# EFFICIENT GENERATION OF MOTION TRANSITIONS USING SPACETIME CONSTRAINTS

Discussion based on [RGBC96]

## 0 TABLE OF CONTENTS



Remarks: Maple V Release 3 has been used for all concrete implementations. Illustrations have been rendered using „Three Dimensional Dreams“ running on Oberon System 3 on Windows.

# 1 PROBLEM DISSECTION

## 1.1 Introduction

Imagine having a database consisting of a set of basic motions of a human body model (the authors of [RGBC96] relied on soccer motions). From the motion capturing process, the segments will usually be short and each one independent from the other. But the wish might come up to *concatenate* two arbitrary motion sequences, forming a *composite* motion. And that's what this discussion is all about: the *generation of motion transitions*. For an adequate illustration of the problem, take a look at figure 1-a.

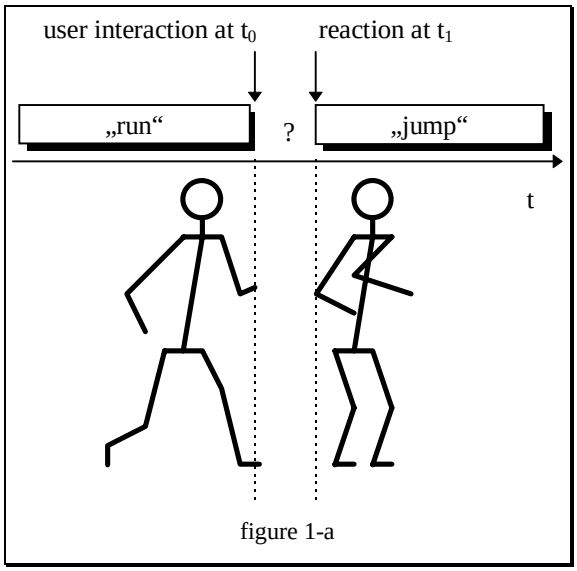



figure 1-a

## 1.2 Formal description

From the above stated introduction, one can pass over to a formal description:

$$\text{motionTransition} = \{\ b_{i,j} \mid q_i(t)=g(b_{i,1},...,b_{i,m}) \wedge \underline{q}(t_0)=\underline{q}_0 \wedge \underline{q}(t_1)=\underline{q}_1\ \}, \tag{1}$$

with

$q_i(\cdot)$ the $i^{th}$ joint angle function,
$b_{i,j}$ the parameters of the angle base function,
$g(\cdot)$ the angle base function,
$\underline{q}(\cdot)=(q_1(\cdot),...,q_n(\cdot))$ the vector of all joint angle functions,
$\underline{q}_0$ the value of $\underline{q}(\cdot)$ at $t_0$ and
$\underline{q}_1$ the value of $\underline{q}(\cdot)$ at $t_1$.

Obviously, (1) is a set with an infinite cardinality and so there is an arbitrary number of solutions to the problem. An application thus needs to generate a subset of (1).

# 2 SOLUTION DUE TO [RGBC96]

The authors of [RGBC96] restrict their algorithm to the set

$$\text{motionTransition}_{RGBC96} = \{\ \text{motionTransition} \mid e \rightarrow \min\ \}. \tag{1}$$

The *energy function* e is given as

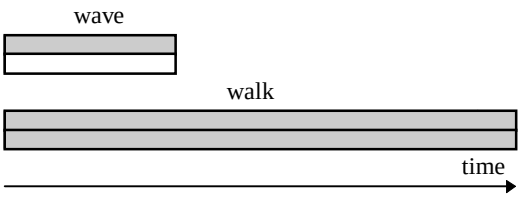
$$e := \int_{t_0}^{t_1} \sum_i \tau_i{}^2 dt \tag{2}$$

with the *torque* $\tau_i$ in joint i. That is: it computes the motion transition that requires the least energy, which is a formidable minimization task and is done by a *gradient based* method („BFGS“ [GMW81]). The authors remark that „*experience has shown that motion that minimizes energy looks natural*“.
Angle functions are represented from a cubic B-spline basis as they have, for the generated short transitions, shown good convergence properties. The use of B-spline wavlets has been taken into consideration but been rejected from the large number of basis functions in a single degree of freedom. On the other hand, five to ten B-spline coefficients suffice to represent the angular position.

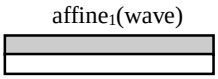
The problem of finding the required torque is known as the *inverse dynamics problem* [Ba91]. A solution according to [LWP80] serves as basic algorithm. From its vectorial form, [Ba91] presented a *tensorial* version which has actually been implemented and allows the computation of all unknowns with less computation effort from elegant Cartesian tensor identities.

## 2.1 Motion representation

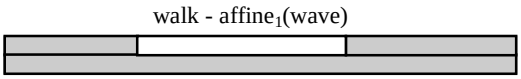
To make the user able to manipulate motions, break it into pieces and to reassemble it into new, more complex motions, a *motion expression language* has been defined. Let its application be explained from the example of combining an arm wave with a walk motion. The walk motion defines all degrees of freedom of the body model, whereas the arm wave does so for an arm only. Graphically, one might sketch the situation as follows, with filled bars denoting defined degrees of freedom:



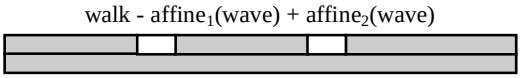

*Affine transformations* allow to move the wave motion in time:




Now the arm's degrees of freedom need to be undefined since they are fixed from the walk motion. For that purpose, the *minus operator* is provided.




Now the *addition* of a newly transformed wave with the modified walk motion offers two transition gaps which can be filled by the generation of a motion transition:




So the total resulting motion can symbolically be described as

$$SP(\text{walk} - \text{affine}_1(\text{wave}) + \text{affine}_2(\text{wave}))$$

with „SP" denoting the spacetime optimization. Motions are represented as a hierarchy of motion expressions. Motion expressions can be one of three types of objects: intervals, degrees of freedom (DOF) and motion units (MU). An interval is just a list of function values, whereas a DOF is a list of intervals which defines the degree of freedom of a joint for example for an arbitrary time. Finally, a motion unit is just an array of DOF's to describe m degrees of freedom of a body. In pseudo-BNF notation, these expressions can be stated as follows:

| | | |
|---|---|---|
| interval | := | $(f(t_0),...,f(t_{n-1}),t_0,t_{n-1}) \mid \varepsilon$ |
| DOF | := | interval \| DOF, interval \| ε |
| MU | := | array m OF DOF |

# 3 THE RESULTS OF THE [RGBC96] APPROACH

From the authors of [RGBC96], we get the following statement about the usefulness of their algorithm:

> „*The results of using our system to generate animations starting from a base library of soccer motions are quite good.*"

On the other hand, the generation of 0.6 seconds motion transition with a body model of 44 degrees of freedom takes 72 seconds on a 100 MHz Pentium. From that, the aim of interactivity has been missed. But one must say that the generated motions can be stored as persistent objects and from that be cached. The authors plan to extend their motion model to more accurately model the dynamics of the human body model since its is still not accurate for *free body motion*.

# 4 MOTIVATION OF THE ENERGY MINIMIZATION APPROACH [RGBC96]

## 4.1 Linear interpolation

I want to motivate the energy minimization approach [RGBC96] from a simple example. Imagine having the body model and border conditions as shown by figure 4-a.

