The Post–processed Galerkin Method Applied to Nonlinear Shell Vibrations

Carlo R. Laing[1,2,†]

Allan McRobie[1]

J. M. T. Thompson[2]

1. Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, United Kingdom.

2. Centre for Nonlinear Dynamics, University College London, Gower Street, London WC1E 6BT, United Kingdom.

† Corresponding author. Current address: Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260, USA.

October 23, 1998

Abstract

We present the results of a computational study of the post–processed Galerkin methods put forward by Garcia-Archilla et al. [7, 8] applied to the nonlinear von Karman equations governing the dynamic response of a thin cylindrical panel periodically forced by a transverse point load.

We spatially discretise the shell using finite differences to produce a large system of ordinary differential equations. By analogy with spectral nonlinear Galerkin methods we split this large system into a “slowly” contracting subsystem and a “quickly” contracting subsystem. We then compare the accuracy and efficiency of (i) ignoring the dynamics of the “quick” system (analogous to a traditional spectral Galerkin truncation and sometimes referred to as “subspace dynamics” in the finite element community when applied to numerical eigenvectors), (ii) slaving the dynamics of the quick system to the slow system during numerical integration (analogous to a nonlinear Galerkin method), and (iii) ignoring the influence of the dynamics of the quick system on the evolution of the slow system until we require some output, when we “lift” the variables from the slow system to the quick using the same slaving rule as in (ii). This corresponds to the post–processing of Garcia-Archilla et al.

We find that method (iii) produces essentially the same accuracy at method (ii) but requires only the computational power of method (i) and is thus more efficient than either. In contrast with spectral methods, this type of finite difference technique can be applied to irregularly–shaped domains. We feel that post–processing of this form is a valuable method that can be implemented in computational schemes for a wide variety of partial differential equations of practical importance.

1 General Introduction

It is known that the solutions of some dissipative nonlinear partial differential equations (PDEs) evolve to a compact set known as a global attractor. Various schemes have been used for constructing finite systems of ordinary differential equations (ODEs) that reproduce the asymptotic dynamics of the original infinite dimensional PDE — one of the most well known being the Galerkin method. This method effectively ignores the small spatial structure of a solution less than a certain size, concentrating instead on the large scale structures.

During the past decade nonlinear Galerkin methods have been introduced which attempt to use the finite dimensionality of the attractor of the PDE to incorporate the influence of the small scale structures on the temporal evolution of the large scale structures. This method has often been successful in producing a greater level of accuracy when compared with a Galerkin method with the same spatial threshold, but often at an increased computational cost, and when comparing accuracy achieved for a given amount of computer time, it is not always clear that the nonlinear Galerkin method has any advantage.

This dilemma has recently been overcome by the introduction of the post–processed Galerkin method, where the influence of the small scale structures on the temporal evolution of the large scale structures is ignored, leading to a computational cost similar to that of an ordinary Galerkin method, until a computation is completed. At this stage the computed solution is "post–processed" (at small computational cost) to recover the small scale structure. This method is often (although not always) more efficient than either the traditional Galerkin or nonlinear Galerkin methods.

We make analogies between a PDE to which these three methods can be applied and a large set of ODEs derived from a finite difference scheme for computing the vibrations of a nonlinear shell and come to a similar conclusion regarding the efficiency of the post–processed Galerkin method.

2 Introduction

There has recently been much interest in the existence of inertial manifolds in classes of dissipative nonlinear partial differential equations (PDEs). Nonlinear Galerkin methods [13], which attempt to completely describe the dynamics on the attractor of a PDE with a *finite–dimensional* dynamical system, have arisen from this theory.

Conceptually, nonlinear Galerkin methods split the infinite–dimensional phase space of the PDE into two complementary subspaces: a finite–dimensional one spanned by "slowly" contracting modes, and its infinite–dimensional complement. The dynamics in this infinite–dimensional space are then assumed to be "slaved" to the dynamics in the finite–dimensional space via an inertial manifold. Many computation schemes have been introduced where this slaving is used in the calculation of the dynamics. Recently, work by Garcia-Archilla et al. [7, 8] has shown that it is often more efficient to ignore the slaving when calculating the dynamics and only to use it when output from the system is actually required — this technique has been called "post–processing".

Much of the work relating to nonlinear Galerkin methods has been presented using spectral techniques. In this paper we make analogies between a large system of ordinary differential equations derived from a semi–explicit finite–difference scheme for the dynamics of a cylindrical panel forced at a point and a general PDE for which nonlinear Galerkin and post–processing techniques have been developed. In contrast with spectral methods, the finite–difference method we use can be applied to irregular domains. Our results regarding convergence rates and efficiency are similar to those obtained by other workers [7, 10] who have studied simpler PDEs using spectral methods.

3 Spectral nonlinear Galerkin methods

In this section we review the theory of spectral nonlinear Galerkin and post–processing methods with which we will later make analogies. For more details see [2, 4, 5, 7, 8, 9, 10].

Let us write our PDE as

$$\frac{du}{dt} + Lu + R(u) = f \tag{1}$$

where L is a linear spatial–differential operator, $R(u)$ contains nonlinear terms, and f is a forcing function. Assume that L has an infinite number of orthonormal eigenfunctions $\{w_i\}$ spanning the Hilbert space H that our solution u lives in, with eigenvalues $\{\lambda_i\}$ such that $Lw_i = \lambda_i w_i$, and let us order the λ_is so that $0 < Re\{\lambda_1\} \leq Re\{\lambda_2\} \leq Re\{\lambda_3\} \leq \ldots$ We define the finite–dimensional subspace H_k to be the span of the first k eigenfunctions, i.e. $H_k = \text{span}\{w_1, w_2, \ldots, w_k\}$, P_k to be the projection onto H_k, and $Q_k \equiv I - P_k$ to be the projection onto the complement of H_k.

Defining $p \equiv P_k u$ and $q \equiv Q_k u$ and applying first P_k and then Q_k to (1) we obtain two coupled ODEs, the first finite–dimensional and the second infinite–dimensional:

$$\frac{dp}{dt} + Lp + P_k[R(p+q)] \;=\; P_k f \qquad (2)$$

$$\frac{dq}{dt} + Lq + Q_k[R(p+q)] \;=\; Q_k f \qquad (3)$$

The traditional Galerkin method corresponds to setting $q = 0$ in (2) and integrating the resulting finite–dimensional system:

$$\frac{dy_k}{dt} + Ly_k + P_k[R(y_k)] = P_k f$$

We then approximate the true solution of (1) by $u \approx y_k$.

The rationale behind the development of inertial manifolds is that if the attractor of (1) is finite–dimensional, then it is reasonable to hypothesise that for some k there exists a graph, Φ_k, such that on the attractor $q = \Phi_k(p)$, i.e. on the attractor the behaviour of all the higher modes is completely governed by the behaviour of a finite number of the lower modes. If this is the case, then to determine the dynamics on the attractor we substitute Φ_k into (2) and integrate the equation

$$\frac{dz_k}{dt} + Lz_k + P_k[R(z_k + \Phi_k(z_k))] = P_k f,$$

reconstructing the solution as $u = z_k + \Phi_k(z_k)$.

It has been proven for some PDEs that such a graph Φ_k does exist. Finding Φ_k is another matter: in practice it may need to be approximated numerically. For some PDEs it has not yet been possible to show the existence of an inertial manifold, but the existence has been postulated of an *approximate inertial manifold* (AIM), Φ_{app}, which captures the behaviour of the higher modes in terms of the lower modes, although not exactly as in the case of an exact inertial manifold. In these cases we integrate the equation

$$\frac{dx_k}{dt} + Lx_k + P_k[R(x_k + \Phi_{app}(x_k))] = P_k f \qquad (4)$$

and reconstruct the solution as $u \approx x_k + \Phi_{app}(x_k)$. Although this will not exactly capture the dynamics on the attractor, it is likely to do a better job than assuming $q = 0$ in (2), as we do in the standard Galerkin method.

The question of finding Φ_{app} arises. A number of functions for Φ_{app} have been proposed which have varying degrees of accuracy (see, for example, references in [7, 8]). Here we use Φ_k^1, first defined by Foias et al. [5], which utilises (3) in its definition:

$$\Phi_k^1(p) \equiv L^{-1}\{Q_k f - Q_k[R(p)]\}$$

In practice, Q_k is the projection onto an infinite–dimensional space, so a truncation of it must be used. Thus we define

$$\Phi^1_{k,m}(p) \equiv L^{-1}\{Q_{k,m}f - Q_{k,m}[R(p)]\}$$

where $Q_{k,m} \equiv I - P_k - P_m$ and $m > k$. Equation (4) now becomes

$$\frac{dv_k}{dt} + Lv_k + P_k[R(v_k + L^{-1}\{Q_{k,m}f - Q_{k,m}[R(v_k)]\})] = P_k f \tag{5}$$

and we reconstruct the solution as $u \approx v_k + \Phi^1_{k,m}(v_k)$.

