# ADP (Adaptive Dynamics Planning) – Paper Outline

Printable reference for future discussions and rapid onboarding.

## 1. Problem and Motivation

- Mobile robot navigation often splits planning into global (no dynamics) and local (full dynamics), creating feasibility gaps.

- Fixed dynamics parameters in local planners degrade performance across environments with different constraint profiles.

- Decremental Dynamics Planning (DDP) improves feasibility by reducing dynamics fidelity along the rollout, but relies on hand-crafted rules.

- Need: a principled, adaptive mechanism to tune dynamics fidelity and parameters online to balance safety, efficiency, and compute.

## 2. Key Idea

ADP learns to adapt dynamics modeling during planning. Instead of issuing low-level controls, a learned agent serves as a meta-controller that tunes dynamics-related parameters of classical planners conditioned on observations.

## 3. Contributions

- Formulate Adaptive Dynamics Planning (ADP), a learning-augmented planning paradigm that adapts dynamics fidelity and parameters online.

- Instantiate ADP by coupling a reinforcement learning agent with classical planners (e.g., DWA, DDP, MPPI) via a standardized parameter interface.

- Design reward shaping that aligns safety (no collision), progress-to-goal, smoothness, and traversal time.

- Demonstrate improved success and robustness over fixed-parameter planners and hand-crafted DDP in BARN and real-world Jackal tests.

- Provide analysis on interpretability by inspecting learned parameter schedules and their correlation with environment difficulty.

## 4. Method Overview

### Architecture

- Classical planner executes trajectories; RL agent outputs a parameter vector that modulates dynamics modeling.

- State: laser scans plus robot states (pose/vel).

- Action: a K-dimensional continuous vector of dynamics parameters (e.g., integration step size, number of collision points, speed/acceleration limits, cost weights).

- Observation-to-action cadence synchronized with planner cycles; parameters can vary along the rollout horizon.

### Reward

- Positive: progress toward goal, smooth motion.

- Negative: collisions, near-collision penalties, excessive time, oscillations.

- Optionally include compute-aware regularization to discourage overly fine fidelity when unnecessary.

### Training and Execution

- Train with off-policy RL (e.g., TD3). In deployment, the planner stays in charge of control; the agent only tunes parameters.

- No end-to-end control; preserves stability and interpretability of classical methods.

## 5. Relation to Prior Work

### Classical Planning

- Global: A*, Dijkstra, PRM, RRT* (typically ignore dynamics for tractability).

- Local: DWA, TEB, MPC (model dynamics but rely on fixed, hand-tuned parameters).

### Dynamics Modeling and DDP

- High-fidelity models increase accuracy but are computationally expensive.

- DDP reduces fidelity along the rollout to balance compute and feasibility, yet uses manual decay schedules.

### Learning-Augmented Planning

- End-to-end IL/RL can learn complex behaviors but may lack stability and safety guarantees.

- ADP positions learning as a meta-controller to augment, not replace, classical planning.

## 6. Experimental Plan

- Environments: BARN 300 simulated worlds; additional hidden test sets if available; real-world Jackal setups.

- Baselines: fixed-parameter DWA/DDP/MPPI; hand-crafted DDP; ablated ADP variants.

- Metrics: success rate, collision rate, traversal time, smoothness, and score functions used in BARN.

- Ablations: parameter dimensionality; update frequency; reward components; per-planner interfaces; generalization across maps and speeds.

## 7. Implementation Notes (brief)

- Maintain a clean ROS topic interface (e.g., /dy_dt) for parameter vectors; keep planners in C++ for real-time performance.

- Normalize observations; clip and scale actions to safe parameter ranges; add safety fallbacks (e.g., recovery behaviors).

- Log all parameter schedules for analysis and reproducibility.

## 8. Expected Results and Hypotheses

- ADP improves success and reduces collisions vs. fixed parameters and manual DDP.

- Learned schedules correlate with environment difficulty (e.g., narrower passages trigger finer fidelity near-term).

- Gains persist under domain shift (unseen maps, sensor noise) due to adaptive parameterization.

## 9. Broader Impact and Limitations

- Impact: practical navigation with improved safety-efficiency trade-offs; interpretable learning signals via parameter traces.

- Limitations: reward misspecification risk; dependence on accurate state estimation; added system complexity.

## 10. Writing Plan (Section Checklist)

- Introduction: motivation, gap, contributions.

- Related Work: classical planning, dynamics modeling+DDP, learning-augmented planning.

- Method: ADP architecture, parameter interface, reward, training.

- Experiments: setups, metrics, baselines, ablations; real-world results.

- Analysis: interpretability of learned parameter schedules; failure cases.

- Conclusion: summary, limitations, future directions.