CHAP 5 – REPEATS (LOOPS) & VECTORIZE CODE

↖ MATLAB CAN "AUTOMATE" SOME OPERATIONS ON ARRAYS WHICH NEED REPEATS IN OTHER LANGUAGES.

TWO TYPES

① **FOR**

- USE WHEN KNOW HOW MANY REPEATS, e.g. SUM BANK CLIENT BALANCES
- LOOP VAR. OFTEN USED AS ARRAY SUBSCRIPT

KNOW HOW MANY CLIENTS → `client(i)`

② **WHILE**

- USE WHEN DON'T KNOW HOW MANY REPEATS
- CAN USE WHILE WHEN DO KNOW HOW MANY REPEATS
- CAN INCREMENT A COUNTER VAR (LOOP VAR) INSIDE REPEAT

REPEAT THESE INSTRUCTIONS WHILE LOGICAL EXPRESSION IS TRUE

WHILE REPEAT

```
i = 1 % INITIALIZE
while i < 11   % LOGICAL EXPRESSION
    fprintf('i = %d \n', i)
    i = i + 1; % INCREMENT
end
```

FOR REPEAT

"LOOP VARIABLE"

```
for i = 1:10
    fprintf('i = %d \n', i)
end
```

YOU CAN USE LOOP VARIABLE IN INSTRUCTIONS – OR NOT

e.g. `disp('hello')`

PSEUDO-CODE

% CHECK IF TURKEY IS DONE
i = 1
WHILE T < 350

CHECK TURKEY TEMP (TT)
T(i) = TT
i = i + 1

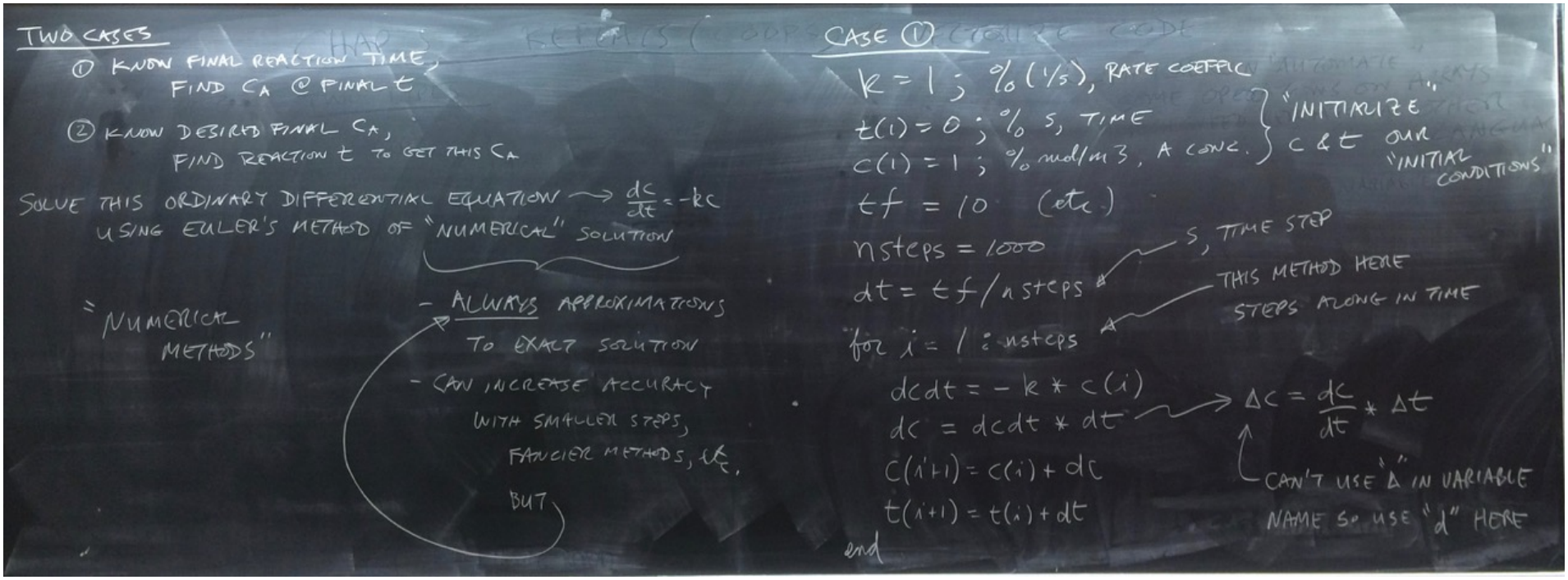
end.

