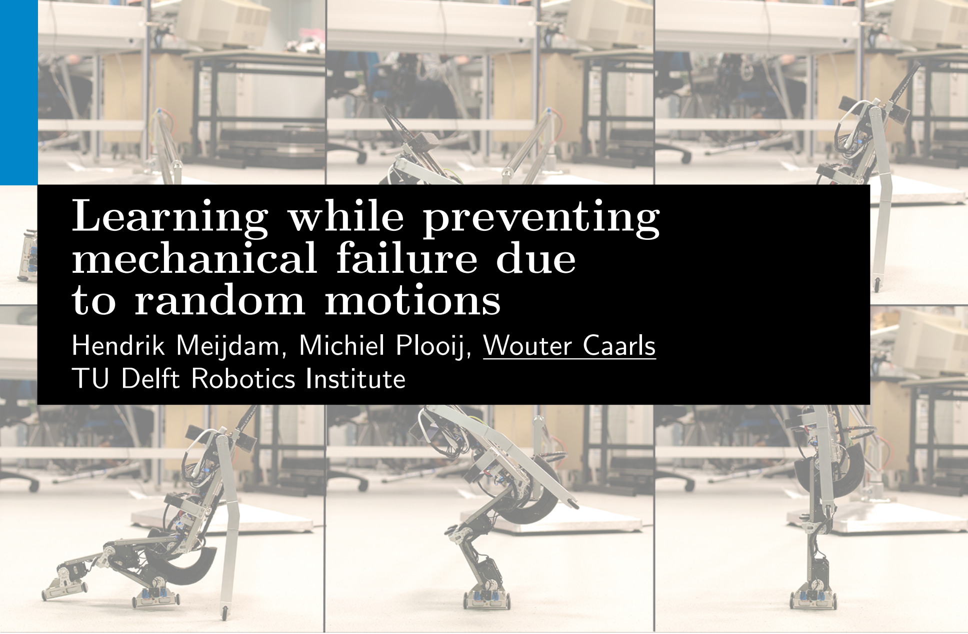


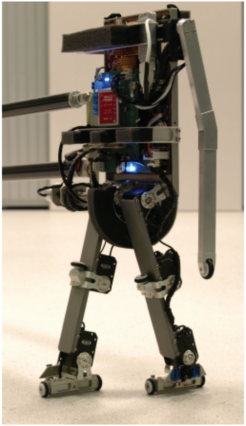
Learning while preventing mechanical failure due to random motions

Hendrik Meijdam, Michiel Plooij, Wouter Caarls
TU Delft Robotics Institute

TU Delft

Reinforcement learning for robots

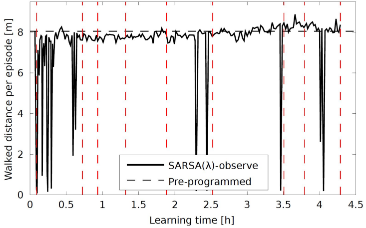
- Robot LEO
  - 50cm tall, 1.7kg
  - 7 Dynamixel servos
  - Connected to a boom (2d)
- Learning to walk
  - SARSA(λ)
  - Start by observing known controller
  - Optimize over 4 hours.
- Gearboxes break every 30 minutes!



TUDelft

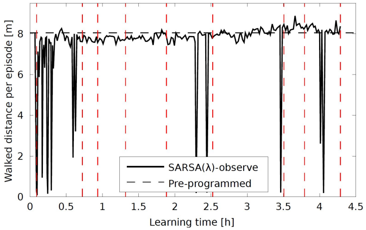
Reinforcement learning for robots

- Robot LEO
  - 50cm tall, 1.7kg
  - 7 Dynamixel servos
  - Connected to a boom (2d)
- Learning to walk
  - SARSA(λ)
  - Start by observing known controller
  - Optimize over 4 hours.
- Gearboxes break every 30 minutes!



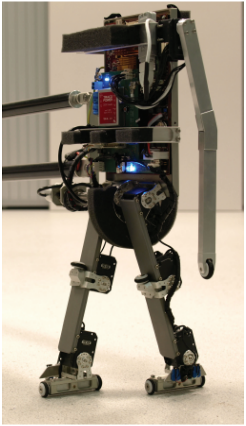
Reinforcement learning for robots

- Robot LEO
  - 50cm tall, 1.7kg
  - 7 Dynamixel servos
  - Connected to a boom (2d)
- Learning to walk
  - SARSA(λ)
  - Start by observing known controller
  - Optimize over 4 hours.
- Gearboxes break every 30 minutes!



