meso:
Simulated Muscles for a Humanoid Robot

Matthew J. Marjanović
Humanoid Robotics Group
MIT Artificial Intelligence Lab

or

“Trying to make a hard thing easier, by making it much harder.”

The Need for Human-like Motor Control

Cog is an anthropomorphic robot. The goal is to explore mechanisms for generating and learning social behavior.

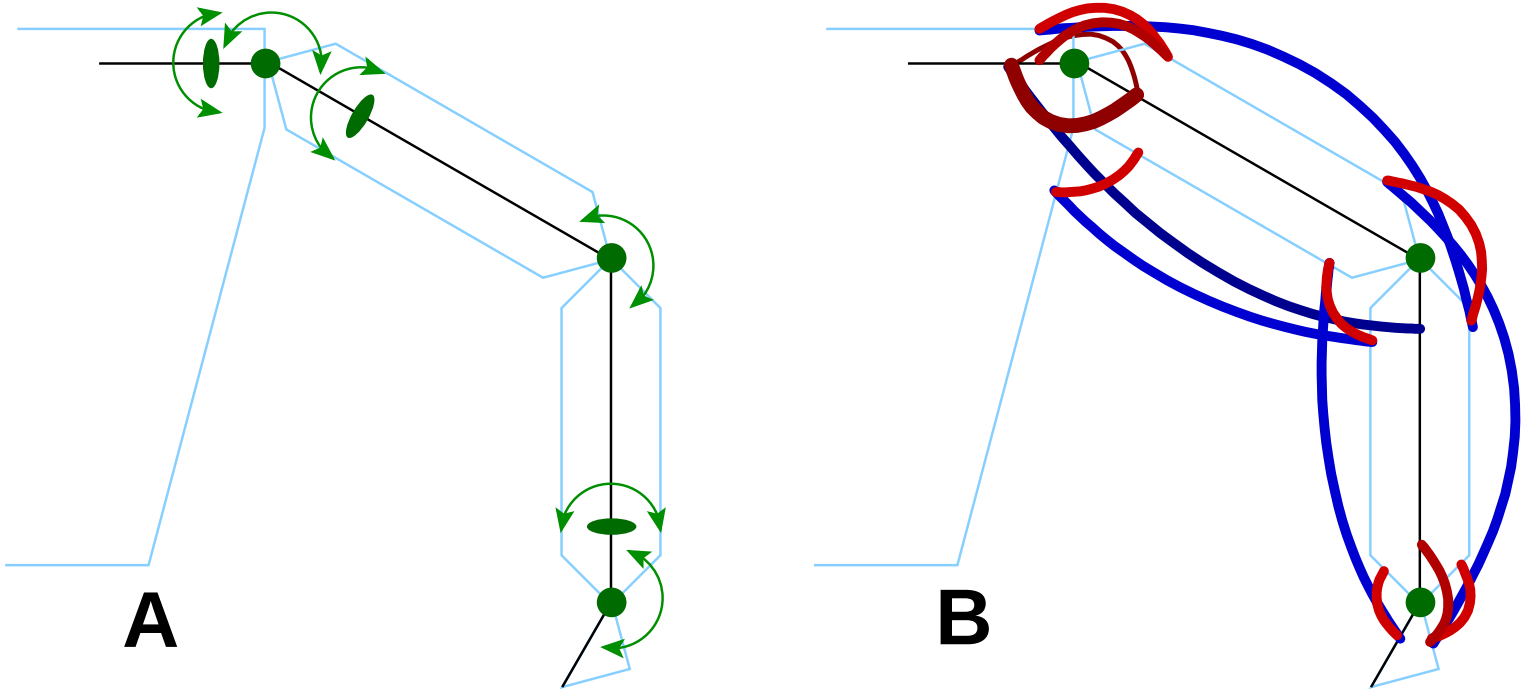
Cog emulates the human form at the right level of granularity.

⇓

These mechanisms should have a very natural form.

Cog's actuators should incorporate human-like control as well as kinematics.

The Plan



Take the individual single-axis series-elasic actuators (A), and control them as if they were actuated by antagonistic pairs of real muscles (B).

How? Calculate the torques which real muscles would exert, and use those values to command the motors. Call the whole thing “meso”.

Features of a Real Musculature

meso incorporates essential features for human-like behavior and response:

- reflex stiffness
- polyarticulate coupling
- a fatigue model

A higher-level control system can tune and optimize movement generation via biologically relevant feedback cues.

