SoftwareX 12 (2020) 100615

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

QBMMlib: A library of quadrature-based moment methods

Spencer H. Bryngelson [a,*], Tim Colonius [a], Rodney O. Fox [b,c]

[a] Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA
[b] Department of Chemical and Biological Engineering, Iowa State University, Ames, IA 50011, USA
[c] Center for Multiphase Flow Research and Education, Iowa State University, Ames, IA 50011, USA

ARTICLE INFO

Article history:
Received 11 August 2020
Received in revised form 5 October 2020
Accepted 21 October 2020

Keywords:
Population balance equation
Quadrature moment methods
Method of moments
Particulate flows
Dispersions

ABSTRACT

QBMMlib is an open source package of quadrature-based moment methods and their algorithms. Such methods are commonly used to solve fully-coupled disperse flow and combustion problems, though formulating and closing the corresponding governing equations can be complex. QBMMlib aims to make analyzing these techniques simple and more accessible. Its routines use symbolic manipulation to formulate the moment transport equations for a population balance equation and a prescribed dynamical system. The resulting moment transport equations are closed by first trading the moments for a set of quadrature points and weights via an inversion algorithm, of which several are available. Quadratures then compute the moments required for closure. Embedded code snippets show how to use QBMMlib, with the algorithm initialization and solution spanning just 13 total lines of code. Examples are shown and analyzed for a linear harmonic oscillator and a bubble dynamics problem.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00021
Legal Code License	GPL 3
Code versioning system used	git
Software code languages, tools, and services used	Wolfram Language / Mathematica or Python[1]
Compilation requirements, operating environments & dependencies	Wolfram Mathematica v8.0 or above or Python
If available Link to developer documentation/manual	https://github.com/sbryngelson/QBMMlib
Support email for questions	spencer@caltech.edu

[1] A Python version, PyQBMMlib is available at https://github.com/sbryngelson/PyQBMMlib.

1. Motivation and significance

QBMMlib is an open-source library and solves population balance equations (PBEs) using quadrature-based moment methods (QBMMs). PBEs model the evolution of a number density function (NDF) [1–5]. Such models are useful, for example, in fluid dynamics simulations, wherein the NDF evolution can represent growth, shrinkage, coalescence, breakup, and relative motion of a dispersed phased [6–14], Example engineering applications of this are combustion (e.g. soot dynamics in flames) [15–18] and aerosols (e.g. sprays) [19–21].

Both the method of classes [22,23] and the method of moments (MOM) [24,25]. can solve PBEs. QBMMlib employs the MOM because it can more naturally handle problems with multiple internal coordinates (e.g. velocities). Fig. 1 shows a typical QBMM-based solution procedure. The MOM represents the NDF via a set of statistical moments and the transport equations for them follow from the PBE. Inverting the moments to a set of weights and abscissas provides a basis for approximating the unclosed transport equations via quadrature (QMOM) [26].

Variations on QMOM are plentiful. One can change the inversion procedure: Wheeler's algorithm can solve single internal coordinate problems [27] and algorithms exist for enforcing distribution shape (extended-QMOM (EQMOM) [28], anisotropic-Gaussian [29,30]) and hyperbolicity (hyperbolic-QMOM (HyQMOM) [31]). The quadrature weights and abscissas can also evolve directly (direct-QMOM (DQMOM) [32–34]). One complication is that multiple internal coordinate problems do not admit a unique choice in moment set [35]. A modern and robust choice is

* Corresponding author.
E-mail address: spencer@caltech.edu (S.H. Bryngelson).

