ZZH

张桢
swip_zhang_zhen@foxmail.com

2026 年 2 月 2 日

摘要

...

关键词： ZZH

目录

1 note

1.1 note

1.1.1 note

$$double\ m\&l \tag{1}$$

$$t\ \theta(t)\ \varphi(t) \tag{2}$$

$$m_1 = m_2 = m \tag{3}$$

$$l_1 = l_2 = l \tag{4}$$

$$for\ m_1 \tag{5}$$

$$T_1 = \frac{1}{2}ml^2\dot{\theta}^2 \tag{6}$$

$$V_1 = -mgl\cos(\theta) \tag{7}$$

$$L_1 = T_1 - V_1 \tag{8}$$

$$for\ m_2 \tag{9}$$

$$V_2 = -mgl(\cos(\theta) + \cos(\varphi)) \tag{10}$$

$$T_2 = \frac{1}{2}ml^2[\dot{\theta}^2 + \dot{\varphi}^2 + 2\dot{\theta}\dot{\varphi}\cos(\theta - \varphi)] \tag{11}$$

$$L_2 = T_2 - V_2 \tag{12}$$

$$L = \frac{1}{2}ml^2[2\dot{\theta}^2 + \dot{\varphi}^2 + 2\dot{\theta}\dot{\varphi}\cos(\theta - \varphi)] + mgl[2\cos(\theta) + \cos(\varphi)] \tag{13}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) = \frac{\partial L}{\partial \theta} \ \rightarrow \tag{14}$$

$$2ml^2\ddot{\theta} + ml^2\ddot{\varphi}\cos(\theta - \varphi) - ml^2\dot{\varphi}\sin(\theta - \varphi)(\dot{\theta} - \dot{\varphi}) = -2mgl\sin(\theta) - ml^2\dot{\theta}\dot{\varphi}\sin(\theta - \varphi) \tag{15}$$

$$2\ddot{\theta} + \ddot{\varphi}\cos(\theta - \varphi) + \dot{\varphi}^2\sin(\theta - \varphi) + 2\frac{g}{l}\sin(\theta) = 0 \tag{16}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{\varphi}}) = \frac{\partial L}{\partial \varphi} \ \rightarrow \tag{17}$$

$$ml^2\ddot{\varphi} + ml^2\ddot{\theta}\cos(\theta - \varphi) - ml^2\dot{\theta}\sin(\theta - \varphi)(\dot{\theta} - \dot{\varphi}) = -mgl\sin(\varphi) + ml^2\dot{\theta}\dot{\varphi}\sin(\theta - \varphi) \tag{18}$$

$$\ddot{\varphi} + \ddot{\theta}\cos(\theta - \varphi) - \dot{\theta}^2\sin(\theta - \varphi) + \frac{g}{l}\sin(\varphi) = 0 \tag{19}$$

$$\rightarrow \tag{20}$$

$$\ddot{\varphi} + \ddot{\theta}\cos(\theta - \varphi) - \dot{\theta}^2\sin(\theta - \varphi) + \frac{g}{l}\sin(\varphi) = 0 \tag{21}$$

$$2\ddot{\theta} + \ddot{\varphi}\cos(\theta - \varphi) + \dot{\varphi}^2\sin(\theta - \varphi) + 2\frac{g}{l}\sin(\theta) = 0 \tag{22}$$

$$\vec{y} = \begin{pmatrix} \theta \\ \varphi \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \tag{23}$$

$$\dot{\vec{y}} = \begin{pmatrix} y_3 \\ y_4 \\ \frac{1}{\cos^2(y_1 - y_2) - 2}[\sin(y_1 - y_2)(\cos(y_1 - y_2)y_3^2 + y_4^2) + \frac{g}{l}(2\sin y_1 - \cos(y_1 - y_2)\sin y_2)] \\ \frac{1}{\cos^2(y_1 - y_2) - 2}[-\sin(y_1 - y_2)(2y_3^2 + \cos(y_1 - y_2)y_4^2) + \frac{g}{l}(2\sin y_2 - 2\cos(y_1 - y_2)\sin y_1)] \end{pmatrix} \tag{24}$$

