

A NOVEL ATOM TRACKING ALGORITHM FOR THE
ANALYSIS OF COMPLEX CHEMICAL KINETIC
NETWORKS: APPLICATION TO SOOT PRECURSORS
FORMATION IN MULTICOMPONENT FUEL COMBUSTION

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Christopher James Frewin

February 2016

© 2015 Christopher James Frewin

ALL RIGHTS RESERVED

:

ABSTRACT

Bio-derived fuels are expected to progressively displace conventional fuels in an effort to reduce dependence on crude oil and mitigate environmental impact. Bio-fuels are very diverse, leading to new opportunities to tailor fuel blends to achieve desirable properties, such as low emissions. Beyond fuel formulation, a key step is to understand how fuel molecules with very different chemical functionalities interact during combustion to eventually be able to better control emission formation. As a first step to address these challenges, a novel numerical tracking algorithm is presented that allows tracking of specific atoms during homogeneous combustion as they are transferred from reactants to products throughout a chemical kinetic network. Tracking is performed by labeling individual atoms or groups of atoms of interest that are present in the system initially, and solving appropriate tracking equations providing the concentration of tracked atoms at each possible location on each chemical species involved, in addition to the coupled ordinary differential equations describing the time evolution of species concentrations. The transfer equations for a given chemical kinetic mechanism are automatically generated using a constrained hierarchical pattern matching algorithm, or CHPMA, which relies on simple structural and energy-based arguments to characterize and quantify how, in each elementary reaction, atoms are re-organized as reactants are converted into products. In this work, the overall methodology will be presented, with a focus on the CHPMA algorithm. The capabilities of the tracking algorithm will then be illustrated by analyzing soot precursors formation during fuel-rich combustion, providing new, quantitative evidence of well-known behaviors observed in experiments, such as the link between the molecular structure of a fuel and its sooting propensity.

BIOGRAPHICAL SKETCH

Christopher James Frewin is a native of upstate New York, born in Schenectady and an inhabitant of Burnt Hills. Chris graduated from Clarkson University in 2014 with separate B.S. degrees in both physics and mechanical engineering. An avid hiker, skier, runner, and outdoorsman in general, he is an Adirondack 46er and has completed more than half of the 35 Catskill 3500ers. While his immediate position after graduation is programming for supply chain management, his interests remain in software engineering and sustainable technologies, and he is confident in pursuing a PhD some day. He would also like to write. Chris also founded Siren Apparel, a charity clothing company that donates all profits to police officers, firefighters, EMTs, and forest rangers in need. He hopes to one day improve on this platform, and is far from done in the area of founding companies.

This thesis is dedicated to my two grandfathers, Wesley Weaver and James Frewin, both of whom were mechanical engineers.

ACKNOWLEDGEMENTS

I would like to thank Perrine Pepiot and Charl  lie Laurent for their constant help and collaboration on this project. I have truly enjoyed every step of the project from those very first \mathcal{T} matrices all the way up to the completion of this thesis. I would also like to thank Krithika Narayanaswamy for her frequent assistance with my numerous questions on chemistry and her surrogate fuel mechanisms. I would like to thank all the members of the Pepiot Group for their various assistance during my time as a member of the group. Finally, I would like to thank my family and girlfriend for their love and support during my time here at Cornell.

TABLE OF CONTENTS

1	Overview of chapters and appendices	1
2	A novel atom tracking algorithm for the analysis of complex chemical kinetic networks	3
2.1	Introduction	3
2.2	Definitions and notations	8
2.3	Methodology	12
2.3.1	Atom transfer probabilities	12
2.3.2	Formulation of transfer equations	17
2.4	Constrained Hierarchical Pattern Matching	19
2.4.1	CHPMA Overview	20
2.4.2	CHPMA first stage: a preliminary Hierarchical Pattern Matching . .	21
2.4.3	CHPMA second stage: thermochemically constrained selection of plausible mappings	25
2.5	Applications and examples	33
2.5.1	Atom selectivity in combustion products	35
2.5.2	Atom tracking in multi-component mixtures	39
2.6	Conclusion	41
A	User guide: Running Kinsolve with tracking equations activated	43
A.1	Introduction	43
A.2	Steps for compiling and running pre-processing code	43
A.2.1	Pre-processing code functionality with reduced mechanisms	46
A.3	Steps for compiling and running simulation code	47
A.3.1	Post-processing simulation data	49
A.4	Molfiles and the molfile database	50
A.4.1	Adding new molfiles to the molfile database	50
A.4.2	Visualization and verification of molfiles	51
A.4.3	Full and "dummy" molfiles	54
A.5	Summary	54
B	preprocessor.pl code organization and description	57
B.1	Introduction	57
B.2	Printing trackingvars.f90	57
B.3	Summary	61
C	chpma.f90 code organization and description	64
C.1	Introduction	64
C.2	Overview of reconfiguration energy calculation	64
C.3	Summary	67

D	User guide: Python post-processing script and uses	68
D.1	Introduction	68
D.2	Overview of script environment and folders	68
D.2.1	List of scripts	69
D.2.2	Output file naming convention	71
D.3	Code descriptions and summaries	72
D.3.1	postprochelpers.py	72
D.3.2	cleanupheaderssingle.sh	73
D.3.3	flux.py	74
D.3.4	trackflux.py	74
D.3.5	fixedplots.py, oxiplots.py, pyroplots.py, surrplots.py	74
D.3.6	blend_carbon.py	75
D.4	Additional scripts	75
D.5	Summary	76

LIST OF TABLES

2.1	Summarizing relative variations of bond energy calculations, adapted from	31
2.2	Reconfiguration energy for the 24 mappings provided by the pattern-matching algorithm, corresponding to the example reaction shown in Fig. 2.13.	33
2.3	Parameters for the homogeneous, isochor reactor simulations used in Sections 2.5.1 to 2.5.2.	35
A.1	The complete list of chemistry and tracking output files created by the simulation side of the <code>kinsolve/pp</code> branch binary. Here, (<code>tracname</code>) will be the name specified under the <code>Tracking output file : trac</code> input file flag. .	50
A.2	Organization of scripts which create a <code>.tex</code> file from the molfile database. .	54
B.1	The six <code>preprocessor.pl</code> subroutines called before printing <code>trackingvars.f90</code> , their corresponding lines in <code>preprocessor.pl</code> , and description.	58
B.2	Fortran components of <code>trackingvars.f90</code> source code, printed from corresponding sections of <code>preprocessor.pl</code>	59
B.3	<code>preprocessor.pl</code> Perl subroutines which assist with printing <code>trackingvars.f90</code> Fortran subroutines. The lines of the Perl subroutines in <code>preprocessor.pl</code> are provided as well as a description of the Perl subroutine.	62
B.4	Descriptions of the arrays in the <code>get_nTrrp</code> Fortran subroutine.	63
B.5	Bond types and their corresponding number. The numbering method used enables modification of bond energies in <code>chpma.f90</code> if needed.	63
C.1	All subroutines in <code>chpma.90</code> , their corresponding lines, and description. . . .	65

LIST OF FIGURES

2.1	Molecule representations used in the tracking methodology. In the first stage of CHPMA, the skeleton representation of the molecule, keeping track of C and O only, is used, while additional attributes such as bond type or radical location are used in the second stage of the method.	10
2.2	Single and multiple atom labeling for MCH oxidation.	10
2.3	Molecule labeling in a multi-component (NC7 and MCH) mixture: all carbons from MCH will be tracked ($n_L = 1$).	11
2.4	Plausible atom mapping for the reaction in Eq. 2.13. \mathcal{M}_1 , or $\mathcal{M}_{A_1CH_3,A_1}^n$ describes how atoms in toluene are transferred to benzene, while \mathcal{M}_2 , or $\mathcal{M}_{A_1CH_3,CH_3}^n$ describes how they are transferred to the methyl radical. This mapping agrees with the expected reaction pathway, in which the $^1C - ^2C$ bond is the only one being broken. The n superscript is added to stress the fact that mappings are defined for a given reaction n	15
2.5	Additional atom mappings between A_1CH_3 , and A_1 and CH_3 for the reaction in Eq. 2.13.	15
2.6	Set of transfer matrices for the reaction of Eq. 2.13.	16
2.7	Two separate reactions illustrating the selection of WP and SP	23
2.8	Four of the $N_S = 8$ matches of WP in SP found by the VF2 sub-graph isomorphism algorithm, the 4 others being the same ones with a "flipped" WP-match in SP	24
2.9	New pattern reactions generated by replacing WP by one of the pattern WP*	25
2.10	New pattern reaction generated thanks to the removal of WP and a WP-match from SP	25
2.11	<i>Hierarchical Pattern Matching Algorithm</i> flowchart.	26
2.12	Two of the mappings output by the hierarchical pattern matching for the reaction $C_8H_{17} \rightarrow C_4H_8 + C_4H_9$. An index i_r (resp. j_p) refers to the carbon atom of the reactants (resp. products) located in the same row (resp. column). The gray cases show the correspondence between atoms i_r from the reactants and atoms j_p of the products. The wavy lines show the broken covalent bonds induced by the mapping. Mapping 1 induces one broken covalent bond 4_r-6_r . Mapping 2 induces three broken covalent bonds: two in the reactants: 2_r-4_r and 6_r-7_r , and one in the products: 6_p-7_p	29
2.13	Example reaction $x13C_7H_{11}6 \rightarrow N-C_4H_5 + C_3H_6$	30

2.14	Four of the mappings output by the hierarchical pattern matching for the reaction $\text{x13C}_7\text{H}_{11}6 \rightarrow \text{N-C}_4\text{H}_5 + \text{C}_3\text{H}_6$. An index i_r (resp. j_p) refers to the carbon atom of the reactants (resp. products) located in the same row (resp. column). The grey cases show the correspondence between atoms i_r from the reactants and atoms j_p of the products. The bond changes induced by the mapping are directly indicated on the molecules: wavy lines show broken covalent bonds, arrowed lines are bonds created through the mapping, dotted circles show bonds that are transformed, and bonds without any specific mark are the ones that are transferred unchanged.	32
2.15	Detailed energy calculations showing bonds created, broken, transformed, or transferred unchanged, for mappings 1, 22, 23, and 24.	33
2.16	Origin of atoms ($\alpha_{i,j}^{*k}$) at different locations in naphthalene, for cases C1, C2, and C3 (Table 2.3), showing much larger variability for toluene than for heptane or methyl-cyclohexane. All atoms of each fuel molecule have been labeled individually (note that non-labeled atoms in the fuels and A_2 can be deduced from symmetry of the molecules), and all results are taken at $t = 500\text{ms}$	37
2.17	Atom selectivity $\sigma_i^{\mathcal{F}}$ for the three fuel molecules considered in this study. A value of 0 indicates equal contribution of all atoms in the fuel to the species, while a value of 1 would indicate that only one atom in the fuel participates in the formation of the species.	39
2.18	Time evolution of some key species concentrations for case C_4 . Note the logarithmic scale used for time, in order to better show the various characteristic times involved.	40
2.19	Average fraction of C^* in species present at $t = 500\text{ ms}$ for Case C_4 , conditioned on the number of carbon in the species. Error bars indicate for each number of carbon atoms the minimum and maximum values observed. The dashed line indicates the average fraction of labeled atoms, that is, initially coming from toluene molecules, in the system.	41
A.1	Example of the Graphviz language in a <code>.dot</code> file for benzyl radical, A1XC7H7	53
A.2	Difference between full and "dummy" molfiles.	55
B.1	Example of four arrays in the source code which completely characterize a species' structure, in this case, the tert-butoxy species. Note this species has one radical and one oxygen atom. All of these specificities are captured within the arrays.	60
B.2	Graph visualization of the tert-butoxy species.	60

CHAPTER 1

OVERVIEW OF CHAPTERS AND APPENDICES

Chapter 2 consists of a manuscript intended for submission to the scientific journal *Combustion and Flame*. This work was written in collaboration between Charl  lie Laurent, a visiting student from   cole normale sup  rieure de Cachan in France, and Christopher Frewin, a M.S. student from Cornell University. Charl  lie Laurent created the first half of CHPMA, including the recursive pattern matching process. Charl  lie developed many of the equations used to analyze the data and describe the results, and wrote a large portion of the manuscript describing CHPMA in detail. Christopher Frewin created the second half of the algorithm, including the energy calculation of CHPMA, and generated many of the data, plots, and visualizations used in the manuscript. Christopher also created codes for graph generation, molecule visualization, and maintenance of the molecule structure database. Christopher also wrote all additional code to incorporate the atom tracking algorithm variables into a zero dimensional kinetic solver which previously solved only chemistry source terms.

Appendix A illustrates how to compile and run both parts of the in-house code, the pre-processing side in which CHPMA is used to generate mechanism-specific transfer matrices, and the simulation side which runs a simulation with tracking equations activated.

Appendix B summarizes components of a Perl script, `preprocessor.pl`, used to create a Fortran module of the chemical mechanism `mechanism.f90`, and an additional key Fortran module, `trackingvars.f90`, which includes all the relevant variables needed by the kinetic solver to solve the atom transfer equations. This appendix focuses on the components of `preprocessor.pl` which print `trackingvars.f90`.

Appendix C summarizes components of a Fortran module `chmpa.f90`, the module con-

taining all parts of CHPMA and responsible for generating the transfer matrices on the pre-processing side of the code.

Appendix D describes select post-processing scripts written in Python which assist in analyzing data generated by the atom tracking algorithm.

CHAPTER 2

A NOVEL ATOM TRACKING ALGORITHM FOR THE ANALYSIS OF COMPLEX CHEMICAL KINETIC NETWORKS

2.1 Introduction

Today, worldwide energy needs for transportation are overwhelmingly met with petroleum-based fuels [1], and one aspect of growing concern is the emission of pollutants, such as soot particulates, coming with the massive use of fossil fuels. Bio-derived fuels are expected to progressively displace conventional fuels in an effort to reduce our dependence on crude oil and mitigate environmental impact [2]. Due to the wide variety of sources and production routes, bio-fuels are inherently diverse, with chemical functionalities potentially distinct from their fossil-derived counterparts [3], resulting in an explosion in the number and variety of fuels and fuels blends available. This, in turn, provides new opportunities to optimize fuel blends and design new high-efficiency, low-emissions engines [4, 5].

Transportation fuels, from fossil sources in particular, but also from renewable sources, typically consist of a large number of fuel constituents [6, 7], each one potentially playing a specific role during the combustion process. The observables, *i.e.* global measures of engine and combustion performance, therefore result from complex, multi-level physical and chemical interactions between those constituents. Numerous studies, both experimental and numerical, provide strong supporting evidence, with fuel molecular structures and fuel component interactions both impacting combustion in non-trivial ways. For example, the sooting propensity of hydrocarbons fuels, which can be measured through Threshold [8] or Yield [9] Sooting Indices, has been shown to strongly correlate with the structural groups present in the fuel initially, such as rings, primary, secondary or tertiary carbons, type

of carbon-carbon bonds etc. [10, 11, 12, 13, 14]. Adding oxygenated compounds, even in small amounts, to hydrocarbons often leads to a reduction in particulate emissions, but to varying extends depending on the type and location of the oxygen atoms in the additives [11, 15, 16]. In the same vein, recent studies on surrogate fuels, which are simple mixtures of a small number of well-characterized components designed to reproduced experimentally or numerically the combustion dynamics of real, complex fuels, emphasize the importance of matching the overall structural makeup of a fuel, and more specifically, the structural composition of the initial hydrocarbon fragments pool resulting from the initial break-up of the fuel, in the surrogate fuel formulation [17].

Common techniques used to investigate the link between fuel structure and combustion characteristics often rely on group contribution analysis [13, 18], which relates output observations, such as sooting propensity, to input parameters, such as fuel structural groups, using statistical regression techniques. While convenient to establish correlations, such methods cannot be used to characterize the role of the individual chemical processes contributing to the observations. The relatively recent and dramatic advances in chemical kinetics modeling now allow for a much more fundamental analysis of the combustion dynamics. In particular, sensitivity analysis [19, 20] and reaction flux analysis [21, 22] have been used extensively to identify the major reaction pathways and sensitive steps in a chemical kinetic model. While very powerful analysis tools, such techniques provide quantitative information focused on individual reactions only, for example, how sensitive a global combustion quantity is to a given reaction’s parameters, or how much a reaction contributes to the production or consumption of a species. Some additional manipulation is required to get a broader picture, for instance, identifying the main pathways leading from the fuel all the way to the products themselves, and this is generally accompanied by a switch from quantitative measures to a more qualitative analysis.

