

# AE 370: HW5

Linyi Hou

March 6, 2020

## Problem 1

a. Integrating the ODE:

$$\int_{t_k}^{t_{k+1}} \dot{u} dt = \int_{t_k}^{t_{k+1}} f(u(t), t) dt \quad (1)$$

Substituting in the Lagrange basis for trapezoidal method:

$$u(t_{k+1}) - u(t_k) \approx \int_{t_k}^{t_{k+1}} f(u(t_k), t_k) \left( \frac{t - t_{k+1}}{-\Delta t} \right) + f(u(t_{k+1}), t_{k+1}) \left( \frac{t - t_k}{\Delta t} \right) dt \quad (2)$$

Extracting constants:

$$u(t_{k+1}) - u(t_k) \approx \frac{f(u(t_k), t_k)}{-\Delta t} \int_{t_k}^{t_{k+1}} (t - t_{k+1}) dt + \frac{f(u(t_{k+1}), t_{k+1})}{\Delta t} \int_{t_k}^{t_{k+1}} (t - t_k) dt \quad (3)$$

$$= \frac{\Delta t}{2} (f(u(t_k), t_k) + f(u(t_{k+1}), t_{k+1})) \quad (4)$$

Since  $u(t_k)$  is generally not known, we substitute all  $u(t_k)$  with  $u_k$  to represent the approximate solution to the IVP at time  $t_k$ . Thus Equation (4) becomes:

$$u_{k+1} - u_k = \frac{\Delta t}{2} (f(u_k, t_k) + f(u_{k+1}, t_{k+1})) \quad (5)$$

To find the truncation error, we subtract the left side of Equation (3) from the right side of Equation (4) and divide by  $\Delta t$ :

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} (f(u(t_k), t_k) + f(u(t_{k+1}), t_{k+1})) \quad (6)$$

Retrieving the ODE from Equation (1):

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} (\dot{u}(t_k) + \dot{u}(t_{k+1})) \quad (7)$$

We now Taylor expand  $u$  and  $\dot{u}$  at  $t_{k+1}$ :

$$u(t_{k+1}) = u(t_k) + \Delta t \dot{u}(t_k) + \frac{\Delta t^2}{2} \ddot{u}(t_k) + \frac{\Delta t^3}{6} \dddot{u}(t_k) + \text{H.O.T.} \quad (8)$$

$$\dot{u}(t_{k+1}) = \dot{u}(t_k) + \Delta t \ddot{u}(t_k) + \frac{\Delta t^2}{2} \dddot{u}(t_k) + \text{H.O.T.} \quad (9)$$

Substituting Equations (8) and (9) into Equation (7), we find:

$$\tau_k = -\frac{\Delta t^2}{6}\ddot{u}(t_k) + \text{H.O.T.} \quad (10)$$

$$\boxed{\tau_k = O(\Delta t^2)} \quad (11)$$

b. The two-step Adams-Bashforth method has form

$$u_{k+1} = u_k + \frac{\Delta t}{2} [-f(u_{k-1}, t_{k-1}) + 3f(u_k, t_k)] \quad (12)$$

Rewriting the equation by substituting approx. values  $u_k$  with true values  $u(t_k)$ :

$$u(t_{k+1}) = u(t_k) + \frac{\Delta t}{2} [-f(u(t_{k-1}), t_{k-1}) + 3f(u(t_k), t_k)] \quad (13)$$

Rearranging and dividing by  $\Delta t$  to find the truncation error:

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} [-f(u(t_{k-1}), t_{k-1}) + 3f(u(t_k), t_k)] \quad (14)$$

Retrieving the ODE from Equation (1):

$$\boxed{\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} [-\dot{u}(t_{k-1}) + 3\dot{u}(t_k)]} \quad (15)$$

