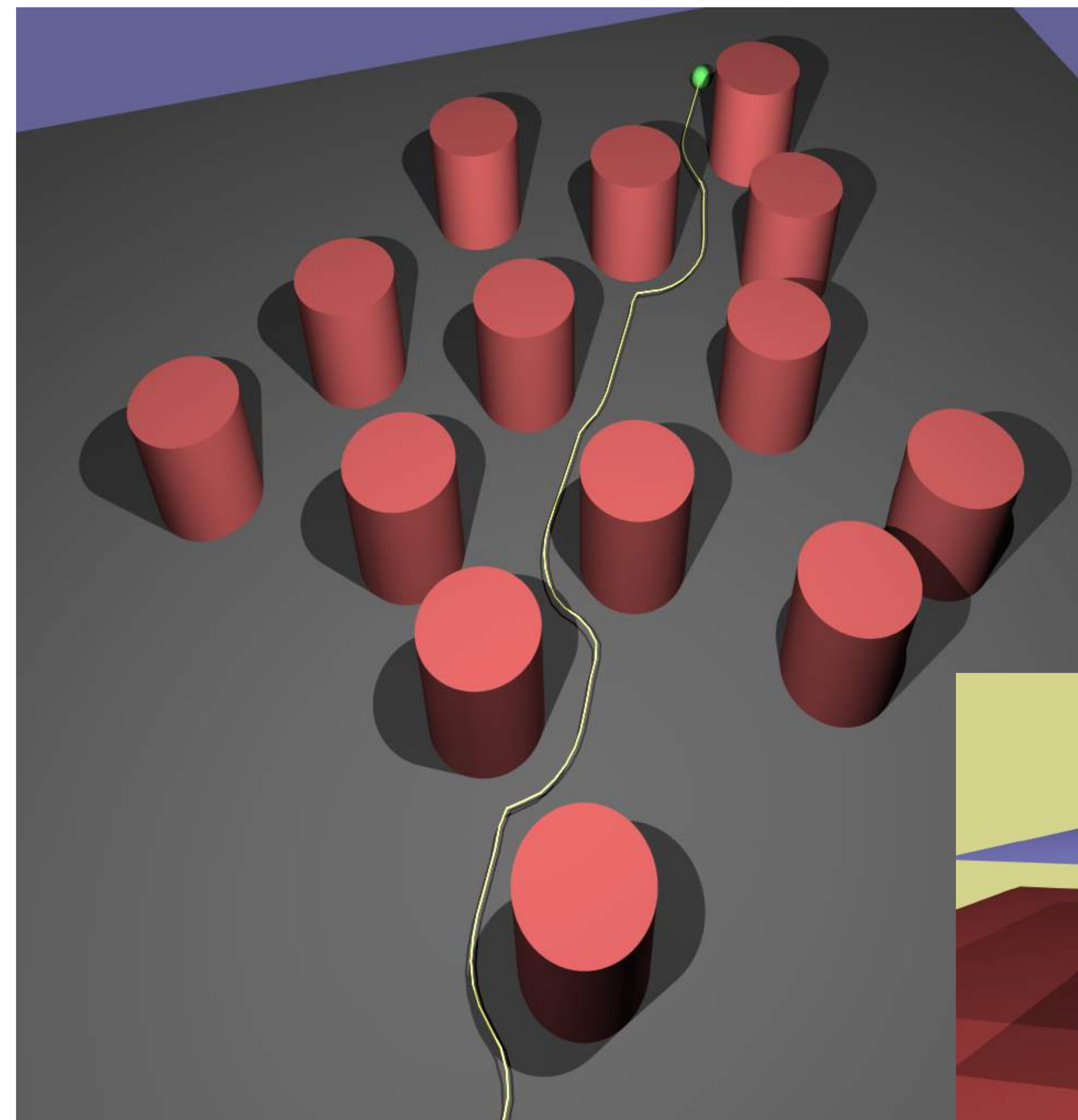
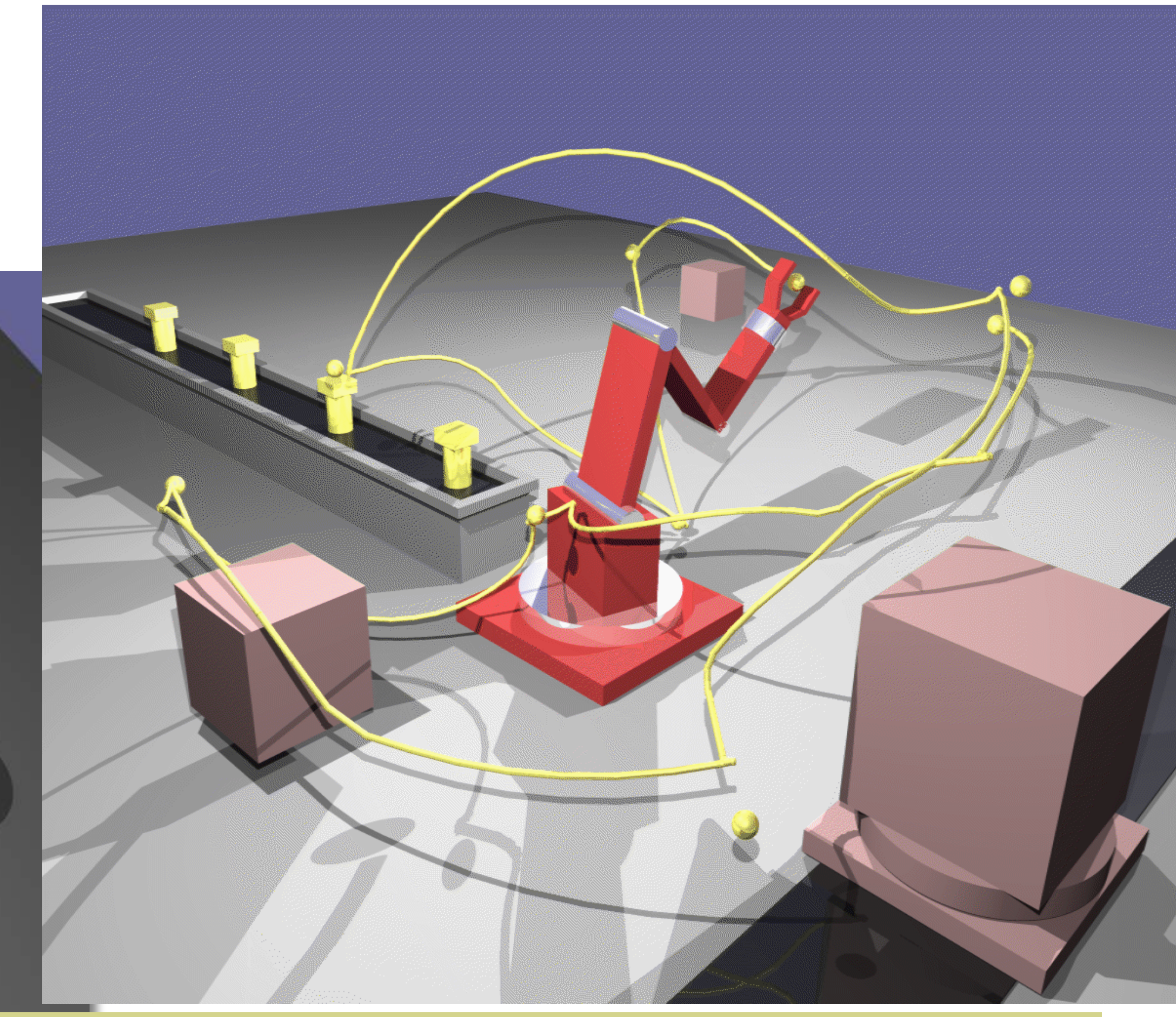
Energy Minimization for Animation

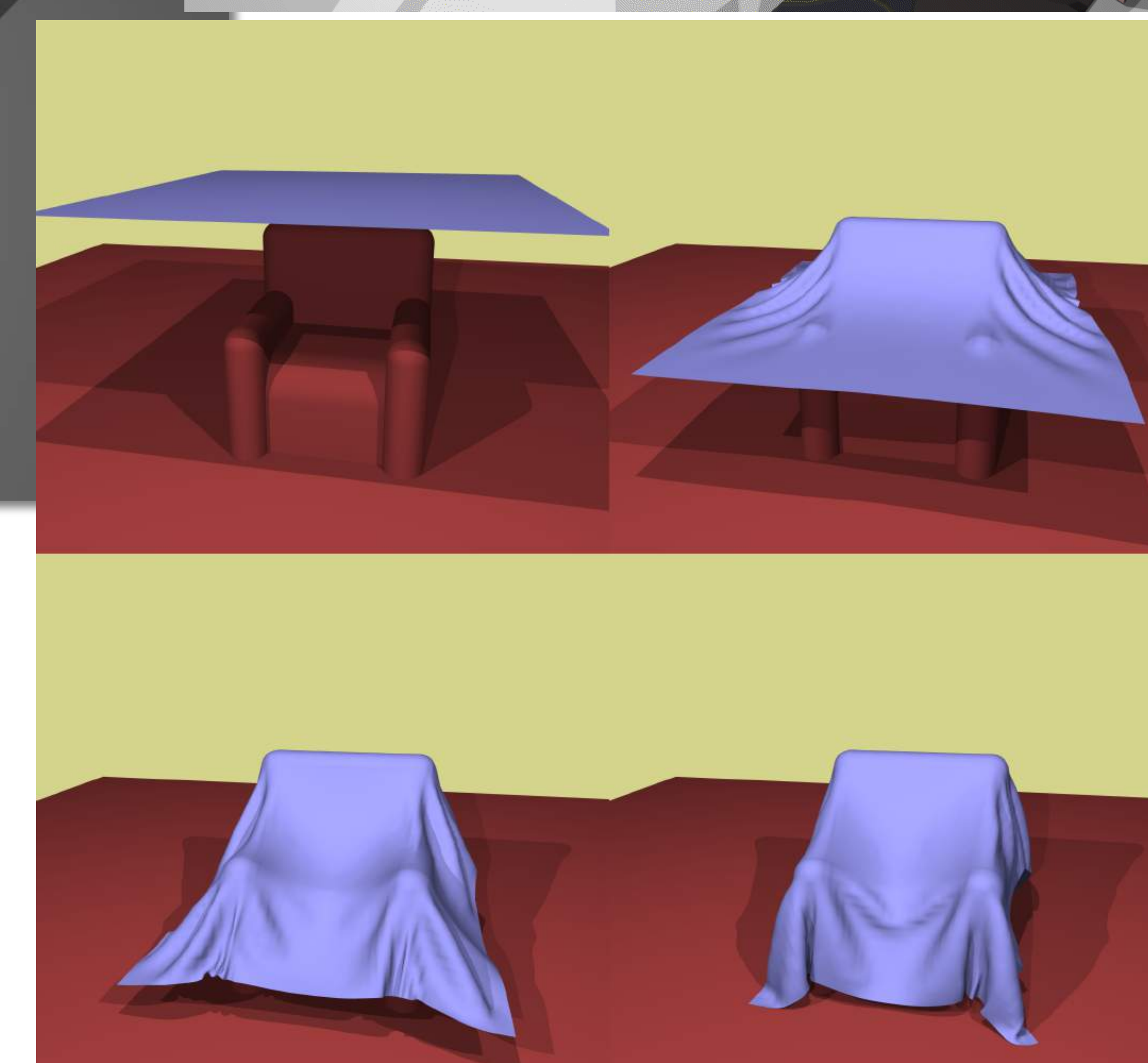
CSE 4280/5280

Animation by minimizing cost functions

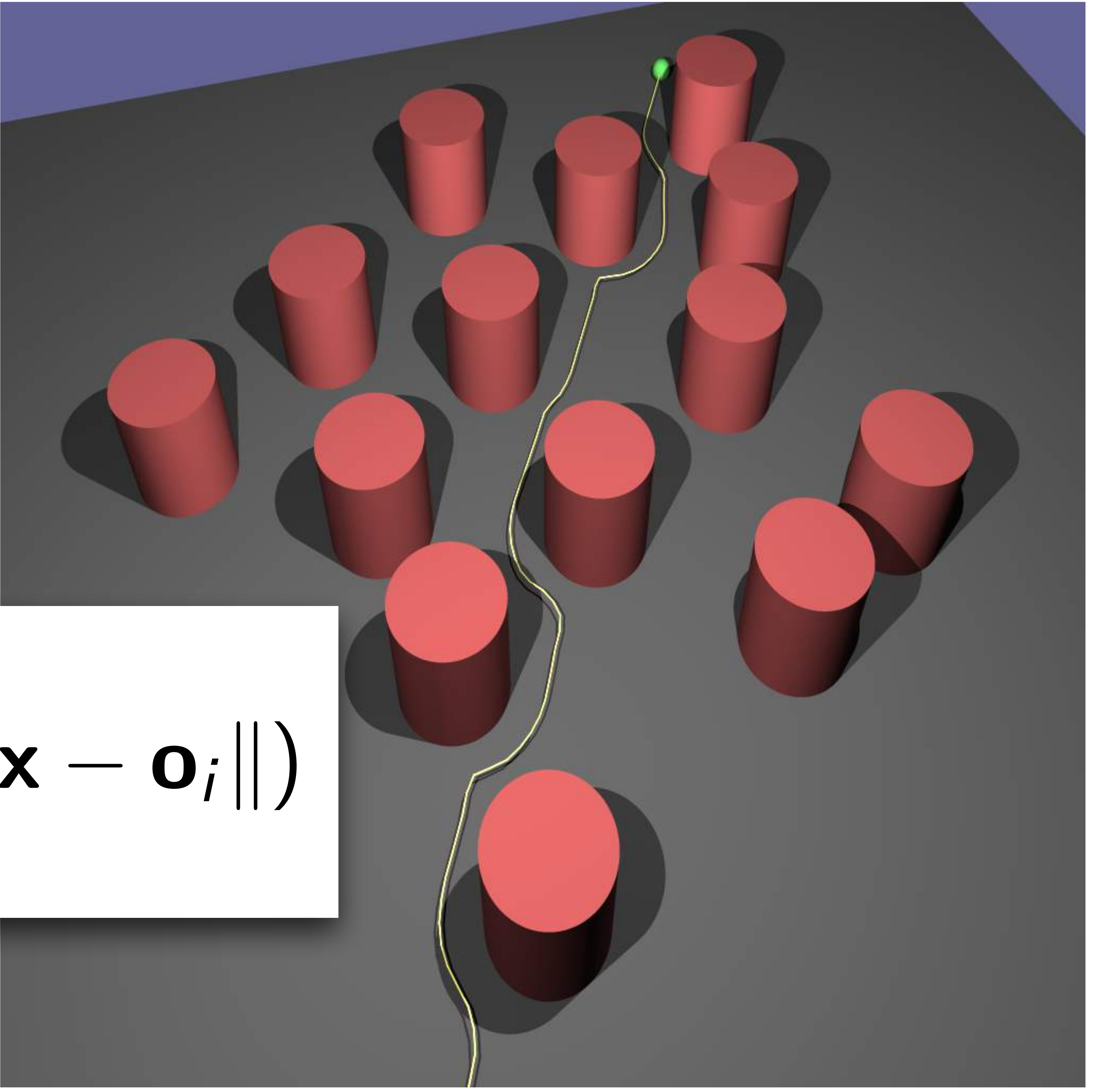
- Goal-oriented motion.
- We can add constraints. These constraints change the topography of the cost functions.
- Animation becomes a task of defining a function
- However, animator surrenders control over details to the algorithm.







