Stochastic Generator of Chemical Structure. 3. Reaction Network Generation.

Jean-Loup Faulon*

Computational Biology and Materials Technology Department, Sandia National Laboratories, Albuquerque NM, 87185-1111

Allen G. Sault

Catalytic and Porous Materials Department, Sandia National Laboratories, Albuquerque NM, 87185-1349

Abstract

A new method to generate chemical reaction network is proposed. The particularity of the method is that network generation and mechanism reduction are performed simultaneously using sampling techniques. Our method is tested for hydrocarbon thermal cracking. Results and theoretical arguments demontrate that our method scales in polynomial time while other deterministic network generators scale in exponential time. This finding offers the possibility to investigate complex reacting systems such as those studied in petroleum refining and combustion.

*jfaulon@cs.sandia.gov

DISCLAIMER

I. INTRODUCTION

Motivated by applications in synthesis design, combustion, and petrochemical refining, reaction network generation has been a prolific field of research for the past 25 years. As reviewed in Ugi *et al.* paper,[1] reaction network generation is based on three types of techniques. Empirical techniques, which strategies are based on data from reaction libraries, semiformal methods based on heuristic algorithms, where reactions are derived from a few mechanistic elementary reactions, and formal techniques, based on graph theory. Ugi *et al.* argue that formal techniques are general enough to be applicable to any type of reacting system in organic or inorganic chemistry, furthermore formal techniques are the only methods capable of generating new reaction mechanisms, and therefore elucidating unresolved chemical processes.

While formal techniques are robust their computational scaling limits their applicability to real reacting systems. Indeed as it has been shown by several authors[1–9], for many processes, the number of reactions and intermediate species that are created generally scale exponentially with the number of atoms of the reactants. Two views of approaching this problem have been proposed. One is to wait until sufficient computational power becomes available. However, the exponential scaling of formal techniques makes it improbable that we will ever reach enough computing power to solve the problem. The other view, which is considered in the present paper, is to reduce the reaction mechanism. The question raised when reducing mechanism is how to choose a reaction subset such that it describes correctly the dynamic behavior of the studied reacting system. Reduction strategies in the area of combustion modeling have been reviewed by Frenklach,[5] these are quite general and applicable to other systems. According to Frenklach there are five types of reduction strategies: 1) global reduction, 2) response modeling, 3) chemical lumping, 4) statistical lumping, and 5) detailed reduction. Global modeling techniques transform a complete reaction scheme into a small number of global reaction steps. Global techniques comprise ad-hoc methods such as empirical fitting, reduction by approximations, and lumping. All global techniques are

specific to a particular problem and cannot be generalized. Response modeling techniques consist of mapping model responses and model variables through functional relationships. Generally, models responses are species concentrations and model variables are the initial boundary conditions of the reacting mixture and the model parameters, such as rate coefficients, and transport properties. The solution mapping method provides a general procedure to solve model response. In this method, model responses are expressed as simple algebraic functions (usually polynomials) in terms of model variables. These algebraic functions are obtained by using either computer experiments or experimental data. As with global techniques, response modeling solutions are problem specific since they require data to build algebraic functions. Chemical lumping and statistical lumping were both developed for polymerization-type reactions. Chemical lumping models are used when a polymer grows by reaction between the polymer and monomer species. The lumping strategy is guided by similarity in chemical structure or chemical reactivity of the reacting species. The main assumption of chemical lumping is that the reactions describing the polymer growth are essentially the same and the associated thermochemical and rate parameters exhibit only a weak dependence on the polymer size. Statistical lumping is used when polymer-polymer interactions are of concern, such as in soot formation, silica powder growth, and metal oxide growth. Such processes are described by the Smoluchowski equation. The integration of the Smoluchowki equation encounters severe prohibitive demands on computer time and memory and statistical approximation of the equation have been proposed. Both chemical and statistical lumping methods are specific to polymerization reactions. Finally, the detailed reduction technique consists of identifying and removing noncontributing reactions. An effective reduction strategy is to compare the individual reaction rates with the rate of a chosen reference reaction. The reference reaction being for instance the rate limiting step or the fastest reaction. The detailed reaction reduction approach is a general technique and is a priori applicable to any reacting system.

The goal of this paper is to propose an original technique for reaction network generation that is computationally tractable and general enough to be applicable to real processes,

such as combustion, petroleum refining, and retrosynthesis. To achieve such a goal, formal generation techniques must be used since with the above chemical processes not all reactions are known and there is a need to create new reaction mechanisms. However, as mentioned earlier formal techniques scale exponentially with the problem size and therefore reduction methods need to be applied. As discussed earlier, the only reduction technique applicable to general systems is the detailed reduction method. The caveat of the method however, is that the entire network must be known prior to reduction. Therefore, network generation and reduction cannot be processed in sequence if our goal is to derive a computationally tractable technique.

In section II, we outline a new method where network generation and reduction are performed simultaneously. More precisely we give four algorithms: deterministic-network-generator (DNG), random-sampling-network-generator ($RSNG$), concentration-sampling-network-generator ($CSNG$) and MC-sampling-network-generator ($MCNG$). While the first algorithm is deterministic and similar to techniques previously reported, the three sampling algorithms are stochastic in nature and appear to be the first efficient (i.e., polynomial-time) methods ever published for reaction network generation. Susnow *et al.*[8] propose a technique similar to $CSNG$ using rates instead of concentrations for mechanism reduction. However, the computational scaling of Susnow *et al.* technique has not been reported, furthermore, it is improbable that their algorithm scales in polynomial time since their technique does not explicitely limit the number of species and reactions allowed in the network. Our $CSNG$, and $MCNG$, algorithms require to compute on-the-fly the rate constants of the reaction generated. In section III, we present a Quantitative Structure Property Relationship (QSPR) approach to compute rate constants at low computational cost. In section IV, we theoretically prove that our sampling generation-reduction algorithms no longer scales exponentially with the problem size but scales polynomially. Finally, in section V, in the context hydrocarbon thermal cracking, we test our sampling algorithms versus experimental results and deterministic algorithms. We also compare the running-time of our algorithms with the theoretical computational complexity calculations of section IV.

II. METHOD

The proposed network generator is based on the Dugundji-Ugi theory.[10] According to this theory compounds and reaction are represented by bond electron ($be-$) and reaction ($r-$) matrices. In a given be-matrix representing a compound the ith row (and column) is assigned the ith atom of the compound. The entry b_{ij} (b_{ji}, $i \neq j$) of the ith row and jth column of a $be-$matrix is the bond order of the covalent bond between atoms i and j. The diagonal entry b_{ij} is the number of free valence electron for atom i. The reader may noticed that the adjacency and connectivity matrices used in chemical information differ from the $be-$matrices in their diagonal entries. The redistribution of valence electrons by which the starting materials of chemical reactions are converted into their products are represented by $r-$matrices. The fundamental equation of the Dugundji-Ugi model for any reaction $\alpha_1 + \alpha_2 + ..\alpha_n \rightarrow \beta_1 + \beta_2 + ..\beta_m$ is $B + R = E$, where B and E are the $be-$matrices of reactants and products, and R is the reaction matrix. The addition of matrices proceeds entry by entry, that is, $b_{ij} + r_{ij} = e_{ij}$. Since there are no formal negative bond orders or negative numbers of valence electrons, the negative entries of R must be matched by positive entries of B of at least equal values. Within the above restrictions, Dugundji and Ugi proved that $be-$ and $r-$ matrices forms a free additive abelian group.[11] There are two types of formal reaction generators, RGB and RGR. RGB generators solve the fundamental equation for a given B, while RGR generator solve the equation for a given R. Additionally, constraints can be added while solving either RGB or RGR equations. Example of constraints are maximum number of species and reactions generated, maximum species size, maximum number of reactant per reactions, maximum number of lone electron, etc...

The generator proposed in this paper is of type RGB. Consequently, the generator builds all the products that can be generated from a set of reactants, and a set of constraints. The current version of our generator is limited to monomolecular and bimolecular reactions but could easily be extended to more complex reactions. Following Dugundji-Ugi methodology reactants are represented by $be-$matrices. The constraints are entered by the user from the

list given in Table I. The studied chemical process is represented into the form of elementary transitions, which are stored in a library of $r-$matrices. Elementary transitions are the electronic transitions that atoms of the reacting system can undergo. Each elementary transition is composed of two configurations containing all the atoms participating in a reaction before and after the reaction has taken place. Examples of elementary transitions and associated $r-$matrices are given in Figure 1. Fontain and Reitsan[12] have compiled the complete set of elementary transitions that carbon atoms can take in organic reactions. This set is composed of 324 transitions, however the set can be greatly reduced depending on the studied process. For instance, it well known[13] that all hydrocarbon thermal cracking reactions can be generated from the five elementary transitions listed in Figure 1. As another example Susnow *et al.*[8] are using 17 transitions to model methane combustion. The algorithms of our generator are given next and are tested for thermal cracking of hydrocarbons in section III.

A. Deterministic algorithm (DNG)

The input of our reaction network generator is a list of reactant species, a list of constraints, and a list of elementary transitions. The output is a network composed of all possible species and reactions that can be created from the input. The algorithm described below is illustrated for ethane thermal cracking in Figure 2. The algorithm starts by computing all the possible species (Ls_1) that can be derived by applying the elementary transitions (Let) to the initial species (Ls_0) while respecting the constraints. To prohibit duplication of species, Ls_1 is composed of species that are not already present in Ls_0. When applying elementary transitions, one has to make the distinction between monomolecular and bimolecular reactions. For each monomolecular elementary transition, Ls_1 is composed of all the possible products that can be generated by applying the elementary transition in all possible ways to the species of Ls_0. For each bimolecular reaction, Ls_1 is composed of the products derived by applying the elementary transitions to all possible pairs of species

in Ls_0. The list of reactions Lr_1 is updated each time an elementary transition applies to a species in Ls_0. Each reaction is represented by a n-tuples composed of the elementary transition (from Let), the reacting species (from Ls_0), and the product species (from Ls_1 or Ls_0). Once the list Ls_1 has been computed the algorithm proceeds and compute recursively $Ls_2, Ls_3, \ldots, Ls_i$. Note that in order to compute Ls_i $(i > 1)$ one has to consider monomolecular reactions only from the set Ls_{i-1} since all monomolecular reactions from the sets Ls_{i-2}, Ls_{i-3},... have already been computed in the previous steps. For the same reason, the products of bimolecular reactions in Ls_i are generated for every possible pairs of species having at least one element in Ls_{i-1}. To avoid redundancies of species in the list $Ls_0, Ls_1, \ldots, Ls_i$, at any step $i > 1$ a new species is added to Ls_i only if the species is not in $Ls_{i-1} \cup Ls_{i-2} \cup \ldots \cup Ls_0$. The process halts at any step i, where the corresponding lists of species Ls_i and reaction Lr_i are empty. Since bimolecular reactions can potentially create products of infinite molecular weight, in order to keep a finite network size one has to set a limit for the maximum species size, let n be this limit. Note that with the exception of polymerization reaction it is reasonable to assume that all species in a given network will be limited in size. In turn, note that if the species have a limited size then the network generation algorithm is guaranteed to converge. Indeed, the maximum number of species, N, that can be formed is bounded by the total number of molecular structures having a number of atoms raging from 1 to the specified value of n. This latter number is equal to the sum of the numbers of isomers containing $1, 2, \ldots, n$ atoms. Finally, note that the number of reactions is also finite and is bounded by N^2 the maximum number of edges in the network (i.e., each species reacts with all the others). The network generator algorithm given in Scheme I uses specific data structures to describe species and reactions. A molecular species s is represented by a molecular graph $G(s)$. This graph is in turn represented by a $be-$matrix. An elementary transition (et) is represented by two molecular graphs, $G_r(et)$ the electronic configurations of the surrounding atoms before the reaction has taken place, and $G_p(et)$ the electronic configuration after the reaction has taken place. From these molecular graphs one constructs a $r-$matrix which is the difference between the two configurations (i.e. $r-$matrix

corresponding to $G_p(et) - G_r(et)$). Finally, a reaction, r, is an n-tuple composed of a elementary transition (et), a list of reactant species ($L_r(r)$), a list of product species ($L_p(r)$), and a rate constant ($k(r)$). Additionally, the algorithm maintains several lists of species and reactions. The list of reactants (Ls_0), the list of species (Ls_i) and reactions (Lr_i) created at the current step i, and the list of species generated at the previous step (Ls_{i-1}). The operator $\uplus$ adds a new species to any list Ls_i if and only if the species is not already present in Ls_i. Since species are represented by molecular graphs the operator $\uplus$ performs a series of graph isomorphism checks between the species to be inserted and the species already in the list. The molecular graph isomorphism routine used in this study has been published elsewhere.[14] The routine's computational scaling is $O(n^2)$, where n is the number of atoms of the species. The following scheme gives the deterministic network generator routines. The functions `species-constraints` and `rate-constant` are rather trivial and not detailed in scheme I. The function `species-constraints` verifies the constraints listed in Table I, the function returns FALSE if any of the constraint is not verified and return TRUE otherwise. The function `rate-constant` computes the rate constant of a given reaction using a technique described in section III.