A very attractive way to elucidate the mechanisms behind complex chemical reaction processes is to use atom tracking. Widely used experimentally, those tracing techniques consist in selectively replacing some atoms in the initial reactants by isotopes, *e.g.* deuterium (^2H), carbon-13 (^{13}C), or carbon-14 (^{14}C), and searching for and quantifying the amount of those isotopes in products, thereby allowing to deduce what kinetic pathways are activated during reaction. Homan *et al.* [23], for instance, labeled carbon atoms in molecules typical of those found in modern fuels, and observed that while all carbon atoms contributed to soot regardless of the fuel, some contributed distinctively more than others. For example, primary carbons showed close to no preferential distribution in the products, while phenyl or naphthyl carbons had a much higher probability end up in soot. Buchholz *et al.* [24] performed combustion experiments with ^{14}C -labeled diesel oxygenate dibutyl-maleate (DBM). DBM is a symmetric molecule that combines carbon-carbon single bonds, and carbon-oxygen single and double-bonds. By selectively labeling carbons in the molecules, they were able to determine how oxygenated functional groups within the fuel molecule impacted CO and CO_2 production, showing for example that in contrast to the C–O single bonds, which are not retained in the products, the C=O double-bonds in DBM do not break during combustion. A related study was conducted by Eveleigh *et al.* [25], in which seven oxygenated molecules with size ranging from 2 to 7 carbons were labeled using ^{13}C , and subjected to pyrolysis conditions at temperatures ranging from 1200 to 1450 °C. A similar conclusion was drawn, stating that carbon atoms contribute to particulate matter to various extent, depending on the nature of the neighboring moiety. Oxygen-containing functional groups have a significant influence on the formation of particulate matter through a reduction in the contribution of adjacent carbon atoms: for example, the methyl carbon in ethanol contributed two-third by weight of the resulting soot, compared to barely a fourth for the carbons linked to the hydroxyl group. As a comparison, all carbon atoms in toluene were found to contribute nearly equally to soot regardless of their nature: methyl carbon, tertiary carbon, or phenyl

(ring) carbon.

Atom tracking techniques are shown to provide very valuable experimental information, and the ability to label atoms in simulations therefore appears as a highly desirable feature for kinetic mechanism development, validation, and analysis. Such a feature has previously been developed by Bunev *et al.* [26, 27] to investigate specific features of hydrogen-methane chemistry and hydrogen/nitric oxides interaction during ignition. Their approach, which they defined as a numerical tracers approach, consists of adding any molecule containing a labeled atom at a given location to the chemical kinetic mechanism of interest, and treating them as individual species. This requires duplicating all reactions involving species that contain at least one labeled atoms, and adjusting the rates of these reactions accordingly to recover the appropriate overall rate of the original reaction. Rate adjustments were done using symmetry numbers or statistical factors commonly found in the transition state literature. By treating molecules with labeled atoms as individual species, evolving during combustion based on their own set of reactions, the labeled atoms can effectively be tracked during combustion, and their interactions with the remaining species identified. Bunev *et al.* used their tracking technique to determine, for instance, whether species such as H_2 , O_2 , CO_2 , and CH_4 , added as diluent in the initial system, were actively participating in combustion, acted as inert diluents, or behaved as a combination of both. While functional when considering small fuel molecules and mechanisms (here, methane), the approach required extensive and non-systematic changes to the chemical model used in the simulation, rendering any extension to molecules and mechanisms relevant for transportation fuels very difficult, if not impossible.

In this context, the objective of this paper is to present a novel automatic and systematic algorithm to numerically track labeled atoms or group of atoms during combustion, thereby creating new capabilities and perspectives to analyze and validate combustion kinetic models

for complex fuels. The framework and guiding principles behind the proposed method are simple. We restrict ourselves to time-evolving, homogeneous (0D) reactors for now, thereby focusing on the kinetic network itself in the absence of transport. User input consists of selecting which atom to label in the initial system. Output contains the evolution in time of the concentration of labeled atoms at any location in any given species involved in the chemical mechanism used for the simulation. Emphasis is placed on ensuring that the algorithm can be directly applied to any chemical kinetic mechanism in conventional format (*e.g.* Chemkin), the only additional information required being the actual structure of each molecular species appearing in the mechanism. Tracking equations are formulated automatically and systematically from the set of reactions contained in the chemical mechanism, using simple structural and energy-based arguments to characterize and quantify how, in each reaction, atoms are re-organized as reactants get converted into products. Those tracking equations are solved in parallel to the coupled ordinary differential equations (ODE) describing the time evolution of species concentrations, requiring minimum modification to the stiff ODE solver typically used for combustion kinetic simulations. In addition, built-in flexibility allows for expert users to refine some the preset atom transfer rules for elementary reactions with unusually complex transition states.

The resulting algorithm has a broad range of applications. In the limited scope of this paper, the focus will be placed on the description of the method, followed by a number of examples aiming at illustrating some of the key analysis capabilities it enables. While those examples do provide some quantitative numerical evidence of the experimental observations mentioned above, more in-depth analysis of specific systems is left to future work.

The remainder of the paper is organized as follows. First, some important definitions and notations are provided (Section 2.2), followed by an overview of the methodology and governing equations (Section 2.3). In Section 2.4, the emphasis is placed on the description

of the core algorithm, called the *Constrained Hierarchical Pattern Matching Algorithm*, or CHPMA, which automatically determines, for any given reaction, how atoms are transferred from reactants to products. Finally, Section 2.5 provides examples of how to post-process the raw data generated by the tracking algorithm, and apply them to soot precursors formation in rich hydrocarbon oxidation configurations.

2.2 Definitions and notations

We consider a reacting mixture of ideal gases, consisting of n^S chemical species. For simplicity of exposition, the mixture is assumed to evolve at a fixed volume, or constant density ρ , so that, given the pressure p , the full thermochemical state, or composition, of the mixture is completely characterized by the n^S -vector of species concentrations \mathbf{C} and the mixture temperature T . The mixture is evolving according to a chemical kinetic mechanism containing n^R reactions, the conservation equation governing the time evolution of a species concentration C_i being written:

$$\frac{dC_i}{dt} = \sum_{n=1}^{n^R} \nu_{i;n} \omega_n, \quad (2.1)$$

where $\nu_{i;n}$ is the stoichiometric coefficient of species S_i in reaction n (negative for a reactant, and positive for a product), and ω_n is the rate of reaction n , expressed as

$$\omega_n = k_n \prod_{j=1}^{n^S} \delta_{j;n}^R C_j^{\nu_{j;n}}. \quad (2.2)$$

In the previous equation, k_n is the rate constant of reaction n in Arrhenius form, and δ^R indicates whether or not a species appears as a reactant in a given reaction:

$$\delta_{j;n}^R = \begin{cases} 1, & \text{if species } j \text{ is } \mathbf{reactant} \text{ in reaction } n \\ 0 & \text{otherwise.} \end{cases} \quad (2.3)$$

For convenience later on, we also define the similar product indicator $\delta_{j;n}^P$:

$$\delta_{j;n}^P = \begin{cases} 1, & \text{if species } j \text{ is } \mathbf{product} \text{ in reaction } n \\ 0 & \text{otherwise.} \end{cases} \quad (2.4)$$

For simplicity of exposition, and without loss of generality, we restrict ourself to hydrocarbon combustion. In this case, any species S_i consists of carbon, C, oxygen, O, and hydrogen, H, atoms only, and its formula can be written:

$$S_i : C_{n_i^C} O_{n_i^O} H_{n_i^H} \quad (2.5)$$

We call a *skeleton atom* any atom able to form more than one covalent bond, in our case C and O, and denote by n_i^A the total number of skeleton atoms in species S_i :

$$n_i^A = n_i^C + n_i^O \quad (2.6)$$

The location of each skeleton atom in species S_i is indexed from 1 to n_i^A , as illustrated in Fig. 2.1. The concentration of species S_i in the reactor is denoted by C_i . We denote by $\kappa_{i;j}$ the concentration of a skeleton atom, C or O, at a given location j in species S_i . This concentration is naturally equal to that of the species itself for any i and j :

$$\kappa_{i;j} = C_i. \quad (2.7)$$

We define $\underline{\kappa}_i$ to be the n_i^A -vector containing these atom concentrations at all possible location in species S_i . The total concentration of skeleton atoms in species S_i is easily obtained as the L_1 -norm of $\underline{\kappa}_i$, and denoted by K_i :

$$K_i = |\underline{\kappa}_i|_1 = n_i^A C_i, \quad (2.8)$$

where the Einstein summing convention is not implied.

The above variables are extended to account for labels affixed to atoms in order to distinguish them from one another and track their paths in time throughout the kinetic

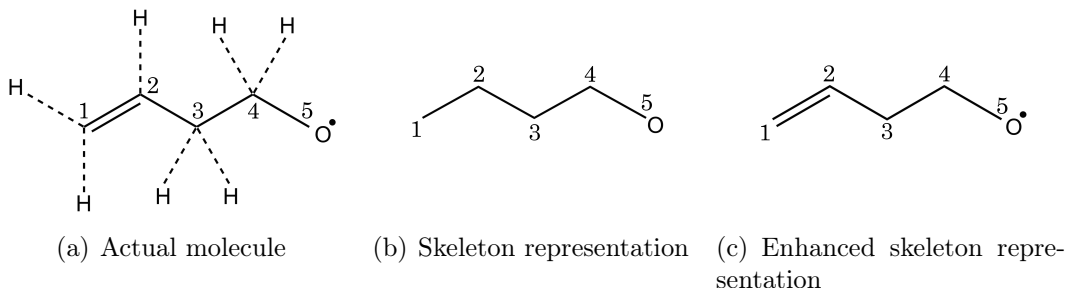


Figure 2.1: Molecule representations used in the tracking methodology. In the first stage of CHPMA, the skeleton representation of the molecule, keeping track of C and O only, is used, while additional attributes such as bond type or radical location are used in the second stage of the method.

network. Atom labeling is performed at discrete times, typically once at the very beginning of a simulation. During a labeling event, all atoms at user-specified locations are assigned a specific label, with the total number of labels being n^L . For clarity, labels are chosen to be contiguous integers, referred to as $*k$, with $*k = 1$ to n^L . Skeleton atoms at non-labeled locations are assigned a 0^* default label. Examples of labeling strategies are shown below, from tracking a single atom (Fig. 2.2(a)) or multiple atoms (Fig. 2.2(b)) in the fuel, to tracking all atoms from a specific molecule in multicomponent mixtures (Fig. 2.3). *n*-heptane and methyl-cyclohexane.

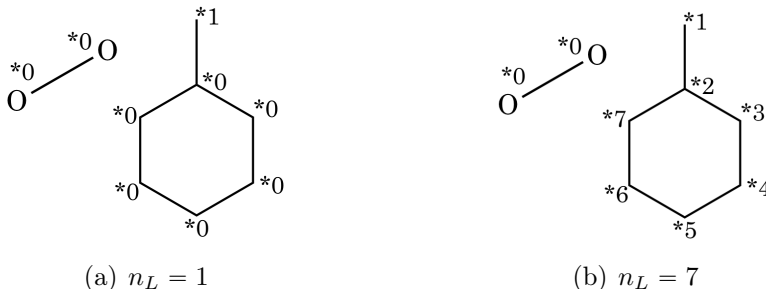


Figure 2.2: Single and multiple atom labeling for MCH oxidation.

Label-specific atom concentrations can now be defined. Following the notation introduced above, we denote by $\kappa_{i,j}^{*k}$ the concentration of $*k$ -labeled atom at location j in species S_i ,

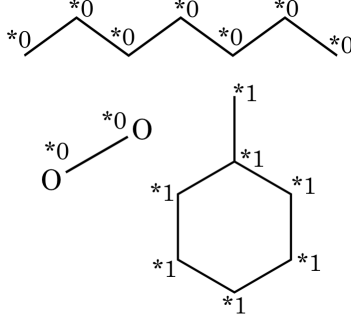


Figure 2.3: Molecule labeling in a multi-component (NC7 and MCH) mixture: all carbons from MCH will be tracked ($n_L = 1$).

and $\underline{\kappa}_i^{*k}$ the vector of size n_i^A containing those concentrations at all locations in species S_i .

The total concentration of k^* -labeled atoms in species S_i is defined by:

$$K_i^{*k} = |\underline{\kappa}_i^{*k}|_1 = \sum_{j=1}^{n_i^A} \kappa_{i;j}^{*k}. \quad (2.9)$$

A number of conservation properties are easily derived from the definition of label-specific concentrations. For example:

- The sum of all labeled and *0-labeled atom concentrations at any location j in any species i is equal to the concentration of species S_i :

$$\sum_{*k=0}^{n_L} \kappa_{i;j}^{*k} = C_i \text{ for all } i, j \quad (2.10)$$

- For any label $*k$, the total concentration of $*k$ -labeled atoms in the system, that is, summed over all possible skeleton atom locations, is constant and equal to its initial value K_0^{*k} :

$$\sum_{i=1}^{n^S} \sum_{j=1}^{n_i^A} \kappa_{i;j}^{*k} = \sum_{i=1}^{n^S} K_i^{*k} = K_0^{*k} \text{ for all } *k \quad (2.11)$$

The objective of the tracking methodology can now be redefined more clearly: given an initial mixture of hydrocarbon species and oxygen, part of which has been labeled using n^L

distinct labels, compute $\kappa_{i,j}^{*k}(t)$, that is, the concentration of $*k$ -labeled atom at each skeleton atom location on each species involved in the chemical system. The corresponding equations and required associated quantities are detailed next.

2.3 Methodology

This section describes two key aspects of the tracking approach: *i*) the exchange of atoms between reactants and products at the reaction level, and *ii*), the derivation of the appropriate governing equations for $\underline{\kappa}_i^{*k}(t)$.

2.3.1 Atom transfer probabilities

During a chemical reaction, reactant molecules collide and transform into products, the rate at which collision and conversion occur being quantified by the law of mass action and the Arrhenius parameters associated with the reaction. As the reaction proceeds, the reactant molecules are affected by structural modifications: covalent bonds are created or broken, hydrogen atoms are transferred, carbon atoms are reorganized, and the geometry of the molecules change to that of the products. Determining the fate of a given atom in reactants, and its final location in the products, usually requires to precisely know the structure of the transition states involved in the reaction. Those structures may become quite complex, and can be determined for example from *ab initio* quantum chemistry calculations [28]. While essential to determine reaction constants, no information about the transition states are retained in commonly used chemical kinetic mechanisms, which consist of a list of reactions only listing reactants and products, along with Arrhenius parameters.

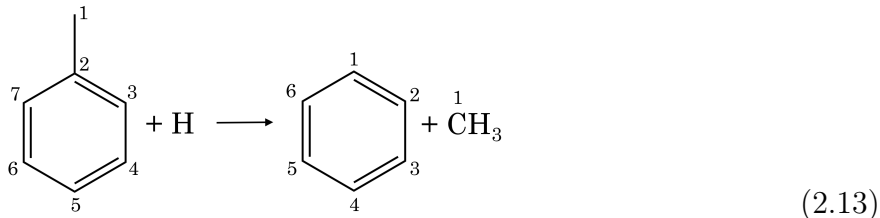
In the context of this work, we make *two important simplifying assumptions*:

- The chemical kinetic mechanisms on which the method is applied are formed of *elementary* reactions, that is, reactions describing actual molecular collisions, and for which the transition states are assumed to be relatively simple, close in structure to both reactants and products.
- Under these conditions, the transition state actually occurring during the reaction is assumed to be the one that minimizes the number of re-arrangements (such as the number of covalent bonds broken or created, or the number of hydrogen atoms transferred during the reaction).

While significant, those assumptions allow us to automatize the process of determining the destination of every skeleton atoms, carbon or oxygen, from reactants to products. This process is described in much details in Section 2.4, along with a discussion of how to handle reactions that do not satisfy those assumptions. At this stage, however, we want to focus on a simple example, the reaction of toluene (A_1CH_3) with hydrogen, yielding benzene (A_1) and a methyl radical:



or, using the molecules directly, with skeleton atom locations numbered from 1 to the total number of skeleton atoms n^A in the species:



and use that reaction to define the concept of atom transfer probability, a key quantity in the derivation of the tracking equations.