Example 1: Single-particle motion



$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x} - \mathbf{o}_i\|\right)$$

Where:

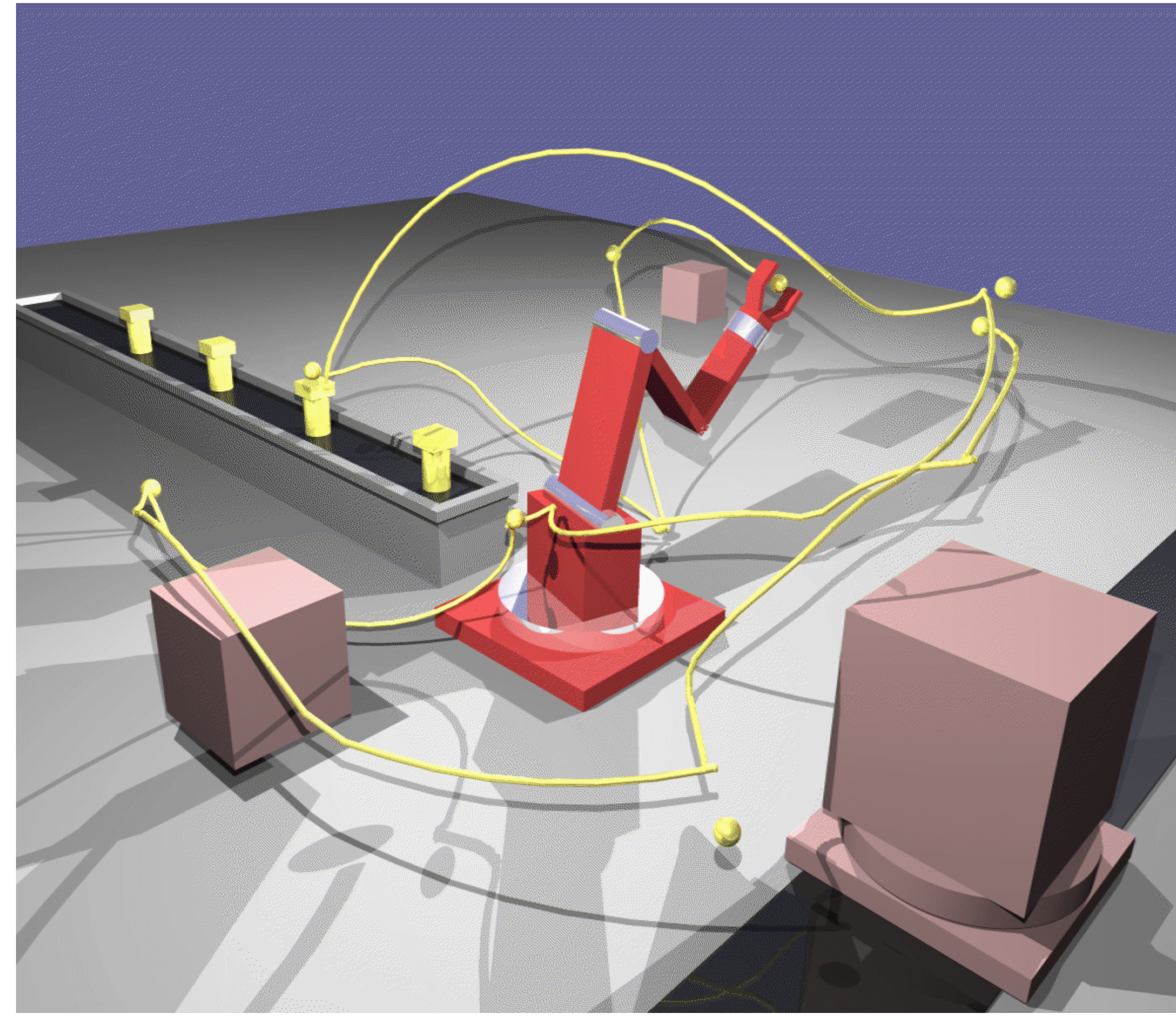
$\mathbf{x}$: Current location of the animated object

$\mathbf{g}$: Goal location

$\mathbf{o}_i$: Location of object i

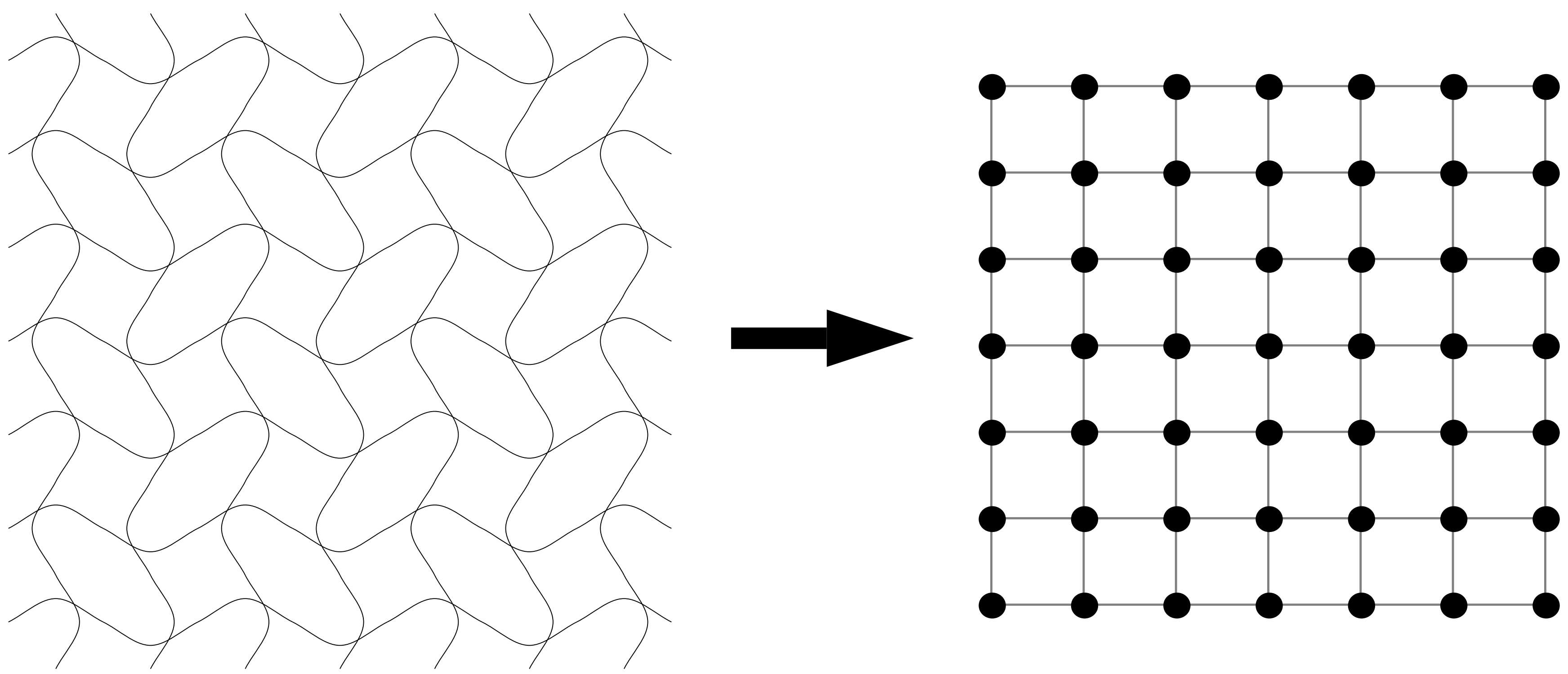
$\mathcal{F}$: Penalty field for collision avoidance

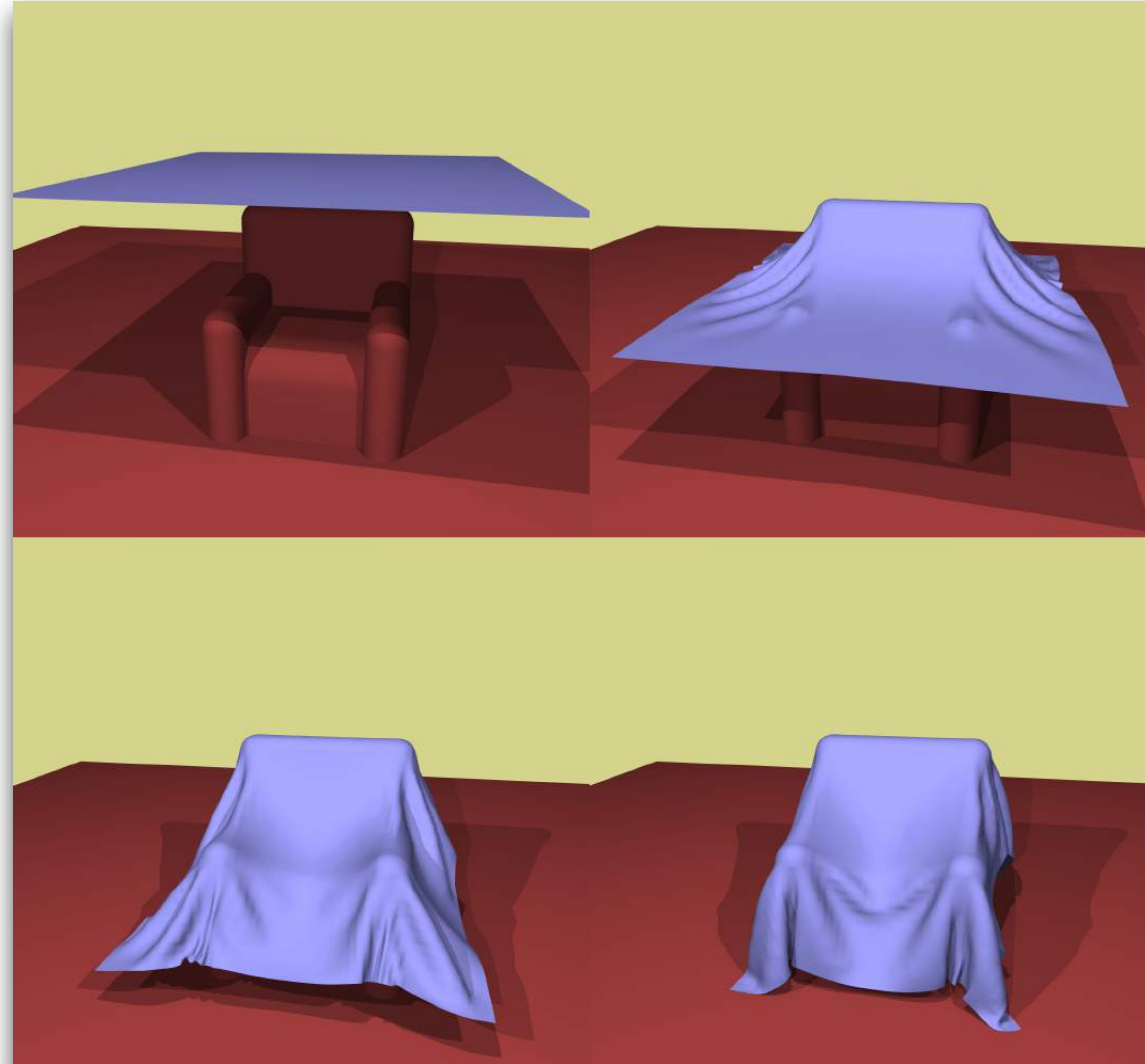
Example 2: Articulated motion



$$C_{\text{Reach}}\left(\phi_0,\phi_1,\phi_2\right) = \left\| P_{\text{tip}}\left(\phi_0,\phi_1,\phi_2\right) - \mathbf{g} \right\| + \sum_{i=1}^{n} \mathcal{F}\left(\left\| P_{\text{tip}}\left(\phi_0,\phi_1,\phi_2\right) - \mathbf{o}_i \right\|\right) + \sum_{i=0}^{2} \text{limit}\left(\phi_j\right).$$

