Friedrich-Alexander-Universität
Technische Fakultät

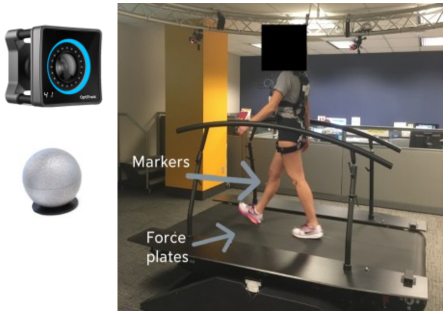
“In the wild” movement analysis using dynamic simulations

Anne Koelewijn and Ton van den Bogert
With help from: Biomechanical Motion Analysis and Creation (BioMAC) group
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
Cleveland State University, Cleveland, USA

Movement Analysis

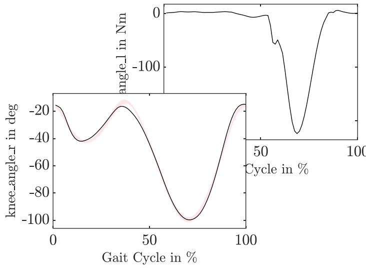
Standard Approach: Optical Motion Capture and inverse methods

Motion Laboratory

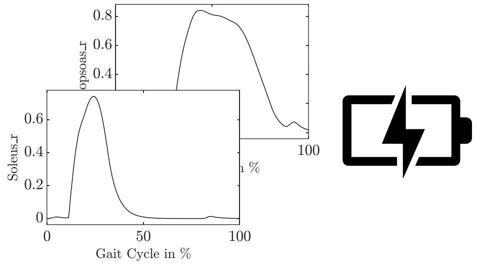


Musculoskeletal Model

Joint Angles and Moments

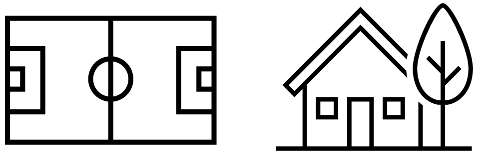


Muscle States and Energy Expenditure

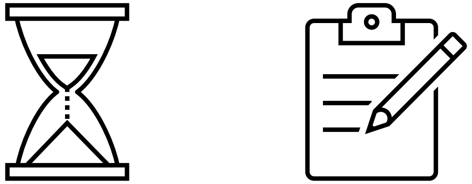


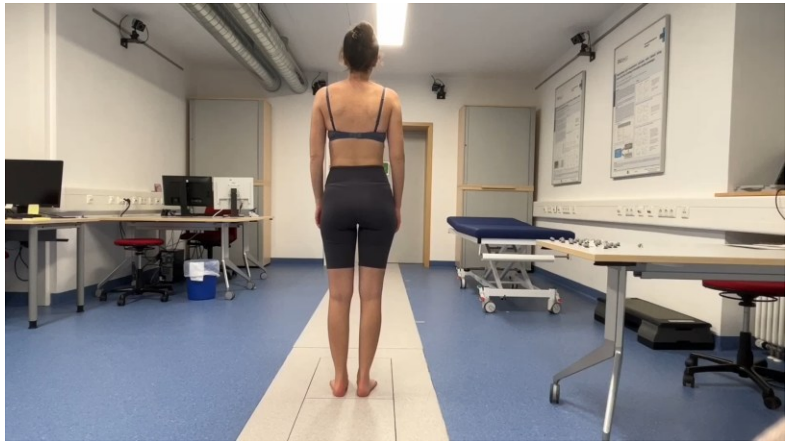
Challenges

- Restricted environment



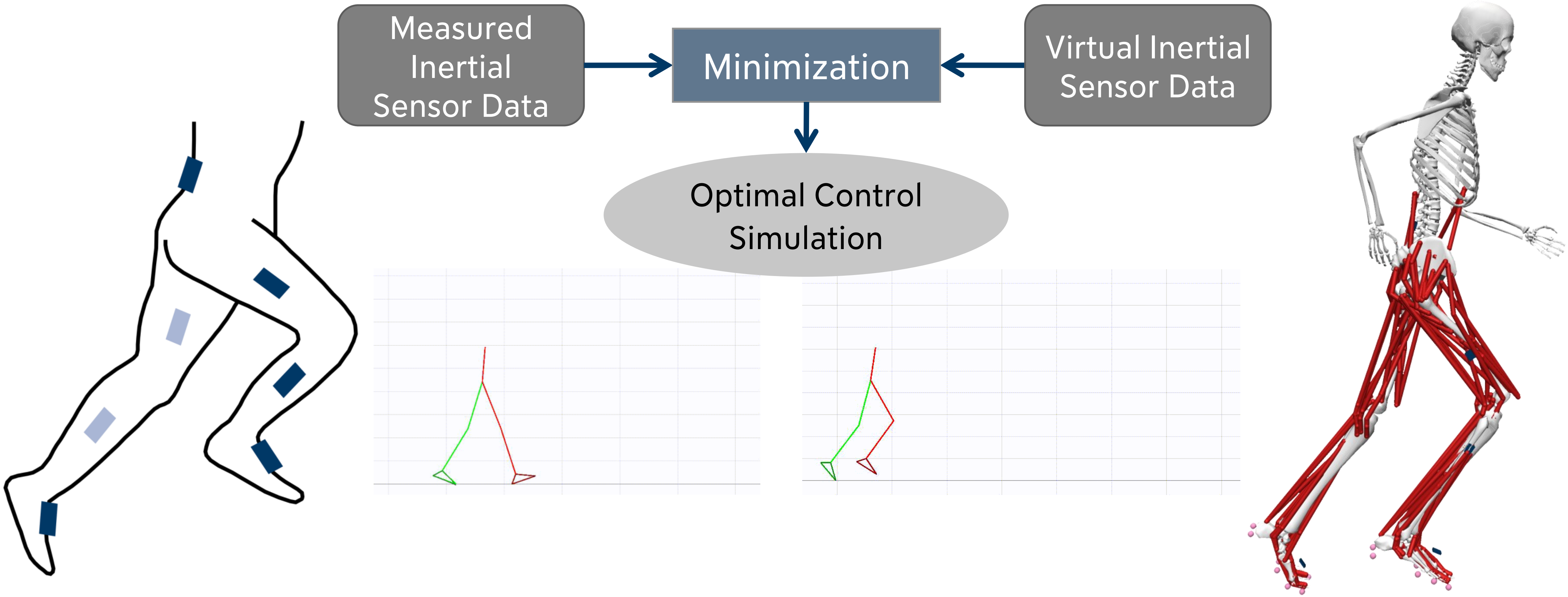
- Time consuming





Optimal Control Simulations

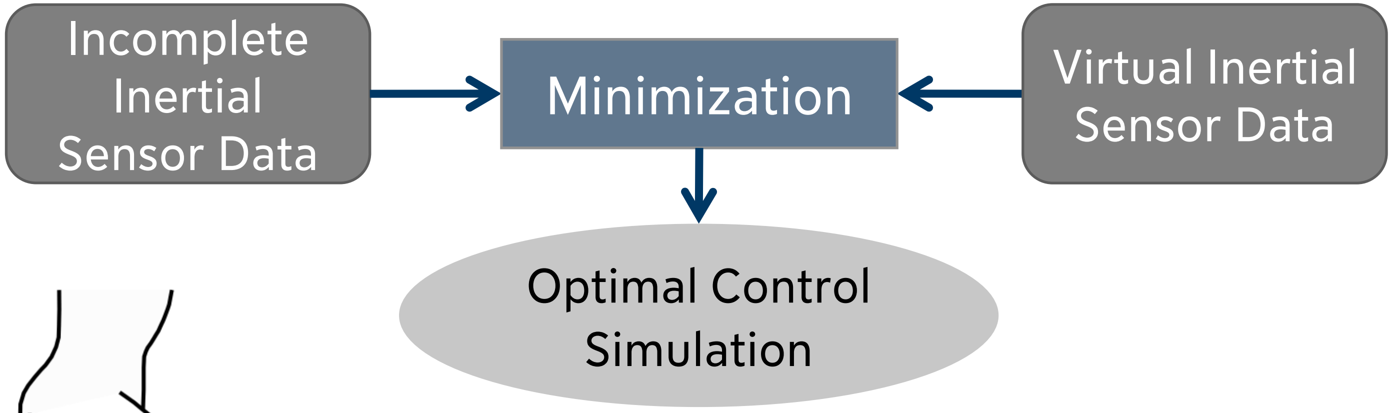
Movement Reconstruction Concept

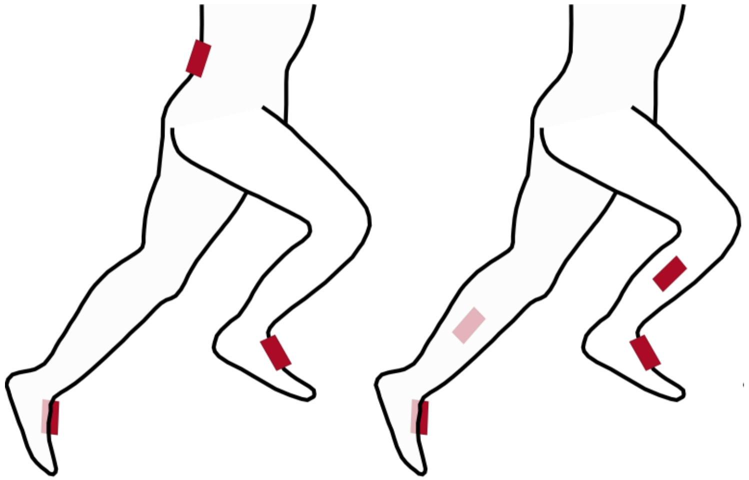


Dorschky et al., *J Biomech*, 2019

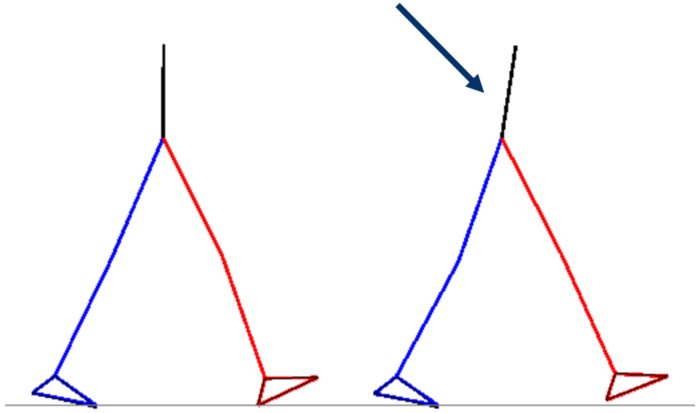
Optimal Control Simulations

Incomplete Data

