Continuum-Microscopic Computation of Non-Equilibrium Viscoelastic Flow

Sorin Mitran

Applied Mathematics
University of North Carolina
Chapel Hill

IMA 2009-10 Complex Fluids and Complex Flows
Flowing Complex Fluids Workshop
Fluid Mechanics-Interaction of Microstructure and Flow
October 14, 2009
Outline

1. Implications from Computer Architecture
   - Personal Bias
   - Computation Hardware

2. FENE micro-macro problem
   - Equations
   - Micro-macro algorithms

3. Some Interesting Algorithms
   - Parareal
   - Density estimation
   - Double projection

4. FENE micro-macro algorithm
   - Configuration density function

5. Conclusion
   - Conclusions
1. **Implications from Computer Architecture**
   - Personal Bias
   - Computation Hardware

2. **FENE micro-macro problem**
   - Equations
   - Micro-macro algorithms

3. **Some Interesting Algorithms**
   - Parareal
   - Density estimation
   - Double projection

4. **FENE micro-macro algorithm**
   - Configuration density function

5. **Conclusion**
   - Conclusions
• Look at current developments in computer architecture
• Look at current developments in computer architecture
• Find interesting applications requiring computation (viscoelastic flow)
• Look at current developments in computer architecture
• Find interesting applications requiring computation (viscoelastic flow)
• Develop algorithms, numerical analysis to support a link between the above
The Supercomputers

- LANL Roadrunner
The Supercomputers

- LANL Roadrunner

- $1.3 \times 10^5$ cores, $1.5 \times 10^{15}$ FLOPS
The Supercomputers

- LANL Roadrunner

- $1.3 \times 10^5$ cores, $1.5 \times 10^{15}$ FLOPS
- $133$M, $2.3$MWatts
The Supercomputers

- LANL Roadrunner

- $1.3 \times 10^5$ cores, $1.5 \times 10^{15}$ FLOPS
- $133M$, $2.3M$ Watts
- $11.3$ MFLOPS/$$, $0.7$ GFLOPS/Watt
• ORNL Jaguar
• ORNL Jaguar

• $1.5 \times 10^5$ cores, $1.4 \times 10^{15}$ FLOPS
• ORNL Jaguar

• $1.5 \times 10^5$ cores, $1.4 \times 10^{15}$ FLOPS

• $104\text{M}, 5-10\text{MWatts}$
• ORNL Jaguar

- $1.5 \times 10^5$ cores, $1.4 \times 10^{15}$ FLOPS
- $104$M, 5-10MWatts
- 13.5 MFLOPS/$, 0.2$ GFLOPS/Watt
• Intel Core i7 — IBM Cell
The CPUs

- Intel Core i7 — IBM Cell

- 4 cores, $70 \times 10^9$ FLOPS — 8 cores, $120 \times 10^9$ FLOPS
The CPUs

- Intel Core i7 — IBM Cell

- 4 cores, $70 \times 10^9$ FLOPS — 8 cores, $120 \times 10^9$ FLOPS

- $1,000$, 850 Watts — $70$, 40 Watts
The CPUs

- Intel Core i7 — IBM Cell

- 4 cores, $70 \times 10^9$ FLOPS — 8 cores, $120 \times 10^9$ FLOPS
- $1,000, 850$ Watts — $70, 40$ Watts
- $82$ MFLOPS/$\$, $0.1$ GFLOPS/Watt — $1.7$ GFLOPS/$\$, $3.0$ GFLOPS/Watt
- Folding@Home
- Folding@Home

- $8.1 \times 10^{15}$ FLOPS (5 × better than LANL Roadrunner)
Nonequilibrium Micro-Macro

Sorin Mitran

Outline
Implications from Computer Architecture
Personal Bias
Computation Hardware

FENE micro-macro problem
Equations
Micro-macro algorithms

Some Interesting Algorithms
Parareal
Density estimation
Double projection

FENE micro-macro algorithm
Configuration density function

Conclusion
Conclusions

The GPUs
- Tesla C1060
Nonequilibrium Micro-Macro

Sorin Mitran

Outline

Implications from Computer Architecture

Personal Bias Computation Hardware

FENE micro-macro problem Equations Micro-macro algorithms

Some Interesting Algorithms

Parareal Density estimation Double projection

FENE micro-macro algorithm Configuration density function

Conclusion

Conclusions

The GPUs

- Tesla C1060

- 240 cores, $10^{12}$ FLOPS
The GPUs

- **Tesla C1060**
  - 240 cores, $10^{12}$ FLOPS
  - $1,300, 200$ Watts
The GPUs

- Tesla C1060

- 240 cores, $10^{12}$ FLOPS
- $1,300, 200$ Watts
- 769 MFLOPS/$, 5.0$ GFLOPS/Watt
The GPUs

