Cinématique Inverse

C Gentil / M Neveu

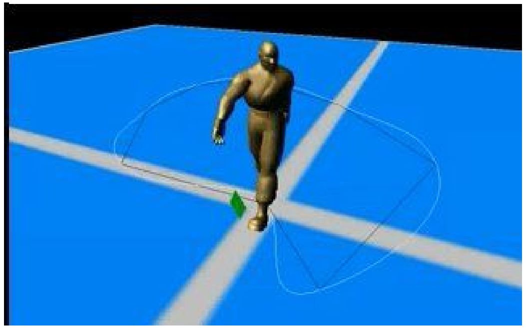
et d’après les cours de :

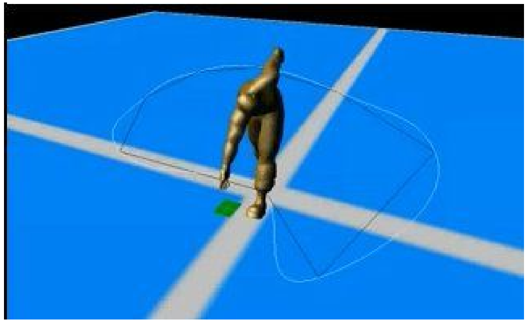
N Holschuch /

Olivier Vaillancourt, Olivier Godin

Plan du cours

- Cinématique inverse :
  - Pourquoi faire ?
  - Animation d'un modèle
- Manipulation directe du modèle :
  - Sélection
  - Tirer une partie du modèle

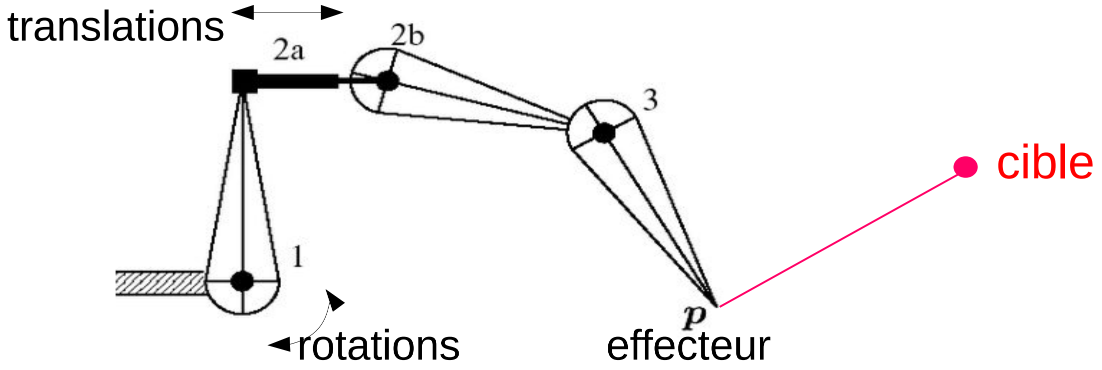




Démo Blender

Cinématique inverse

- Objet articulé
  - Liens, articulations
  - Objectif à atteindre



Cinématique inverse

- Donnée : position à atteindre (M)
- Sortie : valeurs des paramètres des articulations
- Θ=vecteur des paramètres du modèle
  - $\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$
- Trouver $\Theta=g(M)$
- Deux rotations: calcul direct
- Trois rotations : calcul direct
- N articulations : ?

Deux rotations

- Solution directe :

