A Chemical Reactor Network Approach for Modeling Pollutant Formation in Large Industrial Furnaces in Julia

by

Bowen Shuchai Zhu

Submitted to the Faculty of Electrical Engineering, Mathematics and Computer Science
in partial fulfillment of the requirements for the degrees of
Bachelor of Science in Applied Mathematics
and
Bachelor of Science in Computer Science and Engineering
at the
DELFT UNIVERSITY OF TECHNOLOGY
July 2021

Author .....
Faculty of Electrical Engineering, Mathematics and Computer Science
July 2, 2021

Certified by .....
Domenico J. P. Lahaye
Assistant Professor
Thesis Supervisor

Certified by .....
Johan L. A. Dubbeldam
Associate Professor

Certified by .....
Matthias Möller
Associate Professor

A Chemical Reactor Network Approach for Modeling Pollutant Formation in Large Industrial Furnaces in Julia

by

Bowen Shuchai Zhu

Submitted to the Faculty of Electrical Engineering, Mathematics and Computer Science
on July 2, 2021, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Applied Mathematics
and
Bachelor of Science in Computer Science and Engineering

Abstract

The chemical reactor network (CRN) approach is a practical tool for precisely predicting the species concentration in combustion processes with low computational cost. This work examines the capability of the emerging Julia programming language and its ecosystem in solving large CRNs. The packages DifferentialEquations.jl and ModelingToolkit.jl are employed to defining and solving stiff ordinary differential equations, for which the implicit time-integration methods Rodas5 and TRBDF2 with the GMRES linear solver are used. The graph structure of reactor networks is constructed by LightGraphs.jl and SimpleWeightedGraphs.jl. The differential equation solver and the graph data structure are connected via NetworkDynamics.jl. It is concluded that Julia is a competent tool for CRNs containing up to 1000 nodes each with 4 species. Julia is capable of simulating pollutant formation in large reactor networks with reasonable time and memory space.

Thesis Supervisor: Domenico J. P. Lahaye

Contents

Chapter 1

Introduction

To move an aircraft, a propulsion system is needed to create thrust. Different engines develop thrust in different ways, but all depend on burning fuels. The chemical process of burning a fuel is called combustion. In such a process, oxygen reacts rapidly with a fuel which can be solid, liquid, or gas. During combustion, new chemical substances, called exhaust, are produced, most of which are water (H_2O, hydrogen + oxygen) and carbon dioxide (CO_2, carbon + oxygen). However, the emission of by-products are inevitable such as nitrogen oxides (NO_x, nitrogen + oxygen) since nitrogen constitutes around 78% of air besides the presence of dioxygen. Nitric oxide (NO) and nitrogen dioxide (NO_2) contribute to the formation of smog and acid rain and badly influences ozone. Another air pollutant gas, carbon monoxide (CO), also plays roles affecting climate change. Therefore, adequate tools are required for estimating emissions. This thesis aims at developing a simulator written in the Julia programming language following the chemical reactor network approach to model the formation of pollutants. In this project, we focus on the combustion of natural gas with methane (CH_4) as its main component[1] in large industrial furnaces.

An overview is given in this chapter. Chapter 2 describes the combustion of methane and its reactor network in furnaces. Chapter 3 introduces the ecosystem of the Julia programming language. Chapter 4 presents the application of Julia on modelling methane combustion. The function of Julia is further evaluated in chapter 5

[1] we neglect here the presence of higher alkanes such as propane (C_3H_8) and butane (C_4H_{10}).

by testing the 5-node-5-edge networks and large networks.

Chapter 2

Combustion

Combustion of gaseous fuels is the combined process of the mixing of fuel (methane) and oxidizer (oxygen) with transport of chemical species in a flow. To simulate combustion, we model the chemical reactions aiming at a sufficiently accurate description of the formation of major and minor species during the chemical reactions. The computational domain that represents the furnace is subdivided into subdomains (containers, vessels or reactors). The dimension of the subdomain is large compared to the typical dimension of a cell required to resolve the flow. These subdomains are interconnected by the flow on mass leaving one reactor and entering the other reactor. The furnace is represented as a chemical reactor network. A chemical reactor network consists of nodes and edges. The nodes and edges represent chemical reactors (subdomains) and interconnecting pipes, respectively. As for chemical reaction, we will consider the combustion of methane and ambient air at atmospheric pressure and constant volume. To alleviate computational cost, the representation of the flow is simplified.

2.1 Basic mechanism of methane combustion

The natural gas we consider here is a fuel primarily consisting of methane (CH_4). When one methane molecule meets two oxygen molecules with sufficient heat, they start to react and then one molecule of carbon dioxide and two molecules of water

are formed.

$$\mathrm{CH_4} + 2\,\mathrm{O_2} \longrightarrow \mathrm{CO_2} + 2\,\mathrm{H_2O}. \tag{2.1}$$

The rate of reaction is governed by the formulas

$$\frac{\mathrm{d}[\mathrm{CH_4}]}{\mathrm{d}t} = -k \cdot r, \tag{2.2a}$$
$$\frac{\mathrm{d}[\mathrm{O_2}]}{\mathrm{d}t} = -2k \cdot r, \tag{2.2b}$$
$$\frac{\mathrm{d}[\mathrm{CO_2}]}{\mathrm{d}t} = k \cdot r, \tag{2.2c}$$
$$\frac{\mathrm{d}[\mathrm{H_2O}]}{\mathrm{d}t} = 2k \cdot r, \tag{2.2d}$$

$$k = AT^{\beta} \exp(-E_a/RT), \tag{2.3}$$
$$r = [\mathrm{CH_4}]^{n_{\mathrm{CH_4}}} [\mathrm{O_2}]^{n_{\mathrm{O_2}}}, \tag{2.4}$$

where $[\,\cdot\,]$ means the concentration of a species, k is the rate coefficients, E_a is the activation energy, R is the gas constant, T is temperature, β is the temperature exponent and n is reaction order of a species. This relation is also known as the Arrhenius law, in which the reaction rate depends on the temperature through the Arrhenius equation and the pressure through the influence of pressure on molecular space density which affects the species concentrations.

Depending on the level of detail of the mechanism, the reaction of methane and oxygen may produce carbon dioxide and water in a single step. The reaction can also be expressed in two steps. See subsection 2.2.1 for more details.

2.2 Pollutants

Some products of combustion are harmful to the health of humans and other living beings and cause damage to the climate. The minor species, namely pollutants, of

conventional combustion systems include CO, UHC, NO_x, SO_x. For the research to be conducted, CO and NO_x are relevant pollutants and are thus considered in the following.

2.2.1 Carbon monoxide

Carbon monoxide (CO) is a colorless, odorless atmospheric pollutant mainly from the exhaust of incomplete combustion of fuels. Large CO pollution events can be observed from space over cities (Pommier et al., 2013). Breathing air with high concentration of CO leads to a reduced oxygen (O_2) transport in the blood stream to critical organs such as heart and brain (Chenoweth et al., 2021).

The 2S_CH4_BFER scheme by Franzelli et al. (2012) accounts for six species (CH_4, O_2, N_2, CO, CO_2, H_2O) and the following two reactions:

$$\mathrm{CH_4} + 1.5\,\mathrm{O_2} \longrightarrow \mathrm{CO} + 2\,\mathrm{H_2O} \tag{2.5a}$$

$$\mathrm{CO} + 0.5\,\mathrm{O_2} \rightleftharpoons \mathrm{CO_2}, \tag{2.5b}$$

where the forward reaction rate for reactions (2.5) are

$$k_1 = A_1 T^{\beta_1} \exp(-E_{a,1}/RT)[\mathrm{CH_4}]^{n_{\mathrm{CH_4}}}[\mathrm{O_2}]^{n_{\mathrm{O_2},1}}, \tag{2.6a}$$

$$k_2 = A_2 T^{\beta_2} \exp(-E_{a,2}/RT)[\mathrm{CO}]^{n_{\mathrm{CO}}}[\mathrm{O_2}]^{n_{\mathrm{O_2},2}}. \tag{2.6b}$$

The reaction parameters are summarized in table 2.1 (Franzelli et al., 2012).

Parameters	CH_4 oxidation (2.5a)		CO-CO_2 equilibrium (2.5b)	
Activation energy E_a	3.55×10^4		1.2×10^4	
Temperature exponent β	0.0		0.8	
Pre-exponential factor A	4.9×10^9		2×10^8	
Reaction exponents	$n_{\mathrm{CH_4}}$	0.50	n_{CO}	1.00
	$n_{\mathrm{O_2},1}$	0.65	$n_{\mathrm{O_2},2}$	0.50

Table 2.1: Reaction parameters for the 2S_CH4_BFER mechanism

Furthermore, Jones and Lindstedt (1988) presented a four-step mechanism which could be useful for more accurate simulation.

2.2.2 Nitrogen oxides

Oxygen and nitrogen in general do not react at ambient temperatures due to lack of energy. But at high temperatures typically in internal combustion engines, they undergo an endothermic reaction producing various oxides of nitrogen (NO_x).

Nitrogen oxides mainly result in two forms of air pollution. In the presence of sunlight, NO_x reacts with volatile organic compounds (VOCs) yielding photochemical smog. People with lung diseases and people who work or exercise outside are particularly susceptible to adverse effects of smog such as damage to lung tissue and reduction in lung function (Hamra et al., 2015). Furthermore, NO_x generates nitric acid and acid rain by reacting in certain environments.