Spinal Reflexes

The brain’s motor cortex does not directly activate muscles:

- Cortex connects to spinal motor neurons which control the muscles in conjunction with spinal reflex loops.
- With input from stress and strain sensors, the reflexes make antagonistic pairs of muscles act like simple damped, linear springs.

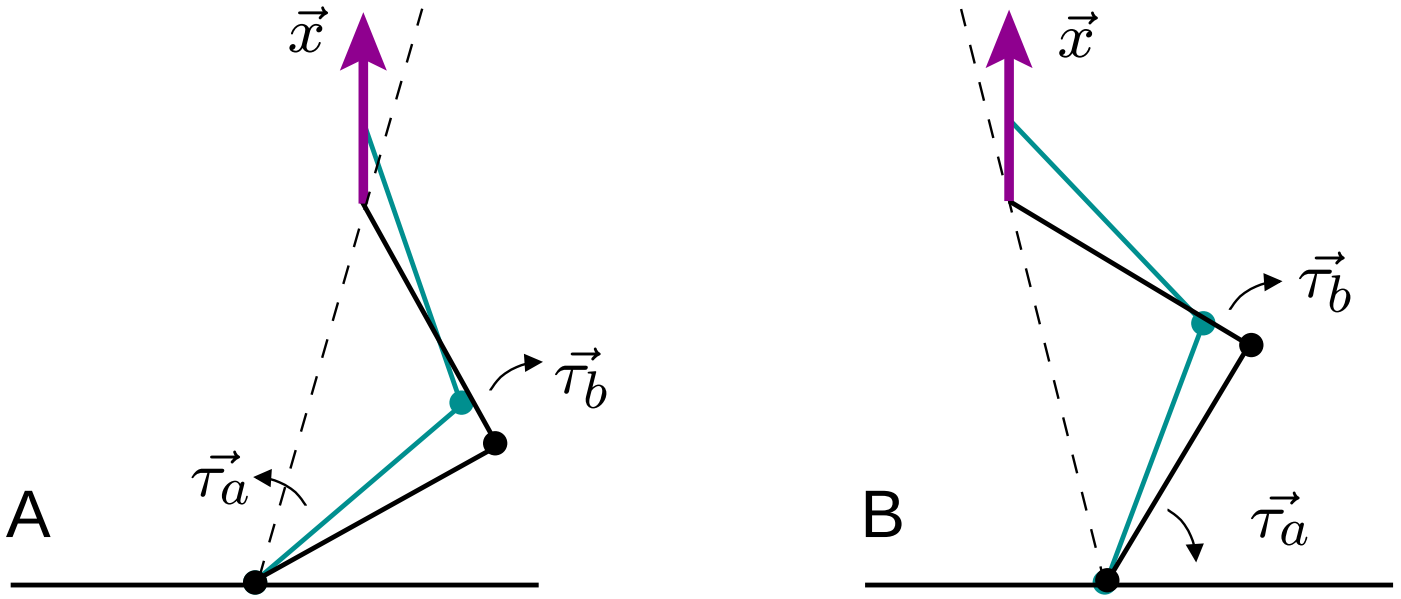
...An accurate simulation of muscle tissue itself is not necessary.

Polyarticulate Muscles

Many muscles in the human body span more than one joint, e.g. biceps and triceps.

- *Kinematically* redundant.
- Positive effects on limb *dynamics:*
  - Single-joint linear springs alone cannot produce isotropic stiffness in end-point coordinates.
  - Polyarticulate muscles can make the musculoskeletal system more efficient.