Consider one specific molecule of toluene colliding with one hydrogen atom, and converting to one molecule of benzene and one methyl radical. Basic atom conservation ensures that each atom in toluene is transferred to one specific location in the products. We call mapping between a reactant A and a product B , denoted by $\mathcal{M}_{A,B}$ (or, as needed by the context, $\mathcal{M}_{i,j}$, where i and j are indices referring to species A and B), any one possible re-arrangement of atoms between these reactant and product in the reaction. A mapping is most conveniently represented as a matrix, whose rows correspond to the various skeleton atom locations in the reactant, and columns, to the various skeleton locations in the product. An entry (i, j) in this matrix is unity if the atom at location i in the reactant is transferred to location j in the product, zero otherwise. The number of mapping matrices for each reaction is equal to the product between the number of reactants and the number of products in that reaction.

A quick inspection of Eq. 2.13 leads to the obvious conclusion that the only bond broken during the reaction is the one between the methyl branch and the aromatic ring ($^1\text{C} - ^2\text{C}$) in toluene. Therefore, after reaction, the methyl carbon in toluene is necessarily transferred to the free methyl radical molecule (probability of 1), and the carbons forming the aromatic cycle in toluene necessarily form the aromatic ring in benzene. Accordingly, one plausible set of mapping matrices for reaction 2.13 is shown in Fig. 2.4.

In addition, due to the symmetry properties of the aromatic cycle, mappings that correspond to a rotated cycle is still appropriate, leading to additional plausible mappings (Fig. 2.5(a)). However, any mapping that ends up breaking up more bonds than just $^1\text{C} - ^2\text{C}$ in toluene, as shown in Fig. 2.5(b) for instance, are not expected to occur.

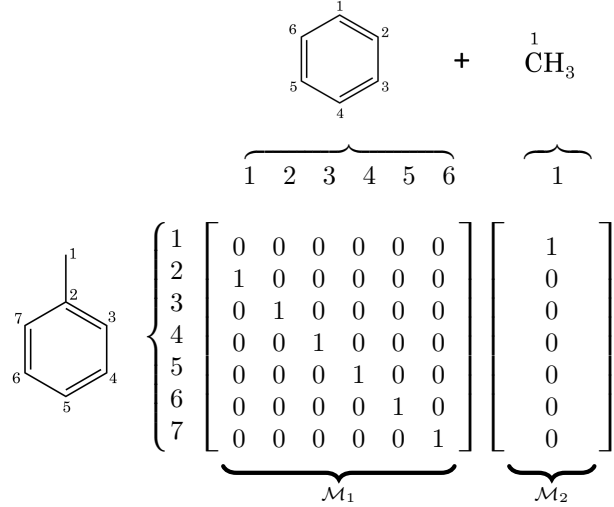


Figure 2.4: Plausible atom mapping for the reaction in Eq. 2.13. \mathcal{M}_1 , or $\mathcal{M}_{A_1CH_3, A_1}^n$ describes how atoms in toluene are transferred to benzene, while \mathcal{M}_2 , or $\mathcal{M}_{A_1CH_3, CH_3}^n$ describes how they are transferred to the methyl radical. This mapping agrees with the expected reaction pathway, in which the $^1C-^2C$ bond is the only one being broken. The n superscript is added to stress the fact that mappings are defined for a given reaction n .

$$\mathcal{M}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(a) Additional plausible mappings.

$$\mathcal{M}_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

(b) Unrealistic mappings.

Figure 2.5: Additional atom mappings between A_1CH_3 , and A_1 and CH_3 for the reaction in Eq. 2.13.

We assume further that *i*) plausible mappings can reliably be identified, and *ii*) plausible mappings occur with equal probability, assumptions whose validity is discussed in Section 2.4.

We can then determine how, on average, atoms are transferred through a reaction n between a reactant S_i in this reaction to a product S_j in this same reaction. For this purpose, we introduce a *transfer probability matrix* between those two species, $\underline{\underline{\mathcal{T}}}_{i,j}^n$, whose elements (k,p) contain the probability of transferring a skeleton atom at location k in reactant S_i to location p in product S_j , computed by averaging all plausible mappings, numbered from 1 to $n^{\mathcal{M}_p}$ between those two species:

$$\mathcal{T}_{i,j}^n(k,p) = \frac{\sum_{m=1}^{n^{\mathcal{M}_p}} \mathcal{M}_{i,j}^{n,(m)}(k,p)}{n^{\mathcal{M}_p}} \quad (2.14)$$

Coming back to Eq. 2.13, it is easily shown that there are six plausible mappings, differing from one another through simple rotation of the aromatic cycle. The set of transfer matrices for that reaction are shown in Fig. 2.6.

$$\mathcal{T}_{A_1CH_3,A_1}^n = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

$$\mathcal{T}_{A_1CH_3,CH_3}^n = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.6: Set of transfer matrices for the reaction of Eq. 2.13.

As is described next, the transfer matrices $\mathcal{T}_{A,B}^n$ are the key entities that allow us to track atoms throughout a kinetic network. How to obtain those matrices for every reaction is detailed in Section 2.4.

2.3.2 Formulation of transfer equations

The conservation equations describing the time evolution of the labeled atom concentrations are similar to those for the species concentrations themselves, but modified to account for the transfer probabilities defined above. Accordingly, the time derivative of $\underline{\kappa}_i^{*k}$, the vector containing the concentration of $*k$ -labeled atoms at each location in species S_i , can be written as the sum of production (positive, \mathcal{P}) and consumption (negative, \mathcal{C}) terms:

$$\frac{d\underline{\kappa}_i^{*k}}{dt} = \left. \frac{d\underline{\kappa}_i^{*k}}{dt} \right|_{\mathcal{P}} + \left. \frac{d\underline{\kappa}_i^{*k}}{dt} \right|_{\mathcal{C}}. \quad (2.15)$$

Consumption term

The rate of consumption of $*k$ -labeled atoms at location j in species S_i , itself appearing as reactant in reaction n , is equal to the full reaction rate ω_n , multiplied by the fraction of $*k$ -labeled atoms at this location, $\alpha_{i;j}^{*k}$:

$$\alpha_{i;j}^{*k} = \frac{\kappa_{i;j}^{*k}}{\kappa_{i;j}} = \frac{\kappa_{i;j}^{*k}}{C_i}, \quad (2.16)$$

or $\underline{\alpha}_i^{*k}$ in vector form (collecting values for all locations in species S_i). The consumption term for $\underline{\kappa}_i^{*k}$ can therefore be written as:

$$\left. \frac{d\underline{\kappa}_i^{*k}}{dt} \right|_{\mathcal{C}} = \sum_{n=1}^{n_R} \delta_{i;n}^R \nu_{i;n} \omega_n \underline{\alpha}_i^{*k}, \quad (2.17)$$

Equation 2.17 is inconvenient to use directly, since $\alpha_{i;j}^{*k}$ becomes ill-defined when C_i , the concentration of species S_i , is small or zero. Instead, we note that all non-zero contributions in Eq. 2.17 come from reactions involving S_i as a reactant, whose reaction rates are proportional to C_i raised to a some power. Hence, the division by C_i can readily be absorbed into

the reaction rate expression to yield a well-defined quantity:

$$\omega_n \alpha_{i;j}^{*k} = \frac{\omega_n}{C_i} \kappa_{i;j}^{*k} \quad (2.18)$$

$$= k_n \left(\prod_{\substack{\text{reactants } S_j \\ S_j \neq S_i}} C_j^{\nu_{j;n}} \right) C_i^{\nu_{i;n}-1} \kappa_{i;j}^{*k} \quad (2.19)$$

Equation 2.17 yields negative contributions because the only non-zero terms are coming from reactions n in which the stoichiometric coefficient $\nu_{i;n}$ is negative.

Production term.

The rate of production of labeled atoms is somewhat more complicated to express, since it involves the atom transfer probability matrices \mathcal{T} specific to each reaction, as defined above.

In vector form, following the same notations as before, this term can be written as:

$$\left. \frac{d\mathbf{\kappa}_i^{*k}}{dt} \right|_{\mathcal{P}} = \sum_{n=1}^{n^R} \delta_{i;n}^P \left(\sum_{m=1}^{n^S} \delta_{m;n}^R \nu_{i;n} \omega_n \mathcal{T}_{m,i}^n \mathbf{\alpha}_m^{*k} \right) \quad (2.20)$$

The j th term of the vector equation 2.20 corresponds to the production rate of $*k$ -labeled atoms at location j in species S_i , $\kappa_{i;j}^{*k}$. It is obtained by summing over each reaction n in which S_i is a *product* (as indicated by the product indicator $\delta_{i;n}^P$), the contribution from all possible sources in that reaction, that is, from all locations in the *reactants* m (reactant indicator $\delta_{m;n}^R$), proportionally to the fraction of $*k$ -labeled atoms currently there, multiplied by the transfer probability between those locations and the j th location in S_i . Again, this is a well-conditioned equation even for zero or low concentration species, because S_m is necessarily a reactant in reaction n , so that the division by C_m in $\mathbf{\alpha}_m$ can be absorbed conveniently in the reaction rate, ω_n , as in Eq. 2.19.

In addition to the conservation equations for species concentrations, tracking atoms throughout the combustion process adds a number of equations directly proportional to the number of labels being used and to the number of skeleton atom locations in the system. The total number of variables can be written:

$$n_{tot} = n^S + n^L \times \sum_{i=1}^{n^S} n_i^A. \quad (2.21)$$

The tracking atom equations being linear, a simulation tracking n_L labels can be replaced by n_L simulations tracking one label, for which the total number of variables is reduced to $n^S + \sum_{i=1}^{n^S} n_i^A$. Additional information about the computational cost of the method can be found in Section 2.5.

2.4 Constrained Hierarchical Pattern Matching

As can be seen from above, the key ingredient of the tracking methodology is the definition of the transfer probabilities (or equivalently transfer matrices) for each reaction in the kinetic model. With 10^3 to 10^4 reactions included in most relevant mechanisms, manual inspection of the structural changes resulting from each and every reaction is impossible, and a fully automatic approach is essential. For this purpose, we designed a robust *Constrained Hierarchical Pattern Matching Algorithm*, or CHPMA, able to efficiently generate the set of transfer matrices $\{\mathcal{T}\}^n$ for any elementary reaction n in the mechanism, without the need for external expertise. Details are provided below.

2.4.1 CHPMA Overview

There are three main ingredients to CHPMA: a pre-processing step, and two complementary structure analysis algorithms:

Pre-processing. The pre-processing step provides a systematic handling of the structure of each species in the mechanism. A graph representation is automatically generated for each molecule of the mechanism, with vertices being the skeleton atoms, and edges corresponding to covalent bonds. Whenever possible, this graph representation is obtained by using the Chemical Abstracts Service (CAS) Registry Number [29] that may be provided with the kinetic mechanism. Species CAS numbers are used to look up the corresponding molfile, a widely used chemical structure format file originally developed at Molecular Design Limited [30]. In the absence of a CAS number, structures are postulated from context and any supplementary material provided with the mechanism, and a proper molfile is built manually. The molfiles are then converted to a suitable graph representation containing a list of vertices (skeleton atoms) and adjacency list (representing the bonds between those atoms), which is stored for use in the subsequent steps.

First stage. CHPMA operates in two stages, applied to each reaction successively. The first stage effectively performs a preliminary *hierarchical pattern matching* of the reactants and products graph representations to generate an initial set of mappings. It focuses on the bare skeleton atom backbone structures of the species involved in the reaction, and molecules graph representations are therefore simplified to non-oriented vertex-labeled graphs (the atom type is taken into account, unlike the covalent bond type), which is called a *pattern* in the following. The algorithm consists of finding all possibilities to fit reactants patterns into products, while breaking a minimum number of bonds. It can be seen as a jigsaw puzzle, starting with the largest patterns and finishing with the smallest ones. This hierarchical

pattern matching is mostly based on graph operators and is decoupled from any chemical consideration, and returns all mappings for the reaction involving a minimum number of bonds broken between skeleton atoms, regardless of the type of bonds.

Second stage. The purpose of the second stage is to exclude unrealistic mappings from the set of potential solutions provided by the initial pattern matching. Indeed, even if this first step has been designed in order to naturally ignore the vast majority of chemically unrealistic matchings, not all mappings returned are plausible. Simple thermochemical rules, based on covalent bond energies, are applied in order to extract a final list of plausible mappings. At the end of this stage, only plausible mappings remain, and transfer probabilities can be computed.

The complementary combination of these two successive algorithms, with the first one taking care of the perturbations occurring on the skeletal structure of the molecules during the reaction, and the second one dealing with transformations affecting the covalent bond types, allows to generate a list of plausible mappings in agreement with the hypotheses on the transition states stated in Section 2.3. A detailed description and examples for both algorithms mentioned above is provided next.

2.4.2 CHPMA first stage: a preliminary Hierarchical Pattern Matching

As mentioned above, the overall purpose of the CHPMA is to match reactants molecular structures with products molecular structures, in order to determine, for every carbon or oxygen atoms in the reactants, all the possible destinations in the products. However, as the resulting solutions (or *mappings*) have to be consistent with the expected transition state

occurring during the reaction, this matching cannot be carried out blindly with random atom arrangements. Instead, the matching process has to be selective. Thus, this first stage of the CHPMA is intended to perform a preliminary hierarchical matching in order to discard all solutions deemed overly unrealistic, only keeping those most chemically credible mappings.

It is intuitively easy to realize that the most chemically unlikely mappings are roughly the ones involving substantial transformations of the molecules carbon-oxygen skeletons: indeed if a mapping implies that a large number of atoms have to split up and then recombine, this rearrangement is extremely improbable to happen. For this hierarchical matching, the molecules graph representations are therefore reduced to **non-oriented vertex-labeled graphs**. This representation only takes into account the molecules bare backbone structure (hydrogen atoms, radicals, and covalent bond types are not represented), but the vertex-labeled property allows us to distinguish carbon atoms and oxygen atoms. This simplified molecule graph representation is called a *pattern*. We note that a pattern can not only be an entire molecule, but also any sub-structure from a molecule: indeed, as it will be shown later, the process brings some patterns to break down into several smaller entities, which will also be called patterns. Similarly, a set of patterns is still called a pattern. In this context, the elementary chemical reaction considered is then called a *pattern reaction*. As depicted in the Figs. 2.7- 2.10, a single pattern reaction can be split into several smaller pattern reactions during the algorithm processing. The algorithm proposed here is based on *pattern matching*: reactants side patterns are matched with products side patterns, and *vice versa*, like in a jigsaw puzzle. However, as stated before, this pattern matching cannot be performed arbitrarily or blindly, that is why it is referred as *hierarchical*, since we perform some processes (see steps **1** and **5'** in the following description) intended to minimize the *pattern reorganization level* (which is an intuitive translation of the minimum reaction chemical potential energy in term of graphs). We give here a detailed functional

and structural analysis (Fig. 2.11) of the proposed *Hierarchical Pattern Matching Algorithm*:

Step 1: The selection of the pattern we want to match (**WP** - **Wanted Pattern**) is primordial to respect the *minimal pattern reorganization* criterion. This criterion imposes to minimize the number of edges, representing covalent bonds, to be broken during the matching process. The selection of **WP** mimics the natural way to solve a jigsaw puzzle: if we have large and small pieces in the puzzle, the easiest way to proceed is obviously to first of all place the large ones, and then finish with the smallest ones. Thus, if there are only carbon atoms involved in the pattern reaction, **WP** is therefore defined by first taking the largest patterns from reactants side and products side, and then the smallest one of these two patterns. For reactions involving oxygen, the idea is to prioritize patterns containing oxygen to match them first, so that the pattern reaction is quickly reduced to the case where only carbon atoms are present. For that purpose, we initially ignore patterns without oxygen, and **WP** is defined as the smallest pattern in the entire pattern reaction. If **WP** is on the reactants side, then the set of patterns that are searched (**SP** - **Searched Pattern**) to find matches of **WP** consists of all the patterns from the products side, and *vice versa*. The manner to select the pattern to match and to search denotes the hierarchical character of the algorithm. This is illustrated in Fig. 2.7.

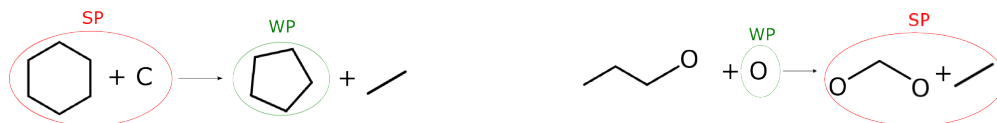


Figure 2.7: Two separate reactions illustrating the selection of **WP** and **SP**.