OPTIONAL IF WANT TO KEEP TURKEY T HISTORY

Missed a blackboard. We consider chemical reaction A > B in a well-mixed batch reactor, where equilibrium is almost all B, and T and V are constant. Balance on A in reactor gives

$$dc/dt = - kc$$

where c is concentration of A, k is rate coefficient



An analytical solution is possible. Separate variables to get

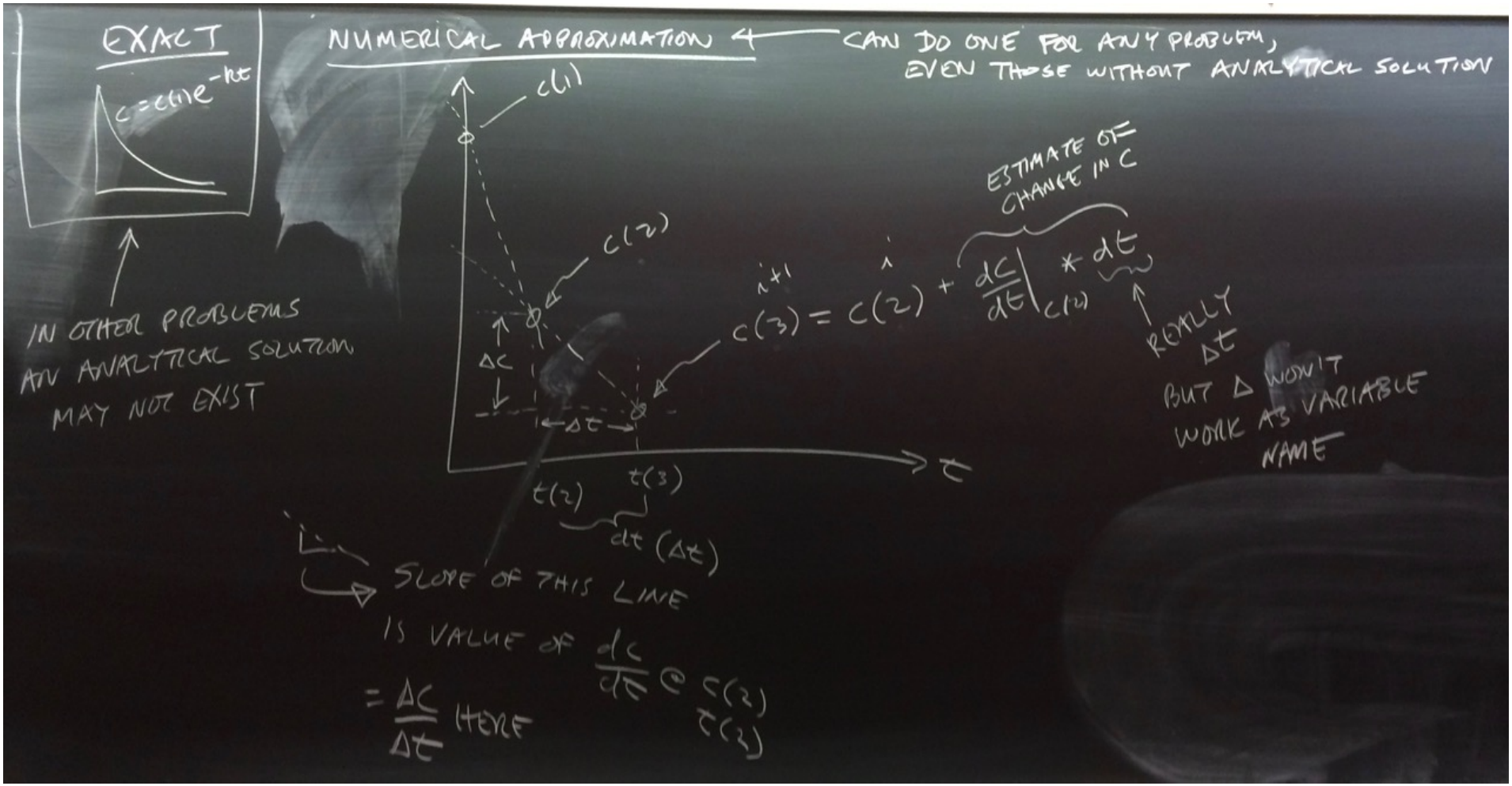
$$dc/c = -kdt.$$

Integrate from initial conditions of known value of c(0) at t=0 to get

$$\ln(c(t)/c(0)) = -kt$$ and exponentiate to get $c(t) = c(0)*\exp(-kt)$

Numerical methods can give approximate solution to this problem, as well as to problems for which an analytical solution does not exist or is difficult to obtain.

