MAE 598: Multi-Robot Systems

Fall 2016

Instructor: **Spring Berman**

spring.berman@asu.edu

Assistant Professor, Mechanical and Aerospace Engineering

Autonomous Collective Systems Laboratory

http://faculty.engineering.asu.edu/acs/

Lectures 22-23: Implicit Motion Planning using Potential Fields

Motion Planning

Let $\mathcal{A}$ be a single rigid object — the *robot* — moving in a Euclidean space $\mathcal{W}$, called *workspace*, represented as $\mathbf{R}^N$, with $N = 2$ or 3.

Let $\mathcal{B}_1, ..., \mathcal{B}_q$ be fixed rigid objects distributed in $\mathcal{W}$. The $\mathcal{B}_i$'s are called *obstacles*.

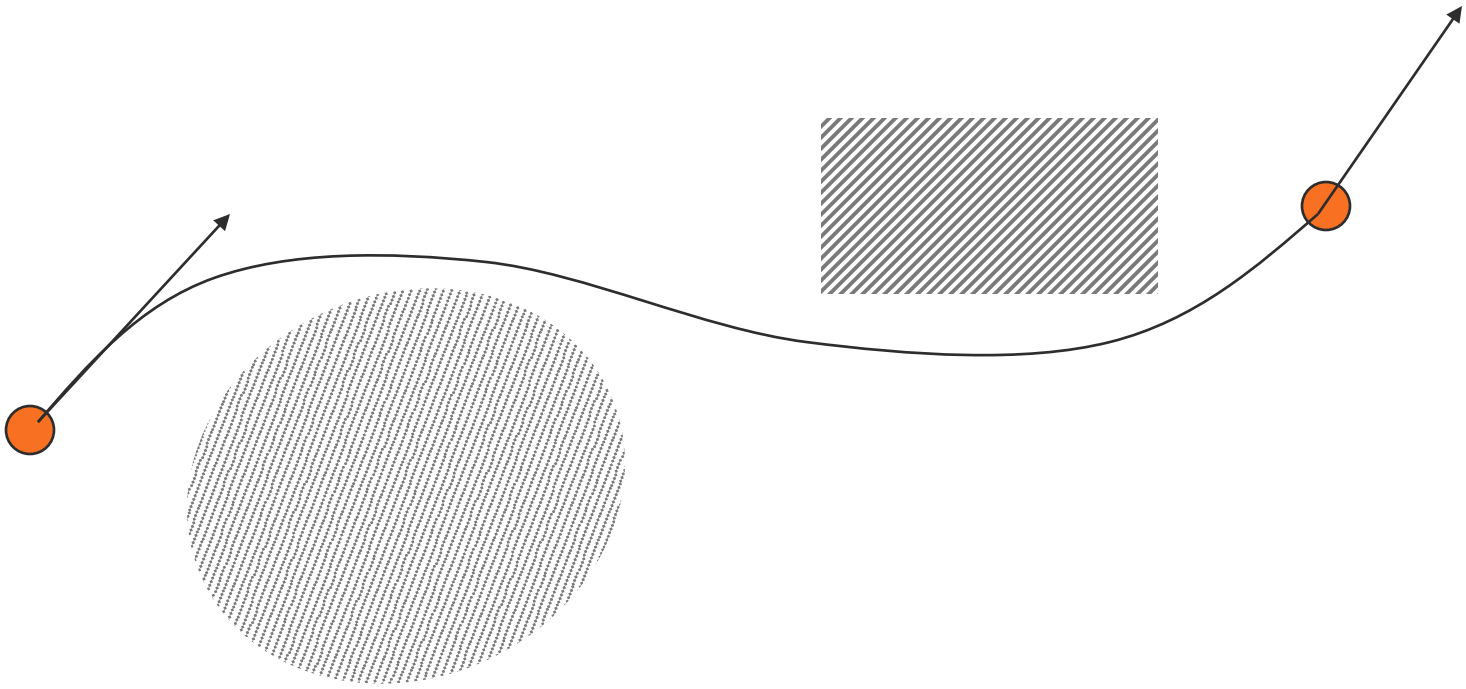
Assume that both the geometry of $\mathcal{A}, \mathcal{B}_1, ..., \mathcal{B}_q$ and the locations of the $\mathcal{B}_i$'s in $\mathcal{W}$ are accurately known. Assume further that no kinematic constraints limit the motions of $\mathcal{A}$ (we say that $\mathcal{A}$ is a *free-flying object*).

The problem is: Given an initial position and orientation and a goal position and orientation of $\mathcal{A}$ in $\mathcal{W}$, generate a *path* τ specifying a continuous sequence of positions and orientations of $\mathcal{A}$ avoiding contact with the $\mathcal{B}_i$'s, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists.

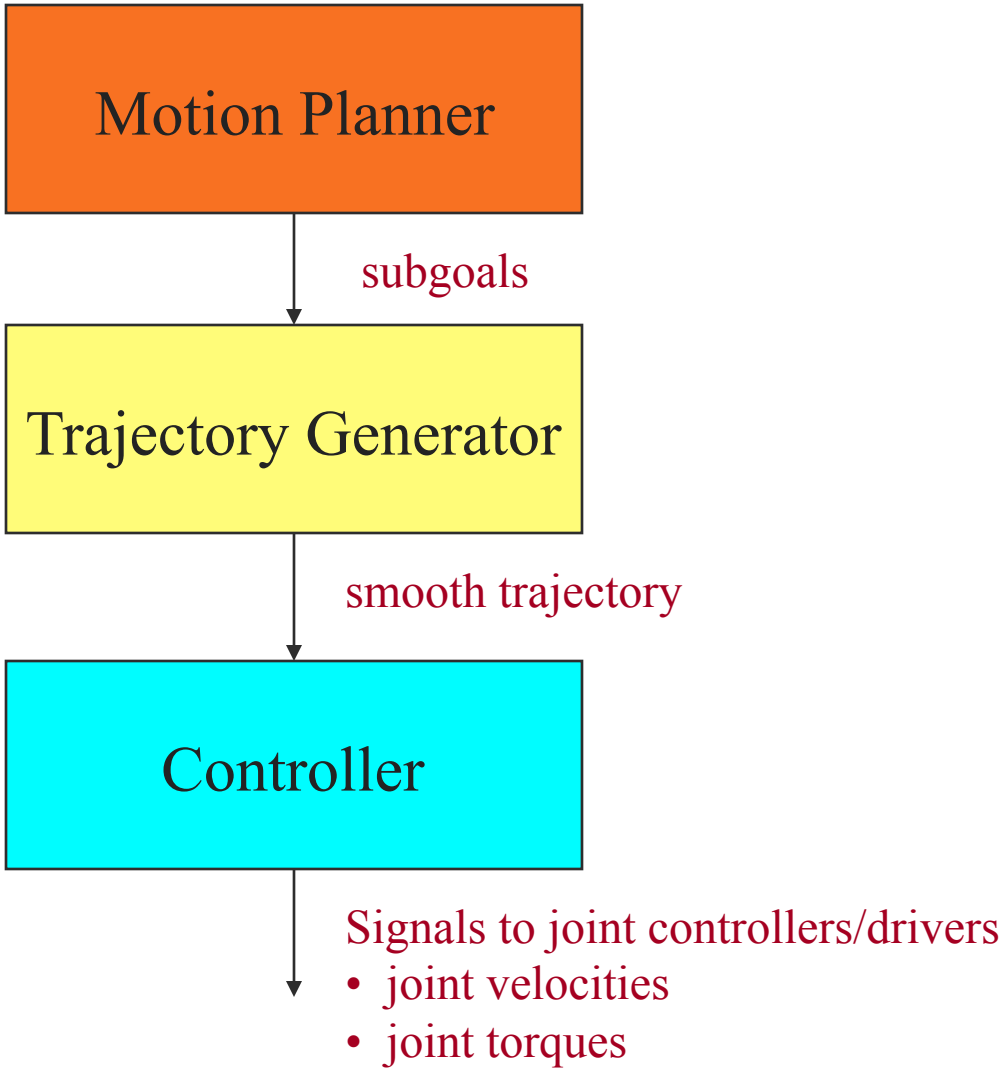
Robot Motion Planning. J.C. Latombe. Kluwer Academic Publishers, Boston, MA, 1991

Trajectory generation

Given end points (and possibly intermediate points) and velocities, generate a smooth trajectory between them while avoiding obstacles and obeying constraints



Explicit Motion Planning



Control

Let $\mathcal{A}$ be a single rigid object — the *robot* — moving in a Euclidean space $\mathcal{W}$, called *workspace*, represented as $\mathbf{R}^N$, with $N = 2$ or 3.

Let $\mathcal{B}_1, ..., \mathcal{B}_q$ be fixed rigid objects distributed in $\mathcal{W}$. The $\mathcal{B}_i$'s are called *obstacles*.

Assume that both the geometry of $\mathcal{A}, \mathcal{B}_1, ..., \mathcal{B}_q$ and the locations of the $\mathcal{B}_i$'s in $\mathcal{W}$ are accurately known. Assume further that no kinematic constraints limit the motions of $\mathcal{A}$ (we say that $\mathcal{A}$ is a *free-flying object*).

Find a control input $u(t)$

The problem is: Given an initial position and orientation and a goal position and orientation of $\mathcal{A}$ in $\mathcal{W}$, generate a *path* τ specifying a continuous sequence of positions and orientations of $\mathcal{A}$ avoiding contact with the $\mathcal{B}_i$'s, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists.

Robot Motion Planning. J.C. Latombe. Kluwer Academic Publishers, Boston, MA, 1991

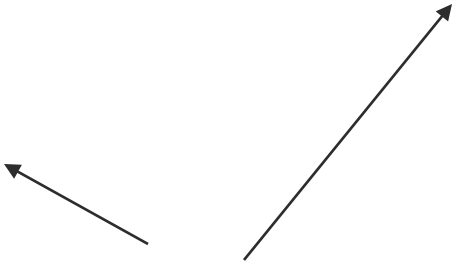
Motion Planning

Known Environments (Model)

- Explicit motion plans
- Implicit motion plans

Unknown Environments

- Sensor based motion planning



Usually combines

Motion Planning

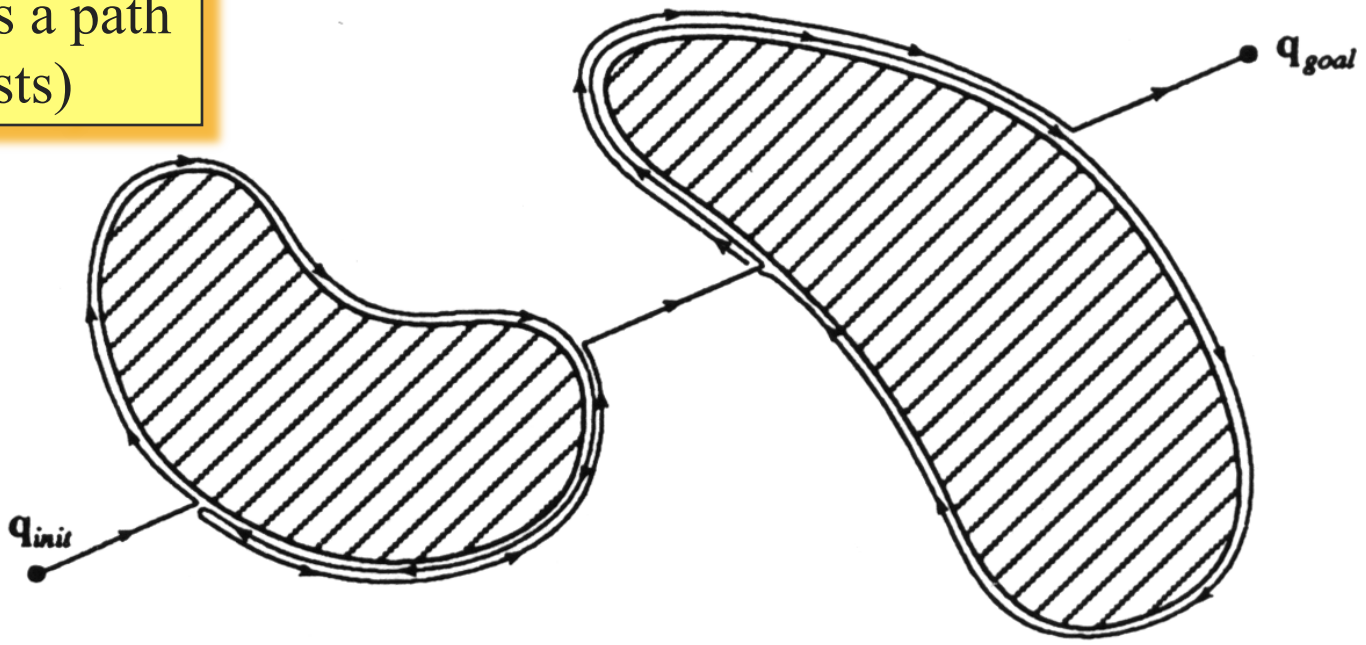
Trajectory Generation

Control

Earliest Implicit Motion Planning Algorithm

1. The robot follows a straight line segment to the goal.
2. When it hits an obstacle (at the hitting point), it follows its boundary while keeping track of the straight line segment.
3. When it returns to the hitting point, it follows the boundary to the point on the boundary that is on the line segment and closest to the goal.
4. It then resumes the straight line segment path to the goal.

