» A year here and still he dreamed of cyberspace,
hope fading nightly. All the speed he took, all the turns he'd taken
and the corners he'd cut in Night City, and still he'd see the matrix in his sleep,
bright lattices of logic unfolding across that colorless void... «[1]

Appendix: Polymerization Models

Synopsis: This appendix discusses several approaches in the modeling of living radical polymerizations with their specific advantages and disadvantages.

A.1. Numerical Integration of Differential Equations

The reaction scheme of free radical polymerization, living or not, presented in chapter 2 shows that only a relatively small number of different species take part in these reactions: initiator, monomer, transfer agent, radicals, dormant polymer chains and dead polymer material. The latter three types, however, are polymeric species which means that they come in a variety of chainlengths. The way in which the chain length is dealt with constitutes the primary difference between the models described in this appendix. Three approaches can be distinguised.

First, the chainlengths can be completely ignored. Although simulations using such models will not yield any information on the polymer as such, they may be used to illustrate simple kinetic effects as in chapter 2, where it was shown that an additional reaction needs to be invoked to explain the retardation that is observed in RAFT polymerizations. The advantage of such a model is that it is both simple and executes very fast. The number of differential equations can range from about five to ten, depending on how detailed different termination and transfer events are treated. The most important disadvantage is of course that no information is gained on the polymer, other than its concentration (in moles per unit volume).

Second, all chainlengths can be considered individually. This means that for each of the polymeric species (radicals, dormant chains and dead polymer), a large number of differential equations needs to be solved. One for each individual chain-length that exists during the polymerization. The advantage is that the most complete picture of the resulting polymer is obtained. Full molar mass distributions

can be constructed from the data and within these it is possible to locate the dead and dormant materials. The disadvantage is that this approach may be applied only to a limited number of polymerizations. The number of differential equations is about three times larger than the number of chain lengths that is monitored during the polymerization. Uncontrolled free radical polymerizations grow chains of a few thousand repeat units from the start of the reaction resulting in simulations that require far more than 10,000 differential equations to be solved simultaniously. More often than not, these differential equations form a stiff system which rapidly becomes insolvable for any computer as the number of differential equations increases. Conventional free radical polymerizations therefore, cannot be simulated with such an approach on common computers. The situation is completely different for living free radical polymerizations. As outlined in chapter 2 the average chain length is a linear function of conversion and its distribution is of low polydispersity. This means if a reaction is set to produce material of say 150 monomer units, that during the entire reaction no material is formed which would significantly exceed this length. To accomodate material formed by combination – which may be slightly longer – and provide a bit of overhead for the non-monodispersity of the distribution, somewhat more than 450 differential equations are required and the simulation can be executed on a modern desktop computer in a timespan anywhere between a few minutes to a day. The simulations remain restricted however to living systems with a fast equilibrium between growing and dormant chains that aim at producing relatively low molar mass material. In this thesis, such a model is used to investigate the kinetics by matching simulations to molar mass distributions obtained by HPLC that show individual oligomers up to a chain length of about 15 monomer units.

A third method forms a compromise between the abovementioned simulations. It relies on the fact that any distribution can be characterized by a number of moments. The i^{th} moment of the distribution of X (μ_i^X) is defined as follows:

$$\mu_i^X = \sum_{j=0}^{\infty} j^i \cdot X_j \qquad \textbf{(7-1)}$$

in which X_j is the concentration of species X with degree of polymerization i.

The more moments are known, the more accurately a distribution can be reconstructed from these values. The zeroth moment corresponds to the total concentration of a certain species, covering all chain lengths. Higher moments take more

abstract forms but they do allow experimentally accessible and physically important polymer characteristics like number average molar mass, weight average molar mass and polydispersity index to be calculated. The number average molar mass ($\overline{M}_n$) is defined as follows:

$$\overline{M}_n = \sum x_i \cdot M_i = \frac{\sum n_i \cdot M_i}{\sum n_i} = \frac{FW_{mon} \cdot \mu_1^X}{\mu_0^X} \qquad \textbf{(7-2)}$$

where x_i is the mole fraction of molecules having degree of polymerization i. The equation can also be expressed in numbers of molecules n_i or alternatively in concentrations. The molar mass of a polymer chain can be replaced by the degree of polymerization (i) times the molar mass of the monomer which allows the number average molar mass to be expressed by the ratio of the first over the zeroth moment of the polymer chain distribution times the mass of a single monomer unit (FW_{mon}).

In an analogue derivation it can be shown that the weight average molar mass ($\overline{M}_w$) is equal to the ratio of the second moment over the first moment of the distribution, again multiplied by the mass of the repeat unit:

$$\overline{M}_w = \sum w_i \cdot M_i = \frac{\sum n_i \cdot M_i^2}{\sum n_i \cdot M_i} = \frac{FW_{mon} \cdot \mu_2^X}{\mu_1^X} \qquad \textbf{(7-3)}$$

The polydispersity index can then be calculated from the ratio of $\overline{M}_w$ over $\overline{M}_n$.

More complex molar mass averages as $\overline{M}_z$ and $\overline{M}_{z+1}$ are derived from the higher moments of a distribution in a similar way. For each of the moments of a distribution, a differential equation is required. The approach taken in this thesis is restricted to the first three moments. This results in a model with approximately fifteen differential equations which can readily be solved by ordinary desktop workstations. The derivation of the differential equations is however slightly more complicated then for the previous modelling approaches where, albeit the large number of differential equations, their structure was very straightforward.

The models result in a set of differential equations which is solved numerically using MATLAB, a widely used environment for scientfic computing.[2] MATLAB contains several different solvers for ordinary differential equations. For all models

derived in this appendix, `ode15s` was used, which is a quasi-constant step size integrator. It implements numerical differentiation formulas (NDFs) which can be considered an improvement over the more commonly used backward differentiation formulas (BDFs, also known as Gear's method) in terms of stability, speed and efficiency.[3] The transparent implementation adapts the stepsize of the integration (through time) and the order of the fit to remain within the error margins given by the user. The differential equations can either be hard-coded or constructed programatically.

A.2. Models

A.2.1. Model withouth Chain Lengths

Construction

A simple model that does not consider any chainlengths is easily derived from the reaction schemes in section 2.3 (Schemes 2.11 and 2.12) which shows how species are generated and how they are destroyed or transformed.

