10.34 – Fall 2006

Homework #6 - Solutions

Problem 1 – Chemostat ODE

This problem is similar to the Quiz 1 case, with some changed parameters and slight structural changes to the equations. The ODEs that should have been solved were:

$$\frac{dN_{cells}}{dt} = \frac{k_1 N_{cells}\left[Nutrients\right]}{\left(1+c_1\left[Nutrients\right]\right)\left(1+d\left[P\right]\right)} - V_{flow}\frac{N_{cells}}{V_{rxtr}}$$

$$\frac{d\left[Nutr\right]}{dt} = \frac{1}{V_{Rxtr}}\left\{V_{flow}\left(\left[Nutr\right]_{In} - \left[Nutr\right]\right) - \left[k_2 N_{cells}\left(\left[Nutr\right] - 1\times10^{-6}\right) + c_2\left(Cell\ Mult\right)\right]\right\}$$

$$\frac{d\left[P\right]}{dt} = \frac{1}{V_{Rxtr}}\left\{\frac{k_3 N_{cells}\exp\left(-d\left[P\right]\right)}{\left(1+c_1\left[Nutrients\right]\right)}\cdot\left(\left[Nutrients\right]-0.01\right)^2 - V_{flow}\left[P\right]\right\}$$

Note that the reactor volume must be introduced into the equations written in terms of concentration.

The steady state of the seed reactor was found by integrating the ODE's to a long time so that the steady-state condition will be reached. The fsolve function should also work to accomplish this. The 230 L reactor part was solved by using two subsequent ODE solves. The first was used to solve the problem from t = 0 → t = $t_{nutrient}$, and the second (with the nutrient solution flowing into the reactor) was used to solve from $t_{nutrient}$ → t_{final}.

```
Steady-State Solution to the 150 L CSTR
Number of Cells in Reactor = 7.403565e+007
Concentration of Cells in Reactor = 4.935710e+005
Concentration of Nutrients (M) = 0.051391
Concentration of Product (M) = 0.032908

Steady-State Solution to the 230 L CSTR
Number of Cells in Reactor = 1.273906e+008
Concentration of Cells in Reactor = 5.538720e+005
Concentration of Nutrients (M) = 0.033451
Concentration of Product (M) = 0.018212
```

***Part A and C: see subsequent page for plots*

Part B: Time to within 1% of steady state:

- you need to take care to calculate the correct time for the case(s) that overshoot the SS value, i.e. you do not want the time point to get within 1% of the SS

	Time to 99% of steady state (sec and hrs, respectively)			
t_{nut} (hr)	*ode45*	*ode23s*	*ode45*	*ode23s*
0	12947	12935	3.60	3.59
2	9897	9881	2.75	2.74
8	31813	31810	8.84	8.84

Part D: *(Note that the tolerances used in the solution were an* `RelTol` *of 1e-10 and an* `AbsTol` *of 1e-12. The wall time and number of steps will change with the tolerance)*

	Total wall time of solver (sec)		***Total number of integrator steps***		***Time per integrator step (msec)***	
t_{nut} (hr)	*ode45*	*ode23s*	*ode45*	*ode23s*	*ode45*	*ode23s*
0	0.3305	12.7283	1938	10576	0.171	1.204
2	0.2904	11.9472	1822	10120	0.159	1.181
8	0.4406	18.2162	2766	15560	0.159	1.171

You can see in this case (and with this tolerance set) that the explicit solver is much faster than the stiff solver because it takes fewer steps and takes less time-per-step. The explicit method takes about an order of magnitude longer per time-step.