https://doi.org/10.1016/j.softx.2020.100615

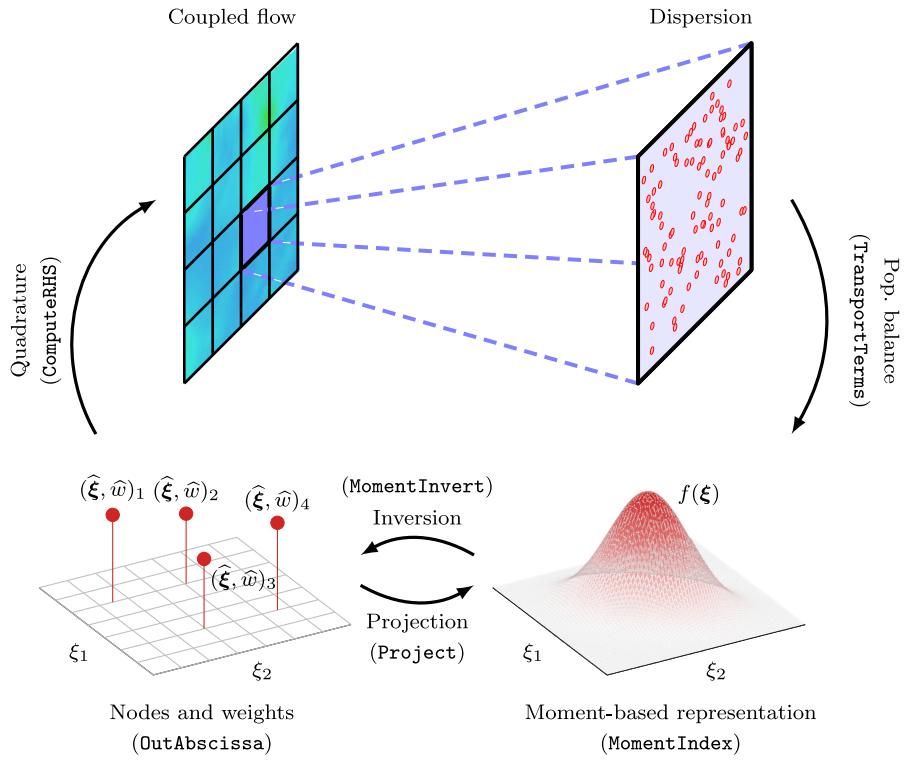

Fig. 1. Schematic illustration of the QMOM solution method for flowing dispersions. QBMMlib routines are in parentheses.

to condition one direction on the others (conditional-QMOM (CQMOM) [36] and -HyQMOM (CHyQMOM) [37]). For these reasons, QBMMlib uses Wheeler's algorithm (or its adaptive counterpart) or HyQMOM for one-dimensional moment inversion and CQMOM and CHyQMOM handle multi-dimensional problems.

There is one other actively developed open source QBMM solver: OpenQBMM [38,39]. It is a library for OpenFOAM [40] and implements CQMOM and (3-node) CHyQMOM. MFiX [32,41] and Fluidity [42] use DQMOM, though modern conditional methods (e.g. CQMOM and CHyQMOM) generally outperform it [7]. Note that these are fully-coupled flow solvers. QBMMlib instead decouples these problems and solves the moment transport equations directly for an input dynamical system. This makes it preferable for prototyping and testing on novel physical problems. In pursuit of this, QBMMlib places emphasis on expressive programming, simple interfaces, and symbolic computation where possible. As a result, it solves PBE-based problems with just a few lines of code. The end user can included spatial complexities as needed.

Section 2 describes QBMMlib's implementation of the PBE and QBMMs. Section 3 verifies its methods and demonstrates its capabilities for three example problems. Section 4 discusses the utility and novelty of QBMMlib, which concludes the paper.

2. Software description

QBMMlib is a collection of functions for solving PBEs via QBMMs. Table 1 describes the public-facing routines. These routines are also documented and accessible in Mathematica notebooks via

```
In[1]:= Get["QBMMlib"];
        ?QBMMlib`*
```

Note that, in their present form, `QBMMlib.wl` routines should not be called from other languages. Fig. 1 illustrates the place of these routines in the model and its solution procedure. `TransportTerms` computes the moment transport equations for the moment set of `MomentIndex`. `MomentInvert` inverts these moments to weights and abscissas that close the moment set via quadrature (`ComputeRHS`) and project it onto a realizable moment space (`Project`). The next sections describe the details of these routines.

2.1. Population balance equations (`TransportTerms`)

QBMMlib can solve one- and two-dimensional general populations balance equations and a supplemental direction if its NDF is stationary. We detail the two-dimensional case without loss of generality. For illustration, consider

$$\ddot{x} = g(x, \dot{x}), \tag{1}$$

where the dots indicate partial time derivatives, g is a function, and $\boldsymbol{\xi} = \{x, \dot{x}\}$ are the internal coordinates. The number density function f describes the state and statistics of this system in the $\boldsymbol{\xi}$-space. A population balance equation (PBE) governs f as

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(f\dot{x}) + \frac{\partial}{\partial \dot{x}}(f\ddot{x}) = 0, \tag{2}$$

where the zero right-hand-side indicates conservation of f, though sinks and sources can model aggregation and breakup [35]. Quadrature-based methods to solving (2) represent f by a set of raw moments $\vec{\boldsymbol{M}}$ as $f(\vec{\boldsymbol{M}})$. The moment indices $\vec{\boldsymbol{k}} = \{l, m\} = \{(0, 0), \ldots\}$ associated the carried moment set $\vec{\boldsymbol{M}} = \{M_{l,m}\}$ depend upon the moment inversion procedure and the number of quadrature points (details follow in Section 2.2).

The raw moments are

$$M_{p_1,\ldots,p_{N_\xi}} \equiv \int_\Omega f(\boldsymbol{\xi}) \prod_{j=1}^{N_\xi} \xi_j^{p_j} \mathrm{d}\xi_j \tag{3}$$

Table 1
Example public-facing routines. Parenthetical variables correspond to the code snippets and notation of Section 2.