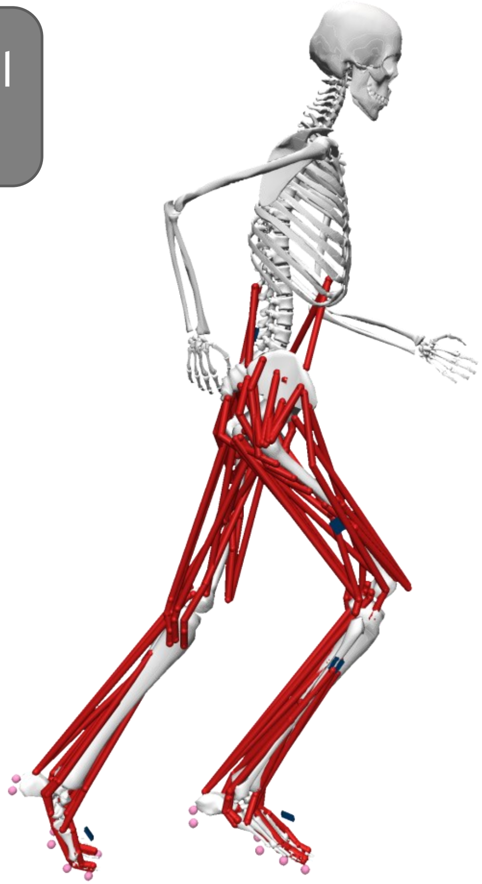




3 sensors 4 sensors



3 sensors 4 sensors



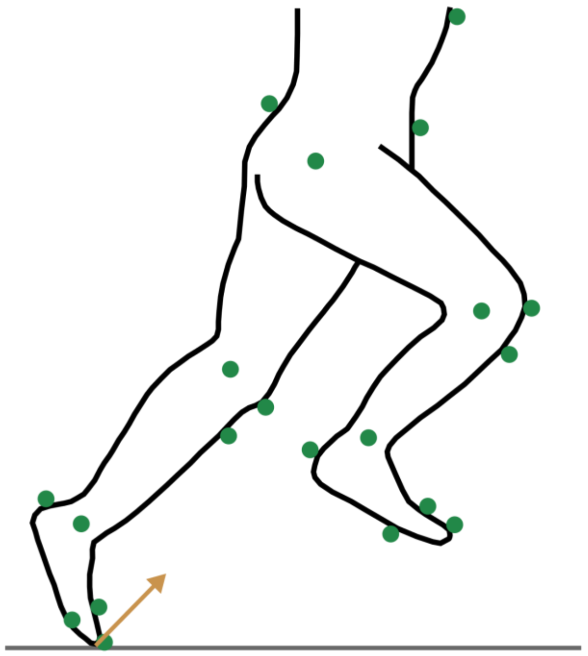
Dorschky, Nitschke et al., *Front Bioeng Biotech*, 2025

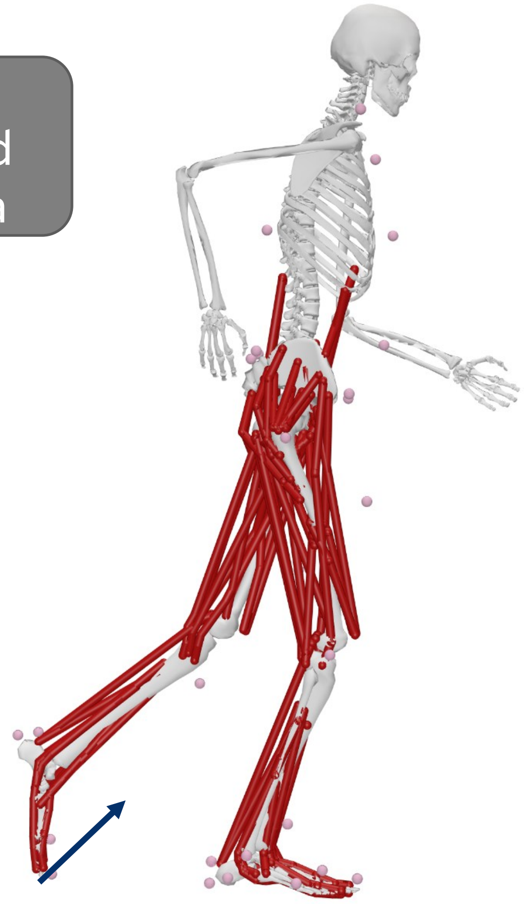
Optimal Control Simulations

3D Reconstruction from Marker and Force Data

Measured Marker and Force Data → Minimization ← Virtual Marker and Force Data

Minimization → Optimal Control Simulation





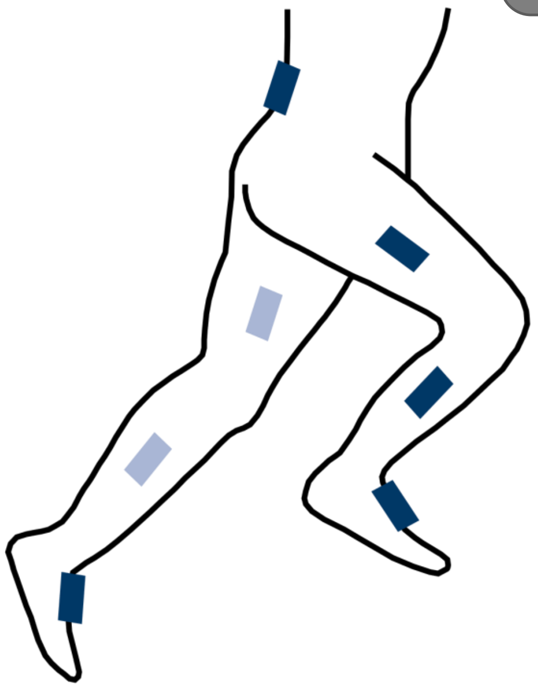
Nitschke, ..., Koelewijn, *PeerJ*, 2023

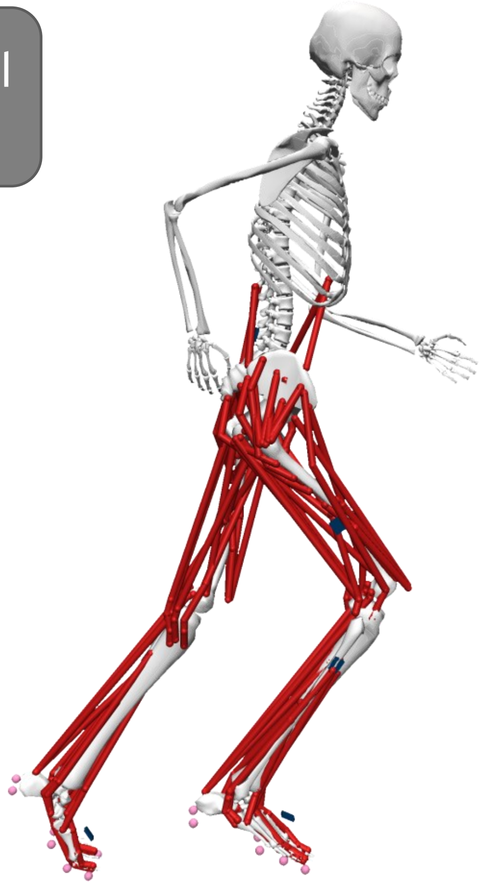
Optimal Control Simulations

3D Reconstruction from Inertial Sensor Data (Proof of Concept)

"Perfect" Inertial Sensor Data → Minimization ← Virtual Inertial Sensor Data

Minimization → Optimal Control Simulation

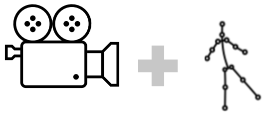




Nitschke et al., *Front Bioeng Biotech*, 2024

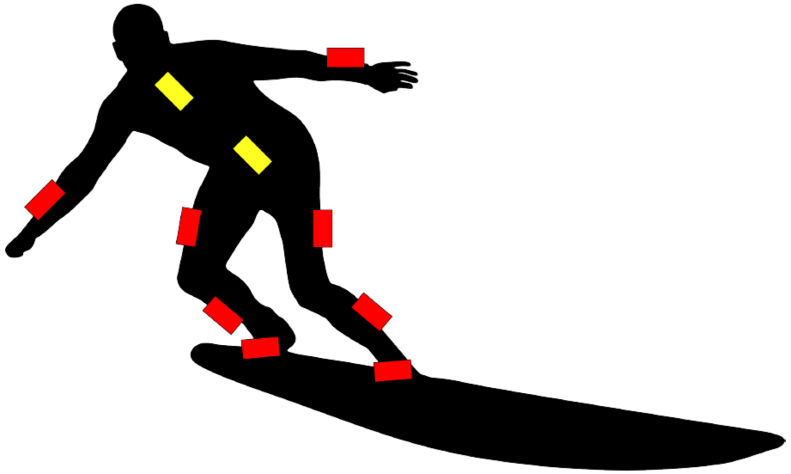
Biomechanical Analysis of Surfing

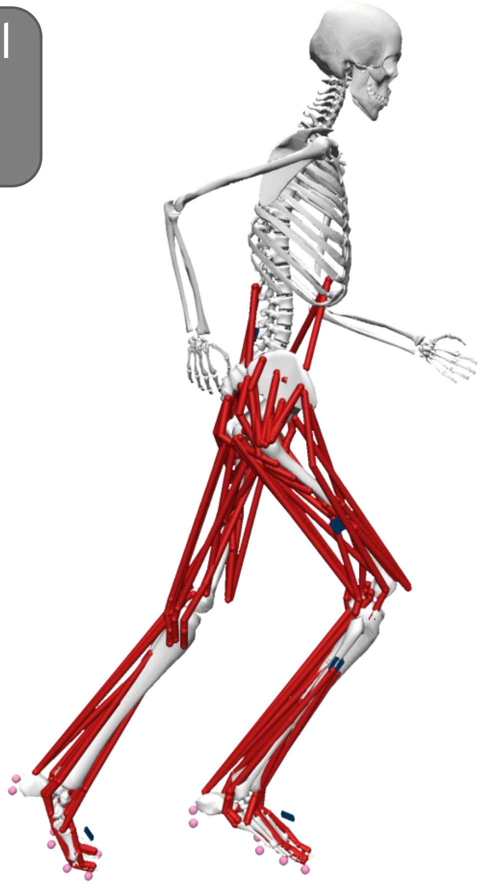
3D Reconstruction from Inertial Sensor and Video Data



Inertial Sensor and Pose Data → Minimization ← Virtual Inertial Sensor and Pose Data

