Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 7791–7812
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Acceleration techniques for reduced-order models based on proper orthogonal decomposition

Paul G.A. Cizmas [a,*], Brian R. Richardson [a], Thomas A. Brenner [a], Thomas J. O'Brien [b], Ronald W. Breault [b]

[a] Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843-3141, USA
[b] National Energy Technology Laboratory, Department of Energy, Morgantown, WV 26507-0880, USA

Received 3 July 2007; received in revised form 24 April 2008; accepted 30 April 2008
Available online 16 May 2008

Abstract

This paper presents several acceleration techniques for reduced-order models based on the proper orthogonal decomposition (POD) method. The techniques proposed herein are: (i) an algorithm for splitting the database of snapshots generated by the full-order model; (ii) a method for solving quasi-symmetrical matrices; (iii) a strategy for reducing the frequency of the projection. The acceleration techniques were applied to a POD-based reduced-order model of the two-phase flows in fluidized beds. This reduced-order model was developed using numerical results from a full-order computational fluid dynamics model of a two-dimensional fluidized bed. Using these acceleration techniques the computational time of the POD model was two orders of magnitude shorter than the full-order model.

Keywords: Proper orthogonal decomposition; Multiphase flow; Computational methods in fluid dynamics; Nonlinear dynamics

1. Introduction

The numerical simulation of transient transport phenomena requires a large amount of computational time, in spite of the developments in computer hardware. It is often difficult to identify the dominant spatial features from these numerical simulations. These issues are exacerbated when modeling multiphase flows and chemical reactions, such as in multiphase flow reactors. It is also difficult to assess how the process variables influence each other. To address these problems, reduced-order models of fluidized bed have been recently developed [1,2].

Reduced-order models (ROMs) have been commonly used in structural dynamics for years. Currently these techniques are also applied to fluid dynamics studies. The idea is to determine the dominant spatial modes of the flow field and use these modes, as opposed to many local grid points, to represent the flow [3]. The

* Corresponding author. Tel.: +1 979 845 5952.
E-mail address: cizmas@tamu.edu (P.G.A. Cizmas).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.04.036

Nomenclature

Roman

a	coefficients of discretized equations
A	cell face area
b	coefficients of discretized equations
D	spatial domain
D_p	particle diameter
F_{gs}	drag force between fluid and solids
$\vec{g}$	gravitational acceleration vector
h_{s0}	initial height of packed bed
$i_{\max}$	number of cells in x-direction
$j_{\max}$	number of cells in y-direction
M	number of snapshots (that is, dimension of temporal domain)
m	number of POD modes
N	dimension of spatial domain
p	pressure
R	autocorrelation function
t	time
$\overline{\overline{S}}$	stress tensor
$\vec{v}$	velocity vector
ΔV	cell volume
(x, y)	Cartesian coordinates

Greek

α	time-dependent orthogonal coefficients
ϵ	volume fraction
ε	error
ρ	density
λ	eigenvalue
ξ	convection weighting factor
φ	time-independent orthonormal basis function
μ	viscosity

Superscripts

$*$	complex conjugate
$\bigstar$	tentative value, i.e., value before correction
p	component corresponding to pressure p
u	component corresponding to velocity u
v	component corresponding to velocity v
ϵ	component corresponding to volume fraction

Subscripts

0	zeroth mode or initial value
E	center of east neighbor cell in mass balance equations or east face of control volume in momentum balance equations
e	east face of control volume in mass balance equation or east neighbor cell of control volume in momentum balance equations
g	gas phase
i	ith mode
k	kth mode

ℓ	either gas or solids phase
m	either gas or solids phase
nb	neighbor cells
N	center of north neighbor cell in mass balance equations or north face of control volume in momentum balance equations
n	north face of control volume in mass balance equation or north neighbor cell of control volume in momentum balance equations
P	center of control volume in mass balance equation
p	center of control volume in momentum balance equations
S	center of south neighbor cell in mass balance equations or south face of control volume in momentum balance equations
s	solids phase or south face of control volume in mass balance equations or south neighbor cell of control volume in momentum balance equations
W	center of west neighbor cell in mass balance equations or west face of control volume in momentum balance equations
w	west face of control volume in mass balance equation or west neighbor cell of control volume in momentum balance equations

reduction is from the large number of local grid points (on the order of tens of thousands or more) to a small number of spatial modes (typically less than 100).

In the first attempts to use ROMs for fluid flows [4–8] the models were developed for small perturbations about a nonlinear steady flow field. As a result, these models were limited in their applicability. Reduced-order models were also developed for flows with large perturbations using the method of proper orthogonal decomposition (POD) [9–11,1,2,12,13]. Recent reviews of ROMs based on POD are presented in [14,15].

The objective of this paper is to present a set of acceleration techniques for POD-based reduced-order models. These techniques include: (i) an algorithm for splitting the snapshot database; (ii) a method for solving quasi-symmetrical matrices; (iii) a strategy for reducing the frequency of the POD projection. The next section presents the full-order model used for the simulation of the transport phenomena in a fluidized bed. This is followed by a description of the POD-based reduced-order model developed for the full-order model. A detailed comparison of the full-order and reduced-order model algorithms is described next. The acceleration techniques proposed for the solution of the reduced-order model are subsequently presented.

2. Full-order model

The term "full-order model" denotes the numerical model used to solve the transport equations. For the isothermal flow considered herein, these governing equations consist of the gas-phase and solids-phase mass and momentum balance. This section describes the governing equations of the full-order model and the numerical solver used to solve these equations.

2.1. Governing equations

The gas–solids interactions in a fluidized bed were modeled herein using a two-fluid hydrodynamic model [16]. The model described the isothermal flow of dense or dilute fluid–solids mixtures, based on the fundamental laws of mass and momentum conservation. The governing equations of this model were the following system of partial differential equations:

Gas-phase and solids-phase mass balance

$$\frac{\partial}{\partial t}(\epsilon_m \rho_m) + \nabla \cdot (\epsilon_m \rho_m \vec{v}_m) = 0. \tag{1}$$

Gas-phase and solids-phase momentum balance

$$\frac{\partial}{\partial t}(\epsilon_m \rho_m \vec{v}_m) + \nabla \cdot (\epsilon_m \rho_m \vec{v}_m \vec{v}_m) = -\epsilon_m \nabla p_g + \nabla \cdot \bar{\bar{S}}_m + F_{gs}(\vec{v}_s - \vec{v}_g) + \epsilon_m \rho_m \vec{g}. \quad (2)$$

In the momentum equation, the first term on the right side represents the normal surface forces; the second term represents the shear surface forces. The last two terms represent the drag force between the fluid and solids phases, and the gravitational body forces.

2.2. Numerical solver

The full-order model described herein consisted of the transport equations (1) and (2). The numerical algorithm developed at the Department of Energy's National Energy Technology Laboratory and implemented in the Multiphase Flow with Interphase eXchanges (MFIX) code [16] was used to solve the transport equations. The solution of these equations provided the database for the proper orthogonal decomposition [1].

MFIX uses a staggered grid arrangement to prevent unphysical oscillations of the solution [17, p. 116]. Scalars are stored at the cell centers; components of velocity vectors are stored at the cell faces. The equations for scalar variables are solved on the main grid. The equations for velocity components are solved on two staggered grids.

2.2.1. Mass balance

A control volume for the mass balance equations is shown in Fig. 1, where P is the center of the control volume. E, W, N, and S represent the centers of the east, west, north, and south neighbor cells of the control volume; e, w, n, and s represent the east, west, north, and south faces of the control volume. The volume fractions ϵ_m and densities ρ_m are stored at the cell centers P, E, W, N, and S. In order to discretize the convection terms, volume fraction and density values at the cell faces e, w, n, and s must be evaluated.

MFIX uses a convection weighting factor ξ to calculate the volume fraction and density at each face [18]. For example, $(\epsilon_m \rho_m)$ at the east face is calculated as [17, p. 44]

$$(\epsilon_m \rho_m)_e = (\xi_m)_e (\epsilon_m \rho_m)_E + (\bar{\xi}_m)_e (\epsilon_m \rho_m)_P, \quad (3)$$

where $(\xi_m)_e$ is the convection weighting factor for $(\epsilon_m \rho_m)$ at the east face and $(\bar{\xi}_m)_e = 1 - (\xi_m)_e$. Using this, the mass balance equations are discretized as [17, p. 99]

$$(a_m)_P (\epsilon_m \rho_m)_P = \sum_{nb} (a_m)_{nb} (\epsilon_m \rho_m)_{nb} + (b_m)_P, \quad (4)$$

where the subscript nb represents E, W, N, and S neighbors. The coefficients $(a_m)_{nb}$, $(a_m)_P$ and $(b_m)_P$ are defined in [2, Appendix A].

2.2.2. Momentum balance

The control volumes used to discretize the x- and y-momentum balance equations are shown in Fig. 2, where p denotes the center of the control volume; e, w, n, and s represent the east, west, north, and south neighbor cells of the control volume; E, W, N, and S denote the east, west, north, and south faces of the con-

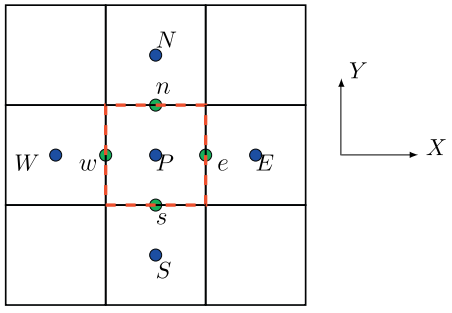
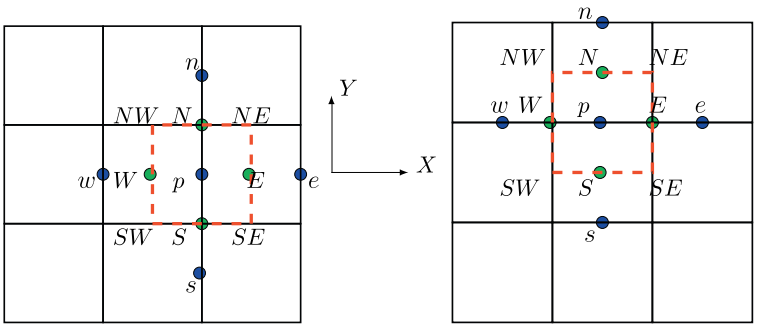


Fig. 1. Control volume for mass balance.