$$\frac{dI}{dt} = -k_d \cdot I \tag{7-4}$$

$$\frac{dM}{dt} = -k_p \cdot M \cdot P - k_i \cdot M \cdot R \tag{7-5}$$

$$\frac{dR}{dt} = 2 f k_d \cdot I - k_i \cdot R \cdot M + k_{frag,R} \cdot (PSR + RSR) - k_{add,R} \cdot (SR + SP) - k_t \cdot R \cdot (R + P + RSR + PSR + PSP) \tag{7-6}$$

$$\frac{dP}{dt} = k_i \cdot R \cdot M - k_t \cdot P \cdot (R + P + RSR + PSR + PSP) - k_{add,P} \cdot P \cdot (SR + SP) + k_{frag,P} \cdot P \cdot (PSP + PSR) \tag{7-7}$$

$$\frac{dSR}{dt} = k_{frag,P} \cdot PSR + k_{frag,R} \cdot RSR - k_{add,R} \cdot R \cdot SR - k_{add,P} \cdot P \cdot SR \tag{7-8}$$

$$\frac{dSP}{dt} = k_{frag,P} \cdot PSP + k_{frag,R} \cdot PSR - k_{add,R} \cdot R \cdot SP - k_{add,P} \cdot P \cdot SP \tag{7-9}$$

$$\frac{dD}{dt} = k_t \cdot R \cdot (R + P + PSP + PSR + RSR) + k_t \cdot P \cdot (P + RSR + PSR + PSP) \quad \textbf{(7-10)}$$

$$\frac{dRSR}{dt} = k_{add,R} \cdot R \cdot SR - k_{frag,R} \cdot RSR - k_t \cdot RSR \cdot (R + P) \quad \textbf{(7-11)}$$

$$\frac{dPSR}{dt} = k_{add,R} \cdot R \cdot SP + k_{add,P} \cdot P \cdot SR - k_{frag,R} \cdot PSR - k_{frag,P} \cdot PSR - k_t \cdot PSR \cdot (R + P) \quad \textbf{(7-12)}$$

$$\frac{dPSP}{dt} = k_{add,P} \cdot P \cdot SP - k_{frag,P} \cdot PSP - k_t \cdot PSP \cdot (R + P) \quad \textbf{(7-13)}$$

The model as presented here utilizes a single termination rate constant, but the actual computer files allow one to distinguish intermediate radical termination from the other termination events.

Implementation

Equations 7-4 to 7-13 can be rewritten in a form that is desired by the MATLAB solver. It can integrate ordinary differential equations if they are offered in the following form:

$$y' = F(t, y) \quad \textbf{(7-14)}$$

in which t is a scalar independent variable, in this case time; y is a vector of dependent variables; y' is a function of t and y returning a column vector the same length as y. In this case y could be the vector [I, M, P, T, S, D] and y' the vector containing the elements on the right hand side of the differential equations 7-4 to 7-13. Besides these two vectors a third one is required which indicates the starting conditions $y_0 = [I_0, M_0, 0, T_0, 0, 0]$ and last, the options for the integrator need to be set. These typically dictate the time interval for integration and the absolute and relative error margins. Furthermore, optionally user defined conditions may be constructed (so-called *events*) that prematurely stop the integration. In all the models in this chapter, events were created that stopped integration when either monomer or initiator had reached conversions higher than 99.999% and when the concentration of any species would drop below zero. Further integration beyond this point would

not result in additional meaningful results but stretched the required integration time considerably. Shown below are the contents of the two basic `.m` files required to run this simulation, stripped of all unnessecary functionality.

the file `startit.m`:

```
clear all;

name='noraftterm';
kd   = 1.35e-4;
ki   = 7e2;
kp   = 6.6e2;
kPaddSR  = 7e6;        %P adds
kPaddSP  = 7e6;
kRaddSP  = 7e6;        %R adds
kRaddSR  = 7e6;
kbetaPSP = 1.2e5;        %P fragments
kmaddPSR = 1.2e5;
kbetaRSR = 1.2e5;        %R fragments
kbetaPSR = 1.2e5;
ktbasis  = 2*pi*0.25*7e-9*6.02e23;
ktbI     = 1.5*pi*0.25*7e-9*6.02e23; %ktbasis; %set zero to eliminate intermediate
termination

kmatrix =[kd ki kp kPaddSR kbetaPSR kmaddPSR kRaddSR kbetaRSR kPaddSP kbetaPSP
kRaddSP ktbasis ktbI];

maxci = 0.9999;
maxcm = 0.9999;

mx = [maxci maxcm];

mmo=31;
msol=58;
mass=[mmo msol];

I   = 4.4e-3;  %initiator
M   = 3;       %monomer
R   = 0;       %ini- or raft-derived radicals
SR  = 0;       %raft
P   = 0;       %propagating radicals
SP  = 0;       %dormant species
D   = 0;       %dead chains
RSR = 0;       %intermediate
RSP = 0;       %intermediate
PSP = 0;       %intermediate
%-----------------------------------------------------------------------------------
y0=[I M R SR P SP D RSR RSP PSP];
tmax=[0 3.5e5];
options = odeset('AbsTol',1e-12,'RelTol',3e-
13,'BDF','off','Stats','on','Events','on');
%-----------------------------------------------------------------------------------
tic;
[t,x]=ode15s('simpleraft',tmax,y0,options,kmatrix,y0,mass,mx);
toc;
%-----------------------------------------------------------------------------------
fpm = fopen('overview.txt','a');
temp=[name '\r\n'];
fprintf(fpm,temp);
temp=['ini \t' num2str(I,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['mono \t' num2str(M,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['raft \t' num2str(SR,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kd \t' num2str(kd,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['ki \t' num2str(ki,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kp \t' num2str(kp,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kRaddSR \t' num2str(kRaddSR,'%.3g') '\r\n'];
fprintf(fpm,temp);
```

```
temp=['kRaddSP \t' num2str(kRaddSP,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kPaddSR \t' num2str(kPaddSR,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kPaddSP \t' num2str(kPaddSP,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kbetaRSR \t' num2str(kbetaRSR,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kbetaPSP \t' num2str(kbetaPSP,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kbetaPSR \t' num2str(kbetaPSR,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp=['kmaddPSR \t' num2str(kmaddPSR,'%.3g') '\r\n'];
fprintf(fpm,temp);
temp='\r\n\r\n';
fprintf(fpm,temp);
fclose(fpm);
%-------------------------------------------------------------------------------------
nm= [name '.dat']
fm  = fopen(nm,'w');
fprintf(fm,'t mc I M R SR P SP D RSR RSP PSP\n');
for i=1:max(size(t))
   mc=((M-x(i,2))/M)*100;
   fprintf(fm,'%.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e
%.4e\n',t(i),mc,x(i,1),x(i,2),x(i,3),x(i,4),x(i,5),x(i,6),x(i,7),x(i,8),x(i,9),x(i,10
));
end % for i
fclose(fm);
%-------------------------------------------------------------------------------------
clear all;
```

