QBMMlib: A library of quadrature-based moment methods

Spencer H. Bryngelsona
California Institute of Technology

Tim Colonius
California Institute of Technology

Rodney O. Fox
Iowa State University, rofox@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/cbe_pubs

Part of the Fluid Dynamics Commons, and the Theory and Algorithms Commons

The complete bibliographic information for this item can be found at https://lib.dr.iastate.edu/cbe_pubs/443. For information on how to cite this item, please visit http://lib.dr.iastate.edu/howtocite.html.
QBMMlib: A library of quadrature-based moment methods

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.

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

Disciplines
Fluid Dynamics | Theory and Algorithms

Comments

Creative Commons License
This work is licensed under a Creative Commons Attribution 4.0 License.
Original software publication

QBMMlib: A library of quadrature-based moment methods

Spencer H. Bryngelson, Tim Colonius, Rodney O. Fox

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.

© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

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 (E-QMOM) [28], anisotropic-Gaussian [29,30]) and hyperbolicity (hyperbolic-QMOM (H-QMOM) [31]). The quadrature weights and abscissas can also evolve directly (direct-QMOM (D-QMOM) [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

https://doi.org/10.1016/j.softx.2020.100615
2352-7110/© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
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 population balance equations and a supplemental direction if its NDF is stationary. We detail the two-dimensional case without loss of generality. For illustration, consider

\[ \dot{x} = g(x, \dot{x}), \]

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

\[ \frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(f \dot{x}) + \frac{\partial}{\partial \dot{x}}(f \ddot{x}) = 0, \]

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 \( M \) as \( f(M) \). The moment indices \( k = [l, m] = [(0, 0), \ldots] \) associated the carried moment set \( M = [M_{l,m}] \) depend upon the moment inversion procedure and the number of quadrature points (details follow in Section 2.2).

The raw moments are

\[ M_{p_{1}, \ldots, p_{\dot{N}}} = \int_{\Omega} f(\xi) \prod_{j=1}^{N} \xi_{j}^{p_{j}} d\xi \]
where \( p_j \) (for \( j = 1, \ldots, N_\Omega \)) are the moment indices, \( N_\Omega \) is the number of internal coordinates (\( N_\Omega = 2 \) in (1)), and \( \Omega \) is the domain of \( f \). These moments evolve as

\[
\frac{\partial \hat{M}}{\partial t} = \mathbf{F}(\hat{M}),
\]

where, for (1),

\[
F_{l,m} = IM_{l-1,m+1} + m \int_\Omega x^l x^{m-1} f \, d\xi.
\]

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

\[
\ddot{x} = x + \dot{x},
\]

then

\[
\int_\Omega \ddot{x} x^m \, d\xi = MI_{l-1,m-1} + M_{l,m},
\]

so

\[
F_{l,m} = IM_{l-1,m+1} + m(M_{l-1,m-1} + M_{l,m}).
\]

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

```mathematica
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)

\texttt{MomentInvert} inverts the set of raw moments \( \hat{M} \) into a set of quadrature weights \( \hat{w} \) and abscissas (nodes) \( \hat{\xi} \):

\[
\hat{M} \rightarrow [\hat{w}, \hat{\xi}]
\]

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 \( \xi_1 \) conditioned on \( \xi_2 \)) and \( \xi_2 \xi_1 \).

The indices that make up a so-called optimal set \( \hat{M} \) depend on the moment inversion method and the number of quadrature nodes used in each internal coordinate direction \( N_\xi \). 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 \( \hat{M} = \{M_0, M_1, \ldots, M_{2N_\xi-1}\} \) for QMOM and \( \hat{M} = \{M_0, M_1, \ldots, M_{2N_\xi-2}\} \) for HyQMOM. Table 2 shows the optimal moment sets QBMMlib uses for \( N_\xi = 2 \) two-dimensional problems \[31,36\].

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governing equation (eqn, e.g. (1))</td>
<td>Coefficients (coefs) and exponents (exps) of moment transport equations</td>
</tr>
<tr>
<td>Moment index</td>
<td>Number of nodes (n, ( N_\xi )), inversion method (method)</td>
</tr>
<tr>
<td>Moment set indices (momidx, ( \hat{k} ))</td>
<td>Number of permutations (default: 1)</td>
</tr>
</tbody>
</table>

Table 2

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

The Mathematica code snippet below computes the abscissas for different moment inversion methods (as in (7)). In the following Mathematica code snippet, a two-dimensional Gaussian distribution (BinormalDistribution) initializes the moment set \( \hat{M} \) (moments), though in principle \( \hat{M} \) can be any realizable moment set. The method inversion algorithm then converts it to quadrature weights \( \hat{w} \) and abscissas \( \hat{\xi} \).

```mathematica
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}}
```

Fig. 2 shows the abscissas for different moment inversion algorithms. Their locations in the internal coordinate space \( \hat{\xi} \) 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.
2.3. Moment system closure via quadrature (ComputeRHS)

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

\[
\hat{\xi}_j = \sum_{i=1}^{N_j} w_{ij} \xi_i \quad \text{for each} \quad (i) \text{ moment index (\text{midx})}
\]

where \( \hat{\xi}_j \) (for \( i = 1, \ldots, N_j \)) are the abscissa for internal coordinate direction \( \xi_j \) (for \( j = 1, \ldots, N \)). These quadrature approximations build \( F \) of (5) \((F)\) via the QBMM function ComputeRHS. ComputeRHS approximates \( F \) via quadrature and sums the required moments \( (\text{exps}) \) and their coefficients \( (\text{coefs}) \) for each moment index \( (\text{midx}) \). The code snippet below shows this implementation in QBMMlib.

\text{In[8]:= 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 \( \hat{M} \) 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 \( (\text{moments}) \). The code snippet below shows an example QBMMlib projection and Euler time step (with step time size \( dt \)).

\text{In[9]:= momentsP = Project[w,xi,momidx];}
\text{moments = momentsP + dt F;}

QBMMlib also includes an adaptive strong-stability-preserving (SSP) third-order-accurate Runge–Kutta (RK) time integrator,
Fig. 4. Nominal differences $\|\varepsilon_{MC}\|_2$ 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 $\|\varepsilon_{MC}\|_2 \propto 1/\sqrt{N_{MC}}$ is also shown.

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]:

$$R \dddot{R} + \frac{3}{2} \dot{R}^2 + \frac{4}{Re} \dot{R} = \frac{1}{R^3} - C_p,$$

(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 ($\xi$). 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 $Re = 10$ and (b) inviscid $Re \to \infty$. Here, $Re = 10$ is the Reynolds number that corresponds to 1 $\mu$m bubbles in water and $Re \to \infty$ represents ignoring viscous effects. Invoking $Re \to \infty$ 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 $Re = 10$ case, this is sufficient for the QBMMlib-predicted moments to match the Monte Carlo results. However, for $Re \to \infty$, 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 CHyQMOMM algorithms. This is because closing the moment system requires extrapolating out of the represented moment space, which is of similar fidelity for both algorithms.
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 ChHyQMOM 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