Minimization → Optimal Control Simulation





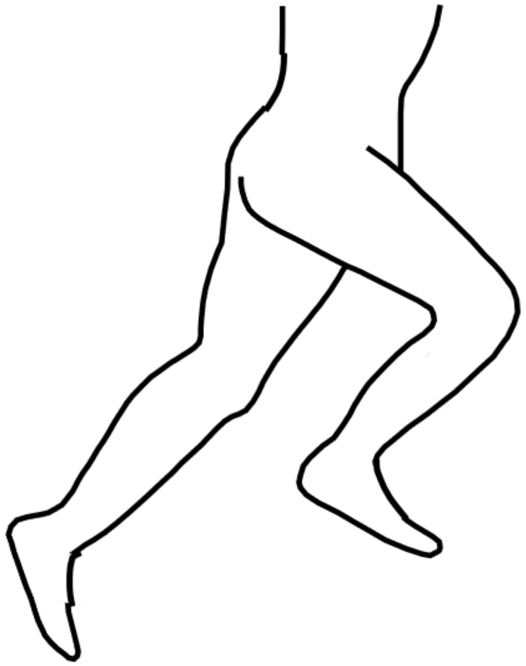
Weiss et al., *Mult Sys Dyn*, 2025

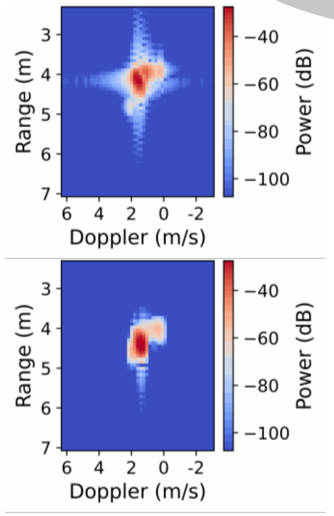
Optimal Control Simulations

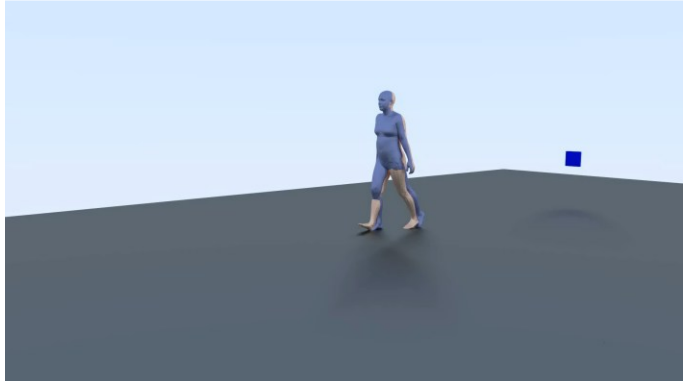
Movement Reconstructions from Radar Sensors

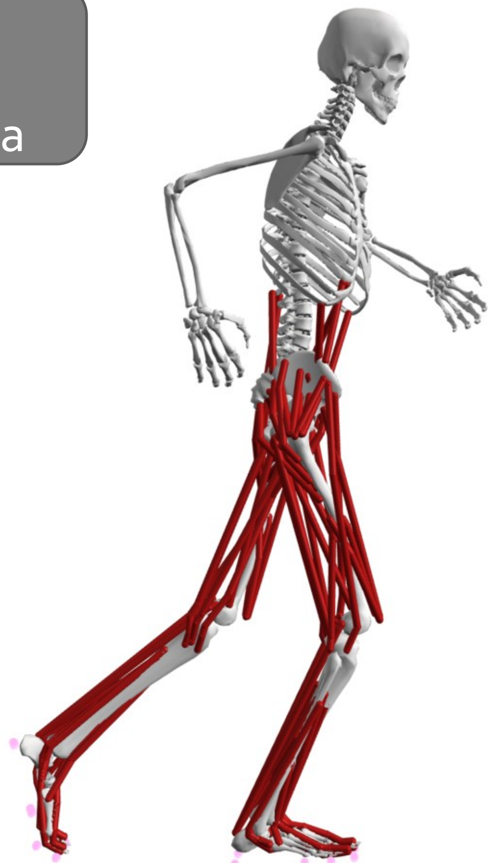
Doppler Radar Data → Minimization ← Virtual Doppler Radar Data

Minimization ↓ Optimal Control Simulation









Gambietz et al., *ASILOMAR*, 2024; Schüßler et al., *IEEE J Microw*, 2021

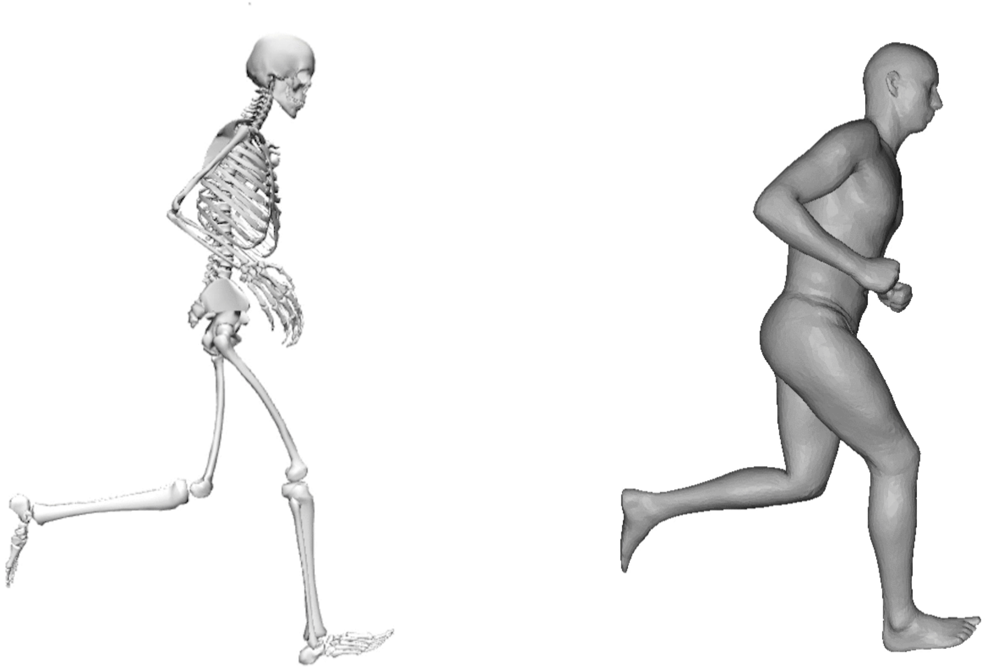
Agenda

1 Optimal Control Simulations
- Forward Dynamics Simulation
- Trajectory Optimization/Optimal Control
- Direct Collocation

2 Interial Sensor Model

3 Hands-On Tutorial
- Toolbox and Tutorial Introduction
- Familiarizing with the Output and Objects
- Comparing 2D Simulations from IMU Data
- Writing New Code

4 Conclusion



Forward Dynamics Simulation

Human Body Dynamics

Dynamics: description of an object's motion

- Accelerations are related to the applied forces

The motion of the body is defined by the skeleton

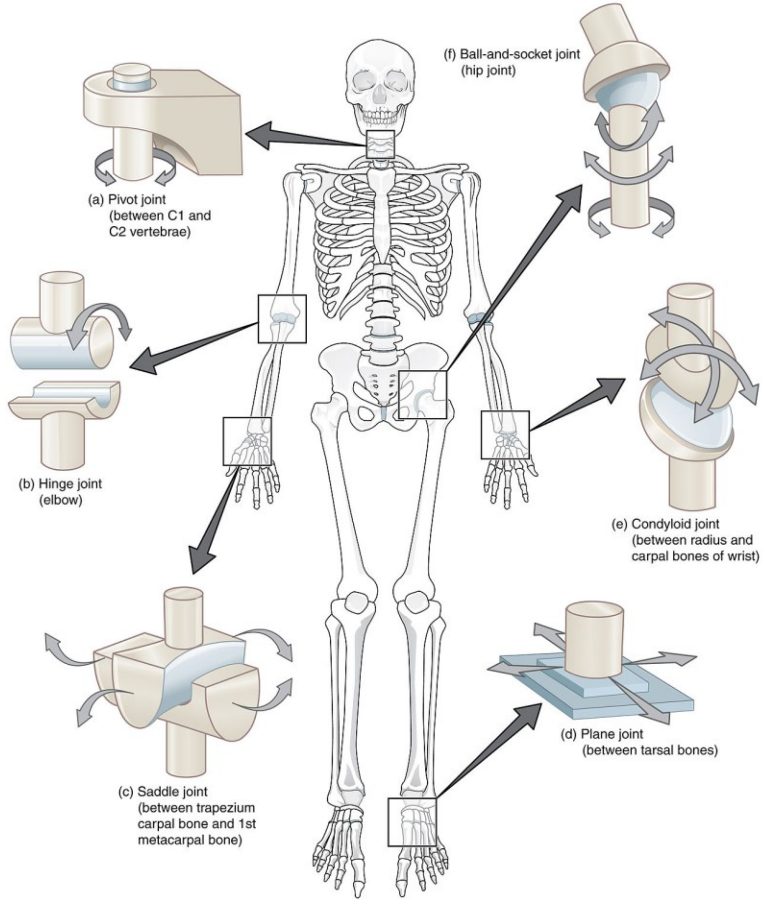
- Rigid bodies connected by joints
- Joint type defines degrees of freedom

Multibody dynamics

- Define the degrees of freedom
  - Global position and orientation of one reference segment (trunk)
  - Angles between segments

Propulsion: joint torques or muscle stimulations

- Muscle stimulation require muscle dynamics in addition



General Multibody Dynamics Equation

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T(q)F_{ext}$$

q: degrees of freedom

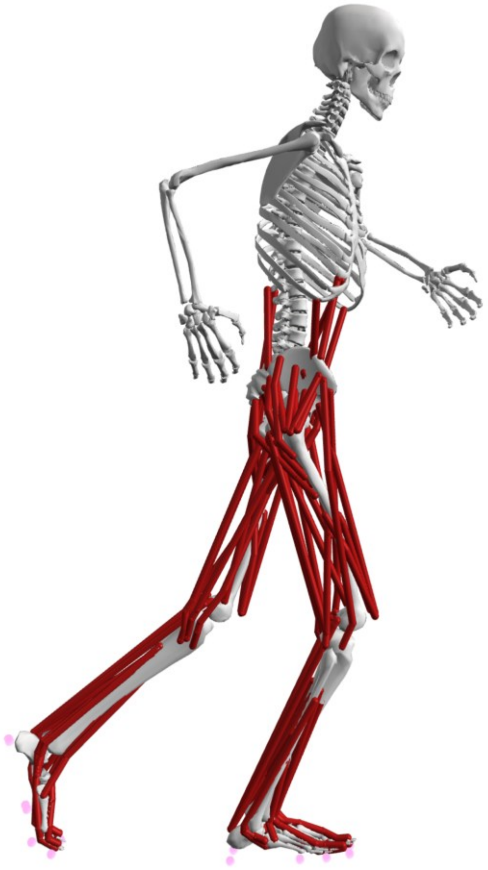
M: mass/inertia matrix, dependent on the joint angles

