Solving an optimization problem

for maximizing the forward velocity of arm model

(Mohammad Shabani)

Introduction

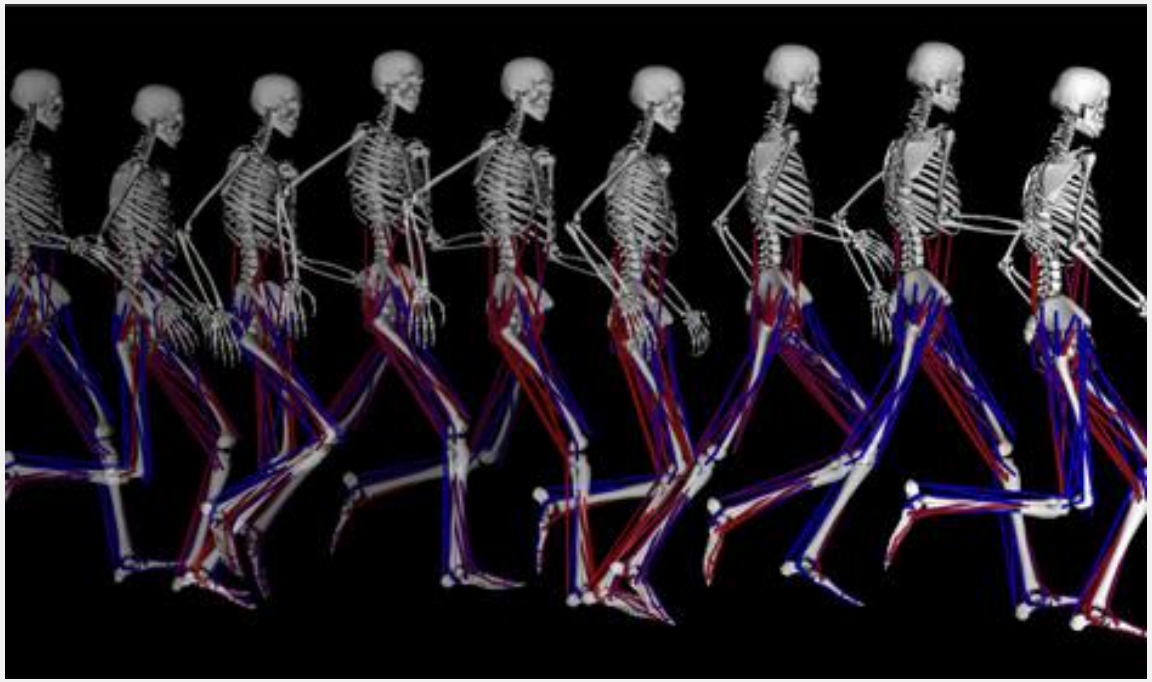
Method

Goals

Introduction

What is OpenSim?

- OpenSim is a musculoskeletal simulation toolkit



Introduction

What is OpenSim?

- OpenSim is a musculoskeletal simulation toolkit

Introduction

What does OpenSim do?

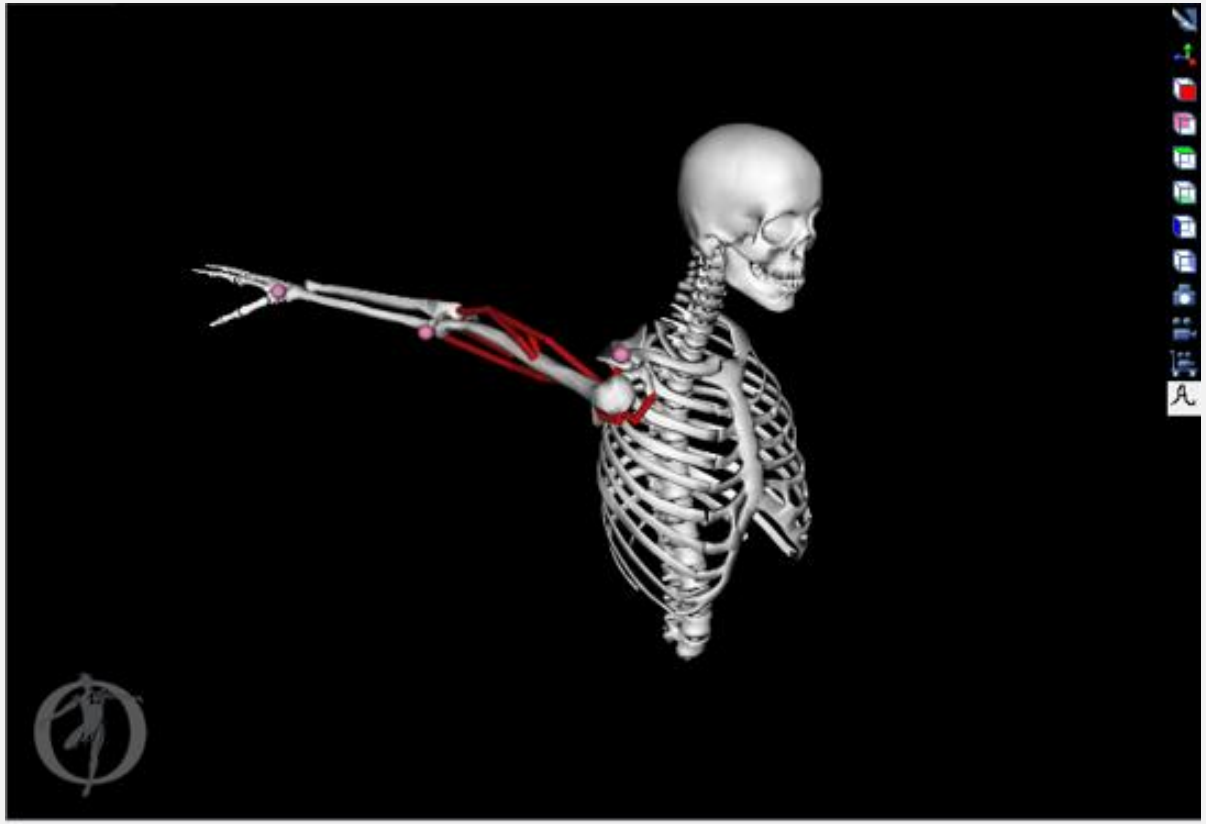
- Using it as a developer
  - You can implement your own model
  - You can improve it (it is open source)
- Using it as an end user
  - You can use an implemented model to analyze it (using GUI)