Scheme I:

`deterministic-network-generator`(Ls_0, Let, Ls, Lr)

`input:` -Ls_0: `list of initial species`

-Let: `list of elementary transitions`

`output:` -Ls: `list of all species in network`

-Lr: `list of all reactions in network`

`local:` -Ls_i: `list of species created at step` i

-Ls_{i-1}: `list of species created at`

step $i-1$

-Lr_i: list of reactions created

at step i

begin

1. $Ls := Ls_0$; $Lr := \emptyset$; $Ls_{i-1} = Ls_0$;
2. do
4. $Ls_i := \emptyset$; $Lr_i = \emptyset$;
5. generate-species-reactions

$(Ls_{i-1}, Ls, Let, Ls_i, Lr_i)$

6. $Ls := Ls \uplus Ls_i$; $Lr := Lr \uplus Lr_i$;
7. $Ls_{i-1} := Ls_i$;
8. until ($Ls_i = \emptyset$ and $Lr_i = \emptyset$)

end

generate-species-reactions$(Ls_1, Ls_2, Let, Ls_i, Lr_i)$

input: -Ls_1, Ls_2: list of species

-Let: list of elementary transitions

output: -Ls_i: list of species created

at step i

-Lr_i: list of reactions created

at step i

local: -L_{G_p}: list of graphs returned

by generate-product

begin

1. For all species s in Ls_1 do
2. For all reactions $et \in Let$

with $order(et) = 1$ do

3. $L_{G_p} := \emptyset$;

4. generate-product

$(G(s), G_r(et), G_p(et), L_{G_p})$;

5. update-species-reactions

$(L_{G_p}, er, s, \emptyset, Ls_i, Lr_i)$;

6. done

7. done

8. For all species $s_1 \in Ls_1$ and $s_2 \in Ls_2$ do

9. For all reactions $et \in Let$

with $order(et) = 2$ do

10. $L_{G_p} := \emptyset$;

11. generate-product

$(G(s_1) \cup G(s_2), G_r(et), G_p(et), L_{G_p})$;

12. update-species-reactions

$(L_{G_p}, er, s_1, s_2, Ls_i, Lr_i)$;

13. done

14. done

end

generate-product$(G(s), G_r(et), G_p(et), L_{G_p})$

input: -$G(s)$: molecular graph of species s

-$G_r(et)$: molecular graph

of the reactants of rection et

-$G_p(et)$: molecular graph

of the products of reaction et

output: -L_{G_p}: list of graphs obtained

after applying et to s

local: -X: a graph

-G': a subgraph of $G(s)$

begin

1. For all $G' \subseteq G(s)$ s. t. $G' \equiv G_r(et)$ do

2. $X := G(s) - G' + G_p(et)$;

3. if (species-constraints(X) = TRUE)

then $L_{G_p} := L_{G_p} \uplus X$ fi;

4. done

end

update-species-reactions($L_{G_p}, et, s_1, s_2, Ls_i, Lr_i$)

input: -L_{G_p}: list of graphs returned

by generate-product

-et: elementary transition

-s_1, s_2: reactant species

output: -Ls_i: list of species created

at step i

-Lr_i: list of reactions created

at step i

local: -L_p: list of connected graphs

-k: rate constant

begin

1. For all graphs $G_p \in L_{G_p}$ do

2. compute L_p the list of

connected components of G_p;

3. $Ls_i := Ls_i \uplus L_p$;

4. k := rate-constant(et, s_1, s_2, L_p);

5. $Lr_i := Lr_i \uplus (et, s_1, s_2, L_p, k)$;

6. done

end

B. Sampling algorithms

Although the size of the network generated by the deterministic algorithm is finite due the limited size of the species, as we already mentioned the number species in the network grows exponentially with the with n, the number of atoms of the largest species. In fact, as it will be seen in the last section, it is virtually impossible to generated thermal cracking networks for species larger than heptane. As argued in the introduction there are several techniques to reduce the network size. We propose in this section three network sampling algorithms based on the detailed reduction idea: random-sampling, concentration-sampling, and Monte-Carlo sampling.

(i) Random-sampling algorithm (RSNG). The algorithm is identical to the deterministic network generator with the exception that at each step i the resulting species list Ls is reduced to a size equal to a pre-defined number, M_s. The reduction is done at random, that is, species are removed at random from Ls until the size of Ls is below M_s. The algorithm is given in Scheme II. The subroutine `generate-species-reactions` is given in scheme I.

Scheme II:

random-sampling-network-generator(Ls_0, Les, Ls, Lr)

input: -Ls_0: list of initial species

-Let: list of elementary transitions

output: -Ls: list of all species in network

-Lr: list of all reactions in network

local: -Ls_i: list of species created at step i

-Ls_{i-1}: list of species created at step $i-1$

-Lr_i: list of reactions created at step i

begin

1. $Ls := Ls_0$; $Lr := \emptyset$; $Ls_{i-1} = Ls_0$;
2. do forever
4. $Ls_i := \emptyset$; $Lr_i = \emptyset$;
5. generate-species-reactions $(Ls_{i-1}, Ls, Let, Ls_i, Lr_i)$;

if ($Ls_i = \emptyset$ and $Lr_i = \emptyset$) then end fi

6. $Ls := Ls \uplus Ls_i$; $Lr := Lr \uplus Lr_i$;
7. reduce-mechanism-random(Ls, Ls_i);
8. $Ls_{i-1} := Ls_i$;
9. done

end

reduce-mechanism-random(Ls, Ls_i)

input: -Ls: list of species

-Ls_i: list of species created at step i

output: -Ls: reduced list of species

-Ls_i: reduced list of species

created at step i

const.: -M_s: maximum number of species
allowed in Ls

begin

1. While ($|Ls| > M_s$) do

2. select s in Ls at random;

3. $Ls := Ls - s$; $Ls_i := Ls_i - s$;

4. done

end

(ii) Concentration-sampling algorithm (CSNG). With the concentration-sampling algorithm, the reduction of the network is not performed at random but based on the species concentrations. Futhermore, at each step i species are removed from Ls_i and not Ls as with the previous algorithm. Hence, in the present case, the total number of network species is not limited, but the number of species generated at each step is bounded by the predefined value M_s. The main assumption of this method is that if a species created at any given step i has low concentration values over the reaction time, this species will generate products with concentrations at most equal to that low value. Therefore, removing low concentration species from the list Ls_i should have a neglectable effect on the final product distribution. Note that this assumption is valid only if the species are consumed during the reactions and do not act as catalysts. Indeed if a species is a catalyst even with low concentration, this species could have a relatively large impact on the final product distribution. Hence, catalytic species must be identified prior using this scheme. Note that for thermal cracking reactions all the species participating to the reactions are consummed (cf. Figure 1), and therefore, there is no need to identify catalytic species. Using the above concentration assumption, the algorithm works as follows. At each step i the deterministic routine

`generate-species-reaction` is run to compute the list of new species. At this point the network is composed of all species in $Ls_0 \cup \ldots \cup Ls_i$ and associated reactions. Although the network may not be completed, since the reaction rates are calculated on-the-fly (cf. section III), the time evolution of the species concentrations is computed by solving the system of differential equations associated with the partial network. In the present algorithm, we use the Monte-Carlo Gillespie[15] (MC-Gillespie) technique to solve the system. The MC-Gillespie technique monitors the number of particles for each species versus time. The initial number of particles of the reactants are computed from their initial concentrations (given by the user), and the initial number M_p of particule in the system. In the present paper, we are using the MC-Gillespie technique at constant volume V, hence the initial number of particles is calculated from the particle density, and the reaction volume (both user inputs). The MC-Gillespie technique is an exact method for numerical integration of the time evolution of any spatially homogeneous mixture of molecular species that interact through a specified set of coupled chemical reaction channels. The technique is based on a fundamental equation giving the probability at time t that the next reaction in V will occur in the differential time interval $[t+\tau, t+\tau+d\tau]$ and will be an r_μ reaction:

$$P(\tau,\mu)d\tau = P_1(\tau)\ P_2(\mu/\tau) \tag{1}$$

where

$$P_1(\tau) = ae^{-a\tau}d\tau \tag{2}$$

and

$$P_2(\mu/\tau) = a_\mu/a \tag{3}$$

with

$$a = \sum_\mu \mu a_\mu \tag{4}$$

In eqs 2-4, $a_\mu d\tau$ is the probability, to first order in $d\tau$, that a reaction r_μ will occur in V in the next time interval $d\tau$. The rate constant k_μ is related to a_μ in the different ways depending if the reaction is monomolecular or bimolecular. For monomolecular reactions:

$$a_\mu = [s]k_\mu \tag{5}$$

where $[s]$ is the concentration of species s, which, in the present case is simply the number of particle s. For bimolecular reaction involving two species s_1 and s_2, we have:

$$a_\mu = [s_1][s_2]k_\mu/V \tag{6}$$

and for bimolecular reaction involving only one reactant species:

$$a_\mu = [s]([s]-1)k_\mu/V \tag{7}$$

In order to integrate eq. 1, Gillespie proposes the following scheme. First generate a random value τ according to $P_1(\tau)$ and second generate a random integer μ according to $P_2(\mu/\tau)$. In our implementation of the MC-Gillespie technique, the random value τ is computed by simply drawing a random number r_1 from an uniform distribution in the unit interval and taking:

$$\tau = (1/a)ln(1/r_1) \tag{8}$$

In turn, the random integer μ is generated by drawing another random number r_2 in the unit interval and by taking μ the integer verifying:

$$\sum_{\nu=1}^{\mu-1} a_\nu \leq r_2 a \leq \sum_{\nu=1}^{\mu} a_\nu \tag{9}$$

In his original paper,[15] Gillespie has proven that expressions 8 and 9 are indeed correct to simulate stochastically the time evolution of homogenous reactions. During the MC-Gillespie integration, the algorithm retains for each species present in the network the maximum concentration (i.e., number of particles) reached over the time period the system is integrated. This operation requires to maintain two lists, $L[s]$, the list of species concentrations, and $L[s_{max}]$ the list of species maximum concentration. Species are then sorted by decreasing concentration, the first M_s elements of the sorted list are kept in Ls while the other are eliminated. The algorithm is given in the following scheme, the routine `generate-species-reaction` is given in scheme I. The routine

assign-initial-number-particle is not detailed here, but returns the initial number of particles for each reactants. This number is simply equal to the product of the maximum number of particles (M_p) by the initial concentration of the reactant. Both numbers are user's input.

