Computing the fundamental distortion mode in Coriolis mass flow meters

Robert Cheesewright and Simon Shaw

April 24, 2006

Contents

1 Overview

This technical report details and extends the numerical computations that appear in [4]. That reference contains more background and motivation for the study than appears below.

Coriolis mass flow meters represent an applied mathematics problem in the general area of fluid-structure interaction. A basic configuration is that of a metal tube with fluid flowing through it at velocity V. The tube is electromagnetically vibrated and the resulting Coriolis accelerations which influence the displacement can be used to determine the mass flow.

For an Euler-Bernoulli beam the basic problem is described by the partial differential equation (PDE),

$$(m_p + m_f)u_{tt} + EI_p u_{xxxx} + m_f\Big[2Vu_{xt} + V^2 u_{xx}\Big] = f,$$

in a domain $Q := \Omega \times I$ where $\Omega := (\check{x}, \hat{x})$ represents the beam and $I := (0, T)$ is the time interval of interest. On the other hand, for a Timoshenko beam we have a pair,

$$(m_p + m_f)u_{tt} + m_f\Big[2Vu_{xt} + V^2u_{xx}\Big] - \kappa GA_p\big(u_{xx} - \theta_x\big) = f_0,$$

$$(\varrho_p I_p + \varrho_f I_f)\theta_{tt} - EI_p\theta_{xx} - \kappa GA_p\big(u_x - \theta\big) = f_1,$$

also holding in Q. In these the forces, f, f_0 and f_1 are usually zero.

For ease of software implementation, we note that the Timoshenko equations can be written more succinctly as,

$$\frac{\partial}{\partial t}\underbrace{\begin{pmatrix} m_p + m_f & 0 \\ 0 & \varrho_p I_p + \varrho_f I_f \end{pmatrix}}_{\varrho}\begin{pmatrix} u_t \\ \theta_t \end{pmatrix} - \frac{\partial}{\partial x}\underbrace{\begin{pmatrix} \kappa GA_p - m_f V^2 & 0 \\ 0 & EI_p \end{pmatrix}}_{\sigma}\begin{pmatrix} u_x \\ \theta_x \end{pmatrix}$$

$$+ \frac{\partial}{\partial t}\underbrace{\begin{pmatrix} 2Vm_f & 0 \\ 0 & 0 \end{pmatrix}}_{\xi}\begin{pmatrix} u_x \\ \theta_x \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & \kappa GA_p \\ -\kappa GA_p & 0 \end{pmatrix}}_{\beta}\begin{pmatrix} u_x \\ \theta_x \end{pmatrix}$$

$$+ \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & \kappa GA_p \end{pmatrix}}_{\gamma}\begin{pmatrix} u \\ \theta \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \end{pmatrix},$$

although a little further thought is needed if the mechanically correct natural boundary conditions need to be formulated.

Each of these problems can be recognised as the standard beam equations from mechanics with additional terms, in the square brackets, introduced to model the fluid flow. The first such term, $2m_f V u_{xt}$, arises from the Coriolis forces and the second, $m_f V^2 u_{xx}$, comes from the centrifugal forces, see [3] for more background.

The purpose of this report is to describe some approaches related to computing the fundamental Coriolis distortion eigenmode. It turns out that this is computationally non-trivial (due, we think, to errors at machine precision) and this report describes both good and bad experiences.

The finite element systems referred to below are generated by a bespoke C/C++ code and are then dumped into Matlab `m`-files for the eigen analysis. We are using Matlab (7.1.0.246 (R14) SP3) because it is both simple and effective, and well suited to 'rapid prototyping'. We note also that, if desired, the entire numerical operation could be moved to Matlab using, for example, the codes in [1] as a starting point.

2 Initial and boundary conditions

In mathematical terms the correct boundary conditions are homogeneous and essential at each end of the beam. Mechanically, these correspond to zero displacement and rotation at each end.

Correct forms of the initial data have not yet been identified because so far we have been concerned only with the eigen-analysis.

3 Data

The coefficients in the PDEs above have the following meanings and values:

- κ: Timoshenko's shear correction factor (taken as unity).
- E: Young's modulus ($1.158 \times 10^{11}\mathrm{N/m^2}$).
- G: the shear modulus ($4.3209 \times 10^{10}\mathrm{N/m^2}$).
- V: the fluid velocity ($6\mathrm{m/s}$).
- A_p: cross-sectional area of the pipe ($7.001 \times 10^{-5}\mathrm{m^2}$).
- A_f: cross-sectional area of the fluid ($4.367 \times 10^{-4}\mathrm{m^2}$).
- m_p: the mass per unit length of pipe ($0.31555\mathrm{kg/m}$).
- m_f: the mass per unit length of fluid ($0.4367\mathrm{kg/m}$).
- ϱ_p: the volume density of the pipe ($4507\mathrm{kg/m^3}$).
- ϱ_f: the volume density of the fluid ($1000\mathrm{kg/m^3}$).
- I_p: the second area moment of the pipe ($5.256 \times 10^{-9}\mathrm{m^4}$).
- I_f: the polar second area moment of the pipe ($1.0512 \times 10^{-8}\mathrm{m^4}$).
- $\hat{x} - \check{x}$: the length of the pipe ($0.82\mathrm{m}$).

4 The eigenvalue discretisation

In the absence of loads, and after finite element discretisation, each PDE problem can be written as a system of ordinary differential equations:

$$\boldsymbol{M}\boldsymbol{U}_{tt} + \boldsymbol{E}\boldsymbol{U}_t + \boldsymbol{A}\boldsymbol{U} = \boldsymbol{0}$$

where:

- $\boldsymbol{M}$ is the mass matrix
- $\boldsymbol{A}$ is the stiffness matrix
- $\boldsymbol{E}$ is the velocity-damping matrix

The mass matrix is always positive definite and, after imposing cantilevered boundary conditions at both ends of the beam, the stiffness matrix will be also (if $m_f V^2$ is small enough).

The finite element formulation used here is quite standard. For the Euler-Bernoulli beam we have used two-noded C^1 cubic elements with degrees of freedom u and u_x at each end, while for the Timoshenko beam we have used each of:

- two-noded C^0 linear elements
- three-noded C^0 quadratic elements
- four-noded C^0 cubic elements

with degrees of freedom u and θ at each node. (Recall that the linear four degree-of-freedom element cannot usually be used for the Timoshenko beam due to its tendency to 'locking'. See for example [2].)

We now want to derive the eigen-problem associated with the Coriolis beam meter. For this, assume the form $\boldsymbol{U} = \boldsymbol{V}e^{i\omega t}$ so that $\boldsymbol{U}_t = i\omega\boldsymbol{V}e^{i\omega t}$ and $\boldsymbol{U}_{tt} = -\omega^2\boldsymbol{V}e^{i\omega t}$, and substitute in to get,

$$(\boldsymbol{A} + i\omega\boldsymbol{E} - \omega^2\boldsymbol{M})\boldsymbol{V} = \boldsymbol{0}. \tag{1}$$

We employ several methods to 'solve' this quadratic eigenvalue problem. Each uses matlab, and they are described below. Before we get to them we can deduce some properties of the eigensystem (1).

Let $\phi_1, \phi_2 \ldots$ be the shape functions associated with the finite element discretisation, and observe that the entries of the matrix $\boldsymbol{E}$, up to the coefficient $2m_f V$, are given by,

$$E_{ij} = \int_0^L \frac{d\phi_i}{dx}\phi_j \, dx.$$

Integrating by parts and using the zero boundary conditions then gives,

$$E_{ij} = \int_0^L \frac{d\phi_i}{dx}\phi_j \, dx = -\int_0^L \frac{d\phi_j}{dx}\phi_i \, dx = -E_{ji}.$$

Hence, $\boldsymbol{E} = -\boldsymbol{E}^T$.

Now, because $\boldsymbol{M}$ is symmetric and positive definite we always have $\boldsymbol{V}\cdot\boldsymbol{M}\boldsymbol{V} > 0$ for all real vectors $\boldsymbol{V}$. If $\boldsymbol{V}$ is complex then we can also claim that $\overline{\boldsymbol{V}}\cdot\boldsymbol{M}\boldsymbol{V} > 0$ for, if $\boldsymbol{V} = \boldsymbol{X} + i\boldsymbol{Y}$, then,

$$\overline{\boldsymbol{V}}\cdot\boldsymbol{M}\boldsymbol{V} = \boldsymbol{X}\cdot\boldsymbol{M}\boldsymbol{X} + \boldsymbol{Y}\cdot\boldsymbol{M}\boldsymbol{Y} > 0.$$

Clearly, $\overline{\boldsymbol{V}}\cdot\boldsymbol{A}\boldsymbol{V} > 0$ also and so taking the scalar product of (1) with $\overline{\boldsymbol{V}}$ we get,

$$\omega^2\overline{\boldsymbol{V}}\cdot\boldsymbol{M}\boldsymbol{V} - i\omega\overline{\boldsymbol{V}}\cdot\boldsymbol{E}\boldsymbol{V} - \overline{\boldsymbol{V}}\cdot\boldsymbol{A}\boldsymbol{V} = 0$$

which is a quadratic equation for ω.

Notice that,

$$\overline{\overline{\boldsymbol{V}}\cdot\boldsymbol{E}\boldsymbol{V}} = \boldsymbol{V}\cdot\boldsymbol{E}\overline{\boldsymbol{V}} = \overline{\boldsymbol{V}}\cdot\boldsymbol{E}^T\boldsymbol{V} = -\overline{\boldsymbol{V}}\cdot\boldsymbol{E}\boldsymbol{V},$$

and so $\overline{\boldsymbol{V}}\cdot\boldsymbol{E}\boldsymbol{V}$ equals the negative of its conjugate and therefore $\overline{\boldsymbol{V}}\cdot\boldsymbol{E}\boldsymbol{V} = bi$ for some real b.

Hence, with $a = \overline{\boldsymbol{V}}\cdot\boldsymbol{M}\boldsymbol{V}$ and $c = \overline{\boldsymbol{V}}\cdot\boldsymbol{A}\boldsymbol{V}$ the quadratic for ω is,

$$aw^2 + b\omega - c = 0.$$

Since b is real and a and c are positive, it follows that each eigenvalue, ω, is real.