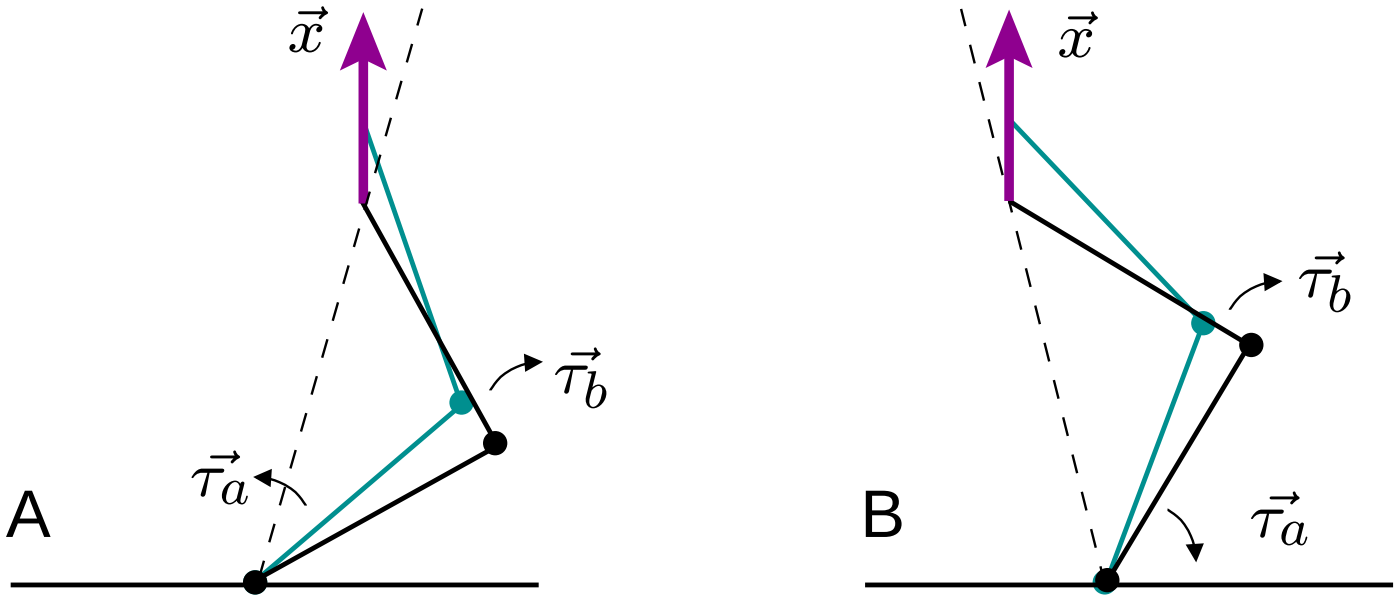
Mechanical Efficiency



To do work along $\vec{x}$ — that is, to apply a force along that vector — the arm must move into the blue configuration.

(A) The joints apply torques $\vec{\tau}_a$ and $\vec{\tau}_b$ in the same directions as they are displaced, thus both contributing to the output work.

Mechanical Efficiency



(B) Joint a applies torque $\vec{\tau}_a$ in *opposition* to its displacement.

Joint a is *absorbing* energy, which must have been provided by joint b.

Polyarticulate Muscles

In these cases, a stiffened biarticulate muscle can provide a mechanical linkage which produces the same motion, but does not waste the work.

Cog has no mechanical polyarticulate actuators:

- Simulation of such won’t make the robot more energy efficient.
- But, simulation should make human-like motion appear more energetically optimal to the controller.

Fatigue

Electric motors have completely alien fatigue characteristics compared to human muscle:

- Motors can apply a wide range of forces indefinitely.
- Human muscles get tired much more quickly, affecting *how* muscles end up being used.

meso's fatigue model which accounts for basic metabolic processes in muscle:

- reservoir of energy for penalty-free short-term activity
- build-up of slow-to-dissipate byproducts from unsustainably intense activity

"Fatigue level" affects motor performance implicitly, and is also accessible as feedback directly by the control system. The model is tunable, making it possible to simulate different stages of growth and ability.

Meat and Bones

meso is divided into two layers:

- a skeletal model which accounts for the kinematics of the robot and the virtual muscles.
- a muscular model which calculates the muscle forces and fatigue properties.

The Skeletal Model — Kinematics

Calculate two functions:

- $\vec{l}(\vec{\theta})$, the vector of lengths of the virtual muscles,
- $\vec{\tau}(\vec{\theta}, \vec{F})$, the vector of joint torques.

Inputs:

- $\vec{\theta}$, the skeletal configuration expressed as a vector of joint angles
- $\vec{F}$, the vector of muscle forces provided by the muscular model

Requires a description of the mechanical linkages in the real robot and a list of the anchor points of the virtual muscles.

