centre adscrit a:
UNIVERSITAT DE BARCELONA

Animation Foundations

11. Inverse Kinematics.

Cyclic Coordinate Descent

Dr Joan Llobera – joanllobera@enti.cat

Reminder

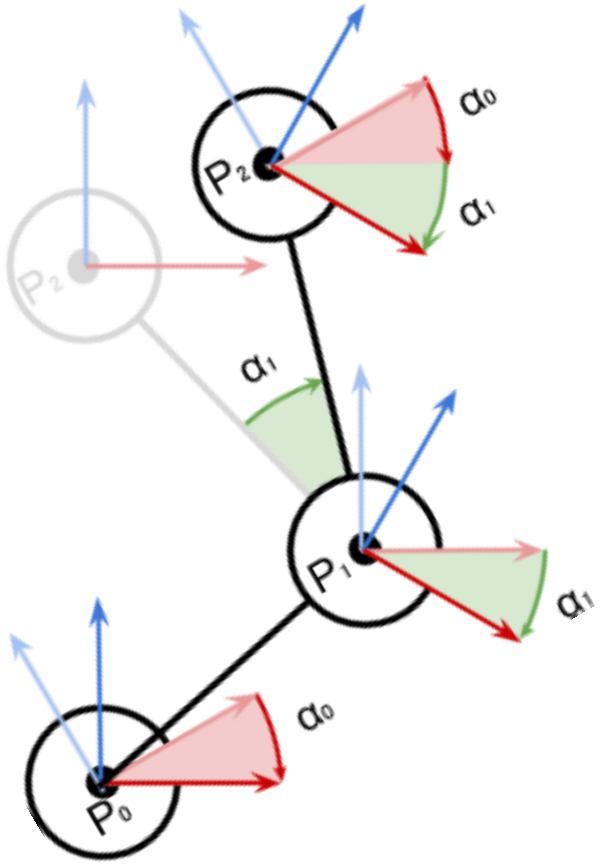
Inverse kinematics. The intro

Idea!

We can define an optimization function to minimize a distance depending on a certain number of angles

Min function(distance(angles) ,angles) = ?

But how?

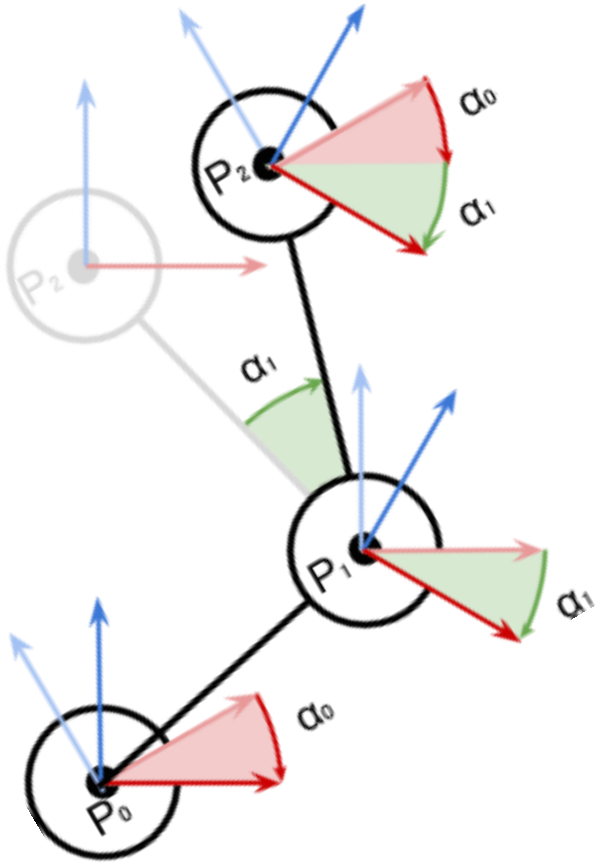


Reminder

Inverse kinematics. The methods

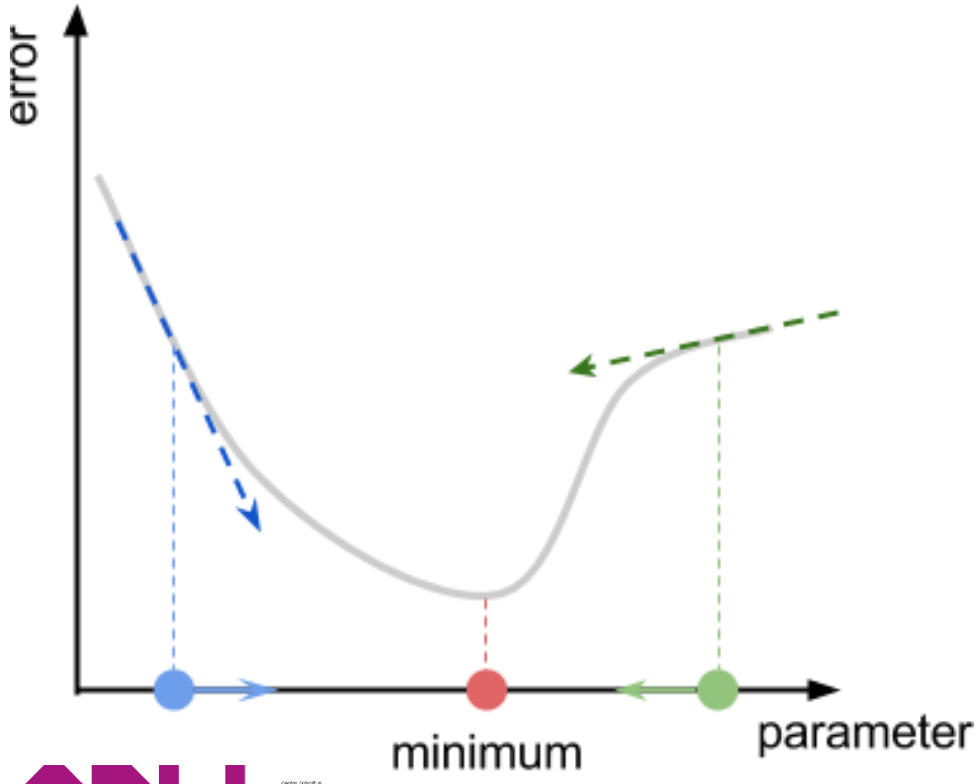
3 methods:

