ELSEVIER

J. Non-Newtonian Fluid Mech. 108 (2002) 227–255

Journal of Non-Newtonian Fluid Mechanics

www.elsevier.com/locate/jnnfm

Brownian dynamics simulations of bead-rod and bead-spring chains: numerical algorithms and coarse-graining issues

Madan Somasi [a], Bamin Khomami [a,*], Nathanael J. Woo [b], Joe S. Hur [c], Eric S.G. Shaqfeh [c]

[a] *Department of Chemical Engineering and Materials Research Laboratory, Washington University, Campus Box 1198, St. Louis, MO 63130, USA*

[b] *Scientific Computing/Computational Mathematics Program, Stanford University, Stanford, CA 9305, USA*

[c] *Department of Chemical and Mechanical Engineering, Stanford University, Stanford, CA 9305, USA*

Received 6 August 2001; received in revised form 10 October 2001

Abstract

The efficiency and robustness of various numerical schemes have been evaluated by performing Brownian dynamics simulations of bead-rod and three popular nonlinear bead-spring chain models in uniaxial extension and simple shear flow. The bead-spring models include finitely extensible nonlinear elastic (FENE) springs, worm-like chain (WLC) springs, and Pade approximation to the inverse Langevin function (ILC) springs. For the bead-spring chains two new predictor–corrector algorithms are proposed, which are much superior to commonly used explicit and other fully implicit schemes. In the case of bead-rod chain models, the mid-point algorithm of Liu [J. Chem. Phys. 90 (1989) 5826] is found to be computationally more efficient than a fully implicit Newton's method. Furthermore, the accuracy and computational efficiency of two different stress expressions for the bead-rod chains, namely the Kramers–Kirkwood and the modified Giesekus have been evaluated under both transient and steady conditions. It is demonstrated that the Kramers–Kirkwood with stochastic filtering is the preferred choice for transient flow while the Giesekus expression is better suited for steady state calculations. The issue of coarse graining from a bead-rod chain to a bead-spring chain has also been investigated. Though bead-spring chains are shown to capture only semi-quantitatively the response of the bead-rod chains in transient flows, a systematic coarse-graining procedure that provides the best description of bead-rod chains via bead-spring chains is presented.

Keywords: Brownian dynamics; Dilute polymer solution; Bead-spring model; Bead-rod model; Shear flow; Extensional flow; Algorithm; Coarse graining

* Corresponding author. Tel.: +1-314-935-6065; fax: +1-314-935-7211.
E-mail address: bam@poly1.che.wustl.edu (B. Khomami).

0377-0257/02/$ – see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0377-0257(02)00132-5

1. Introduction

Quantitative predictions of the velocity and stress distributions in complex flows of polymeric solutions is a challenging goal, but offers the promise of more rational design of polymer processing operations. The conventional approach to modeling the flow of dilute polymeric solutions is based on solving the equations of conservation of mass, momentum and energy in conjunction with a closed-form constitutive equation for the polymeric stress (e.g. [1–3]). To date, the most commonly used closed-form constitutive equations for dilute polymeric solutions have been obtained by approximating models based on kinetic theory. However, such approximations commonly referred to as 'closure approximations' often distort the predictions of the original kinetic theory based model [4].

In recent years, continuum level discretization techniques have been successfully merged with Brownian dynamics (BD) techniques to allow direct use of micro-structural models for polymer dynamics in flow simulations, thus by-passing the need for closed-form constitutive equations [5–8]. Algorithms for efficient continuum/mesoscopic simulation have advanced at a rapid rate since their first introduction in the early 90s. However, even with the most efficient multi-scale algorithms, complex flow simulations have been limited to include only very simple micro-structural models for the polymeric molecule, such as the elastic dumbbell model. This is due to the fact that this class of simulations is very CPU intensive primarily due to the large ensemble required to obtain accurate polymeric stresses from Brownian dynamic simulations. Hence, use of more complex micro-structural models such as bead-rod chains in complex flow simulations is beyond the reach of the current computing power.

In order to improve the efficiency and predictive capability of combined mesoscopic/continuum simulation techniques, two issues need to be addressed. First, more CPU efficient BD algorithms need to be developed. Secondly, accurate coarse-graining procedures from a molecular level (i.e. atomistic) to mesoscopic level (e.g. bead-rod and bead-spring chains) need to be developed. In what follows, we will briefly describe the basic approach in coarse graining from an atomistic to a mesoscopic level of description.

The coarse graining from an atomistic level to the bead-rod description is based on sound statistical mechanic principles. Specifically, atomic vibrations are neglected leading to 'freely rotating bonds'. Moreover, the distance between the adjacent beads, i.e. the 'Kuhn step' can be determined based on the fact that pair-wise inter-atomic potential interactions can be neglected beyond a certain distance. Further coarse graining from a bead-rod to a bead-spring chain relies on the assumption of local equilibrated motion of many Kuhn steps. Hence, a number of Kuhn steps are replaced by a 'phantom entropic spring'. It should be noted that depending on the flow strength, the number of Kuhn steps that can be equivalently replaced by an entropic spring is different; hence systematic coarse-graining procedures from the bead-rod to the bead-spring chains are rarely available. Finally, in the most coarse-grained approximation, one can neglect the internal structure of the molecule and represent the whole molecule as a single dumbbell with an entropic spring. However, an alternative procedure has been suggested by Ghosh et al. [9] who have recently proposed a coarse-graining procedure based on an adaptive length scale principle, where the properties of the entropic spring are adjusted based on the flow strength.

As mentioned earlier, flow simulations of dilute polymeric solutions with micromechanical models containing sufficient molecular level information to describe the nonlinear rheology of polymeric solutions, require fast BD algorithms as well as the need for accurate coarse-graining procedures. Hence, we have focused our attention on the development of CPU efficient BD simulation algorithms for bead-rod and bead-spring chains as well as the development of a rational coarse-graining procedure to represent a bead-rod chain by a smaller bead-spring chain. Specifically, we have compared the performance

of a fully implicit Newton's method against the commonly used two-step mid-point technique for BD simulation of bead-rod chains. In addition, the accuracy and CPU efficiency of the *stochastically filtered* Kramers–Kirkwood and the modified Giesekus expressions for evaluating the polymeric stress in bead-rod simulations have been compared.

Similarly, for the bead-spring models the performance of conventional explicit techniques with a fully implicit technique based on Newton's method have been examined. In addition, two new predictor–corrector based techniques are introduced. Continued iteration of these methods results in fully self-consistent methods and their performance relative to the other two techniques is discussed. Finally, coarse graining from a bead-rod to a bead-spring chain is examined. The aim is to find the number of Kuhn steps that can be replaced by an equivalent nonlinear entropic spring such that the polymeric stress computed from both micro-structural models are equivalent over a wide range of flow types and Weissenberg numbers.

The manuscript is organized as follows. The governing equations and the expressions for the stress tensor for various nonlinear bead-spring chains and the Kramers chains are introduced in Section 2. Following this, the various numerical methods used to solve the governing equations for both bead-rod and bead-spring models are discussed. The two new algorithms for flow simulation of the three different bead-spring chains are introduced in this section. The relative performance of the various numerical techniques discussed in Section 2 is also examined in Section 3. Specifically, the accuracy of the solution and the computational efficiencies of the various techniques are examined. This is followed by the discussion of the scaling relationships and coarse-graining procedures in Section 4. We conclude by highlighting the best simulation algorithms.

2. Governing equations

In the following, a brief description of the governing equations for the two micro-structural models, namely the bead-rod chain and the bead-spring chain necessary to evaluate the chain dynamics under flow will be discussed.

The bead-rod chain consists of N_k beads connected by $N_k - 1$ rigid rods of length a (Fig. 1). The beads serve as interaction points with the solvent and the massless rods act as rigid constraints in the chain that keep every bead at a constant distance a away from its neighboring beads.

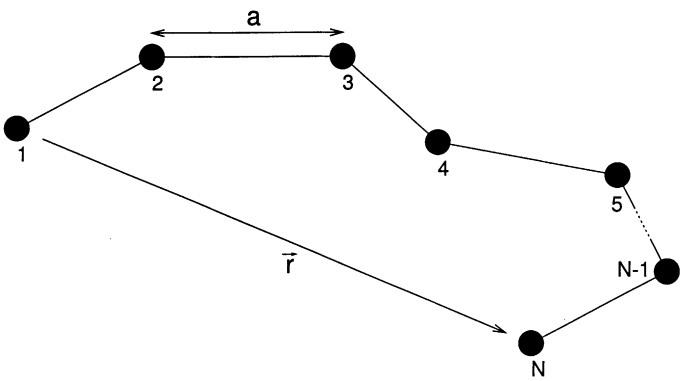

Fig. 1. The freely jointed bead-rod chain (Kramers chain) model for a polymer chain.

The governing equations for the dynamics of the chain are obtained by summing the external forces acting on the beads,

$$\boldsymbol{F}_i^{\mathrm{H}} + \boldsymbol{F}_i^{\mathrm{C}} + \boldsymbol{F}_i^{\mathrm{B}} = 0, \quad i = 1, 2, \ldots, N_k \tag{1}$$

where the subscript i refers to the bead number and $\boldsymbol{F}^{\mathrm{H}}$, $\boldsymbol{F}^{\mathrm{C}}$, $\boldsymbol{F}^{\mathrm{B}}$ are the hydrodynamic drag force, the constraint force and the Brownian force, respectively. The hydrodynamic force is the Stokes drag acting on the beads in the absence of hydrodynamic interaction and fluid inertia and is simply given by,

$$\boldsymbol{F}_i^{\mathrm{H}} = -\zeta(\dot{\boldsymbol{r}}_i - \boldsymbol{u}_i^{\infty}) \tag{2}$$

where ζ is the drag coefficient (i.e. the drag tensor has been assumed to be isotropic), $\dot{\boldsymbol{r}}_i$ the velocity of the bead and $\boldsymbol{u}_i^{\infty}$ is the solvent velocity at bead i. The constraint force is the force that keeps two neighboring beads at a constant distance a away from each other, and can be written in terms of the tensions T_i in the rods as follows,

$$\boldsymbol{F}_i^{\mathrm{C}} = T_i\boldsymbol{u}_i - T_{i-1}\boldsymbol{u}_{i-1} \tag{3}$$

where $\boldsymbol{u}_i = (\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)/a$ is the orientation of rod i connecting beads i and $i+1$ with position vectors $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i+1}$, respectively. The Brownian force is used to represent the frequent collisions between the bead and the solvent molecules and is mathematically represented as a quantity with zero mean (because of the isotropic nature of the collisions) and a second moment that balances the dissipative forces (also known as the fluctuation–dissipation theorem of the second kind [11]). Therefore, $\boldsymbol{F}^{\mathrm{B}}$ is chosen such that

$$\langle \boldsymbol{F}_i^{\mathrm{B}}(t) \rangle = 0 \tag{4}$$

$$\langle \boldsymbol{F}_i^{\mathrm{B}}(t)\boldsymbol{F}_j^{\mathrm{B}}(t+\Delta t) \rangle = 2kT\zeta\delta_{ij}\delta(\Delta t) \approx \frac{2kT\zeta\delta_{ij}}{\Delta t} \tag{5}$$

where k is the Boltzmann's constant, T the absolute temperature, δ_{ij} the Dirac delta and δ is the unit tensor. In the above equations, $\langle \cdots \rangle$ represents an ensemble average.

After substitution of different forces acting on the beads into Eq. (1) and rearranging, one can obtain the following set of equations governing the evolution of the position vectors of the beads:

$$\mathrm{d}\boldsymbol{r}_i = \left[\boldsymbol{u}_i^{\infty} + \frac{\boldsymbol{F}_i^{\mathrm{C}}}{\zeta}\right]\mathrm{d}t + \sqrt{\frac{2kT}{\zeta}}\,\mathrm{d}\boldsymbol{W}_i, \quad i = 1, 2, \ldots, N_k \tag{6}$$

where $\mathrm{d}\boldsymbol{W}_i$ is a Wiener process mathematically represented by a Gaussian random number with a mean of zero and variance $\mathrm{d}t$. The above set of equations have to be solved subject to the constraints that the distance of separation between any two adjacent beads at any instant of time is fixed at a, or

$$(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)\cdot(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i) - a^2 = \phi^2, \quad i = 1, 2, \ldots, N_k - 1 \tag{7}$$

where ϕ^2 is a specified tolerance (typically 10^{-6} to 10^{-8}). Eqs. (6) and (7) result in $2N_k - 1$ equations for N_k position variables $\boldsymbol{r}_i$ and $N_k - 1$ tensions T_i, which can be solved in a number of ways. In the next section, two numerical schemes, namely the Newton's method and the method of Lagrange multipliers will be discussed. To facilitate comparison with the bead-spring results, we have chosen to non-dimensionalize the above equations as follows. The distance between adjacent beads a serves as the length scale of the problem, the bead diffusion time, $\zeta a^2/kT$ serves as the characteristic time scale and the forces are made dimensionless with kT/a.