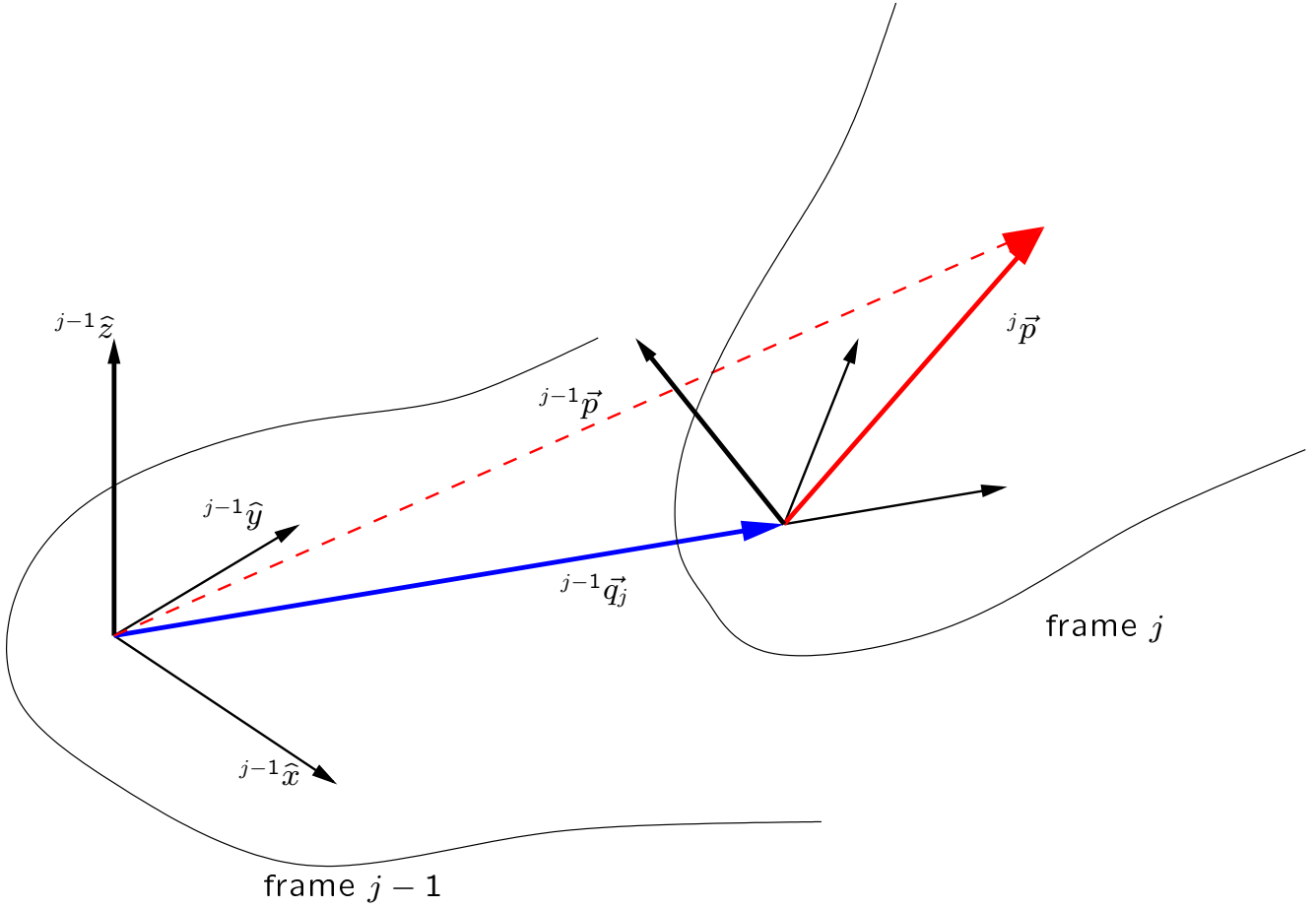
Coordinate Frames and Transforms

A robot can be described as a tree of segments, or *links*:

- Links are connected by revolute joints.
- Each link has an associated coordinate frame.
- A kinematic description of the robot is a specification of how these coordinate frames relate to each other.

My convention:

- Frames are aligned such that the $\hat{z}$ axis matches the joint axis
- Usually the $\hat{x}$ axis is parallel to the major axis of the associated limb segment.



- Frame $(j-1)$ is the parent of frame j.
- Vectors ${}^{j}\vec{p}$ and ${}^{j-1}\vec{p}$ describe the same point, relative to the respective frames,
- ${}^{j-1}\vec{q}_j$ defines the origin of frame j. (And ${}^{j}\vec{q}_j = 0$; it's the origin, after all.)