(a) x-momentum balance (b) y-momentum balance

Fig. 2. Control volume for momentum balance.

trol volume; NE, NW, SE, and SW denote the four corners of the control volume. The discretized form of the gas and solids x-momentum equations is [18, p. 26]

$$(a_m^u)_p(u_m)_p = \sum_{nb}(a_m^u)_{nb}(u_m)_{nb} + (b_m^u)_p - A_p(\epsilon_m)_p((p_g)_E - (p_g)_W) + (F_{gs}(u_\ell - u_m)_p)\Delta V, \tag{5}$$

where m indicates the phase. ℓ denotes the phase other than m, that is, if m corresponds to the solids phase then ℓ corresponds to the gas phase, and *vice versa*. The y-momentum equations are discretized similarly

$$(a_m^v)_p(v_m)_p = \sum_{nb}(a_m^v)_{nb}(v_m)_{nb} + (b_m^v)_p - A_p(\epsilon_m)_p((p_g)_N - (p_g)_S) + (F_{gs}(v_\ell - v_m)_p)\Delta V. \tag{6}$$

The coefficients $(a_m^u)_{nb}$, $(a_m^u)_p$, $(b_m^u)_p$, $(a_m^v)_{nb}$, $(a_m^v)_p$ and $(b_m^v)_p$ are defined in [2, Appendix B].

2.2.3. Gas pressure correction

The full-order model solver uses the gas pressure correction equation instead of the gas mass balance equation. The former is derived from the discretized gas mass balance equation and the discretized momentum balance equations, and can be written as [18, p. 41]

$$a_P^p(p_g')_P = \sum_{nb} a_{nb}^p(p_g')_{nb} + b_P^p. \tag{7}$$

The coefficients a_{nb}^p, a_P^p and b_P^p are defined in [2, Appendix C].

Eq. (7) is solved to determine the gas pressure correction, p_g', using the same control volume as for the mass balance equations. The velocity correction along the x-direction is given by [18]

$$(u_m)_p = (u_m^\star)_p - d_{mp}((p_g')_E - (p_g')_W), \tag{8}$$

where the expressions of d_{gp} and d_{sp} are given in [2]. The superscript ★ indicates tentative velocities (i.e., velocities before correction). In (8), p is the center of the control volume shown in Fig. 2a.

Similarly, the velocity correction along the y-direction is given by

$$(v_m)_p = (v_m^\star)_p - d_{mp}((p_g')_N - (p_g')_S), \tag{9}$$

where p now is the center of the control volume shown in Fig. 2b.

2.2.4. Solids volume fraction correction

To solve dense packing of solids, a solids volume fraction correction equation is used by including the effect of solids pressure in the discretized solids mass balance equation [18, p. 48]. This solids volume fraction correction equation

$$a_P^{\epsilon_s}\epsilon_{sP}' = \sum_{nb} a_{nb}^{\epsilon_s}\epsilon_{snb}' + b_P^{\epsilon_s} \tag{10}$$

is solved instead of the solids mass balance equation. The coefficients $a_{nb}^{\epsilon_s}$, $a_P^{\epsilon_s}$ and $b_P^{\epsilon_s}$ are defined in [2, Appendix D]. The solids volume fraction correction, ϵ_s', was used to calculate the solids volume fraction ϵ_s. The gas void fraction, ϵ_g, was then calculated as $\epsilon_g = 1 - \epsilon_s$.

3. Reduced-order model based on proper orthogonal decomposition

Proper orthogonal decomposition is a procedure for extracting the optimal basis set from an ensemble of observations. The POD extracts key spatial features from physical systems with spatial and temporal characteristics [19]. Let us consider a sequence of numerical or experimental observations represented by scalar functions $u(x,t_i), i = 1,\ldots,M$. These observations are assumed to form a linear, finite-dimensional Hilbert space L^2 on a spatial domain D. The observations $u(x,t_i)$ are parameterized by t_i, which represents time. From the ensemble of observations, POD extracts time-independent orthonormal basis functions $\{\varphi_k(x)\}$ and time-dependent orthogonal coefficients $\{\alpha_k(t_i)\}$, such that the reconstruction

$$u(x,t_i) = \sum_{k=1}^{M} \alpha_k(t_i)\varphi_k(x), \quad i = 1,\ldots,M, \tag{11}$$

which is an approximation, is optimal in the sense that the average least-square truncation error

$$\varepsilon_m = \left\langle \left\| u(x,t_i) - \sum_{k=1}^{m} \alpha_k(t_i)\varphi_k(x) \right\|^2 \right\rangle \tag{12}$$

is a minimum for any given number $m \leqslant M$ of basis functions over all possible sets of basis functions. Herein $\|\cdot\|$ denotes the L^2-norm given by $\|f\| = (f,f)^{\frac{1}{2}}$, where $(,)$ denotes the Euclidean inner product. $\langle\cdot\rangle$ denotes an ensemble average over the number of observations $\langle f\rangle = \sum_{i=1}^{M} f(x,t_i)/M$. The optimum condition specified by (12) is equivalent to finding functions φ that maximize the normalized averaged projection of u onto φ, that is, $\max_{\varphi\in L^2(D)} \langle |(u,\varphi)|^2\rangle / \|\varphi\|^2$, where $|\cdot|$ denotes the modulus. This maximization problem reduces to [19, p. 89]

$$\int_D \langle u(x)u^*(y)\rangle \varphi(y)\,\mathrm{d}y = \lambda\varphi(x). \tag{13}$$

Therefore, the optimal basis functions $\{\varphi_k\}$, called the POD basis functions, are the eigenfunctions of the integral equation (13), whose kernel function is the auto-correlation function $\langle u(x)u^*(y)\rangle \equiv R(x,y)$. For a finite-dimensional case, the auto-correlation function $R(x,y)$ is replaced by the tensor product matrix $\overline{\overline{R}}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{M} \mathbf{u}(\mathbf{x},t_i)\mathbf{u}^{\mathrm{T}}(\mathbf{y},t_i)/M$.

The eigenfunctions $\boldsymbol{\varphi}_k(\mathbf{x})$ are also vector-valued functions and have the same dimension N as the observations $\mathbf{u}$. It can be shown that the eigenvectors of $\overline{\overline{R}}$ are the eigenfunctions $\boldsymbol{\varphi}_k(\mathbf{x})$ [19, p. 90], again, called the POD basis functions.

For two-dimensional isothermal gas–solids flow, the full-order computational fluid dynamics model solves the discretized momentum equations of both the gas-phase and the solids-phase (5) and (6), the gas pressure correction equation (7), and the solids volume fraction correction equation (10). The dependent field variables of the full-order model were the gas pressure p_g, the solids volume fraction ϵ_s, and the velocity components of the gas-phase and the solid-phase, u_g, v_g, u_s, and v_s. Using the full-order model, a database of snapshots of these variables was generated using the MFIX code. The POD basis functions $\varphi_i^{p_g}$, $\varphi_i^{\epsilon_s}$, $\varphi_i^{u_g}$, $\varphi_i^{v_g}$, $\varphi_i^{u_s}$, and $\varphi_i^{v_s}$ were extracted from this database. These POD basis functions were obtained from the auto-correlation functions $R^{\aleph}(x,y) \equiv \langle \aleph(x)\aleph^*(y)\rangle$, where $\aleph$ denotes any of the dependent variables. Consequently, these POD basis functions were sub-optimal because they implicitly assumed that the cross-correlations of the components of vector $\{p_g,\epsilon_s,u_g,v_g,u_s,v_s\}$ were zero.

In developing the reduced-order model, one can project onto the basis functions either the partial differential equations of the full-order model or the discretized form of these equations. The latter option has been used herein in order to take advantage of the implementation developed for the full-order model. The discretized x- and y-momentum equations, the discretized gas pressure correction equation and the discretized solids volume fraction correction equation are projected onto the basis functions φ^{u_m}, φ^{v_m}, φ^{p_g}, and φ^{ϵ_s}, respectively. Six systems of linear algebraic equations are obtained:

$$\widetilde{\mathcal{A}}^{u_m}\alpha^{u_m} = \widetilde{\mathcal{B}}^{u_m}, \tag{14}$$

$$\widetilde{\mathcal{A}}^{v_m}\alpha^{v_m} = \widetilde{\mathcal{B}}^{v_m}, \tag{15}$$

$$\widetilde{\mathcal{A}}^{p_g}\alpha^{p'_g} = \widetilde{\mathcal{B}}^{p_g}, \tag{16}$$

$$\widetilde{\mathcal{A}}^{\epsilon_s}\alpha^{\epsilon'_s} = \widetilde{\mathcal{B}}^{\epsilon_s}, \tag{17}$$

where m denotes the phase (g or s). The first five systems of equations, (14)–(16), are identical to those derived in [2] and for this reason will not be repeated in here. The system of equations (17) represents to solids volume fraction correction equations. The derivation of the expressions of $\widetilde{\mathcal{A}}^{\epsilon_s}$ and $\widetilde{\mathcal{B}}^{\epsilon_s}$ is shown in the following.

The reconstruction (11) can be written for solids volume fraction as:

$$\epsilon_s(x,t) = \varphi_0^{\epsilon_s}(x) + \sum_{k=1}^{m^{\epsilon_s}} \alpha_k^{\epsilon'_s}(t)\varphi_k^{\epsilon_s}(x), \tag{18}$$

where the time independent term $\varphi_0^{\epsilon_s}(x)$, which represents the zeroth-order term, is kept on the right-hand side and not deducted from $\epsilon_s(x,t)$. Similarly to the pressure, the solids volume fraction can be written as the sum of a tentative value, $\epsilon_s^\star$ and a correction, ϵ'_s

$$\epsilon_s(x,t) = \epsilon_s^\star(x) + \epsilon'_s(x,t), \tag{19}$$

where $\varphi_0^{\epsilon_s}$ and $\epsilon_s^\star$ are assumed to be equal.