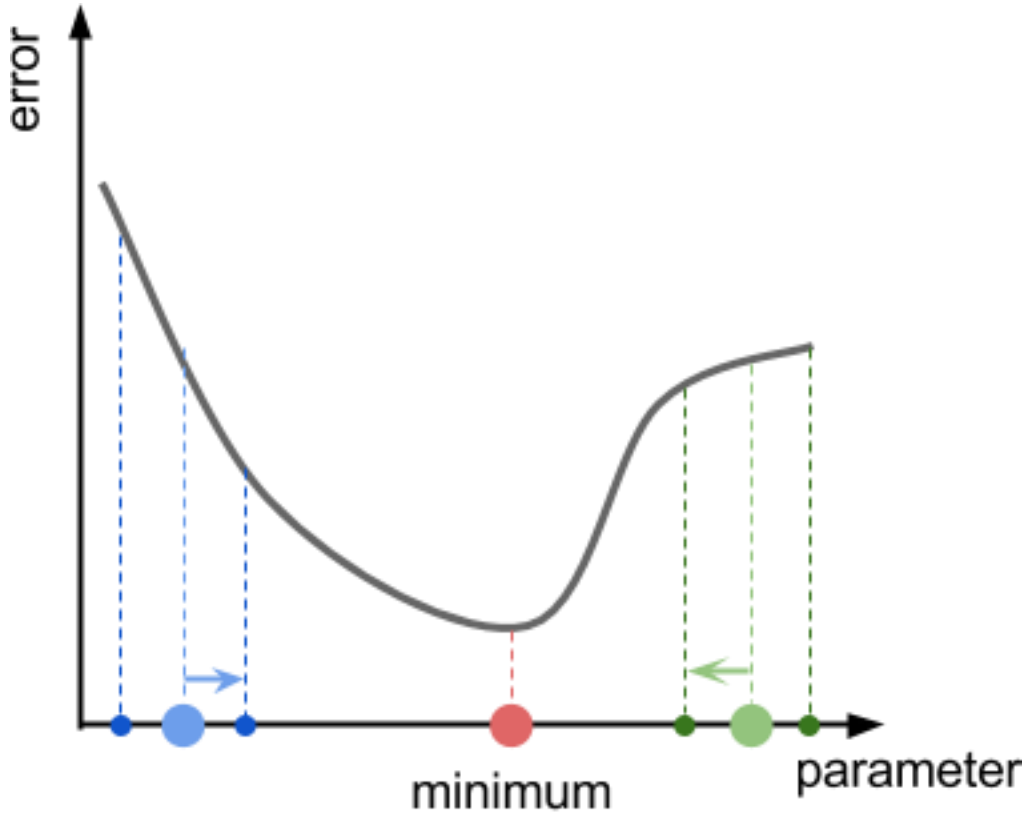
- Gradient Descent (GD)
- Cyclic Coordinate Descent (CCD)
- Forward and Backward Recursive Inverse Kinematic (FABRIK)



Reminder

Gradient Descent. What is it?





Reminder

Gradient Descent. The maths

Gradient 1D:

$$\nabla f(p) = \lim_{n \to 0} \frac{f(p + \Delta x) - f(p)}{\Delta x}$$

Gradient (for our robot):

$$\nabla f(\alpha_0, \alpha_1, \alpha_2) = \lim_{n \to 0} \frac{f(p + \Delta x) - f(p)}{\Delta x}$$

Which means:

$$\nabla f_{\alpha_0}(\alpha_0, \alpha_1, \alpha_2) = \lim_{n \to 0} \frac{f(\alpha_0 + \Delta\alpha_0, \alpha_1, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta\alpha_0}$$

$$\nabla f_{\alpha_1}(\alpha_0, \alpha_1, \alpha_2) = \ \ldots$$

$$\nabla f_{\alpha_2}(\alpha_0, \alpha_1, \alpha_2) = \ \ldots$$

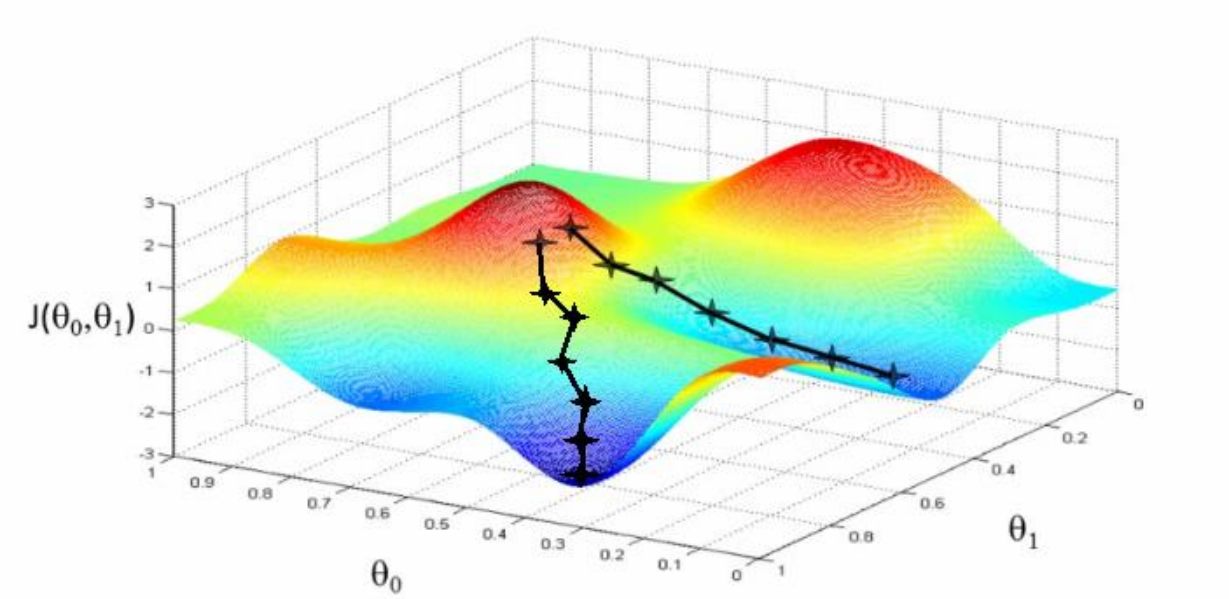
Iteration step:

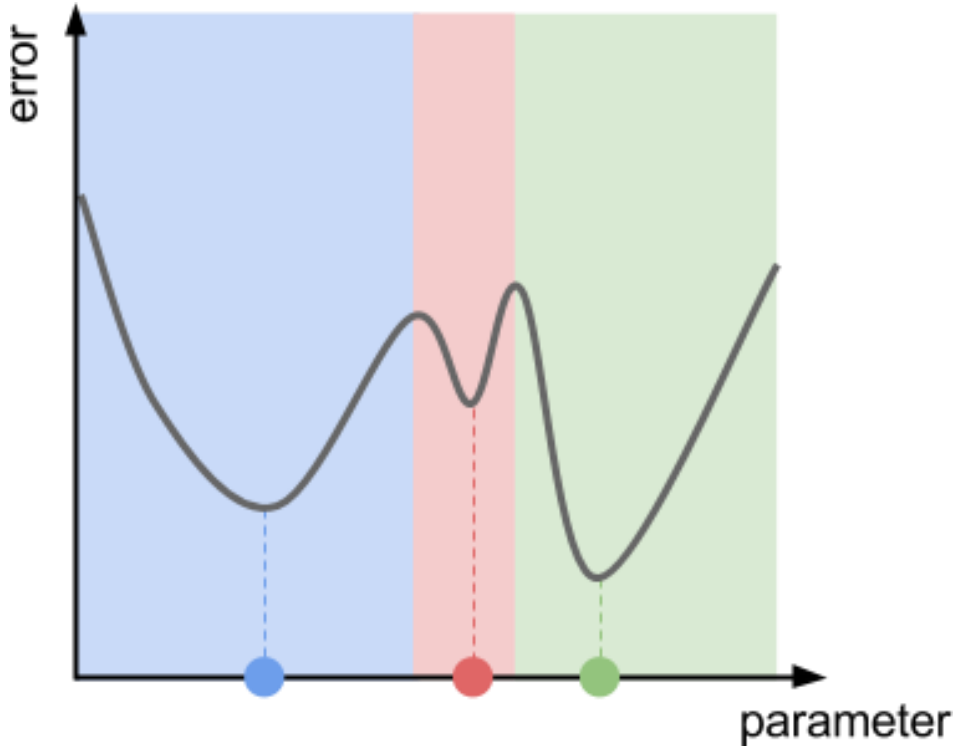
$$(\alpha_0)_{i+1} = (\alpha_0)_i - L\nabla f_{\alpha_0}(\alpha_0, \alpha_1, \alpha_2)$$

$$(\alpha_1)_{i+1} = \cdots$$

$$(\alpha_2)_{i+1} = \cdots$$

Gradient Descent. Potential problems





ESCOLA DE NOVES TECNOLOGIES INTERACTIVES
UNIVERSITAT DE BARCELONA

An interlude on Mathematics (1)