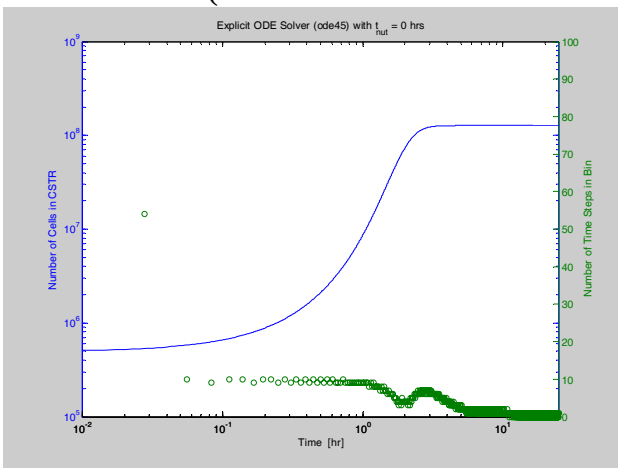
The table below shows the results when the default tolerances (`rtol = 1e-3` and `atol = 1e-6`) are used to solve the problem. There are a couple of important things to note. The stiff solver now takes much fewer steps to solve the problem, but still takes longer to do so. You will also note that the time-per-step is very similar to the above case, with the implicit solver taking a bit longer per step. This is likely due to initially estimating the Jacobian or other overhead associated with the solver, and since there are fewer steps to distribute this overhead among, the result is a slightly longer time-per-step.

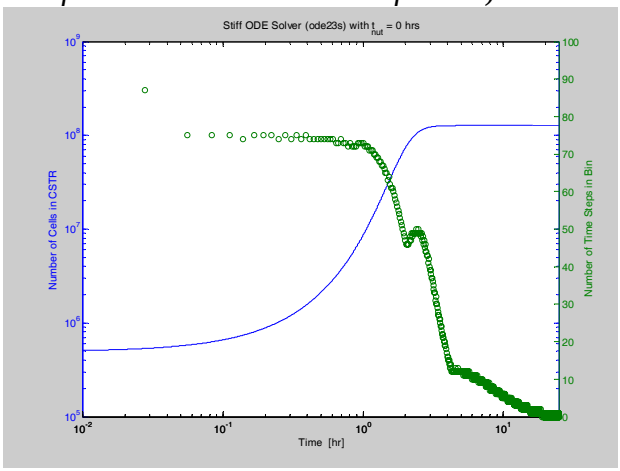
	Total wall time of solver (sec)		***Total number of integrator steps***		***Time per integrator step (msec)***	
t_{nut} (hr)	*ode45*	*ode23s*	*ode45*	*ode23s*	*ode45*	*ode23s*
0	0.0801	0.0901	438	72	0.183	1.251
2	0.0701	0.0701	410	61	0.171	1.149
8	0.0801	0.1202	454	100	0.176	1.202

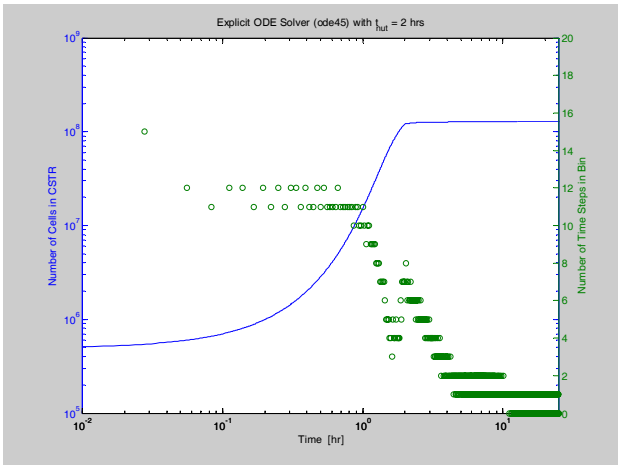
The reason why the implicit solver works in fewer steps at a low tolerance and more steps at a high tolerance is likely due to the error in the neglected term of the expansion. The ode45 solver is a 5^{th} order expansion and the ode23s is a 3^{rd} order expansion. This means that the 5^{th} order method will be able to take larger steps and still meet the high tolerances, since the error goes as $(\Delta t)^m$. Below is a table showing the results for the high-tolerances using ode15s (a 5^{th} order stiff method). You can see that in this case, the implicit solver does take fewer steps than the explicit method.

