CHE 504: Advanced Reactor Design

Tutorial No. 3: - Instructor: J.E. Gatica

Sample Problem *[see handout, P13-8 (3rd Ed.)]*

A pulse test for a "black box" [reactor] unit yields the following RTD

$$C^{OUT}(t) \quad \begin{cases} 0.1\ t & \text{for } 0 \le t \le 5 \\ 0.05\ (15 - t) & \text{for } 5 < t \le 15 \\ 0 & \text{elsewhere} \end{cases}$$

(b) Find the [average] "residence time" (i.e., <t>)

This was a simple integral

$$\langle t \rangle = \int_0^\infty t\ E(t)\ dt$$

For a second order reaction with $k\ C^o_A = 1.2\ min^{-1}$ (remember, $Da = k\ C^o_A\ V/Q$)

(c and d). Find the conversion for an ideal CSTR and an ideal PFR

Recitation

Note that we are now able to calculate the conversion in ideal flow reactors [c & d], i.e.,

```
Da = k*(CAo^(n-1)) * Tau;
fprintf ('\n Da = %7.3f \n',Da);
```

Which yields

```
Reactor Performance:
Da =   8.001

 xA(CSTR)  =   0.703
 xA(PFR)   =   0.889
```

Segregation Models

e. Find the conversion for a "segregated" fluid ("macrofluid").

The definition of the conversion for a macrofluid is

$$x_{A,MACRO} = \int_0^\infty x_A^b(t)\ E(t)\ dt$$

where "x_A^b" is the conversion in a batch reactor. One could easily follow the approach used for the average residence time calculations. This reduces to find the conversion in the batch reactor as a function of time

The solution for a batch reduces to solving the equation

$$\frac{dN_A}{dt} = -k\ C_A^n\ V, \text{ or}$$

$$\frac{dx_A}{dt} = k\ \left(C_A^o\right)^{n-1}\ \left(1 - x_A\right)^n$$

which can be easily solved for integer values of "n." For n=2, for instance, one can find

$$x_A(t) = \frac{kC_A^o\ t}{1 + kC_A^o t}$$

Now the integral

$$x_{A,MACRO} = \int_0^\infty x_A^b(t)\ E(t)\ dt$$

grows increasingly complex, and any hopes for an analytical solution would demand a significant time spent in solving the equation. This approach is all right, except that the batch reactor doesn't always have an analytical (or at least a simple analytical) solution. A more versatile approach is to transform the problem into two coupled differential equations, namely

$$x_{A,MACRO} = \int_0^\infty x_A^b(t)\ E(t)\ dt$$

becomes

$$\frac{dx_{A,MACRO}}{dt} = x_A^b(t)\ E(t)$$

which is solved in conjunction with the equation for the batch, i.e.

$$\frac{dx_A^b}{dt} = k\left(C_A^o\right)^{\text{n-1}}\left(1 - x_A^b\right)^n$$

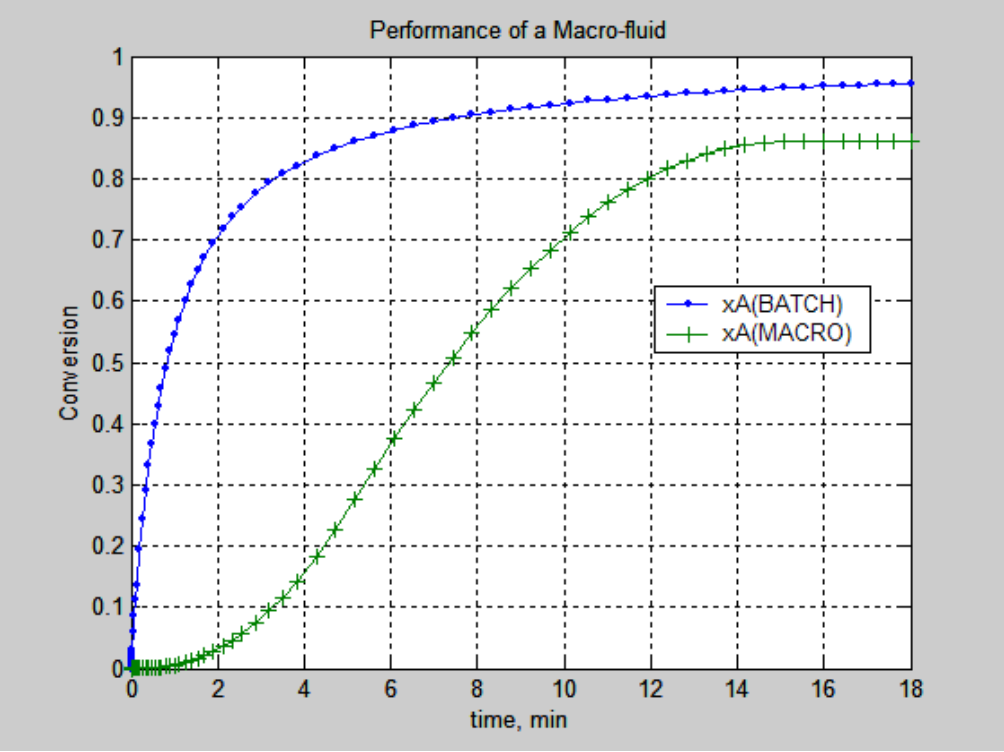
which, as expected, yields identical results

```
xA_MACRO        =   0.861 (ode15s)
xA_MACRO        =   0.860 (quad)
```

It is interesting to see a plot of the solution:

Can you make any comments out of this plot?