and the file `simpleraft.m`:

```
function varargout = simpleraft(t,y,flag,k,sv,m,mx)

switch flag
 case ''                       % Return dy/dt = f(t,y).
   varargout{1} = f(t,y,k,sv,m,mx);
 case 'events'                   % Return [value,isterminal,direction]
   [varargout{1:3}] = events(t,y,k,sv,m,mx);
 otherwise
   error(['Unknown flag ''' flag '''.']);
 end
% ------------------------------------------------------------
% 1  I       kd
% 2  M       ki
% 3  R       kp
% 4  SR      kPaddSR
% 5  P       kbetaPSR
% 6  SP      kmaddPSR
% 7  D       kRaddSR
% 8  RSR     kbetaRSR
% 9  PSR     kPaddSP
% 10 PSP     kbetaPSP
% 11         kRaddSP
% 12         ktbasis used for ordinary termination
% 13         ktbI    used for intermediate termination
% ------------------------------------------------------------
function dydt = f(t,y,k,sv,m,mx)

convM =(sv(2)-y(2))/sv(2);  % bereken conversie M
nc = (sv(4)-y(4))+2*(sv(1)-y(1));

if convM<1e-8
   convM=1e-8;
end

if nc<1e-8
   nc=1e-8;
end

if sv(4)>0
   length=round(convM*sv(2)/nc);   % radicaallengte=dormantlengte
else
   length=round((y(2)*k(3))/((2*(k(1)*y(1)*6e8)^0.5)+1e-3*y(2)));  %
radicaallengte=kinetische lengte
end
```

```
if length<3
   length=3;
end

wp=m(1)*convM/(m(1)+m(2)); % bereken wp

Dmon    = 9e-8;
Dshort  = Dmon;
Dlong   = Dmon/(length^min(2, 0.66+2*wp));
Ddouble = Dmon/((2*length)^min(2, 0.66+2*wp));

ktSS  = k(12)*(Dshort+Dshort);
ktISS  = k(13)*(Dshort+Dshort);
% ktLL  = k(12)*(Dlong+Dlong);
ktLS  = k(12)*(Dlong+Dshort);
ktILS  = k(13)*(Dlong+Dshort);
% ktLLL = k(13)*(Ddouble+Dlong);    %intermediate termination PSP P
ktLLS = k(13)*(Ddouble+Dshort);   %intermediate termination PSP R

eff=0.7;
Ithermal=4e-9*y(2)^3;
IT=Ithermal;

 dydt = zeros(10,1);
 dydt(1)=-k(1)*y(1);
 dydt(2)=-k(3)*y(2)*y(5)-k(2)*y(3)*y(2);
 dydt(3)=IT+k(1)*eff*2*y(1)+k(5)*y(9)+k(8)*y(8)-k(7)*y(3)*y(4)-k(11)*y(3)*y(6)-
ktLS*y(3)*y(5)-ktLLS*y(3)*y(10)-k(2)*y(3)*y(2)-ktSS*y(3)*y(3)-ktILS*y(3)*y(9)-
ktISS*y(3)*y(8);
 dydt(4)=k(6)*y(9)+k(8)*y(8)-k(4)*y(5)*y(4)-k(7)*y(3)*y(4);
 dydt(5)=k(2)*y(3)*y(2)+k(6)*y(9)+k(10)*y(10)-k(4)*y(4)*y(5)-k(9)*y(5)*y(6)-
ktLS*y(5)*y(5)-ktLLS*y(5)*y(10)-ktLS*y(3)*y(5)-ktLLS*y(5)*y(9)-ktILS*y(5)*y(8);
 dydt(6)=k(5)*y(9)+k(10)*y(10)-k(9)*y(5)*y(6)-k(11)*y(3)*y(6);

dydt(7)=ktLS*y(5)*y(5)+ktLS*y(5)*y(3)+ktLLS*y(5)*y(10)+ktLLS*y(3)*y(10)+ktISS*y(3)*y(
8)+ktILS*y(5)*y(8)+ktILS*y(3)*y(9);
 dydt(8)=k(7)*y(3)*y(4)-k(8)*y(8)-ktISS*y(3)*y(8)-ktILS*y(5)*y(8);
 dydt(9)=k(4)*y(4)*y(5)+k(11)*y(3)*y(6)-k(5)*y(9)-k(6)*y(9)-ktILS*y(3)*y(9)-
ktLLS*y(5)*y(9);
 dydt(10)=k(9)*y(5)*y(6)-k(10)*y(10)-ktLLS*y(5)*y(10)-ktLLS*y(3)*y(10);
 %-------------------------------------------------------------------------------
  function [value,isterminal,direction] = events(t,y,k,sv,m,mx)

 % sv(1)= concentration I at t=0,  sv(2)= concentration M at t=0
 % y(1) = concentration I at t=t,  y(2) = concentration M at t=t
 % mx(1)= maximum conversion of I, mx(2)= maximum conversion of M
 % abort integration when one of both components reaches max. conversion

 value = zeros(1,2);             % max conversion=zero crossing
 % value(1:2) = [((sv(1)-y(1))/sv(1))-mx(1), ((sv(2)-y(2))/sv(2))-mx(2)];
 value(1:2) = [1, ((sv(2)-y(2))/sv(2))-mx(2)];

 isterminal = zeros(1,2);
 isterminal(1:2) = [1,1];

 direction = zeros(1,2);         % direction unimportant
% ------------------------------------------------------------
```

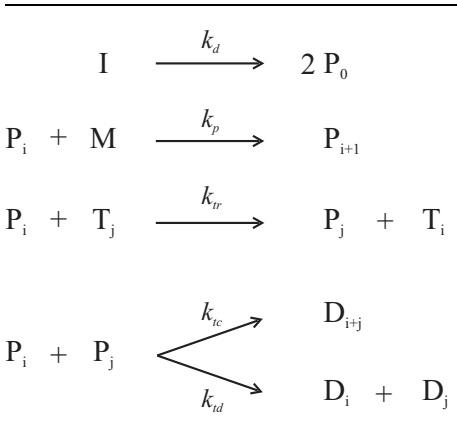
when the `startit` command is given to MATLAB, the code in the first file is executed. The first section allows the user to set different rate constants and concentrations. The second section prepares the integration by constructing a vector of starting conditions and setting the options. The third section executes the integration using the `ode15s` solver and calling `simpleraft.m` for a description of the differential equations. When the integration is finished, the vector `t` contains the time points of the integration and the corresponding rows in matrix `x` contain the concentrations for each of the six different species. The fourth section creates a basic

output file containing all the data in ASCII format. Fully functional `.m` files can be obtained from the author upon request. They may also be downloaded from the author's website (currently www.xs4all.nl/~engel13).

A.2.2. Exact Model

Construction

The second approach mentioned in the introduction uses a differential equation for each individual chainlength for all species and several others for monomer, initiator, etc. The following reaction scheme allows the required set of differential equations to be derived.



species	
initiator	I
monomer	M
radical	P
dormant chain	T
dead material	D

rate constants	
dissociation	k_d
propagation	k_p
combination	k_{tc}
disproportionation	k_{td}
transfer	k_{tr}
initiator effcicency	f

These then are as follows:

$$\frac{dI}{dt} = -k_d \cdot I \qquad \textbf{(7-15)}$$

$$\frac{dM}{dt} = -k_p \cdot M \cdot P \qquad \textbf{(7-16)}$$

$$\frac{dP_0}{dt} = 2 f k_d \cdot I - k_p \cdot P_0 \cdot M - k_{tr} \cdot P_0 \cdot \sum_{j=0}^{n} T_j + k_{tr} \cdot T_0 \cdot \sum_{j=0}^{n} P_j - k_{tc} \cdot P_0 \cdot \sum_{j=0}^{n} P_j - k_{td} \cdot P_0 \cdot \sum_{j=0}^{n} P_j \qquad \textbf{(7-17)}$$

$$\frac{dP_i}{dt} = k_p \cdot P_{i-1} \cdot M - k_p \cdot P_i \cdot M - k_{tr} \cdot P_i \cdot \sum_{j=0}^{n} T_j + k_{tr} \cdot T_i \cdot \sum_{j=0}^{n} P_j - k_{tc} \cdot P_i \cdot \sum_{j=0}^{n} P_j - k_{td} \cdot P_i \cdot \sum_{j=0}^{n} P_j \quad \textbf{(7-18)}$$

$$\frac{dT_i}{dt} = k_{tr} \cdot P_i \cdot \sum_{j=0}^{n} T_j - k_{tr} \cdot T_i \cdot \sum_{j=0}^{n} P_j \quad \textbf{(7-19)}$$

$$\frac{dD_i}{dt} = k_{tc} \cdot \sum_{j=0}^{i} P_j \cdot P_{i-j} + k_{td} \cdot P_i \cdot \sum_{j=0}^{n} P_j \quad \textbf{(7-20)}$$

the subscript *i* in the disributed species denotes the number of monomer units. P_0 is therefore not a polymer radical, but a chemically different species derived from the initiator or transfer agent. T_0 – a dormant species without monomer units – is the initial transfer agent. When n different chainlengths are considered, the total number of differential equations equals 4n+1. Although the model is in principle ideal for most of the living polymerizations in this thesis, it does not allow for comparison of the results with those of (simulated) polymerizations applying less reactive transfer agents.