We now Taylor expand  $u$  and  $\dot{u}$  at  $t_{k+1}$  and  $t_{k-1}$  respectively:

$$u(t_{k+1}) = u(t_k) + \Delta t \dot{u}(t_k) + \frac{\Delta t^2}{2} \ddot{u}(t_k) + \frac{\Delta t^3}{6} \dddot{u}(t_k) + \text{H.O.T.} \quad (16)$$

$$\dot{u}(t_{k-1}) = \dot{u}(t_k) - \Delta t \ddot{u}(t_k) + \frac{\Delta t^2}{2} \dddot{u}(t_k) + \text{H.O.T.} \quad (17)$$

Substituting Equations (16) and (17) into Equation (15):

$$\tau_k = \frac{u(t_k) + \Delta t \dot{u}(t_k) + \frac{\Delta t^2}{2} \ddot{u}(t_k) + \frac{\Delta t^3}{6} \dddot{u}(t_k) + \text{H.O.T.} - u(t_k)}{\Delta t} - \frac{1}{2} \left[ -\left( \dot{u}(t_k) - \Delta t \ddot{u}(t_k) + \frac{\Delta t^2}{2} \dddot{u}(t_k) + \text{H.O.T.} \right) + 3\dot{u}(t_k) \right] \quad (18)$$

Collecting terms:

$$\tau_k = \frac{5}{12} \Delta t^2 \ddot{u}(t_k) + \text{H.O.T.} \quad (19)$$

$$\boxed{\tau_k = O(\Delta t^2)} \quad (20)$$

## Problem 2

- a. See figures and code attached in Appendix A and Appendix B, respectively.
- b. See code attached in Appendix B.

First, for each method, we manually adjust the number of time steps taken between  $t = 0$  and  $t = 50$  until the minimum number of time steps (conversely, the maximum  $dt$ ) was found that satisfies the convergence requirement of  $6 \times 10^{-5}$ .

Then, using the tic and toc functions, we propagate the approximations to determine the average run time. The results are as follows:

Adams-Bashforth 2-Step: 0.029 seconds | 43233 steps  
 Heun's Method: 0.015 seconds | 17362 steps  
 Four-Stage Runge-Kutta: 0.0053 seconds | 1711 steps

Both Heun's method (two-stage Runge-Kutta) and the four-stage Runge-Kutta method are variants of the trapezoid method. Both methods show quicker convergence times and fewer steps required as compared to the two-step Adams-Bashforth method, which suggests that the trapezoid method is better suited for solving the predator-prey problem.

- c. For a two-step Adams-Moulton method:

$$u_{k+1} = u_k + \frac{\Delta t}{12} \left[ -f(u_{k-1}, t_{k-1}) + 8f(u_k, t_k) + 5f(u_{k+1}, t_{k+1}) \right] \quad (21)$$

The only unknown in this equation is  $u_{k+1}$ . Since for the predator-prey problem  $f(u_k, t_k) = f(u_k)$ , Equation (21) becomes:

$$u_{k+1} = u_k + \frac{\Delta t}{12} \left[ -f(u_{k-1}) + 8f(u_k) + 5f(u_{k+1}) \right] \quad (22)$$

This equation would be solved in MATLAB using syms, where we set up two variables  $x, y$  such that  $u_{k+1} = [x, y]^T$ . Thus Equation (22) becomes a system of two equations with two variables, which can be easily solved.

