Iran First International Combustion School (ICS2019)
Tehran, 24-26 August 2019

Combustion Modeling

2. Numerical algorithms for reactive flows with detailed kinetic mechanisms: 0D systems

Alberto Cuoci

References

[Bisetti2015] F. Bisetti, *Short Course on Reactive Flow Modelling*, International CI Summer School 2015, Procida (Italy)

[Cuoci2019] A. Cuoci, *Numerical modeling of reacting systems with detailed kinetic mechanisms*, Computer Aided Chemical Engineering, 45, p. 675-721 (2019)

[Kee2017] R.J. Kee, M.E. Coltrin, P. Glarborg, *Chemically Reacting Flow: Theory and Practice*, Wiley, 2 edition, 2017

[RD2000] Reaction Design, *CHEMKIN, A software package for the analysis of gas-phase chemical and plasma kinetics*, CK-TUT-10112-1112-UG-1, CHE-036-1, CHEMKIN Collection Release 3.6, September 2000, https://www3.nd.edu/~powers/ame.60636/chemkin2000.pdf

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) Solution of ODE systems
   c) Numerical methods for stiff ODE systems
   d) The Jacobian matrix

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) Solution of ODE systems
   c) Numerical methods for stiff ODE systems
   d) The Jacobian matrix

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

Numerical challenges in combustion

1. Number of equations

Since detailed kinetic mechanisms involve hundreds or thousands of species, the number of coupled equations can be very large, especially when multidimensional geometries are simulated

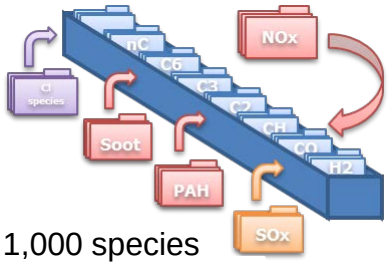
2. Non-linearity

The transport equations of species and energy are very non-linear, because of reaction rates expressions (power-law and exponential)

3. Stiffness

The characteristic times of species involved in a kinetic scheme can differ by several order of magnitudes.

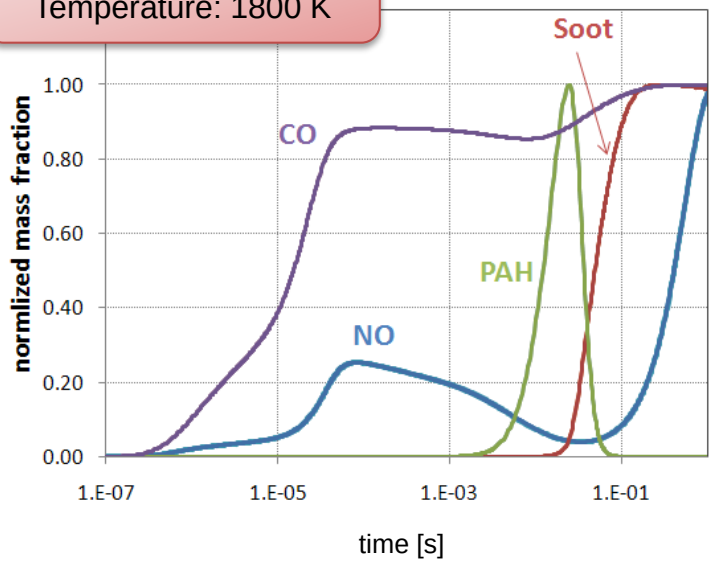
Detailed kinetic mechs



~ 1,000 species
~ 10,000 reactions

Unsteady flame

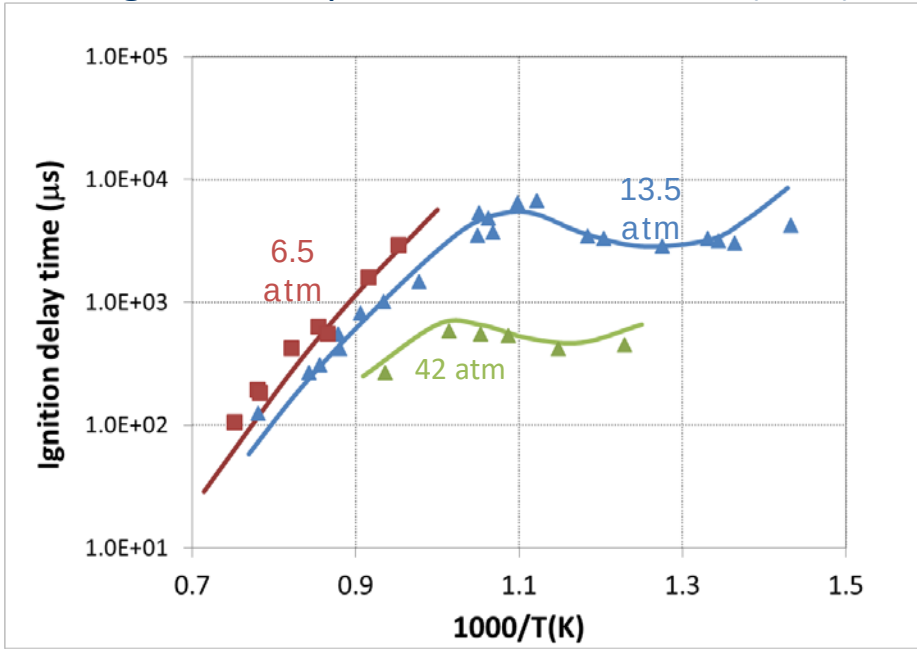
Inlet mixture: C_3H_8 + Air
Temperature: 1800 K



Need of detailed chemistry (I)

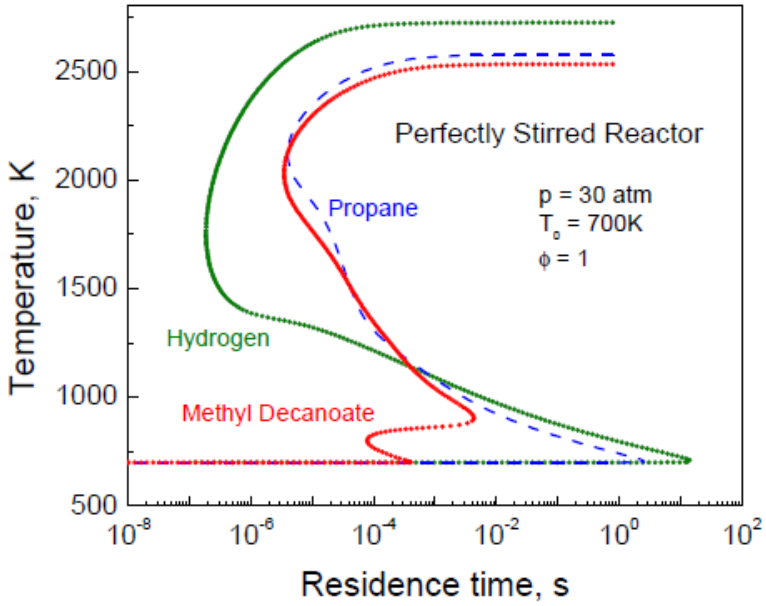
Detailed chemistry is important for: ignition, extinction, instabilities …

Negative temperature Coefficients (NTC)



Experimental data from:
Ciezki H.K. and Adomeit G., *Shock-tube investigation of self-ignition of n-heptane-air mixtures under engine relevant conditions*, Combustion and Flame 93 p. 421–433 (1993)

Combustion "S"-curves



Plot from:
Lu T., *Computational Tools for Diagnostics and Reduction of Detailed Chemical Kinetics*, Princeton-CEFRC Summer School on Combustion (2012)

Need of detailed chemistry (II)

Real fuels and surrogates

need of modeling synergistic effects between the different components

Biofuels

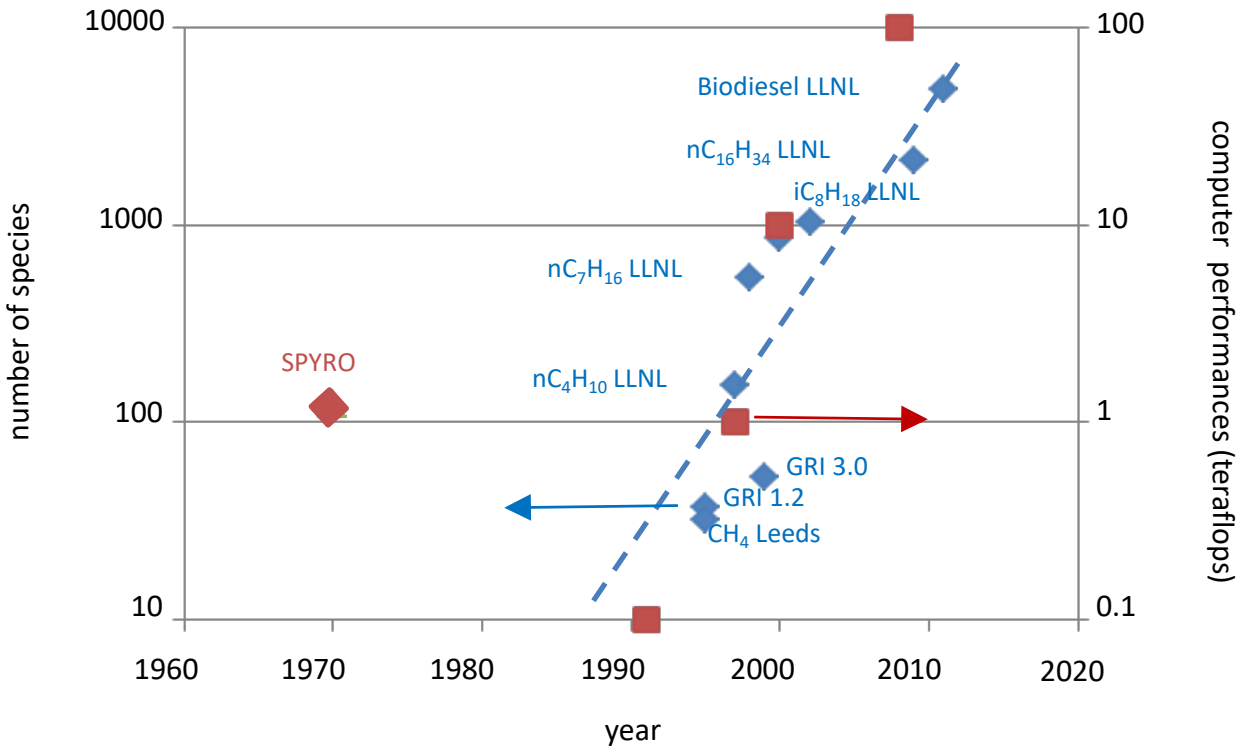
bioalcohols, biodiesel, green diesel, bioethers

Pollutant emissions

NOx, SOx, PAHs, soot

Realistic numerical simulations of pyrolysis and combustion require not only accurate, detailed modeling of fluid dynamics, but also a detailed characterization of chemical reactions and physical and chemical properties of the gas mixture.
The inaccuracy and inadequacy of simple approaches assuming either equilibrium chemistry or global mechanisms have been clearly demonstrated in recent years. This has promoted an increasing effort to develop and incorporate more complex reaction mechanisms in the numerical simulation of combustion and pyrolysis

Detailed chemistry and computational resources

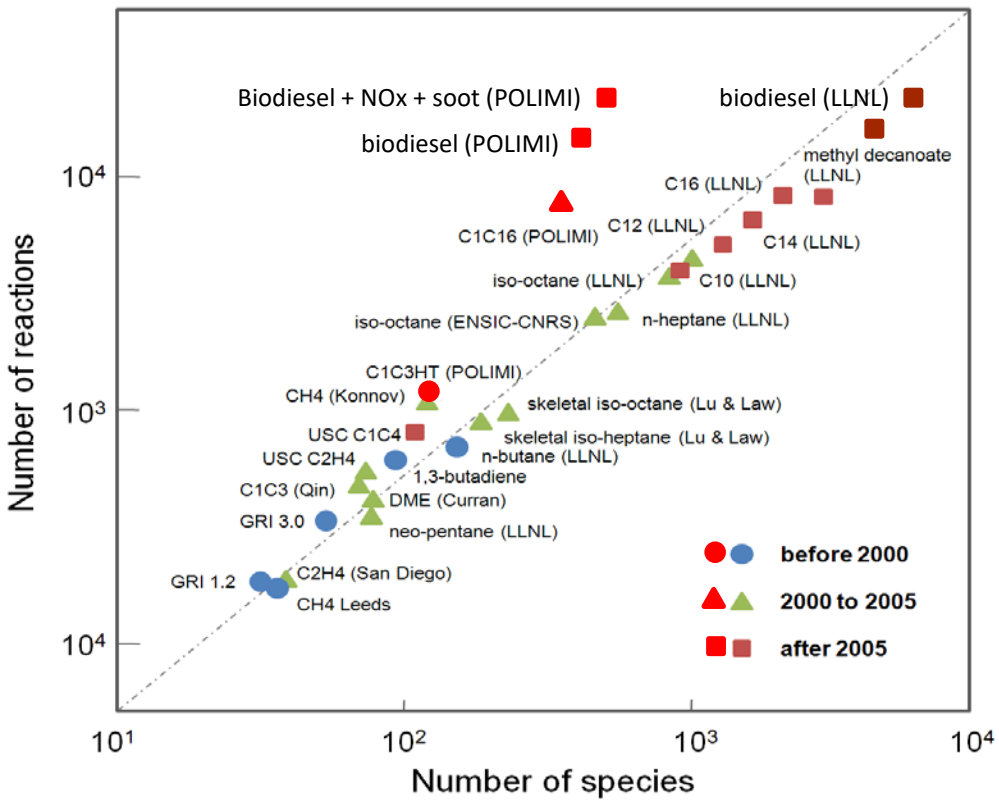


Detailed kinetic mechanisms include many species: the resulting equation systems are very large

Adapted from:

T. Faravelli, *Numerical Modeling of Pollutant Emissions with Detailed Kinetics: from Ideal Reactors to Flames*, Invited Lecture at 14th ICNC 2013, San Antonio (TX)

Kinetic mechanism size



Adapted from: **T.F. Lu, C.K. Law**, *Toward accommodating realistic fuel chemistry in large-scale computations,* Progress in Energy and Combustion Science, 35, p. 192–215 (2009)

increasing effort to incorporate more complex reaction mechanisms in simulation of combustion processes

this has led to the development of reaction mechanisms with different levels of detail and comprehensiveness

computational cost associated with such mechanisms is usually very high

need of numerical techniques and computational tools to make:

-the use of large kinetic schemes computationally efficient
-easy their integration in new and/or existing numerical codes

Non-linearity and sparsity

A reaction in general form: $\upsilon_1^f S_1 + \upsilon_2^f S_2 + \cdots + \upsilon_K^f S_K = \upsilon_1^b S_1 + \upsilon_2^b S_2 + \cdots + \upsilon_K^b S_K$

υ_j^f forward stoichiometric coefficients

υ_j^b backward stoichiometric coefficients

The reaction rates are strongly non-linear!

$$r_f = k_f(T) \prod_j C_j^{\upsilon_j^f} \qquad r_b = k_b(T) \prod_j C_j^{\upsilon_j^b}$$

$$k_f(T) = AT^n exp\left(-\frac{E}{RT}\right) \qquad k_b(T) = \frac{k_f(T)}{K_{eq}(T)}$$

Detailed chemistry is very sparse!

An elementary reaction only involve a few species (usually not more than 4)

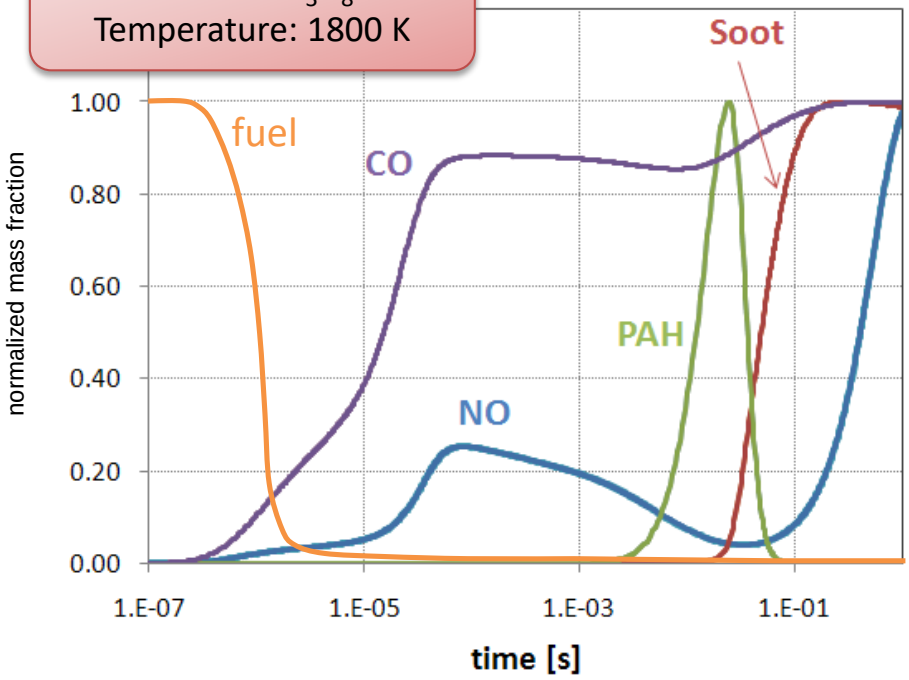
$$S_1 + S_5 = S_7 + S_{12}$$

Directly coupled species increases linearly with the mechanism size

Adapted from:
Lu T., *Computational Tools for Diagnostics and Reduction of Detailed Chemical Kinetics*, Princeton-CEFRC Summer School on Combustion (2012)

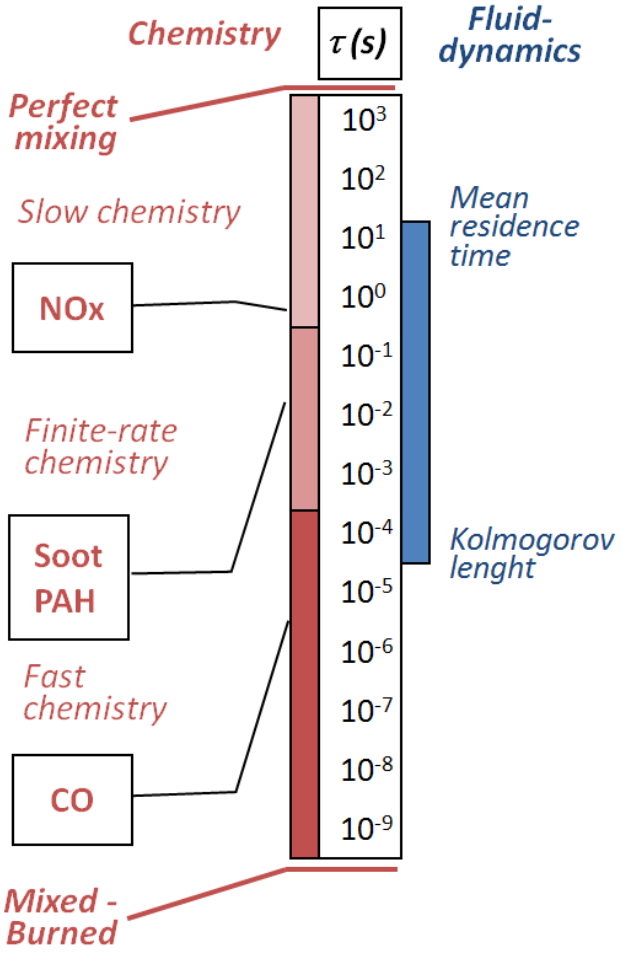
Chemistry is stiff (I)

Inlet mixture: C_3H_8 + Air
Temperature: 1800 K



1. Slow modes:
Nox and soot formation
CO → CO2 (often rate limiting)

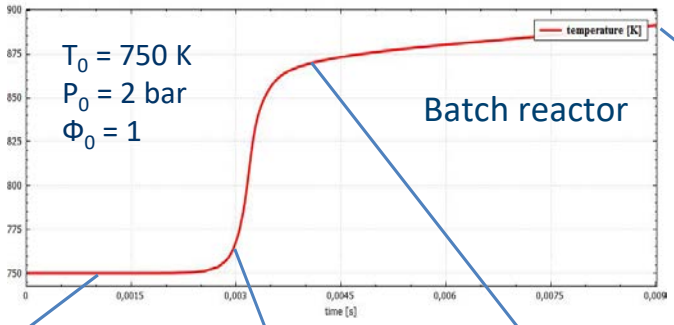
2. Fast modes:
Reactions involving highly reactive radicals (H, O, OH, …)
HCO → CO
CH3O → CH2O



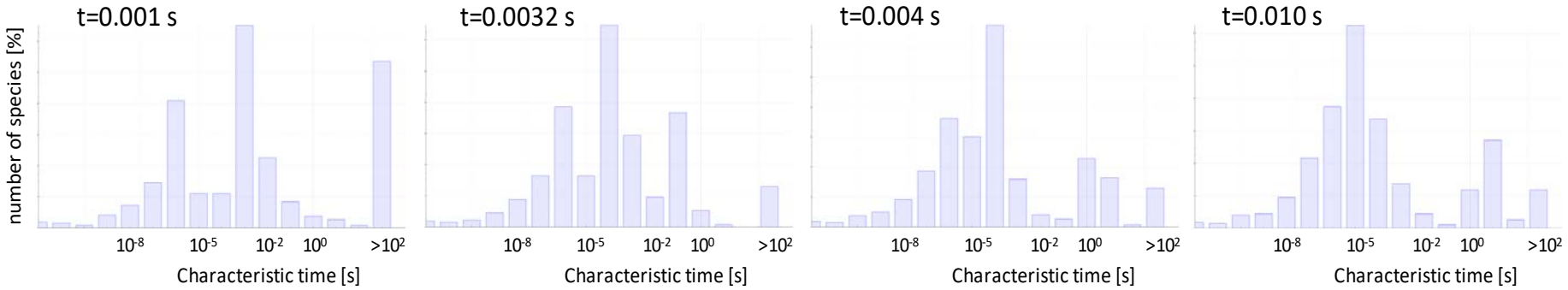
Adapted from:
R. Fox, *"Computational models for turbulent reacting flows"*, Cambridge University Press (2002)

Chemistry is stiff (II)

LLNL-NC7 Mechanism
Species: 654
Reactions: 2,837



Distribution of characteristic times calculated through the eigenvalue analysis of the Jacobian matrix associated to the system of species and energy equations



Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) **The 0D reactor model**
   b) Solution of ODE systems
   c) Numerical methods for stiff ODE systems
   d) The Jacobian matrix

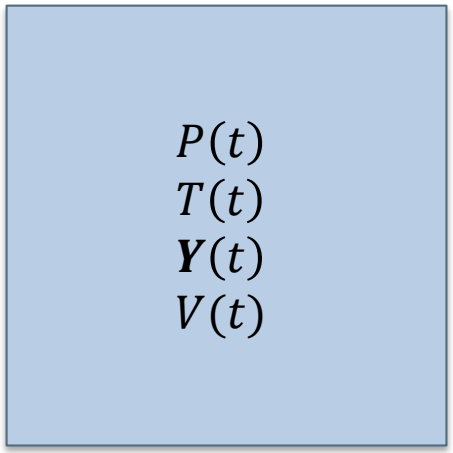
3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

The batch reactor (or 0D reactor) (I)

The “zero-dimensional reactor” or batch reactor is a fundamental configuration in numerical combustion.