Introduction

What is the model we are interested to?

A simple model of arm consisting of:

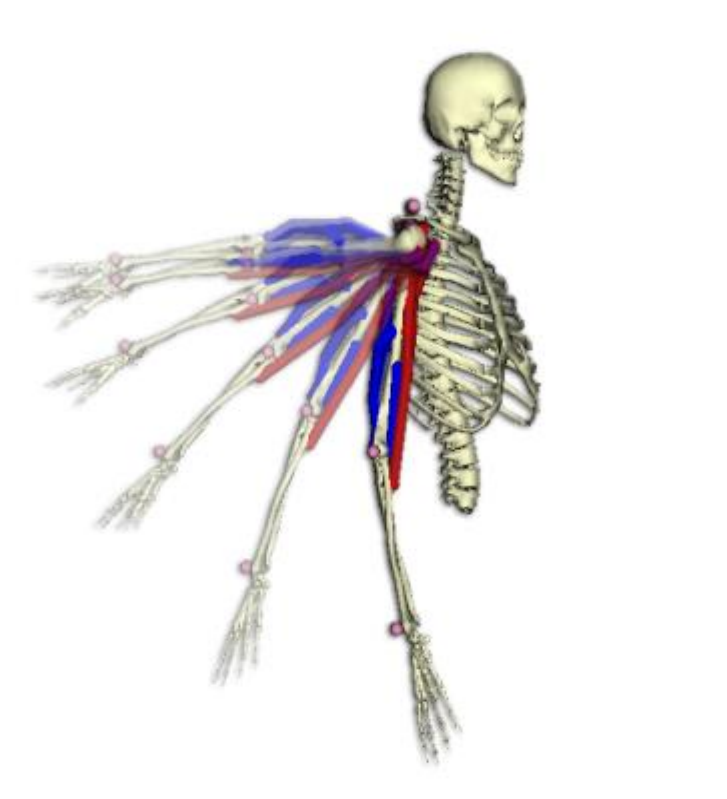
- 2 bones
- 6 muscles
- 2 joints



Introduction

What we are going to do?

- Maximizing forward velocity of wrist
- By finding optimized muscle control
- We should solve an optimization problem



Introduction

What is the problem?

- It takes a lot of time
- Optimizer is not fast enough

```
c:\samuel\OptimizerExampleAmBuild\RelWithDebInfo\main.exe

Optimization Step #: 2  controls = ~[0.01 0.01 0.01 0.01 0.01 0.01] bestSoFar =
-3.0077

Optimization Step #: 4  controls = ~[0.00999728 0.01 0.01 0.01 0.01 0.01] bestSo
Far = -3.00783

Optimization Step #: 6  controls = ~[0.01 0.00999728 0.01 0.01 0.01 0.01] bestSo
Far = -3.0079

Optimization Step #: 15  controls = ~[0.01 0.01 0.01 1 1 0.01] bestSoFar = -3.87
628

Optimization Step #: 20  controls = ~[0.01 0.00999728 0.01 1 1 0.01] bestSoFar =
 -3.87636

Optimization Step #: 29  controls = ~[0.01 0.01 0.0577658 1 1 0.0594141] bestSoF
ar = -4.0675

Optimization Step #: 31  controls = ~[0.0100027 0.01 0.0577658 1 1 0.0594141] be
stSoFar = -4.06773

Optimization Step #: 33  controls = ~[0.01 0.0100027 0.0577658 1 1 0.0594141] be
stSoFar = -4.06773
```

- We should repeat simulation

for a huge number of input arguments

Introduction

Method

Goals

Method

What is the solution?