Once the position vectors of all the beads have been determined, the polymer contribution to the total stress, τ_p can be obtained using the Kramers–Kirkwood expression (neglecting the isotropic contribution) as,

$$\tau_p = \sum_{i=1}^{N_k} \langle \boldsymbol{R}_i \boldsymbol{F}_i^H \rangle \tag{8}$$

where $\boldsymbol{R}_i$ is the position of bead i relative to the center of mass of the chain, i.e.,

$$\boldsymbol{R}_i = \boldsymbol{r}_i - \boldsymbol{r}_c \tag{9}$$

where

$$\boldsymbol{r}_c = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{r}_i . \tag{10}$$

In Eq. (8) the polymer stress τ_p has been non-dimensionalized using $n_p kT$, with n_p being the number density of the polymers in solution. Furthermore, in order to obtain smooth averages, we have followed the noise reduction technique proposed by Doyle et al. [10]. In addition to the Kramers–Kirkwood expression discussed above, the polymer stress can also be obtained using the modified Giesekus expression as given by [11,12]

$$\tau_p = -\frac{1}{2} \left[\sum_{i=1}^{N_k} \langle \boldsymbol{R}_i \boldsymbol{R}_i \rangle \right]^{(1)} \tag{11}$$

where the superscript (1) denotes the upper convected derivative,

$$\boldsymbol{X}^{(1)} = \frac{\partial \boldsymbol{X}}{\partial t} - Pe(\boldsymbol{\kappa}^{\dagger} \cdot \boldsymbol{X} + \boldsymbol{X} \cdot \boldsymbol{\kappa}) \tag{12}$$

where $Pe = \dot{\gamma}(\zeta a^2/kT)$ is the bead Peclet number, which can be interpreted as the ratio of the bead diffusion time to the characteristic flow time.

Although the modified Giesekus expression for the polymer stress is more useful in evaluating the polymeric stress under steady state conditions, we have chosen to evaluate its usefulness in transient flows as well, since it is simpler to evaluate than the Kramers–Kirkwood expression with the noise reduction technique of Doyle et al. [10].

2.1. Simulation procedure

The initial conditions for the positions vectors of each bead in a chain are obtained by simulating a random walk, i.e. each successive $\boldsymbol{r}_{i+1} - \boldsymbol{r}_i$ is obtained by invoking a random unit vector on a surface of a sphere. The first bead is assumed to be at the origin, which thus specifies the remaining position vectors uniquely. However, since it is known that, at equilibrium, a Kramers chain does not exactly follow a random walk structure [12], the correct initial distribution is obtained by allowing all the chains to equilibrate for a significant time period (typically 10^6–10^7 time steps, which corresponds to around 1000 times the bead diffusion time). Once the correct equilibrium distribution has been obtained, the chains are subject to flow at time $t = 0$, after which, Eq. (6) is integrated in time subject to the constraints and any macroscopic

properties such as the stress are obtained via appropriate averages. Typically, 1000–4000 chains are used in every transient simulation, with time steps of 10^{-3} to 10^{-5} depending on the flow strength.

2.2. Bead-spring chains

A bead-spring chain consists of M beads connected by $N_s = M - 1$ entropic springs. While the beads serve as interaction points with the solvent, the springs represent entropic effects due to the internal degrees of freedom which have been lost in the process of coarse graining. The governing equations are obtained as before by summing the external forces acting on the beads

$$\boldsymbol{F}_i^{\mathrm{H}} + \boldsymbol{F}_i^{\mathrm{E}} + \boldsymbol{F}_i^{\mathrm{B}} = 0, \quad i = 1, 2, \ldots, M \tag{13}$$

where the subscript i refers to the bead number and $\boldsymbol{F}^{\mathrm{H}}$, $\boldsymbol{F}^{\mathrm{E}}$, $\boldsymbol{F}^{\mathrm{B}}$ are the hydrodynamic drag force, the effective spring force and the Brownian force, respectively.

The expressions for the hydrodynamic force (i.e. Eq. (2)) and the Brownian force (Eqs. (4) and (5)) are identical to the corresponding bead-rod chain expressions. The effective spring force on bead i is given as

$$\boldsymbol{F}_i^{\mathrm{E}} = \begin{cases} \boldsymbol{F}_1^{\mathrm{s}} & \text{if} \quad i = 1 \\ \boldsymbol{F}_i^{\mathrm{s}} - \boldsymbol{F}_{i-1}^{\mathrm{s}} & \text{if} \quad 1 < i < M \\ \boldsymbol{F}_{M-1}^{\mathrm{s}} & \text{if} \quad i = M \end{cases} \tag{14}$$

where $\boldsymbol{F}_i^{\mathrm{s}}$ is the spring force associated with spring i. Depending on the force law used for the springs, one can arrive at different models. In this work, we shall make use of three widely used nonlinear spring force laws to model the springs, i.e. the finitely extensible nonlinear elastic (FENE) force law, the worm-like chain (WLC) force law [13] and a Pade approximation to the inverse Langevin force law (ILC) [14]. The force laws are summarized below for comparison:

$$\boldsymbol{F}_i^{\mathrm{FENE}} = \frac{H_s \boldsymbol{Q}_i}{1 - Q^2/Q_0^2} \tag{15}$$

$$\boldsymbol{F}_i^{\mathrm{WLC}} = \frac{kT}{b_{\mathrm{K}}} \left[\frac{1}{2} \frac{1}{(1 - (Q/Q_0))^2} - \frac{1}{2} + \frac{2Q}{Q_0} \right] \frac{\boldsymbol{Q}_i}{Q_0} \tag{16}$$

$$\boldsymbol{F}_i^{\mathrm{ILC}} = \frac{kT}{b_{\mathrm{K}}} \frac{\boldsymbol{Q}_i}{Q_0} \left[\frac{3 - (Q/Q_0)^2}{1 - (Q/Q_0)^2} \right] \tag{17}$$

where $\boldsymbol{Q}_i = \boldsymbol{r}_{i+1} - \boldsymbol{r}_i$ is the connector vector of the ith spring, H_s the spring constant, Q the magnitude of $\boldsymbol{Q}_i$ and Q_0 is the maximum extensibility of each spring. In Eq. (16) we have used the fact that the Kuhn length b_{K} is twice the persistence length l^{p}, i.e. $b_{\mathrm{K}} = 2l^{\mathrm{p}}$. Lastly, we note that all spring force laws in Eqs. (15)–(17) follow the Hookean spring force law in the small deformation limit.

Eq. (13) can be rearranged to obtain the stochastic differential equation (SDE) governing the evolution of the position vectors of the beads of the bead-spring chain as

$$\mathrm{d}\boldsymbol{r}_i = \left[u_i^{\infty} + \frac{\boldsymbol{F}_i^{\mathrm{E}}}{\zeta} \right] \mathrm{d}t + \sqrt{\frac{2kT}{\zeta}}\, \mathrm{d}\boldsymbol{W}_i, \quad i = 1, 2, \ldots, M \tag{18}$$

Furthermore, the polymer stress, τ_p, is given by the Kramers expression as follows

$$\tau_p = \sum_{i=1}^{N_s} \langle \boldsymbol{Q}_i \boldsymbol{F}_i^s \rangle \tag{19}$$

To begin the simulations, typically 1000–10,000 chains are chosen which satisfy the equilibrium distribution function of a bead-spring chain comprised of Hookean springs. As in our bead-rod simulations, these chains are allowed to equilibrate for 10^4–10^6 steps such that the correct distribution for a given force law is obtained. The flow field is then introduced and the evolution of the chains is monitored. Furthermore, macroscopic properties such as the viscosity and its first normal stress coefficient are obtained at any time through appropriate averaging. In order to facilitate comparisons with the bead-rod simulations, it is important to introduce the characteristic scales for non-dimensionalization. The average equilibrium length of a Hookean dumbbell, $\sqrt{kT/H_s}$, has been chosen as the length scale in the simulations. Time is made dimensionless by $\zeta/4H_s$ (i.e. the relaxation time of a Hookean dumbbell), and stress is non-dimensionalized with $n_p kT$ as before. We define the Weissenberg number as the product of the longest relaxation time (λ) and the shear rate ($\dot{\gamma}$). For bead-rod chains, $We = \lambda\dot{\gamma} = 0.0142 N_k^2 \dot{\gamma}(\xi_k a^2/kT)$. Similarly for bead-spring chains, $We = \lambda^d\dot{\gamma} = \lambda^d\dot{\gamma}(\xi/4H_s)$. The detail of how the longest relaxation times for both bead-rod and bead-spring chains are calculated from the simulation is outlined in Section 4.3. For both chains, *We* is the product of the dimensionless longest relaxation time and *Pe*. The dimensionless form of the equations and the numerical techniques used to solve them are introduced in the next section.

3. Numerical techniques

To compare the accuracy and CPU efficiency of various techniques, simulations have been performed in two homogeneous flows, namely the start-up of steady shear and uniaxial extensional flow. In the case of steady shear flow, the velocity field is given as

$$v_x = \dot{\gamma} y; \qquad v_y = v_z = 0 \tag{20}$$

and for uniaxial extensional flow, the velocity field takes the form

$$v_x = \dot{\epsilon} x; \qquad v_y = -\tfrac{1}{2}\dot{\epsilon} y; \qquad v_z = -\tfrac{1}{2}\dot{\epsilon} z \tag{21}$$

where $\dot{\gamma}$ and $\dot{\epsilon}$ are the shear rate and elongation rate, respectively.

3.1. Bead-rod simulations

In the following, the integration scheme and subsequently, the solvers used for the bead-rod simulations are described. We have used the iterative technique originally proposed by Liu [15] to solve the SDEs describing the evolution of the bead position vectors $\boldsymbol{r}_i$, which, when written in the dimensionless form are as follows

$$\mathrm{d}\boldsymbol{r}_i = [Pe(\boldsymbol{\kappa}\cdot\boldsymbol{r}_i) + \boldsymbol{F}_i^C]\mathrm{d}t + \sqrt{2}\,\mathrm{d}\boldsymbol{W}_i, \quad i = 1, 2, \ldots, N_k \tag{22}$$

subject to the constraint

$$(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)\cdot(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i) - 1 = \phi^2, \quad i = 1, 2, \ldots, N_k - 1 \tag{23}$$

where $\boldsymbol{\kappa} = \nabla \boldsymbol{v}^{\dagger}$ is the transpose of the velocity gradient tensor. Since the above equations have been rendered dimensionless, $\boldsymbol{\kappa}$ has the form

$$\boldsymbol{\kappa} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \qquad \boldsymbol{\kappa} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix} \tag{24}$$

in simple shear and extensional flow, respectively.