Coordinate Frames and Transforms

The transform ${}^i_jT$ changes the reference frame of a point from j to i.

$$ {}^i\vec{p} = {}^i_jT {}^j\vec{p} $$

For any two frames $i < j$ connected by a kinematic chain,

$$ {}^i_jT = {}^i_{i+1}T {}^{i+1}_{i+2}T \ldots {}^{j-1}_jT. $$

${}^i_jT$ is actually a rotation and a translation,

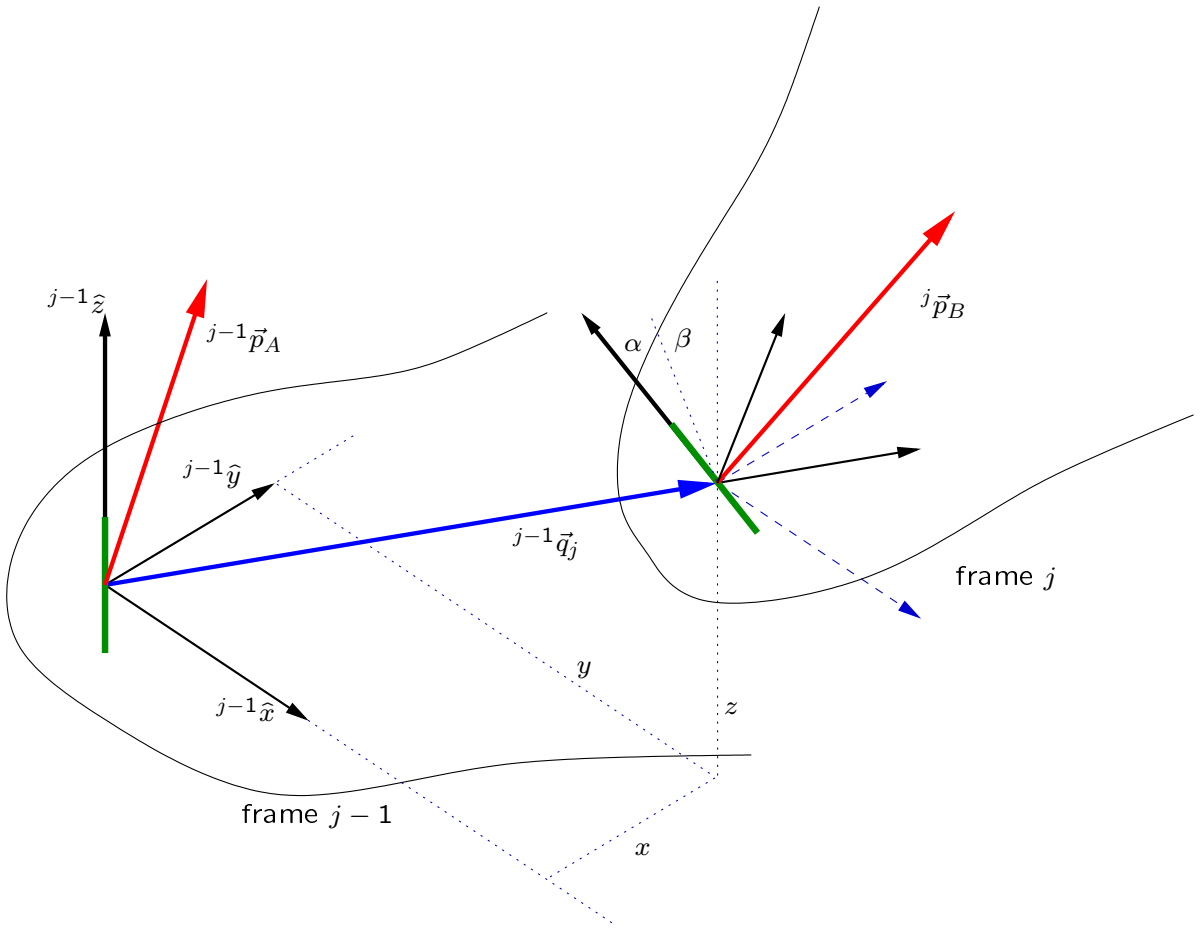
$$ {}^i_jT {}^j\vec{p} = {}^i_jR {}^j\vec{p} + {}^i\vec{q}_j, $$

where ${}^i_jR$ is a 3×3 matrix, which depends on the parameterization used to specify the relative orientation.

Linkage and Muscle Description

Frames are described with six parameters:

- x, y, z (constant) to position a joint in its parent frame
- α, β (constant) to set the joint's orientation
- θ (variable) to indicate the joint angle
- (implicit seventh parameter is the identity of the parent frame)



Origin specified by relative (x, y, z) position; β is rotation around $\hat{y}$; α is around $\hat{x}'$.

