<<N17>>

Establishment and Numerical Solution of Differential Equations for the Composition of Multiple Free Radical Copolymer

Wentao Zeng, Yuhong Ning, Jiaxu Ma

Supervisor: Yachao Qi

Tsinghua University High School

Abstract In this paper, the differential equations for copolymerization of multiple monomers are derived based on the steady-status assumption of Alfrey-Goldfinger and Valvassori-Sartori respectively, and a numerical method is implemented in C++ language. The differential equations of copolymer composition with wide application presented in this paper provide a theoretical guidance for industrial production and academic research of multiple free radical copolymerization.

Keywords free radical, monomer, multiple copolymerization, steady-status assumption of Alfrey-Goldfinger, steady-status assumption of Valvassori-Sartori, differential equations of polymer composition, numerical solution, explicit Euler method.

1. Introduction

The study of the free radical copolymerization behavior, especially the micro structure and macroscopic composition of copolymerarbitrary, has important guiding significance for the molecular structure design of free radical copolymerization. Pan [1] proposed differential equations of the composition of binary copolymer as follows:

$$\frac{d[M_1]}{d[M_2]}=\frac{[M_1]}{[M_2]}\cdot\frac{r_{12}[M_1]+[M_2]}{[M_1]+r_{21}[M_2]} \tag{1-1}$$

Where $[M_1]$ and $[M_2]$ stand for concentration of monomer, r is reactivity ratio.

In addition, Pan gave the analytic solution of equations (1-1). In the analytic solution, C is conversion ratio, $[M]$ is the sum of $[M_1]$ and $[M_2]$, $[M^0]$ is initial value of $[M]$.

$$\frac{[M^0]-[M]}{[M^0]}=1-\left(\frac{f_1}{f_1^0}\right)^{\alpha}\left(\frac{f_2}{f_2^0}\right)^{\beta}\left(\frac{f_1^0-\delta}{f_1-\delta}\right)^{\gamma} \tag{1-2}$$

$$f_1=\frac{[M_1]}{[M_1]+[M_2]} \qquad f_2=1-f_1 \qquad \alpha=\frac{r_{21}}{1-r_{21}}$$

$$\beta=\frac{r_{12}}{1-r_{12}} \qquad \gamma=\frac{1-r_{12}r_{21}}{(1-r_{12})\cdot(1-r_{21})} \qquad \delta=\frac{1-r_{21}}{(2-r_{12}-r_{21})}$$

The above formulas are too complicated and have many singular points, it is difficult to handle for programming.

For ternary copolymerization, references [1] presented a differential form as follows:

<<N17>>

$$
\begin{aligned}
&d[M_1]/d[M_2]/d[M_3]= \\
&[M_1]\cdot\left(\frac{[M_1]}{r_{31}r_{21}}+\frac{[M_2]}{r_{21}r_{32}}+\frac{[M_3]}{r_{13}r_{23}}\right)\cdot\left([M_1]+\frac{[M_2]}{r_{12}}+\frac{[M_3]}{r_{13}}\right)\Big/ \\
&[M_2]\cdot\left(\frac{[M_1]}{r_{12}r_{31}}+\frac{[M_2]}{r_{12}r_{32}}+\frac{[M_3]}{r_{32}r_{13}}\right)\cdot\left(\frac{[M_1]}{r_{21}}+[M_2]+\frac{[M_3]}{r_{23}}\right)\Big/ \\
&[M_3]\cdot\left(\frac{[M_1]}{r_{13}r_{21}}+\frac{[M_2]}{r_{23}r_{12}}+\frac{[M_3]}{r_{13}r_{23}}\right)\cdot\left(\frac{[M_1]}{r_{31}}+\frac{[M_2]}{r_{32}}+[M_3]\right)
\end{aligned}
\tag{1-3}
$$

Since the formula derivation process is too complicated, there is no research on the differential equations and the corresponding numerical solution for the composition beyond ternary copolymer in literature.

In this paper, the differential equations for the composition of quaternary copolymer are derived based on the steady-status assumption of Alfrey-Goldfinger[1] and the steady-status hypothesis of Valvassori-Sartori[1], and a generalized result can be applied to copolymerization of arbitrary monomers. A numerical algorithm is implemented in C++ language.

2. Differential equations for quaternary copolymerization

In quaternary copolymerization, ignoring the penultimate effect, there are four chain initiation reactions, sixteen chain propagation reactions and twelve chain termination reactions. The reaction equations and rate formulas of chain propagation are listed as follows:

$$
\begin{array}{llll}
M_1^{\cdot}+M_1\rightarrow MM_1^{\cdot} & R_{11}=k_{11}\left[M_1^{\cdot}\right]\cdot[M_1] & M_1^{\cdot}+M_3\rightarrow MM_3^{\cdot} & R_{13}=k_{13}\left[M_1^{\cdot}\right]\cdot[M_3] \\
M_2^{\cdot}+M_1\rightarrow MM_1^{\cdot} & R_{21}=k_{21}\left[M_2^{\cdot}\right]\cdot[M_1] & M_2^{\cdot}+M_3\rightarrow MM_3^{\cdot} & R_{23}=k_{23}\left[M_2^{\cdot}\right]\cdot[M_3] \\
M_3^{\cdot}+M_1\rightarrow MM_1^{\cdot} & R_{31}=k_{31}\left[M_3^{\cdot}\right]\cdot[M_1] & M_3^{\cdot}+M_3\rightarrow MM_3^{\cdot} & R_{33}=k_{33}\left[M_3^{\cdot}\right]\cdot[M_3] \\
M_4^{\cdot}+M_1\rightarrow MM_1^{\cdot} & R_{41}=k_{41}\left[M_4^{\cdot}\right]\cdot[M_1] & M_4^{\cdot}+M_3\rightarrow MM_3^{\cdot} & R_{43}=k_{43}\left[M_4^{\cdot}\right]\cdot[M_3] \\
M_1^{\cdot}+M_2\rightarrow MM_2^{\cdot} & R_{12}=k_{12}\left[M_1^{\cdot}\right]\cdot[M_2] & M_1^{\cdot}+M_4\rightarrow MM_4^{\cdot} & R_{14}=k_{14}\left[M_1^{\cdot}\right]\cdot[M_4] \\
M_2^{\cdot}+M_2\rightarrow MM_2^{\cdot} & R_{22}=k_{22}\left[M_2^{\cdot}\right]\cdot[M_2] & M_2^{\cdot}+M_4\rightarrow MM_4^{\cdot} & R_{24}=k_{24}\left[M_2^{\cdot}\right]\cdot[M_4] \\
M_3^{\cdot}+M_2\rightarrow MM_2^{\cdot} & R_{32}=k_{32}\left[M_3^{\cdot}\right]\cdot[M_2] & M_3^{\cdot}+M_4\rightarrow MM_4^{\cdot} & R_{34}=k_{34}\left[M_3^{\cdot}\right]\cdot[M_4] \\
M_4^{\cdot}+M_2\rightarrow MM_2^{\cdot} & R_{42}=k_{42}\left[M_4^{\cdot}\right]\cdot[M_2] & M_4^{\cdot}+M_4\rightarrow MM_4^{\cdot} & R_{44}=k_{44}\left[M_4^{\cdot}\right]\cdot[M_4]
\end{array}
\tag{2-1}
$$

Where,

R_{ij}: reaction rate

k_{ij}: reaction rate constant

[M_i.]: concentration of free radical M_i.

[M_i]: concentration of monomer M_i

The equations for reaction rate of four monomers, also known as disappearing rate of monomers is as follows:

$$\begin{aligned}
\frac{d[M_1]}{dt} &= -(R_{11}+R_{21}+R_{31}+R_{41}) \\
\frac{d[M_2]}{dt} &= -(R_{12}+R_{22}+R_{32}+R_{42}) \\
\frac{d[M_3]}{dt} &= -(R_{13}+R_{23}+R_{33}+R_{43}) \\
\frac{d[M_4]}{dt} &= -(R_{14}+R_{24}+R_{34}+R_{44})
\end{aligned} \tag{2-2}$$

Here, *dt* is differential of time *t*.