Why does the conversion for the batch reactor show a monotonic increase, while the performance for the macro-fluid reaches a "plateau"?



f. Find the conversion for a fluid in a maximum state of mixedness ("microfluid").

Let us now proceed to the micro-fluid calculations. This is a slightly more complex problem, the differential equation seems simple

$$\frac{dx_{A,MICRO}}{dt} = \frac{r_A}{C_A^o} + \left(\frac{E(\lambda)}{1-F(\lambda)} \right) x_{A,MICRO}$$

$$\text{subject to } x_{A,MICRO} \to 0, \text{ for } \lambda \to \infty$$

The problem here is what are the "initial conditions"? $\lambda=0$ would actually indicate $t\to\infty$, in other words, the conversion at the exit of the reactor (which is what we are trying to find!). So, the simplest alternative is to integrate backwards (as suggested in Fogler, cf. Example 13-7). There are a few problems associated with this idea, namely

- We cannot use "ode45" (or any other ODE solver available in MatLAB).
- We need a table of values for $E(\lambda)$ and $F(\lambda)$, i.e. we need to solve $F(\lambda)$ first.

If we use a simple discretization in finite differences, the equation becomes

$$\frac{dx_A}{d\lambda} = \Im(x_A, \lambda)$$

then

$$\frac{x_A^{\lambda+\Delta\lambda} - x_A^{\lambda}}{\Delta\lambda} \cong (1-\alpha)\,\Im(x_A^{\lambda}, \lambda) + \alpha\,\Im(x_A^{\lambda+\Delta\lambda}, \lambda+\Delta\lambda)$$

where the system will be fully implicit for $\alpha=1$, and fully explicit for $\alpha=0$ (if we integrated backwards it would be just the opposite). Fogler implemented a fully explicit algorithm for backwards integration, i.e.

$$x_A^{\lambda} \cong x_A^{\lambda+\Delta\lambda} - \Delta\lambda\,\Im(x_A^{\lambda+\Delta\lambda}, \lambda+\Delta\lambda)$$

or

$$x_A^{\lambda-\Delta\lambda} \cong x_A^{\lambda} - \Delta\lambda\,\Im(x_A^{\lambda}, \lambda)$$

The problem with this approach is that is numerically unstable (we will see below the results when trying to replicate Fogler's example). The first question is "What is a good choice for $\Delta\lambda$?"

Unfortunately, for non-linear problems (there is no theory for stability in non-linear problems!), we need to use trial-and-error. The "rule of thumb" is to select $\Delta\lambda$ "small" and try, then use $\Delta\lambda/2$ and try again until two successive results do not differ "significantly" (significant figures, accuracy, tolerance, etc...).

For experimental data, as presented in the textbook (for Example 13-7), there are no alternatives, and the question is actually irrelevant.

Fogler suggests interpolating some of the values (with no explanation or reasons). There is no numerical justification to do what he suggests, interpolating data is similar to generating "experimental data" we did not collect … It might be a good approximation but it will depend of the system being analyzed or the data available. Fogler's suggestion might work for this particular example … but it is likely to fail for other problems …

This is not discussed by Fogler, but the common approach is (provided we have a reliable interpolation algorithm or that the data is not corrupted by serious experimental error … for data "smooth" as the one provided it will work seamlessly … ☺ …)

A more rigorous approach is to avoid generating artificial data, and use a Fully Implicit approach, instead (also implemented in micromix1.p); which is often unconditionally stable.

Let us examine both approaches together …

This is implemented **in the MatLAB scripts "micromix2.p" and "micro_f.p"** [we will skip the details of this script as they do not pertain to CHE 504]

```
% function micromix2.m
%   Calculation of micromixing for a
%   system with a tabulated function, t vs. E(t)
%
%   Usage
%   xA_MICRO = micromix2 (method, t, E, iprint, k, CAo, n)
%
%   where
%   Inputs
%   method : method selection flag
%       < 0:    Fully explicit (Fogler's example) integration
%       > 0:    Fully implicit integration
%   t      : vector of t values, t(i), i =1, ..., N
%   E      : function E(t), E(i) is E(t) for t=t(i), i =1, ..., N
%   iprint : printing flag
%            > 0, print iteration results
%            < 0, no printing
%   k, CAo, n
%       are kinetic parameters describing a power-law kinetics
```

```
%       Da = k * (CAo^(n-1))*Tau
```

If we know the function that describes the RTD function (E(t)) as a function of time (t) [as in Problem P13-8, assigned in the Homework], then we can use a fully explicit integration approach along with a trial-and-error approach; i.e., select $\Delta\lambda$ "small" and try, then use $\Delta\lambda/2$ and try again. The integration process is repeated until two successive results do not differ "significantly" (significant figures, accuracy, tolerance, etc...).

This is implemented in the MatLAB script "micromix1.p"
[we will skip the details of this script, they do not pertain to CHE 504]

```
% function micromix1.m
%
%   Calculation of micromixing for a
%   system with
%        *** a known RTA function, E(t)***
%
%   Usage
%   xA_MICRO = micromix1 (E, nsig, iprint, infinity, k, CAo, n)
%
%   where
%
%   Inputs
%   E, function E(t), it returns E(t) for a given t
%   nsig    : number of significant figures sought
%           in the solution
%   iprint  : printing flag
%           > 0, print iteration results
%           < 0, no printing
%   infinity : a value of t "large enough"
%           to ensure all the tracer has left the reactor
%   k, CAo, n : kinetic parameters, for
%       rA = rate(xA) = - k * CAo^n *(1-xA)^n;
```

Supplementary Reading for Next Week:

Review Example 13-7