In Liu's [15] approach, given $\boldsymbol{r}_i(t)$, $\boldsymbol{r}_i(t+\delta t)$ is obtained utilizing a predictor–corrector type algorithm. The predictor step consists of an 'unconstrained' move (i.e. $\boldsymbol{F}_i^{\mathrm{C}} = 0$) such that,

$$\boldsymbol{r}_i^{*} = \boldsymbol{r}_i(t) + Pe\boldsymbol{\kappa}(t) \cdot \boldsymbol{r}_i(t)\mathrm{d}t + \sqrt{2}\,\mathrm{d}\boldsymbol{W}_i, \quad i = 1, 2, \ldots, N_k \tag{25}$$

Following this, $\boldsymbol{r}_i(t+\delta t)$ is obtained as

$$\boldsymbol{r}_i(t+\delta t) = \boldsymbol{r}_i^{*} + \boldsymbol{F}_i^{\mathrm{C}}\delta t \tag{26}$$

Combining Eqs. (23) and (26), along with the expression for the constraint force $\boldsymbol{F}_i^{\mathrm{C}}$ from Eq. (3) gives $N_k - 1$ nonlinear equations for the tensions T_i,

$$\begin{aligned}(2\delta t)\boldsymbol{b}_i \cdot (T_{i-1}\boldsymbol{u}_{i-1} - 2T_i\boldsymbol{u}_i + T_{i+1}\boldsymbol{u}_{i+1}) + (T_{i-1}\boldsymbol{u}_{i-1} - 2T_i\boldsymbol{u}_i + T_{i+1}\boldsymbol{u}_{i+1})^2(\delta t)^2 \\ = \phi^2 + 1 - (|\boldsymbol{b}_i|)^2, \quad i = 1, 2, \ldots, N_k\end{aligned} \tag{27}$$

where $\boldsymbol{b}_i = \boldsymbol{r}_{i+1}^{*} - \boldsymbol{r}_1^{*}$. The above set of nonlinear equations can be solved using a number of different approaches, and in this study we shall examine a few methods. The first method is a Picard's method, which is identical to that used by Liu [15]. The nonlinear terms in tensions (second term on the left-hand side) appearing in the above equation are assumed to be small in comparison to the linear terms. Therefore, the set of equations is rewritten in terms of a linear set of equations and solved iteratively by evaluating the nonlinear part from the previous iteration. This results in a tridiagonal system of equations for the $N_k - 1$ tensions. Alternatively, Newton's method can be used to solve the set of discrete equations. Utilizing Newton's method, Eq. (27) can be rewritten as,

$$\begin{aligned}\boldsymbol{F}_i(x) \equiv (2\delta t)\boldsymbol{b}_i \cdot (T_{i-1}\boldsymbol{u}_{i-1} - 2T_i\boldsymbol{u}_i + T_{i+1}\boldsymbol{u}_{i+1}) + (T_{i-1}\boldsymbol{u}_{i-1} - 2T_i\boldsymbol{u}_i + T_{i+1}\boldsymbol{u}_{i+1})^2(\delta t)^2 - \phi^2 - 1 \\ + (|\boldsymbol{b}_i|)^2 = 0, \quad i = 1, 2, \ldots, N_k - 1\end{aligned} \tag{28}$$

where $\boldsymbol{F}_i$ is the ith equation and we define $\boldsymbol{x} = \{T_1, T_2, \ldots, T_{N_k-1}\}^{\dagger}$ as the vector of the tension variables. To implement this technique we note that given the value of $\boldsymbol{x} = \boldsymbol{x}^{\mathrm{old}}$ at any instant, the value of $\boldsymbol{x} = \boldsymbol{x}^{\mathrm{new}}$ at the next iteration can be obtained according to Newton's method as

$$\boldsymbol{x}^{\mathrm{new}} = \boldsymbol{x}^{\mathrm{old}} - J^{-1}(\boldsymbol{x}^{\mathrm{old}}) \cdot \boldsymbol{F}(\boldsymbol{x}^{\mathrm{old}}) \tag{29}$$

where J^{-1} is the inverse of the Jacobian matrix $\boldsymbol{J}$ defined as

$$\boldsymbol{J} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}} \tag{30}$$

where $\boldsymbol{F} = \{F_1, F_2, \ldots, F_{N_k-1}\}^{\dagger}$. The process is repeated until $|\boldsymbol{x}^{\mathrm{new}} - \boldsymbol{x}^{\mathrm{old}}| <$ tolerance, where the tolerance is typically 10^{-6} to 10^{-8}.

Both the methods outlined above need an initial condition for $\boldsymbol{x}$. Usually, the solution at the previous time step is taken to be the starting guess for the iterative process at the present time. It should be noted that, while the convergence rate of the Newton's method is quadratic, as opposed to the linear convergence of the Picard's method, Newton's method involves evaluating the Jacobian at *every* iteration, while the resulting tridiagonal matrix of the linear Picard's method is constant for all iterations at any particular time step. The relative performance of both these methods for different chain lengths and for various flow strengths will be discussed below.

3.2. Bead-spring simulations

In this section we illustrate all algorithms only for the case of FENE chains realizing that extensions to other nonlinear bead-spring cases can be similarly accomplished. Since the spring forces can be written in terms of the inter-bead displacements, it will be useful to write the SDEs describing the bead-spring chain dynamics in terms of the connector vectors $\boldsymbol{Q}_i$ rather than the bead positions $\boldsymbol{r}_i$

$$\mathrm{d}\boldsymbol{Q}_i = [Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}_i) + \tfrac{1}{4}(\boldsymbol{F}^{\mathrm{s}}_{i-1} - 2\boldsymbol{F}^{\mathrm{s}}_i + \boldsymbol{F}^{\mathrm{s}}_{i+1})]\delta t + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}_{i+1} - \mathrm{d}\boldsymbol{W}_i) \tag{31}$$

for $i = 1, 2, \ldots, N_{\mathrm{s}}$. The above equation has been written in the dimensionless form utilizing the scales introduced in Section 2. For example, the dimensionless spring force $\boldsymbol{F}^{\mathrm{s}}_i$ for a FENE dumbbell has the form

$$\boldsymbol{F}^{\mathrm{s}}_i = \frac{\boldsymbol{Q}_i}{1 - \boldsymbol{Q}_i^2/b} \tag{32}$$

where $b = H_{\mathrm{s}}Q_0^2/kT$ is the dimensionless extensibility parameter. The other spring force laws in Eqs. (16) and (17) can be made dimensionless using the same scalings outlined in Section 2.2.

Eq. (31) can again be integrated utilizing various techniques. The simplest procedure involves marching forward in time using an explicit forward Euler integration scheme. The next improvement to this technique is the Picard's method, in which the explicit solution procedure is iterated at every time step until the residual of the solution falls below a pre-specified tolerance. However, when using these explicit techniques, care should be taken that the length of the connector vector $\boldsymbol{Q}$ does not exceed the maximum permissible value $\sqrt{b}$, because this would result in aphysical spring forces. This can be avoided by following the rejection algorithm proposed by Öttinger [11], i.e. a move that results in an aphysical spring force law is rejected. Alternately, one can use Newton's method to solve the set of nonlinear equations iteratively at every time step until convergence is achieved. Furthermore, the resulting set of linear equations at every iteration in the Newton's method within a time step can be either solved using a direct solver, or an iterative solver.

In this work, we shall compare both the explicit methods and the fully implicit Newton's method to two new semi-implicit predictor–corrector methods that we have developed to integrate the above set of equations. These methods, which have been motivated by the two-step semi-implicit method proposed by Öttinger (to integrate a FENE dumbbell) [11] are given below. Note that in each of these methods we can and do continue the predictor–corrector iteration to convergence. Thus these techniques can be considered to be *alternative fully implicit* solution techniques as they produce self-consistent solutions at each time step.

In the first algorithm, given the solution $\boldsymbol{Q}_i^n$ at time $t = n\delta t$, the solution $\boldsymbol{Q}_i^{n+1}$ at the next time step $t = (n+1)\delta t$, is obtained as follows:

Step 1. In the predictor step, $\boldsymbol{Q}_i(t_n)$ is *explicitly* updated to obtain $\boldsymbol{Q}_i^*(t_n)$ as

$$\boldsymbol{Q}_i^* = \boldsymbol{Q}_i^n + [Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}_i^n) + \tfrac{1}{4}(\boldsymbol{F}_{i-1}^{s,n} - 2\boldsymbol{F}_i^{s,n} + \boldsymbol{F}_{i+1}^{s,n})]\delta t + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}_{i+1}^n - \mathrm{d}\boldsymbol{W}_i^n) \tag{33}$$

where $\boldsymbol{F}_i^{s,n}$ is the spring force for the ith segment at time $t = n\delta t$.

Step 2. In the first corrector step, the spring forces for segments i and $i-1$ are treated implicitly when solving for $\bar{\boldsymbol{Q}}_i$ such that

$$\bar{\boldsymbol{Q}}_i + \tfrac{1}{2}(\delta t)\bar{\boldsymbol{F}}_i^s = \boldsymbol{Q}_i^n + [\tfrac{1}{2}Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}_i^n + \boldsymbol{\kappa}\cdot\boldsymbol{Q}_i^*) + \tfrac{1}{4}(\bar{\boldsymbol{F}}_{i-1}^s + \boldsymbol{F}_{i+1}^{s,n})]\delta t + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}_{i+1}^n - \mathrm{d}\boldsymbol{W}_i^n) \tag{34}$$

which, upon rearrangement, results in the following cubic equation for the magnitude of $\bar{\boldsymbol{Q}}_i$ for each ith spring in the chain

$$|\bar{\boldsymbol{Q}}_i|^3 - \boldsymbol{R}|\bar{\boldsymbol{Q}}_i|^2 - b(1 + \tfrac{1}{2}(\delta t))|\bar{\boldsymbol{Q}}_i| + b\times\boldsymbol{R} = 0 \tag{35}$$

where $\boldsymbol{R}$ is the magnitude of the right-hand side vector of Eq. (34) and b is the dimensionless extensibility parameter. This equation has one unique root between 0 and $\sqrt{b}$, and thus by choosing this root, we can ensure that $|\bar{\boldsymbol{Q}}_i|$ is never greater than $\sqrt{b}$.

Step 3. In the final corrector step, once again the spring forces for segments i and $i-1$ are treated implicitly, while the spring force for segment $i+1$ is obtained from Step 2.

$$\begin{aligned}\boldsymbol{Q}_i^{n+1} + \tfrac{1}{2}(\delta t)\boldsymbol{F}_i^{s,n+1} = \boldsymbol{Q}_i^n + [\tfrac{1}{2}Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}_i^n + \boldsymbol{\kappa}\cdot\bar{\boldsymbol{Q}}_i) + \tfrac{1}{4}(\boldsymbol{F}_{i-1}^{s,n+1} + \boldsymbol{F}_{i+1}^s)]\delta t \\ + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}_{i+1}^n - \mathrm{d}\boldsymbol{W}_i^n)\end{aligned} \tag{36}$$

The above equation also results in a cubic equation similar to Eq. (35), which can be solved to obtain a root for $|\boldsymbol{Q}_i|$ that lies between 0 and $\sqrt{b}$. Once every $\boldsymbol{Q}_i^{n+1}$ is known, the residual ϵ is calculated as the difference between the solutions $\bar{\boldsymbol{Q}}_i$ and $\bar{\boldsymbol{Q}}_i^{n+1}$.

$$\epsilon = \sqrt{\sum_{i=1}^{N_s}(\boldsymbol{Q}_i^{n+1} - \bar{\boldsymbol{Q}}_i)^2} \tag{37}$$

If this residual is greater than a specified tolerance (e.g. 10^{-6}), $\boldsymbol{Q}_i^{n+1}$ is copied onto $\bar{\boldsymbol{Q}}_i$ and Step 3 is repeated until convergence. This results in a self-consistent semi-implicit method. Basically, the premise of the new method lies in the fact that, the force law, $\bar{\boldsymbol{F}}_i^s$ for any spring in the chain, is either written explicitly (from the previous time step or from a previous step in the present time step, where the length of the connector is guaranteed to be within the bounds) or solved implicitly through the cubic equation. This ensures that no Brownian step is rejected at any time during the entire simulation.

For the WLC and inverse Langevin chain, we obtain a different cubic equation than Eq. (35) above. As in the FENE case, we are still guaranteed a unique real root between 0 and $\sqrt{b}$.