Euler's method is an approximation because it assumes that the value of dc/dt (slope, rate of change) is constant between time steps, whereas the value of dc/dt is always changing.



The accuracy of Euler's method can be improved by taking smaller steps in time (Δt, dt) at the cost of longer computation time.

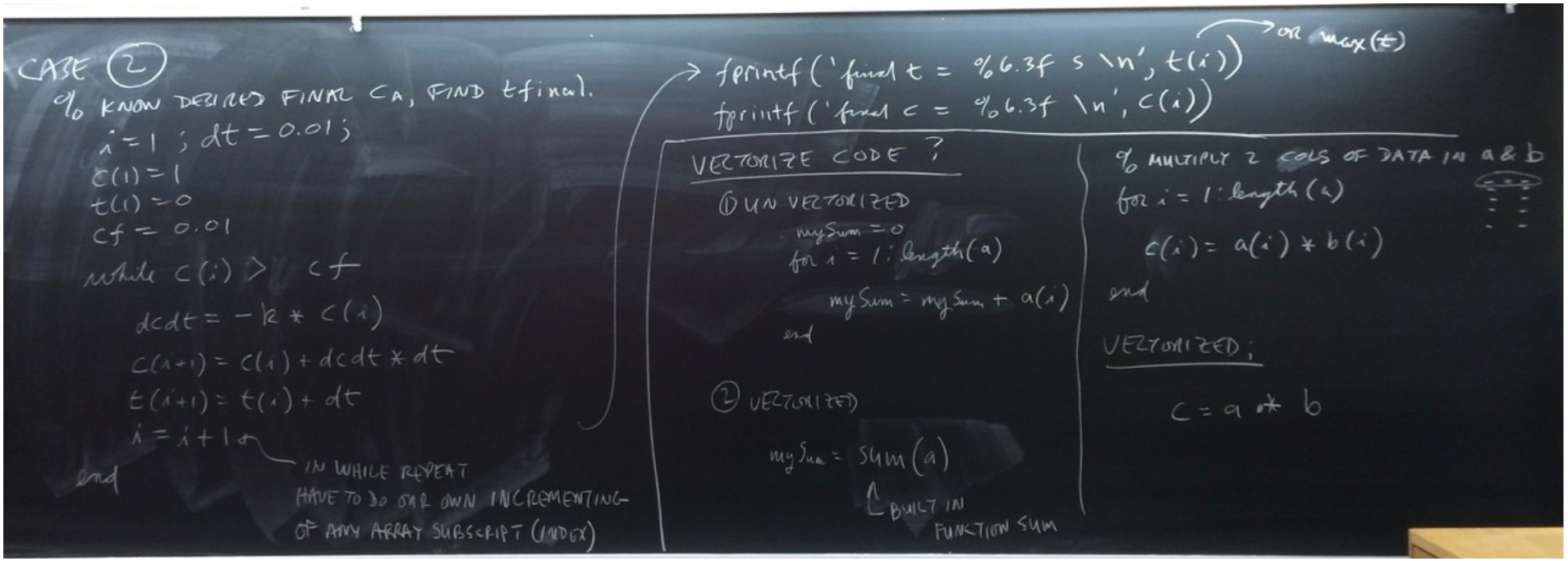

CASE (2)
% KNOW DESIRED FINAL cA, FIND tfinal.
i = 1 ; dt = 0.01;
c(1) = 1
t(1) = 0
cf = 0.01
while c(i) > cf
dcdt = -k * c(i)
c(i+1) = c(i) + dcdt * dt
t(i+1) = t(i) + dt
i = i + 1
end
IN WHILE REPEAT HAVE TO DO OUR OWN INCREMENTING OF ANY ARRAY SUBSCRIPT (INDEX)
fprintf('final t = %6.3f s \n', t(i))
OR max(t)
fprintf('final c = %6.3f \n', c(i))
VECTORIZE CODE ?
① UN VECTORIZED
mySum = 0
for i = 1 : length(a)
mySum = mySum + a(i)
end
② VECTORIZED
mySum = sum(a)
BUILT IN FUNCTION SUM
% MULTIPLY 2 COLS OF DATA IN a & b
for i = 1 : length(a)
c(i) = a(i) * b(i)
end
VECTORIZED:
c = a .* b