• Tesla C1060

• 240 cores, $10^{12}$ FLOPS
• $1,300$, 200 Watts
• 769 MFLOPS/$\$, 5.0 GFLOPS/Watt
• $(60, 7) \times$ better than LANL Roadrunner on (MFLOPS/$\$, GFLOPS/Watt)
Mixed-architecture supercomputers

- UNC BASS 452 CPUs, 180 GPUs, $2M
Some lessons to keep in mind

- Avoid large memory requirements
- Favor easily (trivially) parallelizable approaches
- Ensemble averaging
- No intercommunication/spatial decomposition
- Target low-accuracy (single precision) calculations

![Graph showing MFLOP/$ and GFLOP/Watt for different computing platforms.]
Some lessons to keep in mind

- Avoid large memory requirements
Some lessons to keep in mind

- Avoid large memory requirements
- Favor easily (trivially) parallelizable approaches
Some lessons to keep in mind

- Avoid large memory requirements
- Favor easily (trivially) parallelizable approaches
  - ensemble averaging
Some lessons to keep in mind

- Avoid large memory requirements
- Favor easily (trivially) parallelizable approaches
  - ensemble averaging
  - no intercommunication spatial decomposition
Some lessons to keep in mind

- Avoid large memory requirements
- Favor easily (trivially) parallelizable approaches
  - ensemble averaging
  - no intercommunication spatial decomposition
- Target low-accuracy (single precision) calculations
- Ensemble averaging
• Ensemble averaging
  • Simple cases can be run on GPUs (e.g. dumbbell SDE)
• Ensemble averaging
  • Simple cases can be run on GPUs (e.g. dumbbell SDE)
  • Complicated cases require CPUs (e.g. spatially extended Monte Carlo)
• **Ensemble averaging**
  - Simple cases can be run on GPUs (e.g. dumbbell SDE)
  - Complicated cases require CPUs (e.g. spatially extended Monte Carlo)

• **Spatial domain decomposition**
• Ensemble averaging
  • Simple cases can be run on GPUs (e.g. dumbbell SDE)
  • Complicated cases require CPUs (e.g. spatially extended Monte Carlo)

• Spatial domain decomposition
  • Works well if boundary data message time less than internal computation time
• **Ensemble averaging**
  - Simple cases can be run on GPUs (e.g. dumbbell SDE)
  - Complicated cases require CPUs (e.g. spatially extended Monte Carlo)

• **Spatial domain decomposition**
  - Works well if boundary data message time less than internal computation time
  - Usually implemented to require time step coupling
Parallelization

- Ensemble averaging
  - Simple cases can be run on GPUs (e.g. dumbbell SDE)
  - Complicated cases require CPUs (e.g. spatially extended Monte Carlo)
- Spatial domain decomposition
  - Works well if boundary data message time less than internal computation time
  - Usually implemented to require time step coupling
- Time domain decomposition
• Ensemble averaging
  • Simple cases can be run on GPUs (e.g. dumbbell SDE)
  • Complicated cases require CPUs (e.g. spatially extended Monte Carlo)

• Spatial domain decomposition
  • Works well if boundary data message time less than internal computation time
  • Usually implemented to require time step coupling

• Time domain decomposition
  • Multiple shooting (Parareal algorithm)
Parallelization

- Ensemble averaging
  - Simple cases can be run on GPUs (e.g. dumbbell SDE)
  - Complicated cases require CPUs (e.g. spatially extended Monte Carlo)

- Spatial domain decomposition
  - Works well if boundary data message time less than internal computation time
  - Usually implemented to require time step coupling

- Time domain decomposition
  - Multiple shooting (Parareal algorithm)
  - Deferred correction
Nonequilibrium Micro-Macro

Sorin Mitran

Outline

1. Implications from Computer Architecture
   Personal Bias
   Computation Hardware

2. FENE micro-macro problem
   Equations
   Micro-macro algorithms

3. Some Interesting Algorithms
   Parareal
   Density estimation
   Double projection

4. FENE micro-macro algorithm
   Configuration density function

5. Conclusion
   Conclusions
• Mass $\nabla \cdot \nu = 0$
• Mass $\nabla \cdot \nu = 0$

• Momentum

$$\partial_t \nu + \nu \cdot \nabla \nu = -\nabla p + \eta_s \nabla^2 \nu + \nabla \cdot \tau$$
**Continuum conservation laws**