We have also investigated a second slightly different algorithm which treats the spring force term using a trapezoidal rule instead of the backward Euler scheme as in Eqs. (34) and (36). In this case the overall algorithm becomes second order accurate because the trapezoidal rule is a higher order scheme. However,

in actuality, there is no improvement in accuracy due to the discontinuities in the Brownian forcing term [16]. For this algorithm the corrector steps corresponding to Eqs. (34) and (36) are given by

$$\bar{\boldsymbol{Q}}_i + \tfrac{1}{4}(\delta t)\bar{\boldsymbol{F}}^{\mathrm{s}}_i = \boldsymbol{Q}^n_i + [\tfrac{1}{2}Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}^n_i + \boldsymbol{\kappa}\cdot\boldsymbol{Q}^*_i) + \tfrac{1}{8}(\bar{\boldsymbol{F}}^{\mathrm{s}}_{i-1} + \boldsymbol{F}^{\mathrm{s},n}_{i+1}) + \tfrac{1}{8}(\boldsymbol{F}^{\mathrm{s},n}_{i-1} - 2\boldsymbol{F}^{\mathrm{s},n}_i + \boldsymbol{F}^{\mathrm{s},n}_{i+1})]\delta t + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}^n_{i+1} - \mathrm{d}\boldsymbol{W}^n_i) \tag{38}$$

$$\boldsymbol{Q}^{n+1}_i + \tfrac{1}{4}(\delta t)\boldsymbol{F}^{\mathrm{s},n+1}_i = \boldsymbol{Q}^n_i + [\tfrac{1}{2}Pe(\boldsymbol{\kappa}\cdot\boldsymbol{Q}^n_i + \boldsymbol{\kappa}\cdot\bar{\boldsymbol{Q}}_i) + \tfrac{1}{8}(\boldsymbol{F}^{\mathrm{s},n+1}_{i-1} + \bar{\boldsymbol{F}}^{\mathrm{s}}_{i+1}) + \tfrac{1}{8}(\boldsymbol{F}^{\mathrm{s},n}_{i-1} - 2\boldsymbol{F}^{\mathrm{s},n}_{i-1} + \boldsymbol{F}^{\mathrm{s},n}_{i+1})]\delta t + \sqrt{\tfrac{1}{2}}(\mathrm{d}\boldsymbol{W}^n_{i+1} - \mathrm{d}\boldsymbol{W}^n_i) \tag{39}$$

Similarly we have the following cubic equation for the FENE in the trapezoidal algorithm,

$$|\bar{\boldsymbol{Q}}_i|^3 - \boldsymbol{R}|\bar{\boldsymbol{Q}}_i|^2 - b(1 + \tfrac{1}{4}(\delta t))|\bar{\boldsymbol{Q}}_i| + b\times\boldsymbol{R} = 0 \tag{40}$$

We have found that the performance of these two algorithms is nearly identical. For shear flow, the trapezoidal algorithm performs better than backward Euler scheme at a very large time step (i.e. $\Delta t = 0.1$), while for extensional flow the backward Euler scheme outperforms the trapezoidal scheme. For typical time steps these two algorithms give identical results, thus in what follows we shall only directly compare the backward Euler scheme to the explicit and Newton's implicit schemes.

3.3. Cubic equation iteration schemes

We have also investigated various iteration schemes to solve Eqs. (34), (36), (38) and (39). In the context of a *linear* matrix equation $Ax = b$, the matrix A can be split into three matrices $A = D + L + U$, where D, L and U are the diagonal, lower triangular, and upper triangular matrices, respectively. The simplest iteration scheme known as *Jacobi* method is obtained by retaining the diagonal matrix on the left side of the equation while moving the rest of the matrices to the other side resulting in $Dx = -(L+U)x+b$. This is motivated by the fact that inverting a diagonal matrix is much more CPU efficient than inverting a more general matrix. Therefore the iteration scheme is $x_{n+1} = D^{-1}(-(L+U)x_n + b)$. The convergence of the iteration scheme depends on the eigenvalues of the matrix $-(L+U)$. For Gauss–Seidel (GS), we retain $(D+L)$ on the left side of the equation and the iteration scheme becomes $x_{n+1} - (D+L)^{-1}(Ux_n + b)$. Similarly for successive over-relaxation (SOR), we have $x_{n+1}(-(1/\omega D + L)^{-1}((1-1/\omega)D + U)x_n + b)$. The choice of different methods of splitting the matrix A is motivated by a search for the iteration technique with the highest convergence rate. For linear matrix problems, with a special choice of ω, SOR can reduce the spectral radius of the iteration matrix and a faster convergence is obtained over Jacobi and GS iterations [17].

A similar approach is followed when solving nonlinear problems. The simplest scheme is Jacobi-like iteration, where $\bar{\boldsymbol{F}}^{\mathrm{s}}_{i-1}$ in Eqs. (34) and (38) are evaluated in the previous iteration step as $\bar{\boldsymbol{F}}^{\mathrm{s},n}_{i-1}$, and $\bar{\boldsymbol{F}}^{\mathrm{s},n+1}_{i-1}$ in Eqs. (36) and (39) are evaluated as $\bar{\boldsymbol{F}}^{\mathrm{s}}_{i-1}$. Alternatively, one can divide the equations into two groups based on bead number, odd i's and even i's, and iteratively calculate $|\boldsymbol{Q}|$ in Eqs. (34) and (38) alternating with even and odd i values. Since each spring is connected only to its nearest neighbors, the equations are completely decoupled. This is known as a *black-and-white* or two-color GS iteration [17]. Since this is equivalent to using the most updated values of $|\boldsymbol{Q}|_{\mathrm{s}}$, it is easy to understand why GS converges faster than Jacobi iteration. The iteration scheme we presented in Eqs. (34), (36), (38) and (39) are identical to

the two-color GS method with the bead numbers merely renumbered in a different order. Similarly we can try a variant of the SOR method in which a fraction of the non-coupling force term is moved to the other side in addition to the terms moved by the GS method. In the nonlinear problem, we find SOR is even slower than Jacobi iteration and we find that the GS iteration scheme is about 30–100% faster than Jacobi iteration method. Therefore we have presented results based on the fastest iteration scheme (i.e. the two-color GS as shown in Eqs. (34), (36), (38) and (39)).

4. Results and discussions

4.1. Bead-rod chains

Fig. 2 shows the comparison of the CPU time ratio between the Newton's and the Picard's method for calculation in uniaxial extensional flow at $We = 1$ and 10. We observe that the Picard's method is always at least 10% faster than the Newton's method for any given chain size. We have also found this to be generally true regardless of flow strength and type.

Furthermore, we have tested the relative performance of the two stress expressions, namely the modified Giesekus, i.e. Eq. (11) and the Kramers–Kirkwood given by Eq. (8) plus the noise reduction technique proposed by Doyle et al. [10]. Fig. 3 shows a typical result for start-up of an extensional flow for 1000 chains using both stress expressions. As seen from the figure, both the expressions for stress give almost identical results both in the transient and the steady state regions. Moreover, the CPU time associated with evaluation of the stress using both approaches are very similar. However, if one just observes the steady part of the curve (Fig. 3(b)), it is obvious that the Giesekus expression outperforms the Kramers–Kirkwood expression, especially in regards to the statistical noise associated with the stochastic simulations. In addition, the Giesekus expression is much simpler to evaluate than the Kramers–Kirkwood

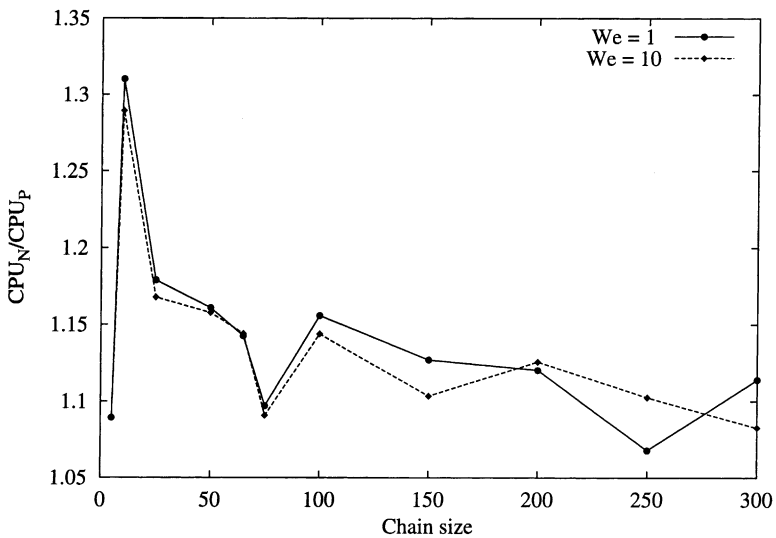

Fig. 2. The ratio of the CPU time for the Newton's method to the Picard's iterative method for a 50 bead Kramers chain for $We = 1$ and 10.

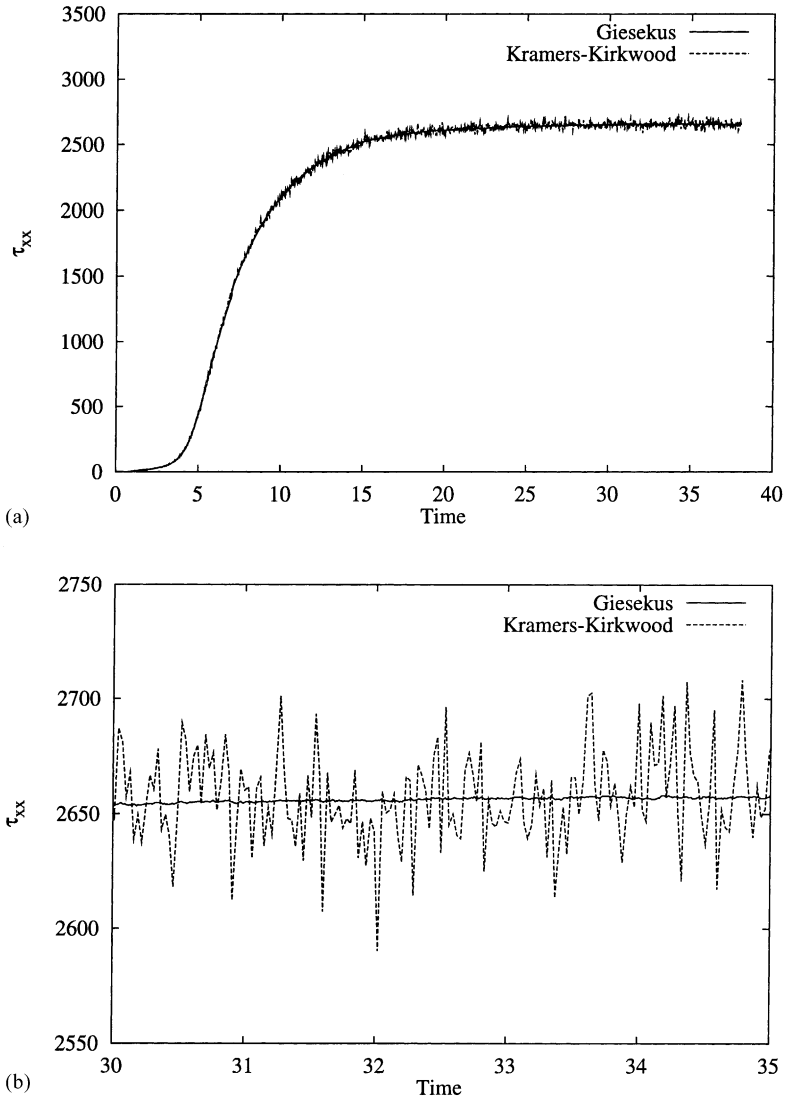

Fig. 3. Comparison between the Giesekus and Kramers–Kirkwood stress expressions for Kramers chains ($N_k = 50$, $We = 11.4$): evolution of τ_{xx} with time: (a) transient region and (b) steady state region.

expression (with noise reduction technique of Doyle et al. [10]. However, it should be noted that in transient simulations, the Giesekus stress expression involves evaluating time derivatives, which is not a trivial exercise because of the rapidly fluctuating results. Hence, the derivatives need to be *smoothed*. To obtain the results in Fig. 3, the smoothing of the derivatives has been performed based on using a number of previous variable values instead of employing just the immediate previous value. The effect of increasing the number of previous variable values in calculating the time derivative on the overall quality of the solution is shown in Fig. 4, where the evolution of the stress τ_{xx} has been plotted as a function of time.