Always finds a path (if it exists)



Bug Algorithm: *Vladimir Lumelsky*

Implicit Method: Potential Field Controllers

Basic idea

- Create attractive potential field to pull robot (R) toward a goal

$$V_{goal} = k\left[d(R, goal)\right]^2$$

- Create repulsive potential field to repel robot (R) from obstacles

$$V_{obs} = \frac{c}{d(R, obs)}$$

- In two-dimensional space (robot is a point, goal/obstacles are points)

$$d(R, goal) = \left[(x - x_{goal})^2 + (y - y_{goal})^2\right]^{\frac{1}{2}}$$

$$d(R, obs) = \left[(x - x_{obs})^2 + (y - y_{obs})^2\right]^{\frac{1}{2}}$$

- Remember: Force on a particle is given by $f = -grad\ (V)$

Implicit Method: Potential Field Controllers

Basic idea

- Construct potential field for goal
- Construct potential field for each obstacle
- Add potential fields to create the total potential $V(x, y)$

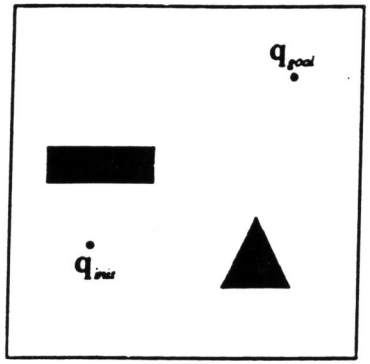
  Assume two-dimensional space (robot is a point)

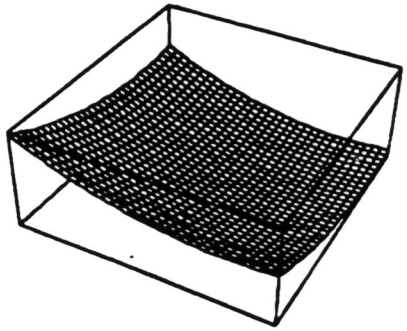
- Force on a particle is given by $f = -grad\ (V)$
- Command robot velocity according to the following control law (policy)

$$\frac{dx}{dt} = -k\frac{\partial V}{\partial x}$$

$$\frac{dy}{dt} = -k\frac{\partial V}{\partial y}$$

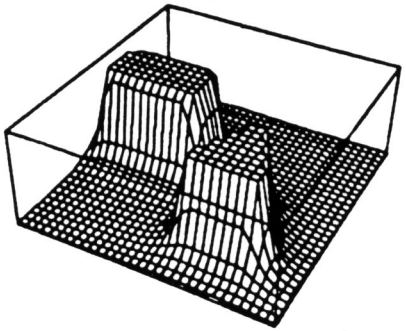
Basic idea

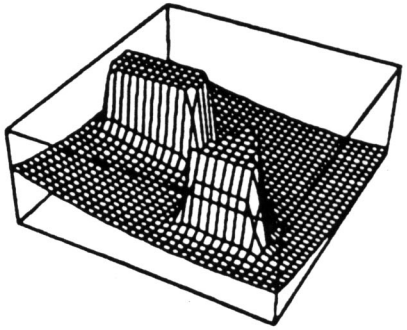




Attractive potential field for goal

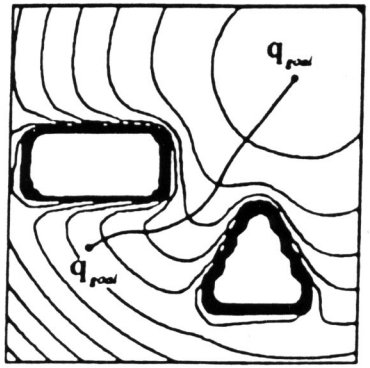
Repulsive potential field for obstacles

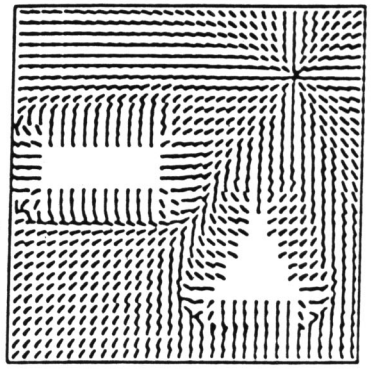




Attractive + repulsive potential fields

Equi-potential contours



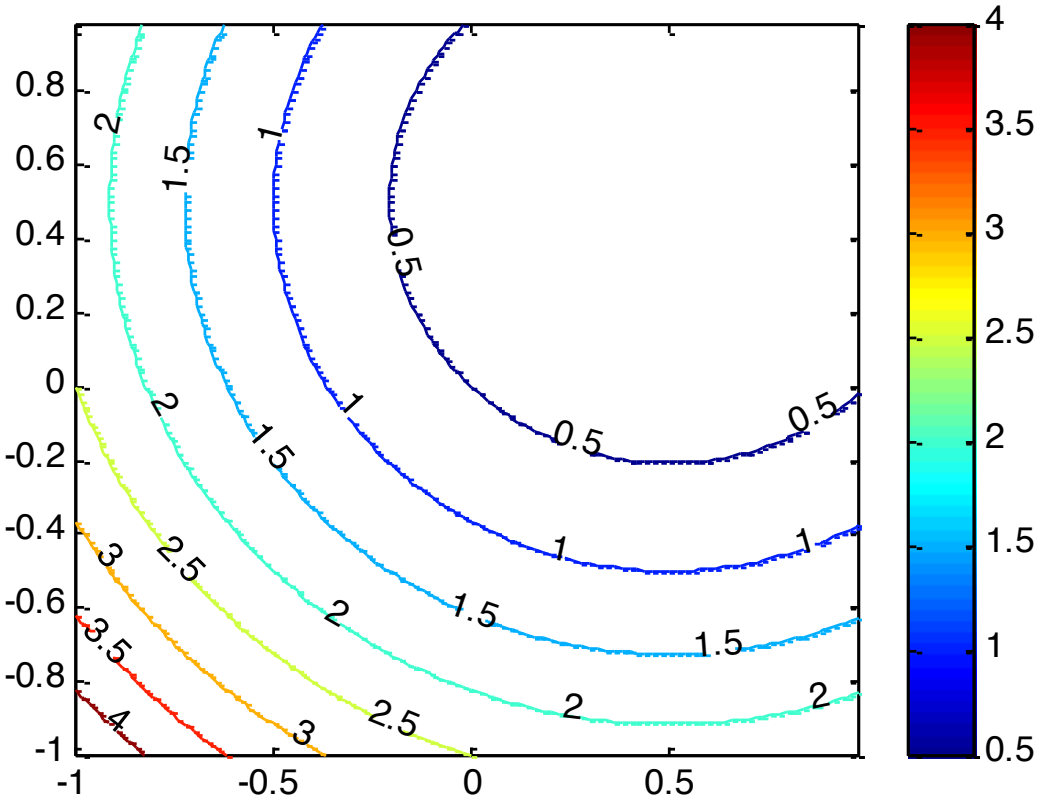


Force field

[Latombe 91]

Potential Field: Goal (x=0.5, y=0.5)

Contour plot of V_{goal}

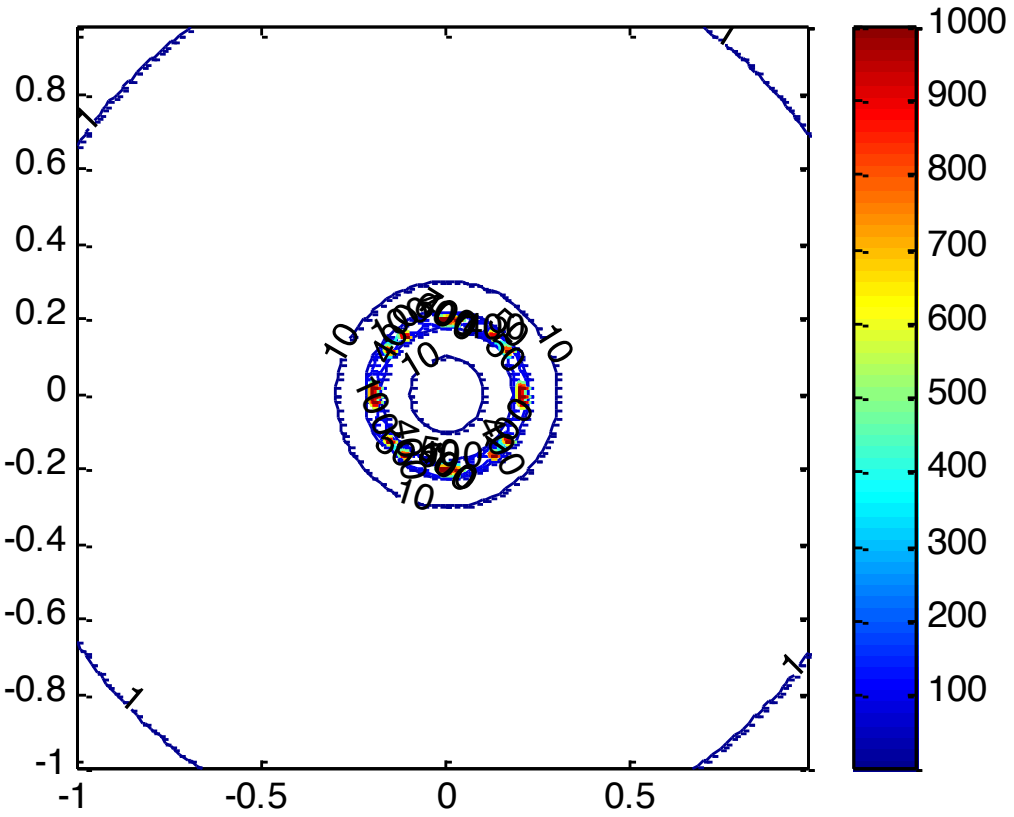


$$V_{goal} = \left[(x - x_{goal})^2 + (y - y_{goal})^2 \right]^{\frac{1}{2}}$$

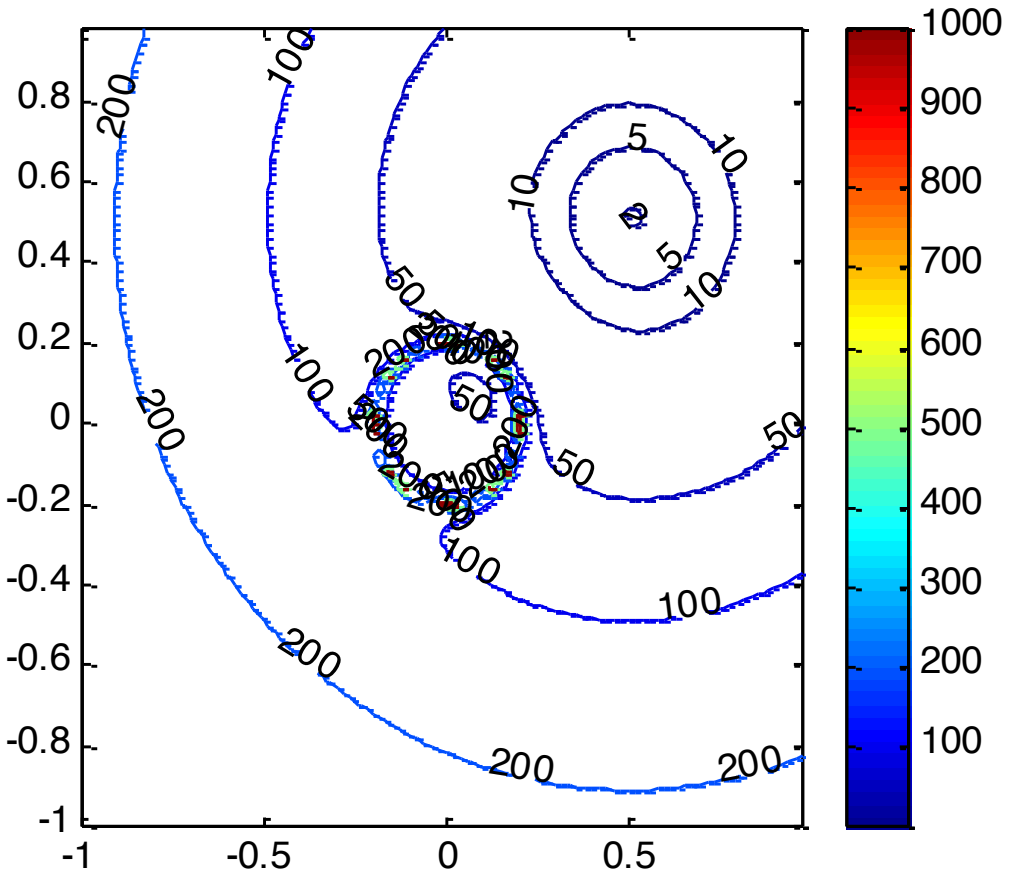
Potential Field: Obstacle (x=0.0, y=0.0)

$$V_{obs} = \frac{1}{\left[\left(x - x_{goal}\right)^2 + \left(y - y_{goal}\right)^2\right]^{\frac{1}{2}}}$$