There are 12 reactivity ratios which are defined in a matrix:

$$r = \begin{pmatrix} 1 & r_{12} & r_{13} & r_{14} \\ r_{21} & 1 & r_{23} & r_{24} \\ r_{31} & r_{32} & 1 & r_{34} \\ r_{41} & r_{42} & r_{43} & 1 \end{pmatrix} \tag{2-3}$$

Where $r_{ij} = \frac{k_{ii}}{k_{ij}}$, $i \in \{1,2,3,4\}, j \in \{1,2,3,4\}, i \neq j$

According to steady-status assumption in Alfrey-Goldfinger's method[1], four equations can be obtained. Among these equations, only three is independent. So, three concentrations free radicals can be eliminated by using these equations

$$\begin{aligned}
R_{11}+R_{12}+R_{13}+R_{14} &= R_{11}+R_{21}+R_{31}+R_{41} \\
R_{21}+R_{22}+R_{23}+R_{24} &= R_{12}+R_{22}+R_{32}+R_{42} \\
R_{31}+R_{32}+R_{33}+R_{34} &= R_{13}+R_{23}+R_{33}+R_{43} \\
R_{41}+R_{42}+R_{43}+R_{44} &= R_{14}+R_{24}+R_{34}+R_{44}
\end{aligned} \tag{2-4}$$

The differential equations can be derived from the known conditions (2-1), (2-2), (2-3) and (2-4): 1) All reaction rate constants in equations (2-1) can be eliminated except k_{11}, k_{22}, k_{33} and k_{44} when reactivity ratio table (2-3) was substituted into equations (2-1), 2) The concentration of three free radicals can be calculated by steady-status assumption (2-4), the results was put into equations (2-1), 3) The embryonic form of differential equations can generated by introducing (2-1) into (2-2), 4) Extract the common variables and represent them by a function α, 5) The equations can be further organized and simplified.

The instantaneous composition of the quaternary copolymer in vector form satisfy equations (2-5):

$$\frac{dM}{dt} = \alpha \cdot \begin{pmatrix}
[M_1]\cdot\left\{\frac{[M_1]}{r_{21}\cdot r_{31}\cdot r_{41}}+\frac{[M_2]}{r_{21}\cdot r_{32}\cdot r_{42}}+\frac{[M_3]}{r_{31}\cdot r_{23}\cdot r_{43}}+\frac{[M_4]}{r_{41}\cdot r_{24}\cdot r_{34}}\right\}\cdot\left\{[M_1]+\frac{[M_2]}{r_{12}}+\frac{[M_3]}{r_{13}}+\frac{[M_4]}{r_{14}}\right\} \\
[M_2]\cdot\left\{\frac{[M_1]}{r_{12}\cdot r_{31}\cdot r_{41}}+\frac{[M_2]}{r_{12}\cdot r_{32}\cdot r_{42}}+\frac{[M_3]}{r_{13}\cdot r_{32}\cdot r_{43}}+\frac{[M_4]}{r_{14}\cdot r_{42}\cdot r_{34}}\right\}\cdot\left\{\frac{[M_1]}{r_{21}}+[M_2]+\frac{[M_3]}{r_{23}}+\frac{[M_4]}{r_{24}}\right\} \\
[M_3]\cdot\left\{\frac{[M_1]}{r_{21}\cdot r_{13}\cdot r_{41}}+\frac{[M_2]}{r_{12}\cdot r_{23}\cdot r_{42}}+\frac{[M_3]}{r_{13}\cdot r_{23}\cdot r_{43}}+\frac{[M_4]}{r_{14}\cdot r_{24}\cdot r_{43}}\right\}\cdot\left\{\frac{[M_1]}{r_{31}}+\frac{[M_2]}{r_{32}}+[M_3]+\frac{[M_4]}{r_{34}}\right\} \\
[M_4]\cdot\left\{\frac{[M_1]}{r_{21}\cdot r_{31}\cdot r_{14}}+\frac{[M_2]}{r_{12}\cdot r_{32}\cdot r_{24}}+\frac{[M_3]}{r_{13}\cdot r_{23}\cdot r_{34}}+\frac{[M_4]}{r_{14}\cdot r_{24}\cdot r_{34}}\right\}\cdot\left\{\frac{[M_1]}{r_{41}}+\frac{[M_2]}{r_{42}}+\frac{[M_3]}{r_{43}}+[M_4]\right\}
\end{pmatrix} \tag{2-5}$$

Where *dM* is defined as:

<<N17>>

$$dM = \left(d[M_1] \quad d[M_2] \quad d[M_3] \quad d[M_4]\right)^T$$

And α is a function:

$$\alpha = f\left([M_1 \cdot], k_{11}, k_{22}, k_{33}, k_{44}, [M_1], [M_2], [M_3], [M_4]\right)$$

If extent of reaction is introduced as a variable, equations (2-5) could be simplified and expressed without time parameter *t*, for all vector components have a factor of function α which could be eliminated. Take function α as a normalized coefficient, there's no need to derive its explicit expression, because formula (2-5) can be applied to program directly without the explicit form of α. So the further discussion about α isn't necessary.

The steady-status assumption of Valvassori-Sartori [1] is given below.

$$\begin{aligned} R_{12} = R_{21} \quad R_{13} = R_{31} \quad R_{14} = R_{41} \\ R_{23} = R_{32} \quad R_{24} = R_{42} \\ R_{34} = R_{43} \end{aligned} \tag{2-6}$$

Under this assumption, the instantaneous composition of the quaternary copolymer in vector form are obtained:

$$\frac{dM}{dt} = \alpha \cdot \begin{pmatrix} [M_1] \cdot \left\{ [M_1] + \frac{[M_2]}{r_{12}} + \frac{[M_3]}{r_{13}} + \frac{[M_4]}{r_{14}} \right\} \\ [M_2] \cdot \left(\frac{r_{21}}{r_{12}}\right) \cdot \left\{ \frac{[M_1]}{r_{21}} + [M_2] + \frac{[M_3]}{r_{23}} + \frac{[M_4]}{r_{24}} \right\} \\ [M_3] \cdot \left(\frac{r_{31}}{r_{13}}\right) \cdot \left\{ \frac{[M_1]}{r_{31}} + \frac{[M_2]}{r_{32}} + [M_3] + \frac{[M_4]}{r_{34}} \right\} \\ [M_4] \cdot \left(\frac{r_{41}}{r_{14}}\right) \cdot \left\{ \frac{[M_1]}{r_{41}} + \frac{[M_2]}{r_{42}} + \frac{[M_3]}{r_{43}} + [M_4] \right\} \end{pmatrix} \tag{2-7}$$

In formula (2-5) and (2-7), the meaning of α remains unchanged, the expressions of α are different.