Step 2: Once the patterns **WP** and **SP** are selected, the aim is to find all the manners to fit **WP** within **SP**: all the N_S possible matches of **WP** in **SP** are found thanks to a

sub-graph isomorphism algorithm (VF2-algorithm [31], from the Boost Graph Library [32]). This algorithm automatically takes into account the multiple possibilities to fit **WP** in **SP** due to molecular symmetries. Figure 2.8 shows an example of these multiple matches.

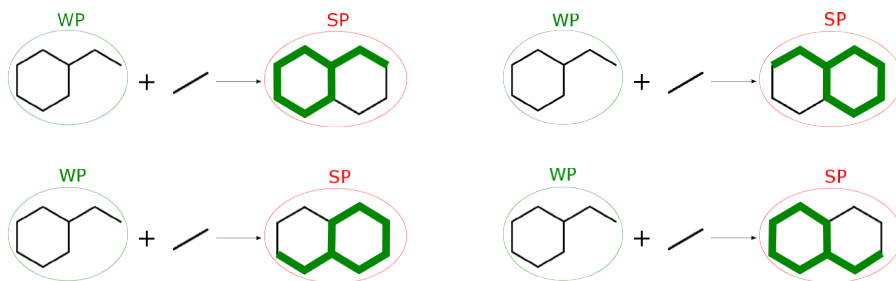


Figure 2.8: Four of the $N_S = 8$ matches of **WP** in **SP** found by the VF2 sub-graph isomorphism algorithm, the 4 others being the same ones with a “flipped” **WP-match** in **SP**.

Step 3: The way **WP** is selected (**Step 1**) has been designed so that it is possible to find at least one match of **WP** in **SP** for the vast majority of pattern reactions. However, a few cases where the molecular geometry is totally modified require to break one bond of **WP**, in order to make it easier to match.

Step 4: The current pattern reaction is replaced by a set of updated and smaller (in term of number of edges) pattern reactions, where **WP** is replaced by one of the patterns **WP***, generated by breaking arbitrarily one bond of **WP**, as illustrated in Fig. 2.9.

Step 5 and 5’: Once **WP** has been matched, the current pattern reaction is updated by removing **WP**. Then, for each possible match of **WP** within **SP**, a new smaller (in term of number of vertices and edges) pattern reaction is generated by removing **WP** and the associated **WP-match** from **SP**, as shown in Fig. 2.10. The mapping of the matching of **WP** in **SP**, *i.e.* the link from each atom of **WP** to its destination in **SP**, is stored in

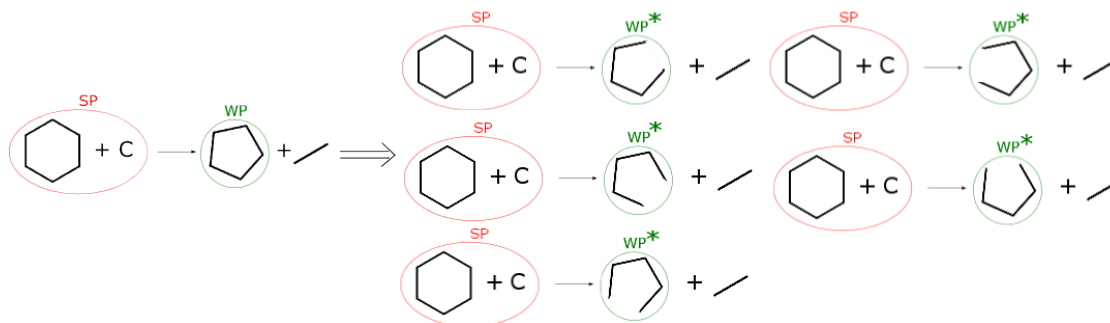


Figure 2.9: New pattern reactions generated by replacing **WP** by one of the pattern **WP***.

a permutation matrix. Finally, if there is no pattern to match left in the current pattern reaction, then it is removed from the list of reactions to be processed. The permutation matrix is updated on-the-fly throughout the matching process, so that it contains at the end a mapping of a potential solution to the problem.

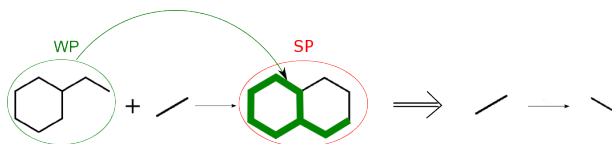


Figure 2.10: New pattern reaction generated thanks to the removal of **WP** and a **WP-match** from **SP**.

These steps and the interactions linking them within the CHPMA general framework are summarized on the algorithm structural diagram in Fig. 2.11.

2.4.3 CHPMA second stage: thermochemically constrained selection of plausible mappings

As mentioned above, the preliminary hierarchical pattern matching has been designed in order to move aside most unrealistic structural rearrangements, that is to say those that significantly affect the carbon-oxygen skeletons of molecules involved in the reaction. The

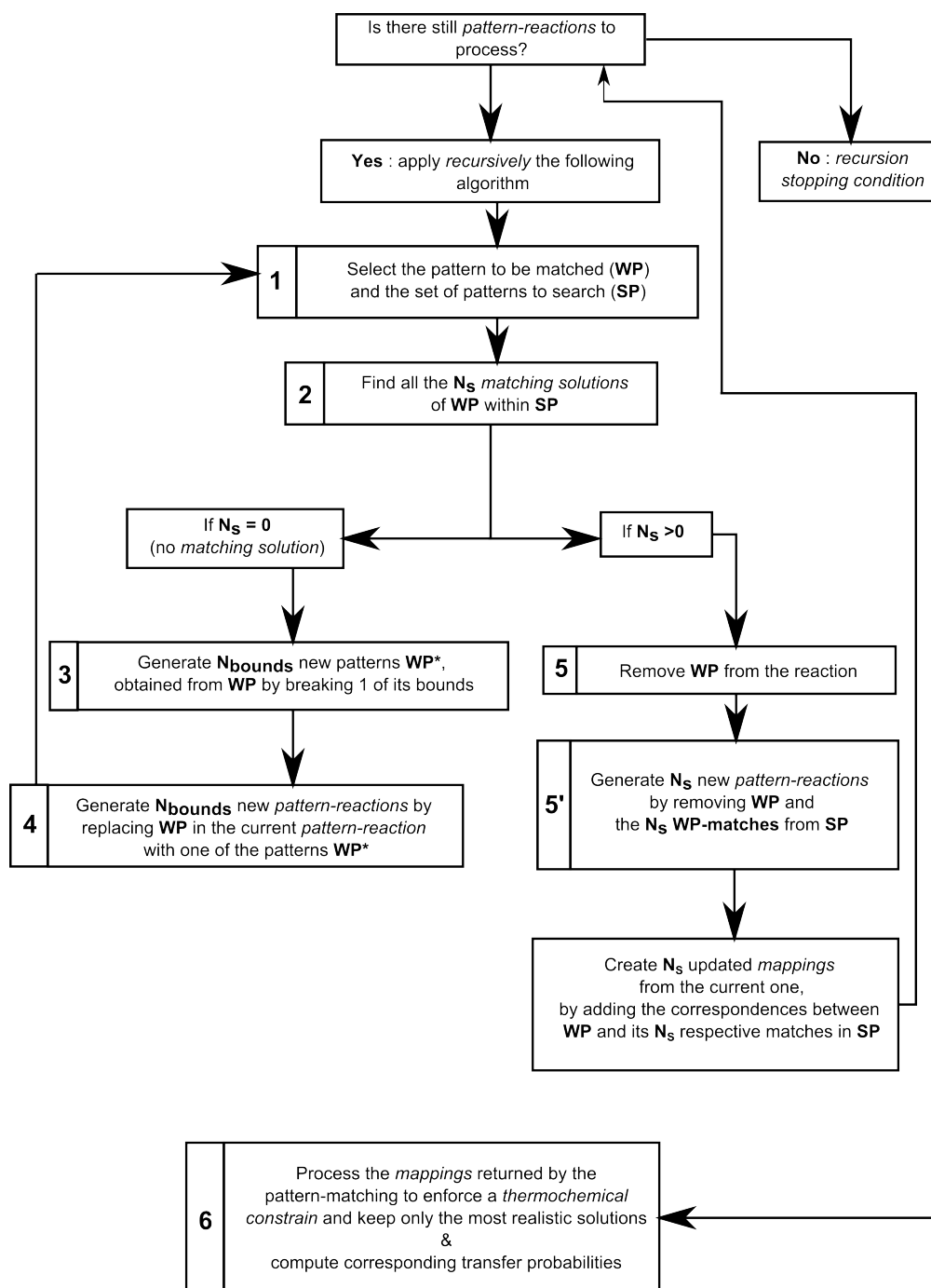


Figure 2.11: *Hierarchical Pattern Matching Algorithm* flowchart.

output of the first stage of the CHPMA is therefore a set of mappings, *i.e.* a set of permutation matrices containing the correspondence between atoms in the reactants and atoms in

the products, each one of those representing a different structural rearrangement. However, since some of the resulting mappings are not compatible with a reasonable transition state during the reaction pathway, this set of potential solutions need to be refined. This is realized thanks to a constrained selection of the final mappings, depicted in **Step 6** in Fig. 2.11. As the main defect of the pattern matching is that it ignores any chemical consideration, the selection process implemented in this last step aims at taking into account energetic perturbations, such as covalent bonds rupture or formation, associated to each mapping. Once more, it is unfeasible to implement detailed quantum chemistry equations, and simple averaged thermochemical rules have to be used. In regard to this matter, for each mapping \mathcal{M} , we define a *reconfiguration energy* $\mathcal{E}_{\mathcal{M}}$, which is a measure of the level of thermochemical perturbation induced by the mapping \mathcal{M} . Then, mappings minimizing the value $\mathcal{E}_{\mathcal{M}}$ are those minimizing the chemical potential energy of the reaction. In this second algorithm, detailed structural features, such as covalent bond types, radicals and hydrogen atoms, are taken into account in the molecules graph representations. However, we need to address separately the case where the reaction is not truly elementary, that is, contains species that are not actual molecules, but rather, only representative of a set of isomers species. Those *lumped* species, used in order to reduce the complexity of the kinetic mechanism or due to a lack of reaction rate information, usually have a set skeleton structure, but bond type and radical location, if applicable, are not specified. The following sections provide details and examples of the calculation of $\mathcal{E}_{\mathcal{M}}$ for these two cases.

Calculating $\mathcal{E}_{\mathcal{M}}$ for reactions involving lumped species

The mechanisms used in this work often include lumped species, that is, species representing several distinct isomeric forms of a same molecule, introducing some uncertainty on the locations of double covalent bonds or radicals in those. In the case where the reaction

considered contains lumped species, it is therefore impossible to take into account covalent bond types and hydrogen atoms. Then, like previously, the molecules are simplified to their pattern representations. In this case, we define the reconfiguration energy as the number of broken covalent bonds induced by the mapping \mathcal{M} :

$$\mathcal{E}_{\mathcal{M}} = N_{\text{broken bonds}} \quad (2.22)$$

We notice that during the pattern-matching process, a broken covalent bond can be induced at **Step 3** or **Step 5'**, as shown in Fig. 2.11. For each mapping, the number of induced broken bonds is computed on-the-fly during the matching process, and the final constrained selection simply consists of keeping the mappings with a minimum associated reconfiguration energy. Furthermore, most of lumped species in the mechanisms we used are large poly-cyclic molecules (due to several isomeric forms). Yet we noticed that the preliminary pattern-matching is particularly efficient to keep only realistic mappings for reactions involving such large molecules. For almost all PAHs reactions, the hierarchical pattern matching itself is sufficient to provide the set of mappings whose the reconfiguration energy is minimum, and in this case the constrained selection is not necessary. However, there are also lumped species which are not large poly-cyclic molecules, such as C_8H_{17} (due to several possibilities for the radical location), and the computation of the reconfiguration energy is primordial for those: indeed, in the example of the reaction $\text{C}_8\text{H}_{17} \rightarrow \text{C}_4\text{H}_8 + \text{C}_4\text{H}_9$, the hierarchical pattern matching provides 180 mappings as potential solutions, and only 72 of them are kept as final solutions by the constrained selection. Figures 2.12(a) and 2.12(b) provide examples of two mappings for which we have $\mathcal{E}_{\mathcal{M}_1} = 1$ and $\mathcal{E}_{\mathcal{M}_2} = 3$: the mapping 2 is therefore moved aside.

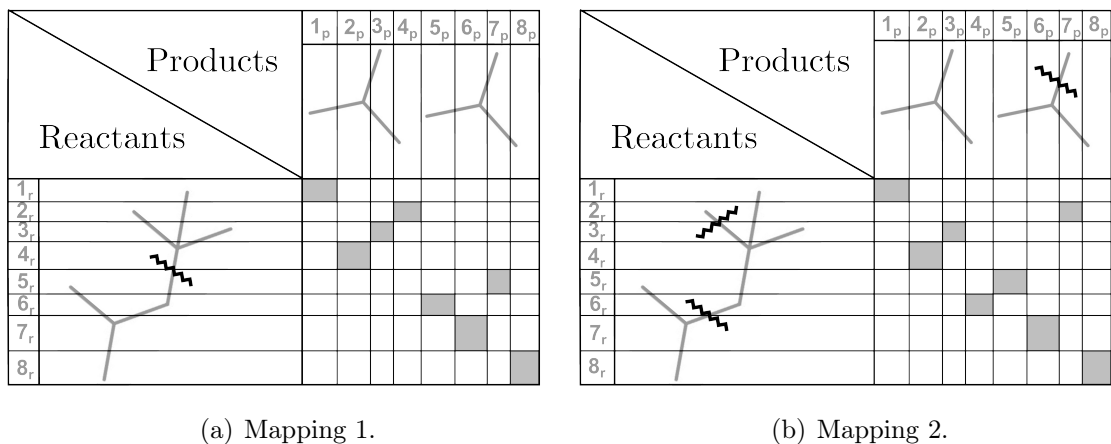


Figure 2.12: Two of the mappings output by the hierarchical pattern matching for the reaction $\text{C}_8\text{H}_{17} \rightarrow \text{C}_4\text{H}_8 + \text{C}_4\text{H}_9$. An index i_r (resp. j_p) refers to the carbon atom of the reactants (resp. products) located in the same row (resp. column). The gray cases show the correspondence between atoms i_r from the reactants and atoms j_p of the products. The wavy lines show the broken covalent bonds induced by the mapping. Mapping 1 induces one broken covalent bond 4_r-6_r . Mapping 2 induces three broken covalent bonds: two in the reactants: 2_r-4_r and 6_r-7_r , and one in the products: 6_p-7_p .

$\mathcal{E}_{\mathcal{M}}$ calculation when no lumped species is involved in the reaction

In the case where there is no lumped species involved in the reaction, additional structural information, such as covalent bond type, and locations of radicals and hydrogen atoms, is extracted from the molecule and used to compute the reconfiguration energy. $\mathcal{E}_{\mathcal{M}}$ has to account for several types of thermochemical perturbations affecting the molecules structures: indeed, a covalent bond can be either created through the reaction dynamic, on the contrary broken, or transformed into another bond type, or transferred unchanged. We define then the reconfiguration energy of a mapping as \mathcal{M} :

$$\mathcal{E}_{\mathcal{M}} = \sum_{b \in \mathcal{B}_c} E_b + \sum_{b \in \mathcal{B}_b} E_b + \sum_{b \in \mathcal{B}_t} |E_{b,p} - E_{b,r}| \quad (2.23)$$

Where:

- \mathcal{B}_c is the set of bonds of the products that are created through the mapping \mathcal{M} .
- \mathcal{B}_b is the set of bonds of the reactants that are broken through the mapping \mathcal{M} .
- \mathcal{B}_t is the set of bonds that are transferred (changed or unchanged) through the mapping \mathcal{M} .
- E_b is the bond energy of bond b , characterizing its strength. $E_{b,p}$ and $E_{b,r}$, are respectively the bond energy of the transferred bond b in products and reactants. We notice that if the bond is transferred unchanged, the term $|E_{b,p} - E_{b,r}|$ is null.

First, we can notice that, from a strict point of view, computing the reconfiguration energy requires to use bond energies, which depends of steric factors linked to the the molecules geometries. However, in the simplified context of this work, constant averaged bond energies (summarized in Table 2.1) are used. It is also interesting to note that the reconfiguration energy defined in the case of lumped species (Eq. 2.22) can be found back from Eq. 2.23 by assigning a value of unity for all bond energies. Finally, from a mathematical point of view, Eq. 2.23 defines a norm on the mapping-space, and the purpose of the constrained selection is therefore to find mappings that minimize this norm.