We can think of (5) as an equation for the dynamics on the approximate inertial manifold. The last term on the left side of (5) normally requires a lot of effort to evaluate, and the question of the computational efficiency of integrating (5) as opposed to a Galerkin projection onto s modes, where $s > k$, to obtain a desired degree of accuracy is raised.

The work of Garcia-Archilla et al. [7, 8] on post–processing Galerkin methods shows that it is sometimes possible to obtain the accuracy of a system like (5) with no more effort than that involved in integrating a k–mode Galerkin system. The idea is to integrate a k–mode Galerkin approximation:

$$\frac{dy_k}{dt} + Ly_k + P_k[R(y_k)] = P_k f \tag{6}$$

and then when output is required, say at time T, "lift" this data up to the approximate inertial manifold, i.e. write $u(T) \approx y_k(T) + \Phi^1_{k,m}(y_k(T))$. Garcia-Archilla et al. [7, 8] showed that this is often as accurate as the solution $v_k(T)+\Phi^1_{k,m}(v_k(T))$ obtained from integrating (5), but has the advantage that the system (6) is simpler to integrate. The lifting of the solution onto the approximate inertial manifold need only be carried out when output is required, rather than at every time–step in the numerical integration, as is the case when integrating (5).

We have reviewed the theory of only spectral nonlinear Galerkin methods in this section — similar ideas have also been applied to both finite element [14, 15] and finite difference [20] schemes, although as far as we are aware, the idea of post–processing has not been applied to a finite element scheme, and this is the first application to a finite difference scheme.

4 Finite difference model of a panel

In this section we discuss one approach that combines the ideas of approximate inertial manifolds and finite–difference numerical schemes for the von Karman equations (7-

8). The model we study is shown schematically in Figure 1. It is part of a cylindrical panel with radius R, forced at the point $(\bar{x}, \bar{y})$ with the function $f(t)$.

We should mention the paper by Foale et al. [3], which contains results similar to those presented here. Foale et al. studied a finite–difference model of a panel similar to that shown in Figure 1, the main difference being that their panel was axially forced on the boundaries rather than at a point. This may seem a minor difference, but it is thought that the efficiency of the nonlinear Galerkin method as opposed to the flat Galerkin method depends on (among other things) the smoothness of solutions of the PDE, which in turn depends on the smoothness of forcing [10]. Foale et al. [3] concluded that a nonlinear Galerkin method provided little advantage over a flat Galerkin method, but we feel that this was due to the very smooth solutions they obtained which was a result of the axial forcing they used.

Another approach to the study of the dynamics of such a shell is the standard spectral Galerkin method [1, 18], and there has also been some recent work on localised buckling in cylindrical shells using ideas relating to homoclinic orbits [11, 12], although these techniques can only be used in the quasistatic case.

The equations governing the motion of the plate are the von Karman equations:

$$\begin{aligned} D\nabla^4 w + \mu\frac{\partial^2 w}{\partial t^2} + \beta\frac{\partial w}{\partial t} + \beta_2\frac{\partial}{\partial t}\nabla^4 w &= \frac{\partial^2 w}{\partial x^2}\frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 w}{\partial y^2}\frac{\partial^2 \phi}{\partial x^2} - 2\frac{\partial^2 w}{\partial x \partial y}\frac{\partial^2 \phi}{\partial x \partial y} \\ &+ \frac{1}{R}\frac{\partial^2 \phi}{\partial x^2} + \delta(x-\bar{x})\delta(y-\bar{y})f(t) \qquad (7) \\ \frac{1}{Eh}\nabla^4\phi &= \left(\frac{\partial^2 w}{\partial x \partial y}\right)^2 - \frac{\partial^2 w}{\partial x^2}\frac{\partial^2 w}{\partial y^2} - \frac{1}{R}\frac{\partial^2 w}{\partial x^2} \qquad (8) \end{aligned}$$

where w is the normal displacement of the shell (positive inwards), t is time, ϕ is the in–plane Airy stress function, μ is the mass per unit area, β is the coefficient of linear damping, β_2 is the visco–elastic damping coefficient, h is the thickness of the shell, E is Young's modulus, R is the radius of the panel, ∇^4 is the biharmonic operator and

$$D = \frac{Eh^3}{12(1-\nu^2)}$$

is the flexural rigidity of the panel, where ν is Poisson's ratio.

The boundary conditions corresponding to simply–supported edges are

$$\frac{\partial^2\phi}{\partial y^2} = 0 \quad \text{at} \quad x = 0, a \tag{9}$$

$$\frac{\partial^2\phi}{\partial x^2} = 0 \quad \text{at} \quad y = 0, a \tag{10}$$

$$\frac{\partial^2\phi}{\partial x \partial y} = 0 \quad \text{at} \quad y = 0, a; x = 0, a \tag{11}$$

$$\frac{\partial^2 w}{\partial x^2} = 0 \quad \text{at} \quad x = 0, a \tag{12}$$

$$\frac{\partial^2 w}{\partial y^2} = 0 \quad \text{at} \quad y = 0, a \tag{13}$$

$$w = 0 \quad \text{at} \quad y = 0, a; x = 0, a \tag{14}$$

where a is the side length of the panel. (We set a =1m and do not refer to it again.)

In order to implement a finite–difference scheme we lay a regular cartesian grid over the domain, as shown in Figure 2, and approximate the spatial derivatives using standard central difference formulae:

$$\begin{aligned}
\nabla^4\psi_{i,j} &\approx [20\psi_{i,j} - 8(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j-1} + \psi_{i,j+1}) \\
&+ 2(\psi_{i-1,j-1} + \psi_{i-1,j+1} + \psi_{i+1,j-1} + \psi_{i+1,j+1}) \\
&+ \psi_{i-2,j} + \psi_{i+2,j} + \psi_{i,j-2} + \psi_{i,j+2}]/(\delta x)^4 \\
\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} &\approx [\psi_{i,j-1} - 2\psi_{i,j} + \psi_{i,j+1}]/(\delta x)^2 \\
\left.\frac{\partial^2\psi}{\partial x^2}\right|_{i,j} &\approx [\psi_{i-1,j} - 2\psi_{i,j} + \psi_{i+1,j}]/(\delta x)^2 \\
\left.\frac{\partial^2\psi}{\partial x \partial y}\right|_{i,j} &\approx [\psi_{i+1,j+1} + \psi_{i-1,j-1} - \psi_{i+1,j-1} - \psi_{i-1,j+1}]/(2\delta x)^2
\end{aligned}$$

where δx is the grid spacing. We define the integer n by saying that there are n^2 internal points in the plate. This implies that there are $4n + 4$ boundary points and a total of $(n + 4)^2 = n^2 + 8n + 16$ grid points, of which $4n + 12$ are "ghost points" lying δx further out than the boundary of the plate. We choose the point of forcing, $(\bar{x}, \bar{y})$, to be one of the grid points.

The first step is to solve (8) for ϕ as a function of w. We do this by rewriting (8) as

$$A\phi = f(w) \tag{15}$$

where A is a $(n+4) \times (n+4)$ matrix obtained by applying the discretisation of (8) at the internal points and applying the boundary conditions (9)-(11), and ϕ and w are now vectors representing their discretisations over all grid points. The application of (8)-(11) provides only $n^2 + 8n + 12$ equations, so we use a second approximation to the boundary condition

$$\frac{\partial^2 \phi}{\partial x \partial y} = 0$$

at all 4 corners, *viz.*

$$[\phi_{i,j} + \phi_{i+1,j+1} - \phi_{i+1,j} - \phi_{i,j+1}]/(\delta x)^2 = 0$$

to make up the last 4 equations, where one diagonal of the square of points to which this formula is applied always points from the corner point in towards the centre of the plate. Because of the nature of the boundary conditions, A is not invertible, having co–rank 4 if n is odd and 3 if n is even. Because of this, we use singular value decomposition [19] to solve (15). The solution we obtain has some arbitrary constants in it, the number being equal to the co–rank of A, but this does not matter as we need only the second spatial derivatives of ϕ for (7) and taking these derivatives removes the constants.