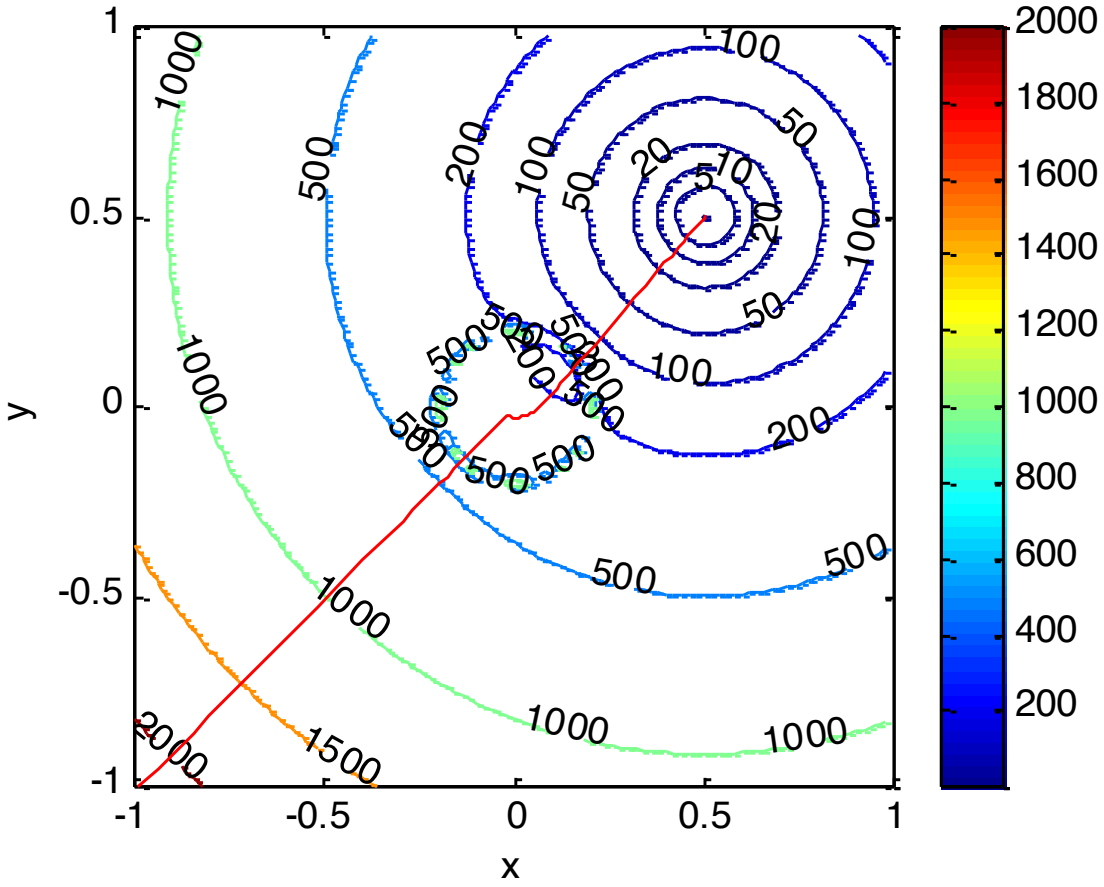
Contour plot of V_{obs}



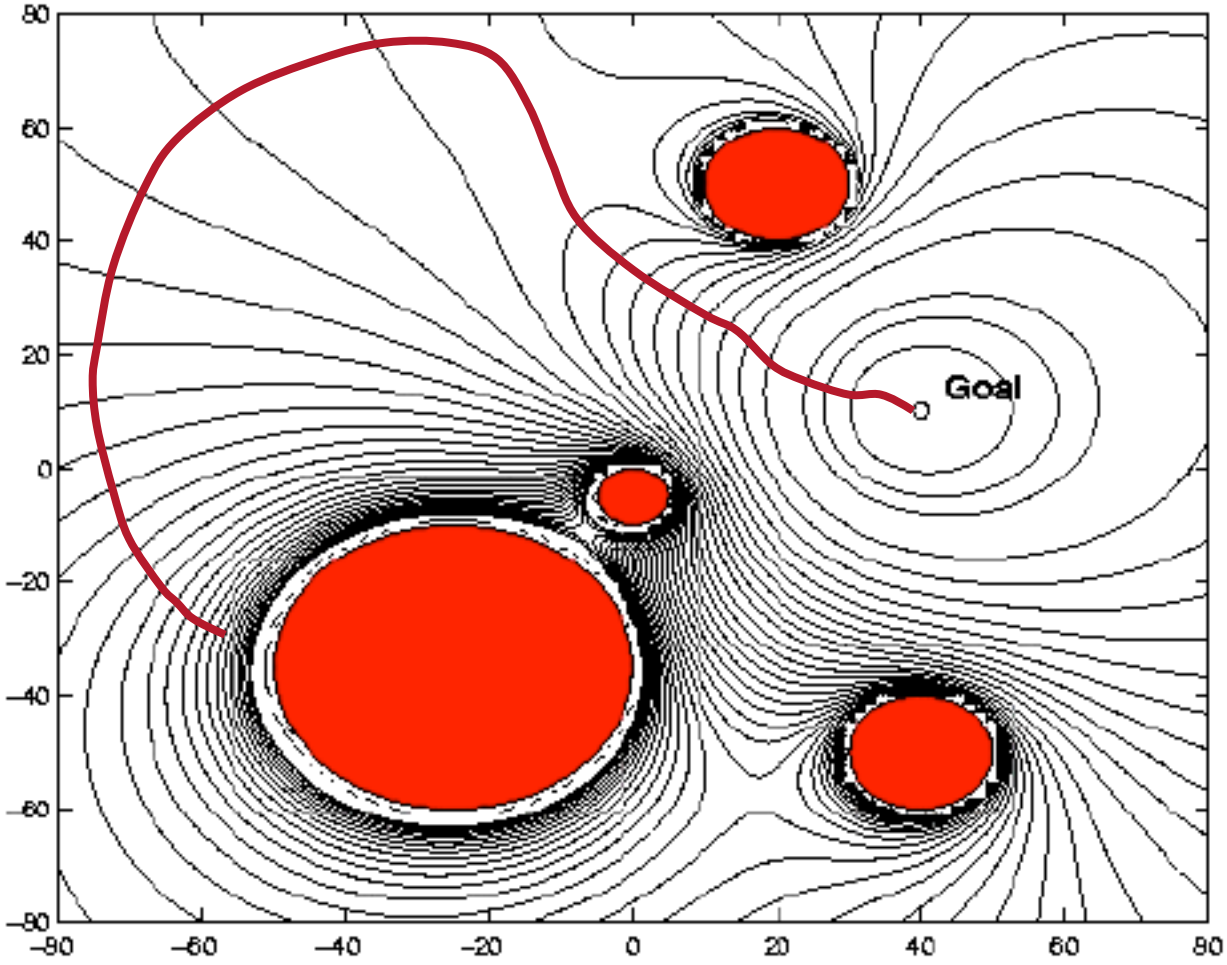
$$V_{obs} + 100\ V_{goal}$$



Trajectory for $V_{obs} + 500\ V_{goal}$



What are the potential (no pun!) difficulties?



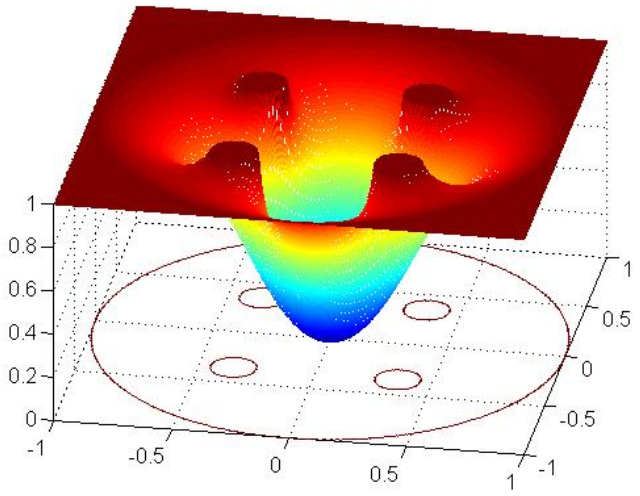
Navigation Functions

[1] Rimon and Koditschek, "Exact robot navigation using artificial potential functions." IEEE Trans. on Robotics and Automation, 8(5):501-518, Oct. 1992.

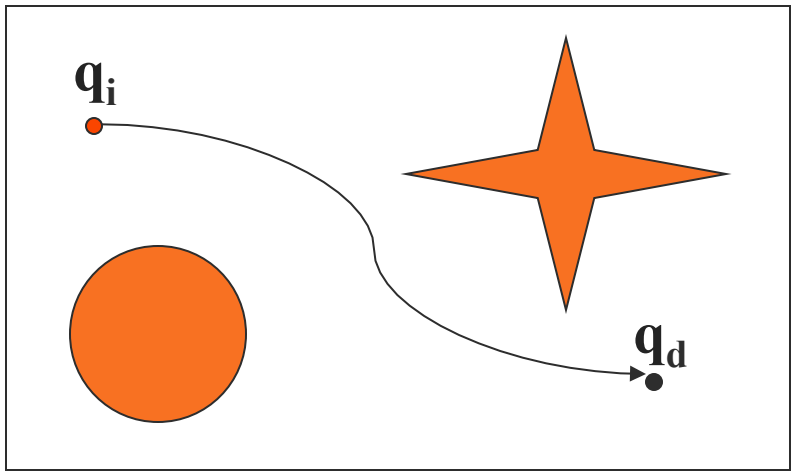
For theoretical background:

[2] Rimon and Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds." Trans. of Amer. Math. Society, 327(1): 71-115, Sept. 1991.

[3] Rimon and Koditschek, "Robot navigation functions on manifolds with boundary." Advances in Applied Mathematics, 11:412-442, 1990.



Problem Statement



Assume:

- Point robot
- Stationary 2D world
- Perfect info about environment
- Ideal sensors and actuators

Construct a smooth, bounded-torque controller τ such that the robot trajectory goes to $\mathbf{q_d}$ while avoiding the obstacles.