Now suppose that $(\omega, \boldsymbol{V})$ is an eigen-pair for (1) and take the complex conjugate of that equation to get,

$$\begin{aligned} \boldsymbol{0} &= \boldsymbol{A}\overline{\boldsymbol{V}} - i\omega\boldsymbol{E}\overline{\boldsymbol{V}} - \omega^2\boldsymbol{M}\overline{\boldsymbol{V}}, \\ &= \boldsymbol{A}\overline{\boldsymbol{V}} + i(-\omega)\boldsymbol{E}\overline{\boldsymbol{V}} - (-\omega)^2\boldsymbol{M}\overline{\boldsymbol{V}}, \end{aligned}$$

which demonstrates that if $(\omega, \boldsymbol{V})$ is an eigen-pair for (1) then so too is $(-\omega, \overline{\boldsymbol{V}})$.

In summary, for (1): $\boldsymbol{M}$ and $\boldsymbol{A}$ are symmetric and positive definite; $\boldsymbol{E}$ is satifies the anti-symmetry $\boldsymbol{E}^T = -\boldsymbol{E}$; the eigenvalues, ω, are real; and, if $(\omega, \boldsymbol{V})$ is an eigenpair then so too is $(-\omega, \overline{\boldsymbol{V}})$.

The results that are of most interest are the modulus (approximately $946\mathrm{rad/sec}$ for our data) of the least-in-modulus eigenvalue, and the real and imaginary parts of the corresponding eigenvector. The imaginary part is the Coriolis distortion mode and the next section attempts to explain why this is the case.

4.1 The distortion mode

Assume a large time solution of the Euler-Bernoulli equations of the form,

$$u(x,t) = W_1(x)\sin(\gamma_1 t) + W_2(x)\sin(\gamma_1 t) + W_C(x)\cos(\gamma_1 t),$$

where W_1 is the no-flow solution ($V = 0$), W_2 is the modification to W_1 arising from the flow and W_C is the shape of the Coriolis distortion mode.

These functions are governed by three coupled boundary value problems. They are not given here. However, to see how this form of solution can be matched to the quadratic eigenvalue problem derived above we take,

$$\boldsymbol{U} = \boldsymbol{W}\sin(\gamma t) + \boldsymbol{W}_C\cos(\gamma t),$$

so that the vectors $\boldsymbol{W}$ and $\boldsymbol{W}_C$ correspond in turn to $W_1 + W_2$ and W_C.

Since,

$$\boldsymbol{U}_t = \gamma\boldsymbol{W}\cos(\gamma t) - \gamma\boldsymbol{W}_C\sin(\gamma t),$$

and,

$$\boldsymbol{U}_{tt} = -\gamma^2\boldsymbol{W}\sin(\gamma t) - \gamma^2\boldsymbol{W}_C\cos(\gamma t),$$

we can substitute into the ordinary differential equation system and see that we require that,

$$\begin{aligned} -\gamma^2\boldsymbol{M}\boldsymbol{W} - \gamma\boldsymbol{E}\boldsymbol{W}_C + \boldsymbol{A}\boldsymbol{W} &= \boldsymbol{0}, \\ -\gamma^2\boldsymbol{M}\boldsymbol{W}_C + \gamma\boldsymbol{E}\boldsymbol{W} + \boldsymbol{A}\boldsymbol{W}_C &= \boldsymbol{0}. \end{aligned}$$

Setting $\boldsymbol{Y} = \boldsymbol{W} + i\boldsymbol{W}_C$, and noting that $-(\boldsymbol{W}_C - i\boldsymbol{W}) = i\boldsymbol{Y}$, we can add these two equations to arrive at,

$$-\gamma^2\boldsymbol{M}\boldsymbol{Y} + i\gamma\boldsymbol{E}\boldsymbol{Y} + \boldsymbol{A}\boldsymbol{Y} = \boldsymbol{0}.$$

This is of exactly the same form as the quadratic eigenvalue problem derived above and we see that $\boldsymbol{W}_C = \mathrm{imag}(\boldsymbol{Y})$ is the modal approximation to the Coriolis distortion given by W_C.

By physical reasoning we expect that W_C has 180° rotational symmetry about the centre of the beam and exhibits a single maximum on one side of the centre, and a single minimum on the other side. It follows that W_C should exhibit an inflection at the beam's midpoint and this can be used to make a quick judgement as to the the quality of computed eigenvectors (see the results given later).

On the other hand, W has no inflection. It should have reflective symmetry about the beam's midpoint and exhibit a single maximum (or minimum).

4.2 Method 1: quadratic eigenvalue problem

The matlab fragment

```
[X, e] = polyeig(A, i*E, -M);
```

solves the quadratic eigenvalue problem in terms of a column of eigenvalues, `e`, and a matrix of eigenvectors, `X`. (Recall that $\boldsymbol{A}$ and $\boldsymbol{M}$ are invertible.)

4.3 Method 2: linear eigenvalue problem

Setting $\boldsymbol{W} = \omega \boldsymbol{V}$ so that $-\omega^2 \boldsymbol{M}\boldsymbol{V} = -\omega \boldsymbol{M}\boldsymbol{W}$ and recalling that $\boldsymbol{M}$ is invertible we can write the quadratic eigenvalue problem as a standard eigenvalue problem of block-system type:

$$\begin{pmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{M}^{-1}\boldsymbol{A} & i\boldsymbol{M}^{-1}\boldsymbol{E} \end{pmatrix} \begin{pmatrix} \boldsymbol{V} \\ \boldsymbol{W} \end{pmatrix} = \omega \begin{pmatrix} \boldsymbol{V} \\ \boldsymbol{W} \end{pmatrix}.$$

This can be abbreviated to $\boldsymbol{B}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{L}$, where $\boldsymbol{B}$ is the block-matrix, $\boldsymbol{L}$ is a diagonal matrix of eigenvalues and $\boldsymbol{X}$ is a matrix of eigenvectors, and can be solved in matlab via the fragment,

```
B = [ zeros(4,4) eye(4) ; M\A i*M\E ];
[X L] = eig(B);
```

(for e.g. a 4 by 4 system).

4.4 Method 3: Inverse iteration

If $\boldsymbol{D}$ is a non-singular matrix then, given an initial guess $\boldsymbol{x}_0$, the inverse power iteration,

$$\boldsymbol{z}_{n+1} = \boldsymbol{D}^{-1}\boldsymbol{x}_n, \qquad \boldsymbol{x}_{n+1} = \boldsymbol{z}_{n+1}\|\boldsymbol{z}_{n+1}\|_\infty^{-1}, \qquad n = 0, 1, 2, \ldots$$

is well defined. The second step, where

$$\|\boldsymbol{z}\|_\infty := \max\{|z_i| \colon 1 \leqslant i \leqslant N\}$$

for $\boldsymbol{z} = (z_1, z_2, \ldots, z_N)^T$, is a normalization and is important to ensure that the iteration is convergent.

If $\boldsymbol{D}$ has a single eigenvalue of least modulus then this iteration converges to the eigenvector for that eigenvalue. In that case we can write $\boldsymbol{x}_n \to \boldsymbol{x}$, and then obtain the eigenvalue from the Rayleigh quotient,

$$\lambda = \frac{\boldsymbol{x} \cdot \boldsymbol{D}\boldsymbol{x}}{\boldsymbol{x} \cdot \boldsymbol{x}}.$$

If there is more than one eigenvalue of minimal modulus the situation is less clear. The iterates may cycle over each of the associated eigenvectors, or even over, or converge to, a linear combination of the eigenvectors.

If these minimal modulus eigenvalues are identical then it is unclear how to proceed but, if they are algebraically distinct (e.g. $a + ib$ and $-a + ib$) then we can force the inverse iteration to converge by employing a 'shift'.

Suppose that $(\lambda, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{D}$ then $(\lambda - p, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{D} - p\boldsymbol{I}$ because,

$$\boldsymbol{0} = (\boldsymbol{D} - \lambda\boldsymbol{I})\boldsymbol{x} = (\boldsymbol{D} - \lambda\boldsymbol{I})\boldsymbol{x} - p\boldsymbol{I}\boldsymbol{x} + p\boldsymbol{I}\boldsymbol{x} = ((\boldsymbol{D} - p\boldsymbol{I}) - (\lambda - p)\boldsymbol{I})\boldsymbol{x}.$$

Here p is the 'shift' and if $\boldsymbol{D}$ has two minimal modulus eigenvalues, say,

$$a + bi \qquad \text{and} \qquad -a + bi,$$

then $\boldsymbol{D} - p\boldsymbol{I}$ has (for real p),

$$(a - p) + bi \qquad \text{and} \qquad -(a + p) + bi.$$

An informed choice for p therefore reduces the modulus of one eigenvalue while increasing the modulus of the other. The inverse iteration then becomes convergent.

This can be used for the Coriolis beam problem, and in this context the two main issues are:

- the choice of initial guess, $\boldsymbol{x}_0$;
- the value fof the shift, p.

In terms of the matrix system given above we can write the non-shifted inverse iteration for the beam either as,

$$\boldsymbol{Z}_{n+1} = \boldsymbol{B}^{-1}\boldsymbol{X}_n, \qquad \boldsymbol{X}_{n+1} = \boldsymbol{Z}_{n+1}\|\boldsymbol{Z}_{n+1}\|_\infty^{-1},$$

or in components,

$$\boldsymbol{W}_{n+1} = \boldsymbol{V}_n,$$
$$\boldsymbol{V}_{n+1} = \boldsymbol{A}^{-1}\big(\boldsymbol{M}\boldsymbol{W}_n - i\boldsymbol{E}\boldsymbol{W}_{n+1}\big),$$

with some form of normalisation.

In the same way, the shifted inverse iteration is either,

$$\boldsymbol{Z}_{n+1} = (\boldsymbol{B} - p\boldsymbol{I})^{-1}\boldsymbol{X}_n, \qquad \boldsymbol{X}_{n+1} = \boldsymbol{Z}_{n+1}\|\boldsymbol{Z}_{n+1}\|_\infty^{-1},$$

or (again omitting the normalisation),

$$\boldsymbol{W}_{n+1} - p\boldsymbol{V}_{n+1} = \boldsymbol{V}_n,$$

$$\boldsymbol{A}\boldsymbol{V}_{n+1} + (i\boldsymbol{E} - p\boldsymbol{M})\boldsymbol{W}_{n+1} = \boldsymbol{M}\boldsymbol{W}_n.$$

Using the first equation, $\boldsymbol{W}_{n+1} = \boldsymbol{V}_n + p\boldsymbol{V}_{n+1}$, in the second then leads to,

$$(\boldsymbol{A} + ip\boldsymbol{E} - p^2\boldsymbol{M})\boldsymbol{V}_{n+1} = \boldsymbol{M}\boldsymbol{W}_n + (p\boldsymbol{M} - i\boldsymbol{E})\boldsymbol{V}_n.$$

In terms of implementation this iteration requires complex arithmetic. This presents no problems for tools such as Matlab, but if we want to use a plain vanilla C/C++ code, for example, then we need to work with real and imaginary components.

