Simulation of chemical reaction kinetics

Lars Schiller

E-10-Simulation GmbH

June 30, 2015

Problem description and mathematical model

Simulation of a oxyhydrogen reaction with the given initial conditions and chemical equations: $\rightarrow$ $\dot{\boldsymbol{c}} = f(\boldsymbol{c})$, $\boldsymbol{c}_0 = [100\ 50\ 0\ 0\ 0\ 0]^T$

$$
\begin{aligned}
&\mathrm{H_2} \xrightarrow{1e-4} 2\mathrm{H}\cdot \\
&\mathrm{H}\cdot + \mathrm{O_2} \xrightarrow{1e4} \cdot\mathrm{OH} + \cdot\mathrm{O}\cdot \\
&\cdot\mathrm{O}\cdot + \mathrm{H_2} \xrightarrow{2e-2} \cdot\mathrm{OH} + \mathrm{H}\cdot \\
&\cdot\mathrm{OH} + \mathrm{H_2} \xrightarrow{1e-3} \mathrm{H_2O} + \mathrm{H}\cdot \\
&\mathrm{H}\cdot + \cdot\mathrm{OH} \xrightarrow{1e-3} \mathrm{H_2O}
\end{aligned}
\quad \rightarrow \quad
\begin{bmatrix} \dot{c}_{\mathrm{H_2}} \\ \dot{c}_{\mathrm{O_2}} \\ \dot{c}_{\mathrm{H}\cdot} \\ \dot{c}_{\cdot\mathrm{OH}} \\ \dot{c}_{\cdot\mathrm{O}\cdot} \\ \dot{c}_{\mathrm{H_2O}} \end{bmatrix}
=
\begin{bmatrix}
-1 & 0 & -1 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 \\
2 & -1 & 1 & 1 & -1 \\
0 & 1 & 1 & -1 & -1 \\
0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1e\text{-}4 c_{\mathrm{H_2}} \\
1e4 c_{\mathrm{H}\cdot} c_{\mathrm{O_2}} \\
2e\text{-}2 c_{\cdot\mathrm{O}\cdot} c_{\mathrm{H_2}} \\
1e\text{-}3 c_{\cdot\mathrm{OH}} c_{\mathrm{H_2}} \\
1e\text{-}3 c_{\mathrm{H}\cdot} c_{\cdot\mathrm{OH}}
\end{bmatrix}
$$

- Since the problem is *stiff*, we have to choose an implicit multi-step method to obtain a large domain of stability (A-stable).
- Try *Adams-Moulton*-method and *BDF*-method with $k = 1$ steps and a step size $h = 1\text{e}{-2}$:

$$\sum_{j=0}^{k} a_j y_{n+j} = h \sum_{j=0}^{k} b_j f_{n+j} \tag{1}$$

Adams-Moulton:

k	a_0	a_1	b_0	b_1
1	-1	1	0.5	0.5

$$F(y_{n+1}) = y_n + \frac{h}{2}\left(f_n + f_{n+1}\right) - y_{n+1}$$

BDF:

k	a_0	a_1	b_0	b_1
1	-1	1	0	1

$$F(y_{n+1}) = y_n + h f_{n+1} - y_{n+1}$$

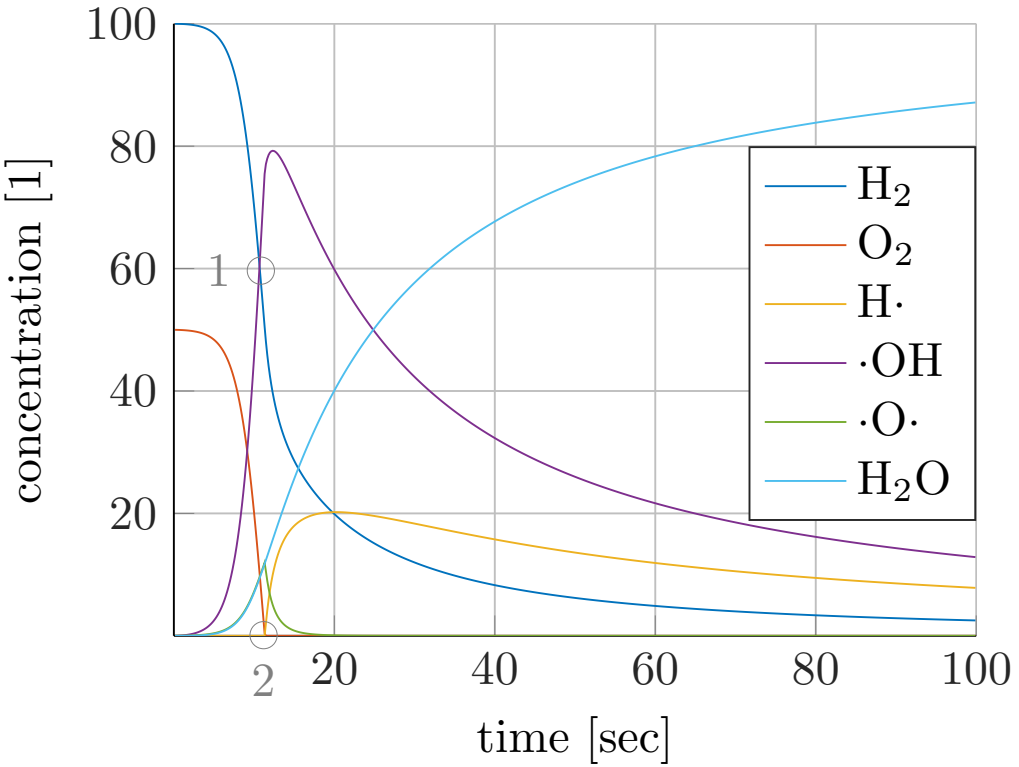
Implementation of the numerical solvers

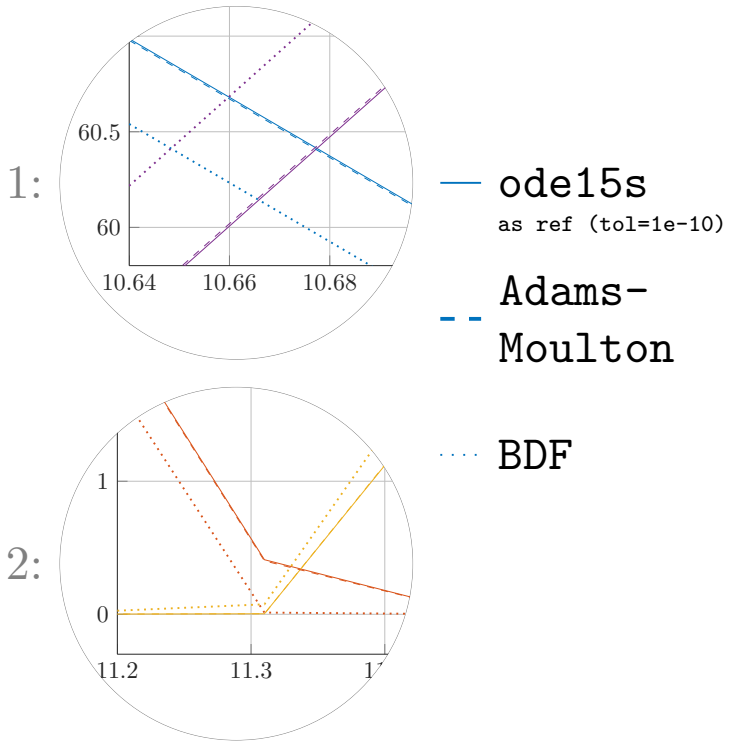
```
% Classic Iteration Loop
while (n+k-1)*h < tend
   % Newton-method
   [...] % initializing
   Fi = F(y(n+k-1));
   while norm(Fi) > tol
      [...] % calculate JFi
      yi(i+1) = yi(i)-JFi\Fi;
      Fi = F(yi(i+1));
      i = i+1;
   end
   y(n+k) = yi(i);
   n = n+1;
end
```

To calculate the Jacobian:

$$JF(y) = \left[\frac{\delta F(y)}{\delta y_1} \quad \dots \quad \frac{\delta F(y)}{\delta y_N} \right]$$

$$\frac{\delta F(y)}{\delta y_i} = \frac{F(y + \hat{h} e_i) - F(y)}{\hat{h}}$$





method	fun eval	max NS	average NS	ct [sec]	|err|
A-M	91016	7	1.0145	4.0205	0.0347
BDF	80476	7	1.0068	5.7835	13.5788
ode15s	988	-	-	0.4859	ref