As an example, the thermochemically constrained selection is then applied to a reaction shown in Fig. 2.13, for which the preliminary hierarchical pattern-matching gives 24 mappings. Four of these mappings, as well as the structural changes they induce, are presented in Figures 2.14(a), 2.14(b), 2.14(c), and 2.14(d).

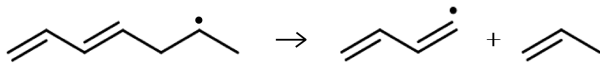


Figure 2.13: Example reaction $\text{x13C}_7\text{H}_{11}6 \rightarrow \text{N-C}_4\text{H}_5 + \text{C}_3\text{H}_6$.

Bond Type	Average Bond Enthalpy (KJ/mol)
O-O	146
C-C	348
C-O	358
C-H	413
O-H	463
O=O	495
C=C	614
C=O	799
C≡C	839
C≡O	1072

Table 2.1: Summarizing relative variations of bond energy calculations, adapted from

For Mapping 1, the detail of the reconfiguration energy calculation gives:

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}_1} = & \underbrace{E_{1_p-H} + E_{5_p-6_p} + E_{7_p-H}}_{\text{bonds created}} \\
& + \underbrace{E_{1_r-H} + E_{1_r-2_r} + E_{3_r-H} + E_{5_r-6_r}}_{\text{bonds broken}} \\
& + \underbrace{|E_{1_p-2_p} - E_{2_r-3_r}| + |E_{6_p-7_p} - E_{6_r-7_r}|}_{\text{bonds transformed}}
\end{aligned} \tag{2.24}$$

The corresponding detailed reconfiguration energy for mappings 22, 23 and 24 are shown in Fig. 2.15. The values of $\mathcal{E}_{\mathcal{M}}$ for the 24 mappings provided by the pattern-matching are tabulated in Table 2.2. According to these values, only the mapping 24 (Fig. 2.14(d)) is kept as a final solution. From a chemical point of view, that means that the most favorable reaction dynamic is the rupture of the covalent bond $3_r - 4_r$, followed by the pooling of a free electron between atoms 2_r and 3_r to create a double covalent bond.

Once the final mappings have been identified, those are processed in order to compute the transfer probabilities (**Step 6** in Fig. 2.11), according to Eq. 2.14.

Products \ Reactants		1p	2p	3p	4p	5p	6p	7p
1r								
2r								
3r								
4r								
5r								
6r								
7r								

(a) Mapping 1.

Products \ Reactants		1p	2p	3p	4p	5p	6p	7p
1r								
2r								
3r								
4r								
5r								
6r								
7r								

(b) Mapping 22.

Products \ Reactants		1p	2p	3p	4p	5p	6p	7p
1r								
2r								
3r								
4r								
5r								
6r								
7r								

(c) Mapping 23.

Products \ Reactants		1p	2p	3p	4p	5p	6p	7p
1r								
2r								
3r								
4r								
5r								
6r								
7r								

(d) Mapping 23.

Figure 2.14: Four of the mappings output by the hierarchical pattern matching for the reaction $\text{x13C}_7\text{H}_{11}6 \rightarrow \text{N-C}_4\text{H}_5 + \text{C}_3\text{H}_6$. An index i_r (resp. j_p) refers to the carbon atom of the reactants (resp. products) located in the same row (resp. column). The grey cases show the correspondence between atoms i_r from the reactants and atoms j_p of the products. The bond changes induced by the mapping are directly indicated on the molecules: wavy lines show broken covalent bonds, arrowed lines are bonds created through the mapping, dotted circles show bonds that are transformed, and bonds without any specific mark are the ones that are transferred unchanged.

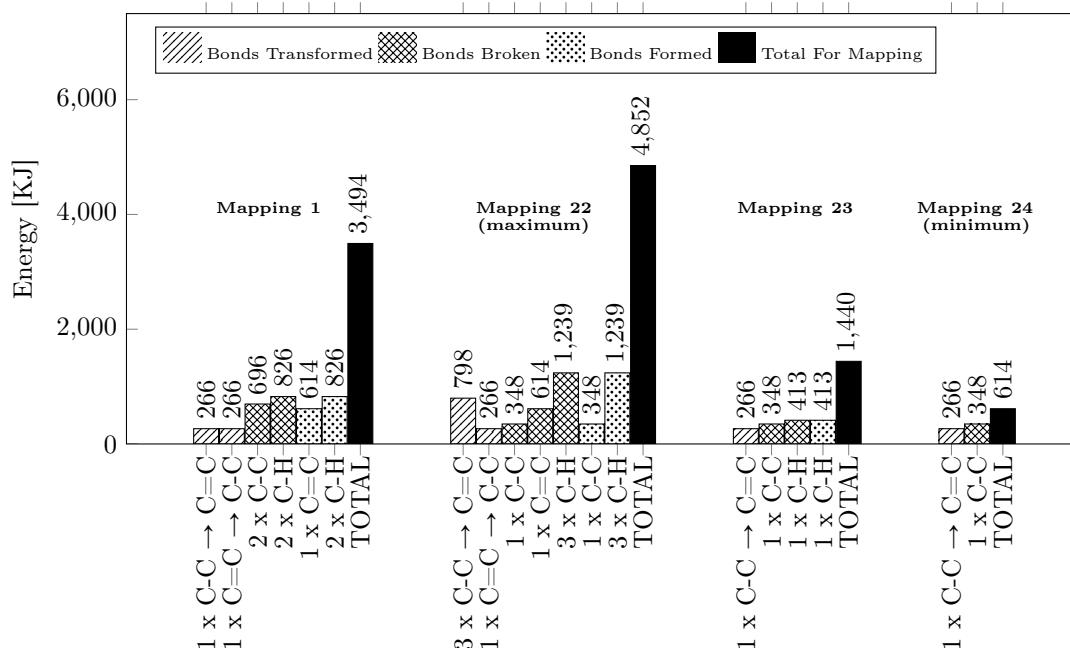


Figure 2.15: Detailed energy calculations showing bonds created, broken, transformed, or transferred unchanged, for mappings 1, 22, 23, and 24.

Mapping Number	Total Reconfiguration Energy (KJ/mol)	Mapping Number	Total Reconfiguration Energy (KJ/mol)
1	3494	13	2266
2	2136	14	1440
3	4320	15	3494
4	2962	16	2136
5	2374	17	4320
6	3200	18	2962
7	4026	19	3200
8	4026	20	4026
9	2798	21 (max)	4852
10	3330	22 (max)	4852
11	3624	23	1440
12	4156	24 (min)	614

Table 2.2: Reconfiguration energy for the 24 mappings provided by the pattern-matching algorithm, corresponding to the example reaction shown in Fig. 2.13.

2.5 Applications and examples

The numerical tracking framework, composed of the CHPMA and the transfer equations formulated in Section 2.3.2, has been implemented in an in-house kinetic code, and used

to simulate a various initial conditions in an homogeneous isochor reactor. The kinetic mechanism used in this section is the high-temperature part of the mechanism developed by Narayanaswamy *et al.* [33, 34, 35, 36], composed of 235 species and 2187 reactions. This mechanism is especially well-suited to demonstrate the benefits of the tracking approach, since it contains consistent submodules describing the oxidation of a variety of hydrocarbon fuels. The tracking equations are integrated in time, along with the species concentration equations, using the stiff ODE solver DVODE [37]. Conservation properties of labeled quantities, as described in Eqs. 2.10 and 2.11 are checked and satisfied at all times.

In the following, some useful notations for post-processing are first introduced. This is followed by several practical examples displaying some of the new capabilities provided by the tracking approach described here. They include an illustration of individual atom tracking, and how it can be used to quantify preferential concentrations of some initial fuel structural blocks in products, and fuel-specific preferential contributions to products in multi-component mixtures. Examples focus here on the formation of soot precursors, especially large aromatic species such as naphthalene. To obtain a consistent picture, fixed conditions, summarized in Table 2.3, are used throughout the examples. Three different hydrocarbons are used as fuel: *n*-heptane (NC7), toluene (TOL), and methyl-cyclohexane (MCH). The fuel/oxygen mixture is initialized for all cases at a temperature $T = 1200K$ and a pressure of 1 bar. A fixed 3% molar carbon content and a molar C/O ratio of 2 are used, yielding slightly different equivalence ratios for each case, from 5.14 to 6.28. All simulations are run for a total time of 0.5 second.

As mentioned above, for each label tracked in the chemical kinetic network, an additional $\sum_{i=1}^{n^S} n_i^A$ equations need to be solved as well. For the chemical mechanism used in these examples, this represents a total of 1530 equations, in addition to the 236 equations needed to the species concentrations and temperature. While not prohibitive provided that the

Simulation Index		C1	C2	C3	C4
Fuel		NC7	TOL	MCH	NC7 (80% mol) TOL (20% mol)
Initial composition	NC7	0.0043			0.0034
	TOL		0.0043		0.0009
	MCH			0.0043	
	O ₂	0.0075			
	N ₂	0.0098			
Equivalence ratio		6.28	6	5.14	6.05
Pressure		1 bar			
Temperature		1200 K			
Carbon molar content		3 %			
C/O ratio		2			

Table 2.3: Parameters for the homogeneous, isochor reactor simulations used in Sections 2.5.1 to 2.5.2.

number of labels tracked per simulation remains small, the large number of time-evolving variables emphasizes the need for targeted post-processing approaches, directly extracting relevant and meaningful information from the large volume of data generated. This process is illustrated in the examples below.

2.5.1 Atom selectivity in combustion products

The first example investigates how the initial structure of a fuel molecule affects the formation of combustion products. If each fuel atom is labeled individually at the beginning of a simulation, a straightforward approach is to select a molecular species of interest, S_i , and look at where, on average, each atom j in this species is coming from at a given time t . This information is directly provided by the quantities $\underline{\alpha}_i^{*k}$ defined in Eq. 2.16. Here, cases C1, C2, and C3 (Table 2.3) are run successively, labeling in each case, each of the fuel atoms individually. This configuration is illustrated for methyl-cyclohexane in Fig. 2.2(b). We select naphthalene (A_2) at the end of the simulation ($t = 500$ ms) as our species of interest,

and record $\underline{\alpha}_{A_2}^{*k}$ for each $*k$, corresponding to initial fuel atom locations. Results are provided in Fig. 2.16.

Each fuel molecule containing 7 skeleton atoms (in this case, carbon only), an equal contribution from each of the fuel atoms to a location j in A_2 would result in:

$$\alpha_{i;j}^{*k} = \frac{1}{7} = 0.1428, \text{ or } 14.28\% \text{ for all } *k. \quad (2.25)$$

Fig. 2.16 shows that for both *n*-heptane and methyl-cyclohexane, an close-to-equal contribution of all fuel atoms to each location in A_2 is obtained, very likely indicating that naphthalene is formed by recombination of small hydrocarbon molecules indiscriminately of their origins: there is no preferential pathway between atoms in the fuel and atoms in the product. In the case of toluene, however, a strong variability is observed. For example, atoms at the central location 1 in naphthalene are predominantly labeled *2 and *3, that is, coming from the tertiary carbon in toluene, and those adjacent to it in the aromatic cycle. This points toward the existence of a significant, more direct pathway between toluene and A_2 , for instance, the addition of a C_3 species on the methyl branch of toluene to form the second ring, while pathways going through small hydrocarbon pieces contribute to a lesser extent.

To consolidate the information contained in the $\underline{\alpha}$ values into one single, better conditioned quantity, we define the *selectivity* $\sigma_i^{\mathcal{F}}$ of a species S_i with respect to the skeleton atoms in a fuel molecule \mathcal{F} as:

$$\sigma_i^{\mathcal{F}} = \frac{\left[\frac{1}{n_{\mathcal{F}}^A} \sum_{*k \in \mathcal{F}} \left(K_i^{*k} - \frac{n_i^A C_i}{n_{\mathcal{F}}^A} \right)^2 \right]^{1/2}}{\sqrt{n_{\mathcal{F}}^A - 1} \frac{n_i^A C_i}{n_{\mathcal{F}}^A}}, \quad (2.26)$$

where :

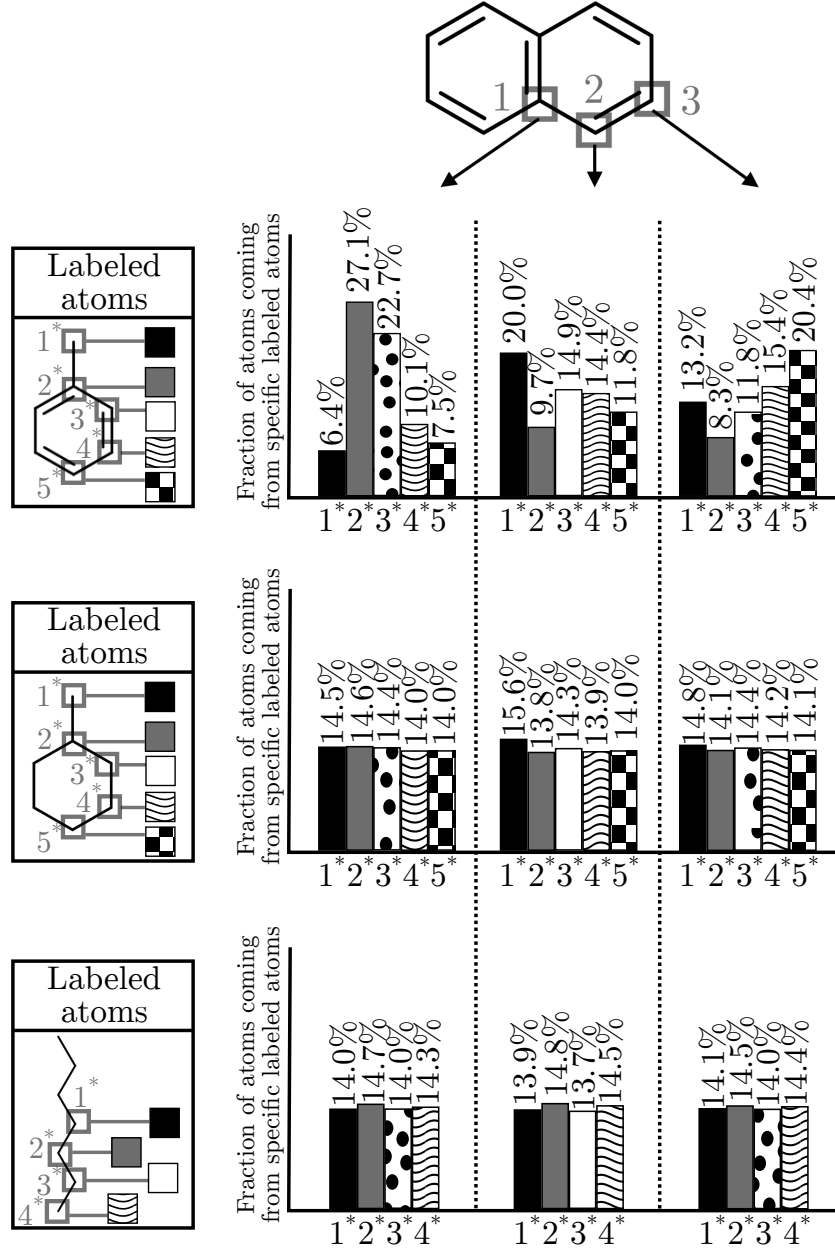


Figure 2.16: Origin of atoms ($\alpha_{i,j}^{*k}$) at different locations in naphthalene, for cases C1, C2, and C3 (Table 2.3), showing much larger variability for toluene than for heptane or methylcyclohexane. All atoms of each fuel molecule have been labeled individually (note that non-labeled atoms in the fuels and A₂ can be deduced from symmetry of the molecules), and all results are taken at $t = 500\text{ms}$.

- K_i^{*k} is defined in Eq. 2.9,
- The term $\frac{n_i^A C_i}{n_{\mathcal{F}}^A}$ is the average of K_i^{*k} over all labels $*k$ in the fuel molecule \mathcal{F} .
- The numerator corresponds to the standard deviation of the set of data $\{K_i^{*k}\}_{*k \in \mathcal{F}}$.
- $\sqrt{n_{\mathcal{F}}^A - 1}$ is a normalization term enforcing that $\sigma_i^{\mathcal{F}}$ be bounded between 0 and 1.