$$\vec{y} = \begin{pmatrix} \theta \\ \varphi \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} ; \dot{\vec{y}} = \begin{pmatrix} y_3 \\ y_4 \\ \frac{1}{\cos^2(y_1 - y_2) - 2}[\sin(y_1 - y_2)(\cos(y_1 - y_2) y_3^2 + y_4^2) + \frac{g}{l}(2 \sin y_1 - \cos(y_1 - y_2) \sin y_2)] \\ \frac{1}{\cos^2(y_1 - y_2) - 2}[-\sin(y_1 - y_2)(2y_3^2 + \cos(y_1 - y_2) y_4^2) + \frac{g}{l}(2 \sin y_2 - 2\cos(y_1 - y_2) \sin y_1)] \end{pmatrix}$$

end

```
module constants_module
implicit none
public

! 物理常量
real(8), parameter :: g = 9.81d0          ! 重力加速度 (m/s^2)
real(8), parameter :: l = 1.0d0           ! 摆长 (m)
real(8), parameter :: m = 1.0d0           ! 质量 (kg)

! 模拟参数
real(8), parameter :: t_end = 20.0d0      ! 结束时间 (s)
real(8), parameter :: dt = 0.001d0        ! 时间步长 (s)
integer, parameter :: n = 4               ! 系统维度
integer, parameter :: n_steps = nint(t_end / dt)  ! 总步数

! 初始条件 ( ,  , _dot, _dot)
real(8), parameter :: y0(4) = [0.1d0, 0.2d0, 0.0d0, 0.0d0]

end module constants_module
```

```
module rk4_solver_module
use constants_module, only: n, dt, g, l
implicit none
private
public :: rk4_step

contains

! 双摆微分方程 - 严格按你的公式
subroutine derivs(t, y, dydt)
    real(8), intent(in) :: t
    real(8), intent(in) :: y(n)          ! y = [ ,  , _dot, _dot]
    real(8), intent(out) :: dydt(n)

    real(8) :: theta, phi, theta_dot, phi_dot
    real(8) :: diff, cos_diff, sin_diff, denom

    ! 提取变量
    theta = y(1)      !
    phi = y(2)        !
    theta_dot = y(3) !  _dot
    phi_dot = y(4)   !  _dot

    ! 角度差
```

```
        diff = theta - phi
        cos_diff = cos(diff)
        sin_diff = sin(diff)

        ! 分母 cos^2( - ) - 2
        denom = cos_diff**2 - 2.0d0

        ! 状态方程
        dydt(1) = theta_dot  ! d /dt =  _dot
        dydt(2) = phi_dot    ! d /dt =  _dot

        ! d _dot/dt - 严格按照你的公式
        dydt(3) = (1.0d0 / denom) * &
                 (sin_diff * (cos_diff * theta_dot**2 + phi_dot**2) + &
                 (g/l) * (2.0d0 * sin(theta) - cos_diff * sin(phi)))

        ! d _dot/dt - 严格按照你的公式
        dydt(4) = (1.0d0 / denom) * &
                 (-sin_diff * (2.0d0 * theta_dot**2 + cos_diff * phi_dot**2) + &
                 (g/l) * (2.0d0 * sin(phi) - 2.0d0 * cos_diff * sin(theta)))

    end subroutine derivs

    ! 经典四阶龙格库塔单步推进
    subroutine rk4_step(t, y, y_next)
        real(8), intent(in) :: t
        real(8), intent(in) :: y(n)
        real(8), intent(out) :: y_next(n)

        real(8) :: k1(n), k2(n), k3(n), k4(n)
        real(8) :: y_temp(n)

        ! 第一步
        call derivs(t, y, k1)

        ! 第二步
        y_temp = y + 0.5d0 * dt * k1
        call derivs(t + 0.5d0 * dt, y_temp, k2)

        ! 第三步
        y_temp = y + 0.5d0 * dt * k2
        call derivs(t + 0.5d0 * dt, y_temp, k3)

        ! 第四步
        y_temp = y + dt * k3
        call derivs(t + dt, y_temp, k4)

        ! 组合
        y_next = y + (dt / 6.0d0) * (k1 + 2.0d0 * k2 + 2.0d0 * k3 + k4)

    end subroutine rk4_step

    end module rk4_solver_module
```

```
module energy_module
use constants_module, only: n, g, l, m
implicit none
private
public :: compute_energy

contains

! 计算系统能量
subroutine compute_energy(y, kinetic, potential, total)