Scheme III:

concentration-sampling-network-

generator(Ls_0, Let, Ls, Lr)

input: -Ls_0: list of initial species

-Let: list of elementary transitions

output: -Ls: list of all species in network

-Lr: list of all reactions in network

local: -Ls_i: list of species created at step i

-Ls_{i-1}: list of species created at step $i-1$

-Lr_i: list of reactions created at step i

begin

1. $Ls := Ls_0$; $Lr := \emptyset$; $Ls_{i-1} = Ls_0$;

2. do forever

4. $Ls_i := \emptyset$; $Lr_i = \emptyset$;

5. generate-species-reactions

$(Ls_{i-1}, Ls, Let, Ls_i, Lr_i)$;

if ($Ls_i = \emptyset$ and $Lr_i = \emptyset$) then end fi

6. $Ls := Ls \uplus Ls_i$; $Lr := Lr \uplus Lr_i$;

```
7.      reduce-mechanism-concentration(Ls, Ls_i);
8.      Ls_{i-1} := Ls_i;
9.  done
end

reduce-mechanism-concentration(Ls, Ls_i)
input:  -Ls:  list of species
        -Ls_i:  list of species
         created at step i
output: -Ls:  reduced list of species
        -Ls_i:  reduced list of species
         created at step i
local:  -L[s]:  list of species concentration
        -L[s_max]:  list of species
         maximum concentration
        -n_c:  number of MC steps
        -t:  time
const.: -M_p:  maximum number particles
        -M_c:  maximum number of MC steps
begin
1.  L[s] := ∅;  L[s_max] := ∅;
1.  For all species s ∈ Ls do
2.      if step(s) = 0 then
2.          [s]  := assign-initial-number
                        -particles(s, M_p);
2.          [s_max] := [s];
2.      else
```

2. $[s] := 0;\ [s_{max}] := 0;$

2. fi

2. $L[s] := L[s] \cup [s];$

2. $L[s_{max}] := L[s_{max}] \cup [s_{max}];$

2. done

2. $t := 0;\ n_c := 0;$

2. While $n_c < M_c$ do

2. $t :=$MC-Gillespie-step$(Ls, L[s], Lr, t)$;

2. For all species $s \in Ls$ do

2. if $[s] > [s_{max}]$ then $[s_{max}] = [s]$ fi

2. done

2. $n_c := n_c + 1;$

2. done

2. While $(|Ls_i| > M_s)$ do

2. find $s \in Ls_i$

having the lowest $[s_{max}]$ value

2. $Ls := Ls - s;\ Ls_i := Ls_i - s;$

2. done

end

MC-Gillespie-step$(Ls, L[s], Lr, t)$

input: -Ls: list of species

-$L[s]$: list of species concentration

-Lr: list of reactions

-t: time

output: -$L[s]$: updated list of species concentration

-t: time after event occurs

begin

1. compute time τ of next event using eq.8;
2. select reaction r in Lr occurring
   at time $t+\tau$ using eq.9;
2. $t := t+\tau$;
2. For all $s \in L_r(r)$ do $[s] := [s]-1$ done;
2. For all $s \in L_p(r)$ do $[s] := [s]+1$ done;
2. return t;

end

(iii) MC-sampling algorithm (MCNG). The idea of the MC-sampling algorithm is to perform at the same time both the Monte-Carlo integration and the network generation. The advantage of this technique is that there are no assumptions regarding catalyst species. As in the previous case, one starts with an initial reactant concentration given in the form of numbers of particles. The initial total number of particles (M_p) is calculated from the particle density and reaction volume. At each step, the set of new species is computed using the generate-species-reactions routine, but in the present case these species are generated by applying the elementary transitions only for species having non-zero concentration (i.e., set Ls^* in Scheme IV). The concentrations of the new species are set to zero and updated using the MC-Gillespie integration step. The process is iterated until the number of step exceed the pre-defined M_c value. In the following scheme, the routine `generate-species-reactions` is given in scheme I, and the routine `MC-Gillespie-step` can be found in scheme III.

Scheme IV:

`MC-sampling-network-generator`(Ls_0, Les, Ls, Lr)

input: -Ls_0: list of initial species

-Let: list of elementary transitions

output: -Ls: list of all species in network

-Lr: list of all reactions in network

local: -Ls_i: list of species created at step i

-Lr_i: list of reactions created at step i

-$L[s]$: list of species concentration

-Ls^*: list of species with non-zero concentration

-t: time

const.: -M_p: maximum number particles

-M_c: maximum number of MC steps

begin

1. $Ls := Ls_0$; $L[s] := \emptyset$; $Lr = \emptyset$;
2. For all species $s \in Ls$ do
3. $[s]$:= assign-initial-number -particles(s, M_p);
4. $L[s]$:= $L[s] \cup [s]$;
5. done
6. $t = 0$; $i = 0$;
7. While $i < M_c$ do
8. $Ls_i := \emptyset$; $Lr_i := \emptyset$; $Ls^* := \emptyset$, $i := i + 1$;
9. For all $s \in Ls$ do
10. if $[s] = 0$ then $Ls^* := Ls^* - s$ fi
11. done

```
12.        generate-species-reactions
                        (Ls*, Ls*, Les, i, Ls_i, Lr_i);
13.        Ls := Ls ⊎ Ls_i;  Lr := Lr ⊎ Lr_i;
14.        For all s ∈ Ls_i do [s] := 0 done
15.        t := MC-Gillespie-step(Ls, L[s], Lr, t);
16. done
end
```

III. RATE CALCULATIONS

Rate constants are calculated for each new reaction generated in the routine generate-species-reactions. Because generate-species-reactions is a basic routine called by of all the network generation algorithms, we have implemented a fast techniques to compute the rates. Rate constants are estimated for each reaction using a quantitative structure property relationship (QSPR) based on the reaction type taken for the elementary transitions listed in Table I, and on the Wiener indices of the reactants and products. The Wiener index is simply the sum of the number of bonds between all pairs of atoms in a given molecule or radical. Thus, H_2 has a Wiener coefficient of 1, the methyl radical has a Wiener coefficient of 9, methane has a wiener coefficient of 16, etc. For large ($> C_4$) hydrocarbon species that exhibit multiple isomeric forms, the Wiener coefficient is lowest for highly branched isomers, and largest for the linear isomer. Thus, the Wiener coefficient reflects the extent of branching of a given C_xH_{2x+1} radical stoichiometry. Since the stability of hydrocarbon radicals decreases with branching[16] in the order $R_3C > R_2CH > RCH_2 > CH_3$, one would expect the reactivity of the radicals to increase in the same order. Putting these two facts together leads to the conclusion that the Wiener coefficient should correlate with radical activity, with higher Wiener coefficients indicating higher reactivity for a given C_xH_{2x+1} radical composition.

The QSPR used here is based on experimental kinetic data taken from Allara and Shaw[17] for each of the reactions given in Figure 1. For a given reaction type, the preexponential factor is taken as the average of the measured preexponential factors for each experiment reported in Allara and Shaw. The activation energy is fitted to the expression:

$$E_a = E_0 + \alpha \Sigma W_r + \beta \Sigma W_p \tag{10}$$

where E_0, α, and β are fitting parameters and ΣW_r and ΣW_p are the sums of the Wiener coefficients of the reactants and products, respectively. The form of eq. 10 was chosen rather than the more general form:

$$E_a = E_0 + \sum_{i=1}^{R} \alpha_i W_{r_i} + \sum_{i=1}^{P} \beta_i W_{p_i} \tag{11}$$

where the α_i's and β_i's are now individual coefficients for each reactant or product. This choice was made to avoid the necessity of sorting the reactants and products to ensure that the appropriate coefficient is applied. For example, a hydrogen transfer reaction where reactant 1 is an alkane and reactant 2 is a radical would require that the algorithm differentiate between the radical and the alkane to make sure that W_1 is multiplied by α_1 rather than α_2. A similar consideration applies to the products. The computational cost of performing this sorting was deemed unjustified given the uncertainties in the available experimental data and the resulting unavoidable uncertainties in the QSPR estimations (see below).

Results from fitting eq. 10 to each reaction are given in Table III. Note that for two of the reactions (addition and hydrogen abstraction) the data is divided into multiple sets rather than being fit as a whole. This division is based on inspection of the data in Allara and Shaw, which reveals that for these two reactions the rate preexponential factors and/or activation energies clearly vary with reactant type. For example, the kinetic parameters of hydrogen abstraction from H_2 clearly differ from those for hydrogen abstraction from an alkane. Consequently Table III lists eight different reactions vs. only five in Table II.

Figure 3 shows a plot of the predicted rate constants for each reaction vs. the measured rate constants. In general, the predicted rate constants are within about a half an

order of magnitude of the measured values. Unfortunately, the available experimental data also displays variations of this magnitude, so better accuracy cannot be expected from the QSPR. It is nevertheless interesting to note that for the addition reactions and hydrogen transfer from alkanes to alkyl radicals, Figure 3 shows very little variation in the predicted rate constants. This result suggests that the Wiener coefficient is not a good structural parameter for developing a QSPR for these reactions, and that other approaches will need to be considered to obtain better rate constant estimates. Despite these limitations, the QSPR presented here is still accurate enough to demonstrate the performance and scaling of our algorithm, and allow for qualitative predictions of product distributions.

IV. THEORETICAL COMPUTATIONAL COMPLEXITY

In this section we analyse the upper bound theoretical time-complexity of the above algorithms, and prove that the three sampling algorithms can be run in polynomial time. Theoretical complexity results are summarized in Tables III and IV. The reader should refer to the Glossary for the notations used in this section. Practicle results, i.e., running time complexity are given in the next section.

For hydrocarbon cracking reactions, the maximum number of atoms for each elementary transition is $r \leq 3$ and the number of elementary transitions is $R = 8$ (cf. Table I). As a general rule r and R are always bounded and will be taken as constant in our complexity calculations.

We first evaluate the time-complexity of the procedure generate-product. The procedure performs a subgraph isomorphism between a given species and an elementary transition. Since each elementary transition contains at most r atoms, there are $r!$ ways of mapping $G_r(et)$ onto each atom of $G(s)$. $G(s)$ comprises at most n atoms, therefore, the number of subgraph isomorphism performed is $nr!$. Let us assume that all tries pass the subgraph isomorphism and the constraint tests. In such an instance, $nr!$ graphs are added to the list L_{G_p}, and consequently, $|L_{G_p}| \leq nr! = \mathrm{O}(n)$. Each graph added to L_{G_p} is tested for

isomorphism with a complexity of $O(n^2)$. The constraints listed in Table I can be checked in $O(n)$ steps, consequently the time-complexity of the generate-product routine is $nr!(n+n^2)$ $= O(n^3)$. Furthermore, the number of graphs returned by generate-product is $|L_{G_p}| \leq nr!$ $= O(n)$.

Next, we evaluate the time-complexity of update-species-reactions. As seens above, generate-product may return $nr!$ graphs, these graphs in turn may contain at most n connected components (i.e., $|L_p| = n$). Therefore, the maximum number of species and reaction that can be created by update-species-reactions is $|L_{G_p}||L_p| \leq n^2 r! = O(n^2)$. Note that we choose an upper bound for $|Lr_i|$ equal to the upper bound for $|Ls_i|$, since every reaction in Lr_i contains one element of Ls_i, and for every element of Ls_i there exist a reaction in Lr_i. Note however that we do not necessarly have $|Ls_i| = |Lr_i|$ due to the isomorphism checks performed when inserting elements in Ls_i and Lr_i. The routine update-species-reactions first compute all the connected components of the graphs of L_{G_p}. It is well known that the connected components of a graph can be found in $O(n)$ steps.[18] The routine then checks isomorphism for all species and reactions created and compute the rate constants of all new reactions. Isomorphism between species can be achieved in $O(n^2)$, isomorphism between reactions can be performed in a constant time (i.e., $O(1)$) since reactions are represented by n-tuple. As seen in the previous section, rate constants are calculated using the Wiener indices of the reactants and products of the reactions. Calculating a Wiener index requires $O(n^2)$ steps for a species containing n atoms.[19] Thus, the overall time-complexity for update-species-reactions is $|L_{G_p}||L_p|(O(n) + O(n^2) + O(1) + O(n^2)) = O(n^4)$.

The routine generate-species-reaction distinguishes monomolecular reactions from bimolecular reactions. For monomolecular reactions all species in $Ls_1 = Ls_{i-1}$ and all elementary transitions Let are sent to generate-product and update-species-reaction. As computed above, the maximum number of species and reactions returned by update-species-reaction is $n^2 r!$. Therefore, the number of species created through monomolecular reactions and returned by generate-species-reaction is bounded by $|Ls_{i-1}||Let||L_{G_p}||L_p| = N_{i-1} R n^2 r!$. For bimolecular reactions the same operations are performed for all pairs of species between Ls_{i-1}

and $Ls_{0,i-1}$, and the number of species created is maximized by $|Ls_{0,i-1}||Ls_{i-1}||Let||L_{G_p}||L_p|$ $= N_{0,i-1}N_{i-1}Rn^2r!$. Consequently, the number of species created at step i is bounded by $|Ls_i| \leq N_{i-1}Rn^2r! + N_{0,i-1}N_{i-1}Rn^2r! = \mathrm{O}(N_{0,i-1}N_{i-1}n^2)$. Generate-species-reactions calls the routines generate-product and update-species-reactions for both mono- and bimolecular reactions. The two routines scale respectively $\mathrm{O}(n^3)$ and $\mathrm{O}(n^4)$. Consequently the time-complexity is $|Ls_{i-1}||Let|(\mathrm{O}(n^3) + \mathrm{O}(n^4)) = \mathrm{O}(N_{i-1}n^4)$for monomolecular reaction and $|Ls_{0,i-1}||Ls_{i-1}||Let| = \mathrm{O}(N_{0,i-1}N_{i-1}n^4)$ for bimolecular reactions. The later being the dominant term the overall time-complexity of generate-species-reaction is $\mathrm{O}(N_{0,i-1}N_{i-1}n^4)$.