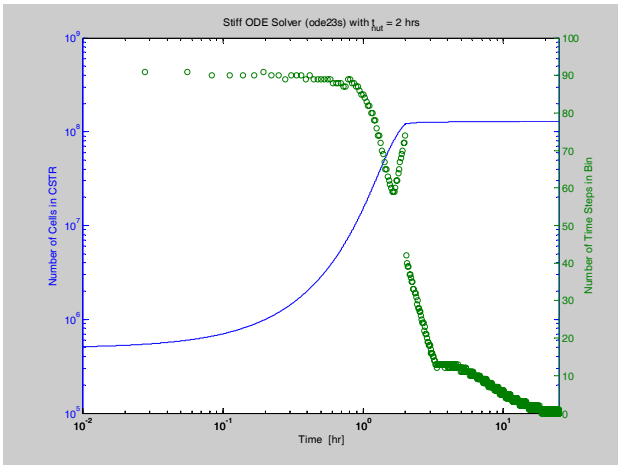
	Total wall time of solver (sec)		***Total number of integrator steps***		***Time per integrator step (msec)***	
t_{nut} (hr)	*ode45*	*ode15s*	*ode45*	*ode15s*	*ode45*	*ode15s*
0	0.2804	0.5808	1666	766	0.168	0.758
2	0.3004	0.5001	1734	739	0.173	0.677
8	0.4206	0.8713	2562	1167	0.164	0.747

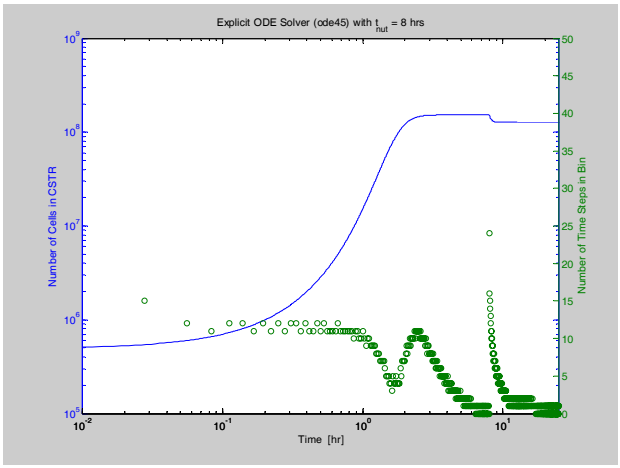
Parts A and C: (`ode45` and `ode23s`: `RelTol` *of 1e-10 and an* `AbsTol` *of 1e-12*)

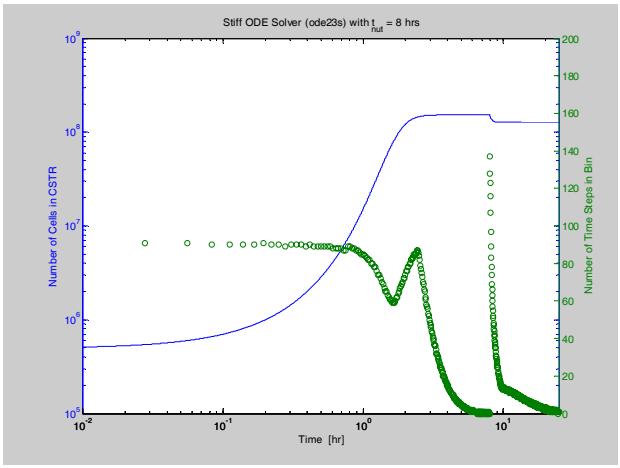












Homework 6

18 October 2006

1 Problem 2

(a) The set of differential equations are given below

$$\frac{dI}{dx} = -(a + cT^2)I$$

$$k\frac{d^2T}{dx^2} = \frac{dI}{dx}$$

with Boundary coditions

$$I_{x=0} = I_0$$

$$T_{x=0} = T_{x=L} = T_a$$

The differential equations are 2^{nd} order in T. To convert it into two 1^{st} order differential equations we define a new variable called $T' = \frac{dT}{dx}$. With this new variable the differential equations get transformed as shown below.