$$d=\sqrt{x^2+y^2}$$

$$d^2=\left(L_1+L_2\cos\theta_2\right)^2+\left(L_2\sin\theta_2\right)^2$$

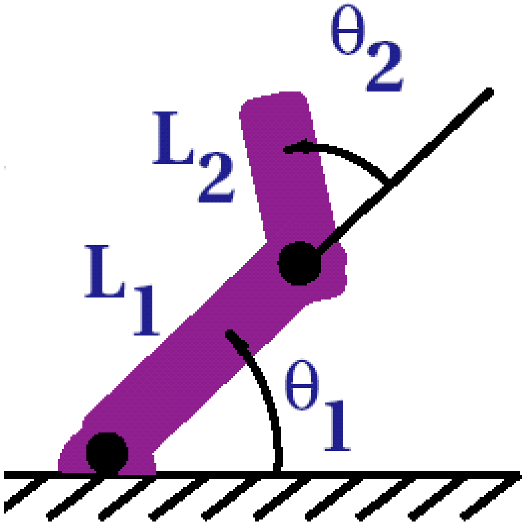
$$d^2=L_1^2+L_2^2+2L_1L_2\cos q_2$$

$$\cos\theta_2=\frac{d^2-L_1^2-L_2^2}{2L_1L_2}$$

$$\tan(\alpha)=\frac{L_2\sin\theta_2}{L_1+L_2\cos\theta_2}$$

$$\tan(\theta_1+\alpha)=\frac{y}{x}$$

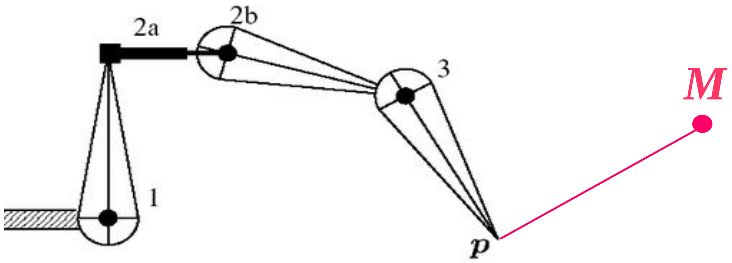
$$\theta_1=\arctan\left(\frac{y}{x}\right)-\arctan\left(\frac{L_2\sin\theta_2}{L_1+L_2\cos\theta_2}\right)$$



Trois rotations

- Encore une solution directe
  - trigonométrie
- Paramètre supplémentaire
  - Choix de la solution
- Limites des articulations

Cas général : n articulations



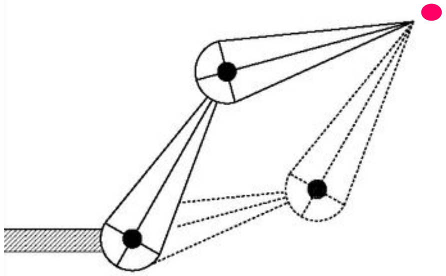
- On veut que l'effecteur $f(\Theta)$=M
- Θ=vecteur des paramètres du modèle
  - $\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$
- $f(\Theta)$ position de l'extrémité du modèle (coord. 2D ou 3D)
- M position de la cible (coord. 2D / 3D)
- Connaissant M, trouver Θ

Pourquoi c’est difficile ?

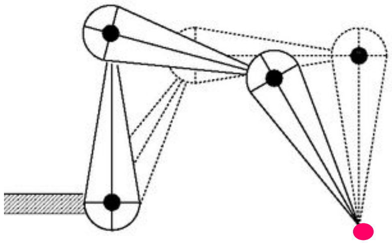
- Problème non-linéaire
- Plusieurs solutions
- Pas toujours bien conditionné
- Limites des articulations

Non-unicité

- Deux solutions :



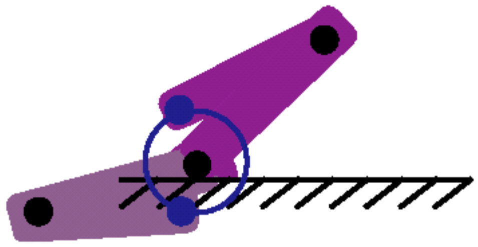
- Intervalle de solutions :



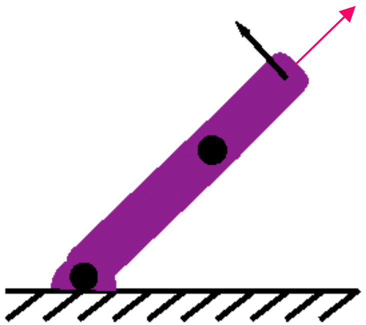
- Pas de solutions :

Pas toujours bien conditionné

- Petite différence sur M, grande différence sur Θ :



Changer Θ ne rapproche pas de M :



Au fait, que vaut f ?

- Matrice de transformation
- Concaténation des matrices
- $f(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\,\mathbf{T}_2\mathbf{R}_3(\theta_3)\,\mathbf{T}_3 \ldots M_0$

  Ou $f(\Theta) = \mathbf{T}_1\,\mathbf{R}_1(\theta_1)\mathbf{T}_2\,\mathbf{R}_2(\theta_2)\,\mathbf{T}_3\,\mathbf{R}_3(\theta_3) \ldots M_0$
  - M_0 position extrémité du bras avant transformations
- Non-linéaire à cause des rotations
- Calcul de f : cinématique directe

Racines d'une fonction non-linéaire

- On veut trouver Θ tel que : $f(\Theta)$-M = 0
- Linéarisation du problème :

$$f(\Theta+h)=f(\Theta)+f'(\Theta)h+f''(\Theta)h^2+\ldots$$

- On part d'une valeur de Θ et de $f(\Theta)$
- **On ne connaît pas Λ tel que $f(\Theta+\Lambda)=M$**
- On peut trouver Λ' qui s'en rapproche
- **$\Theta \leftarrow \Theta+\Lambda$' et on itère**