Having solved (15) for ϕ as a function of w we substitute this into (7) and write down the equations of motion for w at each of the n^2 internal grid points, which we label $w_1, w_2, \ldots, w_{n^2}$, making use of the boundary conditions on w (12)-(14). However, (7) contains the second derivative of w with respect to time, and it is easier to work with a system containing only first derivatives. Because of this, we define a new variable u such that

$$[u_1, u_2, \ldots, u_{n^2}] \equiv [w_1, w_2, \ldots, w_{n^2}]$$

and

$$[u_{n^2+1}, u_{n^2+2}, \ldots, u_{2n^2}] \equiv \frac{d}{dt}[w_1, w_2, \ldots, w_{n^2}]$$

We now have a system of $2n^2$ first–order differential equations which we write as

$$\dot{u} = G(u, t) \tag{16}$$

(Note that the first n^2 of these equations are particularly simple, being

$$\dot{u}_i = u_{n^2+i} \qquad i = 1, \ldots n^2.)$$

For any reasonable value of n, (16) is a high–dimensional system (of dimension $2n^2$) which is impractical to study in any depth. This is the motivation behind trying to use some of the ideas relating to approximate inertial manifolds that have been developed for infinite–dimensional PDEs to help us in our study of (16).

We are trying to make an analogy between (16) and a generic PDE (1). Instead of using the eigenfunctions of L as a basis for the space of solutions, we use the numerically determined eigenvectors of the linearisation of (16) about the rest state (the origin). To do this we write (16) as

$$\dot{u} = Bu + H(u,t)$$

where B is the spatial Jacobian of $G(u,t)$ evaluated at $u = 0$ and $H(u,t) = G(u,t) - Bu$. For large enough n, the eigenvalues of B come in a mixture of complex conjugate pairs and simple reals which we order according to their real parts, for example

$$\{-\rho_1 \pm i\omega_1, -\rho_2, \ldots, -\rho_k \pm i\omega_k, \ldots, -\rho_m, \ldots, -\rho_\eta\} \tag{17}$$

where $0 < \rho_1 \leq \rho_2 \leq \ldots \leq \rho_\eta$ and $\rho_1 < \rho_\eta$, i.e. not all the ρ_js are equal, and $n^2 \leq \eta < 2n^2$. It is the presence of the visco–elastic damping (the term in (7) with the coefficient β_2) that provides this ordering of the real parts of the eigenvalues, as with only linear damping (the term in (7) with the coefficient β) all of the real parts of the eigenvalues are equal (see [3] for more details). For small n all eigenvalues are complex. The eigenvectors (once converted back to displacements of the shell) correspond to modes of vibration and are similar in appearance to the eigenfunctions that would be used in a spectral Galerkin analysis of (7)-(8), *viz.*

$$w(x,y) = \sin(a\pi x)\sin(b\pi y) \tag{18}$$

where $a, b \in \mathbf{Z}^+$. Note that the eigenvalues of the linearisation of (7)-(8) about the rest state corresponding to the eigenfunction (18) are

$$\lambda(a,b) = \frac{\xi(a,b) \pm i\sqrt{\Omega^2(a,b) - \xi^2(a,b)}}{2} \tag{19}$$

where

$$\xi(a,b) = \frac{\beta + \beta_2\pi^4(a^2+b^2)^2}{\mu} \quad \text{and} \quad \Omega(a,b) = \frac{2\pi^2(a^2+b^2)\sqrt{D}}{\sqrt{\mu}}$$

The eigenvalues given by (19) with real part closest to zero are shown in Figure 3 for $n = 9$ and other parameters as in Section 5. These agree well with the numerically determined eigenvalues of B for the same parameter values.

We want to split our variable u into a slowing contracting component and a quickly contracting component in the same way that we did for the PDE where we projected onto the space spanned by the first few modes and then onto the complement of that space. This is simply done by performing a linear coordinate change on u so that the Jacobian of the resulting dynamical system is block diagonal. Let the (complex) eigenvector of B corresponding to the eigenvalue $-\rho_j + i\omega_j$ be z_j and the real eigenvector of B corresponding to the real eigenvalue $-\rho_k$ be z_k and form the matrix

$$Z \equiv \left(\; Re(z_1) \;\middle|\; Im(z_1) \;\middle|\; z_2 \;\middle|\; \cdots \;\middle|\; Re(z_k) \;\middle|\; Im(z_k) \;\middle|\; \cdots \;\middle|\; z_m \;\middle|\; \cdots \;\middle|\; z_\eta \; \right)$$

using the ordering and structure given by (17). Defining the variable v by $u = Zv$ we have

$$\dot{v} = Z^{-1}\dot{u} = (Z^{-1}BZ)v + Z^{-1}H(Zv,t) \equiv Cv + \Gamma(v,t) \tag{20}$$

where

$$C = \begin{pmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_\eta \end{pmatrix}, \tag{21}$$

and the C_js are either 2×2 blocks:

$$C_j = \begin{pmatrix} -\rho_j & \omega_j \\ -\omega_j & -\rho_j \end{pmatrix},$$

corresponding to the complex eigenvalues $-\rho_j \pm i\omega_j$, or

$$C_k = -\rho_k$$

corresponding to the real eigenvalue $-\rho_k$, and the "0"s in (21) represent zero matrices of the appropriate size, i.e. C is block diagonal.

To make notation easier we introduce a function M, defined on a subset of the integers, which contains information about whether eigenvalues of B are real or complex. It is defined iteratively as

$$M(j) = M(j-1) + m(j), \qquad M(0) = 0,$$

for $j = 1, \ldots, \eta$, where $m(j)$ is the dimensionality of the eigenspace corresponding to the eigenvalues represented by C_j. With this definition, the dynamics obtained by restricting (20) to the first j modes is given by the time evolution of the first $M(j)$ components of v. Note that $M(\eta) = 2n^2$.

By analogy with a PDE, we say that a k–mode Galerkin truncation of (20) is the system that results when we set $v_{M(k)+1}, \ldots, v_{M(\eta)}$ to zero, leaving an $M(k)$–dimensional system of ODEs. To allow us to use the ideas of approximate inertial manifolds, let us write

$$y_k = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{M(k)} \end{pmatrix} \quad \text{and} \quad y_{k,m} = \begin{pmatrix} v_{M(k)+1} \\ v_{M(k)+2} \\ \vdots \\ v_{M(m)} \end{pmatrix}$$

so that

$$v = \begin{pmatrix} y_k \\ y_{k,m} \\ y_{m,\eta} \end{pmatrix}$$

Also, let us write

$$\Upsilon_k = \begin{pmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{M(k)} \end{pmatrix} \text{ and } \Upsilon_{k,m} = \begin{pmatrix} C_{M(k)+1} & 0 & \cdots & 0 \\ 0 & C_{M(k)+2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{M(m)} \end{pmatrix}$$

and let $\Gamma_k(v,t)$ be the first $M(k)$ components of $\Gamma(v,t)$ and $\Gamma_{k,m}(v,t)$ be the $[M(k)+1]$th to $[M(m)]$th components. Thus we can write (20) as

$$\begin{aligned} \dot{y}_k &= (\Upsilon_k)y_k + \Gamma_k([y_k; y_{k,m}; y_{m,\eta}], t) \\ \dot{y}_{k,m} &= (\Upsilon_{k,m})y_{k,m} + \Gamma_{k,m}([y_k; y_{k,m}; y_{m,\eta}], t) \\ \dot{y}_{m,\eta} &= (\Upsilon_{m,\eta})y_{m,\eta} + \Gamma_{m,\eta}([y_k; y_{k,m}; y_{m,\eta}], t) \end{aligned}$$

and a k–mode Galerkin truncation of (20) is the system

$$\dot{y}_k = (\Upsilon_k)y_k + \Gamma_k([y_k; 0; 0], t) \tag{22}$$

We make an analogy with the PDE by assuming that there is a graph relating the asymptotic behaviour of the higher modes ($y_{k,m}$ for $k < m \leq \eta$) to the behaviour of the lower modes (y_k), i.e. there exists a $\Phi_{k,m}$ such that

$$y_{k,m} = \Phi_{k,m}(y_k)$$

Again, in analogy with the construction for the PDE, we take a first approximation to $\Phi_{k,m}$, $\Phi^1_{k,m}$ defined as

$$y_{k,m} = \Phi^1_{k,m}(y_k) \equiv -(\Upsilon_{k,m})^{-1}\Gamma_{k,m}([y_k;0;0],t) \tag{23}$$

Thus the evolution on the approximate inertial manifold is given by the system

$$\dot{y}_k = (\Upsilon_k)y_k + \Gamma_k([y_k; -(\Upsilon_{k,m})^{-1}\Gamma_{k,m}([y_k;0;0],t);0],t) \tag{24}$$

with the solution reconstructed as $v \approx y_k + \Phi^1_{k,m}(y_k)$. The post–processing approach consists of solving (22) and then reconstructing the solution as $v \approx y_k + \Phi^1_{k,m}(y_k)$. Note that since $\Upsilon_{k,m}$ is block diagonal, it is trivial to invert.