Traditionally: Path planning + trajectory planning + robot control

Navigation function approach

$$\tau = -\nabla\varphi(p) + d(p, \dot{p})$$

Definition

$$\varphi : \mathcal{M} \rightarrow [0, 1]$$

$\mathcal{M}$ is a compact, connected, analytic manifold with boundary

- Manifold of dimension *n*: set that is locally homeomorphic to R^n

A C^∞ φ with a unique minimum at $\boldsymbol{q}_\mathbf{d}$ exists for any $\boldsymbol{q}_\mathbf{d}$ in the interior of an $\mathcal{M}$ with these properties (Theorem 3 of [3])

Definition

Navigation functions are:

M

A

P

S

Definition

Navigation functions are:

Morse: nonsingular Hessian at critical points

Admissible: uniformly maximal on boundary of $\mathcal{M}$

Polar: unique minimum at q_d

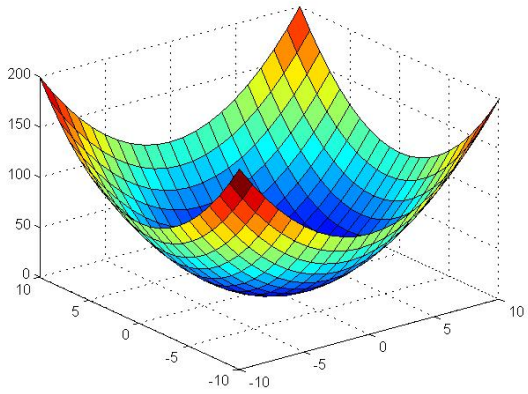
Smooth: at least C^2

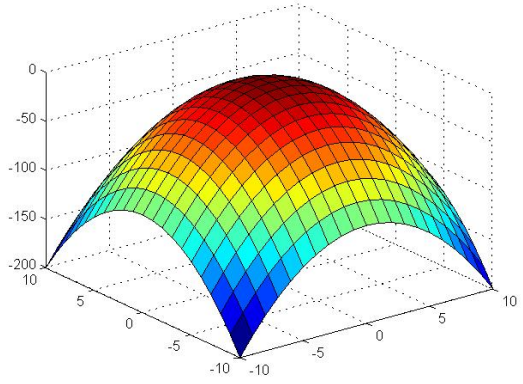
Definition

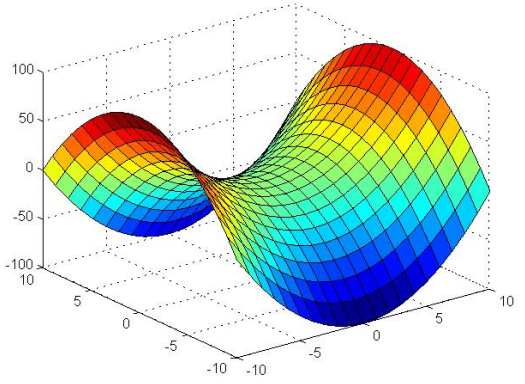
Morse

- Hessian matrix of φ evaluated at its critical points is nonsingular → critical points are nondegenerate (maxima, minima, or saddle points)

- Ensures that trajectories beginning away from a set of measure 0 asymptotically approach a local minimum







Definition

Admissible

- Uniformly maximal (a constant $c > 0$) on boundary $\partial\mathcal{M}$ of freespace
- Guarantees that trajectories starting in $\mathcal{M}$ with bounded initial velocity will remain away from obstacles
- Makes smooth potential field defined on $\mathcal{M}$ bounded, producing a bounded controller

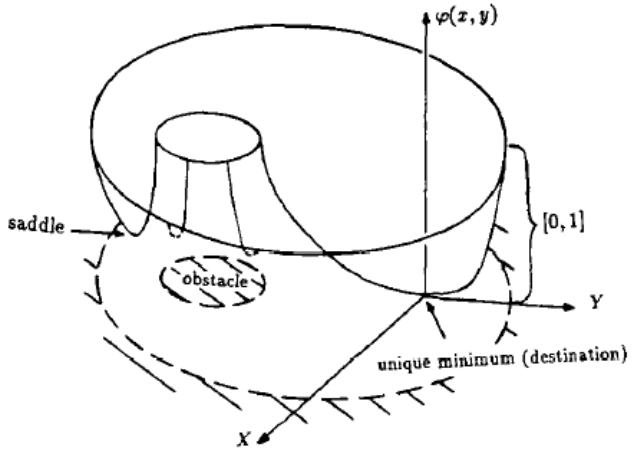
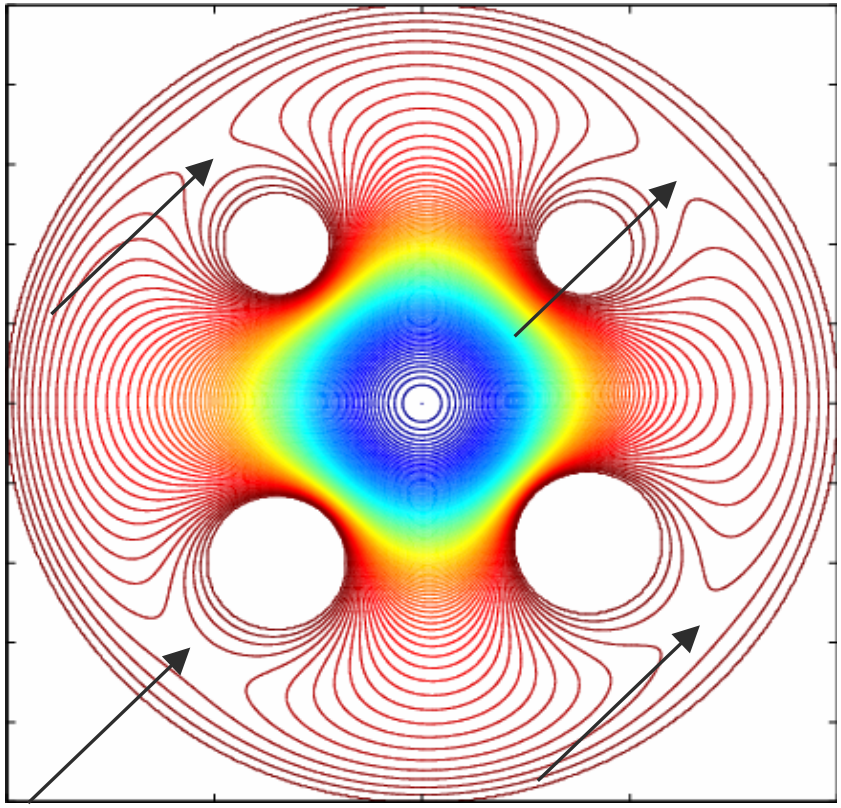


Fig. 2. A suitable potential function over a free configuration space.

From [1]

Properties

- Each obstacle introduces at least one saddle point of φ (Corollary 2.3 of [3])



Properties

- Properties of φ are invariant under an analytic diffeomorphism h (a smooth, bijective map with a smooth inverse): (Prop. 2.6 of [3])

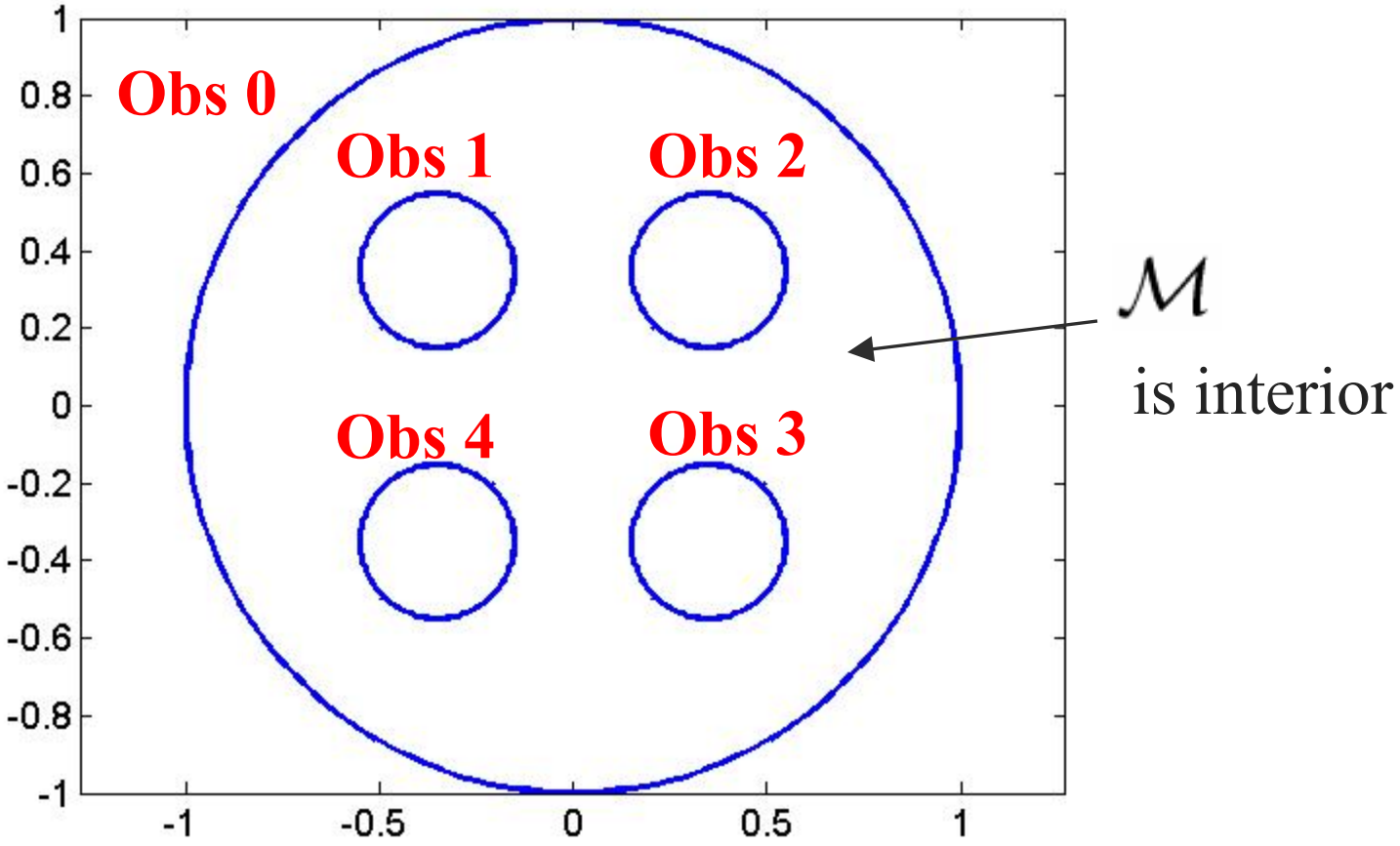
$$h : \mathcal{F} \rightarrow \mathcal{M} \qquad \tilde{\varphi} \equiv \varphi \circ h$$

Nav fn on F ← $\tilde{\varphi}$; φ → Nav fn on M

→ Can construct a navigation function on any space deformable to $\mathcal{M}$

Sphere Worlds

Obstacles are disks; obstacle 0 is complement of largest disk



Sphere Worlds

Obstacle functions: $\text{obs}_i = \{q : \beta_i(q) \leq 0\}$

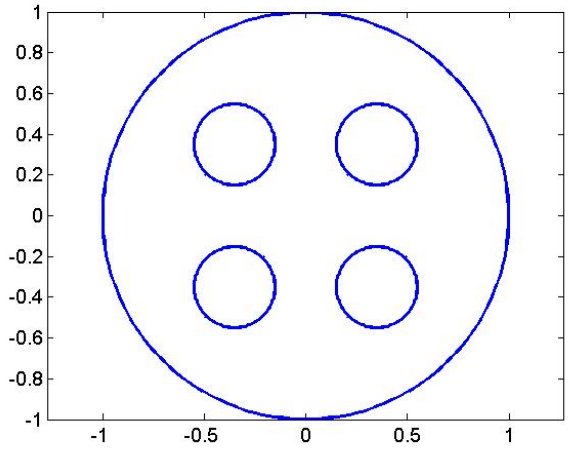
$$\beta_0(q) = -\|q - q_0\|^2 + \rho_0^2$$