Writing,

$$\boldsymbol{V}_n = \boldsymbol{V}_n^R + i\boldsymbol{V}_n^I \qquad \text{and} \qquad \boldsymbol{W}_n = \boldsymbol{W}_n^R + i\boldsymbol{W}_n^I,$$

and taking p to be real, the iteration above can be written as,

$$\boldsymbol{W}_{n+1}^R = \boldsymbol{V}_n^R + p\boldsymbol{V}_{n+1}^R \qquad \text{and} \qquad \boldsymbol{W}_{n+1}^I = \boldsymbol{V}_n^I + p\boldsymbol{V}_{n+1}^I$$

and

$$(\boldsymbol{A} + ip\boldsymbol{E} - p^2\boldsymbol{M})(\boldsymbol{V}_{n+1}^R + i\boldsymbol{V}_{n+1}^I) = \boldsymbol{M}(\boldsymbol{W}_n^R + i\boldsymbol{W}_n^I) + (p\boldsymbol{M} - i\boldsymbol{E})(\boldsymbol{V}_n^R + i\boldsymbol{V}_n^I).$$

This last equation can be written as,

$$\begin{pmatrix} \boldsymbol{A} - p^2\boldsymbol{M} & -p\boldsymbol{E} \\ p\boldsymbol{E} & \boldsymbol{A} - p^2\boldsymbol{M} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}_{n+1}^R \\ \boldsymbol{V}_{n+1}^I \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{W}_n^R + p\boldsymbol{M}\boldsymbol{V}_n^R + \boldsymbol{E}\boldsymbol{V}_n^I \\ \boldsymbol{M}\boldsymbol{W}_n^I - \boldsymbol{E}\boldsymbol{V}_n^R + p\boldsymbol{M}\boldsymbol{V}_n^I \end{pmatrix}.$$

If we abreviate this system to,

$$\begin{pmatrix} \boldsymbol{\alpha} & -\boldsymbol{\beta} \\ \boldsymbol{\beta} & \boldsymbol{\alpha} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}_{n+1}^R \\ \boldsymbol{V}_{n+1}^I \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \end{pmatrix},$$

then a single block-row operation gives $\boldsymbol{V}_{n+1}$ in components as the solution of,

$$(\boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\alpha}^{-1}\boldsymbol{\beta})\boldsymbol{V}_{n+1}^I = \boldsymbol{q}_2 - \boldsymbol{\beta}\boldsymbol{\alpha}^{-1}\boldsymbol{q}_1,$$

$$\boldsymbol{\alpha}\boldsymbol{V}^R_{n+1} = \boldsymbol{q}_1 + \boldsymbol{\beta}\boldsymbol{V}^I_{n+1}.$$

Since constructing the inverse of a matrix is usually regarded as ill-advised in computational linear algebra, the first equation above is problematic due to the presence of $\boldsymbol{\alpha}^{-1}$ on the right hand side (recall also that $\boldsymbol{E}$—hence $\boldsymbol{\beta}$—is singular). However, notice that a rearrangement,

$$\boldsymbol{\alpha}\boldsymbol{V}^I_{n+1} = \boldsymbol{q}_2 - \boldsymbol{\beta}\boldsymbol{\alpha}^{-1}(\boldsymbol{q}_1 + \boldsymbol{\beta}\boldsymbol{V}^I_{n+1}),$$

can form the basis of a fixed-point iteration. Algorithmically, this iteration is:

set $\boldsymbol{V}^I_{n+1} = \boldsymbol{V}^I_n$
do {
    set $\hat{\boldsymbol{V}}^I_{n+1} = \boldsymbol{V}^I_{n+1}$
    solve $\boldsymbol{\alpha}\boldsymbol{q_3} = \boldsymbol{q}_1 + \boldsymbol{\beta}\hat{\boldsymbol{V}}^I_{n+1}$ *for* $\boldsymbol{q}_3$
    solve $\boldsymbol{\alpha}\boldsymbol{V}^I_{n+1} = \boldsymbol{q}_2 - \boldsymbol{\beta}\boldsymbol{q}_3$ *for* $\boldsymbol{V}^I_{n+1}$
} *while* $\|\boldsymbol{V}^I_{n+1} - \hat{\boldsymbol{V}}^I_{n+1}\|_\infty > \epsilon$

where ϵ is a user-defined tolerance.

4.5 Method 4: shifting the quadratic eigenvalue problem

In method 1 we saw that the matlab fragment

```
[X, e] = polyeig(A, i*E, -M);
```

can be used to solve the quadratic eigenvalue problem. It is relevant here to see how a shift can be introduced into this formulation of the problem.

The shifted version of the linear eigenvalue problem given in Method 2 is,

$$\begin{pmatrix} -p\boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{M}^{-1}\boldsymbol{A} & i\boldsymbol{M}^{-1}\boldsymbol{E} - p\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{V} \\ \boldsymbol{W} \end{pmatrix} = \omega \begin{pmatrix} \boldsymbol{V} \\ \boldsymbol{W} \end{pmatrix}.$$

From the top-row equation we derive,

$$\boldsymbol{W} = (\omega + p)\boldsymbol{V},$$

and substituting this into the bottom-row equation results (fairly obviously) in,

$$\boldsymbol{A}\boldsymbol{V} + i(\omega + p)\boldsymbol{E}\boldsymbol{V} - (p + \omega)^2\boldsymbol{M}\boldsymbol{V} = \boldsymbol{0}.$$

Collecting terms in like powers of ω results in another quadratic eigenvalue problem.

$$(\hat{\boldsymbol{A}} + i\omega\hat{\boldsymbol{E}} - \omega^2\boldsymbol{M})\boldsymbol{V} = \boldsymbol{0},$$

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + ip\boldsymbol{E} - p^2\boldsymbol{M}$ and $\hat{\boldsymbol{E}} = \boldsymbol{E} + 2ip\boldsymbol{M}$.

Therefore, if `polyeig` is giving poor results due to two algebraically different but equal-in-minimal-modulus eigenvalues then this shifted system may improve matters. It is worth investigating.

5 Numerical experiments

This section contains the outcome of several computations for both beam models, with varying numbers of elements and, for the Timoshenko beam, various types of elements (linear, quadratic and cubic). We used a bespoke C/C++ code to generate the finite element matrices and then fed these into Matlab (7.1.0.246 (R14) SP3) to generate the eigen results shown below.

For each of the methods described above the computed least-in-modulus value of $|\omega|$, for various numbers, N_e, of equal width finite elements, are shown for the Euler-Bernoulli beam in Table 1, with no shift, and Table 2, with shift $p = 900\,\mathrm{rad/sec}$. The analogous results for the Timoshenko beam are shown in Tables 3 and 4, for linear elements; 5 and 6, for quadratic elements; and, 7 and 8, for cubic elements. In these tables 'II' means 'inverse iteration'. (Note that some results for larger values of N_e are missing for the Timoshenko beam due to the PC we were using having insufficient memory to deal with the large matrices.)

In each case ten inverse iterations were carried out, and the initial eigenvector for the inverse iteration was,

$$\begin{pmatrix} 1+10^{-5}i \\ 1+10^{-5}i \\ \vdots \\ 1+10^{-5}i \end{pmatrix}.$$

This choice was arrived at by trial and error, but motivated by the fact that the imaginary part is known to be several orders of magnitude less than the real part.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	946.1995	946.1995	$\approx 1 \leftrightarrow \approx 13000$
32	946.1948	946.1948	$\approx 1 \leftrightarrow \approx 13000$
64	946.1945	946.1945	$\approx 1 \leftrightarrow \approx 13000$
128	946.1941	946.1944	$\approx 1 \leftrightarrow \approx 13000$
256	946.1941	946.1939	$\approx 1 \leftrightarrow \approx 13000$
512	945.6362	946.1943	$\approx 1 \leftrightarrow \approx 13000$

Table 1: Computed $|\omega|_{\min}$ for the Euler-Bernoulli beam (cubic elements) with no shift, $p = 0$.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	946.1995	946.1995	946.1995
32	946.1948	946.1948	946.1948
64	946.1945	946.1945	946.1945
128	946.1945	946.1944	946.1943
256	946.1942	946.1939	946.1720
512	946.1923	946.1943	946.6341

Table 2: Computed $|\omega|_{\min}$ for the Euler-Bernoulli beam (cubic elements) with shift $p = 900$.

For a meter of length L, the meter sensitivity is defined as $(q_1 - q_2)/\dot{m}$ where $\dot{m}$ is the fluid mass flow rate, $q_1 = \arg V(L/4)$ and $q_2 = \arg V(3L/4)$. These values are shown in Tables 9 and 10 for the Euler-Bernoulli beam, and Tables 11, 12, 13, 14, 15 and 16 for the Timoshenko beam.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	1362.101	1362.101	$\approx 1 \leftrightarrow \approx 28000$
32	1059.941	1059.941	$\approx 1 \leftrightarrow \approx 16000$
64	970.4797	970.4797	$\approx 1 \leftrightarrow \approx 13000$
128	946.8384	946.8384	$\approx 1 \leftrightarrow \approx 13000$
256	940.8380	940.8380	$\approx 1 \leftrightarrow \approx 13000$
512	939.3323	939.3321	$\approx 1 \leftrightarrow \approx 13000$

Table 3: Computed $|\omega|_{\min}$ for the Timoshenko beam (linear elements) with no shift, $p = 0$.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	1362.101	1362.101	1362.102
32	1059.941	1059.941	1059.941
64	970.4797	970.4797	970.4797
128	946.8384	946.8384	946.8384
256	940.8380	940.8380	940.8380
512	939.3321	939.3321	939.3321

Table 4: Computed $|\omega|_{\min}$ for the Timoshenko beam (linear elements) with shift $p = 900$.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	940.1753	940.1753	$\approx 1 \leftrightarrow \approx 13000$
32	938.9266	938.9266	$\approx 1 \leftrightarrow \approx 13000$
64	938.8359	938.8359	$\approx 1 \leftrightarrow \approx 13000$
128	938.8298	938.8300	$\approx 1 \leftrightarrow \approx 13000$
256	938.8296	938.8296	$\approx 1 \leftrightarrow \approx 13000$

Table 5: Computed $|\omega|_{\min}$ for the Timoshenko beam (quadratic elements) with no shift, $p = 0$.