The primary sources of NO_x in combustion processes are dominated by the thermal NO formation, also well known as the Zel'dovich mechanism (Zel'dovich, 1946). The reaction mechanism consists of two chain reactions:

$$\mathrm{O} + \mathrm{N_2} \overset{k_1}{\rightleftharpoons} \mathrm{NO} + \mathrm{N} \tag{2.7a}$$

$$\mathrm{N} + \mathrm{O_2} \overset{k_2}{\rightleftharpoons} \mathrm{NO} + \mathrm{O} \tag{2.7b}$$

which can be extended by adding the reaction

$$\mathrm{N} + \mathrm{OH} \overset{k_3}{\rightleftharpoons} \mathrm{NO} + \mathrm{H}. \tag{2.7c}$$

The forward and backward reaction rate constants for reactions (2.7) are given in table 2.2 (Turns, 2012, p. 171).

Reaction	Forward reaction rate coefficients ($m^3/(kmol)(s)$)	Backward reaction rate coefficients ($m^3/(kmol)(s)$)
2.7a	$k_{1f} = 1.8 \times 10^{11} \exp(-38370/T)$	$k_{1b} = 3.8 \times 10^{10} \exp(-425/T)$
2.7b	$k_{2f} = 1.8 \times 10^{7} \exp(-4680/T)$	$k_{2b} = 3.8 \times 10^{6} \exp(-20820/T)$
2.7c	$k_{3f} = 7.1 \times 10^{10} \exp(-450/T)$	$k_{3b} = 1.7 \times 10^{11} \exp(-24560/T)$

Table 2.2: Reaction rates for the extended Zel'dovich mechanism

Note that reaction (2.7a) has a very high activation energy ($E_a = 38370R =$

$38370 \times 8.3145 = 319027\,\mathrm{kJ/mol}$) by contrast. Reactions (2.7b) and (2.7c) have been found to be much faster than reaction (2.7a). If simulation timescale is sufficiently long, one can assume that N_2, O_2, O and OH concentrations are at equilibrium and N atoms are in steady state, which simplifies the calculation of the NO formation. The reverse reaction can be further neglected by assuming that the NO concentration is much less than its equilibrium value. So the rate expression is:

$$\frac{\mathrm{d[NO]}}{\mathrm{d}t} = k_{1f}[\mathrm{O}][\mathrm{N_2}] + k_{2f}[\mathrm{N}][\mathrm{O_2}] + k_{3f}[\mathrm{N}][\mathrm{OH}] \tag{2.8a}$$

$$\frac{\mathrm{d[N]}}{\mathrm{d}t} = k_{1f}[\mathrm{O}][\mathrm{N_2}] - k_{2f}[\mathrm{N}][\mathrm{O_2}] - k_{3f}[\mathrm{N}][\mathrm{OH}] = 0 \tag{2.8b}$$

$$\Longrightarrow \quad \frac{\mathrm{d[NO]}}{\mathrm{d}t} = 2k_{1f}[\mathrm{O}][\mathrm{N_2}] \tag{2.8c}$$

2.3 Chemical reactor network

Computer-based simulation techniques are commonly used by gas turbine manufacturers in the field of new combustor designs. To describe pollutant formation accurately, around 10^2 species and 10^3 reactions must be considered besides the simplest fuels. Currently, it is too computationally expensive to include all phenomena of importance in a single accurate computational fluid dynamics (CFD) simulation for a combustion system. For the sake of compensating detailed chemistry, the chemical reactor network (CRN) approach has been developed with the cost of reduced treatment of fluid dynamics (Benedetto et al., 2000). A trend can be seen that emission simulation moves from the high-budget CFD approaches to CRN approaches where a network of idealized chemical reactors simplifies the computation for combustion modelling (Khodayari et al., 2020).

The CRN modelling approach is used for modelling combustion processes. The inner combustor space is divided into several regions. The deviation of physical and chemical properties is relatively small such that each region can be modelled as a node with detailed chemistry in the chemical reactor network. The physical flow of substances is captured by edges connecting nodes.

Chapter 3

Julia programming language

Julia is a high-level, high-performance, dynamic programming language designed by Bezanson et al. (2017). Many of its features are well suited for numerical and scientific computing. It is claimed that Julia can achieve machine performance without sacrificing human convenience, namely the so-called `Walks like Python, Runs like C`. The recent development of many Julia packages including various mathematical libraries, data manipulation tools and packages for general purpose computing enrich the Julia ecosystem for community use. In this project, we would like to apply Julia to model pollutant formation using the chemical reactor network approach. The following sections introduce the main relevant packages and how they contribute to our project.

3.1 DifferentialEquations.jl

DifferentialEquations.jl (Rackauckas & Nie, 2017) is a suite for numerically solving differential equations. It assists convenient ODE construction and works well together with Plots.jl to visualize solutions. One of the key features is that it has many built-in solver algorithms which supports diverse types of differential equations in different scales.

3.2 ModelingToolkit.jl

ModelingToolkit.jl (Ma et al., 2021) is a modeling language for high-performance symbolic-numeric computation in scientific computing. It allows for users to give a high-level description of a model for symbolic preprocessing to analyze and enhance the model. It is compatible with DifferentialEquations.jl as all the symbolic systems have a direct conversion to a numerical system which can then be handled through the SciML interfaces. ModelingToolkit.jl can be used to symbolicify numerical code and do symbolic calculation. More particularly, it is capable to automatically derive the Jacobian function of an `ODEProblem`.

3.3 LightGraphs.jl and SimpleWeightedGraphs.jl

To simulate the reactor network, special tools and data structures are needed to store and process graphs. LightGraphs.jl and SimpleWeightedGraphs.jl (Bromberger et al., 2017) offer a set of simple and graph implementations which make it easy for us to define arbitrary complex networks. Besides constructing directed, weighted networks by adjacency matrices, many built-in graph structures are available, such as the Barabási–Albert model.

3.4 NetworkDynamics.jl

NetworkDynamics.jl (Lindner et al., 2020) provides an interface between LightGraphs.jl and DifferentialEquations.jl. It allows to define several types of dynamic and static nodes and edges and to link them up in order to create complex network dynamics. In this project, the behavior of a node is the chemical reaction of fuel combustion. The transport phenomena between neighbor nodes is captured by the edges.

Chapter 4

Methane combustion simulation with Julia

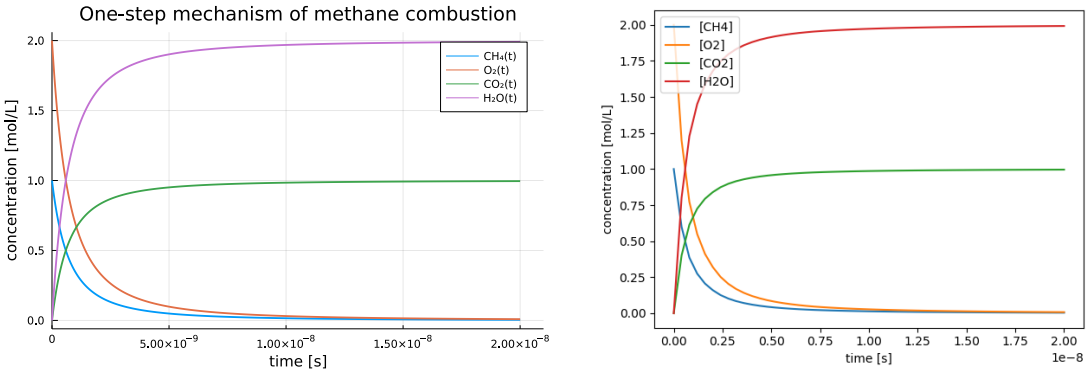
Julia is still relatively new with a 9-year history as of 2021. Despite the recent development of Julia's ecosystem, its application in combustion modelling remains to be explored. In order to examine the its capability, we start from simple combustion simulation of methane before modelling real physical networks. It is also important to test the correctness of Julia's packages by comparing results from other programming languages.

4.1 One-step mechanism

The one-step reaction mechanism of methane combustion is described in section 2.1. The package ModelingToolkit.jl is used in combination with DifferentialEquations.jl to define the ordinary differential equations of the mathematical model. Van der Linden (2020) adopted the forward Euler scheme with Python to simulate the reaction. The parameters and initial conditions are kept the same which are summarized in table 4.1. The Julia implementation can be found in appendix A.1 on page 30. The results shown in figure 4-1 are the same which indeed demonstrate the basic functions of DifferentialEquations.jl and ModelingToolkit.jl.