# Appendix A: Figures

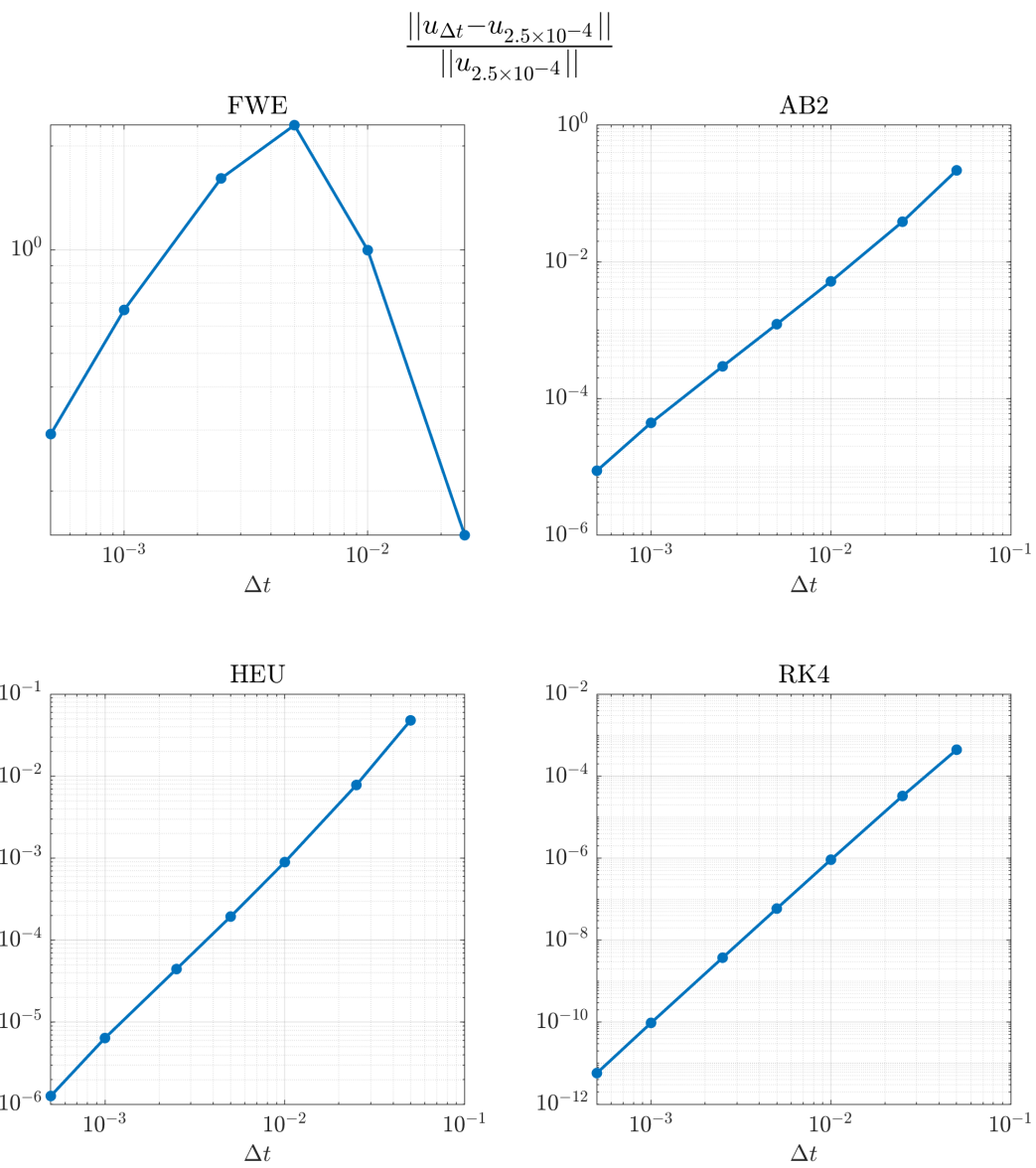


Figure 1: Error Plot for Problem 2a

## Appendix B: Code

```
1 %% Problem 2 Part a
2 close all
3 clear;clc
4
5 %—problem parameters
6
7     b = 2; %birth rate of prey
8     p = 1; %effect of predation on prey
9     d = 3; %death rate of predators
10    g = 1; %growth of predators due to eating prey
11
12    t0 = 0; %starting time
13    T = 50; %final time
14
15    u0 = [1;1]; %IC
16
17    f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS
18
19 %—
20
21 %part a)
22     %—simulation params
23     dtvect = [5e-2, 2.5e-2, 1e-2, 5e-3, 2.5e-3, 1e-3, 5e-4, 2.5e-4];
24     %—
25
26     %initialize vector that stores approximate soln at T for various dt
27     u_FWE_keep = zeros( 2,length( dtvect ) ); % forward Euler
28     u_AB2_keep = zeros( 2,length( dtvect ) ); % Adams-Bashforth 2-step
29     u_HEU_keep = zeros( 2,length( dtvect ) ); % Heun's
30     u_RK4_keep = zeros( 2,length( dtvect ) ); % Runge-Kutta 4
31
32
33     %advance in time
34     for j = 1 : length(dtvect)
35
36         %current dt
37         dt = dtvect(j);
38
```

```

39     tk = t0; %initialize time iterate
40
41     %initialize iterates for various methods
42     % FWE
43     u_FWE_k = u0;
44     % AB2
45     u_AB2_k = u0;
46     u_AB2_km1 = u0;
47     % HEU
48     u_HEU_k = u0;
49     % RK4
50     u_RK4_k = u0;
51
52
53     %run to final time T
54     for jj = 1 : T/dt
55
56         % ===== FWE =====
57         % propagate and update
58         u_FWE_k = u_FWE_k + f(u_FWE_k) * dt;
59
60         % ===== AB2 =====
61         % propagate
62         if jj < 2
63             %advance with Heun's for 1st time step
64             u_AB2_kp1 = u_AB2_k + ...
65                 0.5 * dt * ...
66                 ( f(u_AB2_k) + f(u_AB2_k + dt * f(u_AB2_k)) )
67             ;
68         else
69             u_AB2_kp1 = u_AB2_k + dt / 2 * ...
70                 ( -f(u_AB2_km1) + 3 * f(u_AB2_k) );
71         end
72         % update iterates
73         u_AB2_km1 = u_AB2_k;
74         u_AB2_k = u_AB2_kp1;
75
76         % ===== HEU =====
77         % propagate and update
78         u_HEU_k = u_HEU_k + ...

```

```

78             0.5 * dt * ...
79             ( f(u_HEU_k) + f(u_HEU_k + dt * f(u_HEU_k)) );
80
81         % ===== RK4 =====
82         % propagate and update
83         y1 = f(u_RK4_k);
84         y2 = f(u_RK4_k + dt/2 * y1);
85         y3 = f(u_RK4_k + dt/2 * y2);
86         y4 = f(u_RK4_k + dt * y3);