`TransportTerms`	Input:	Governing equation (`eqn`, e.g. (1)) and its variables
	Output:	Coefficients (`coefs`) and exponents (`exps`) of moment transport equations
`MomentIndex`	Input:	Number of nodes (`n`, N_ξ), inversion method (`method`)
	Output:	Moment set indices (`momidx`, $\vec{\boldsymbol{k}}$)
	Options:	Number of permutations (default: 1)
`MomentInvert`	Input:	Moment set (`moments`, $\vec{\boldsymbol{M}}$) and its indices
	Output:	Optimal set of abscissas (`xi`, $\widehat{\boldsymbol{\xi}}$) and weights (`w`, $\widehat{\boldsymbol{w}}$)
	Options:	Method, Permutation (default: 12 ($\xi_1\vert\xi_2$), $N_\xi > 1$ only)
`ComputeRHS`	Input:	Abscissas, weights, moment set indices, transport coefficients
	Output:	Right-hand-side of moment transport equation (`rhs`, $\vec{\boldsymbol{F}}$)
	Options:	Third coordinate direction abscissas ($N_\xi = 2$ only)
`Project`	Input:	Abscissas, weights, moment set indices
	Output:	Projected moment set (`momentsP`, $\vec{\boldsymbol{M}}$)
`OutAbscissa`	Input:	Abscissas
	Output:	Threaded abscissas

where p_j (for $j = 1, \ldots, N_\xi$) are the moment indices, N_ξ is the number of internal coordinates ($N_\xi = 2$ in (1)), and Ω is the domain of f. These moments evolve as

$$\frac{\partial \vec{\boldsymbol{M}}}{\partial t} = \vec{\boldsymbol{F}}(\vec{\boldsymbol{M}}), \tag{4}$$

where, for (1),

$$F_{l,m} = lM_{l-1,m+1} + m\int_\Omega \ddot{x}\, x^l \dot{x}^{m-1} f \,\mathrm{d}\boldsymbol{\xi}. \tag{5}$$

This forcing follows from the PBE via integration-by-parts [43]. For the prescribed dynamics $\ddot{x}$ (as in (1)), the integral term of (5) is equivalent to a sum of moments. For example, if

$$\ddot{x} = x + \dot{x},$$

$$\text{then} \quad \int_\Omega \ddot{x} x^l \dot{x}^{m-1} f \,\mathrm{d}\boldsymbol{\xi} = M_{l+1,m-1} + M_{l,m},$$

$$\text{so} \quad F_{l,m} = lM_{l-1,m+1} + m(M_{l+1,m-1} + M_{l,m}). \tag{6}$$

The routine `TransportTerms` manipulates the PBE (as in (6)) to determine the coefficients and moment indices that constitute $\boldsymbol{F}$. The Mathematica code snippet below demonstrates this functionality for the dynamics of (1).

```
In[2]:= eqn = x[t]+x'[t] == x''[t];
        {coefs,exps} = TransportTerms[eqn,x[t],t]

Out[2]= {{c[2],c[2],c[1]},{{1+c[1],-1+c[2]},
        {c[1],c[2]},{-1+c[1],1+c[2]}}}
```

Here, the unassigned coefficients `c[1]` and `c[2]` correspond to the moment indices l and m of (5).

2.2. Moment inversion and quadrature weights (`MomentIndex`, `MomentInvert`)

`MomentInvert` inverts the set of raw moments $\vec{\boldsymbol{M}}$ into a set of quadrature weights $\widehat{\boldsymbol{w}}$ and abscissas (nodes) $\widehat{\boldsymbol{\xi}}$:

$$\vec{\boldsymbol{M}} \to \{\widehat{\boldsymbol{w}}, \widehat{\boldsymbol{\xi}}\} \tag{7}$$

Different algorithms can perform this procedure, each with its own relative merits, as discussed in Section 1. Common approaches for one-dimensional moment sets ($N_\xi = 1$) are QMOM [24,26] and hyperbolic QMOM (HyQMOM) [31]. For higher-dimensional moment sets ($N_\xi > 1$) conditioned moment methods are cheaper and more stable than performing QMOM on each coordinate direction individually [36]. Such conditioned methods perform 1D moment inversion in one coordinate direction, then condition the next directions on the previous ones [36]. Examples of these are conditional-QMOM (CQMOM) and conditional-HyQMOM (CHyQMOM) [31,37]. The order of conditioning is the permutation: 2D problems have two permutations, $\xi_1|\xi_2$ (coordinate direction ξ_1 conditioned on ξ_2) and $\xi_2|\xi_1$.