Combining (18) and (19) yields

$$\epsilon'_s(x,t) = \sum_{k=1}^{m^{\epsilon_s}} \alpha_k^{\epsilon'_s}(t)\varphi_k^{\epsilon_s}(x). \tag{20}$$

Dropping for convenience the ϵ'_s superscript, replacing in (10) the subscript P by i, and substituting (20) into (10) yields

$$\sum_{k=1}^{m} \alpha_k \left(a_i\varphi_k(x_i) - \sum_{i_{nb}=1}^{NB} a_{i_{nb}}\varphi_k(x_{i_{nb}}) \right) = b_i, \quad i = 1,\ldots,N, \tag{21}$$

where i_{nb} represents the neighbor of cell i. NB is the total number of neighbors of a cell, which herein is constant and equal to 4, and m is the number of modes kept in the proper orthogonal decomposition.

Eq. (21) can be written as

$$\sum_{k=1}^{m} \alpha_k \left([A]\{\varphi_k\} - \sum_{nb=1}^{NB} [A_{nb}]\{\varphi_{k_{nb}}\} \right) = \{b\}, \tag{22}$$

where $[A]$ and $[A_{nb}]$ are diagonal matrices with $A_{ii} = a_i$ and $A_{nbii} = a_{i_{nb}}$, $i = 1,\ldots,N$, $\{\varphi_k\} = \{\varphi_k(x_1), \varphi_k(x_2), \ldots, \varphi_k(x_N)\}^{\mathrm{T}}$ and $\{\varphi_{k_{nb}}\} = \{\varphi_k(x_{1_{nb}}), \varphi_k(x_{2_{nb}}), \ldots, \varphi_k(x_{N_{nb}})\}^{\mathrm{T}}$.

Eq. (22) is then projected on the basis functions by left-multiplying it with the transposed eigenvectors $\{\varphi_\ell\}^{\mathrm{T}}$

$$\{\varphi_\ell\}^{\mathrm{T}} \sum_{k=1}^{m} \alpha_k \left([A]\{\varphi_k\} - \sum_{nb=1}^{NB} [A_{nb}]\{\varphi_{k_{nb}}\} \right) = \{\varphi_\ell\}^{\mathrm{T}}\{b\}, \quad \ell = 1,\ldots,m. \tag{23}$$

The system of m equations (23) has m unknowns α_i and, after adding back the superscripts ϵ_s and ϵ'_s, can be written as

$$[\widetilde{\mathcal{A}}^{\epsilon_s}]\{\alpha^{\epsilon'_s}\} = \{\widetilde{\mathcal{B}}^{\epsilon_s}\}, \tag{17}$$

where

$$\widetilde{\mathcal{A}}_{\ell k}^{\epsilon_s} = \{\varphi_\ell\}^{\mathrm{T}}[A]\{\varphi_k\} - \sum_{nb=1}^{NB} \{\varphi_\ell\}^{\mathrm{T}}[A_{nb}]\{\varphi_{k_{nb}}\}, \quad \ell,k = 1,\ldots,m \tag{24}$$

and

$$\widetilde{\mathcal{B}}_\ell^{\epsilon_s} = \{\varphi_\ell\}^{\mathrm{T}}\{b\}, \quad \ell = 1,\ldots,m.$$

4. Structure of numerical algorithms

The purpose of this section is to compare the numerical implementation of the full-order model (FOM), against numerical implementation of the reduced-order model (ROM). This section presents the time profiles of the FOM and ROM codes for a typical flow case. The computational time per subiteration, the number of subiterations, and number of operations are also provided.

The solution algorithms used in the ROM and the FOM are similar in organization. For a two-dimensional isothermal case, the FOM solves for six dependent field variables: solids volume fraction, gas pressure, and gas and solids phase velocities. The ROM solves for the time coefficients $\alpha(t)$ of each dependent field variable and then reconstructs the dependent variables using the basis functions. Both the FOM and the ROM use fully implicit, time marching algorithms. At each time step subiterations are performed until a residual criterion is satisfied. The time step size is adjusted based on the convergence rate during the calculation.

A comparison of the FOM and ROM codes was done for a minimum fluidization case [20], where the solution was simulated for 0.8 s, from 0.2 to 1.0 s. From 0 to 0.2 s a background gas was injected at the bottom of the fluidized bed with a velocity of 1 cm/s. At t = 0.2 s, a central jet with the width of 1.016 cm and velocity of 12.6 cm/s was turned on. The geometry and boundary conditions are shown in Fig. 3 and the parameters are given in Table 1. Three hundred and twenty snapshots were generated while using the full-order model. These snapshots were as equally spaced in time as possible, given the variable time step of the full-order model. The

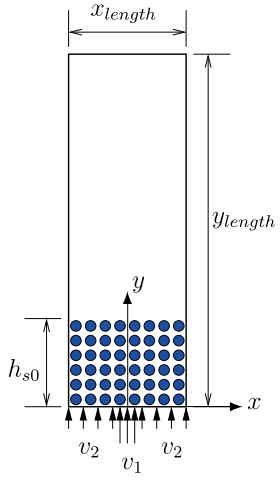


Fig. 3. Geometry and boundary conditions.

Table 1
Parameters of fluidized bed

Parameter	Description	Units	
x_{length}	Length of the domain in x-direction	cm	25.4
y_{length}	Length of the domain in y-direction	cm	76.5
$i_{\max}$	Number of cells in x-direction	–	50
$j_{\max}$	Number of cells in y-direction	–	76
v_1, v_2	Gas inflow velocities	cm/s	12.6, 1
p_{g_s}	Static pressure at outlet	g/cm/s^2	1.01e^6
T_{g_0}	Gas temperature	K	297
μ_{g_0}	Gas viscosity	g/cm/s	1.8e^{-4}
t_{start}	Start time	s	0.2
t_{stop}	Stop time	s	1.0
ρ_s	Particle density	g/cm^3	1.0
D_p	Particle diameter	cm	0.05
h_{s0}	Initial height of packed bed	cm	14.7
ϵ_g^*	Initial void fraction of packed bed	–	0.4

Table 2
Number of modes used and their symbols

Field variable	Symbol	Number of modes	
		Set 1	Set 2
Gas pressure	N_{p_g}	2	2
Solids volume fraction	$N_{\epsilon s}$	7	3
u gas velocity	N_{u_g}	2	1
v gas velocity	N_{v_g}	5	5
u solids velocity	N_{u_s}	8	4
v solids velocity	N_{v_s}	6	3

number of modes used in the ROM is given by Set 1 in Table 2. For this case, one subiteration took 0.386 s in the ROM and 0.213 s in the FOM. The FOM required a total of 43,600 subiterations while the ROM only required a total of 2300 subiterations. Consequently, although a subiteration in the reduced-order model took 1.8 times more than a subiteration in the full-order model, the reduced-order model was 10.6 times faster than the full-order model for simulating 0.8 s of flow in a fluidized bed.

The speed-up was due to the fact that the ROM required fewer subiterations per time step than the FOM. This was the result of the fact that the time step limitations in the reduced-order model that solved ODEs were less restrictive that those of the full-order model that solved PDEs. Consequently, the ROM could use larger time steps than the FOM, as it will be shown in Section 5.4. The speed-up obtained by using the reduced-order model increases as the duration of the simulation increases.

A comparison of the full-order and reduced-order models is presented to explain the differences in computational time per subiteration. To facilitate this comparison, the subroutines of the FOM and the ROM were divided into similar groups. Although the role of each group is the same in the FOM and the ROM, the details of the subroutines in the groups differ. The definitions of the groups are presented in Table 3. Both models spent most of computational time in groups 5, 6, and 7, as shown in Table 4. These three groups form the subiteration loop.

Group 7 was chosen to be explored in detail for two reasons. First, the calculations in this group are a significant percentage of the total computation time. The calculations in group 7 take approximately 24% of the total computation time of the ROM and 12% of the FOM. Second, numerical tests showed that in order to achieve a good solution, the volume fraction often required more modes than most of the other field variables.

To determine the most computationally expensive calculations, a time profile of group 7 was generated for both the FOM and the ROM. These time profiles are shown in Fig. 4. Fig. 4a indicates that the FOM spent most of the computing time solving the linear set of equations (10). In contrast, the ROM spent most of the computing time on the projection of the basis functions. In the ROM, the computational time spent for the solution of the linear systems of equations was less than 1% of the total computational time. If less than three modes are used, the reconstruction of the field variables, which is part of the correction group, uses the majority of the calculation time. The conclusions drawn based on the solids volume fraction hold for all the dependent field variables.

Table 3
Description of common groups of the FOM and the ROM codes

Group	Description
1	Reads initial data, sets boundary and initial conditions
2	Calculates initial values of dependent variables
3	Calculates initial values inside the time loop
4	Calculates initial values inside the subiteration loop
5	Updates velocity values
6	Updates pressure and corrects velocities
7	Updates solids volume fraction and corrects velocities
8	Checks convergence and performs final steps in iteration loop
9	Outputs results

Table 4
Time profile of the FOM and the ROM codes

Group	FOM		ROM	
	(s)	(%)	(s)	(%)
1	0.396	0.004	0.874	0.097
2	0.586	0.006	0.007	0.001
3	171.655	1.794	4.987	0.555
4	0.231	0.002	25.727	2.863
5	5794.622	60.574	563.552	62.729
6	2331.241	24.370	82.377	9.169
7	1162.497	12.152	213.978	23.817
8	83.266	0.870	6.256	0.696
9	21.670	0.227	0.644	0.073
Total	9566.164	100.00	898.402	100.00

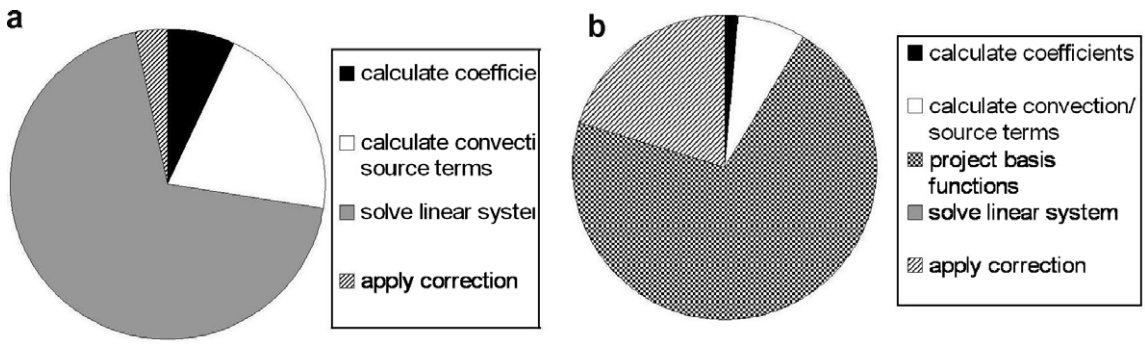


Fig. 4. Time profile of group 7: (a) FOM; (b) ROM.