87         u_RK4_k = u_RK4_k + 1/6 * dt * (y1 + 2*y2 + 2*y3 + y4);
88
89
90         % ===== update time =====
91
92         tk = tk + dt;
93
94     end
95
96     % store solution at T
97     % FWE
98     u_FWE_keep(:,j) = u_FWE_k;
99     % AB2
100    u_AB2_keep(:,j) = u_AB2_kp1;
101    % HEU
102    u_HEU_keep(:,j) = u_HEU_k;
103    % RK4
104    u_RK4_keep(:,j) = u_RK4_k;
105
106 end
107
108 %compute difference between solution at smallest dt and the other dts
109
110 %initialize vector
111 u_FWE_diff = zeros( length( dtvect )-1,1 );
112 u_AB2_diff = zeros( length( dtvect )-1,1 );
113 u_HEU_diff = zeros( length( dtvect )-1,1 );
114 u_RK4_diff = zeros( length( dtvect )-1,1 );
115
116 for j = 1 : length( dtvect )-1
117

```

```

118         u_FWE_diff(j) = norm(u_FWE_keep(:,j) - u_FWE_keep(:,end)) ...
119                             / norm(u_FWE_keep(:,end));
120         u_AB2_diff(j) = norm(u_AB2_keep(:,j) - u_AB2_keep(:,end)) ...
121                             / norm(u_AB2_keep(:,end));
122         u_HEU_diff(j) = norm(u_HEU_keep(:,j) - u_HEU_keep(:,end)) ...
123                             / norm(u_HEU_keep(:,end));
124         u_RK4_diff(j) = norm(u_RK4_keep(:,j) - u_RK4_keep(:,end)) ...
125                             / norm(u_RK4_keep(:,end));
126
127     end
128
129     f = figure(1);
130
131     subplot(2,2,1)
132     loglog( dtvect(1:end-1), u_FWE_diff, '.-', 'markersize', 25, '
        linewidth', 2 )
133     title('FWE', 'fontsize', 16, 'interpreter' , 'latex')
134     set( gca, 'Color', [1 1 1] )
135     set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
136     xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
137     grid(gca,'minor')
138     grid on
139
140     subplot(2,2,2)
141     loglog( dtvect(1:end-1), u_AB2_diff, '.-', 'markersize', 25, '
        linewidth', 2 )
142     title('AB2', 'fontsize', 16, 'interpreter' , 'latex')
143     set( gca, 'Color', [1 1 1] )
144     set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
145     xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
146     grid(gca,'minor')
147     grid on
148
149     subplot(2,2,3)
150     loglog( dtvect(1:end-1), u_HEU_diff, '.-', 'markersize', 25, '
        linewidth', 2 )
151     title('HEU', 'fontsize', 16, 'interpreter' , 'latex')
152     set( gca, 'Color', [1 1 1] )
153     set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
154     xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')