$$\frac{dT}{dx} = T'$$

$$\frac{dI}{dx} = -(a + cT^2)I$$

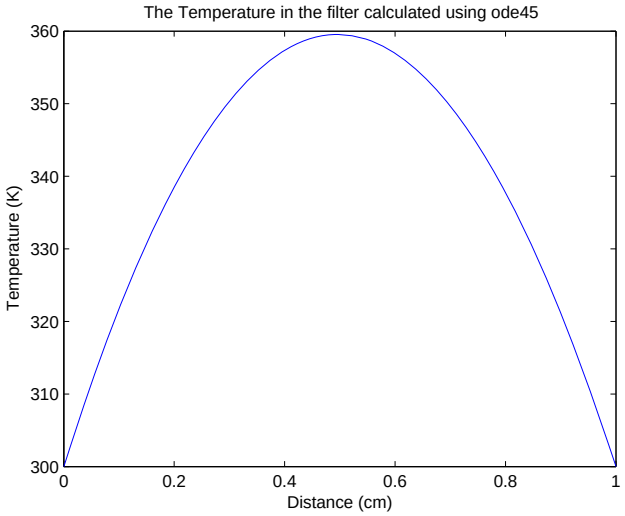
$$\frac{dT'}{dx} = \frac{-(a + cT^2)I}{k}$$

The above set of differential equation are now in the standard form

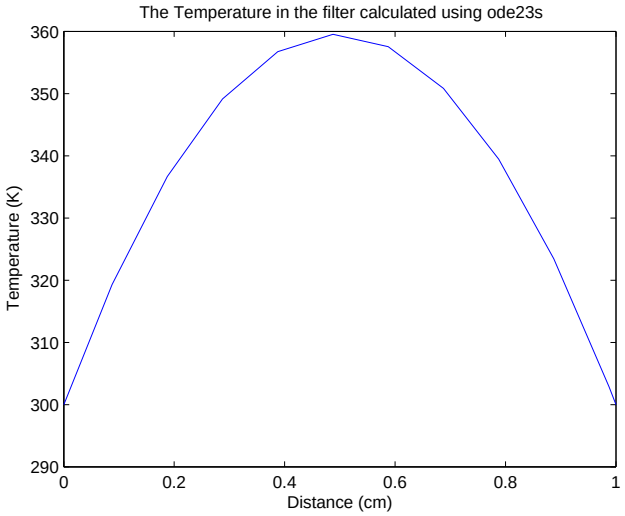
$$\frac{d\underline{Y}}{dx} = \underline{F}(\underline{Y})$$

(b) The difficulty with the above differential equations is that we don't have the necessary intitial conditions to convert it into an IVP problem. To circumvent this problem we assume a initial value for T' and solve the IVP to obtain T at $x = L$. If this T matches the ambient temperature then our system is solved. But if $T_{x=L}$ is not equal to ambient temperature then we will use some update scheme to guess a new value for T'. We will use this iterative scheme to get an approritate value of T'. In my program I have used the matlab function `fzero` to solve for the value of T'. The cost function is "$T_{x=L} - T_a$". The differential equation has been solved using `ode45` and `ode23s` and are implemented in programs `problem2a_usingode45` and `problem2a_usingode23s` respectively. The time taken to solve the system of ode once the correct value of T' is obtained is calculated by using the matlab function `cputime`. The command and the output for each part are given below.

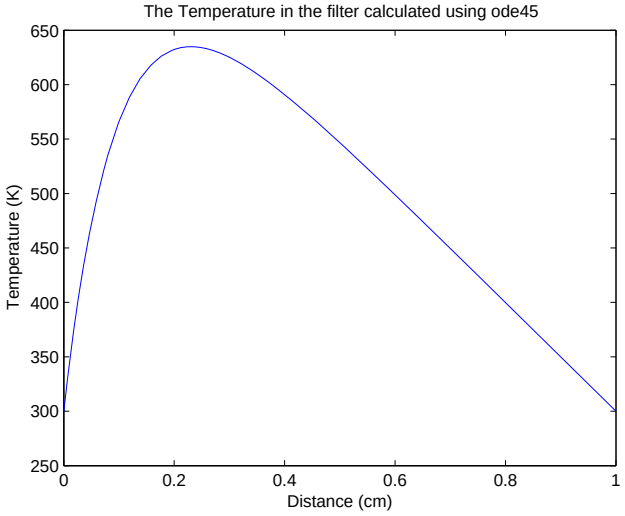
```
>> problem2a_usingode45(10,1e-7);
The time taken by ode45 = 0.010014 s
The maximum temperature achieved is 359.5373 K, at 0.49723 cm
The number of integration steps used before maximum T is 21
out of total 41
>>
```




```
>> [x,y]=problem2a_usingode23s(10,1e-7);
The time taken by ode23s =0.020029 s
The maximum temperature achieved is 359.5218 K, at 0.4876 cm
The number of integration steps used before maximum T is 6 out
of total 12
>>
```