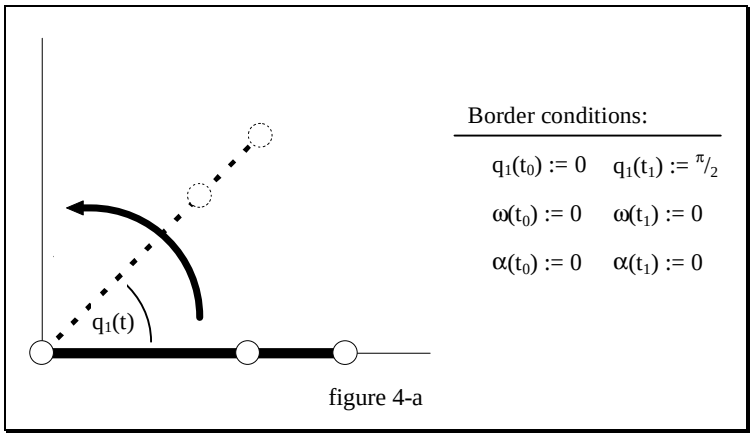


figure 4-a

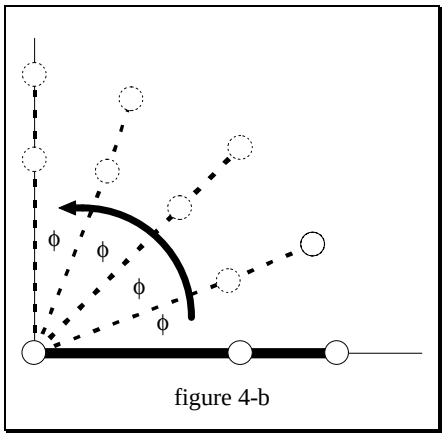


figure 4-b

For the sake of simplicity, I neglect all influences of gravitation. A trivial approach is interpolation. Interpolating linearly in $q_1$ results in the „motion“ depicted by figure 4-b. From the border conditions, we have

$$\lim_{t \to t_0^+} \alpha(t) = +\infty \,, \tag{1}$$

and thus this technique proves insufficient. The example will serve as demonstration body model in a subsequent section and it will be shown, that the minimization of the required total energy makes the motion look „natural“.

# 5 PROPOSED ALGORITHMS

## 5.1 Root motion

From the aspect of user interaction, the authors have chosen a straight forward way to implement the creature's root motion. The root's position for a time interval $[t_0,t_1]$ is computed from

$$\underline{x}(t) = \underline{x}(t_0) + \int_{t_0}^{t} \left\{ \underline{v}(t_0)\left(1 - \frac{\alpha - t_0}{t_1 - t_0}\right) + \underline{v}(t_1)\frac{\alpha - t_0}{t_1 - t_0} \right\} d\alpha \,. \tag{1}$$

From the vectorial velocity $\underline{v}$ given at discrete times $t_0$ and $t_1$, (1) gives the position vector $\underline{x}(t)$ from linear interpolation in velocity. Obviously, (1) induces a $C^1$ path. A $C^2$ motion could be obtained from interpolating linearly in the root motion's *acceleration*. (1) is explicitly solved as

$$\underline{x}(t) = \underline{v}(t_0)\left( (t - t_0) - \frac{\frac{1}{2}(t^2 - {t_0}^2) - t_0(t - t_0)}{t_1 - t_0} \right) + \underline{v}(t_1)\frac{\frac{1}{2}(t^2 - {t_0}^2) - t_0(t - t_0)}{t_1 - t_0} \,. \tag{2}$$

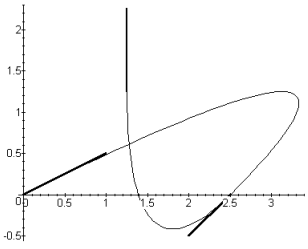


figure 5-a

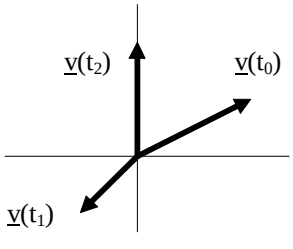


figure 5-b

As an example, consider the scenario of figure 5-b whose speed vectors have been defined as $\underline{v}(0):=(1,{}^1/_2)$, $\underline{v}(10):=(-{}^1/_2,-{}^1/_2)$ and $\underline{v}(15):=(0,1)$. What motion will result? From (2), one gets the composed root's motion (with initial condition $\underline{root}(0):=(0,0)$) as

$$\underline{root}(t) = \begin{cases} (t - \frac{3}{40}t^2, \frac{1}{2}t - \frac{1}{20}t^2) & (t \in [0,10]) \\ (\frac{25}{2} - \frac{3}{2}t + \frac{1}{20}t^2, -\frac{7}{20}t + \frac{3}{20}t^2 + 20) & (t \in [10,15]) \end{cases}, \tag{3}$$

whose graph is shown by figure 5-a.

## 5.2 Kinematic constraints

Any realistic visualization of human body motion lives from cinematic constraints. Aspects concerning *inverse* kinematics are considered from having a body model as dynamic chains starting at the root and ending at tips. From that, limbs are kept together from the specific motions computed from joint angle functions. A kinematic constraint could be the restriction to fix a certain part of the body at its place , e.g. a foot on the floor. Such points can be found as positions that are (almost) at the same place at the start and the end of the motion transition time interval and we designate them as *constrained coordinate frames*. But one needs to notice that those cannot be held at the desired position in any case, as figure 5-c shows: the given root motion prevents from keeping the constrained frame on the floor. When denoting the Euclidean length of the difference vector between desired and actual position as

$$r_k(t) = \left\| \underline{p}_k(t) - \hat{\underline{p}}_k(t) \right\|_2 , \tag{1}$$

kinematic constraints degenerate to the minimization of the total deviation R over the entire time interval:

$$R := \int_{t_0}^{t_1} \sum_{k=1}^{K} r_k(t)\,dt . \tag{2}$$

Note that the minimization of (2) merely influences the *support limb* as it can be seen from figure 5-c.

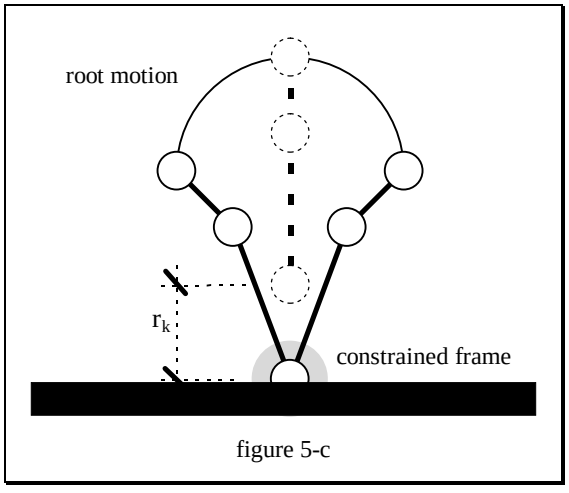


figure 5-c

## 5.3 Motion cyclification

Motion *cyclification* is a special case of motion transition generation which aims to concatenate a given motion with itself. From the fact that the motion will oftenly be of cyclic nature (think of a walking person), this problem is treated with less effort. As an example, let me here explain the application of the proposed algorithm on a single (scalar) angle function f(t) such as shown by figure 5-d.