The selectivity $\sigma_i^{\mathcal{F}}$, which can be interpreted as the relative standard deviation of the set of data $\{K_i^{*k}\}_{*k \in \mathcal{F}}$, measures the *dispersion*, or equivalently, the inhomogeneity, of the contributions of each atom of a fuel \mathcal{F} to a species of interest, S_i . Thus, a value $\sigma_i^{\mathcal{F}} = 1$ means that S_i is exclusively formed from one, and one only, atom in the fuel molecule \mathcal{F} . On the contrary, a value $\sigma_i^{\mathcal{F}} = 0$, means that all atoms initially in \mathcal{F} contribute equally to the formation of S_i . This quantity can be used to quantify *a priori* the effect of the potential emergence of competing pathways in the kinetic network.

$\sigma_i^{\mathcal{F}}$ was computed at $t = 500$ ms for cases C_1 , C_2 , and C_3 , each case considering a different fuel \mathcal{F} (NC7, TOL, MCH). Here, i refers to the following set of intermediate or product species:

$$S_i \in \{\text{CH}_4, \text{C}_2\text{H}_2, \text{C}_2\text{H}_4, \text{C}_2\text{H}_6, \text{C}_3\text{H}_8, \text{A}_2\}. \quad (2.27)$$

Results are shown in Fig. 2.17. Consistent with the observation on naphthalene in Fig. 2.16, we observe that all species considered do exhibit preferential concentrations of certain atoms in the fuel, but to various extent. For example, again, toluene is associated with the highest selectivity, with a 15% value obtained in CH_4 and C_2H_4 . The intuitive explanation is that small hydrocarbon fragments are likely to come not from the aromatic ring, but rather from the loss of the methyl branch, or, when the ring opens up, from the extremities of the resulting linear molecule. Note that for heptane and methyl-cyclohexane, the product species considered here can be split into two groups: the first one, containing CH_4 , C_2H_6 , and C_3H_8 , show higher selectivities than the other species, C_2H_2 , C_2H_4 , and A_2 . This indicates that

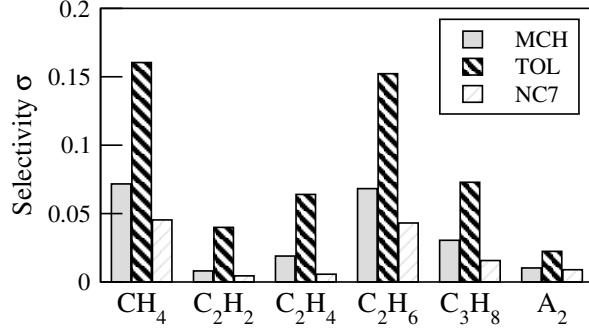


Figure 2.17: Atom selectivity σ_i^F for the three fuel molecules considered in this study. A value of 0 indicates equal contribution of all atoms in the fuel to the species, while a value of 1 would indicate that only one atom in the fuel participates in the formation of the species.

those species are obtained initially in significant amount through direct break-up of the fuel, retaining specificities from the fuel break-up pattern, even after a long residence time. In contrast, the second group of species appears to be secondary species, whose formation involves a larger number of pathways, thereby mixing better all atoms available.

2.5.2 Atom tracking in multi-component mixtures

The tracking algorithm is then applied to a mixture of two hydrocarbon fuels (case C_4), containing heptane (80% in volume) and toluene (20% in volume). The time evolution of some key species concentrations is displayed in Fig. 2.18, showing two very different characteristic times for fuel consumption, heptane being consumed quickly (1 ms) and entirely, while the toluene mole fraction remains roughly constant, equal to its initial value, over a period 10 times as long. Due to the high equivalence ratio, significant amounts of hydrocarbon species, including toluene, remain after oxidation.

This configuration is used to investigate the contribution of individual fuel components in the formation of combustion intermediates and products. For this purpose, at $t = 0$,

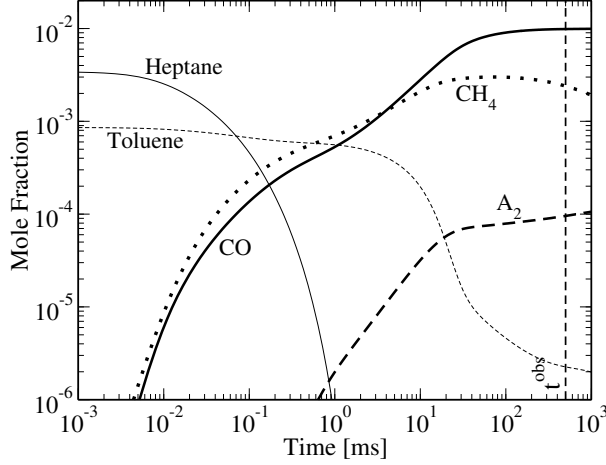


Figure 2.18: Time evolution of some key species concentrations for case C_4 . Note the logarithmic scale used for time, in order to better show the various characteristic times involved.

all atoms in toluene are labeled * (only one label being used, the integer is dropped for simplicity), while the atoms forming heptane and molecular oxygen are left untouched. This corresponds to the configuration depicted in Fig. 2.3. At $t = 500$ ms, the value K_i^* for each species S_i in the system is recorded, and normalized by the local concentration of skeleton atoms in S_i , yielding the fraction of C^* in each species:

$$\beta_i = \frac{K_i^*}{n_i^A C_i} \quad (2.28)$$

The values β_i , $i = 1$ to n^S , are then averaged conditioned on the the number of carbon in the corresponding species, the resulting conditional averages being plotted in Fig. 2.19.

A very clear trend can be observed, with β nearly monotonically increasing with the number of carbons in the species. Small hydrocarbon species (less than 5 carbons) are primarily formed by atoms initially contained in n -heptane, while large molecules contain a disproportionate amount of atoms originating from the initial toluene molecules. Large gas-phase species are precursors to soot particulates. Therefore, this result is very consistent with numerous experimental observations, for instance the work of Homan *et al.* [23], who showed that ring-forming carbons have a much higher probability to end up in soot in rich

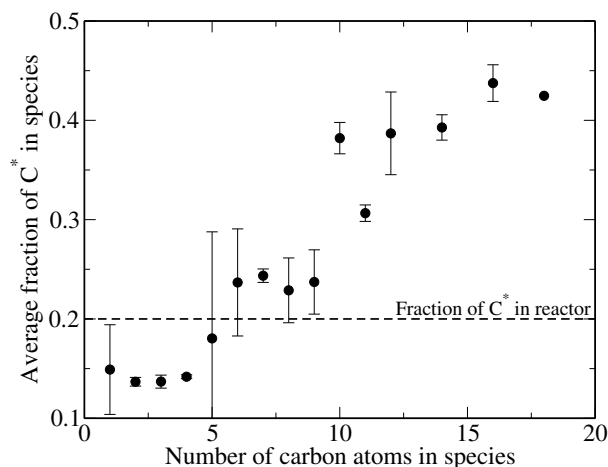


Figure 2.19: Average fraction of C* in species present at $t = 500$ ms for Case C_4 , conditioned on the number of carbon in the species. Error bars indicate for each number of carbon atoms the minimum and maximum values observed. The dashed line indicates the average fraction of labeled atoms, that is, initially coming from toluene molecules, in the system.

combustion.

2.6 Conclusion

A novel tracking algorithm has been developed and implemented in simple homogeneous, constant-volume reactor simulations. The algorithm contains three main ingredients: a systematic and tailored treatment of the structural features of all species involved in the kinetic model, a pattern matching algorithm relying on graph isomorphisms to identify a list of potential atom re-arrangements, or mappings, between reactants and products occurring in elementary reactions, and a selection technique to identify only those mapping compatible with the expected transition states, that is, those involving minimal energy and structural changes. The method has been applied to several cases involving one or several fuels, in order to demonstrate its new capabilities compared to conventional kinetic mechanism analysis. While no direct comparison with experimental data is done, the results are qualitatively

consistent with existing experimental work. Future work involves a better treatment of non-conventional reactions, that is, reactions with complex transition states and additional validation of the results. The method is ideally suited to address some recurrent questions in combustion chemistry, including how to best define surrogate fuels for experimental and numerical investigation of realistic combustion devices, and soot mitigation strategies using oxygenated molecules.

APPENDIX A

USER GUIDE: RUNNING KINSOLVE WITH TRACKING EQUATIONS ACTIVATED

A.1 Introduction

This guide assumes that the `kinsolve/pp` branch of the Pepiot Research Group's Kinsolve repository is on the user's system and all dependencies for Kinsolve have been installed. The atom tracking algorithm branch of the Kinsolve repository is named `kinsolve/pp`. This branch is able to create two binary versions of Kinsolve originating from two separate directories within the root directory of the branch: `preprocessing/` and `simulation/`. The `preprocessing/` folder contains all the code to create a transfer matrix output file, while the `simulation/` folder includes the tracking equations needed to track atoms that have been labeled via the input file.

A.2 Steps for compiling and running pre-processing code

The following steps are required to compile and run the pre-processing side of the code.

1. Once in the root directory of the repository, first navigate to the `examples/` folder and open the `input` file. For the pre-processing side of the tracking algorithm code to run without errors, there are two lines which need to be included in the input file:

Tracking : true

Transfer file to write : (file name to store transfer coefficients in)

Here, (file name to store transfer coefficients in) can be any valid name for a file. This is the single file produced by the pre-processing side of the kinsolve code, containing the transfer matrices for every reaction in the mechanism. The transfer coefficients file is printed in binary format.

2. Ensure that the `Thermo file :` and `Mechanism file :` flags in the input file specify the correct thermodynamic and mechanism data files to track atoms for.
3. Issue `perl ../perl/preprocessor.pl input`

The Perl script `preprocessor.pl` reads both the thermodynamic and mechanism files from the input file and creates a Fortran version of the mechanism, `mechanism.f90`. Within the atom tracking branch of Kinsolve, this script also creates a `trackingvars.f90` file which includes all the variables required for the tracking algorithm. `preprocessor.pl` should run to completion without throwing errors, however, if a new chemical mechanism is being used, `preprocessor.pl` will provide warning messages for each missing molfile. For more information about adding new molfiles, see Subsection A.4.1.

4. Move the newly created `mechanism.f90` and `trackingvars.f90` files into the `preprocessor/src/chem` folder.

5. Navigate to `preprocessor/src/`

6. Issue the following two commands:

```
make distclean
```

```
make debug
```

7. The package takes a long time to compile when using the `make debug` compiling flag due to the size of the Boost Graph Library [32], which is needed for the pattern matching portion of the transfer coefficient calculation. If no errors are thrown and the following is shown:

```
make[2]: Nothing to be done for 'util'.
```

Then the binary was built properly and is ready to run. As of December 2015, there is no `make opt` flag for the pre-processing portion of the code.

8. Navigate back to the `examples/` folder

9. Issue `../bin/kinsolve input`

There will be a large number of outputs which appear in the terminal. It is the pre-processing code showing what reaction it is on and some of the calculations the

CHPMA algorithm is making.

10. Once the code has completed, there will be a new file by the name specified for the `Transfer file to write` : flag directly in the `examples/` folder. A simulation with tracking equations activated can now be run.

A.2.1 Pre-processing code functionality with reduced mechanisms

The pre-processing side of the code is currently set to create a transfer matrices file for three different mechanisms, the full high temperature surrogate proposed by Narayanaswamy *et al.* [33, 34, 35, 36] which has 235 species, a reduced mechanism with 169 species, and a further reduced mechanism with 108 species. There are three areas in `chpma.f90` which need to be modified before the pre-processing code can compile without errors. There are three such sections:

1. The correct aromatic and lumped species lists, lines 24–40 in `chpma.f90`.
2. The correct `brk_bonds_limit` for the mechanism of consideration, lines 208–243 in `chpma.f90`.
3. The correct if statements in the hydrogen-oxygen/hydrogen-carbon bond calculations, lines 558–579 in `chpma.f90`. For the 108 species mechanism, there is no molecular

carbon species (`C` in mechanism) and thus it cannot be included in the `if` statement.

For each of these sections, there are three sets of lines which correspond to the three mechanisms, well commented in each area. Additional components of the `chpma.f90` source code, describing how CHPMA operates, are described in Appendix B.

A.3 Steps for compiling and running simulation code

If a transfer matrix file has been built for the mechanism of interest, then a simulation can be run with atom tracking equations activated for that mechanism. The following are the steps required to compile and run such a simulation.

1. Navigate to the `examples/` folder and look at the `input` file. For the simulation side of the tracking method to work, there are a few key lines which need to be in the input file. In this example, atoms 2-6 are simultaneously labeled in methylcyclohexane:

```
Transfer file to read :  transfer_matrices.f90
```

```
Tracking output file :  trac_MCHC7H14_23456
```

```
Tracking species :  MCH-C7H14
```

```
Atom number :  2 3 4 5 6
```

2. Navigate to `simulation/src/`

3. Issue the following two commands:

```
make distclean
```

```
make debug
```

4. If no errors are thrown and the following output is shown:

```
make[2]: Nothing to be done for 'util'.
```

then issue:

```
make distclean
```

```
make opt
```

It is important to do `make opt` for the simulation side of the code, since code will be solving many ODEs and runs quickly only if the `make opt` flag has been used for compilation. If the following is shown:

```
make[2]: Nothing to be done for 'util'.
```

then the binary has compiled correctly and is ready to run.

5. For the simulation portion of the code, due to the high number of output files generated, it is recommended to move the Kinsolve binary file, the input file, and the

transfer matrix file outside of the repository all to the same folder and run the code there. As with the pre-processing portion of the code, the simulation code binary can be run with the following command:

```
../bin/kinsolve input
```

6. The current simulation code is set up to produce a variety of output files to assist with post-processing of the data. The files and their contents are summarized in Table A.1.

The **Atom number** : input file flag, specifying which atoms to label in the tracking species, has multiple options. The first and most simple option is to specify a single atom index on the species of interest, *e.g.* 0 or 5 or 7. The second option is any list of atoms indices separated by single spaces *e.g.* 0 1 2 3 or 7 6 5 or 5 3 6 1. Finally, specifying -1 as the atom index prompts the code to label all atoms in the species to be tracked simultaneously. Because these indices are numbered according to the `trackervars.f90` file, the atom indices start at 0. [/perl/molefiletex/allgraphs.pdf](#) can be viewed to see how atoms are numbered in all species in the molfile database. In some molecules, the atom indexing is often not intuitive, so it is recommended to inspect `allgraphs.pdf` before providing atom indices to label in the input file.

A.3.1 Post-processing simulation data

There is a collection of Python scripts that have been written for post-processing of tracking data. Details on these scripts can be found in Appendix D.

Output Filename	Description of Contents
profile_(tracname)_C.dat	Concentration profiles.
profile_(tracname)_Y.dat	Mass fraction profiles.
profile_(tracname)_X.dat	Mole fraction profiles.
trac_(tracname)_check	Sum of all tracked atoms #1 and #2 for all times.
trac_(tracname)_det	The concentration amount of tracked atom #1 in atom each species.
trac_(tracname)_gl	The sum of tracked atoms #1 and #2 in each species.
trac_(tracname)_total	The concentration amount of tracked atom #1 summed over all atoms in a species.
trac_(tracname)_specp	The percentage composition of tracked atom #1 in each species.
trac_(tracname)_detpr	Each species-reaction rate by pure chemistry, species of interest are hardcoded at the top of reactor.f90.
trac_(tracname)_tracdetpr	Each species reaction rate by amount of tracked atom #1 in species, species of interest are hardcoded at the top of reactor.f90.



Table A.1: The complete list of chemistry and tracking output files created by the simulation side of the kinsolve/pp branch binary. Here, (tracname) will be the name specified under the Tracking output file : trac input file flag.

A.4 Molfiles and the molfile database

A.4.1 Adding new molfiles to the molfile database

The current code is compatible with the species in the fuel surrogate proposed by Narayanaswamy *et al.* [33, 34, 35, 36]. All molfiles in the database have been derived

from the naming conventions used in this mechanism. If a different mechanism is to be used with new species, any newly required species can be discovered directly by running `preprocessor.pl` during the pre-processing side of the code. `preprocessor.pl` will print each molfile it cannot find in the `perl/molfiledatabase/` folder to the terminal. The final print from `preprocessor.pl`, will include a warning of how many molfiles could not be found. To remedy these warnings, add the missing molfiles with file name equal to the name provided by `preprocessor.pl` into the `perl/molfiledatabase/` folder. Additional source code added to the standard Kinsolve `preprocessor.pl` required for the atom tracking algorithm are discussed in detail in Appendix B.