Example 3: Cloth (elastic) motion

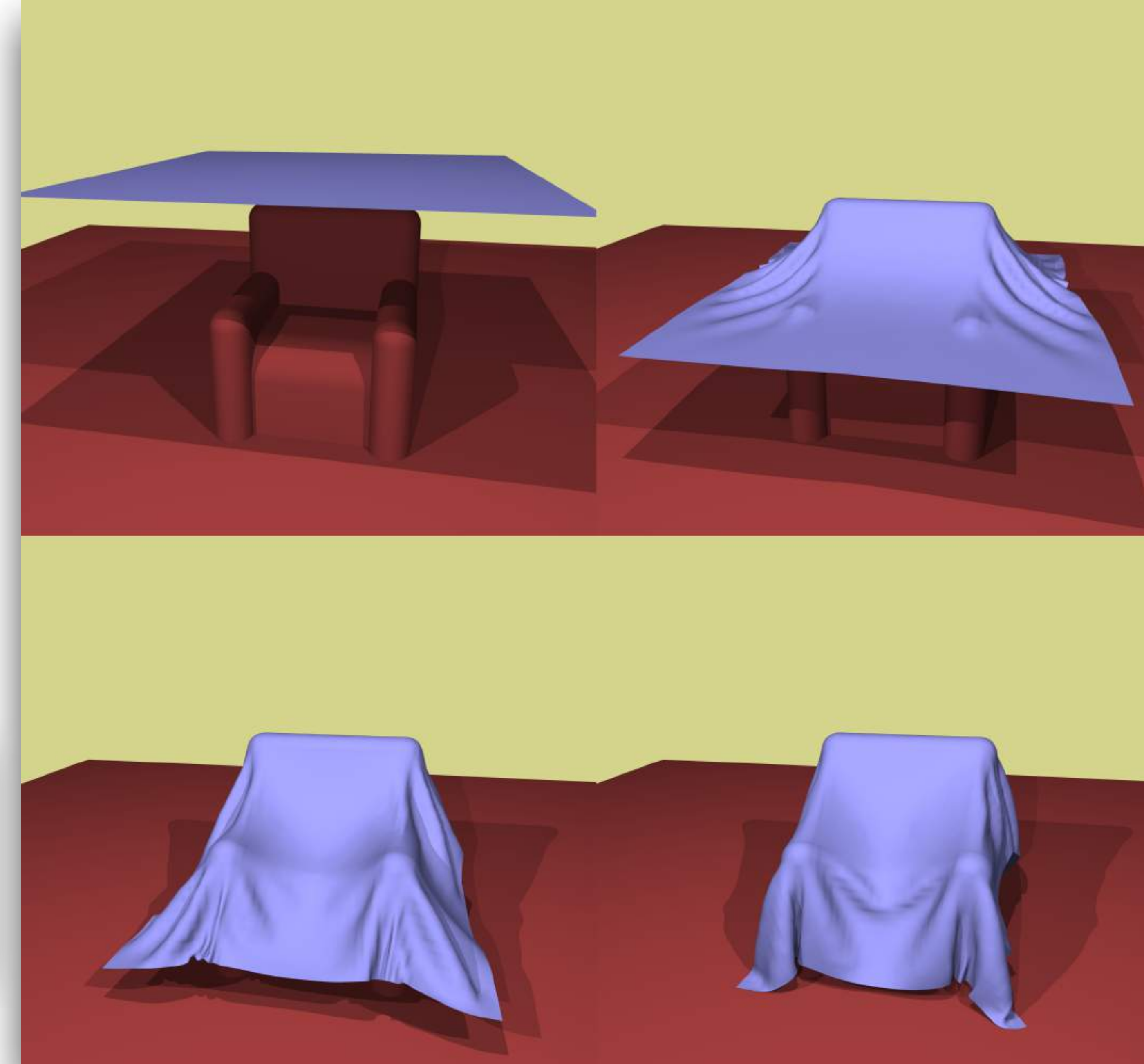




1. Model a plain weave as a particle grid
2. Define each "energy" term as a (e.g., repulsion or attraction)

$$U_i = U_{\mathrm{repel}_i} + U_{\mathrm{stretch}_i} + U_{\mathrm{bend}_i} + U_{\mathrm{trellis}_i} + U_{\mathrm{grav}_i}$$

Example 3: Cloth (elastic) motion



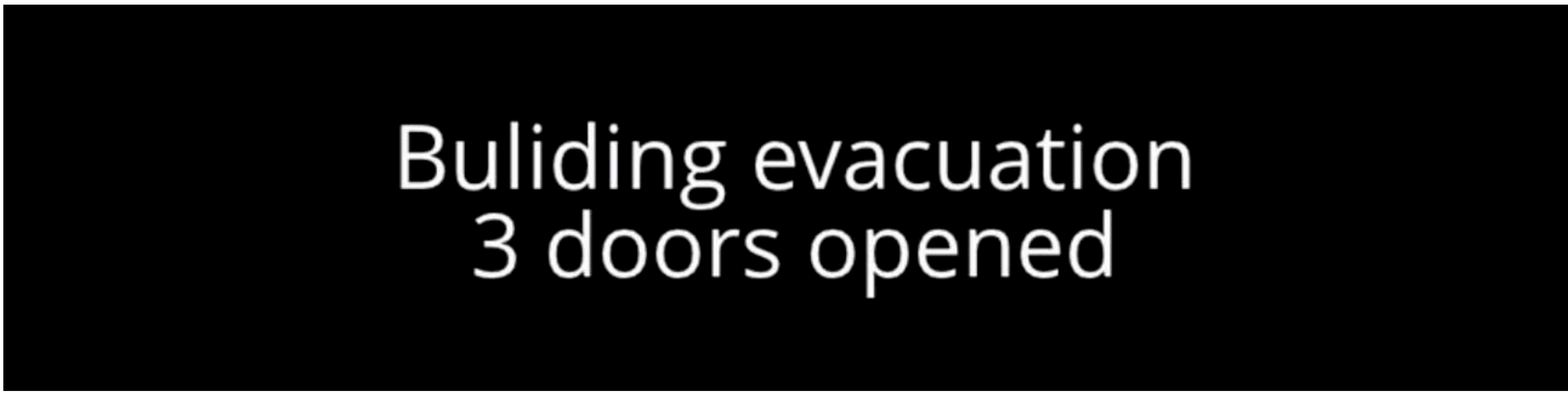
$$U_i = U_{\text{repel}_i} + U_{\text{stretch}_i} + U_{\text{bend}_i} + U_{\text{trellis}_i} + U_{\text{grav}_i}$$

Where:

- U_{repel_i}: Intra-particle repulsion energy. It keeps particles at a minimum distance. Helps prevent self-intersection of the cloth.
- U_{stretch_i}: Tension (energy) between a particle and its 4 neighbors.
- U_{bend_i}: Energy due to threads bending out of plane of local plane of the cloth.
- U_{trellis_i}: Energy due to bending around a thread crossing in the plane.
- U_{grav_i}: Potential energy due to gravity.

Example 4: Pedestrian simulation using Helbing's social-force model

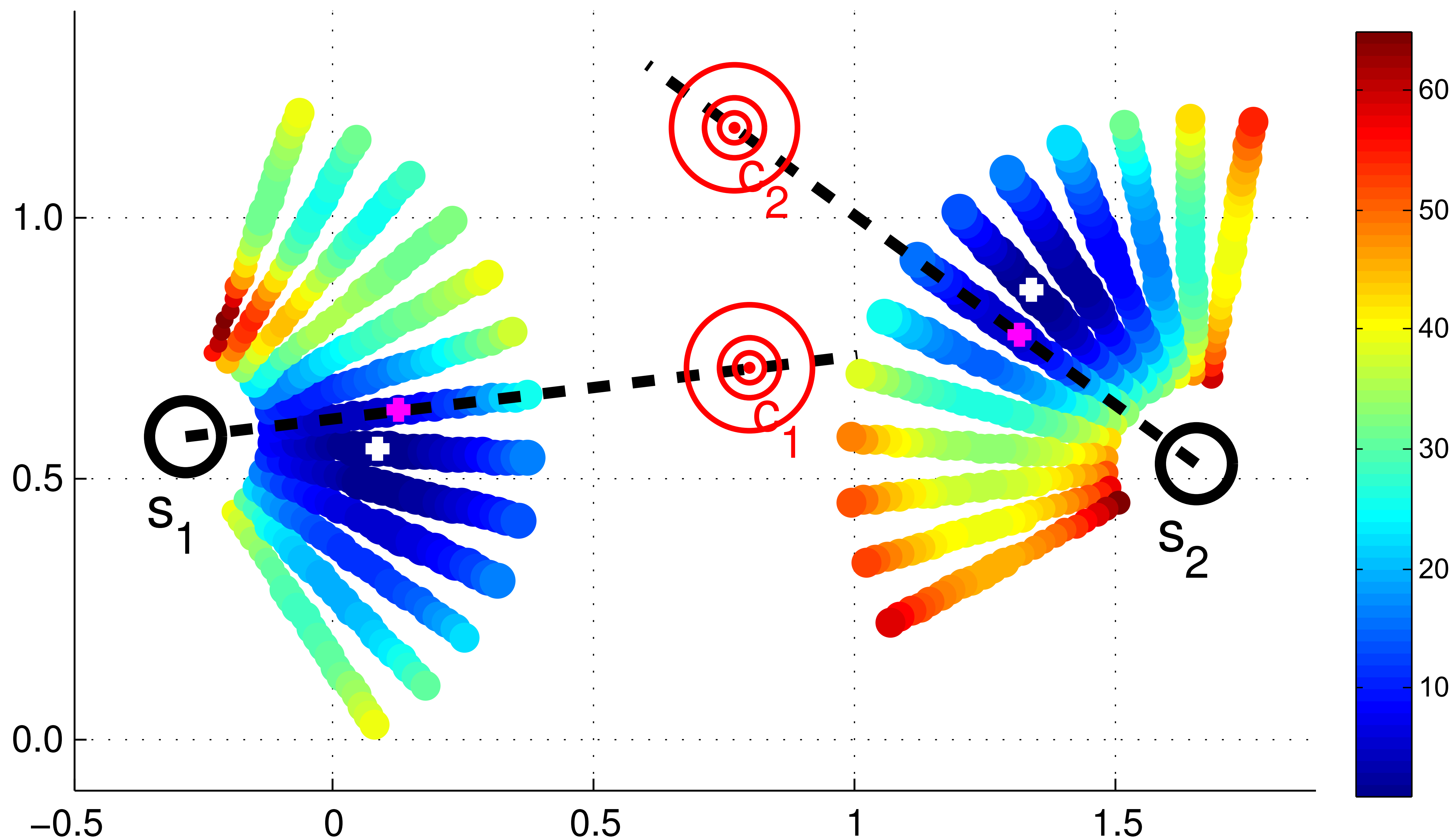
D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.



See video online here: https://youtu.be/Xcbc4ff0EY8

Social-force model

$$d_{ij}^2(t, \tilde{\mathbf{v}}_i) = ||\mathbf{p}_i + t\tilde{\mathbf{v}}_i - \mathbf{p}_j - t\mathbf{v}_j||^2$$



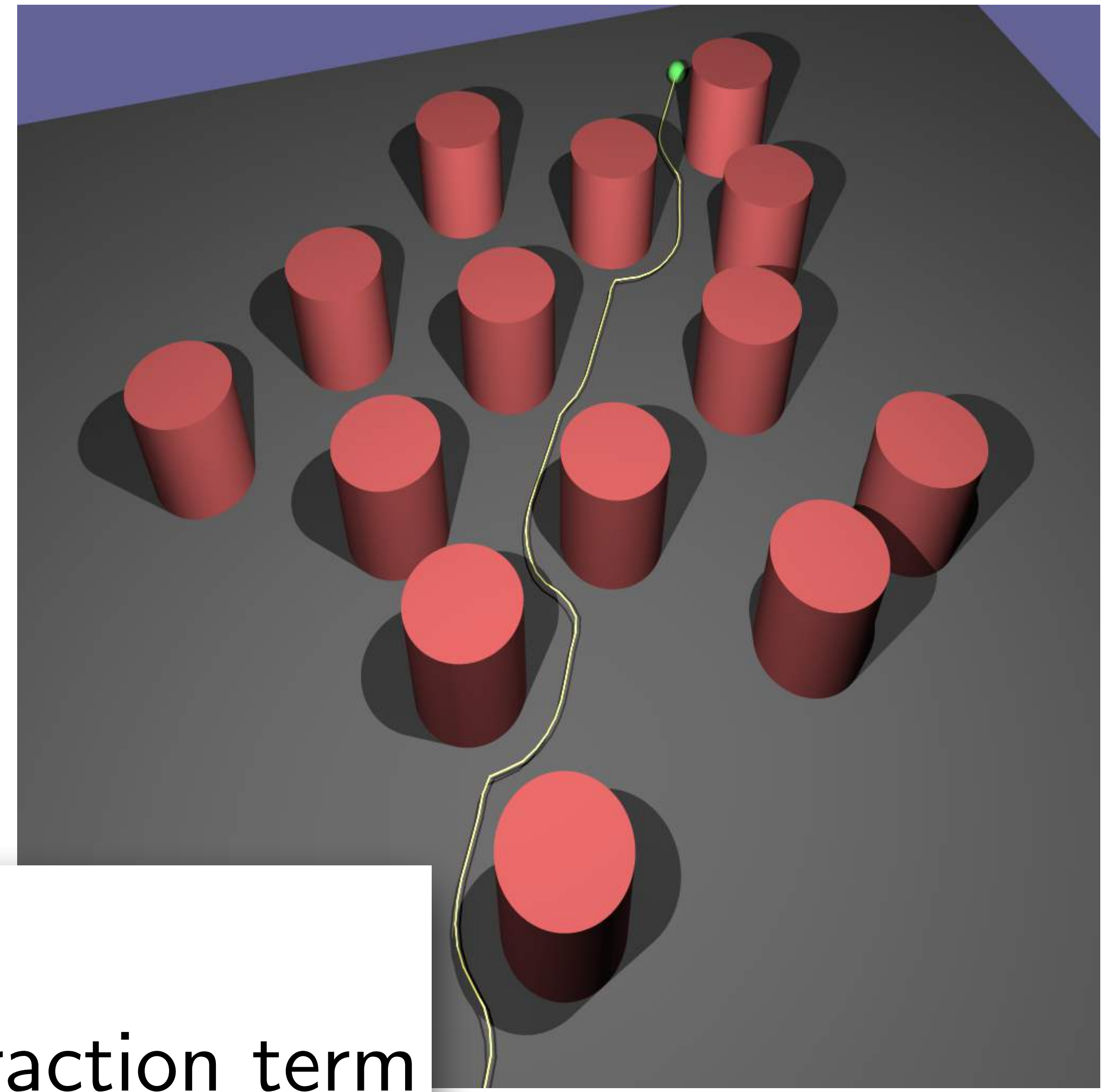
- Colors denote energies for different velocities.
- White dots mark the minima