Besides, the transfer reaction cannot be unraveled further by the use of the full addition–fragmentation equilibrium. The intermediate species has two chains attached to the dithio moiety and the length of both needs to be remembered when it is formed which would result in $\frac{1}{2}n^2$ extra differential equations. For the example given in the introduction this would result in an increase from 450 differential equations to 11,700!

Implementation

Luckily, not all differential equations need to be hardcoded. For each of the species, the differential equations for the various chain lengths are very similar and they can be constructed programatically using a loop to iterate through all chain lengths. Again two files are made:

the file `run.m`

```
clear all;       % clear all variables in the Matlab environment

n  = 80;         % number of identifiable species
nr = (4*n)+4;    % number of differential equations
```

```
ktr = 6.5e3;     % transfer rate constant
kp  = 6.5e2;     % propagation rate constant
kd  = 7.4e-5;    % dissociation rate constant
f   = 0.6;       % initiator efficiency

T = 5.9e-2;      % initial transfer agent concentration
I = 2e-2;        % initial initiator concentration
M = 3;           % initial monomer concentration

maxci=0.999999;  % stop integration at this conversion for initiator I
maxcm=0.999;     % stop integration at this conversion for monomer M
maxpc=0.01;      % max fraction of raft chains as undistinguishable species

tmax = 1e8;      % alternate time of integration
%---------section 2-------------------------------------------------------------
% gather the required parameters

kvalue=[kd ktr kp f];
conc=  [I M T];
maxc=  [maxci maxcm maxpc];
y0=zeros(1,nr);
y0(1:3)=conc;
%---------section 3-------------------------------------------------------------
% construct a matrix e containing the chain length dependend
% termination rate coefficients

TERM=zeros((n+1),(n+1));
for i=1:(n+1)
  for j=1:(n+1)
    if (i<=85)
      D1=3.1e-5/((i+1)^0.5);
    else
      D1=3.1e-5*(85^0.1)/((i+1)^0.6);
    end    % if i
    if (j<=85)
      D2=3.1e-5/((j+1)^0.5);
    else
      D2=3.1e-5*(85^0.1)/((j+1)^0.6);
    end    % if j
    TERM(i,j)=5.58e13*(D1+D2);
  end %for j
end %for i
%-----section 4--calculation--------------------------------------------------
[t,x]=ode15s('raft',[],y0,[],kvalue,conc,maxc,n,TERM);
%-----create output-----------------------------------------------------------
%-----section 5----open files-------------------------------------------------
fm = fopen('main.dat','w');
ft = fopen('raft.dat','w');
fp = fopen('rad.dat','w');
%-----section 6----create headers---------------------------------------------
fst='time mc';
fsp='time mc';

for j = 0:n
   fst=[fst ' t' num2str(j)];
   fsp=[fsp ' p' num2str(j)];
end

fst=[fst ' \n'];
fsp=[fsp ' \n'];

fprintf(fm,'time mc mono ini dead mnR mwR pdR mnD mwD pdD\n');
fprintf(ft,fst);
fprintf(fp,fsp);
%------section 7-----MW averages--raft-------------------------------------------

r=zeros(max(size(t)),6);
for i=1:max(size(t))
   for j=0:n
      r(i,1)=r(i,1)+x(i,3+j);                                    % SUM N
      r(i,2)=r(i,2)+(x(i,3+j)*(250+(j*104.15)));                 % SUM (N*M)
      r(i,3)=r(i,3)+(x(i,3+j)*(250+(j*104.15))*(250+(j*104.15))); % SUM (N*M^2)
   end
   r(i,4)=r(i,2)/r(i,1);   % number average molar mass   Mn= SUM(N*M)/SUM(N)
   r(i,5)=r(i,3)/r(i,2);   % weight average molar mass   Mw= SUM(N*M^2)/SUM(N*M)
   r(i,6)=r(i,5)/r(i,4);   % polydispersity index        PD= Mw/Mn
end
%------section 8----MW averages--dead--------------------------------------------
```

```
d=zeros(max(size(t)),6);
for i=1:max(size(t))
   for j=0:((2*n)-1)
      d(i,1)=d(i,1)+x(i,3+j);                                                    % SUM N
      d(i,2)=d(i,2)+(x(i,5+(2*n)+j)*(130+(j*104.15)));                           % SUM (N*M)
      d(i,3)=d(i,3)+(x(i,5+(2*n)+j)*(130+(j*104.15))*(130+(j*104.15)));  % SUM (N*M^2)
   end
   d(i,4)=d(i,2)/d(i,1);   % number average molar mass   Mn= SUM(N*M)/SUM(N)
   d(i,5)=d(i,3)/d(i,2);   % weight average molar mass   Mw= SUM(N*M^2)/SUM(N*M)
   d(i,6)=d(i,5)/d(i,4);   % polydispersity index        PD= Mw/Mn
end
%------section 9----output----main--raft--rad---------------------------------------

for i=1:max(size(t)),
   mc = ((M - x(i,2))/M)*100;
   fprintf(fm,'%.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e %.4e
%.4e\n',t(i),mc,x(i,2),x(i,1),x(i,(2*n+5)),r(i,4),r(i,5),r(i,6),d(i,4),d(i,5),d(i,6))
;
   fprintf(ft,'%.4e %.4e',t(i),mc);
   fprintf(fp,'%.4e %.4e',t(i),mc);

   for j = 0:(n-1)
      fprintf(ft,' %.3e',x(i,3+j));   % loop raft species
      fprintf(fp,' %.3e',x(i,4+n+j)); % loop radical species
   end
   fprintf(ft,' %.3e \n',x(i,3+n));   % add final species of each series and add
   fprintf(fp,' %.3e \n',x(i,4+n+n)); % an end-of-line character
end
%----section 10-- close files-------------------------------------------------------
fclose(fm);
fclose(ft);
fclose(fp);
%------section 11-----the-dead-files-----------------------------------------------
aantal=(2*n)-1;           % aantal dode species
spf=200;                  % aantal species per file
bestand=fix(aantal/spf);  % aantal files (max 200 species per file) minus 1

for i = 0:bestand         % loop door veschillende bestanden (waarde nul is 1
bestand)
  naam=['dead' num2str(i) '.dat'];  %maak filenaam aan
  fd = fopen(naam,'w');             %open file

  if (i<bestand)
     sif=spf;             % sif is aantal bestanden in deze file alleen in de laatste
  else                    % file is het kleiner dan spf
     sif=aantal-(bestand*spf);
  end  %if

  fsd='time mc';          % header aanmaken
  for j = 0:(sif-1)
     fsd=[fsd ' d' num2str(j+(i*spf))];
  end      % for j
  fsd=[fsd ' \n'];
  fprintf(fd,fsd);

  for k=1:max(size(t)),          % tijden doorlopen
    mc = ((M - x(k,2))/M)*100;
    fprintf(fd,'%.4e %.4e',t(k),mc); % conversie & tijd printen

    for j = 0:(sif-2)            % species doorlopen op 1 na
      temp= x(k,5+(2*n)+j+(i*spf));
      if (temp<1e-120)           % prevent ultra-small numbers (unreadable by Origin)
           temp=0;
      end                        % if
      fprintf(fd,' %.3e',temp);  % loop dead species
    end                          %for j

    j=sif-1;                     % laatste species
    temp = x(k,5+(2*n)+j+(i*spf));
    if (temp<1e-120)
        temp=0;
    end                          % if
    fprintf(fd,' %.3e \n',temp);
  end                            % for k
  fclose(fd);
end                              %for i
%----section 12---MWDs--------------------------------------------------------------
```

```
conv=[1 2 5 10 15 20 25 30 40 50 60 70 80 90 95 96 97 98 99];
nummer=1;
wr=zeros(2,n);           % gewicht raft (1 kolom absoluut/2 geschaald)
wd=zeros(2,(2*n-1));     % gewicht dood (1 kolom absoluut/2 geschaald)

for k=1:max(size(t)),    % tijden doorlopen
   mc = ((M - x(k,2))/M)*100;

   if (nummer>max(size(conv))),break,end     % alle output files klaar

   if (mc>conv(nummer))
     nummer=nummer + 1;

     fn=['r' num2str(conv(nummer-1)) '.dat'];
     fm = fopen(fn,'w');

     for i = 0:(n-1)
       wr(1,(i+1))=(250+(i*104));             % mw as
       wr(2,(i+1))=x(k,3+i)*(250+(i*104));    % gewichts distributie
       fprintf(fm,'%.3e %.3e \n', wr(1,(i+1)), wr(2,(i+1)));
     end % for i

     fclose(fm);
     fn=['d' num2str(conv(nummer-1)) '.dat'];
     fm = fopen(fn,'w');

     for i = 0:(2*n-1)
       wd(1,(i+1))=(130+(i*104));                  % mw as
       wd(2,(i+1))=x(k,5+(2*n)+i)*(130+(i*104));   % gewichts distributie
       fprintf(fm,'%.3e %.3e \n', wd(1,(i+1)), wd(2,(i+1)));
     end % for i
       fclose(fm);
     end % if
end % for k
%-------------------------------------------------------------------------------
clear all;
```