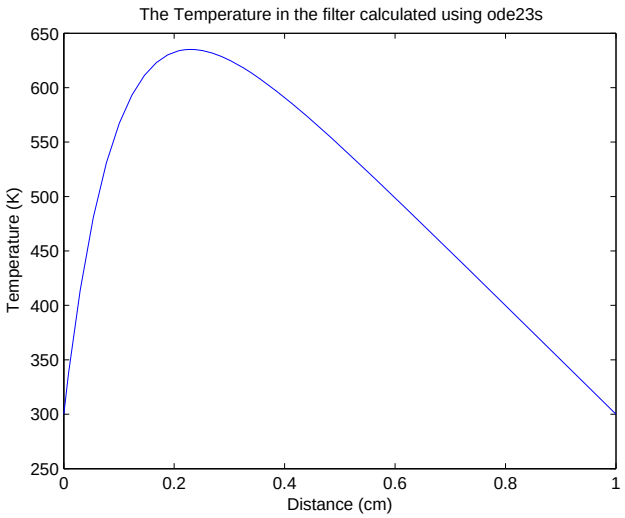



```
>> [x,y]=problem2a_usingode45(1000,1e-7);
The time taken by ode45 = 0.010014 s
The maximum temperature achieved is 634.8372 K, at 0.23261 cm
The number of integration steps used before maximum T is 17
out of total 57
>>
```

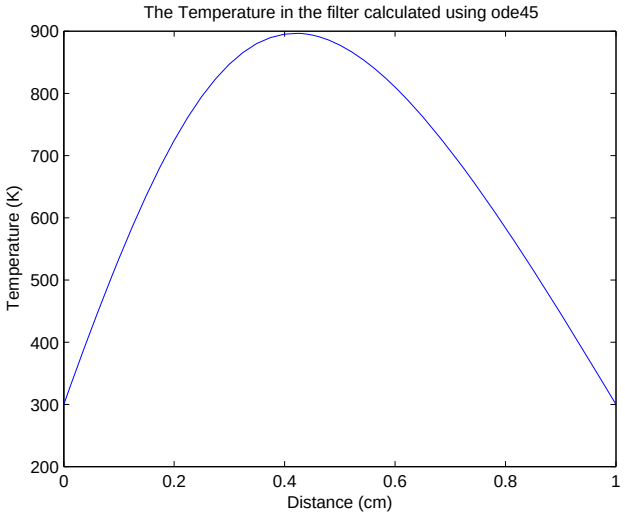



```
>> [x,y]=problem2a_usingode23s(1000,1e-7);
The time taken by ode23s =0.060086 s
The maximum temperature achieved is 635.0234 K, at 0.23496 cm
The number of integration steps used before maximum T is 13
```