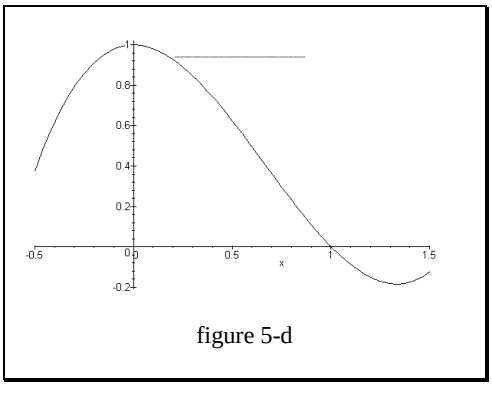


figure 5-d

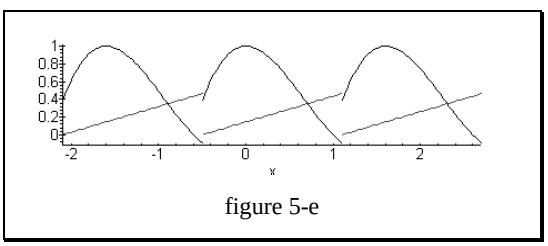


figure 5-e

Let's say the „motion" starts at $t_s$ and finishes at $t_f$. These two times are usually be defined by the user. The authors of [RGBC96] now define the connection intervals $I_s$ and $I_f$ each as to be one fifth of the entire length:

$$I_s = [t_s, t_s + 5^{-1}(t_f - t_f)] \quad (1)$$
$$I_f = [t_f - 5^{-1}(t_f - t_f), t_f]. \quad (2)$$

The approach is now to solve the minimization problem

$$\min_{\substack{t_0 \in I_s \\ t_1 \in I_f}} \left\| \underline{a} - \underline{b} \right\|_2 , \quad (3)$$

with $\underline{a}=[f(t_0),...,f^{(n-1)}(t_0)]^T$ and $\underline{b}=[f(t_1),...,f^{(n-1)}(t_1)]^T$. For n=0, one gets to

$$\min_{\substack{t_0 \in I_s \\ t_1 \in I_f}} \left| f(t_0) - f(t_1) \right| , \quad (4)$$

(minimization of difference in function values) which obviously has the solution $\{t_0=-^1/_2, t_1=^{11}/_{10}\}$ for the example of figure 4-d. The result is shown by figure 5-e: the encountered discontinuity is removed by distributing the error linearly over $[t_s,t_f]$, as the depicted lines show. Their superposition is given by figure 5-g.

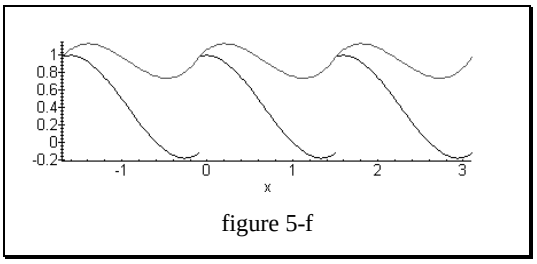



figure 5-f

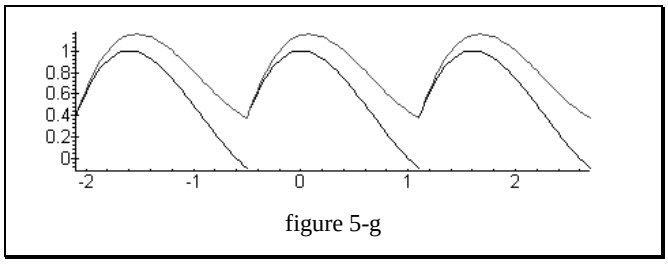



figure 5-g

For n=1, one needs to solve

$$\min_{\substack{t_0 \in I_s \\ t_1 \in I_f}} \sqrt{(f(t_0) - f(t_1))^2 + (\dot{f}(t_0) - \dot{f}(t_1))^2} . \quad (5)$$

The minimum is now obtained for $\{t_0=-^1/_{10}, t_1=^3/_2\}$. This leads to the more pleasant cyclified motion shown by figure 5-f. Note that the derivative fits by „accident", since nothing has been done to make the derivative continuous. The authors of [RGBC96] use a least squares cyclic B-spline approximation to construct a $C^2$ motion curve. As mentioned, one can even advance by considering f's higher derivatives.

### 5.3.1 Maple source text

```
# Motion cyclification by difference minimization (chk 97)
with( plots):
f := t->t^3-2*t^2+1;
t[s] := -1/2: t[f] := 3/2:
a := t[s] + 1/5*(t[f]-t[s]): b := t[f] - 1/5*(t[f]-t[s]):
plot( f(x), x=t[s]..t[f]);
# Considering only f-----------------------------------------------------------------------
start := t[s]: finish := b:
A := -(f(finish)-f(start))/(finish-start):
g := x->evalf(A)*x - start*A:
P1 := plot( f(x+(finish-start)), x=start-(finish-start)..start):
P1plus := plot( g(x+(finish-start)), x=start-(finish-start)..start, color=red):
P2 := plot( f(x), x=start..finish):
P2plus := plot( g(x), x=start..finish, color=red):
P3 := plot( f(x-(finish-start)), x=finish..2*finish-start):
P3plus := plot( g(x-(finish-start)), x=finish..2*finish-start, color=red):
display( {P1,P1plus,P2,P2plus,P3,P3plus});
P1new := plot( f(x+(finish-start))+g(x+(finish-start)), x=start-(finish-start)..start):
P2new := plot( f(x)+g(x), x=start..finish, color=red):
P3new := plot( f(x-(finish-start))+g(x-(finish-start)), x=finish..2*finish-start):
display( {P1,P2,P3,P1new,P2new,P3new});
# Considering f and df/dt-----------------------------------------------------------------
start := a: finish := t[f]:
A := -(f(finish)-f(start))/(finish-start):
g := x->evalf(A)*x - start*A:
P1 := plot( f(x+(finish-start)), x=start-(finish-start)..start):
P1plus := plot( g(x+(finish-start)), x=start-(finish-start)..start, color=red):
P2 := plot( f(x), x=start..finish):
P2plus := plot( g(x), x=start..finish, color=red):
P3 := plot( f(x-(finish-start)), x=finish..2*finish-start):
P3plus := plot( g(x-(finish-start)), x=finish..2*finish-start, color=red):
display( {P1,P1plus,P2,P2plus,P3,P3plus});
P1new := plot( f(x+(finish-start))+g(x+(finish-start)), x=start-(finish-start)..start):
P2new := plot( f(x)+g(x), x=start..finish, color=red):
P3new := plot( f(x-(finish-start))+g(x-(finish-start)), x=finish..2*finish-start):
display( {P1,P2,P3,P1new,P2new,P3new});
```

## 5.4 The inverse dynamics problem (IDP) due to [Ba,LWP]

Due to [Ba91], the *inverse dynamics problem* is task of finding a generalized force vector $\underline{\tau}$ from a given *motion trajectory* of a body model. Formally, it can be described as

$$\tau = f(q, \dot{q}, \ddot{q}) . \quad (1)$$

The dual problem (that is: finding the induced motion from all forces acting on the model) is called the *forward* or *direct dynamics problem*. The following sections state the recursive algorithm due to [LWP80] which proceeds in two steps:

a) Propagation of velocities, accelerations and moments from root to tips (*forward* dynamics equations)
b) Computation of total forces and torque from tips to root (*backward* dynamics equations)

From that, an algorithmic formulation of the process could be stated as

```
PROCEDURE IDP( me : root);
BEGIN
    forwardDynamicsEquations(me);
    FOR all successors s of me DO IDP(s) END;
    backwardDynamicsEquations(me)
END IDP;
```

The casual context of the computed variables is shown by figure 5-h. It also depicts the separation into forward and backward dynamics equations.