Parameters and inputs	Values
Activation energy E_a	2×10^4
Temperature exponent β	0.0
Pre-exponential factor A	1.1×10^{10}
Reaction exponents	n_{CH_4} 1.0
	n_{O_2} 0.5
Start time t_0	0.0
End time t_n	2×10^{-8}
Constant temperature T	1000
Initial values	$[CH_4]$ 1.0
	$[O_2]$ 2.0
	$[CO_2]$ 0.0
	$[H_2O]$ 0.0

Table 4.1: Parameters and inputs for the one-step reaction mechanism of methane combustion



(a) Julia (b) Python (Van der Linden, 2020, p. 26)

Figure 4-1: One-step mechanism of methane combustion

4.2 Two-step mechanism

The combustion of methane is actually a sequence of elementary reactions rather than the straightforward one-step mechanism. The 2S_CH4_BFER scheme by Franzelli et al. (2012) considered in this section is explained in subsection 2.2.1. We would like

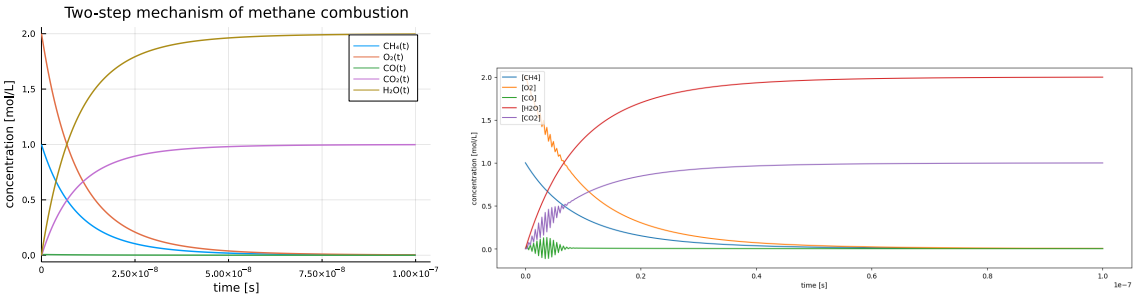
to find an approximation solution to the following model for $0 < t < 10^{-7}$.

$$\frac{\mathrm{d}[\mathrm{CH_4}]}{\mathrm{d}t} = -k_1, \tag{4.1a}$$
$$\frac{\mathrm{d}[\mathrm{O_2}]}{\mathrm{d}t} = -1.5k_1 - 0.5k_2, \tag{4.1b}$$
$$\frac{\mathrm{d}[\mathrm{CO}]}{\mathrm{d}t} = k_1 - k_2, \tag{4.1c}$$
$$\frac{\mathrm{d}[\mathrm{CO_2}]}{\mathrm{d}t} = k_2, \tag{4.1d}$$
$$\frac{\mathrm{d}[\mathrm{H_2O}]}{\mathrm{d}t} = 2k_1. \tag{4.1e}$$

The same parameters in table 2.1 on page 5 are applied. We begin from the simulation with the initial conditions in table 4.2 to compare the solutions with the existing Python implementation.

Species	Initial values
$\mathrm{CH_4}$	1.0
$\mathrm{O_2}$	2.0
CO	0.0
$\mathrm{CO_2}$	0.0
$\mathrm{H_2O}$	0.0

Table 4.2: Inputs for the two-step reaction mechanism of methane combustion



(a) Julia (b) Python (Van der Linden, 2020, p. 28)

Figure 4-2: Two-step mechanism of methane combustion

The results in figure 4-2 look the same for both implementations except the small fluctuation in Python's figure 4-2b due to the numerical methods it adopted. Note that while we are investigating pollutant formation, it is not particularly obvious from

the figure that the carbon monoxide (CO) is produced. Figure 4-3 is a separate graph for CO. The concentration of CO jumps sharply away from 0 immediately after the reaction starts.

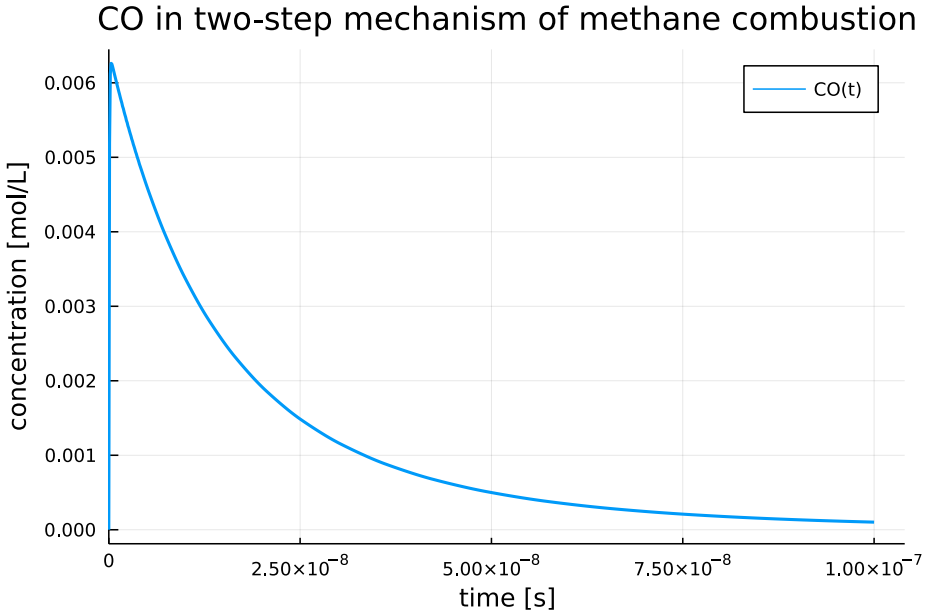


Figure 4-3: Carbon monoxide in two-step mechanism of methane combustion

Chapter 5

Chemical reactor network with Julia

This chapter presents solving reactor networks with Julia. It starts from the formal mathematical model. Application on 5-node 5-edge networks is illustrated. At the end, the performance of solving large networks with various solver algorithms are evaluated and compared.

5.1 Mathematical model

As briefly introduced in section 2.3, an internal combustor space is modelled as a network of graph G with n vertices V and some edges E. A vertex v is a container with chemical reactions perfectly stirred which only involve chemistry. An edge e is a pipe serving as a channel for flowing species with prescribed velocity which captures transport.

Each vertex v_i contains s chemical species. Directed edge $e_{i,j}$ connects vertex v_i and vertex v_j. $\mathbf{u}(t)$ is a vector of unknown species concentration.

$$\mathbf{u}(t) = \begin{pmatrix} \mathbf{u}_1(t) \\ \vdots \\ \mathbf{u}_i(t) \\ \vdots \\ \mathbf{u}_n(t) \end{pmatrix} \tag{5.1}$$

$$\mathbf{u}_i(t) = \begin{pmatrix} u_{i,1}(t) \\ \vdots \\ u_{i,j}(t) \\ \vdots \\ u_{i,s}(t) \end{pmatrix} \tag{5.2}$$

where $\mathbf{u}_i(t)$ is a subvector corresponding to vertex v_i. $u_{i,j}$ denotes the concentration of the j-th species in vertex v_i.

The dynamics of the entire system is modelled by a set of ordinary differential equations (ODEs)

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} = \mathbf{F}(\mathbf{u}(t), t) = \mathbf{F}_{\text{chemistry}}(\mathbf{u}(t), t) + \mathbf{F}_{\text{transport}}(\mathbf{u}(t), t) \tag{5.3}$$

with prescribed initial conditions

$$\mathbf{u}(t = 0) = \mathbf{u}_0. \tag{5.4}$$

The aim is to solve the large-scale ODE system with sufficient accuracy for a certain time range. The differential equations are stiff as it involves fast chemistry and slow transport, for which we will apply implicit time integration methods with variable time steps.

5.2 Implicit time-integration method

Choosing a good solver is required to getting top speed. Figure 4-3 on page 13 from the results of the two-step reaction mechanism of methane combustion reveals the stiff nature of the CRN problem. Explicit methods generally demand impractically small time steps Δt to keep the error bounded. We thus apply implicit time-integration methods which take less computational time to achieve given accuracy with larger time steps, even taking into account that extra computations are needed at each time step. The documentation of DifferentialEquations.jl recommends Rosenbrock methods for smaller problems and ESDIRK methods for slightly larger problems which are tested and evaluated below.

5.3 5-node-5-edge networks

In this section, we apply Julia to solve networks consisting of 5 nodes and 5 edges in two different connectivity configurations. The code can be found in appendix A.3 on page 32. Each vertex in the graph is a node with the one-step reaction mechanism (2.1) of methane combustion as its internal chemistry. The ODE system is defined as follows, $\forall i \in \{1, 2, 3, 4, 5\}$,

$$\frac{\mathrm{d}\mathbf{u}_i(t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \\ u_{i,4} \end{pmatrix} = k u_{i,1}^{n_{\mathrm{CH_4}}} u_{i,2}^{n_{\mathrm{O_2}}} \begin{pmatrix} -1 \\ -2 \\ 1 \\ 2 \end{pmatrix} + \sigma \sum_{j=1}^{5} G_{ji}(\mathbf{u}_j - \mathbf{u}_i) \tag{5.5}$$

where $u_{i,1}$, $u_{i,2}$, $u_{i,3}$, $u_{i,4}$ denote the concentration of CH_4, O_2, CO_2 and H_2O in the i-th vertex respectively. The rate coefficient $k = AT^{\beta}\exp(-E_a/RT)$, the reaction orders $n_{\mathrm{CH_4}}$ and $n_{\mathrm{O_2}}$ are the same with the values in table 4.1 on page 11. The graph structure and the flow rate are captured by the parameters σ and G_{ji}.

Moreover, as for the initial condition, there are fuels in the first node and none in other nodes at the start time. That is, methane is injected into node 1 only at the

outset and flow travels downstream to other nodes. Node 1 acts as an inlet for CH_4. The initial values of O_2 in nodes 1, 3, 5 are set to zero.