Two advantages of using a finite–difference approach as opposed to a spectral method are that for finite–difference, the domain need not be regular and also that the eigenfunctions of the differential operator L need not be found analytically. The "eigenfunctions" are instead the eigenvectors of B (the z_js) which are determined numerically from the semi–discrete system (16).

Note that the construction above is very similar to the construction of a centre manifold for an ODE — see, e.g. [21]. Also, the idea of constructing an inertial manifold for a large set of ODEs derived from a finite difference scheme is not new and was discussed in, e.g. [6].

5 Numerical results

In this section we show some numerical results regarding the finite difference model discussed above. In Figure 4 we compare the accuracy of the three methods — flat Galerkin (FG), post–processed Galerkin (PPG) and the nonlinear Galerkin method (NLG). We have chosen $n = 9$ in this example, with forcing at node (2,1) with $\mu f(t) = 10000 \sin(800t)$. Other constants have the values

μ	β	β_2	ν	E	h	R
78.5 kg/m^2	7071 Ns/m	$\beta/(16\pi^4)$	0.3	2.1×10^{11} N/m^2	0.01 m	8.333 m

(μ, ν and E are the relevant values for steel). We have compared the solutions at a time of 0.1 seconds (corresponding to approximately 13 forcing cycles), having started from rest. Integration has been done with the Matlab [17] (Version 5) routine `ode23` with default settings. For the post–processing and the calculations on the approximate inertial manifold (AIM) we have "lifted" the solution up onto the all the remaining modes, i.e. we have taken $m = \eta$ in (24) for all values of k. The results of Garcia-Archilla et al. [7] suggest taking $m = k^{\alpha}$ for some $1 < \alpha < 2$, with the exact value depending on the PDE being studied, but for these calculations we chose $m = \eta$. Note that we have not shown results from all possible values of k in Figure 4, only the lowest ones, as for higher values we no longer have enough modes to lift up to and scaling results for convergence rates are no longer likely to be valid.

In Figure 4 we have defined the error as the L_2 norm in space of the difference between the solution at t=0.1s calculated using all the modes and the solution at t=0.1s calculated (and then possibly post–processed) using a smaller number of modes. The theory of Jones et al. [10] shows that the error should be inversely proportional to the real part of the kth eigenvalue, where k is the level of truncation, and hence a log–log plot of error as a function of the real part of the eigenvalues of B should produce a straight line. The main point to notice is that the post–processed solution has a similar accuracy as the calculations on the AIM, and that these are significantly better than a flat Galerkin. Note also the different slopes of the best–fit straight lines. The results of Jones et al. [10] suggest that for solutions with low smoothness (such as those generated by our point forcing), the NLG method converges faster than the FG (as well as having better accuracy for a given number of modes). This is confirmed in Figure 4, where the slopes of best–fit lines for the PPG and NLG are -1.0199 and -1.0056, respectively, compared with -0.7111 for the FG.

We now give a simple example demonstrating how the post–processing works. Figure 5 a. shows the displacement of the shell 9.25 periods after starting from rest, where the forcing is $\mu f(t) = 10000 \sin(800t)$ at node (2,2) and $n = 4$. For $n = 4$, $\eta = 16$, i.e. there are 16 distinct modes, all of which have complex eigenvalues associated with them. The 16 eigenvectors are similar to the 16 spectral Galerkin modes $\sin(\pi x)\sin(\pi y), \ldots, \sin(4\pi x)\sin(4\pi y)$. We have chosen to integrate on the first 6 modes and post–process the remaining 10.

Figure 5 b. shows the difference between the "true" solution in a. and the result of integrating on the first 6 modes (i.e. a flat Galerkin projection). This Figure can be thought of as showing the "fine structure" that we have lost by using only the first 6 modes. Figure 5 c. shows the difference between the solution obtained by post–processing the data from the integration on the first 6 modes and that obtained by using a 6 mode truncation and can be thought of as showing the fine structure that has been recovered using the post–processing. Note the similarity with b., and that b. and c. have the same scale.

In Figure 6 we show a "spectral analysis" of the full solution, the solution obtained through a 6 mode truncation, and the post–processed solution. Assuming that the plate is on the unit square and the displacement at the grid points is given by z, the quantities we have calculated are

$$\theta_{i,j} \equiv \frac{|\texttt{trapz}\{\texttt{trapz}\{z \sin(i\pi x)\sin(j\pi y)\}\}|}{\texttt{trapz}\{\texttt{trapz}\{\sin^2(i\pi x)\sin^2(j\pi y)\}\}}$$

where the arguments of the `trapz` are evaluated at the grid points calculated (not the ones introduced by the interpolation) and `trapz` is the Matlab [17] function for approximate integration using the trapezoidal method. We can think of $\theta_{i,j}$ as giving the amount of the mode $\sin(i\pi x)\sin(j\pi y)$ in z.

Figure 6 a. shows θ for the full solution from Figure 5 a. Figure 6 b. shows θ for the solution from the 6 mode truncation. Note that θ is very small (although non–zero) away from the first 6 modes. Figure 6 c. shows θ for the post–processed solution. Note the similarity with a.

Figure 7 shows this information more clearly. Here we have shown the ratios of corresponding values of θ for the 6 mode truncation and the true solution (a.), and the post–processed values and the true solution (b.). We see that in a. the ratios for the first 6 modes are close to 1 while the other ratios are close to 0, whereas in b. all ratios are close to 1.

6 Extraction of the reduced system

The issue of explicitly extracting the Galerkin truncation (22) and the approximate inertial manifold (23) from the full system (16) arises. In principle, equation (20) can be calculated symbolically using a package such as Maple [16] and equations (22) and (23) easily extracted, but we have found that this approach rapidly uses up all available RAM on a moderate sized work–station.

A more practical approach is to calculate (20) numerically and then use one of two methods to extract (22) and (23) from it. Both methods rely on knowing the functional form of (22) and (23) — for the von Karman equations they are both cubic polynomials in the components of y_k and linear in $f(t)$. For our first method, let us set $k = 1$, assume that $M(1) = 2$ and that we are trying to find (22). Write

$$y_1 = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

We know that the first component of (22) can be written

$$\dot{u}_1 = \sum_{i=1}^{2} a_i u_i + \sum_{i=1}^{2}\sum_{j=i}^{2} b_{i,j} u_i u_j + \sum_{i=1}^{2}\sum_{j=i}^{2}\sum_{k=j}^{2} c_{i,j,k} u_i u_j u_k + A\sin(\omega t) \qquad (25)$$

There are $P \equiv [M(k)]^3/6 + [M(k)]^2 + 11M(k)/6$ unknown coefficients $a_1, \ldots, c_{2,2,2}$ in (25), so if we choose P random $M(k)$–tuples $[u_1(s) \cdots u_{M(k)}(s)]$ for $s = 1, \ldots, P$ and create a matrix, H (for $P = 9$ in this case) which can be thought of as a generalisation of a Vandermonde matrix [19]

$$\begin{pmatrix} u_1(1) & u_2(1) & u_1^2(1) & u_1(1)u_2(1) & u_2^2(1) & u_1^3(1) & u_1^2(1)u_2(1) & u_1(1)u_2^2(1) & u_2^3(1) \\ u_1(2) & u_2(2) & u_1^2(2) & u_1(2)u_2(2) & u_2^2(2) & u_1^3(2) & u_1^2(2)u_2(2) & u_1(2)u_2^2(2) & u_2^3(2) \\ \vdots & & & & \vdots & & & & \vdots \\ u_1(P) & u_2(P) & u_1^2(P) & u_1(P)u_2(P) & u_2^2(P) & u_1^3(P) & u_1^2(P)u_2(P) & u_1(P)u_2^2(P) & u_2^3(P) \end{pmatrix}$$

and a vector

$$X = \begin{pmatrix} \dot{u}_1([u(1);0;0],0) \\ \dot{u}_1([u(2);0;0],0) \\ \vdots \\ \dot{u}_1([u(P);0;0],0) \end{pmatrix}$$

where $\dot{u}_1$ is the first component of (22) as calculated numerically via (20), then the vector of coefficients

$$\begin{pmatrix} a_1 \\ a_2 \\ b_{1,1} \\ b_{1,2} \\ b_{2,2} \\ c_{1,1,1} \\ c_{1,1,2} \\ c_{1,2,2} \\ c_{2,2,2} \end{pmatrix}$$

is given by $H^{-1}X$, provided H is invertible (generically, it will be, as the $M(k)$–tuples are chosen randomly). A is determined by evaluating $\dot{u}_1([0;0;0],\pi/(2\omega))$. This procedure can then be repeated for each component of (22). The disadvantage of this method is that the side length of H goes as $[M(k)]^3$ and inverting H becomes impractical for $M(k)$ greater than about 20.