It consists of a reactive system with pressure $P(t)$, temperature $T(t)$, composition $\boldsymbol{Y}(t)$ and volume $V(t)$.

The system is surrounded by an impermeable (no mass transfer) and adiabatic (no heat transfer) walls.



Applications

- A model for studying the properties of kinetic mechanisms
- A tool for Computational Fluid Dynamics in the context of operator splitting (more later)
- A model for shock-tubes and kinetics studies

F. Bisetti, *Short Course on Reactive Flow Modelling*, International CI Summer School 2015, Procida (Italy)

The batch reactor (or 0D reactor) (II)

0D means that the batch reactor is assumed to be perfectly homogeneous, i.e. no spatial gradients of variables are present.

The batch reactor variables change only in time, but the homogeneity is always kept. This means that we do not need any spatial coordinate to describe the reactor, so the name 0D reactor.

The governing equations of a batch reactor can easily derived from the general 3D governing equations, by removing all the spatial gradients. By definition, there is no sense in defining a velocity field (no need of momentum equation). Moreover, since the system is closed, the continuity equation simply states that the total mass in the reactor does not change in time. Thus:

0D reactor governing equations

$$\begin{cases} \rho \dfrac{dY_i}{dt} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \dfrac{dP}{dt} + \dot{Q} \end{cases}$$

* We need an additional equation describing the evolution of pressure

The batch reactor (or 0D reactor) (II)

0D reactor governing equations

$$\begin{cases} \rho \dfrac{dY_i}{dt} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \dfrac{dP}{dt} + \dot{Q} \end{cases}$$

Why is the 0D reactor so important?

It is a much more simpler system than a 3D reacting flows (no spatial dimensions), but it retains the major features/complexities associated to chemical reactions of a reacting flow:

- Strongly non-linear equations
- Stiffness
- Strong coupling between the equations

Species equations in concentrations

In many cases it is more convenient to rewrite the equations of species using the concentrations (instead of the mass fractions)

$$\frac{dC_i}{dt} = \frac{1}{W_i}\frac{d}{dt}(\rho Y_i) = \frac{\rho}{W_i}\frac{dY_i}{dt} + \frac{Y_i}{W_i}\frac{d\rho}{dt} = \frac{\dot{\Omega}_i}{W_i} + \frac{\rho Y_i}{W_i}\frac{1}{\rho}\frac{d\rho}{dt} = \frac{\dot{\Omega}_i}{W_i} + C_i\frac{1}{\rho}\frac{d\rho}{dt} = \frac{\dot{\Omega}_i}{W_i} - C_i\frac{1}{V}\frac{dV}{dt}$$

Species equations $\quad \frac{dC_i}{dt} = \frac{\dot{\Omega}_i}{W_i} - C_i\frac{1}{V}\frac{dV}{dt}$

- The rate of change of molar concentrations is due to **reactions (first term)** and **changes in volume (second term)**.
- If $V = const$, then the second term is zero and the rate of change of molar concentrations is due to reactions only.

Species equations in mole fractions

It is also possible to rewrite the species equations using the mole fractions:

$$\frac{dX_i}{dt} = \frac{1}{W_i}\frac{d}{dt}(WY_i) = \frac{d}{dt}\left(\frac{Y_i/(1/W)}{W_i}\right) = \frac{W}{W_i}\frac{dY_i}{dt} - W^2\frac{Y_i}{W_i}\frac{d}{dt}\left(\frac{1}{W}\right)$$

$$\frac{d}{dt}\left(\frac{1}{W}\right) = \sum_{j=1}^{N}\frac{1}{W_j}\frac{dY_j}{dt} = \frac{1}{\rho}\sum_{j=1}^{N}\frac{\dot{\Omega}_j}{W_j}$$

$$\frac{dX_i}{dt} = \frac{W}{W_i}\frac{dY_i}{dt} - W^2\frac{Y_i}{W_i}\frac{d}{dt}\left(\frac{1}{W}\right) = \frac{W}{\rho}\frac{\dot{\Omega}_i}{W_i} - W^2\frac{Y_i}{W_i}\frac{1}{\rho}\sum_{j=1}^{N}\frac{\dot{\Omega}_j}{W_j} = \frac{W}{\rho}\frac{\dot{\Omega}_i}{W_i} - X_i\frac{W}{\rho}\sum_{j=1}^{N}\frac{\dot{\Omega}_j}{W_j}$$

Species equations $$\frac{dX_i}{dt} = \frac{RT}{P}\left(\frac{\dot{\Omega}_i}{W_i} - X_i\sum_{j=1}^{N}\frac{\dot{\Omega}_j}{W_j}\right)$$

Special case: constant pressure reactor

The constant pressure assumption is often used in the context of 0D reactors. The corresponding governing equations become:

Mass fractions

$$\begin{cases} \rho \dfrac{dY_i}{dt} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \dot{Q} \end{cases}$$

which constitutes an ***"autonomous"*** system of nonlinear ODEs as it does not explicitly include a dependency on time t, but only on state $(T; P; \boldsymbol{Y})$.

Concentrations

$$\begin{cases} \dfrac{dC_i}{dt} = \dfrac{\dot{\Omega}_i}{W_i} - C_i \dfrac{1}{V}\dfrac{dV}{dt} \\ \rho C_P \dfrac{dT}{dt} = \dot{Q} \end{cases}$$

Mole fractions

$$\begin{cases} \dfrac{dX_i}{dt} = \dfrac{RT}{P}\left(\dfrac{\dot{\Omega}_i}{W_i} - X_i \sum_{j=1}^{N} \dfrac{\dot{\Omega}_j}{W_j}\right) \\ \rho C_P \dfrac{dT}{dt} = \dot{Q} \end{cases}$$

Special case: constant volume reactor

The constant volume assumption is often used in the context of 0D reactors. Assuming an ideal gas behavior, we can derive the equation governing the evolution of pressure, needed to complete the set of equations describing the reactor:

$$\frac{dP}{dt} = \frac{d}{dt}\left(\frac{NRT}{V}\right) = \frac{P}{T}\frac{dT}{dt} + \frac{P}{C_{tot}}\frac{dC_{tot}}{dt} = \frac{P}{T}\frac{dT}{dt} + \frac{P}{C_{tot}}\sum_{j=1}^{N}\frac{\dot{\Omega}_j}{W_j}$$

Governing equations

$$\begin{cases} \rho\dfrac{dY_i}{dt} = \dot{\Omega}_i \quad \longleftrightarrow \quad \dfrac{dC_i}{dt} = \dfrac{\dot{\Omega}_i}{W_i} \\ \rho C_P\dfrac{dT}{dt} = \dfrac{dP}{dt} + \dot{Q} \\ \dfrac{dP}{dt} = \dfrac{d}{dt}\left(\dfrac{NRT}{V}\right) = \dfrac{P}{T}\dfrac{dT}{dt} + \dfrac{P}{C_{tot}}\displaystyle\sum_{j=1}^{N}\dfrac{\dot{\Omega}_j}{W_j} \end{cases}$$

ODE (Ordinary Differential Equations) Systems

Independently of the additional assumptions on the nature of the 0D reactor, the relevant point is that the governing equations are a system of ODEs (Ordinary Differential Equations) with Initial Conditions (IC). In the most general case:

$$\text{ODEs} \begin{cases} \dfrac{dY_i}{dt} = f_{Yi}(\boldsymbol{Y}, T, P) \\ \dfrac{dT}{dt} = f_T(\boldsymbol{Y}, T, P) \\ \dfrac{dP}{dt} = f_T(\boldsymbol{Y}, T, P) \end{cases} + \text{ICs} \begin{cases} Y_i(t=0) = Y_i^0 \\ T(t=0) = T^0 \\ P(t=0) = P^0 \end{cases}$$

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) **Solution of ODE systems**
   c) Numerical methods for stiff ODE systems
   d) The Jacobian matrix

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

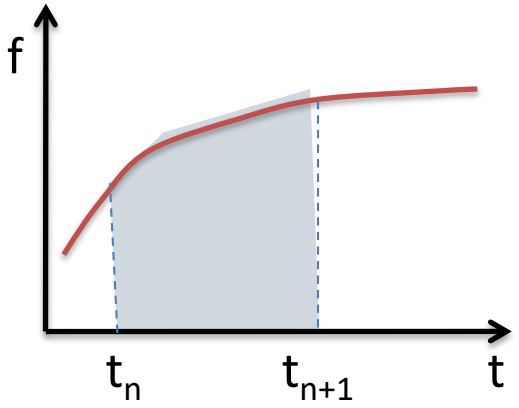
Solution of ODEs with ICs

Ordinary Differential Equations (ODE) system (m equations)

$$\begin{cases} \dfrac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t) \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

If f does not explicitly depend on t, the system is autonomous

$$\int_{t_n}^{t_{n+1}} \frac{d\mathbf{y}}{dt} dt = \mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) = \int_{t_n}^{t_{n+1}} f(\mathbf{y}) dt$$



$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \int_{t_n}^{t_{n+1}} f(\mathbf{y}) dt$$

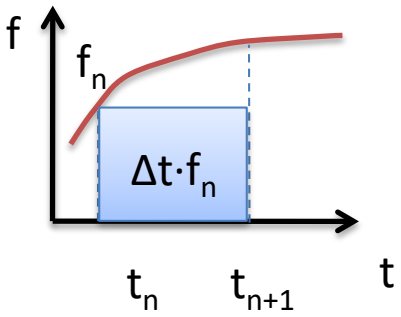
Euler's Methods

Explicit (forward) Euler's Method

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{y})dt \approx \mathbf{f}_n h_n$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}_n h_n$$

The values at the new time step can be obtained **explicitly**, since all the information at the starting time is known



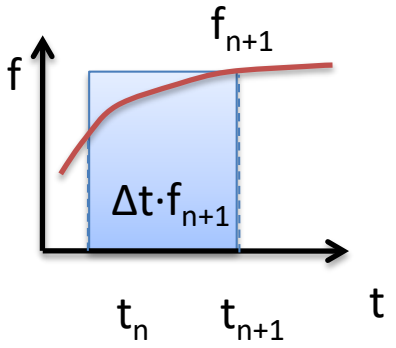
$$h_n = t_{n+1} - t_n$$

Implicit (backward) Euler's Method

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{y})dt \approx \mathbf{f}_{n+1} h_n$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}_{n+1} h_n$$

The values at the new time step cannot be calculated explicitly, but require the solution of a non-linear system of algebraic equations



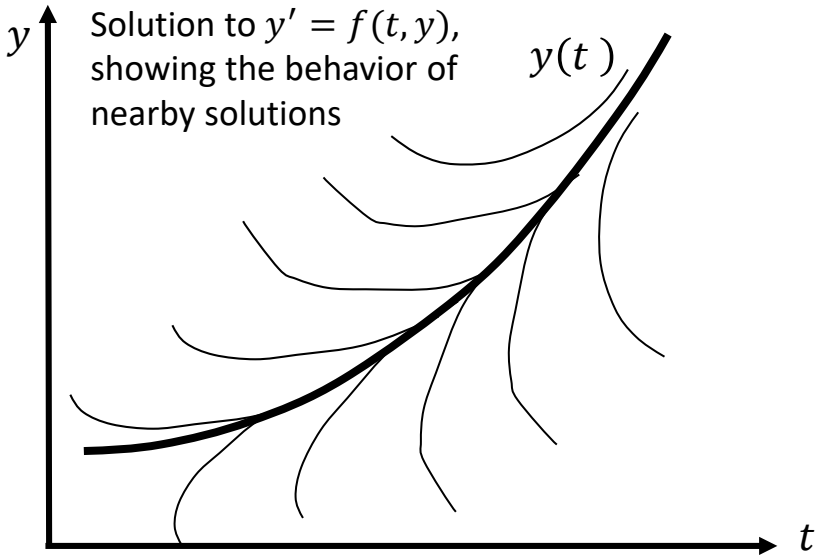
Analysis of explicit Euler's Method (I)

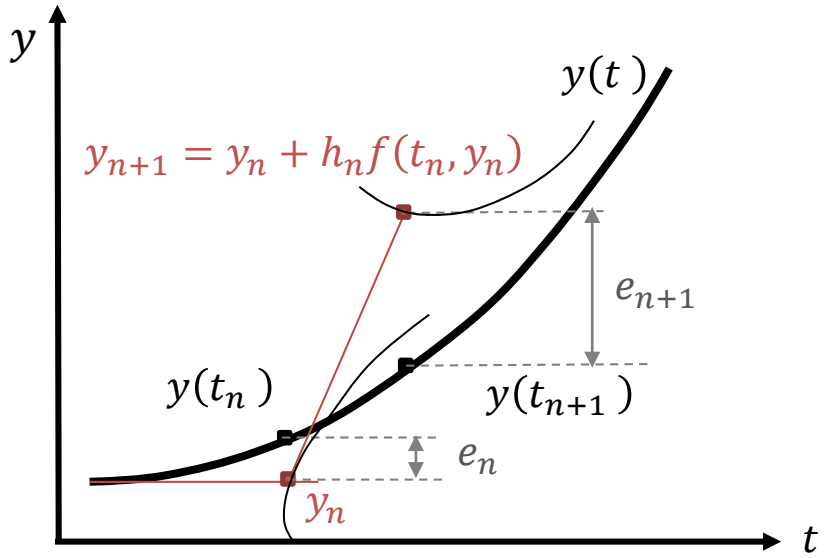
$$\frac{y_{n+1} - y_n}{h_n} = f(t_n, y_n)$$

The explicit (forward) Euler's method begins by approximating the derivative with a 1st order finite-difference

$$y_{n+1} = y_n + h_n f(t_n, y_n)$$

The **true solution** $\boldsymbol{y(t_n)}$ is different from the **numerical solution** y_n because of accumulated errors. If the integration step is too large, the difference (i.e. the error) could grow and the method "blows-up"





Adapted from: **Kee, Coltrin, Glarborg**, *Chemically Reacting Flow : Theory and Practice*, Wiley Interscience (2003)

Analysis of explicit Euler's Method (II)

$$y' = -\lambda y + g(t)$$

To explain the stability characteristics of the explicit Euler's method, we consider a **model problem**

$$\frac{y_{n+1} - y_n}{h_n} = -\lambda y_n + g(t_n) \qquad y_{n+1} = y_n + h_n[-\lambda y_n + g(t_n)]$$

Obviously the exact solution always satisfy this equation $y'(t_n) = -\lambda y(t_n) + g(t_n)$

$$y'(t_n) + \frac{y(t_{n+1}) - y(t_n)}{h_n} = -\lambda y(t_n) + g(t_n) + \frac{y(t_{n+1}) - y(t_n)}{h_n}$$

$$y(t_{n+1}) = y(t_n) + h_n[-\lambda y(t_n) + g(t_n)] + [y(t_{n+1}) - y(t_n) - h_n y'(t_n)]$$

This terms represent the explicit Euler's algorithm operating on the exact solution

Measure of the local truncation error

Analysis of explicit Euler's Method (III)

accuracy

The local truncation error can be identified through a Taylor series expansion of the solution about time t_n

$$y(t_{n+1}) = y(t_n) + h_n y'(t_n) + \frac{h_n^2}{2} y''(t_n) + O(h_n^3)$$

$$y(t_{n+1}) - y(t_n) - h_n y'(t_n) = \frac{h_n^2}{2} y''(t_n) + O(h_n^3)$$

local truncation error

$$d_n = \frac{h_n^2}{2} y''(t_n) + O(h_n^3)$$

The accuracy is controlled to within a certain tolerance through the local truncation error:

$$\left\| \frac{h_n^2}{2} y'' \right\| \leq \varepsilon \qquad h_n \leq \sqrt{2\varepsilon / \|y''\|}$$

stability

global error $e_n = y_n - y(t_n)$

$$e_{n+1} = (1 - h_n \lambda) e_n + d_n$$

Method conditionally stable

It is evident that any numerical errors will be amplified, unless:

$$|1 - h_n \lambda| \leq 1 \quad \Rightarrow \quad h_n \leq \frac{2}{\lambda}$$

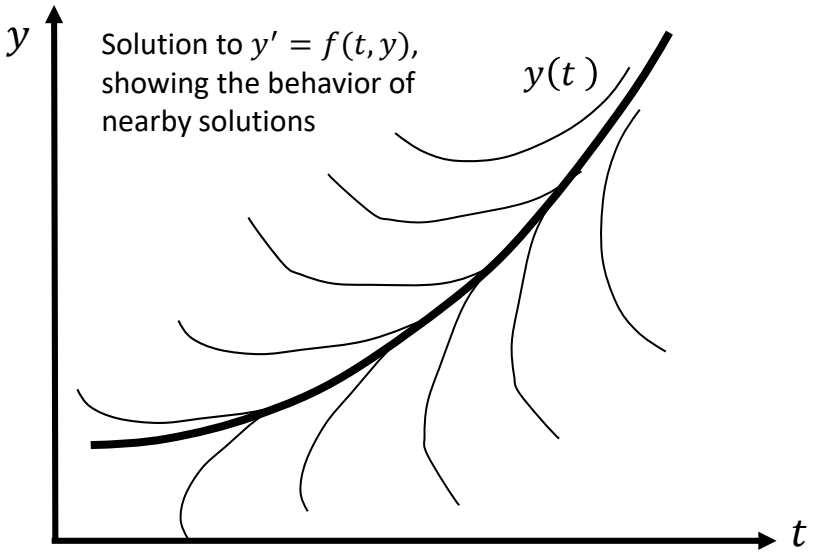
Analysis of implicit Euler's Method (I)

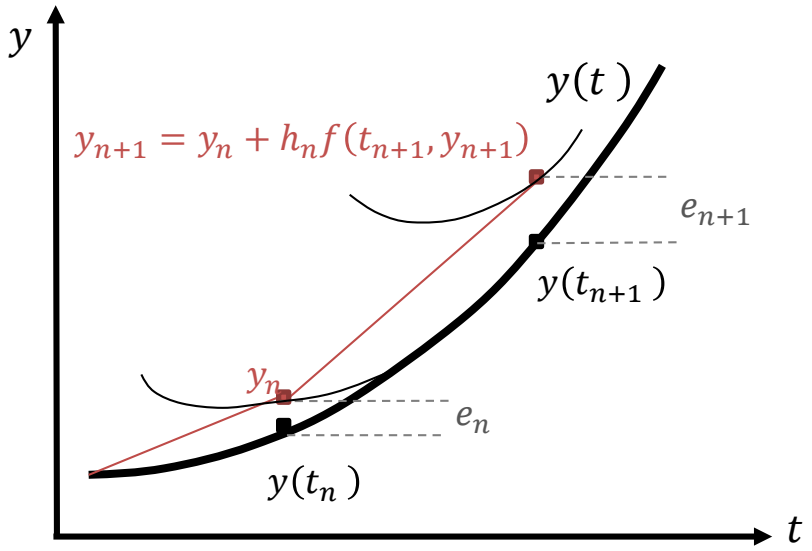
$$\frac{y_{n+1} - y_n}{h_n} = f(t_{n+1}, y_{n+1})$$

$$y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1})$$

The implicit (backward) Euler's method begins by approximating the derivative with a 1st order finite-difference

Despite the complication related to the solution of a non/linear system of equations at each time step, the benefit of the implicit method lies in its excellent stability properties.





Adapted from: **Kee, Coltrin, Glarborg**, *Chemically Reacting Flow : Theory and Practice*, Wiley Interscience (2003)

Analysis of implicit Euler's Method (II)

$$y' = -\lambda y + g(t)$$

To explain the stability characteristics of the implicit Euler's method, we consider a model problem

$$\frac{y_{n+1} - y_n}{h_n} = -\lambda y_{n+1} + g(t_{n+1}) \qquad y_{n+1} = y_n + h_n[-\lambda y_{n+1} + g(t_{n+1})]$$

Obviously the exact solution always satisfy this equation

$$y'(t_n) = -\lambda y(t_n) + g(t_n)$$

$$y'(t_n) + \frac{y(t_{n+1}) - y(t_n)}{h_n} = -\lambda y(t_{n+1}) + g(t_n + 1) + \frac{y(t_{n+1}) - y(t_n)}{h_n}$$

$$y(t_{n+1}) = y(t_n) + h_n[-\lambda y(t_{n+1}) + g(t_{n+1})] + [y(t_{n+1}) - y(t_n) - h_n y'(t_{n+1})]$$

This terms represent the implicit Euler's algorithm operating on the exact solution

Measure of the local truncation error

Analysis of implicit Euler's Method (III)

accuracy

The local truncation error can be identified through a Taylor series expansion of the solution about time t_{n+1} (in the negative t direction)

$$y(t_n) = y(t_{n+1}) - h_n y'(t_{n+1}) + \frac{h_n^2}{2} y''(t_{n+1}) + O(h_n^3)$$

$$-[y(t_{n+1}) - y(t_n) - h_n y'(t_{n+1})] = \frac{h_n^2}{2} y''(t_n) + O(h_n^3)$$

local truncation error

$$d_n = \frac{h_n^2}{2} y''(t_n) + O(h_n^3)$$

The accuracy is controlled to within a certain tolerance through the local truncation error:

$$\left\| \frac{h_n^2}{2} y'' \right\| \leq \varepsilon \qquad h_n \leq \sqrt{2\varepsilon / \|y''\|}$$

stability

global error $e_n = y_n - y(t_n)$

$$e_{n+1}(1 + h_n \lambda) = e_n + d_n$$

Method unconditionally stable

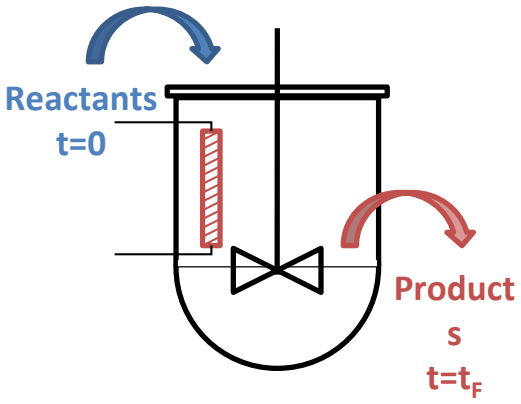
It is evident that any numerical errors will be amplified, unless:

$$\left| \frac{1}{1 + h_n \lambda} \right| \leq 1$$

This condition is satisfied for every integration step!