Reference: http://vision.cse.psu.edu/courses/Tracking/vlpr12/PellegriniNeverWalkAlone.pdf

Details

Single-particle motion

$$C_{\mathrm{PathPlan}}\left(\mathbf{x}\right) = \boxed{\left\|\mathbf{x} - \mathbf{g}\right\|} + \sum_{i=1}^{n} \mathcal{F}\left(\left\|\mathbf{x} - \mathbf{o}_i\right\|\right)$$



Attraction term: distance to goal

Let $\mathbf{x} = (x, y, z)^{\mathsf{T}}$ be the current location of the particle and $\mathbf{g} = (u, v, w)^{\mathsf{T}}$ be the particle's goal location. Then, the attraction term is given by:

$$\|\mathbf{x} - \mathbf{g}\| = \sqrt{(x-u)^2 + (y-v)^2 + (z-w)^2},$$

which we can also write, by using the dot product, as:

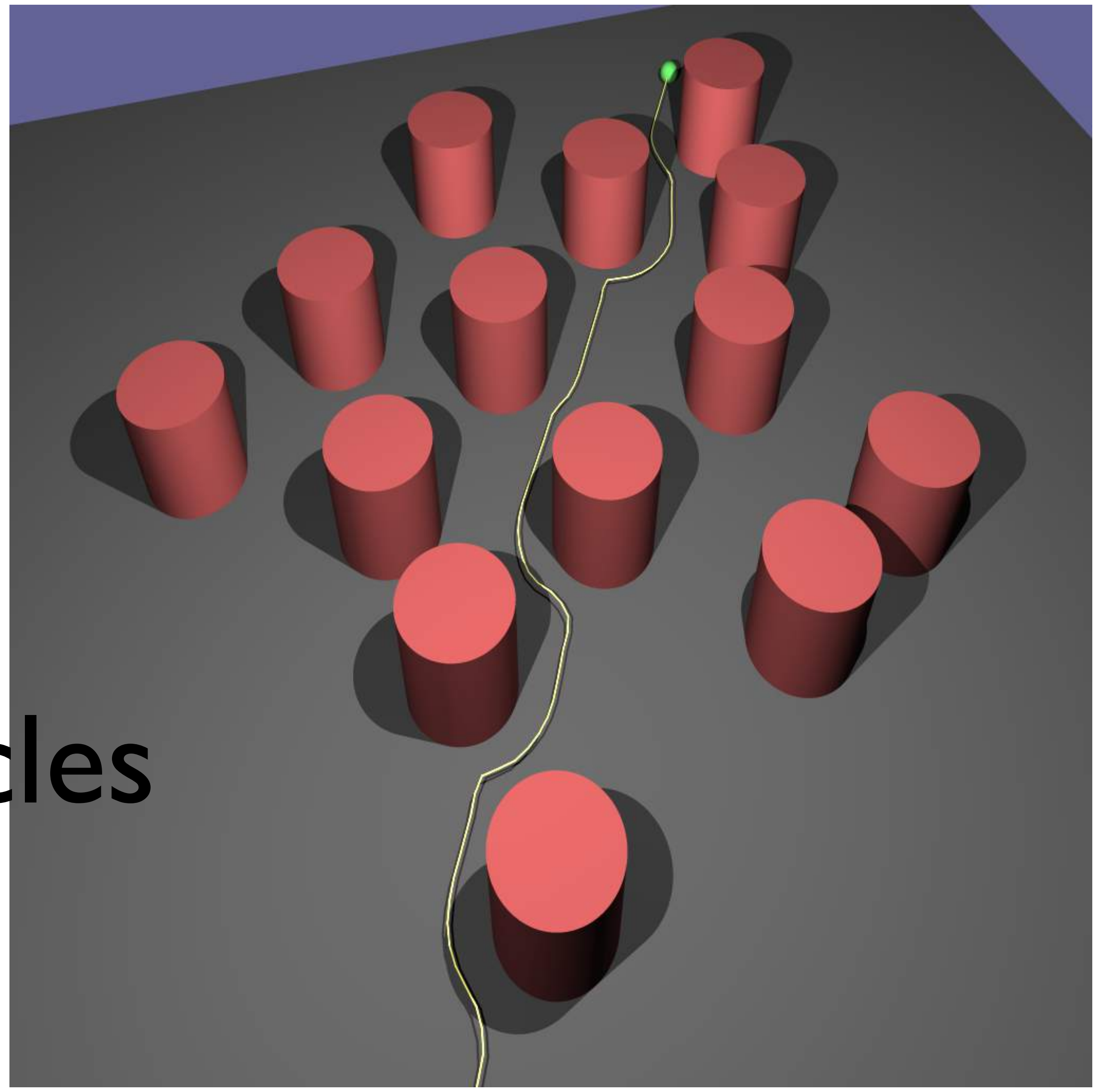
$$\|\mathbf{x} - \mathbf{g}\| = \sqrt{(\mathbf{x} - \mathbf{g}) \cdot (\mathbf{x} - \mathbf{g})},$$

or, in matrix notation:

$$\|\mathbf{x} - \mathbf{g}\| = \sqrt{(\mathbf{x} - \mathbf{g})^{\mathsf{T}} (\mathbf{x} - \mathbf{g})}.$$

Single-particle motion

$$C_{\mathrm{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \boxed{\mathcal{F}(\|\mathbf{x} - \mathbf{o}_i\|)}$$



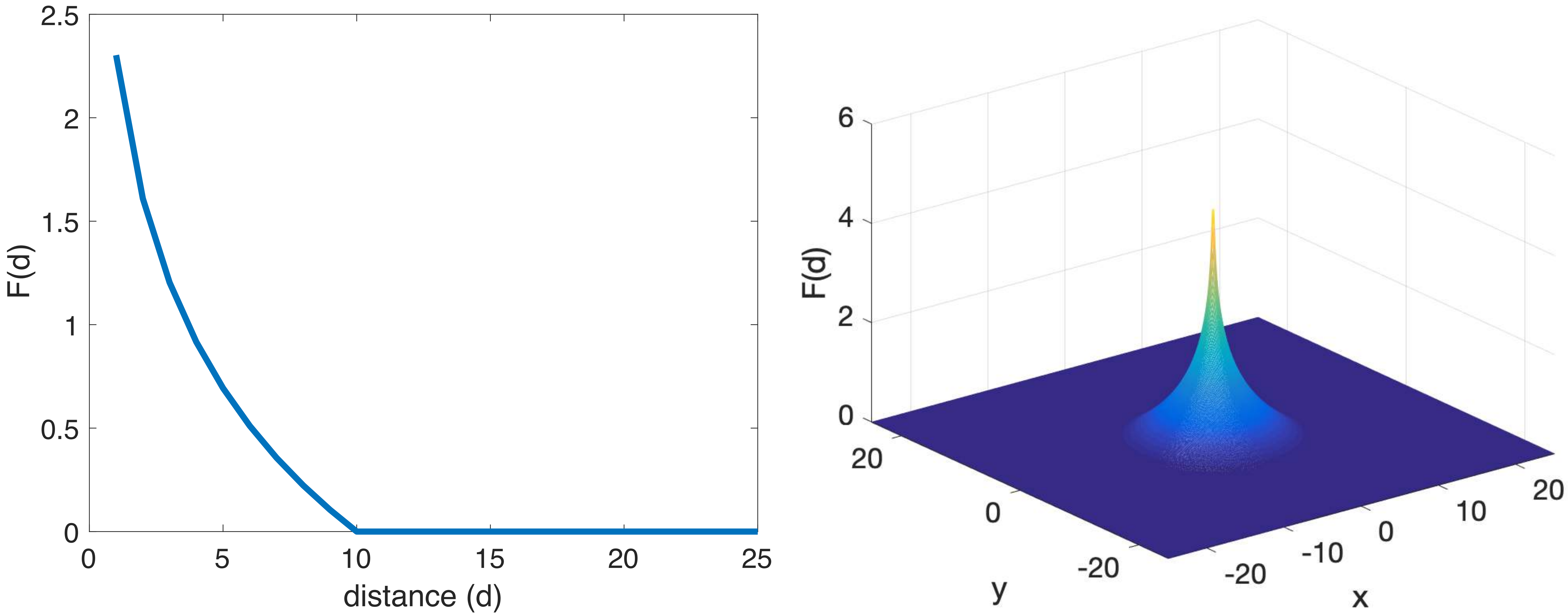
Repulsion term: combined distance to obstacles

$$\mathcal{F}(d) = \begin{cases} \ln(R/d), & 0 < d \leq R, \\ 0, & d > R. \end{cases}$$

R: is the radius of the obstacle

Single-particle motion

Repulsion term: combined distance to obstacles



(a) (b)

Figure 2: Penalty field function. (a) One-dimensional plot of the penalty function. Penalty value increases as particle approaches the obstacle. (b) 2-D representation of the field function for $d = \sqrt{x^2 + y^2}$, i.e., distance from any point (x, y) to the origin.

Single-particle motion: **gradient descent**

$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x}-\mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x}-\mathbf{o}_i\|\right) \tag{1}$$

Minimize (1) by using the Gradient Descent algorithm:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla C\left(\mathbf{x}_n\right), \quad n \geq 0,$$

where:

$$\nabla C = \left[\frac{\partial C}{\partial x} \quad \frac{\partial C}{\partial y}\right]^{\mathsf{T}}$$

Example with two obstacles

Cost function

$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x} - \mathbf{o}_i\|\right) \quad (1)$$

Goal location

$$\mathbf{g} = (70, 70)^{\mathsf{T}}$$

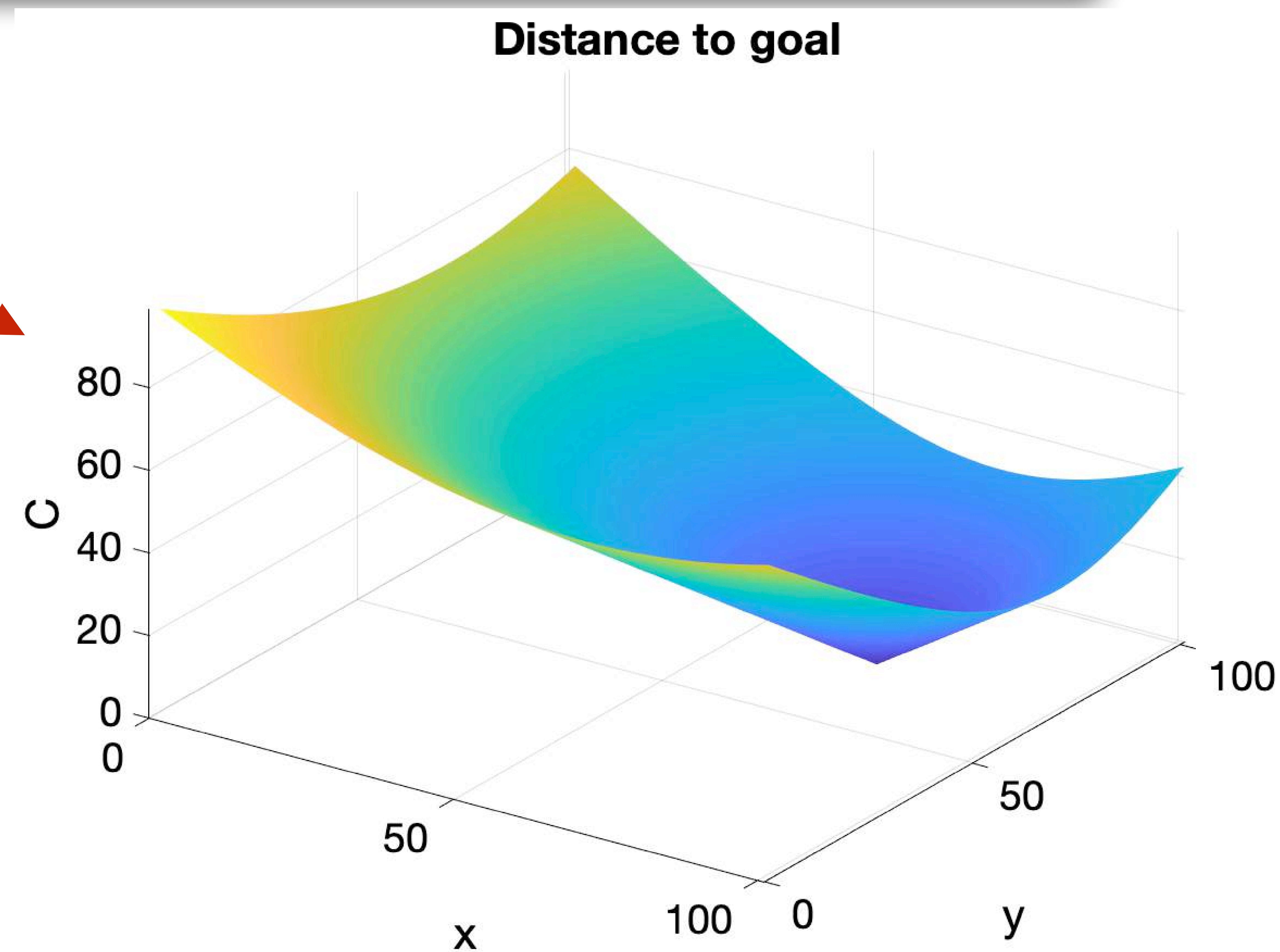
Obstacle locations

$$\mathbf{o}_1 = (40, 40)^{\mathsf{T}}$$

$$\mathbf{o}_2 = (70, 30)^{\mathsf{T}}$$

Obstacle radius

$$R = 20$$



Example with two obstacles

Cost function

$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}(\|\mathbf{x} - \mathbf{o}_i\|) \quad (1)$$