When functions can be plotted in 2D, the rate of change, or the inclination of a function, can be measured with a…

Derivative: $\frac{df(x)}{dx}$

In a computer, we use relative increments: $\frac{\Delta f(x)}{\Delta x}$

When you have more than one variable, you can use a partial derivative: $\frac{\partial f(x)}{\partial x}$

Example: if x is the age of a population, and f(x) is the number of games you buy, partial derivative can tell you how you change your consumption patterns depending on the age.

But you could also calculate the derivative from another variable y, i.e., from the income of a population. $\frac{\partial f(y)}{\partial y}$

→ Notice both would not be independent (this means trouble in the world of maths)

An interlude on Mathematics (2)

When functions have more variables… You can use gradient.

The gradient tells you in what direction there is more change. In Cartesian coordinates, it is simply:

$$\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

Example: if f is the pollution in the air (particles / cubic meter), the gradient will tell you in what direction you will have clean air the faster (assume no wind)

A gradient:

- Is used on a scalar field

(one number on each position)

- It results in a vectorial field

(one vector on each position)

An interlude on Mathematics (3)

When functions have more variables… You can also use divergence.

The divergence gives you the rate of change of a vectorial field. In Cartesian coordinates, it is simply:

$$\mathrm{d}iv\ \vec{f}(x,y,z) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} + \frac{\partial f}{\partial z}$$

$$div\ \vec{f}(x,y,z) = \nabla \cdot \vec{f}(x,y,z)$$

Example: if f is the density of particles of oxygen,, where heat increases the gas will expand and there will be a flow of particles towards the outside (source, positive divergence).

Where heat decreases the gas will contract and there will be a flow of particles towards the inside (sink, negative divergence)

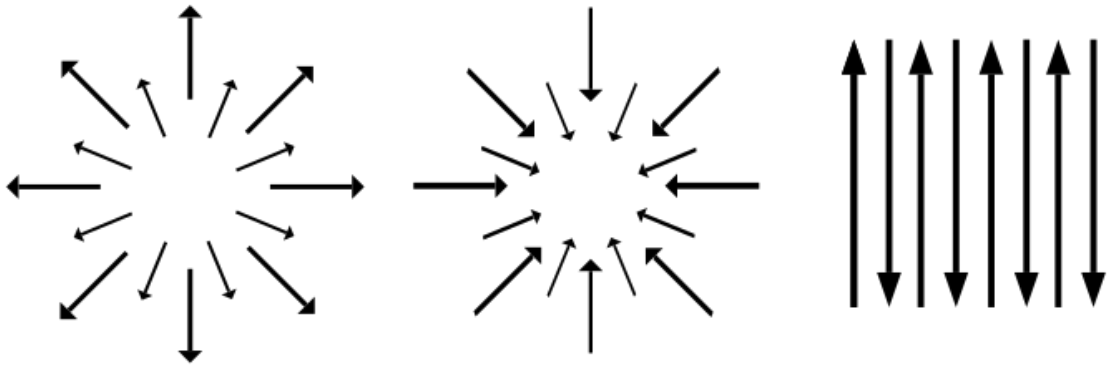
A gradient:

- Is used on a vectorial field

(one vector on each position)

- It results in a numerical field

(one number on each position)



$\partial/\partial x(\mathbf{V}_x) > 0$
$\partial/\partial y(\mathbf{V}_y) > 0$
$\nabla\bullet(\mathbf{V}) > 0$

$\partial/\partial x(\mathbf{V}_x) < 0$
$\partial/\partial y(\mathbf{V}_y) < 0$
$\nabla\bullet(\mathbf{V}) < 0$

$\partial/\partial x(\mathbf{V}_x) = 0$
$\partial/\partial y(\mathbf{V}_y) = 0$
$\nabla\bullet(\mathbf{V}) = 0$

An interlude on Mathematics (4)

When functions have more variables… You can also use the Jacobian.

Consider a function such as:

$$\boldsymbol{f}\colon \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}^{\boldsymbol{m}}$$

The Jacobian gives you the rate of change of a function in every place where it is differentiable.

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

An interlude on Mathematics (4)

When functions have more variables… You can also use the Jacobian

Cons

The Jacobian gives you the rate of change of a function in every place where it is

Example: a robotic arm on a platform built on wheels

n number of actuators (rotate joint 1, move in direction x, etc.),

m dimensions: 3 in the end effector (x,y,z), plus 2 in the position of the platform (x,y)

The Jacobian method of IK

The Jacobian method is the first Iterative method that was introduced.

- Focus: Linearization
- Dealing with small displacements

Principle of the linearization

1) First linearize the function:

x = **F**(θ, body skeleton)

by storing its first derivatives

in the jacobian matrix *J*.

$$J = \begin{pmatrix} \frac{\partial Fx_1}{\partial \theta_1} & \frac{\partial Fx_1}{\partial \theta_2} & \cdots & \frac{\partial Fx_1}{\partial \theta_N} \\ \frac{\partial Fx_2}{\partial \theta_1} & \frac{\partial Fx_2}{\partial \theta_2} & \cdots & \frac{\partial Fx_2}{\partial \theta_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Fx_M}{\partial \theta_1} & \frac{\partial Fx_M}{\partial \theta_2} & \cdots & \frac{\partial Fx_M}{\partial \theta_N} \end{pmatrix}$$

Principle of the linearization

2) Then, use the inverse of the Jacobian as a local approximation of

$$\theta = J^{-1}(x, \text{body skeleton})$$

with :