Linéarisation

Séries de Taylor :

$$f(\Theta+h)=f(\Theta)+f'(\Theta)h+f''(\Theta)h^2+\ldots$$

Cas des fonctions à plusieurs variables :

$$f(\Theta+h)=f(\Theta)+J(\Theta)h+{}^{t}hH(\Theta)h+\ldots$$

J Jacobien de *f*, forme linéaire

H Hessien de *f*, forme quadratique

Jacobien

- Matrices des dérivées d'une fonction à plusieurs variables :

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$J(\theta) = \begin{bmatrix} \frac{\partial f_x}{\partial x_0}(\theta) & \frac{\partial f_x}{\partial x_1}(\theta) & \cdots \\ \frac{\partial f_y}{\partial x_0}(\theta) & \frac{\partial f_y}{\partial x_1}(\theta) & \cdots \\ \frac{\partial f_z}{\partial x_0}(\theta) & \frac{\partial f_z}{\partial x_1}(\theta) & \cdots \end{bmatrix}$$

Linéarisation du problème

- $f(\Theta)$-M = E, erreur actuelle
- Trouver Λ, tel que $\mathbf{J}(\Theta)\Lambda$ = E
- Λ : résolution système linéaire
- Approximation linéaire : petits déplacements
  - Petits déplacements dans la direction de Λ
- Série de petits déplacements
- Recalculer **J** et E à chaque étape

Résolution itérative

- Petits pas par petits pas
- À chaque étape :
  - Choix entre plusieurs solutions
  - Prendre solution proche position actuelle
- Taille des pas :
  - Chercher taille optimale ?
  - Rotations : pour $x < 2$ degrés:
    - $\sin(x) \approx x$
    - $\cos(x) \approx 1$
  - Garder pas < 2 degrés

Algorithme

```
inverseKinematics()
  {
    start with previous Θ;
    E = target - computeEndPoint();
    for(k=0; k<kmax && |E| > eps; k++){
      J = computeJacobian();
      solve J Λ = E;
      if (max(Λ)>2) Λ = 2Λ/max(Λ);
      Θ = Θ+ Λ;
      E = target - computeEndPoint();
    }
  }
```

La bonne question

- **`solve J Λ = E;`**
- **J** n’est pas inversible
  - En effet, **J** n’est pas carrée
  - **J** matrice 2*n (ou 3n) :

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_0}(\theta) & \frac{\partial f_x}{\partial x_1}(\theta) \cdots \\ \frac{\partial f_y}{\partial x_0}(\theta) & \frac{\partial f_y}{\partial x_1}(\theta) \cdots \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} E_x \\ E_y \end{bmatrix}$$

Pseudo-Inverse

- $\mathbf{J}^T\mathbf{J}$ est carrée (n*n). Donc :

  $\mathbf{J}\,\Lambda = E$

  $\mathbf{J}^T\mathbf{J}\,\Lambda = \mathbf{J}^T E$

  $\Lambda = (\mathbf{J}^T\mathbf{J})^{-1}\,\mathbf{J}^T E$

  $\Lambda = \mathbf{J}^+ E$

- $\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1}\,\mathbf{J}^T$ *pseudo-inverse* de **J**
  - Pareil que l'inverse si **J** est carrée et inversible
  - Propriétés : $\mathbf{J}\mathbf{J}^+\mathbf{J} = \mathbf{J}$, $\mathbf{J}^+\mathbf{J}\mathbf{J}^+ = \mathbf{J}^+$
  - **J** est $m*n \Rightarrow \mathbf{J}^+$ est $n*m$

Et la cinématique inverse ?

- On veut résoudre $X = f(\Theta) + \mathbf{J}(\Theta)\Lambda$
  - $f(\Theta)$ position de l'extrémité du bras
  - i^e colonne de **J** vient de l'articulation i
  - $f(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\mathbf{T}_2\mathbf{R}_3(\theta_3)\mathbf{T}_3 \ldots M_0$

$$\frac{\partial f}{\theta_1} = \frac{\partial R_1}{\theta_1} T_1 R_2 T_2 R_3 T_3 \cdots M_0$$

$$\frac{\partial f}{\theta_1} = R_1\left(\theta_1 + \frac{\pi}{2}\right) T_1 R_2 T_2 R_3 T_3 \cdots M_0$$