The other method involves numerical differentiation where we use the approximate formulae

$$\begin{aligned}
\frac{\partial f([x_1\cdots x_n])}{\partial x_i} &\approx \frac{f([0\ \overset{i}{\delta}\ 0])}{\delta} \\
\frac{\partial^2 f([x_1\cdots x_n])}{\partial x_i\partial x_j} &\approx \frac{f([0\ \overset{i}{\delta}\ 0\ \overset{j}{\delta}\ 0])-f([0\ \overset{i}{\delta}\ 0])-f([0\ \overset{j}{\delta}\ 0])}{\delta^2} \qquad i\neq j \\
\frac{\partial^2 f([x_1\cdots x_n])}{\partial^2 x_i} &\approx \frac{f([0\ \overset{i}{2\delta}\ 0])-2f([0\ \overset{i}{\delta}\ 0])}{2\delta^2}
\end{aligned}$$

and similarly for the third partial derivatives, where δ is small and $[0\ \overset{i}{\delta}\ 0\ \overset{j}{\delta}\ 0]$ is a vector with n entries, all of which are zero except for ith and jth, which are equal to δ. These approximate formulae can be used to isolate the coefficients $a_1,\ldots,c_{2,2,2}$ in (25), while the technique mentioned above can be used to extract the values of A. This method has the advantage that it does not use large amounts of RAM, but it does require more calls to (16) than the first method discussed.

7 Efficiency

In this section we compare the efficiency of the three different methods. In Figure 8 we show a log–log plot of the error (defined in the same way as in section 5) as a function of the number of floating point operations required to integrate the reduced system of equations for 0.1 seconds starting from rest for the three different schemes. We used the Matlab [17] (Version 5) procedure `ode45` with default settings for the integration and the forcing was again $10000\sin(800t)$, but this time at node (7,2). The 10 data points for the flat Galerkin method correspond to the first 10 modes.

While it is hard to discern a clear difference in the efficiencies of the flat Galerkin as opposed to the calculations on the AIM, it is clear that there is a difference between the calculations on the AIM and the post–processed solution, the post–processing providing greater accuracy for a given amount of CPU time. This is in agreement with results of other workers using spectral methods [7].

8 Conclusion

We have demonstrated an application of the ideas of post–processed Galerkin methods to the large set of ordinary differential equations resulting from a semi–discrete finite difference approximation to the von Karman equations that govern the vibrations of thin shells. We have shown that post–processing gives results of similar accuracy to those obtained from a nonlinear Galerkin method yet only requires a similar amount of computational power to that used for a flat Galerkin of lower order and is thus more efficient than both the flat Galerkin and nonlinear Galerkin methods. We believe that post–processing of this form is a useful technique and will be of use in the many areas for which the numerical solution of partial differential equations is necessary.

Acknowledgements

We would like to thank Professor Edriss Titi for telling us about post–processed Galerkin methods, and the referees for their useful comments. This work was supported by the EPSRC Applied Nonlinear Mathematics Initiative.

References

[1] Baumgarten, R., Kreuzer, E. and Popov, A. A. (1997) A Bifurcation Analysis of the Dynamics of a Simplified Shell Model. *Nonlinear Dynamics* **12**, 307.

[2] Devulder, C., Marion, M. and Titi E. S. (1993) On the rate of convergence of the nonlinear Galerkin methods. *Mathematics of Computation* **60**, 495.

[3] Foale, S., McRobie, F. A. and Thompson, J. M. T. (1998) Numerical dimension–reduction methods for nonlinear shell vibrations. Accepted by *Journal of Sound and Vibration.*

[4] Foias, C., Jolly, M. S., Kevrekidis, I. G., Sell, G.R. and Titi, E. S. (1988) On the computation of inertial manifolds. *Physics Letters A* **131**(7,8), 433.

[5] Foias, C., Manley, O. and Temam, R. (1988) Modelling of the interaction of small and large eddies in two dimensional turbulent flows. *Modélisation Mathématique et Analyse Numérique* **22**(1) 93.

[6] Foias, C. and Titi, E. S. (1991) Determining nodes, finite difference schemes and inertial manifolds. *Nonlinearity* **4** 135.

[7] Garcia-Archilla, B., Novo, J. and Titi, E. S. (1998) Postprocessing the Galerkin Method: a Novel Approach to Approximate Inertial Manifolds. *SIAM Journal on Numerical Analysis* **35**(3) 941.

[8] Garcia-Archilla, B., Novo, J. and Titi, E. S. An Approximate Inertial Manifolds Approach to Postprocessing the Galerkin Method for the Navier–Stokes equations. Submitted to *Mathematics of Computation.*

[9] Graham, M. D., Steen, P.H. and Titi, E.S. (1993) Computational Efficiency and Approximate Inertial Manifolds for a Bénard Convection System. *Journal of Nonlinear Science* **3**(2) 152.

[10] Jones, D. A., Margolin, L. G. and Titi, E. S. (1995) On the Effectiveness of the Approximate Inertial Manifold – a Computational Study. *Theoretical and Computational Fluid Dynamics* **7**(4) 243.

[11] Lord, G. J., Champneys, A. R. and Hunt, G. W. The Role of Homoclinic Bifurcations in Understanding the Buckling of Long Thin Cylindrical Shells. Submitted to *IUTAM Chaos '97 Conference Proceedings.*

[12] Lord, G. J., Champneys, A. R. and Hunt, G. W. Computation of homoclinic orbits in Partial Differential Equations: an application to cylindrical shell buckling. Submitted to *SIAM Journal on Scientific Computation.*

[13] Marion, M. and Temam, R. (1989) Nonlinear Galerkin methods. *SIAM Journal on Numerical Analysis* **26**(5) 1139.

[14] Marion, M. and Temam, R. (1990) Nonlinear Galerkin methods: the finite elements case. *Numerische Mathematik* **57**(3) 205.

[15] Marion, M. and Xu, J. (1995) Error estimates on a new nonlinear Galerkin method based on two–grid finite elements. *SIAM Journal on Numerical Analysis* **32**(4) 1170.

[16] Waterloo Maple Inc. 450 Phillip Street, Waterloo, Ontario, Canada N2L 5J2. `http://www.maplesoft.on.ca/`.

[17] The MathWorks, Inc. 24 Prime Park Way Natick, MA 01760-1500, USA. `http://www.mathworks.com/`

[18] Popov, A. A., Thompson, J. M. T. and McRobie, F. A. (1998) Low dimensional models of shell vibrations. Parametrically excited vibrations of cylindrical shells. *Journal of Sound and Vibration* **209**(1) 163.

[19] Press, W. H., Tuekolsky, S.A., Vettering, W.T. and Flannery, B.P. (1992) *Numerical Recipes in C.* (2nd ed.) Cambridge University Press.

[20] Temam, R. (1990) Inertial manifolds and multigrid methods. *SIAM Journal on Mathematical Analysis.* **21**(1) 154.

[21] Wiggins, S. (1990). *Introduction to Applied Nonlinear Dynamical Systems and Chaos.* Texts in Applied Mathematics **2**. Springer–Verlag, New York.