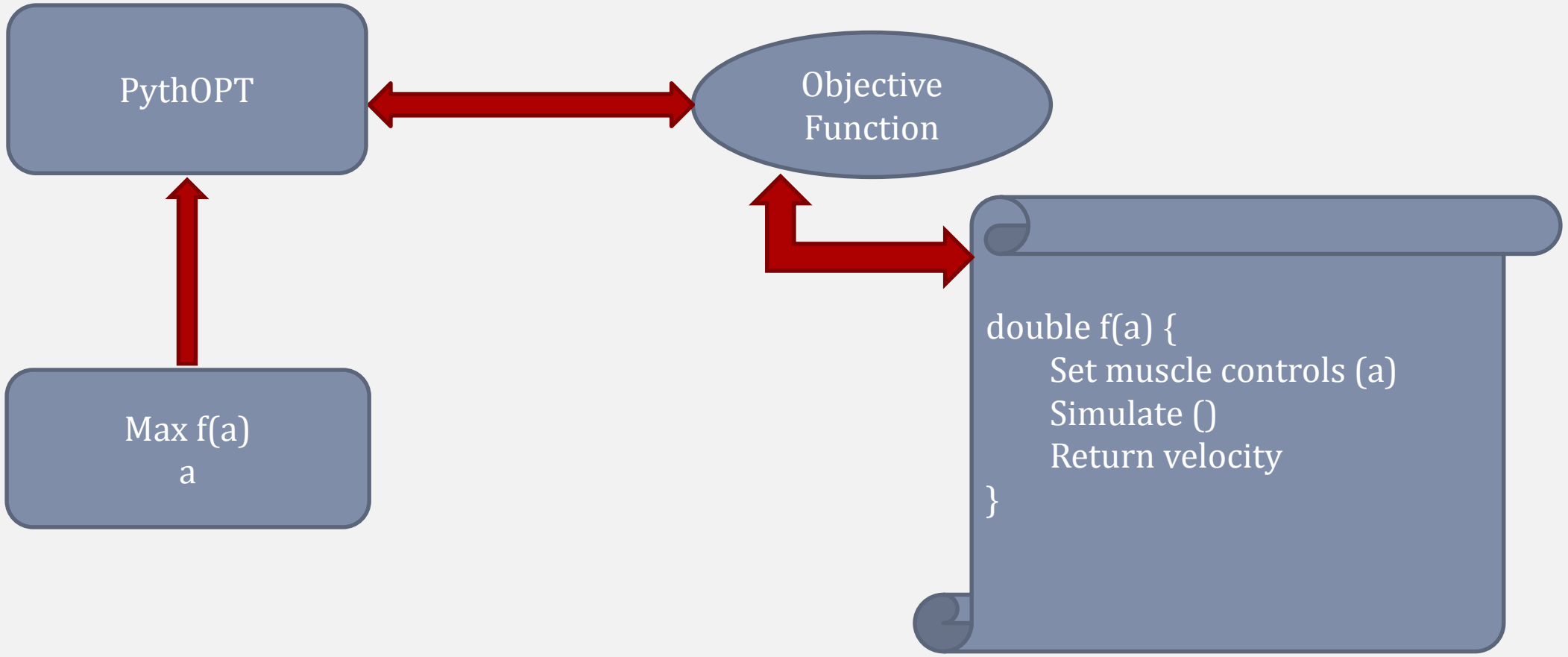
- We need to reduce the execution time
- Same process is repeated for every input parameters
- This program is easily parallelizable