The indices that makeup a so-called optimal set $\vec{\boldsymbol{M}}$ depend on the moment inversion method and the number of quadrature nodes used in each internal coordinate direction $N_{\widehat{\xi}}$. Here, "optimal" constrains the number of moments (and their order) required to yield a full-rank and square coefficient matrix [44]. Optimal moment sets are more stable and smaller (and so cheaper) than non-optimal ones. For $N_\xi = 1$ single-coordinate problems the optimal moment indices are $\vec{\boldsymbol{M}} = \{M_0, M_1, \ldots, M_{2N_{\widehat{\xi}}-1}\}$ for QMOM and $\vec{\boldsymbol{M}} = \{M_0, M_1, \ldots, M_{2N_{\widehat{\xi}}-2}\}$ for HyQMOM. Table 2 shows the optimal moment sets QBMMlib uses for $N_\xi = 2$ two-dimensional problems [31,36]. `MomentIndex` computes these moment indices given the inversion algorithm (`method`) and number of quadrature points (`n`) in each internal coordinate direction ($N_{\widehat{\xi}}$). The code snippet below computes the moment set corresponding to $N_{\widehat{\xi}_1} = N_{\widehat{\xi}_2} = 2$ via CHyQMOM.

```
In[3]:= method = "CHYQMOM";
        n = {2,2};
        momidx = MomentIndex[n,method]

Out[3]= {{0,0},{1,0},{0,1},{2,0},{1,1},{0,2}}
```

`MomentInvert` then inverts the moment set $\vec{\boldsymbol{M}}$ to a set of weights and abscissas (as in (7)). In the following Mathematica code snippet, a two-dimensional Gaussian distribution (`BinormalDistribution`) initializes the moment set $\vec{\boldsymbol{M}}$ (`moments`), though in principle $\vec{\boldsymbol{M}}$ can be any realizable moment set. The `method` inversion algorithm then converts it to quadrature weights (`w`) and abscissas (`xi`).

```
In[4]:= mu1 = mu2 = 1; sig1 = sig2 = 0.3; rho = 0.5;
        f = BinormalDistribution[{mu1,mu2},{sig1,sig2},rho];
        GenMoment[i_] := Moment[f,i];
        moments = Map[GenMoment,momidx];
        {w,xi} = MomentInvert[moments,momidx,Method->method];
```

Fig. 2 shows the abscissas for different moment inversion algorithms. Their locations in the internal coordinate space $\boldsymbol{\xi}$ are different, though their weights are such that each quadrature reproduces the exact (up to) second-order moments. We verify that QBMMlib has this property to the carried precision. We discuss the QBMMlib quadrature routines used for this next.

Table 2
Example optimal $N_\xi = 2$ moment sets for permutations as labeled.

(a) CQMOM $[- \ \xi_1|\xi_2 \ - \ - \ \xi_2|\xi_1]$

(i) $N_\xi = 2$

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$M_{0,3}$
$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$
$M_{2,0}$	$M_{2,1}$		
$M_{3,0}$	$M_{3,1}$		

(ii) $N_\xi = 3$

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$M_{0,3}$	$M_{0,4}$	$M_{0,5}$
$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$
$M_{2,0}$	$M_{2,1}$	$M_{2,2}$	$M_{2,3}$	$M_{2,4}$	$M_{2,5}$
$M_{3,0}$	$M_{3,1}$	$M_{3,2}$			
$M_{4,0}$	$M_{4,1}$	$M_{4,2}$			
$M_{5,0}$	$M_{5,1}$	$M_{5,2}$			

(b) CHyQMOM

(i) $N_\xi = 2$

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$
$M_{1,0}$	$M_{1,1}$	
$M_{2,0}$		

(ii) $N_\xi = 3$

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$M_{0,3}$	$M_{0,4}$
$M_{1,0}$	$M_{1,1}$			
$M_{2,0}$				
$M_{3,0}$				
$M_{4,0}$				

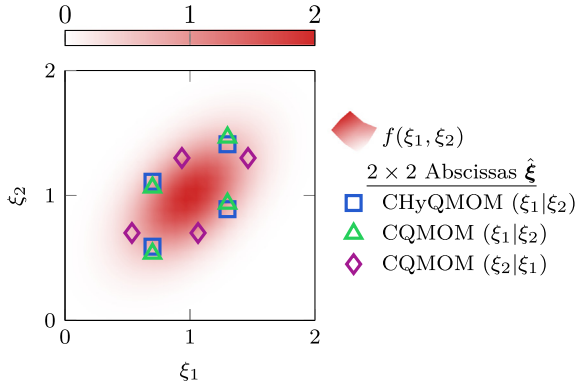

Fig. 2. Abscissas $\widehat{\boldsymbol{\xi}}$ corresponding to the number density function example f of Section 2.2. Different moment inversion algorithms and permutations correspond to $N_{\widehat{\xi}_1} = N_{\widehat{\xi}_2} = 2$ as labeled.