	$\lvert\omega\rvert_{\min}$		
N_e	`polyeig`	`eig`	II
16	940.1753	940.1753	940.1753
32	938.9266	938.9266	938.9266
64	938.8359	938.8359	938.8359
128	938.8300	938.8300	938.8300
256	938.8296	938.8296	938.8296

Table 6: Computed $|\omega|_{\min}$ for the Timoshenko beam (quadratic elements) with shift $p = 900$.

Figures 1 to 6 show results for the Euler-Bernoulli beam for a shift of $p = 0$. Figures 7 to 12 show the corresponding results when a shift of $p = 900$ was employed.

A similar set of results are shown for the Timoshenko beam in: Figures 13 to 18 and Figures 19 to 24 for linear elements; Figures 25 to 29 and Figures 30 to 34 for quadratic elements; and, Figures 35 to 38 and Figures 39 to 42 for cubic elements.

N_e	$\|\omega\|_{\min}$ polyeig	eig	II
16	938.8300	938.8300	$\approx 1 \leftrightarrow \approx 13000$
32	938.8296	938.8296	$\approx 1 \leftrightarrow \approx 13000$
64	938.8296	938.8296	$\approx 1 \leftrightarrow \approx 13000$
128	938.8299	938.8296	$\approx 1 \leftrightarrow \approx 13000$
256	$*$	938.8296	$\approx 1 \leftrightarrow \approx 13000$

Table 7: Computed $|\omega|_{\min}$ for the Timoshenko beam (cubic elements) with no shift, $p = 0$.

N_e	$\|\omega\|_{\min}$ polyeig	eig	II
16	938.8300	938.8300	938.8300
32	938.8296	938.8296	938.8296
64	938.8296	938.8296	938.8296
128	938.8296	938.8296	938.8296
256	$*$	938.8296	938.8296

Table 8: Computed $|\omega|_{\min}$ for the Timoshenko beam (cubic elements) with shift $p = 900$.

N_e		sensitivities degrees) polyeig	eig	II
16	q_1	-4.844533×10^{1}	2.942431×10^{-1}	1.036817×10^{-3}
	q_2	-4.932315×10^{1}	-5.835774×10^{-1}	1.091165×10^{-4}
	$q_1 - q_2$	-8.778205×10^{-1}	-8.778205×10^{-1}	-9.277003×10^{-4}
32	q_1	5.368917×10^{1}	6.956238×10^{-1}	1.036820×10^{-3}
	q_2	5.456701×10^{1}	-1.822097×10^{-1}	1.090967×10^{-4}
	$q_1 - q_2$	8.778334×10^{-1}	-8.778335×10^{-1}	-9.277233×10^{-4}
64	q_1	3.547197×10^{1}	6.956246×10^{-1}	1.036821×10^{-3}
	q_2	3.459413×10^{1}	-1.822098×10^{-1}	1.090950×10^{-4}
	$q_1 - q_2$	-8.778397×10^{-1}	-8.778344×10^{-1}	-9.277257×10^{-4}
128	q_1	-4.777672×10^{1}	-6.688155×10^{-1}	1.036821×10^{-3}
	q_2	-4.865456×10^{1}	2.090189×10^{-1}	1.090948×10^{-4}
	$q_1 - q_2$	-8.778383×10^{-1}	8.778345×10^{-1}	-9.277261×10^{-4}
256	q_1	5.011923	-6.823000×10^{-1}	1.036821×10^{-3}
	q_2	5.889219	1.955341×10^{-1}	1.090948×10^{-4}
	$q_1 - q_2$	8.772961×10^{-1}	8.778341×10^{-1}	-9.277262×10^{-4}
512	q_1	-6.543358×10^{1}	2.023207×10^{-1}	1.036821×10^{-3}
	q_2	-6.455142×10^{1}	-6.757378×10^{-1}	1.090946×10^{-4}
	$q_1 - q_2$	8.821615×10^{-1}	-8.780585×10^{-1}	-9.277263×10^{-4}

Table 9: Computed meter sensitivities for the Euler-Bernoulli beam without shift ($p = 0$).

N_e		sensitivities (degrees) polyeig	eig	II
16	q_1	-8.326008	2.942431×10^{-1}	4.398344×10^{-1}
	q_2	-7.448187	-5.835774×10^{-1}	-4.379861×10^{-1}
	$q_1 - q_2$	8.778205×10^{-1}	-8.778205×10^{-1}	-8.778205×10^{-1}
32	q_1	-5.950667×10^{1}	6.956238×10^{-1}	4.395348×10^{-1}
	q_2	-5.862883×10^{1}	-1.822097×10^{-1}	-4.382988×10^{-1}
	$q_1 - q_2$	8.778335×10^{-1}	-8.778335×10^{-1}	-8.778335×10^{-1}
64	q_1	-8.799232	6.956246×10^{-1}	4.394958×10^{-1}
	q_2	-7.921398	-1.822098×10^{-1}	-4.383385×10^{-1}
	$q_1 - q_2$	8.778341×10^{-1}	-8.778344×10^{-1}	-8.778343×10^{-1}
128	q_1	-5.142200×10^{1}	-6.688155×10^{-1}	4.394908×10^{-1}
	q_2	-5.054417×10^{1}	2.090189×10^{-1}	-4.383436×10^{-1}
	$q_1 - q_2$	8.778373×10^{-1}	8.778345×10^{-1}	-8.778344×10^{-1}
256	q_1	5.233954×10^{1}	-6.823000×10^{-1}	4.394915×10^{-1}
	q_2	5.321735×10^{1}	1.955341×10^{-1}	-4.383430×10^{-1}
	$q_1 - q_2$	8.778170×10^{-1}	8.778341×10^{-1}	-8.778345×10^{-1}
512	q_1	7.111125×10^{1}	2.023207×10^{-1}	4.394881×10^{-1}
	q_2	7.198967×10^{1}	-6.757378×10^{-1}	-4.383452×10^{-1}
	$q_1 - q_2$	8.784199×10^{-1}	-8.780585×10^{-1}	-8.778333×10^{-1}

Table 10: Computed meter sensitivities for the Euler-Bernoulli beam with shift $p = 900$.

N_e		sensitivities (degrees) polyeig	eig	II
16	q_1	8.620849×10^{1}	3.988202×10^{-1}	7.959019×10^{-4}
	q_2	8.560109×10^{1}	-2.085772×10^{-1}	3.500188×10^{-4}
	$q_1 - q_2$	-6.073973×10^{-1}	-6.073973×10^{-1}	-4.458831×10^{-4}
32	q_1	-8.786691	-6.196106×10^{-1}	9.464655×10^{-4}
	q_2	-9.578552	1.722510×10^{-1}	1.994542×10^{-4}
	$q_1 - q_2$	-7.918616×10^{-1}	7.918616×10^{-1}	-7.470113×10^{-4}
64	q_1	-7.149025×10^{1}	6.791112×10^{-1}	1.020102×10^{-3}
	q_2	-7.235821×10^{1}	-1.888517×10^{-1}	1.258172×10^{-4}
	$q_1 - q_2$	-8.679628×10^{-1}	-8.679629×10^{-1}	-8.942845×10^{-4}
128	q_1	4.512374×10^{1}	-6.698200×10^{-1}	1.043130×10^{-3}
	q_2	4.601418×10^{1}	2.206100×10^{-1}	1.027887×10^{-4}
	$q_1 - q_2$	8.904338×10^{-1}	8.904300×10^{-1}	-9.403413×10^{-4}
256	q_1	-7.336709×10^{1}	2.084588×10^{-1}	1.049253×10^{-3}
	q_2	-7.426342×10^{1}	-6.878501×10^{-1}	9.666585×10^{-5}
	$q_1 - q_2$	-8.963245×10^{-1}	-8.963089×10^{-1}	-9.525869×10^{-4}
512	q_1	1.462008×10^{1}	-6.821975×10^{-1}	1.050808×10^{-3}
	q_2	1.372235×10^{1}	2.155985×10^{-1}	9.511075×10^{-5}
	$q_1 - q_2$	-8.977313×10^{-1}	8.977959×10^{-1}	-9.556971×10^{-4}

Table 11: Computed meter sensitivities for the Timoshenko beam (linear elements) without shift ($p = 0$).

N_e		sensitivities (degrees) `polyeig`	`eig`	II
16	q_1	1.820939×10^{1}	3.988202×10^{-1}	3.042750×10^{-1}
	q_2	1.881678×10^{1}	-2.085772×10^{-1}	-3.031221×10^{-1}
	$q_1 - q_2$	6.073973×10^{-1}	-6.073973×10^{-1}	-6.073971×10^{-1}
32	q_1	3.969592×10^{1}	-6.196106×10^{-1}	3.965059×10^{-1}
	q_2	4.048778×10^{1}	1.722510×10^{-1}	-3.953557×10^{-1}
	$q_1 - q_2$	7.918616×10^{-1}	7.918616×10^{-1}	-7.918616×10^{-1}
64	q_1	-7.635198×10^{1}	6.791112×10^{-1}	4.345560×10^{-1}
	q_2	-7.548401×10^{1}	-1.888517×10^{-1}	-4.334069×10^{-1}
	$q_1 - q_2$	8.679629×10^{-1}	-8.679629×10^{-1}	-8.679629×10^{-1}
128	q_1	-6.346056×10^{1}	-6.698200×10^{-1}	4.457894×10^{-1}
	q_2	-6.257013×10^{1}	2.206100×10^{-1}	-4.446406×10^{-1}
	$q_1 - q_2$	8.904306×10^{-1}	8.904300×10^{-1}	-8.904300×10^{-1}
256	q_1	5.998508×10^{1}	2.084588×10^{-1}	4.487288×10^{-1}
	q_2	$6.088139 \times 10^{+1}$	-6.878501×10^{-1}	-4.475801×10^{-1}
	$q_1 - q_2$	8.963110×10^{-1}	-8.963089×10^{-1}	-8.963089×10^{-1}
512	q_1	-8.159702×10^{1}	-6.821975×10^{-1}	4.494723×10^{-1}
	q_2	-8.069922×10^{1}	2.155985×10^{-1}	-4.483236×10^{-1}
	$q_1 - q_2$	8.977960×10^{-1}	8.977959×10^{-1}	-8.977959×10^{-1}

Table 12: Computed meter sensitivities for the Timoshenko beam (linear elements) with shift $p = 900$.