Method

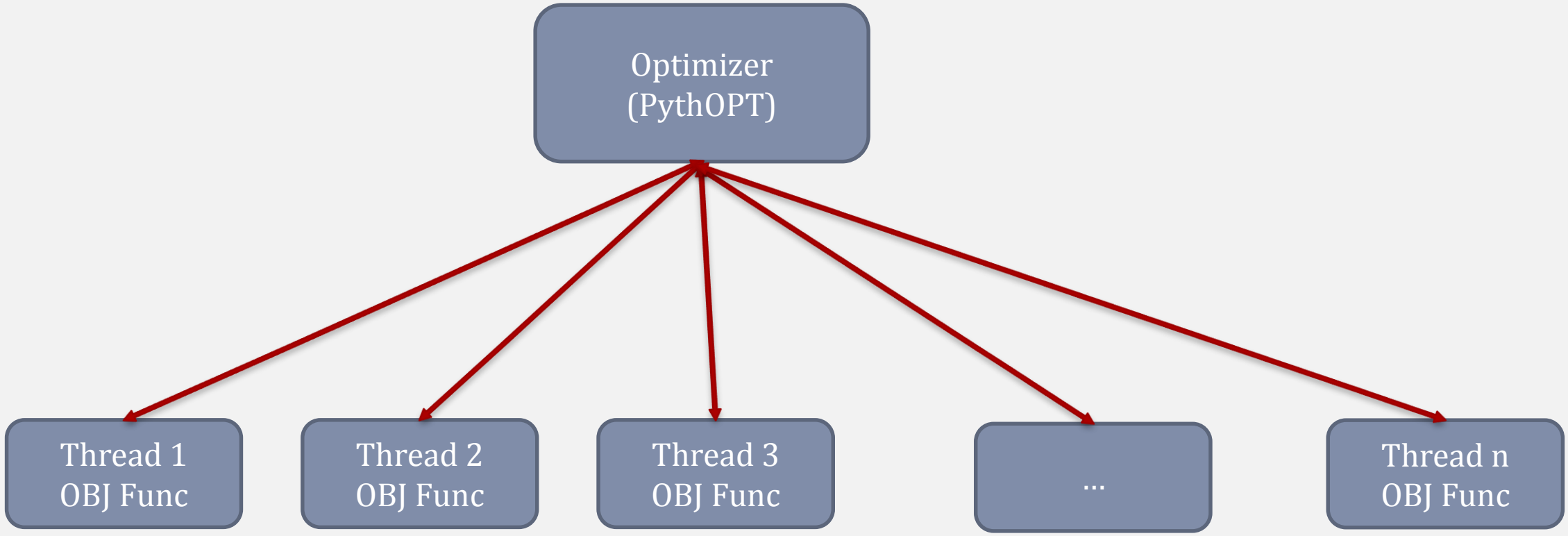
The idea to solve the problem:

- This program is easily parallelizable so we can run it on a cluster
- Using PythOPT as our new solver for optimizing
  - PythOpt is a solver
  - It can call an application
  - We can run our C++ code from PythOPT

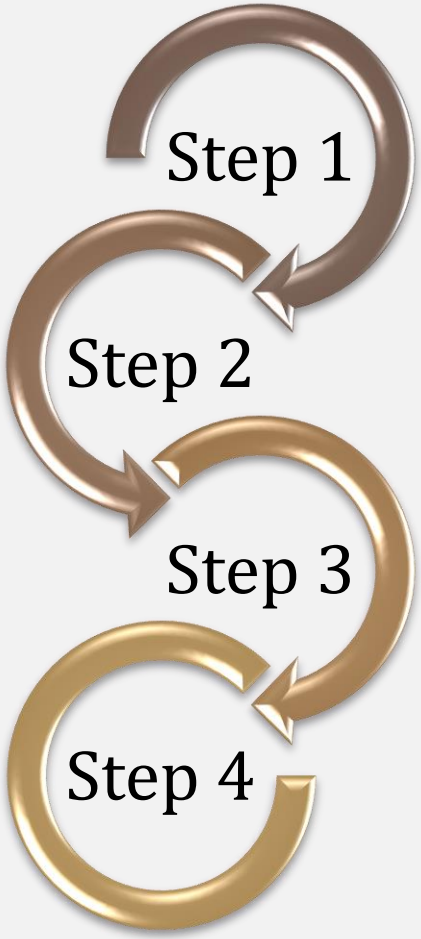
Method


PythOPT
Objective Function
Max f(a)
a
double f(a) {
Set muscle controls (a)
Simulate ()
Return velocity
}

Method


Optimizer
(PythOPT)
Thread 1
OBJ Func
Thread 2
OBJ Func
Thread 3
OBJ Func
...
Thread n
OBJ Func

Method



- Investigating PythOPT and implemented code in OpenSim
- Connect PythOPT to a simple C++ program
- Connecting main OpenSim program to PythOPT
- Running code on a cluster (using MPI)

Introduction

Method

Goals

Goals

- Using PythOPT as our solver
- Running the program on a cluster
- Getting results in a less time

Goals

- Test cases: we want to see how using PythOpt and parallelism affect our problem
  - Running **serial** code with PythOPT
  - Measuring execution time with and without PythOPT
  - Can see the improvement on execution time

Goals

- Test cases: we want to see how using PythOpt and parallelism affect our problem
  - Running final code on a cluster using PythOPT
  - Can see gains on speed up totally

Goals

- Test cases: we want to see how using PythOpt and parallelism affect our problem
  - All comparisons are based on execution time
  - With a certain amount of samples