Winter Semester 2017/18

Leibniz Universität Hannover

Simulation Engineering

Homework 2 : Circus Trapeze Real Time Simulation

Vaibhav Kasturia
M.Sc. ITIS

Dr. Umut Durak
Department of Informatics, TU Clausthal

Euler Functions

- Functions to calculate theta and velocity.

```
17  // Function to calculate theta
18  double calc_function1(double time, double velocity){
19      return velocity*time;
20  }
21
22  // Function to calculate Velocity (in loop iteration) using Runge Kutta Method
23  double calc_function2(double time, double theta){
24      return -g/length*sin(theta)*time;
25  }
```

Euler Functions

- Functions used in calculating parameters for finding next theta and next velocity using Range Kutta Method.

```
27  //Function used in Calculating parameters for finding next theta for pendulum using Runge Kutta Method.
28  double calc_range1(double time, double velocity){
29      double h = time/2;
30      double k1= calc_function1(time, velocity);
31      double k2 = calc_function1((time+(h/2)), (velocity+h/2*(k1)));
32      double k3 = calc_function1((time+(h/2)), (velocity+h/2*(k2)));
33      double k4 = calc_function1((time+(h)), (velocity+h*(k3)));
34
35      double val = h*(k1 + 2*k2 + 2*k3 + k4)/6;
36      return val;
37
38  }
39
40  //Function used in Calculating parameters for finding next velocity for pendulum
41  double calc_range2(double time, double theta){
42      double h = time/2;
43      double k1= calc_function2(time, theta);
44      double k2 = calc_function2((time+(h/2)), (theta+h/2*(k1)));
45      double k3 = calc_function2((time+(h/2)), (theta+h/2*(k2)));
46      double k4 = calc_function2((time+(h)), (theta+h*(k3)));
47
48      double val = (h/6*(k1 + 2*k2 + 2*k3 + k4));
49      return val;
50  }
51
```

Circus Trapeze Real Time

Main

- Create Thread with simulation function

```
86  int main ()
87  {
88      simulation_struct.sim_mode = 0;
89      simulation_struct.sim_time = 0.0;
90      simulation_struct.frame_time = 0.010;
91      simulation_struct.sim_end_time = 10;
92      pthread_t simulation_thread;
93
94      int status;
95
96      status = pthread_create(&simulation_thread, NULL, simulation_function, NULL);
97      simulation_struct.sim_mode =1;
98      while (simulation_struct.sim_time<simulation_struct.sim_end_time);
99  }
```

Circus Trapeze Real Time

Simulation Function

- Timer Handler installed as Signal Handler for SIGVTALARM
- Configure Timer to expire after the given frame time, when initialization is done
- Call function to initialize the simulation

```
101 void *simulation_function(void *ptr){
102
103     struct sigaction sa;
104     struct itimerval timer;
105
106     // Install timer_handler as the signal handler for SIGVTALRM.
107     memset (&sa, 0, sizeof (sa));
108     sa.sa_handler = &timer_handler;
109     sigaction (SIGVTALRM, &sa, NULL);
110
111     // Configure the timer to expire after the given frame time, when initialization is done
112     // Value that is fed is in microseconds
113
114     timer.it_value.tv_sec = 0;
115     timer.it_value.tv_usec = simulation_struct.frame_time*10000;
116
117     //After first time frame, we repeat this after each time frame
118     timer.it_interval.tv_sec = 0;
119     timer.it_interval.tv_usec = simulation_struct.frame_time * 1000;
120
121     //Start a virtual timer.
122     //Whenever process executes, countdown is done
123     initialise_simulation(&parameters, &states);
124
125     setitimer (ITIMER_VIRTUAL, &timer, NULL);
126     return NULL;
127 }
```

Circus Trapeze Real Time

Initialize Simulation Function

- Initialize the simulation
- Parameters: Acceleration due to gravity, length, length of man, end time, delta time, start angle
- States: Angle (Theta), Velocity, Time

```
69  void initialise_simulation(parameter_struct_type* par, state_struct_type* state){
70
71      parameters.g = 9.8;
72      parameters.length = 5;
73      parameters.length_of_man= 2.4;
74      parameters.end_time = 2.16 * 3.14 * sqrt(length / g);
75      parameters.delta_time= 0.1;
76      parameters.start_angle= 55;
77
78      states.theta = parameters.start_angle * 6.28 / 360;
79      states.velocity = 0;
80      states.time = 0;
81
82  }
```