Table 5
Number of operations for the projection routines in the ROM

Variable	Additions	Multiplications
u_m	$(i_{\max}-1)(j_{\max}+2)(8N_{u_m}^2+9N_{u_m})$	$(i_{\max}-1)(j_{\max}+2)(6N_{u_m}^2+7N_{u_m})$
v_m	$i_{\max}(j_{\max}+2)(8N_{v_m}^2+9N_{v_m})$	$i_{\max}(j_{\max}+2)(6N_{v_m}^2+7N_{v_m})$
p_g	$i_{\max}\cdot j_{\max}(6N_{p_g}^2+2N_{p_g})$	$i_{\max}\cdot j_{\max}(7N_{p_g}^2+2N_{p_g})$
ϵ_g	$i_{\max}\cdot j_{\max}(6N_{\epsilon_s}^2+2N_{\epsilon_s})$	$i_{\max}\cdot j_{\max}(7N_{\epsilon_s}^2+2N_{\epsilon_s})$

To further determine the difference between the reduced-order model and the full-order model, the number of operations was estimated in the projection subroutines of the ROM.

Table 5 shows that the number of operations in the ROM varies as a quadratic function of the number of modes. In the FOM, the number of total operations was approximately $28(i_{\max}+2)(j_{\max}+2)$ [21]. Therefore the number of operations per subiteration is always greater in the ROM than in the FOM, except for the atypical case when only one mode is used to represent each field variable.

5. Acceleration methods

This section describes several methods developed to further decrease the computational time of the reduced-order model. The techniques presented herein are: (i) an algorithm for splitting the database; (ii) an algorithm for solving quasi-symmetrical matrices; (iii) a strategy for reducing the frequency of updating $\widetilde{\mathcal{A}}$. The influence of the time step adjustment strategy is also investigated.

5.1. Database splitting

The POD basis functions are extracted from a database of snapshots generated by numerically integrating the governing differential equations. Currently, it is common to use a database that includes all the snapshots. Using a single database that covers the entire time domain, however, could be too restrictive. For example, consider the transience during the startup of the flow in a fluidized bed. The large variation in time at startup requires more modes than are necessary to model the flow features present in the latter part of the simulation. A method to avoid this problem is to split the database of snapshots.

Splitting the database into multiple subsets produces an auto-correlation matrix $\overline{\overline{R}}$ that contains more relative energy in the first modes. Herein, energy is defined as the sum of all the POD eigenvalues. The relative energy captured by the kth mode is defined as $\lambda_k / \sum_{i=1}^{M} \lambda_i$ [11]. As the relative energy of the first modes increases, fewer POD modes are needed in the reconstruction (11) to approximate the solution. Consequently, the computational cost of the reduced-order model decreases.

The snapshots created by solving the full-order model for the minimum fluidization case were divided into two parts. The first part, which captured the transient flow, ranged from 0.2 to 0.35 s and included 60 snapshots. The second part, which captured the slower varying flow, ranged from 0.35 to 1.0 s and included 260 snapshots. Fig. 5 shows the cumulative energy of the POD modes obtained using a single database that covered the entire time domain. Figs. 6 and 7 show the cumulative energy of the POD modes for the split databases.

The energy variation extracted from the 0.2–1.0 s database, shown in Fig. 5, was similar to the energy variation extracted from the transient snapshots, shown in Fig. 6. Most of the energy extracted from the 0.35–1.0 s database was, however, concentrated in the first mode, as shown in Fig. 7. This concentration of the energy allowed capturing most of the flow features using fewer modes compared to the transient regime.

Computing the auto-correlation matrix for each database subset is a straight forward process. Determining the bounds of each subset such that to reduce the computational cost is less trivial. Two methods for the separation of the snapshots into subsets were explored.

The first method measured the time variation of the time coefficients, α, of the dominant modes of each field variable. The variation of the time coefficients must be calculated and monitored for several modes for every field variable. Monitoring all of these values can, in some cases, produce conflicting information. An alternative to monitoring all six field variables was to monitor only the field variables that most affect the flow. For the minimum fluidization case, the flow was most affected by the first modes of gas pressure and gas velocity in the y-direction. The time variation of the first time coefficient of gas pressure, $\alpha_1^{p_g}$, is shown in Fig. 8. The variation of the first time coefficient of the gas pressure, $\Delta\alpha_1^{p_g} = |\alpha_1^{p_g}(t + \Delta t) - \alpha_1^{p_g}(t)|$, is shown in Fig. 9. In this case, the end of the transient regime was at $t = 0.35$ s.

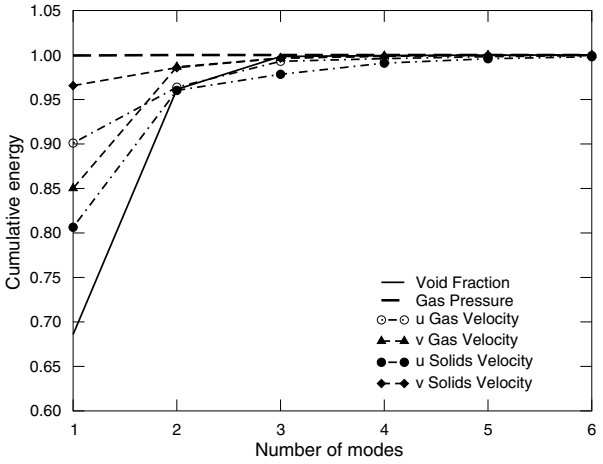


Fig. 5. Cumulative energy spectrum for a database that spans 0.2–1.0 s.

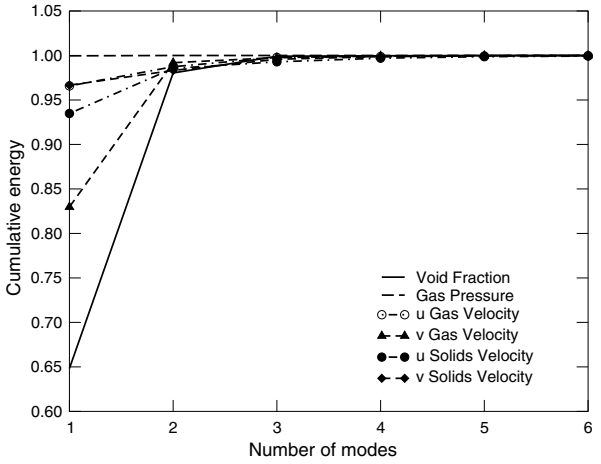


Fig. 6. Cumulative energy spectrum for a database that spans 0.2–0.35 s.

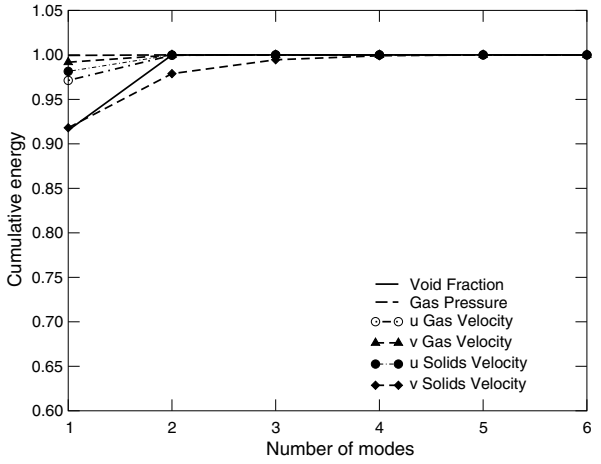


Fig. 7. Cumulative energy spectrum for a database that spans 0.35–1.0 s.

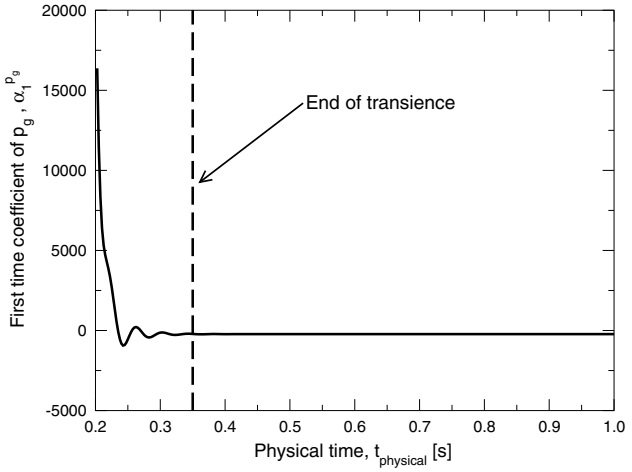


Fig. 8. Time history of the first time coefficient of gas pressure, $\alpha_1^{p_g}$.

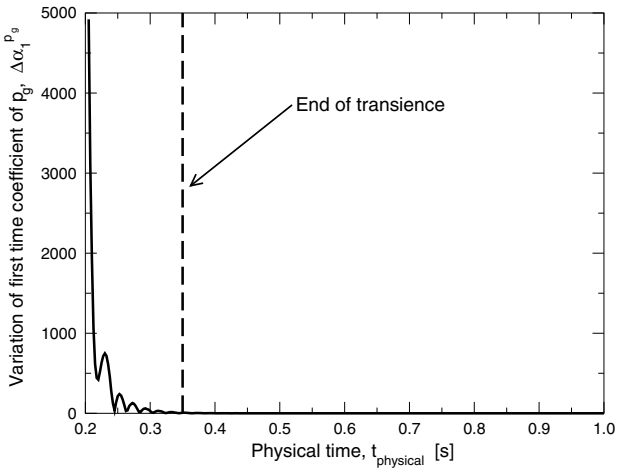


Fig. 9. Time history of the variation of the first time coefficient of gas pressure, $\Delta\alpha_1^{p_g}$.