An example: the isothermal batch reactor (I)

Batch reactor with fixed volume and temperature



$A \rightarrow 2B$	$R_1 = k_1 C_A$
$B \rightarrow C$	$R_2 = k_2 C_B$

Species equations

$$\begin{cases} \dfrac{dC_A}{dt} = -k_1 C_A \\ \dfrac{dC_B}{dt} = 2k_1 C_A - k_2 C_B \end{cases}$$

Initial conditions

$$\begin{cases} C_A(t=0) = C_A^0 \\ C_B(t=0) = C_B^0 \end{cases}$$

$$\frac{d}{dt}\begin{bmatrix} C_A \\ C_B \end{bmatrix} = \begin{bmatrix} -k_1 & 0 \\ 2k_1 & -k_2 \end{bmatrix}\begin{bmatrix} C_A \\ C_B \end{bmatrix} = \mathbf{J}\begin{bmatrix} C_A \\ C_B \end{bmatrix}$$

System of ordinary differential equations with initial conditions

An example: the isothermal batch reactor (II)

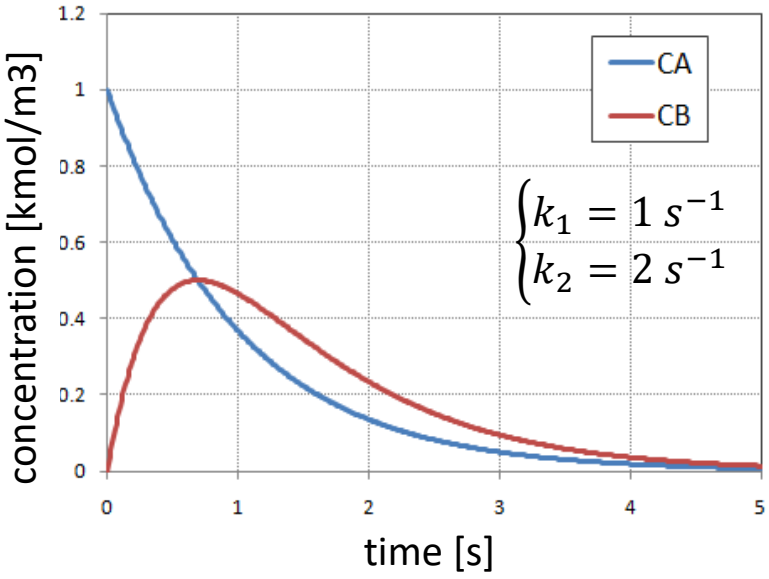
Analytical solution

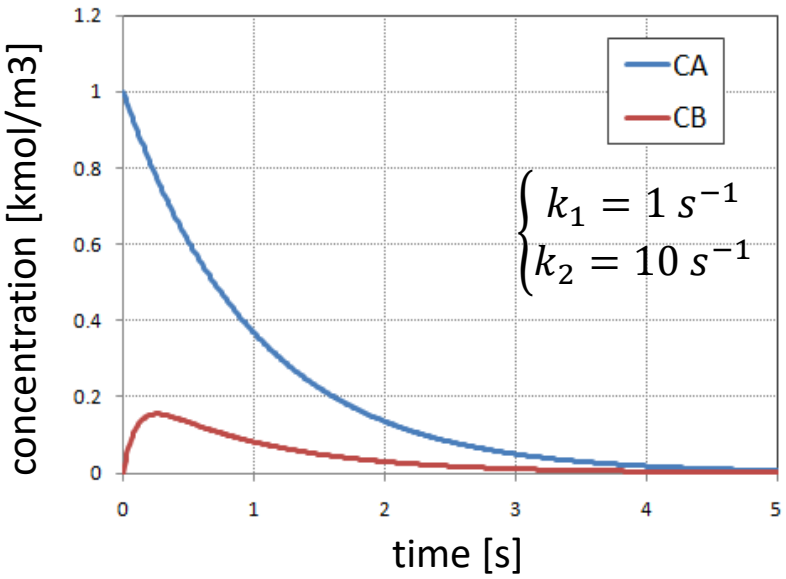
$$\begin{cases} C_A = C_A^0 \mathrm{e}^{-k_1 t} \\ C_B = \dfrac{2k_1 C_A^0}{k_2 - k_1} \mathrm{e}^{-k_1 t} - \left(\dfrac{2k_1 C_A^0}{k_2 - k_1} - C_B^0 \right) \mathrm{e}^{-k_2 t} \end{cases}$$

Eigenvalues

$$\boldsymbol{\lambda} = \begin{bmatrix} -k_1 \\ -k_2 \end{bmatrix}$$

Initial conditions $\begin{cases} C_A = 1\ kmol/m^3 \\ C_B = 0 \end{cases}$





An example: the isothermal batch reactor (II)

$$\frac{y_{n+1} - y_n}{h_n} = f(t_n)$$ **Explicit Euler's method**

$$\begin{cases} C_A^{n+1} = (1 - h_n k_1) C_A^n \\ C_B^{n+1} = 2hk_1 C_A^n + (1 - h_n k_2) C_B^{n+1} \end{cases}$$

$$\mathbf{C}_{n+1} = (\mathbf{I} + h_n \mathbf{J}) \mathbf{C}_n$$

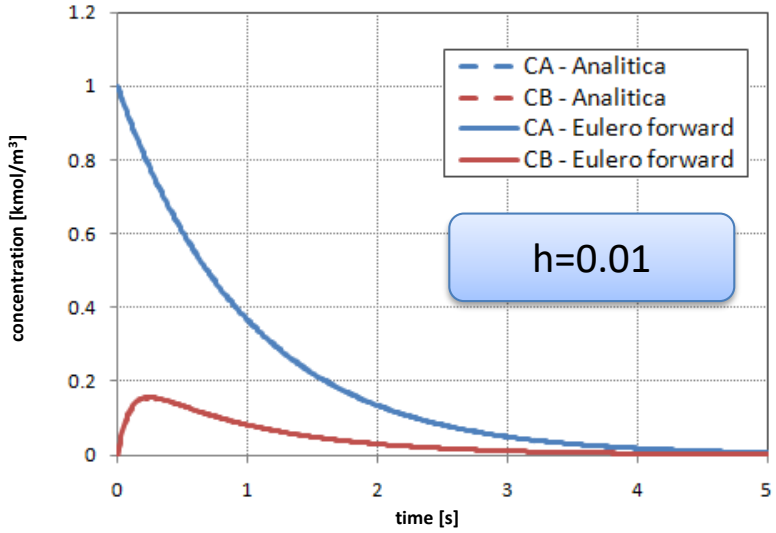
Jacobian matrix $\mathbf{J} = \begin{bmatrix} -k_1 & 0 \\ 2k_1 & -k_2 \end{bmatrix}$

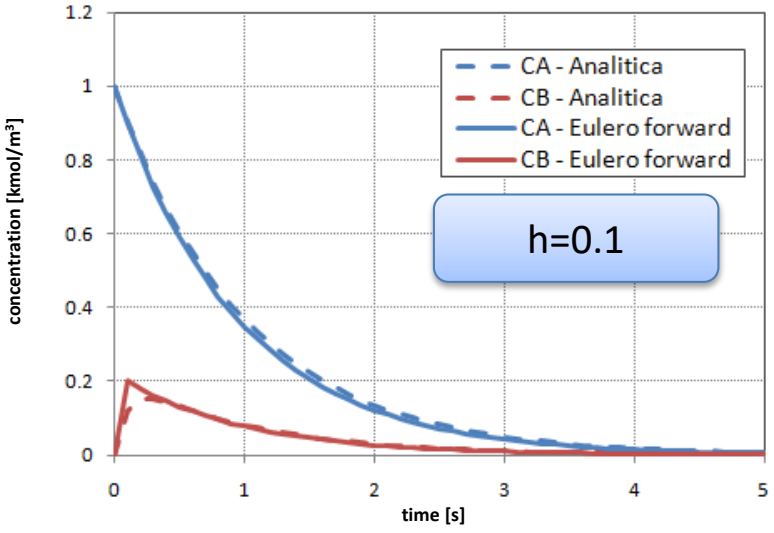
$$\begin{bmatrix} C_A^{n+1} \\ C_B^{n+1} \end{bmatrix} = \begin{bmatrix} 1 - hk_1 & 0 \\ 2hk_1 & 1 - hk_2 \end{bmatrix} \begin{bmatrix} C_A^n \\ C_B^n \end{bmatrix}$$

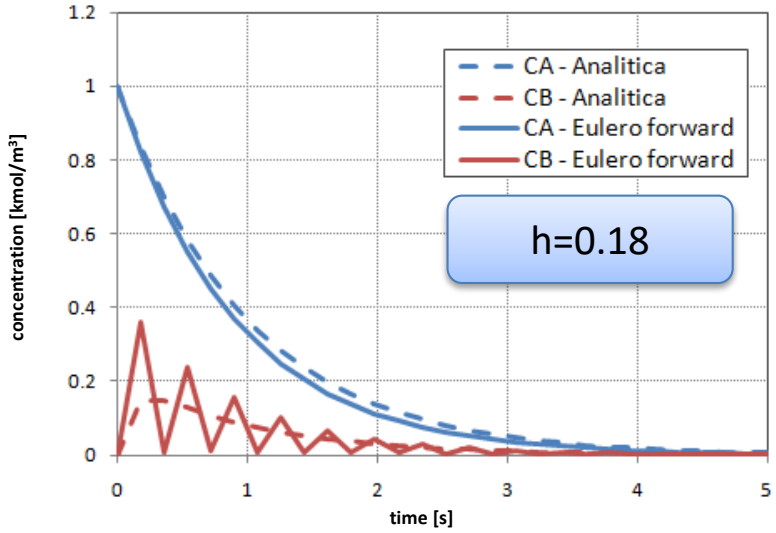
Stability condition: $h_n < h_{\max} = \frac{2}{|\lambda|_{\max}}$

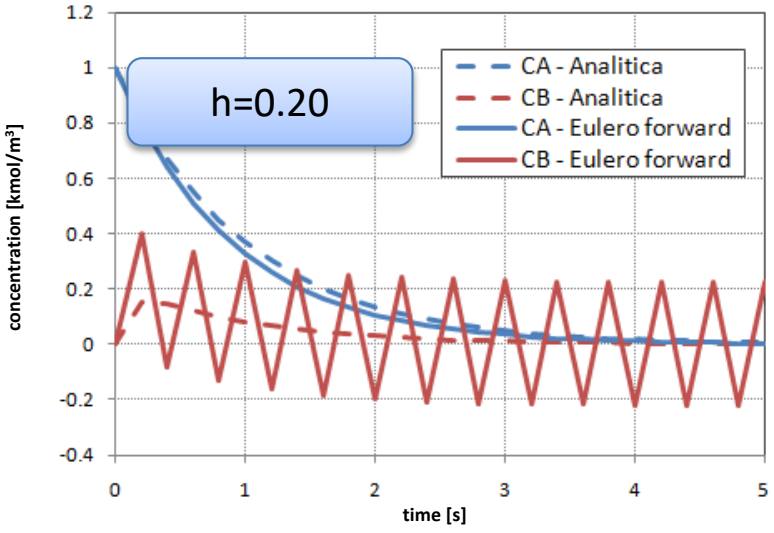
For this example $h_{max} = 0.20$

An example: the isothermal batch reactor (IV)









An example: the isothermal batch reactor (V)

$$\frac{y_{n+1} - y_n}{h_n} = f(t_{n+1})$$

Implicit Euler's method

$$\begin{cases} (1 + hk_1)C_A^{n+1} = C_A^n \\ -2hk_1 C_B^{n+1} + (1 + hk_2)C_B^{n+1} = 2hk_1 C_A^n + (1 - hk_2)C_B^{n+1} \end{cases}$$

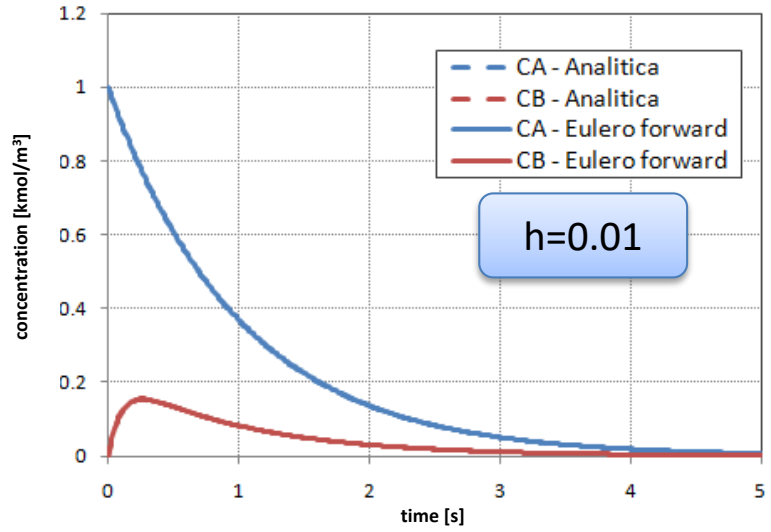
$$(\mathbf{I} - h_n\mathbf{J})\mathbf{C}_{n+1} = \mathbf{C}_n$$

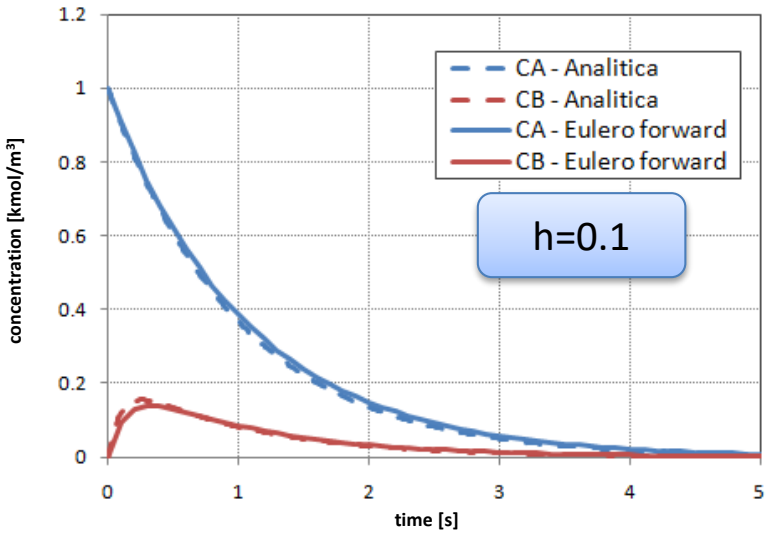
Jacobian matrix $\mathbf{J} = \begin{bmatrix} -k_1 & 0 \\ 2k_1 & -k_2 \end{bmatrix}$

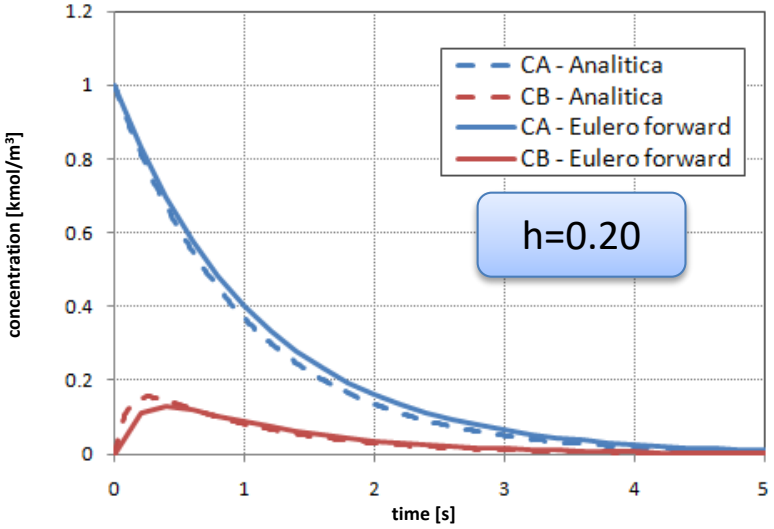
$$\begin{bmatrix} 1 + hk_1 & 0 \\ -2hk_1 & 1 + hk_2 \end{bmatrix} \begin{bmatrix} C_A^{n+1} \\ C_B^{n+1} \end{bmatrix} = \begin{bmatrix} C_A^n \\ C_B^n \end{bmatrix}$$

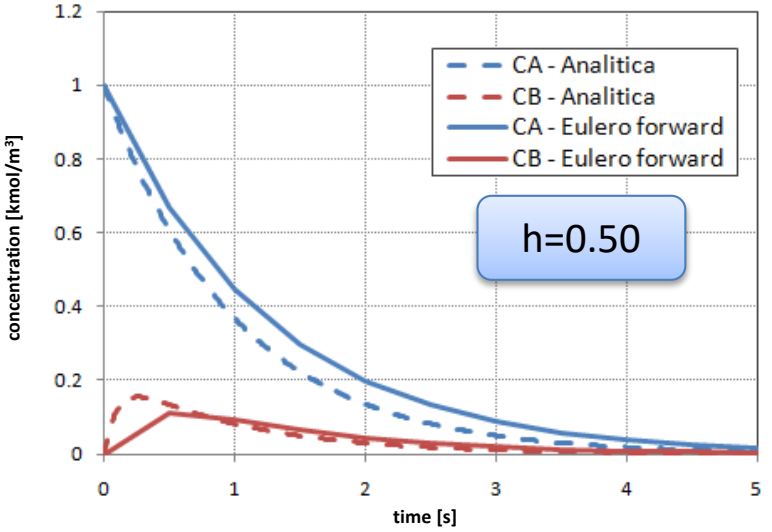
The implicit Euler's method is **unconditionally stable**. No stability condition is required

An example: the isothermal batch reactor (VI)









Multi-steps algorithms

Adams-Bashfort (II order)

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h_n}{2}[3\mathbf{f}_n - \mathbf{f}_{n-1}]$$

It is an explicit method, since the no information at time t_{n+1} (or larger) is needed

Trapezoidal method (II order)

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h_n}{2}[\mathbf{f}_{n+1} + \mathbf{f}_n]$$

It is an implicit method, since the information at time t_{n+1} is required

Adams-Moulton (III order)

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h_n}{12}[5\mathbf{f}_{n+1} + 8\mathbf{f}_n - \mathbf{f}_{n+1}]$$

It is an implicit method, since the information at time t_{n+1} is required

Trapezoidal method (Cranck-Nicholson) (I)

$$\frac{y_{n+1}-y_n}{h_n}=\frac{f(t_{n+1})+f(t_n)}{2}$$

Mid-point rule (Cranck-Nicholson method)

$$\begin{cases}(1+hk_1)C_A^{n+1}=C_A^n\\ -2hk_1C_B^{n+1}+(1+hk_2)C_B^{n+1}=2hk_1C_A^n+(1-hk_2)C_B^{n+1}\end{cases}$$

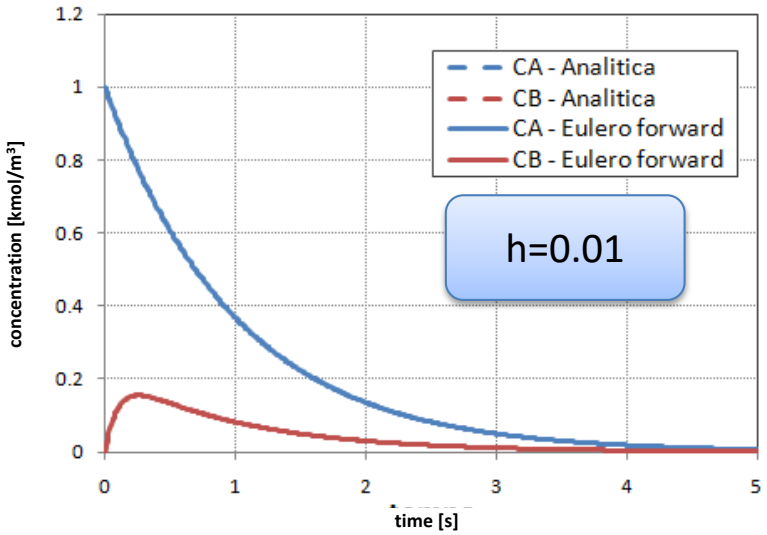
$$\left(\mathbf{I}-\frac{h_n}{2}\mathbf{J}\right)\mathbf{C}_{n+1}=\left(\mathbf{I}+\frac{h_n}{2}\mathbf{J}\right)\mathbf{C}_n$$

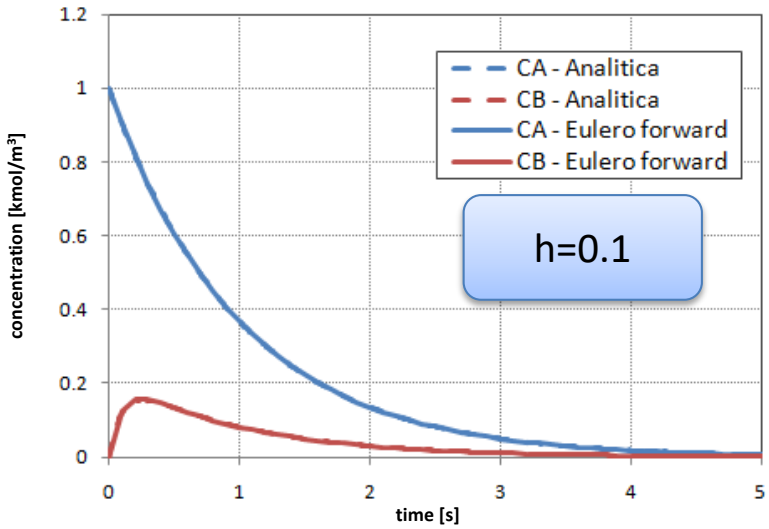
Jacobian matrix $\mathbf{J}=\begin{bmatrix}-k_1 & 0\\ 2k_1 & -k_2\end{bmatrix}$