Goal location

$$\mathbf{g} = (70, 70)^{\mathsf{T}}$$

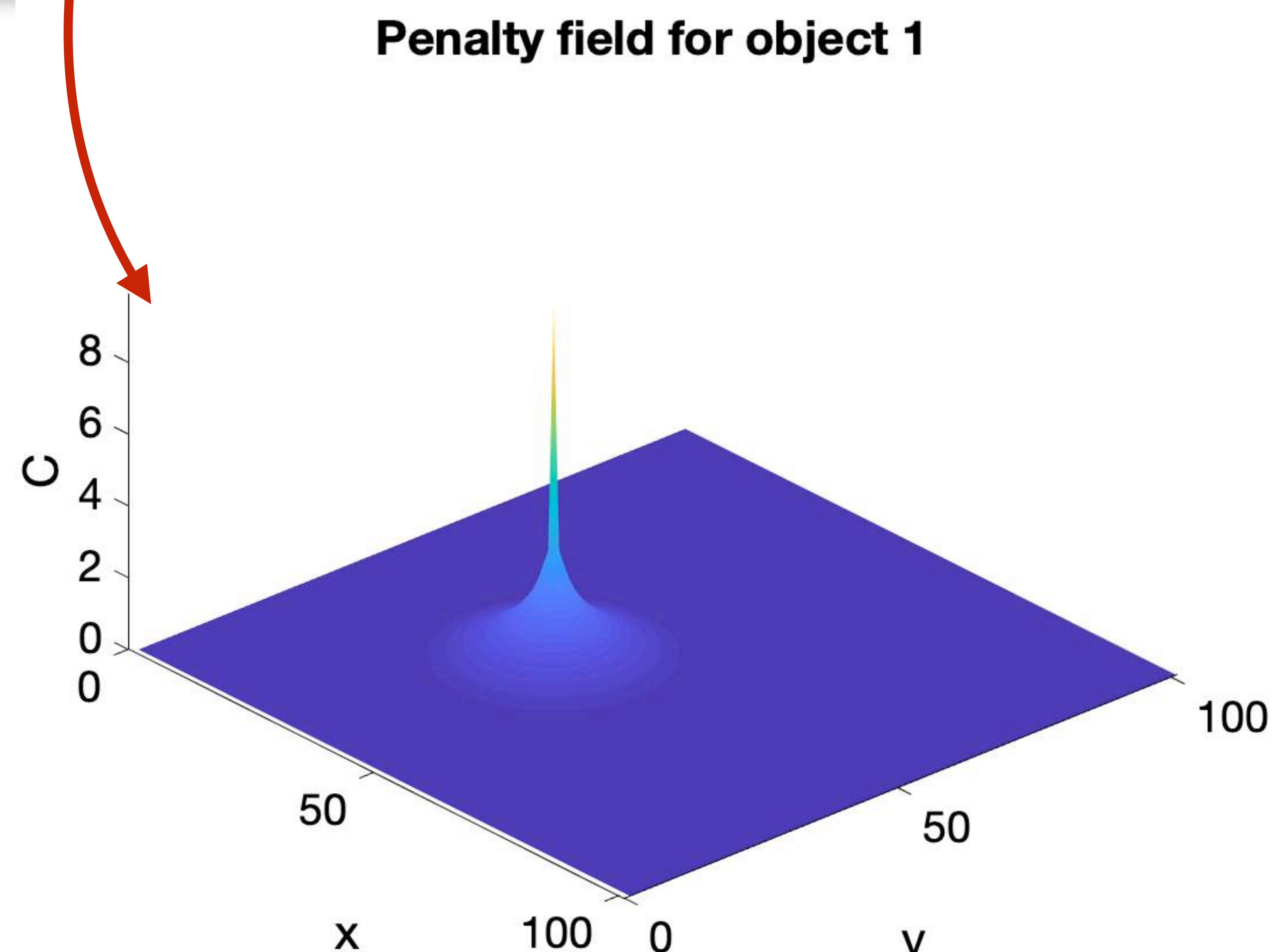
Obstacle locations

$$\mathbf{o}_1 = (40, 40)^{\mathsf{T}}$$

$$\mathbf{o}_2 = (70, 30)^{\mathsf{T}}$$

Obstacle radius

$$R = 20$$



Example with two obstacles

Cost function

$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x} - \mathbf{o}_i\|\right) \quad (1)$$

Goal location

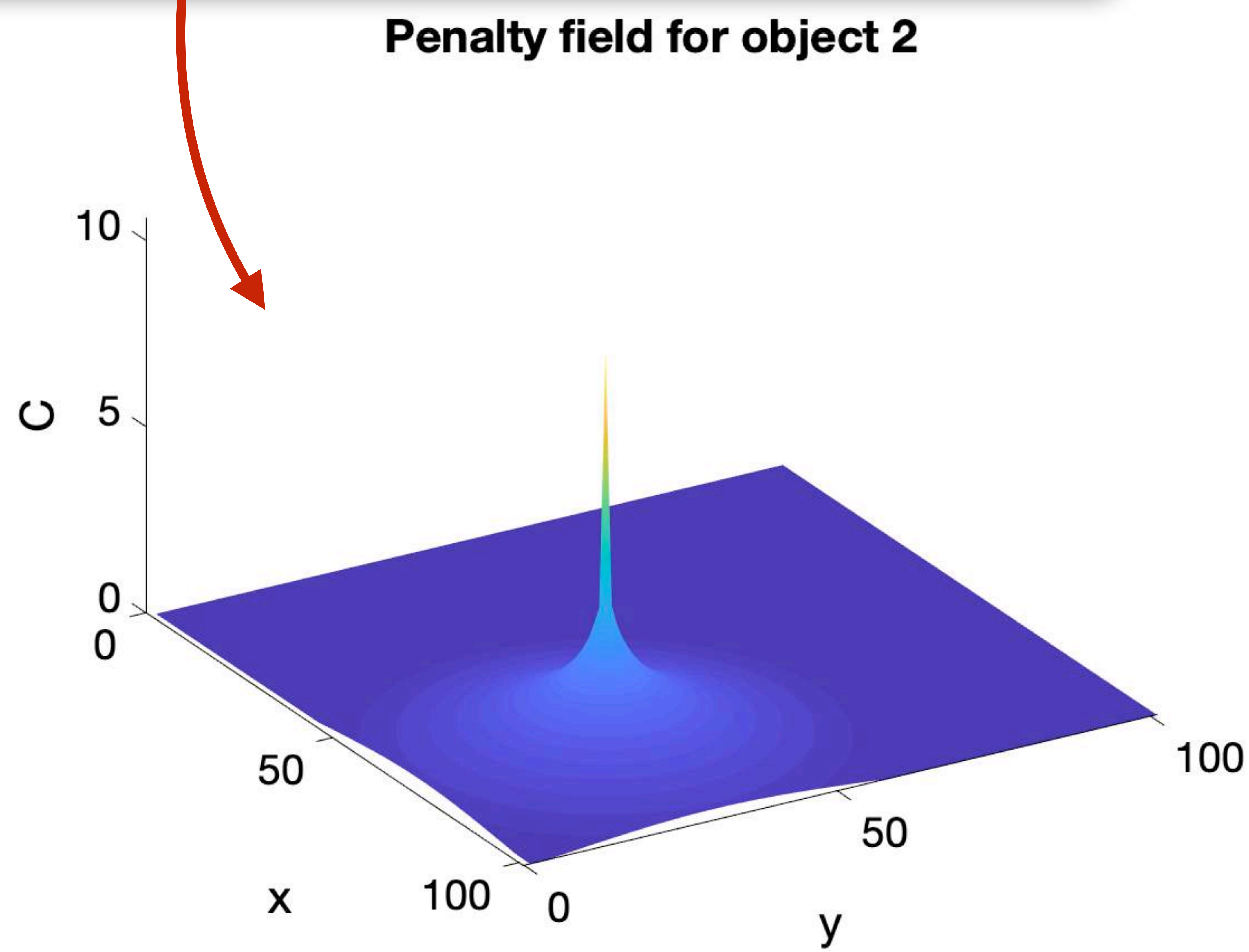
$$\mathbf{g} = (70, 70)^{\mathsf{T}}$$

Obstacle locations

$$\mathbf{o}_1 = (40, 40)^{\mathsf{T}}$$
$$\mathbf{o}_2 = (70, 30)^{\mathsf{T}}$$

Obstacle radius

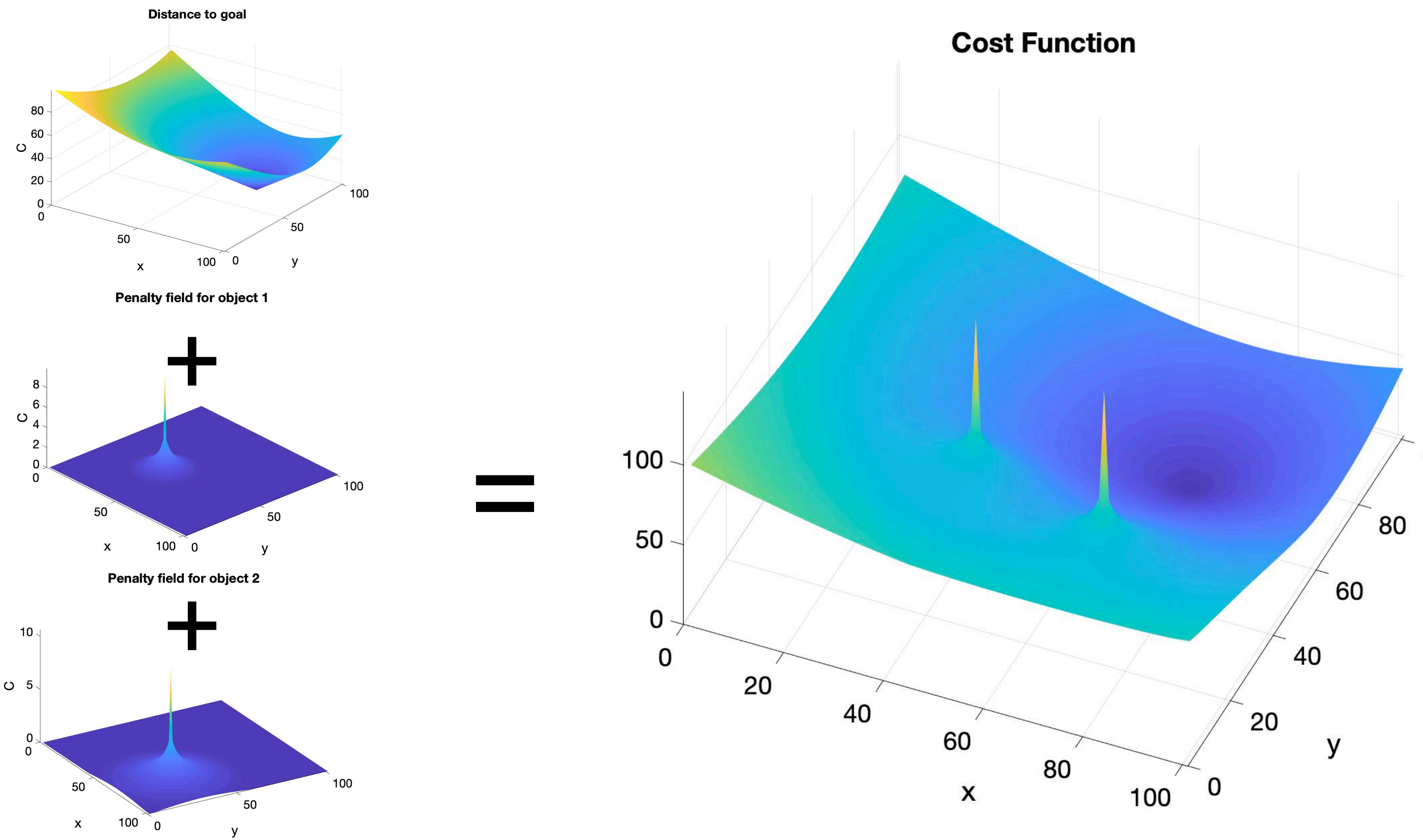
$$R = 20$$



Example with two obstacles

Cost function

$$C_{\mathrm{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x} - \mathbf{o}_i\|\right) \quad (1)$$