and the file `raft.m`

```
function varargout = raft(t,y,flag,a,b,c,d,e)

 switch flag
 case ''                        % Return dy/dt = f(t,y)
   varargout{1} = f(t,y,a,b,c,d,e);
 case 'init'                    % Return default [tspan,y0,options]
   [varargout{1:3}] = init(a,b,c,d,e);
 case 'events'                  % Return [value,isterminal,direction]
   [varargout{1:3}] = events(t,y,a,b,c,d,e);
 otherwise
   error(['Unknown flag ''' flag '''.']);
 end

% ---------------+------------+----------+----------------------
% conc. in time  |k-values |conc. t=0 | max conversions
% ---------------+------------+----------+----------------------
%I   y(1)        kd   a(1)    I0  b(1)   max.conv. I     c(1)
%M   y(2)        ktr  a(2)    M0  b(2)   max.conv. M     c(2)
%T0  y(3)        kp   a(3)    T0  b(3)   max. long raft  c(3)
%Tn  y(3+d)      f    a(4)
%P0  y(4+d)
%Pn  y(4+2d)
%D0  y(5+2d)
%D2n y(4+4d)
% ---------------------------------------------------------------
%number of species d
%kt matrix         e
% ---------------------------------------------------------------
 function dydt = f(t,y,a,b,c,d,e)

 no = (4*d)+4;               % number of differential equations
 dydt=zeros(no,1);           % define output column vector
 rad=sum(y((4+d):(4+2*d)))   % total radical concentration
 raft=sum(y(3:(3+d)));       % total raft concentration
%---------------------------------------------------------------
% calculate the average radical chain length

 av=0;
```

```
 for i = 0:d             % for all chain lengths
    av=av+(y(4+d+i)*i);  % summarize concentration * length
 end

 if (rad<=0)             % avoid error right at the
    av=0;                % start of integration
 else
    av=fix(av/rad)       %average radical chainlength
 end
%-------------------------------------------------------------
% construct differential equations

 dydt(1) = -a(1)*y(1);     % initiator decay
 dydt(2) = -a(3)*y(2)*rad; % monomer consumption

 for i = 0:d               % dormant species & radicals
   dydt(3+i)= a(2)*y(4+d+i)*raft-a(2)*y(3+i)*rad;
   dydt(4+d+i)=-a(3)*y(4+d+i)*y(2)-a(2)*y(4+d+i)*raft+a(2)*y(3+i)*rad
               -e((av+1),(i+1))*y(4+d+i)*rad;
 end

 for i = 1:(d)             % radicals
   dydt(4+d+i)=dydt(4+d+i)+(a(3)*y(2)*y(3+d+i));
 end
                           % initiator contribution to P0
 dydt(4+d)=dydt(4+d)+(2*a(1)*a(4)*y(1));

                           % cancel propagation for Pn
 dydt(4+2*d)=dydt(4+2*d)+(a(3)*y(4+2*d)*y(2));

                           % dead species-------------------------------------
 for i=0:(d-1)             % chain length dead material
   for j=0:i               % loop different combinations to form dead species
      r1=i-j;              % length radical 1
      r2=j;                % length radical 2
      dydt(5+(2*d)+i)=dydt(5+(2*d)+i)+e((r1+1),(r2+1))*y(4+d+r1)*y(4+d+r2);
   end
 end

  for i=d:(2*(d-1))        % chain length dead material
   for j=fix(i/2):-1:0     % loop different combinations to form dead species
      r1=i-j;              % length radical 1
      r2=j;                % length radical 2
      if (r1>(d-1))|(r2>(d-1)),break,end  % non existing radical length
      dydt(5+(2*d)+i)=dydt(5+(2*d)+i)+e((r1+1),(r2+1))*y(4+d+r1)*y(4+d+r2);
   end
 end

 dydt(4+4*d)=e(d,d)*y(4+2*d)*y(4+2*d);
%-------------------------------------------------------------------------------

 function [tspan,y0,options] = init(a,b,c,d,e)

 tspan = [0 1e8];    % default timespan
 nr = (4*d)+5;       % number of differential equations/compounds
 y0 = zeros(1,nr);   % starting concentrations
 y0(1:3) = b(1:3);
 options =odeset('AbsTol',1e-7,'RelTol',1e-7,'BDF','off','Stats','on','Events','on');
                     % error tolerances
%-------------------------------------------------------------------------------
 function [value,isterminal,direction] = events(t,y,a,b,c,d,e)

 % b(1)= concentration I at t=0,  b(2)= concentration M at t=0
 % y(1)= concentration I at t=t,  y(2)= concentration M at t=t
 % c(1)= maximum conversion of I, c(2)= maximum conversion of M
 % abort integration when one of both components reaches max. conversion

 % second criterium: integrating noise, [radicals]<0

 % third criterium: heap of non-distinguisable species > 1%

 value = zeros(1,d+4);             % max conversion=zero crossing
 value(1:2) = [((b(1)-y(1))/b(1))-c(1), ((b(2)-y(2))/b(2))-c(2)];

 for i =0:d
    value(i+3) = y(4+d(1)+i);      % [radical]<0
 end
```

```
 value(1,d+4)= (y(d+3)/b(3))-c(3); % third criterium

 isterminal = zeros(1,d+4);
 isterminal(1:2) = [1,1];
 isterminal(d+4) =[1];             % all are terminal events
 direction = zeros(1,d+4);         % direction unimportant
%-------------------------------------------------------------------------------
```