The advantage of this method is that the end of the transient regime can be accurately detected during calculation. The disadvantage of this method is that there is not a unique value that determines the limit of the transience for all six field variables.

The second method proposed for separating the snapshots was to monitor the ratio of the change in CPU time and the change in physical time, $\Delta t_{\mathrm{CPU}}/\Delta t$. During transience, longer computation times are needed per time step as shown in Fig. 10. The advantage of monitoring this parameter is that the time slope is a single value that describes the behavior of all six field variables. The disadvantage of this method is that it over predicts the end of the transience. The magnitude of the slope decreased rapidly until $t = 0.3$ s and continued to decrease somewhat slower to a quasi-constant value at $t = 0.45$ s. Placing the end of the transient region at $t = 0.45$ s is more conservative than the 0.35 s predicted by the first method.

For the minimum fluidization case without database splitting, the reduced-order model with the modes given by Set 2 of Table 2 was 21 times faster then the full-order model. When the database was split at $t = 0.35$ s, the number of modes used from 0.2 to 0.35 s was given by Set 2 of Table 2 and the number of modes used from 0.35 to 1.0 s was 1 for each variable. Splitting the database at $t = 0.35$ s resulted in a speed-up of 30 compared to the full-order model. The increase in the speed-up factor was result of using fewer modes in the post transient period.

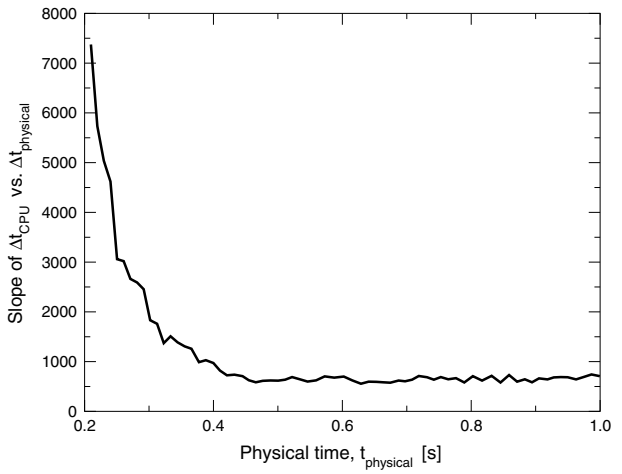


Fig. 10. Slope of the total CPU time vs physical time measured at 0.01 s increments.

The error of the reduced-order model with respect to the full-order model was defined as [2]

$$\varepsilon_{\aleph} = \sqrt{\sum_{i=1}^{N}(\aleph_i^{\mathrm{FOM}} - \aleph_i^{\mathrm{ROM}})^2} \Bigg/ \sum_{i=1}^{N}|\aleph_i^{\mathrm{FOM}}|, \tag{25}$$

where $\aleph$ represents the dependent field variable, FOM denotes the full-order model, and ROM denotes the reduced-order model with or without freezing. Here N is the total number of spatial grid points. The errors of each variable were then used to calculate the average error as

$$\varepsilon_{\mathrm{avg}} = (\varepsilon_{p_g} + \varepsilon_{\epsilon_s} + \varepsilon_{u_g} + \varepsilon_{v_g} + \varepsilon_{u_s} + \varepsilon_{v_s})/6. \tag{26}$$

In the case without database splitting the average error at $t = 1.0$ s was 1.882E−2. The average error at $t = 1.0$ s while using database splitting was 4.370E−2. The error increased in the latter case because fewer modes were used in the proper orthogonal decomposition.

5.2. Solver for linear systems with quasi-symmetric matrices

Let us consider the system of equations generated by projecting the solids volume fraction correction equation onto the basis functions extracted from an ensemble of ϵ_s snapshots. This system was derived in Section 3 and was written as

$$\widetilde{\mathcal{A}}^{\epsilon_s}\alpha^{\epsilon_s} = \widetilde{\mathcal{B}}^{\epsilon_s}, \tag{15}$$

where α^{ϵ_s} is the vector of unknowns $\alpha_i^{\epsilon_s}$. The ℓk element of the $\widetilde{\mathcal{A}}^{\epsilon_s}$ matrix is

$$\widetilde{\mathcal{A}}_{\ell k}^{\epsilon_s} = \{\varphi_\ell\}^{\mathrm{T}}[A]\{\varphi_k\} - \sum_{nb=1}^{NB}\{\varphi_\ell\}^{\mathrm{T}}[A_{nb}]\{\varphi_{k_{nb}}\}, \quad \ell, k = 1, \ldots, m. \tag{24}$$

The $\widetilde{\mathcal{A}}^{\epsilon_s}$ matrix is not symmetrical because of the second term $-\sum_{nb=1}^{NB}\{\varphi_\ell\}^{\mathrm{T}}[A_{nb}]\{\varphi_{k_{nb}}\}$ of the element $\widetilde{\mathcal{A}}_{\ell k}^{\epsilon_s}$. The matrix would be symmetrical if $\{\varphi_k\} = \{\varphi_{k_{nb}}\}$. The difference between the two vectors $\{\varphi_k\}$ and $\{\varphi_{k_{nb}}\}$ is small, however, because the latter vector is evaluated at slightly different spatial locations compared to the first vector. Similarly, the systems of linear algebraic equations (14)–(16) have matrices that are not symmetrical. These matrices, however, are quite close to being symmetrical, and for this reason will be called quasi-symmetrical. A typical example of a quasi-symmetrical matrix is the $\widetilde{\mathcal{A}}$ matrix obtained for $m = 8$ [22]

$$\widetilde{\mathcal{A}} = \begin{vmatrix} 196.4486 & 63.3060 & 6.0469 & 0.5038 & -21.3047 & 11.9071 & 2.3488 & -6.8064 \\ 63.3060 & 903.4807 & -44.1690 & 6.3410 & 14.0286 & -7.4939 & 6.1636 & 19.8724 \\ 6.0459 & -44.1687 & 243.2099 & -20.7951 & -164.8536 & 68.0529 & 19.3275 & -42.8377 \\ 0.5039 & 6.3411 & -20.7953 & 930.9194 & 31.0348 & 20.0166 & 14.3861 & 15.2768 \\ -21.3042 & 14.0288 & -164.8535 & 31.0347 & 890.8742 & 32.1664 & 42.8224 & -23.8698 \\ 11.9068 & -7.4940 & 68.0527 & 20.0167 & 32.1663 & 904.3555 & -10.8230 & 26.7999 \\ 2.3477 & 6.1634 & 19.3267 & 14.3861 & 42.8222 & -10.8228 & 872.6460 & 92.5161 \\ -6.8042 & 19.8722 & -42.8362 & 15.2768 & -23.8695 & 26.7996 & 92.5161 & 763.9839 \end{vmatrix}. \tag{27}$$

The algorithm proposed herein for solving a system of equations $Ax = b$, in which the matrix A must be positive definite and quasi-symmetrical, begins by splitting the matrix into a symmetrical and a non-symmetrical part [22]:

$$(A_s + A_n)x = b. \tag{28}$$

The decomposition of the A matrix into a symmetrical and non-symmetrical part is not unique. This issue will be discussed later in the section, while for the moment one will consider that one of the possible choices was used to split the A matrix.

A solution of the symmetrical part is then obtained by solving the system of equations

$$A_s x_s^{(1)} = b. \tag{29}$$

The solution of (28) is decomposed in a component obtained by solving the system (29) and a correction needed because the matrix A is non-symmetrical

$$x = x_s^{(1)} + x_n^{(1)}. \tag{30}$$

Substituting (30) into (28) and deducting (29) yields

$$\begin{aligned} &(A_s + A_n)x_n^{(1)} = -A_n x_s^{(1)} \quad \text{or} \\ &(A_s + A_n)x_n^{(1)} = b^{(1)}. \end{aligned} \tag{31}$$

Note that the system of equations (31) has the same matrix as (28). Consequently, an iterative process can be used to find the solution. $x_n^{(1)}$ can be considered a correction of the solution $x_s^{(1)}$ that is needed because the A matrix is non-symmetrical. The correction $x_n^{(1)}$ can be split into two components: a component $x_s^{(2)}$ obtained by solving the system $A_s x_s^{(2)} = b^{(1)}$ and a correction needed because the matrix A is non-symmetrical, similarly to the approach used in (30):

$$x_n^{(1)} = x_s^{(2)} + x_n^{(2)}. \tag{32}$$

Substituting (32) into (31) and using $A_s x_s^{(2)} = b^{(1)}$ yields

$$(A_s + A_n)x_n^{(2)} = -A_n x_s^{(2)} \quad \text{or} \tag{33}$$

$$(A_s + A_n)x_n^{(2)} = b^{(2)}. \tag{34}$$

This process of approximating the solution yields after p steps

$$x = x_s^{(1)} + x_s^{(2)} + \cdots + x_s^{(p)} + x_n^{(p)}, \tag{35}$$

where the values $x_s^{(i)}$, $1 \leqslant i \leqslant p$, are obtained by solving the linear system

$$A_s x_s^{(i)} = b^{(i)}, \tag{36}$$

where $b^{(i)} = -A_n x_s^{(i-1)}$. The iterative process of adding corrections is stopped when $x_s^{(p)}$ is smaller than an imposed error.

The computation of $x_s^{(i)}$ requires the solution of the linear system (36) several times. The Cholesky decomposition is used for the factorization of the positive-definite symmetrical matrix A_s. The method takes advantage of the fact that the A_s is constant and only the right-hand-side vector $b^{(i)}$ changes.

The method proposed herein replaced the LU decomposition [23, p. 34] that was previously used to solve the linear algebraic systems (14)–(17) [2]. For a system with m equations, the number of operations for the LU decomposition is $m^3/3$ while the number of operations for the Cholesky decomposition is $m^3/6$.

As mentioned previously in this section, the decomposition of the A matrix into a symmetrical and a non-symmetrical part is not unique. Two decompositions were explored herein in order to evaluate their effect on the convergence of the solver algorithm.