Let us now evaluate the number of iterations in the deterministic-network-generator routine. Let k be the maximum valence allowed, usually in organic chemistry $k \leq 4$. It will take at most kn steps to remove all the bonds in Ls_0 and it will take at most kn steps to create the largest structure comprising n atoms. Therefore, all accessible isomers comprising 1 to n atoms can be created in $2kn$ steps, and the maximum number of iterations of the main routine is bounded by $2kn = \mathrm{O}(n)$. Note that we can write $N_{0,i-1} \leq 2kn < N_i >$ where $< N_i >$ is the average size of the list Ls_i. Reversely, the upper bound for $|Ls|$ is $N = 2kn < N_i >$, and $< N_i > = N/(2kn)$. The deterministic-network-generator routine performs $\mathrm{O}(n)$ calls to generate-species-reactions, $\mathrm{O}(n)$ isomorphism checks between Ls_i and $Ls_{0,i}$, and $\mathrm{O}(n)$ insertions of Lr_i in $Lr_{0,i}$. Therefore, the upper-bound time-complexity is $n(N_{0,i}N_{i-1}n^4 + N_{0,i}N_{i-1}n^2 + N_{0,i}N_{i-1}) \leq n(2kn < N_i > N/(2kn)n^4 + 2kn < N_i > N/(2kn)n^2 + 2kn < N_i > N/(2kn)) = N/(2kn)Nn^5 + N/(2kn)Nn^3 + N/(2kn)Nn \leq N^2n^4/2k + N^2n^2/2k + N^2/2k = \mathrm{O}(N^2n^4)$. This complexity is not polynomial since N is bounded by the number of isomers comprising $1, \ldots, n$ atoms, which increases exponentially with n.

We now consider the sampling routines. At any given step i, the random-sampling keep the size of the list $Ls_{0,i-1}$ below a predefined value M_s. Thus, we have $|Ls_{0,i-1}| \leq M_s$. The procedure reduce-mechanism-random performs M_s random selections in $Ls_{0,i-1} \uplus Ls_i$. As seen above, we have $|Ls_i| \leq N_{i-1}Rn^2r! + N_{0,i-1}N_{i-1}Rn^2r! \leq N_{i-1}Rn^2r! + M_sN_{i-1}Rn^2r! = M_s^2\ \mathrm{O}(n^2)$, since in the present case both $N_{0,i-1}$ and N_{i-1} are bounded by M_s. Note that we

also have $|Lr_i| \leq M_s^2\ \mathrm{O}(n^2)$, since $|Ls_i|$ and $|Lr_i|$ have the same upper bounds. Furthermore, $|Ls_{0,i-1} \uplus Ls_i| \leq |Ls_{0,i-1}| + |Ls_i| \leq M_s + M_s^2\ \mathrm{O}(n^2) = M_s^2\ \mathrm{O}(n^2)$. Consequently, M_s random selections in $Ls_{0,i-1} \uplus Ls_i$ will be performed with a time-complexity $M_s^3\ \mathrm{O}(n^2)$. Let us recall that the complexity of generate-reactions-species is $N_{0,i-1}N_{i-1}Rr!n^4$, which in the present case reduces to $M_sM_sRr!n^4 = M_s^2\mathrm{O}(n^4)$. Following the same reductions, the overall complexity of the random-sampling-network-generator algorithm is bounded by: $n(N_{0,i}N_{i-1}n^4 + N_{0,i}N_{i-1}n^2 + N_{0,i}N_{i-1} + M_s^3\ \mathrm{O}(n^2) = M_s^2\mathrm{O}(n^5) + M_s^3\mathrm{O}(n^3)$.

Concentration-sampling differs from random-sampling since the size of the list $Ls_{0,i-1}$ is not restrained. Instead, at any given step i,the sizes of the individual lists $Ls_0, \ldots, Ls_{i-1}$ are kept below a predefined value M_s. Thus, we have $|Ls_{0,i-1}| = N_{0,i-1} = iM_s$. We also have $|Ls_i| \leq N_{i-1}Rn^2r! + N_{0,i-1}N_{i-1}Rn^2r! = N_{i-1}Rn^2r! + 2kn < N_i > N_{i-1}Rn^2r! = M_s^2\ \mathrm{O}(n^3)$, since in the present case $< N_i >$ and N_{i-1} are bounded by M_s. Furthermore, $|Ls_{0,i-1} \uplus Ls_i| \leq |Ls_{0,i-1}| + |Ls_i| \leq 2kn < N_i > + M_s^2\ \mathrm{O}(n^3) = M_s^2\ \mathrm{O}(n^3)$. In turn, the complexity of generate-reactions-species is $N_{0,i-1}N_{i-1}Rr!n^4$, which reduces to $2knM_sM_sRr!n^4 = M_s^2\mathrm{O}(n^5)$.

Concentration-sampling also differs from random-sampling with the reduce-mechanism routine, which calls the MC-Gillespie-step. The MC-Gillespie-step routine first compute the time τ of the next event, then the reaction occuring at time τ, and finally the routine updates the species concentrations. Computing the time of next event is performed using eq. 8 and requires $|Lr_i| \leq M_s^2\ \mathrm{O}(n^3)$ steps. Selecting the next reaction is also achieved in $|Lr_i| \leq M_s^2\ \mathrm{O}(n^3)$ steps. Finally, updating the species concentrations is done in a constant time since there are at most 4 species (i.e. 2 reactants and 2 products) participating to a reaction. The overall time-complexity of MC-Gillespie-step is $M_s^2\ \mathrm{O}(n^3)$.

The MC-Gillespie-step is called by the reduce-mechanism-concentration routine. There are three loops in this routine. Clearly, the complexity of the first loop is $|Ls_{0,i-1} \uplus Ls_i| = M_s^2\ \mathrm{O}(n^3)$. The second loop calls M_c times the Monte-Carlo-Gillespie-step routine, and update the maximum concentration for each species in $Ls_{0,i-1} \uplus Ls_i$. The MC-Gillespie-step routine has a complexity of $M_s^2\ \mathrm{O}(n^3)$. Updating the maximum concentration is achieved in $M_s^2\ \mathrm{O}(n^3)$. Therefore the complexity of the second loop is $M_cM_s^2\ \mathrm{O}(n^3)$. The third

loop searches the M_s maximum concentrations over time of all species in $Ls_{0,i-1} \uplus Ls_i$, its complexity is $M_s^3 \mathrm{O}(n^3)$. The overall time-complexity of reduce-mechanism-concentration is therefore: $(M_s^2 + M_c M_s^2 + M_s^3)\ \mathrm{O}(n^3)$. Thus, the complexity of the concentration-sampling-network-generator algorithm is bounded by: $n(N_{0,i}N_{i-1}n^4 + N_{0,i}N_{i-1}n^2 + N_{0,i}N_{i-1} + (M_s^2 + M_c M_s^2 + M_s^3)\ \mathrm{O}(n^3)) = M_s^2 \mathrm{O}(n^6) + (M_c + M_s + 1)M_s^2\ \mathrm{O}(n^4)$.

The main difference between MC-sampling-network-generator and the other algorithms is that the number of atoms remains the same, M_p. Hence, the number of species with non-zero concentration is at most M_p (i.e., $|Ls^*| \leq M_p$). The routine generate-species-reactions is called for the list Ls^*, consequently, its complexity is $N^*N^*Rr!n^4 = M_p^2 \mathrm{O}(n^4)$. Note also that $|Ls_i|$ the number of species returned as well as $|Lr_i|$ the number of reactions are bounded by $|Ls^*||Ls^*||Let|nr!|Lp| \leq M_p^2 Rn^2 r! = M_p^2 \mathrm{O}(n^2)$. The number of species before the first iteration of MC-sampling-network-generator (i.e., the number of reactants) is at most M_p. The number of species after the first iteration is $|Ls_0 \uplus Ls_1| \leq M_p + M_p^2 \mathrm{O}(n^2) = M_p^2 \mathrm{O}(n^2)$. It can easily be verified that after i iterations the number of species $|Ls_{0,i}|$ will remain bounded by $iM_p^2 \mathrm{O}(n^2)$. After M_c iteration we have $|Ls| \leq M_c M_p^2 \mathrm{O}(n^2)$ and $|Lr| \leq M_c M_p^2 \mathrm{O}(n^2)$ since $|Lr|$ has the same upper bound than $|Ls|$. Hence, the complexity of the isomorphism checks in MC-sampling-network-generator is $|Ls||Ls_i|\ \mathrm{O}(n^2) = M_c M_p^4 \mathrm{O}(n^6)$, while the complexity of MC-Gillespie-step is $|Lr| \leq M_c M_p^2 \mathrm{O}(n^2)$. Since the number of iteration is equal to M_c, the overall complexity of MC-sampling-network-generator is $M_c^2 M_p^4\ \mathrm{O}(n^6)$

V. HYDROCARBON THERMAL CRACKING

The goal in this section is to probe how well the sampling algorithms predict the kinetics of hydrocarbon thermal cracking. More precisely, we want to verify if the sampling algorithms can reproduce the results obtained with the *DNG* deterministic network generator. Prior testing our sampling algorithms it is worth checking how our *DNG* algorithm performs versus experimental results. Although match with experimental data is not our priority here, we want at least to verify that we capture the kinetics of hydrocarbon cracking.

(i) *DNG* versus experimental results. Figures 4 and 5 give the product distribution of ethane and butane using the *DNG* algorithm. Comparison of the results with experimental measurements demonstrates that the generated reaction networks and rate constants give only qualitative agreement.

In the case of ethane thermal cracking at 1118 K and an ethane pressure of 38 Torr, our *DNG* technique predicts ethylene and hydrogen in nearly equal amounts as products, with only a small amount of methane formation, regardless of conversion. Experimentally,[20] significant amounts of methane are formed, the ethylene yield falls from 100% to 75% of the hydrogen yield as conversion increases, and small amounts of acetylene and butadiene are formed. The failure of our *DNG* technique to predict the latter two experimental results is entirely due to the absence in our model of reactions that convert ethylene into acetylene and butadiene. Inclusion of these reactions is expected to rectify this inaccuracy. The low methane yield predicted by our simulation is likely due to inaccuracies in the QSPR used to calculate reaction rates, as our simulation generates all of the reactions commonly believed to occur during ethane thermal cracking.

For n-butane thermal cracking the agreement between experiment and *DNG* output is also qualitative. While our *DNG* technique predicts the formation of all of the major products observed experimentally,[21] the relative amounts of the products do not agree well. In particular our *DNG* technique underpredict the amount of methane and overpredict the amount of hydrogen. As is the case for ethane, the *DNG* algorithm generates all of the important species and reactions for thermal cracking, leading to the conclusion that the discrepancies in product distribution with experiments are the result of inaccuracies in rate parameters predicted from our QSPR.

As noted in Section III, the inacuracies in rate parameters predicted from our QSPR are in large part a reflection of uncertainties in experimental measurements of rate parameters, so that better agreement with experimental product distributions cannot be expected. It should be possible to empirically optimized either the QSPR or individual rate parameters

to obtain better agreement with experiments. However, for the purposes of this study, we believe that the qualitative agreement between experiment and simulation, and the fact that the algorithms generate all of the required species and reactions, sufficiently demonstrates the correct performance of the algorithms.

(ii) *RSNG*, *CSNG* amd *MCNG* versus *DNG*. The *RSNG* algorithm was tested with butane cracking for three different M_s values. Let us recall that *RSNG* selects at random M_s species at each generation step disregarding rates and concentration considerations. It is therefore not surprising to find descrepencies between *RSNG* and *DNG* results since intermediate species and even final products may be removed arbitrarely during the selection process. None of the product distributions obtained with $M_s = 8$, 10, and 16 in Figure 6 match the deterministic results. These findings indicate that *RSNG* is not a valid technique to sample reaction mechanism.