The files are for a large part self explanetory. Comments can be found inline with the code. The same general structure is applied as in the previous model. `run.m` collects and prepares the input parameters, calls the `ode15s` solver which uses the differential equations in `raft.m`, and produces several output files from the raw integration results. `raft.dat`, `rad.dat` and `deadX.dat` (`X` being an integer) contain the concentrations of each and every species in time. `Main.dat` contains the molar mass averages, polydispersities and conversion as a function of time. Section 12 generates a number of molar mass distributions at the conversions specified in the `conv` vector. For every point both an `rX.dat` and `dX.dat` (`X` being the conversion) file are created containing the molar mass distribution of the dormant chains and of the dead material respectively.

The `raft.m` file illustrates the use of events. As only a limited number of chain lengths is considered the model will need to check during the integration whether or not this number still suffices. If any material grows to larger chain lengths, the model needs to terminate. This can be achived by removal of the positive contribution of propagation from the largest radical species (in its differential equation). This prevents polymer 'growing out of the model'. The largest radical species P_n not only represents polymer radicals with length n, but cumilates all longer chains as well. In every iteration the model checks the concentration of P_n and as soon as it amounts to more then 1% of the total radical concentration, the integration is halted. Events are constructed in such a way that they represent a certain zero-crossing. The actual percentage of P_n is substracted from 1 so that the event-value evaluates to zero and is recognised by MATLAB. Reaching the maximum conversion for either initiator or monomer and negative radical concentrations also trigger events that halt the integration because they indicate that the polymerization is essentially finished and that further integration will yield meaningless results.

A.2.3. The Method of Moments

The third model discussed in the introduction aims to keep track of the molar mass averages of a distribution rather than the full distribution itself which, as shown in the previous section, is not possible for a lot of systems. The computa-

tional power released by this simplification can be employed to tackle more demanding polymerizations. The derivation of the differential equations from the reaction scheme is in this case slightly more complicated. The same reaction scheme is used as for the exact model to arrive at the following differential equations for the individual species:

$$\frac{dI}{dt} = -k_d \cdot I \qquad \textbf{(7-21)}$$

$$\frac{dM}{dt} = -k_p \cdot M \cdot P \qquad \textbf{(7-22)}$$

$$\begin{aligned}\frac{dP_0}{dt} = {} & 2fk_d \cdot I - k_p \cdot P_0 \cdot M - k_{tr} \cdot P_0 \cdot \sum_{j=0}^{\infty} T_j + k_{tr} \cdot T_0 \cdot \sum_{j=0}^{\infty} P_j \\ & - k_{tc} \cdot P_0 \cdot \sum_{j=0}^{\infty} P_j - k_{td} \cdot P_0 \cdot \sum_{j=0}^{\infty} P_j\end{aligned} \qquad \textbf{(7-23)}$$

$$\begin{aligned}\frac{dP_i}{dt} = {} & k_p \cdot M \cdot (P_{i-1} - P_i) - k_{tr} \cdot P_i \cdot \sum_{j=0}^{\infty} T_j + k_{tr} \cdot T_i \cdot \sum_{j=0}^{\infty} P_j \\ & - k_{tc} \cdot P_i \cdot \sum_{j=0}^{\infty} P_j - k_{td} \cdot P_i \cdot \sum_{j=0}^{\infty} P_j\end{aligned} \qquad \textbf{(7-24)}$$

$$\frac{dT_0}{dt} = k_{tr} \cdot P_0 \cdot \sum_{j=1}^{\infty} T_j - k_{tr} \cdot T_0 \cdot \sum_{j=1}^{\infty} P_j \qquad \textbf{(7-25)}$$

$$\frac{dT_i}{dt} = k_{tr} \cdot P_i \cdot \sum_{j=0}^{\infty} T_j - k_{tr} \cdot T_i \cdot \sum_{j=0}^{\infty} P_j \qquad \textbf{(7-26)}$$

$$\frac{dD_i}{dt} = k_{tc} \cdot \sum_{j=0}^{i} P_j \cdot P_{i-j} + k_{td} \cdot P_i \cdot \sum_{j=0}^{\infty} P_j \qquad \textbf{(7-27)}$$

The differential equations for the initiator and the monomer can be used directly in the model. Note that P_0 and T_0 are treated seperately and have been taken out of their distribution. This affects only the zeroth moment of the distribution as for the higher moments the contribution of the individual species is multi-

plied with its index (see formula 7-1) which cancels out the zero-length species. Before these equations can be put in the model they need to be rewritten so that they are expressed in moments:

$$\frac{dP_0}{dt} = 2fk_d \cdot I - k_p \cdot P_0 \cdot M - k_{tr} \cdot P_0 \cdot \mu_0^T + k_{tr} \cdot T_0 \cdot \mu_0^P - (k_{tc} + k_{td}) \cdot P_0 \cdot (P_0 + \mu_0^P) \quad \textbf{(7-28)}$$

$$\frac{dT_0}{dt} = k_{tr} \cdot P_0 \cdot \mu_0^T - k_{tr} \cdot T_0 \cdot \mu_0^P \quad \textbf{(7-29)}$$

The equations for P_i, T_i and D_i are used to derive the respective differential equations for the moments of these distributions. The differential equation for the zeroth moment of the radical distribution follows from summation over all i from 1 to infinity:

$$\frac{d\mu_0^P}{dt} = \frac{d\sum_{i=1}^{\infty} P_i}{dt} = k_p \cdot M \cdot \sum_{i=1}^{\infty} (P_{i-1} - P_i) - k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} P_i + k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} T_i - (k_{tc} + k_{td}) \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} P_i \quad \textbf{(7-30)}$$

$$\frac{d\mu_0^P}{dt} = k_p \cdot M \cdot P_0 - k_{tr} \cdot \mu_0^P \cdot (T_0 + \mu_0^T) + k_{tr} \cdot \mu_0^T \cdot (P_0 + \mu_0^P) - (k_{tc} + k_{td}) \cdot (P_0 + \mu_0^P) \cdot \mu_0^P \quad \textbf{(7-31)}$$

$$\frac{d\mu_0^P}{dt} = k_p \cdot M \cdot P_0 + k_{tr} \cdot P_0 \cdot \mu_0^T - k_{tr} \cdot T_0 \cdot \mu_0^P - (k_{tc} + k_{td}) \cdot (P_0 + \mu_0^P) \cdot \mu_0^P \quad \textbf{(7-32)}$$

Keeping in mind that the zeroth moment is in fact the total concentration of polymeric radicals, the correctness of the obtained differential equation can easily be rationalized. There are positive contributions from initiator derived radicals that 'propagate into the distribution' as well as from dormant polymer chains that become activated by such a radical. Negative contributions result from both termination and the reaction between transfer agent and a propagating species which generates a P_0 instead of a P_i radical.