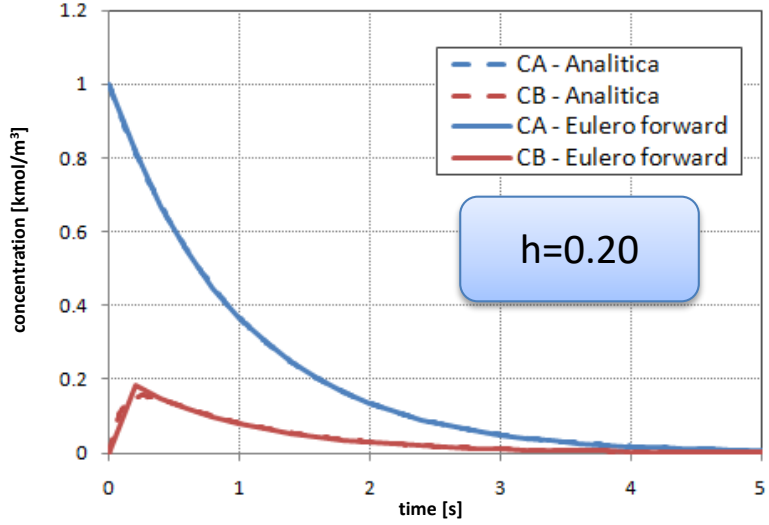
$$\begin{bmatrix}1+\frac{h}{2}k_1 & 0\\ -2\frac{h}{2}k_1 & 1+\frac{h}{2}k_2\end{bmatrix}\begin{bmatrix}C_A^{n+1}\\ C_B^{n+1}\end{bmatrix}=\begin{bmatrix}1-\frac{h}{2}k_1 & 0\\ 2\frac{h}{2}k_1 & 1-\frac{h}{2}k_2\end{bmatrix}\begin{bmatrix}C_A^{n}\\ C_B^{n}\end{bmatrix}$$

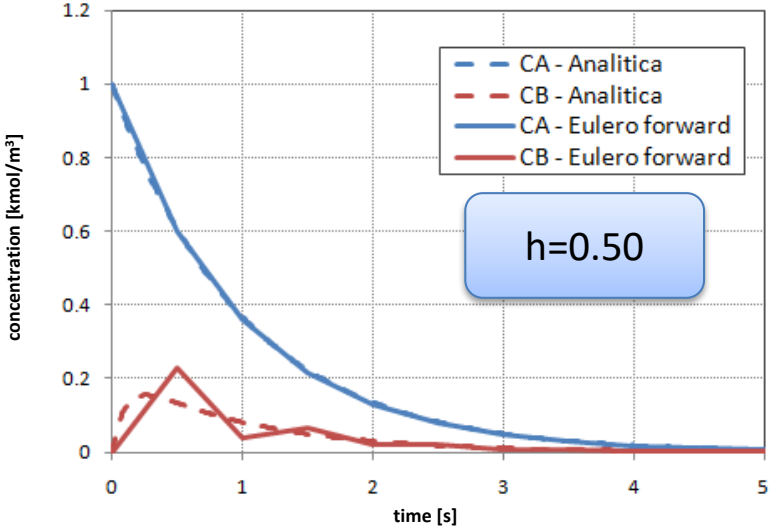
The mid-point (or trapezoidal) method is **unconditionally stable**. No stability condition is required, as for the implicit Euler method

Trapezoidal method (Cranck-Nicholson) (II)









Runge-Kutta Methods

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \sum_{i=1}^{S} b_i \mathbf{k}_i$$

S = number of stages

- Only information associated to the last integration step is used
- Very low memory requirements
- Very simple to use and implement
- Extremely efficient for very large systems of equations

$$\mathbf{k}_i = h_n \mathbf{f}\left(t_n + c_i h_n, \mathbf{y}^n + \Delta t \sum_{j=1}^{S} b_{isj} \mathbf{k}_j \right)$$

Model parameters

$$\{a_{ij}\}, \{b_i\}, \{c_i\}$$

For the implicit methods, the coefficient matrix a_{ij} is not necessarily lower triangular

- Depending on the a_{ij} parameters, the resulting methods can be implicit or explicit
- In the implicit case, at least one of the k functions requires to be calculated implicitly. The computational time increases, but usually the stability features improve
- The implicit methods can be applied (after some proper improvements) for solving stiff problems

Ascher U., Petzold L., *"Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations"*, SIAM (1997)

IVth order Runge-Kutta method

1. Predictor: Explicit Euler

$$\mathbf{y}_{n+1/2}^{pred} = \boldsymbol{y}_n + \mathbf{f}(\boldsymbol{y}_n)\frac{h_n}{2}$$

2. Corrector: Explicit Euler

$$\mathbf{y}_{n+1/2}^{corr} = \boldsymbol{y}_n + \mathbf{f}\left(\mathbf{y}_{n+1/2}^{pred}\right)\frac{h_n}{2}$$

3. Midpoint rule

$$\mathbf{y}_{n+1}^{*} = \boldsymbol{y}_n + \mathbf{f}\left(\mathbf{y}_{n+1/2}^{corr}\right)h_n$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h_n}{6}[\mathbf{k}_1 + 2\mathbf{k_2} + 2\mathbf{k}_3 + \mathbf{k}_4]$$

- ✓ $\mathbf{k}_1$ is the increment based on the slope at the beginning of the interval, using y' (explicit Euler's method)
- ✓ $\mathbf{k_2}$ is the increment based on the slope at the midpoint of the interval, using $y' + 1/2\, h_n\mathbf{k}_1$
- ✓ $\mathbf{k_3}$ is again the increment based on the slope at the midpoint, but now using $y' + 1/2\, h_n\mathbf{k}_2$
- ✓ $\mathbf{k_4}$ is the increment based on the slope at the end of the interval, using $y' + h_n\mathbf{k}_3$

4. Simpson Rule

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{\Delta t}{6}\left[\mathbf{f}(\boldsymbol{y}_n) + 2\mathbf{f}\left(\mathbf{y}_{n+1/2}^{pred}\right) + 2\mathbf{f}\left(\mathbf{y}_{n+1/2}^{corr}\right) + \mathbf{f}(\mathbf{y}_{n+1}^{*})\right]$$

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) Solution of ODE systems
   c) **Numerical methods for stiff ODE systems**
   d) The Jacobian matrix

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

Stiff equations (I)

$$\begin{cases} \dfrac{dC}{dt} = -100(C - \sin t) \\ C(t=0) = 1 \end{cases}$$

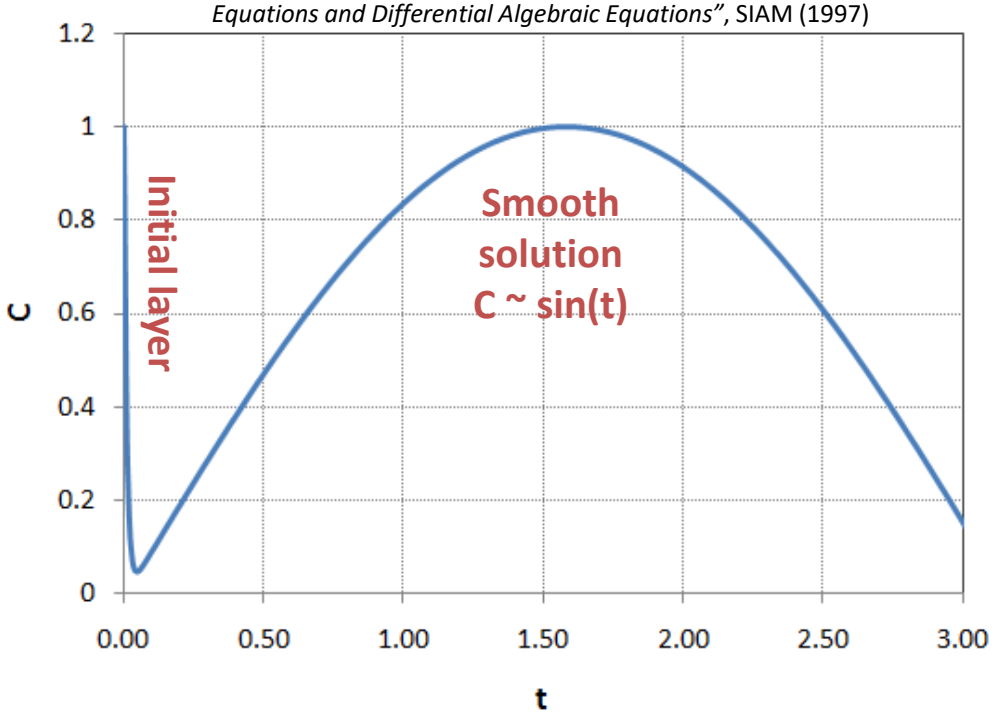
Eigenvalue: $\lambda = -100$

Explicit Euler's method:

$$h < \frac{2}{-\lambda} = 0.02$$

Even at long time, when the solution is changing very slowly, the equation itself still has a characteristic time scale that can be quite short if λ is large.

Ascher U., Petzold L., *"Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations"*, SIAM (1997)



Regardless of the value of t, the characteristic time scale of this equation is $\tau = \frac{1}{\lambda}$

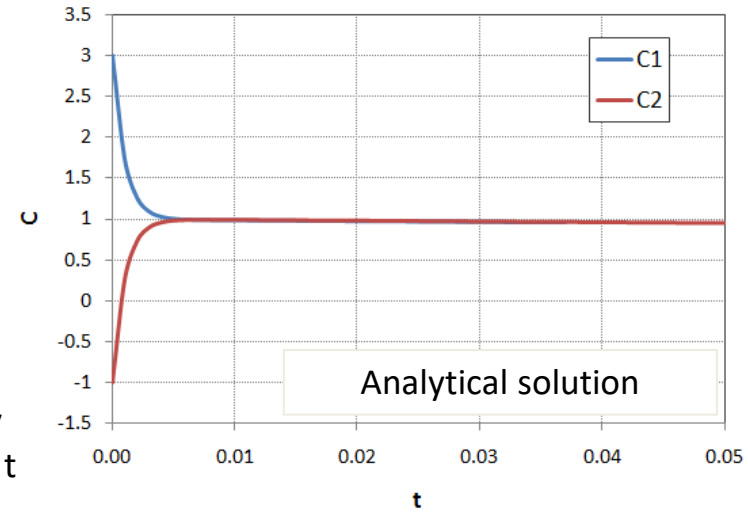
Stiffness occurs in regions where the solution is changing slowly (or not at all), yet the characteristic time scales are very small

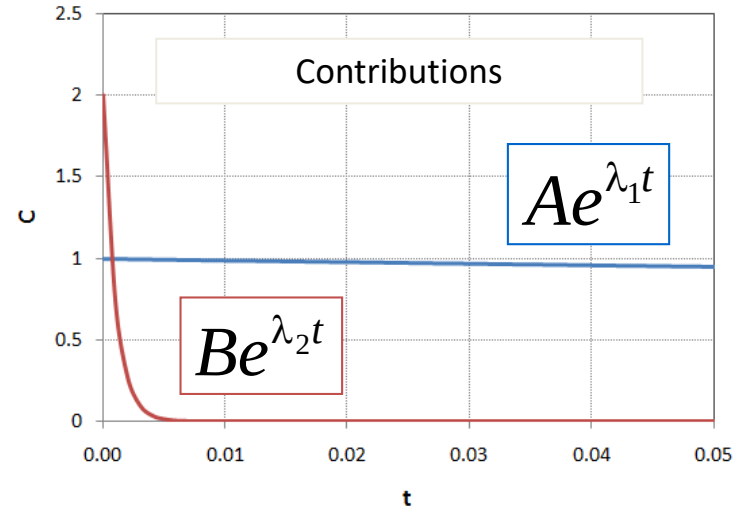
Stiff equations (II)

$$\begin{cases} \dfrac{dC_1}{dt} = \dfrac{\lambda_1 + \lambda_2}{2} C_1 + \dfrac{\lambda_1 - \lambda_2}{2} C_2 \\ \dfrac{dC_2}{dt} = \dfrac{\lambda_1 - \lambda_2}{2} C_1 + \dfrac{\lambda_1 + \lambda_2}{2} C_2 \end{cases} \qquad \begin{cases} \lambda_1 = -1 \\ \lambda_2 = -1000 \end{cases}$$

$$\begin{cases} C_1 = Ae^{\lambda_1 t} + Be^{\lambda_2 t} \\ C_2 = Ae^{\lambda_1 t} - Be^{\lambda_2 t} \end{cases}$$

This contribution is significant only for very small values of t. For larger t it is practically equally to 0 and negligible if compared to the first contribution





Explicit Euler's method

The integration step must satisfy the usual stability condition:

$$h < h_{\max} = -\frac{2}{\lambda_{\max}} = \frac{2}{1000}$$

The integration step depends on the $\lambda_{\max}$, even at large times (i.e. even when the 2nd contribution has a negligible value)
We are forced to use a very small time step to correctly account for a contribution whose value is practically equal to zero

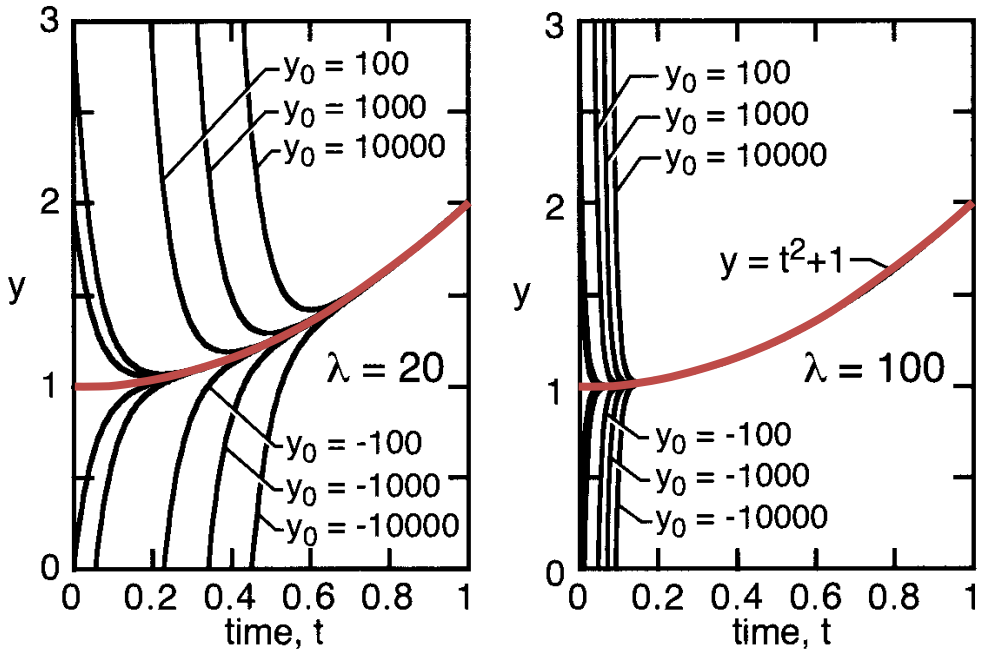
Stiff equations (III)

$$\begin{cases} \dfrac{dy}{dt} = -\lambda[y - (t^2 + 1)] + 2t \\ y(0) = y_0 \end{cases}$$

First-order linear model problem

$$y(t) = [y_0 - 1]e^{-\lambda t} + (t^2 + 1)$$

Characteristic time $\tau = \dfrac{1}{\lambda}$



The family of solutions can be thought as a manifold of solutions, all of which tend toward the slowly varying solution $\boldsymbol{y(t) = (t^2 + 1)}$

The faster the characteristic scale, the more rapidly the family of solutions tend toward the slowly varying solution

Adapted from: **Kee, Coltrin, Glarborg**, *Chemically Reacting Flow : Theory and Practice*, Wiley Interscience (2003)

Another example of stiff behavior

This below is the well-known Robertson's kinetic system:

$$\begin{cases} A \xrightarrow{0.04} B & \text{slow} \\ B + B \xrightarrow{3\cdot 10^7} C + B & \text{very fast} \\ B + C \xrightarrow{10^4} A + C & \text{fast} \end{cases}$$

Initial conditions

$$C(t=0) = (1,0,0)$$

$$\begin{cases} \dfrac{dC_A}{dt} = -0.04C_A + 10^4 C_B C_C \\ \dfrac{dC_B}{dt} = 0.04C_A - 10^4 C_B C_C - 3\cdot 10^7 C_B C_B \\ \dfrac{dC_C}{dt} = 3\cdot 10^7 C_B C_B \end{cases}$$

Very stiff ODE system!

The Robertson's example is often invoked as "sample kinetic mechanism". Note it lacks the exponential dependence on temperature in real systems and non-linearities are limited to products.

Stiffness: definition

Definition 1

A general ODE system is stiff if the ratio between the maximum and the minimum eigenvalues of the local Jacobian matrix is very large

$$\frac{|\lambda_{\max}|}{|\lambda_{\min}|} >> 1$$

Definition 2

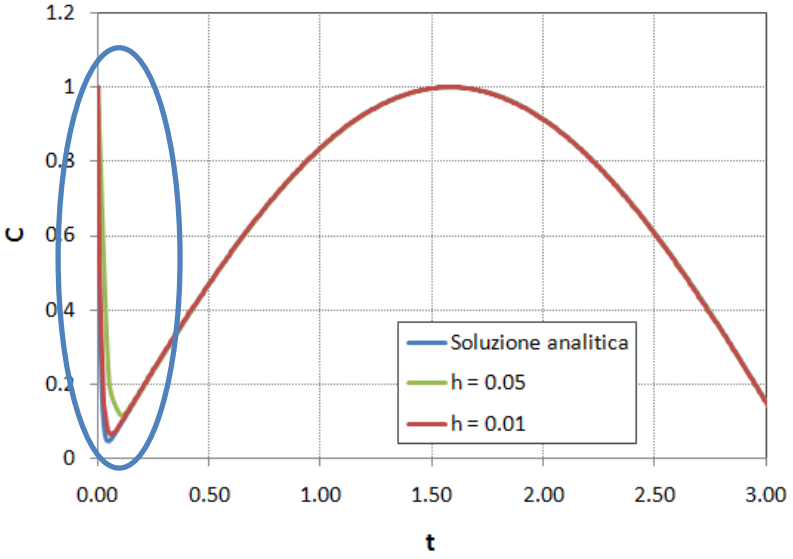
An IVP (Initial Value Problem) in some interval is stiff if the step size needed to mantain stability of the forward Euler method is much smaller than the step size required to represent the solution accurately

Physical interpretation

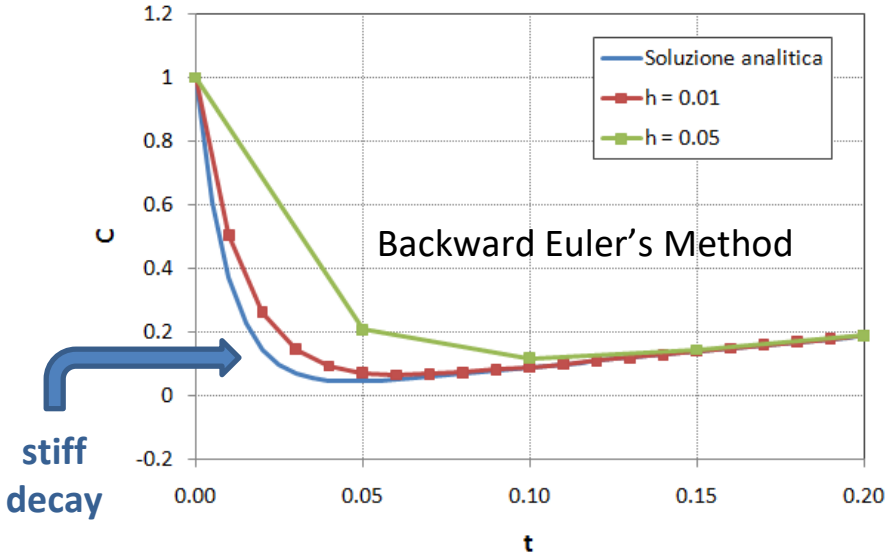
Scientists often describe stiffness in terms of multiple time scales. If the problem has widely varying time scales, and the phenomena that change on fast scales are stable, then the problem is stiff

Need of implicit solvers for stiff problems

- Implicit methods usually have better features in terms of stability if compared to the explicit methods (i.e. they can use larger integration time steps, without blowing-up)



- A system of non linear algebraic equations must be solved at each time step
- A larger amount of memory is required with respect to the explicit methods
- Not all the implicit methods are appropriate for solving stiff ODE systems. In most cases, in order to be used efficiently, they have to be properly reformulated



Stiff decay

The stiff-decay is the ability to skip fine-level (i.e. rapidly changing) solution details and still maintaining a reasonable description of the solution on a coarse level in the very stiff case.