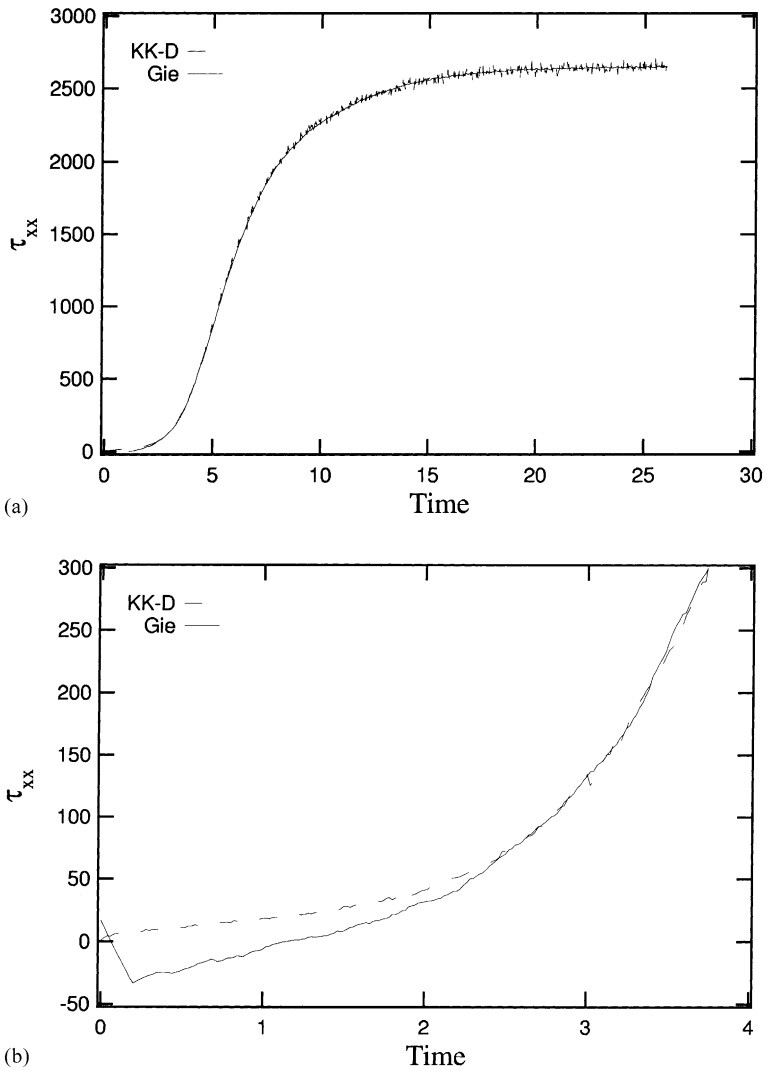

Fig. 4. Comparison between the Giesekus and Kramers–Kirkwood stress expressions for Kramers chains with $N_p = 200$ ($N_k = 50$, $We = 11.4$): (a) transient and (b) blow-up of initial transient.

Specifically, the time derivative of any quantity X is evaluated as follows

$$\frac{dX(t)}{dt} = \frac{X(t) - X(t - N_p\,\Delta t)}{N_p\,\Delta t} \tag{41}$$

We have used different values of N_p, the number of past values, ranging from 2 to 200 in our derivative calculations. The results shown in Fig. 3 correspond to a value of 200. Clearly, by choosing an appropriate size of past values (in this particular case, around 200 past values), we can see that the solution during the transient region of the simulation also improves considerably. Furthermore, we have also observed that as far as the statistical error is concerned, the Giesekus expression with a smoothed derivative

over 20 past values is very similar to that calculated from the Kramers–Kirkwood expression. Values below 10 result in very large statistical noise when evaluating the stress utilizing the modified Giesekus expression. Although increase in N_p does smooth out the stochastic noise in the derivative calculations, it also introduces a numerical error, proportional to Δt, because of the backward difference involved. Therefore, there will always be some value of N_p, beyond which the numerical error more than offsets the effects of smoothing out the stochastic noise in the calculations. However, since the Δt used in the bead-rod simulations are typically very small, the value of $N_p = 20$ suggested here should work well for most calculations. Moreover, it should be pointed out that use of the modified Giesekus expression with smoothing of the derivative could be very memory intensive in a complex flow situation where one might need to store the values of all stress components at every node for 20 time steps. Therefore a combination of both techniques might be an optimal method that one could use in complex flow situations, i.e. Kramers–Kirkwood with noise reduction procedure of Doyle et al. [10] during strong transients and modified Giesekus for steady and nearly steady flows.

4.2. Bead-spring chains

Fig. 5 depicts the CPU time ratio between the explicit method and either of semi-implicit predictor–corrector techniques proposed in this study for different FENE chain lengths. The test flow is start-up of simple shear flow at $We = 10$ and we iterate the predictor–corrector method to full convergence. From the figure, one can observe that the explicit scheme is at least four times faster than the predictor–corrector method. This can be explained based on the fact that there are at least three sub-steps at each time for the new method. In addition, except for the predictor step, the remaining steps in the new algorithm are semi-implicit steps which are more computationally intensive than an explicit step. Finally, the new scheme also involves iterations at each time step to ensure convergence, which requires additional CPU time.

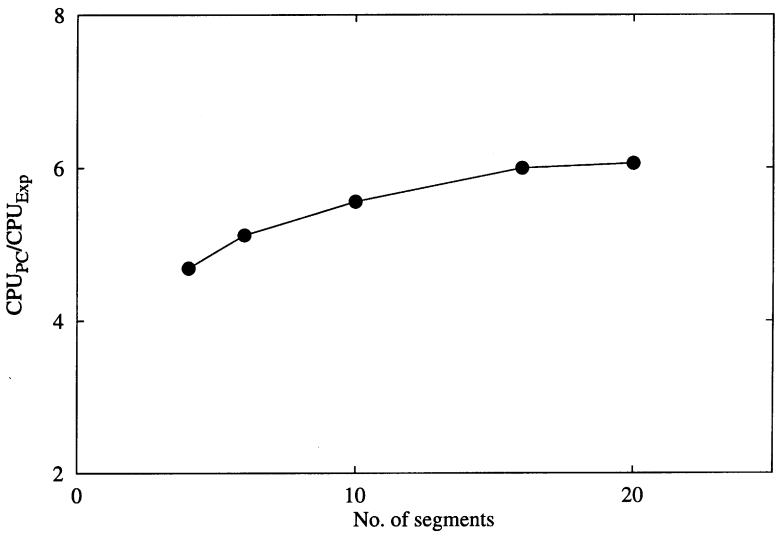

Fig. 5. CPU time ratio for the predictor–corrector and the explicit for FENE chain simulations of different sizes in shear flow at $We = 10$.

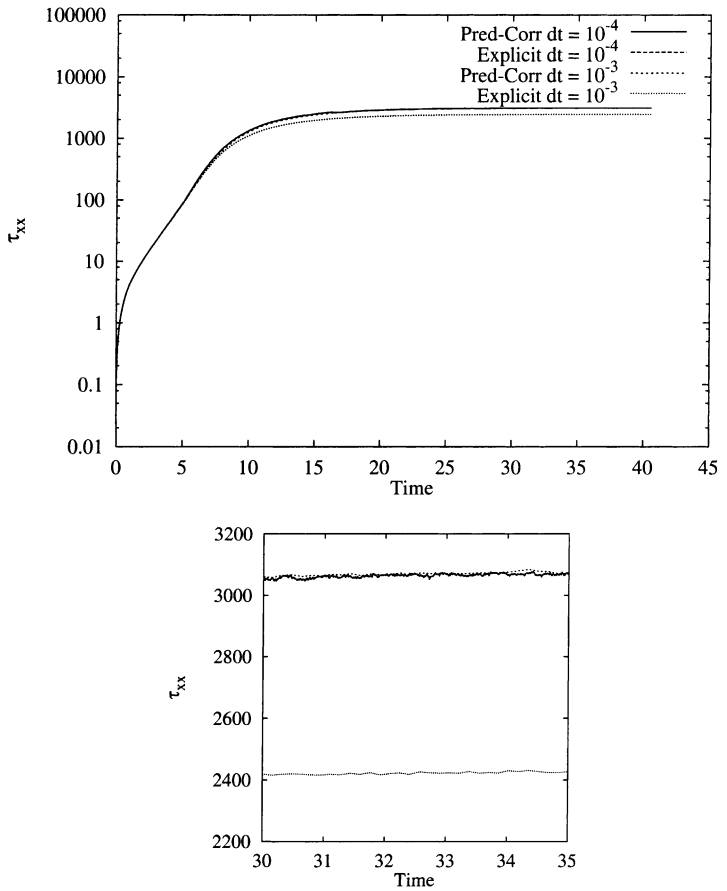

Fig. 6. Comparison of the predictor–corrector and explicit methods for four segment FENE chains in extensional flow at $We = 10$: evolution of τ_{xx} with time. Bottom figure shows comparison after steady state has been achieved.

Fig. 6 shows the comparison (in terms of the solution accuracy) between the explicit and either semi-implicit method for a four segment chain in start-up of uniaxial extensional flow at $We = 10$ for two different time steps, namely $dt = 0.0001$ and 0.001. It can be observed that, for the smaller time step, both the explicit and the predictor–corrector method give identical results. Table 1 shows the relative CPU times corresponding to Fig. 6. One can clearly observe that the explicit scheme is almost four times faster than the predictor–corrector method at any given time step. However, when the time step is increased to 0.001, we observe that only the predictor–corrector scheme gives the correct transients

Table 1
CPU times (s) for the solutions obtained using the predictor–corrector and explicit schemes as shown in Fig. 6

Δt	Predictor–corrector	Explicit
0.0001	8077	2255
0.001	1112	224

and steady value for the stresses. In addition, it can be further observed that the predictor–corrector techniques are generally two to three times faster than the explicit method for a given solution accuracy. The inaccuracy associated with the explicit technique with relatively large time steps is due to the rejection of moves that lead to aphysical spring force laws. Hence, over the course of the simulation, these rejections could significantly alter the true evolution of the chain dynamics. Although one can use an adaptive time step to make sure no rejections take place during the simulation, this would increase the number of computations performed at each time step. Hence, the advantages of a single step explicit method that is highly CPU efficient would be lost. More importantly, in self-consistent complex flow simulations, where the deformation rate could rapidly vary from position to position in the flow domain, the use of the predictor–corrector methods is essential as they do not lead to rejections and they can be used with much larger time steps. This in turn, minimizes the number of times that the macoscopic conservation laws have to be solved resulting in even more CPU and memory savings.

The explicit method combined with the Picard iterations ensures self-consistent solutions at each time step. Hence, larger time steps could be used in the simulations. However, the rejections could still alter the chain dynamics. Fig. 7 depicts the results obtained using the predictor–corrector method with full convergence and the explicit/Picard method for a six-segment FENE chain in the start-up of shear flow. Once again, an order of magnitude smaller time step is required for the explicit/Picard method in order to reproduce the same accuracy in the results as the predictor–corrector technique. Hence, generally speaking, to produce the same quality results the combined explicit/Picard's method is approximately three to four times slower than the predictor–corrector method.

To this point, the superiority of the newly proposed predictor–corrector methods relative to explicit methods has been demonstrated. Next, we shall focus on the performance of the fully implicit Newton's method. Two different variations of the method have been examined. The first is the Newton's method with a direct solver, wherein the Jacobian matrix is inverted using a direct banded solver at every iteration. The second is an iterative solver, where the resulting linear system of equations is iteratively solved *within*

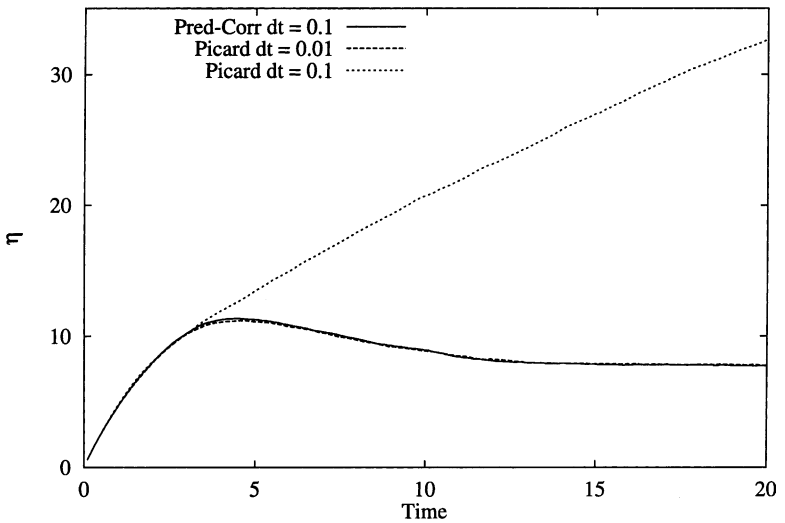

Fig. 7. Comparison of the solution obtained using the predictor–corrector method and the implicit Picard's iterative method for six segment FENE chain in shear flow (dimensionless $We = 10$).