Figure 7 gives the product distribution for butane cracking using the *CSNG* algorithm with three different M_s values. Let us recall that *CSNG* selects at each generation step the M_s most abundant species. $M_s = 6$ gives a different product distribution that *DNG*, $M_s = 7$ gives a product distribution close to *DNG* except for propylene which is overpredicted. $M_s = 8$ gives the same product distribution than *DNG*. Not surpringly, we find that for all values $M_s \geq 8$ the products distributions of *DNG* and *CSNG* are indentical. Indeed, according to the *CSNG* assumption (cf. section II) additional species of low concentration have only a minor effect on the final product distribution. $M_s = 8$ is therefore the treshold above which all *CSNG* product distributions are correct. In Figure 8, the above observation is verified for all hydrocarbons up to 24 atoms. As in can be seens in Figure 8.a, for all tested hydrocabons a treshold for M_s exist above which there is no difference in product distribution between *DNG* and *CSNG*. Futhermore according to Figure 8.b, this trehold scales $2/3\ (n-2)$ where n is the number of atoms of the tested hydrocarbon. All these results lead to the conclusion that *CSNG* can be used to generate reaction network for thermal cracking provided that $M_s \geq 2/3\ (n-2)$.

For $MCNG$ only two parameters can be varied, M_p the initial number of particles, and M_c the number of steps on the Monte-Carlo integration. As shown in Figure 9, for butane cracking, a perfect match with DNG is obtained for $M_p = 10,00$ and a fairly good match for $M_p = 2,000$. In both cases we have $M_c = 10,000$. With lower M_p values, butane decomposition is slower, this is due to the fact that not all of the 145 reactions for butane decomposition are used, slower butane decomposition is especially noticable when the initial number of particle is lower than the number of reactions. $M_p = 2,000$ and $M_c = 10,000$ appear to be the thresholds above which $MCNG$ is in good agreement with DNG, additional tests (up to up to 24 atoms) indicate that these numbers are independant of the tested hydrocarbon.

(iii) Computational complexity for hydorcarbon thermal cracking. Figures 10, 11, and 12 give the computational scaling of the DNG $CSNG$ and $MCNG$ algorithms for hydrocarbon thermal cracking up to 24 atoms. $RSNG$ was not tested for computaional scaling since we have shown that it is not an appropriate technique for network generation. In all cases the hydrocabon results agree well with the theoretical results listed in Tables III and IV. With DNG, the number of species and CPU time scale exponentially with n (the maximum number of atoms per species allowed). While $CSNG$ and $MCNG$ give product distribution identical to DNG, in both cases, the number of species generated and the CPU time scale polynomially with n.

VI. CONCLUSION

We have introduced here a new technique where network generation and reduction are performed simultaneously. Theoretically, we have proven that the deterministic network generators based on the Dugundji-Ugi methodology have a number of species/reactions and CPU running time that scale exponentially. We have also proven that our concentration sampling and Monte-Carlo sampling algorithms scale polynomially while giving identical

results than deterministic generators. All our theoretical findings are in agreement with results obtained for hydrocarbon thermal cracking. The computational running time scaling we found with the concentration sampling algithm is $n^6/10^6$ and $n^6/10^4$ with the Monte-Carlo sampling algorithm, where n is the maximum species size. Although the polynomial exponant is rather large these results enables one to study organic reactions on compounds comprising up to 50 atoms in few CPU hours using standard desktop technology. Whereas improvement in the scaling exponant need to carried out, our sampling algorithms already offer the possibility of generating complex reaction networks such those studied in combution and petroleum refining.

GLOSSARY

DNG : Deterministic network generator algorithm.

$RSNG$: Random sampling network generator algorithm.

$CSNG$: Concentration sampling network generator algorithm.

$MCNG$: Monte-Carlo sampling network generator algorithm.

e_s : Maximum number of lone electron per species.

e_p : Maximum number of lone electron per atom.

Ls : List of all species produced by algorithm.

Lr : List of all reactions produced by algorithm.

$L(s)$: List of all species concentration.

Ls^* : List of all species with non-zero concentration.

Ls_0 : List of reactants.

Ls_i : List of all species produced by algorithm at step i.

Lr_i : List of all reactions produced by algorithm at step i.

$Ls_{0,i}$: List of all species produced by algorithm up to step i.

$Lr_{0,i}$: List of all reactions produced by algorithm up to step i.

n : Maximum number of atom per species.

N : Maximum number of species produced by algorithm.

N^* : Number of species with non-zero concentration.

N_0 : Number of reactants.

N_i : Maximum number of species produced by algorithm at step i.

$N_{0,i}$: Maximum number of species produced by algorithm up to step i.

M_c : Maximum number of Monte-Carlo steps.

M_s : Maximum number of species created per generation steps.

M_p : Maximum number of particles for Monte-Carlo integeration.

O_{max} : Maximum reaction order.

r : Maximum number of atoms per elementary transition.

R : Maximum number of elementary transitions.

ACKNOWLEDGMENTS

We would like to acknowledge the funding provided by the Math. Information and Computer Science program of the U.S. Department of Energy under contract DE-AC04-76DP00789.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

REFERENCES

[1] Ugi, I.; Bauer, J.; Bley, K.; Dengler, A.; Dietz A.; Fontain, E.; Gruber, B.; Herges, R.; Knauer, M.; Reitsam, K.; Stein, N. Computer-Assisted Solution of Chemical Problems - The Historical Development and the Present State of the Art of a New Discipline of Chemistry. Angew. Chem. Int. Ed. Engl. 1993, 32, 201-227.

[2] Ugi, I.; Fontain, E.; Bauer, J. Transparent formal methods for reducing the combinatorial abundance of conceivable solutions to a chemical problem. Computer-assisted elucidation of complex mechanism. Analytica Chimica Acta 1990, 235, 155-161.

[3] Clymans, P. J.; Froment, G. F. Computer-generation of reaction paths and rates equations in the thermal cracking of normal and branched paraffins. Computers and Chemical Engineering 1984, 83, 137-142.

[4] Hillewaert, L. P.; Dierickx, J. L.; Froment, G. F. Computer Generation of Reaction Schemes and Rates Equations for Thermal Cracking. AIChE Journal 1988, 34, 17-24.

[5] Frenklach, M. Modeling of Large Reaction Systems. In textitComplex Chemical Reaction Systems, Mathematical Modelling and Simulation; Warnatz J.; Jager W., Eds;, Springer Series in Chemical Physics, Vol. 47, Springer-Verlag: Berlin, 1987, 2-16.

[6] Di Maio, F. P.; Lignola, P. G. KING, a KInetic Network Generator. Chemical Engineering Science 1992, 47, 2713-27181.

[7] Broadbelt, L. J.; Stark, S. M.; Klein, M. T. Computer Generated Pyrolysis Modeling: On-the-fly Generation of Species, Reactions, and Rates. Ind. Eng. Chem. Res. 1994, 33, 790-799.

[8] Susnow, R. G.; Dean, A. M.; Green, W. H.; Peczak. P.; Broadbelt, L. J. Rate-Based Construction of Kinetic Models for Complex Systems. J. Phys. Chem A 1997, 101, 3731-3740.

[9] Temkin, O. N.; Zeigarnik, A. V.; Bonchev, D. *Chemical Reaction Network. A Graph-*

Theoretical Approach; CRC Press: Boca Raton, FL, 1996.

[10] Dugundji, J.; Gillespie, P.; Marquarding, D. Ugi, I. Metric Space and Graphs Representing the LOgical Structure of Chemistry. In *Chemical Applications of Graph Theory*; Balaban, A. T. Ed.; Academic Press: London 1976, Chapter 6.

[11] Dugundji, J.; Ugi, I. Theory of the be- and r- matrices. Top. Curr. Chem. 1973, 39, 19-29.

[12] Fontain, E.; Reitsam, K. The Generation of Reaction Network with RAIN. 1. The Reaction Generator. J. Chem. Info. Comput. Sci. 1991, 31, 96-101.

[13] Nigam, A.; Klein, M. T. A Mechanism-Oriented Lumping Strategy for Heavy Hydrocarbon Pyrolysis: Imposition of Quantitative Structure-Reactivity Relationship for Pure Components. IEC RESEARCH 1993, 32, 1297-1303.

[14] Faulon, J.-L. Isomorphism, automorphism-partitioning, and cannonical labeling can be solved of polynomial-time for molecular graph. J. Chem. Inf. Comput. Sci., 1998, 38, 432-444.

[15] Gillespie, D. T. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. Journal of Computational Physics 1976, 22, 403-434.

[16] no ref. yet.

[17] Allara, D. L.; Shaw, R. A Compilation of Kinetic Parameters for the Thermal Degradation of n-Alkane Molecules. J. Phys. Chem. Ref. Data 1980, 9, 523-559.

[18] Kucera, L. *Combinatorial Algorithm*; Adam Hilger: Bristol, 1989.

[19] Trinajstic, N. *Chemical Graph Theory*; 2nd ed.; CRC Press: Boca Raton, FL, 1992.

[20] McConnell, C. F.; Head, B. D. Pyrolysis of Ethane and Propane. in Pyrolysis, Theory and Indistrial Practice. (Albright. L. F.; Crynes, B. L.; Corcoran, W. H. Eds.), Academic Press, New York, 1983, 25-45.

[21] Corcoran, W. H. Pyrolysis of n-Butane. in Pyrolysis, Theory and Indistrial Practice. (Albright. L. F.; Crynes, B. L.; Corcoran, W. H. Eds.), Academic Press, New York, 1983, 47-68.

[22] Rebick, C. Pyrolysis of Heavy Hydrocarbons. in Pyrolysis, Theory and Indistrial Practice. (Albright. L. F.; Crynes, B. L.; Corcoran, W. H. Eds.), Academic Press, New York, 1983, 69-87.

Table I: Network generator constraints

	Maximum value allowed			
	DNG	RSNG	CSNG	MCNG
Network constraints				
r	∞ (3)	∞ (3)	∞ (3)	∞ (3)
R	∞ (8)	∞ (8)	∞ (8)	∞ (8)
O_{max}	2 (2)	2 (2)	2 (2)	2 (2)
M_s	n/a	100,000 (20)	100,000 (20)	n/a
M_c	n/a	n/a	100,000 (10,000)	100,000 (10,000)
M_p	n/a	n/a	100,000 (10,000)	100,000 (10,000)
Species constraints				
n	∞ (50)	∞ (50)	∞ (50)	1,000,000 (50)
e_s	∞ (2)	∞ (2)	∞ (2)	1,000,000 (2)
e_p	2 (1)	2 (1)	2 (1)	2 (1)

Cf. Glossary for notations. Values in parentheses are actual values taken for hydrocarbon thermal cracking.

Table II: Rate parameters for hydrocarbon thermal cracking reactions

Reaction	$\log A$ (s^{-1})	E_0 (kcal)	α	β
Bond homolysis	16.802	86.950	-0.02199	0.02890
$\beta-$scission	13.392	35.144	-0.01736	0.02468
H addition to olefins	9.924	3.252	0.07558	-0.07045
C_xH_{2x+1} addition to olefins	7.893	7.826	-0.00246	0.00060
H abstraction from alkanes by C_xH_{2x+1} radicals to form alkanes	8.365	11.560	-0.02767	0.02780
H abstraction from alkanes by H radicals to form alkanes	10.986	5.481	0.34390	-0.39190
H abstraction from H_2 by C_xH_{2x+1} radicals to form alkanes	9.323	9.853	-0.29900	0.28200
Radical recombination	9.843	0	0	0

Cf. equation 10 for notations.

Table III: Number of species: upper bounds

	DNG	RSNG	CSNG	MCNG
$\|Ls_i\|$	$O(N_{0,i-1}N_{i-1}n^2)$	$M_s^2O(n^2)$	$M_s^2O(n^3)$	$M_p^2O(n^2)$
$\|Ls_{0,i-1}\|$	$O(N_{0,i-1})$	M_s	iM_s	$iM_p^2O(n^2)$
$\|Ls\|$	$O(N)$	M_s	$M_sO(n)$	$M_cM_p^2O(n^2)$

Cf. Glossary for terminology.