5.3.1 Main flux and recirculation

We firstly consider the network shown in figure 5-1. The vertices 1, 2, 3, 4 reside in the same line. The vertex 5, the edges 4 and 5 creates a cycle of circulation.

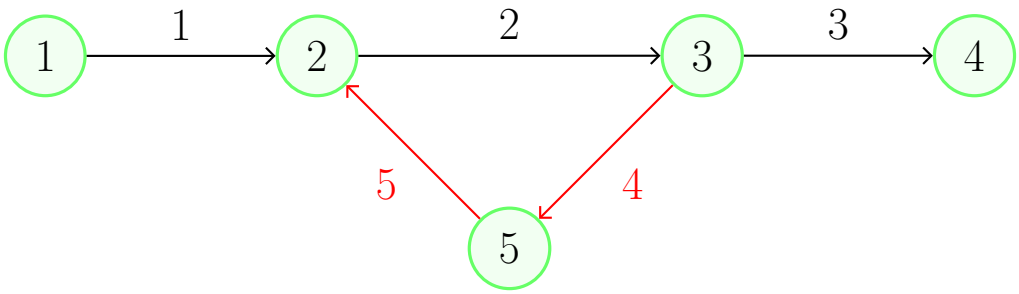


Figure 5-1: Graph of 5-node 5-edge network with main flux and recirculation

The adjacency matrix is

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.6}$$

where the numbers outside the border of the matrix indicate the indices of vertices. The element G_{ji} on the j-th row and the i-th column is 1 if there exists a directed edge from vertex j to vertex i and 0 otherwise.

Figures 5-2, 5-3 and 5-4 display the evolution of the substances in the network. As we can see, the fuel flows from node 1 to other nodes. The oxygen diffuses to the entire network except node 1. The reaction of methane combustion gradually take place in nodes 2, 3, 4, and 5.

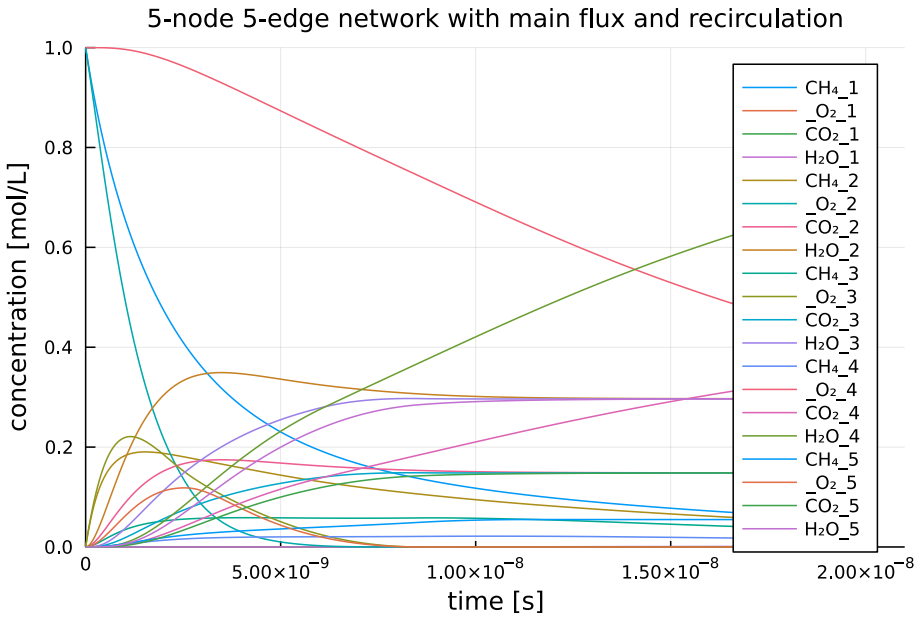
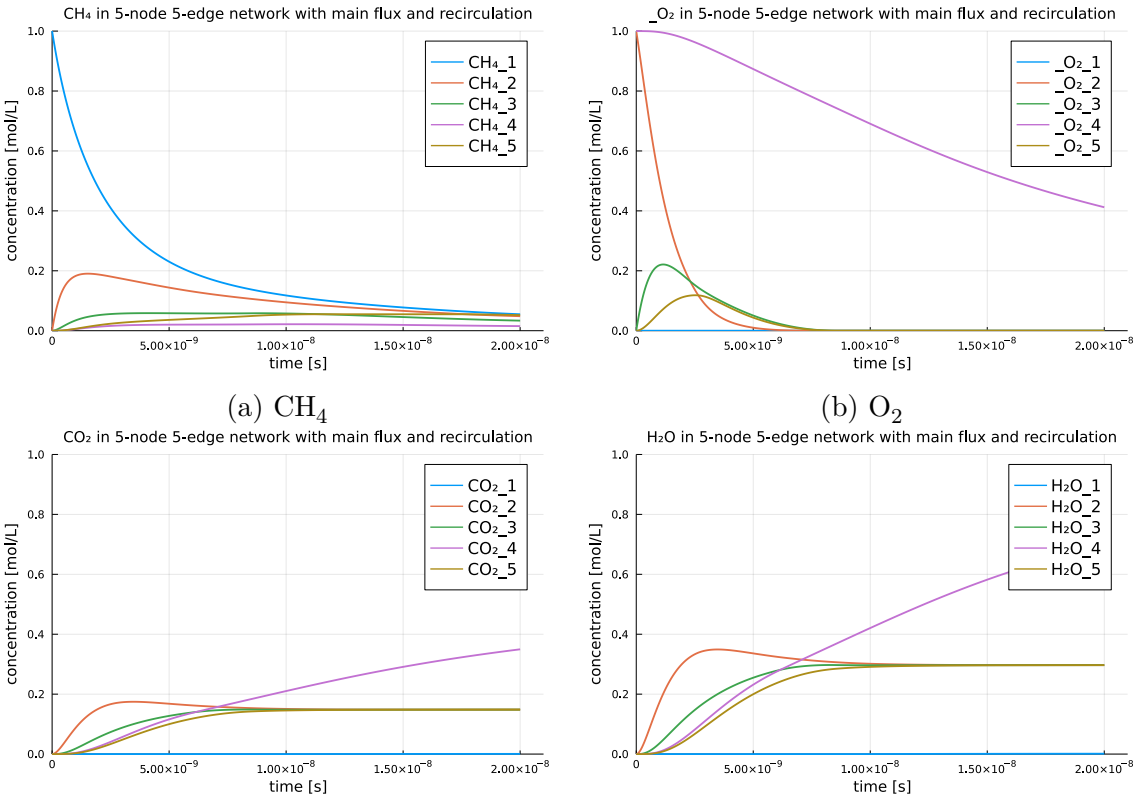


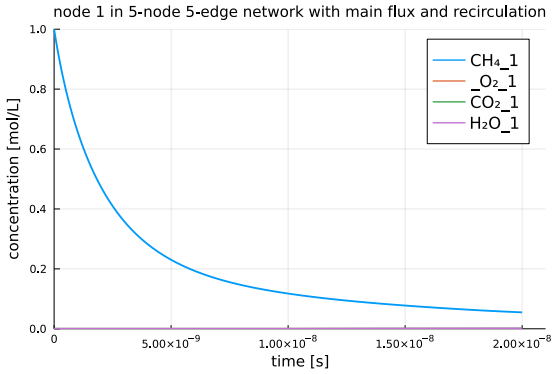
Figure 5-2: 5-node 5-edge network with main flux and recirculation



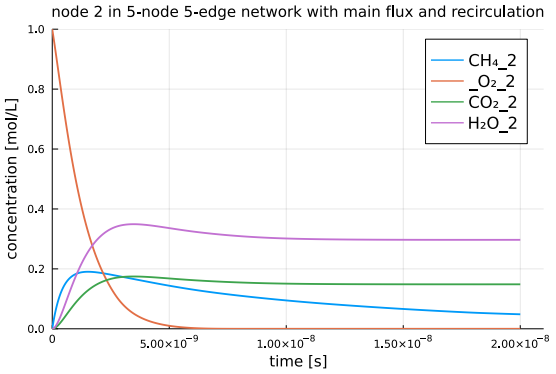
(a) CH_4 (b) O_2

(c) CO_2 (d) H_2O

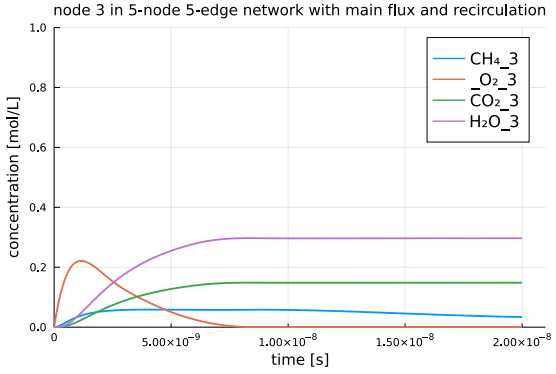
Figure 5-3: Species in 5-node 5-edge network with main flux and recirculation