3. General differential equations of copolymerization of multiple monomers

3.1 Differential equations with differential of time

In order to derive the equations which are easy to be solved by computer for multiple copolymerization, formula (2-5) should be more symmetrical. A new notation is introduced as following.

$$r_{ii} = \frac{k_{ii}}{k_{ii}} = 1, (i = 1,2,3,4)$$

Then formula (2-5) has a novel form:

$$\frac{dM}{dt} = \alpha \cdot \begin{pmatrix} [M_1] \cdot \left\{ \left(\frac{r_{11}}{r_{11}}\right) \cdot \frac{[M_1]}{r_{11} \cdot r_{21} \cdot r_{31} \cdot r_{41}} + \left(\frac{r_{12}}{r_{21}}\right) \cdot \frac{[M_2]}{r_{12} \cdot r_{22} \cdot r_{32} \cdot r_{42}} + \left(\frac{r_{13}}{r_{31}}\right) \cdot \frac{[M_3]}{r_{13} \cdot r_{23} \cdot r_{33} \cdot r_{43}} + \left(\frac{r_{14}}{r_{41}}\right) \cdot \frac{[M_4]}{r_{14} \cdot r_{24} \cdot r_{34} \cdot r_{44}} \right\} \cdot \left\{ \frac{[M_1]}{r_{11}} + \frac{[M_2]}{r_{12}} + \frac{[M_3]}{r_{13}} + \frac{[M_4]}{r_{14}} \right\} \\ [M_2] \cdot \left\{ \left(\frac{r_{21}}{r_{12}}\right) \cdot \frac{[M_1]}{r_{11} \cdot r_{21} \cdot r_{31} \cdot r_{41}} + \left(\frac{r_{22}}{r_{22}}\right) \cdot \frac{[M_2]}{r_{12} \cdot r_{22} \cdot r_{32} \cdot r_{42}} + \left(\frac{r_{23}}{r_{32}}\right) \cdot \frac{[M_3]}{r_{13} \cdot r_{23} \cdot r_{33} \cdot r_{43}} + \left(\frac{r_{24}}{r_{42}}\right) \cdot \frac{[M_4]}{r_{14} \cdot r_{24} \cdot r_{34} \cdot r_{44}} \right\} \cdot \left\{ \frac{[M_1]}{r_{21}} + \frac{[M_2]}{r_{22}} + \frac{[M_3]}{r_{23}} + \frac{[M_4]}{r_{24}} \right\} \\ [M_3] \cdot \left\{ \left(\frac{r_{31}}{r_{13}}\right) \cdot \frac{[M_1]}{r_{11} \cdot r_{21} \cdot r_{31} \cdot r_{41}} + \left(\frac{r_{32}}{r_{23}}\right) \cdot \frac{[M_2]}{r_{12} \cdot r_{22} \cdot r_{32} \cdot r_{42}} + \left(\frac{r_{33}}{r_{33}}\right) \cdot \frac{[M_3]}{r_{13} \cdot r_{23} \cdot r_{33} \cdot r_{43}} + \left(\frac{r_{34}}{r_{43}}\right) \cdot \frac{[M_4]}{r_{14} \cdot r_{24} \cdot r_{34} \cdot r_{44}} \right\} \cdot \left\{ \frac{[M_1]}{r_{31}} + \frac{[M_2]}{r_{32}} + \frac{[M_3]}{r_{33}} + \frac{[M_4]}{r_{34}} \right\} \\ [M_4] \cdot \left\{ \left(\frac{r_{41}}{r_{14}}\right) \cdot \frac{[M_1]}{r_{11} \cdot r_{21} \cdot r_{31} \cdot r_{41}} + \left(\frac{r_{42}}{r_{24}}\right) \cdot \frac{[M_2]}{r_{12} \cdot r_{22} \cdot r_{32} \cdot r_{42}} + \left(\frac{r_{43}}{r_{34}}\right) \cdot \frac{[M_3]}{r_{13} \cdot r_{23} \cdot r_{33} \cdot r_{43}} + \left(\frac{r_{44}}{r_{44}}\right) \cdot \frac{[M_4]}{r_{14} \cdot r_{24} \cdot r_{34} \cdot r_{44}} \right\} \cdot \left\{ \frac{[M_1]}{r_{41}} + \frac{[M_2]}{r_{42}} + \frac{[M_3]}{r_{43}} + \frac{[M_4]}{r_{44}} \right\} \end{pmatrix} \tag{3-1}$$

Based on the above formulas, the general differential equations of copolymerization of multiple monomers are discussed in detail in the following.

In copolymerization of multiple monomers, ignoring the penultimate effect, there are n reactions of chain initiation, n^2 reactions of chain propagation and $n(n-1)$ reactions of chain termination. There are $n(n-1)$ reactivity ratios which are shown in a matrix:

$$r = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1j} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2j} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} & \cdots & r_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nj} & \cdots & r_{nn} \end{pmatrix} \tag{3-2}$$

Where $r_{ij} = \dfrac{k_{ii}}{k_{ij}}$, $i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., n\}$.

k_{ij} is defined as above, if $i = j$, $r_{ij} = 1$.

Under the steady-status assumption by Alfrey-Goldfinger[1], the transient composition of the copolymer of multiple monomers satisfy the following equations:

$$\frac{dM}{dt} = \alpha \cdot \begin{pmatrix} [M_1]\cdot\left\{\left(\frac{r_{11}}{r_{11}}\right)\cdot\frac{[M_1]}{\prod_{i=1}^{n} r_{i1}} + \left(\frac{r_{12}}{r_{21}}\right)\cdot\frac{[M_2]}{\prod_{i=1}^{n} r_{i2}} + \cdots + \left(\frac{r_{1j}}{r_{j1}}\right)\cdot\frac{[M_j]}{\prod_{i=1}^{n} r_{ij}} + \cdots + \left(\frac{r_{1n}}{r_{n1}}\right)\cdot\frac{[M_n]}{\prod_{i=1}^{n} r_{in}}\right\}\cdot\left\{\frac{[M_1]}{r_{11}} + \frac{[M_2]}{r_{12}} + \cdots + \frac{[M_j]}{r_{1j}} + \cdots + \frac{[M_n]}{r_{1n}}\right\} \\ [M_2]\cdot\left\{\left(\frac{r_{21}}{r_{12}}\right)\cdot\frac{[M_1]}{\prod_{i=1}^{n} r_{i1}} + \left(\frac{r_{22}}{r_{22}}\right)\cdot\frac{[M_2]}{\prod_{i=1}^{n} r_{i2}} + \cdots + \left(\frac{r_{2j}}{r_{j2}}\right)\cdot\frac{[M_j]}{\prod_{i=1}^{n} r_{ij}} + \cdots + \left(\frac{r_{2n}}{r_{n2}}\right)\cdot\frac{[M_n]}{\prod_{i=1}^{n} r_{in}}\right\}\cdot\left\{\frac{[M_1]}{r_{21}} + \frac{[M_2]}{r_{22}} + \cdots + \frac{[M_j]}{r_{2j}} + \cdots + \frac{[M_n]}{r_{2n}}\right\} \\ \cdots\cdots \\ [M_k]\cdot\left\{\left(\frac{r_{k1}}{r_{1k}}\right)\cdot\frac{[M_1]}{\prod_{i=1}^{n} r_{i1}} + \left(\frac{r_{k2}}{r_{2k}}\right)\cdot\frac{[M_2]}{\prod_{i=1}^{n} r_{i2}} + \cdots + \left(\frac{r_{kj}}{r_{jk}}\right)\cdot\frac{[M_j]}{\prod_{i=1}^{n} r_{ij}} + \cdots + \left(\frac{r_{kn}}{r_{nk}}\right)\cdot\frac{[M_n]}{\prod_{i=1}^{n} r_{in}}\right\}\cdot\left\{\frac{[M_1]}{r_{k1}} + \frac{[M_2]}{r_{k2}} + \cdots + \frac{[M_j]}{r_{kj}} + \cdots + \frac{[M_n]}{r_{kn}}\right\} \\ \cdots\cdots \\ [M_n]\cdot\left\{\left(\frac{r_{n1}}{r_{1n}}\right)\cdot\frac{[M_1]}{\prod_{i=1}^{n} r_{i1}} + \left(\frac{r_{n2}}{r_{2n}}\right)\cdot\frac{[M_2]}{\prod_{i=1}^{n} r_{i2}} + \cdots + \left(\frac{r_{nj}}{r_{jn}}\right)\cdot\frac{[M_j]}{\prod_{i=1}^{n} r_{ij}} + \cdots + \left(\frac{r_{nn}}{r_{nn}}\right)\cdot\frac{[M_n]}{\prod_{i=1}^{n} r_{in}}\right\}\cdot\left\{\frac{[M_1]}{r_{n1}} + \frac{[M_2]}{r_{n2}} + \cdots + \frac{[M_j]}{r_{nj}} + \cdots + \frac{[M_n]}{r_{nn}}\right\} \end{pmatrix} \tag{3-3}$$