    real(8), intent(in) :: y(n)
    real(8), intent(out) :: kinetic, potential, total

    real(8) :: theta, phi, theta_dot, phi_dot
    real(8) :: diff, cos_diff

    ! 提取变量
    theta = y(1)
    phi = y(2)
    theta_dot = y(3)
    phi_dot = y(4)

    diff = theta - phi
    cos_diff = cos(diff)

    ! 动能
    kinetic = 0.5d0 * m * l**2 * &
              (2.0d0 * theta_dot**2 + phi_dot**2 + &
               2.0d0 * theta_dot * phi_dot * cos_diff)

    ! 势能
    potential = -m * g * l * (2.0d0 * cos(theta) + cos(phi))

    ! 总能量
    total = kinetic + potential

end subroutine compute_energy

end module energy_module
```

```
module output_module
use constants_module, only: n, dt, t_end, n_steps
use energy_module, only: compute_energy
implicit none
private
public :: write_results

contains

! 写入结果文件
subroutine write_results(time_array, state_array, filename)
    real(8), intent(in) :: time_array(n_steps)
```

```fortran
        real(8), intent(in) :: state_array(n_steps, n)
        character(len=*), intent(in) :: filename

        integer :: i, unit
        real(8) :: kinetic, potential, total

        open(newunit=unit, file=filename, status='replace')

        ! 写入表头
        write(unit, '(A)') 'Time(s)    Theta(rad)    Phi(rad)    '// &
                           'Theta_dot(rad/s)    Phi_dot(rad/s)    '// &
                           'Kinetic(J)    Potential(J)    Total(J)'

        ! 写入数据
        do i = 1, n_steps
            ! 计算能量
            call compute_energy(state_array(i,:), kinetic, potential, total)

            ! 写入一行数据
            write(unit, '(8ES16.8)') &
                time_array(i), &
                state_array(i,1), &    ! theta
                state_array(i,2), &    ! phi
                state_array(i,3), &    ! theta_dot
                state_array(i,4), &    ! phi_dot
                kinetic, &
                potential, &
                total
        end do

        close(unit)

        print *, '结果已保存到文件: ', trim(filename)

    end subroutine write_results

    end module output_module
```

```fortran
    program double_pendulum_main
    use constants_module, only: n, dt, t_end, n_steps, y0
    use rk4_solver_module, only: rk4_step
    use output_module, only: write_results
    implicit none

    real(8) :: time_array(n_steps), state_array(n_steps, n)
    real(8) :: current_time, current_state(n)
    integer :: i

    ! 初始化
    current_time = 0.0d0
    current_state = y0
    time_array(1) = current_time
    state_array(1,:) = current_state
```

```fortran

    ! 时间推进
    do i = 1, n_steps-1
        ! 单步龙格库塔
        call rk4_step(current_time, current_state, current_state)

        ! 更新时间
        current_time = current_time + dt

        ! 存储结果
        time_array(i+1) = current_time
        state_array(i+1,:) = current_state

        ! 进度显示
        if (mod(i, 1000) == 0) then
            print '(A,F6.1,A,F6.1)', '进度:␣', current_time, '␣/␣', t_end
        end if
    end do

    print *, '模拟完成, 总步数:␣', n_steps

    ! 写入结果
    call write_results(time_array, state_array, 'double_pendulum_results.txt')

    end program double_pendulum_main
```

```makefile
# 编译器设置
FC = gfortran
FFLAGS = -O2 -Wall -Wextra
LDFLAGS =