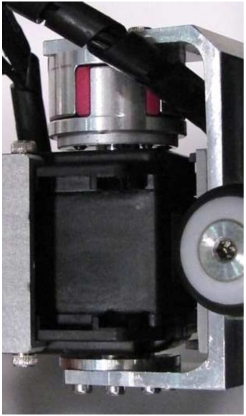
Failure causes

- Falling
  - Foam padding
  - Switch off power to motors
- Stepping
  - Unavoidable
  - Elastic joint elements
- Random motions
  - Caused by exploration
  - Elastic elements help, but not enough
  - Less problematic in policy search



Failure causes

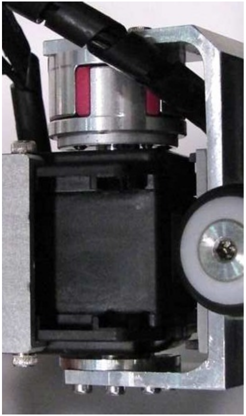
- Falling
  - Foam padding
  - Switch off power to motors
- Stepping
  - Unavoidable
  - Elastic joint elements
- Random motions
  - Caused by exploration
  - Elastic elements help, but not enough
  - Less problematic in policy search



TUDelft

Failure causes

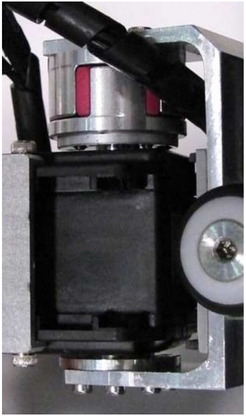
- Falling
  - Foam padding
  - Switch off power to motors
- Stepping
  - Unavoidable
  - Elastic joint elements
- Random motions
  - Caused by exploration
  - Elastic elements help, but not enough
  - Less problematic in policy search



TUDelft

Failure causes

- Falling
  - Foam padding
  - Switch off power to motors
- Stepping
  - Unavoidable
  - Elastic joint elements
- Random motions
  - Caused by exploration
  - Elastic elements help, but not enough
  - ~~Less problematic in policy search~~



TUDelft

Preventing failure due to random motions

- Model the damage due to backlash re-engagement
- Investigate different action filtering algorithms
- Compare the performance on a simulation of LEO

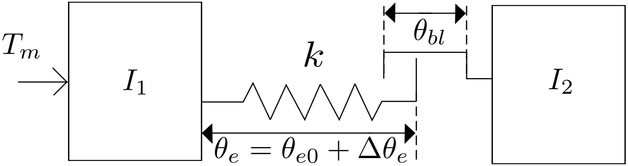
Preventing failure due to random motions

- Model the damage due to backlash re-engagement
- Investigate different action filtering algorithms
- Compare the performance on a simulation of LEO

Preventing failure due to random motions

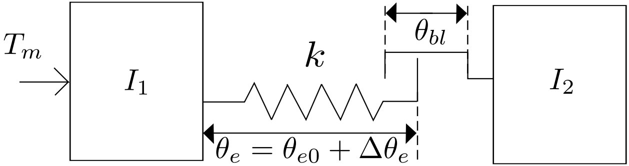
- Model the damage due to backlash re-engagement
- Investigate different action filtering algorithms
- Compare the performance on a simulation of LEO

Calculating the MTBF



- The mean time between failure (MTBF) is predicted based on material fatigue during backlash re-engagements
- The maximum stress in the gears is a function of the torque at which the gears re-engage
- MTBF is a function of torque and number of re-engagements

Calculating the MTBF

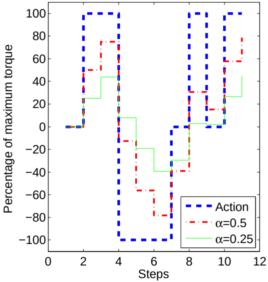


- The mean time between failure (MTBF) is predicted based on material fatigue during backlash re-engagements
- The maximum stress in the gears is a function of the torque at which the gears re-engage
- MTBF is a function of torque and number of re-engagements

Low-pass filtering

- Filter the actions with a discrete first-order filter

$$a_{filtered_t} = \alpha \cdot a_t + (1 - \alpha) \cdot a_{filtered_{t-1}}$$

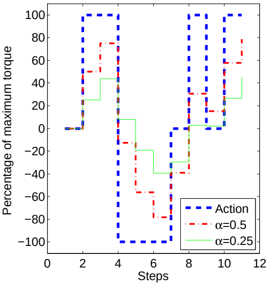


- The Markov property can be preserved by adding $a_{filtered_{k-1}}$ to the state

Low-pass filtering

- Filter the actions with a discrete first-order filter

$$a_{filtered_t} = \alpha \cdot a_t + (1 - \alpha) \cdot a_{filtered_{t-1}}$$

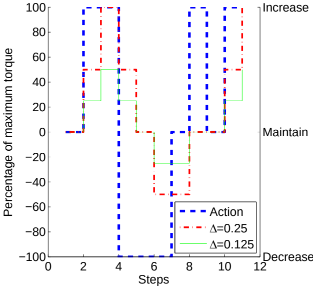


- The Markov property can be preserved by adding $a_{filtered_{k-1}}$ to the state