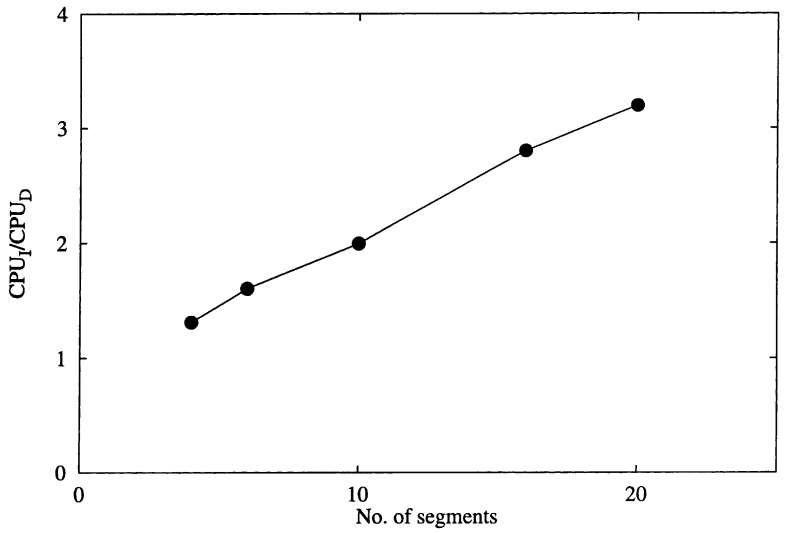

Fig. 8. The ratio of the CPU run time for Newton's method using direct and iterative solvers for various FENE chain sizes.

every iteration. Fig. 8 shows the CPU efficiency of the Newton's method with a direct solver and an iterative solver. Clearly, over the range of chain sizes considered, iterative solvers are much slower as compared to direct solvers. This is due to the fact that the Jacobian matrix for the nonlinear bead-spring chains considered is quite small and hence it is more efficient to invert the matrix in one step rather than iteratively solving the linear system. However, if a very large number of segments are used (i.e. <100) it is possible that the iterative solver could be more efficient. However, it should be borne in mind that the whole idea behind coarse graining is to reduce the degrees of freedom in the problem. Hence, for all practical chain sizes, the iterative solver is not competitive. Fig. 9 shows the relative CPU times between the Newton's method (with a direct solver at every iteration) and either of our predictor–corrector algorithms with iteration to convergence for different chain sizes. It can be seen that the predictor–corrector methods are at least twice as fast as the Newton's method for small chain sizes and in fact, the predictor–corrector technique becomes much more CPU efficient for larger chain sizes. This is expected, because as the chain size increases, the Jacobian matrix is more expensive to compute, and furthermore inverting the large matrix also becomes more time consuming. Thus Newton's method becomes less competitive as the chain size becomes larger.

In order to make the newly proposed predictor–corrector methods even more CPU efficient, we have incorporated the idea of a 'look-up' table in the algorithm. This idea follows from the fact that, in both Steps 2 and 3 of our algorithms (i.e. Eqs. (34) and (36), or (38) and (39)) we have to solve a cubic equation for the length of the connector vector (see Eqs. (35) and (40)). Closer inspection of either cubic equation reveals that, while the coefficient of the linear term is a constant throughout the simulations, the other two coefficients (quadratic and constant) are functions of only the magnitude $\boldsymbol{R}$ of the right-hand side vector of Eq. (34) or (38). Therefore, one can solve this cubic equation at the beginning of the simulation just once for different values of $\boldsymbol{R}$ and store the roots in a look-up table. During the simulation, the root can be extracted by performing a linear interpolation from the values in the look-up table. Fig. 10 shows a typical plot of the cubic root plotted against the parameter $\boldsymbol{R}$ for a maximum extensibility of $b = 900$

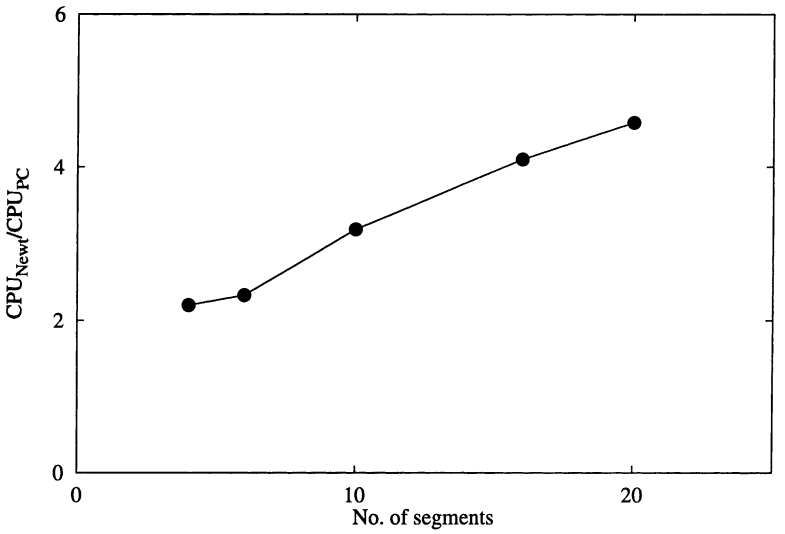

Fig. 9. The ratio of the CPU run time for Newton's method (with a direct solver) and predictor–corrector method for various FENE chain sizes.

from Eq. (35). From the figure, it is obvious that the linear interpolation should work well, since the root is a very smooth function of the parameter $\boldsymbol{R}$. In Fig. 11, the CPU time ratio with and without the look-up table formulation for the predictor–corrector technique defined by Eqs. (36) and (39) is plotted as a function of the chain size for $We = 1.0$ and maximum extensibility (of every individual spring in the chain) of 900. It can be observed that incorporating the look-up table into the algorithm more than

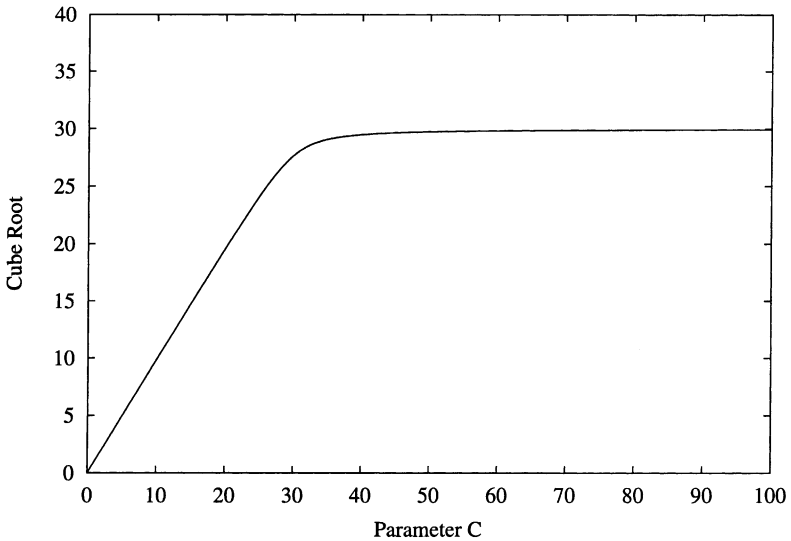

Fig. 10. The cube root for a dimensionless extensibility of 900 as a function of the parameter $\boldsymbol{R}$, which is the magnitude of the right-hand side of Eq. (35).

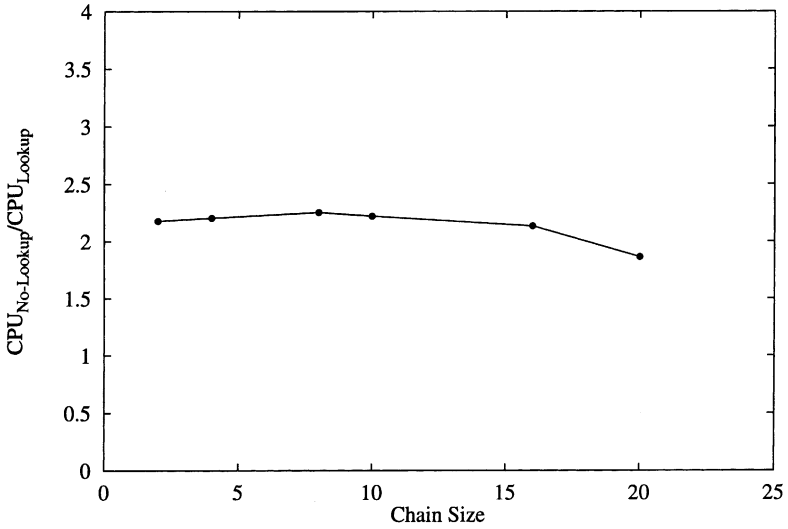

Fig. 11. The CPU ratio for runs without and with the use of look-up table for different FENE chain sizes.

doubles the computational efficiency of the BD simulations of the bead-spring chains, and this effect is virtually independent of the chain size. Since stochastic simulations are usually the most expensive part of all complex micro–macro computations, any improvement in making these simulations faster helps make the complex flow problem more tractable.

4.3. Comparison of bead-rod and bead-spring chain models

It is well known that a Kramers chain can both qualitatively and quantitatively capture the behavior of polymer chains in simple flows [4,10,12]. More recently, a Kramers chain matching the molecular parameters of a λ-DNA molecule has shown excellent agreement with single-molecule microscopy experiment [18–21]. However, since it is impossible to perform self-consistent simulations with this micro-structural model in complex flows with present day computing power, it becomes imperative that one has a coarse-graining recipe to reproduce the predictions of Kramers chains using structurally simpler models. The models often chosen for this purpose are the bead-spring chains. In order to compare results between bead-rod and bead-spring models, it is worthwhile first to point out the relationship between the parameters of the two models. The length of a fully stretched Kramers chain is given by $(N_k - 1)a$. Therefore the maximum extensibility of every spring in the corresponding N_s segment bead-spring chain is such that

$$N_s \times Q_0 = (N_k - 1)a \tag{42}$$

Correspondingly, the spring constant H_s of an individual spring in the bead-spring chain model is related to the bead-rod parameters as [12]

$$H_s = \frac{3kT}{(N_k - 1)a^2} N_s \tag{43}$$

Since $b = H_s Q_0^2/kT$, this implies that b can be written in terms of the bead-rod parameters as

$$b_s = \frac{3(N_k - 1)}{N_s} \tag{44}$$

Since the length scales in the two models have been fixed, we now turn our attention to the time scales involved in the coarse-graining procedure. We denote the time scale of the FENE chains to be $t_s^* = \zeta/4H_s$, and the time scale for bead-rod chains to be $t_k^* = \zeta_k a^2/kT$, where ζ_k is the bead drag coefficient for a bead in the bead-rod chain. We have explored various possibilities for determining the scaling relationship between t_s^* and t_k^*. For example, one can relate the two by fixing the longest relaxation time for the two chains to be the same, i.e. we choose the parameters such that the longest relaxation time becomes independent of the chain length. This is done as follows. The longest relaxation time of the bead-rod chain has been determined by Doyle et al. [10] for different chain lengths. Specifically, they have determined the longest relaxation time by examining the long time relaxation of the stress and/or the birefringence of an ensemble of chains starting with initial configuration in which every chain is fully stretched in the z-direction. This is analogous to studying the relaxation after subjecting the chains to uniaxial extensional flow with Weissenberg number $We \to \infty$. By performing the above mentioned analysis, they have given the following best fit relation between the longest relaxation time λ and the chain size N_k

$$\lambda = 0.0142 N_k^2 \left(\frac{\zeta_k a^2}{kT}\right) = 0.0142 N_k^2 t_k^* \tag{45}$$

We have followed an analogous procedure with the FENE springs. We start with FENE chains almost fully stretched in the z-direction and follow the time evolution of the stress by letting the chains relax to equilibrium. The *dimensionless* longest relaxation time λ^d can then be obtained as -1/slope from the semi-log plot of stress versus time. Subsequently, t_s^* can be related to t_k^* by equating the longest relaxation time, i.e.,

$$t_s^* = \left(\frac{0.0142 N_k^2}{\lambda^d}\right) t_k^* \tag{46}$$

A second approach to relate the two time scales is by equating the time scale characterized by the zero-shear material functions of the bead-spring chains to that of the bead-rod chains. The characteristic time scale λ_0 is defined as the ratio of zero-shear first normal stress coefficient and zero-shear viscosity ($\lambda_0 = \Psi_{1,0}/2\eta_0^p$) and it is given by the following equations for bead-rod and FENE chains, respectively [12,22]

$$\lambda_0^{\text{Kramers}} = \frac{10N_k^3 - 12N_k^2 + 35N_k - 12}{900N_k} \tag{47}$$

$$\lambda_0^{\text{FENE}} = \frac{1}{60}\left(\frac{b_s}{b_s + 5}\right)\left[2N_s^2 + 7 - 12\frac{N_s^2 + 1}{N_s(b_s + 7)}\right]. \tag{48}$$

The performance of these time scales have been examined by performing a number of simulations. Overall, we find, for extensional flow, bead-spring chains, with either time scales, fail to quantitatively capture the transient behavior of the corresponding bead-rod chains (see Figs. 12 and 13). However, it should be noted that the scaling based on the longest relaxation time performs better than the zero-shear time scale for all flow strengths considered. Good agreement between bead-spring and bead-rod can be