Where dM is defined as:

$$dM = \left(d[M_1] \quad d[M_2] \quad \ldots \quad d[M_k] \quad \ldots \quad d[M_n]\right)^T$$

Therefore, the final form of the equations could be:

$$\frac{d[M_k]}{dt} = \alpha \cdot [M_k] \cdot \left(\sum_{j=1}^{n}\left(\frac{r_{kj}}{r_{jk}} \cdot \frac{[M_j]}{\prod_{i=1}^{n} r_{ij}}\right)\right) \cdot \left(\sum_{j=1}^{n}\frac{[M_j]}{r_{kj}}\right), (k = 1,2,...,\mathrm{n}) \tag{3-4}$$

Based on the steady-status assumption of Valvassori-Sartori[1], a new form of formula (2-7) is generated below:

<<N17>>

$$\frac{dM}{dt}=\alpha\cdot\begin{pmatrix}[M_1]\cdot\left(\frac{r_{11}}{r_{11}}\right)\cdot\left\{\frac{[M_1]}{r_{11}}+\frac{[M_2]}{r_{12}}+\frac{[M_3]}{r_{13}}+\frac{[M_4]}{r_{14}}\right\}\\ [M_2]\cdot\left(\frac{r_{21}}{r_{12}}\right)\cdot\left\{\frac{[M_1]}{r_{21}}+\frac{[M_2]}{r_{22}}+\frac{[M_3]}{r_{23}}+\frac{[M_4]}{r_{24}}\right\}\\ [M_3]\cdot\left(\frac{r_{31}}{r_{13}}\right)\cdot\left\{\frac{[M_1]}{r_{31}}+\frac{[M_2]}{r_{32}}+\frac{[M_3]}{r_{33}}+\frac{[M_4]}{r_{34}}\right\}\\ [M_4]\cdot\left(\frac{r_{41}}{r_{14}}\right)\cdot\left\{\frac{[M_1]}{r_{41}}+\frac{[M_2]}{r_{42}}+\frac{[M_3]}{r_{43}}+\frac{[M_4]}{r_{44}}\right\}\end{pmatrix} \tag{3-5}$$

Formula (3-5) can be extended to the copolymerization of multiple monomers:

$$\frac{dM}{dt}=\alpha\cdot\begin{pmatrix}[M_1]\cdot\left(\frac{r_{11}}{r_{11}}\right)\cdot\left\{\frac{[M_1]}{r_{11}}+\frac{[M_2]}{r_{12}}+\ldots+\frac{[M_j]}{r_{1j}}+\ldots+\frac{[M_n]}{r_{1n}}\right\}\\ [M_2]\cdot\left(\frac{r_{21}}{r_{12}}\right)\cdot\left\{\frac{[M_1]}{r_{21}}+\frac{[M_2]}{r_{22}}+\ldots+\frac{[M_j]}{r_{2i}}+\ldots+\frac{[M_n]}{r_{2n}}\right\}\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ [M_k]\cdot\left(\frac{r_{k1}}{r_{1k}}\right)\cdot\left\{\frac{[M_1]}{r_{k1}}+\frac{[M_2]}{r_{k2}}+\ldots+\frac{[M_j]}{r_{kj}}+\ldots+\frac{[M_n]}{r_{kn}}\right\}\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ [M_n]\cdot\left(\frac{r_{n1}}{r_{1n}}\right)\cdot\left\{\frac{[M_1]}{r_{n1}}+\frac{[M_2]}{r_{n2}}+\ldots+\frac{[M_j]}{r_{nj}}+\ldots+\frac{[M_n]}{r_{nn}}\right\}\end{pmatrix} \tag{3-6}$$

The final general differential equations are:

$$\frac{d[M_k]}{dt}=\alpha\cdot[M_k]\cdot\left(\frac{r_{k1}}{r_{1k}}\right)\cdot\left(\sum_{j=1}^{n}\frac{[M_j]}{r_{kj}}\right),(k=1,2,\ldots,n) \tag{3-7}$$

3.2 Differential equations with differential of conversion ratio

The conversion ratio C is defined as follows:

$$C=\frac{\sum_{i=1}^{n}[M_i^0]-\sum_{i=1}^{n}[M_i]}{\sum_{i=1}^{n}[M_i^0]} \tag{3-8}$$

Where, $[M_i^0]$ stands for the initial value of monomer $[M_i]$. So the differential of conversion ratio C is:

$$dC=-\frac{d\sum_{i=1}^{n}[M_i]}{\sum_{i=1}^{n}[M_i^0]} \tag{3-9}$$

To simplify the expression, a function g_k is introduced and defined below:

<<N17>>

$$g_k = [M_k] \cdot \left(\sum_{j=1}^{n} \left(\frac{r_{kj}}{r_{jk}} \cdot \frac{[M_j]}{\prod_{i=1}^{n} r_{ij}} \right) \right) \cdot \left(\sum_{j=1}^{n} \frac{[M_j]}{r_{kj}} \right), (k = 1,2,\cdots,n) \tag{3-10}$$

The formula (3-4) is simplified as:

$$\frac{d[M_k]}{dt} = \alpha g_k \tag{3-11}$$

Therefore,

$$dC = -\frac{d\sum_{i=1}^{n}[M_i]}{\sum_{i=1}^{n}[M_i^0]} = -\frac{\sum_{i=1}^{n} d[M_i]}{\sum_{i=1}^{n}[M_i^0]} = -\frac{\alpha dt \sum_{i=1}^{n} g_i}{\sum_{i=1}^{n}[M_i^0]} \tag{3-12}$$

$$dt = -\frac{dC\sum_{i=1}^{n}[M_i^0]}{\alpha\sum_{i=1}^{n} g_i} \tag{3-13}$$

Replace *dt* in (3-11) with (3-13), differential equations with variable *dC* is obtained and no function α is needed:

$$\frac{d[M_k]}{dC} = -\left(\sum_{i=1}^{n}[M_i^0] \right) \frac{g_k}{\sum_{i=1}^{n} g_i} \tag{3-14}$$

3.3 Verification of the differential equations

If *n* equals to two, the differential equations (3-4) become the following equation, which is the equation (1-1) of the binary copolymerization given in the reference [1].

$$\begin{pmatrix} \frac{d[M_1]}{dt} \\ \frac{d[M_2]}{dt} \end{pmatrix} = \alpha \cdot \begin{pmatrix} [M_1] \cdot (r_{12}[M_1] + [M_2]) \\ [M_2] \cdot ([M_1] + r_{21}[M_2]) \end{pmatrix}$$

If *n* equals to three, the differential equations become the following equations, which is also the same as equation (1-2) of the ternary copolymerization given in the reference [1].

$$\begin{pmatrix} \frac{d[M_1]}{dt} \\ \frac{d[M_2]}{dt} \\ \frac{d[M_3]}{dt} \end{pmatrix} = \alpha \cdot \begin{pmatrix} [M_1] \cdot \left(\frac{[M_1]}{r_{31}r_{21}} + \frac{[M_2]}{r_{21}r_{32}} + \frac{[M_3]}{r_{13}r_{23}} \right) \cdot \left([M_1] + \frac{[M_2]}{r_{12}} + \frac{[M_3]}{r_{13}} \right) \\ [M_2] \cdot \left(\frac{[M_1]}{r_{12}r_{31}} + \frac{[M_2]}{r_{12}r_{32}} + \frac{[M_3]}{r_{32}r_{13}} \right) \cdot \left(\frac{[M_1]}{r_{21}} + [M_2] + \frac{[M_3]}{r_{23}} \right) \\ [M_3] \cdot \left(\frac{[M_1]}{r_{13}r_{21}} + \frac{[M_2]}{r_{23}r_{12}} + \frac{[M_3]}{r_{13}r_{23}} \right) \cdot \left(\frac{[M_1]}{r_{31}} + \frac{[M_2]}{r_{32}} + [M_3] \right) \end{pmatrix}$$

Of course, for n equals to four, it becomes the differential equations of the quaternary copolymerization derived in this paper.

