# Outline for Second Lecture on Numerical Kinetic Models

How does CHEMIN work?
Very short intro to Numerical solution of ODE's
Overview: What are we trying to accomplish in Kinetics?!
Difference between measuring kinetics in the real system, and measuring rate constants in designed experiments.
Implications of reactive intermediates.

## How does CHEMKIN work?

Reads chem.inp, surf.inp. Computes thermo (and any other needed properties) for all the species, and computes the reverse rate constants from the thermo. Most of the computations are simple, mostly what this program is doing is parsing the input deck and putting everything in the right format for the next steps. In the old versions, this is called Ckinterp, since it "interprets" the input deck.

Reads aurora.inp
Sets up a program that returns F(Y), the right hand side of the ODE (or DAE) system, using all the rate constants and molecular properties computed in the first step. Then it calls VODE or DASSL or similar to solve the equations.

Postprocessor: Takes the table of yn(ti) computed by VODE and pulls out the columns you want, and plots them.

## ***<u>How are ODE's solved numerically? (the short short version)</u>***

If we define $Y=\{y_i, T\}$ and there is no transport, then Eq.(1) is of the form:

$dY/dt = F(Y)$ and we usually know the initial conditions: $Y(t_o)=Y_o$

The general procedure is to step forward in time with some formula

$$Y(t+\Delta t) = Y(t) + G(Y)\,\Delta t \qquad \text{Eq.(3)}$$

where G is our best estimate of the average of $F(Y(t'))$ over the trajectory from $Y(t'=t)$ to $Y(t'=t+\Delta t)$, using lots of little timesteps $\Delta t$ until we reach $t_{final}$. In the simplest approximation called Forward (or Explicit) Euler $G=F(Y(t))$. This turns out to be pretty inaccurate (just like the rectangle rule is not a very accurate way to compute numerical integrals) and also numerically unstable unless $\Delta t$ is very small. Note that with the Forward Euler, the relative error in $\Delta Y$ is $\frac{1}{2}\, d^2Y/dt^2\, \Delta t / F$, and $d^2Y/dt^2 = J\,F$, so the relative error could be as large as $\frac{1}{2}\lambda_{max}\,\Delta t = \frac{1}{2}\Delta t/\tau_{shortest}$.

The fundamental problem is that we don't know the trajectory $Y(t')$ – all we can do is extrapolate to estimate $Y(t')$ at future times. The extrapolation at each timestep introduces a little bit of error, and then the extrapolation at the next timestep is built on an incorrect starting point and an incorrect estimate of the slope G(Y). If one is not careful, the errors can cumulate in a very unfavorable way, making the whole procedure numerically unstable (even if the ODEs and the physical situation they describe is perfectly stable). Stability considerations indicate that for explicit methods (where G depends only on values of Y you have already computed) you want $\|I+J\Delta t\| < 2$, again saying you need $\Delta t/\tau_{shortest} < 1$ to be safe.

There are two general ways of coping with this. One is to use relatively simple estimates of G and just use tiny timesteps $\Delta t$ much smaller than any of the physical timescales in the system. This general approach is called "explicit", the most famous algorithms that work this way are called "Runge-Kutta" methods. The alternative approach is to use very complicated methods for estimating G that are guaranteed to be numerically stable, these are called "implicit" methods because in these methods G is a function of $Y(t+\Delta t)$, so Eq.(3) becomes an implicit (usually) nonlinear system of equations, very difficult to solve. The hope is that by improving the numerical stability, one could use large $\Delta t$'s and so reduce the number of timesteps required to reach $t_{final}$.

The difficult aspect of chemical kinetics is that one often has reactive intermediates with lifetimes $10^{12}$ times shorter than the relatively inert starting materials. The short lifetimes of the reactive species force $\Delta t$ in the explicit methods to be extremely small, so then a huge number of timesteps (and a correspondingly huge amount of CPU time) are required before the starting materials are consumed significantly. When the range of timescales is very large, the ODE system is called "stiff". Most important situation is when you want to know the system behavior on a relatively long time scale many orders of magnitude larger than $\tau_{shortest}$. Stiffness introduces many numerical problems, even for implicit ODE solvers; several algorithmic tricks must be used simultaneously in order to solve stiff systems without introducing huge numerical errors.

A key idea for dealing with stiff systems is "adaptive time stepping", where the ODE solver decides after each step whether to increase or decrease $\Delta t$ for the next step. Stiff solvers usually start with very small step sizes and repeated computations of the Jacobian, but when everything is going fine they do a great job of extrapolating forward in time and so can use very large $\Delta t$ and the same Jacobian over and over without introducing significant errors. This sometimes causes problems, if something dramatic happens in the equations at a particular time, e.g. in combustion/oxidation systems the chemistry changes very abruptly with the $O_2$ is all consumed, this can cause VODE to fail. One of the diagnostics from VODE and DASSL lists the number of "step failures", this shows where the program realized its extrapolation was not accurate enough, and it had to back up to a smaller Dt (and usually also recomputed the Jacobian).

The first algorithm that could correctly handle stiff ODE systems was discovered by William Gear in the 1970's. The best programs now available for solving large stiff ODE systems are VODE, DASSL/DASPK, and DAEPACK. I believe the current version of CHEMKIN calls VODE and DASSL.

**<u>What is the overall goal of Kinetics?</u>**

a) Develop Predictive Models for Y(t) for important systems (e.g. atmosphere, combustion, chemical reactors, biological systems)
b) To understand the fundamental elementary step chemistry (in large part to aid goal (a))

$dY/dt = F(Y,K)$

The real systems usually have lots of species and unknown rate constants, so Y and K are big arrays. Experimentally, we can typically either measure one or two species as a function of time, or measure several species at a fixed time (by end product analysis). Most of the time we cannot measure enough species to determine any of the rate constants. There are many more rate constants than species, so almost never could one determine all the rate constants even if one could measure all the species.

So instead of studying the real system, we set up artificial laboratory systems, where the rate constant(s) we want to determine very strongly influence the handful of measurements we can make.

**Implications of Reactive Intermediates:**

In real system, usually in steady state and low concentration, very difficult to detect and very difficult to infer k's from them. Consider A B C quasi steady state.

Very difficult to get absolute concentrations

Solution: pseudo-first-order, make artificially high concentrations of reactive intermediate and watch system relax.