In the first decomposition, the A matrix, a_{ij}, $i, j = 1, \ldots, m$, was split such that the symmetrical and non-symmetrical parts were

$$\begin{aligned} A_s &= \begin{bmatrix} a_{11} & a_{21} & \ldots & a_{m1} \\ a_{21} & a_{22} & \ldots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mm} \end{bmatrix}, \\ A_n &= \begin{bmatrix} 0 & a_{12} - a_{21} & \ldots & a_{1m} - a_{m1} \\ 0 & 0 & \ldots & a_{2m} - a_{m2} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}. \end{aligned} \tag{37}$$

In the second decomposition, the A matrix was split into a symmetrical matrix A_s and skew-symmetrical matrix A_n

$$A_s = \frac{1}{2}(A + A^{\mathrm{T}}),$$
$$A_n = \frac{1}{2}(A - A^{\mathrm{T}}). \tag{38}$$

These two decompositions were applied to solve the system of equations $Ax = b$ where the A matrix was given by (27) and the right-hand-side term b was

$$b = \{-33.601113.487 - 8.81839.586 - 21.22133.225 - 49.50717.622\}^{\mathrm{T}}.$$

For an imposed error of 10^{-9}, the iterative solver algorithm converged for both decompositions in three steps. The vector of corrections $x_s^{(i)}$, shown in Table 6, indicates that in these cases the matrix decomposition had a minimal effect on the convergence. The Euclidean norm of the relative error between the solutions obtained using the two decompositions was 8.9×10^{-31}.

Numerical tests showed that three to four $x_s^{(i)}$ terms in (35) were usually sufficient to obtain a solution with an error less than 10^{-6}. Consequently, three to four $x_s^{(i)}$ solutions of the linear algebraic system of equations (36) must be computed. Since only the right-hand-side term $b^{(i)}$ changes while the matrix A_s is constant, the number of operations for the proposed method was increased by a factor proportional to m^2 multiplied by the number of $x_s^{(i)}$ terms in (35). As long as m is larger than 4, the computational cost of the proposed method is approximately half that of the LU decomposition. It should be noted that this estimate does not include the cost of searching for optimum symmetrization. The results shown in Table 6 indicated, however, that for the case studied, the results were not affected by the type of symmetrization.

The magnitude of the non-symmetrical terms was gradually increased in numerical tests. Herein, the degree of non-symmetry was defined as [24]

$$\eta = \|A - A^{\mathrm{T}}\|_{\mathrm{F}} / \|A + A^{\mathrm{T}}\|_{\mathrm{F}},$$

where $\| \ \|_{\mathrm{F}}$ indicates the Frobenius (or the Hilbert-Schmidt) norm [25, p. 56]. The solution of the $Ax = b$ system mentioned above was computed for three error levels: $10^{-6}, 10^{-9}$, and 10^{-12}. The A matrix was decomposed using split option 1 (37). The degree of non-symmetry of the A matrix was increased by multiplying the non-symmetric matrix A_s by different factors X, ranging from 1 to 5×10^8. The variation of the degree of non-symmetry of matrix A as a function of the factor X is shown in Fig. 11.

The variation of the number of iterations needed for convergence as a function of the degree of non-symmetry η is shown in Fig. 12a. It is remarkable that five or less iterations are needed for the convergence of the solution for degree of non-symmetry $\eta = 0.01$ that corresponds to a multiplying factor $X = 10,000$. The number of iterations increased to approximately 20 for η larger than 0.1 which correspond to multiplying factors larger than 10^5. Certainly at these large X values the matrix is far from a quasi-symmetric matrix. If the multiplying factor X is further increased, the matrix is no longer positive definite and the algorithm cannot be used because it relies on the Cholesky decomposition.

Table 6
Vector of corrections $x_s^{(i)}$ corresponding to matrix (27) for decompositions (37) and (38)

	Split 1 (37)			Split 2 (38)		
	$x_s^{(1)}$	$x_s^{(2)}$	$x_s^{(3)}$	$x_s^{(1)}$	$x_s^{(2)}$	$x_s^{(3)}$
1	0.2205E+00	−0.5529E−06	0.7659E−12	0.2205E+00	−0.3159E−06	−0.3772E−11
2	−0.1401E+00	0.3505E−07	−0.1631E−12	−0.1401E+00	0.4948E−07	0.2289E−12
3	0.3053E−01	−0.2586E−06	−0.3920E−12	0.3053E−01	0.3431E−06	−0.2669E−11
4	−0.4188E−01	−0.1645E−07	0.8489E−14	−0.4188E−01	0.1406E−07	0.5404E−14
5	0.3669E−01	−0.8025E−07	−0.6626E−13	0.3669E−01	−0.2367E−07	−0.6342E−12
6	−0.4223E−01	0.4975E−07	0.3221E−13	−0.4223E−01	0.4024E−07	0.3478E−12
7	0.5685E−01	0.1344E−07	0.1147E−13	0.5685E−01	0.1726E−06	0.6542E−13
8	−0.1916E−01	−0.2588E−07	−0.1567E−13	−0.1916E−01	−0.4003E−06	−0.9619E−13

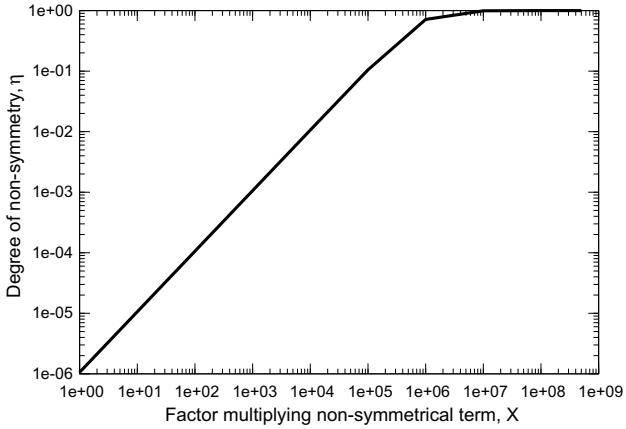


Fig. 11. Degree of non-symmetry η as a function of the multiplying factor X.

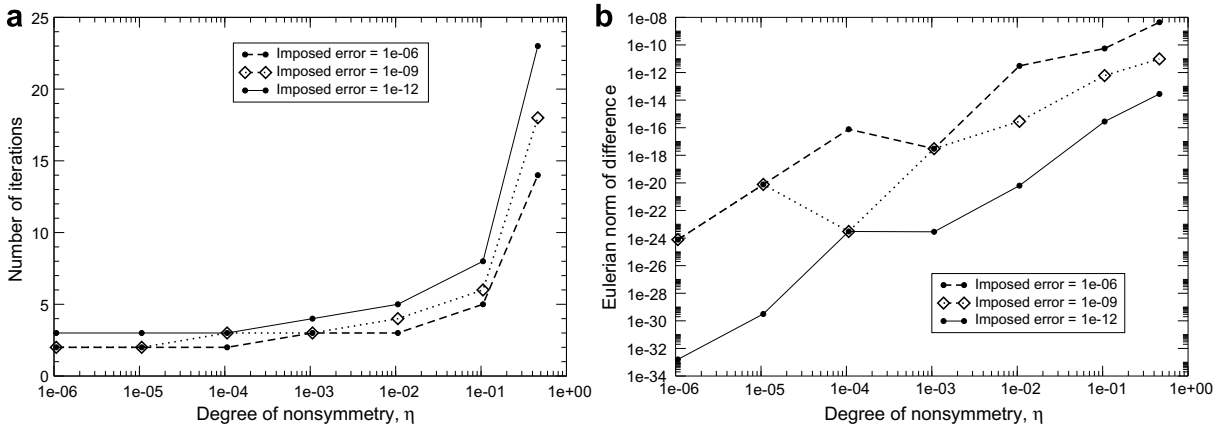


Fig. 12. Effect of degree of non-symmetry on: (a) number of iterations; (b) Eulerian norm of difference between the solutions of the LU decomposition and the present method.

The Eulerian norm of the difference between the solution obtained using the LU decomposition and the method proposed herein is shown in Fig. 12b as a function of the degree of non-symmetry η. The norm of the difference increases as the degree of non-symmetry increases. The norm of difference is, however, smaller than 10^{-10} even for values of η as large as 0.1. Consequently, the method can be applied to a larger class of matrices than just matrices obtained in the proper orthogonal decomposition method, as long as these matrices are positive definite.

5.3. Freezing the matrix of the linear system

The projection of the discretized differential equation onto the basis functions takes most of the computational time of a subiteration, as shown in Fig. 4b. For a minimum fluidization case [20] numerical tests showed that the components of the projected $\widetilde{\mathcal{A}}$ matrix do not vary significantly past the transient period. To quantify the variation of the $\widetilde{\mathcal{A}}$ matrix, the eigenvalues of $\widetilde{\mathcal{A}}(t)^{-1}\widetilde{\mathcal{A}}(t+\Delta t)$, where Δt is the subiteration time step, were compared against 1, the eigenvalues of identity matrix. Table 7 shows the results of a comparison performed using the $\widetilde{\mathcal{A}}^{v_g}$ matrices extracted from the transient and post transient periods. The eigenvalues varied by $4.1 \times 10^{-3}\%$ between subiterations in the transient period, and by $6.7 \times 10^{-7}\%$ in the post transient period. Similar results were obtained for the other dependent variables. Since the $\widetilde{\mathcal{A}}$ matrices of the linear systems

Table 7
Eigenvalues of $\widetilde{\mathcal{A}}^{v_g^{-1}}(t)\widetilde{\mathcal{A}}^{v_g}(t+\Delta t)$ during the transient ($t = 0.25$ s) and post transient (t=0.9 s) regime

Time	Eigenvalues				
t (s)	λ_1	λ_2	λ_3	λ_4	λ_5
0.25	1.000000000	1.000000000	1.000000000	0.999982334	0.999936216
0.90	1.000000000	1.000000000	1.000000000	1.000000000	0.999999933

(14)–(17) did not vary significantly between two subiterations, they were updated every other iteration to reduce computational time. The $\widetilde{\mathcal{B}}$ vectors, however, were updated every subiteration.