List of Figures

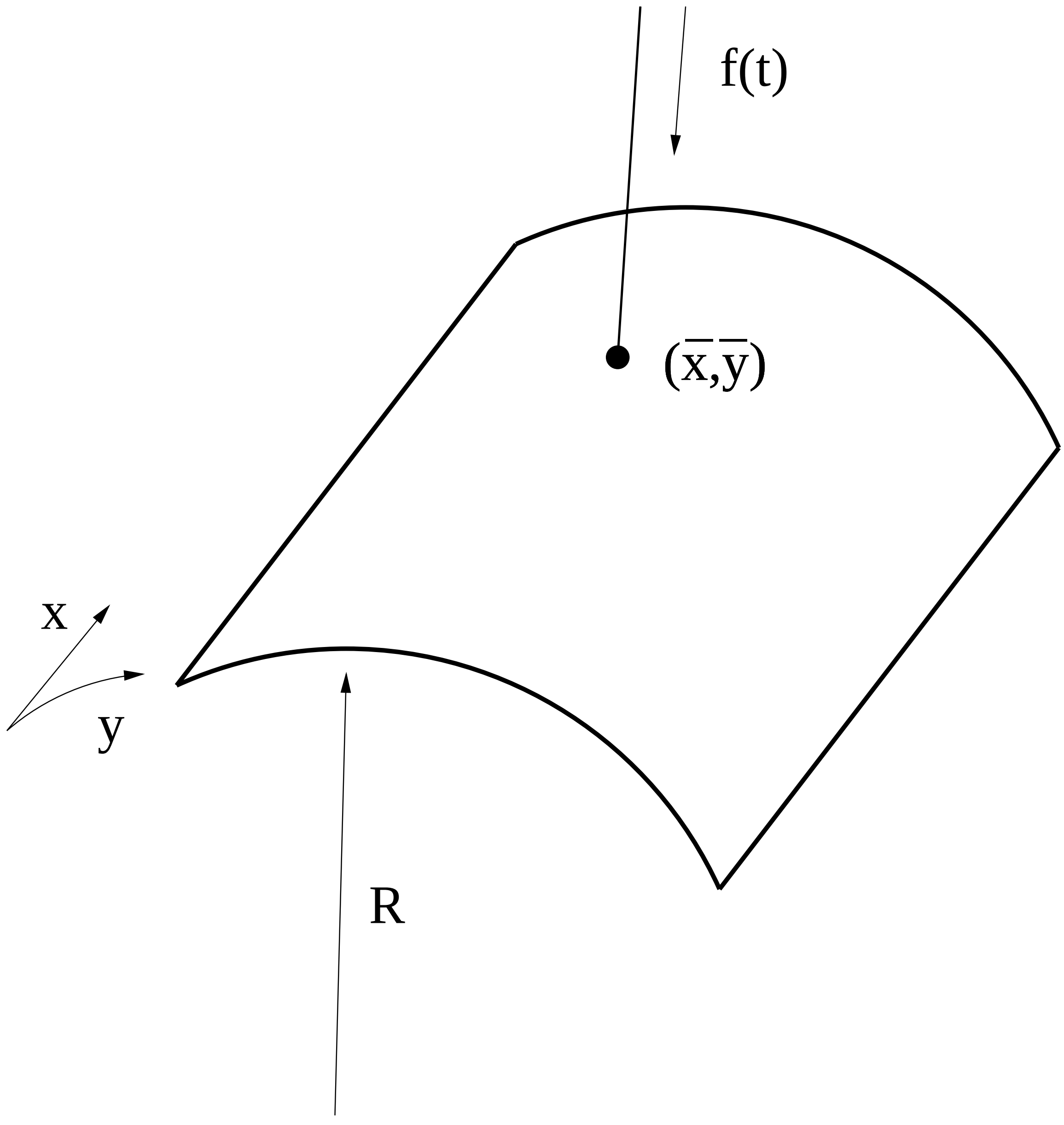

Figure 1: Schematic diagram of cylindrical panel with point forcing of $f(t)$. All side lengths are assumed to be equal.

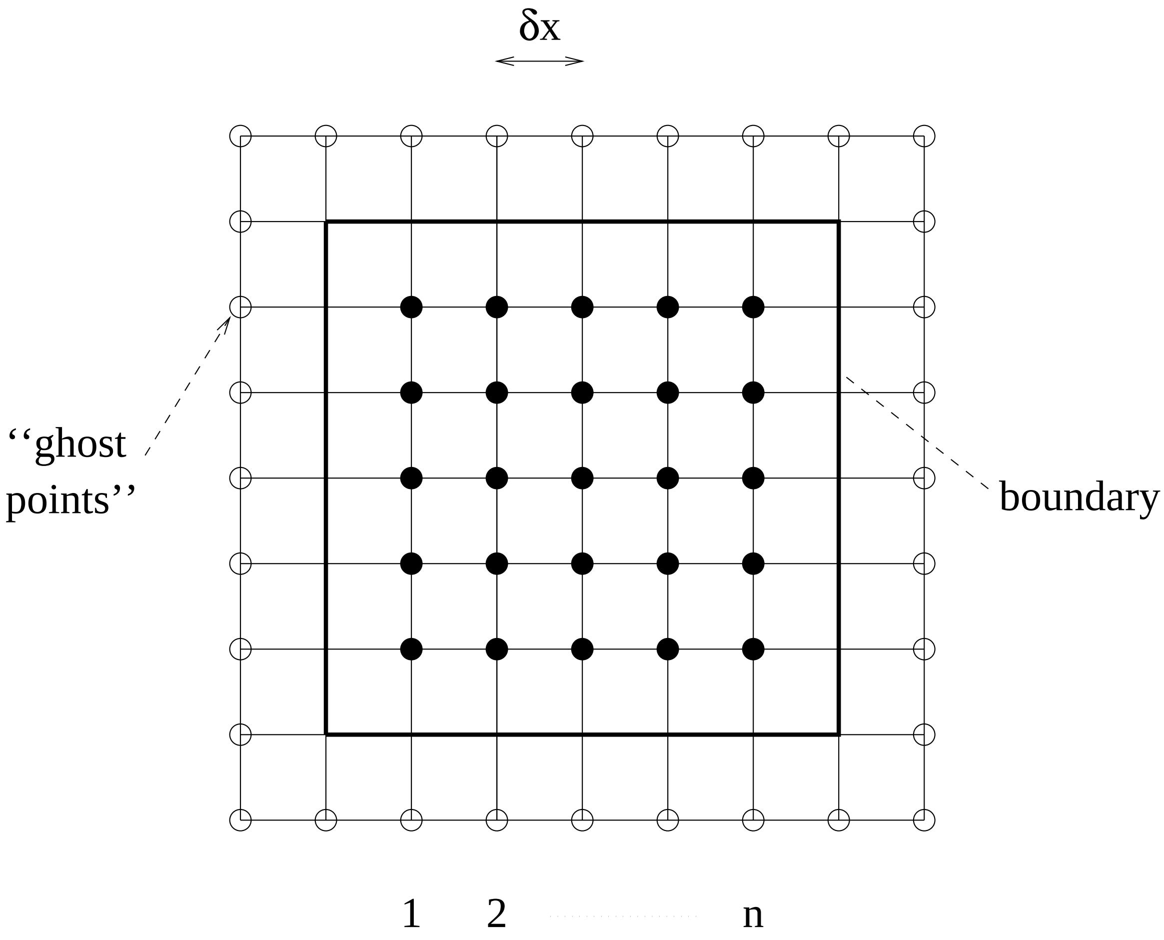

Figure 2: Schematic diagram of the cartesian grid we lay over the shell.

