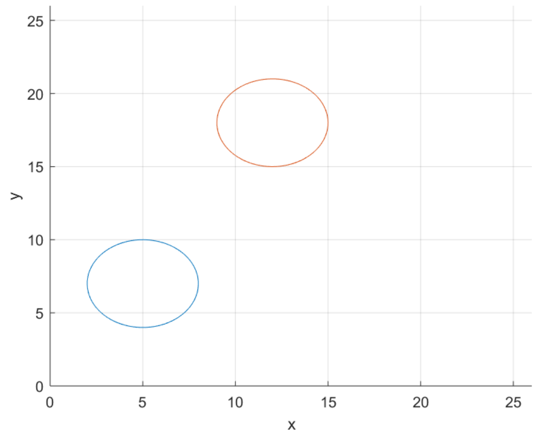
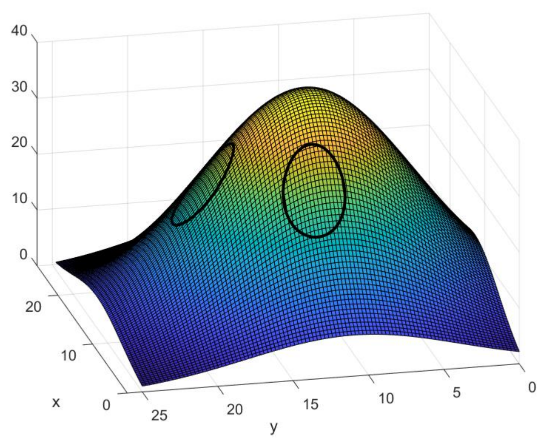
Path Planning using Constrained Differential Dynamic Programming

Gael Colas, Ianis Bougdal-Lambert

6/8/18

Problem Statement





- Simple model:
  - Given a 2D map with non-linear obstacles, find the shortest path from A to B that avoids obstacles.
  - Obstacles can be nonlinear (e.g. circles).

- More complex model:
  - The map is now a 3D level surface.
  - The cost of traversing a point is proportional to its elevation.
  - Find the shortest path that also minimizes the total effort along the path.

State-of-the-art

- For simple path finding: A* very efficient.
- A* can be adapted to take into account the space-varying cost by adding a cost to each node.

- For optimal control problems, many options: Direct Methods, Indirect Methods, Differential Dynamic Programming…
- Indirect Methods are very accurate but difficult to set up and do not handle well inequality constraints.

- Differential Dynamic Programming is widely used.
- Can handle nonlinear cost function.

Differential Dynamic Programming

- Locally approximates the dynamics and cost functions by quadratic models
- Model:

$$x_{k+1} = f(x_k, u_k)$$
$$J(X, U) = \sum_{k=0}^{N-1} l(x_k, u_k) + l_f(x_N)$$

- DP:

$$V_k(x) = \min_U l(x, u) + V_{k+1}\big(f(x, u)\big)$$
$$V_N(x) = l_f(x)$$

- Start with a feasible nominal trajectory and iterate:
  - › Backward pass to approximate value function as quadratic function
  - › Forward pass to produce a new nominal trajectory based on approximation computed before

Constrained DDP

- Need to handle inequality constraints on state and control

$$g_k(x, u) \leq 0$$

- The Backwards pass is modified to take into account active constraints at each point along the path using sensitivity analysis
- Solve QP problem to find quadratic approximation of optimal control along the path
- Active constraints are only *estimated*

- Backwards pass does not ensure feasibility of updated trajectory
- During Forward pass, ensure that updated trajectory is actually feasible and reduces objective function

- Need to regularize: parameters to keep trajectory close to nominal one are updated depending on success or failure