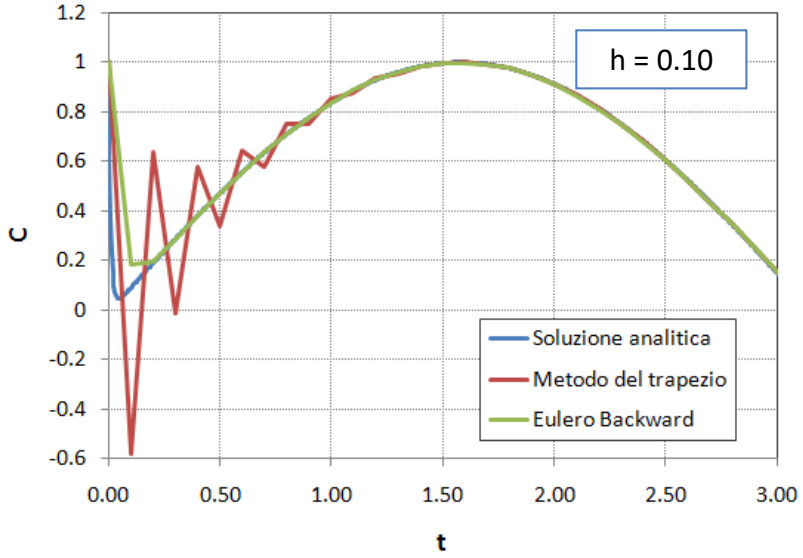
Backward Euler's Method (**stiff decay)**

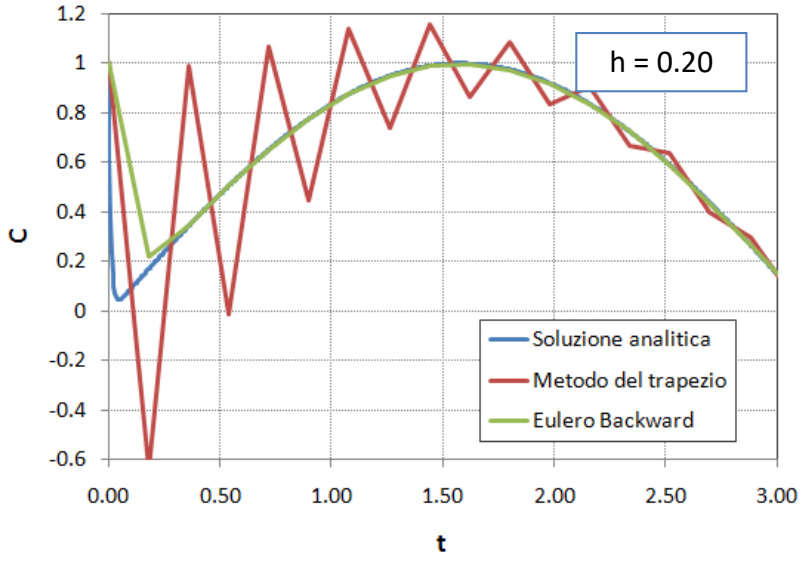
The initial layer is poorly approximated, but still the solution is qualitatively recovered where it varies slowly

Trapezoidal method (**no stiff decay)**

Any solution detail must be resolved even if only a coarse picture of the solution is desired, because the fast mode components of local errors get propagated, almost un-damped, throughout the integration interval corresponding to the initial layer

To apply the trapezoidal rule intelligently for this example, we must use a small step size through the initial layer, than the step size can become larger





Backward Differentiation Formulas (BDF)

These are the most common methods for solving stiff equations

$$\mathbf{y}_{n+k} = h_n \beta_k \mathbf{f}_{n+k} - \sum_{j=0}^{k-1} \alpha_j \, \mathbf{y}_{n+j}$$

They are linear multistep methods, where $\alpha_0 \neq 0$, and $\beta_k \neq 0$ and the order is equal to the step number k

- ✓ The first-order method is the backward (implicit) Euler method.
- ✓ The first, second and third order methods have very good stability properties
- ✓ The higher-order methods may give poor results for systems with eigenvalues near the imaginary y-axis

k	β_k	α_0	α_1	α_2	α_3	α_4	α_5
1	1	-1					
2	$2/3$	$1/3$	$-4/3$				
3	$6/11$	$-2/11$	$9/11$	$-18/11$			
4	$12/25$	$3/25$	$-16/25$	$36/25$	$-48/25$		
5	$60/137$	$-12/137$	$75/137$	$-200/137$	$300/137$	$-300/137$	
6	$600/147$	$10/147$	$-72/147$	$225/147$	$-400/147$	$450/147$	$-360/147$

Oran, Boris, *Numerical simulation of reactive flows*, Cambridge University Press (2001)

Other methods for stiff ODE

Exponential Methods (Lambert, 1973)
A particular form of interpolating function is adopted with free parameters, determined by requiring the interpolant to satisfy certain conditions on the approximate solutions and their derivatives. Exponential methods can be at least comparable in speed and accuracy to the BDF for stiff ODEs

Asymptotic Methods (Young and Boris, 1977)
Are problem-dependent and requires certain features by the equations to be integrated. If they can be applied, usually they are very fast since they do not require any Jacobian evaluation (and factorization) and are very easy to implement and manage

Quasi-Steady-State Methods (Mott, 1999)
They are a sort of extension and improvement of the classic Asymptotic Methods, with better properties in terms of stability and accuracy.

Implicit Extrapolation Methods (Lindberg, 1973)
They are similar to the BDF methods, but they do not perform as well

Low-order, Single-Step, Extrapolation Methods (Wagoner, 1969)
These methods are very stable for very stiff ODE systems.

BDF methods and non linear systems (I)

$$\mathbf{y}_{n+k} = h_n \beta_k \mathbf{f}_{n+k} - \sum_{j=0}^{k-1} \alpha_j \mathbf{y}_{n+j}$$

General BDF method of order k

Every implicit method requires the solution of a system of non-linear algeraic equations at each integration step

BDF method of order 1 (backward Euler method)

$$\mathbf{y}_{n+1} = h_n \mathbf{f}_{n+1} + \mathbf{y}_n$$

$$\mathbf{F}(t_{n+1}, \mathbf{y}_{n+1}) \equiv \mathbf{y}_{n+1} - \mathbf{y}_n - h_n \mathbf{f}_{n+1} = \mathbf{0}$$

BDF method of order 2

$$\mathbf{y}_{n+1} = h_n \frac{2}{3} \mathbf{f}_{n+1} - \frac{1}{3} \mathbf{y}_{n-1} + \frac{4}{3} \mathbf{y}_n$$

$$\mathbf{F}(t_{n+1}, \mathbf{y}_{n+1}) \equiv \mathbf{y}_{n+1} - \frac{4}{3} \mathbf{y}_n + \frac{1}{3} \mathbf{y}_{n-1} - \frac{2}{3} h_n \mathbf{f}_{n+1} = \mathbf{0}$$

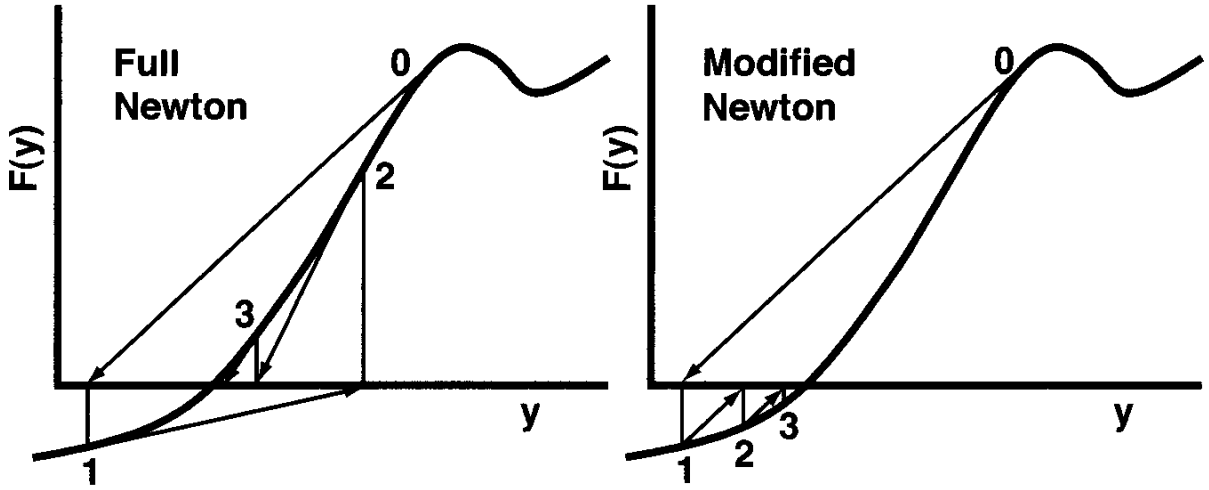
Scalar Newton's Algorithm

$$F(y) = 0 \qquad F\left(y^{(m+1)}\right) \approx F\left(y^{(m)}\right) + \left.\frac{\partial F}{\partial y}\right|_{y^{(m)}} \left(y^{(m+1)} - y^{(m)}\right) = 0$$

$$\left.\frac{\partial F}{\partial y}\right|_{y^{(m)}} \Delta y^{(m)} = -F\left(y^{(m)}\right)$$

$$\Delta y^{(m)} \equiv \left(y^{(m+1)} - y^{(m)}\right)$$

- The iteration begins at some guess at the solution
- A zero is found of the locally linear problem
- Based on the y from the linear approximation, point 1 is found
- The slope at point 1 is then used to form a new linear problem, to obtain point 2
- The process continues until the solution is found



Adapted from: **Kee, Coltrin, Glarborg**, *Chemically Reacting Flow : Theory and Practice*, Wiley Interscience (2003)

Newton's Algorithm for Algebraic Systems (I)

$\mathbf{F}(\mathbf{y}) = \mathbf{0}$ System of N non-linear algebraic equations $\mathbf{F}$ in N unknowns $\mathbf{y}$

$$\mathbf{F}\left(\mathbf{y}^{(m+1)}\right) \approx \mathbf{F}\left(\mathbf{y}^{(m)}\right) + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{\mathbf{y}^{(m)}} \left(\mathbf{y}^{(m+1)} - \mathbf{y}^{(m)}\right) = \mathbf{0}$$

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{\mathbf{y}^{(m)}} \Delta\mathbf{y}^{(m)} = -\mathbf{F}\left(\mathbf{y}^{(m)}\right)$$

- The procedure begins with a guess at the solution vector $\mathbf{y}^{(0)}$ and continues until the correction $\Delta\mathbf{y}^{(m)}$ becomes negligibly small
- The Newton's method converges very rapidly (quadratically) if the initial iterate lies within its domain of convergence

$$\mathbf{J}^{(m)}\Delta\mathbf{y}^{(m)} = -\mathbf{F}\left(\mathbf{y}^{(m)}\right)$$

$$\mathbf{J}^{(m)} \equiv \left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{\mathbf{y}^{(m)}}$$

Jacobian at the m iteration

$$\Delta\mathbf{y}^{(m)} \equiv \left(\mathbf{y}^{(m+1)} - \mathbf{y}^{(m)}\right)$$

Correction vector at the m iteration

Newton's Algorithm for Algebraic Systems (II)

One method of improving the convergence properties of the Newton's method is to implement a **damping strategy**

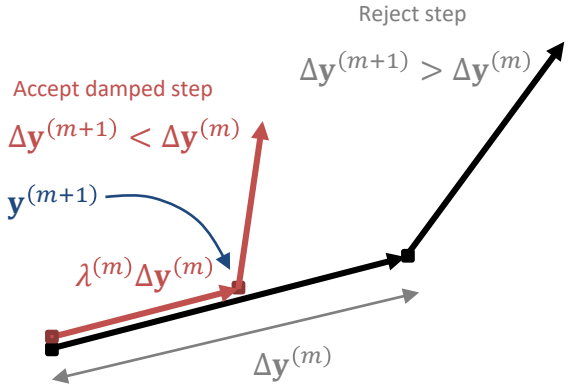
$$\mathbf{J}^{(m)}\Delta\mathbf{y}^{(m)} = -\lambda^{(m)}\mathbf{F}\left(\mathbf{y}^{(m)}\right)$$

$0 \leq \lambda^{(m)} \leq 1$ Damping parameter (to be chosen as large as possible)

1. Each component of the new $\mathbf{y}^{(m+1)}$ must stay within a trust region where the solution is known to exist (for example mass fractions must be greater than 0 and less than unity)

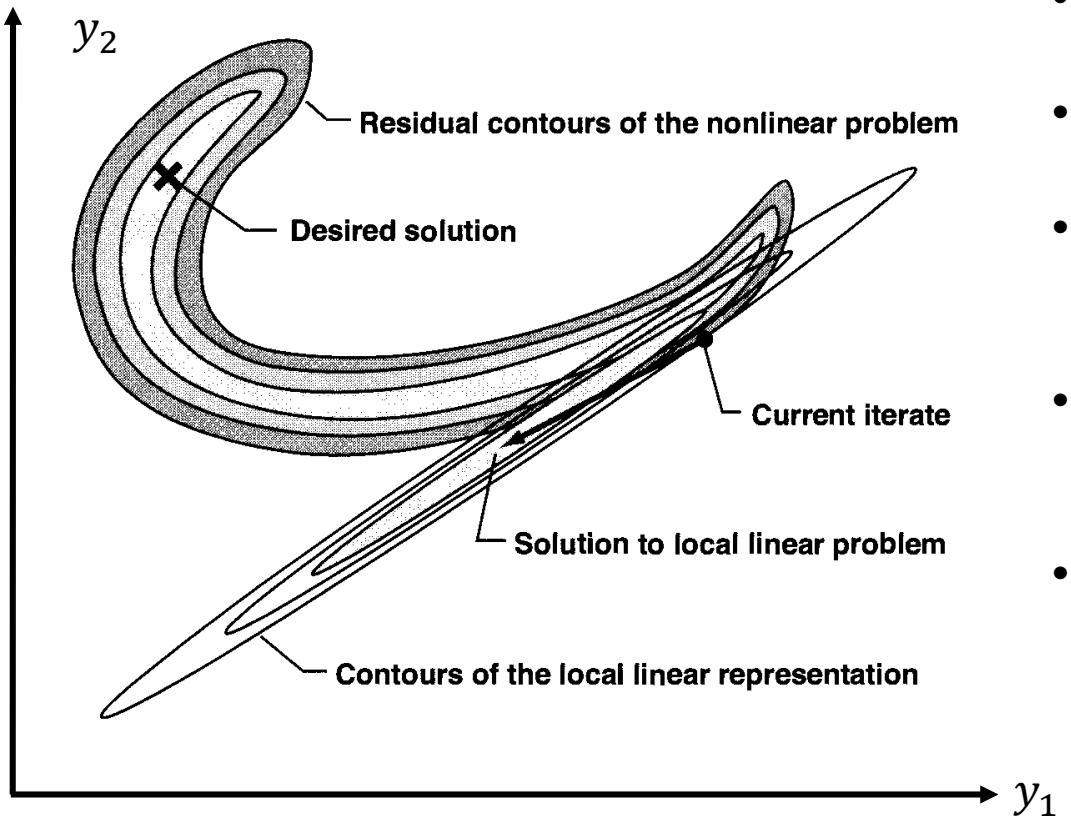
2. The new $\mathbf{y}^{(m+1)}$ is not accepted unless a norm of the next correction vector $\Delta\mathbf{y}^{(m+1)}$ is smaller than the current correction vector $\Delta\mathbf{y}^{(m)}$

$$\left\| \left[\mathbf{J}^{(m)}\right]^{-1}\mathbf{F}\left(\mathbf{y}^{(m+1)}\right) \right\| < \left\| \left[\mathbf{J}^{(m)}\right]^{-1}\mathbf{F}\left(\mathbf{y}^{(m)}\right) \right\|$$



Newton's Algorithm for Algebraic Systems (III)

Modified Newton's methods try to reuse as much as possible the same Jacobian matrix for several iterations



- long, narrow valley, greatly elongated, with very steep walls.
- non linearity is represented by the curvature of the valley.
- numerical solution to the locally linear problem at every iteration is highly sensitive to small errors
- scale disparity associated to the condition number of the Jacobian matrix
- as the condition number increases, the Jacobian becomes increasingly ill-conditioned

Adapted from: **Kee, Coltrin, Glarborg**, *Chemically Reacting Flow : Theory and Practice*, Wiley Interscience (2003)

BDF methods and non linear systems

$$\mathbf{F}(t_{n+1}, \mathbf{y}_{n+1}) \equiv \mathbf{y}_{n+1} - \mathbf{y}_n - h_n \mathbf{f}_{n+1} = \mathbf{0}$$

BDF method of order 1

$$\mathbf{F}(t_{n+1}, \mathbf{y}_{n+1}) \equiv \mathbf{y}_{n+1} - \frac{4}{3}\mathbf{y}_n + \frac{1}{3}\mathbf{y}_{n-1} - \frac{2}{3} h_n \mathbf{f}_{n+1} = \mathbf{0}$$

BDF method of order 2

$$\boxed{\mathbf{F}_{n+1} \equiv \mathbf{y}_{n+1} + \tilde{\mathbf{y}}_n - \sigma \mathbf{f}_{n+1} = \mathbf{0}}$$

Generic BDF method

$$\mathbf{F}_{n+1}^{(m)} + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{n+1}^{(m)} \left(\mathbf{y}_{n+1}^{(m+1)} - \mathbf{y}_{n+1}^{(m)}\right) = \mathbf{0}$$

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{n+1}^{(m)} \left(\mathbf{y}_{n+1}^{(m+1)} - \mathbf{y}_{n+1}^{(m)}\right) = -\mathbf{F}_{n+1}^{(m)}$$

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right|_{n+1}^{(m)} = \mathbf{I} - \sigma \mathbf{J}_{n+1}^{(m)}$$

$$\mathbf{F}_{n+1}^{(m)} \equiv \mathbf{y}_{n+1}^{(m)} + \tilde{\mathbf{y}}_n - \sigma \mathbf{f}_{n+1}^{(m)}$$

$$\boxed{\left(\mathbf{I} - \sigma \mathbf{J}_{n+1}^{(m)}\right) \Delta \mathbf{y}_{n+1}^{(m)} = -\left[\mathbf{y}_{n+1}^{(m)} + \tilde{\mathbf{y}}_n - \sigma \mathbf{f}_{n+1}^{(m)}\right]}$$

Linear system with N unknown

Linear system solution

$$\left(\mathbf{I} - \sigma \mathbf{J}_{n+1}^{(m)}\right) \Delta \mathbf{y}_{n+1}^{(m)} = -\left[\mathbf{y}_{n+1}^{(m)} + \tilde{\mathbf{y}}_n - \sigma \mathbf{f}_{n+1}^{(m)}\right]$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \left(\mathbf{I} - \sigma \mathbf{J}_{n+1}^{(m)}\right)$$

$$\mathbf{b} = -\left[\mathbf{y}_{n+1}^{(m)} + \tilde{\mathbf{y}}_n - \sigma \mathbf{f}_{n+1}^{(m)}\right]$$

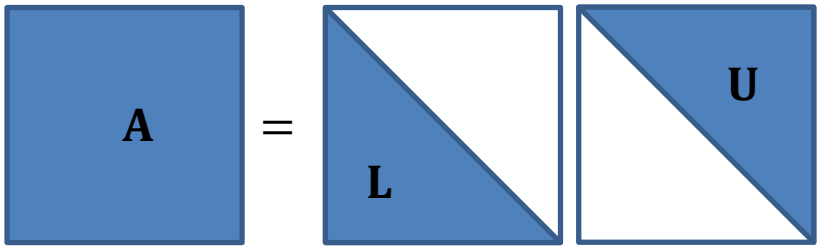
$$\mathbf{x} = \Delta \mathbf{y}_{n+1}^{(m)}$$

Matrix A is decomposed into the product of upper (U) and lower (L) triangular matrices

$$\mathbf{A} = \mathbf{LU} \qquad \mathbf{LUx} = \mathbf{b}$$

The solution is accomplished sequentially as a forward substitution followed by a back substitution

$$\mathbf{Ux} = \mathbf{z} \qquad \mathbf{Lz} = \mathbf{b}$$



The LU factorization is a relatively expensive process, with the number of operations scaling as **the cube of the size** of the matrix

Once decomposed, however, the solution for any b vector scales as **the square of the system size**

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) Solution of ODE systems
   c) Numerical methods for stiff ODE systems
   d) **The Jacobian matrix**

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

Introduction to Jacobians

Jacobians are a fundamental concept in ordinary differential equations. In the case of zero-dimensional reactors, they are key to a successful time integration strategy.

Let us consider an autonomous system of n non-linear ODEs $\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y})$. Then the "Jacobian" of the system is a $n \times n$ matrix:

$$\boldsymbol{J} = \frac{d\boldsymbol{f}}{d\boldsymbol{y}} \qquad \text{or} \qquad J_{ik} = \frac{df_i}{dy_k}$$

Note that the Jacobian of a non-linear system of equation is not constant, rather changes with the solution vector y.

Often, we shall look at the eigenvalues of $\boldsymbol{J}$ and shall denote with $\sigma(\boldsymbol{J}) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$ the "spectrum" of the Jacobian.