- Mass: \( \nabla \cdot \mathbf{v} = 0 \)
- Momentum:
  \[
  \partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v} = -\nabla p + \eta_s \nabla^2 \mathbf{v} + \nabla \cdot \tau
  \]
- Viscoelastic stress evolution (Oldroyd-B):
  \[
  \partial_t \tau + \mathbf{v} \cdot \nabla \tau - \kappa \tau - \tau \kappa^T = -\tau / \lambda + \eta_p (\kappa + \kappa^T)
  \]
Nonequilibrium Micro-Macro

Sorin Mitran

Outline
Implications from Computer Architecture
Personal Bias Computation Hardware

FENE micro-macro problem

Equations
Micro-macro algorithms

Some Interesting Algorithms
Parareal Density estimation Double projection

FENE micro-macro algorithm
Configuration density function

Conclusion Conclusions

Continuum conservation laws

- Mass $\nabla \cdot \mathbf{v} = 0$
- Momentum
  \[ \partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v} = -\nabla p + \eta_s \nabla^2 \mathbf{v} + \nabla \cdot \mathbf{\tau} \]
- Viscoelastic stress evolution (Oldroyd-B)
  \[ \partial_t \mathbf{\tau} + \mathbf{v} \cdot \nabla \mathbf{\tau} - \kappa \mathbf{\tau} - \mathbf{\tau} \kappa^T = -\mathbf{\tau} / \lambda + \eta_p (\kappa + \kappa^T) \]
- $\kappa = \nabla \mathbf{v}$
• Viscoelastic stress \((Q\) end-to-end vector) \[\tau = -nk_B T I + nH \langle F(Q)Q \rangle\]
Kramers relations

- Viscoelastic stress ($Q$ end-to-end vector)
  \[ \tau = -nk_B T I + nH \langle F(Q)Q \rangle \]

- Configuration average
  \[ \langle f \rangle (x, t) = \int \Psi(x, t, Q) f(Q) dQ \]
• Viscoelastic stress \((Q\) end-to-end vector\)

\[
\tau = -nk_B T I + nH \langle F(Q) Q \rangle
\]

• Configuration average

\[
\langle f \rangle (x, t) = \int \Psi(x, t, Q)f(Q)dQ
\]

• Fokker-Planck equation for elongation

\[
\partial_t \Psi + v \cdot \nabla \Psi = -\nabla_Q \cdot \left[ (\kappa Q - 2F(Q)/\zeta)\Psi \right] + (2k_B T /\zeta) \nabla^2_Q \Psi
\]
• Viscoelastic stress \((Q \text{ end-to-end vector})\)
\[
\tau = -nk_B T I + nH \langle F(Q)Q \rangle
\]

• Configuration average
\[
\langle f \rangle (x, t) = \int \Psi(x, t, Q)f(Q)dQ
\]

• Fokker-Planck equation for elongation
\[
\partial_t \Psi + v \cdot \nabla \Psi = -\nabla Q \cdot [(\kappa Q - 2F(Q)/\zeta)\Psi] + (2k_B T/\zeta) \nabla^2_Q \Psi
\]

• Equivalent stochastic formulation
\[
dQ(x, t) + v(x, t) \cdot \nabla Q dt =
\]
\[
(\kappa(x, t)Q(x, t) - 2F(Q(x, t))/\zeta) dt + \sqrt{\frac{4k_B T}{\zeta}} dW(x, t)
\]
- CONNFFESSIT-type Brownian Dynamics (with variance reduction)
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
  • large execution time
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
  • large execution time
• Brownian Configuration Fields
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
  • large execution time
• Brownian Configuration Fields
  • reduce execution time
• **CONNFFESSIT-type Brownian Dynamics (with variance reduction)**
  - large memory requirement, many dumbbells
  - large execution time

• **Brownian Configuration Fields**
  - reduce execution time
  - large memory requirement, store configuration fields
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
  • large execution time
• Brownian Configuration Fields
  • reduce execution time
  • large memory requirement, store configuration fields
• Lagrangian Particle Method
• CONNFFESSIT-type Brownian Dynamics (with variance reduction)
  • large memory requirement, many dumbbells
  • large execution time

• Brownian Configuration Fields
  • reduce execution time
  • large memory requirement, store configuration fields

• Lagrangian Particle Method

• Brownian Dynamics - Distribution Function Storing - starting point for present algorithm - updates configuration p.d.f. \( \Psi \)
1 Implications from Computer Architecture
   Personal Bias
   Computation Hardware