4. Numerical solution of the differential equations

The equations (3-4) and (3-7) are suitable for using computer to solve. For multiple copolymerization, $n(n\text{-}1)$ reactivity ratio parameters are needed, but the experimental data of each reactivity ratio are rarely measured completely. In order to ensure the applicability of the calculations, it is necessary to supplement the missing reactivity ratios by using the *Q-e* formula of Alfrey-Price[1]:

$$r_{ij} = \left(\frac{Q_i}{Q_j}\right)\cdot \exp\left(-e_i\left(e_i - e_j\right)\right) \tag{4-1}$$

Where:

Q: conjugative effect parameter of monomer M_i

e: polarity effect parameter of monomer M_i

Explicit Euler method[3] is applied to the differential equations (3-4) and equations (3-7) to get the numerical solution. The general form of the recursive formula is:

$$\begin{aligned} \vec{y}' &= \vec{f}\left(t, \vec{y}(t)\right), \vec{y}(t_0) = \vec{y}_0 \\ \vec{y}_{n+1} &= \vec{y}_n + h\cdot \vec{f}\left(t_n, \vec{y}_n\right) \end{aligned} \tag{4-2}$$

The algorithm is written in C++ program language, and is compiled by BCB6.0 which gives the graphical result.

The C++ codes for equations (3-7) are listed as follows (Please refer to the appendix for define of class and other detail):

```
void TCopolymerization::Get_Distribution_Alfrey_Goldfinger(matrix& ans)
{                                   //ans is a data storage container, store in matrix form
Vector dM(N);
Vector M(M0); M/=M.Mean()*N;    //The feeding molecular ratio is normalized to a percentage
double dC=1e-4;                     //Monomer conversion ratio differential (integration step)
ans.Set(1,N+1); ans.DelLine(1) ;
for(double C=0; C<=1-dC; C+=dC){    //Extent of reaction from 0% -100%
        for(int k=1; k<=N; k++){
                double A(0),B(0);
                for(int j=1;j<=N;j++){
                        double D(1);
                        for(int i=1;i<=N;i++)
                                D*=r(i,j);
                        A+=(r(k,j)/r(j,k))*M[j]/D;
```

```
                    B+=M[j]/r(k,j);
                    }
                dM[k]=M[k]*A*B;
                }
        dM/=dM.Mean()*N; //When a certain degree of polymerization is present, the
                        //proportion of newly produced polymer is normalized to a percentage
        Vector tmp=dM;
        tmp.Ins(C,1);
        ans.AddLine(tmp);
        dM*=dC;
        M-=dM;
        }
// For more intuitive display image, deal with ans
int n=ans.ColumSize();
for(int j=1;j<=ans.LineSize();j++)
        for(int i=3;i<=n;i++)
                ans(j,i)+=ans(j,i-1);
}
```

The C ++ language fragment for equations (3-7) is as follows:

```
void TCopolymerization::Get_Distribution_Valvassori_Sartori(matrix& ans)
{
Vector dM(N);
Vector M(M0); M/=M.Mean()*N;    //The feeding molecular ratio is normalized to a percentage
double dC=1e-4;                 //Monomer conversion ratio differential (integration step)
ans.Set(1,N+1); ans.DelLine(1) ;
for(double C=0; C<=1-dC; C+=dC){
        for(int k=1; k<=N; k++){
                double A(0);
                for(int j=1; j<=N; j++)
                        A+=M[j]/r(k,j);
                dM[k]=M[k]*(r(k,1)/r(1,k))*A;
                }
        dM/=dM.Mean()*N; //When a certain degree of polymerization is present,
                        //the proportion of newly produced polymer is normalized to a percentage
        Vector tmp=dM;
        tmp.Ins(C,1);
        ans.AddLine(tmp);
        dM*=dC;
        M-=dM;
```

```
        }
// For more intuitive display image, deal with ans
int n=ans.ColumSize();
for(int j=1;j<=ans.LineSize();j++)
        for(int i=3;i<=n;i++)
                ans(j,i)+=ans(j,i-1);
}
```

5. Analysis and verification of the results

5.1 Compare numerical solution with analytic solution

The differential equation of binary copolymerization has analytic solution. So, it is suitable for studying the difference between numerical solution and analytic solution.

Comparing the numerical solution from the above algorithm with the analytical solution of reference [1] under the following conditions:

1) molecular ratio of butadiene and styrene is 1:1

2) set parameters of reactivity ratio r_{12} = 1.35 and r_{21} = 0.58

The result is illustrated in fig.5-1.

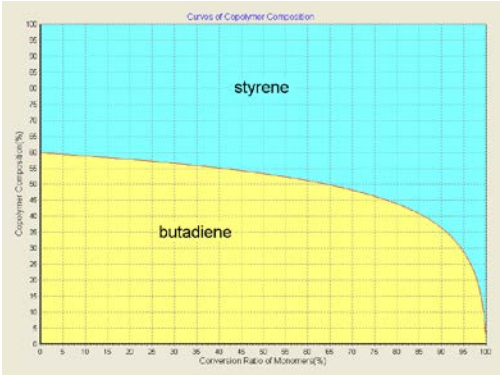

Fig.5-1 Numerical Solution of Copolymerization between Butadiene and Styrene

The error of the explicit Euler method is shown in table 5-1.

Table 5-1 Error of Explicit Euler Method

Conversion ratio	styrene ratio				error (dC=0.0001)
	dC=0.01	dC=0.001	dC=0.0001	Analytic solution	
0	0.59796437	0.59796437	0.597964377	0.597964377	0
0.1	0.58849400	0.588535113	0.588539213	0.588539623	4.09861E-07
0.2	0.57771196	0.577806608	0.577816044	0.577816987	9.43267E-07

0.3	0.56524109	0.565407549	0.565424138	0.565425796	1.65833E-06
0.4	0.55052163	0.550788204	0.550814757	0.550817411	2.6543E-06
0.5	0.53267358	0.533087106	0.533128269	0.533132384	4.11444E-06
0.6	0.51020092	0.51084643	0.510910619	0.510917035	6.41529E-06
0.7	0.48025656	0.481311126	0.481415805	0.481426265	1.04601E-05
0.8	0.43635534	0.438281832	0.438472363	0.438491395	1.9032E-05
0.9	0.35759456	0.362333766	0.362797168	0.362843404	4.62359E-05

The cumulative error is shown in Fig.5-2, the effect of step of calculation on accuracy is shown in Fig.5-3.