# 目标文件
OBJS = constants_module.o rk4_solver_module.o energy_module.o output_module.o main.o

# 可执行文件
TARGET = double_pendulum

# 默认目标
all: $(TARGET)

# 链接可执行文件
$(TARGET): $(OBJS)
	$(FC) $(FFLAGS) -o $(TARGET) $(OBJS) $(LDFLAGS)

# 编译主程序
main.o: main.f90 constants_module.o rk4_solver_module.o output_module.o
	$(FC) $(FFLAGS) -c main.f90

# 编译输出模块
output_module.o: output_module.f90 constants_module.o energy_module.o
	$(FC) $(FFLAGS) -c output_module.f90

# 编译能量模块
```

```
energy_module.o: energy_module.f90 constants_module.o
	$(FC) $(FFLAGS) -c energy_module.f90

# 编译求解器模块
rk4_solver_module.o: rk4_solver_module.f90 constants_module.o
	$(FC) $(FFLAGS) -c rk4_solver_module.f90

# 编译常量模块
constants_module.o: constants_module.f90
	$(FC) $(FFLAGS) -c constants_module.f90

# 清理
clean:
	rm -f *.o *.mod $(TARGET)

# 运行
run: $(TARGET)
	./$(TARGET)

# 查看结果
check:
	@echo "前10行结果:"
	@head -n 11 double_pendulum_results.txt

.PHONY: all clean run check
```

```
% 读取双摆模拟结果文件
clear; clc;
cd 'C://Users/wudou/zhz/zhz/clt/zzh/';
% 设置文件名
filename = './data/double_pendulum_results.txt';
% 检查文件是否存在
if ~exist(filename, 'file')
    error('文件不存在:␣%s\n请确保已经运行了Fortran程序并生成了结果文件。', filename);
end
% 读取数据
fprintf('正在读取文件:␣%s\n', filename);
% 使用textscan读取文件（跳过第一行表头）
fid = fopen(filename, 'r');
if fid == -1
    error('无法打开文件:␣%s', filename);
end
% 读取表头
header = fgetl(fid);
% 使用textscan读取数据（科学计数法格式）
data = textscan(fid, '%f␣%f␣%f␣%f␣%f␣%f␣%f');
% 关闭文件
fclose(fid);
% 提取各列数据到工作区
time = data{1};                 % 时间 (s)
theta = data{2};                %  角 (rad)
phi = data{3};                  %  角 (rad)
theta_dot = data{4};            %  角速度 (rad/s)
```

```
phi_dot = data{5};          % 角速度 (rad/s)
kinetic = data{6};           % 动能 (J)
potential = data{7};         % 势能 (J)
total = data{8};             % 总能量 (J)
% 显示基本信息
fprintf('数据读取成功! \n');
fprintf('总数据点数:␣%d\n', length(time));
fprintf('时间范围:␣%.2f␣~␣%.2f␣s\n', time(1), time(end));
fprintf(' 角范围:␣%.4f␣~␣%.4f␣rad\n', min(theta), max(theta));
fprintf(' 角范围:␣%.4f␣~␣%.4f␣rad\n', min(phi), max(phi));
fprintf('总能量范围:␣%.6f␣~␣%.6f␣J\n', min(total), max(total));
% 计算能量相对误差 (检查能量守恒)
energy_error = (max(total) - min(total)) / mean(total) * 100;
fprintf('能量相对误差:␣%.4e␣%%\n', energy_error);
% 可选: 将数据保存为.mat文件以便后续使用
%save('double_pendulum_data.mat', 'time', 'theta', 'phi', 'theta_dot', 'phi_dot', ...'kinetic', 'potential', 'total', 'header');
%fprintf('数据已保存到工作区变量中, 并已保存为␣double_pendulum_data.mat␣文件。\n');
fprintf('可用变量:\n');
fprintf('␣␣time:␣时间序列\n');
fprintf('␣␣theta,␣phi:␣角度\n');
fprintf('␣␣theta_dot,␣phi_dot:␣角速度\n');
fprintf('␣␣kinetic,␣potential,␣total:␣能量\n');
fprintf('␣␣header:␣表头信息\n');
dir='./result/';
figure;
plot(time,theta,'LineWidth',1.5,'Color','b');
title('$$\theta\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$\theta$$','Interpreter','latex','FontSize',16);
legend('$$\theta$$','Interpreter','latex','Location','north');
fname=sprintf('theta-t');
print('-dpng',[dir '/' fname]);
close;
figure;
plot(time,phi,'LineWidth',1.5,'Color','b');
title('$$\varphi\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$\varphi$$','Interpreter','latex','FontSize',16);
legend('$$varphi$$','Interpreter','latex','Location','north');