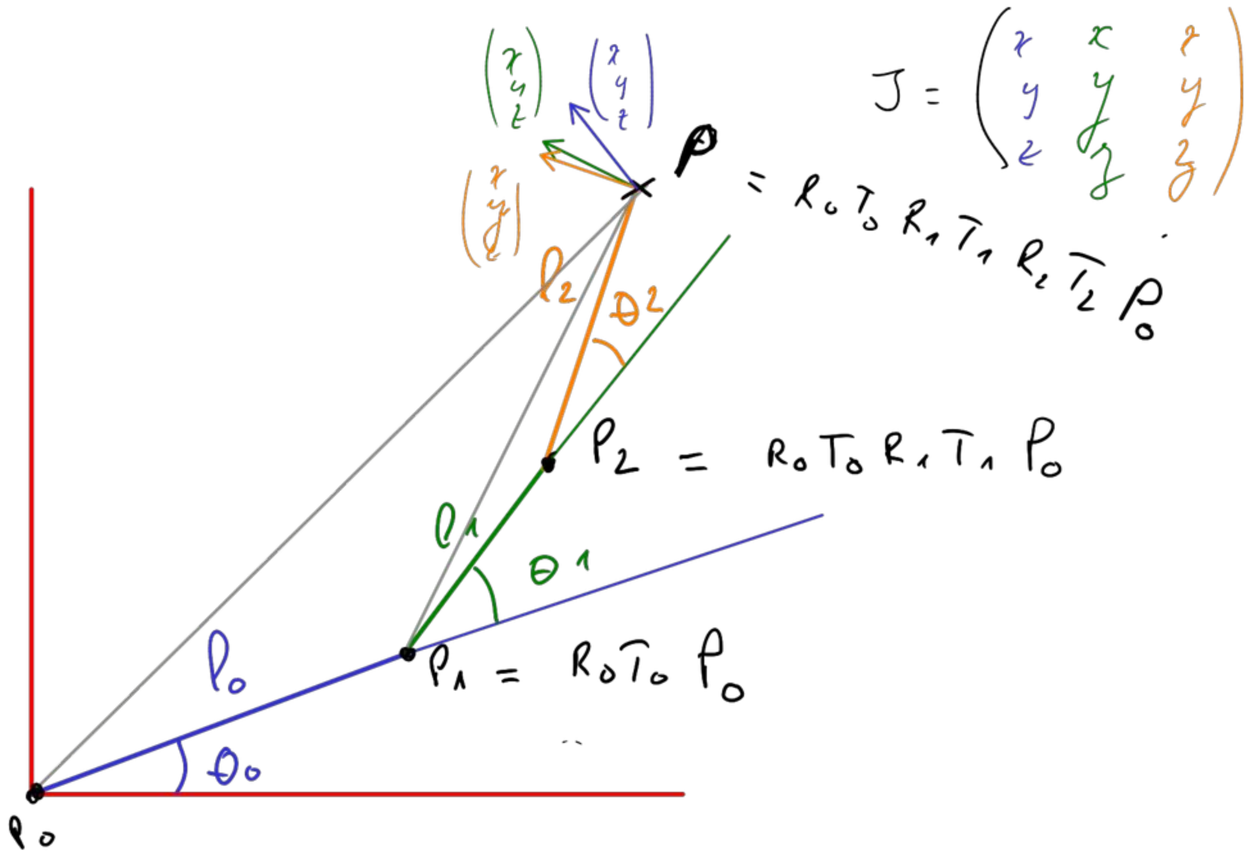
Jacobien

- Variation de $f(\Theta)$ au premier ordre
- Approximation linéaire de f
- Matrice 3*n (ou 2*n en 2D) *(n= nb de variables)*
- Calcul de **J**:
  - $f(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\mathbf{T}_2\mathbf{R}_3(\theta_3)\mathbf{T}_3 \ldots M_0$

$$\frac{\partial f}{\partial \theta_1} = \frac{\partial R_1}{\partial \theta_1} T_1 R_2 T_2 R_3 T_3 \cdots M_0$$

Pas si compliqué que ça
TP-VSP

Trop facile la jacobienne



Calcul du Jacobien dans R^3 (ou R^2)

- Jacobien d'une rotation autour de l'axe a_i (normalisé) passant par le pivot P_i :

$$a_i \times (E - P_i)$$

  - Attention :
    - (E-Pi) doit être exprimé dans le même repère que la cible (repère de la scène)
    - l'axe a_i et le pivot P_i dépendent de la configuration des parents (similaire au calcul des pivots).
- Jacobien d'une translation
  - Vecteur dans la direction de translation (attention config. parents)

Algorithme

```
inverseKinematics()
  {
    Vector Θ = getLinkParameters();
    Vector E = target - computeEndPoint();
     for(k=0; k<kmax && E.norm() > eps; k++){
      Matrix J = computeJacobian();
      Matrix J+ = pseudoInverse(J);
      Vector Λ = J+E ;
      if (max(Λ)>2) Λ *= 2/max(Λ);
      Θ = Θ+ Λ;
      putLinkParameters();
      E = target - computeEndPoint();
    }
  }
```

Alternatives et plus de contrôle

- ~~Pseudo-inverse de la Jacobienne~~ =>
  - Transposée de la Jacobienne
    - (+) Évite les pb de matrice non iversible ou mal conditionnée,
    - (-) Convergence plus lente
- Descente cyclique des coordonnées
  - Moins de calculs
  - Plus rapide mais moins précis
- Gestion des limites des articulations