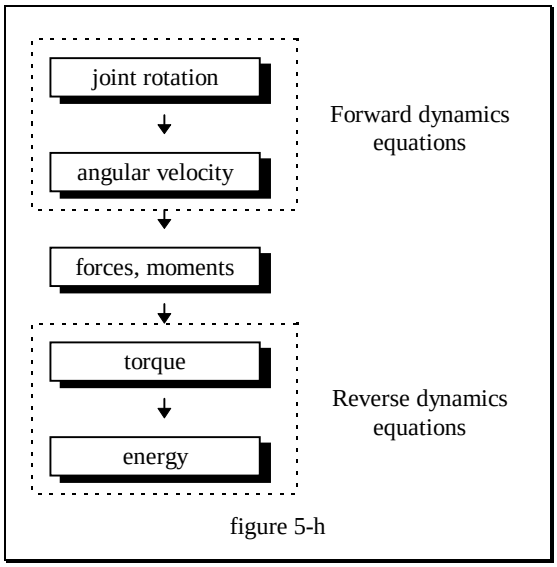


figure 5-h

### 5.4.1 Notation and definitions

From the hierarchical property of the body model, notation quickly becomes confusing.

$$\underline{\text{vector}}^{\text{with respect to which link}}$$

The usage of sub- and superscripts has just been demonstrated. Further, an index i+ denotes the link one position further from root[1] as link i, whereas i- does the same in the reversed way (closer to the root). To express the vector product as a matrix multiplication, the *dual* operator needs to be introduced which is, for a vector $\underline{u}$ given as

$$\tilde{\underline{u}} = \text{dual}\left(\begin{bmatrix}\underline{u}[0]\\ \underline{u}[1]\\ \underline{u}[2]\end{bmatrix}\right) = \begin{bmatrix}0 & -\underline{u}[2] & \underline{u}[1]\\ \underline{u}[2] & 0 & -\underline{u}[0]\\ -\underline{u}[1] & \underline{u}[0] & 0\end{bmatrix} \tag{1}$$

It assigns a vector a skew-symmetric Cartesian tensor of second order [Ba91]. One easily recognizes that the identity

$$\underline{u} \times \underline{v} \equiv \tilde{\underline{u}} \cdot \underline{v} \tag{2}$$

is valid. This is the primary relationship of the algorithm's vectorial to the tensorial version. Also, the *inertia tensor* about the center of mass can be written as

$$\underline{\underline{I}}_c = -\int_m \tilde{\underline{r}}\,\tilde{\underline{r}}\,dm \tag{3}$$

with $\underline{r}$ denoting the position vector of the infinitesimal mass element „dm".

### 5.4.2 Link parameters (simplified)

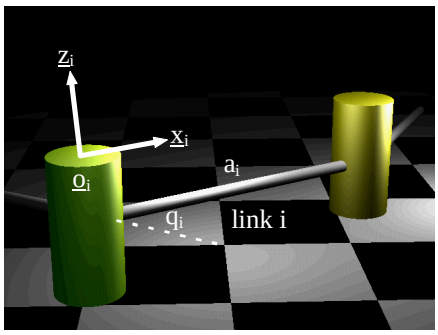


figure 5-j

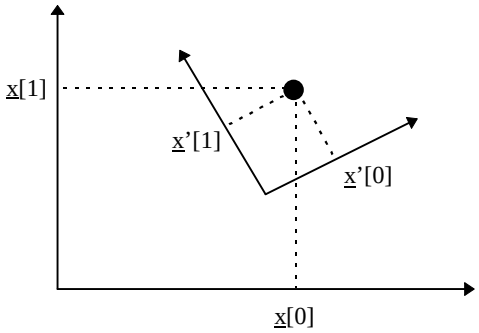


figure 5-i

The figure 5-j shows an arbitrary link i within the dynamic chain. Rotation axis' are visualized by the vertical cylinders. From above made definitions, the coordinate transformation $A_i$ of coordinates with respect to the $i+^{th}$ coordinate frame to the $i^{th}$ coordinate frame can be written as the subsequent execution of a rotation and a translation [Ba91]:

[1] With respect to the dynamic chain

$$A_{i+}{}^{T} = \mathrm{Rot}(\underline{z}_i, q_{i+}); \mathrm{Trans}(\underline{x}_i, a_i); \tag{1}$$

(1) reminds us of homogeneous coordinate transformations and the matrix $A_{i+}$ is in fact from figure 5-i trivially obtained as

$$A_{i+} = \begin{bmatrix} \cos(q_{i+}) & -\sin(q_{i+}) & 0 & a_i \\ \sin(q_{i+}) & \cos(q_{i+}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The reader is referred to [Ba91] for a more generalized treatise (such as prismatic joints or arbitrary axis directions) of the topic. Further, for some reasons that will become obvious later, when the body model includes no prismatic joints, only the rotation part of $A_{i+}$ must be considered.

### 5.4.3 Involved symbols and semantics

The figure 5-k summarizes the introduced notation with all relevant symbols used in the algorithm. Variables concerning e.g. the center of mass haven been placed in the box nearby it.

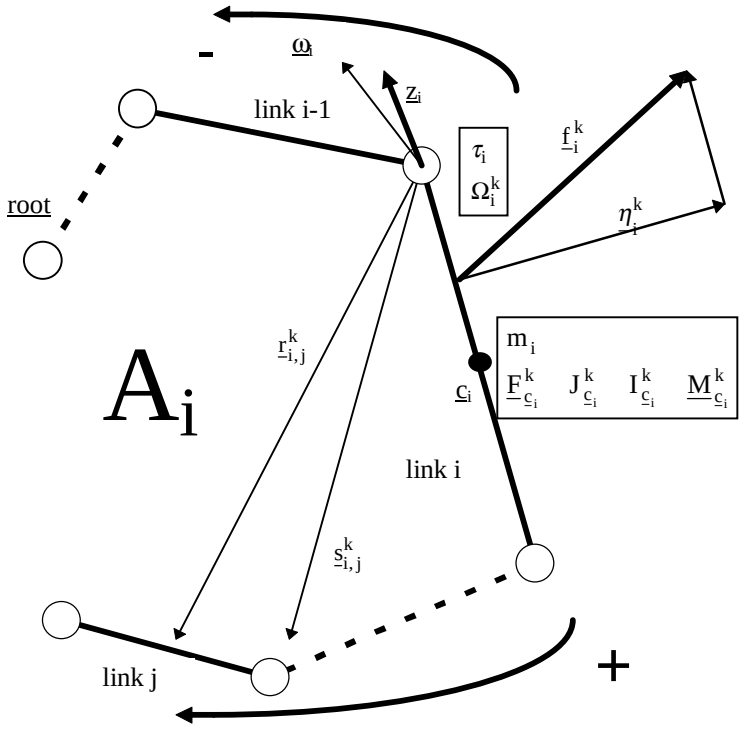



figure 5-k

$\tau_i$ torque in joint i
$\underline{z}_i$ joint axis i
$\underline{\omega}_i$ angular velocity of $i^{th}$ link
$\underline{\Omega}_{c_i}^k$ angular acceleration tensor $i^{th}$ link