Modelization

- States

$$x_k = [x, y]^T$$

- Controls

$$u_k = [\dot{x}, \dot{y}]^T$$

- Minimizing the length of the path is equivalent to minimizing the norm of the velocity, i.e. the controls:

$$l(x, u) = u^T R u$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

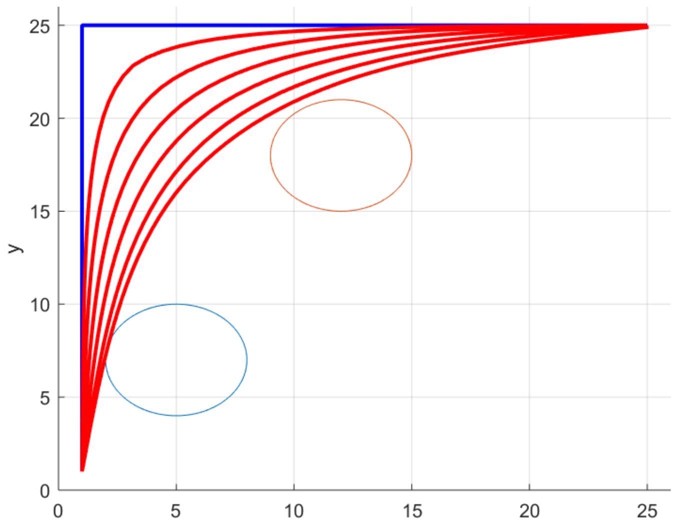
- For each obstacle (circles):

$$g_i(x, u) = r_i^2 - \left(x - x_{i,c}\right)^2 - \left(y - y_{i,c}\right)^2 \leq 0$$

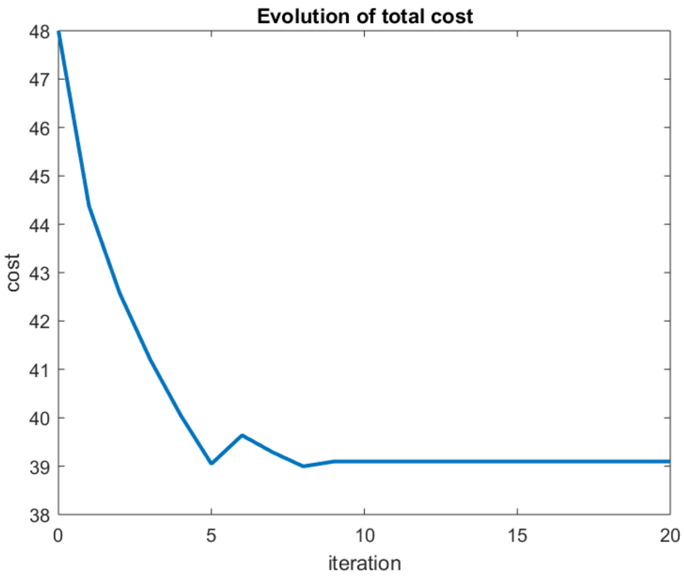
- Complex case:

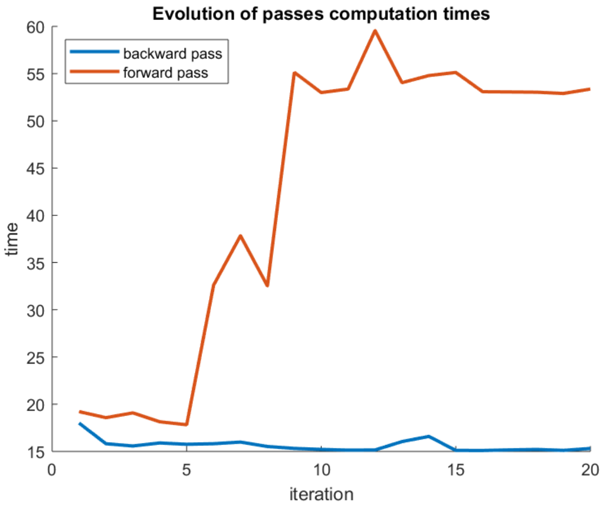
$$l(x, u) = u^T R u \; + \alpha \times elev(x)$$