2 FENE micro-macro problem
   Equations
   Micro-macro algorithms

3 Some Interesting Algorithms
   Parareal
   Density estimation
   Double projection

4 FENE micro-macro algorithm
   Configuration density function

5 Conclusion
   Conclusions
Parareal: Time domain decomposition

- Solve ODE \( u'(t) = f(t, u(t)) \)
Parareal: Time domain decomposition

- Solve ODE $u'(t) = f(t, u(t))$
- Advance over time step $[0, T]$
Parareal: Time domain decomposition

- Solve ODE $u'(t) = f(t, u(t))$
- Advance over time step $[0, T]$
- Divide into sub-intervals $[t_n, t_{n+1}]$, $t_n = n\Delta t$
Parareal: Time domain decomposition

- Solve ODE $u'(t) = f(t, u(t))$
- Advance over time step $[0, T]$
- Divide into sub-intervals $[t_n, t_{n+1}]$, $t_n = n\Delta t$
- Use $P$ processors, $\Delta t = T / P$
Parareal: Time domain decomposition

- Solve ODE $u'(t) = f(t, u(t))$
- Advance over time step $[0, T]$
- Divide into sub-intervals $[t_n, t_{n+1}]$, $t_n = n\Delta t$
- Use $P$ processors, $\Delta t = T/P$
- Each processor works on interval $[t_n, t_{n+1}]$
Parareal: Concurrent fast/slow algorithms

- $F(t_{n+1}, t_n, U_n)$ expensive algorithm
Parareal: Concurrent fast/slow algorithms

- $F(t_{n+1}, t_n, U_n)$ expensive algorithm
- $G(t_{n+1}, t_n, U_n)$ cheap algorithm
Parareal: Concurrent fast/slow algorithms

- \( F(t_{n+1}, t_n, U_n) \) expensive algorithm
- \( G(t_{n+1}, t_n, U_n) \) cheap algorithm
- Use \( G \) to compute initial approximation \( U_n^0 \)

\[
U_{n+1}^0 = G(t_{n+1}, t_n, U_n^0)
\]
Parareal: Concurrent fast/slow algorithms

- $F(t_{n+1}, t_n, U_n)$ expensive algorithm
- $G(t_{n+1}, t_n, U_n)$ cheap algorithm
- Use $G$ to compute initial approximation $U_n^0$

$$U_{n+1}^0 = G(t_{n+1}, t_n, U_n^0)$$

- Once each CPU has $U_n^0$ apply expensive algorithm $F(t_{n+1}, t_n, U_n^0)$
Parareal: Concurrent fast/slow algorithms

- $F(t_{n+1}, t_n, U_n)$ expensive algorithm
- $G(t_{n+1}, t_n, U_n)$ cheap algorithm
- Use $G$ to compute initial approximation $U_n^0$
  \[ U_{n+1}^0 = G(t_{n+1}, t_n, U_n^0) \]
- Once each CPU has $U_n^0$ apply expensive algorithm $F(t_{n+1}, t_n, U_n^0)$
- Apply correction
  \[ U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k) \]
• Converges if system eigenvalues are not dominated by imaginary part (convection dominated flow)
Parareal: Theory, performance

- Converges if system eigenvalues are not dominated by imaginary part (convection dominated flow)
- $P$ iterations is the same as applying expensive algorithm $F$
• Converges if system eigenvalues are not dominated by imaginary part (convection dominated flow)
• $P$ iterations is the same as applying expensive algorithm $F$
• Theoretical superlinear convergence of $U^n_k$ to $U^n_P$
• Converges if system eigenvalues are not dominated by imaginary part (convection dominated flow)
• \( P \) iterations is the same as applying expensive algorithm \( F \)
• Theoretical superlinear convergence of \( U_n^k \) to \( U_n^P \)
• Quadratic convergence often observed in practical computation
Density estimation: Histograms, kernels, orthogonal functions

- Sample of $f$-distributed random variable $X_1, X_2, \ldots, X_n$
Density estimation: Histograms, kernels, orthogonal functions

- Sample of $f$-distributed random variable $X_1, X_2, \ldots, X_n$
- Histogram estimate

\[
\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} N_i l_T(x)
\]
Density estimation: Histograms, kernels, orthogonal functions

- Sample of $f$-distributed random variable $X_1, X_2, \ldots, X_n$
- Histogram estimate
  \[
  \hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} N_i I_{T_i}(x)
  \]
- Kernel estimate
  \[
  \hat{f}_h(x) = \frac{1}{(nh)^{d-1}} \sum_{i=1}^{n} K \left( \frac{x - X_i}{h} \right)
  \]
Density estimation: Histograms, kernels, orthogonal functions