2.3. *Moment system closure via quadrature* (`ComputeRHS`)

Quadrature approximates the raw moments defined in (3) and required by (5) as

$$\prod_{j=1}^{N_\xi} \sum_{i=1}^{N_{\widehat{\xi}_j}} \widehat{w}_{j,i} \, \widehat{\xi}_{j,i}^{p_j} \to M_{p_1,\ldots,p_{N_\xi}} \tag{8}$$

where $\widehat{\xi}_{j,i}$ (for $i = 1, \ldots, N_{\widehat{\xi}_j}$) are the abscissa for internal coordinate direction ξ_j (for $j = 1, \ldots, N_\xi$). These quadrature approximations build $\vec{\boldsymbol{F}}$ of (5) (`F`) via the QBMM function `ComputeRHS`. `ComputeRHS` approximates (via quadrature) and sums the required moments (`exps`) and their coefficients (`coefs`) for each moment index (`momidx`). The code snippet below shows this implementation in QBMMlib.

```
In[5]:= F = ComputeRHS[w,xi,momidx,{coefs,exps}];
```

2.4. *Realizable time integration* (`Project`)

Stable and realizable time integration of (4) requires recasting the moment set $\vec{\boldsymbol{M}}$ from its weights and abscissas [13]. QBMMlib function `Project` performs this projection. A time integrator (e.g. Euler's method) then computes the next iteration of the moment set (`moments`). The code snippet below shows an example QBMMlib projection and Euler time step (with time step size `dt`).

```
In[6]:= momentsP = Project[w,xi,momidx];
        moments = momentsP + dt F;
```

QBMMlib also includes an adaptive strong-stability-preserving (SSP) third-order-accurate Runge–Kutta (RK) time integrator, RK23 [45]. The difference between the SSP–RK3 solution and an embedded second-order-accurate SSP–RK solution provides a first-order approximation of the time step error. The time step size adjustment is then proportional to this error. The illustrative examples of the next section use this adaptive time stepping procedure.

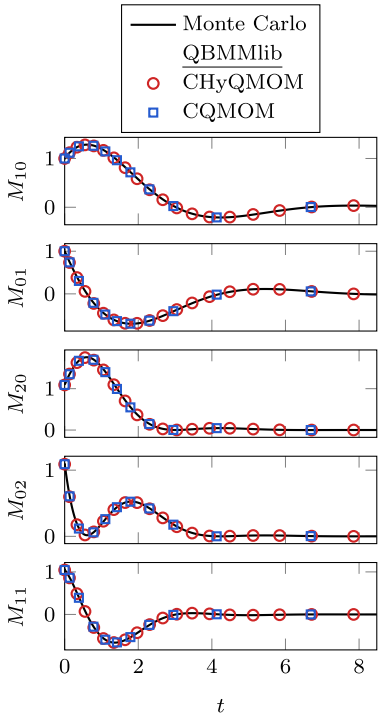

Fig. 3. Evolution of the first- and second-order moments in time t for the linear harmonic oscillator example problem. Monte Carlo simulation serves as the truth and the symbols show QBMMlib solutions using $N_{\widehat{\xi}_1} = N_{\widehat{\xi}_2} = 2$ CHyQMOM and CQMOM.

3. Illustrative examples

3.1. *Linear harmonic oscillator*

The example case of Section 2, including (6), is a linear harmonic oscillator. This moment system is linear and thus closed, so it can also verify the solution methods of QBMMlib via comparison to Monte Carlo simulations.

Fig. 3 shows the evolution of CQMOM and CHyQMOM moment sets and compares them to Monte Carlo surrogate truth solutions. The behavior of the first-order moments (M_{10} and M_{01}) match the positions and velocities expected of a linear oscillator. Further, the QBMMlib solutions match the moments of the Monte Carlo

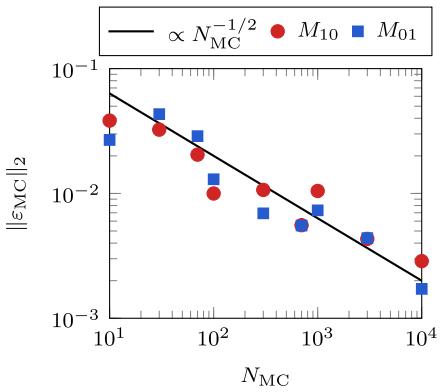

Fig. 4. Nominal differences $\|\varepsilon_{\mathrm{MC}}\|_2$ between the first-order moments of Monte Carlo simulation ensembles (of size N_{MC}) and the approximations of CHyQMOM (via QBMMlib). The expected convergence power law $\|\varepsilon_{\mathrm{MC}}\|_2 \propto 1/\sqrt{N_{\mathrm{MC}}}$ is also shown.

simulations to plotting accuracy. The L_2 norm $\|\cdot\|_2$ of the error quantifies this matching as

$$\varepsilon_{\mathrm{MC}}(t) \equiv \frac{M_{ij}^{(\mathrm{MC})}(t) - M_{ij}^{(\mathrm{QBMM})}(t)}{\max_t M_{ij}^{(\mathrm{QBMM})}(t)} \tag{9}$$

where superscripts (MC) and (QMOM) are correspond to Monte Carlo and QBMMlib simulations. Fig. 4 shows $\|\varepsilon_{\mathrm{MC}}\|_2$ for varying Monte Carlo ensemble size N_{MC} and QBMMlib method CHyQMOM, though the Monte Carlo moment errors dominate the QBMM ones and so CQMOM has the same results. Indeed, the error converges at the expected rate.