C: Centrifugal/coriolis matrix, dependent on joint angles and velocities

G: Gravity matrix, dependent on the joint angles

$J^T(q)F_{ext}$: External forces (ground reaction forces) using Jacobian

τ: internal moments / generalized forces / joint moments



General Multibody Dynamics Equation

$$M(q(t))\ddot{q}(t) + C(q(t),\dot{q}(t))\dot{q}(t) + G(q(t)) = \tau(t) + J^T(q(t))F_{ext}(t)$$

q: degrees of freedom

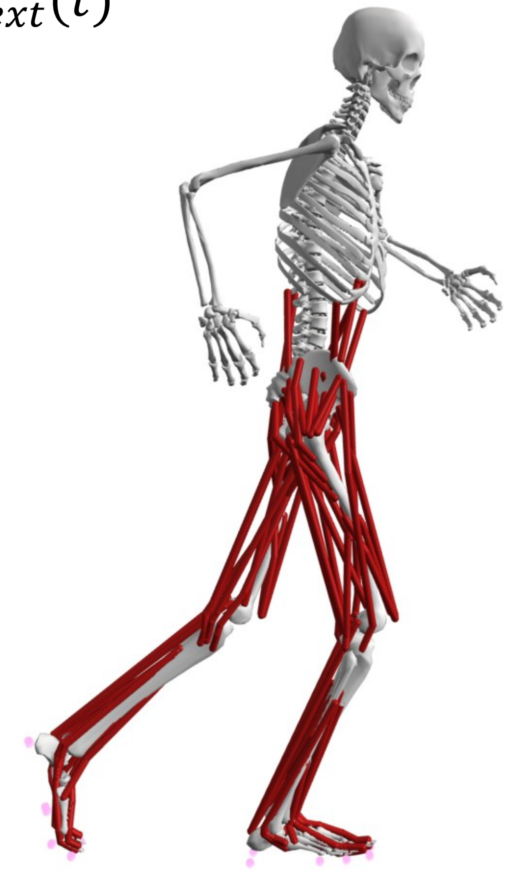
M: mass/inertia matrix, dependent on the joint angles

C: Centrifugal/coriolis matrix, dependent on joint angles and velocities

G: Gravity matrix, dependent on the joint angles

$J^T(q)F_{ext}$: External forces (ground reaction forces) using Jacobian

τ: internal moments / generalized forces / joint moments



Goal of a simulation: find $q(t)$ and $\tau(t)$

Second Order Dynamics System

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T(q)F_{ext}$$

Dynamics equation includes the second derivative of the joint angle, while numerical integration methods are first order.

- Use state $x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$ with joint angles and velocities (and global orientation and position and derivative)

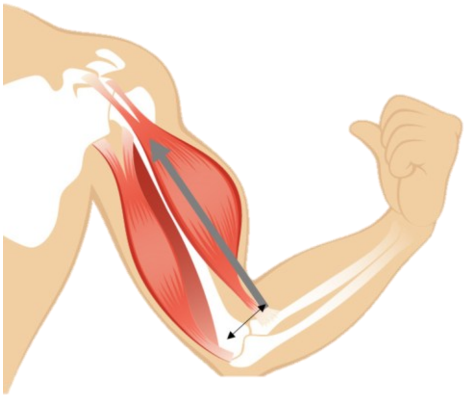
$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)(\tau + J^T(q)F_{ext} - C(q,\dot{q})\dot{q} - G(q) \end{bmatrix} = \begin{bmatrix} \dot{q}(t) \\ f(q(t),\dot{q}(t)) \end{bmatrix}$$

And the state at the next time point:

$$x(t+\Delta t) = \begin{bmatrix} q(t+\Delta t) \\ \dot{q}(t+\Delta t) \end{bmatrix} = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix} + \Delta t \begin{bmatrix} \dot{q}(t) \\ f(q(t),\dot{q}(t)) \end{bmatrix}$$

Rigid Body Dynamics with Muscles

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T(q)F_{ext}$$



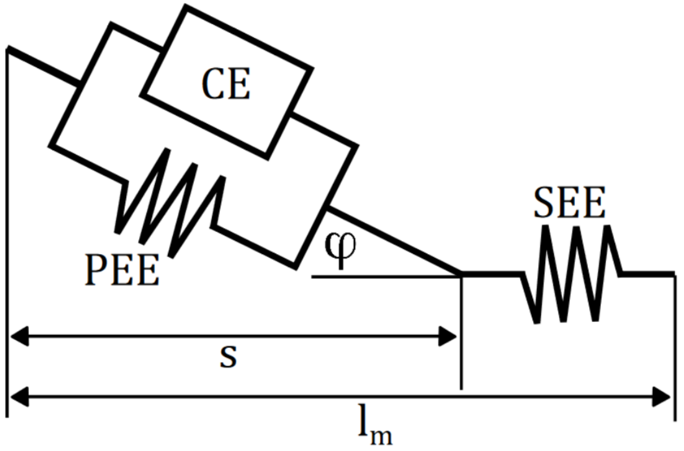
Inputs are τ: joint torques

- Joint torques can be input directly
- Determine joint torques from muscle forces $\tau = DF_{SEE}$
  - D: matrix of moment arms

Muscle dynamics

- $F_{SEE} = (F_{ce} + F_{PEE})\cos(\phi)$
- Two differential equations
  - $F_{CE} = af(l_{CE})g(v_{\mathrm{CE}})F_{iso}$
  - $\frac{da}{dt} = \left(\frac{u(t)}{T_{act}} + \frac{1-u(t)}{T_{deact}}\right)(u(t) - a(t))$

→Input is neural stimulation $u(t)$



Standard Dynamics Equation

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T(q)F_{ext}$$

Creating a simulation (for a human musculoskeletal system):

$$x(t+\Delta t) = \begin{bmatrix} q(t+\Delta t) \\ \dot{q}(t+\Delta t) \\ a(t+\Delta t) \\ l_{CE}(t+\Delta t) \end{bmatrix} = \begin{bmatrix} q(t) \\ \dot{q}(t) \\ a(t) \\ l_{CE}(t) \end{bmatrix} + \Delta t \begin{bmatrix} \dot{q}(t) \\ f(q,\dot{q},a,l_{CE}) \\ \dot{a}(t,u) \\ v_{CE}(t) \end{bmatrix}$$

$$f(q,\dot{q},a,l_{CE}) = M^{-1}(q)\left(\tau + J^T(q)F_{ext} - C(q,\dot{q})\dot{q} - G(q)\right)$$

- τ comes from the muscle forces
- The equation for $\dot{a}$ and v_{CE} are the Hill model

→ What is u?

- The muscle inputs for the model to perform the desired task

Questions?

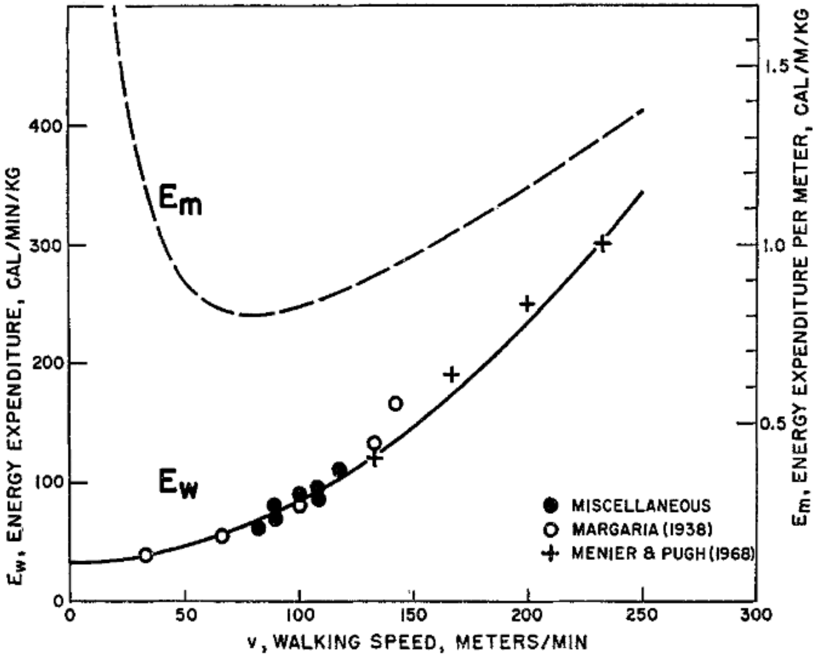
Optimal Control/Trajectory Optimization

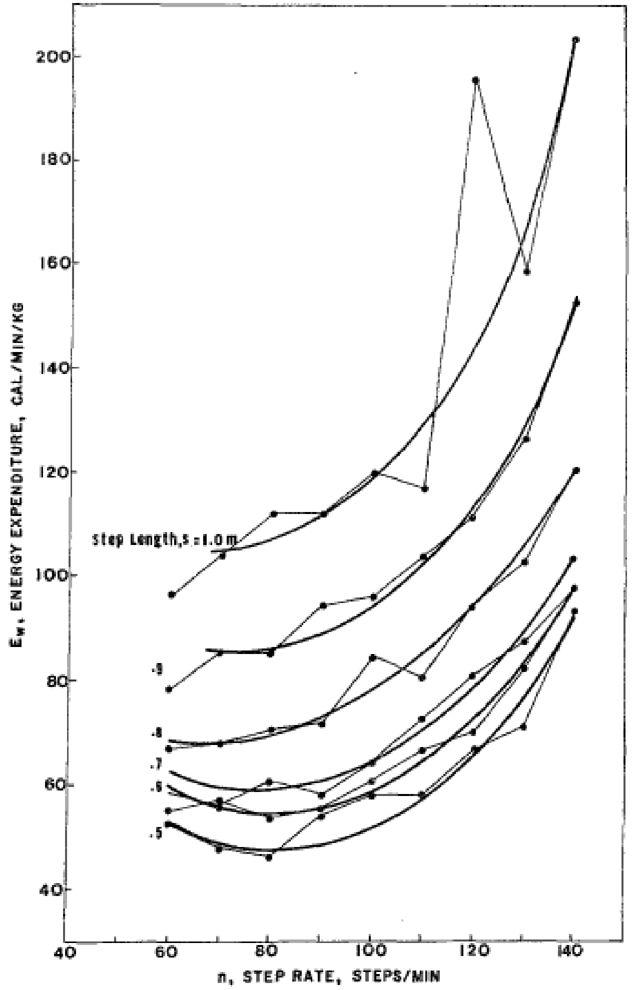
Why an Optimization?