- Sample of $f$-distributed random variable $X_1, X_2, \ldots, X_n$
- Histogram estimate

\[ \hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} N_i I_{T_i}(x) \]

- Kernel estimate

\[ \hat{f}_h(x) = \frac{1}{(nh)^{d-1}} \sum_{i=1}^{n} K \left( \frac{x - X_i}{h} \right) \]

- Orthogonal functions

\[ \hat{f}(x) = \sum_{k=-\infty}^{\infty} a_k \varphi_k(x) \]
• Incompressibility acts as an overall elliptic
Operator splitting of conservation equations

- Incompressibility acts as an overall elliptic
- Operator splitting is often used to avoid large implicit systems
- Incompressibility acts as an overall elliptic
- Operator splitting is often used to avoid large implicit systems
  - Navier stokes $\nu$ without pressure
• Incompressibility acts as an overall elliptic
• Operator splitting is often used to avoid large implicit systems
  • Navier stokes $\nu$ without pressure
  • Oldroyd-B $\tau$ advanced first then $\nu$ without pressure
Incompressibility acts as an overall elliptic

Operator splitting is often used to avoid large implicit systems

- Navier stokes $\nu$ without pressure
- Oldroyd-B $\tau$ advanced first then $\nu$ without pressure
- micro-macro $\tau$ advanced from microscopic model
Operator splitting of conservation equations

- Incompressibility acts as an overall elliptic
- Operator splitting is often used to avoid large implicit systems
  - Navier stokes $\nu$ without pressure
  - Oldroyd-B $\tau$ advanced first then $\nu$ without pressure
  - micro-macro $\tau$ advanced from microscopic model
- Elliptic constraint imposed at end of time step through projection on divergence-free velocity fields
Operator splitting of conservation equations

- Incompressibility acts as an overall elliptic
- Operator splitting is often used to avoid large implicit systems
  - Navier stokes $\nu$ without pressure
  - Oldroyd-B $\tau$ advanced first then $\nu$ without pressure
  - micro-macro $\tau$ advanced from microscopic model
- Elliptic constraint imposed at end of time step through projection on divergence-free velocity fields
- Incompressibility prohibits longitudinal wave modes in viscoelastic fluids
Operator splitting of conservation equations

- Incompressibility acts as an overall elliptic
- Operator splitting is often used to avoid large implicit systems
  - Navier stokes $\nu$ without pressure
  - Oldroyd-B $\tau$ advanced first then $\nu$ without pressure
  - Micro-macro $\tau$ advanced from microscopic model
- Elliptic constraint imposed at end of time step through projection on divergence-free velocity fields
- Incompressibility prohibits longitudinal wave modes in viscoelastic fluids
- $\tau$ from operator splitting can contain spurious longitudinal modes-
Navier-Stokes projection method (Chorin, Temam)

- Advance velocity field disregarding pressure term

\[
\text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \nabla^2 \vec{v}^n - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n
\]
Non-equilibrium Micro-Macro

Sorin Mitran

Outline

Implications from Computer Architecture

Personal Bias Computation Hardware

FENE micro-macro problem Equations Micro-macro algorithms

Some Interesting Algorithms

Parareal Density estimation

Double projection

Navier-Stokes projection method (Chorin, Temam)

• Advance velocity field disregarding pressure term

\[
\text{Re} \left( \frac{\tilde{v}^* - \tilde{v}^n}{\Delta t} \right) = \nabla^2 \tilde{v}^n - \text{Re}(\tilde{v}^n \cdot \nabla) \tilde{v}^n
\]

• Compute a pressure that enforces \( \nabla \cdot \tilde{v}^{n+1} = 0 \)

\[
\nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \tilde{v}^*
\]
Navier-Stokes projection method (Chorin, Temam)

- Advance velocity field disregarding pressure term
  \[
  \text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \nabla^2 \vec{v}^n - \text{Re}(\vec{v}^n \cdot \nabla)\vec{v}^n
  \]

- Compute a pressure that enforces \( \nabla \cdot \vec{v}^{n+1} = 0 \)
  \[
  \nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^*
  \]

- Correct velocity field
  \[
  \text{Re} \left( \frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} \right) = -\nabla p^{n+1}
  \]
Navier-Stokes projection method (Chorin, Temam)

- Advance velocity field disregarding pressure term
  \[ \text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \nabla^2 \vec{v}^n - \text{Re}(\vec{v}^n \cdot \nabla)\vec{v}^n \]

- Compute a pressure that enforces \( \nabla \cdot \vec{v}^{n+1} = 0 \)
  \[ \nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^* \]

- Correct velocity field
  \[ \text{Re} \left( \frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} \right) = -\nabla p^{n+1} \]