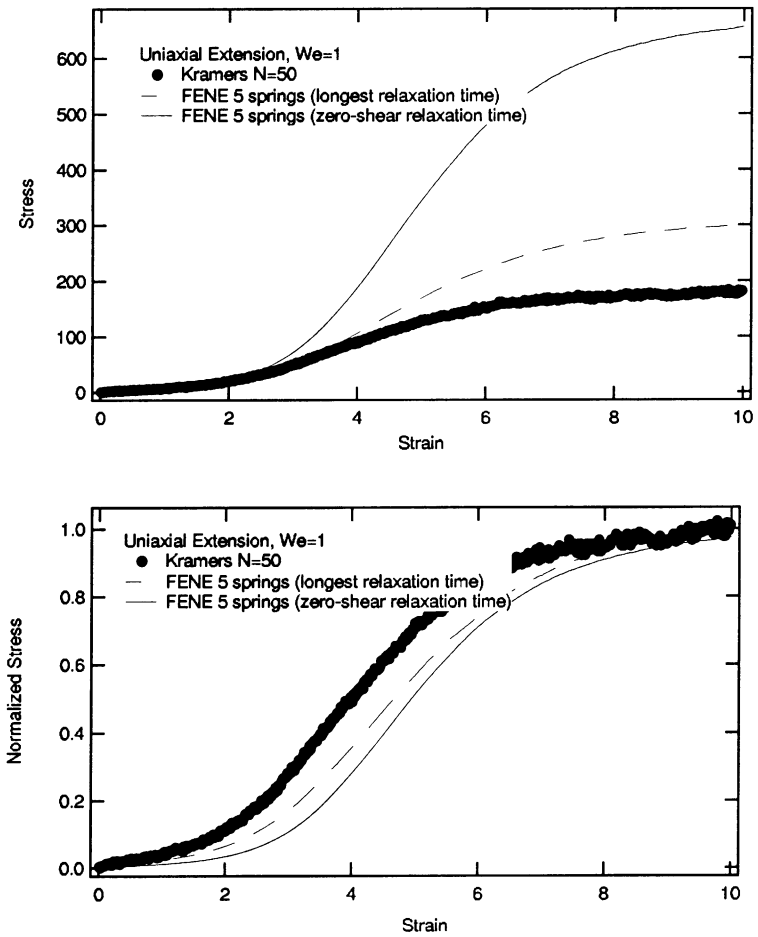

Fig. 12. Comparison between FENE chain predictions with 50 bead Kramers chain predictions for different time scaling relationships in extensional flow at $We = 1$.

obtained if the stress is normalized by its steady state value except for low values of We as shown in Fig. 12(b). This fact has been demonstrated elsewhere [19] in simulating the rheology of DNA solutions. Similarly only near-quantitative agreement between bead-rod chains and bead-spring chains is obtained for shear flow. Specifically, at low shear rate the zero-shear time scale is better in terms of capturing the transient behavior of the bead-rod as shown in Figs. 14 and 15, while at high shear rates the longest relaxation time scale outperforms the zero-shear time scale. Again good quantitative agreement can be obtained if the shear stress is rescaled by its steady value. For both shear and extensional flow, we find better agreement in stress response between the bead-rod chain and the corresponding bead-spring chain as the number of springs were increased up to the upper limit given by Eq. (49). The effect of the chain size N_k on transient response of shear stress is discussed elsewhere [21].

Though we have just shown that bead-spring chains mimic the bead-rod chains only semi-quantitatively, a consistent coarse-graining of bead-spring chains that gives the optimal agreement with bead-rod chains

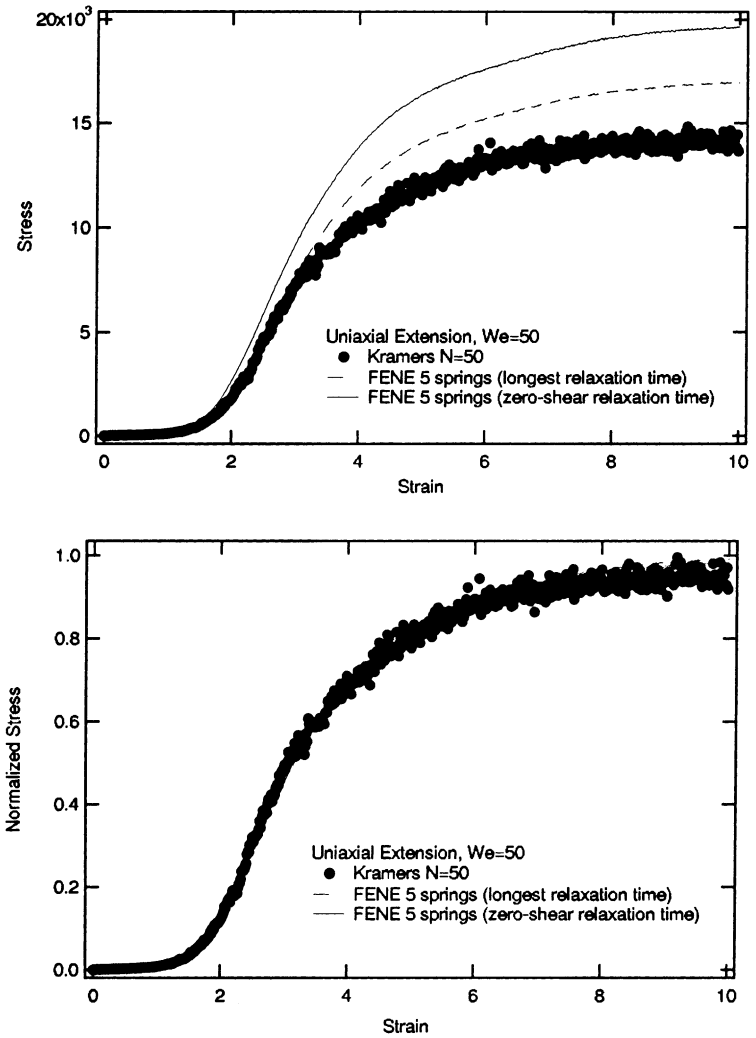

Fig. 13. Comparison between FENE chain predictions with 50 bead Kramers chain predictions for different time scaling relationships in extensional flow at $We = 50$.

can be developed. In the next section we discuss the number of springs in the bead-spring chains needed to best describe bead-rod chains under flow.

4.4. Coarse-graining procedure

Previously Doyle and Shaqfeh [23] compared the bead-rod chain with the FENE-PM and FENE chain in linear flows. They have used the same coarse-graining procedure as ours except that they used the Rouse time for their nonlinear bead-spring models instead of the longest relaxation time. As will be shown below, the nonlinear spring force law changes the relaxation time from the linear Rouse time about

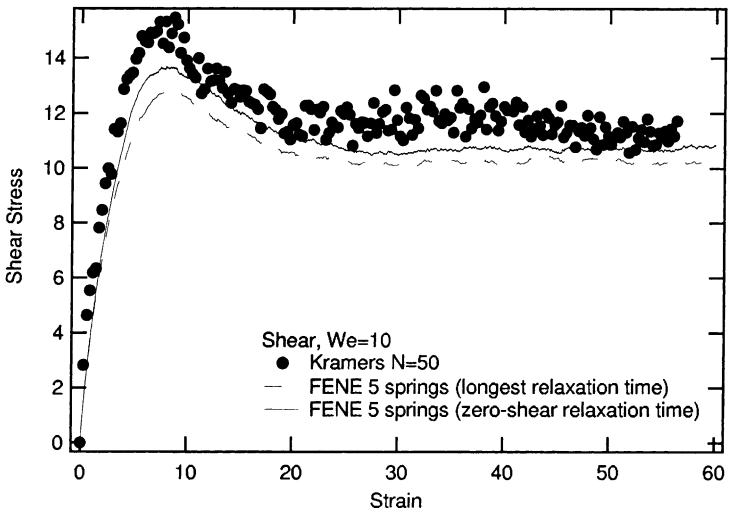

Fig. 14. Comparison between FENE chain predictions with 50 bead Kramers chain predictions for different time scaling relationships in shear flow at $We = 10$.

30–50% depending on the degree of discretization. Apart from their underestimation of *We*, we find our multi-spring FENE results qualitatively consistent with their FENE-PM result. Recently Ghosh et al. [24] proposed similar coarse-graining as ours except they rescaled their relaxation time for the bead-spring chain in order to get the same steady state extensional stress as the bead-rod chain. Therefore it is not surprising that their bead-spring chain result is in better agreement with the bead-rod chain in extensional flow than the results presented based on the two scalings used in this study. However in shear flow, we have found the agreement is poor if their scaling is used. Since in a complex flow, the kinematics is neither pure shear nor pure extensional, their procedure cannot be effectively used in complex flow simulations.

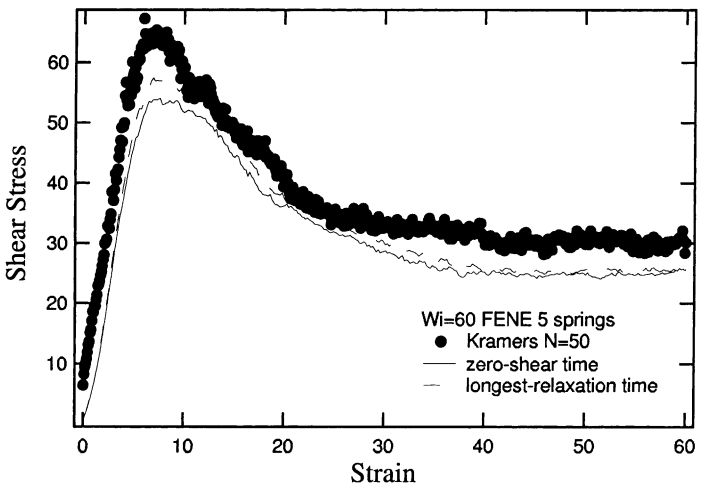

Fig. 15. Comparison between FENE chain predictions with 50 bead Kramers chain predictions for different time scaling relationships in shear flow at $We = 60$.

Table 2
The longest relaxation time of the worm-like chain with varying number of springs for $b = 450$

M (number of modes)	$b = 450$
5	1.369
10	4.189
15	8.749
20	14.12
30	27.01
40	44.58

In order to accurately simulate a polymer chain using the proposed numerical scheme in this paper, we need systematic criteria for the extent of coarse-graining. In other words, given a number of Kuhn steps (N_K) we want to find the minimum number of springs to best capture the behavior of a polymer chain. Previously Doyle et al. [10] found that only Kramers chains consisting of more than 10 rods follow the analytic inverse Langevin force law. Thus we have the upper bound limit as

$$N_s \leq \frac{N_K}{10}. \tag{49}$$

Yet a quantitative criterion for the lower bound on the number of springs is not available. Qualitatively, we have to include as many internal modes as needed to accurately describe the details of the dynamics. Before discussing the coarse-graining procedure, we point out the fact that when a single-spring chain is discretized into a multi-spring chain, the force on the chain segments becomes stiffer resulting in higher force than the single-spring force law. To accommodate this, Hur et al. artificially increased the Kuhn length size by 45% for their worm-like chain model for λ-DNA [19]. But since the deviation of the force is a nonlinear function of the chain extensions, a satisfactory rescaling of the force cannot be obtained by simply adjusting the Kuhn length alone. Also the amount by which the Kuhn length should be increased cannot be determined unambiguously. Our decision not to rescale the Kuhn length is also based on the reasoning that at small chain extensions, the spring force law should reduce to the Hookean limit with the correct molecular constants with no adjustable parameters.