People minimize energy expenditure when planning movements

- This is an optimization

→It can be converted to a computer optimization

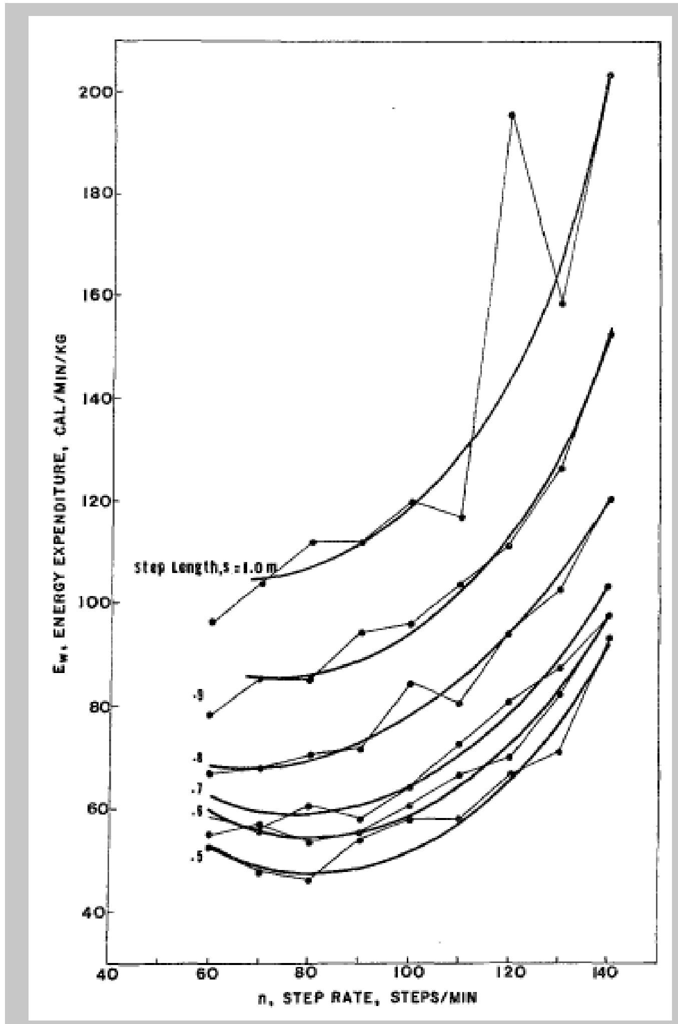




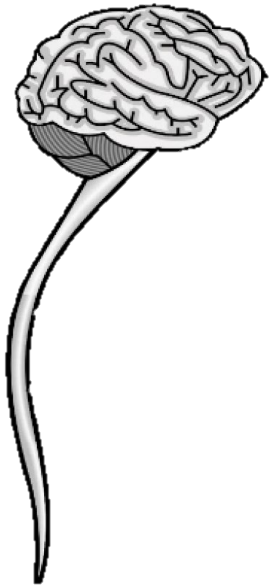
Zarrugh et al., *Europ. J. Appl. Physiol.*, 1974

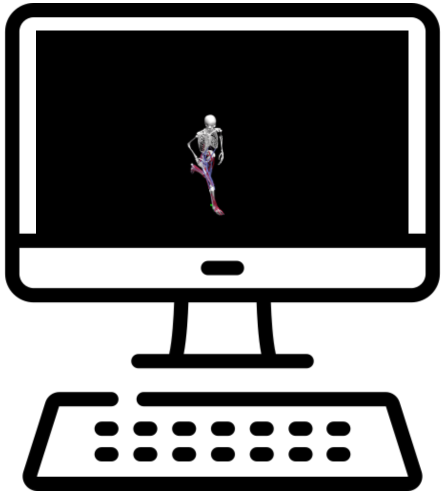
Creating Movement Simulations

Solve the same optimization on a computer as in the nervous system



Zarrugh et al., *Eur J Appl Physiol*, 1974



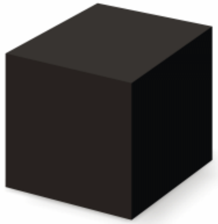


Two types of simulations

How is muscle stimulation created?

- Trajectory focused

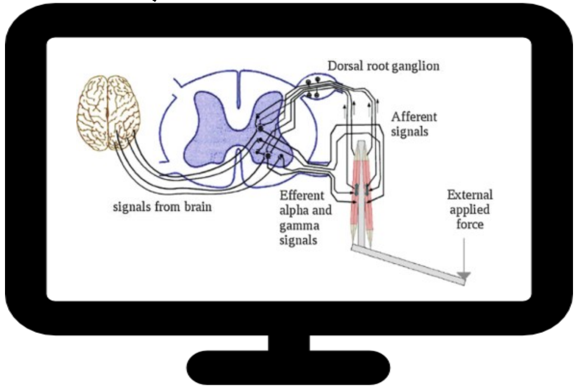
Be energy efficient
(optimize objective)



Muscle stimulations for
gait cycle

  - Outcome: time-series of muscle stimulations and movement

- Neural control (control focused)



  - Outcome: reflex or other control model parameters

Trajectory Optimization Problem

Term	Definition
Constraints	Requirements on variables that are required to be satisfied
Optimization variables	The variables that are changed in order to solve the optimization
Objective	The function that is minimized (or maximized)

Optimization variables: Find inputs $u(t), 0 \leq t \leq T$

Dynamics: For system $\dot{x}(t) = f(x(t), u(t))$

That minimize:

Objective:

$$J(x, u, t) = \frac{1}{T} \int_{t=0}^{T} c\big(x(t), u(t)\big) dt$$

Subject to:

Task constraints:

$$g\big(x(t_h)\big) = 0$$

Trajectory Optimization Problem

Term	Definition
Constraints	Requirements on variables that are required to be satisfied
Optimization variables	The variables that are changed in order to solve the optimization
Objective	

Optimiz

Dynamic

How do we solve this problem?

Objectiv

$)dt$

Subject to:

Task constraints: $g\big(x(t_h)\big) = 0$

Shooting

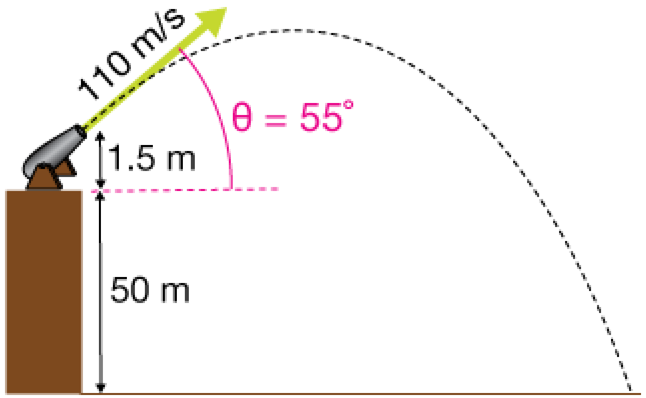
Compare to shooting a cannon

- Initial conditions $x(0)$ are known

Try control input $u(t)$ and simulate

- Numerical integration

$$x(\Delta t) = x(0) + \Delta t f\big(x(0), u(0)\big)$$
$$x(2\Delta t) = x(\Delta t) + \Delta t f\big(x(\Delta t), u(\Delta t)\big)$$
$$\vdots$$
$$x(t + \Delta t) = x(t) + \Delta t f\big(x(t), u(t)\big)$$



http://xaktly.com/ProjectileMotion.html

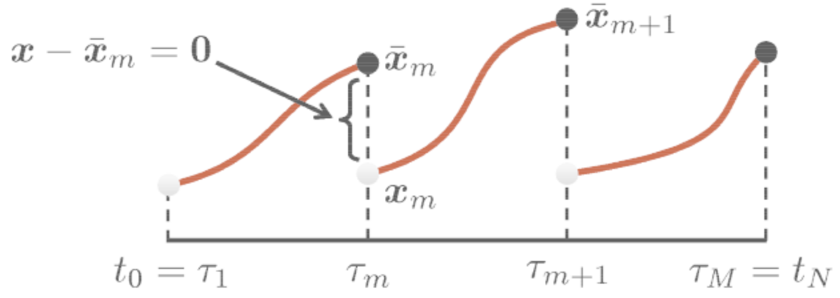
Iterate until objective is minimized

Multiple Shooting

Try states at several key times, $x(0)$, $x(t_1)$, $x(t_2)$ and control input $u(t)$

Iterate until

- Objective is minimized
- Constraints ensure system goes from $x(0)$ to $x(t_1)$ to $x(t_2)$



Optimization variables:

$$v_{opt} = \begin{bmatrix} u(0) \\ \vdots \\ u(T) \\ x(t_1) \\ x(t_2) \\ \vdots \\ x(t_f) \end{bmatrix}$$

Example in gait:

- Separate dynamics for swing, stance and double stance
- Allows for fixed base dynamics

Direct Collocation

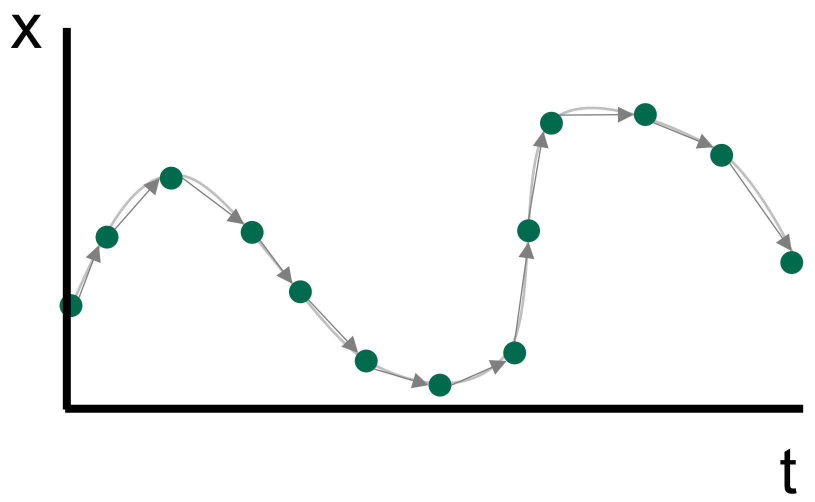
No forward simulation: dynamics added to the constraints

- Forward simulation: $x(t+1) = x(t) + \Delta t\, f(x(t), u(t))$
- Can be converted to a constraint $g\big(x(t_h)\big) = 0$

Try a set of states $x(t)$, and inputs

$$u(t), 0 \leq t \leq T$$

$$v_{opt} = \begin{bmatrix} u(0) \\ \vdots \\ u(T) \\ x(0) \\ x(t_1) \\ \vdots \\ x(T) \end{bmatrix}$$



Iterate until

- Objective is minimized
- States and inputs follow dynamics

Optimization problem with many optimization variables and constraints

Questions?

