Introduction to Game Programming and Robotics

Unit # 16

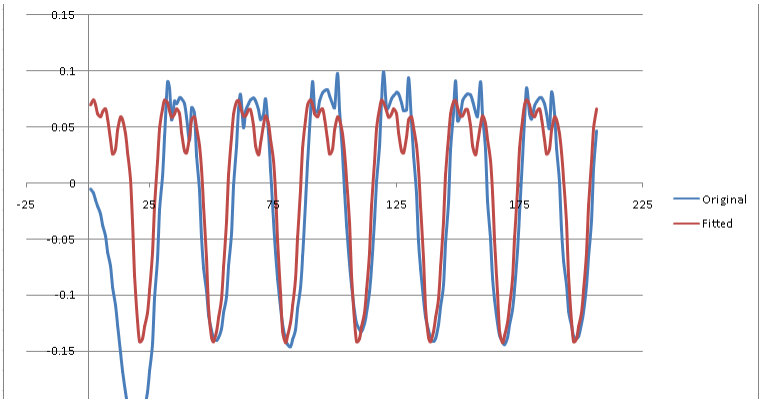
Partial Fourier Series based Locomotion

- Each joint's motion is followed by the points generated by PFS

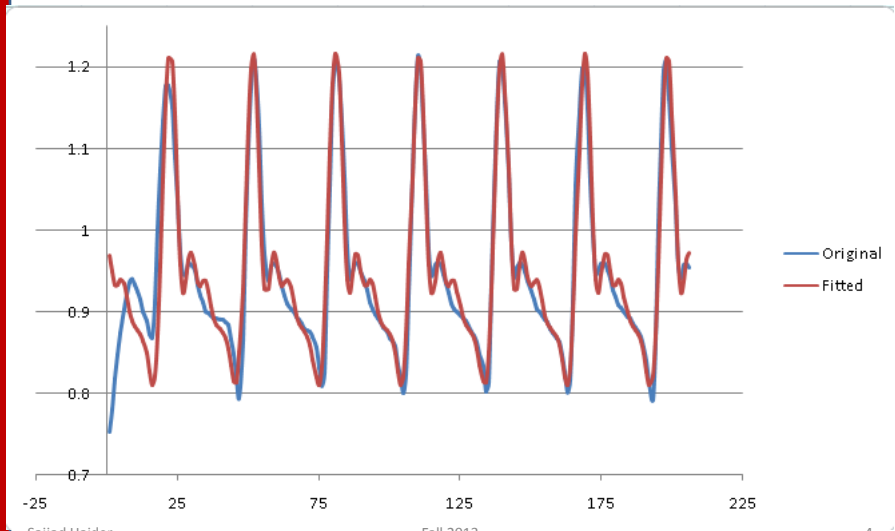
$$f(t) = C + \sum_{n=1}^{N} A_n \sin\left(n\frac{2\pi}{T}t + \phi_n\right), \forall t \in \Re$$

- In theory, N varies for different joints and depending upon its value we need to learn the corresponding number of A, C, T and ϕ parameters.
- However, for our walk, we are using N = 1 for all joints

Knee Pitch



Hip Pitch



Getting Current Joints' Values

- angles[2] = Body.LLJ1.Angle.Radians;
- angles[3] = Body.LLJ3.Angle.Radians;
- angles[4] = Body.RLJ3.Angle.Radians;
- angles[5] = Body.LLJ2.Angle.Radians;
- angles[6] = Body.RLJ2.Angle.Radians;
- angles[7] = Body.LLJ4.Angle.Radians;
- angles[8] = Body.RLJ4.Angle.Radians;
- angles[9] = Body.LLJ5.Angle.Radians;
- angles[10] = Body.RLJ5.Angle.Radians;
- angles[11] = Body.LLJ6.Angle.Radians;
- angles[12] = Body.RLJ6.Angle.Radians;