Table IV: Computational time-complexity: upper bounds

	DNG	RSNG	CSNG	MCNG
generate-product	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$
update-species-reactions	$O(n^4)$	$O(n^4)$	$O(n^4)$	$O(n^4)$
generate-species-reactions	$O(N_{0,i-1}N_{i-1}n^4)$	$M_s^2O(n^4)$	$M_s^2O(n^5)$	$M_p^2O(n^4)$
MC-Gillespie-step	n/a	n/a	$M_s^2O(n^3)$	$M_cM_p^2O(n^2)$
reduce-mechanism	n/a	$M_s^2O(n^2)$	$(M_s^2 + M_cM_s^2 + M_s^3)O(n^3)$	n/a
full algorithm	$O(N^2n^4)$	$M_s^2O(n^5) + M_s^3O(n^3)$	$M_s^2O(n^6) + (M_c + M_s + 1)O(n^4)$	$M_cM_p^4O(n^6)$

Cf. Glossary for terminology.

Figure Captions

Figure 1: Example of $(be-)$ and $(r-)$ matrices for elementary transitions of hydrocarbon thermal cracking. a) $(be-)$matrices for methane and methyl. b) $(r-)$-matrices for the 5 elementary transitions of hydrocarbon thermal cracking.

Figure 2: The 3 first steps of ethane thermal cracking using the 5 elementary transitions given in Fig. 1.

Figure 3: Predicted vs. experimental rate constants for hydrocarbon thermal cracking.

Figure 4: Ethane thermal cracking with DNG. T = 1118K, initial ethane concentration = .000545 M.

Figure 5: Butane thermal cracking with DNG. T = 863K, initial butane concentration = .001 M.

Figure 6: Butane thermal cracking with $RSNG$. T = 863K, initial butane concentration = .001 M. a) $RSNG$ with $M_s = 8$, b) $RSNG$ with $M_s = 10$, c) $RSNG$ with $M_s = 16$.

Figure 7: Butane thermal cracking with $CSNG$. T = 863K, initial butane concentration = .001 M. a) $RSNG$ with $M_s = 6$, b) $RSNG$ with $M_s = 7$, c) $RSNG$ with $M_s = 8$.

Figure 8: Product distribution RMSD between DNG and $CSNG$. a) RMSD for individual hydrocabons for different M_s values. b) M_s vs. reactant size (n) for which RMSD = 0. The equation of the straight line is $M_s = 2/3\ (n-2)$

Figure 9: Butane thermal cracking with $MCNG$. T = 863K, initial butane concentration = .001 M. a) $M_p = 100$, b) $M_p = 500$, c) $M_p = 2{,}000$, d) $M_p = 10{,}000$ (all major products are shown). In all cases $MCNG$ was run until convergence, that is for at most $M_c = 10{,}000$ steps.

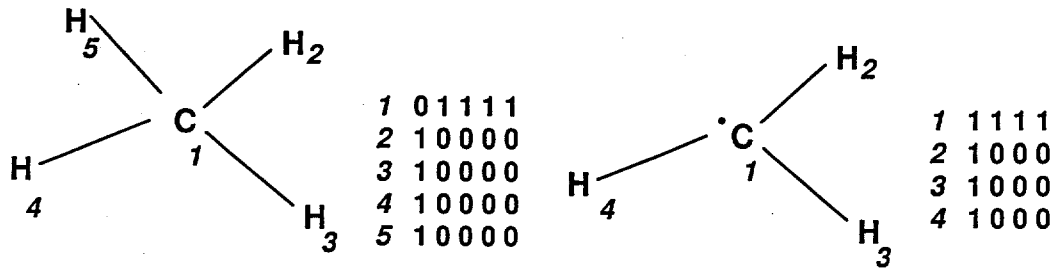
Figure 10: DNG computational scaling.

Figure 11: *CSNG* computational scaling for $M_s = 2/3\ (n-2)$.

Figure 12: *MCNG* computational scaling for $M_p = 2{,}000$ and $M_c = 10{,}000$.

FIGURES

Fig. 1



a)

homolysis:	$R_1\text{-}R_2 \rightarrow R_1^{\bullet} + R_2^{\bullet}$	R1 1 -1 R2 -1 1
recombination:	$R_1^{\bullet} + R_2^{\bullet} \rightarrow R_1\text{-}R_2$	R1 -1 1 R2 1 -1
H-abstraction:	$CH + R^{\bullet} \rightarrow RH + C^{\bullet}$	H 0 -1 1 C -1 1 0 R• 1 0 -1
β-scission:	$R_1\text{-}C_\beta\text{-}C_\alpha^{\bullet} + R_2^{\bullet} \rightarrow C_\alpha{=}C_\beta + R_1\text{-}R_2$	R1 0 1 0 -1 R2 1 -1 0 0 Cα 0 0 -1 1 Cβ -1 0 1 0
addition:	$C_\alpha{=}C_\beta + R_1\text{-}R_2 \rightarrow R_1\text{-}C_\beta\text{-}C_\alpha^{\bullet} + R_2^{\bullet}$	R1 0 -1 0 1 R2 -1 1 0 0 Cα 0 0 1 -1 Cβ 1 0 -1 0

b)

Fig. 2

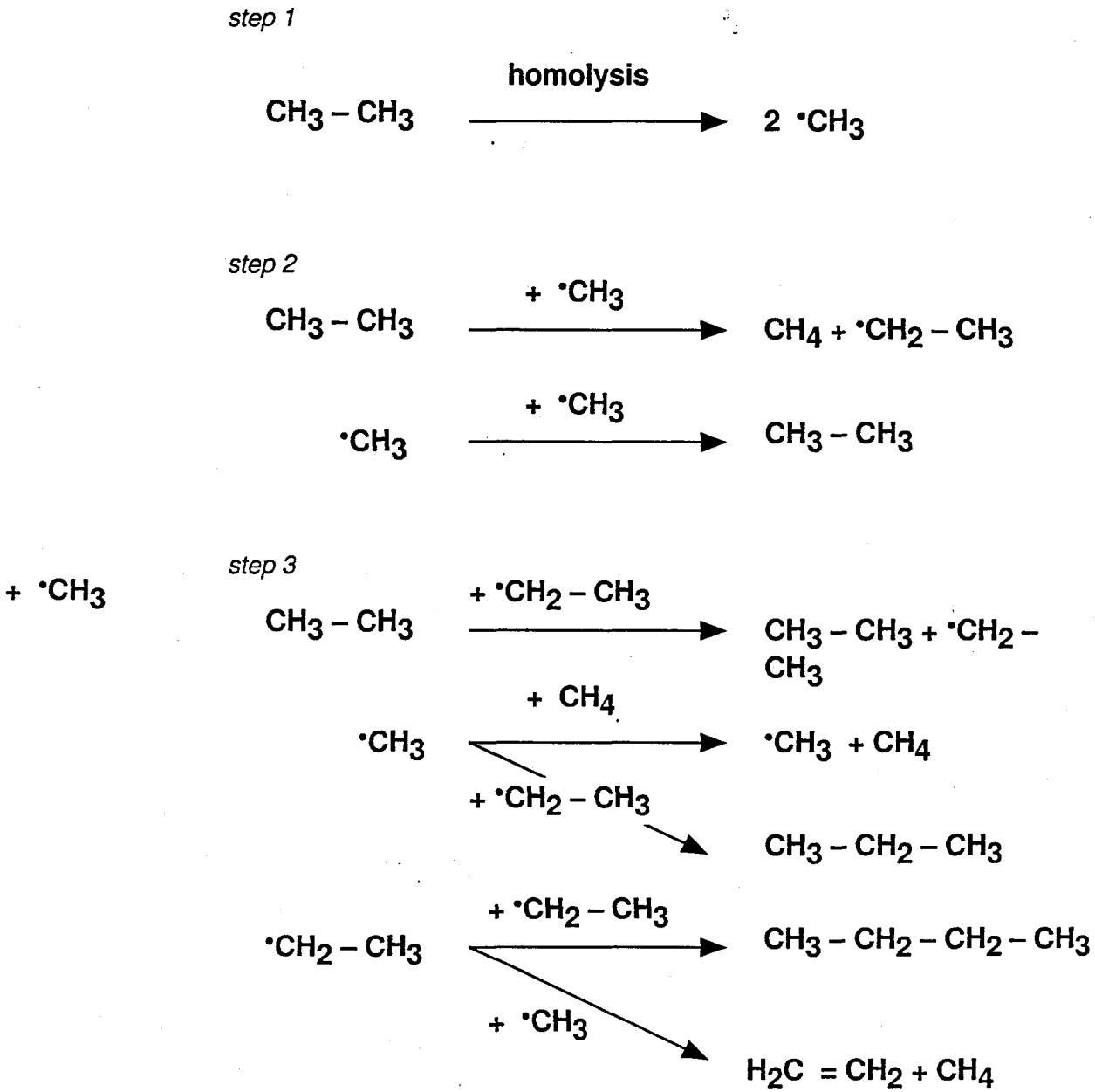

step 1
homolysis
CH3 – CH3
2 •CH3
step 2
+ •CH3
CH3 – CH3
CH4 + •CH2 – CH3
+ •CH3
•CH3
CH3 – CH3
step 3
+ •CH3
+ •CH2 – CH3
CH3 – CH3
CH3 – CH3 + •CH2 – CH3
+ CH4
•CH3
•CH3 + CH4
+ •CH2 – CH3
CH3 – CH2 – CH3
+ •CH2 – CH3
•CH2 – CH3
CH3 – CH2 – CH2 – CH3
+ •CH3
H2C = CH2 + CH4