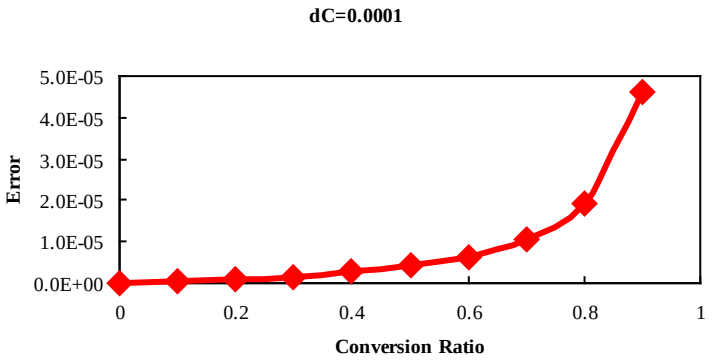

Fig. 5-2 Cumulative Error

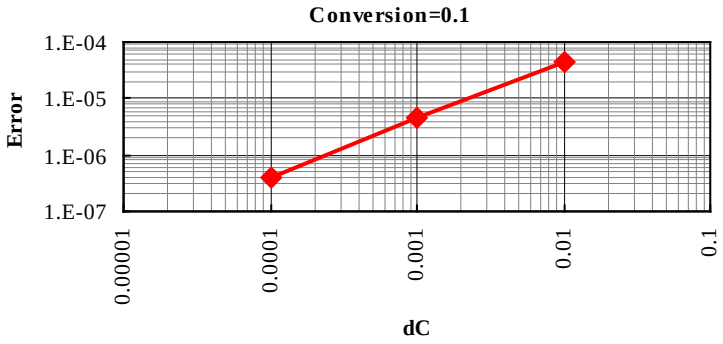

Fig. 5-3 Effect of Step Size on Accuracy

As we see from above, although the explicit Euler method has only order of accuracy 1, if the integral step size is less than 0.0001, the error will be less than 0.00005. It is enough for the application.

5.2 Verification of hexahydric copolymerization

The hexahydric copolymerization behavior is tested by the proposed algorithm. Curve in the figure (5-4) shows the relationship between the polymer composition and the conversion ratio. The six monomers are butadiene, styrene, methyl methacrylate, methyl acrylate, acrylonitrile and diethyl fumarate separately, and their feeding molecular ratio is 1:1:1:1:1:1.

Shown in figure 5-4 are the results under the steady-status assumption of Alfrey-Goldfinger.

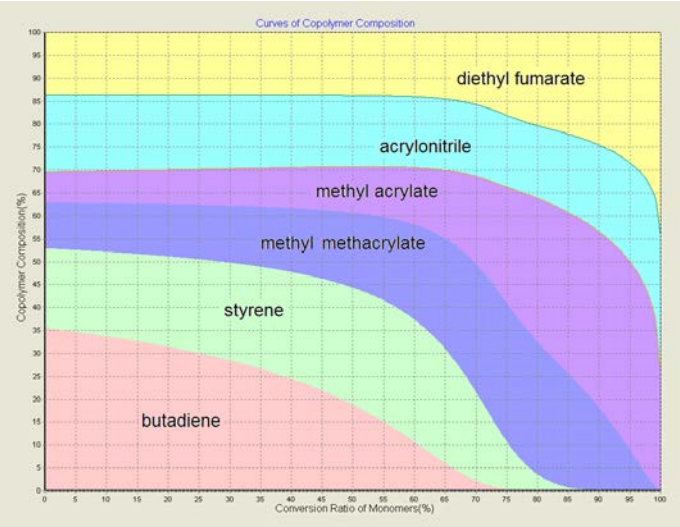

Fig.5-4 Numerical Solution Based on Alfrey-Goldfinger Steady-status Assumption

From this figure, it is known that:

1) Butadiene has been almost completely consumed in the extent of reaction of about 70%;

2) Styrene has been almost completely consumed in the extent of reaction of 80%, when the reaction proceeds into the quaternary copolymerization stage;

3) The content of diethyl fumarate increases rapidly in the polymer after the extent of reaction reaches 70%;

4) In the final stage of the reaction, there are still traces of diethyl fumarate which remains unreacted;

5) The content of acrylonitrile in the whole reaction stage is relatively constant.

Using the same conditions, the results under the steady-status assumption of Valvassori-Sartori are as follows:

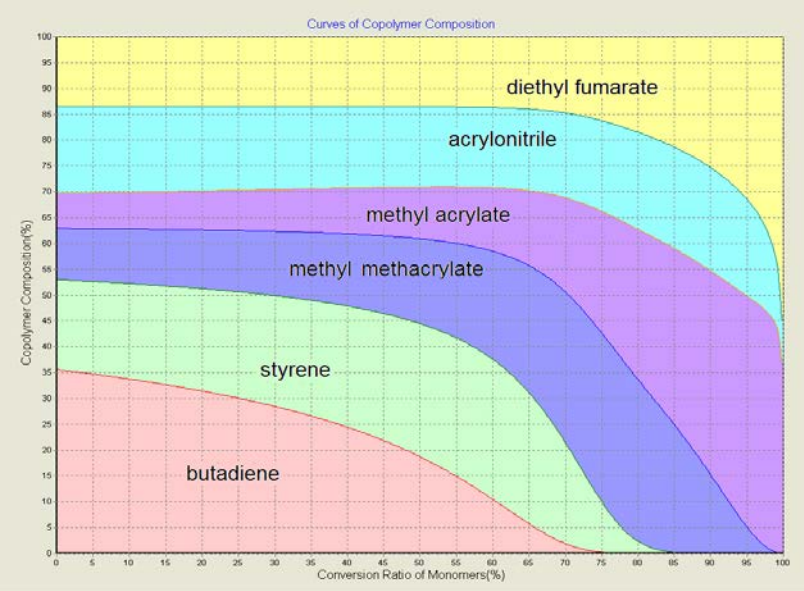

Fig.5-5 Numerical Solution Based on Valvassori-Sartori Steady-status Assumption

6. Conclusion

A general differential equations of copolymerization of multiple monomers is derived which could provide a theoretical guidance for industrial production and academic research of multiple free radical copolymerization. The C++ realization for solving the numerical solution of the equations is given. Accuracy of numerical solution is evaluated and a hexahydric copolymerization demonstration is given to verify the validity of the proposed algorithm.

7. Acknowledgements

First of all, we would like to thank our instructor, Qi Yachao, who has put forward valuable suggestions for improvements throughout our research and writing.

Also, we greatly appreciate the support from our two chemistry teachers, Wu Jingquan and Li Shuxia, who have always encouraged us to go further on the path of learning chemistry and led us to do pioneering thinking and development thinking, so that we now have the power to complete this paper.

We would also like to thank everyone who helped us, supported us and advised us on this paper (our parents, classmates and other teachers), this work could not have been done without you.

8. References

[1] Pan Zuren. Polymer Chemistry. Chemical Industry Press, July 2014

[2] Zhang Pingwen et al. Numerical Analysis. Peking University Press, April 2015

[3] Su Tashan. VC++ and BC++ Numerical Analysis Class Library. Tsinghua University Press, November 2005

9. Appendix

Code 1

```
// The class of free radical copolymerization
//file name: Copolymerization.h
#ifndef COPOLMERIZATION_H
#define COPOLMERIZATION_H
#include“Matrix.h”              //class vector and matrix are in it
typedef    vector Vector;
#include<vector.h>              //container of objects
#include<math.h>
class TCopolymerization              //class name
{
private:
            int                 N;          //kinds of monomers
      std::vector<String>       Name;       //names of monomers
            Vector              Q;          //conjugative effect parameter
            Vector              e;          //polarity effect parameter
            matrix              r0;         //matrix of Incompleter
            Vector              M0;         //initial value of molecular ratio
            matrix              r;          //matrix of complete reactivity ratio
            void Cal_rij();
            void fr(double q1,double e1,double q2,double e2,double& r12,double& r21);
                                                                        // tool function
public:
            void Set_Name(const std::vector<String>
            _Name);
            void Set_Q_e ( Vector& _Q,    Vector& _e);
            void Set_rij0( matrix& _r0);
            void Set_M0    ( Vector& _M0);
            void Get_Distribution_Alfrey_Goldfinger (matrix& ans);    //Core function
            void Get_Distribution_Valvassori_Sartori     (matrix& ans);    //Core function
};
//---------------------------------------------------------------------------

void TCopolymerization::Set_Name(const std::vector<String> _Name){Name=_Name;}
void TCopolymerization::Set_Q_e ( Vector& _Q, Vector& _e){Q=_Q; e=_e;}
void TCopolymerization::Set_rij0( matrix& _r0){r0=_r0; Cal_rij();}
void TCopolymerization::Set_M0    ( Vector& _M0){M0=_M0;}
// Tool function, Calculates the reactivity ratio from the Q-e value:
void TCopolymerization::fr(double q1,double e1,double q2,double e2,double& r12,double& r21)
{
```