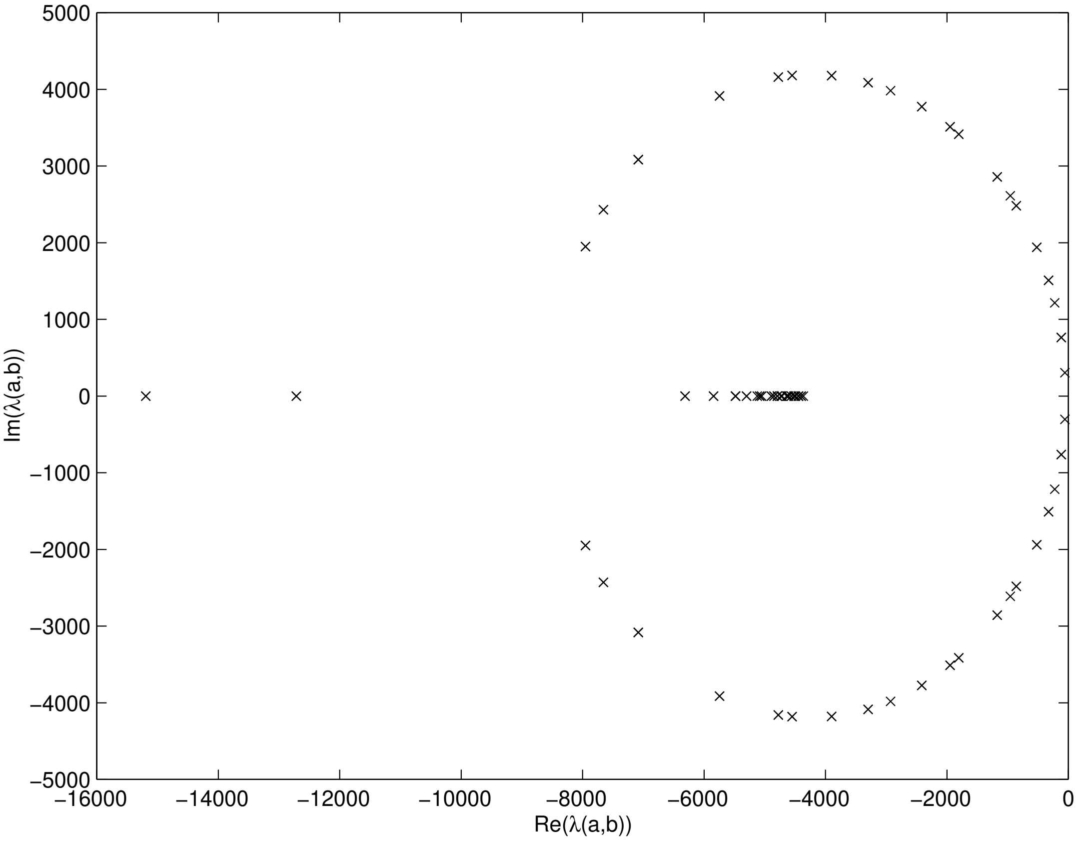

Figure 3: Partial set of eigenvalues (19) of the linearisation of the von Karman equations (7)-(8) about the rest state for $n = 9$; other parameters are as in Section 5. There are more real eigenvalues that are more negative than those shown.

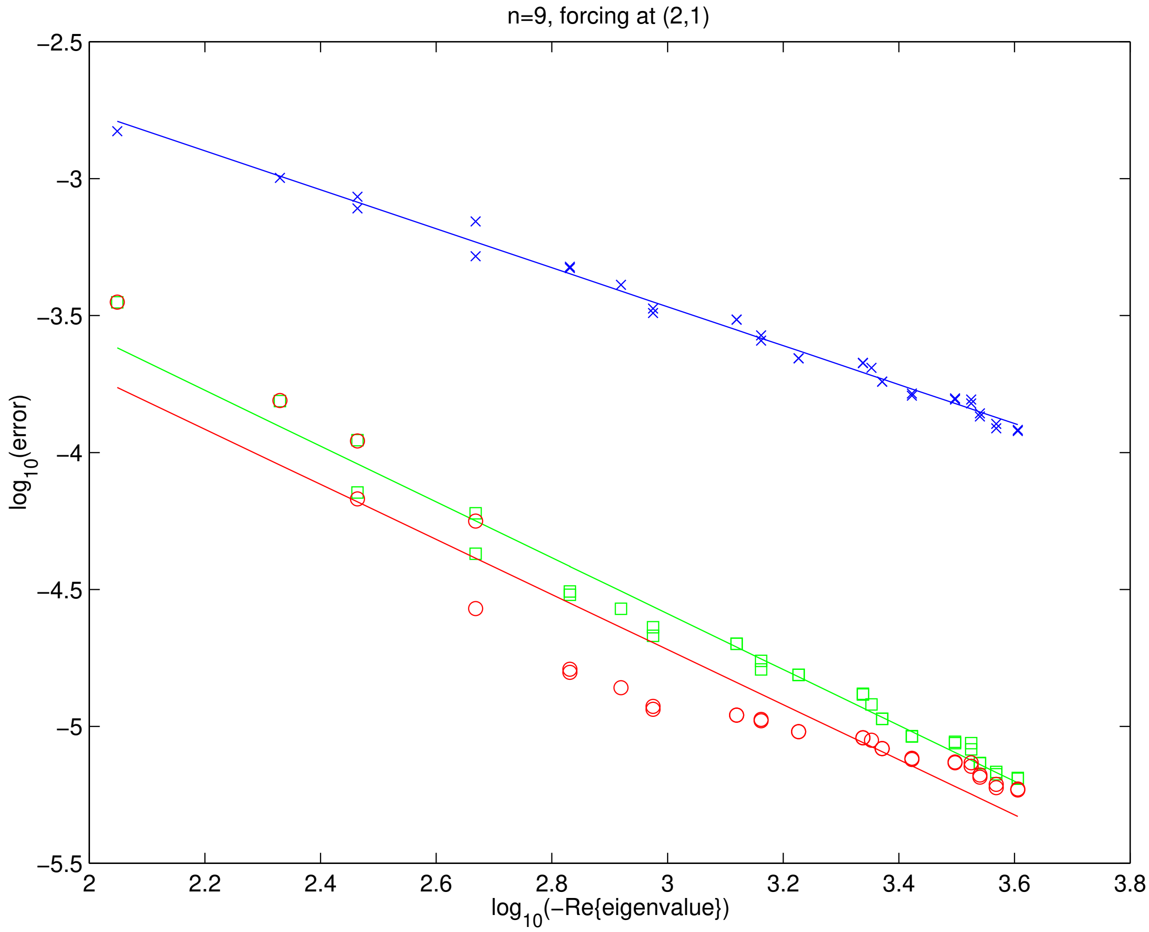

Figure 4: A log–log plot of the error (L_2 norm in space at t=0.1s) as a function of the real part of the eigenvalues of B for a system with $n = 9$ and forcing of $\mu f(t) = 10000 \sin(800t)$ at node (2,1). The crosses are a flat Galerkin truncation, squares are post–processed Galerkin and circles are calculations on the inertial manifold. The straight lines are least squares fits and have slopes -0.7111, -1.0199 and -1.0056, respectively.

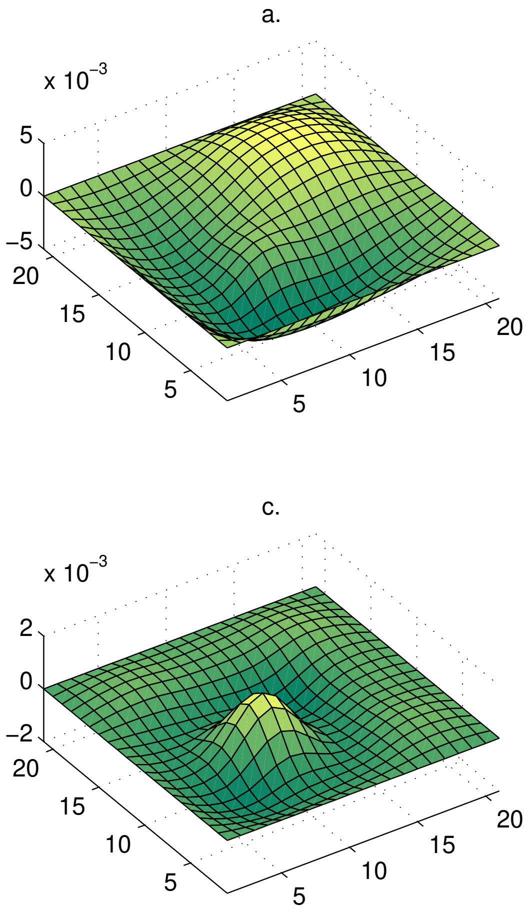

Figure 5: a.) Shell at $t = 9.25$ periods after starting from rest. Forcing is at node (2,2) and $n = 4$. b.) Difference between solution in a. and that calculated using a flat 6 mode truncation. c.) Difference between post–processed solution integrated on 6 modes and lifted to all remaining modes and that calculated using a flat 6 mode truncation, using same scale as b. (The surfaces are generated by two passes of the Matlab [17] function `interp2` with bicubic interpolation on the original data to aid visualisation.)