fname=sprintf('phi-t');
print('-dpng',[dir '/' fname]);
close;
figure;
plot(time,phi_dot,'LineWidth',1.5,'Color','b');
title('$$\dot{\varphi}\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$\dot{\varphi}$$','Interpreter','latex','FontSize',16);
legend('$$\dot{\varphi}$$','Interpreter','latex','Location','north');
fname=sprintf('phi_dot-t');
print('-dpng',[dir '/' fname]);
close;
figure;
```

```
plot(time,phi_dot,'LineWidth',1.5,'Color','b');
title('$$\dot{\theta}\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$\dot{\theta}$$','Interpreter','latex','FontSize',16);
legend('$$\dot{\theta}$$','Interpreter','latex','Location','north');
fname=sprintf('theta_dot-t');
print('-dpng',[dir '/' fname]);
close;
figure;
plot(time,kinetic,'LineWidth',1.5,'Color','b');
title('$$T\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$T$$','Interpreter','latex','FontSize',16);
legend('$$T$$','Interpreter','latex','Location','north');
fname=sprintf('T-t');
print('-dpng',[dir '/' fname]);
close;
figure;
plot(time,potential,'LineWidth',1.5,'Color','b');
title('$$V\sim␣t$$','Interpreter','latex','FontSize',16);
xlabel('$$t$$','Interpreter','latex','FontSize',16);
ylabel('$$V$$','Interpreter','latex','FontSize',16);
legend('$$V$$','Interpreter','latex','Location','north');
fname=sprintf('V-t');
print('-dpng',[dir '/' fname]);
close;
figure;
plot(time,total,'LineWidth',5,'Color','b');
title('E');

close;

% 设置参数
l = 1.0;  % 摆长
skip_frames = 20;  % 跳过帧数 (加速动画)
% 提取部分数据用于动画
indices = 1:skip_frames:length(time);
time_anim = time(indices);
theta_anim = theta(indices);
phi_anim = phi(indices);

% 计算笛卡尔坐标
x1 = l * sin(theta_anim);
y1 = -l * cos(theta_anim);
x2 = x1 + l * sin(phi_anim);
y2 = y1 - l * cos(phi_anim);

% 创建图形窗口
figure('Position', [100, 100, 800, 800]);
axis equal;
xlim([-2.1*l, 2.1*l]);
ylim([-2.1*l, 2.1*l]);
grid on;
```

```
hold on;
title('双摆运动动画');
xlabel('x');
ylabel('y');
% 绘制固定点
plot(0, 0, 'ko', 'MarkerSize', 12, 'MarkerFaceColor', 'k');
% 初始化图形对象
pendulum_line1 = plot([0, x1(1)], [0, y1(1)], 'b-', 'LineWidth', 1);
pendulum_line2 = plot([x1(1), x2(1)], [y1(1), y2(1)], 'r-', 'LineWidth', 1);
mass1 = plot(x1(1), y1(1), 'bo', 'MarkerSize', 12, 'MarkerFaceColor', 'b');
mass2 = plot(x2(1), y2(1), 'ro', 'MarkerSize', 12, 'MarkerFaceColor', 'r');
trajectory = plot(x2(1), y2(1), 'g-', 'LineWidth', 1.5, 'Color', [0, 0.7, 0, 0.5]);
% 时间显示文本
time_text = text(-2*l, -0.5*l, sprintf('时间:␣%.2f␣s', time_anim(1)), ...
    'FontSize', 14, 'FontWeight', 'bold');
% 轨迹数据存储
trajectory_x = [];
trajectory_y = [];
% 生成动画
fprintf('生成动画...\n');
for i = 1:length(time_anim)
    % 更新摆的位置
    set(pendulum_line1, 'XData', [0, x1(i)], 'YData', [0, y1(i)]);
    set(pendulum_line2, 'XData', [x1(i), x2(i)], 'YData', [y1(i), y2(i)]);
    set(mass1, 'XData', x1(i), 'YData', y1(i));
    set(mass2, 'XData', x2(i), 'YData', y2(i));

    % 更新轨迹
    trajectory_x = [trajectory_x, x2(i)];
    trajectory_y = [trajectory_y, y2(i)];
    set(trajectory, 'XData', trajectory_x, 'YData', trajectory_y);

    % 更新时间显示
    set(time_text, 'String', sprintf('时间:␣%.2f␣s', time_anim(i)));

    % 刷新图形
    drawnow;

    % 暂停控制动画速度
    pause(0.01);
end
fprintf('动画完成\n');
```