The Jacobian matrix: evaluation

Pseudo-code to numerically calculate the Jacobian matrix

$Given\ \boldsymbol{y}$

$\dot{\boldsymbol{\Omega}}' = calculate(\boldsymbol{y}')$

N_S cycles

for j=1:NS

$y_j' = y_j + \varepsilon_j$

$\dot{\boldsymbol{\Omega}}' = calculate(\boldsymbol{y}')$

for i=1:NS

N_R exp

$$J_{i,j} = \frac{\dot{\Omega}_i' - \dot{\Omega}_i}{\varepsilon_j}$$

end

end

Cost of numerical Jacobian: $N_S \cdot N_R$ exponentiations

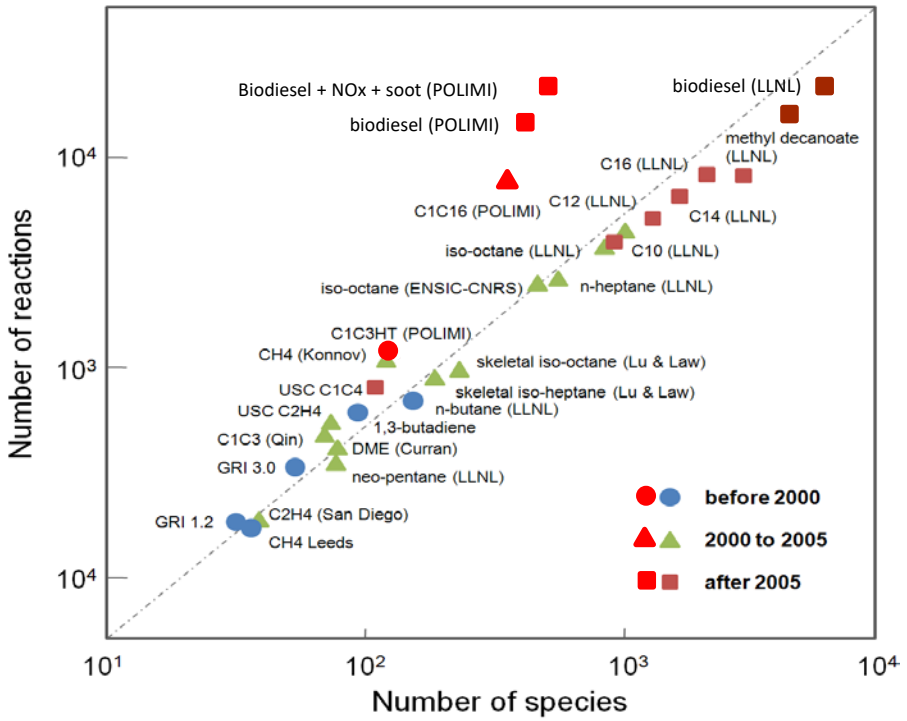
Can be very time consuming (even for 0D systems)

Example

- Auto-ignition of 2-methyl alkanes (LLNL)
- Number of species: $N_S \sim 8000$
- Number of reactions: $N_R \sim 30000$
- Typical Jacobian evaluations ~ 100
- Number of exponentiations ~ $>10^{10}$
- Cost single exponentiation: 20-50 flops
- Total number of flops ~ 10^{12}
- Peak speed of this laptop: ~10 Gflops
- Time for Jacobian evaluation: ~100 s

Time complexity of Jacobian operations

Typically: $N_R \sim 5 \div 20 N_S$



Adapted from: **T.F. Lu, C.K. Law**, *Toward accommodating realistic fuel chemistry in large-scale computations,* Progress in Energy and Combustion Science, 35, p. 192–215 (2009)

Global computational cost

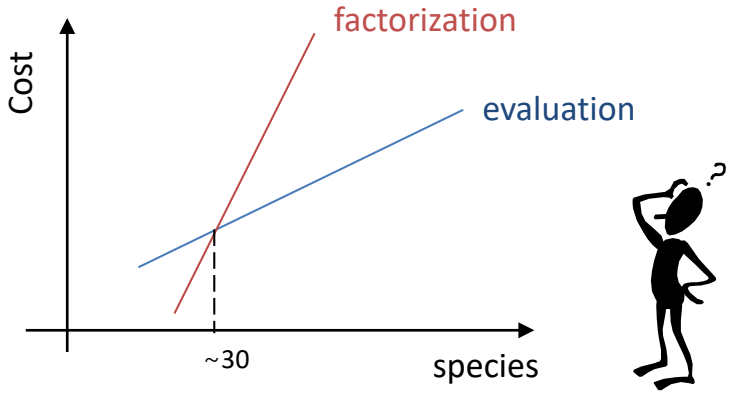
$$C = \beta N_S^2 + \gamma N_S^3$$

Evaluation of Jacobian (numerically):

$$\sim 5 \div 20 N_S^2$$

Factorization of Jacobian (LU):

$$\sim \frac{2}{3} N_S^3$$

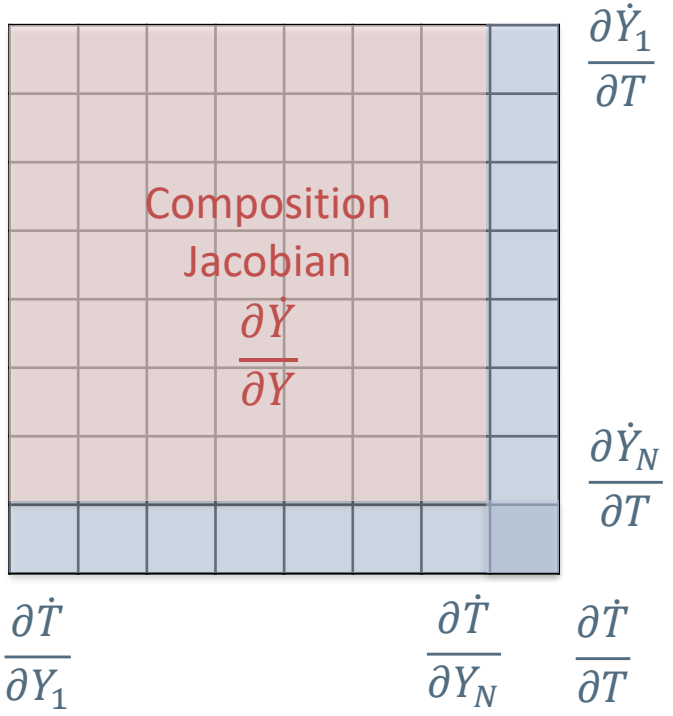


The Jacobian for 0D systems

Let us now consider again the system of equations governing an adiabatic, constant pressure, 0D reactor:

Governing equations

$$\begin{cases} \rho \dfrac{dY_i}{dt} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \dot{Q} \end{cases}$$



It is clear that the system's Jacobian comprises a "composition Jacobian" block and two vectors due to the dependence of T on composition and the dependence of reactions on temperature.

Note that in the case of isochoric reactors, pressure varies in time and the additional ODE for $P(t)$ gives rise to a larger Jacobian

The mass fractions Jacobian

Let us focus the attention on the composition Jacobian block. Mass fractions offer a natural description of the mixture's composition for reactions occurring at constant pressure and adiabatic conditions (no heat loss).

$$\frac{dY_i}{dt} = \dot{Y}_i = \frac{\dot{\Omega}_i}{\rho}$$

Unfortunately, the mass fraction Jacobian $\frac{\partial \dot{\boldsymbol{Y}}}{\partial \boldsymbol{Y}}$ is a **dense matrix**, i.e. all of its elements are nonzero!

This occurs because the density is a function of the mass fractions of **all the species**!

However, kinetic mechanisms are typically very sparse, due to the loose chemical coupling between species (i.e. each reactions involves a very limited number of species).

We have to try to exploit this sparsity!

The Jacobian as a product of matrices

$$\mathbf{J} = \frac{\partial \dot{\boldsymbol{\Omega}}}{\partial \mathbf{Y}} = \mathbf{A}\boldsymbol{\Gamma}$$

$$\mathbf{A} = \sum_{i=1}^{NR} \left(\boldsymbol{\nu}_i \frac{dr_i}{d\boldsymbol{C}} \right)$$

$$\boldsymbol{\Gamma} = \frac{d\boldsymbol{C}}{d\boldsymbol{Y}}$$

It is inexpensive to evaluate analytically, because only the differentiation of each reaction rate r_i with respect to the reactants is needed

Variable transformation matrix (system-dependent). Usually **Y** represents the mass fractions/ Not time consuming because it does not involve expensive rate evaluations

Cost

Original formulation: $\sim 5 \div 20 N_S^2$

Analytic Jacobian: $\sim N_S$

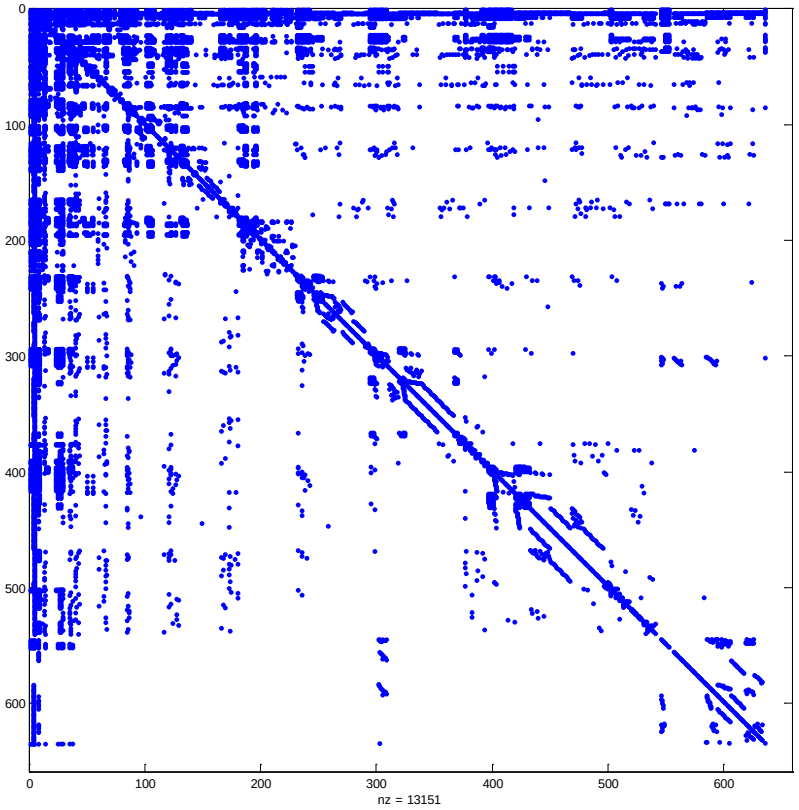
The concentration Jacobian is sparse

The **concentration Jacobian is a "sparse" matrix** due to the loose chemical coupling between species:

$$\frac{dC_i}{dt} = \frac{\dot{\Omega}_i}{W_i} \quad \Rightarrow \quad J_{ik}^{C} = \frac{\partial \dot{C}_i}{\partial C_k} = \frac{\partial}{\partial C_k}\left(\frac{\dot{\Omega}_i}{W_i}\right)$$

A sparse matrix is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, the matrix is called dense.

The "sparsity" of a matrix is the ratio of nonzero to the total number of elements.
It is of paramount importance to exploit the underlying sparsity of the Jacobian matrix for the purpose of improving the computational performance of kinetics computations.



Sparsity pattern for the 658 species n-heptane mechanism from LLNL. The sparsity is 3.03%.

Physical meaning of Jacobian elements

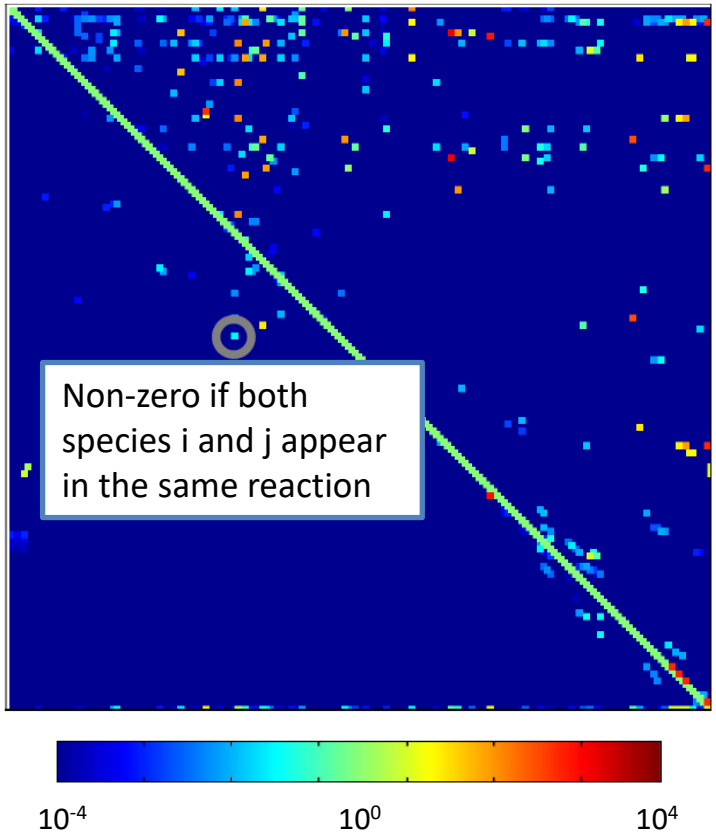
$$\begin{cases} \dfrac{dC_i}{dt} = \dfrac{\dot{\Omega}_i}{W_i} \\ C_i(0) = C_i^0 \end{cases}$$

Governing equations for a 0D reactor

$$J_{ik} = \frac{\partial\left(\frac{\dot{\Omega}_i}{W_i}\right)}{\partial C_k}$$

The magnitude represents the characteristic frequency at which the 2 species (i and k) are coupled

Jacobian matrix element magnitude



Adapted from:
McNenly M. et al., *"Improving Combustion Software to Solve Detailed Chemical Kinetics for HECC"*, LLNL-PRES-593472

Example of a sparse mechanism

Blue pixel (i, k): a possible non-trivial (non-zero) entry in Jacobian matrix

N_S = number of species
N_R = number of reactions, $N_R \sim 5\text{-}20\ N_S$

Non-zero coefficients per row: ~10

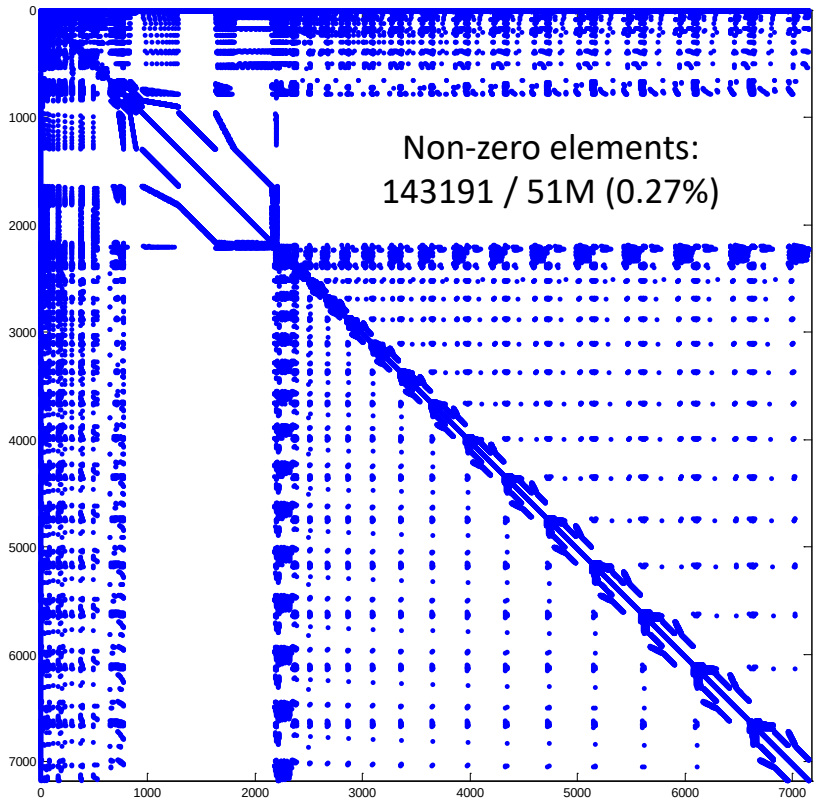
Total number of coefficients: $\sim 10 N_S$

Total coefficients in **J**: $N_S \cdot N_S$

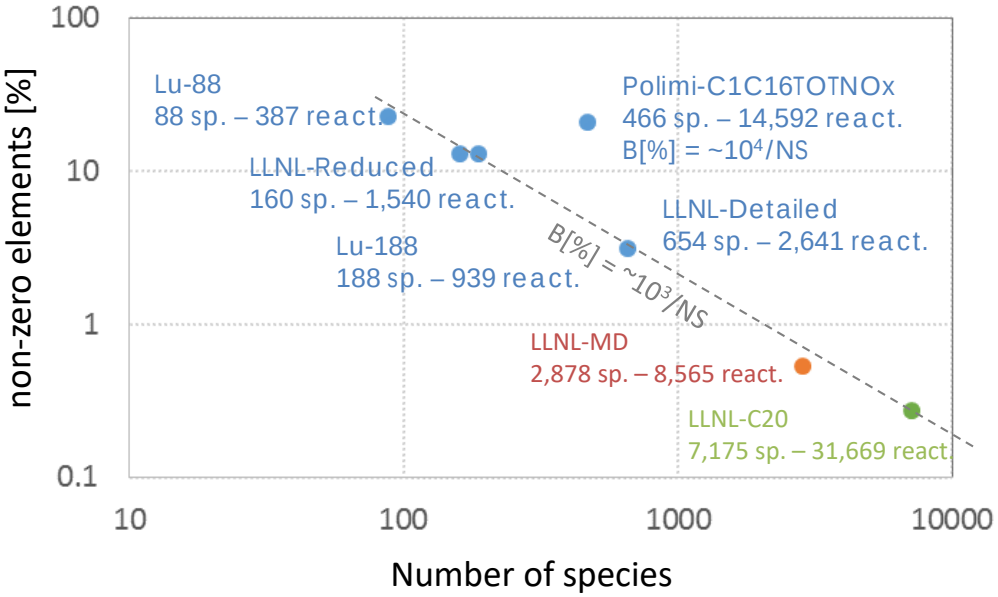
Fraction of non zero coefficients in J: $\sim \frac{10}{N_S}$

Larger mechanisms are sparser!

LLNL n-alkanes
Species: 7175, Reactions: 31669



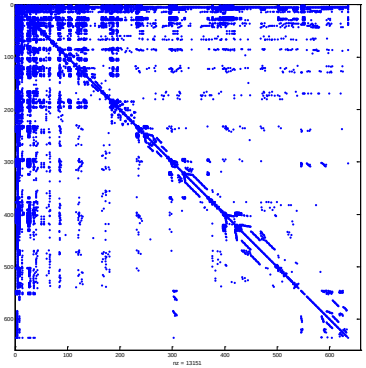
Sparsity of mechanisms

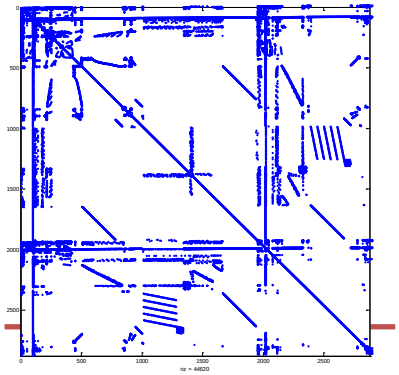


The sparsity of a wide selection of mechanisms has been assessed and shown in the figure to the left.

It is clear that, the larger then kinetics mechanisms, i.e. the more species, the greater the sparsity.

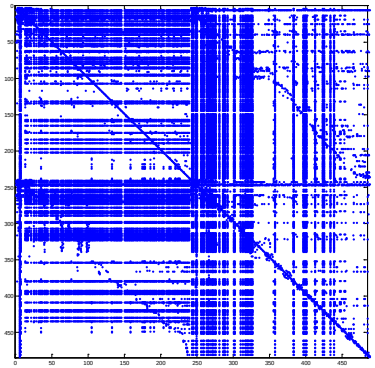
LLNL n-heptane
Species: 658
Reactions: 2827
Non-zeros: 3.03%





LLNL Methyl-decanoate
Species: 2878
Reactions: 8555
Non-zeros: 0.49%

PolimiTOT1412 + NOx
Species: 484
Reactions: 19341
Non-zeros: 16.07%



Direct vs Iterative solvers (I)

Direct solvers

- Compute a proper decomposition of **J**
- Can be thought of as variant of LU decomposition that finds triangular factors **L** and **U**, so that **J**=**LU**
- Sparse direct solvers save memory and CPU time by considering the sparsity pattern of **J**
- Very robust
- Work grows with N^2
- Memory grows with $N^{3/2}$

Do not stuck with some poorly supported packages found somewhere on the internet! Choose widely used, high-quality software!

Iterative solvers

- Improve the solution in each iteration
- Start with an initial guess
- Continue iterations till a stopping criterion is satisfied (typically that error/residual is less than a specified tolerance)
- Return final guess
- Depending on solver and preconditioner type, work can be only $\mathcal{O}(N)$… or much worse!
- Memory increases linearly
- Note: the final guess does not solve **Jx**=**b** exactly

Direct vs Iterative solvers (II)

Direct solvers

- Always work, for any invertible matrix
- No need to think about preconditioners
- Faster for problems with < 10k unknowns
- Usually do not scale very well
- Large requests of memory for large problems

Iterative solvers

- Need o(N) memory
- Can solve very large problems (10^6 of unknowns)
- Often parallelize very well
- Choice of solver/preconditioner depends on the problem

Advantage: only need multiplication with the matrix, no access to matrix element is required

Disadvantage: Efficiency strongly depends on the availability of good preconditioners

Preconditioning

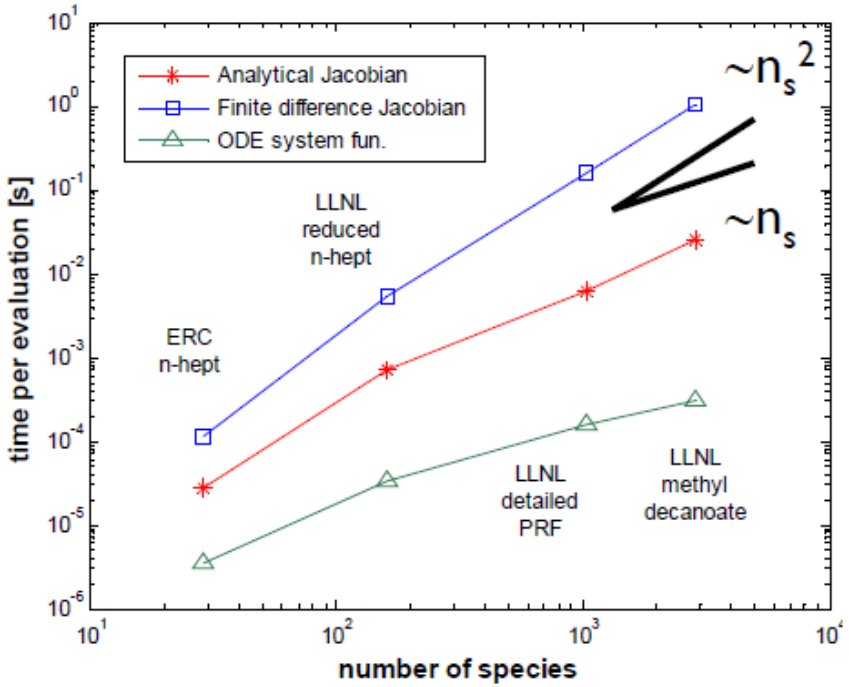
Instead of solving the original linear $\mathbf{Ax} = \mathbf{b}$, one may solve the left preconditioned system:

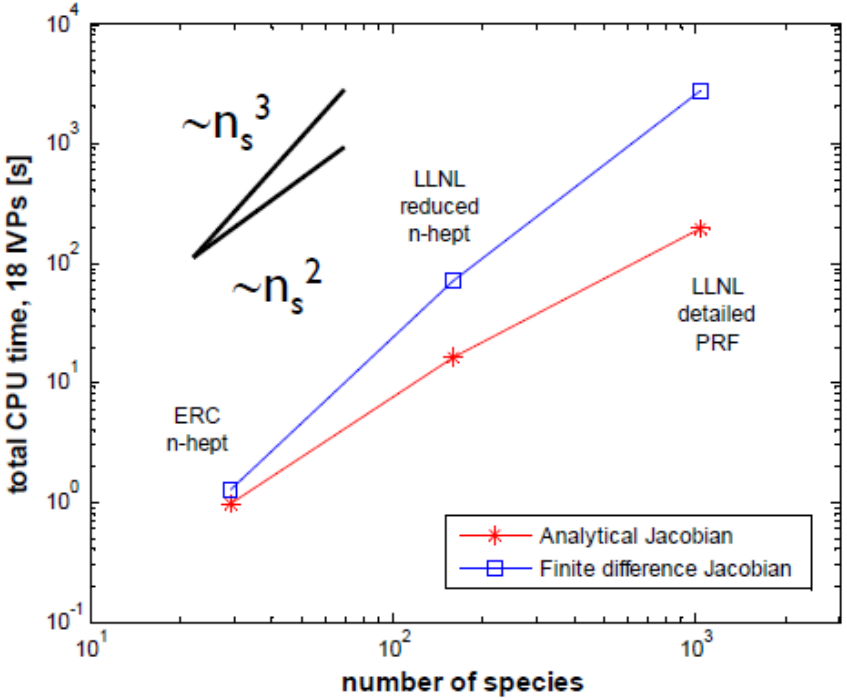
$$\boldsymbol{P}^{-1}(\mathbf{Ax} - \mathbf{b}) = \mathbf{0}$$

which give the same solution as the original system so long as the preconditioner matrix **P** is nonsingular. The right preconditioning is less common.