<<N17>>

```
r12=q1/q2*exp(-e1*(e1-e2));
r21=q2/q1*exp(-e2*(e2-e1));
}
//-------------------------------------
// Disposal reactivity ratio table:
void TCopolymerization::Cal_rij()
{
N=r0.LineSize();
r.Set(N,N);
for(int i=1;i<=N;i++){
        for(int j=i;j<=N;j++){
                if(i<=r0.LineSize() && j<=r0.ColumSize() &&r0(i,j)>=0 && i!=j){
                        r(i,j)=r0(i,j);
                        r(j,i)=r0(j,i);
                        }
                else if(i==j)
                        r(i,j)=1.0;
                else {
                        double rij,rji;
                        this->fr(Q[i],e[i],Q[j],e[j],rij,rji);
                        r(i,j)=rij;
                        r(j,i)=rji;
                        }
                }
        }
}
//-------------------------------------------------------------------------

void TCopolymerization::Get_Distribution_Alfrey_Goldfinger(matrix& ans)
{
Vector dM(N);
Vector M(M0); M/=M.Mean()*N; //The feeding molecular ratio is normalized to a percentage
double dC=1e-4;                         //Monomer conversion ratio differential (integration step)
ans.Set(1,N+1); ans.DelLine(1) ;
for(double C=0; C<=1-dC; C+=dC){    //Extent of reaction from 0% -100%；
        for(int k=1; k<=N; k++){
                double A(0),B(0);
                for(int j=1;j<=N;j++){
                        double D(1);
                        for(int i=1;i<=N;i++)
                                D*=r(i,j);
                        A+=(r(k,j)/r(j,k))*M[j]/D;
                        B+=M[j]/r(k,j);
                        }
```

```
                dM[k]=M[k]*A*B;
                }
        dM/=dM.Mean()*N;      // When a certain degree of polymerization is present,
            //the proportion of newly produced polymer is normalized to a percentage
        Vector tmp=dM;
        tmp.Ins(C,1);
        ans.AddLine(tmp);
        dM*=dC;
        M-=dM;
        }
// For more intuitive display image, deal with ans
int n=ans.ColumSize();
for(int j=1;j<=ans.LineSize();j++)
        for(int i=3;i<=n;i++)
                ans(j,i)+=ans(j,i-1);
}
//---------------------------------------------------------------------

void TCopolymerization::Get_Distribution_Valvassori_Sartori(matrix& ans)
{
Vector dM(N);
Vector M(M0); M/=M.Mean()*N; //The feeding molecular ratio is normalized to a percentage
double dC=1e-4;                         //Monomer conversion ratio differential (integration step)
ans.Set(1,N+1); ans.DelLine(1) ;
for(double C=0; C<=1-dC; C+=dC){
        for(int k=1; k<=N; k++){
                double A(0);
                for(int j=1; j<=N; j++)
                        A+=M[j]/r(k,j);
                dM[k]=M[k]*(r(k,1)/r(1,k))*A;
                }
        dM/=dM.Mean()*N;      //When a certain degree of polymerization is present,
                //the proportion of newly produced polymer is normalized to a percentage
        Vector tmp=dM;
        tmp.Ins(C,1);
        ans.AddLine(tmp);
        dM*=dC;
        M-=dM;
        }
//For more intuitive display image, deal with ans
int n=ans.ColumSize();
for(int j=1;j<=ans.LineSize();j++)
        for(int i=3;i<=n;i++)
                ans(j,i)+=ans(j,i-1);
```

```
}
//---------------------------------------------------------------------
#endif
```

Code 2

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <Chart.hpp>
#include <ExtCtrls.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
#include <Grids.hpp>
#include "Matrix.h"
#include " Copolymerization.h"
//---------------------------------------------------------------------------
const Mmax=6;       //The maximum number of lines to display
const Mlimited=10;//The maximum number for monomer types of the multiple copolymerization,
                        //it can take a larger value
class TeQ;
class TMMR;
class TForm1 : public TForm
{
__published:   // IDE-managed Components
        TChart *Chart1;
        TLineSeries *Series1;
        TLineSeries *Series2;
        TLineSeries *Series3;
        TLineSeries *Series4;
        TLineSeries *Series5;
        TLineSeries *Series6;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TComboBox *ComboBox1;
        void __fastcall FormCreate(TObject *Sender);
```

```
        void __fastcall FormDestroy(TObject *Sender);
        void __fastcall ComboBox1Change(TObject *Sender);
        void __fastcall ComboBox1Exit(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
private: // User declarations
        int N;
        std::vector<TeQ> eQ;        //Alternative monomers and their e-Q tables
        std::vector<TMMR> R;        //Reactivity ratio summary table
        String myNames[Mmax];
        double r[Mmax][Mmax];
        double e[Mmax];
        double Q[Mmax];
        double M0[Mmax];
        TEdit **edtName;
        TEdit **edtM0;
        TEdit ***edtr;
        bool IsEdtNameShow;
public:         // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
class TeQ // The class for storing Q-e values
{
public: String Name;
        double e;
        double Q;
};
//---------------------------------------------------------------------------
class TMMR //The class for storing reactivity ratio data
{
public: String M1;
        String M2;
        double r;
        String Note;
};
//---------------------------------------------------------------------------
// Converts an array of characters to a string:
String PCharToStr(char* chr)
{
String ans;
char* p=chr;
while(*p!=0){ans+=*p;p++;}
```

```
return ans;
}
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Code 3

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include<fstream.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------

// Initialization processing:
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
N=0;
edtName=new TEdit*[Mmax+1];
edtM0    =new TEdit*[Mmax+1];
for(int i=0;i<Mmax+1;i++){
        edtName[i]=new TEdit(this);
        edtName[i]->Parent=this;
        edtName[i]->Left=5;
        edtName[i]->Width=100;
        edtName[i]->Height=20;
        edtName[i]->Visible=false;

        edtM0[i]=new TEdit(this);
        edtM0[i]->Parent=this;
        edtM0[i]->Left=5+100;
        edtM0[i]->Width=100;
        edtM0[i]->Height=20;
        edtM0[i]->Visible=false;
        }

//Begin:
int rsize=Mmax+1;
edtr=new TEdit **[rsize];
```

```
for(int i=0;i<rsize;i++){
        edtr[i]=new TEdit*[rsize];
        for(int j=0;j<rsize;j++){
                edtr[i][j]=new TEdit(this);
                edtr[i][j]->Parent=this;
                edtr[i][j]->Left=j*30+5;
                edtr[i][j]->Width=30;
                edtr[i][j]->Height=20;
                edtr[i][j]->Visible=false;
                if(i==j) {edtr[i][j]->Text="1";    edtr[i][j]->Enabled=false;}
                if(i==0) {edtr[i][j]->Text="r?"+IntToStr(j);    edtr[i][j]->Enabled=false;}
                if(j==0) {edtr[i][j]->Text="r"+IntToStr(i)+"?";
                        edtr[i][j]->Enabled=false;}
                if(i==0&&j==0) {   edtr[i][j]->Text="竞聚率表";
                                edtr[i][j]->Enabled=false;}
                }
        }
this->ComboBox1->Width    =edtName[0]->Width;
this->ComboBox1->Height =edtName[0]->Height;
this->ComboBox1->Left      =edtName[0]->Left;
// Read the e-Q value:
ifstream infile;
infile.open(".\\e-Q 值表.txt");
char pchar[100];
double tmp1,temp2;
TeQ eQi;
infile>>pchar>>pchar>>pchar;
while(!infile.eof()){
        infile>>pchar>>eQi.e>>eQi.Q;
        eQi.Name=PCharToStr(pchar);
        this->ComboBox1->Items->Add(eQi.Name);
        this->eQ.push_back(eQi);
        }
infile.close();
//Read the table of reactivity ratio rij:
ifstream infile2;
infile2.open(".\\竞聚率表.txt");
char pchar1[100],pchar2[100],pchar3[100];
infile2>>pchar1>>pchar1>>pchar1>>pchar3;
R.clear();
TMMR mmr;
while(!infile2.eof()){
        infile2>>pchar1>>pchar2>>mmr.r>>pchar3;
        mmr.M1=PCharToStr(pchar1);
```