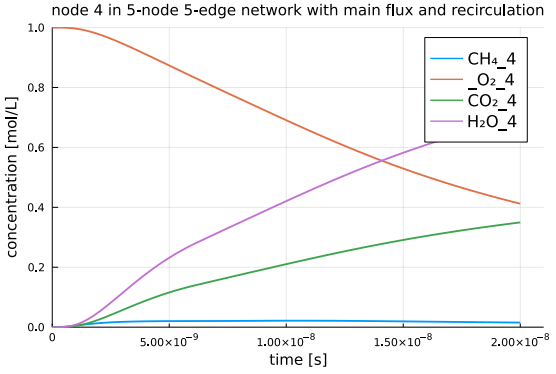
(a) Node 1



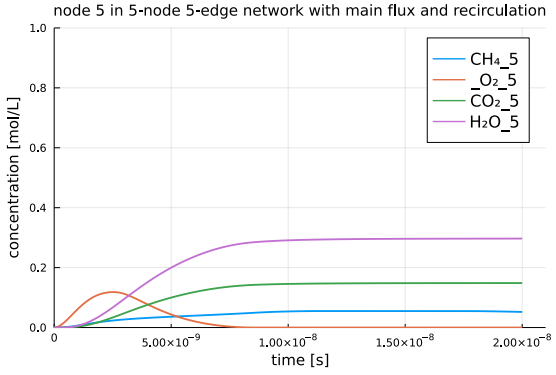
(b) Node 2



(c) Node 3



(d) Node 4



(e) Node 5

Figure 5-4: Nodes in 5-node 5-edge network with main flux and recirculation

5.3.2 Main flux and by-pass

Figure 5-5 shows the structure of the 5-node-5-edge network with main flux and by-pass. Edges 4 and 5 build a by-pass for edge 2. The adjacency matrix in this case becomes

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}. \tag{5.7}$$

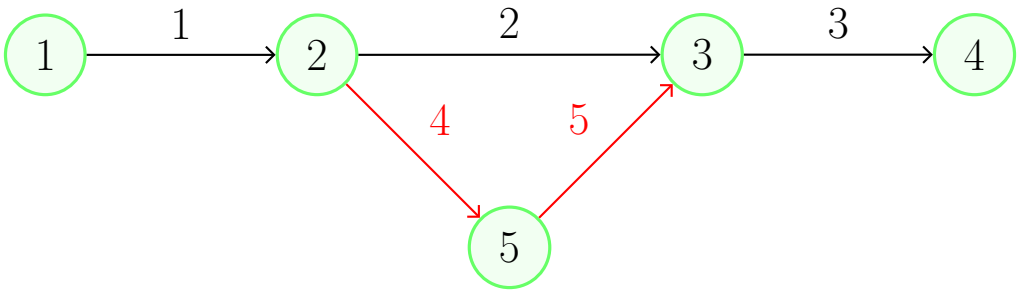


Figure 5-5: Graph of 5-node 5-edge network with main flux and by-pass

The results are illustrated in figures 5-6, 5-7 and 5-8. Compared to the network with recirculation, the oxygen in node 2 gets consumed and spreads to other nodes more quickly as there is an additional edge originating from node 2 and no flow traveling back to it. The concentration of CO_2 and H_2O are higher in node 2 in the network with recirculation than the that with by-pass. As a whole, less substances react in node 2.

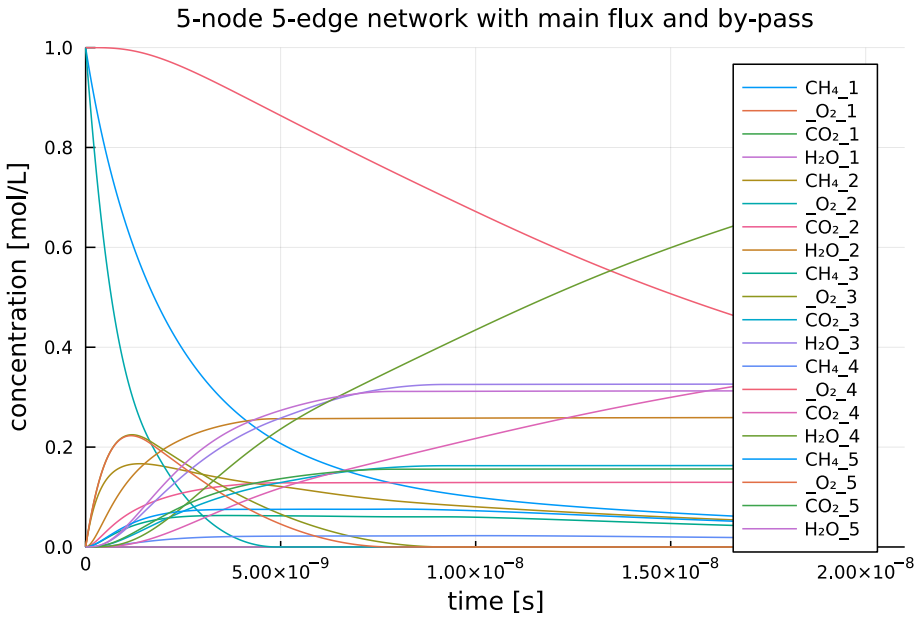
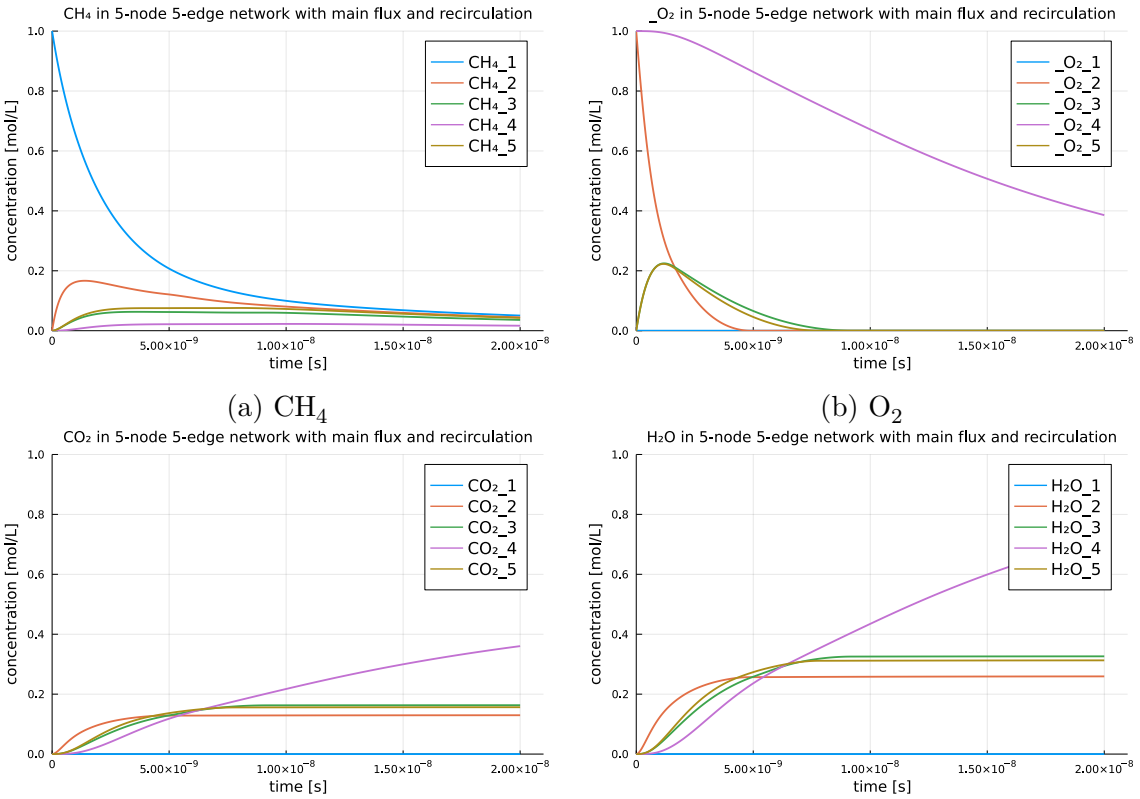


Figure 5-6: 5-node 5-edge network with main flux and by-pass



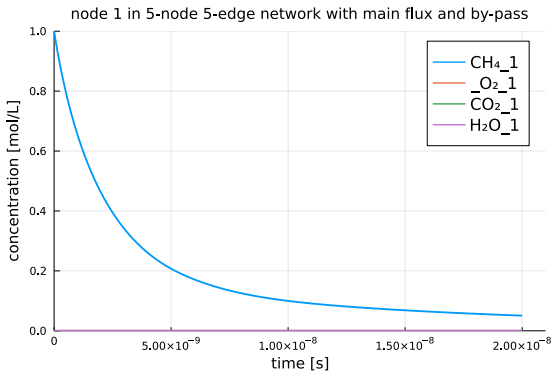
(a) CH_4

(b) O_2

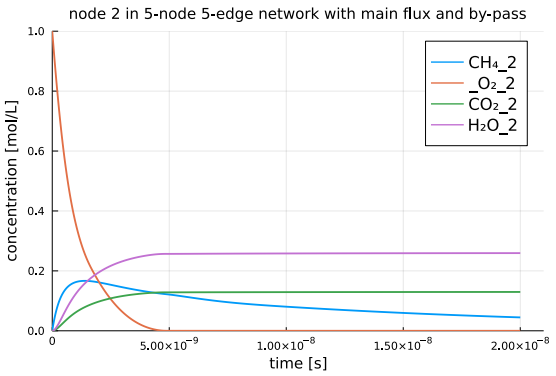
(c) CO_2

(d) H_2O

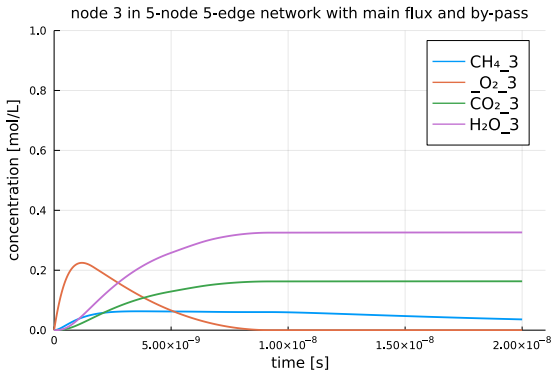
Figure 5-7: Species in 5-node 5-edge network with main flux and by-pass