- Method is essentially based upon Helmholtz decomposition
  \[ \forall \vec{v}, \exists \vec{A}, \phi : \vec{v} = \nabla \phi + \nabla \times \vec{A} \]
Oldroyd-B projection method (Van Kan, Bell, Gresho)

- Advance polymeric stress

\[
\text{We} \left( \frac{\tau^{n+1} - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
\text{We} \left( (\bar{v} \cdot \nabla) \tau - \tau \nabla \bar{v} - (\nabla \bar{v})^T \tau \right)
\]
Oldroyd-B projection method (Van Kan, Bell, Gresho)

- Advance polymeric stress

\[
\text{We} \left( \frac{\tau^{n+1} - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
\text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
\]

- Advance velocity field disregarding pressure term

\[
\text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla)\vec{v}^n + \nabla \cdot \tau^{n+1}
\]
Oldroyd-B projection method (Van Kan, Bell, Gresho)

- Advance polymeric stress
  \[
  \text{We} \left( \frac{\tau^{n+1} - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
  \text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
  \]

- Advance velocity field disregarding pressure term
  \[
  \text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n + \nabla \cdot \tau^{n+1}
  \]

- Compute pressure
  \[
  \nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^*
  \]
Nonequilibrium Micro-Macro
Sorin Mitran

Outline
Implications from Computer Architecture
Personal Bias
Computation Hardware

FENE micro-macro problem
Equations
Micro-macro algorithms

Some Interesting Algorithms
Parareal
Density estimation
Double projection

FENE micro-macro algorithm
Configuration density function

Conclusion
Conclusions

Oldroyd-B projection method (Van Kan, Bell, Gresho)

- Advance polymeric stress
  \[
  \text{We} \left( \frac{\tau^{n+1} - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
  \text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
  \]

- Advance velocity field disregarding pressure term
  \[
  \text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n + \nabla \cdot \tau^{n+1}
  \]

- Compute pressure
  \[
  \nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^*
  \]

- Correct velocity field
  \[
  \text{Re} \left( \frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} \right) = -\nabla p^{n+1}
  \]
Linear wave behavior

- Neglect convective nonlinearity, diffusion, source terms

\[ \nabla \cdot \vec{v} = 0 \]
\[ \rho \frac{\partial \vec{v}}{\partial t} = -\nabla p + \nabla \cdot \tau \]
\[ \frac{\partial \tau}{\partial t} = \frac{\eta}{\lambda_1} \left( \nabla \vec{v} + (\nabla \vec{v})^T \right) \]
Linear wave behavior

• Neglect convective nonlinearity, diffusion, source terms

\[ \nabla \cdot \vec{v} = 0 \]
\[ \rho \frac{\partial \vec{v}}{\partial t} = -\nabla p + \nabla \cdot \tau \]
\[ \frac{\partial \tau}{\partial t} = \frac{\eta_p}{\lambda_1} \left( \nabla \vec{v} + (\nabla \vec{v})^T \right) \]

• Linearized model cannot sustain \( P \)-waves

\[ [i+i] \]
Symmetric tensor decomposition

- Decompose into a divergence-free part and remainder

\[ \tau = \alpha + \beta \]
Symmetric tensor decomposition

- Decompose into a divergence-free part and remainder
  \[ \tau = \alpha + \beta \]
- Divergence-part is superfluous here, but can be found by
  \[
  \left\{ \begin{array}{l}
  \nabla \cdot \alpha = 0 \\
  \alpha \cdot \vec{n} = 0
  \end{array} \right. 
  \]
Symmetric tensor decomposition

- Decompose into a divergence-free part and remainder
  \[ \tau = \alpha + \beta \]

- Divergence-part is superfluous here, but can be found by
  \[
  \begin{align*}
  \nabla \cdot \alpha &= 0 \\
  \alpha \cdot \vec{n} &= 0
  \end{align*}
  \]

- Force-carrying part can be written as
  \[
  \beta = \frac{1}{2} \left[ \nabla \vec{u} + (\nabla \vec{u})^T \right]
  \]
Symmetric tensor decomposition

- Decompose into a divergence-free part and remainder
  \[ \tau = \alpha + \beta \]

- Divergence-part is superfluous here, but can be found by
  \[
  \begin{cases}
  \nabla \cdot \alpha = 0 \\
  \alpha \cdot \vec{n} = 0
  \end{cases}
  \]

- Force-carrying part can be written as
  \[ \beta = \frac{1}{2} \left[ \nabla \vec{u} + (\nabla \vec{u})^T \right] \]