Table of Contents

for & while repeat examples

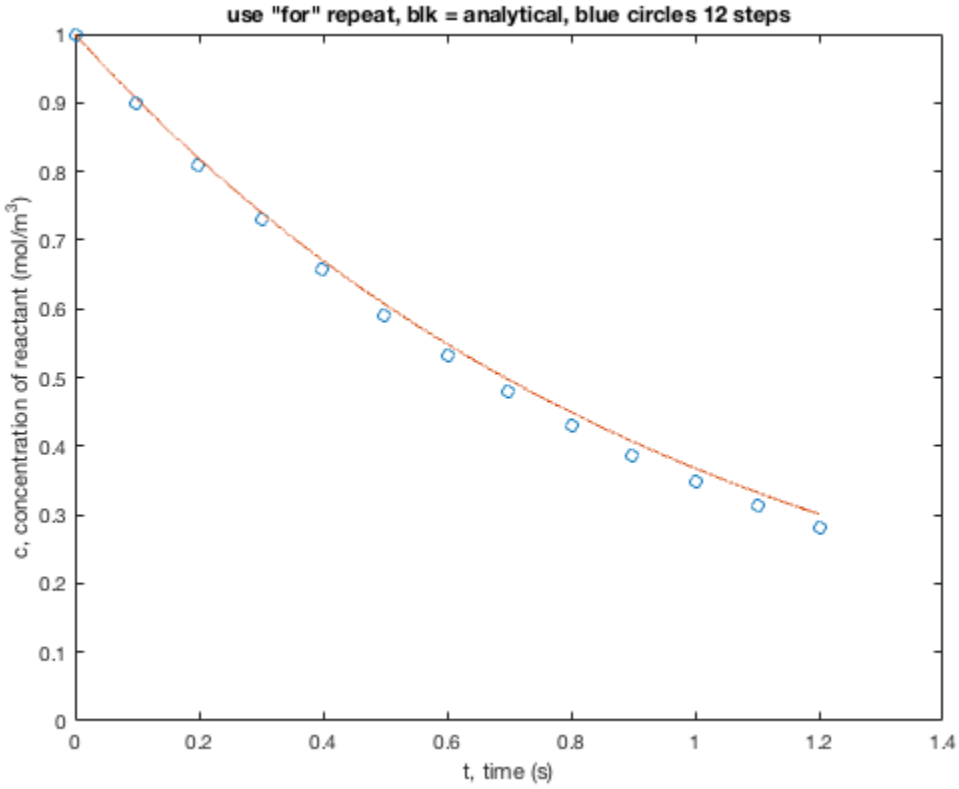
```
% reaction A > B in batch reactor (equil almost all B)
% specify well-mixed, isothermal, constant volume
% balance on A is dc/dt = -kc
% solve using Euler's method of numerical approximation
% Euler's is an approximation because it assumes rate of change
% (dc/dt here) is constant between steps of independent
% variable (t here)
% accuracy will improve by taking smaller time step
% at cost of longer computation
```

case 1 - use "for", specify final time, find conc

```
clear % get rid of any longer arrays from previous runs
k = 1; % 1/s, rate coefficient
t(1) = 0; % s, initial time
c(1) = 1; % mol/m3, initial conc of reactant A
tfinal = 1.2; % s, final time
nsteps = 12; % number time steps
dt = tfinal/nsteps; % s, time step
% for repeat automatically initializes and increments
% loop variable, which here is used as array subscript
for i = 1:nsteps
    dcdt = -k * c(i); % rate of change in c, dc/dt
    dc = dcdt * dt; % estimated change in c over time step
    c(i+1) = c(i) + dc; % estimate next c
    t(i+1) = t(i) + dt; % get next t
end
fprintf('--- case 1, use for repeat ------- \n')
fprintf('specify final time & number time steps, find final conc \n')
fprintf('final time, final = %0.2f vs. specified = %0.2f
\n',t(length(t)),tfinal)
fprintf('final conc = %0.3f \n',c(length(c)))
fprintf('number time steps specified = %i \n',nsteps)
fprintf('computed dt = %0.3f \n',dt)
figure(1)
plot(t,c,'o')
% also plot analytical solution
hold on
ta = 0:0.001:tfinal;
```

```
title(tt)
ylabel('c, concentration of reactant (mol/m^3)')
xlabel('t, time (s)')
axis([0 1.4 0 1])
hold off

--- case 1, use for repeat -------
specify final time & number time steps, find final conc
final time, final = 1.20 vs. specified = 1.20
final conc = 0.282
number time steps specified = 12
computed dt = 0.100
```