Single-particle motion: **gradient descent**

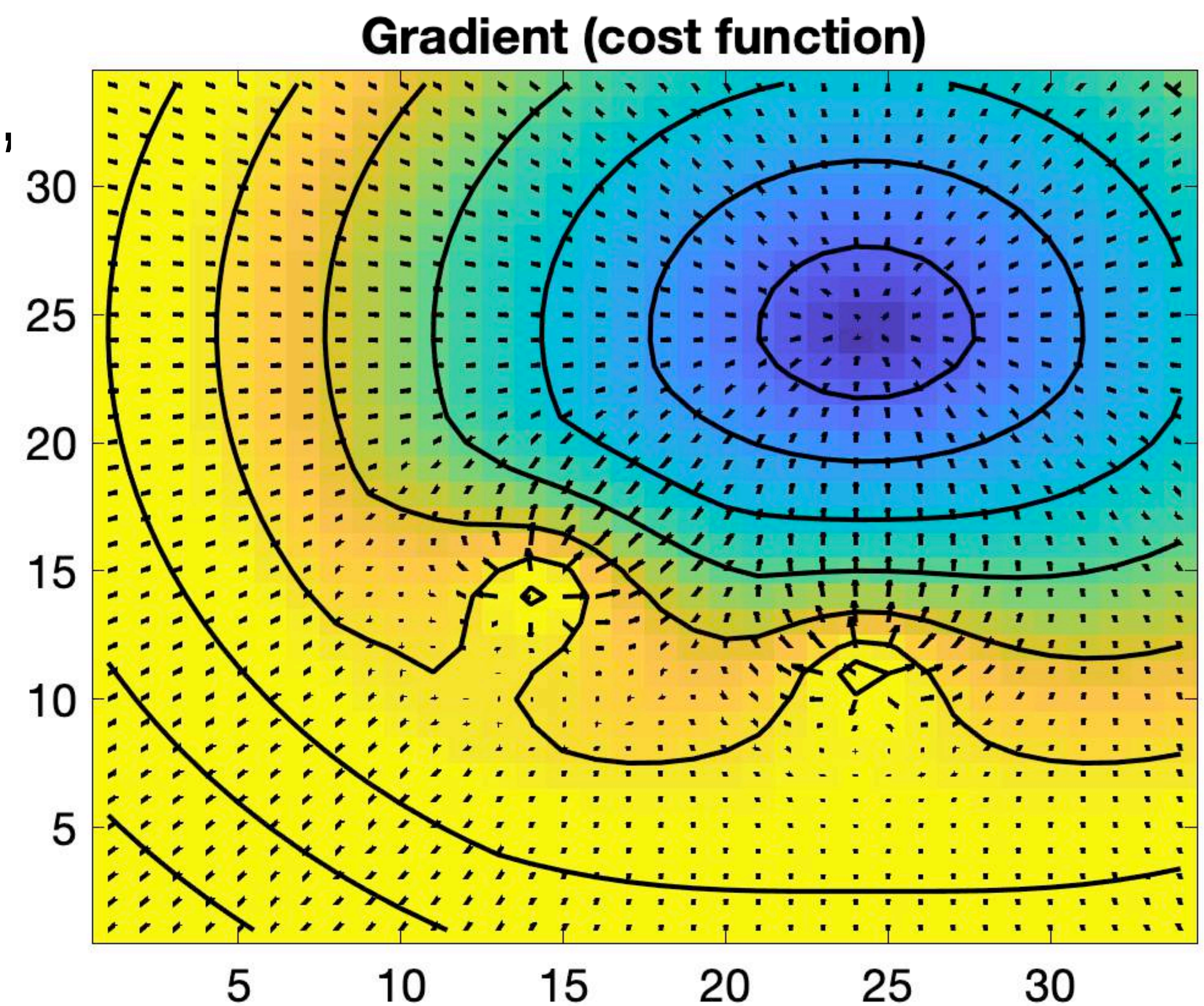
$$C_{\text{PathPlan}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}\| + \sum_{i=1}^{n} \mathcal{F}\left(\|\mathbf{x} - \mathbf{o}_i\|\right) \quad (1)$$

Minimize (1) by using the Gradient Descent algorithm:

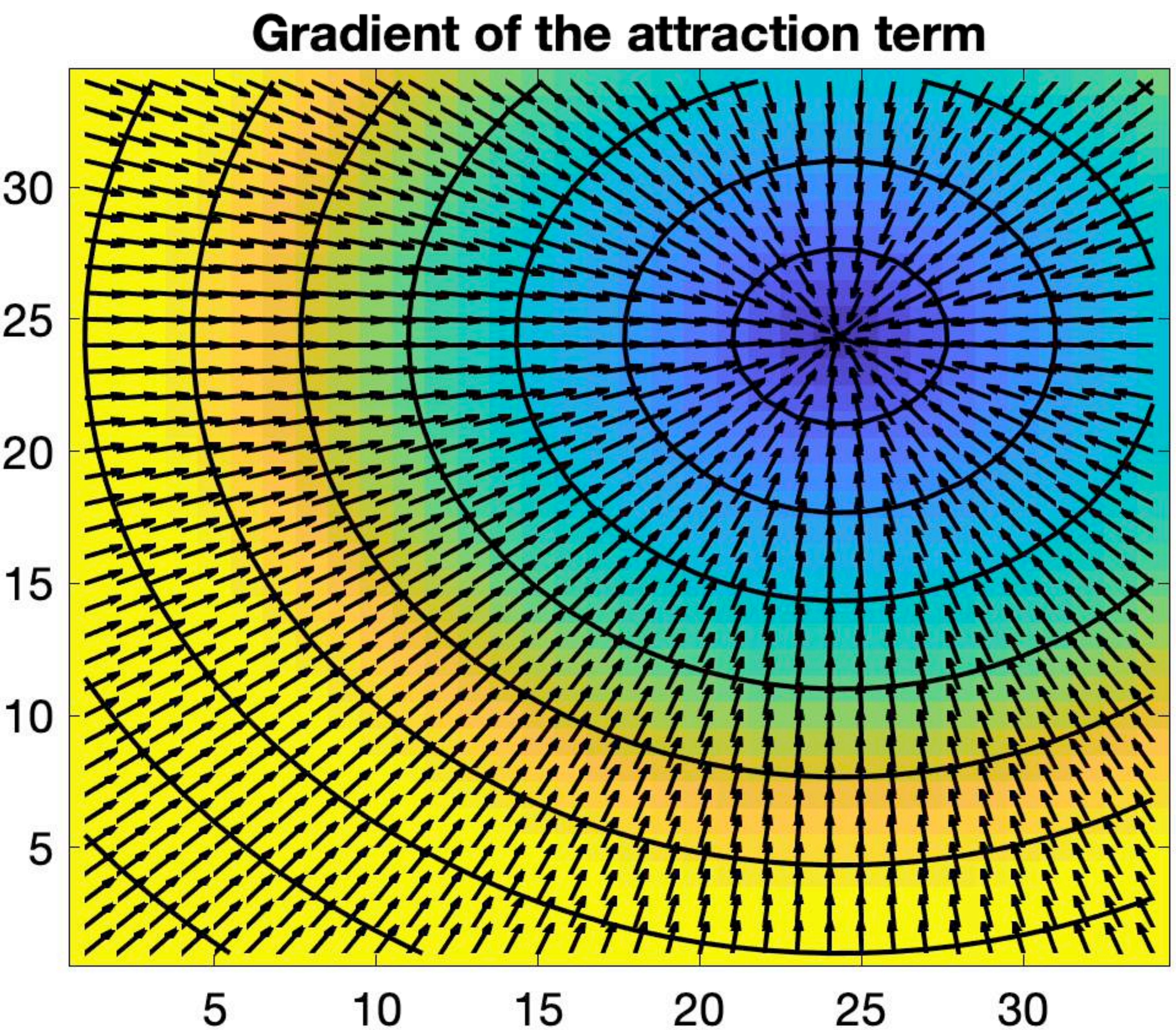
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla C(\mathbf{x}_n), \quad n \geq 0,$$

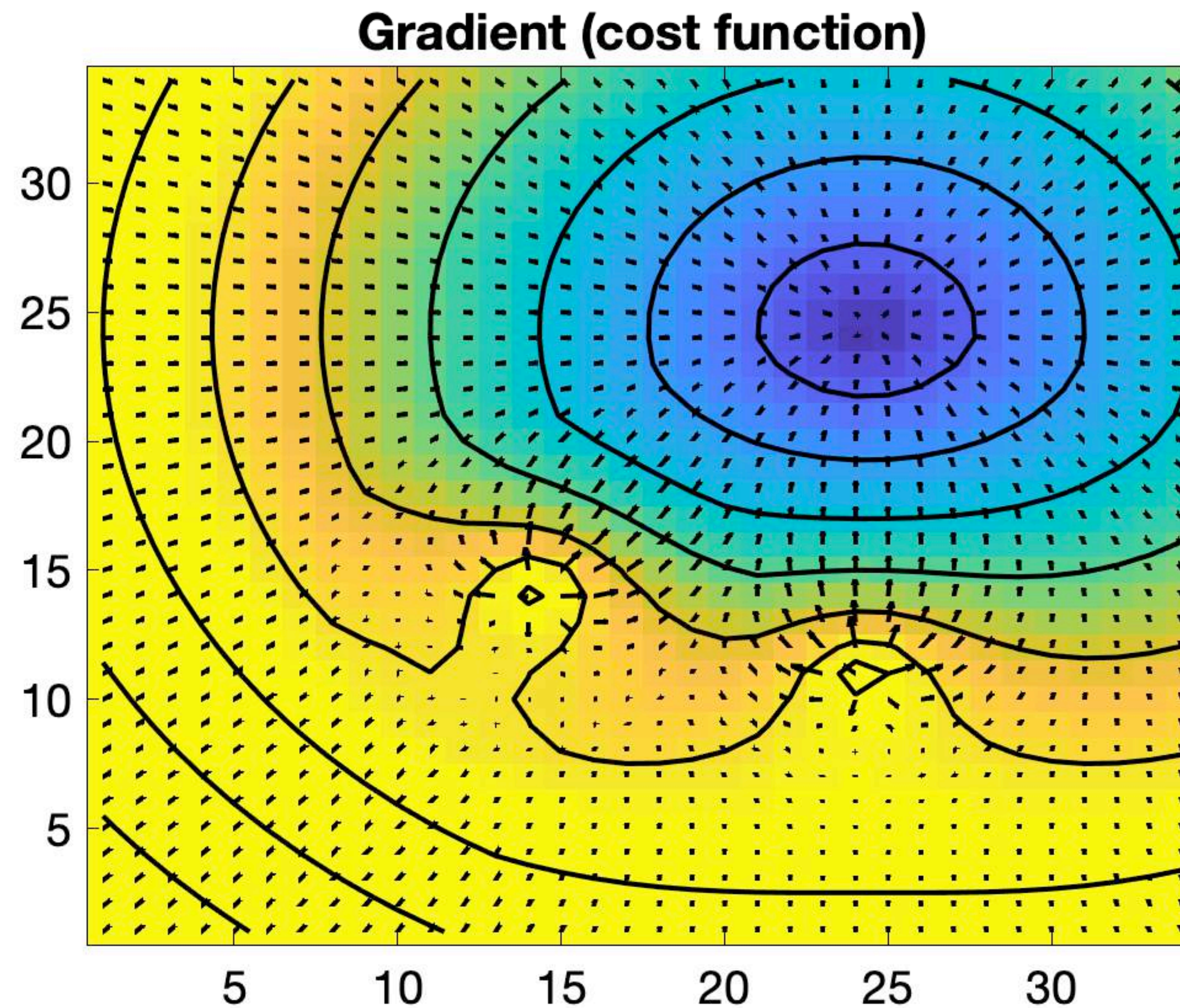
where:

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial x} & \frac{\partial C}{\partial y} \end{bmatrix}^{\mathrm{T}}$$