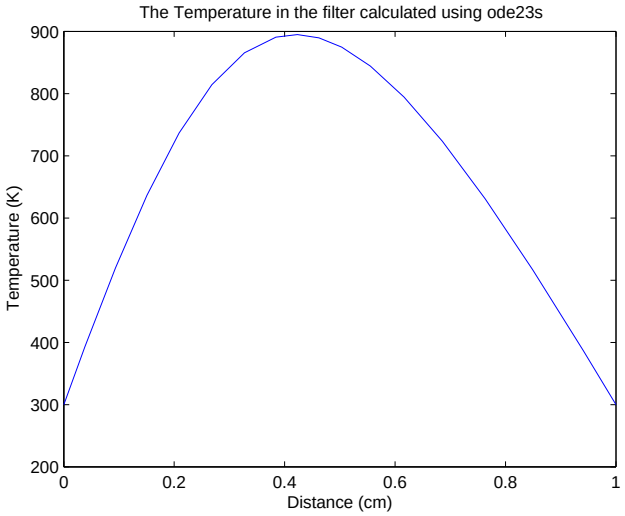
```
out of total 46
>>
```



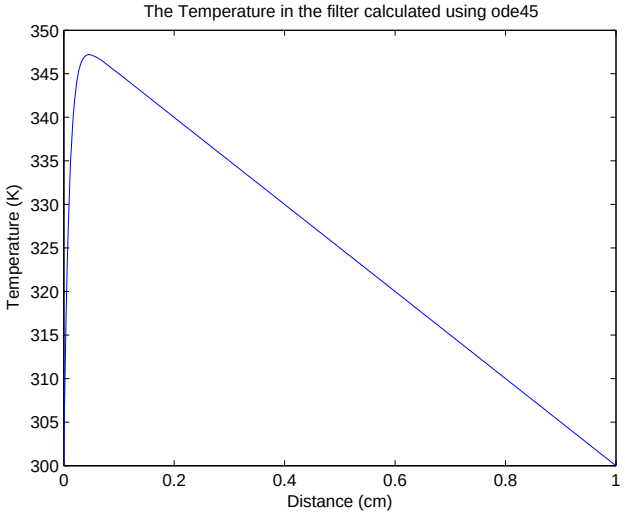
```
>> [x,y]=problem2a_usingode45(10,3e-4);
The time taken by ode45 = 0.010014 s
The maximum temperature achieved is 896.4175 K, at 0.42431 cm
The number of integration steps used before maximum T is 21
out of total 45
>>
```




```
>> [x,y]=problem2a_usingode23s(10,3e-4);
The time taken by ode23s =0.020029 s
The maximum temperature achieved is 894.8323 K, at 0.42321 cm
The number of integration steps used before maximum T is 9 out
of total 18
>>
```

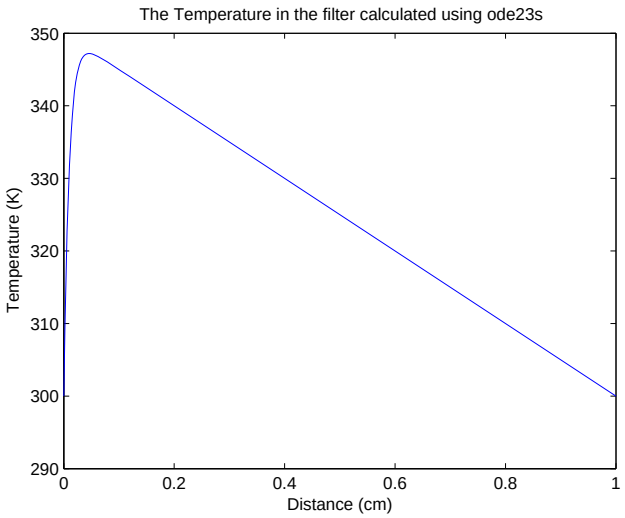



```
>> [x,y]=problem2a_usingode45(10000,1e-7);
The time taken by ode45 = 0.040058 s
The maximum temperature achieved is 347.1956 K, at 0.04677 cm
The number of integration steps used before maximum T is 28
out of total 221
>>
```




```
>> [x,y]=problem2a_usingode23s(10000,1e-7);
The time taken by ode23s =0.13019 s
The maximum temperature achieved is 347.2022 K, at 0.04630 cm
The number of integration steps used before maximum T is 22
```

```
out of total 105
>>
```



(c) For part 4 in the previous section the distance at which the maximum temperature is obtained is 0.046 cm. The number of integration steps used till this point is 28 in `ode45`. At the same average step size we will require 600 step sizes to cover 1 cm. Thus if we used the BVP method with 500 uniformly spaced grip points we will get an erroneous answer. The grid size required to get an accurate result will be more than 600 because our simplified discretization scheme incurs a greater error per time step than `ode45`. Also it can be noticed that `ode23s` takes longer to solve the IVP than `ode45`, this is not very surprising given the fact that this problem is not very stiff, all the variables change approximately at the same length scale and thus the cost of solving a system of non-linear equations at each integration step by `ode23s` is not compensated by the greater number of time steps that `ode45` has to take to solve the system.