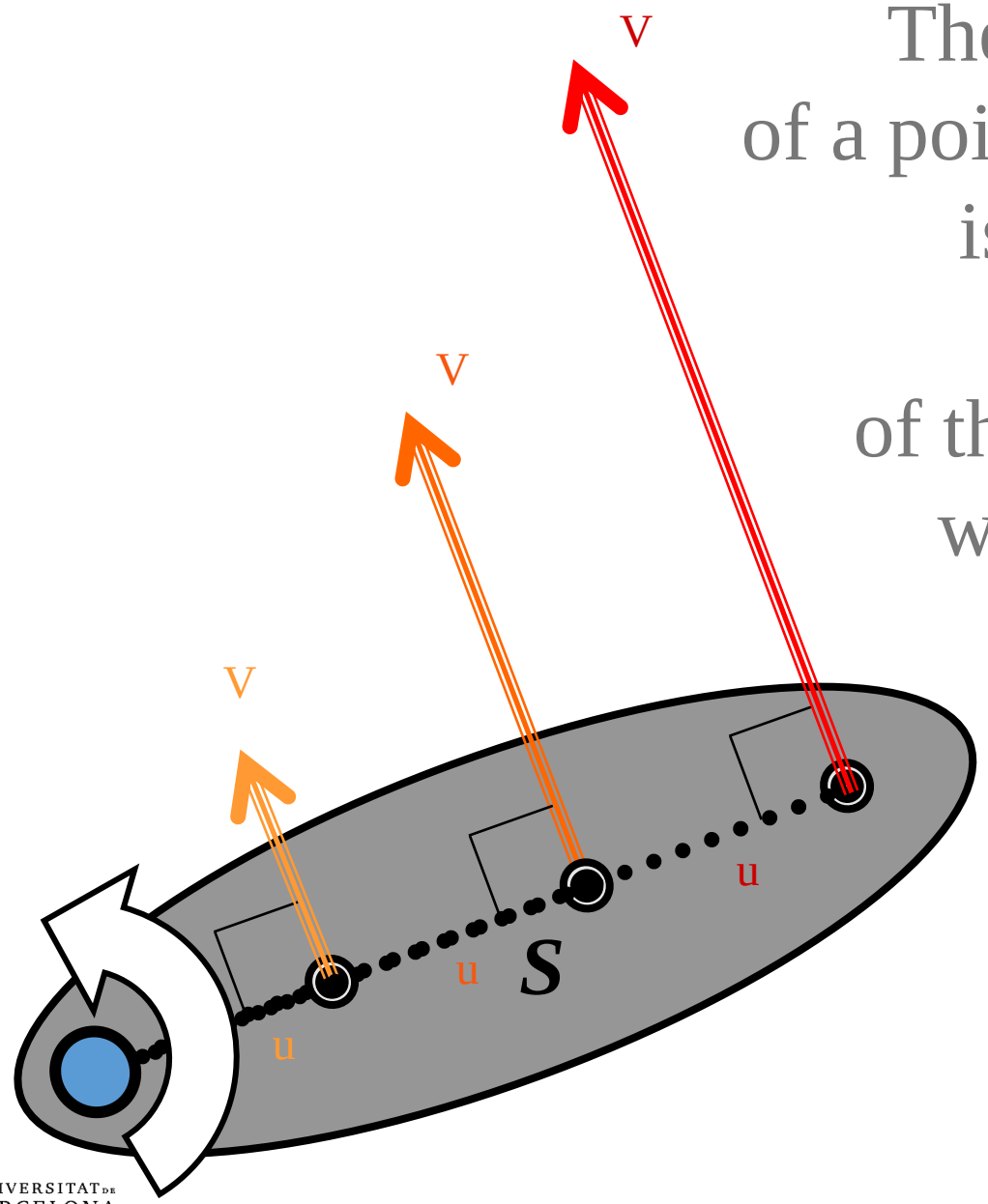
$$\Delta\theta = J^{-1}(\Delta x)$$

How to build the jacobian?

- Consider the influence of each joint independently of the other joints

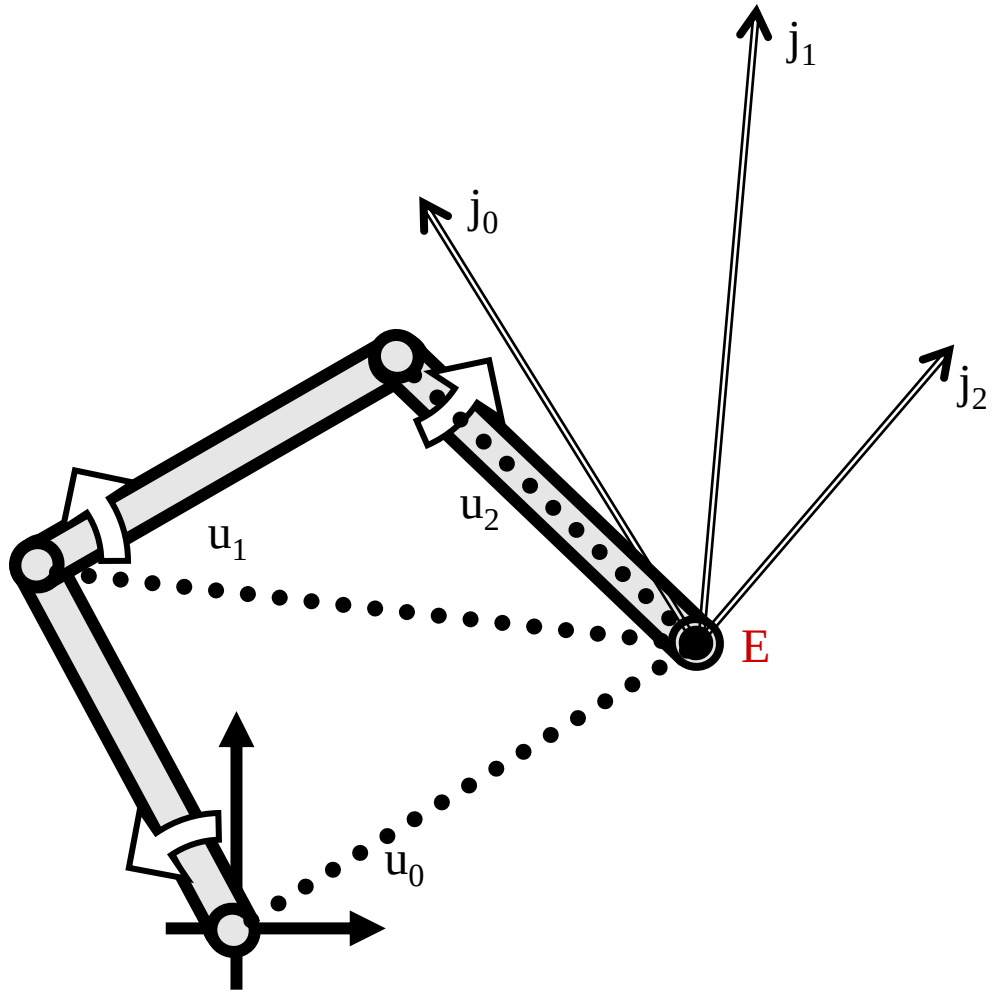
  The joints ignore each other ⇔ it's a first order approximation

- When one joint changes, all the children segments are viewed as ONE UNIQUE RIGID SOLID
- It is easy to evaluate the instantaneous velocity of a point of a rigid solid:
  - With the cross product of the rotation velocity vector by the position vector locating the point with respect to the joint (see next slide)
  - Use a unit rotation velocity for building the Jacobian



The velocity vector V
of a point from the solid ***S***
is obtained with the
cross product
of the rotation velocity
with the *lever arm* u