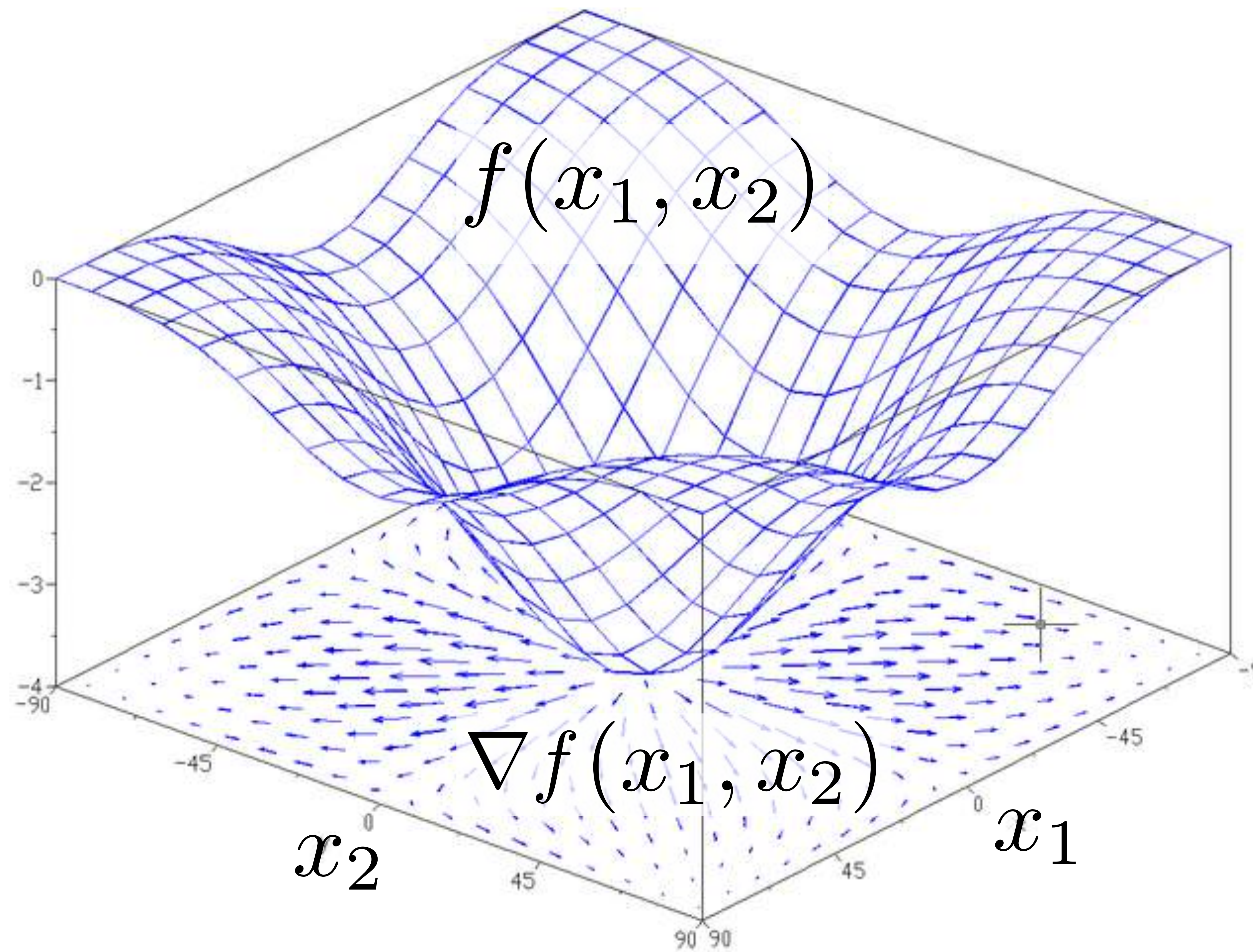
Example with two obstacles: gradients





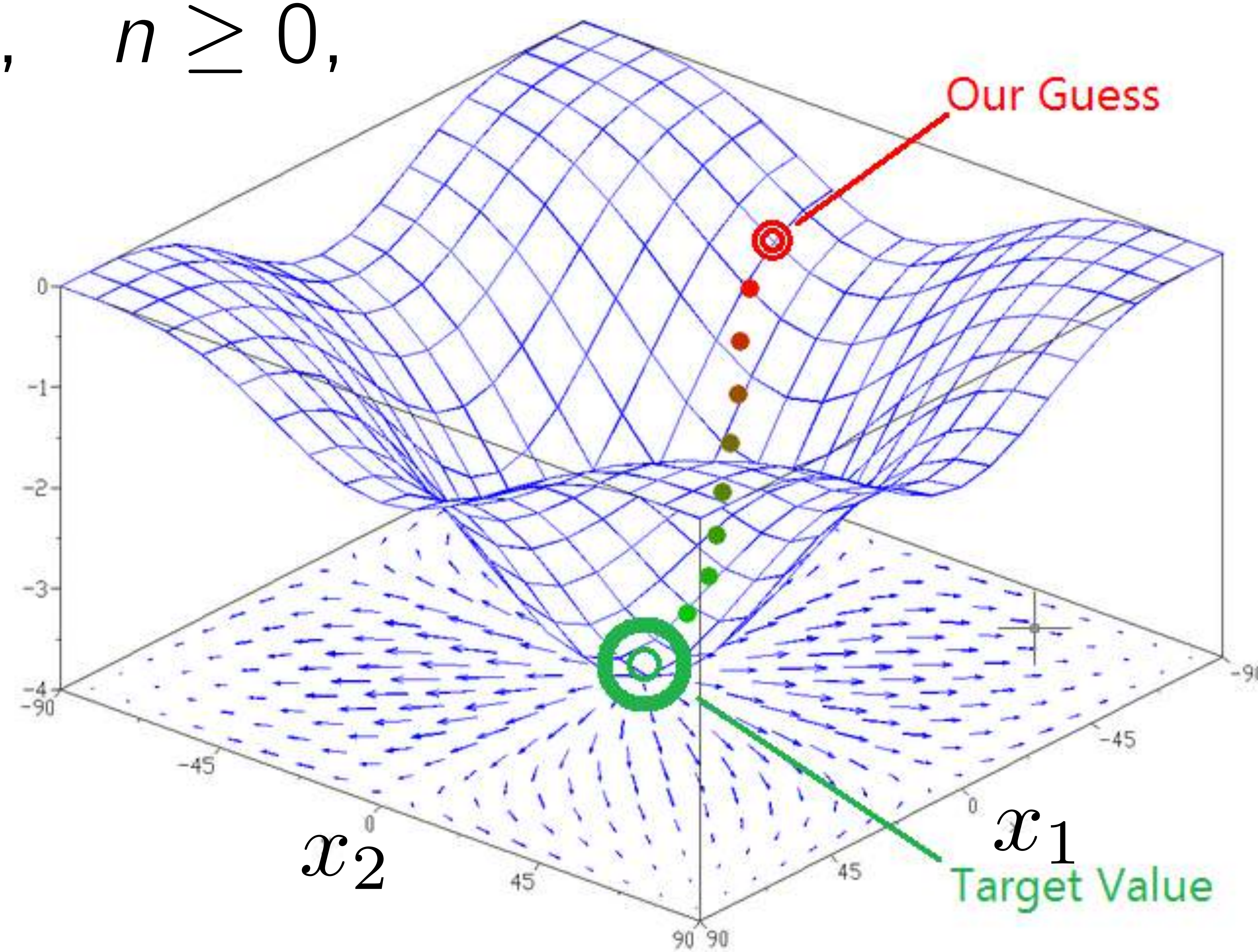
Single-particle motion: **gradient descent**

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}} = \nabla f = \frac{\partial f}{\partial\left(x_1, x_2, \ldots, x_N\right)} = \left[\frac{\partial f}{\partial x_1} \ \frac{\partial f}{\partial x_2} \ \cdots \ \frac{\partial f}{\partial x_N}\right]^{\mathsf{T}}$$



Single-particle motion: **gradient descent**

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla f(\mathbf{x}_n), \quad n \geq 0,$$



Single-particle motion: **gradient descent**

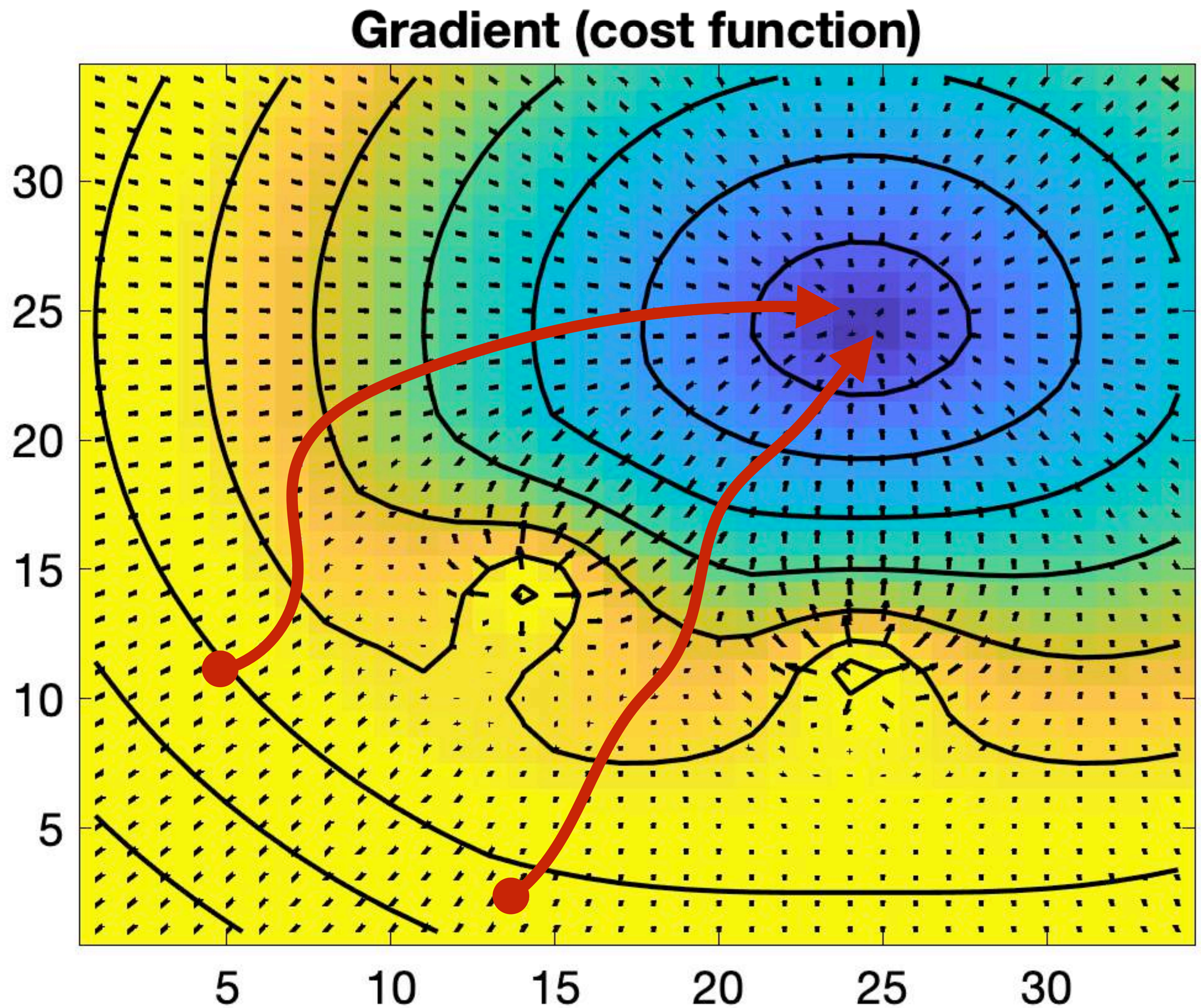
Algorithm 1 Gradient descent (scalar function of a single scalar variable)

$x_0 \leftarrow$ starting value
$f_0 \leftarrow f\,(x_0)$ $\triangleright$ Evaluate f at x_0
while $f_n \neq g$ **do**
    $s_i \leftarrow \frac{\mathrm{d}f}{\mathrm{d}x}\,(x_i)$ $\triangleright$ Compute slope
    $x_{i+1} \leftarrow x_i + \beta\,(g - f_i)\,\frac{1}{s_i}$ $\triangleright$ Take a step along Δx
    $f_{i+1} \leftarrow f\,(x_{i+1})$ $\triangleright$ Evaluate f at new x_{i+1}
end while

Single-particle motion: **gradient descent**

Minimize (1) by using the Gradient Descent algorithm:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla C\left(\mathbf{x}_n\right), \quad n \geq 0, \qquad \text{where: } \nabla C = \begin{bmatrix} \frac{\partial C}{\partial x} & \frac{\partial C}{\partial y} \end{bmatrix}^{\mathrm{T}}$$