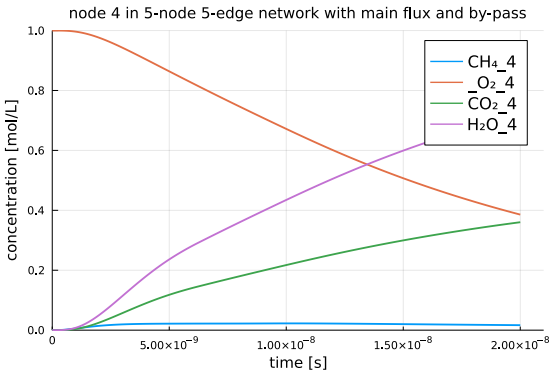
(a) Node 1



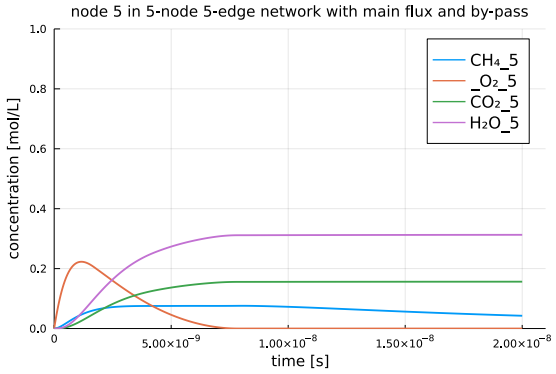
(b) Node 2



(c) Node 3



(d) Node 4

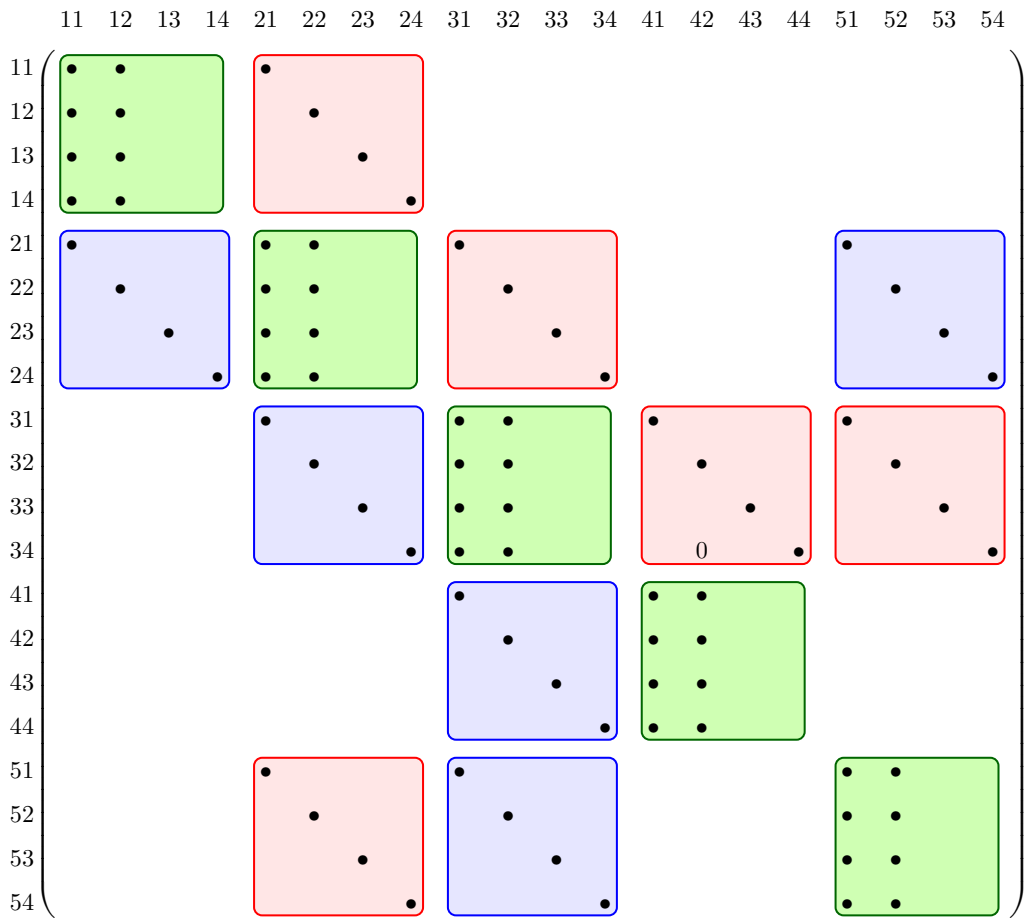


(e) Node 5

Figure 5-8: Nodes in 5-node 5-edge network with main flux and by-pass

5.4 Jacobian sparsity

Solving stiff ordinary differential equations generally requires very small step size for numerical stability. It is crucial to improve performance and avoid inefficiency when dealing with large problems. According to the Julia package DifferentialEquations.jl, Jacobian is built at each iteration when implicit differential equation solver is used which is computationally expensive. The efficiency can be optimized by declaring Jacobian functions to reduce the Jacobian construction cost. In this way, we provide extra information about the ODE system to the solver. On the other hand, one can exploit the sparsity pattern of Jacobian. For example, the sparsity structure of the Jacobian for the 5-node-5-edge network with main flux and recirculation is



(5.8)

where the non-zero elements are shown as black bullets •. The green blocks in the diagonal represent the internal chemistry whose reaction rate depends on the concen-

tration of the two reactants CH_4 and O_2 in our mathematical model. The red and blue blocks correspond to the forward and backward directions of edges respectively.

For large networks, most of the entries in the Jacobian are zero. Therefore, special data structures could be invoked to speed up Jacobian calculation and reduce memory usage. Here, the `SparseArrays` stdlib module comes into play which helps save space and execution time for arrays that contain enough zeros compared to dense arrays[1].

5.5 Performance

We solved the one- and the two-step reaction mechanism of methane combustion with the default solver algorithm[2] and applied the 5th order A-stable stiffly stable Rosenbrock method `Rodas5` to solve the 5-node-5-edge networks. So far only small (<50 ODEs) problems are examined. However, in order to utilize Julia on real combustion simulation, we shall test its capability on solving large reactor networks with more species beforehand.

Besides `Rodas5`, we consider the second order A-B-L-S-stable one-step ESDIRK method `TRBDF2` which is suggested by DifferentialEquations.jl documentation for slightly larger problems (with <2000 ODEs). Furthermore, it is possible to optimize the linear solvers for certain differential equation integration methods. We also evaluate the performance of `TRBDF2` in combination with the GMRES linear solver.

For simplicity, we employ the test network shown in figure 5-9. All nodes reside in the same line with a directed edge connecting every pair of consecutive nodes. The internal chemistry in each node is the one-step mechanism of methane combustion. The initial values are gradually decreasing from node 1 to node n. The parameters are kept the same as the 5-node-5-edge networks discussed in section 5.3.

Figure 5-10 shows the result of execution times and allocated memory. `TRBDF2` significantly outperforms `Rodas5` for large problems. The GMRES linear solver further

[1]Note that Jacobian sparsity should only be declared if the sparsity if high, for example, 99% of the matrix elements are zeros, otherwise the overhead of sparse data structures may be higher than the gains.

[2]Julia DifferentialEquations.jl automatically chooses an efficient method for the input problem if no solver is selected by user.