Results – Simple Model

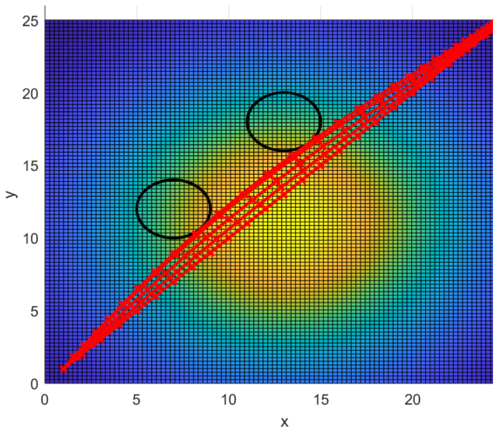


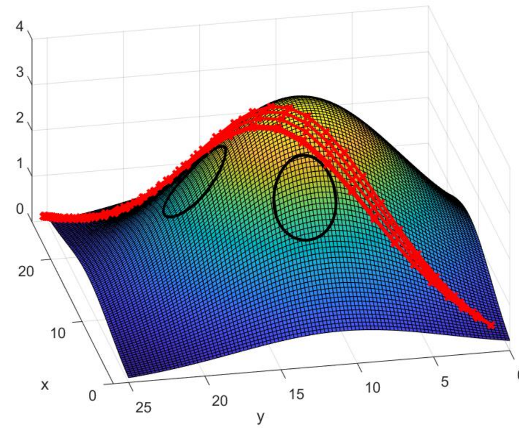
- Slow Convergence (QP solving)
  - 1 min per iteration
  - 10 min to convergence
- Depends on initialization
  - Cannot “jump” an obstacle
- Very sensitive to parameters
  - Trust region updating