Illustration for a 3 segments arm, for point E



j_0	j_1	j_2
Xj_0	Xj_1	Xj_2
Yj_0	Yj_1	Yj_2

Jacobian J of the effector E for the chain current state

See [Rotenberg06] for other jacobian computation

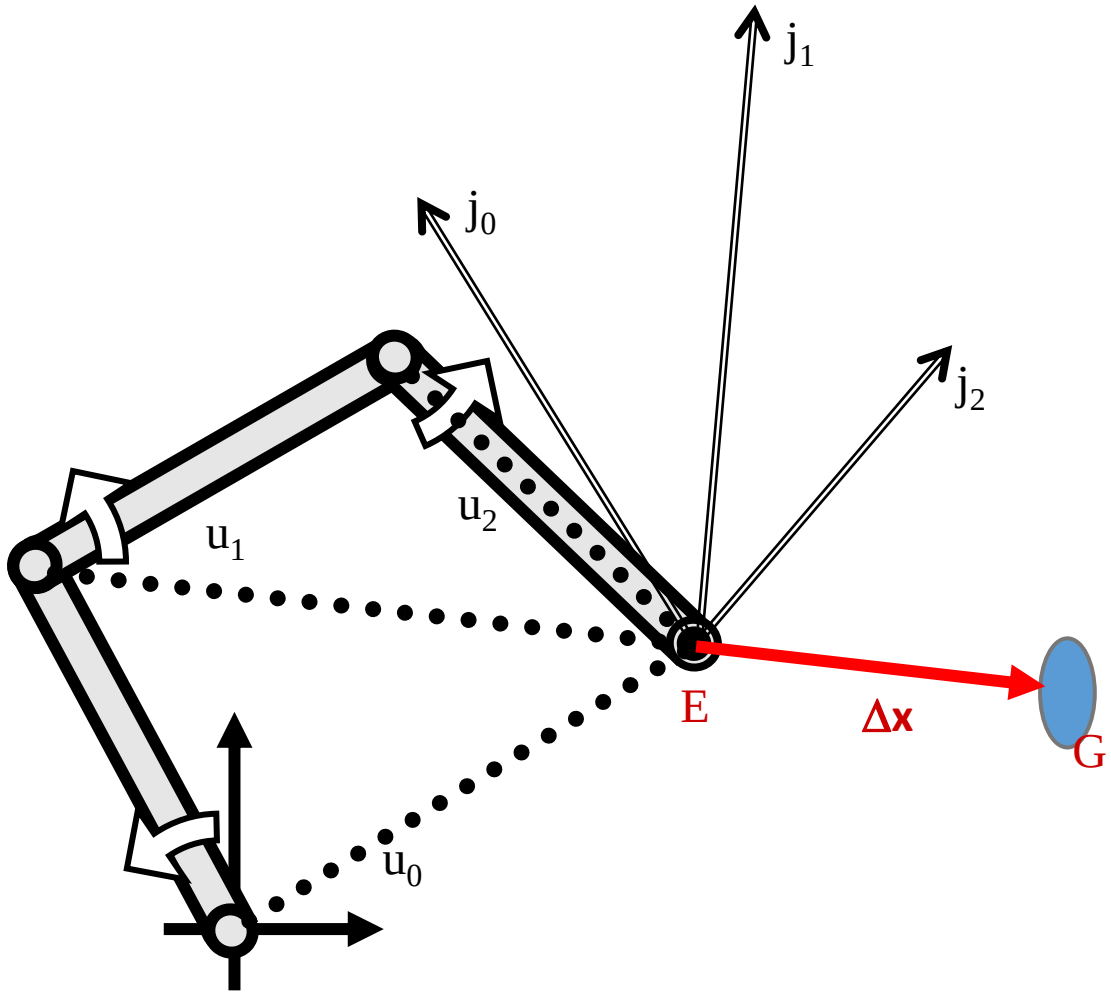
Jacobian transpose

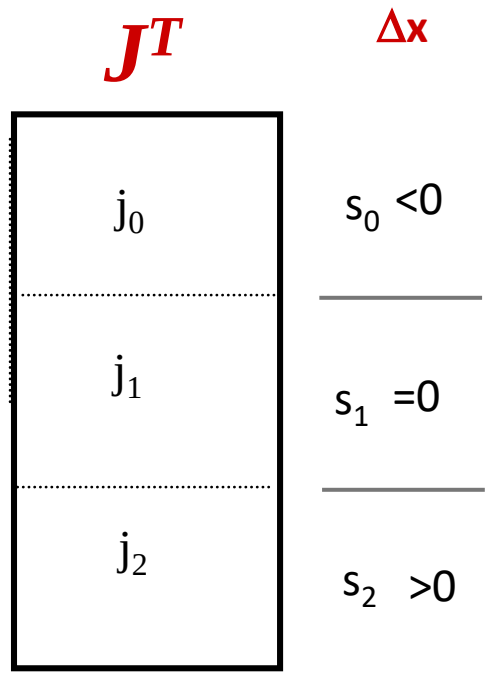
- The Jacobian matrix J, of dimension m x n, gathers all the partial derivatives of the m constraints x_i with respect to the n joint variables θ_j

$$J = [\delta x_i / \delta \theta_j]$$

- The Jacobian transpose method [Welman93]:
  - exploits the absolute influence of each joint variable θ_j for contributing to the constraint error vector Δx
  - the influence of a joint j is given by the scalar product of Δx with the column j of J

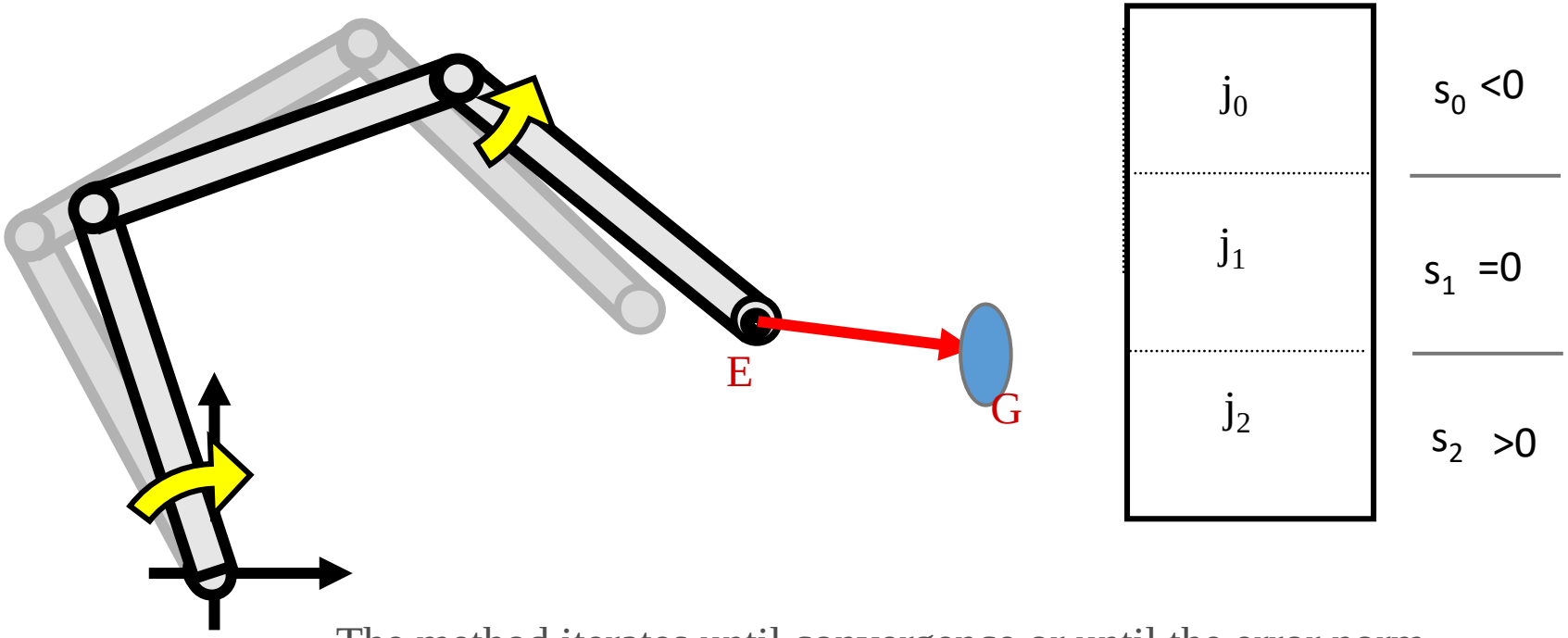
Ex: the effector E is attracted toward the goal G (1)