$$\beta_j(q) = \|q - q_j\|^2 - \rho_j^2 \quad (j = 1, ..., M)$$

$q = [x\ y]$

q_i = obstacle center

ρ_i = obstacle radius

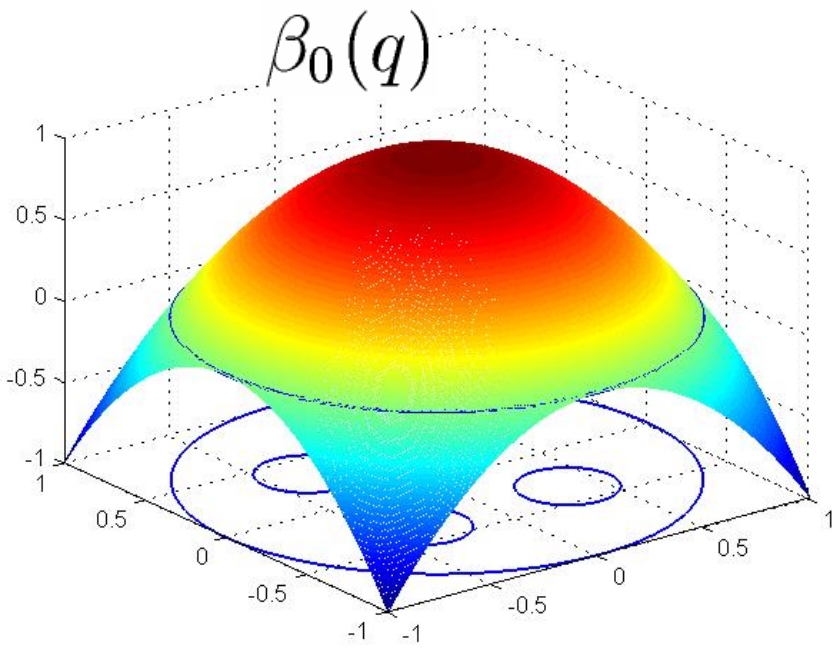


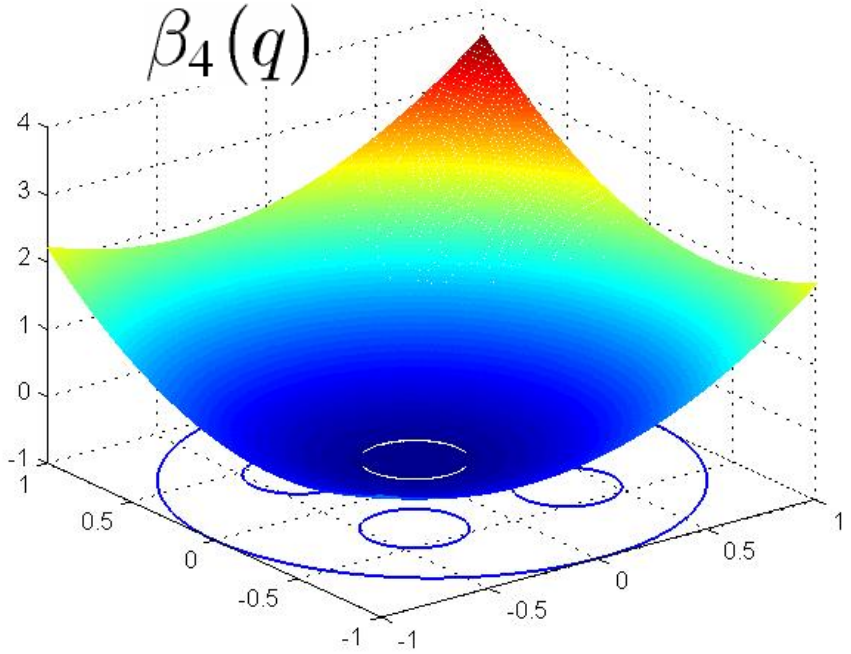
Sphere Worlds

Obstacle functions: $\text{obs}_i = \{q : \beta_i(q) \leq 0\}$

$$\beta_0(q) = -\|q - q_0\|^2 + \rho_0^2$$

$$\beta_j(q) = \|q - q_j\|^2 - \rho_j^2 \quad (j = 1, ..., M)$$

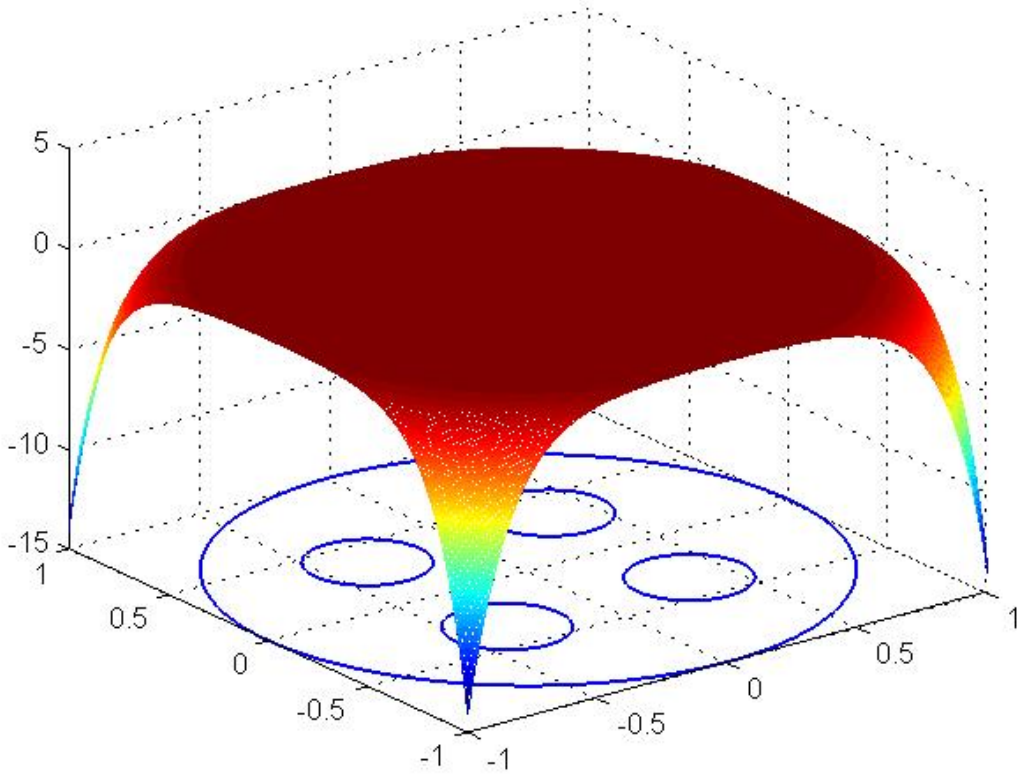




Sphere Worlds

Obstacle functions: $\text{obs}_i = \{q : \beta_i(q) \leq 0\}$

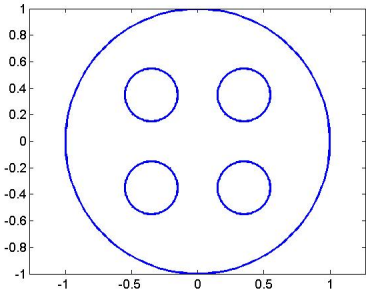
$$\beta = \prod_{i=0}^{M} \beta_i$$



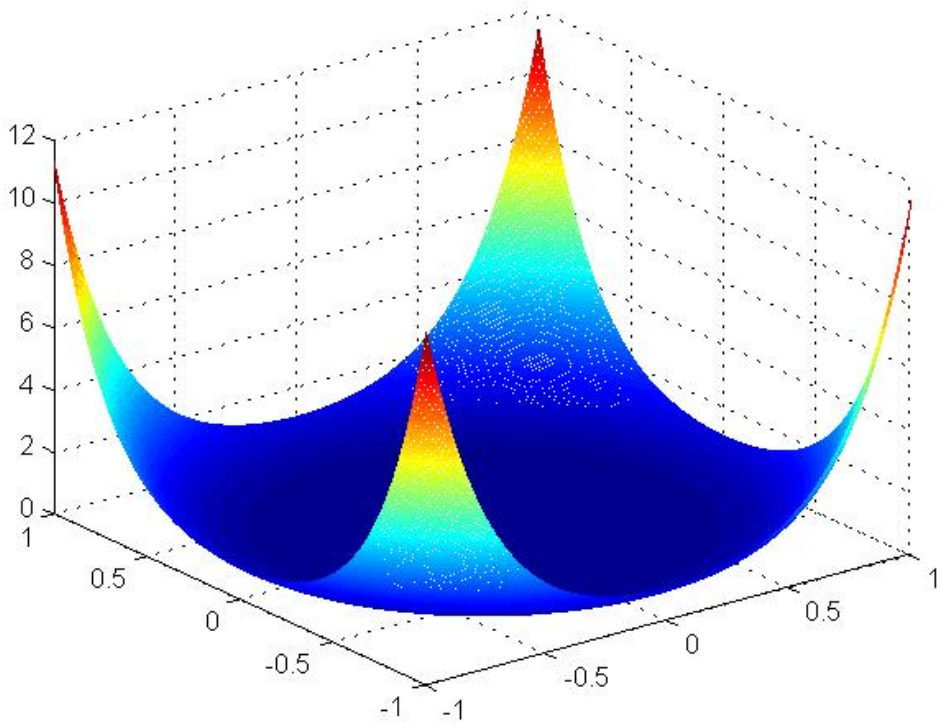
Sphere Worlds

Distance-to-goal function:

$$\gamma_\kappa(q) = \|q - q_d\|^{2\kappa}$$



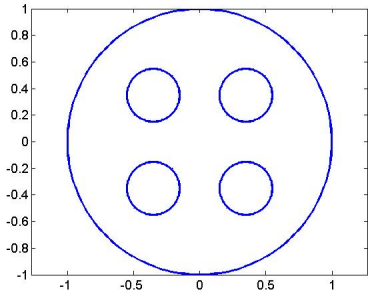
$$\kappa = 3.5$$



Sphere Worlds

Distance-to-goal function:

$$\gamma_\kappa(q) = \|q - q_d\|^{2\kappa}$$



$\kappa > 0$

- φ is a valid navigation function if $\kappa \geq N$, where N is a function of the geometric data

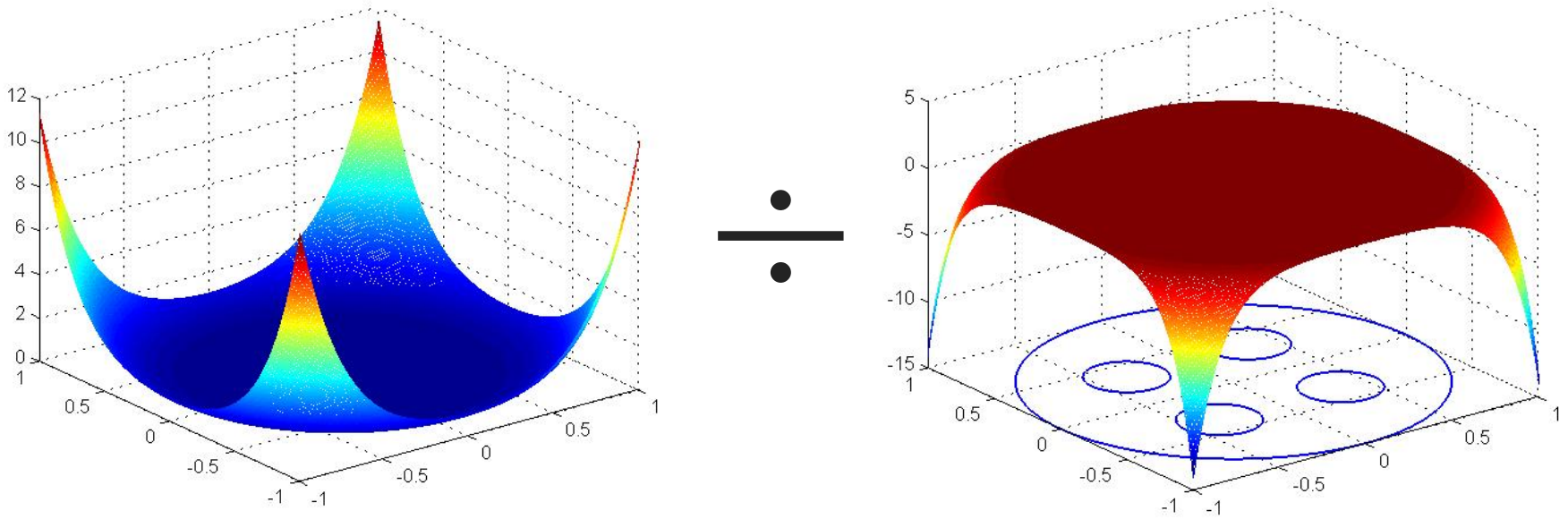
- As κ increases, local minima disappear

- Setting κ arbitrarily high can lead to gradient vector fields that vary too abruptly to be implemented

- Rule of thumb: $\kappa >$ number of obstacles + 1
(but seems to work for lower κ)

Sphere Worlds

$$\hat{\varphi} = \gamma_\kappa / \beta$$

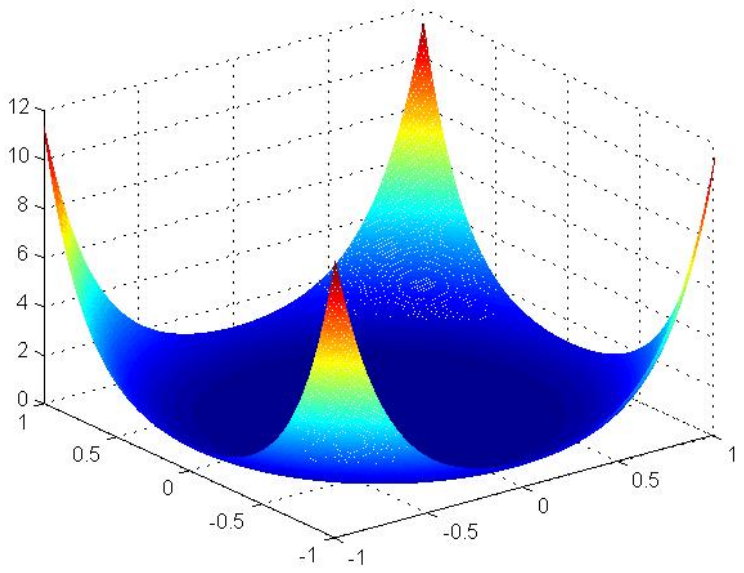


Is this a navigation function?