3.2. Bubble cavitation

The dynamics of a cavitating gas bubble dispersion serves as a two-internal-coordinate nonlinear example problem. The Rayleigh–Plesset equation models the bubble dynamics [46]:

$$R\ddot{R} + \frac{3}{2}\dot{R}^2 + \frac{4}{\mathrm{Re}}\frac{\dot{R}}{R} = \frac{1}{R^3} - C_p, \tag{10}$$

where R is the bubble radius, Re is the Reynolds number (dimensionless ratio of inertial to viscous effects), and C_p is the dimensionless pressure ratio between the suspending fluid and bubbles. Thus, R and $\dot{R}$ are the two internal coordinates ($\boldsymbol{\xi}$). For our purposes it suffices to ignore surface tension effects (following (10)) and use $C_p = 1/0.3$ to represent a large pressure ratio. This formulation is non-dimensionalized by the (monodisperse) equilibrium bubble radius and suspending fluid density and pressure. The initial NDF is a log-normal distribution in the R-coordinate (shape parameters $\mu_R = 1$, $\sigma_R = 0.2$) and a normal distribution in the $\dot{R}$ coordinate ($\mu_{\dot{R}} = 0$, $\sigma_{\dot{R}} = 0.1$). The NDF is initially uncorrelated.

Fig. 5 shows the moment dynamics for two bubble dispersions problems: (a) viscous $\mathrm{Re} = 10$ and (b) inviscid $\mathrm{Re} \to \infty$. Here, $\mathrm{Re} = 10$ is the Reynolds number that corresponds to 1 μm bubbles in water and $\mathrm{Re} \to \infty$ represents ignoring viscous effects. Invoking $\mathrm{Re} \to \infty$ is not appropriate for most cavitation problems of physical relevance, though it provides a useful reference. In both cases the mean bubble radius M_{10} oscillates and damps. This damping is more significant in (a) than (b) due to viscous effects, as expected. In the $\mathrm{Re} = 10$ case, this is sufficient for the QBMMlib-predicted moments to match the Monte Carlo results. However, for $\mathrm{Re} \to \infty$, the evolving moment set does not faithfully represent the bubble oscillations, particularly at long times.

Indeed, a mismatch between the Monte Carlo and QBMMlib results is clear for M_{02} and M_{11}. These differences are qualitatively similar for both the CQMOM and CHyQMOM algorithms. This is because closing the moment system requires extrapolating out of the represented moment space, which is of similar fidelity for both algorithms.

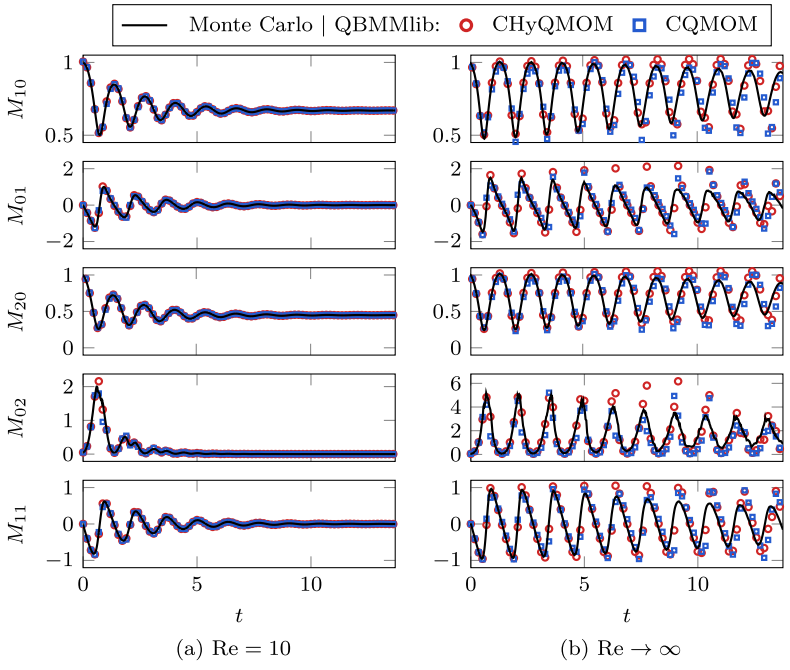

Fig. 5. Evolution of the first- and second-order moments for (a) viscous and (b) inviscid bubble dynamics as labeled. A $N_{\mathrm{MC}} = 5000$ Monte Carlo simulation approximates the exact moments. The symbols show QBMMlib solutions for $N_{\xi_1} = N_{\xi_2} = 2$ CHyQMOM and CQMOM, as labeled.