Due to the possibility of accumulating errors because of freezing the $\widetilde{\mathcal{A}}$ matrix, skipping to update $\widetilde{\mathcal{A}}$ was started only when the transient period ended. The relationship between the computational time and the physical time was used herein to determine when to stop updating the matrix every subiteration. Fig. 13 shows that for $t > 0.4$ s the relationship between the CPU time and the physical time was close to linear. This is in agreement with the results shown in Fig. 10 and indicates that the transient period ends at $t = 0.4$ s.

The application of freezing resulted in a 10.1% decrease of computational time while using the modes from Set 2 of Table 2. There was, however, an increase of the error compared to the reduced-order model without freezing. The error of the reduced-order model with respect to the full-order model was defined by (25). The error of the ROM with freezing, ε^{F}, with respect to the ROM without freezing, $\varepsilon^{\mathrm{NF}}$, was defined as

$$\varepsilon^{\mathrm{F-NF}} = |\varepsilon^{\mathrm{NF}} - \varepsilon^{F}|/\varepsilon^{\mathrm{NF}}.$$

Table 8 shows the errors of some of the dependent field variables at time $t = 1$ s, the end of the integration period. The largest error $\varepsilon^{\mathrm{F-NF}}$ corresponds to the solids volume fraction. The errors between the reduced-order model and the full-order model for the solids volume fraction are extremely small. Consequently, the increase of the error $\varepsilon_{\epsilon_s}^{\mathrm{F-NF}}$ in this case is not a concern.

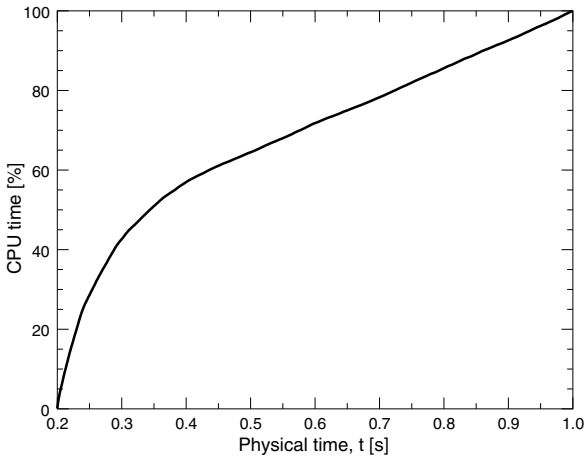


Fig. 13. Relationship between physical time and CPU time.

Table 8
Errors at $t = 1.0$ s with and without freezing, using POD modes specified by Set 2

Variable	$\varepsilon^{\mathrm{NF}}$ (%)	ε^{F} (%)	$\varepsilon^{\mathrm{F-NF}}$ (%)
u_g	0.215	0.224	0.419
v_g	0.340	0.362	6.471
p_g	1.544×10^{-5}	1.643×10^{-5}	6.412
ϵ_s	4.159×10^{-5}	5.059×10^{-5}	21.640

Given that freezing the projection of the $\widetilde{\mathcal{A}}$ matrix greatly reduces the number of operations per subiteration, a 10.1% decrease of CPU time is a small improvement. This limited reduction of the computational time was due to two factors. First, freezing was applied beginning at time $t = 0.4$ s and for this reason affected only a portion of the integration time. Second, an increase in the number of subiterations diminished the benefit of saving computational time by not updating the $\widetilde{\mathcal{A}}$ matrix.

5.4. Time step adjustment

The FOM and the ROM use an identical method to adjust the time step during the integration. This method can vary the size and the frequency of the time step adjustment. Given identical initial time steps and time step adjustment parameters, the time step size in the ROM increased by at least one order of magnitude, while the time step in the FOM remained almost constant, as shown in Fig. 14.

Three parameters were modified in the ROM to reduce the computation time: (1) the frequency of the time step adjustment; (2) the size of the time step adjustment; (3) the size of the initial time step. Numerical tests showed that the variation of the initial time step size was the most effective time step adjustment speed-up method. The initial time step for the minimum fluidization case was varied between 10^{-4} and 10^{-3} s. Increasing the initial time step size from 10^{-4} to 10^{-3} s reduced the computational time between 6.1% and 22.3%, depending on the number of modes used for the minimum fluidization case. This relatively small reduction in computational time did not reflect the increase of the initial time step. Much of the computational speed-up was lost because the size of the time step was reduced during the integration in the transient period to satisfy convergence.

The optimal value for reducing the computational time was found to be 5×10^{-4} s while the time step was adjusted by $\pm 10\%$ every 5 iterations. Increasing the initial time step above the optimal value did not diminish the computational time because the time step must be reduced to achieve convergence. On the other hand, when the initial time step was smaller than the optimal value, the time steps did not grow as large as in the case with optimal initial time step.

5.5. Summary of acceleration methods

A summary of the speed-up factors achieved by the various versions of the reduced-order model is given in Table 9. The speed-up factors are reported with respect to the full-order model. All the reduced-order models used the number of modes specified by Set 2 of Table 2. Table 9 also shows the errors ε_{avg} (26) between the full-order model and the various versions of reduced-order models.

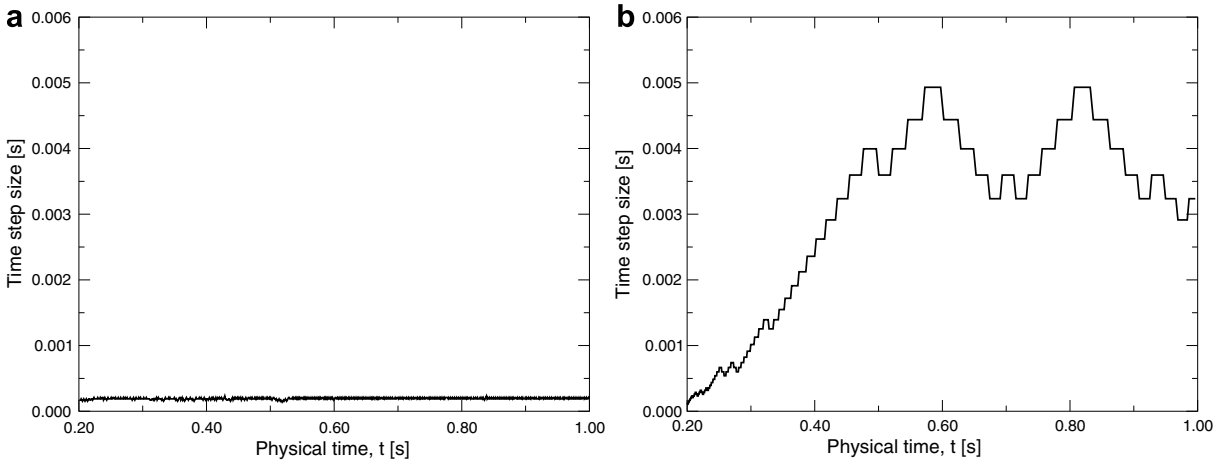


Fig. 14. Time step size vs. physical time: (a) FOM; (b) ROM.

Table 9
Speed-up factors and averaged errors: ROM vs. FOM

	Speed-up factor	Error, ε_{avg}
FOM	1	0
ROM with no acceleration	21	1.882E−2
ROM with database splitting	30	4.370E−2
ROM with projection freezing	23	0.151818
ROM with initial time step adjustment	25	1.601E−2
ROM with database splitting and initial time step adjustment	114	6.240E−2

As shown in Table 9, the smallest errors correspond to the reduced-order model with initial time step adjustment. The largest errors correspond to the reduced-order model with projection freezing. Since the reduced-order model with projection freezing had the largest errors and the smallest speed-up factor, this acceleration technique was considered least useful. Consequently only the acceleration techniques based on database splitting and time step adjustment were combined.

Contour plots of the gas pressure and y-direction gas velocity at time $t = 1.0$ s computed using the full-order model and the reduced-order models are shown in Figs. 15 and 16. These figures also include contour plots of the difference between the full-order model and the reduced-order models. Figs. 15 and 16 show that the contour plots of pressure and y-direction gas velocity are not affected by the ROM used. Indeed, the differences between the FOM and the ROMs are small, their relative values being at most of the order of 10^{-8} for pressure and 10^{-4} for gas velocity. The largest errors corresponded to the solids velocities, and their values were of the order of 10^{-2}.

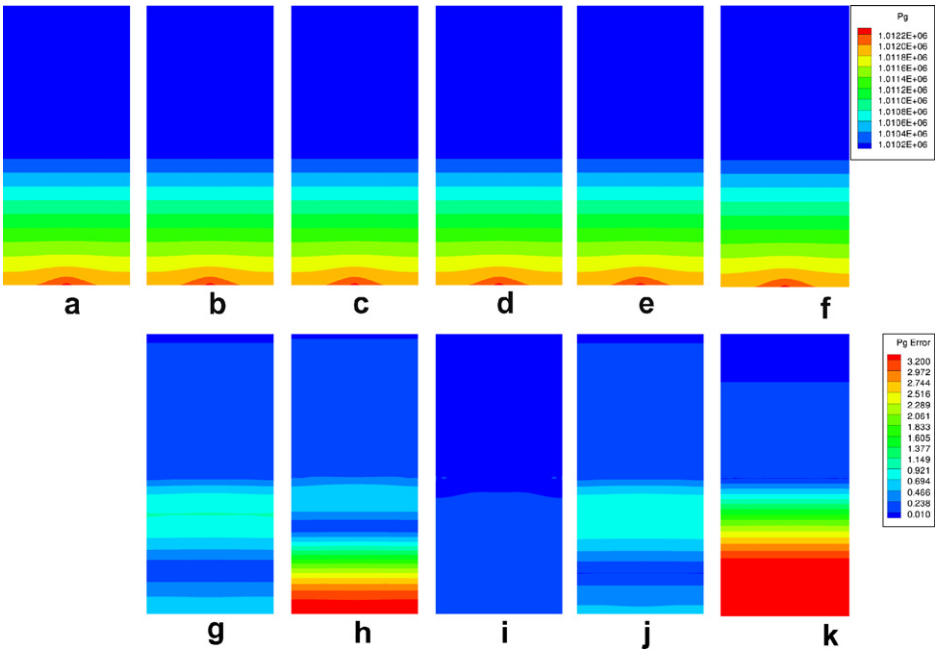


Fig. 15. Contour plots of gas pressure: (a) full-order model (FOM); (b) reduced-order model (ROM), no acceleration; (c) ROM, matrix freezing; (d) ROM, database splitting; (e) ROM, initial time step adjustment; (f) ROM, database splitting and initial time step adjustment; and the pressure difference between (g) FOM and ROM, no acceleration; (h) FOM and ROM, matrix freezing; (i) FOM and ROM, database splitting, (j) FOM and ROM, initial time step adjustment; (k) FOM and ROM, database splitting and initial time step adjustment (all values at $t = 1.0$ s).