The goal of this preconditioned system is to reduce the condition number of the left preconditioned system matrix $\boldsymbol{P}^{-1}\mathbf{A}$. The preconditioned matrix $\boldsymbol{P}^{-1}\mathbf{A}$ is almost never explicitly formed. Only the action of applying the preconditioner solve operation $\boldsymbol{P}^{-1}$to a given vector need to be computed in iterative methods.

Jacobian matrix sparsity effects





Analytical, sparse evaluation of Jacobian: $N_S^2 \Rightarrow N_S$

Jacobian matrix storage requirements: $N_S^2 \Rightarrow N_S$

Jacobian matrix factorization: $N_S^2 \Rightarrow N_S$

Plots from:
Perini, F., Galligani, E., Reitz, R. D., *A study of direct and Krylov iterative sparse solver techniques to approach linear scaling of the integration of chemical kinetics with detailed combustion mechanisms*, Combustion and Flame 161 (5), 1180-1195 (2014)

Analytical Jacobian

$$\mathbf{J} = \frac{\partial \dot{\boldsymbol{\Omega}}}{\partial \mathbf{Y}} = \mathbf{A}\boldsymbol{\Gamma}$$

$$\mathbf{A} = \sum_{i=1}^{NR} \left(\boldsymbol{\nu}_i \frac{dr_i}{d\boldsymbol{C}} \right)$$

$$\boldsymbol{\Gamma} = \frac{d\boldsymbol{C}}{d\boldsymbol{Y}}$$

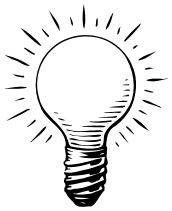
It is inexpensive to evaluate analytically, because only the differentiation of each reaction rate r_i with respect to the reactants is needed

Variable transformation matrix (system-dependent). Usually **Y** represents the mass fractions/ Not time consuming because it does not involve expensive rate evaluations

Cost

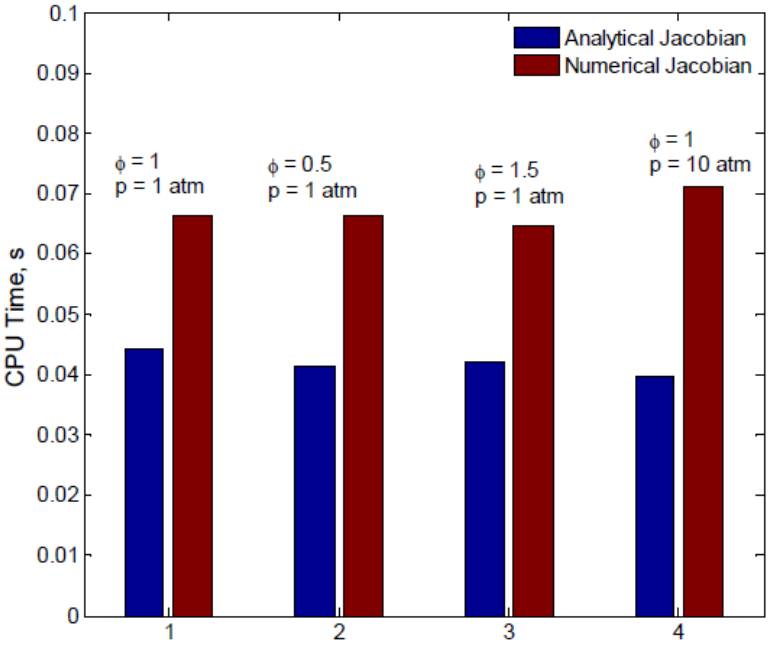
Analytic Jacobian: $\sim N_S$

Numerical Jacobian: $\sim 5 \div 20 N_S^2$



Analytical Jacobian is more accurate

Auto-ignition of CH4/air (T_0=1000K)



Plot from:
Lu T., *Computational Tools for Diagnostics and Reduction of Detailed Chemical Kinetics*, Princeton-CEFRC Summer School on Combustion (2012)

Analytical Jacobian should always be used!

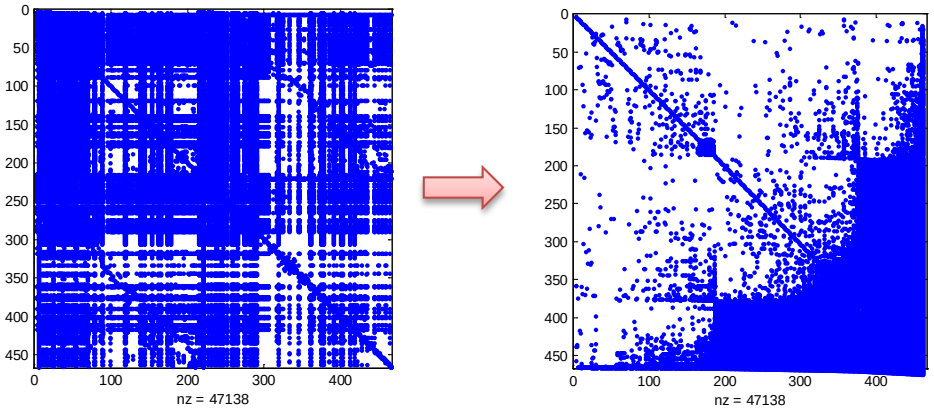
Hard-coded analytical Jacobian

Symbolic preprocessor (MATLAB, Maple, etc.)

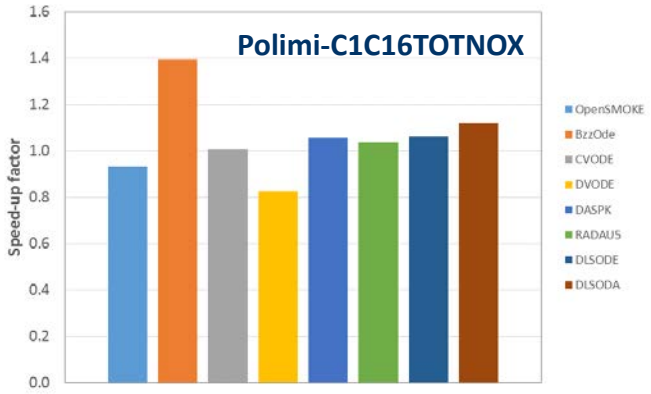
C++ (or FORTRAN) source code

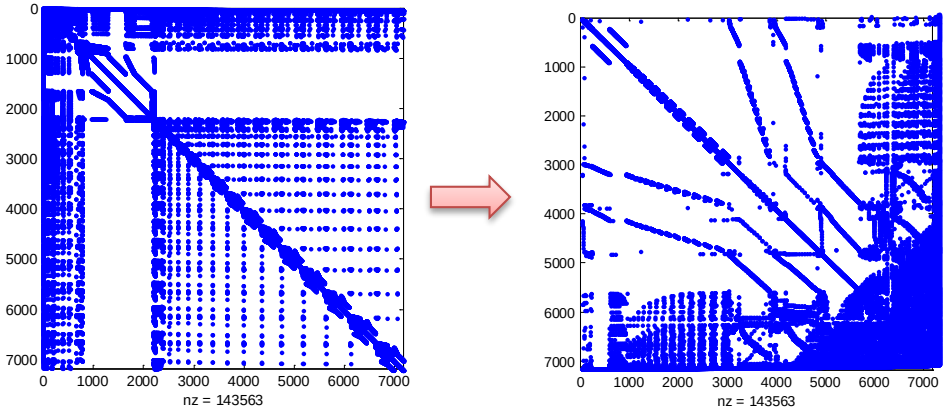
- ✓ The hard-coded Jacobian is super-fast (static allocation and compiler optimizations)
- ✓ Compilation time is super-long (thousands of lines of codes)
- ✓ Recompilation is needed if the mechanism changes

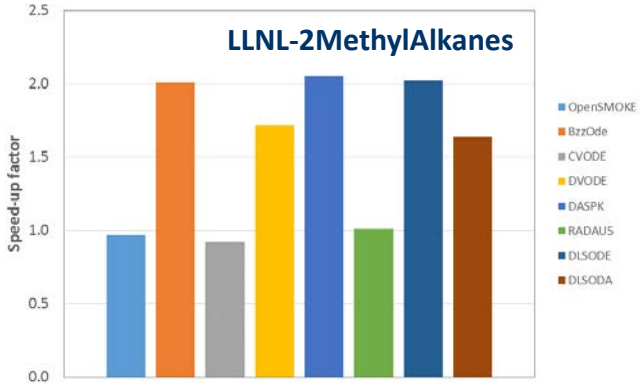
Reordering the species in the mechanism



Polimi-C1C16TOTNOX
Species: 484, Reactions: 19341







LLNL n-alkanes
Species: 7175, Reactions: 31669

Outline

1. **Introduction**

   Complexity of reacting flows: number of equations, coupling, non-linearity, stiffness

2. **Numerical solution of "chemistry"**
   a) The 0D reactor model
   b) Solution of ODE systems
   c) Numerical methods for stiff ODE systems
   d) The Jacobian matrix

3. **The ideal or OD reacting systems**
   a) Batch Reactor
   b) Shock Tube Reactor
   c) Perfectly Stirred Reactor
   d) Plug Flow Reactor

Ideal or 0D reacting systems (I)

Ideal or 0D systems include:

• batch reactors (BR) • rapid compression machines (RCM) • shock-tube reactors (STR)	Ignition delay times
• plug-flow reactors (PFR) • perfectly stirred reactors (PSR)	Speciation

Those ideal systems are mathematical approximations to corresponding las-scale reactors that are commonly adopted to study chemical kinetics.

They are referred to as 0D reactors since they can be described by a system of ordinary differential equations (ODE) with initial conditions, in which the independent coordinate is the time!

Details are available here: **A. Cuoci,** *Numerical modeling of reacting systems with detailed kinetic mechanisms*, Computer Aided Chemical Engineering, 45, p. 675-721 (2019)

Ideal or 0D reacting systems (II)

- The batch reactor, the RCM and the shock-tubes are especially useful to characterize **explosion limits and ignition delay times**.
- Flow reactors (PFR and PSR) have proved high valuable in the **characterization of composition (i.e. speciation)**, also at high temperature.
- They are also very useful idealizations for modeling practical systems. For example, the PSR may be used as an approximation of reacting flows in which intense turbulent mixing occurs (promoting uniformity in space). PFR is useful for flows with strong mixing in cross-stream direction, but insignificant mixing in the primary flow direction.
- For systems with complex fluid dynamics, in which the approximation provided by a single ideal reactor is too crude, networks of ideal reactors can represent a very useful and accurate approximation.

The batch reactor

The batch reactor is a closed system (i.e. without input/output streams), which is characterized by a fixed amount of mass.

A variety of batch reactors can be solved, either at constant pressure or constant volume, in isothermal or adiabatic conditions or with heat exchange with the external environment.

The batch reactor is described by a system of ODEs with ICs:

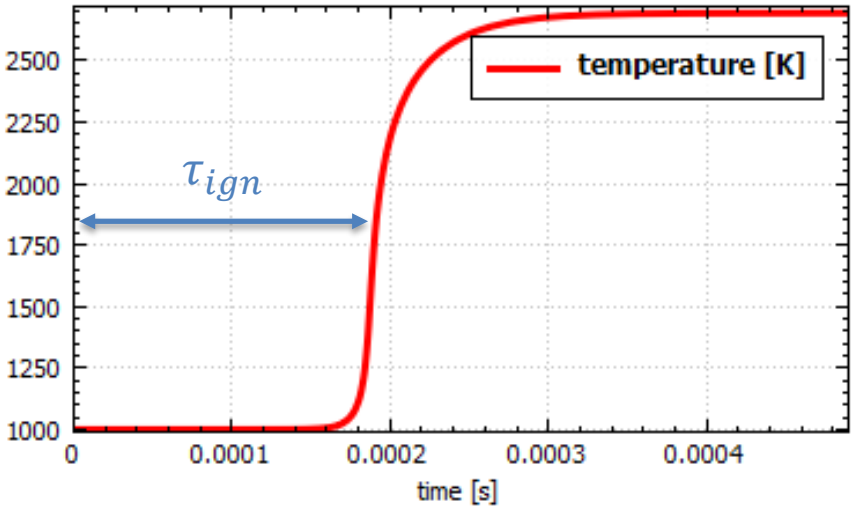
$$\begin{cases} \rho \dfrac{dY_i}{dt} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \dfrac{dP}{dt} + \dot{Q} + \dfrac{Q_{ext}}{V} \end{cases} \qquad \begin{cases} Y_i(0) = Y_i^0 \\ T(0) = T^0 \end{cases}$$

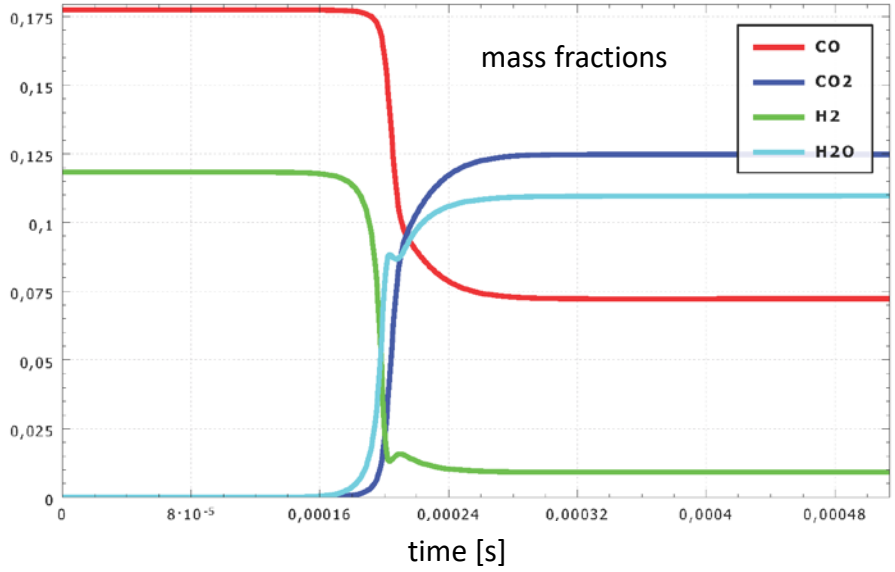
Q_{ext} is the power exchanged with the external environment and V is the reactor volume.

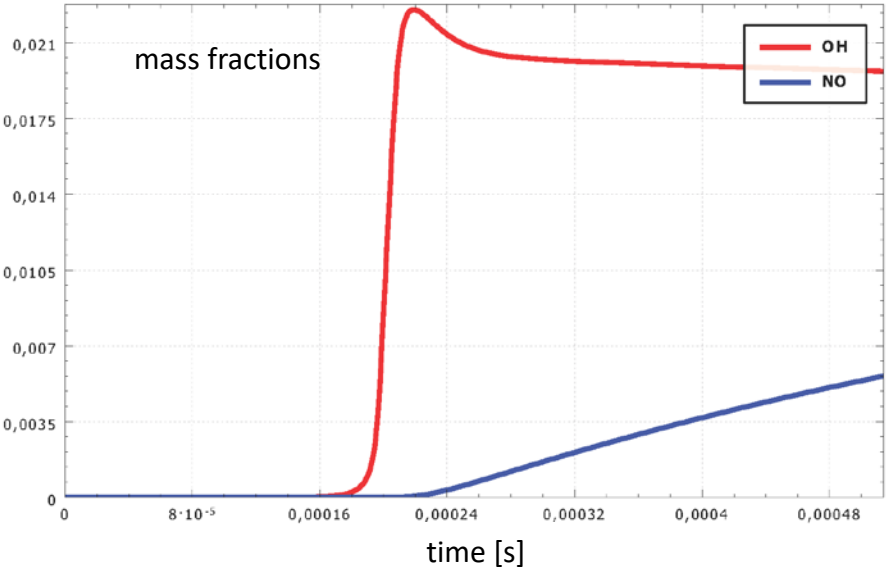
An example: autoignition of H2/CO mixture

Adiabatic batch reactor (constant volume)

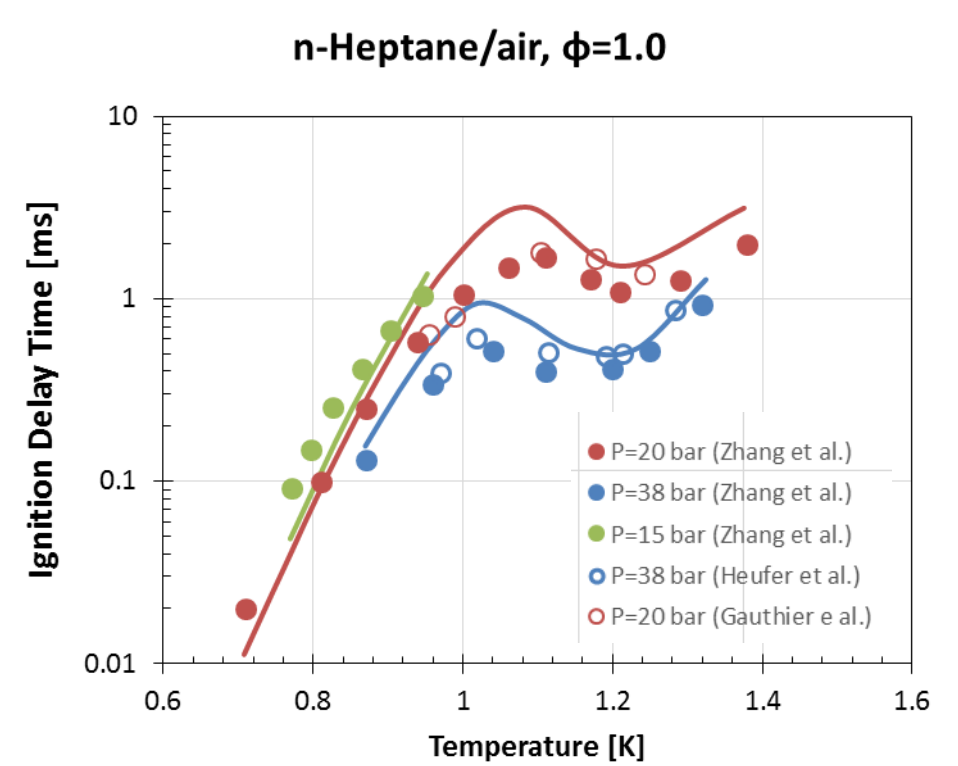
$T^0 = 1000K$
$P^0 = 1atm$
Fuel: H2/CO (40/60 mol)
Oxidizer: O2/N2 (21/79 mol)
Equivalence ratio: 1







Ignition delay times from batch reactors



In the experiments, ignition delay times of stoichiometric n-heptane/air mixtures have been measured in two different high-pressure shock tubes in the temperature range of 726–1412 K and at elevated pressures (15, 20 and 38 bar).

Zhang et al., *An updated experimental and kinetic modeling study of n-heptane oxidation*, Combustion and Flame 172 (2016), p. 116-135

Shock Tube Reactor (I)

The Shock Tube Reactor (STR) model is used to model the chemical kinetics behind a normal incident or a reflected shock. The interest is especially to simulate the behavior of a shock tube experiment for studying reaction kinetics.

The set of equations describing the mass fraction, velocity and temperature profiles downstream of the shock, can be derived from conservation laws of mass, momentum and energy. The flow is assumed to be adiabatic and mass diffusion, thermal conductivity, and viscous effects are assumed to be negligible.

Since the typical test times behind a shock wave are on the order of $0.1 - 1\ ms$, neglecting transport phenomena does not impact on the accuracy of the results.

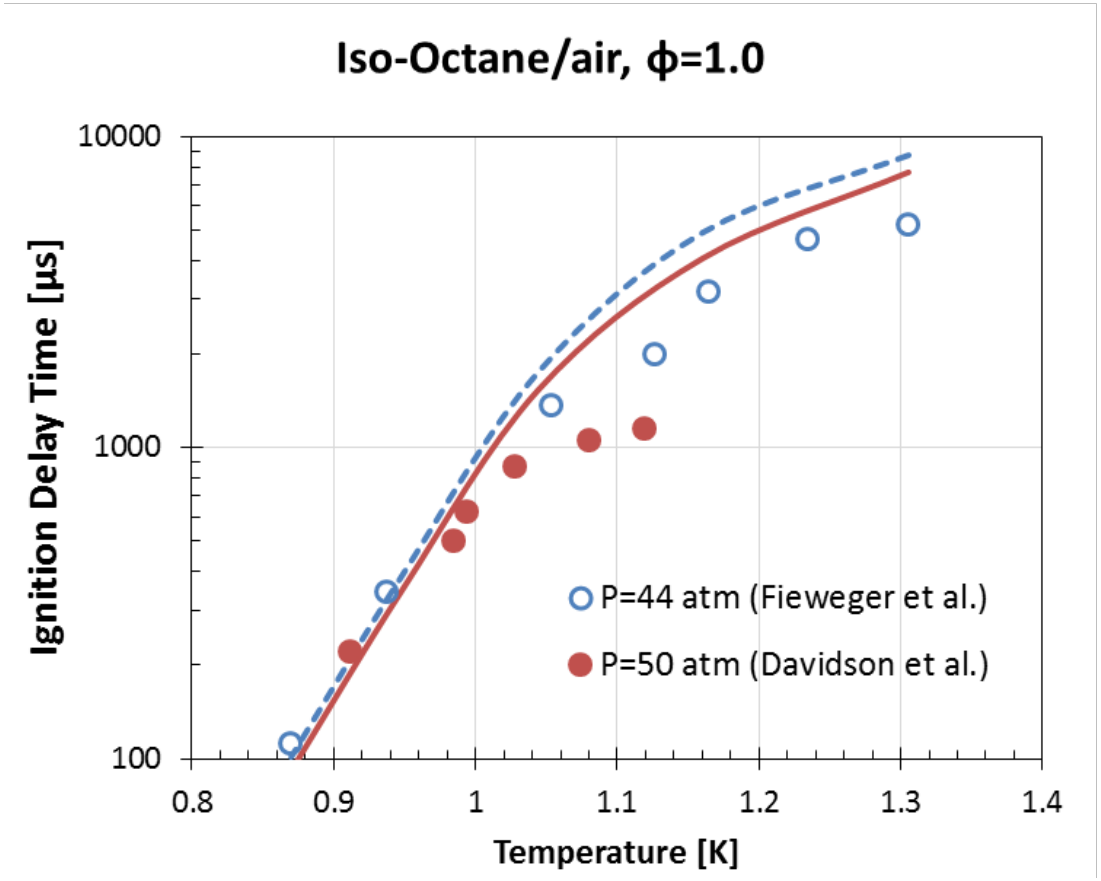
Shock Tube Reactor (II)

The conservation equations constitute an ODE system with initial conditions:

$$\begin{cases} \rho \dfrac{dY_i}{dz} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = v^2 \dfrac{d\rho}{dt} + \dot{Q} + \rho v^3 \beta \\ \left[P\left(1 + \dfrac{v^2}{C_P T}\right) - \rho v^2 \right] \dfrac{d\rho}{dt} = -PW \sum_{i=1}^{N} \dfrac{\dot{\Omega}_i}{W_i} - \dfrac{P}{C_P T}\dot{Q} + \rho^2 v^3 \left(1 - \dfrac{p}{\rho C_P T}\right)\beta \\ \rho \dfrac{dv}{dt} = -v \dfrac{d\rho}{dt} - \rho v^2 \beta \end{cases}$$

The equations reported above are written using the time as the independent variable instead of distance , since in typical shock tube experiments, the usual measurable quantities are density, species concentration, velocity and temperature as functions of time.