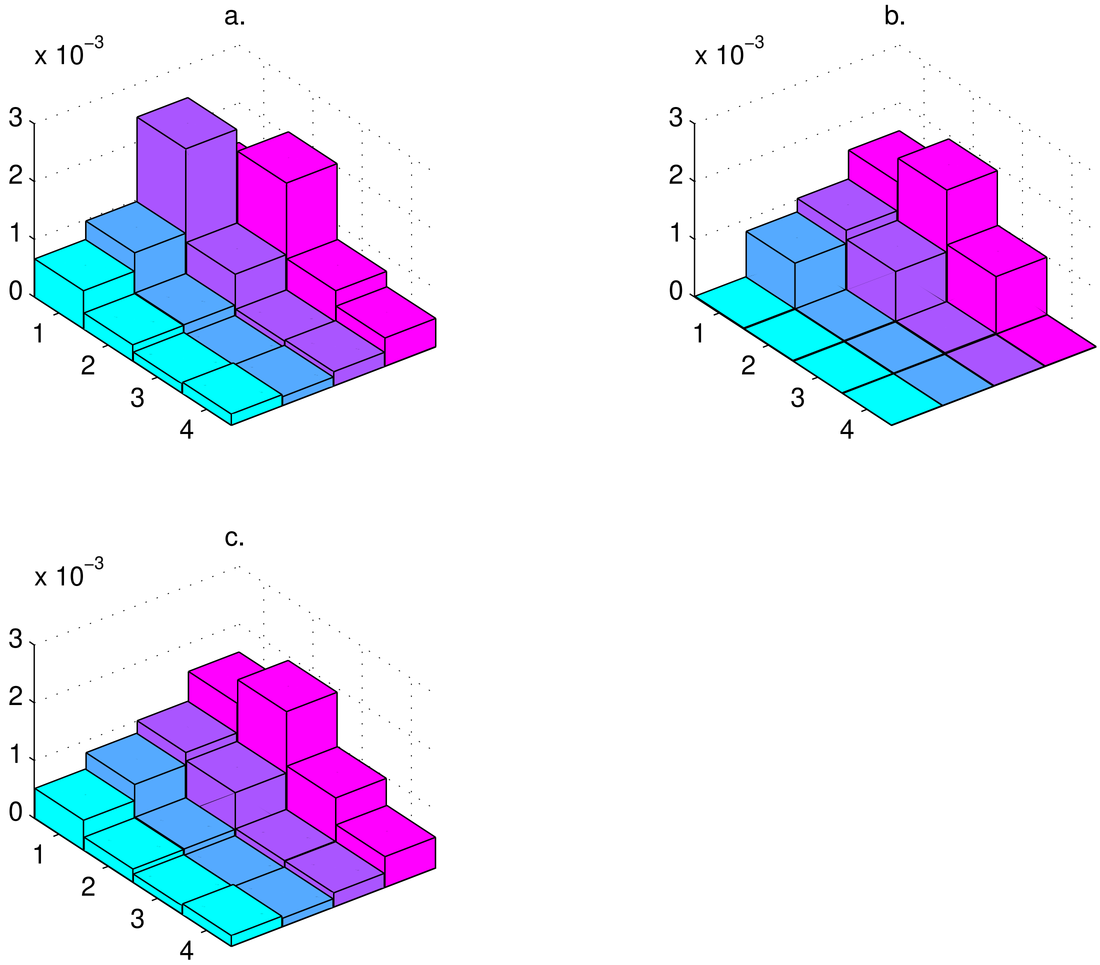

Figure 6: Spectral analysis of a.) full calculation, shown in Figure 5 a., b.) 6 mode truncation, c.) post–processed solution integrated on 6 modes and lifed to all remaining modes. The (1,1) mode is furthest away and the (4,4) mode closest.

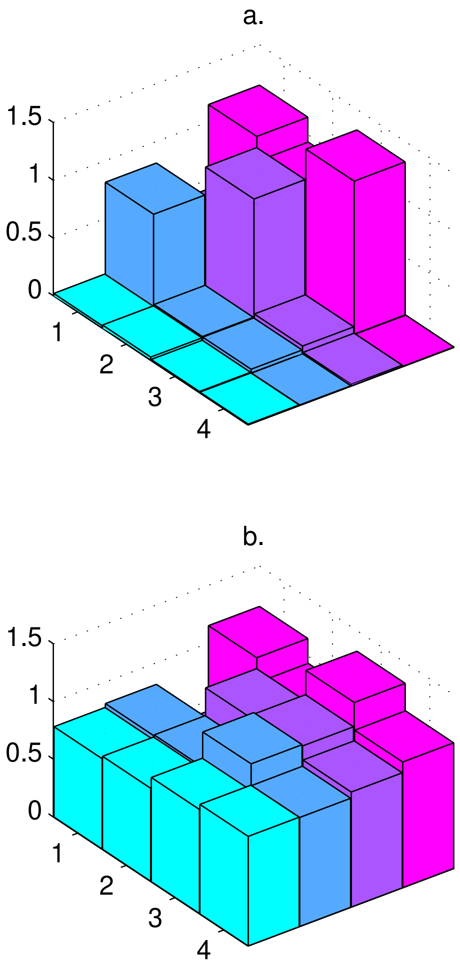

Figure 7: Ratios of spectral components in Figure 6. a. shows the ratio b:a in Figure 6, i.e. 6 mode:full calculation, while b. shows the ratio c:a in Figure 6, i.e. post–processed from 6 modes to all remaining:full calculation.

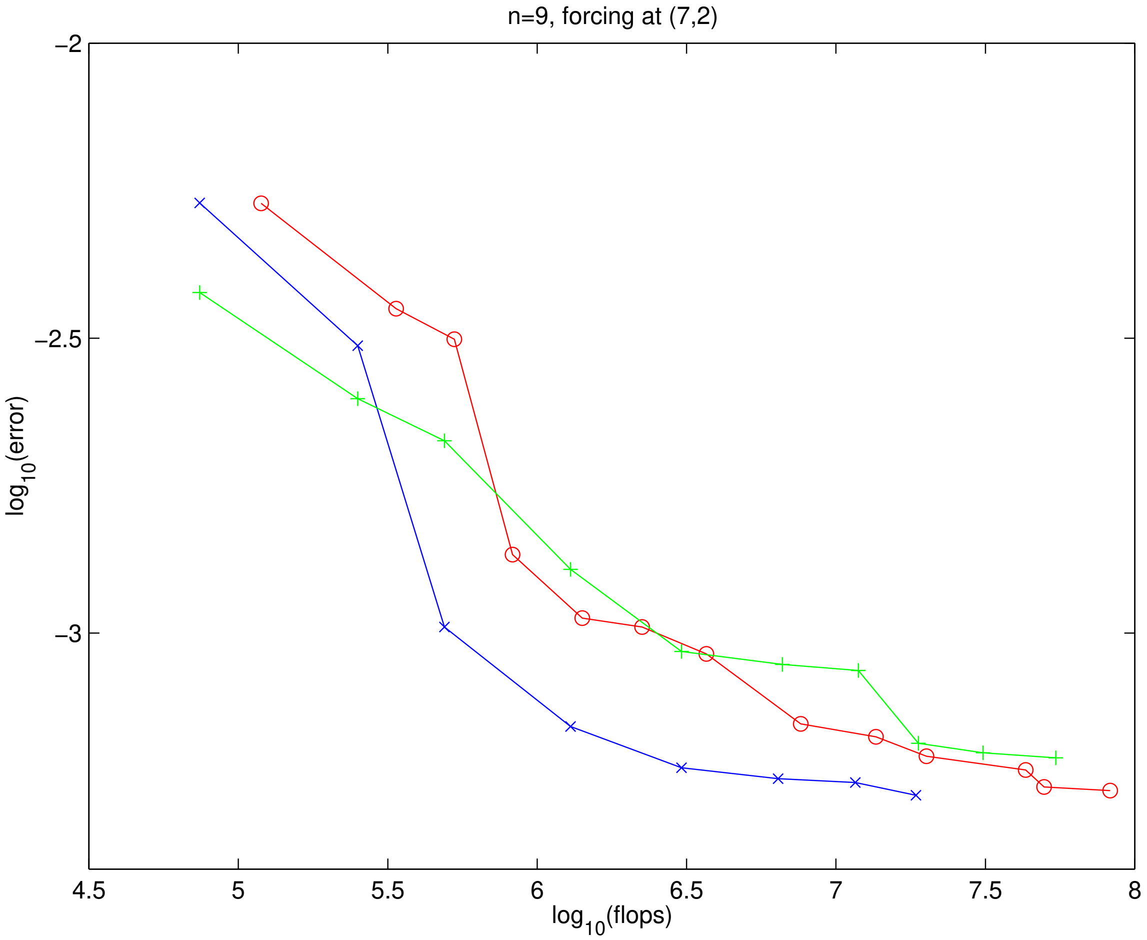

Figure 8: Log–log plot of the error as a function of the number of floating point operations needed to integrate the reduced system of equations for a time of 0.1s. "+" is the flat Galerkin, "o" is for integration on the AIM, while "x" refers to the post–processed case.