---

$\underline{f}_i^k$ force vector exerted on link i by i-1
$\underline{\eta}_i^k$ moment vector exerted on link i by i-1

---

$\underline{c}_i$ center of mass of the $i^{th}$ link
$m_i$ mass of the $i^{th}$ link
$\underline{F}_{c_i}^k$ force vector acting on $\underline{c}_i$
$\underline{I}_{c_i}^k$ inertia tensor of the $i^{th}$ frame about $\underline{c}_i$
$\underline{M}_{c_i}^k$ moment vector about $\underline{c}_i$
$\underline{J}_{c_i}^k$ Euler's inertia tensor about $\underline{c}_i$

### 5.4.4 Border conditions of the example motion: the polynominal basis approach

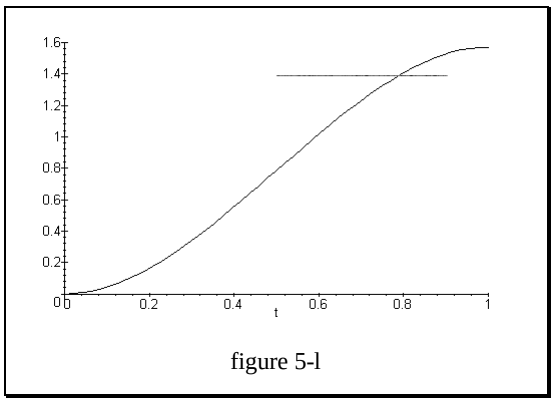



figure 5-l

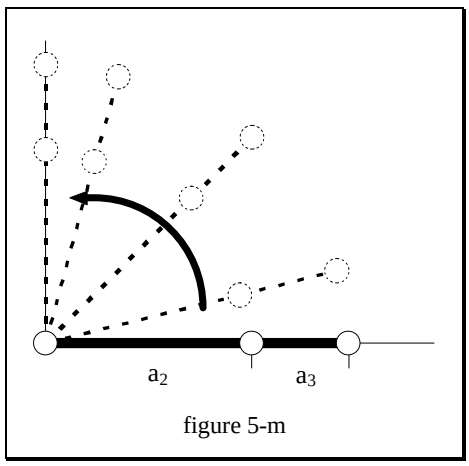



figure 5-m

The „motion“ introduced by figure 4-b lacks a continuous acceleration. I therefore set up an angle function of the form shown by figure 5-l. From the border conditions, Maple calculates the polynomial

$$q_1(t) = -\pi \cdot t^3 + {}^3\!/_2 \cdot \pi \cdot t^2 \qquad (1)$$

This results in rotational position as shown by figure 5-m. The second angle function $q_2(\cdot)$ is obviously $q_2(t) \equiv 0$. In the following sections, in parallel with the explanation of the dynamics equations, their effect to the example body model of figure 5-m will be observed, whose static conditions are given by

1. root motion ..... $\underline{\omega}_0^0(t) = \underline{\dot{\omega}}_0^0(t) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, $\underline{\ddot{s}}_{0,0}^0(t) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$

2. joint axis' ..... $\underline{z}_0^0(t) = \underline{z}_1^1(t) = \underline{z}_2^2(t) = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$

3. joint lengths..... $\underline{s}_{0,1}^0 = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, $\underline{s}_{1,2}^1 = \begin{bmatrix}a_2 & 0 & 0\end{bmatrix}^T := \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$, $\underline{s}_{2,3}^2 = \begin{bmatrix}a_3 & 0 & 0\end{bmatrix}^T := \begin{bmatrix}1/2 & 0 & 0\end{bmatrix}^T$

4. limbs' center of mass..... $\underline{r}_{1,1}^1 = \underline{s}_{1,2}^1 / 2 = \begin{bmatrix}1/2 & 0 & 0\end{bmatrix}^T$, $\underline{r}_{2,2}^2 = \underline{s}_{2,3}^2 / 2 = \begin{bmatrix}1/4 & 0 & 0\end{bmatrix}^T$

5. coordinate transformation ... $A_1 = \begin{bmatrix}\cos(-\pi \cdot t^3 + 3/2 \cdot \pi \cdot t^2) & -\sin(-\pi \cdot t^3 + 3/2 \cdot \pi \cdot t^2) & 0 \\ \sin(-\pi \cdot t^3 + 3/2 \cdot \pi \cdot t^2) & \cos(-\pi \cdot t^3 + 3/2 \cdot \pi \cdot t^2) & 0 \\ 0 & 0 & 1\end{bmatrix}$, $A_2 = \begin{bmatrix}\cos(0) & -\sin(0) & 0 \\ \sin(0) & \cos(0) & 0 \\ 0 & 0 & 1\end{bmatrix}$

6. mass..... $m_1 := 1$, $m_2 := 1/2$

### 5.4.5 Forward dynamics equations

In the following, the *forward* dynamics equations are explained. The absolute angular velocity of the i[th] coordinate frame is given by the (transformed) rotation induced by the previous coordinate system plus the change in i+'s joint angle function $\underline{q}_{i+}$:

$$\boxed{\underline{\omega}_{i+}^{i+} = \underbrace{A_{i+}{}^T \underline{\omega}_i^i}_{\text{link i}} + \underbrace{\underline{z}_{i+}^{i+} \dot{q}_{i+}}_{\text{joint rotation link i+}}}$$

$$\underline{\omega}_1^1(t) = \begin{bmatrix}0 & 0 & -3\pi \cdot t^2 + 3 \cdot \pi \cdot t\end{bmatrix}^T$$

$$\underline{\omega}_2^2(t) = \begin{bmatrix}0 & 0 & -3\pi \cdot t^2 + 3\pi \cdot t\end{bmatrix}^T$$

The next formula is obtained by calculating the derivative:

$$\boxed{\underline{\dot{\omega}}_{i+}^{i+} = A_{i+}{}^T \left[\underline{\dot{\omega}}_i^i + \underline{\omega}_i^i \times \underline{z}_{i+}^{i+} \dot{q}_{i+}\right] + \underline{z}_{i+}^{i+} \ddot{q}_{i+}}$$

$$\underline{\dot{\omega}}_1^1(t) = \begin{bmatrix}0 & 0 & -6\pi \cdot t + 3\pi\end{bmatrix}^T$$

$$\underline{\dot{\omega}}_2^2(t) = \begin{bmatrix}0 & 0 & -6\pi \cdot t + 3\pi\end{bmatrix}^T$$