```
            mmr.M2=PCharToStr(pchar2);
            mmr.Note=PCharToStr(pchar3);
            R.push_back(mmr);
            }
infile2.close();
this->IsEdtNameShow=false;
this->Chart1->Width=this->Width-this->Chart1->Left-20;
this->Chart1->Height=this->Height-this->Chart1->Top-50;
}
//---------------------------------------------------------------------------

//Add polymerized monomers:
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if(N==Mmax)return;
this->ComboBox1->Visible=true;
edtName[N+1]->Visible=false;
//Hide edtr:
if(this->IsEdtNameShow){
            for(int i=0;i<=N;i++)
                        for(int j=0;j<=N;j++)
                                    edtr[i][j]->Visible=false;
            this->IsEdtNameShow=false;
            }
//Display interface:
if(N==0){
            edtName[N]->Text="单体名称：       ";
            edtName[N]->Top=50;
            edtName[N]->Enabled=false;
            edtName[N]->Visible=true;
            edtM0[N]->Text="Mole 数：";
            edtM0[N]->Top=edtName[N]->Top;    // Both alignment
            edtM0[N]->Enabled=false;
            edtM0[N]->Visible=true;
            }
if(N>0 && edtName[N]->Text=="") return;
edtName[N]->Visible=true;
if(N<Mmax){
            edtName[N]->Enabled=false;
            N++;
            edtName[N]->Top=edtName[N-1]->Top+edtName[N-1]->Height;
            edtM0[N]->Top=edtM0[N-1]->Top+edtM0[N-1]->Height;
            edtM0[N]->Visible=true;
            }
```

```
this->ComboBox1->Top=edtM0[N]->Top;
this->ComboBox1->Text="";
this->ComboBox1->Visible=true;
this->ComboBox1->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
this->ComboBox1->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormDestroy(TObject *Sender)
{
for(int i=0;i<Mmax+1;i++){
        delete edtName[i];
        delete edtM0[i];
        }
delete edtName;
delete edtM0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
if(this->ComboBox1->Text=="") return;
for(int i=1;i<N;i++){
        if(this->ComboBox1->Text==edtName[i]->Text){
                this->ComboBox1->Text="";
                return;
                }
        }
edtName[N]->Text=this->ComboBox1->Text;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ComboBox1Exit(TObject *Sender)
{
if(this->ComboBox1->Text=="") return;
for(int i=1;i<N;i++){
        if(this->ComboBox1->Text==edtName[i]->Text){
                ShowMessage("输入重复！ ");
                this->ComboBox1->Text="";
```

```
                this->ComboBox1->Focused();
                return;
                }
        }
edtName[N]->Text=this->ComboBox1->Text;
}
//---------------------------------------------------------------------------

// Upon completion of data entry, data processing begins:
void __fastcall TForm1::Button2Click(TObject *Sender)
{
// Check for errors and warnings:
if(N==0||N==1){
        ShowMessage("错误：共聚单体少于 2 种！");
        return;
        }
if(edtName[N]->Text==""){
        ShowMessage("请完成第"+IntToStr(N)+"单体的选择！");
        return;
        }
//Display edtr:
if(!this->IsEdtNameShow){
        for(int i=0;i<=N;i++){
                for(int j=0;j<=N;j++){
                        edtr[i][j]->Top=edtName[N]->Top+30+i*20;
                        edtr[i][j]->Visible=true;
                        }
                }
        this->IsEdtNameShow=true;
        }
//Dispose:
edtName[N]->Text=this->ComboBox1->Text;
this->ComboBox1->Enabled=false;
this->ComboBox1->Visible=false;
edtName[N]->Enabled=false;
edtName[N]->Visible=true;
int num=0;
for(int i=1;i<=N;i++)
        num+=(edtM0[i]->Text=="")?0:1;
if(num<2){ShowMessage("错误：投料单体少于 2 种！"); return;}

//Begin running:
if(num<N)ShowMessage("警告：有"+IntToStr(N-num)+"种单体未投料！");
//Obtain M0，e，Q:
```

```
for(int i=0;i<N;i++){
        if(edtM0[i+1]->Text=="")M0[i]=0;
        else        M0[i]=edtM0[i+1]->Text.ToDouble();
        myNames[i]=edtName[i+1]->Text;
        for(unsigned j=0;j<eQ.size();j++){
                if(myNames[i]==eQ[j].Name){
                        e[i]=eQ[j].e;
                        Q[i]=eQ[j].Q;
                        }
                }
        }
//Check rij:
for(int i=0;i<N;i++){
        for(int j=0;j<N;j++){
                if(i==j){r[i][j]=1;continue;}
                bool finded=false;
                for(unsigned k=0;k<R.size();k++){
                        if(myNames[i]==R[k].M1 && myNames[j]==R[k].M2){
                                r[i][j]=R[k].r;
                                finded=true;
                                break;
                                }
                        }
                if(!finded)r[i][j]=-1;
                if(r[i][j]>=0)edtr[i+1][j+1]->Text=r[i][j];
                }
        }
}
//---------------------------------------------------------------------------

//Calculate the conversion ratio curve data and plot:
void __fastcall TForm1::Button3Click(TObject *Sender)
{
TCopolymerization aa;
Vector ve(N),vQ(N),vM0(N);
for(int i=1;i<=N;i++){
        ve[i]=e[i-1];
        vQ[i]=Q[i-1];
        vM0[i]=M0[i-1];
        }
matrix mr0(N,N);
for(int i=1;i<=N;i++){
        for(int j=1;j<=N;j++) {
                if(edtr[i][j]->Text=="")
```

```
                        mr0(i,j)=-1;
                else    mr0(i,j)=edtr[i][j]->Text.ToDouble();
                }
        }
aa.Set_Q_e(vQ,ve);
aa.Set_rij0(mr0);
aa.Set_M0(vM0);
matrix ans;
//aa.Get_Distribution_Alfrey_Goldfinger(ans);    //Core code, steady-status assumption
                                                        //of Alfrey-Goldfinger
aa.Get_Distribution_Valvassori_Sartori(ans);     //Core code, steady-status assumption
                                                        //of Valvassori-Sartori
int nH=ans.LineSize();
int k=(nH>500)?(nH/500):1;
//Plot:
for(int i=0;i<Mmax;i++){
        if(i<N)   this->Chart1->Series[i]->Title=edtName[i+1]->Text;
        else            this->Chart1->Series[i]->Title=" ";
        }
for(int i=0;i<N;i++)
        this->Chart1->Series[i]->Clear();
for(int i=2;i<=N;i++) {
        for(int j=1;j<=nH;j+=k)
                this->Chart1->Series[i-2]->AddXY(ans(j,1)*100, ans(j,i)*100); //Draw curves
        }
}
//---------------------------------------------------------------------------
```