N_e		sensitivities (degrees) `polyeig`	`eig`	II
16	q_1	8.456385×10^{1}	-1.940224×10^{-1}	1.048495×10^{-3}
	q_2	8.366959×10^{1}	7.002301×10^{-1}	9.742365×10^{-5}
	$q_1 - q_2$	-8.942525×10^{-1}	8.942525×10^{-1}	-9.510714×10^{-4}
32	q_1	1.286257×10^{1}	1.952803×10^{-1}	1.051124×10^{-3}
	q_2	1.376057×10^{1}	-7.027220×10^{-1}	9.479458×10^{-5}
	$q_1 - q_2$	8.980022×10^{-1}	-8.980023×10^{-1}	-9.563294×10^{-4}
64	q_1	-6.430425×10^{1}	-6.757193×10^{-1}	1.051315×10^{-3}
	q_2	-6.340597×10^{1}	2.225550×10^{-1}	9.460349×10^{-5}
	$q_1 - q_2$	8.982804×10^{-1}	8.982743×10^{-1}	-9.567116×10^{-4}
128	q_1	-6.143330×10^{1}	-6.893710×10^{-1}	1.051328×10^{-3}
	q_2	-6.053498×10^{1}	2.089209×10^{-1}	9.459103×10^{-5}
	$q_1 - q_2$	8.983137×10^{-1}	8.982919×10^{-1}	-9.567365×10^{-4}
256	q_1	-6.455767×10^{1}	-2.157181×10^{-1}	1.051328×10^{-3}
	q_2	-6.545599×10^{1}	6.825750×10^{-1}	9.459023×10^{-5}
	$q_1 - q_2$	-8.983209×10^{-1}	8.982931×10^{-1}	-9.567381×10^{-4}

Table 13: Computed meter sensitivities for the Timoshenko beam (quadratic elements) without shift ($p = 0$).

N_e		sensitivities (degrees) `polyeig`	`eig`	II
16	q_1	3.530840×10^{1}	-1.940224×10^{-1}	4.477007×10^{-1}
	q_2	3.620266×10^{1}	7.002301×10^{-1}	-4.465518×10^{-1}
	$q_1 - q_2$	8.942525×10^{-1}	8.942525×10^{-1}	-8.942525×10^{-1}
32	q_1	1.433462×10^{1}	1.952803×10^{-1}	4.495755×10^{-1}
	q_2	1.523262×10^{1}	-7.027220×10^{-1}	-4.484268×10^{-1}
	$q_1 - q_2$	8.980028×10^{-1}	-8.980023×10^{-1}	-8.980023×10^{-1}
64	q_1	3.689830×10^{1}	-6.757193×10^{-1}	4.497115×10^{-1}
	q_2	3.779658×10^{1}	2.225550×10^{-1}	-4.485628×10^{-1}
	$q_1 - q_2$	8.982745×10^{-1}	8.982743×10^{-1}	-8.982742×10^{-1}
128	q_1	-6.880750×10^{1}	-6.893710×10^{-1}	4.497203×10^{-1}
	q_2	-6.790921×10^{1}	2.089209×10^{-1}	-4.485716×10^{-1}
	$q_1 - q_2$	8.982929×10^{-1}	8.982919×10^{-1}	-8.982919×10^{-1}
256	q_1	4.069199×10^{1}	-2.157181×10^{-1}	4.497209×10^{-1}
	q_2	4.159029×10^{1}	6.825750×10^{-1}	-4.485722×10^{-1}
	$q_1 - q_2$	8.983055×10^{-1}	8.982931×10^{-1}	-8.982931×10^{-1}

Table 14: Computed meter sensitivities for the Timoshenko beam (quadratic elements) with shift $p = 900$.

N_e		sensitivities (degrees) `polyeig`	`eig`	II
16	q_1	-8.121455×10^{1}	-6.665646×10^{-1}	1.051327×10^{-3}
	q_2	-8.211284×10^{1}	2.317273×10^{-1}	9.459181×10^{-5}
	$q_1 - q_2$	-8.982918×10^{-1}	8.982919×10^{-1}	-9.567355×10^{-4}
32	q_1	4.225465×10^{1}	-6.665280×10^{-1}	1.051328×10^{-3}
	q_2	4.135634×10^{1}	2.317651×10^{-1}	9.459042×10^{-5}
	$q_1 - q_2$	-8.983098×10^{-1}	8.982931×10^{-1}	-9.567379×10^{-4}
64	q_1	-7.654419×10^{1}	2.134473×10^{-1}	1.051328×10^{-3}
	q_2	-7.744251×10^{1}	-6.848458×10^{-1}	9.459024×10^{-5}
	$q_1 - q_2$	-8.983196×10^{-1}	-8.982931×10^{-1}	-9.567381×10^{-4}
128	q_1	2.024034	-2.134472×10^{-1}	1.051328×10^{-3}
	q_2	2.922323	6.848459×10^{-1}	9.459019×10^{-5}
	$q_1 - q_2$	8.982888×10^{-1}	8.982932×10^{-1}	-9.567382×10^{-4}
256	q_1	$*$	-6.848458×10^{-1}	1.051328×10^{-3}
	q_2	$*$	2.134473×10^{-1}	9.459018×10^{-5}
	$q_1 - q_2$	$*$	-8.982932×10^{-1}	9.567382×10^{-4}

Table 15: Computed meter sensitivities for the Timoshenko beam (cubic elements) without shift ($p = 0$).

N_e		sensitivities (degrees) `polyeig`	`eig`	II
16	q_1	-3.764567	-6.665646×10^{-1}	4.497206×10^{-1}
	q_2	-2.866275	2.317273×10^{-1}	-4.485713×10^{-1}
	$q_1 - q_2$	8.982921×10^{-1}	8.982919×10^{-1}	-8.982919×10^{-1}
32	q_1	8.126765×10^{1}	-6.665280×10^{-1}	4.497210×10^{-1}
	q_2	8.216594×10^{1}	2.317651×10^{-1}	-4.485721×10^{-1}
	$q_1 - q_2$	8.982931×10^{-1}	8.982931×10^{-1}	-8.982931×10^{-1}
64	q_1	6.778602×10^{1}	2.134473×10^{-1}	4.497209×10^{-1}
	q_2	6.868432×10^{1}	-6.848458×10^{-1}	-4.485722×10^{-1}
	$q_1 - q_2$	8.982939×10^{-1}	-8.982931×10^{-1}	-8.982931×10^{-1}
128	q_1	-3.645084×10^{1}	-2.134472×10^{-1}	4.497209×10^{-1}
	q_2	-3.555254×10^{1}	6.848459×10^{-1}	-4.485722×10^{-1}
	$q_1 - q_2$	8.982929×10^{-1}	8.982932×10^{-1}	-8.982931×10^{-1}
256	q_1	$*$	-6.848458×10^{-1}	4.497210×10^{-1}
	q_2	$*$	2.134473×10^{-1}	-4.485721×10^{-1}
	$q_1 - q_2$	$*$	-8.982932×10^{-1}	8.982931×10^{-1}

Table 16: Computed meter sensitivities for the Timoshenko beam (cubic elements) with shift $p = 900$.

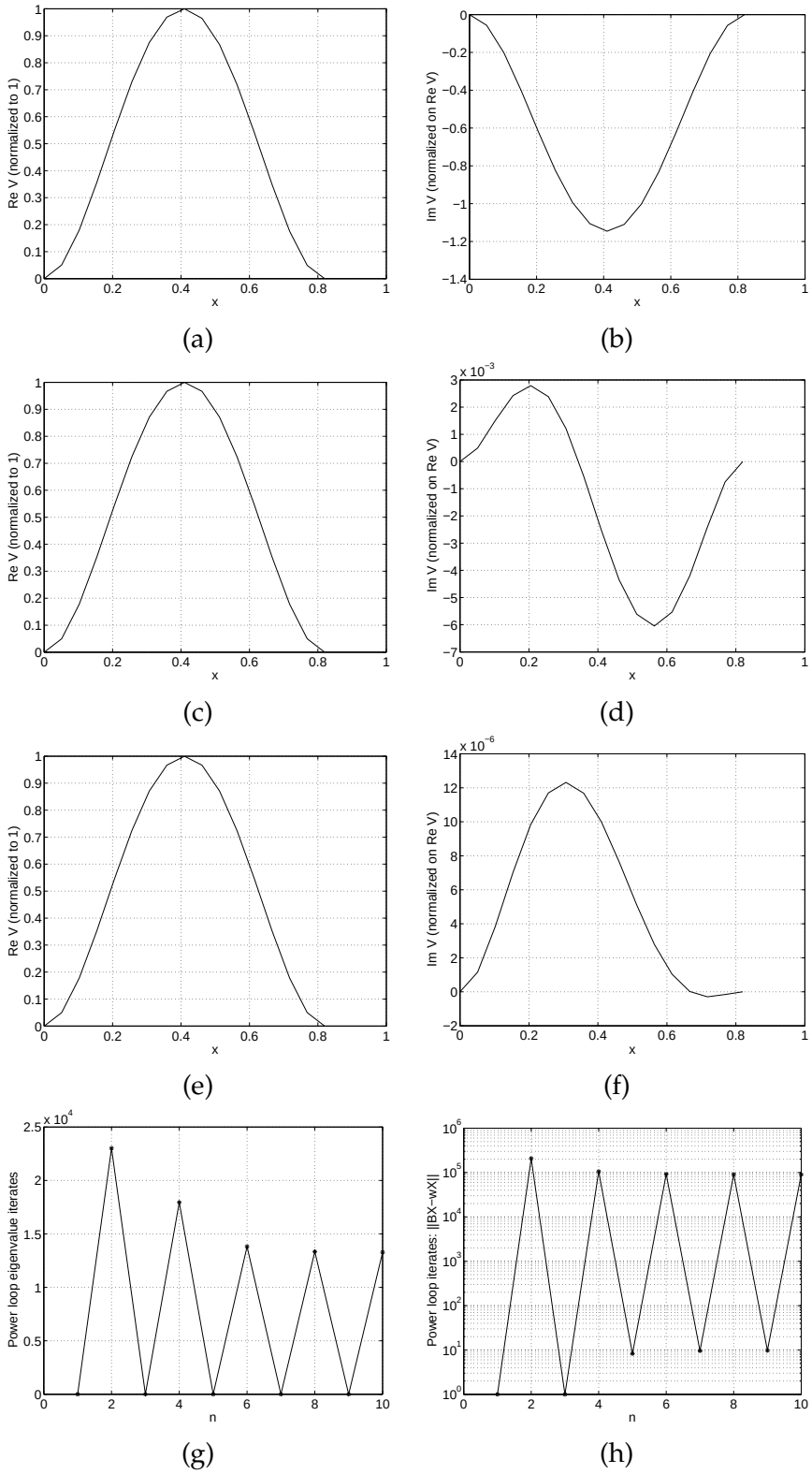

Figure 1: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 16 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