The acceleration of the i+[th] coordinate frame is given as the coordinate transformed sum of the acceleration of the i[th] coordinate frame plus the acceleration induced by the rotation of i plus the centrifugal acceleration caused by i's rotation:

$$\boxed{\underline{\ddot{s}}_{0,i+}^{i+} = A_{i+}^T [\underline{\ddot{s}}_{0,i}^i + \underline{\dot{\omega}}_i^i \times \underline{s}_{i,i+}^i + \underline{\omega}_i^i \times \left(\underline{\omega}_i^i \times \underline{s}_{i,i+}^i\right)]}$$

$$\underline{\ddot{s}}_{0,1}^1 = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$$

Note the signification of this vector's components: the first one is caused by *centrifugal* acceleration and the second one by the body's *inertia*! From figure 5-m, it can be seen that the components exactly formulate the physical reality: the centrifugal acceleration is 0 for t=0, has a maximum for t=0.5 and returns to be zero for t=1. The acceleration on the other hand is a linear function with negative derivative and also shows symmetry with respect to t=0.5. As the next step, the acceleration of the i+[th] link's center of mass needs to be calculated. It is given by the acceleration of the coordinate system's origin plus the acceleration caused by the rotation of the i+[th] joint plus the centrifugal acceleration caused by the same rotation:

$$\underline{\ddot{s}}_{0,2}^2 = \begin{bmatrix}-(-3\pi t^2 + 3\pi t)^2 \\ -6\pi t + 3\pi \\ 0\end{bmatrix}^T$$

$$\boxed{\underline{\ddot{r}}_{0,i+}^{i+} = \underline{\ddot{s}}_{0,i+}^{i+} + \underline{\dot{\omega}}_{i+}^{i+} \times \underline{r}_{i+,i+}^{i+} + \underline{\omega}_{i+}^{i+} \times \left(\underline{\omega}_{i+}^{i+} \times \underline{r}_{i+,i+}^{i+}\right)}$$

...[2]

The next formula calculates the force vector acting on the center of mass of the i+[th] link according to Newton's second law:

$$\boxed{\underline{F}_{\underline{c}_{i+}}^{i+} = m_{i+} \underline{\ddot{r}}_{0,i+}^{i+}}$$

The final forward dynamics equation is Euler's equation to get the moment about the i+[th] center of mass:

[2] the expressions become rather complex

$$\underline{M}_{\underline{c}_{i+}}^{i+} = I_{\underline{c}_{i+}}^{i+} \dot{\underline{\omega}}_{i+}^{i+} + \underline{\omega}_{i+}^{i+} \times I_{\underline{c}_{i+}}^{i+} \underline{\omega}_{i+}^{i+}$$

### 5.4.6 Backward dynamics equations

The force vector exerted on link i by link i-1 is given as a superposition of the transformed force vector exerted on link i+ by link i and the force vector acting on the center of mass of the $i^{th}$ link:

$$\underline{f}_i^i = A_{i+} \underline{f}_{i+}^{i+} + \underline{F}_{\underline{c}_i}^i$$

The moment vector exerted on link i by link i-1 is given as the moment acting on i's center of mass, plus the radial components of $\underline{f}_{i+}$ plus the transformed moment vector on i+:

$$\underline{\eta}_i^i = \underline{M}_{\underline{c}_i}^i + \underline{s}_{i,i+}^i \times A_{i+} \underline{f}_{i+}^{i+} + \underline{r}_{i,i}^i \times \underline{F}_{\underline{c}_i}^i + A_{i+} \underline{\eta}_{i+}^{i+}$$

Assuming the joint axis vector's length to be one, the torque can be computed as orthogonal projection of the moment vector on it:

$$\tau_i = \underline{\eta}_i^i \cdot \underline{z}_i^i$$

From that, the total energy of the introduced example motion is computed as to be 5.89. Remark: I have here explained the *vectorial* version [LWP80]. The *tensorial* version [Ba91] allows the equations to be written more elegantly, resulting in a reduced amount of computation power.

$$e^* := \int_0^1 \sum_{i \in \{1,2\}} |\tau_i| dt = 5.89$$

### 5.4.7 Double polynominal angle function basis

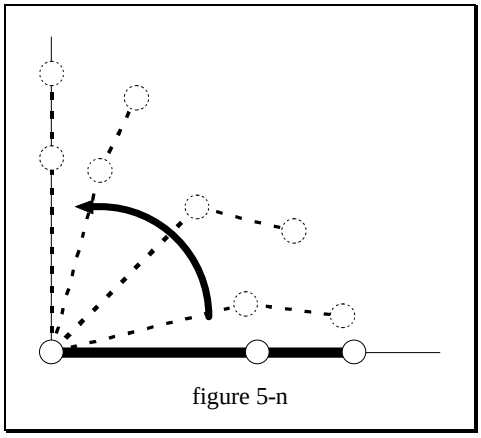


figure 5-n

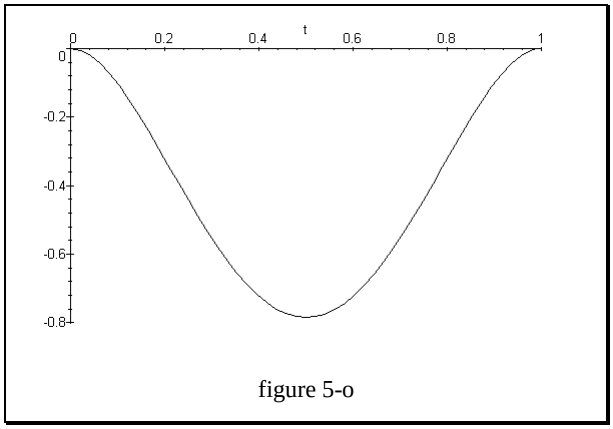



figure 5-o

But the motion still does not look „natural". What we intuitively want see is something as shown by figure 5-n. I therefore set up the joint angle function $q_2$ as follows:

$$q_2(t) := -4\pi t^4 + 8\pi t^3 - 4\pi t^2.$$

The border conditions remain. Doing the complete calculus (dynamics equations) yields

$$e^* := \int_0^1 \sum_{i \in \{1,2\}} |\tau_i| dt = 5.50 .$$

The amount of required energy has been reduced! We have seen that the minimization of energy induces „natural" motions. The torque functions for the scenarios of figure 5-m and figure 5-n for both joints are shown by figure 5-q and figure 5-p respectively.