```



```

155     grid(gca,'minor')
156     grid on
157
158     subplot(2,2,4)
159     loglog( dtvect(1:end-1), u_RK4_diff, '.-', 'markersize', 25, '
        linewidth', 2 )
160     title('RK4', 'fontsize', 16, 'interpreter' , 'latex')
161     set( gca, 'Color', [1 1 1] )
162     set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
163     xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
164     grid(gca,'minor')
165     grid on
166
167
168     sgtitle('$\frac{||u_{\Delta t} - u_{2.5 \times 10^{-4}}||}{||u_{2.5 \times 10^{-4}}||}$', 'fontsize', 28, 'interpreter' , 'latex' )
169     set(f, 'PaperPositionMode', 'manual')
170     set(f, 'Color', [1 1 1])
171     set(f, 'PaperUnits', 'centimeters')
172     set(f, 'PaperSize', [30 30])
173     set(f, 'Units', 'centimeters' )
174     set(f, 'Position', [0 0 30 30])
175     set(f, 'PaperPosition', [0 0 30 30])
176
177     svnm = 'error_q2';
178     print( '-dpng', svnm, '-r200' )
179
180     % store the reference values for all four methods in a .mat file
181     u_FWE_ref = u_FWE_keep(:,end);
182     u_AB2_ref = u_AB2_keep(:,end);
183     u_HEU_ref = u_HEU_keep(:,end);
184     u_RK4_ref = u_RK4_keep(:,end);
185
186     save('ref_data.mat', 'u_FWE_ref', 'u_AB2_ref', 'u_HEU_ref', 'u_RK4_ref');
187
188     %% Problem 2 Part b, AB2
189     close all
190     clear;clc
191
192     % manually edit this variable for convergence

```

```

193 num_steps = 43233;
194
195 % simulation params
196 b = 2; %birth rate of prey
197 p = 1; %effect of predation on prey
198 d = 3; %death rate of predators
199 g = 1; %growth of predators due to eating prey
200 u0 = [1;1]; %IC
201 f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS
202
203 tk = 0; % starting time
204 T = 50; % final time
205 dt = (T-tk)/num_steps; % time step
206
207 % load reference value
208 load('ref_data.mat','u_AB2_ref');
209
210 % ===== BEGIN TIMER =====
211 tic
212
213 %initialize iterates for AB2
214 u_AB2_k = u0;
215 u_AB2_km1 = u0;
216
217 %advance to final time T
218 for jj = 1 : T/dt
219
220     % ===== AB2 =====
221     % propagate
222     if jj < 2
223         %advance with Heun's for 1st time step
224         u_AB2_kp1 = u_AB2_k + ...
225             0.5 * dt * ...
226             ( f(u_AB2_k) + f(u_AB2_k + dt * f(u_AB2_k)) );
227     else
228         u_AB2_kp1 = u_AB2_k + dt / 2 * ...
229             ( -f(u_AB2_km1) + 3 * f(u_AB2_k) );
230     end
231     % update iterates
232     u_AB2_km1 = u_AB2_k;