Fig. 3

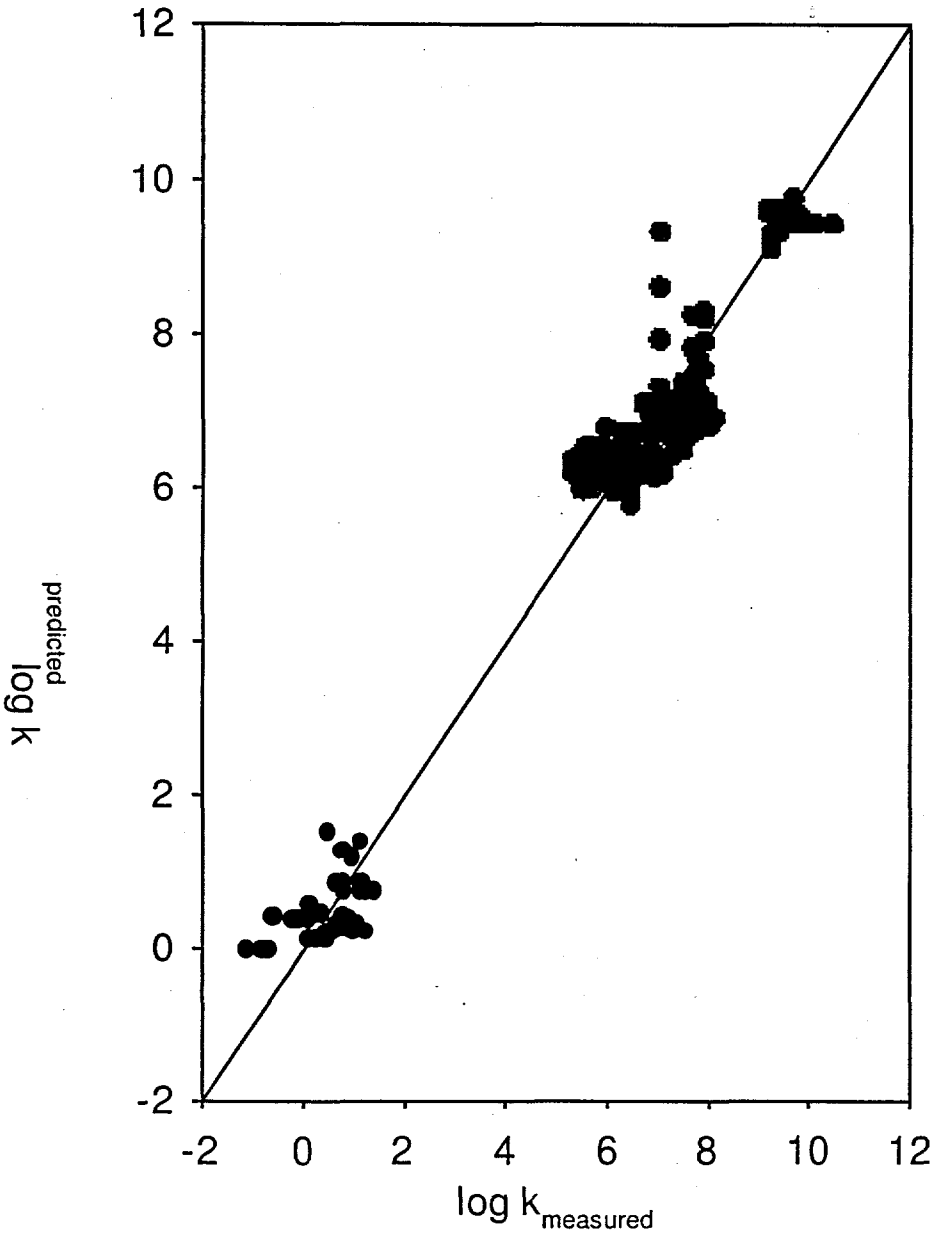


Fig. 4

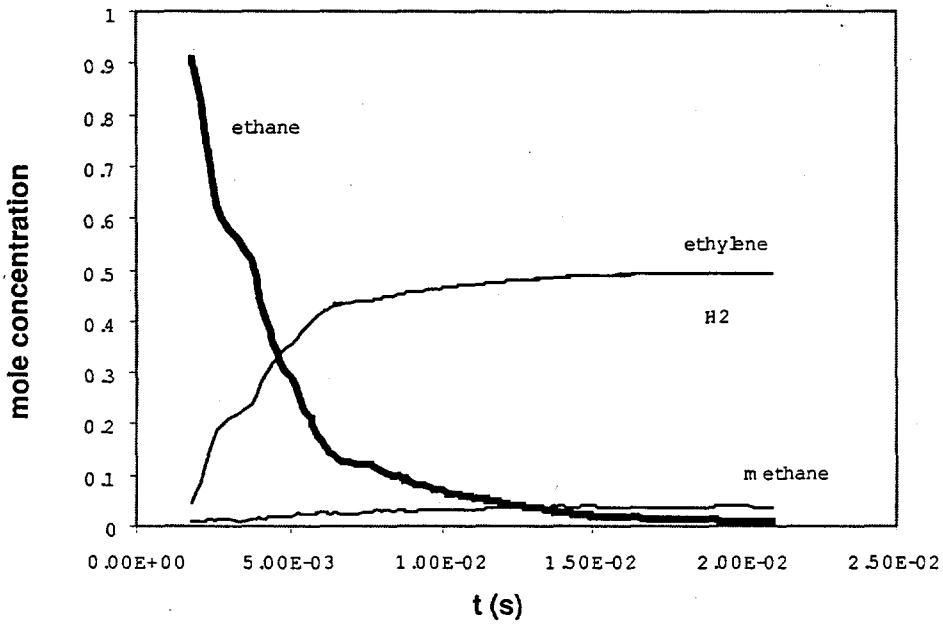

mole concentration
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
ethane
ethylene
H2
methane
0.00E+00
5.00E-03
1.00E-02
1.50E-02
2.00E-02
2.50E-02
t (s)

Fig. 5

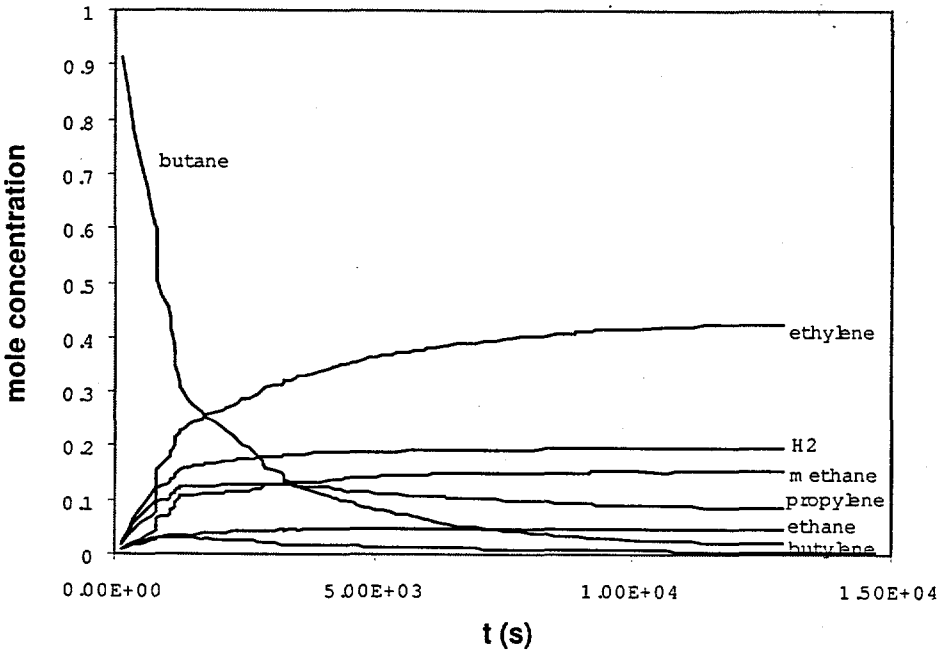
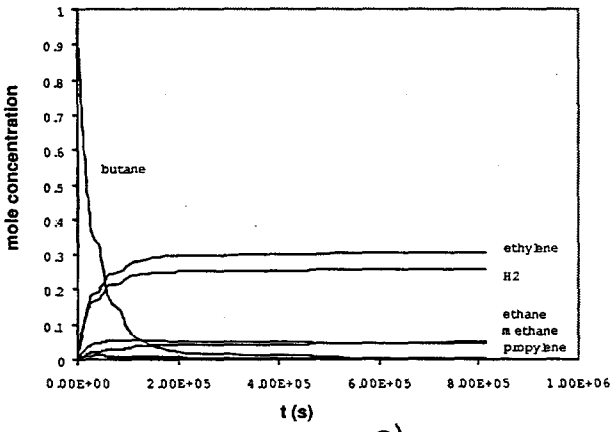
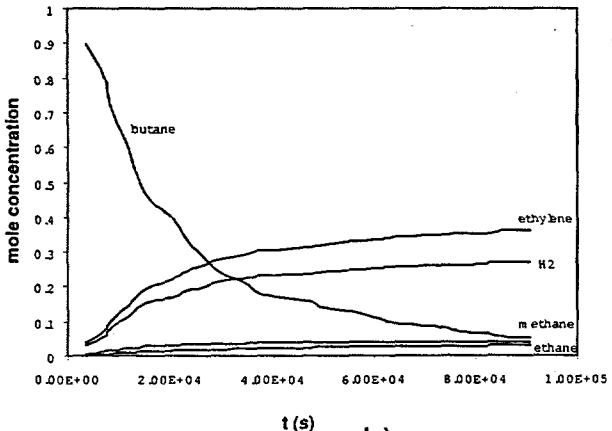

mole concentration
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
butane
ethylene
H2
methane
propylene
ethane
butylene
0.00E+00
5.00E+03
1.00E+04
1.50E+04
t (s)

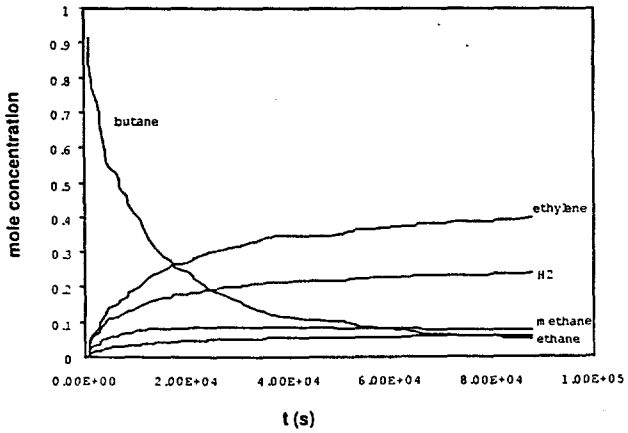
Fig. 6


mole concentration
butane
ethylene
H2
ethane
methane
propylene
t (s)

a)

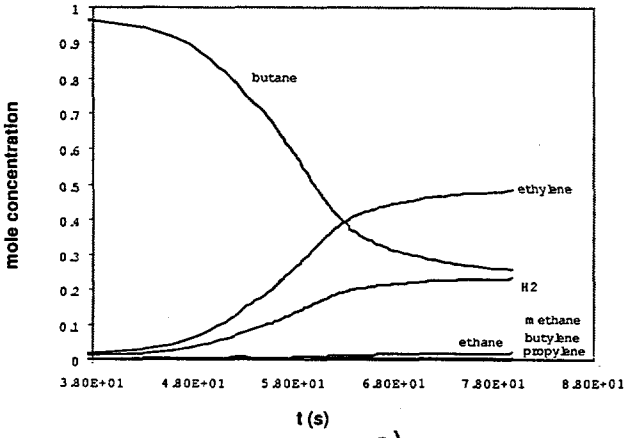

mole concentration
butane
ethylene
H2
methane
ethane
t (s)

b)

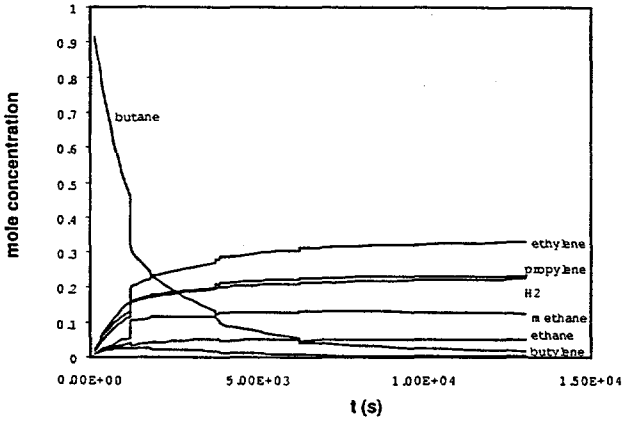

mole concentration
butane
ethylene
H2
methane
ethane
t (s)

c)

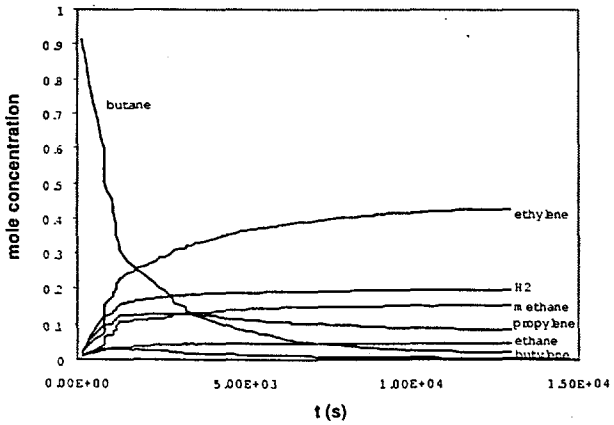
Fig. 7


mole concentration
butane
ethylene
H2
methane
butylene
propylene
ethane
t (s)

a)

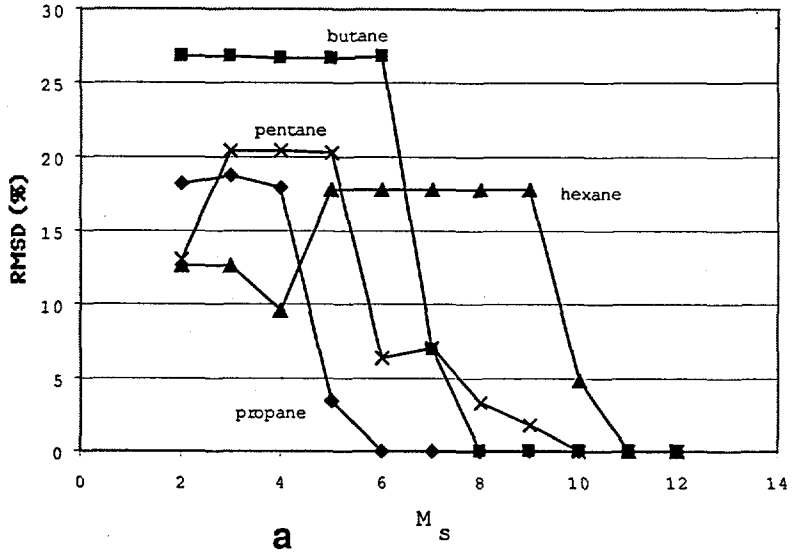

mole concentration
butane
ethylene
propylene
H2
methane
ethane
butylene
t (s)

b)