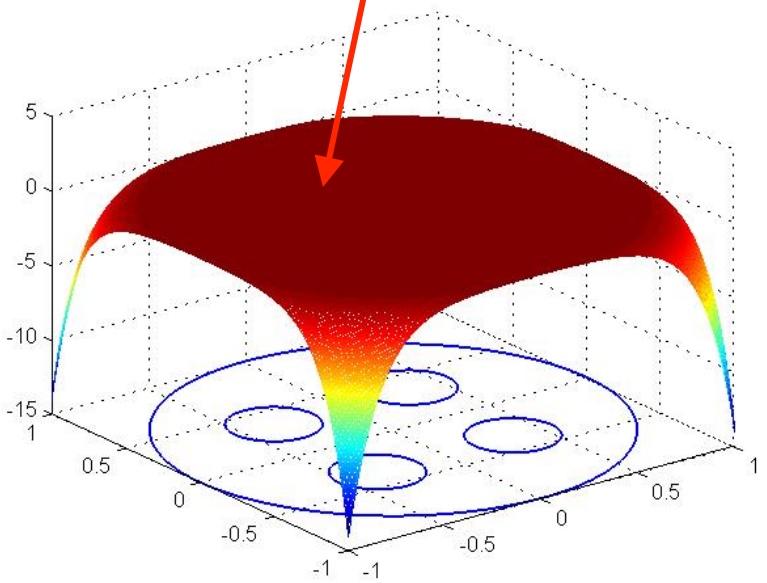
Sphere Worlds

$$\hat{\varphi} = \gamma_\kappa / \beta$$

0 at obstacle boundaries



÷



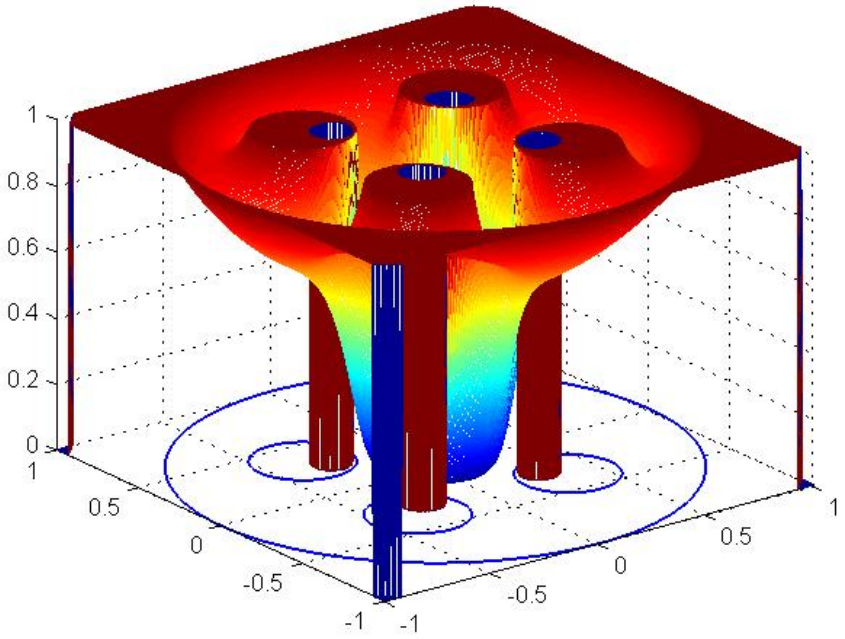
It's polar and almost everywhere Morse and analytic, but… $\varphi : \mathcal{M} \to [0, 1]$

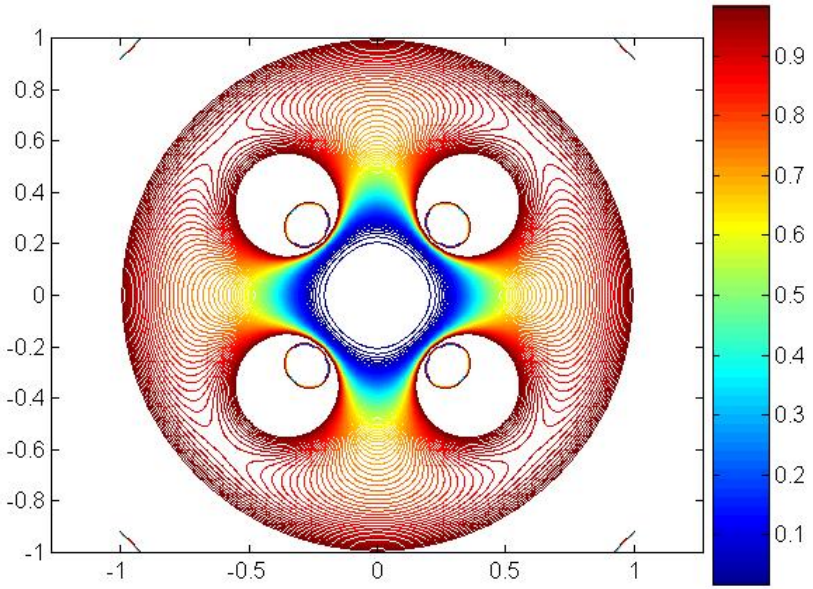
Sphere Worlds

Squash the function with an analytic switch:

$$\sigma_\lambda(x) = x/(\lambda + x) \qquad [0,\infty) \to [0,1]$$

It's polar, admissible, analytic, and Morse everywhere except at q_d

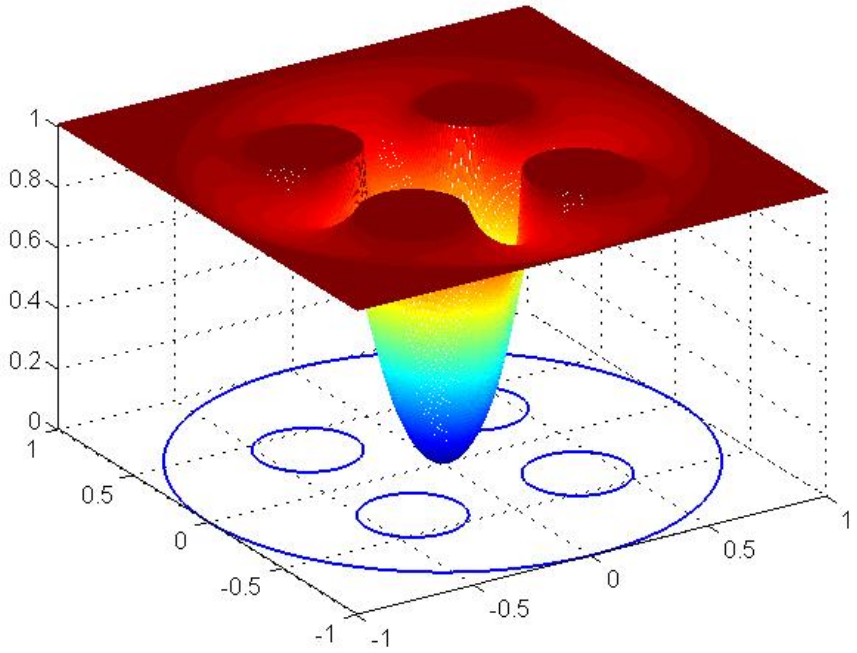


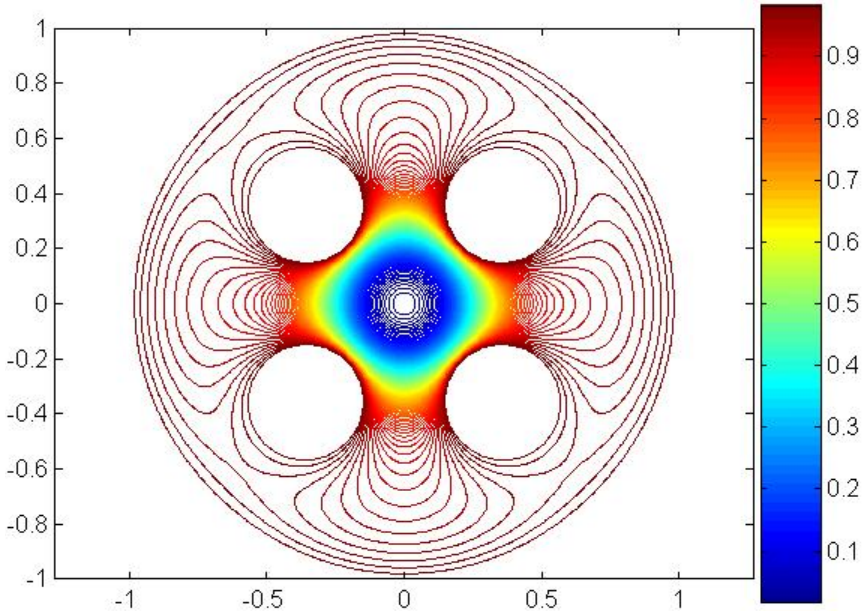


Sphere Worlds

Use a "distortion" function to make q_d nondegenerate:

$$\rho_\kappa(x) = x^{1/\kappa}$$

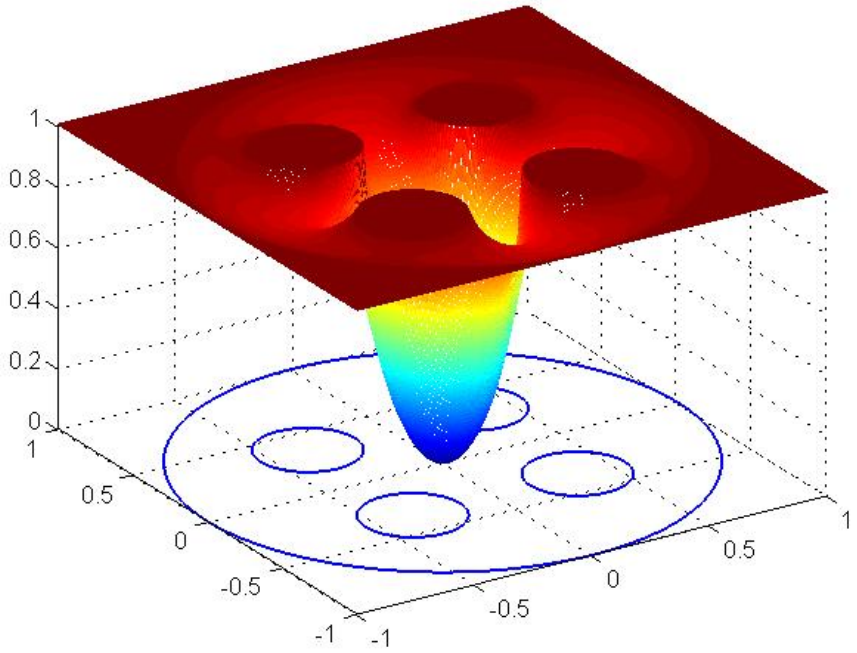


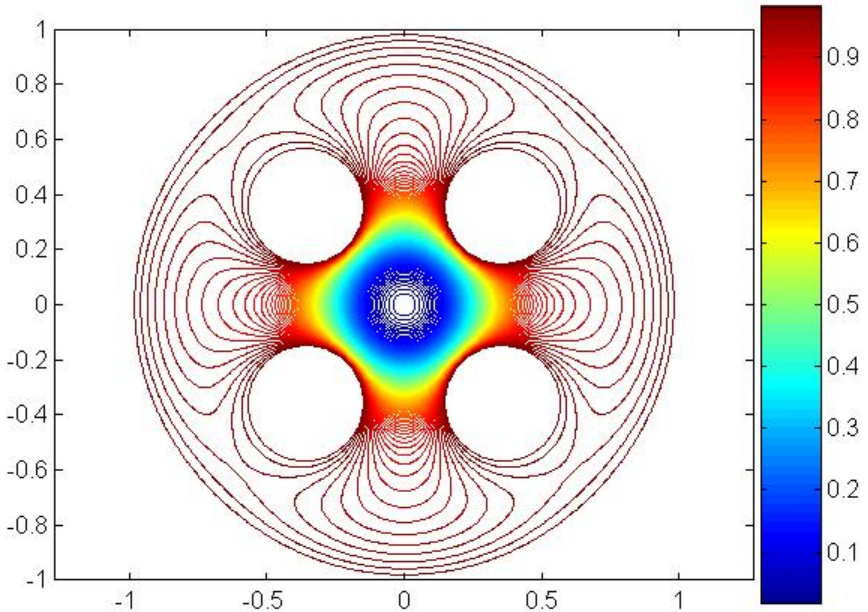


Sphere Worlds

Navigation function is thus defined as:

$$\varphi_\kappa(q) \equiv \quad (\rho_k \circ \sigma_1 \circ \hat{\varphi})(q) = \frac{\|q - q_d\|^2}{[\|q - q_d\|^{2\kappa} + \beta(q)]^{1/\kappa}}$$