- Displacement field is found by solving elliptic problem
  \[
  \nabla^2 \vec{u} + \nabla (\nabla \cdot \vec{v}) = 2 \nabla \cdot \tau \\
  \frac{1}{2} \left[ \left[ \nabla \vec{u} + (\nabla \vec{u})^T \right] \right] \cdot \vec{n} = -\tau \cdot \vec{n}
  \]
First Projection - polymeric displacement field

- Obtain polymeric stress prediction

\[
\text{We} \left( \frac{\tau^* - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
\text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
\]
First Projection - polymeric displacement field

- Obtain polymeric stress prediction

\[
\text{We} \left( \frac{\tau^* - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
\text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
\]

- Decompose into divergence-free, elastic-force carrying parts

\[
\tau^* = \alpha + \beta^*
\]

[i+-i]
First Projection - polymeric displacement field

- Obtain polymeric stress prediction

\[
\text{We} \left( \frac{\tau^* - \tau^n}{\Delta t} \right) = (1 - \beta) \dot{\gamma}^n - \tau^n - \\
\text{We} \left( (\vec{v} \cdot \nabla) \tau - \tau \nabla \vec{v} - (\nabla \vec{v})^T \tau \right)
\]

- Decompose into divergence-free, elastic-force carrying parts

\[
\tau^* = \alpha + \beta^*
\]

- Find deformation field of force-carrying part

\[
\beta^* = \frac{1}{2} \left[ \nabla \vec{u}^* + (\nabla \vec{u}^*)^T \right]
\]
First Projection - polymeric displacement field

- Compute correction to enforce incompressibility

\[ \nabla^2 \phi = -\nabla \cdot \bar{u}^* \]
First Projection - polymeric displacement field

- Compute correction to enforce incompressibility
  \[ \nabla^2 \phi = -\nabla \cdot \vec{u}^* \]

- Obtain updated deformation field
  \[ \vec{u}^{n+1} = \vec{u}^* + \nabla \phi \]
First Projection - polymeric displacement field

- Compute correction to enforce incompressibility

\[ \nabla^2 \phi = -\nabla \cdot \bar{u}^* \]

- Obtain updated deformation field

\[ \bar{u}^{n+1} = \bar{u}^* + \nabla \phi \]

- Obtain updated polymeric stress

\[ \tau^{n+1} = \frac{1}{2} \left[ \nabla \bar{u}^{n+1} + (\nabla \bar{u}^{n+1})^T \right] \]

[i+-i]
Second Projection - enforce continuity equation

- Velocity field prediction

\[
\text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n + \nabla \cdot \tau^{n+1}
\]
Second Projection - enforce continuity equation

- Velocity field prediction

\[ \text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n + \nabla \cdot \tau^{n+1} \]

- Compute pressure

\[ \nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^* \]
Second Projection - enforce continuity equation

- Velocity field prediction

\[
\text{Re} \left( \frac{\vec{v}^* - \vec{v}^n}{\Delta t} \right) = \beta \nabla^2 \vec{v} - \text{Re}(\vec{v}^n \cdot \nabla) \vec{v}^n + \nabla \cdot \tau^{n+1}
\]

- Compute pressure

\[
\nabla^2 p^{n+1} = \left( \frac{\text{Re}}{\Delta t} \right) \nabla \cdot \vec{v}^*
\]

- Obtain updated velocity field

\[
\text{Re} \left( \frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} \right) = -\nabla p^{n+1}
\]
Outline

1. Implications from Computer Architecture
   - Personal Bias
   - Computation Hardware

2. FENE micro-macro problem
   - Equations
   - Micro-macro algorithms

3. Some Interesting Algorithms
   - Parareal
   - Density estimation
   - Double projection

4. FENE micro-macro algorithm
   - Configuration density function

5. Conclusion
   - Conclusions
• Use $\Psi(t_n)$ to generate $Q_\alpha(t_n)$
• Use $\Psi(t_n)$ to generate $Q_\alpha(t_n)$

• Advance

$$Q_\alpha^{n+1} = Q_\alpha^n + (\kappa^n Q_\alpha^n - \frac{1}{2\lambda_H} F(Q_\alpha^n)) \Delta t + \sqrt{\frac{1}{\lambda_H}} \Delta W_\alpha^n$$
• Use $Ψ(t_n)$ to generate $Q_α(t_n)$

• Advance

$$Q_α^{n+1} = Q_α^n + (κ^n Q_α^n - \frac{1}{2λ_H} F(Q_α^n))Δt + \sqrt{\frac{1}{λ_H}}ΔW_α^n$$

• Reconstruct $Ψ(t_{n+1})$ (Histogram approximation)
Combine formulations within parareal