The same technique will be used to arrive at the differential equations for the first and the second moment although for these cases the result takes a more abstract form:

$$\frac{d\mu_1^P}{dt} = \frac{d\sum_{i=1}^{\infty} i \cdot P_i}{dt} = k_p \cdot M \cdot \sum_{i=1}^{\infty} i \cdot (P_{i-1} - P_i) - k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} i \cdot P_i$$
$$+ k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i \cdot T_i - (k_{td} + k_{tc}) \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i \cdot P_i \qquad \textbf{(7-33)}$$

$$\frac{d\mu_1^P}{dt} = k_p \cdot M \cdot (P_0 + \mu_0^P) - k_{tr} \cdot (T_0 + \mu_0^T) \cdot \mu_1^P + k_{tr} \cdot (P_0 + \mu_0^P) \cdot \mu_1^T$$
$$- (k_{td} + k_{tc}) \cdot (P_0 + \mu_0^P) \cdot \mu_1^P \qquad \textbf{(7-34)}$$

and the second moment

$$\frac{d\mu_2^P}{dt} = \frac{d\sum_{i=1}^{\infty} i^2 \cdot P_i}{dt} = k_p \cdot M \cdot \sum_{i=1}^{\infty} i^2 \cdot (P_{i-1} - P_i) - k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} i^2 \cdot P_i$$
$$+ k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i^2 \cdot T_i - (k_{td} + k_{tc}) \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i^2 \cdot P_i \qquad \textbf{(7-35)}$$

$$\frac{d\mu_2^P}{dt} = k_p \cdot M \cdot (P_0 + \mu_0^P + 2\mu_1^P) - k_{tr} \cdot (T_0 + \mu_0^T) \cdot \mu_2^P + k_{tr} \cdot (P_0 + \mu_0^P) \cdot \mu_2^T$$
$$- (k_{td} + k_{tc}) \cdot (P_0 + \mu_0^P) \cdot \mu_2^P \qquad \textbf{(7-36)}$$

The moments for the dormant species are derived as follows:

zeroth moment:

$$\frac{d\mu_0^T}{dt} = \frac{d\sum_{i=1}^{\infty} T_i}{dt} = k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} P_i - k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} T_i \quad \textbf{(7-37)}$$

$$\frac{d\mu_0^T}{dt} = k_{tr} \cdot (T_0 + \mu_0^T) \cdot \mu_0^P - k_{tr} \cdot (P_0 + \mu_0^P) \cdot \mu_0^T \quad \textbf{(7-38)}$$

first moment:

$$\frac{d\mu_1^T}{dt} = \frac{d\sum_{i=1}^{\infty} i \cdot T_i}{dt} = k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} i \cdot P_i - k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i \cdot T_i \quad \textbf{(7-39)}$$

$$\frac{d\mu_1^T}{dt} = k_{tr} \cdot (T_0 + \mu_0^T) \cdot \mu_1^P - k_{tr} \cdot (P_0 + \mu_0^P) \cdot \mu_1^T \quad \textbf{(7-40)}$$

second moment:

$$\frac{d\mu_2^T}{dt} = \frac{d\sum_{i=1}^{\infty} i^2 \cdot T_i}{dt} = k_{tr} \cdot \left(T_0 + \sum_{j=1}^{\infty} T_j\right) \cdot \sum_{i=1}^{\infty} i^2 \cdot P_i - k_{tr} \cdot \left(P_0 + \sum_{j=1}^{\infty} P_j\right) \cdot \sum_{i=1}^{\infty} i^2 \cdot T_i \quad \textbf{(7-41)}$$

$$\frac{d\mu_2^T}{dt} = k_{tr} \cdot (T_0 + \mu_0^T) \cdot \mu_2^P - k_{tr} \cdot (P_0 + \mu_0^P) \cdot \mu_2^T \quad \textbf{(7-42)}$$

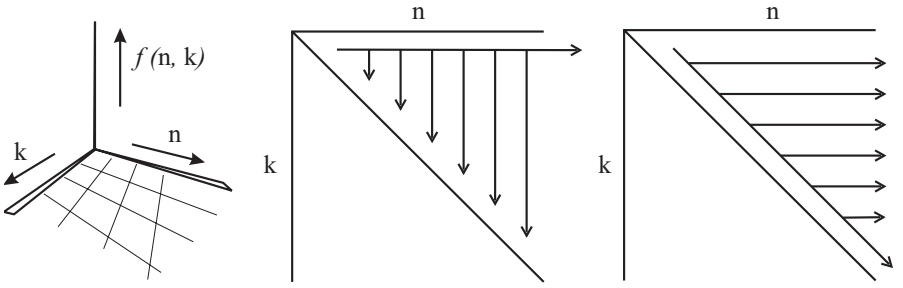
for the dead chains:

$$\frac{d\mu_0^D}{dt} = \frac{d\sum_{i=0}^{\infty} D_i}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{i} P_j \cdot P_{i-j} + k_{td} \cdot \sum_{i=0}^{\infty} P_i \cdot \sum_{j=0}^{\infty} P_j \quad \textbf{(7-43)}$$

in which the contribution from disproportionation can readily be expressed in moments. Rewriting the contribution from combination requires the use of the following identity:

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n} f(k,n) = \sum_{k=0}^{\infty}\sum_{n=k}^{\infty} f(k,n) = \sum_{n=0}^{\infty}\sum_{k=0}^{\infty} f(k,n+k) \qquad \textbf{(7-44)}$$

the first step can be visualized if we consider a three dimensional space:



the middle illustration shows which area on the base, the *nk*-plane, is covered by the first summation. The same area is covered by the second summation. Here *k* runs from zero to infinity while *n* runs from the current *k* value to infinity. When in this summation, *n* is replaced by the new running variable *n+k*, the summation takes its final form. If this identity is applied to equation 7-43 , it can be expressed in the moments of the radical distribution.

$$\frac{d\mu_0^D}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} P_j \cdot P_i + k_{td} \cdot \sum_{i=0}^{\infty} P_i \cdot \sum_{j=0}^{\infty} P_j \qquad \textbf{(7-45)}$$

$$\frac{d\mu_0^D}{dt} = (k_{tc} + k_{td}) \cdot (P_0 + \mu_0^P) \qquad \textbf{(7-46)}$$

the same identity is used in the derivation of the first and the second moment:

$$\frac{d\mu_1^D}{dt} = \frac{d\sum_{i=0}^{\infty} i \cdot D_i}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{i} i \cdot P_j \cdot P_{i-j} + k_{td} \cdot \sum_{i=0}^{\infty} i \cdot P_i \cdot \sum_{j=0}^{\infty} P_j \qquad \textbf{(7-47)}$$

$$\frac{d\mu_1^D}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (i+j) \cdot P_j \cdot P_i + k_{td} \cdot \mu_1^P \cdot (P_0 + \mu_0^P) \qquad \textbf{(7-48)}$$