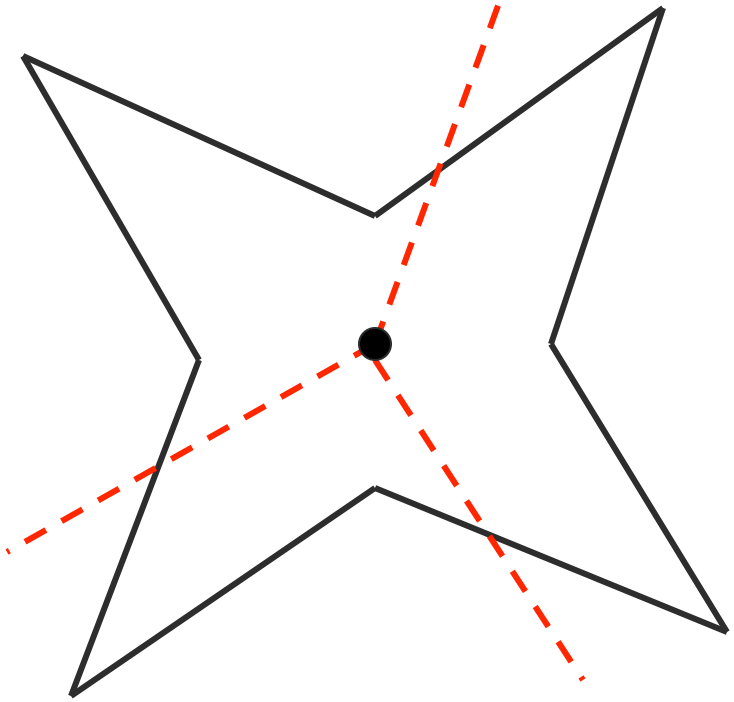




Star Worlds

Star-shaped set: Has a point from which all rays cross the boundary only once

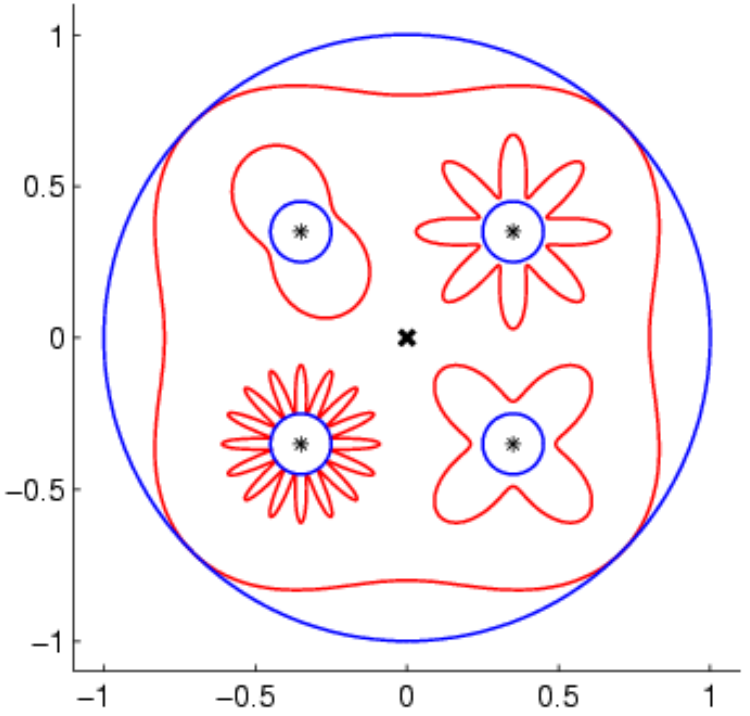
- Include all convex sets



Star Worlds

- Star world $\mathcal{F}$: All obstacles are star-shaped sets
- Construct a diffeomorphic map *h* to define a navigation function $\tilde{\varphi} = \varphi \circ h$ on $\mathcal{F}$

	$\mathcal{F}$	$\mathcal{M}$
Boundary	**red**	**blue**
Goal	q_d	p_d
Obstacle centers	q_i	p_i



Star Worlds

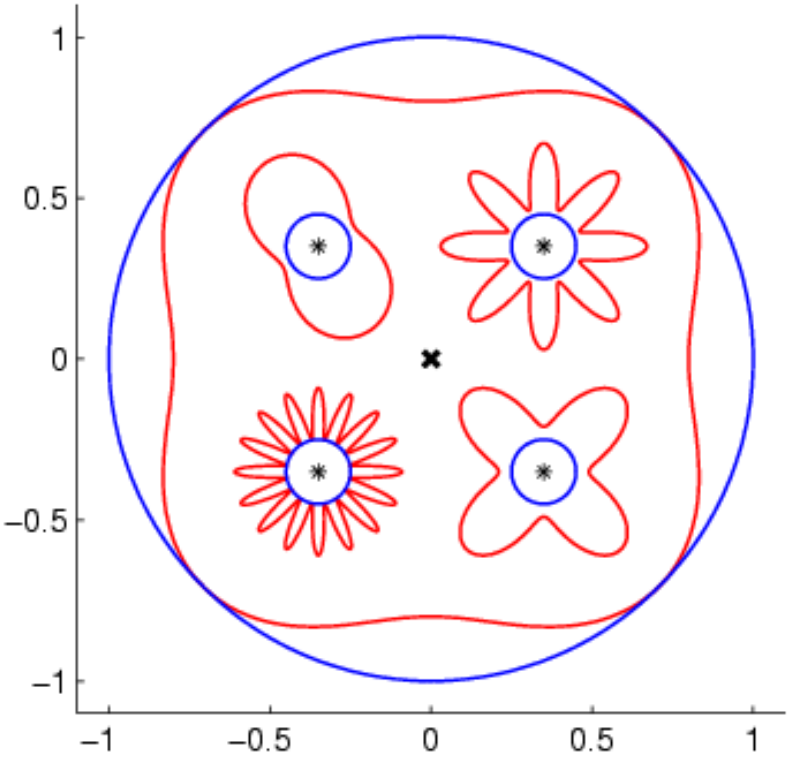
- Star world $\mathcal{F}$: All obstacles are star-shaped sets
- Construct a diffeomorphic map *h* to define a navigation function $\tilde{\varphi} = \varphi \circ h$ on $\mathcal{F}$

Constraints on $\mathcal{M}$:

$p_d = q_d \quad p_i = q_i$

Internal spheres are subsets of internal stars

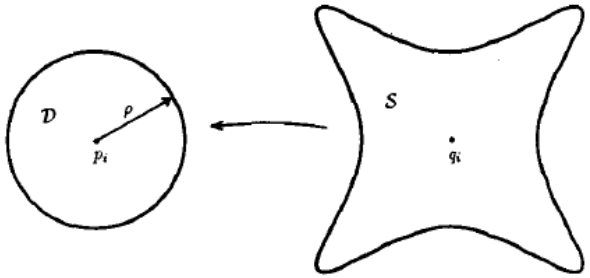
Outer sphere contains outer star



Star Worlds

Star-to-sphere transformation

$$T_i(q) = \nu_i \cdot [q - q_i] + p_i$$



star centers

sphere centers

From [1]

Scaling factors:

$$\nu_0(q) = \rho_0 \frac{1 - \beta_0(q)}{\|q - q_0\|} \qquad \nu_i(q) = \rho_i \frac{1 + \beta_i(q)}{\|q - q_i\|}$$

β_i is the implicit representation of a star obstacle

Star Worlds

"Omitted product" of star obstacle $\mathcal{S}_i$:

$$\overline{\beta}_i = \prod_{j=0, j\neq i}^{M} \beta_j$$

Analytic switch for each star obstacle:

$$s_i(q,\lambda) = (\sigma_\lambda \circ \frac{\gamma_\kappa \overline{\beta}_i}{\beta_i})(q) = \left(\frac{\gamma_\kappa \overline{\beta}_i}{\gamma_\kappa \overline{\beta}_i + \lambda \beta_i}\right)(q)$$

unity on $\partial\mathcal{S}_i$ and zero on $\partial\mathcal{S}_j$, $j \neq i$ and at q_d

- Used to map one star to one sphere when ***q*** is close to an obstacle

Star Worlds

Analytic switch for each star obstacle:

$$s_i(q,\lambda) = (\sigma_\lambda \circ \frac{\gamma_\kappa \overline{\beta}_i}{\beta_i})(q) = \left(\frac{\gamma_\kappa \overline{\beta}_i}{\gamma_\kappa \overline{\beta}_i + \lambda \beta_i}\right)(q)$$

Parameter λ

- h_λ is an analytic diffeomorphism if $\lambda \geq \Lambda$, where Λ is a function of the geometric data and $\boldsymbol{q_d}$