Parameterization is mostly a matter of convenience: all it needs to do is to provide the transformation matrices ${}_{i-1}^{i}T$.

Linkage and Muscle Description

Virtual muscles are just lines defined by two endpoints, ${}^{j}\vec{p}_A$ and ${}^{k}\vec{p}_B$.

- Each endpoint (A or B) is anchored in a particular link's coordinate frame.
- Frames correspond to the parts of the robot to which the muscle is attached.

Calculation of Muscle Length

l_{AB} is the cartesian distance between the anchor points, ${}^{j}\vec{p}_A$ and ${}^{k}\vec{p}_B$.

$$l_{AB} = \|\vec{p}_A - \vec{p}_B\|$$

Must first transform the endpoint vectors into the same reference frame:

$${}^{k}\vec{p}_A = {}^{k}_{j}T\,{}^{j}\vec{p}_A$$

Anchor vectors are constant w.r.t. base frame, but transformation is a function of the skeleton/joint configuration. Hence, the length becomes a function of the joint angles $\vec{\theta}$.

Calculation of Joint Torque

Must calculate the torque at each joint spanned by a virtual muscle.

$$\vec{\tau} = \vec{F} \times \vec{r}$$

Given the force magnitude F:

$$\begin{aligned} \vec{F} &= F\frac{\vec{p}_B - \vec{p}_A}{\|\vec{p}_b - \vec{p}_A\|} = \frac{F}{l_{AB}}(\vec{p}_B - \vec{p}_A) \\ \vec{r} &= \vec{p}_B - \vec{q}_j \\ \vec{\tau}_j &= \vec{F} \times \vec{r} = \frac{F}{l_{AB}}(\vec{p}_B - \vec{p}_A) \times (\vec{p}_B - \vec{q}_j) \end{aligned}$$

Calculation of Joint Torque

For joint j, choose the j^{th} reference frame. Note that ${}^j\vec{q}_j = 0$:

$$\begin{aligned} {}^j\vec{\tau}_j &= {}^j\vec{F} \times {}^j\vec{r} \\ &= \frac{F}{l_{AB}}({}^j\vec{p}_A \times {}^j\vec{p}_B) \end{aligned}$$

Only use $\hat{z}$ component:

$$\tau_{jz} = \left(\frac{F}{l_{AB}}\right)({}^jp_{Ax}{}^jp_{By} - {}^jp_{Ay}{}^jp_{Bx})$$

This calculation is performed for every joint j $(i < j \leq k)$ spanned by the virtual muscle.

Each virtual muscle m contributes a vector of torques $\vec{\tau}_m$ to the robot (one component per joint; most are zero) to yield $\vec{\tau}(\vec{\theta}, \vec{F})$, the skeletal torque of the virtual musculature.

The Muscular Model

Muscular model computes the force output of virtual muscles by simulating muscle tissue:

- spring-like function of the muscle length,
- modulated by a fatigue model

Real muscles exhibit spring-like behavior due to:

- mechanical properties of muscle tissue
- effects of the lowest level spinal feedback loops

Spring-like Muscles

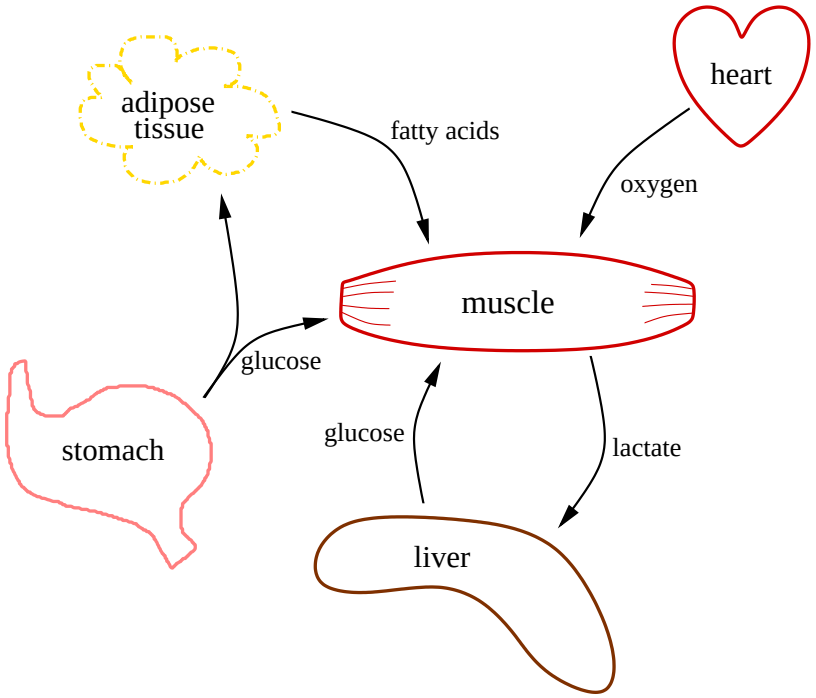
Basic control law for a virtual muscle:

$$F = K(l - l_0) - Bl' + F_0.$$

- stiffness K
- damping B
- equilibrium length l_0, “actual” length l
- bias force F_0

Force law output can be clipped such that $F \geq 0$.

Fatigue Model



A complete metabolic model simulates major organs involved with energy production and consumption. Long-term exertion causes build-up of lactic acid in muscle tissue, leading to fatigue. Heart rate and fuel availability affect the rate of lactic acid accumulation.

Fatigue Model

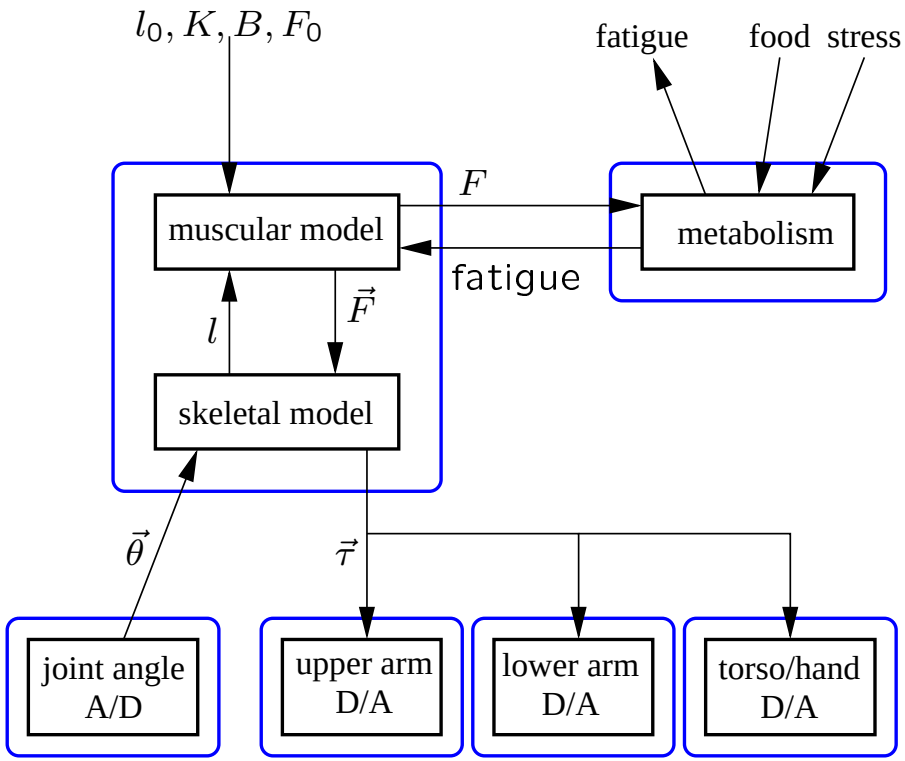
- Lactic acid level in each muscle used to modulate force output.
- Level is exposed to the rest of the system, to be used as an indicator of fatigue or soreness.
- Hook for modifying the blood glucose level (simulating ingestion of food).
- Hook for injecting epinephrine into the system, which could be linked to an emotional system to provide a physically realized reaction to stress.

Implementation

“It’s a bunch of sok processes.”

Control Loop

Seven processes distributed over six processors:



Main loop runs at 500Hz, with less than $4ms$ of end-to-end latency.