An example: iso-octane



Ignition delay times were measured in a shock tube for iso-octane/air and toluene/air at conditions similar to those found in homogeneous charge compression ignition (HCCI) engines.

Davidson et al., *Shock tube ignition measurements of iso-octane/air and toluene/air at high pressures*, Proceedings of the Combustion Institute, 30 (2005), p. 1175-1182

Perfectly Stirred Reactor

The Perfectly Stirred Reactor (PSR) or Continuously Stirred Tank Reactor (CSTR) is an idealization that proves useful in describing laboratory experiments and can often be used in modeling practical devices.

Gases enter the reactor with mass flow rate, temperature T^{in} (and molar enthalpies H_i^{in}) and composition Y_i^{in}. Once inside the reactor, the gases mix instantaneously and perfectly, which means that temperature and composition within the reactor are uniform.

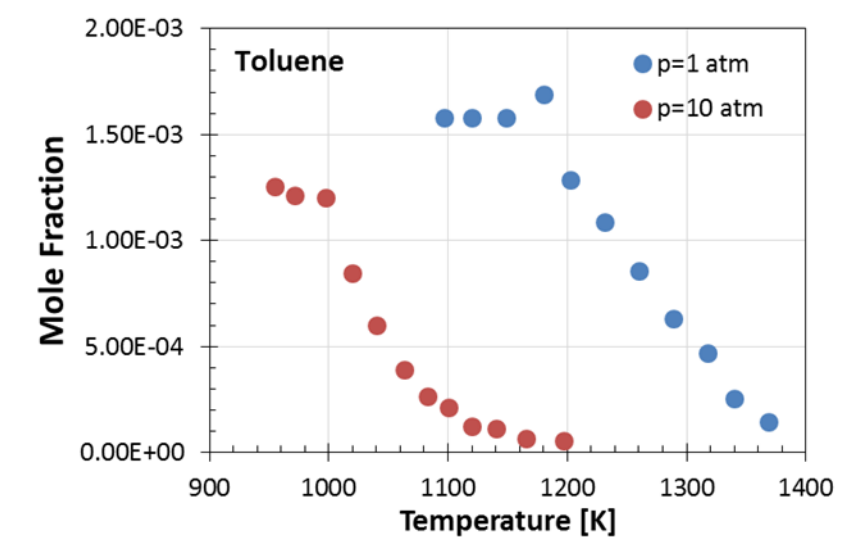
In experiments, isothermal conditions are adopted in most cases.

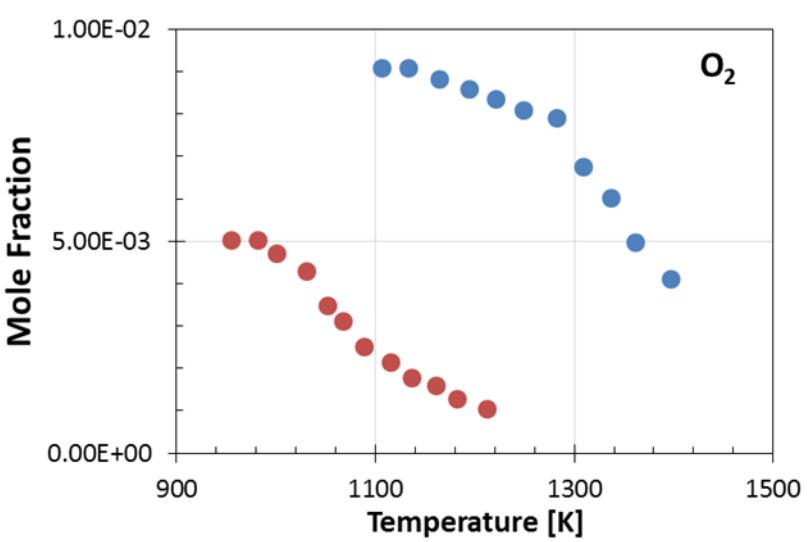
The conservation equations of species and energy are a system of ODEs with ICs:

$$\begin{cases} \rho \dfrac{dY_i}{dz} = \rho \dfrac{Y_i^{in} - Y_i}{\tau} + \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{dt} = \rho \dfrac{\sum X_i^{in}\left(H_i^{in} - H_i\right)}{W\tau} + \dot{Q} + \dfrac{Q_{ex}}{V} \end{cases} \qquad \begin{cases} Y_i(0) = Y_i^0 \\ T(0) = T^0 \end{cases}$$

An example: oxidation of toluene (I)

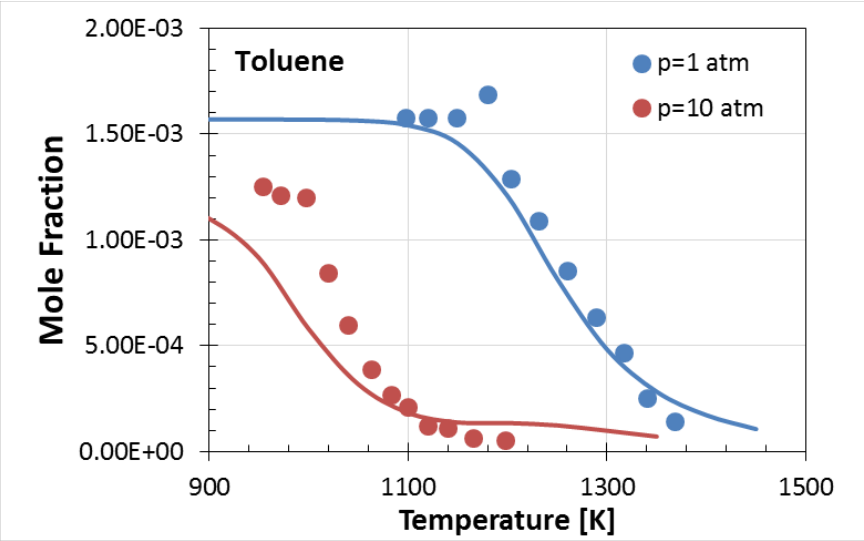
The oxidation of toluene was investigated in a jet stirred reactor (JSR) at the **pressure of 10 atm**, **residence time of 0.6 s**, **equivalence ratios of 0.5, 1.0 and 1.5**, and **temperatures from 950 to 1200 K** using gas chromatography combined with flame ionization detector, thermal conductivity detector and mass spectrometry

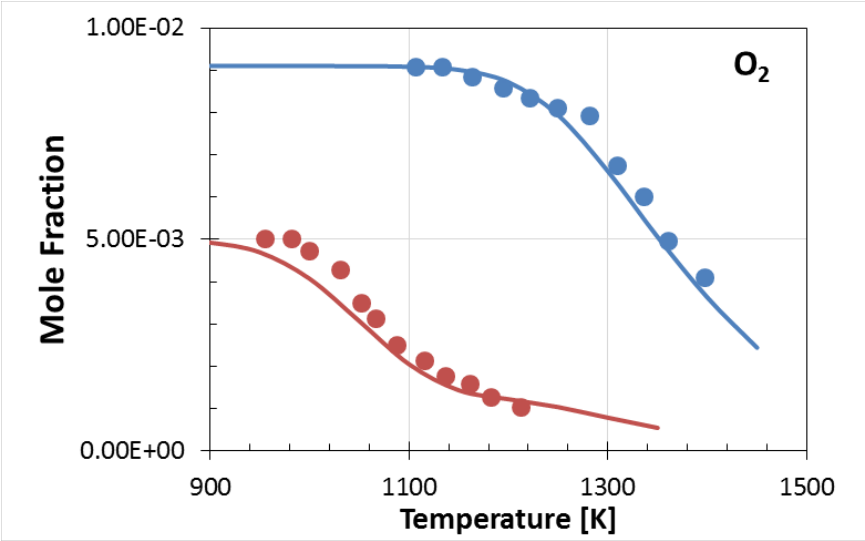


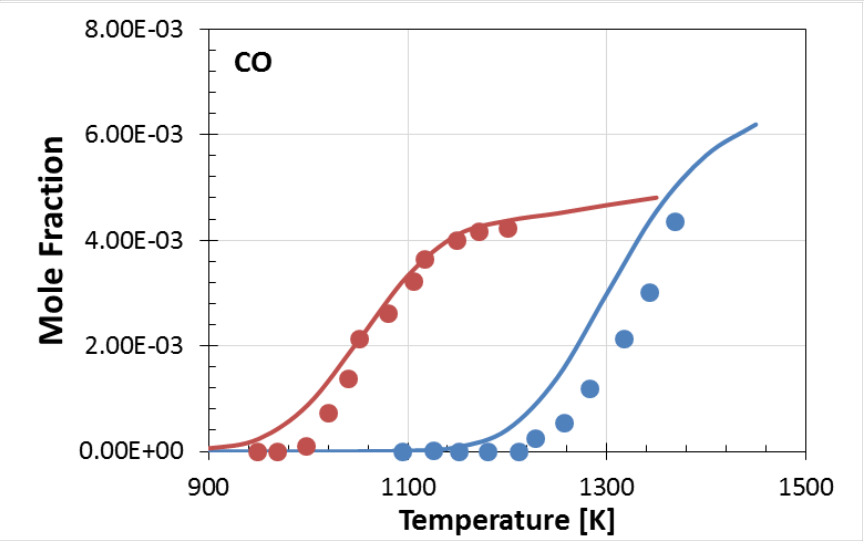


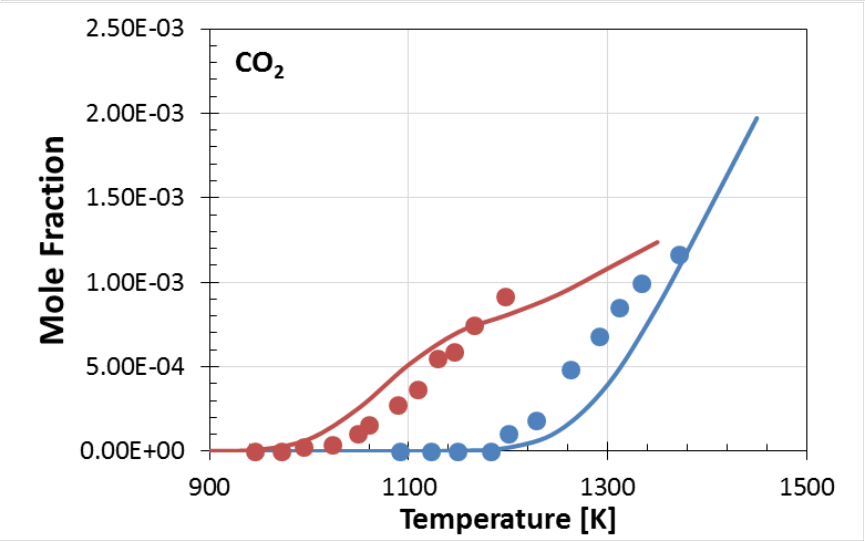
Yuan et al., *Investigation on the pyrolysis and oxidation of toluene over a wide range conditions. I. Flow reactor pyrolysis and jet stirred reactor oxidation*, Combustion and Flame, 162 (2015), p. 3-21

An example: oxidation of toluene (II)





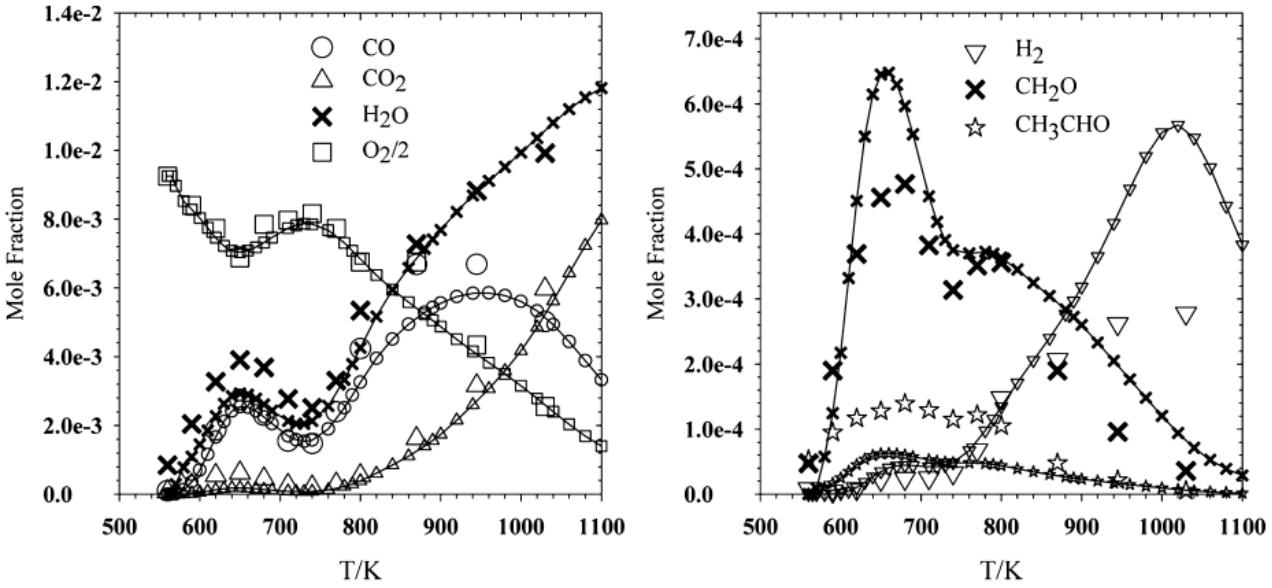




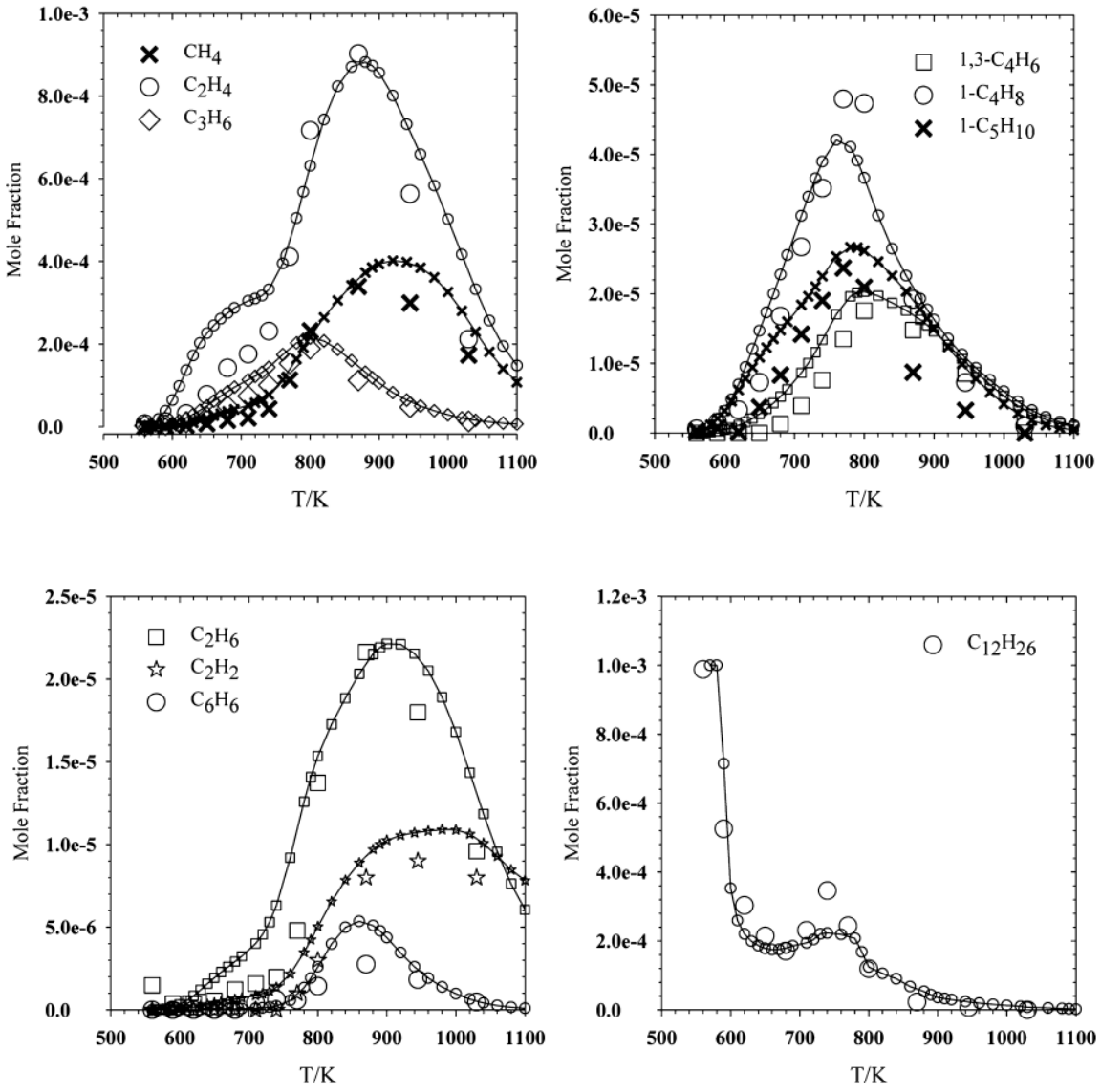
An example: oxidation of n-dodecane at high pressure (I)

Mzé –Ahmed et al. experimentally studied two large hydrocarbons (n-undecane and n-dodecane) in a jet-stirred reactor (JSR) of volume equal to 33 cm3 at high pressure (P=10 bar), at temperatures ranging from 550 to 1150 K, at a constant residence time (τ=1 s) and for three equivalence ratios (φ= 0.5, 1.0, and 2.0).

The experiments are simulated as isothermal (fixed temperature) perfectly stirred reactor: 10 bar, equivalence ratio of 1 (which means molar fractions nC12 0.1%, O2 1.85%, N2 98.05%) and residence time of 1 s.



An example: oxidation of n-dodecane at high pressure (II)



A. Mzé –Ahmed, K. Hadj-Ali, P. Dagaut, G. Dayma, Experimental and Modeling Study of the Oxidation Kinetics of n-Undecane and n-Dodecane in a Jet-Stirred Reactor, Energy & Fuels, 26, 4253−4268 (2012)

Plug Flow Reactor (I)

Plug flow reactors (PFR) are commonly used to validate detailed kinetic mechanisms and to perform kinetic analysis.

The species and the temperature may vary along the reactor, but it is assumed that there are neither variations in radial direction, neither diffusive transport along the length of the channel, i.e. in the flow direction.

The conservation equations of species and energy are written with respect to the spatial coordinate. An additional equation is written to reconstruct the residence time .

Thanks to the assumption reported above, the steady-state equations governing the PFR constitute an ODE system with initial conditions (prescribed on the inlet section):

$$\begin{cases} \rho v \dfrac{dY_i}{dz} = \dot{\Omega}_i \\ \rho v \left(C_P + \dfrac{v^2}{T} \right) \dfrac{dT}{dz} = \dot{Q} - v^2 W \displaystyle\sum_{i=1}^{N} \frac{\dot{\Omega}_i}{W_i} + \frac{U}{a_C} (T_{ext} - T) \end{cases} \qquad \begin{cases} Y_i(0) = Y_i^0 \\ T(0) = T^0 \end{cases}$$

Plug Flow Reactor (I)

The equations reported before can be simplified if we neglect the kinetic energy contribution in the energy equation and we introduce the concept of residence time $\tau(z)$, i.e. the time spent by the reacting mixture at a given coordinate z:

$$\frac{dz}{d\tau} = v$$

Which leads to:

$$\begin{cases} \rho \dfrac{dY_i}{d\tau} = \dot{\Omega}_i \\ \rho C_P \dfrac{dT}{d\tau} = \dot{Q} + \dfrac{U}{a_C}(T_{ext} - T) \end{cases} \qquad \begin{cases} Y_i(0) = Y_i^0 \\ T(0) = T^0 \end{cases}$$

In case of adiabatic conditions, they become equivalent to the equations governing the adiabatic batch reactor.

Training Session 2 (optional)

- using the **batch reactor** model for estimating the **ignition delay times**
- using the **shock-tube reactor** model for estimating the **ignition delay times**
- analyze the predictive capabilities of an existing kinetic mechanism in describing the formation of chemical species (i.e. **speciation**) via experimental data from **Jet Stirred Reactors (JSR)**
- analyze the predictive capabilities of an existing kinetic mechanism in describing the formation of chemical species (i.e. **speciation**) via experimental data from **Plug Flow Reactors (PFR)**