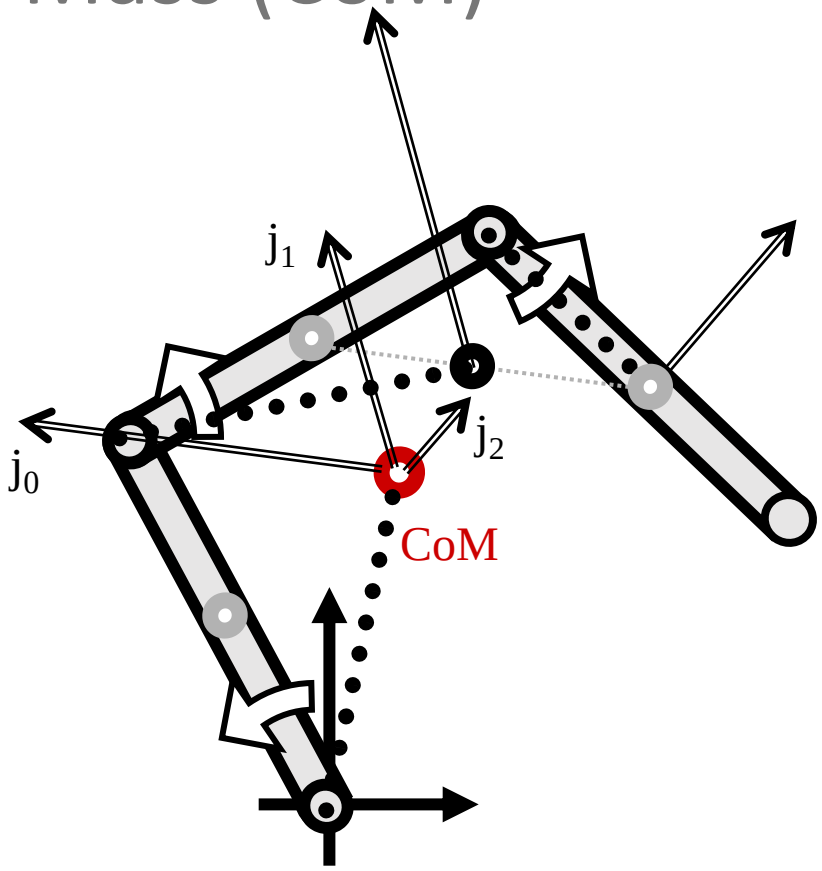
Translation Jacobian *J* of the effector E
for the chain current state

Ex: the effector E is attracted toward the goal G (2)



The method iterates until convergence or until the error norm does not decrease any more

Illustration for a 3 segments arm, for the center of Mass (CoM)



More sophisticated constraints!

j_0	j_1	j_2
Xj_0	Xj_1	Xj_2
Yj_0	Yj_1	Yj_2

Jacobian J of the Center of Mass for the chain current state

[Boulic & Mas 97]

Exercise: recheck how the Gradient Descent works.

Discuss if it is a particular case of the Jacobian method, or if it is not

Wrap-up: what we know about IK so far

- We have learnt the first iterative methods for IK
  - Gradient Descent
  - Jacobian-based methods
- We had analytic solutions for simple cases
(see reminder below)

Analytic methods

- Invert the kinematic equations
- Example for a limb

  Direct kinematics:

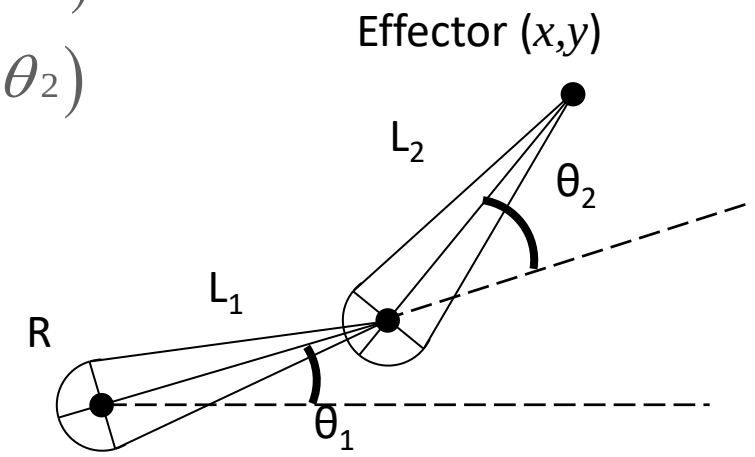
$$x = f_1(\theta_1, \theta_2) = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$

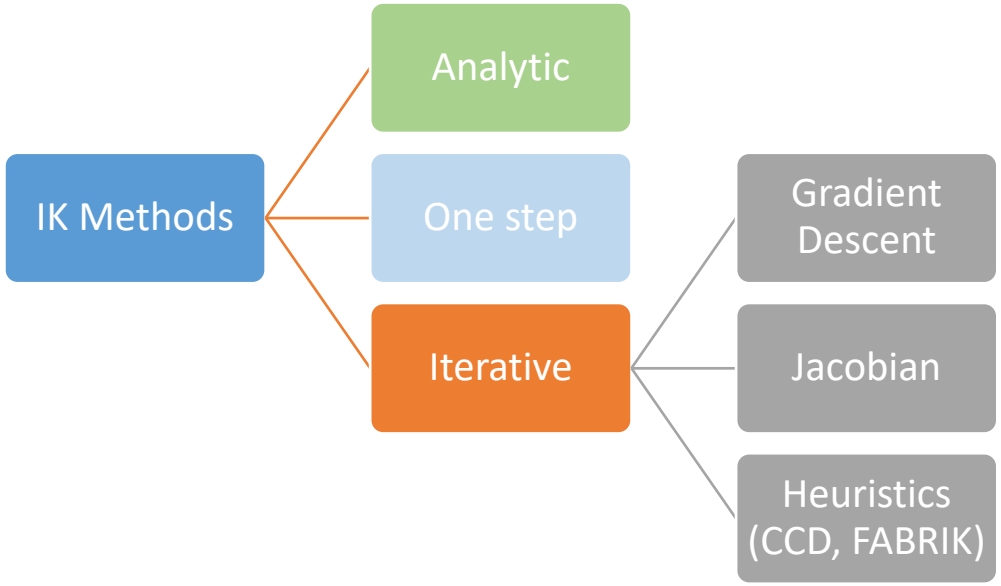
$$y = f_2(\theta_1, \theta_2) = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