Direct Collocation

Direct Collocation

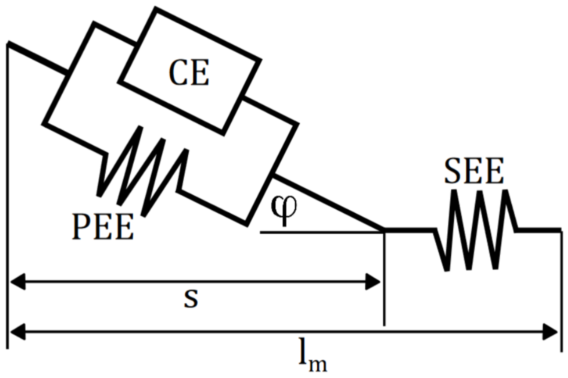
NO forward simulation → Adds dynamics to the constraints

Forward Simulation:

- Forward Euler iteration: $x(t + \Delta t) = x(t) + \Delta t \dot{x}(t)$, where $\dot{x} = f(x, u)$
- This can be a constraint: $g\big(x(t_k)\big) = 0$
  - Evaluated at each time (collocation) point
  - States are added to optimization variables

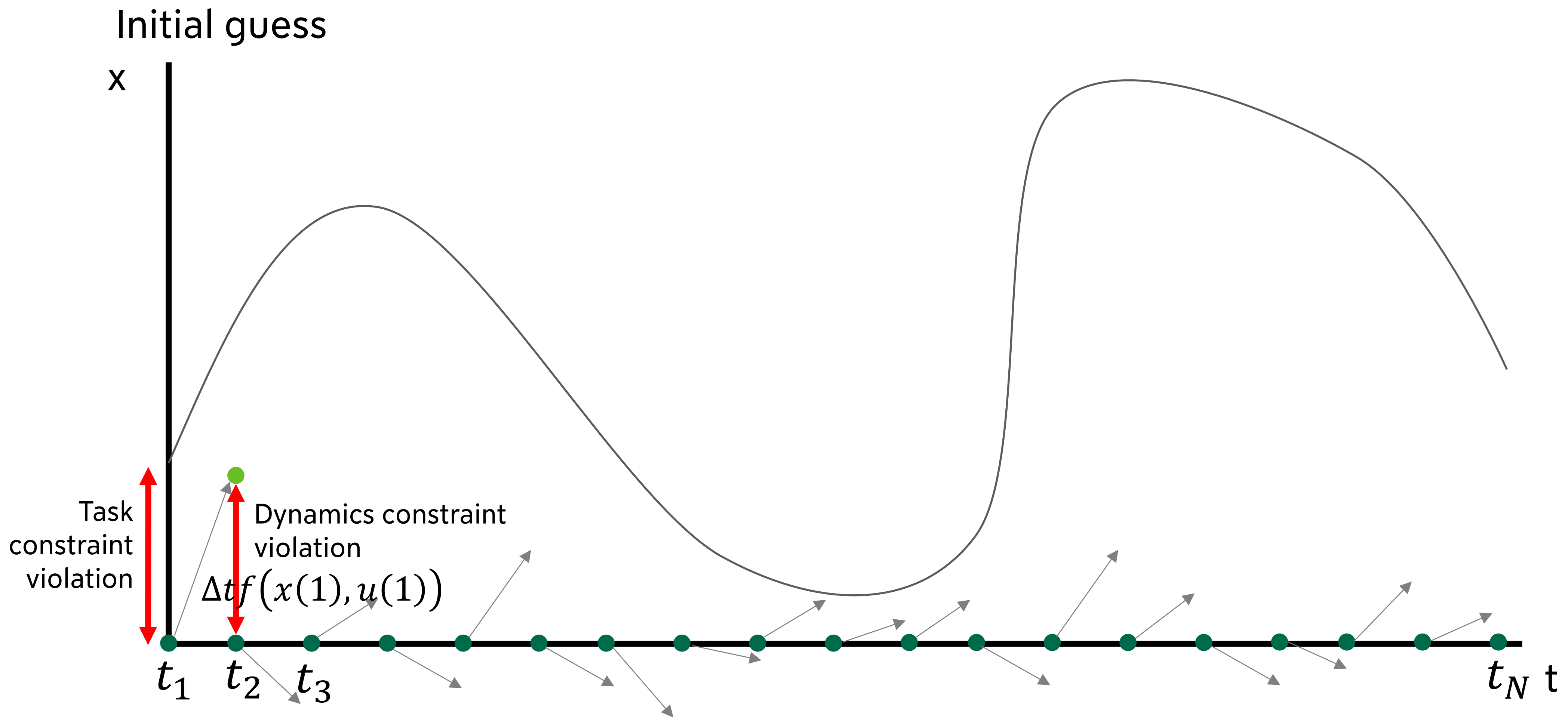
Dynamics in constraints:

- Implicit dynamics can be used $f(x(t), \dot{x}(t), u) = 0$
  - Suitable for muscle dynamics: no division by activation required
- Possible integration schemes
  - We normally use backward Euler: $f\big(x(t + \Delta t), \dot{x}(t), u(t + \Delta t)\big) = 0$
  - Higher order: e.g., Radau collocation schemes



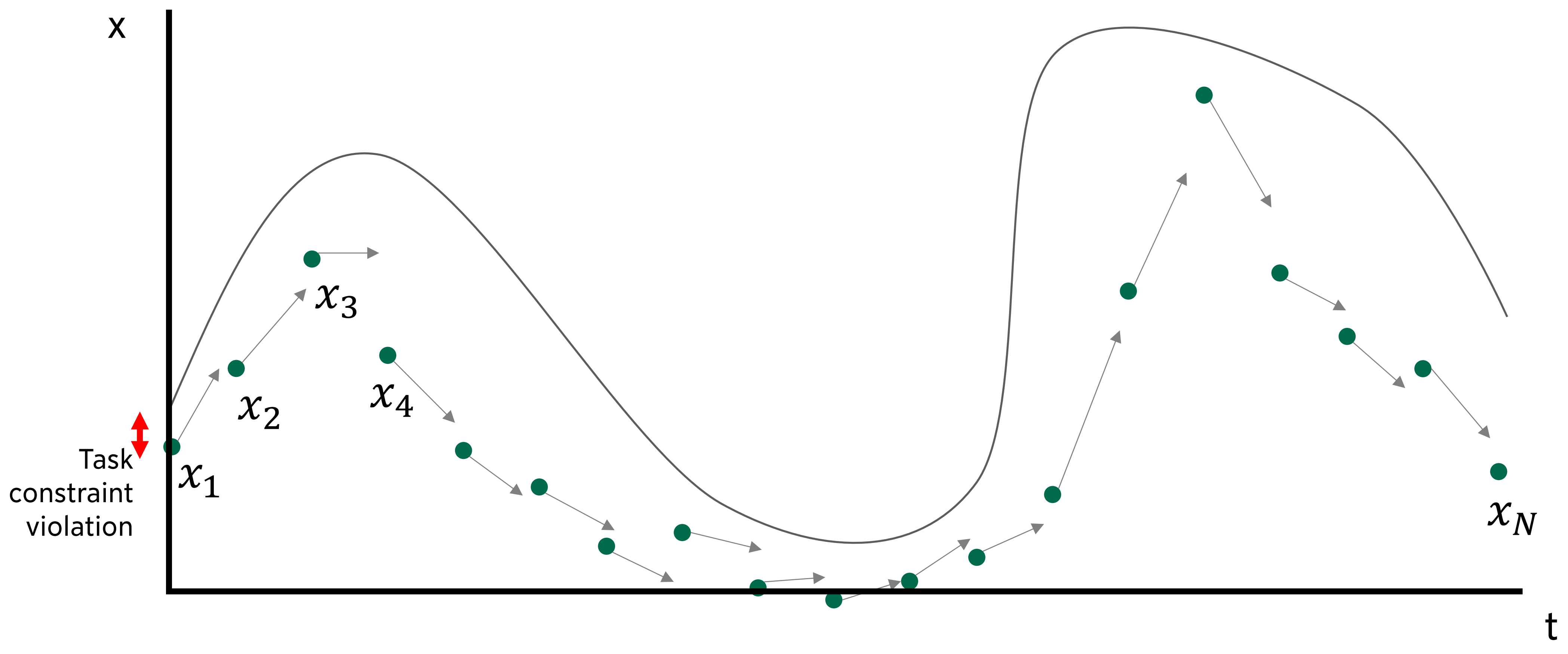
$$F_{CE} = af(l_{CE})g(v_{\mathrm{CE}})F_{iso}$$