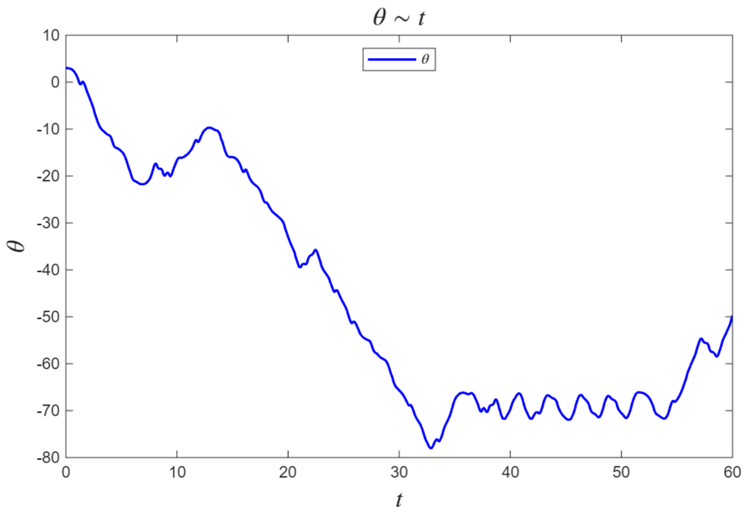

图 1: theta

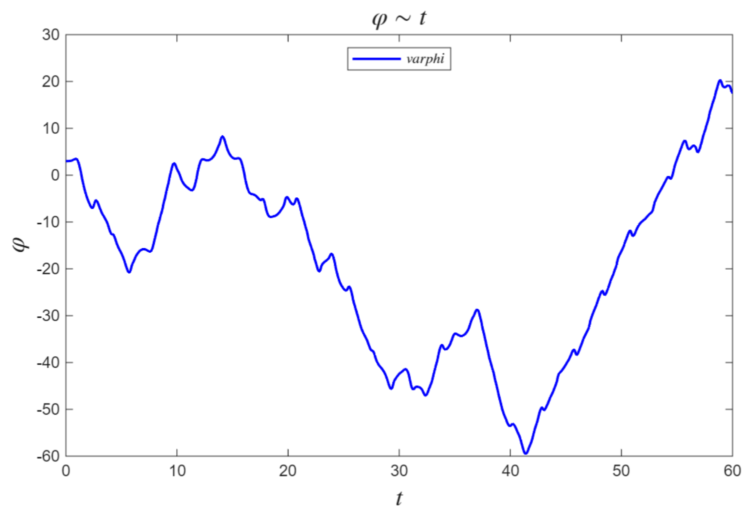

图 2: phi

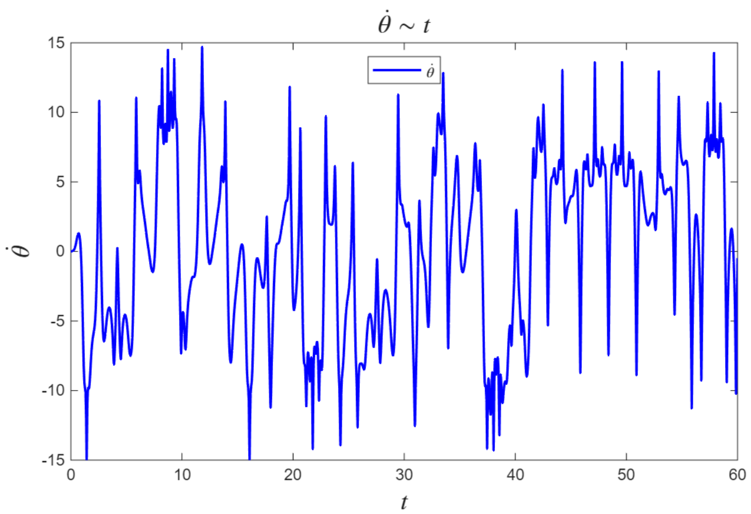