There are two excellent online resources which can be used to find molfiles: 1. The species name search at the National Institute of Standards and Technology website (NIST) at <http://webbook.nist.gov/chemistry/name-ser.html> which has molfiles under the ‘2d Mol file’ links, and 2. ChemSpider <http://www.chemspider.com/>, where molecules can be searched by CAS number or species name, and then exported by clicking the  button. For structures unique to a mechanism where a CAS number may be difficult to find, molecules can be drawn on <http://www.emolecules.com> and then exported as a molfile by clicking the  button.

A.4.2 Visualization and verification of molfiles

Within the `perl/` folder there are a variety of scripts and directories all relating to analysis, visualization, and maintenance of the molfile database. The molfiles, files with `.mol` extensions, are the files physically read by the Perl script `preprocessor.pl` and are located in the

folder `molefiledatabase/`. Files containing `.png` and `.dot` representations of the molfiles can be found in the `molfilegraphs/` folder. The `.pngs` are created by the use of Graphviz [38] command line tools. Graphviz is a graph drawing library with files typically given a `.dot` file extensions. An example of a graph `.dot` file for benzyl radical, named A1XC7H7 in the mechanism, is shown in Fig. A.1. Details of the Graphviz syntax and commands are outside the scope of this manual, but are described in detail in the manual by Ellson *et al.* [38].

The `.png` files are compiled into a single `.tex` file called `allgraphs.tex` within the `molfiletex/` folder. To orchestrate this entire process, issue following within the `molfile_src/` folder:

```
sh build_vis_tex.sh
```

The script `build_vis_tex.sh` executes three different scripts, two in Perl and one in Bash:

1. `perl molfilechecker.pl` - simultaneously checks for proper structure of the molfiles themselves while printing separate Graphviz `.dot` files for each molecule.
2. `sh graph2png.sh` - uses Graphviz command line tools to convert each Graphviz `.dot` file into a `.png`.
3. `perl visualizeallgraphsintex.pl` - creates a file `allgraphs.tex` with one `.png` graph per page. `allgraphs.tex` will be automatically opened by the default `.tex` file editor of the machine running the script.

```

/* ### Graph representing ../molfiledatabase/A1CH2XC7H7.mol ### */
graph {
  layout=neato;
  overlap=false;
  spline=true;
  forcelabels=true;
  graph [fontsize=20, autosize=false, size="1.00,1.00", dpi=400];
  node [shape=circle,width=0.1];
  aaq1 - aaq2;
  aaq1 - aaq2;
  aaq1 [label="0"];
  aaq2 [label="1"];
  aaq1 - aaq3;
  aaq1 [label="0"];
  aaq3 [label="2"];
  aaq1 - aaq7;
  aaq1 [label="0"];
  aaq7 [label="6*"];
  aaq2 - aaq4;
  aaq2 [label="1"];
  aaq4 [label="3"];
  aaq3 - aaq5;
  aaq3 - aaq5;
  aaq3 [label="2"];
  aaq5 [label="4"];
  aaq4 - aaq6;
  aaq4 - aaq6;
  aaq4 [label="3"];
  aaq6 [label="5"];
  aaq5 - aaq6;
  aaq5 [label="4"];
  aaq6 [label="5"];
}

```

Figure A.1: Example of the Graphviz language in a .dot file for benzyl radical, A1XC7H7.

The input and output organization for this process is:

Script	Input	Output File(s)
<code>build_vis_tex.sh</code>	none	runs the following three scripts
<code>molfilechecker.pl</code>	none	database of Graphviz files in location: <code>molfilegraphs/*.dot</code>
<code>graph2png.sh</code>	none	database of <code>.png</code> files in location: <code>molfilegraphs/*.dot.png</code>
<code>visualizeallgraphsintex.pl</code>	none	<code>allgraphs.tex</code>

Table A.2: Organization of scripts which create a `.tex` file from the molfile database.

A.4.3 Full and “dummy” molfiles

Because the tracking algorithm only requires the types of atoms, connection table, bond types, and radicals, it is possible to make an incomplete or “dummy” molfile for the molecule. This is especially useful for species that have names unique to a specific chemical mechanism. As an example, Fig. A.2 shows a complete and “dummy” molfile for the benzyl radical with name `A1XC7H7` and CAS number 2154-56-5 is shown in Fig. A.2.

A “dummy” molfile represents the minimum contents a molfile needs to contain to provide to CHPMA: the number of bonds and atoms, all the atom types, the connection table, the bond types, and the radicals, if any.

A.5 Summary

With this guide it should be possible to run both the pre-processing and simulation parts of the atom tracking code, and resolve the issue when molfiles are missing from the database. A method of visualizing all existing molfiles in the `molfiledatabase` to a `.tex` file has also

```

2154-56-5.mol
ChemDraw07291019112D
7 7 0 0000 0 0 0 0 0 0999 V2000
-0.4091 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.7164 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 -0.7164 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1.2473 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.8242 0.7164 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.8242 -0.7164 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.2473 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1 0
2 3 2 0
3 4 1 0
4 5 2 0
5 6 1 0
6 7 2 0
7 2 1 0
M RAD 1 1 1
M END

```

```

dummy molfile for benzyl radical (A1XC7H7)

7 7
0 0 0 C
0 0 0 C
0 0 0 C
0 0 0 C
0 0 0 C
0 0 0 C
0 0 0 C
0 0 0 C
1 2 1
2 3 2
3 4 1
4 5 2
5 6 1
6 7 2
7 2 1
M RAD 1 1 1
M END

```

Figure A.2: Difference between full and “dummy” molfiles.

been shown, as well as a script which very carefully reads the molfiles to insure there are no bonds, bond types, or atoms missing, in the case of human errors from self-made “dummy”

molfiles. Two scripts critical for the atom tracking algorithm code, `preprocessor.pl` and `chpma.f90` are described in detail in Appendices B and C.

APPENDIX B

PREPROCESSOR.PL CODE ORGANIZATION AND DESCRIPTION

B.1 Introduction

This appendix assumes basic knowledge of Perl and Fortran by the user. Given an input thermodynamic and mechanism file, the `preprocessor.pl` script in the tracking branch `kinsolve/pp` of Kinsolve creates two output files: 1. `mechanism.f90`, a Fortran file which compiles the thermodynamic and mechanism files, completely categorizing the combustion chemistry, and 2. `trackingvars.f90`, a file which contains all information pertaining to the atom tracking algorithm, such as structure, bond types, and radicals for each species. This guide will focus on the components of `preprocessor.pl` specific to the atom tracking algorithm.

B.2 Printing `trackingvars.f90`

A total of six subroutines are called in `preprocessor.pl` before `trackingvars.f90` is printed in order to read, organize, and store the information in the molfiles into `preprocessor.pl`. Table B.1 provides a description of these six subroutines.

Once the six subroutines in Table B.1 are executed in `preprocessor.pl`, the `trackingvars.f90` file is printed. There are then seven additional subroutines that are called as the `trackingvars.f90` file is printed. Table B.2 summarizes the names of the components of the Fortran source code printed in `trackingvars.f90`, and the corresponding Perl subroutines which are called during the printing process.

Subroutine	Lines in <code>preprocessor.pl</code>	Description
<code>getFormulas</code>	2083–2102	Retrieves chemical formulas for each species, including the number of carbons, hydrogens, and oxygens.
<code>getAtomStatistics</code>	2104–2152	Defines the <code>nC</code> array, <code>nO</code> array, maximum number of atoms across all species, and maximum number of carbons across all species.
<code>getMaxNumReactantsAndProducts</code>	2154–2174	Finds the maximum number of reactants and products across all reactions.
<code>getMolfileData</code>	2176–2264	Distills the molfile for each species in the mechanism to Perl arrays.
<code>genEmptyAtomAndColumnArray</code>	2266–2283	Generates a global variable for an empty atom and column array to be used by all species with no carbon.
<code>getEnergyReac</code>	2285–2383	Generates an array of reactions requiring the energy-based calculations in CHPMA.

Table B.1: The six `preprocessor.pl` subroutines called before printing `trackingvars.f90`, their corresponding lines in `preprocessor.pl`, and description.

Fortran structure type	Name in trackingvars.f90	Approx. lines of printing in preprocessor.pl	Description
module	trackingvars	1870–1907	The module allocating amounts for the arrays defined in the <code>trackingvars.f90</code> subroutines.
subroutine	init_graph_mol	1910–1986	The complete structural description of each species distilled into four arrays: <code>atomlists</code> , <code>valence</code> , <code>adjacencylists</code> , and <code>bondtype</code> . An example of these arrays for a species in the mechanism is provided in Fig. B.1 and shown as a graph in Fig. B.2.
subroutine	get_nC_nO	1989–2023	Defines number of carbon and oxygen atoms in each species, as well as number of trackable atoms, the sum of carbon and oxygen atoms in a species.
subroutine	get_nTrrp	2027–2070	Defines <code>nTr</code> , <code>Trr</code> , <code>Trp</code> , <code>nTr3</code> and <code>Tr3</code> . A description of these arrays is provided in Table B.4.

Table B.2: Fortran components of `trackingvars.f90` source code, printed from corresponding sections of `preprocessor.pl`.

```

!-- Atom and adjacency list information for TXC4H9O --!
atomlists(sTXC4H90,1:1:maxnumatoms) =
(/0,1,2,3,4,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)
atomlists(sTXC4H90,2,1:maxnumatoms) =
(/2,1,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)
valence(sTXC4H90,1:1:maxnumatoms) =
(/-1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)
adjacencylists(sTXC4H90,1,1:maxnumbonds) =
(/1,1,1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)
adjacencylists(sTXC4H90,2,1:maxnumbonds) =
(/2,4,3,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)
bondtype(sTXC4H90,1:maxnumbonds) =
(/1,1,1,4,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1/)

```

Figure B.1: Example of four arrays in the source code which completely characterize a species' structure, in this case, the tert-butoxy species. Note this species has one radical and one oxygen atom. All of these specificities are captured within the arrays.

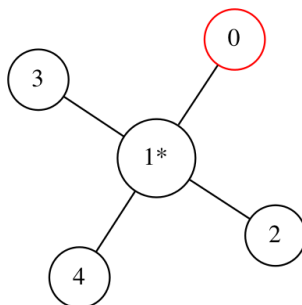


Figure B.2: Graph visualization of the tert-butoxy species.

Table B.3 summarizes the subroutines which are called as `trackingvars.f90` is being printed, and what component of `trackingvars.f90` they print to.

In Table B.3, it is clear that the Fortran subroutine `get_nC_n0` is printed by information determined by the six initial subroutines alone, while `init_graph_mol` is printed by the use of many additional Perl subroutines. Fortran subroutine `get_nTrrp` requires only the `getTrs` Perl subroutine, however `getTrs` is a detailed subroutine, clearly the longest in terms of the number of lines of source code according to Table B.3.

B.3 Summary

An overview of the `preprocessor.pl` script has been provided, with respect to the portions of the source code which are used for printing the `trackingvars.f90` file. Six key Perl subroutines are called to organize the data from the molfiles and store them as variables in the Perl script. These six Perl subroutines are called before printing `trackingvars.f90`. While `trackingvars.f90` is being printed, seven additional Perl subroutines are called to better organize and orchestrate the printing process. These various Perl subroutines are able to print the Fortran module `trackingvars.f90`, which has three subroutines.

Perl Subroutine Name	Name of Corresponding Fortran Subroutine Printed to in trackingvars.f90	Approx. Lines of Subroutine in preprocessor.pl	Description
genAtomIndices	init_graph_mol	2389–2411	Indexing of every carbon and oxygen in species.
genAtomTypes	init_graph_mol	2413–2436	Array of atom types: 1 for carbon, 2 for oxygen.
genValence	init_graph_mol	2438–2459	Creates an array containing the number of valence electrons in a species at each atom position.
genFirstCol	init_graph_mol	2461–2479	First column of the connection table from the molfile.
genSecondCol	init_graph_mol	2481–2507	Second column of the connection table from the molfile.
genBondTypes	init_graph_mol	2509–2569	Bond type of the corresponding bond between the first and second columns of the connection table. The bond numbering scheme is summarized in Table B.5.
getTrs	get_nTrrp	2571–2643	Generates arrays nTr , Trr , Trp , nTr3 and Tr3 . A description of these arrays is provided in Table B.4.

Table B.3: `preprocessor.pl` Perl subroutines which assist with printing `trackingvars.f90` Fortran subroutines. The lines of the Perl subroutines in `preprocessor.pl` are provided as well as a description of the Perl subroutine.

Array in <code>get_nTrrp</code>	Description
<code>nTr(react,1)</code>	number of reactants in reaction <code>react</code> .
<code>nTr(react,2)</code>	number of products in reaction <code>react</code> .
<code>Trr(react,:)</code>	names of reactants in reaction <code>react</code> .
<code>Trp(react,:)</code>	names of products in reaction <code>react</code> .
<code>nTr3(react)</code>	number of third bodies in reaction <code>react</code> .
<code>Tr3(react)</code>	name of third body in reaction <code>react</code> .

Table B.4: Descriptions of the arrays in the `get_nTrrp` Fortran subroutine.

Bond Type	Number
C-C	1
C=C	2
C≡C	3
C-O	4
C=O	5
C≡O	6
O-O	7
O=O	8

Table B.5: Bond types and their corresponding number. The numbering method used enables modification of bond energies in `chpma.f90` if needed.

APPENDIX C

CHPMA.F90 CODE ORGANIZATION AND DESCRIPTION

C.1 Introduction

This appendix assumes basic knowledge Fortran by the user. The `chpma.f90` code is the primary code on the pre-processing side of Kinsolve which calculates the transfer probabilities and matrices. This process includes conducting the recursive pattern matching of the species graphs, the reconfiguration energy calculation, the final selection of acceptable mappings, the calculation of transfer probabilities, and finally printing the transfer probabilities in matrix form to a file. Table C.1 summarizes all subroutines in `chpma.f90` which are able to accomplish these tasks.

C.2 Overview of reconfiguration energy calculation

The reconfiguration energy calculation requires calculating what bonds are formed, broken, or transformed as reactants are converted to products. Because the total structural information for each species is represented as only four arrays, this process becomes an involved calculation. The steps taken to calculate the reconfiguration energy are as follows:

1. **Read in mapping:** A mapping returned by CHPMA is unique list of reactant atom to product atom pairs. The reconfiguration energy sum is initialized at 0.
2. **Calculate reconfiguration of carbon-hydrogen and oxygen-hydrogen bonds:** Calculate and store the number of hydrogen bonds on each carbon or oxygen atom in

Subroutine	Lines in <code>chpma.f90</code>	Description
<code>get_Tr</code>	80–934	Calculates transfer matrices for each reaction in the mechanism.
<code>CHPMA_to_trackingvars</code>	945–990	Converts the global CHPMA indices of a species to its local species-level trackingvars indices, used in the reconfiguration energy calculation.
<code>get_H_energy</code>	992–1176	Counts hydrogen bonds at every atom position, used in the reconfiguration energy calculation.
<code>find_pair_in_array</code>	1179–1202	Finds if two indices in two arrays are adjacent according to the same index. Called often during the reconfiguration energy calculation.
<code>write_Tr</code>	1208–1262	Writes the transfer probabilities to a file in matrix form. The file is written in binary.
<code>pattern_select_CC</code>	1279–1321	Selection of pattern to match when the Wanted Pattern (WP) is only comprised of carbon atoms.
<code>pattern_select_OC</code>	1347–1410	Selection of pattern to match when there is at least one oxygen atom in the reaction.
<code>pattern_selection</code>	1425–1470	Selection of the pattern to match in the Searched Pattern (SP) according to the WP.
<code>pattern_matching</code>	1479–2066	Algorithm which conducts the recursive matching and removal of patterns.

Table C.1: All subroutines in `chpma.90`, their corresponding lines, and description.

the reactants. The same calculation is conducted for the atoms in the products. Any change in number of hydrogens from reactions to products for each atom is added to the configuration energy sum; the energy of a carbon-hydrogen bond is added if the atom of consideration is a carbon atom and the energy of a oxygen-hydrogen bond is added if the atom of consideration is an oxygen atom.