- **Slow (CPU):** solve Fokker-Plank equation

\[
\frac{\partial}{\partial t} \Psi = -\nabla_Q \cdot \left[ \left( \kappa Q - 2F(Q)/\zeta \right) \Psi \right] + \left( 2k_B T / \zeta \right) \nabla^2_Q \Psi
\]
Combine formulations within parareal

- **Slow (CPU):** solve Fokker-Plank equation

\[
\partial_t \Psi = - \nabla_Q \cdot \left[ (\kappa_Q - 2F(Q)/\zeta) \Psi \right] + (2k_B T/\zeta) \nabla^2_Q \Psi
\]

- **Fast (GPU):** advance

\[
Q_{\alpha}^{n+1} = Q_{\alpha}^n + (\kappa^n Q_{\alpha}^n - \frac{1}{2\lambda_H} F(Q_{\alpha}^n)) \delta t + \sqrt{\frac{1}{\lambda_H}} \delta W_{\alpha}^n
\]
• 2:1 ratio, $\text{We} = 0.3$

Figure 3: Comparison of computed viscoelastic cavity flow for an Oldroyd-B fluid at $\text{We} = 0.3$ using the double projection method with the experimental results of Pakdel et al. [16]
Cavity results

- 1:1 ratio, $We = 0.15$

Figure 4: As before, but for $We = 0.15$
• 1:2 ratio, $\textit{We} = 0.075$

Figure 5: As before, but for $\textit{We} = 0.075$
• Convergence of parareal iterations
• Convergence of parareal iterations

• Computational cost: 4-6 times less than SDE for equivalent precision
Outline

1. Implications from Computer Architecture
   Personal Bias
   Computation Hardware

2. FENE micro-macro problem
   Equations
   Micro-macro algorithms

3. Some Interesting Algorithms
   Parareal
   Density estimation
   Double projection

4. FENE micro-macro algorithm
   Configuration density function

5. Conclusion
   Conclusions
• Promising trends in computer architecture
Conclusions

• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code

Outlook
• Use parallel in time repeatedly
• Parareal between continuum and micro descriptions
• Parareal between Adaptive Mesh Refinement levels on continuum scale

Acknowledgements
• UNC Virtual Lung project (NIH R01-HL077546-01A2)
• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code
• Order-or-magnitude speed-up realized here
• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code
• Order-or-magnitude speed-up realized here
  • utilization of highly parallel GPUs
• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code
• Order-or-magnitude speed-up realized here
  • utilization of highly parallel GPUs
  • parallel in time formulation at microscale
Promising trends in computer architecture

Trends are beneficial if we adapt how we develop algorithms/code

Order-or-magnitude speed-up realized here
  - utilization of highly parallel GPUs
  - parallel in time formulation at microscale

Outlook
Conclusions

• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code
• Order-or-magnitude speed-up realized here
  • utilization of highly parallel GPUs
  • parallel in time formulation at microscale
• Outlook
  • use parallel in time repeatedly
Conclusions

- Promising trends in computer architecture
- Trends are beneficial if we adapt how we develop algorithms/code
- Order-or-magnitude speed-up realized here
  - utilization of highly parallel GPUs
  - parallel in time formulation at microscale
- Outlook
  - use parallel in time repeatedly
  - parareal between continuum and micro descriptions
Conclusions

- Promising trends in computer architecture
- Trends are beneficial if we adapt how we develop algorithms/code
- Order-or-magnitude speed-up realized here
  - utilization of highly parallel GPUs
  - parallel in time formulation at microscale

Outlook
- use parallel in time repeatedly
- parareal between continuum and micro descriptions
- parareal between Adaptive Mesh Refinement levels on continuum scale
• Promising trends in computer architecture
• Trends are beneficial if we adapt how we develop algorithms/code
• Order-or-magnitude speed-up realized here
  • utilization of highly parallel GPUs
  • parallel in time formulation at microscale
• Outlook
  • use parallel in time repeatedly
  • parareal between continuum and micro descriptions
  • parareal between Adaptive Mesh Refinement levels on continuum scale
• Acknowledgements
Conclusions

- Promising trends in computer architecture
- Trends are beneficial if we adapt how we develop algorithms/code
- Order-or-magnitude speed-up realized here
  - utilization of highly parallel GPUs
  - parallel in time formulation at microscale
- Outlook
  - use parallel in time repeatedly
  - parareal between continuum and micro descriptions
  - parareal between Adaptive Mesh Refinement levels on continuum scale
- Acknowledgements
  - UNC Virtual Lung project (NIH R01-HL077546-01A2)