图 3: thetadot

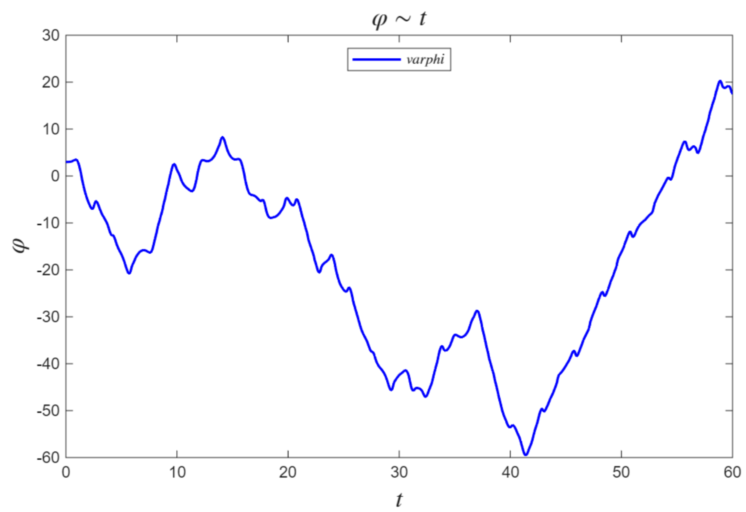

图 4: phi

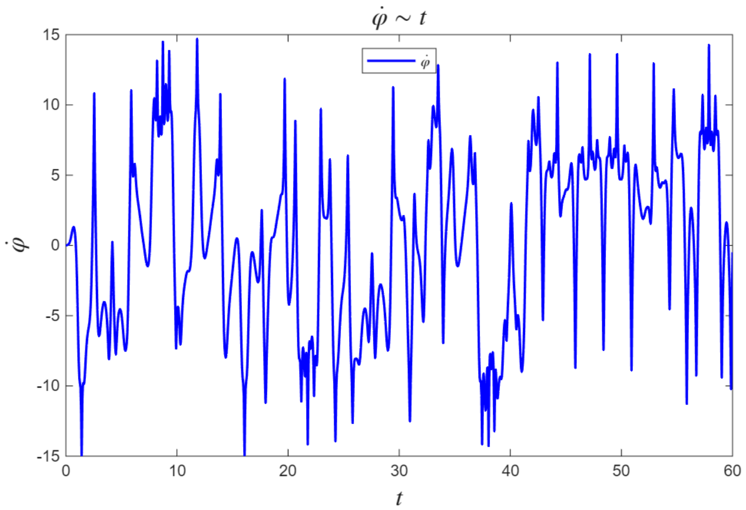

图 5: phidot