4. Impact and conclusions

This paper introduced QBMMlib, a library for solving PBEs using quadrature-based moment methods. It is a Wolfram Language package, which is useful for automating the procedure of using QBMMs for simulating phenomena like bubble and particle dynamics. This includes constructing a moment set for a given QBMM, determining the right-hand-side functions corresponding to a governing equation automatically, and inverting the moment set for quadrature points to close the system. These routines leverage Mathematica's symbolic algebra features and include modern QMOM and conditional-QMOM methods. Having these features available in a unified framework is helpful, particularly when it is unclear what QBMM will be appropriate (or stable) for the model dynamics. Our searches suggest that QBMMlib is the only library, open source or otherwise, that provides such capabilities. Given this, QBMMlib should help researchers prototyping QBMMs for their physical problems (or developing new QBMMs entirely). Indeed, the authors used QBMMlib to guide the implementation of CHyQMOM for phase-averaged bubble cavitation into MFC, the first flow solver with this capability [47,48].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors appreciate the insights of Alberto Passalacqua and Esteban Cisneros-Garibay when developing this library. The US Office of Naval Research supported this work under grant numbers N0014-17-1-2676 and N0014-18-1-2625.

References

[1] Ramkrishna D. Population balances. New York, USA: Academic Press; 2000.
[2] Chapman S, Cowling TG, Burnett D. The mathematical theory of non-uniform gases: An account of the kinetic theory of viscosity, thermal conduction and diffusion in gases. Cambridge University Press; 1990.
[3] Vanni M. Approximate population balance equations for aggregation breakage processes. J Colloid Interface Sci 2000;221:143–60.
[4] Smoluchowski M. Über Brownsche Molekularbewegung unter Einwirkung äußerer Kräfte und deren Zusammenhang mit der verallgemeinerten Diffusionsgleichung. Ann Phys 1916;353(24):1103–12.
[5] Solsvik J, Jakobsen HA. The foundation of the population balance equation: A review. J Disp Sci Tech 2015;36(4):510–20.
[6] Buffo A, Vanni M, Marchisio D. Multidimensional population balance model for the simulation of turbulent gas–liquid systems in stirred tank reactors. Chem Eng Sci 2012;70:31–44.
[7] Buffo A, Vanni M, Marchisio DL, Fox RO. Multivariate Quadrature-Based Moments Methods for turbulent polydisperse gas–liquid systems. Int J Multiph Flow 2013;50:41–57.
[8] Liao Y, Lucas D, Krepper E. Application of new closure models for bubble coalescence and breakup to steam–water vertical pipe flow. Nucl Eng Des 2014;279:126–36.
[9] Li D, Gao Z, Buffo A, Podgorska W, Marchisio DL. Droplet breakage and coalescence in liquid–liquid dispersions: Comparison of different kernels with EQMOM and QMOM. AIChE J 2017;63(6):2293–311.
[10] Gao Z, Li D, Buffo A, Podgórska W, Marchisio DL. Simulation of droplet breakage in turbulent liquid–liquid dispersions with CFD-PBM: Comparison of breakage kernels. Chem Eng Sci 2016;142:277–88.
[11] Fox RO. A quadrature-based third-order moment method for dilute gas–particle flows. J Comput Phys 2008;227(12):6313–50.
[12] Desjardins O, Fox RO, Villedieu P. A quadrature-based moment method for dilute fluid-particle flows. J Comput Phys 2008;227(4):2514–39.
[13] Nguyen TT, Laurent F, Fox RO, Massot M. Solution of population balance equations in applications with fine particles: Mathematical modeling and numerical schemes. J Comput Phys 2016;325:129–56.
[14] Kong B, Fox RO. A solution algorithm for fluid–particle flows across all flow regimes. J Comput Phys 2017;344:575–94.
[15] Kazakov A, Frenklach M. Dynamic modeling of soot particle coagulation and aggregation: Implementation with the method of moments and application to high-pressure laminar premixed flames. Combust Flame 1998;114(3–4):484–501.
[16] Balthasar M, Kraft M. A stochastic approach to calculate the particle size distribution function of soot particles in laminar premixed flames. Combust Flame 2003;133(3):289–98.
[17] Pedel J, Thornock JN, Smith ST, Smith PJ. Large eddy simulation of polydisperse particles in turbulent coaxial jets using the direct quadrature method of moments. Int J Multiph Flow 2014;63:23–38.
[18] Mueller ME, Blanquart G, Pitsch H. A joint volume-surface model of soot aggregation with the method of moments. Proc Combust Inst 2009;32(1):785–92, I.