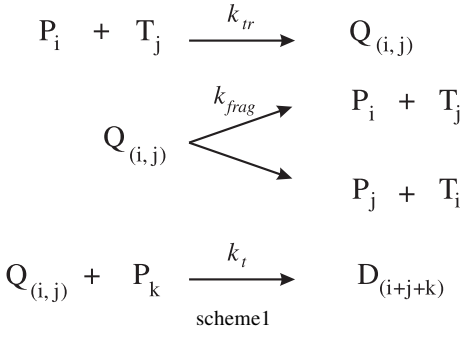
$$\frac{d\mu_1^D}{dt} = (k_{td} + 2 \cdot k_{tc}) \cdot \mu_1^P \cdot (P_0 + \mu_0^P) \qquad \textbf{(7-49)}$$

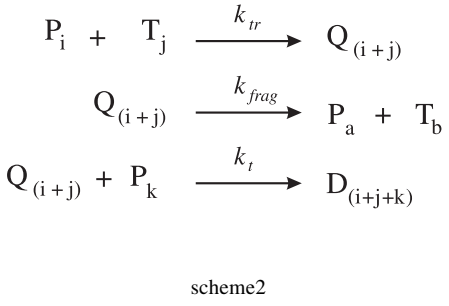
$$\frac{d\mu_2^D}{dt} = \frac{d\sum_{i=0}^{\infty} i^2 \cdot D_i}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{i} i^2 \cdot P_j \cdot P_{i-j} + k_{td} \cdot \sum_{i=0}^{\infty} i^2 \cdot P_i \cdot \sum_{j=0}^{\infty} P_j \qquad \textbf{(7-50)}$$

$$\frac{d\mu_2^D}{dt} = k_{tc} \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (i+j)^2 \cdot P_j \cdot P_i + k_{td} \cdot \mu_2^P \cdot (P_0 + \mu_0^P) \qquad \textbf{(7-51)}$$

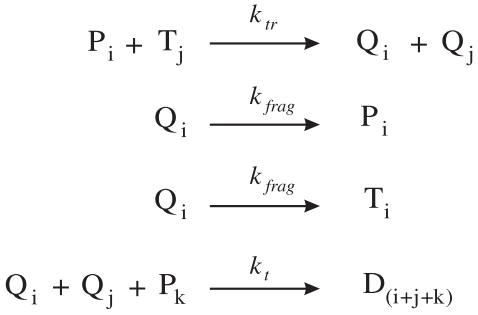
$$\frac{d\mu_2^D}{dt} = (k_{td} + 2 \cdot k_{td}) \cdot \mu_2^P \cdot (P_0 + \mu_0^P) + 2 \cdot k_{tc} \cdot (\mu_1^P)^2 \qquad \textbf{(7-52)}$$

now that the appropiate differential equations have been derived, they can be put in a MATLAB .m file similar to the other models. The output matrix contains the evolution of the moments in time from which the molar mass averages and the polydispersity can be calculated according to equations 7-2 and 7-3.

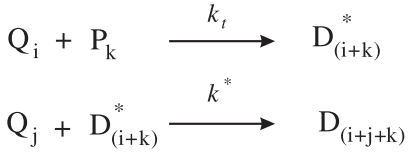




To develop the model further it would be desirable to use the full addition–fragmentation equilibrium and to investigate the intermediate radical as well. As mentioned in the previous model, the intermediate radical has a double distribution and the length of both chains attached to the dithiocarbonate moitey needs to be known (Scheme 1). Treatment of such a 'two dimensional' compound is impossible with the method of moments. When the intermediate is simplified to a one dimensional species, the length of the individual chains that are regenerated upon



scheme 3



scheme 4

fragmentation is not known (Scheme 2). This approach is therefore rejected. A good alternative is the physically unrealisic scheme where the transfer reaction generates two polymer chains (Scheme 3). The reaction is mathematically identical to termination by disproportionation and can easily be described using the moments method. Fragmentation then can be described as a unimolecular transition to either a dormant or a radical species. The concentration dependencies of the rate (fragmentation being first order in the concentration of the intermediate) are maintained in the original state. The consequences of this unrealistic simulation are a faulty concentration of the intermediate species. This can be corrected as the difference is a factor of two. Calculating the average molar mass will also yield erroneous results, but these data are not of particular interest anyway. More important is the molar mass of the termination product formed in the reaction between the intermediate species and a propagating radical. To arrive at the correct molar mass, this will need to be a trimolecular reaction using two intermediate species and a radical.

When the model is transformed into differential equations, the correct rate structure will be lost. The rate of the last termination reaction, that of the intermediate radical will increase by a factor of four when the concentration of the intermediate radical doubles. The correct rate structure can be restored (first order in both intermediate and radical concentration) when substituting the last reaction in Scheme 3 with the two reactions in Scheme 4 allows the complete scheme to be expressed in differential equations, maintaining correct reaction rates and allowing the predication of all molar mass averages. However the prerequisite that the second reaction of Scheme 4 be much faster than the first (in other words, the first reaction is the rate determining step) makes the entire matrix of differential equations close to singular and leads to an extremely stiff system which cannot be solved anymore. The integrator will be required to make extremely small steps on the entire timescale solely because of this reaction. Up to now, no method was found to circumvent this problem.

A.3. Monte Carlo Simulations

Monte Carlo simulations are based on the element of chance. They do not require the chemical reactions to be transformed into a mathematical model of differential equations, but instead make use of probability functions and random numbers. The model used to investigate the effect of the transfer coefficient and the targeted degree of polymerization on the polydispersity of the final product requires four input parameters, in its current implementation: amount of monomer, amount of RAFT agent, the transfer rate coefficient and the propagation rate coefficient. Of the latter two, only their ratio is of importance, reducing the number of actual parameters to three. The simulation neglects termination events and physicalle corresponds to a polymerization with one single radical. One of the RAFT agents is randomly chosen to be the active species. Monomer species are added one by one to the active species. Before each addition occurs, the chance of transfer is calculated using Eq. 7-53:

$$P(transfer) = \frac{k_{tr} \cdot RAFT}{k_{tr} \cdot RAFT + k_p \cdot M} \qquad \textbf{(7-53)}$$

$P(transfer)$ is compared with a random value between 0 and 1 and if transfer occurs, a new, randomly chosen, species from the population is set to be active. The ratio of monomer to raft agent determines the target degree of polymerization, one of the experimental parameters, while the magnitude of the individual species determines the statistical variation. Larger populations of monomer and RAFT, but in the same ratio, will produce more consistent results.

A.4. References

1. From Neuromancer, William Gibson, 1984
2. www.mathworks.com
3. Shampine, L. F.; Reichelt, M. W. *SIAM J. Sci. Comput.* **1997**, *18*, 1.