Integrating controller

- Use relative actions instead of absolute actions

$$a_{filtered_t} = a_{filtered_t} + a_t, a_t \in \{-\Delta, 0, +\Delta\}$$

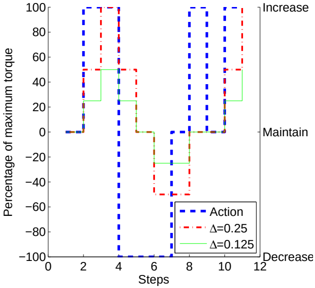


- The Markov property can be preserved by adding a_{t-1} to the state

Integrating controller

- Use relative actions instead of absolute actions

$$a_{filtered_t} = a_{filtered_t} + a_t, a_t \in \{-\Delta, 0, +\Delta\}$$



- The Markov property can be preserved by adding a_{t-1} to the state

Previous action dependent actions

- Use absolute actions, but only allow those near the previous action

$$a_t \in \{a_{t-1} - \Delta, a_{t-1}, a_{t-1} + \Delta\}$$

- Does not violate the Markov property when using a state-action value function

$$Q(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q(s', a') \right]$$

Previous action dependent actions

- Use absolute actions, but only allow those near the previous action

$$a_t \in \{a_{t-1} - \Delta, a_{t-1}, a_{t-1} + \Delta\}$$

- Does not violate the Markov property when using a state-action value function

$$Q(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q(s', a') \right]$$

Previous action dependent actions

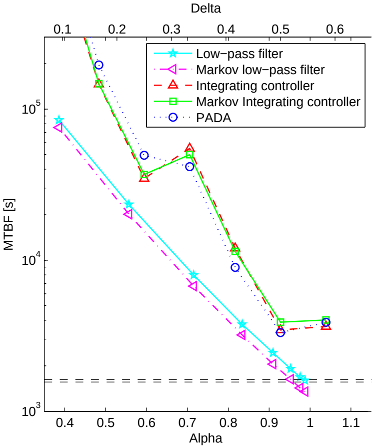
- Use absolute actions, but only allow those near the previous action

$$a_t \in \{a_{t-1} - \Delta, a_{t-1}, a_{t-1} + \Delta\}$$

- Does not violate the Markov property when using a state-action value function

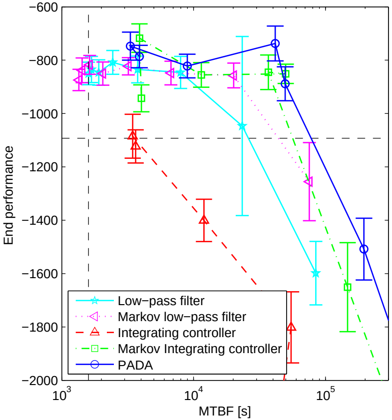
$$Q(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a' \in f(a)} Q(s', a') \right]$$