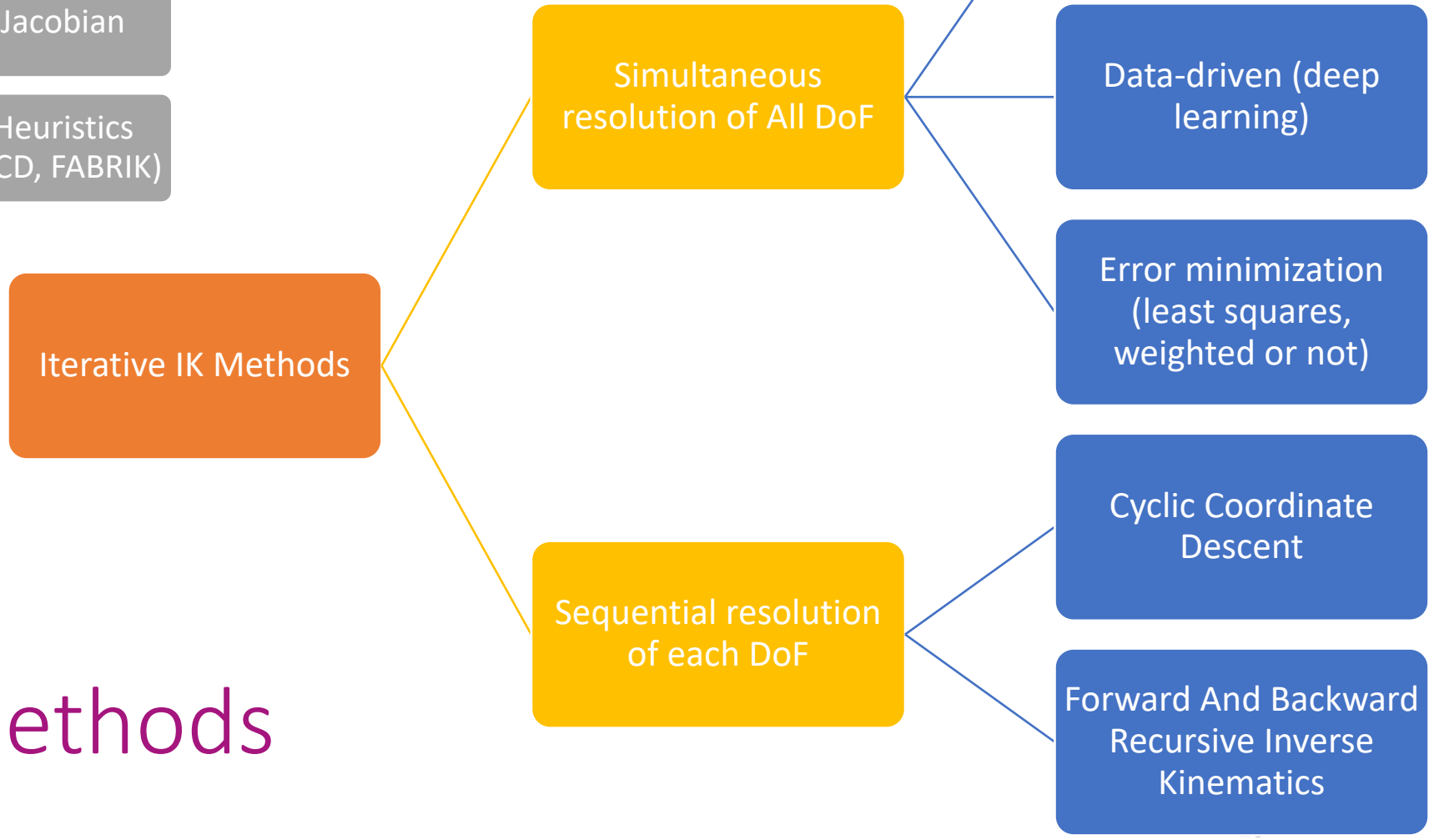
  Inverse kinematics:

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right)$$

$$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{L_2 \sin\theta_2}{L_1 + L_2 \cos\theta_2}\right)$$







Overview of IK methods

ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

UNIVERSITAT DE BARCELONA

Analytical IK vs Iterative

- Analytical
  - Advantages
    - Global solution
    - Reliable, exploited in Robotics
    - Strong (no singularity)
    - Fast
    - « Simple »
  - Drawbacks
    - Only for a few DOFs
    - Not suited for redundant systems more complex than the arm/leg
- Iterative methods
  - Jacobian:
    - Focus: Linearization
    - Dealing with small displacements
  - Heuristic:
    - CCD
    - FABRIK

Comparison of analytic and iterative IK

- Analytic is best suited for simple case like isolated arm, leg, etc…
- Iterative is more general but requires multiple steps to converge towards the solution
  - Due to the non-linearity of the problem
  - If big steps are used, it becomes unstable
  - Or due to solving only for one DOF at a time (CCD)

CCD

- Issued from robotics works [Luenberger84, Wang91]
- Animation: recursive algorithm [Badler87]

- Advantage of this IK solver
  - Fast

- Compared to Jacobian-based approach
  - Solve only one DOF at a time

CCD

- Only one column remains in the Jacobian
- Inversion is fast: low computation cost

$$\begin{pmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_M \end{pmatrix} = \begin{pmatrix} \frac{\partial fx_1}{\partial \theta_1} & \frac{\partial fx_1}{\partial \theta_2} & \cdots & \frac{\partial fx_1}{\partial \theta_N} \\ \frac{\partial fx_2}{\partial \theta_1} & \frac{\partial fx_2}{\partial \theta_2} & \cdots & \frac{\partial fx_2}{\partial \theta_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial fx_M}{\partial \theta_1} & \frac{\partial fx_M}{\partial \theta_2} & \cdots & \frac{\partial fx_M}{\partial \theta_N} \end{pmatrix} \begin{pmatrix} \cancel{d\theta_1} \\ d\theta_2 \\ \cancel{\vdots} \\ \cancel{d\theta_N} \end{pmatrix}$$