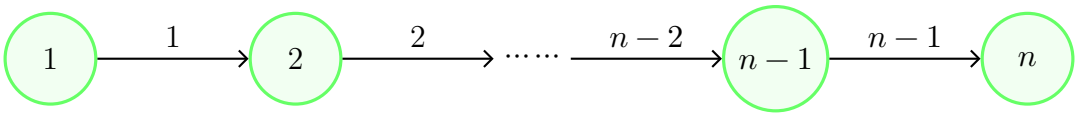
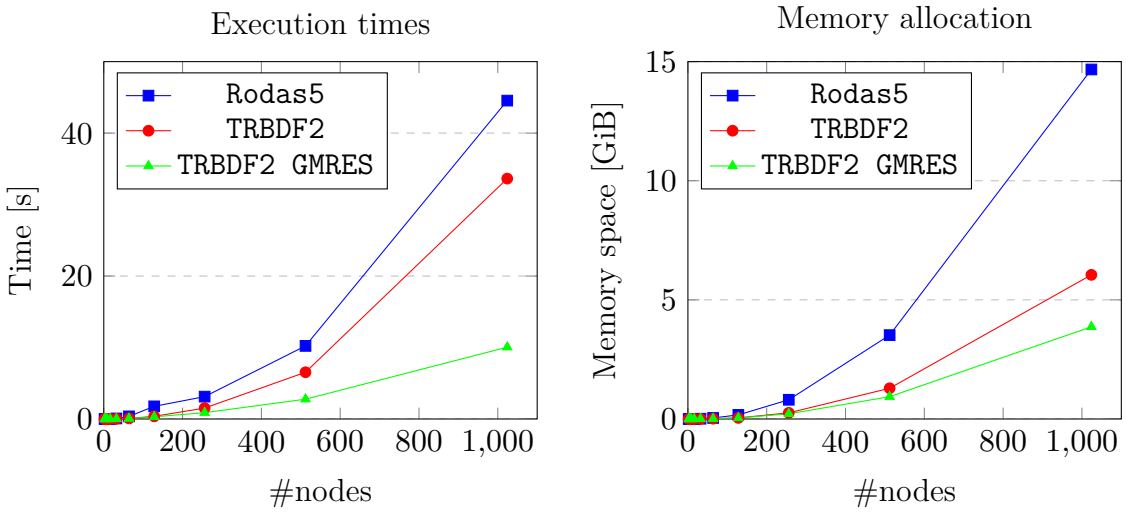


Figure 5-9: Graph of test network

aids in speeding up and reducing memory usage.



(a) Execution times

(b) Memory allocation

Figure 5-10: Performance of Julia for solving reactor networks

Chapter 6

Conclusion

The research was initiated with the growing demand for a computational tool that is able to simulate large chemical reactor networks and to predict pollutant emission. We applied the Julia programming language to solve the one- and the two-step mechanism of methane combustion, the 5-node-5-edge networks and large line-shaped networks.

The results of performance measurement prove that Julia with its ecosystem is competent in simulating large network dynamics up to 1000 nodes with 4 species within reasonable time as long as appropriate numerical solvers are chosen. However, unlike C/C++ and Python, Julia has a rather shorter history. The Julia core code and its packages are still under heavy development thanks to the common effort from the open-source Julia community. It can be foreseen that Julia will be shortly utilized to speed up CRN simulations in industry and to estimate emissions more accurately for industrial furnaces.

Bibliography

Benedetto, D., Pasini, S., Falcitelli, M., La Marca, C., & Tognotti, L. (2000). NO_X emission prediction from 3-d complete modelling to reactor network analysis. *Combustion Science and Technology*, *153*(1), 279–294. https://doi.org/10.1080/00102200008947265

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bromberger, S., Fairbanks, J., & other contributors. (2017). Juliagraphs/lightgraphs.jl: An optimized graphs package for the Julia programming language. https://doi.org/10.5281/zenodo.889971

Chenoweth, J. A., Albertson, T. E., & Greer, M. R. (2021). Carbon monoxide poisoning [Toxicology]. *Critical Care Clinics*, *37*(3), 657–672. https://doi.org/10.1016/j.ccc.2021.03.010

Franzelli, B., Riber, E., Gicquel, L. Y. M., & Poinsot, T. (2012). Large eddy simulation of combustion instabilities in a lean partially premixed swirled flame. *Combustion and Flame*, *159*(2), 621–637. https://doi.org/10.1016/j.combustflame.2011.08.004

Hamra, G. B., Laden, F., Cohen, A. J., Raaschou-Nielsen, O., Brauer, M., & Loomis, D. (2015). Lung cancer and exposure to nitrogen dioxide and traffic: A systematic review and meta-analysis. *Environmental Health Perspectives*, *123*(11), 1107–1112. https://doi.org/10.1289/ehp.1408882

Jones, W. P., & Lindstedt, R. P. (1988). Global reaction schemes for hydrocarbon combustion. *Combustion and Flame*, *73*(3), 233–249. https://doi.org/10.1016/0010-2180(88)90021-1

Khodayari, H., Ommi, F., & Saboohi, Z. (2020). A review on the applications of the chemical reactor network approach on the prediction of pollutant emissions. *Aircraft Engineering and Aerospace Technology*, *92*(4), 551–570. https://doi.org/10.1108/AEAT-08-2019-0178

Lindner, M., Lincoln, L., Drauschke, F., Koulen, J. M., Würfel, H., Plietzsch, A., & Hellmann, F. (2020). Networkdynamics.jl - Composing and simulating complex networks in Julia. https://arxiv.org/abs/2012.12696

Ma, Y., Gowda, S., Anantharaman, R., Laughman, C., Shah, V., & Rackauckas, C. (2021). Modelingtoolkit: A composable graph transformation system for equation-based modeling. https://arxiv.org/abs/2103.05244

Pommier, M., McLinden, C. A., & Deeter, M. (2013). Relative changes in CO emissions over megacities based on observations from space. *Geophysical Research Letters*, *40*(14), 3766–3771. https://doi.org/10.1002/grl.50704

Rackauckas, C., & Nie, Q. (2017). Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. *The Journal of Open Research Software*, *5*(1), 15. https://doi.org/10.5334/jors.151

Turns, S. R. (2012). *An Introduction to Combustion: Concepts and Applications* (3rd ed.). McGraw-Hill. https://www.mheducation.com/highered/product/introduction-combustion-concepts-applications-turns/M9780073380193.html

Van der Linden, L. (2020). *The parareal algorithm on the model for combustion of methane*. Delft University of Technology. http://resolver.tudelft.nl/uuid:6839f1d6-002a-4415-b68b-848505cf6ada

Zel'dovich, Y. B. (1946). The oxidation of nitrogen in combustion explosions. *Acta Physicochimica U.S.S.R.*, *21*, 577–628.

Appendix A

Julia implementation

The project file `Project.toml` including dependencies and versions is shown below.

```
[deps]
BenchmarkTools = "6e4b80f9-dd63-53aa-95a3-0cdb28fa8baf"
DifferentialEquations = "0c46a032-eb83-5123-abaf-570d42b7fbaa"
IncompleteLU = "40713840-3770-5561-ab4c-a76e7d0d7895"
LightGraphs = "093fc24a-ae57-5d10-9952-331d41423f4d"
ModelingToolkit = "961ee093-0014-501f-94e3-6117800e7a78"
NetworkDynamics = "22e9dc34-2a0d-11e9-0de0-8588d035468b"
Plots = "91a5bcdd-55d7-5caf-9e0b-520d859cae80"
SimpleWeightedGraphs = "47aef6b3-ad0c-573a-a1e2-d07658019622"
SparsityDetection = "684fba80-ace3-11e9-3d08-3bc7ed6f96df"

[compat]
BenchmarkTools = "1.0.0"
DifferentialEquations = "6.17.1"
IncompleteLU = "0.2.0"
LightGraphs = "1.3.5"
ModelingToolkit = "5.20.0"
NetworkDynamics = "0.5.4"
Plots = "1.16.5"
SimpleWeightedGraphs = "1.1.1"
SparsityDetection = "0.3.4"
julia = "1.6.1"
```

A.1 One-step mechanism of methane combustion

```julia
using ModelingToolkit, DifferentialEquations, Plots

E = 2e4             # activation energy
β = 0.0             # temperature exponent
A = 1.1e10          # pre-exponential factor
R = 8.3145          # gas constant
T = 1e3             # temperature in Kelvin

@variables t CH₄(t) O₂(t) CO₂(t) H₂O(t) r(t)
@parameters k, n_CH₄, n_O₂
D = Differential(t)

u₀ = [CH₄ => 1.0,  # initial value CH₄
      O₂  => 2.0,  # initial value O₂
      CO₂ => 0.0,  # initial value CO₂
      H₂O => 0.0]  # initial value H₂O

tspan = (0.0,      # start time t₀
         2e-8)     # end time tₙ

p = [k => A * T^β * exp(-E / R / T), # rate coefficient
     n_CH₄ => 1.0, # reaction exponent CH₄
     n_O₂ => 0.5]  # reaction exponent O₂

@named fol_separate = ODESystem([ r ~ CH₄^n_CH₄ * O₂^n_O₂,
                                  D(CH₄) ~ -k * r,
                                  D(O₂)  ~ -2k * r,
                                  D(CO₂) ~ k * r,
                                  D(H₂O) ~ 2k * r ])

prob = ODEProblem(structural_simplify(fol_separate), u₀, tspan, p)
sol = solve(prob)
plot(sol, linewidth = 2, xlims = (0., 2.1e-8),
     title = "One-step mechanism of methane combustion",
     xlabel = "time [s]",
     ylabel = "concentration [mol/L]")
savefig("res/one_step_mechanism_methane.svg")
```