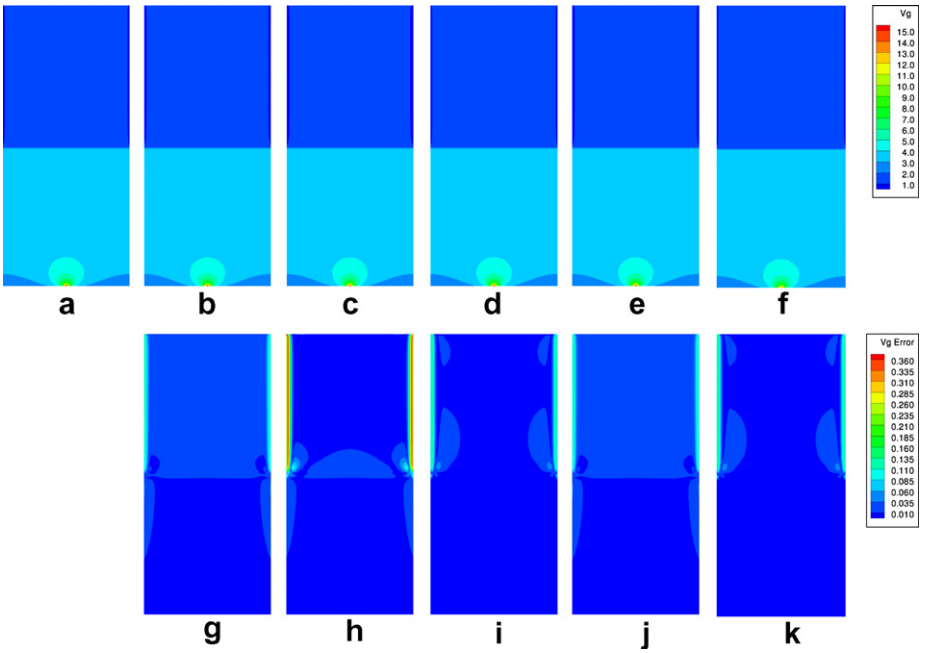


Fig. 16. Contour plots of y-direction gas velocity: (a) full-order model (FOM); (b) reduced-order model (ROM), no acceleration; (c) reduced-order model, matrix freezing; (d) ROM, database splitting; (e) ROM, initial time step adjustment; (f) ROM, database splitting and initial time step adjustment; and the velocity difference between (g) FOM and ROM, no acceleration; (h) FOM and ROM, matrix freezing; (i) FOM and ROM, database splitting; (j) FOM and ROM, initial time step adjustment; (k) FOM and ROM, database splitting and initial time step adjustment (all values at $t = 1.0$ s).

6. Conclusions

This paper presented several acceleration methods for reduced-order models based on the proper orthogonal decomposition method: (i) database splitting; (ii) an algorithm for solving quasi-symmetrical matrices; (iii) a strategy for reducing the frequency of the projection.

The database splitting algorithm used the fact that fewer modes were needed for reconstruction when the time domain was divided. Splitting the database into multiple subsets produced auto-correlation matrices that contained more relative energy in the first modes. As the relative energy of the first modes increased, fewer POD modes were needed in the reconstruction. Consequently, the speed-up factor compared to the ROM without acceleration increased in the example considered herein by approximately 50%.

The algorithm for solving quasi-symmetrical matrices took advantage of the nearly symmetrical structure of the matrices of the linear algebraic systems that yield the time coefficients of the reduced-order model. The solution corresponding to the quasi-symmetrical matrix was approximated as a sum of solutions corresponding to the symmetrical part of the matrix. By operating only on a symmetrical, positive-definite matrix, the Cholesky decomposition allowed a reduction of the computational time by a factor of 2. The additional cost for computing the corrections due to the matrix non-symmetry was minimal because: (i) typically three to four iterations were sufficient to achieve an error less than 10^{-6}; (ii) the operations count for calculating a correction was $\mathcal{O}(m^2)$, where m was the matrix size.

The technique was developed for a POD-based reduced-order model of two-phase flows in fluidized beds. For this application, however, the impact of halving the time for computing the linear system solution was negligible because the time required to solve the system of equations was less than 1% of the total computational time. The method, nevertheless, can be used to solve a broader class of linear algebraic systems that have positive-definite matrices.

The matrix freezing approach increased the computational efficiency by reducing the number of operations performed during the post transient calculations. Although the $\widetilde{A}$ matrix was not updated every subiteration,

the right-hand-side vector, $\widetilde{\mathcal{B}}$, was updated every subiteration because it was computationally inexpensive and beneficial. Matrix freezing led to a small decrease in the computation time (10%) because it was applied only for a portion of the integration and because occasionally the number of subiterations increased in order to achieve a converged solution.

These acceleration methods were tested both individually and in various combinations. The best combination proved to be database splitting in conjunction with initial time step adjustment. Using these two methods, a speed-up factor of 114 was achieved for the ROM, as compared to the FOM, for the minimum fluidization case.

Acknowledgment

This work was sponsored by the Department of Energy under Grant No. DE-FC26-05NT42445.

References

[1] P.G.A. Cizmas, A. Palacios, T. O'Brien, M. Syamlal, Proper-orthogonal decomposition of spatio-temporal patterns in fluidized beds, Chem. Eng. Sci. 58 (19) (2003) 4417–4427.
[2] T. Yuan, P.G. Cizmas, T. O'Brien, A reduced-order model for a bubbling fluidized bed based on proper orthogonal decomposition, Comput. Chem. Eng. 30 (2) (2005) 243–259.
[3] E.H. Dowell, K.C. Hall, J.P. Thomas, R. Florea, B. Epureanu, J. Heeg, Reduced order models in unsteady aerodynamics, in: 40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, St. Louis, MO, 1999.
[4] R. Florea, K.C. Hall, Reduced order modeling of unsteady flows about airfoils, in: Aeroelasticity and Fluid Structure Interaction Problems, ASME International Mechanical Engineering Congress and Exposition, 1994, pp. 49–68.
[5] K.C. Hall, R. Florea, P.J. Lanzkron, A reduced order model of unsteady flows in turbomachinery, J. Turbomach. 117 (3) (1995) 375–383.
[6] K.C. Hall, Eigenanalysis of unsteady flows about airfoils, cascades, and wings, AIAA J. 32 (12) (1994) 2426–2432.
[7] R. Florea, K.C. Hall, P.G.A. Cizmas, Eigenmode analysis of unsteady viscous flows in turbomachinery cascades, in: T.H. Fransson (Ed.), Proceedings of the Eighth International Symposium on Unsteady Aerodynamics and Aeroelasticity of Turbomachines, Stockholm, Sweden, 1997, pp. 767–782.
[8] R. Florea, K.C. Hall, P.G.A. Cizmas, Reduced-order modeling of unsteady viscous flow in a compressor cascade, AIAA J. 36 (6) (1998) 1039–1048.
[9] L. Sirovich, Turbulence and the dynamics of coherent structures, Quart. Appl. Math. 45 (3) (1987) 561–590.
[10] H.M. Park, M.W. Lee, An efficient method of solving the Navier–Stokes equations for flow control, Int. J. Numer. Methods Eng. 41 (6) (1998) 1133–1151.
[11] P.G.A. Cizmas, A. Palacios, Proper orthogonal decomposition of turbine rotor–stator interaction, J. Propul. Power 19 (2) (2003) 268–281.
[12] Y. Utturkar, B.N. Zhang, W. Shyy, Reduced-order description of fluid flow with moving boundaries by proper orthogonal decomposition, Int. J. Heat Fluid Flow 26 (2) (2005) 276–288.
[13] C. Homescu, L.R. Petzold, R. Serban, Error estimation for reduced-order models of dynamical systems, SIAM J. Numer. Anal. 43 (4) (2005) 1693–1714.
[14] E.H. Dowell, D. Tang, Dynamics of Very High Dimensional Systems, World Scientific Publ. Co., New Jersey, 2003.
[15] D.J. Lucia, P.S. Beran, W.A. Silva, Reduced-order modeling: new approaches for computational physics, Prog. Aerospace Sci. 40 (1–2) (2004) 51–117.
[16] M. Syamlal, W. Rogers, T.J. O'Brien, MFIX documentation theory guide, Tech. Rep. DOE/METC-94/1004, DOE/METC, 1994.
[17] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Taylor & Francis, 1980.
[18] M. Syamlal, MFIX documentation numerical technique, Tech. Rep. DE-AC21-95MC31346, EG&G Technical Services of West Virginia, 1998.
[19] P. Holmes, J. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, 1996.
[20] T. Yuan, Reduced order modeling for transport phenomena based on proper orthogonal decomposition, Master's Thesis, Texas A&M University, College Station, TX, December 2003.
[21] F.S.B.F. Oliveira, K. Anastasiou, An efficient computational model for water wave propagation in costal regions, Appl. Ocean Res. 20 (5) (1998) 263–271.
[22] P.G.A. Cizmas, An acceleration approach for reduced-order models based on proper orthogonal decomposition, in: 45th Aerospace Sciences Meeting and Exhibit, AIAA Paper 2007-713, Reno, Nevada, 2007.
[23] W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery, Numerical Recipes in FORTRAN – The Art of Scientific Computing, second ed., Cambridge University Press, 1992.
[24] N. Li, Y. Saad, E. Chow, Crout versions of ILU for general sparse matrices, SIAM J. Sci. Comput. 25 (2) (2003) 716–728.
[25] G.H. Golub, C.F. van Loan, Matrix Computation, second ed., The Johns Hopkins University Press, Baltimore, 1989.