CCD

- Simple algorithm

  For all the DOF θ_i

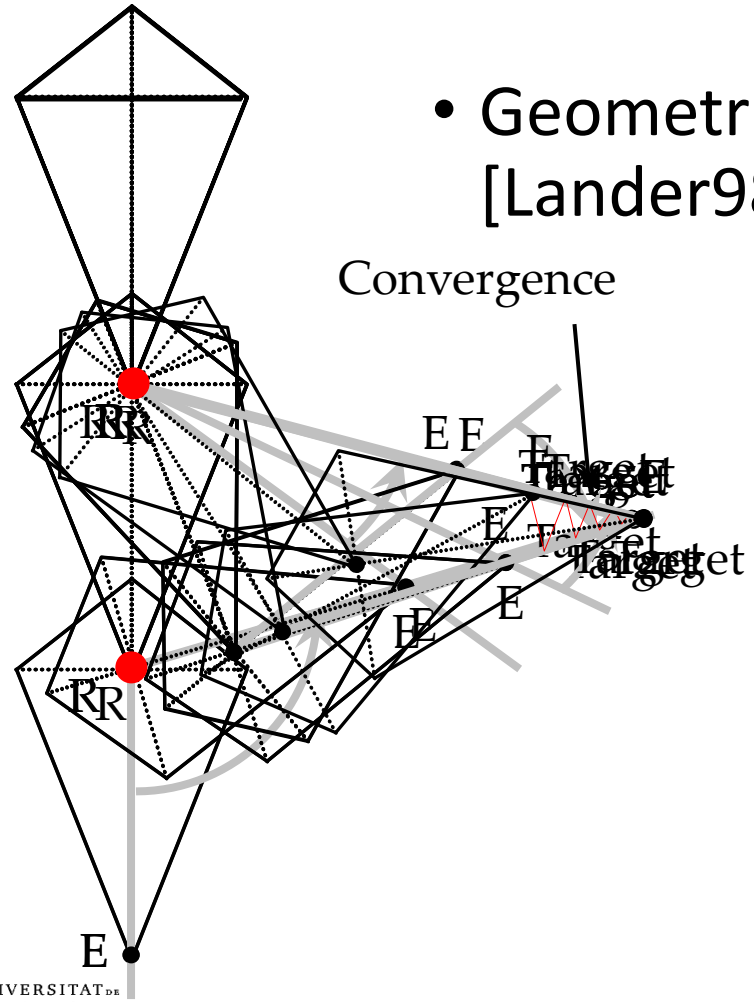
    Compute variation $\Delta\theta_i$ of the DOF θ_i

    Add this variation: $\theta_i = \theta_i + \Delta\theta_i$

  End

- Variation $\Delta\theta_i$ obtained:
  - By inverting the vector of partial derivatives
  - Geometrically

CCD

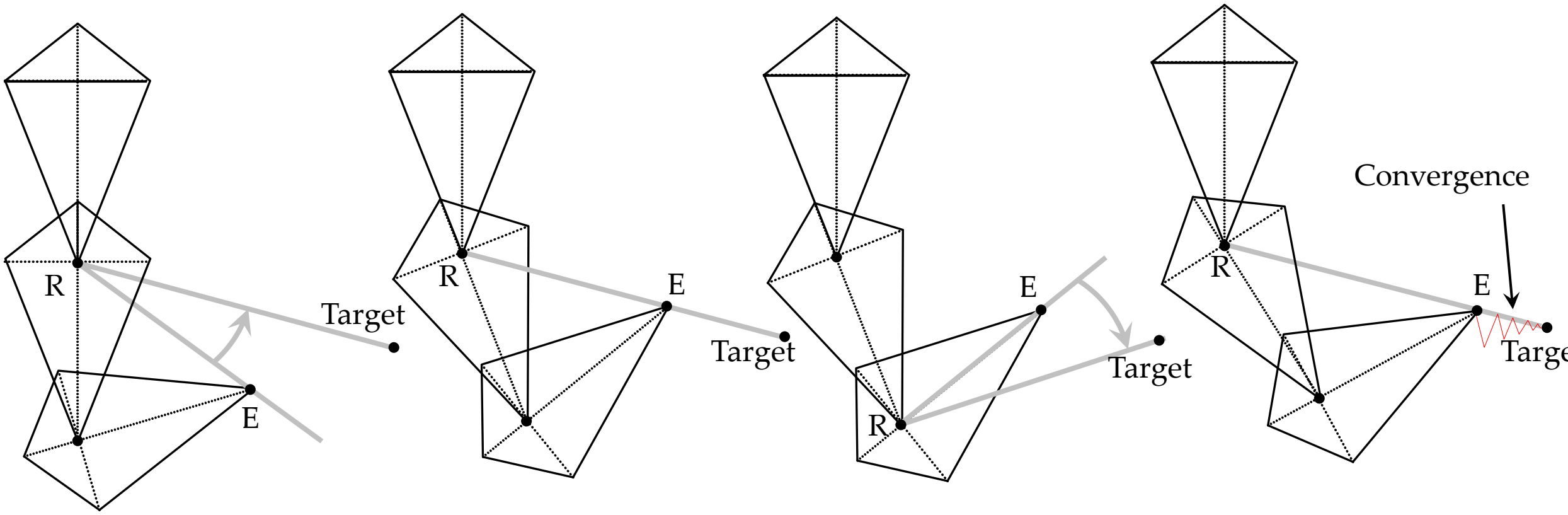


- Geometrical resolution by CCD [Lander98]

Example with 2 DOFs

E: effector
R: currently modified DOF

CCD



CCD

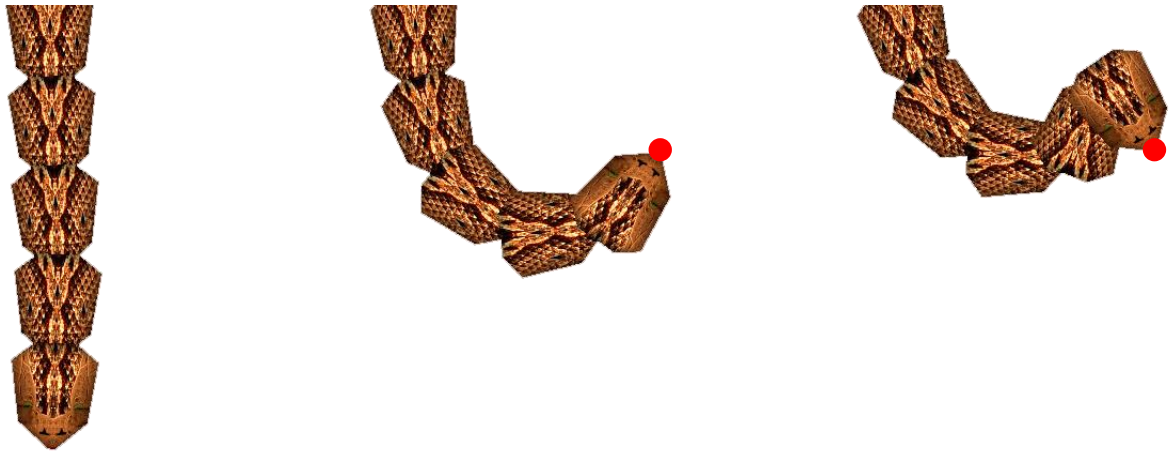
Exercise:

Implement IK based on CCD

CCD

- Advantages
  - Fast computation of one iteration
  - Easy to implement
  - Handling of joint limits

- Drawbacks
  - Slow convergence
  - Bad distribution of the adaptation
  - Unnatural posture



⇒ First joints are more modified than the following ones!

CCD

- First solution: use damping
  - Threshold on the variation of the joint parameters
  - Minimizes the adaptation of each joint
  - But increases the number of iterations
- More homogeneous adaptation
  $\Rightarrow$ Bigger computation cost
- CCD is not suitable for postural adaptation of humanoids
- Our goal
  - Find natural postures
  - Computation time compatible with interactive animation of hundreds of characters

Bibliography & Credits

Notice the references within the slides, they point to specific research articles. Infinite thanks to Ronan Boulic, from EPFL, for advice and materials.

Part of the Materials have been adapted from:

Boulic & Kulpa (2007) *Inverse Kinematics and Kinetics for Virtual Humanoids* Eurographics tutorial

The website is still online!

https://dcgi.fel.cvut.cz/cgg/eg07/index.php?page=tutorial_4