Solution Method: Direct Collocation



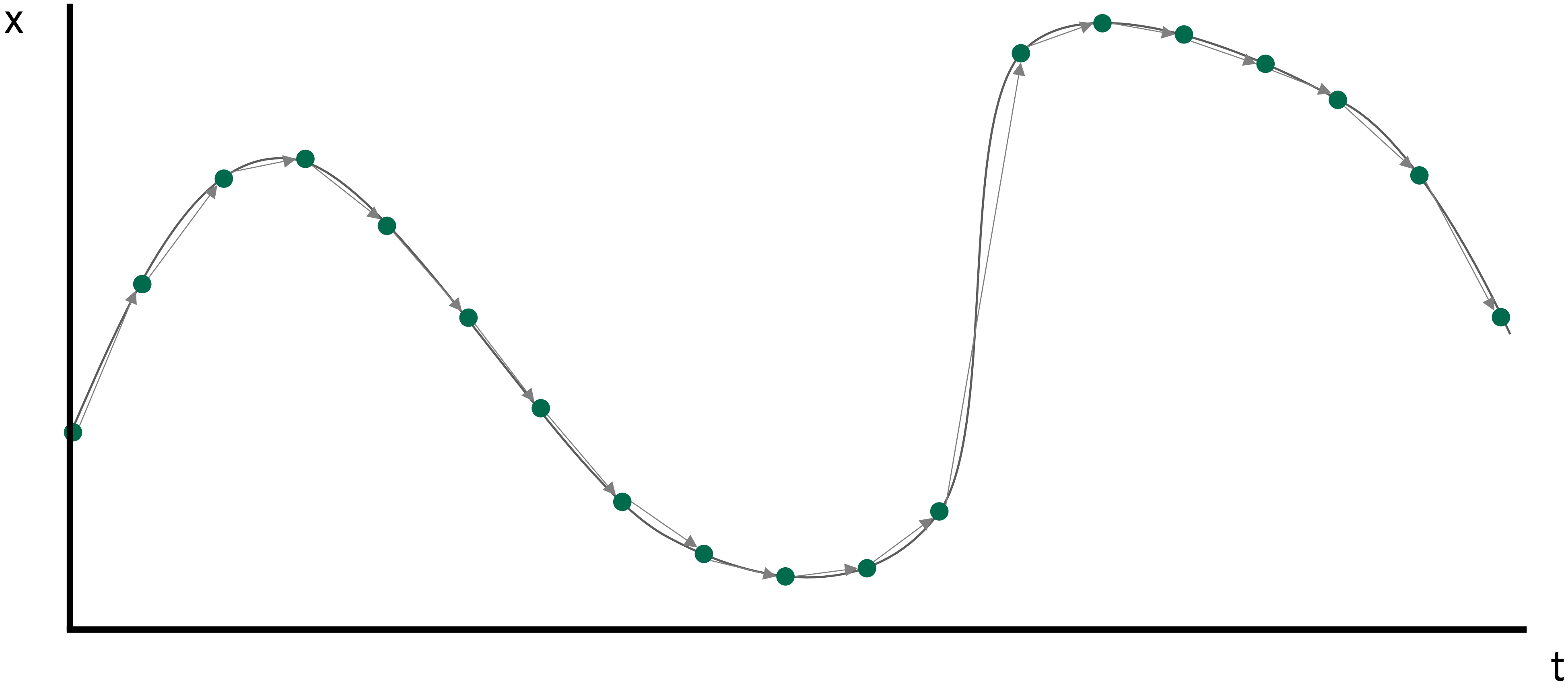
Solution Method: Direct Collocation

Some iterations later



Solution Method: Direct Collocation

Many more iterations later

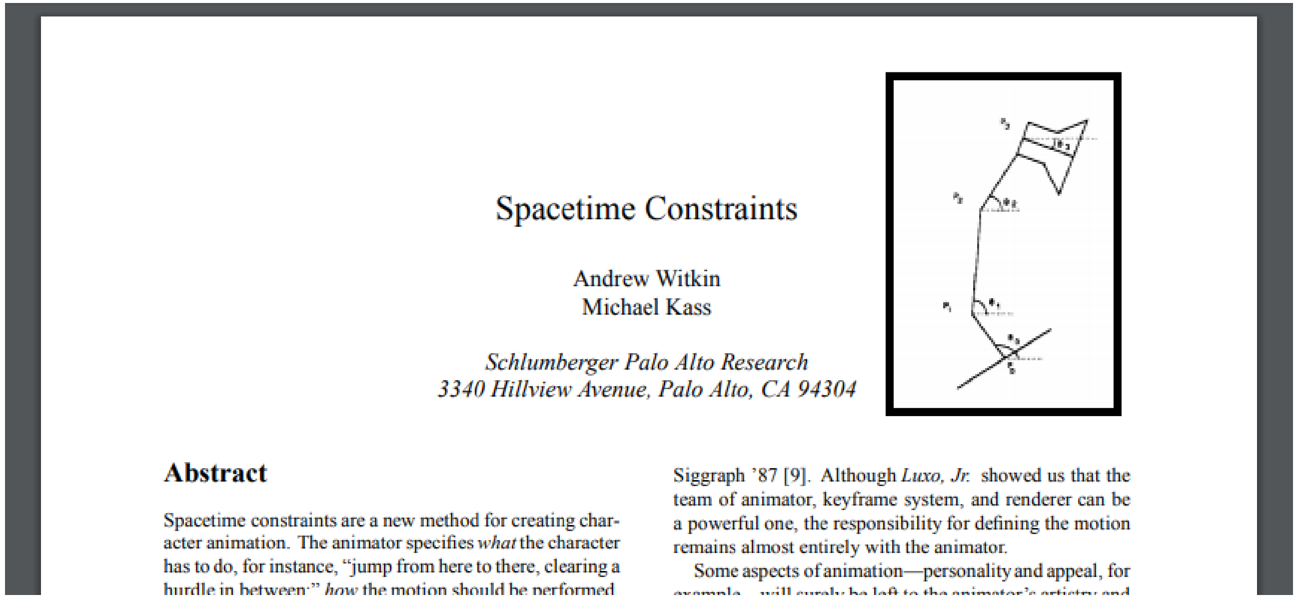


Origin: Spacetime Constraints (1988)

Character animation

Describe:

- What the character should do
- How it should do it (objective)
- Available resources (control input)
- Newton's laws
- Solution: physically valid motion



Spacetime Constraints

Andrew Witkin
Michael Kass

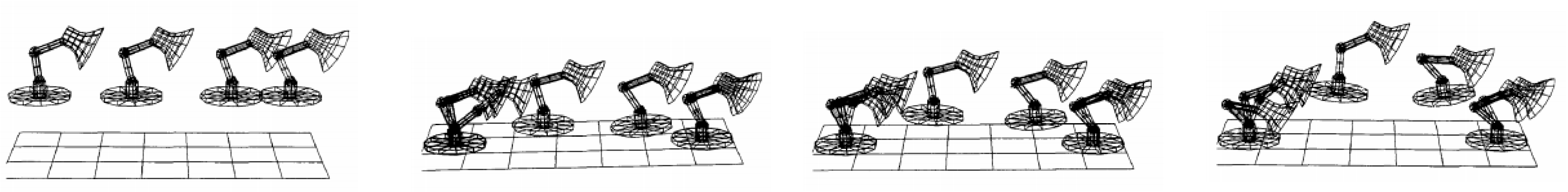
Schlumberger Palo Alto Research
3340 Hillview Avenue, Palo Alto, CA 94304

Abstract

Spacetime constraints are a new method for creating character animation. The animator specifies *what* the character has to do, for instance, "jump from here to there, clearing a hurdle in between;" *how* the motion should be performed

Siggraph '87 [9]. Although *Luxo, Jr.* showed us that the team of animator, keyframe system, and renderer can be a powerful one, the responsibility for defining the motion remains almost entirely with the animator.

Some aspects of animation—personality and appeal, for

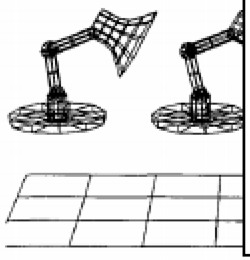


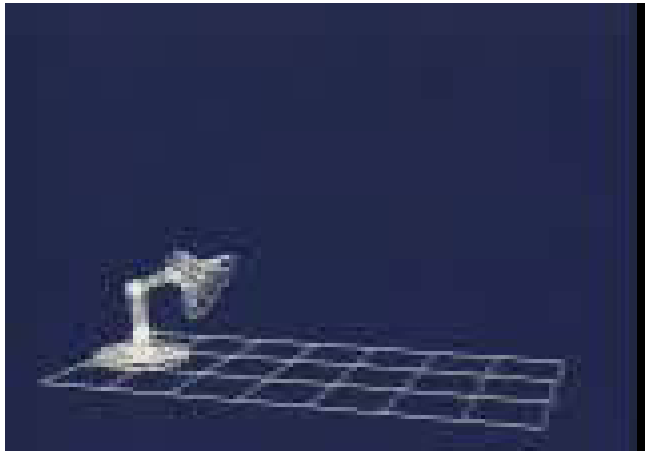
Inspiration: Spacetime Constraints (1988)

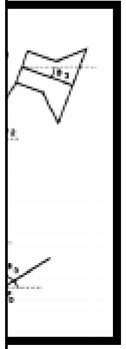
Character animation

Describe

- What th
- How it s
- Availab
- Newton
- Solution

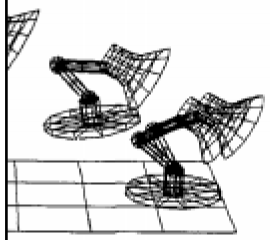






wed us that the
renderer can be
hing the motion

and appeal, for



Questions?

Inertial Sensor Monitoring

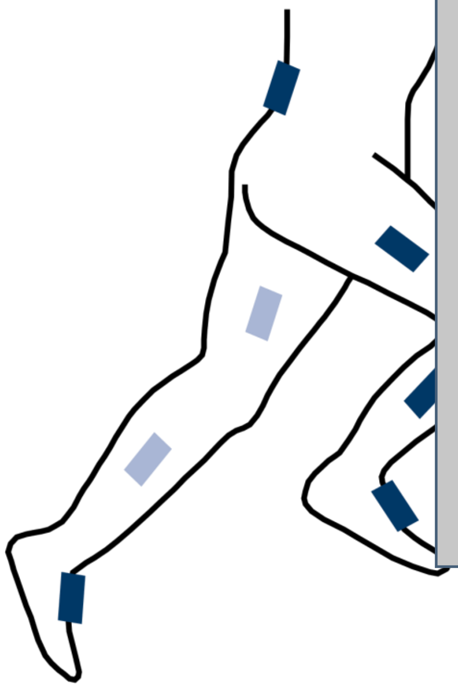
Recap: Virtual Sensor Data

Movement Reconstruction Concept

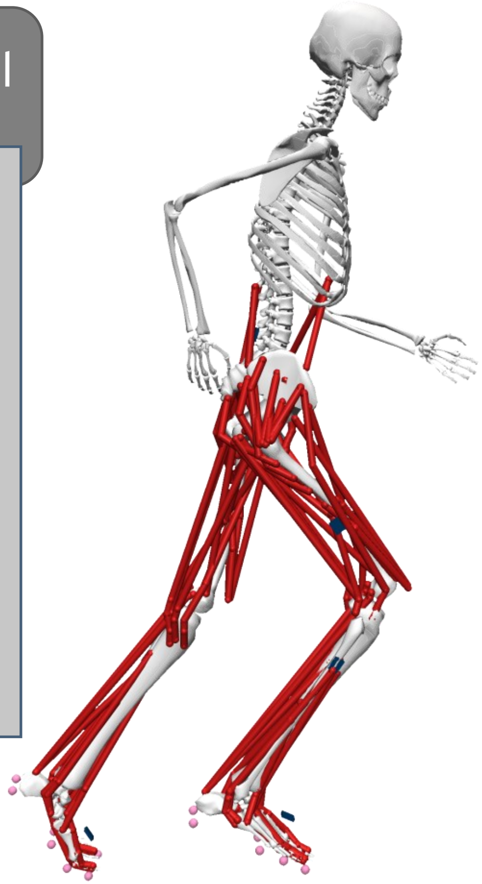
Measured Inertial

Minimization

Virtual Inertial Sensor Data



How do we create virtual inertial sensor data?