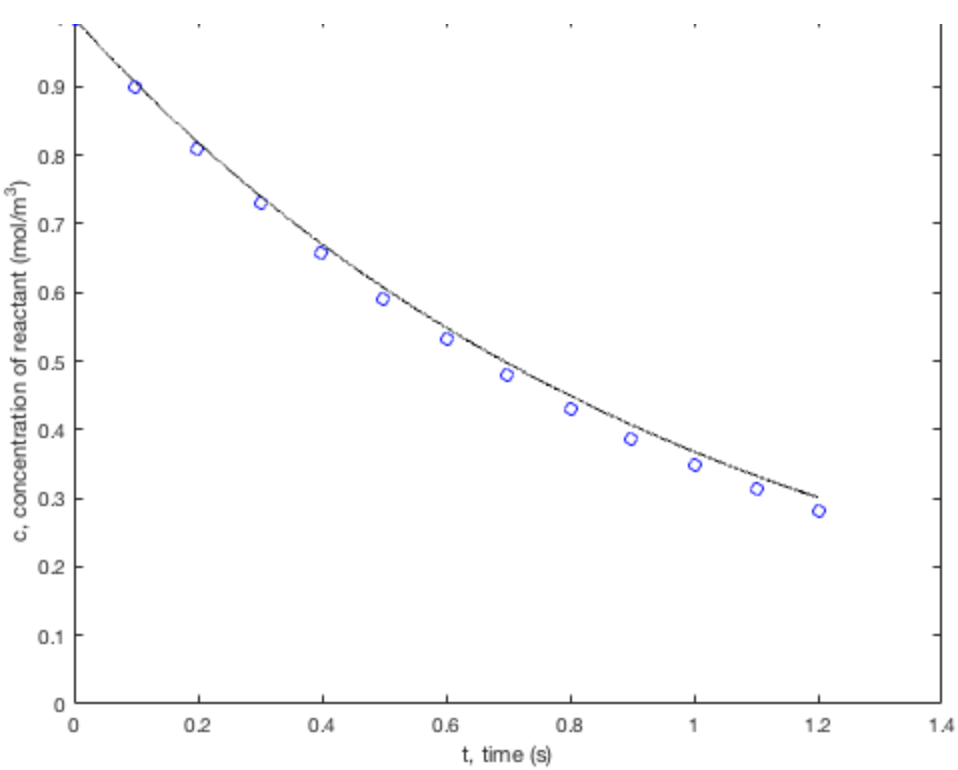


case 2 - use "while", specify final conc, find time

```
clear % get rid of any longer arrays from previous runs
k = 1; % 1/s, rate coefficient
t(1) = 0; % s, initial time
c(1) = 1; % mol/m3, initial conc of reactant A
cfinal = 0.3; % mol/m3, desired final conc of A
```

```
while c(i) > cfinal
    dcdt = -k * c(i); % rate of change in c, dc/dt
    dc = dcdt * dt; % estimated change in c over time step
    c(i+1) = c(i) + dc; % estimate next c
    t(i+1) = t(i) + dt; % get next t
    i = i + 1; % increment array subscript
end
fprintf('--- case 2, use while repeat ------- \n')
fprintf('specify final conc and time step size dt, find final time
 \n')
fprintf('final time = %0.2f \n',t(i))
fprintf('final conc, final = %0.3f vs. specified %0.3f
 \n',c(i),cfinal)
fprintf('number time steps taken = %i \n',i-1)
fprintf('specified dt = %0.3f \n',dt)
fprintf('while will stop closer to cfinal with smaller dt \n')
figure(2)
plot(t,c,'bo')
% also plot analytical solution
hold on
ta = 0:0.001:t(i);
ca = c(1)*exp(-k*ta);
plot(ta,ca,'k')
tt = sprintf('use "while" repeat, blk = analytical, blue circles %i
 steps',i-1);
title(tt)
ylabel('c, concentration of reactant (mol/m^3)')
xlabel('t, time (s)')
axis([0 1.4 0 1])
hold off
fprintf('---------------------- \n')
```

```
--- case 2, use while repeat -------
specify final conc and time step size dt, find final time
final time = 1.20
final conc, final = 0.282 vs. specified 0.300
number time steps taken = 12
specified dt = 0.100
while will stop closer to cfinal with smaller dt
----------------------
```



Published with MATLAB® R2016a