Computing new Values using PFS

- targets[2] = 0.1; // hip yaw
- targets[3] = 0.6 + 0.4 * Math.Sin(2 * Math.PI * t / T - Math.PI / 2 + 1); // hip pitch
- targets[4] = 0.6 + 0.4 * Math.Sin(2 * Math.PI * t / T + Math.PI / 2 + 1);
- targets[5] = 0.03 + 0.03 * Math.Sin(2 * Math.PI * t / T - Math.PI / 2 - 0.3); // hip roll
- targets[6] = -0.03 + 0.03 * Math.Sin(2 * Math.PI * t / T + Math.PI / 2 - 0.3);
- targets[7] = -0.9 + 0.3 * Math.Sin(2 * Math.PI * t / T - Math.PI / 2 - 1); // knee pitch
- targets[8] = -0.9 + 0.3 * Math.Sin(2 * Math.PI * t / T + Math.PI / 2 - 1);
- targets[9] = 0.6 + 0.3 * Math.Sin(2 * Math.PI * t / T + Math.PI / 2 + 0.1); // ankle pitch
- targets[10] = 0.6 + 0.3 * Math.Sin(2 * Math.PI * t / T - Math.PI / 2 + 0.1);
-
- targets[11] = -0.03 + 0.03 * Math.Sin(2 * Math.PI * t / T + Math.PI / 2 - 0.3); //ank roll
- targets[12] = 0.03 + 0.03 * Math.Sin(2 * Math.PI * t / T - Math.PI / 2 - 0.3);

Passing Values to Joints

- Body.LLJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[2]);
  Body.RLJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[2]);
- Body.LLJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[5]);
  Body.RLJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[6]);
- Body.LLJ3.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[3]);
  Body.RLJ3.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[4]);
- Body.LLJ4.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[7]);
  Body.RLJ4.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[8]);
- Body.LLJ5.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[9]);
  Body.RLJ5.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[10]);
- Body.LLJ6.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[11]);
- Body.RLJ6.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[12]);

Additional Skills

- Am I standing or am on the ground?
- If I have fallen on the ground then I need to stand up
  - From back
  - From belly
- How to I align myself towards the ball if it's not in my sight?
  - Turning

Turn Equations

- targets[2] = -0.05 - 0.1 * Math.Sin(2 * Math.PI * t / T + Math.PI/2 + 1); //hip yaw
- targets[1] = -0.05 - 0.1 * Math.Sin(2 * Math.PI * t / T - Math.PI/2 + 1); //hip yaw
-
- targets[3] = 0.42;
- targets[4] = 0.42;
-
- targets[7] = -0.95;
- targets[8] = -0.95;
-
- targets[9] = 0.4;
- targets[10] = 0.4;

For Left Turn

- targets[5] = 0.256 + 0.4 * Math.Sin(2 * Math.PI * t / T + Math.PI/2 - 1.5); // hip roll
- targets[6] = -0.256 + 0.4 * Math.Sin(2 * Math.PI * t / T - Math.PI + 1.5);
-
- targets[11] = -0.256 - 0.4 * Math.Sin(2 * Math.PI * t / T + Math.PI/2 - 0.3); // ankle roll
- targets[12] = 0.256 - 0.4 * Math.Sin(2 * Math.PI * t / T - Math.PI + 0.3);

For Right Turn

- targets[6] = 0.256 + 0.4 * Math.Sin(2 * Math.PI * t / T + Math.PI/2 - 1.5); // hip roll
- targets[5] = -0.256 + 0.4 * Math.Sin(2 * Math.PI * t / T - Math.PI + 1.5);
-
- targets[12] = -0.256 - 0.4 * Math.Sin(2 * Math.PI * t / T + Math.PI/2 - 0.3); // ankle roll
- targets[11] = 0.256 - 0.4 * Math.Sin(2 * Math.PI * t / T - Math.PI + 0.3);
-
- targets[5] = -1*targets[5];
- targets[6] = -1*targets[6];
-
- targets[11] = -1*targets[11];
- targets[12] = -1*targets[12];

Project Details (Mandatory Part)

- You need to develop an agent that can score a goal.
- You would be given 5 minutes to score as many goals as possible
- To achieve this task, you need to have the following routines ready
  - Locomotion (Walking, Turning, GetUpFromBack, GetUpFromBelly)
  - Localization
  - Strategy

Project Details (Bonus Part)

- You can also develop a goalie that can block goals.
- If you achieve this then you will be allowed to place your goalie against other teams' agents.