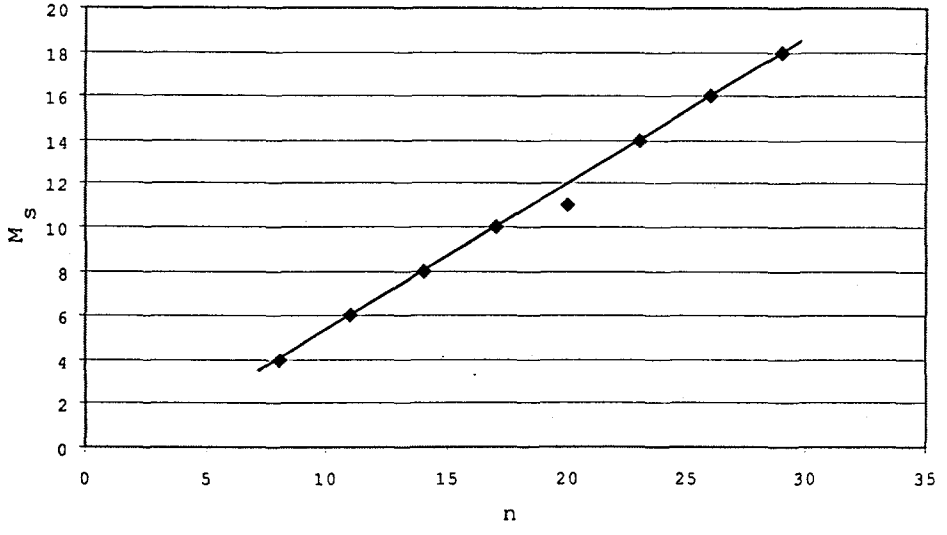

mole concentration
butane
ethylene
H2
methane
propylene
ethane
t (s)

c)

Fig. 8


30
25
20
15
10
5
0
RMSD (%)
butane
pentane
hexane
propane
0
2
4
6
8
10
12
14
M s

a
)


20
18
16
14
12
10
8
6
4
2
0
M s
0
5
10
15
20
25
30
35
n

b)

Fig. 9

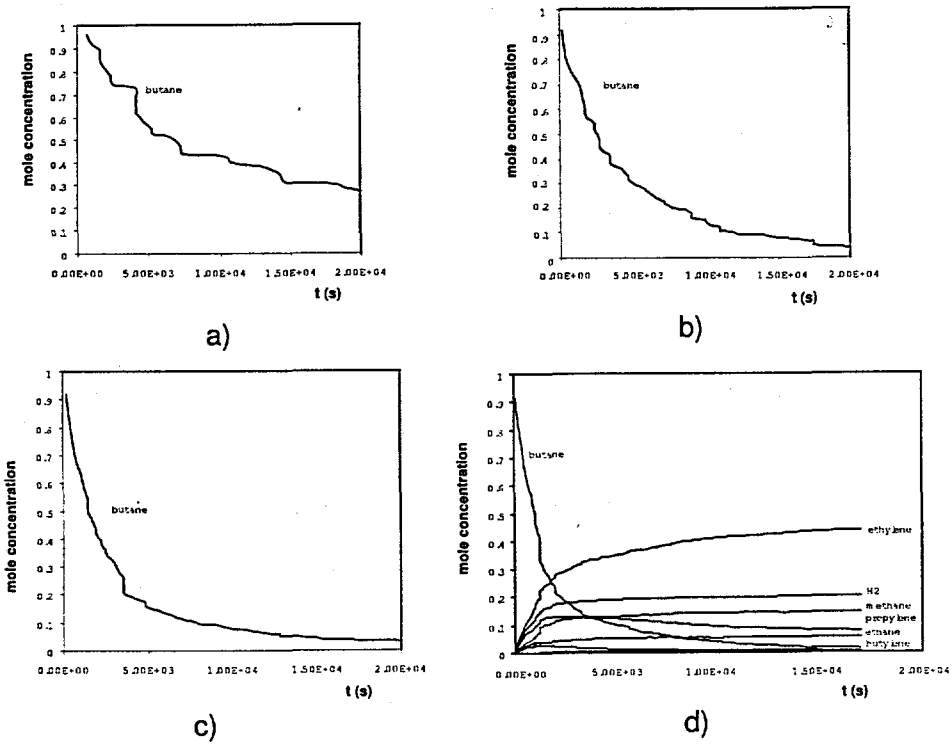
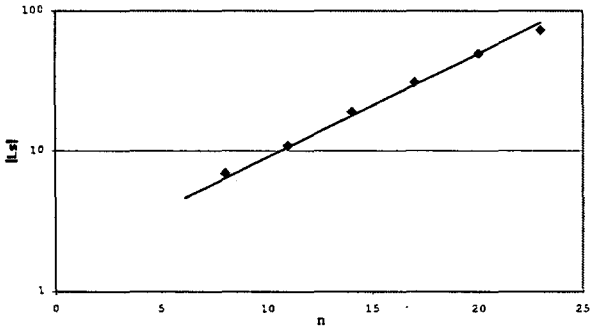
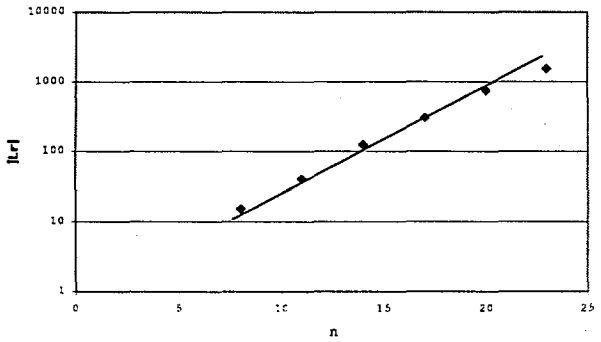

mole concentration
butane
t (s)
a)
b)
c)
d)
ethylene
H2
methane
propylene
ethane
butylene

Fig. 10

a)


100
10
1
|L-s|
0
5
10
15
20
25
n

b)


10000
1000
100
10
1
|L-r|
0
5
10
15
20
25
n

c)

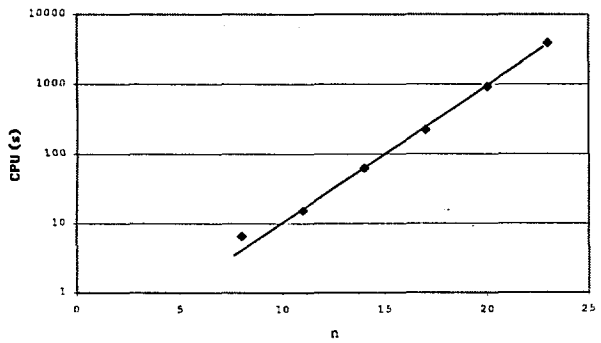
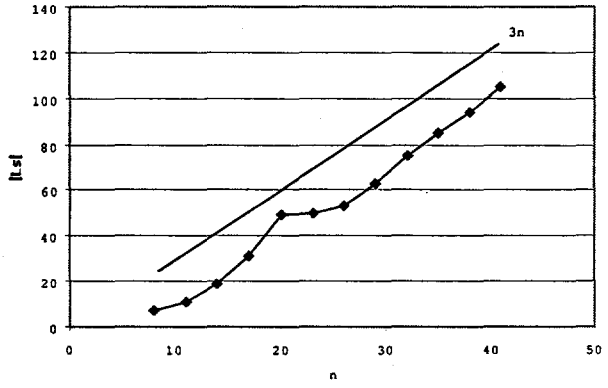
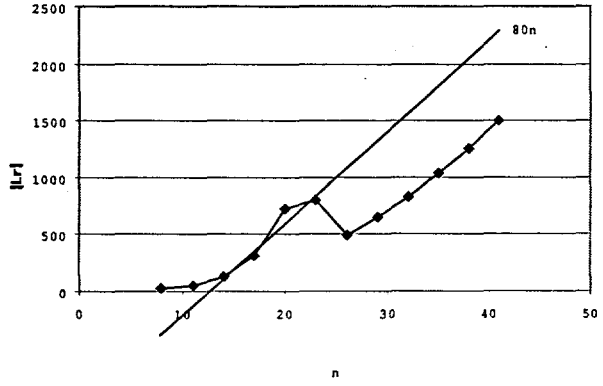

10000
1000
100
10
1
CPU (s)
0
5
10
15
20
25
n

Fig. 11

a)


140
120
100
80
60
40
20
0
|Ls|
3n
0
10
20
30
40
50
n

b)


2500
2000
1500
1000
500
0
|Lr|
80n
0
10
20
30
40
50
n

c)

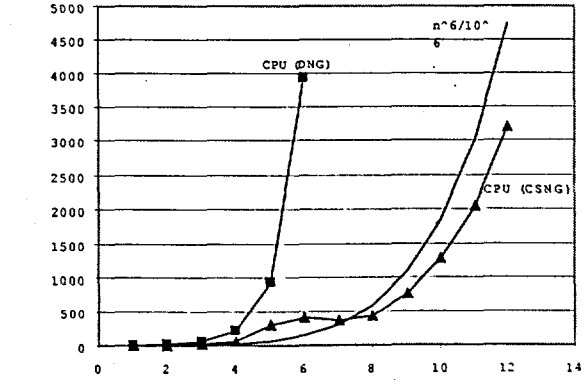
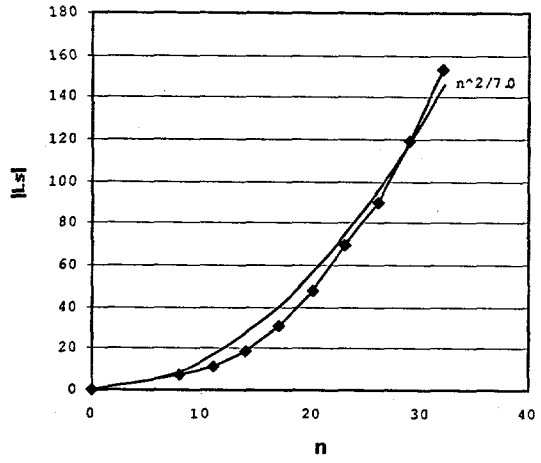
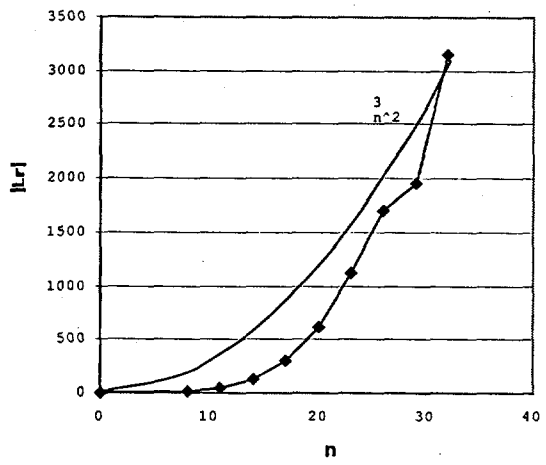

5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
n^6/10^6
CPU (DNG)
CPU (CSNG)
0
2
4
6
8
10
12
14

Fig. 12

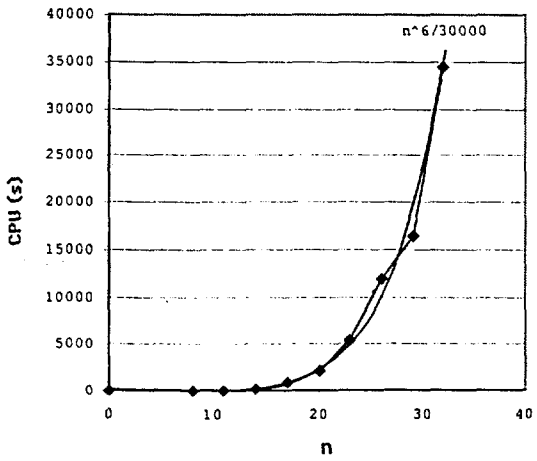
a)


180
160
140
120
100
80
60
40
20
0
|L_s|
n^2/7.0
0
10
20
30
40
n

b)


3500
3000
2500
2000
1500
1000
500
0
|L_r|
3
n^2
0
10
20
30
40
n

c)


40000
35000
30000
25000
20000
15000
10000
5000
0
CPU (s)
n^6/30000
0
10
20
30
40
n