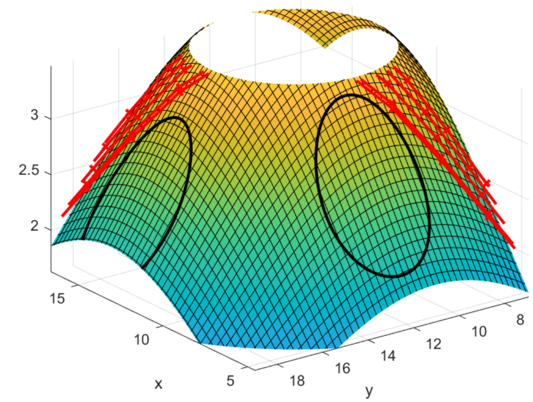


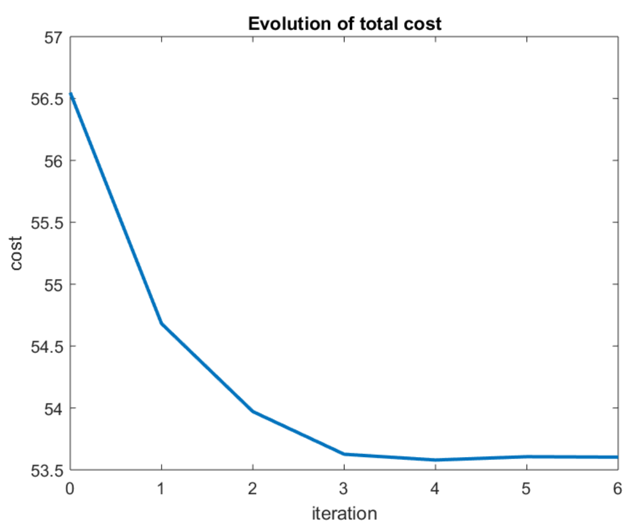


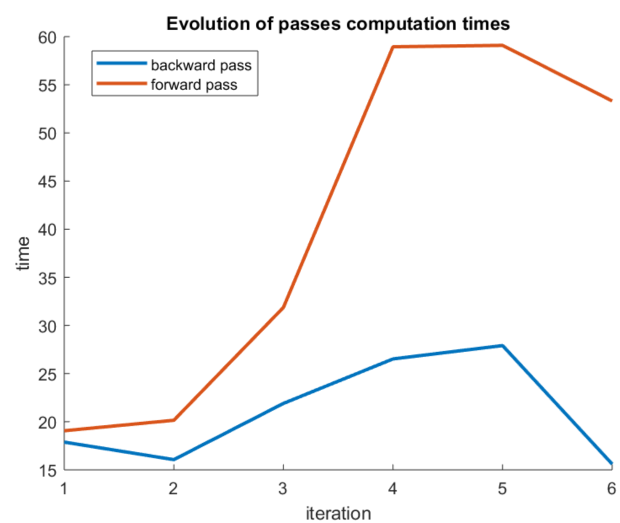
Results – Complex Model



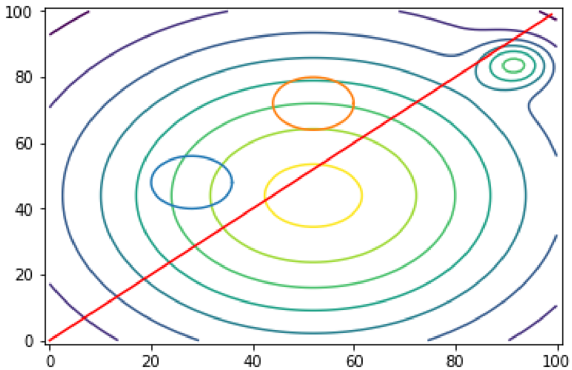




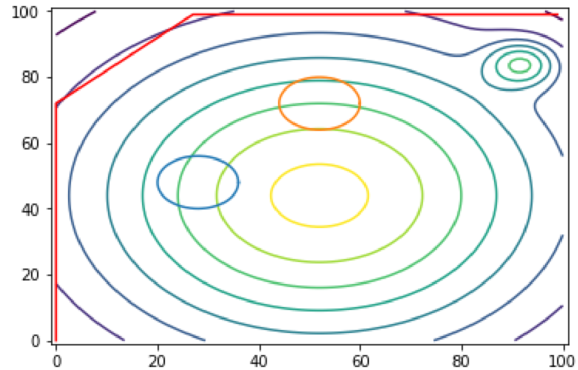




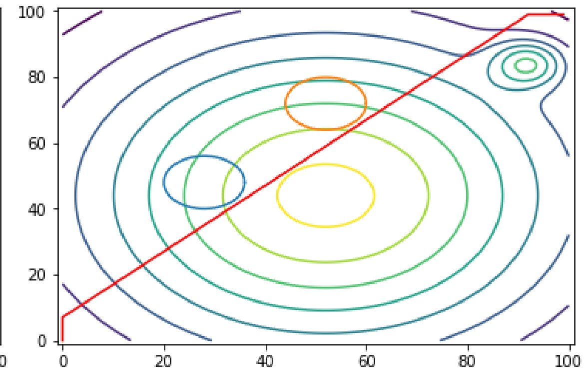
Comparison with A*



Length only

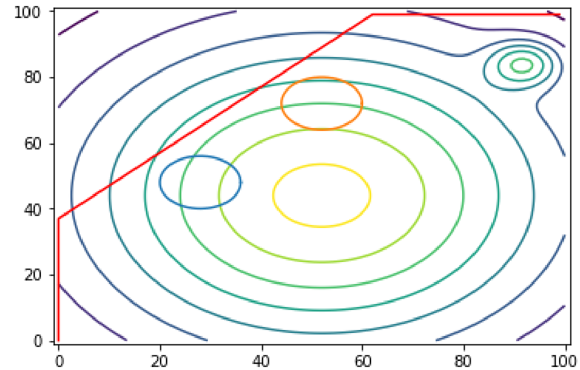


elevation only



Length + 1.12*elevation

- Naive A* can be adapted to any type of running cost $l(x)$
- Very fast and robust
  - 15s
- No need for initialization
  - Always finds optimal path
- No parameters to tune
- Can “jump” obstacles



Length + 1.13*elevation

Conclusion

- C-DDP is an interesting improvement to DDP
- Tailored for motion planning
- Needs a lot of tuning and an initial, feasible, suboptimal path
  - Could use a fast method to initialize the path (RRT)
- A* much more efficient
- Probably more interesting when the cost is also a function of the control inputs and in higher dimensions