Skeletal Model Compiler: mesoc

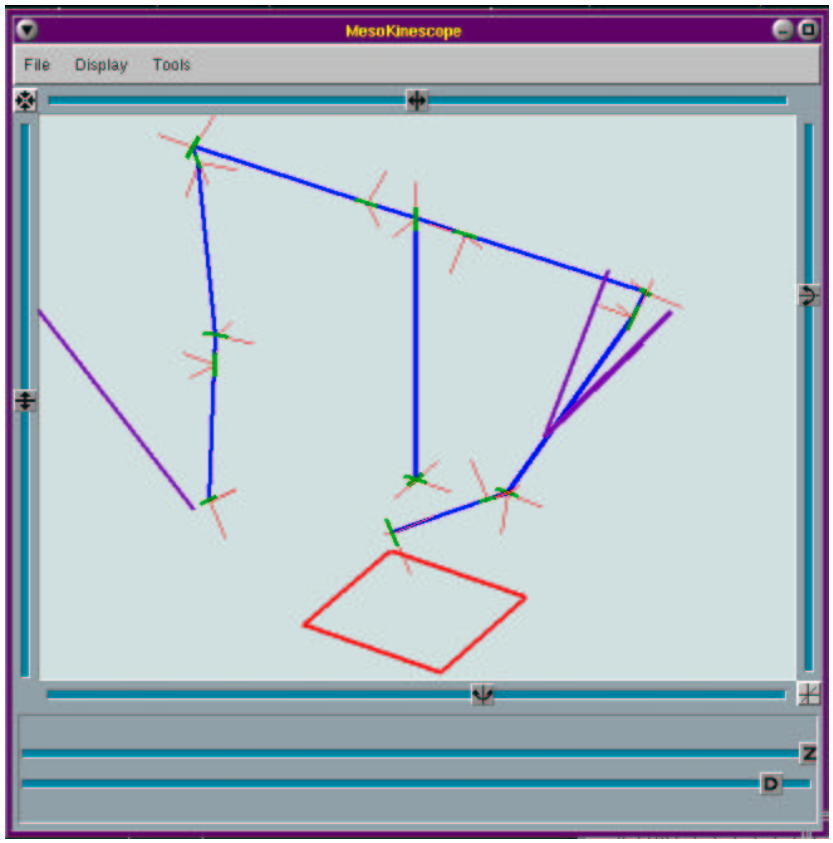
- Parses a configuration file specifying the skeleton and muscles.
- Produces C code which defines the two functions that calculate $\vec{l}(\vec{\theta})$ and $\vec{\tau}(\vec{\theta}, \vec{F})$.

Fixed skeletal structure $\Longrightarrow$ Optimizations:

- All transformations ${}_{j+1}^{j}T$ are precomputed up to the value of $\vec{\theta}$.
- Caching of intermediate results; $2N + 1$ transformations are calculated for a muscle which spans N joints.

The resulting code is very fast: on a 200MHz x86 processor, length and torque can be calculated for over 40 three-joint muscles at 1000Hz.

Graphical Interface: vmeso



- Create and edit skeletal models.
- Visualize muscle operation in real-time.

Preliminary Results

- Testing with several different combinations and configurations of virtual muscles.
- More or less stable, but difficult to quantify the performance of the system.
- With interjoint coupling, the robot's motion is quite different from simple single joint control:
  - Disturbing the arm when it is outstretched causes it to wiggle in unexpected ways.
  - Manually twisting one joint will cause another to move.

Limitations

Muscles are modelled as straight-line connections between points:

- No notion of a tendon which can wrap around a joint.
- Anchor points must be far away from the skeleton, to keep the muscles from crossing through joint.

Actual series-elastic actuators have a torque-control bandwidth of $\sim 25Hz$.

- Damping coefficients B in the muscle force law are necessary to keep the system stable.
- Unclear how these parameters must be adjusted, given that the whole system is a very non-linear controller.

Conclusions and Ongoing Work

No conclusions yet (except that Cog is a pain in my ass). Next:

- Add motor learning modules which will control the muscles.
- Learn how to adjust the bias force F_0 in response to pose (maintain posture while keeping muscle stiffness low).
- Postural mechanism which learns to keep the torso upright by optimizing for minimum muscle fatigue.

Eventually:

- Learn how to move by discovering how muscle activity correlates with sensory feedback
- Experiment with a simple emotional system that connects to the metabolic system and can produce startle and fight-or-flight responses.

Acknowledgements

- Originally inspired by spirited conversations with Matt Williamson.
- Bryan Adams contributed further conversations, much-appreciated toil on the robot, and the metabolic model.
- Jerry Pratt provided several key mechanical insights.
- Money came from DARPA as part of the "Natural Tasking of Robots Based on Human Interaction Cues" project under contract number DABT 63-00-C-10102.