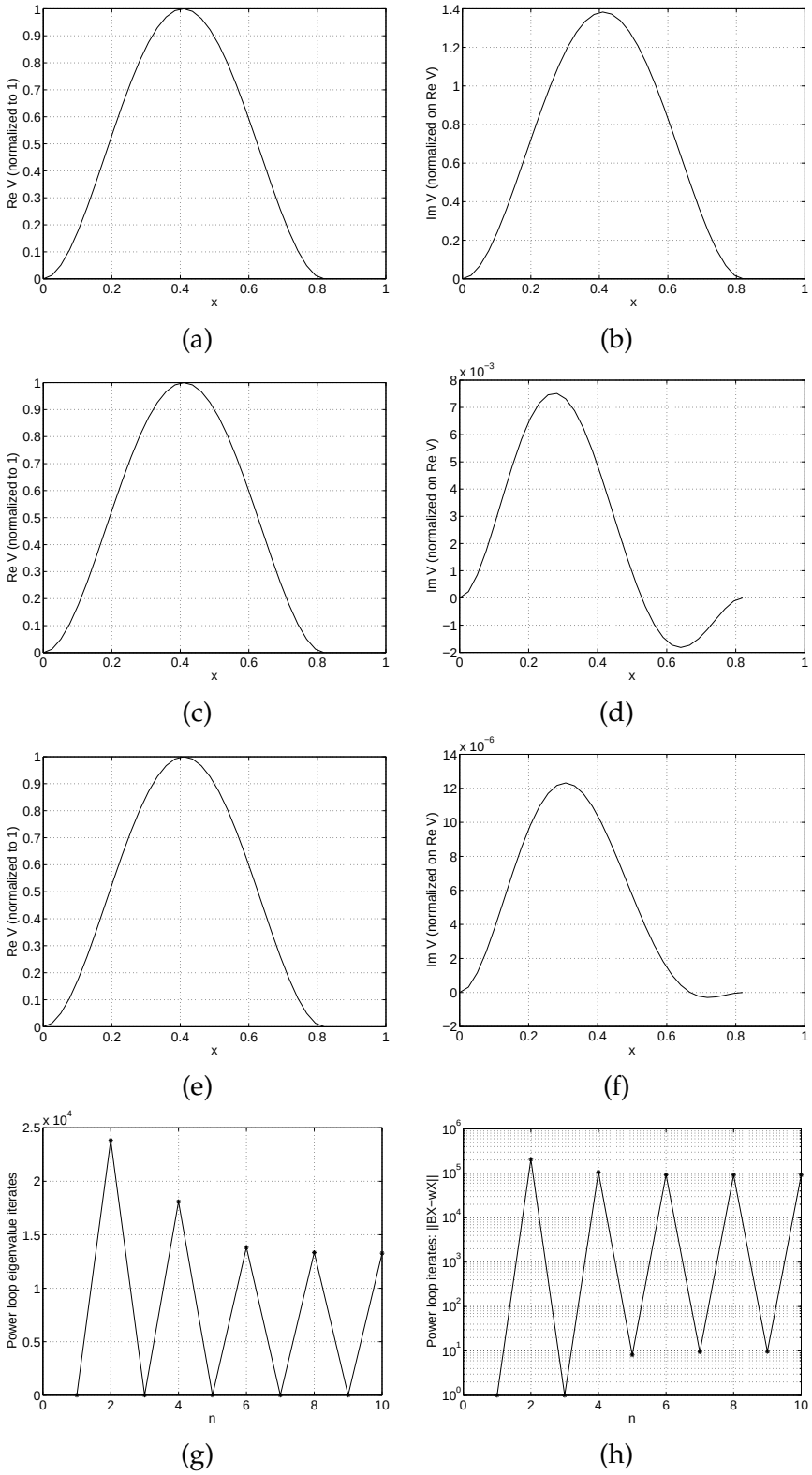

Figure 2: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 32 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

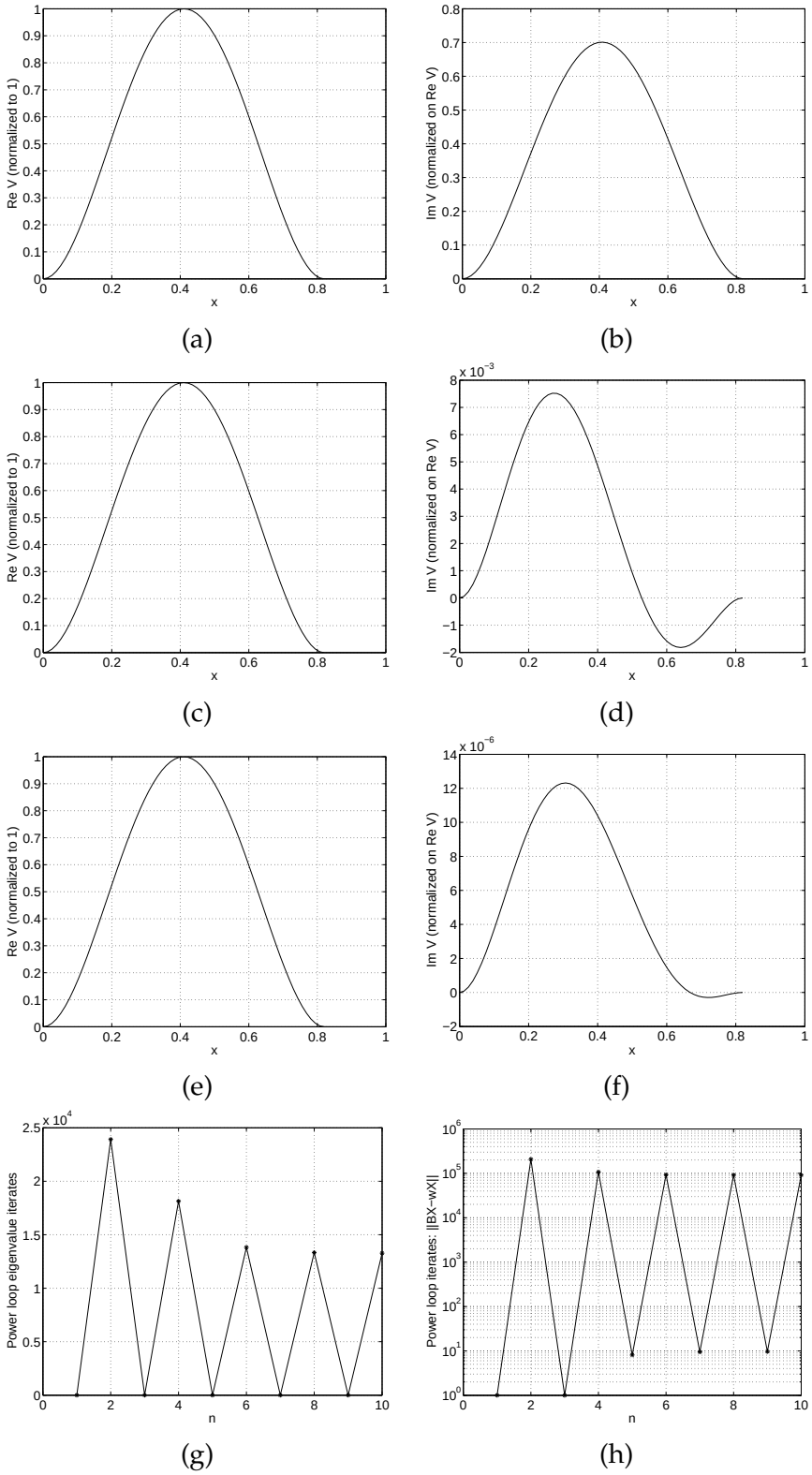

Figure 3: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 64 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