Circus Trapeze Real Time

Timer Handler

- Calls the do_step function
- Prints time, theta and velocity

```
54  void timer_handler (int signum)
55  {
56      simulation_struct.sim_time +=simulation_struct.frame_time;
57      do_step(&parameters, &states);
58      cout<<states.time*10<<"\t\t\t\t\t\t\t"<<states.theta*360/6.28<<"\t\t\t"<< states.velocity<<"\t\t\t"<<"\n";
59  }
60
```

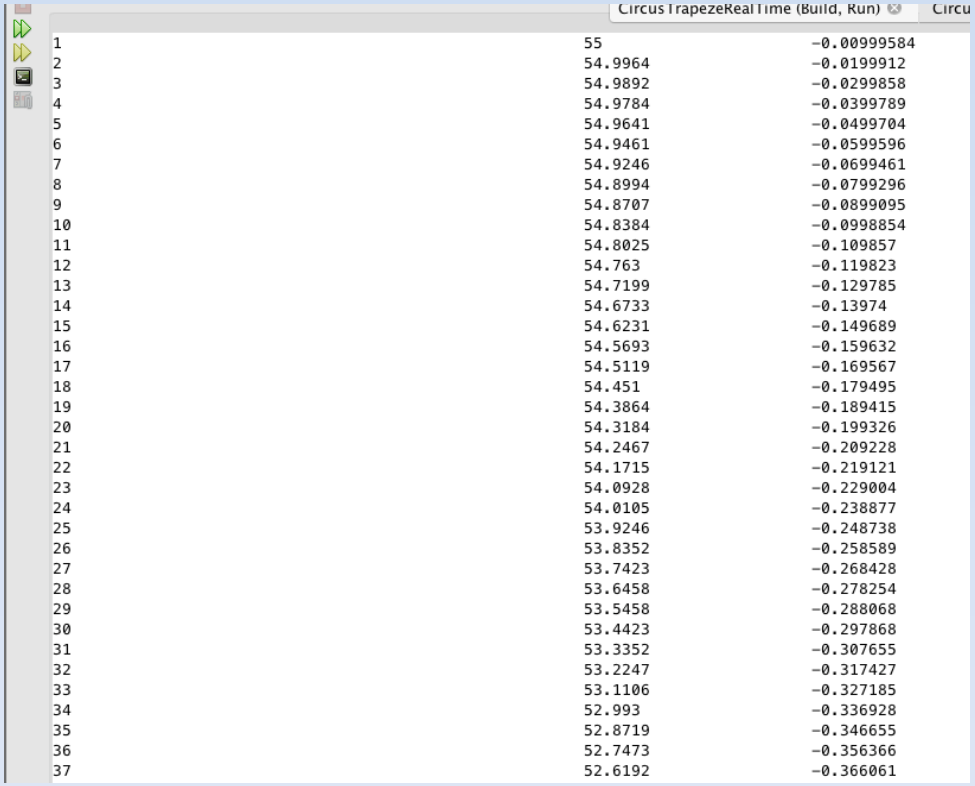
do_step

- Calculates next states: time, velocity and theta using Runge Kutta Method

```
45  void do_step(parameter_struct_type* par, state_struct_type* state){
        //Calculation done using Runge Kutta Method
47      states.theta += calc_range1(parameters.delta_time, states.velocity);
48      states.velocity += calc_range2(parameters.delta_time, states.theta);
49      states.time += parameters.delta_time;
50
51  }
```

Output

- Notice how the velocity and theta change
- Velocity is at a minimum at the initial angle and reaches a maximum as theta approaches 0
- Minus sign indicates the direction of velocity



1	55	-0.00999584
2	54.9964	-0.0199912
3	54.9892	-0.0299858
4	54.9784	-0.0399789
5	54.9641	-0.0499704
6	54.9461	-0.0599596
7	54.9246	-0.0699461
8	54.8994	-0.0799296
9	54.8707	-0.0899095
10	54.8384	-0.0998854
11	54.8025	-0.109857
12	54.763	-0.119823
13	54.7199	-0.129785
14	54.6733	-0.13974
15	54.6231	-0.149689
16	54.5693	-0.159632
17	54.5119	-0.169567
18	54.451	-0.179495
19	54.3864	-0.189415
20	54.3184	-0.199326
21	54.2467	-0.209228
22	54.1715	-0.219121
23	54.0928	-0.229004
24	54.0105	-0.238877
25	53.9246	-0.248738
26	53.8352	-0.258589
27	53.7423	-0.268428
28	53.6458	-0.278254
29	53.5458	-0.288068
30	53.4423	-0.297868
31	53.3352	-0.307655
32	53.2247	-0.317427
33	53.1106	-0.327185
34	52.993	-0.336928
35	52.8719	-0.346655
36	52.7473	-0.356366
37	52.6192	-0.366061

Some Output observations

Q & A