- High λ: shallow $\nabla\tilde{\varphi}$ near obstacle; "safe" navigation

- Low λ: robot only senses obstacle when in close proximity; "fast" navigation

Star Worlds

Identity map: $T_d(q) = q$

Destination switch: $s_d(q, \lambda) = 1 - \sum_{i=0}^{M} s_i$

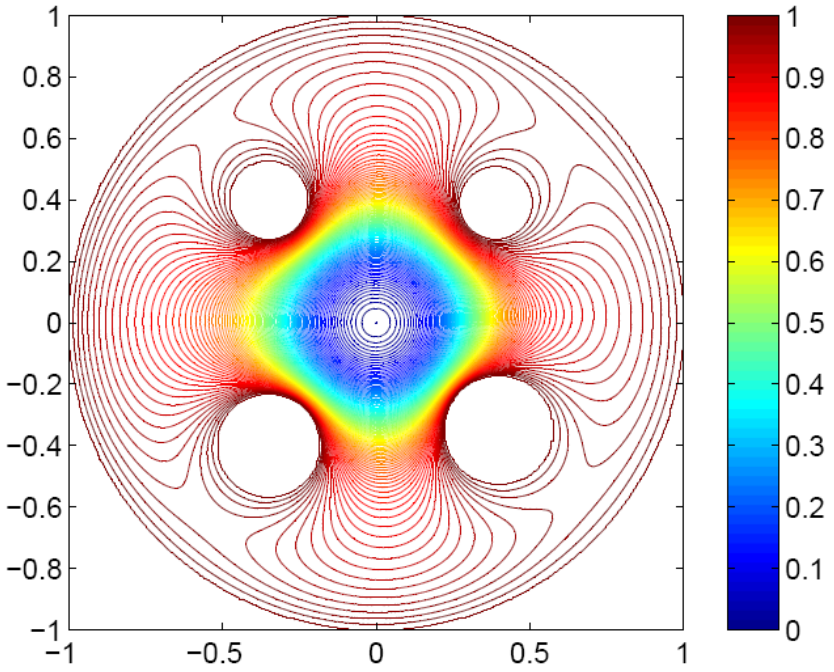
- Ensures that $h(q_d) = p_d$

$$h_\lambda(q) = s_d T_d(q) + \sum_{i=0}^{M} s_i T_i(q)$$

- Resembles T_i near obstacle i and T_d away from obstacles

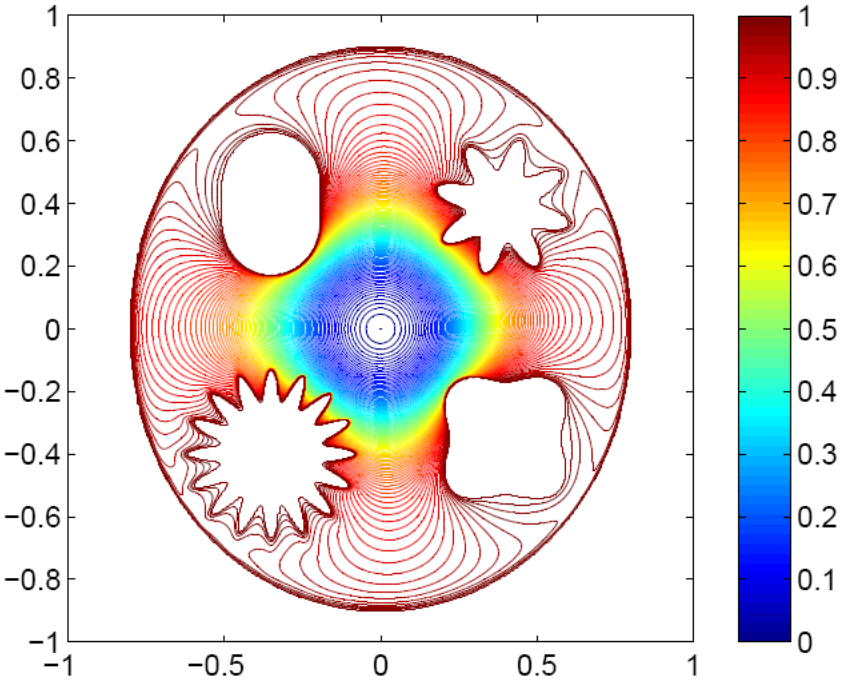
Star Worlds

φ



$\kappa = 3.5$

$\tilde{\varphi} = \varphi \circ h$



$\lambda = 10$

Forests of Stars

Obstacles are unions of overlapping stars, or "trees of stars"; maximal tree depth = d

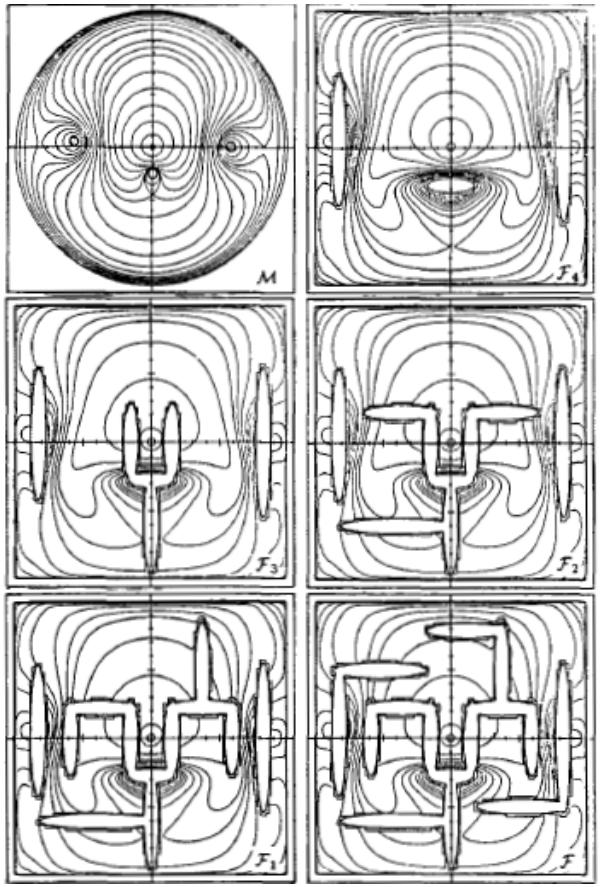
Purge forest of successive levels of stars through coordinate transformations:

$$f_\lambda = s_d T_d(q) + \sum_{i \in \mathcal{L}} s_i T_i(q)$$

$\mathcal{L}$: index set of leaves

$\mathcal{I}$: index set of stars

$$\breve{\varphi} = \varphi \circ h_\lambda \circ f_{\lambda_d} \circ \ldots \circ f_{\lambda_1}$$



From [1]

Forests of Stars

New definitions

$$s_d = 1 - \sum_{i \in \mathcal{L}} s_i$$

$$\overline{\beta}_i = \left(\prod_{k \in \mathcal{I} - \{i, p(i)\}} \beta_k \right) \cdot \left(\prod_{k \in \mathcal{L} - \{i\}} \beta_k \right) \cdot \tilde{\beta}_{p(i)}$$

Zenkin's formula:

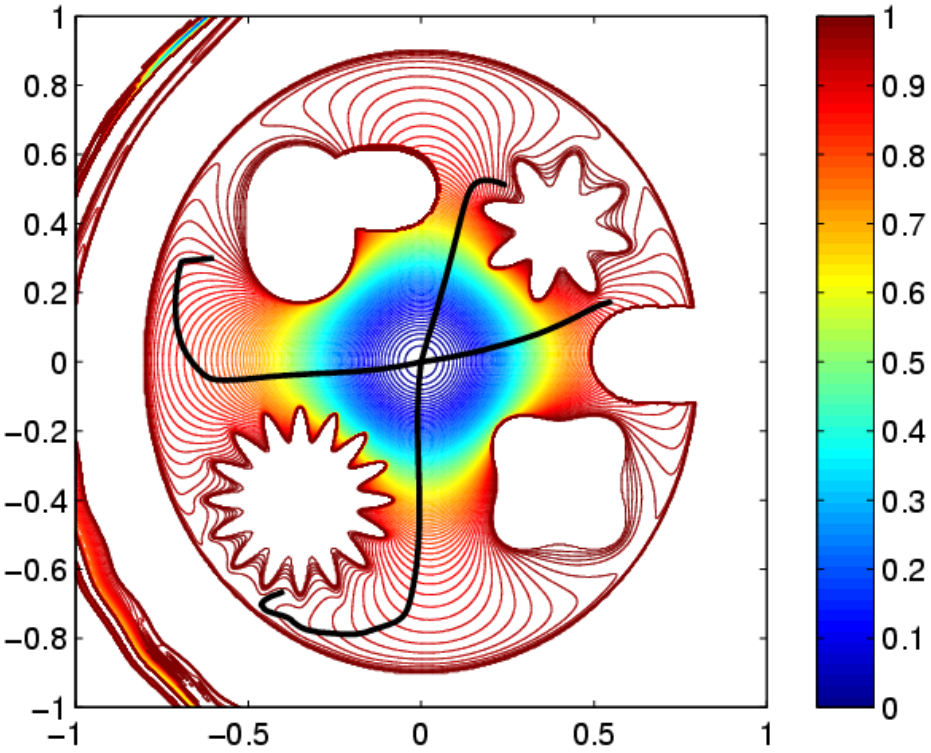
$$\tilde{\beta}_{p(i)}(q) = \beta_{p(i)}(q) + (2E_i - \beta_i(q)) + \sqrt{\beta_{p(i)}^2 + (2E_i - \beta_i(q))^2}$$

$$T_i = \frac{\|L_p(x) - x_r\|}{\|L_c(x) - x_r\|}(x - x_r) + x_r$$

- Maps exposed part of leaf boundary onto part of parent boundary it encloses

Forests of Stars

$$\breve{\varphi} = \varphi \circ h_\lambda \circ f_{\lambda_d} \circ \ldots \circ f_{\lambda_1}$$



Trajectories from: $\dot{x} = -\nabla\breve{\varphi}$

Simulation

Planar kinematic model

- *Continuous model*

$$\dot{q} = [\dot{x} \quad \dot{y}]^T = -K\nabla\varphi = -K\left[\frac{\partial\varphi}{\partial x} \quad \frac{\partial\varphi}{\partial y}\right]^T \qquad K > 0 \text{ is a gain}$$

Can integrate in Matlab using ode45() function

- Sphere worlds in Matlab: can get exact gradient by using symbolic expression of φ (`syms, diff, subs` functions)

- Star worlds, forests of stars: can use numerical approximation

$$\frac{\partial\varphi}{\partial x} \approx [\varphi(x+\delta, y) - \varphi(x, y)]/\delta \qquad \frac{\partial\varphi}{\partial y} \approx [\varphi(x, y+\delta) - \varphi(x, y)]/\delta$$

Simulation

Planar kinematic model

- *Continuous model*

$$\dot{q} = [\dot{x} \quad \dot{y}]^T = -K\nabla\varphi = -K\left[\frac{\partial\varphi}{\partial x} \quad \frac{\partial\varphi}{\partial y}\right]^T$$

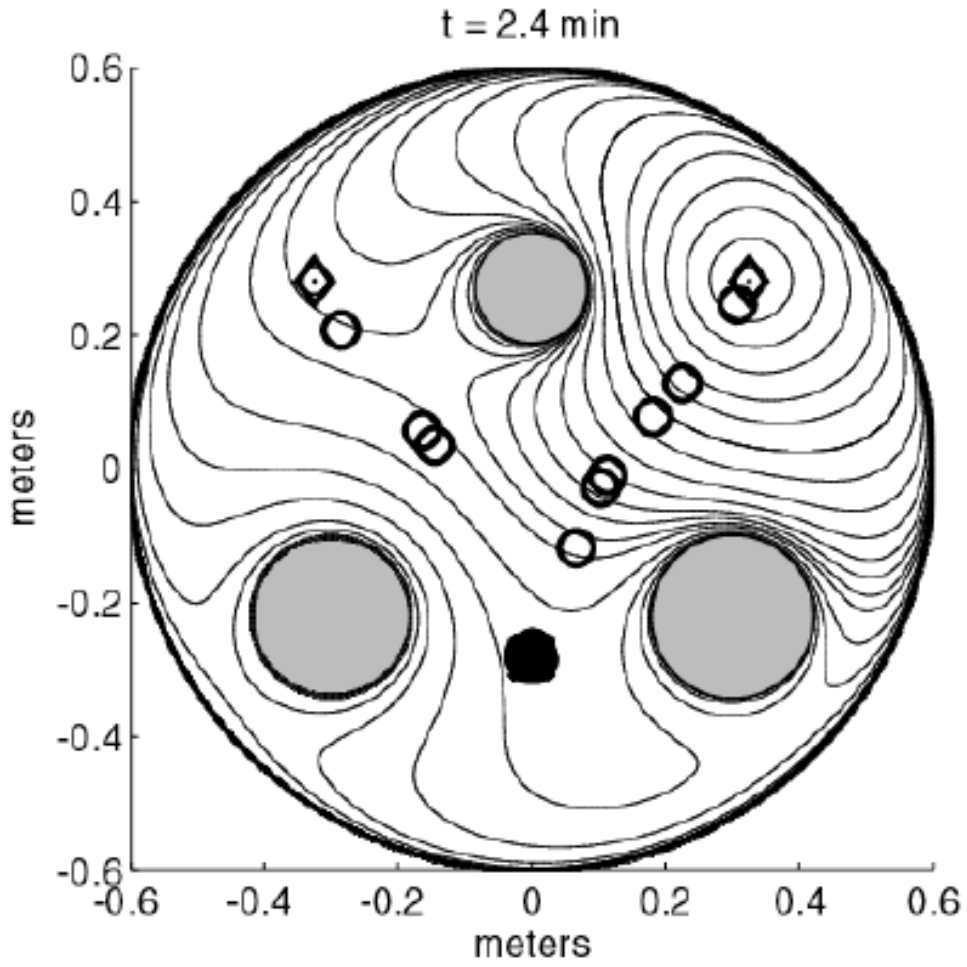
$K > 0$ is a gain

Can integrate in Matlab using ode45() function

- *Discrete time implementation*

$$q(t + \Delta t) = q(t) - K\nabla\varphi\Delta t$$

Simulation of Ants Navigating to Nests



Uses navigation functions with different goal points