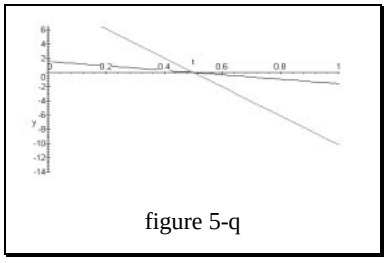


figure 5-q

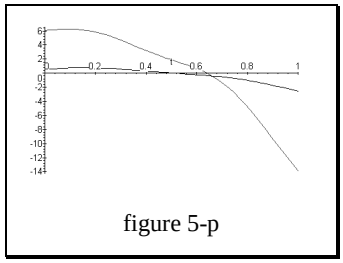


figure 5-p

### 5.4.8 Maple source text

```
# Inverse dynamics problem for a example body model due to Luh, Walker and Paul (chk 11.06.97)

x := proc(a,b) # Overload the 'crossprod' operator
  if a=0 or b=0 then vector([0,0,0]) else crossprod(a,b) fi
end;

with(linalg):
omega[0,0] := vector([0,0,0]); omega_dot[0,0] := vector([0,0,0]);
s_dot_dot[0,0,0] := vector([0,0,0]);
z[0,0] := vector([0,0,1]); z[1,1] := vector([0,0,1]); z[2,2] := vector([0,0,1]);
s[0,1,0] := vector([0,0,0]); s[1,2,1] := vector([1,0,0]); s[2,3,2] := vector([0.5,0,0]);
r[1,1,1] := vector([0.5,0,0]); r[2,2,2] := vector([0.25,0,0]);
q[1] := -Pi*t^3+3/2*Pi*t^2; # q[1] := 3*Pi*t^5-15/2*Pi*t^4+5*Pi*t^3; # with q[1]"(0)=q[1]"(1)=0
m[1] := 1.0; m[2] := 0.5;
I_[c1,1] := matrix([[0,0,0],[0,evalf(1/12*m[1]*r[1,1,1][1]),0],[0,0,evalf(1/12*m[1]*r[1,1,1][1])]]);
I_[c2,2] := matrix([[0,0,0],[0,evalf(1/12*m[2]*r[2,2,2][1]),0],[0,0,evalf(1/12*m[2]*r[2,2,2][1])]]);

g := x-> c[0]*x^4+ (-2*c[0])*x^3+c[0]*x^2; #basis for q[2]
coefficients := { c[0]=0 }; n := 1; #number of coefficients

temp := diff(g(t),t);
g_dot := x-> eval(subs(t=x,temp));
A[1] := matrix([[cos(q[1]),-sin(q[1]),0],[sin(q[1]),cos(q[1]),0],[0,0,1]]):
q[2] := g(t): A[2] := matrix([[cos(q[2]),-sin(q[2]),0],[sin(q[2]),cos(q[2]),0],[0,0,1]]):

# FORWARD RECURSION

omega[1,1] := evalm(transpose(A[1])&*omega[0,0]+z[1,1]*diff(q[1],t));
omega[2,2] := evalm(transpose(A[2])&*omega[1,1]+z[2,2]*diff(q[2],t));
omega_dot[1,1] := evalm(transpose(A[1])&*(omega_dot[0,0]+x(omega[0,0],(z[1,1]*diff(q[1],t))))+z[1,1]*diff(q[1],t$2));
omega_dot[2,2]:= evalm(transpose(A[2])&*(omega_dot[1,1]+x(omega[1,1],(z[2,2]*diff(q[2],t))))+z[2,2]*diff(q[2],t$2));
s_dot_dot[0,1,1]:= evalm(transpose(A[1])&*(s_dot_dot[0,0,0]+x(omega_dot[0,0],s[0,1,0])+x(omega[0,0],x(omega[0,0],s[0,1,0]))));
s_dot_dot[0,2,2]:= evalm(transpose(A[2])&*(s_dot_dot[0,1,1]+x(omega_dot[1,1],s[1,2,1])+x(omega[1,1],x(omega[1,1],s[1,2,1]))));
r_dot_dot[0,1,1] := evalm(s_dot_dot[0,1,1]+x(omega_dot[1,1],r[1,1,1])+x(omega[1,1],x(omega[1,1],r[1,1,1])));
r_dot_dot[0,2,2] := evalm(s_dot_dot[0,2,2]+x(omega_dot[2,2],r[2,2,2])+x(omega[2,2],x(omega[2,2],r[2,2,2])));
F[c1,1] := evalm(m[1]*r_dot_dot[0,1,1]); F[c2,2] := evalm(m[2]*r_dot_dot[0,2,2]);
M[c1,1] := evalm(I_[c1,1]&*omega_dot[1,1]+x(omega[1,1],I_[c1,1]&*omega[1,1]));
M[c2,2] := evalm(I_[c2,2]&*omega_dot[2,2]+x(omega[2,2],I_[c2,2]&*omega[2,2]));

# BACKWARD RECURSION

f[2,2] := evalm(F[c2,2]); # A[3],f[3,3] is not considered since it is 0 in this case
f[1,1] := evalm(A[2]&*f[2,2] + F[c1,1]);
eta[2,2] := evalm(M[c2,2]+x(r[2,2,2],F[c2,2])); #A[3],f[3,3],eta[3,3] is not considered
eta[1,1] := evalm(M[c1,1]+x(s[1,2,1],A[2]&*f[2,2])+x(r[1,1,1],F[c1,1])+A[2]&*eta[2,2]);
tau[2] := simplify(dotprod(eta[2,2],z[2,2]));
tau[1] := simplify(dotprod(eta[1,1],z[1,1]));

# ENERGY CONSIDERATION

unassign('d_tau'):
d_tau[2] := grad(tau[2],[seq(c[n-i],i=1..n)]): # tau[2]'s gradient
d_tau[1] := grad(tau[1],[seq(c[n-i],i=1..n)]): # tau[1]'s gradient
unassign('tau_d_tau'):
tau_d_tau[2] := [seq(subs(coefficients,tau[2])*subs(coefficients, d_tau[2][i]),i=1..n)]:
tau_d_tau[1] := [seq(subs(coefficients,tau[1])*subs(coefficients, d_tau[1][i]),i=1..n)]:

q[2] := subs(coefficients,g(t)); # q[2] from coefficients
plot(q[2], t=0..1);
tau_concrete[2] := simplify(subs(coefficients, tau[2])):
tau_concrete[1] := simplify(subs(coefficients, tau[1])):
plot({tau_concrete[1],tau_concrete[2]},t=0..1,y=-14..6.5);

Digits := 3; e := evalf(int[numeric](simplify(abs(tau_concrete[2])+abs(tau_concrete[1])),t=0..1)); # seems to be correct: ^1 yields e=0

d_tau_sum := [seq(simplify(2*tau_d_tau[2][i]+2*tau_d_tau[1][i]),i=1..n)]:
grad_e := [seq(evalf(int[numeric](d_tau_sum[i],t=0..1)),i=1..n)];

with(plots):
A[2] := matrix([[cos(q[2]),-sin(q[2]),0],[sin(q[2]),cos(q[2]),0],[0,0,1]]):
steps := 25;
flic := []:
for i from 0 to steps do
  t := i/steps;
  temp1 := evalm(A[1]&*s[1,2,1]);
  temp3 := evalm(A[1]&*(evalm(A[2]&*s[2,3,2])));
  temp2 := PLOT(CURVES([
     [0,0],[evalf(temp1[1]),evalf(temp1[2])],
     [evalf(temp1[1]),evalf(temp1[2])],[evalf(temp1[1]+temp3[1]),evalf(temp1[2]+temp3[2])]
   ],THICKNESS(4))):
  flic := [seq(flic[j],j=1..nops(flic)), temp2]:
od:
t := 't': display(flic, insequence = true);
```

# 6 SELECTED HINTS

## 6.1 Optimization in question

The following question might eventually arise: *„But why is the generation of a motion transition a minimization problem at all?“*. From the physical point of view, the energy required to get from the end position of the first to the starting position of the second motion is given by the difference in potential and kinetic energy. This view assumes that kinetic energy can be regained from slowing down a motion.
The solution presented in [RGBC96] defines other conditions from the definition of the energy function. By squaring each joint's torque, the squared *absolute torque* is considered as total effort. The question is, whether this definition really makes sense. For human muscles, it is a correct consideration since they exert power in only *one direction*, being unable to regain energy from slowing down the induced motion, which needs to be carried out by another muscle.