3. **Calculate any preserved or transformed carbon-carbon, carbon-oxygen, or oxygen-oxygen bonds:** For each pair of adjacent carbon-carbon, carbon-oxygen, or oxygen-oxygen bonds pairs the in reactants, an attempt is made to find the same pair of atoms in the products. If such a pair can be found, the bond was either preserved, *e.g.* a single carbon-carbon bond in the reactants remaining as a single carbon-carbon bond in the products, or transformed, *e.g.* a double carbon-oxygen bond in the reactants moves to a single carbon-oxygen bond in the products. For these calculations, any energy of transformation is added according to the transformed bonds term in Eq. 2.23 to the reconfiguration energy. Preserved bonds have an associated reconfiguration energy of zero. Additionally, preserved and transformed bonds are stored and will be used to determine if a search is required for formed bonds.
4. **Calculate any broken carbon-carbon, carbon-oxygen, or oxygen-oxygen bonds:** Any adjacent pairs of atoms, according to the given mapping, that could not be found in the products must have been broken apart in the reactants. These broken bonds are added to the reconfiguration energy.
5. **Search for formed carbon-carbon, carbon-oxygen, or oxygen-oxygen bonds, if any:** Access the stored preserved and transformed bonds. If the number of bonds preserved and transformed does not equal the total number of bonds in the products, then this mapping requires that at least one bond is formed in the products. The location and type of bonds formed are found by checking the stored transformed and

transferred bonds against the original list of product bonds stored by CHPMA when it initially reads the products side of the reaction. Once all the formed bond locations are found, the energy of the type of bond formed is then added to the reconfiguration energy.

6. **Repeat process:** Do for every mapping for the given reaction, storing the final reconfiguration energy sum for each mapping.

As shown in Table C.1, this calculation is assisted with the use of subroutines `CHPMA_to_trackingvars`, `get_H_energy`, and `find_pair_in_array`. The energy algorithm consists of lines 513–872 in `chpma.f90`.

C.3 Summary

The source code of CHPMA first begins by calling the `pattern_matching` subroutine, the recursive pattern matching subroutine which uses `pattern_select_CC`, `pattern_select_OC`, and `pattern_selection`. For the energy calculation, three subroutines are used: `CHPMA_to_trackingvars`, `find_pair_in_array`, and `get_H_energy`. Finally, subroutine `write_Tr` is called to print the transfer probabilities in matrix form to a binary file.

APPENDIX D

USER GUIDE: PYTHON POST-PROCESSING SCRIPT AND USES

D.1 Introduction

The following guide assumes basic knowledge of Python, as well as exposure to the `pandas` and `matplotlib` Python modules. This guide will describe the file hierarchy of the post processing scripts, examine source code of key components of a Python module created specifically for tracking algorithm post processing, and provide a description of the scripts themselves.

D.2 Overview of script environment and folders

The following guide will continually refer to the folders `python/`, where the Python scripts are kept, the `data/` folder, where any exported plots or tables created by the scripts are sent, and the `cases/` folder, where the `casename/` folders are stored, which in turn each hold their own `trackername` output files. Both the `casename` and `trackername` naming convention are discussed in Subsection D.2.2. The `python/`, `data/`, and `cases/` folders are in the same root directory. Within the `python/`, there are scripts in the root, and three other folders `modules/`, `videoCodes/`, and `oldCodes/`.

D.2.1 List of scripts

List of python scripts in the root `python/` folder; note there is also one Bash script:

```
Fixedplots.py
Oxiplots.py
Pyroplots.py
Surrplots.py
blend_carbon.py
cleanupheaderssingle.sh
flux.py
frag.py
plotdiffbars.py
print_dependence.py
print_fixed_data.py
quickplot.py
strct.py
tracflux.py
trianglevideo.py
```

Contents of `modules/` folder, self-made python modules:

```
all_nC.py
define108reactions.py
```



```
define169reactions.py
definefullreactions.py
nCs.py
postprochelpers.py
```

Python bytecode .pyc files that would normally be in this directory have not been shown, as .pyc files compiled from .py files are machine-dependent. The modules will reappear on each respective machine when used from Python scripts in the root directory; Python does this automatically.

Contents of videoCodes/ folder: python scripts for creating animated visualizations of the tracking algorithm data:

```
efficientcolormapoxi.py
efficienttriangleplot.py
effxyzplot.py
makecorcolormapoxi.py
makecorcolormapoxibars.py
makegrayISOdodecanevideo.py
makegrayISOisooctanevideo.py
makegrayISOmxylenesvideo.py
maketrianglemapcolors.py
maketrianglemapcolorsoxi.py
maketriangleplot_alternative.py
xyzplot.py
```

The contents of the `oldCodes/` folder has not been listed because many of these scripts are poorly documented and contain deprecated methods. Many of the codes within the `oldCodes/` folder will not run with the current file organization.

D.2.2 Output file naming convention

The output files from previously run simulations have a universal naming convention, starting with either `profile_` for the chemistry output files, or `trac_` for the tracking output files, within each simulation. The output files are listed in Fig. A.1 of Appendix A. The notation used for all these files is `speciesname_atomsLabeled`. As two examples, if methylcyclohexane, MCH-C7H14 by mechanism name, is labeled at atom positions 2,3,4,5, and 6, as shown in Section A.3, the output files would have file names of `MCHC7H14_23456` or, in some directories, `MCH_23456`. The full name of the concentration output file would be `profile_MCHC7H14_23456_C.dat`, while the full name of the amounts of tracked atom in each species would be `trac_MCHC7H14_23456_det`.

It is recommended to run all of the post processing scripts within IPython [39], a Python shell, so that if errors arise, values of variables can be inspected within the IPython shell to see exactly what is happening in with the code.

D.3 Code descriptions and summaries

Select post-processing scripts are discussed in detail in this Section.

D.3.1 `postprochelpers.py`

The `postprochelpers` module is the back end for reading and cleaning the chemistry and tracking output files. All scripts in the root `python/` folder use this module and thus have the line `import postprochelpers as pp` in the imports section, near the top of every script. The most import and perhaps most simple definition, `def`, in `postprochelpers.py` is the `loaddata()` definition. Three variables are passed to this definition: the `casename`, `tracname`, `fileidentifier` to be read. The `loaddata()` source code is as follows:

```

def loaddata(casename, tracname, fileidentifier):

    if fileidentifier == 'X':
        filepath = '../cases/'+ casename + '/profile_'+ tracname + '_X.dat'
    if fileidentifier == 'Y':
        filepath = '../cases/'+ casename + '/profile_'+ tracname + '_Y.dat'
    if fileidentifier == 'C':
        filepath = '../cases/'+ casename + '/profile_'+ tracname + '_C.dat'
    if fileidentifier == 'det':
        filepath = '../cases/'+ casename + '/trac_'+ tracname + '_det'
    if fileidentifier == 'total':
        filepath = '../cases/'+ casename + '/trac_'+ tracname + '_total'
    if fileidentifier == 'gl':
        filepath = '../cases/'+ casename + '/trac_'+ tracname + '_gl'
    if fileidentifier == 'specp':
        filepath = '../cases/'+ casename + '/trac_'+ tracname + '_specp'

```

D.3.2 cleanupheaderssingle.sh

This Bash script removes the very first of the output file passed to it. It is a single line of Bash script. The entire contents of the file are as follows:

```

# remove first row of file, put it in tmpfile,
# move tmpfile back into originally named filepath
sed '1d'$1 > tmpfile; mv tmpfile $1

```

The first line of any output file created by the tracking algorithm code is a series of numbers representing the number of atoms able to be tracked in each species. This is the default format in which they are printed by the code. Removing this line makes the columns of any DataFrame to be the mechanism species names instead of a series of numbers. Bash was selected because it is faster with intricate in-file modification jobs than Python. This script is called from the `checkforheaders()` definition within the `postprochelpers` module, if numbers are indeed found in the first line of the passed in file.

D.3.3 `flux.py`

This script performs a sensitivity analysis on output `detpr` output files. It is simply a first-order integration of every species-reaction.

D.3.4 `trackflux.py`

This performs a tracking-based sensitivity analysis on output `tracdetpr` output files, nearly the same as the `flux.py` script, except the species-reaction rates are calculated by only using concentrations of labeled atoms in each species of each reaction. The rest of the equations follow as in the `flux.py` script.

D.3.5 `fixedplots.py`, `oxiplots.py`, `pyroplots.py`, `surrplots.py`

These scripts produce publication-worthy plots using the `matplotlib` Python module. This script has hardcoded case names at the top of each script for simplicity. These scripts are

straightforward in what they do: isolating single columns of data from the output files, and in some cases multiplying or dividing them by different values so that they fit on the axis range.

D.3.6 `blend_carbon.py`

This script plots the amount of toluene in all species in the mechanism versus the number. The `all_nC.py` python module is used to return the number of carbons in a species when given that species' name. Both the amount of toluene in each species, as well as the number of carbons in each species are stored in a `pandas DataFrame`. Then, the `pandas DataFrame`'s `sort()` function is used on the column of the `DataFrame` containing the number of carbons, such that the number of carbons are organized in ascending order. The amount of toluene in each species is then plotted versus the number of carbons in that species.

D.4 Additional scripts

There are a variety of additional scripts in the `oldCodes/` folder. Some of these are poorly documented or were written rapidly. Nearly all of the scripts in the `oldCodes/` folder will not run with the current folder configuration. However, they may be useful for taking snippets of code for post-processing tasks. With modification of relative file paths, they could be modified to work.

D.5 Summary

This guide has shown the overall file hierarchy and naming conventions used across the Python scripts. A few scripts have been discussed in detail, from plotting scripts to those that conduct analysis. The `pandas` Python module is frequently used, utilizing the `DataFrame` object type. The `matplotlib` Python module is also used extensively for a variety of plotting jobs. Finally, the `postprochelpers` module is always used in the post-processing scripts as it includes numerous useful definitions used across many of the scripts.

BIBLIOGRAPHY

- [1] Chevron 2014 annual report. Technical report, 2014.
- [2] A. K. Agarwal. Biofuels (alcohols and biodiesel) applications as fuels for internal combustion engines. *Prog. Energy Combust. Sci.*, 33(3):233–271, June 2007.
- [3] A. Demirbas. Importance of biodiesel as transportation fuel. *Energy Policy*, 35(9):4661–4670, September 2007.
- [4] A. Janssen, M. Jakob, M. Mäijther, and S. Pischinger. Tailor-made fuels from biomass – potential of biogenic fuels for reducing emissions. *MTZ Worldwide*, 71:54–60, December 2010.
- [5] G. Knothe, C. A. Sharp, and T. W. Ryan. Exhaust emissions of biodiesel, petrodiesel, neat methyl esters, and alkanes in a new technology engine †. *Energy Fuels*, 20(1):403–408, January 2006.
- [6] Y. Xu, I. Keresztes, A. M. Condo Jr., D. Phillips, P. Pepiot, and T. C. Avedisian. (in press). Spherically symmetric droplet burning characteristics of algae-derived renewable diesel, conventional #2 diesel, and their mixtures. *Fuel*, pages 1–12, November 2015.
- [7] M. E. Myers Jr, J. Stollsteimer, and A. M. Wims. Determination of gasoline octane numbers from chemical composition. *Ana. Chem.*, 47(13):2301–2304, 1975.
- [8] R. J. Gill and D. B. Olson. Estimation of Soot Thresholds for Fuel Mixtures. *Combust. Sci. Technol.*, 40(5-6):307–315, May 2007.
- [9] C. Mcenally and L. Pfefferle. Improved sooting tendency measurements for aromatic hydrocarbons and their implications for naphthalene formation pathways. *Combust. Flame*, 148(4):210–222, March 2007.

- [10] C. S. McEnally and L. D. Pfefferle. Sooting tendencies of nonvolatile aromatic hydrocarbons. *Proc. Combust. Inst.*, 32(1):673–679, 2009.
- [11] C. S. McEnally and L. D. Pfefferle. Sooting tendencies of oxygenated hydrocarbons in laboratory-scale flames. *Env. Sci. Tech.*, 45(6):2498–2503, March 2011.
- [12] Sidney W Benson and Jerry H Buss. Additivity Rules for the Estimation of Molecular Properties. Thermodynamic Properties. *The Journal of Chemical Physics*, 29(3):546–28, 1958.
- [13] Sidney William Benson. *Thermochemical kinetics*. Wiley, 1976.
- [14] Y. Xuan and G. Blanquart. Numerical modeling of sooting tendencies in a laminar co-flow diffusion flame. *Combust. Flame*, 160:1657–1666, 2013.
- [15] P. Pepiot, H. Pitsch, R. Malhotra, S. Kirby, and A. Boehman. Structural group analysis for soot reduction tendency of oxygenated fuels. *Combust. Flame*, 154(1-2):191–205, July 2008.
- [16] C. K. Westbrook, W. J. Pitz, and H. J. Curran. Chemical kinetic modeling study of the effects of oxygenated hydrocarbons on soot emissions from diesel engines †. *J. Phys. Chem. A*, 110(21):6912–6922, June 2006.
- [17] S. Dooley, S. H. Won, J. Heyne, T. I. Farouk, Y. Ju, F. L. Dryer, K. Kumar, X. Hui, C.-J. Sung, H. Wang, M. A. Oehlschlaeger, V. Iyer, S. Iyer, T. A. Litzinger, R. J. Santoro, T. Malewicki, and K. Brezinsky. The experimental evaluation of a methodology for surrogate fuel formulation to emulate gas phase combustion kinetic phenomena. *Combust. Flame*, 159(4):1444–1466, 2012.
- [18] K. G. Joback and R. C. Reid. Estimation of pure-component properties from group-contributions. *Chem. Eng. Commun.*, 57(1-6):233–243, May 2007.

- [19] T. Turányi. Sensitivity analysis of complex kinetic systems. tools and applications. *J. Math. Chem.*, 5(3):203–248, 1990.
- [20] A. E. Lutz, R. J. Kee, and J. A. Miller. *SENKIN: A Fortran program for predicting homogeneous gas phase chemical kinetics with sensitivity analysis*. 1988.
- [21] W. Sun, Z. Chen, X. Gou, and Y. Ju. A path flux analysis method for the reduction of detailed chemical kinetic mechanisms. *Combust. Flame*, 157(7):1298 – 1307, 2010.
- [22] E. M. Fisher, W. J. Pitz, C. K. Westbrook, and H. J. Curran. Detailed chemical kinetic mechanisms for combustion of oxygenated fuels. *Proc. Combust. Inst.*, pages 1–8, March 2001.
- [23] H. S. Homan and W. K. Robbins. A carbon-14 tracer study of the relative fractions of various fuel carbons in soot. *Combust. Flame*, pages 177–190, 2002.
- [24] B. A. Buchholz, C. J. Mueller, A. Upatnieks, G. C. Martin, W. J. Pitz, and C. K. Westbrook. Using Carbon-14 Isotope Tracing to Investigate Molecular Structure Effects of the Oxygenate Dibutyl Maleate on Soot Emissions from a DI Diesel Engine. pages 1–15, August 2005.
- [25] A. Eveleigh, N. Ladommatos, R. Balachandran, and A. Marca. Conversion of oxygenated and hydrocarbon molecules to particulate matter using stable isotopes as tracers. *Combust. Flame*, 161(11):2966–2974, November 2014.
- [26] V. A. Bunev. Tracer Method in Numerical Simulation of Combustion Processes. *Combust. Explos. Shock Waves*, pages 1–9, November 2007.
- [27] V. A. Bunev. Numerical Simulation of the Effect of the Addition of NO and NO₂ on a Rich Hydrogen Flame using the Tracer Method. *Combust. Explos. Shock Waves*, pages 1–7, May 2009.

- [28] I. Mayer. On bond orders and valences in the ab initio quantum chemical theory. *Int. J. Quantu. Chem.*, 29(1):73–84, 1986.
- [29] American Chemical Society. Cas registry. <https://www.cas.org/content/chemical-substances>.
- [30] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer. Description of several chemical structure file formats used by computer programs developed at molecular design limited. *J. Chem. Infor. and Mod.*, pages 1–12, 1992.
- [31] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. *Proc. of the third IAPR-TC-15 Int. Workshop Graph-Based Repr.*, pages 149–159, 2001.
- [32] J. G. Siek, L. Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. 2001.
- [33] G. Blanquart, P. Pepiot, and H. Pitsch. *Combust. Flame*, 156:588–607, 2009.
- [34] K. Narayanaswamy, P. Pepiot, and H. Pitsch. *Combust. Flame*, 157:1879–1898, 2010.
- [35] K. Narayanaswamy, P. Pepiot, and H. Pitsch. *Combust. Flame*, 162(4):1193–1213, 2015.
- [36] K. Narayanaswamy, P. Pepiot, and H. Pitsch. *Combust. Flame*, 161:866–884, 2014.
- [37] P. N. Brown, A. C. Hindmarsh, and G. D. Byrne. Dvode: Variable-coefficient ordinary differential equation solver with fixed-leading-coefficient implementation.
- [38] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz—Open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002.

- [39] B. E. Pérez, F. Granger. Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.