Predicted MTBF



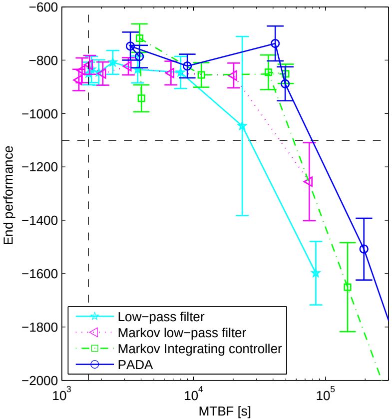
Increased filtering increases MTBF

End performance on pendulum swing-up

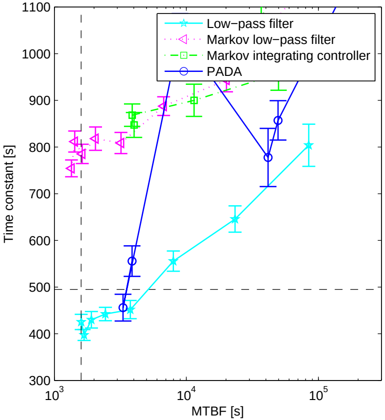


Integrating controller can't reach original end performance

End performance on pendulum swing-up

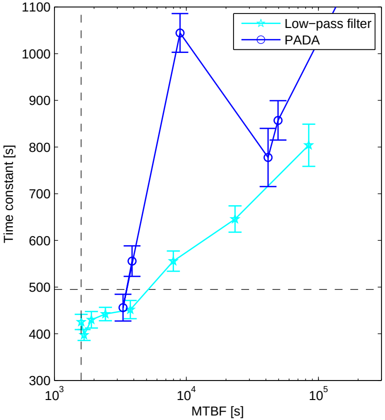


Time constant on pendulum swing-up

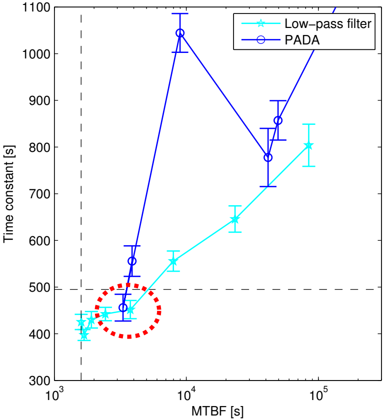


Algorithms with extra state can't reach original time constant

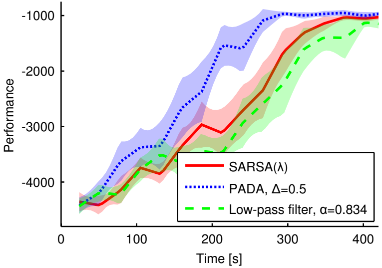
Time constant on pendulum swing-up



Time constant on pendulum swing-up

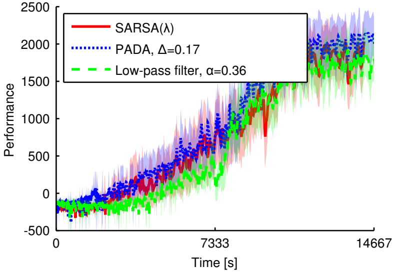


Physical pendulum



Low-pass filter and PADA are at least as good as SARSA(λ) while increasing the predicted MTBF by a factor 2

LEO simulation



- PADA is just as good as SARSA(λ) while increasing the predicted MTBF by a factor 108
- Low-pass filter is slightly worse at the same factor

Summary

- Reinforcement learning on robots is hampered by damage
- The damage due to random motions of TD control algorithms can be successfully mitigated by action filtering
- The low-pass filter and the PADA algorithm both increase the predicted MTBF without negatively affecting the learning process
- The PADA algorithm does not violate the Markov property and gives slightly better results

TU Delft

Summary

- Reinforcement learning on robots is hampered by damage
- The damage due to random motions of TD control algorithms can be successfully mitigated by action filtering
- The low-pass filter and the PADA algorithm both increase the predicted MTBF without negatively affecting the learning process
- The PADA algorithm does not violate the Markov property and gives slightly better results

TUDelft

Summary

- Reinforcement learning on robots is hampered by damage
- The damage due to random motions of TD control algorithms can be successfully mitigated by action filtering
- The low-pass filter and the PADA algorithm both increase the predicted MTBF without negatively affecting the learning process
- The PADA algorithm does not violate the Markov property and gives slightly better results

TUDelft

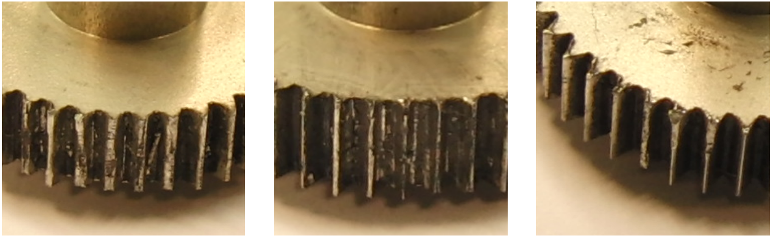
Summary

- Reinforcement learning on robots is hampered by damage
- The damage due to random motions of TD control algorithms can be successfully mitigated by action filtering
- The low-pass filter and the PADA algorithm both increase the predicted MTBF without negatively affecting the learning process
- The PADA algorithm does not violate the Markov property and gives slightly better results

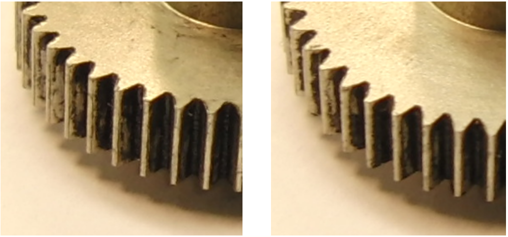
TUDelft

Questions?

Gears

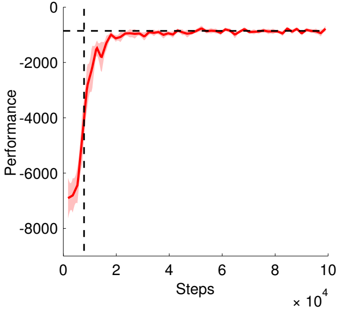


Unrestricted random motions



Restricted random motions

Performance measurement



- End performance is averaged over last 10% of test trials
- Time constant is time taken to rise to 95% of the relative end performance