```

```

233     u_AB2_k = u_AB2_kp1;
234     tk = tk + dt;
235
236 end
237
238 % ===== END TIMER =====
239 t_AB2 = toc;
240
241 % manually check for convergence
242 u_AB2_keep = u_AB2_kp1;
243 u_AB2_diff = norm(u_AB2_keep - u_AB2_ref) / norm(u_AB2_ref);
244 disp(['Error: ' num2str(u_AB2_diff)])
245 if u_AB2_diff <= 6e-5
246     disp('Successfully converged!');
247 else
248     disp('Failed to converge. ');
249 end
250 disp(['Time to converge: ' num2str(t_AB2) ' seconds'])
251
252
253 %% Problem 2 Part b, HEU
254 close all
255 clear;clc
256
257 % manually edit this variable for convergence
258 num_steps = 17362;
259
260 % simulation params
261 b = 2; %birth rate of prey
262 p = 1; %effect of predation on prey
263 d = 3; %death rate of predators
264 g = 1; %growth of predators due to eating prey
265 u0 = [1;1]; %IC
266 f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS
267
268 tk = 0; % starting time
269 T = 50; % final time
270 dt = (T-tk)/num_steps; % time step
271
272 % load reference value

```

```

273 load('ref_data.mat','u_HEU_ref');
274
275 % ===== BEGIN TIMER =====
276 tic
277
278 %initialize iterates for HEU
279 u_HEU_k = u0;
280
281 %advance to final time T
282 for jj = 1 : T/dt
283
284     % ===== HEU =====
285     % propagate and update
286     u_HEU_k = u_HEU_k + ...
287         0.5 * dt * ...
288         ( f(u_HEU_k) + f(u_HEU_k + dt * f(u_HEU_k)) );
289     tk = tk + dt;
290
291 end
292
293 % ===== END TIMER =====
294 t_HEU = toc;
295
296 % manually check for convergence
297 u_HEU_keep = u_HEU_k;
298 u_HEU_diff = norm(u_HEU_keep - u_HEU_ref) / norm(u_HEU_ref);
299 disp(['Error: ' num2str(u_HEU_diff)])
300 if u_HEU_diff <= 6e-5
301     disp('Successfully converged!');
302 else
303     disp('Failed to converge. ');
304 end
305 disp(['Time to converge: ' num2str(t_HEU) ' seconds'])
306
307
308 %% Problem 2 Part b, RK4
309 close all
310 clear;clc
311
312 % maunally edit this variable for convergence

```

```

313 num_steps = 1711;
314
315 % simulation params
316 b = 2; %birth rate of prey
317 p = 1; %effect of predation on prey
318 d = 3; %death rate of predators
319 g = 1; %growth of predators due to eating prey
320 u0 = [1;1]; %IC
321 f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS
322
323 tk = 0; % starting time
324 T = 50; % final time
325 dt = (T-tk)/num_steps; % time step
326
327 % load reference value
328 load('ref_data.mat','u_RK4_ref');
329
330 % ===== BEGIN TIMER =====
331 tic
332
333 %initialize iterates for RK4
334 u_RK4_k = u0;
335
336 %advance to final time T
337 for jj = 1 : T/dt
338
339     % ===== RK4 =====
340     % propagate and update
341     y1 = f(u_RK4_k);
342     y2 = f(u_RK4_k + dt/2 * y1);
343     y3 = f(u_RK4_k + dt/2 * y2);
344     y4 = f(u_RK4_k + dt * y3);
345     u_RK4_k = u_RK4_k + 1/6 * dt * (y1 + 2*y2 + 2*y3 + y4);
346     tk = tk + dt;
347
348 end
349
350 % ===== END TIMER =====
351 t_RK4 = toc;
352

```

```
353 % manually check for convergence
354 u_RK4_keep = u_RK4_k;
355 u_RK4_diff = norm(u_RK4_keep - u_RK4_ref) / norm(u_RK4_ref);
356 disp(['Error: ' num2str(u_RK4_diff)])
357 if u_RK4_diff <= 6e-5
358     disp('Successfully converged!');
359 else
360     disp('Failed to converge. ');
361 end
362 disp(['Time to converge: ' num2str(t_RK4) ' seconds'])
```