## 6.2 Gradient computation

From the application of a gradient-based minimization algorithm, the need for computing the gradient of the energy function

$$e := \int_{t_0}^{t_1} \sum_i \tau_i{}^2 dt \tag{1}$$

comes up. Since the optimization parameters is the set of coefficients of the angular motion basis chosen, the gradient

$$\nabla e = [\partial e/\partial q_1, \ldots, \partial e/\partial q_n]^T \tag{2}$$

is symbolically stated as vector of partial derivatives with respect to the angle functions $q_i$. Evaluating the partial derivatives yields (from the application of the chain rule):

$$\frac{\partial e}{\partial q_i} = \frac{\partial \int_{t_0}^{t_1} \sum_i \tau_i^2 dt}{\partial q_i} = 2 \int_{t_0}^{t_1} \sum_i \tau_i \frac{\partial \tau_i}{\partial q_i} dt \tag{3}$$

What luck that the derivative can be taken *into* the integral! Without this property, we wouldn't even be *able* to compute the gradient: this comes from the computation of the torque. Its analytical form includes functions of the type

$$\int \cos(q_i(t)) dt \tag{4}$$

from the coordinate transformation matrix $A_i$. But we know that there is no closed form for such types of integrals, since the base functions $q_i(t)$ are parametrized with base coefficients. Thus, the gradient can *not* be computed analytically. It can only be evaluated at *discrete positions* according to the following cook's recipe:

1. Compute $\tau_i$'s partial derivative
2. Insert the actual set of base coefficients
3. Sum up and integrate

## 6.3 Additive property of angular position and acceleration

It might be unclear, why a coordinate frame's angular velocity is given as the superposition of an angular velocity and the rotation of the coordinate frame itself (first dynamics equation):

$$\underline{\omega}_{i+}^{i+} = A_{i+}{}^T \underline{\omega}_i^i + \underline{z}_{i+}^{i+} \dot{q}_{i+} \,. \tag{1}$$

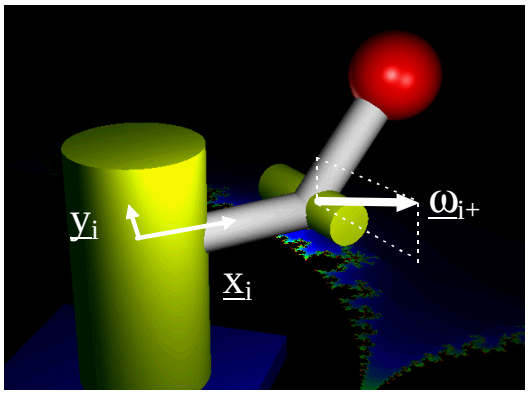



figure 6-a

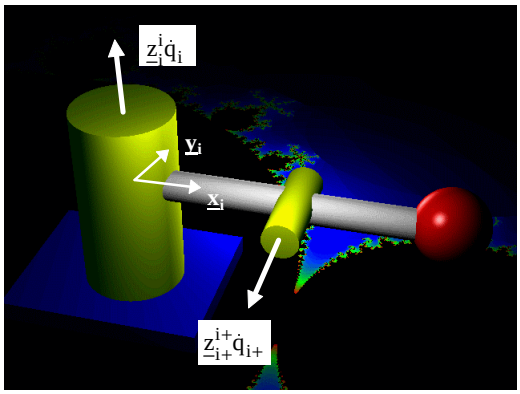



figure 6-b

Take a look at the scenario of figure 6-b. It depicts a simple robot with two revolute joints and a red sphere as manipulator. Imagine having constant rotational velocities as inscribed. Then the situation after some delay might look as shown by figure 6-a.

What *total* angular velocity $\underline{\omega}_{i+}$ is necessary to turn the $i+^{th}$ link into the depicted position? The total angular velocity $\underline{\omega}_{i+}$ of the $i+^{th}$ coordinate frame is given as superposition of i's angular velocity and the rotation speed of the $i+^{th}$ coordinate frame: its vertical components results from the rotation of the $i^{th}$ coordinate frame whereas the horizontal component results from the rotation of the $i+^{th}$ joint. From that I state the following observation: the angular velocity does *not* need to be parallel to the joint axis.

## 6.4 *„Why just $A^T$?“*

The fact that not A's inverse but only a transposed matrix $A^T$ has been used in the dynamics equations. As mentioned, $A^T$ stands for the transpose of the rotation part of the homogeneous coordinate transformation

$$A(t) = \begin{bmatrix} \text{base}(t) & t \\ 0 & 1 \end{bmatrix}. \tag{1}$$

In the reverse dynamics equations, expressions invoking A(t) always have the form

$$\underline{y}'(t) = A(t) \cdot \underline{y}(t) \tag{2}$$

and $\underline{y}(t)$ is always the *derivative with respect to time* of some vector $A(t) \cdot \underline{x}(t)$. This comes from the fact that the dynamics equations do not calculate positions but velocities and accelerations. Thus, (2) can be written as

$$\underline{y}'(t) =: \frac{\partial}{\partial t}\begin{bmatrix} \text{base}(t) & t \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \underline{x}(t) \\ 1 \end{bmatrix} = \frac{\partial}{\partial t}\begin{bmatrix} \text{base}(t) \cdot \underline{x}(t) \\ 0 \end{bmatrix} = \frac{\partial}{\partial t}\begin{bmatrix} \text{base}(t) & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \underline{x}(t) \\ 1 \end{bmatrix}. \tag{3}$$

That is: the last vector component is redundant since t is a constant (no prismatic joints). Thus, from removing the heterogeneous vector component, [base(t)] can be used as homogeneous coordinate transformation. From the orthogonal property of the base, the inverse is obtained as the transpose. Note that these simplifications are only valid under the preconditions that a) no prismatic joints are used and b) no position vectors are computed.

# 7 GLOSSARY

| | |
|---|---|
| *„revolute joint“* | joint with *no* translation facility |
| *„prismatic joint“* | joint *with* translation facility |
| *„coordinate frame“* | object coordinate system |
| *„support point“* | constrained *coordinate frame* of the body model |
| *„support limb“* | kinematic chain from the *support point* back up the kinematic tree to the root |
| *„reverse dynamics equations“* | backward cynamics equations |

# 8 REFERENCES


[RGBC96] C. Rose, B. Guenther, B. Bodenheimer, M. Cohen: *„Efficient generation of motion transitions using spacetime constraints“*, proceedings of Siggraph (1996)

[Ba91] C. A. Balafoutis: *„Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach“*, Kluwer Academic Publishers (1991)

[LWP80] J. Y. S. Luh, M. W. Walker and R.P. Paul: *„On-Line Computational Scheme for Mechanical manipulators“*, ASME J. Dyn. Syst. Meas. And Contr. Vol 102, pp. 69-79 (1980)

[GMW81] P. Gill, W. Murray, M. Wright: *„Practical Optimization“*, Academic Press (1981)