Dorschky et al., *J Biomech*, 2019

Virtual Sensor Model

Inertial Sensors

Contain accelerometers and gyroscopes

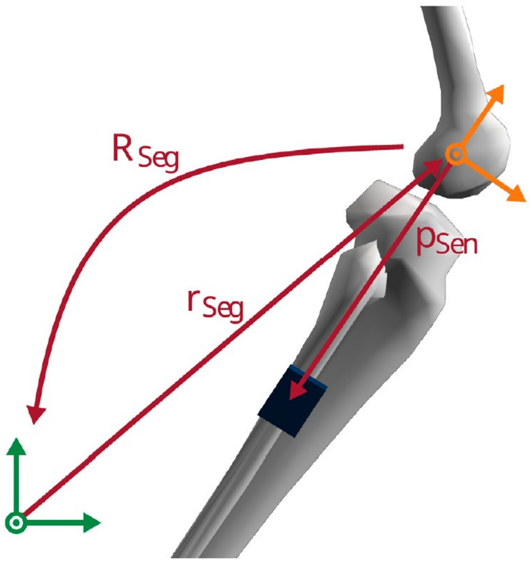
- Accelerometer: linear acceleration $\boldsymbol{a}_s$

$$\boldsymbol{a}_s = \boldsymbol{R}_{Seg}^T\left(\ddot{\boldsymbol{r}}_{Seg} + \ddot{\boldsymbol{R}}_{Seg}\boldsymbol{p}_{Sen} - \boldsymbol{g}\right)$$

- Gyroscope: angular velocity $\boldsymbol{\omega}_s = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$

$$\Omega = \boldsymbol{R}_{Seg}^T\dot{\boldsymbol{R}}_{Seg} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Variable	Meaning
$\mathbf{R}_{Seg}$	Segment orientation matrix
$\boldsymbol{r}_{Seg}$	Location of segment origin
$\boldsymbol{p}_{Sen}$	Local sensor position location
$\boldsymbol{g}$	Gravity



Van den Bogert, Read, Nigg, *J Biomech*, 1996; Nitschke et al., *Front Bioeng Biotech*, 2024

Questions?

Tutorial Today

BioMAC-Sim-Toolbox

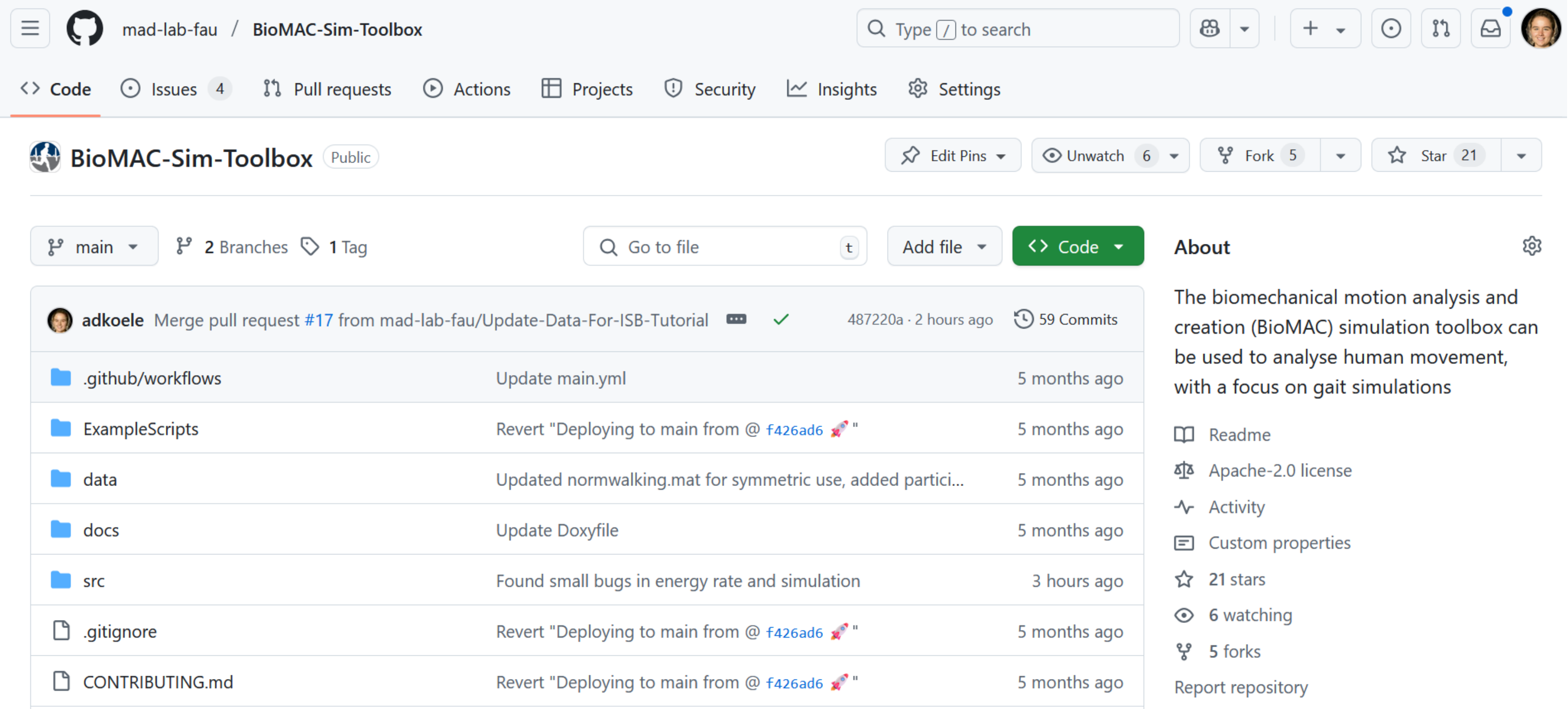
Toolbox available at: https://github.com/mad-lab-fau/BioMAC-Sim-Toolbox

Object oriented

- Object: entity with state, behaviour, and identity
  - Fields: variables providing information

  →Visible in MATLAB's Workspace
  - Methods: actions associated with objects

  →Located in associated folder in **src**



Tutorial

Rest of the Tutorial

Goals

- Get familiar with the code
  - Where can I find which information?
  - How do I create my own optimization?
- Understand impact of several choices
  - What is an appropriate number of nodes?
  - What happens when I change the objective weights?
  - What should my initial guess look like?

Tasks

- Analyse result object → contains all the relevant objects (problem, model, solver)
- Run several simulations and analyse results
- (Optional) Create a constraint function → creating new simulation scripts/functions

Tutorial Introduction

Use the PDF

All result files provided in folder Results

All finished code with "_final.m"

Three sections

1. Getting familiar with result object (15 minutes)
2. Running and comparing simulations (30 minutes)
3. Create your own constraint function (optional)

Two options

- TODO: MATLAB coding/actions
- Questions: Analyse what you see

>> TODO 1. Add path to the location of the BioMAC-Sim-Toolbox in line 24

>> TODO 2: (re)run the code up to line 73 until you get a solution standing on flat feet

Getting Familiar with Result Object

Time: 15 minutes

TODOs

>> TODO 3. Load Section1Example.mat, an example result object.

Questions

Q1. What does it mean when converged is equal to 1? Hint: use the toolbox documentation

Q2. What was the final objective value?

Q3. How long did the optimization take?

Q4. What variables are part of the optimization variables in this problem?

Q5. What was the full objective that was solved for this problem?

Q6. Did we assume left-right symmetry in the problem? Hint: you can find this in the problem itself and in the constraintTerms.

Q7. What are the different states in the model state?

Q8. How many degrees of freedom does the skeletal model have?

Running and Comparing Simulations

Time: 30 minutes

TODOs

- Objective weighting: lines 101-140

>> TODO 4: copy the results that you do not want to run to your result folder. See line 30 in scriptTutorial to find the result folder.

>> TODO 5-8: set up the simulations in lines 105, 107, 112, and 123

- Initial guess: lines 142-178

>> TODO 9-10: set up the simulations in lines 147 and 159

- Number of nodes: lines 180-224

>> TODO 11: set up the simulations in line 194

>> TODO 12: create a convergence plot showing the metabolic cost as a function of the number of nodes in line 223

Questions (qualitative comparisons only):

- Objective weighting

Q9. Would you discard a solution as unrealistic when looking at the ground reaction forces and activations?

Q10. How much difference do you see in the joint angles and joint moments of the acceptable solutions?

- Initial guess

Q11: Do you see any differences between the three simulations?

- Number of nodes

Q12. What does a convergence plot showing metabolic cost versus the number of nodes tell you about a suitable number of nodes?

Q13: Based on the joint angles and joint moments, from which number of nodes do you not see a clear difference between the different simulations anymore?

Create your own constraint function

Optional

TODOs

>> TODO optional 1: give the function move2D_IMU a new name in line 27 and save it accordingly with the new name.

>> TODO optional 2: comment line 83 and uncomment line 82 in the new function to remove the strict bounds on the optimization variable

>> TODO optional 3-4: add the output and Jacobian according to the comments in line 30 and 33 of speedConstraint_IMU.

>> TODO optional 5: move this function from the tutorial folder into the @Collocation folder

>> TODO optional 6: ensure that the file is added to the path

>> TODO optional 7: add the code in line 96 of the new move2D_IMU to ensure that speedConstraint_IMU is used.

>> TODO optional 8: create a problem with a small number of nodes, e.g., 4, to do the derivative test in line 233 of scriptTutorial.

>> TODO optional 9. Add code to run the derivative test function in line 234

>> TODO optional 10. Run the new optimization by adding code in line 246

Questions

Q14: Do you think that the constraint function is correct?

Q15: Do you get a different result than when the speed is set using the bounds?

Questions?

Acknowledgements

Co-creators

- Dr.-Ing. Eva Dorschky and Dr.-Ing. Marlies Nitschke and many others

Funding sources

- Adidas: Heiko Schlarb and Tobias Luckfiel
- DFG - Projektnummer 520189992

Code testers

- Ass.-Prof. Maurice Mohr, Mareike Kühne, and Daniel Oberhubers
- Birte Coppers, Long Han, Julian Shanbhag, and Iris Wechsler

BioMAC support

- Maria Eleni Athanasiadou, Sophie Fleischmann, Markus Gambietz, Long Han, Alexander Weiss

"In the wild" movement analysis using dynamic simulations

Anne Koelewijn and Ton van den Bogert
With help from: Biomechanical Motion Analysis and Creation (BioMAC) group
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
Cleveland State University, Cleveland, USA