First we calculate the longest relaxation time of the worm-like chain discretized into an N_s spring chain. The relaxation times obtained from birefringence decay are used. We choose $b = 450$ which is the molecular parameter for a λ-DNA molecule [19] and $b = 2000$ for comparison. The individual extensibility parameters b_s are chosen according to Eq. (44). The longest relaxation times of the worm-like chain with varying N_s are summarized in Tables 2 and 3. As N_s increases, each spring represents a shorter fragment of the molecule, resulting in a stiffer spring. As shown in Fig. 16, as we discretize the worm-like chain into a finer grained chain with more springs at a fixed number of Kuhn steps, the deviation of

Table 3
The longest relaxation time of the worm-like chain with varying number of springs for $b = 2000$

M (number of modes)	$b = 2000$
5	1.529
20	17.138
40	60.79

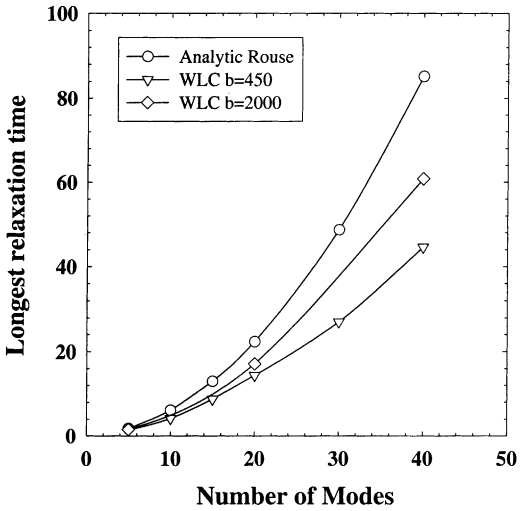

Fig. 16. The longest relaxation time of the worm-like chain with varying number of springs.

the longest relaxation time from the Rouse relaxation time at the same number of modes (N_s) increases. Moreover, the deviation is more pronounced for smaller b_s. We would expect the longest relaxation times of the worm-like chain to approach the Rouse limit for $b_s \gg 1$, however, for small b_s, the longest relaxation time is considerably different as mentioned in the beginning of this section and should be carefully simulated as discussed in Section 4.3.

To test the convergence of the rheological properties as well as the microscopic properties of chains with varying numbers of springs, we have performed simulations of a λ-DNA molecule in shear flow. In Fig. 17, we have plotted the polymer shear viscosity versus strain in the start-up shear flow for $N_s = 5$, 10, 15 and 40 at $We = 60$. A qualitative and quantitative difference is observed for $M > 15$ or $M < 5$.

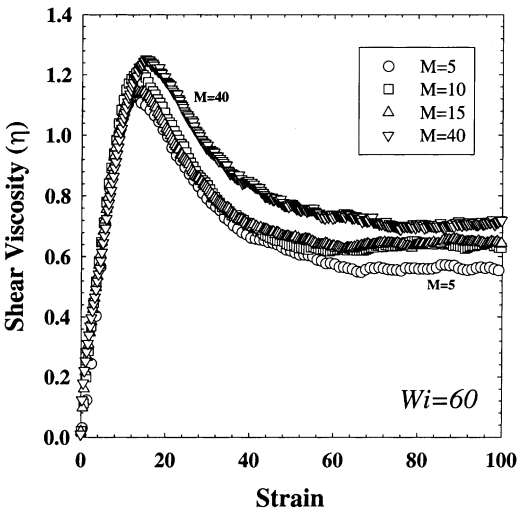

Fig. 17. The transient shear viscosity of the worm-like chain with varying number of springs at $We = 60$.

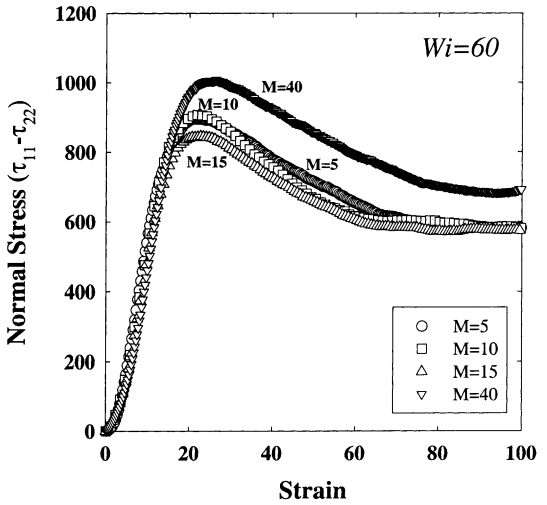

Fig. 18. The transient normal stress difference ($\tau_{11} - \tau_{22}$) of the worm-like chain with varying number of springs in shear flow at $We = 60$.

Based on Eq. (49) a worm-like chain of 15 springs would be an adequate coarse-grained model for a λ-DNA molecule and the prediction is in agreement with the simulation result shown in Fig. 17. The normal stress difference in the start-up of shear flow is shown for varying number of springs in Fig. 18 and again we see a significant deviation for chains with more than 15 springs.

Based on these results, we conclude that $N_s = 15$ is the number of springs necessary to mimic the behavior of a λ-DNA molecule. Though we have based our discussion of coarse-graining on a particular spring model (worm-like chain), the same procedure applies to other spring laws such as FENE chain and inverse Langevin chain.

5. Conclusions

In this study, we have considered two issues related to molecular simulations of viscoelastic flows with prescribed kinematics employing bead-rod and bead-spring chain models. First, various numerical schemes used to simulate such models have been investigated in terms of their solution accuracy and CPU times. Secondly, the issue of coarse graining from the expensive bead-rod chains to simpler bead-spring chains has been examined.

Specifically, the relative performance of two methods, namely the Picard's method and the Newton's method has been evaluated for different chain sizes and different flow strengths for the Kramers chain (bead-rod chain). It has been demonstrated that the Picard's iterative technique is almost 50% faster than the Newton's method, while both providing identical results for polymer stress and configuration. Furthermore, the modified Giesekus and the Kramers–Kirkwood expressions for the stress tensor have been compared both under transient and steady state conditions. In steady flows, it has been shown that numerical implementation of the Giesekus stress expression out-performs the Kramers–Kirkwood expression. However, the Giesekus expression involves calculating time derivatives of stochastic equations, which hinder the quality of the solution during initial transients of a simulation. For simple flow conditions,

we have developed a technique, which overcomes this drawback and makes the Giesekus expression's performance similar to that of the Kramers–Kirkwood expression with the noise filtering suggested by Doyle et al. [10]. However, this technique cannot be extended to complex flow situations because of the additional cost of storage of large amounts of data.

Similarly for the bead-spring chains, various different numerical methods such as; the explicit Euler method, the Picard's method, the Newton's method (with direct and iterative solvers) and two new iterative predictor–corrector techniques have been examined. As expected, for a fixed time step the explicit technique is much faster than the other techniques. However, the explicit technique does not guarantee that the extensibility of the springs is less than the maximum permissible extensibility. In order to overcome this, the Euler's method is modified by including the rejection algorithm as suggested by Öttinger. Although the rejection algorithm minimizes violation of spring law, the explicit technique is not able to perform favorably against the new predictor–corrector techniques. It has also been shown that the new predictor–corrector techniques are superior to Newton's method (with either solver) for all chain sizes.

The issue of coarse graining from a bead-rod to a bead-spring chain has also been investigated. The length scales of the two models are chosen such that the maximum extensibilities for both models are equal. However, for the relative time scales, two different relationships have been examined: equating the longest relaxation time and equating the zero-shear characteristic time. It has been demonstrated that either scaling is successful in providing only a semi-quantitative description of the transient response of bead-rod chains in flow. We have continued however and presented a coarse-graining procedure necessary to determine the optimal number of springs needed to best represent the corresponding bead-rod chain. Since quantitatively accurate representation of the bead-rod chains by bead-spring chains would be important when studying a complex flow problem, this issue of accurate coarse-graining procedures should remain an active area for future research. In this study we have found that for a short chain such as λ-DNA ($N_k = 150$), the maximum number of permissible springs given by Eq. (49) should be used. For much larger chains such as high molecular weight polymers, we expect much smaller number of springs than that specified by Eq. (49) can do an adequate job of describing the dynamics well. But since this upper limit is still too large (on the order of thousands) even for bead-spring models, further research is needed to mimic the high molecular weight chain dynamics with a small number of springs.

Acknowledgements

Some of the FENE chain and Kramers chain simulation results presented in this paper were performed by undergraduate students Tareq Al-Amen and Karen Leslie during their independent study projects in Bamin Khomami's group. Furthermore, B.K. wishes to acknowledge the National Science Foundation, which provided financial support for this work through grant CTS-97325535. E.S.G.S. would like to thank the National Science Foundation for supporting this work through grant CTS-9731896-002.

References

[1] M.J. Crochet, Numerical simulation of viscoelastic flow: a review, Rubber Chem. Technol. 62 (1989) 426–455.
[2] R.A. Brown, M.A. Szady, P.J. Northey, R.C. Armstrong, On the numerical stability of mixed finite element methods for viscoelastic flows governed by differential constitutive equations, Theoret. Comput. Fluid Dyn. (1993) 677–706.

[3] B. Khomami, K.K. Talwar, H.K. Ganpule, A comparative study of higher- and lower-order finite element techniques for computation of viscoelastic flows, J. Rheol. 38 (1994) 255–289.
[4] P.S. Doyle, E.S.G. Shaqfeh, G.H. Mckinley, S.H. Spiegelberg, Relaxation of dilute polymeric solutions following extensional flow, J. Non-Newtonian Fluid Mech. 76 (1998) 79–110.
[5] H.C. Öttinger, M. Laso, Smart polymers in finite element calculations, theoretical and applied rheology, in: P. Moldenaers, R. Keunings (Eds.), Proceedings of the International Congress on Rheology, Elsevier, Brussels, 1992.
[6] M. Laso, H.C. Öttinger, Calculation of viscoelastic flow using molecular models: the CONNFFESSIT approach, J. Non-Newtonian Fluid Mech. 47 (1993) 1–20.
[7] M.A. Hulsen, A.P.G. van Heel, B.H.A.A. van den Brule, Simulation of viscoelastic flows using Brownian configuration fields, J. Non-Newtonian Fluid Mech. 70 (1997) 79–101.
[8] P. Halin, G. Lielens, R. Keunings, V. Legat, The Lagrangian particle method for macroscopic and micro–macro viscoelastic flow computations, J. Non-Newtonian Fluid Mech. 77 (1998) 153–190.
[9] I. Ghosh, G.H. McKinley, R.A. Brown, R.C. Armstrong, An adaptive length scale method for dilute polymer solutions, in: Proceedings of the XIIIth International Congress on Rheology, Cambridge, UK, August 2000.
[10] P.S. Doyle, E.S.G. Shaqfeh, A.P. Gast, Dynamic simulation of freely draining flexible polymers in steady linear flow, J. Fluid Mech. 334 (1997) 251–291.
[11] H.C. Öttinger, Stochastic Processes in Polymeric Fluids, Springer, Berlin, 1996.
[12] R.B. Bird, C.F. Curtiss, R.C. Armstrong, O. Hassager, Dynamics of Polymeric Liquids, vol. 2, Wiley, New York, 1987, p. 40.
[13] J.F. Marko, E.D. Siggia, Stretching DNA, Macromolecules 28 (1991) 3427–3433.
[14] A. Cohen, A Pade approximant to the inverse Langevin function, J. Rheol. Acta 30 (1991) 270–272.
[15] T.W. Liu, Flexible polymer chain dynamics and rheological properties in steady flows, J. Chem. Phys. 90 (1989) 5826–5842.
[16] P.S. Grassia, E.J. Hinch, L.C. Nitsche, Computer simulations of Brownian motion of complex systems, J. Fluid Mech. 282 (1995) 373–403.
[17] G.H. Golub, C.V. Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[18] D.E. Smith, H.P. Babcock, S. Chu, Single polymer dynamics in steady shear flow, Science 283 (1999) 1724–1727.
[19] J.S. Hur, E.S.G. Shaqfeh, R.G. Larson, Brownian dynamics simulations of single DNA molecules in shear flow, J. Rheol. 44 (2000) 713–742.
[20] H.P. Babcock, D.E. Smith, J.S. Hur, E.S.G. Shaqfeh, S. Chu, Relating the microscopic and macroscopic response of a polymeric fluid in a shearing flow, Phys. Rev. Lett. 85 (2000) 2018–2021.
[21] J.S. Hur, E.S.G. Shaqfeh, H.P. Babcock, D.E. Smith, S. Chu, Dynamics of dilute and semidilute DNA solutions in the start-up of shear flow, J. Rheol. 45 (2001) 421–450.
[22] J.M. Wiest, R.I. Tanner, Rheology of bead-nonlinear spring chain macromolecules, J. Rheol. 33 (1989) 281–316.
[23] P.S. Doyle, E.S.G. Shaqfeh, Dynamic simulation of freely draining flexible bead-rod chains: start-up of extensional and shear flow, J. Non-Newtonian Fluid Mech. 76 (1998) 43–78.
[24] I. Ghosh, G.H. McKinley, R.A. Brown, R.C. Armstrong, Deficiencies of FENE dumbbell models in describing the rapid stretching of dilute polymer solutions, J. Rheol. 45 (2001) 721–758.