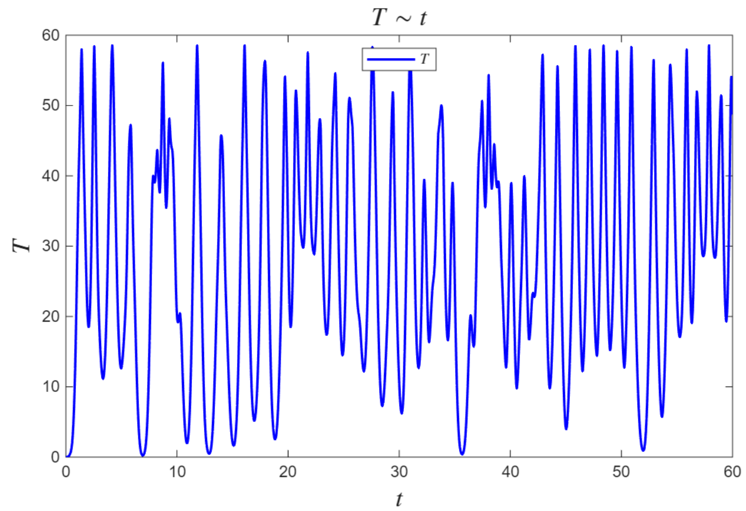

图 6: T

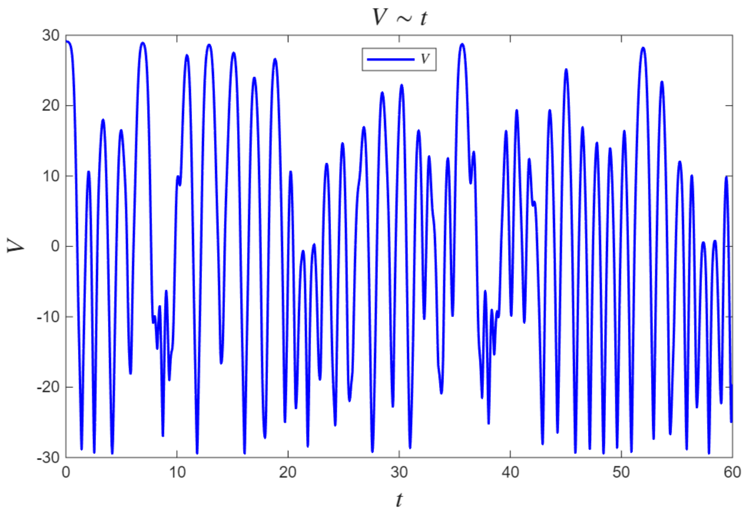

图 7: V

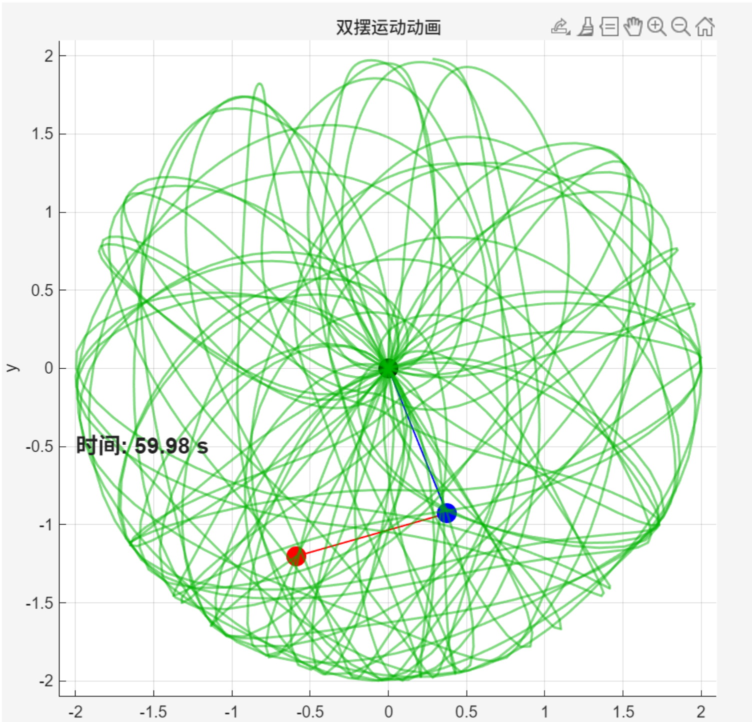

图 8: animation