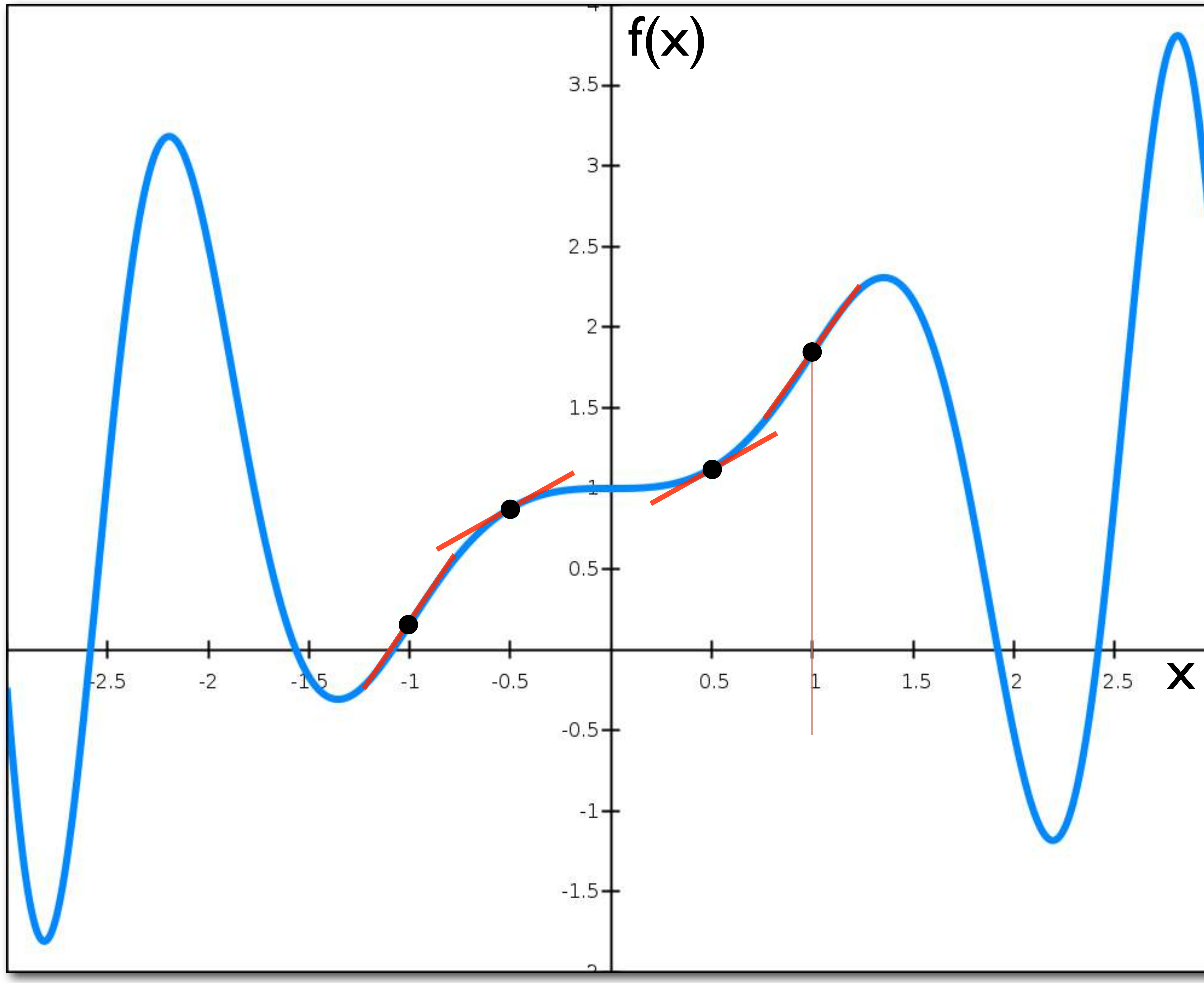


Derivative of a scalar function of a single scalar variable

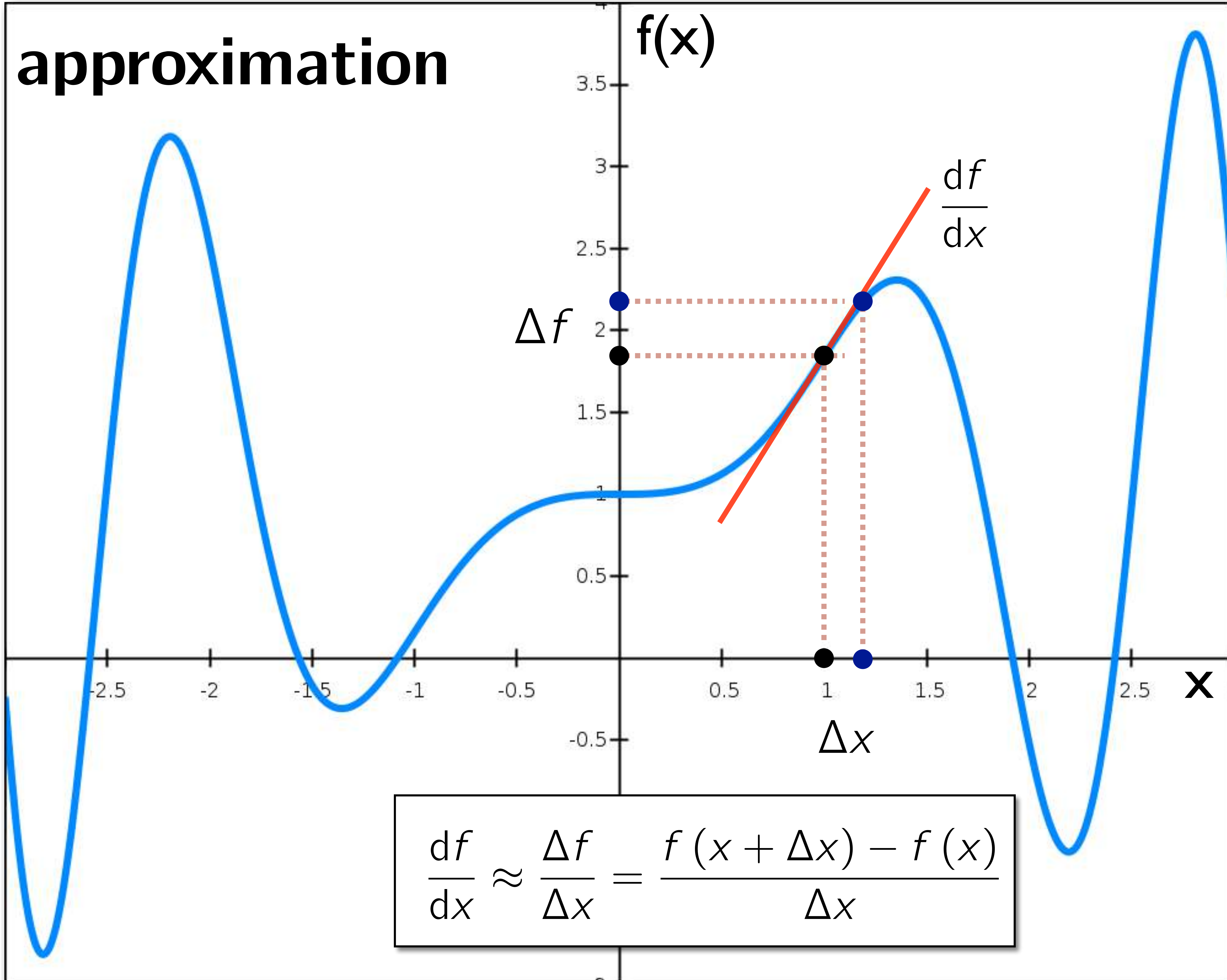
$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$f(x) = x \sin x^2 + 1$$

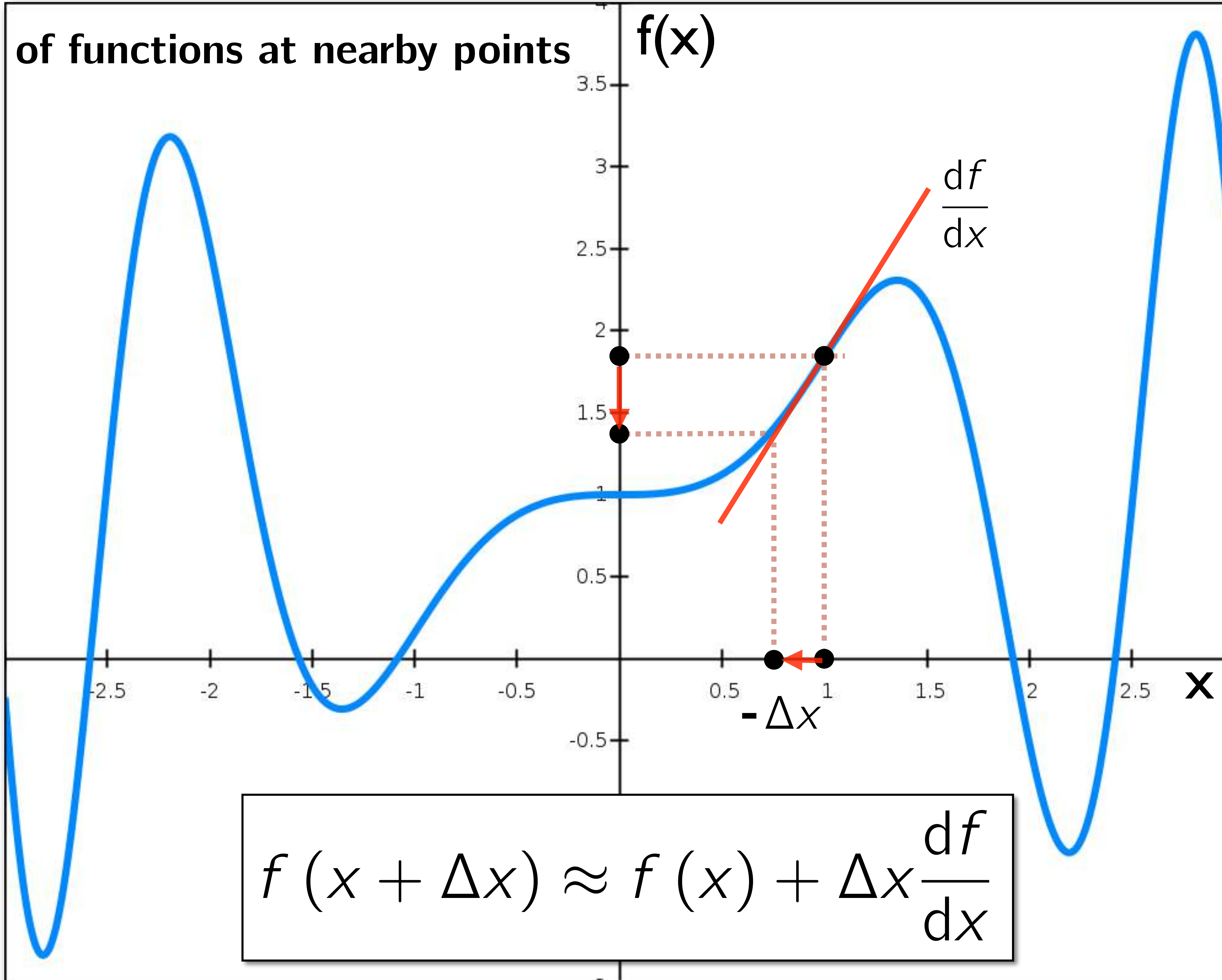
$$\frac{df}{dx}(x) = 2x^2 \cos x^2 + \sin x^2$$



Derivative approximation



Calculating the value of functions at nearby points



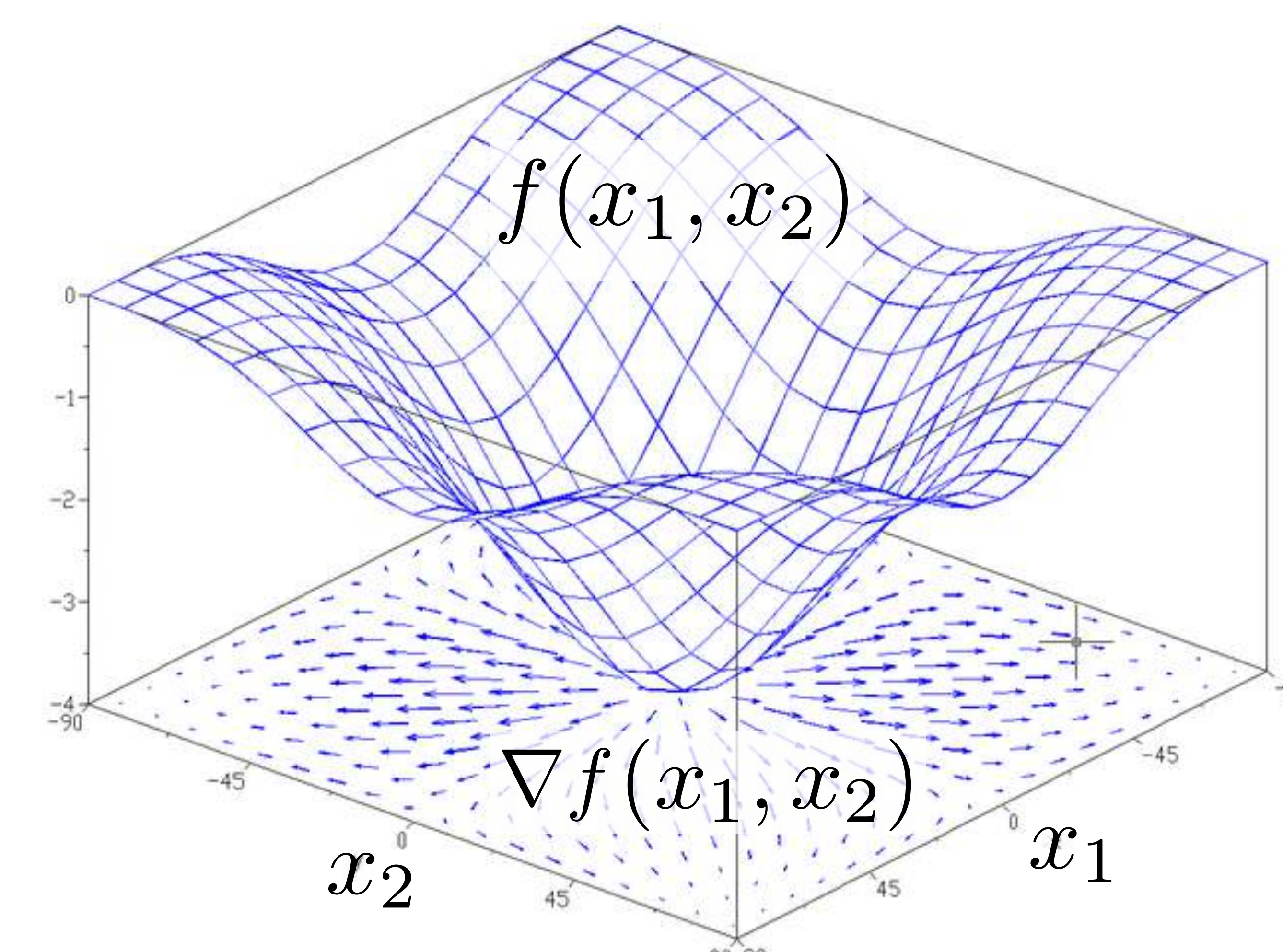
$$f\left(x+\Delta x\right)\approx f\left(x\right)+\Delta x\frac{\mathrm{d}f}{\mathrm{d}x}$$

Derivative of a scalar function of multiple scalar variables (i.e., vector variable)

Let f be a scalar function of a vector variable. The vector variable is $\mathbf{x} = (x_1, x_2, \ldots, x_N)^\mathsf{T}$. This type of function is also called a scalar function of multiple variables or multi-variate function. The value of the function at a point $\mathbf{x}$ is given by $f(\mathbf{x})$ or $f(x_1, x_2, \ldots, x_N)$, and its derivative w.r.t. x is:

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}} = \nabla f = \frac{\partial f}{\partial (x_1, x_2, \ldots, x_N)} = \left[\frac{\partial f}{\partial x_1} \; \frac{\partial f}{\partial x_2} \; \cdots \; \frac{\partial f}{\partial x_N}\right]^\mathsf{T}, \tag{39}$$

which is called the *Gradient* of f at x, and denoted by ∇f.



Derivative of a vector function of a single scalar variable

Let $\mathbf{r}$ be a vector function representing the 3-D motion of a particle as a function of time t:

$$\mathbf{r} = [r_x \ r_y \ r_z]^{\mathsf{T}} . \tag{40}$$

Its derivative is given by:

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \left[\frac{\mathrm{d}r_x}{\mathrm{d}t} \ \frac{\mathrm{d}r_y}{\mathrm{d}t} \ \frac{\mathrm{d}r_z}{\mathrm{d}t}\right]^{\mathsf{T}} , \tag{41}$$

which is also a vector representing the velocity of the particle at time t.

Derivative of a vector function of a vector variable

Some applications require us to calculate derivatives of vector quantifies with respect to other vector quantities. For example, if $\mathbf{f}$ is a vector-valued function of a vector of variables, $\mathbf{x}$. Here, $\mathbf{f}(\mathbf{x}) = (f_1, f_2, \ldots, f_M)^\mathsf{T}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_N)^\mathsf{T}$.

The derivative of $\mathbf{f}$ w.r.t. $\mathbf{x}$ is:

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}} = J(\mathbf{f}, \mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \ldots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_M}{\partial x_1} & \frac{\partial f_M}{\partial x_2} & \ldots & \frac{\partial f_M}{\partial x_N}, \end{bmatrix}$$

and is called the *Jacobian*.

Derivative of a vector function of a vector variable

The Jacobian:

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}} = J(\mathbf{f}, \mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_M}{\partial x_1} & \frac{\partial f_M}{\partial x_2} & \cdots & \frac{\partial f_M}{\partial x_N}, \end{bmatrix} \begin{array}{l} \to \nabla f_1 \\ \\ \\ \\ \end{array}$$

a stack of gradients ∇f_i as rows

$$J(\mathbf{f}, \mathbf{x}) = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_M \end{bmatrix}$$

or matrix where each column is the derivative of a vector function w.r.t. a component of the vector variable:

$$J(\mathbf{f}, \mathbf{x}) = \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial x_1} & \frac{\partial \mathrm{f}}{\partial x_2} & \cdots & \frac{\partial \mathrm{f}}{\partial x_N} \end{bmatrix}$$