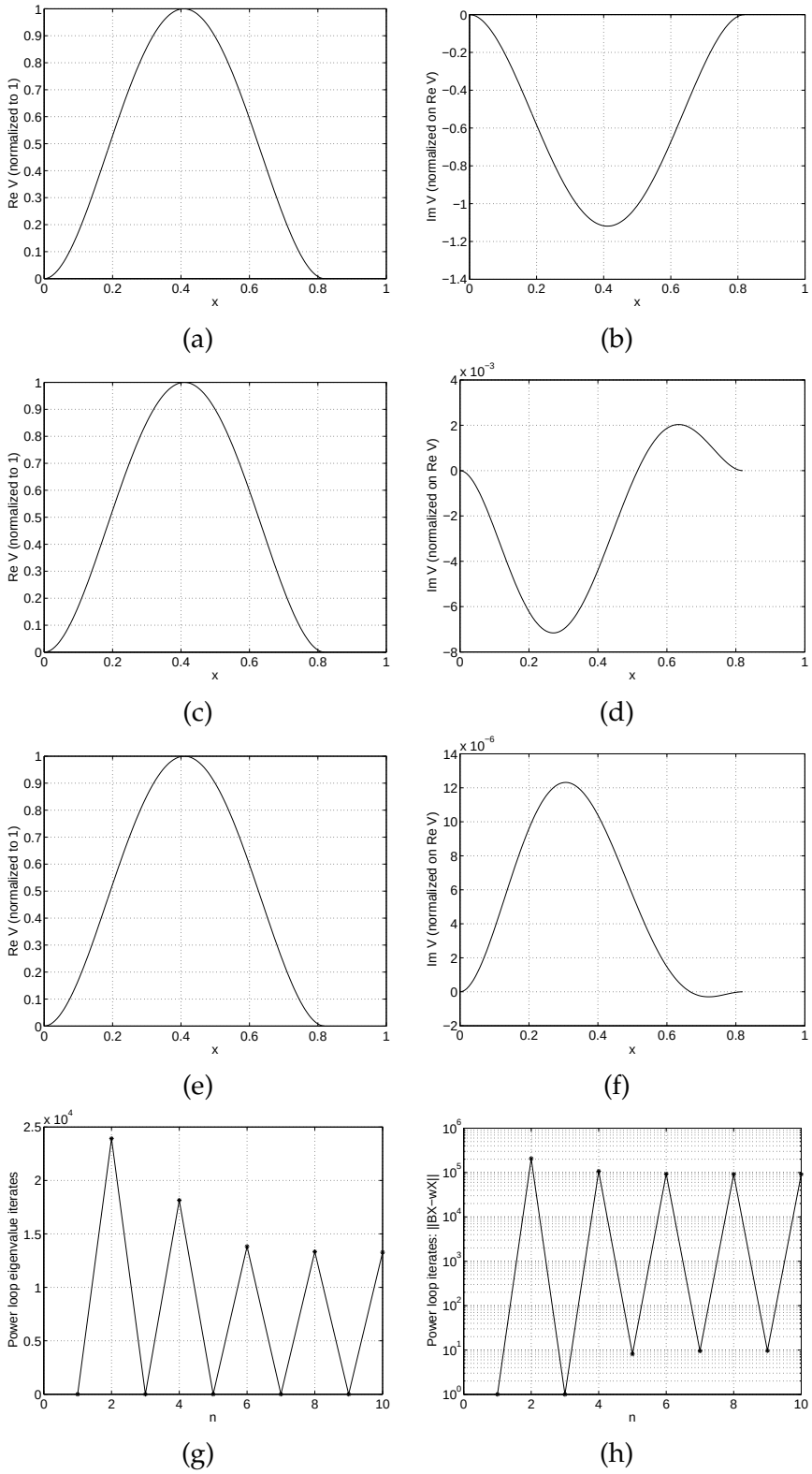

Figure 4: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 128 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

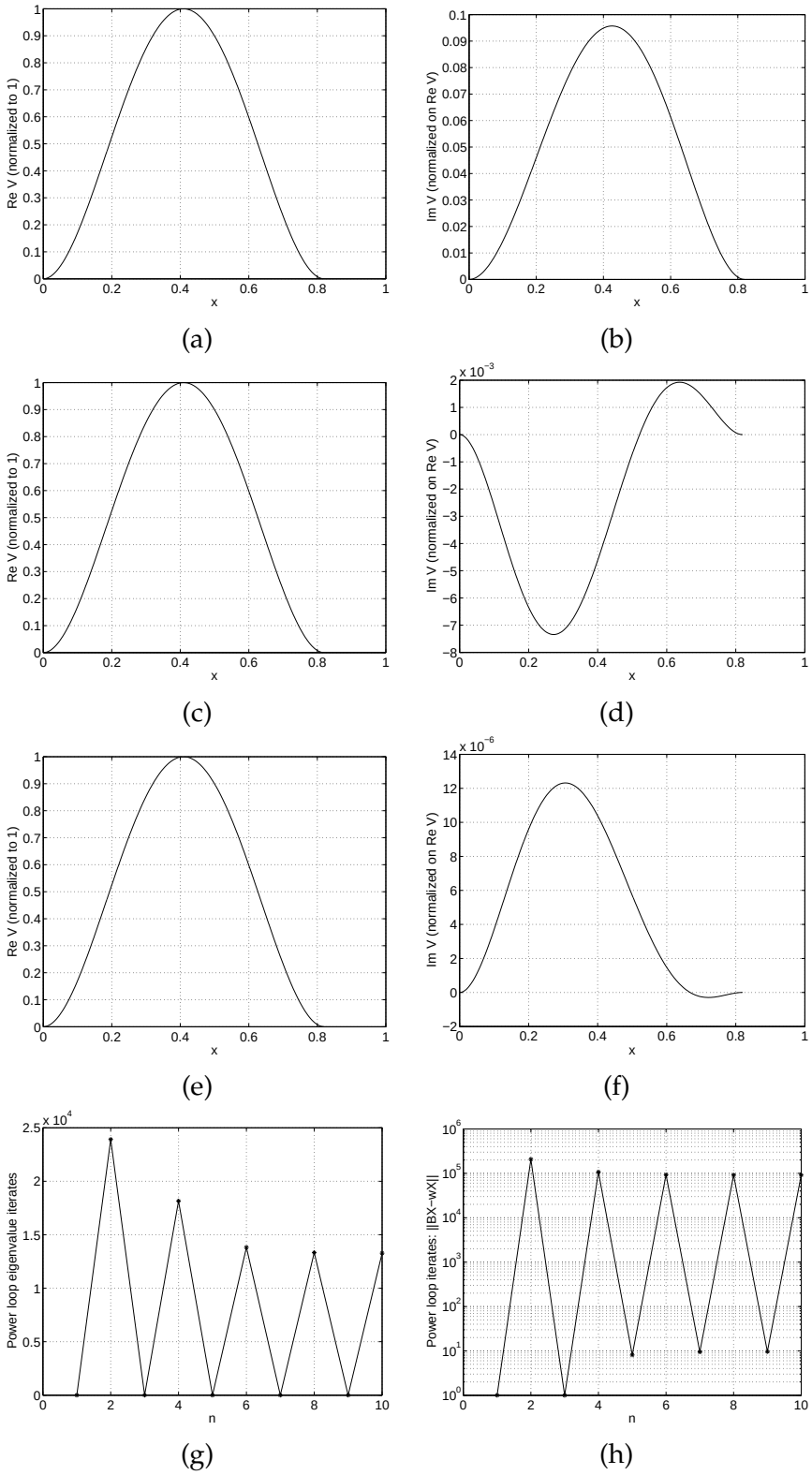

Figure 5: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 256 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

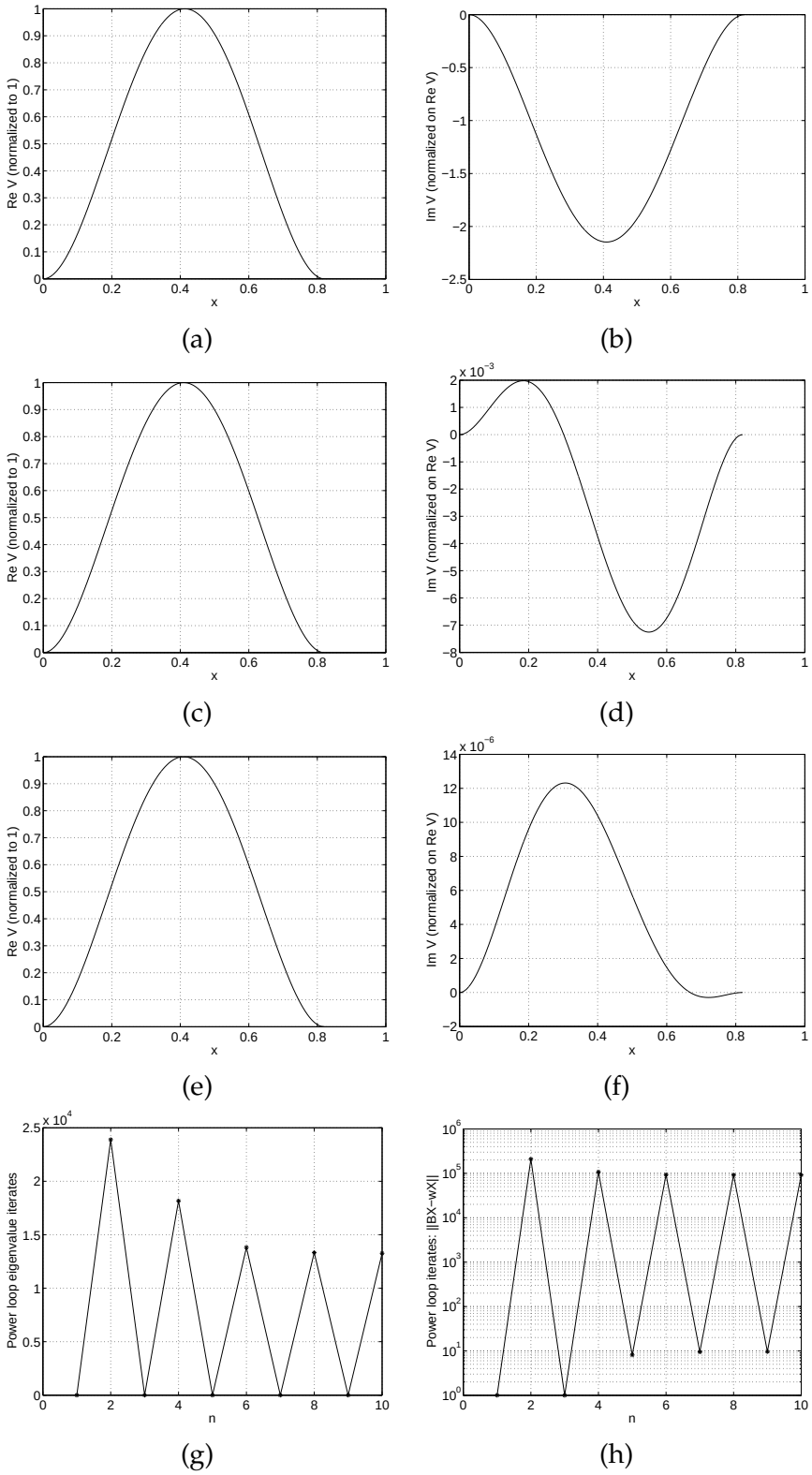

Figure 6: Computed eigen-results, without shift ($p = 0$), for the Euler-Bernoulli beam with 512 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

