Themes:

Initial value problems for ODE, Extrapolation, Sensitivity (as usual).

- Need to show one example with hand calculation for a system with Euler's method, several steps, stepsize halving, and extrapolation : because this is often an exam question.
- Show matlab calls to ode23 & odeset to change tolerances, show how to write dydt = diffeqn(t,y)

1, first, such an example, I give one below.

Hand calculation

```
Mx'' + Kx + D x' = t, x(0) = 1, x'(0) = 0

u = (u1,u2) = (x,x')
u1' = x' = u2;   u1(0) = x(0) = 1;
u2' = x'' = (t –Kx – Dx')/M =
= (t –K*u(1) – D*u(2))/M; u2(0) = x'(0) = 0

Matlab
function dudt = osci(t,u)
K = 1; M = 1; D = 0.2;  % or global K M D
x = u(1); xdot = u(2)
dudt = [xdot; (t-K*x-D*xdot)/M];

or
dudt = [u(2); (t-K*u(1)-D*u(2)/M];
or
dudt = [0 1; -K/M –D/M]*u + [0;t/M];

[tout,uout]=ode23('osci',tspan,u0)

handcompute
dt = 0.4
t = 0    u' (=x',x'')                      u + dt*u'
x   1    0                                 1
x'  0   (0 -1*1 -0.2*0)/1 = -1             0-0.4*1 = -0.4
```

t = 0.4
x   1
x' -0.4

```
============= ========
```

```
dt = 0.2
t = 0    u’ (=x’,x’’)                        u + dt*u’
x   1    0                                   1
x’  0    (0 -1*1 -0.2*0)/1 = -1              0 -1*0.2 = -0.2

t = 0.2  u’ (=x’,x’’)                        u + dt*u’
x   1    -0.2                                1-0.2*0.2 = 0.96
x’  -0.2 (0.2-1*1 -0.2*-0.2)/1 = -0.76       -0.2-0.76*0.2 = -0.352
```

t = 0.4
x 0.96
x’ -0.352

Extrapol

```
t  = 0.4:
            x                        x’
h = 0.4    1                        -0.4
diff             -0.04                      0.048
h = 0.2    0.96          0.92       -0.352        -0.304
```

2, Then use extrapolation on the example in NAM p 96
(numbers below) first show that
if $Y(h) – Y = C h^p$,
then
$(Y(4h)-Y(2h))/(Y(2h)-Y(h)) = … = 2^p$

```
h             y      diff  ratio  extrap
0.1           9379
0.05          9007  -372  2.0    8635
0.025         8823  -184  2.0    8639
0.0125        8731   -92         8639
Excellent improvement (better than expected, maybe)
```

3, Then EXS 7.4
First discuss issue of critical points **y*** where **f**(**y***) = 0. Compute critical point. What type approach to the limit ? In this case exponential, one can estimate the time constant from linearization. Do that if you have time. Write the matlab function & plot (see below)
Then set Qin =0 for all times and run matlab code. What happens? (when h becomes 0)
Note that this is ***not*** exponential approach to the limit, rather the process finishes in finite time. Compute analytically when h = 0 as function of h0. The matlab uses h0 = 3 explain why the code takes such small steps when h should just stay = 0! Mention that the RHS is NOT Lipschitz in h at h = 0 when Qin = 0

```
% run chlepsydra
clear all
close all
figure(1)
```

```
clf
hold on
global A C g H0 F
A = 0.8; g = 9.81; C = 0.1; H0 = 20;
F = 1
F = 0 %-- turns off Qin
hlist = [0.0001,3,10,25]
hlist = 3
opts = odeset('reltol',1e-5,'abstol',1e-5);
for h = hlist
[tout,hout]=ode23('chlepsydra',[0 20],h,opts);
plot(tout,hout(:,1),'k','linewidth',3)
end
xlabel('time')
ylabel('water height')
figure(2)
clf
plot(tout(1:end-1),log10(diff(tout)),'.k')
```

```
function dhdt = chlepsydra(t,h)
global A C g H0 F
Qin = F*max(0,1-h/H0);
Qout = C*sqrt(2*g*h);
dhdt = (Qin-Qout)/A;
then
```

3, EXS 7.8

Do conversion to 1st order system & hand calculation. Numbers can be taken from solution in EXS.

matlab: also with sensitivity to qdot(0)

```
% run genius
clear all
close all
figure(1)
clf
q0    = 100;
qdot0 = 0;
[tout,uout]=ode23('genius',[0 100],[q0;qdot0]);
plot(tout,uout(:,1),'.k',tout,uout(:,2),'.r')
q0100 = uout(end,1)
legend('q','qdot')
qdot0 = 0.01
[tout,uout]=ode23('genius',[0 100],[q0;qdot0]);
q1100 = uout(end,1)
dq100dqdot0 = (q1100-q0100)/qdot0
```

```
function dudt = genius(t,u)
q = u(1); qdot = u(2);
dudt = [qdot;100/q^2];
```

Do part c). Explain …

1) the integration multiplied the eqn by qdot so introduces a “false root” q = constant
2) again, the rhs is NOT Lipschitz in q at q = q0 if qdot(0) = 0.
   But with qdot(0) = eps the limit as eps -> 0 exists. You can show that by computation, plot new solution in same window for eps = 0.1, 0.01, 0.001 (note the sensitivity is large to qdot(0)

I think that is enough for two hours.