[19] Sibra A, Dupays J, Murrone A, Laurent F, Massot M. Simulation of reactive polydisperse sprays strongly coupled to unsteady flows in solid rocket motors: Efficient strategy using Eulerian Multi-Fluid methods. J Comput Phys 2017;339:210–46.
[20] Laurent F, Massot M. Multi-fluid modelling of laminar polydisperse spray flames: Origin, assumptions and comparison of sectional and sampling methods. Combust Theor Model 2001;5(4):537–72.
[21] Hussain M, Kumar J, Tsotsas E. A new framework for population balance modeling of spray fluidized bed agglomeration. Particuology 2015;19:141–54.
[22] Ando K, Colonius T, Brennen CE. Numerical simulation of shock propagation in a polydisperse bubbly liquid. Int J Multiph Flow 2011;37(6):596–608.
[23] Bryngelson SH, Schmidmayer K, Colonius T. A quantitative comparison of phase-averaged models for bubbly, cavitating flows. Int J Multiph Flow 2019;115:137–43.
[24] Hulburt HM, Katz S. Some problems in particle technology: A statistical mechanical formulation. Chem Eng Sci 1964;19:555–74.
[25] Moyal JE. Stochastic processes and statistical physics. J R Stat Soc B 1949;11(2).
[26] McGraw R. Description of aerosol dynamics by the quadrature method of moments. Aerosol Sci Technol 1997;27:255–65.
[27] Wheeler JC. Modified moments and Gaussian quadratures. Rocky Mountain J Math 1974;4(2):287–96.
[28] Yuan C, Laurent F, Fox RO. An extended quadrature method of moments for population balance equations. J Aerosol Sci 2012;51:1–23.
[29] Patel RG, Desjardins O, Kong B, Capecelatro J, Fox RO. Verification of Eulerian–Eulerian and Eulerian–Lagrangian simulations for turbulent fluid–particle flows. AIChE J 2017;63(12):5396–412.
[30] Kong B, Fox RO, Feng H, Capecelatro J, Patel R, Desjardins O. Euler–Euler anisotropic Gaussian mesoscale simulation of homogeneous cluster-induced gas–particle turbulence. AIChE J 2017;63(7):2630–43.
[31] Fox RO, Laurent F, Vié A. Conditional hyperbolic quadrature method of moments for kinetic equations. J Comput Phys 2018;365:269–93.
[32] Marchisio DL, Fox RO. Solution of population balance equations using the direct quadrature method of moments. J Aerosol Sci 2005;36:43–73.
[33] Fox RO. Computational models for turbulent reacting flows. Cambridge University Press; 2003.
[34] Fox RO. Bivariate direct quadrature method of moments for coagulation and sintering of particle populations. J Aerosol Sci 2006;37:1562–80.
[35] Marchisio DL, Fox RO. Computational models for polydisperse particulate and multiphase systems. Cambridge University Press; 2013.
[36] Yuan C, Fox RO. Conditional quadrature method of moments for kinetic equations. J Comput Phys 2011;230(22):8216–46.
[37] Patel RG, Desjardins O, Fox RO. Three-dimensional conditional hyperbolic quadrature method of moments. J Comput Phys X 2019;1:100006.
[38] Passalacqua A, Laurent F, Madadi-kandjani E, Heylmun JC, Fox RO. An open-source quadrature-based population balance solver for OpenFOAM. Chem Eng Sci 2018;176:306–18.
[39] Passalacqua A, Heylmun J, Icardi M, Madadi E, Bachant P, Hu X. OpenQBMM 5.0.1 for OpenFOAM 7. Zenodo; 2019.
[40] Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput Phys 1998;12(6):620–31.
[41] Fan R, Marchisio DL, Fox RO. Application of the direct quadrature method of moments to polydisperse gas–solid fluidized beds. Powder Technol 2004;139(1):7–20.
[42] Davies DR, Wilson CR, Kramer SC. Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics. Geochem Geophys Geosyst 2011;12(6).
[43] Bryngelson SH, Charalampopoulos A, Sapsis TP, Colonius T. A Gaussian moment method and its augmentation via LSTM recurrent neural networks for the statistics of cavitating bubble populations. Int J Multiph Flow 2020;127:103262.
[44] Fox RO. Optimal moment sets for multivariate direct quadrature method of moments. Ind Eng Chem Res 2009;48(21):9686–96.

[45] Gottlieb S, Ketcheson D, Shu C-W. Strong stability preserving Runge–Kutta and multistep time discretizations. World Scientific; 2011.
[46] Brennen CE. Cavitation and bubble dynamics. Oxford University Press; 1995.
[47] Zhang DZ, Prosperetti A. Ensemble phase-averaged equations for bubbly flows. Phys Fluids 1994;6(2956).
[48] Bryngelson SH, Schmidmayer K, Coralic V, Meng JC, Maeda K, Colonius T. MFC: An open-source high-order multi-component, multi-phase, and multi-scale compressible flow solver. Comput Phys Comm 2020;107396.