A.2 Two-step mechanism of methane combustion

```julia
using ModelingToolkit, DifferentialEquations, Plots

E₁ = 3.55e4        # activation energy
E₂ = 1.2e4
β₁ = 0.0           # temperature exponent
β₂ = 0.8
A₁ = 4.9e9         # pre-exponential factor
A₂ = 2e8
R  = 8.3145        # gas constant
T  = 1e3           # temperature in Kelvin

@variables t CH₄(t) O₂(t) CO(t) CO₂(t) H₂O(t) r₁(t) r₂(t)
@parameters k₁, k₂, n_CH₄, n_CO, n_O₂_₁, n_O₂_₂
D = Differential(t)
u₀ = [CH₄ => 1.0,  # initial value CH₄
      O₂  => 2.0,  # initial value O₂
      CO  => 0.0,  # initial value CO
      CO₂ => 0.0,  # initial value CO₂
      H₂O => 0.0]  # initial value H₂O
tspan = (0.0,      # start time t₀
         1e-7)     # end time tₙ
p = [k₁ => A₁ * T^β₁ * exp(-E₁ / R / T), # rate coefficient 1
     k₂ => A₂ * T^β₂ * exp(-E₂ / R / T), # rate coefficient 2
     n_CH₄  => 0.5,  # reaction exponent CH₄
     n_CO   => 1.0,  # reaction exponent CO
     n_O₂_₁ => 0.65, # reaction exponent O₂ 1
     n_O₂_₂ => 0.5]  # reaction exponent O₂ 2
@named fol_separate = ODESystem([ r₁ ~ CH₄^n_CH₄ * O₂^n_O₂_₁,
                                  r₂ ~ CO^n_CO * O₂^n_O₂_₂,
                                  D(CH₄) ~ -k₁ * r₁,
                                  D(O₂)  ~ -1.5k₁ * r₁ - 0.5k₂ * r₂,
                                  D(CO)  ~ k₁ * r₁ - k₂ * r₂,
                                  D(CO₂) ~ k₂ * r₂,
                                  D(H₂O) ~ 2k₁ * r₁ ])
prob = ODEProblem(structural_simplify(fol_separate), u₀, tspan, p)
sol = solve(prob)
plot(sol, linewidth = 2, xlims = (0., 1.04e-7),
     title="Two-step mechanism of methane combustion",
     xlabel="time [s]",
     ylabel="concentration [mol/L]")
savefig("res/two_step_mechanism_methane.svg")

```

```
plot(sol, linewidth = 2, xlims = (0., 1.04e-7), vars = (0, 3),
     title="CO in two-step mechanism of methane combustion",
     xlabel="time [s]",
     ylabel="concentration [mol/L]")
savefig("res/two_step_mechanism_methane_CO.svg")
```

A.3 5-node-5-edge network

```
using SimpleWeightedGraphs, LightGraphs, NetworkDynamics,
      DifferentialEquations, Plots

E = 2e4             # activation energy
β = 0.              # temperature exponent
A = 1.1e10          # pre-exponential factor
R = 8.3145          # gas constant
T = 1e3             # temperature in Kelvin
k = A * T^β * exp(-E / R / T) # rate coefficient
σ = 5e8             # constant for edges
n_CH₄ = 1.          # reaction order CH₄
n_O₂  = 0.5         # reaction order O₂

n_node    = 5
n_species = 4

# adjacency matrix
G = [0. 1. 0. 0. 0.  # main flux and recirculation
     0. 0. 1. 0. 0.
     0. 0. 0. 1. 1.
     0. 0. 0. 0. 0.
     0. 1. 0. 0. 0.]
config = "recirculation"
# G = [0. 1. 0. 0. 0.  # main flux and by-pass
#      0. 0. 1. 0. 1.
#      0. 0. 0. 1. 0.
#      0. 0. 0. 0. 0.
#      0. 0. 1. 0. 0.]
# config = "by-pass"

g_weighted = SimpleWeightedDiGraph(G - G')
edge_weights = getfield.(collect(edges(g_weighted)), :weight)
```

```julia
g_directed = SimpleDiGraph(g_weighted)

@inline Base.@propagate_inbounds function chemistry!(du, u, edges, p, t)
    if u[1] < 0.
        u[1] = 0.
    end
    if u[2] < 0.
        u[2] = 0.
    end
    rate = k * u[1]^n_CH₄ * u[2]^n_O₂
    du[1] = -rate    # dCH₄
    du[2] = -2rate   # dO₂
    du[3] = rate     # dCO₂
    du[4] = 2rate    # dH₂O
    for e in edges
        du .+= e
    end
    nothing
end

@inline Base.@propagate_inbounds function transport!(e, u_s, u_d, p, t)
    e .= 0.
    if p > 0.     # forward  direction of directed edge
        for i in 1:n_species
            if u_s[i] > u_d[i]
                e[i] = p * (u_s[i] - u_d[i])
            end
        end
    elseif p < 0. # backward direction of directed edge
        for i in 1:n_species
            if u_d[i] > u_s[i]
                e[i] = p * (u_d[i] - u_s[i])
            end
        end
    end
    nothing
end

node = ODEVertex(f! = chemistry!, dim = n_species,
                 sym=[:CH₄, :_O₂, :CO₂, :H₂O])
edge = StaticEdge(f! = transport!, dim = n_species,
                  coupling = :directed)
network! = network_dynamics(node, edge, g_directed)

```

```
p = (nothing, σ * edge_weights)
u₀ = zeros(n_species * n_node)
u₀[1] = 1.            # initial value CH₄ in node 1
u₀[2n_species-2] = 1. # initial value O₂  in node 2
u₀[4n_species-2] = 1. # initial value O₂  in node 4
tspan = (0.,          # start time t₀
         2e-8)        # end time tₙ
prob = ODEProblem(network!, u₀, tspan, p)
sol = solve(prob, Rodas5(autodiff=false))

plot(sol, linewidth = 1, titlefont = 11,
     xlims = (0., 2.1e-8), ylims = (0., 1.),
     title = "5-node 5-edge network with main flux and $(config)",
     ylabel = "concentration [mol/L]",
     xlabel = "time [s]")
savefig("res/$(config).svg")

for n in 1:5
    plot(sol, linewidth = 2,
         titlefont = 11, legendfont = 12,
         xlims = (0., 2.1e-8), ylims = (0., 1.),
         vars = syms_containing(network!, string(n)),
         title = "node $(n) in 5-node 5-edge network "
               * "with main flux and $(config)",
         ylabel = "concentration [mol/L]",
         xlabel = "time [s]")
    savefig("res/$(config)$(n).svg")
end

for s in ["CH₄", "_O₂", "CO₂", "H₂O"]
    plot(sol, linewidth = 2,
         titlefont = 11, legendfont = 12,
         xlims = (0., 2.1e-8), ylims = (0., 1.),
         vars = idx_containing(network!, s),
         title = "$(s) in 5-node 5-edge network "
               * "with main flux and $(config)",
         ylabel = "concentration [mol/L]",
         xlabel = "time [s]")
    savefig("res/$(config)$(s).svg")
end
```

A.4 Performance measurement

```
using SimpleWeightedGraphs, LightGraphs, NetworkDynamics,
      DifferentialEquations, Plots, SparseArrays, BenchmarkTools

E = 2e4             # activation energy
β = 0.              # temperature exponent
A = 1.1e10          # pre-exponential factor
R = 8.3145          # gas constant
T = 1e3             # temperature in Kelvin
k = A * T^β * exp(-E / R / T) # rate coefficient
σ = 5e8             # constant for edges
n_CH₄ = 1.          # reaction order CH₄
n_O₂  = 0.5         # reaction order O₂
tspan = (0.,        # start time t₀
         2e-8)      # end time tₙ
n_species = 4

@inline Base.@propagate_inbounds function chemistry!(du, u, edges, p, t)
    if u[1] < 0.
        u[1] = 0.
    end
    if u[2] < 0.
        u[2] = 0.
    end
    rate = k * u[1]^n_CH₄ * u[2]^n_O₂
    du[1] = -rate    # dCH₄
    du[2] = -2rate   # dO₂
    du[3] = rate     # dCO₂
    du[4] = 2rate    # dH₂O
    for e in edges
        du .+= e
    end
    nothing
end

@inline Base.@propagate_inbounds function transport!(e, u_s, u_d, p, t)
    e .= 0.
    if p > 0.     # forward  direction of directed edge
        for i in 1:4
            if u_s[i] > u_d[i]
                e[i] = p * (u_s[i] - u_d[i])
            end
        end
```

```
    elseif p < 0. # backward direction of directed edge
        for i in 1:4
            if u_d[i] > u_s[i]
                e[i] = p * (u_d[i] - u_s[i])
            end
        end
    end
    nothing
end

node = ODEVertex(f! = chemistry!, dim = n_species,
                 sym=[:CH₄, :_O₂, :CO₂, :H₂O])
edge = StaticEdge(f! = transport!, dim = n_species,
                  coupling = :directed)

function measure(n_node, solver)
    println("\n$(n_node) nodes\n$(solver)")
    # adjacency matrix
    G = spdiagm(n_node, n_node, 1 => ones(n_node-1))
    g_weighted = SimpleWeightedDiGraph(G - G')
    edge_weights = getfield.(collect(edges(g_weighted)), :weight)
    g_directed = SimpleDiGraph(g_weighted)
    network! = network_dynamics(node, edge, g_directed)

    p = (nothing, σ * edge_weights)
    u₀ = zeros(4n_node)
    init = collect(range(1., 0., length=n_node))
    for i in 1:n_node
        u₀[4i-3] = init[i]  # initial value CH₄ in node i
        u₀[4i-2] = 2init[i] # initial value O₂  in node i
    end
    prob = ODEProblem(network!, u₀, tspan, p)
    @btime solve($prob, $solver, save_everystep=false)
    nothing
end

for n_node in [2^i for i in 1:10]
    for solver in [Rodas5(autodiff=false), TRBDF2(autodiff=false),
                   TRBDF2(autodiff=false, linsolve=LinSolveGMRES())]
        measure(n_node, solver)
    end
end
```