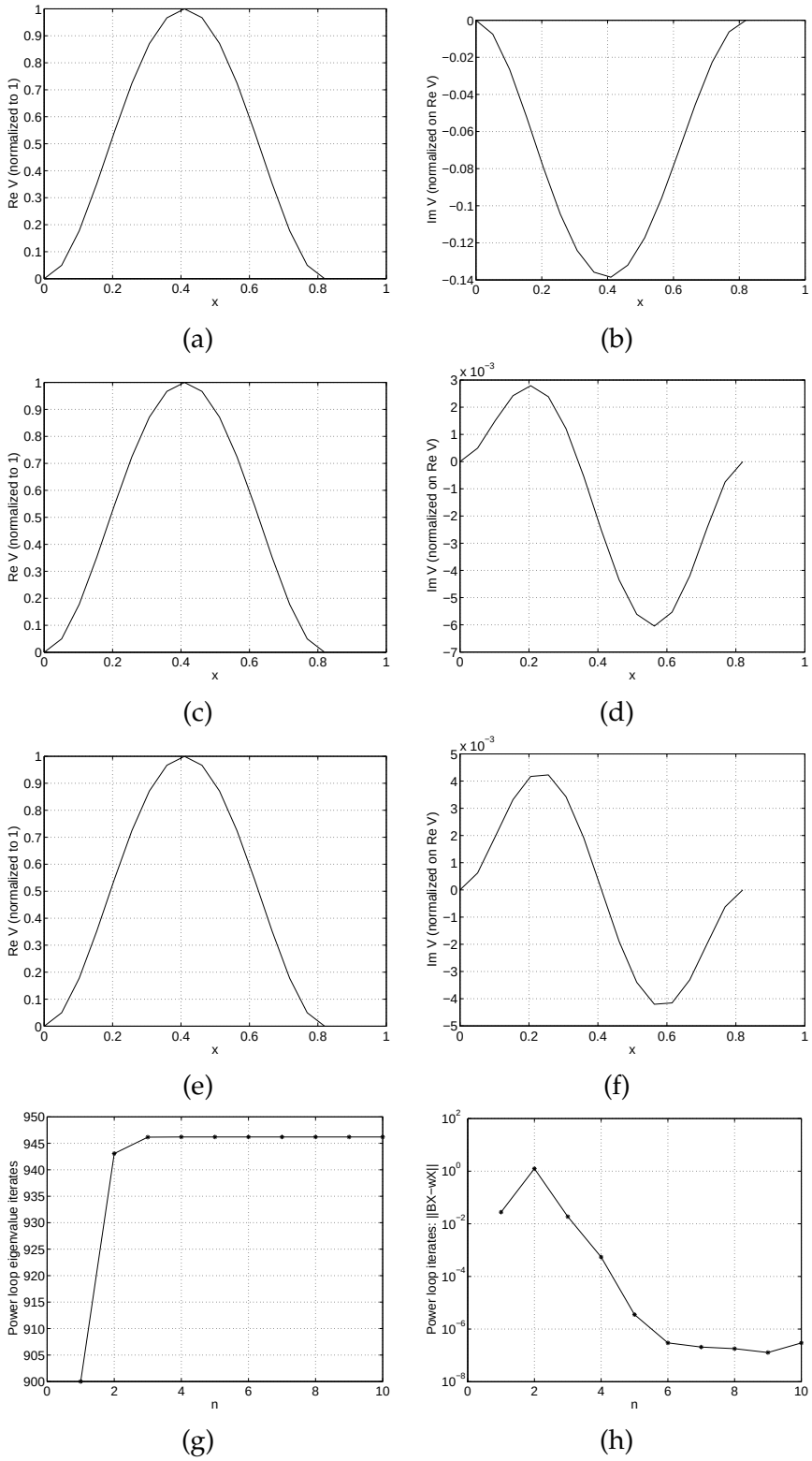

Figure 7: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 16 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

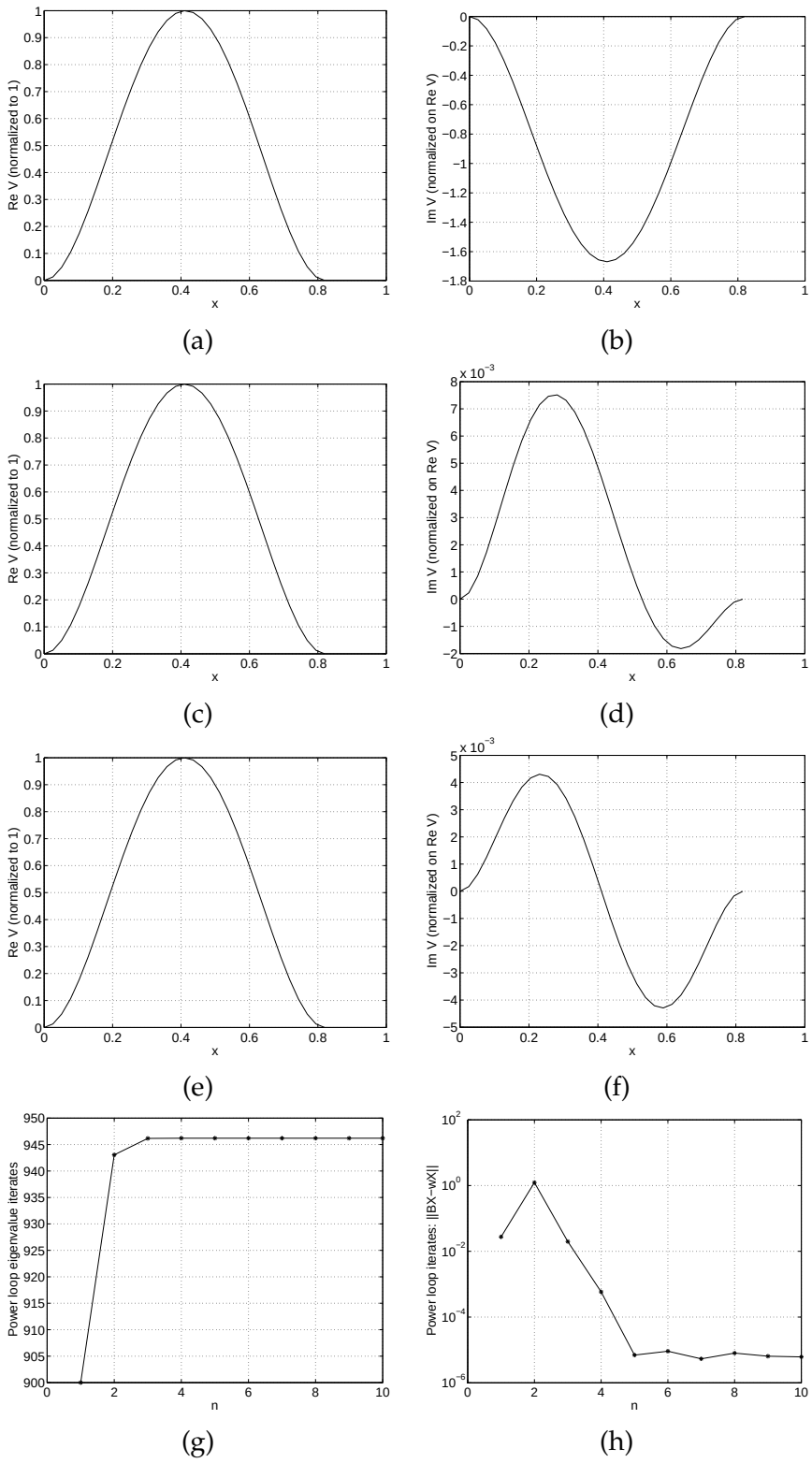

Figure 8: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 32 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

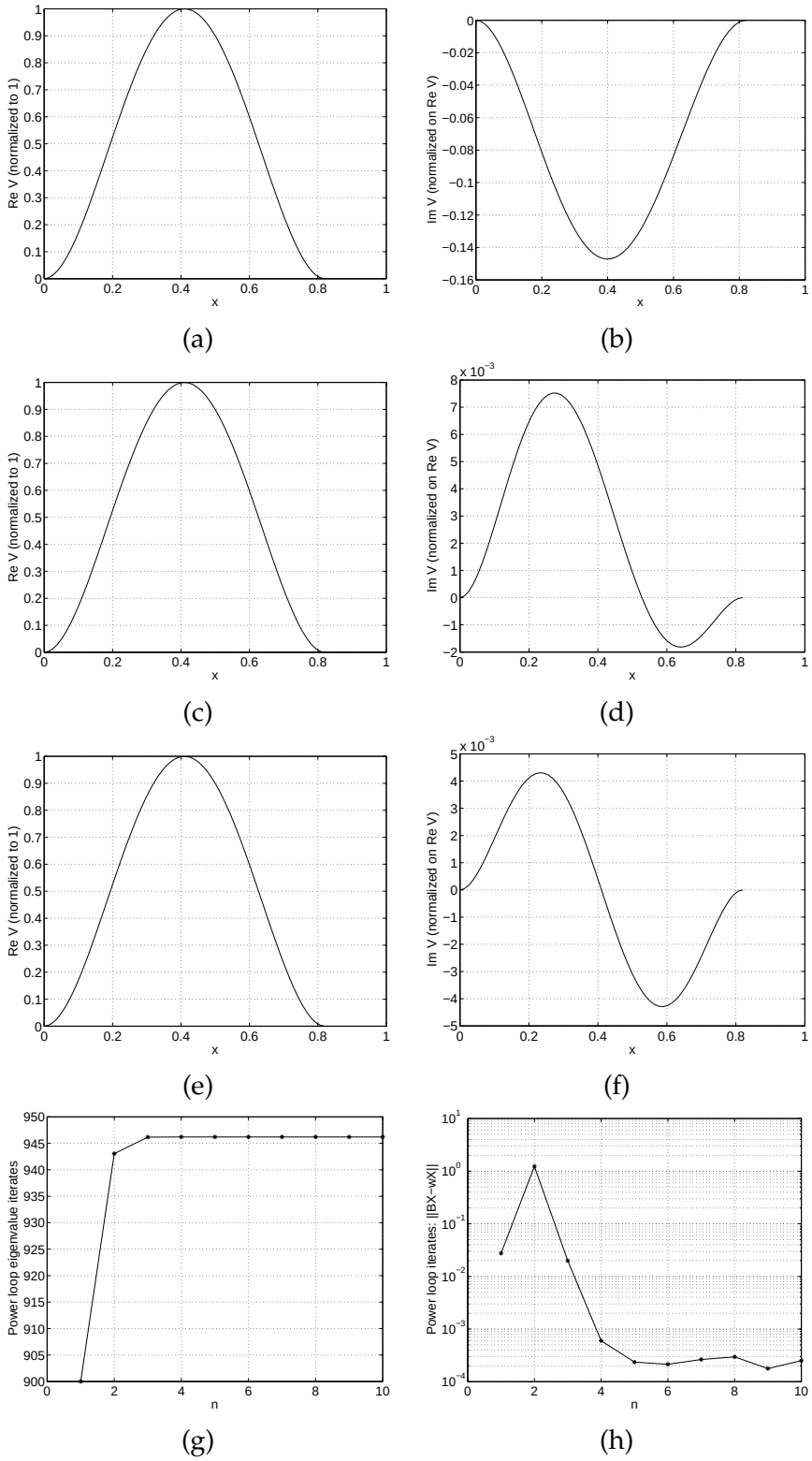

Figure 9: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 64 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

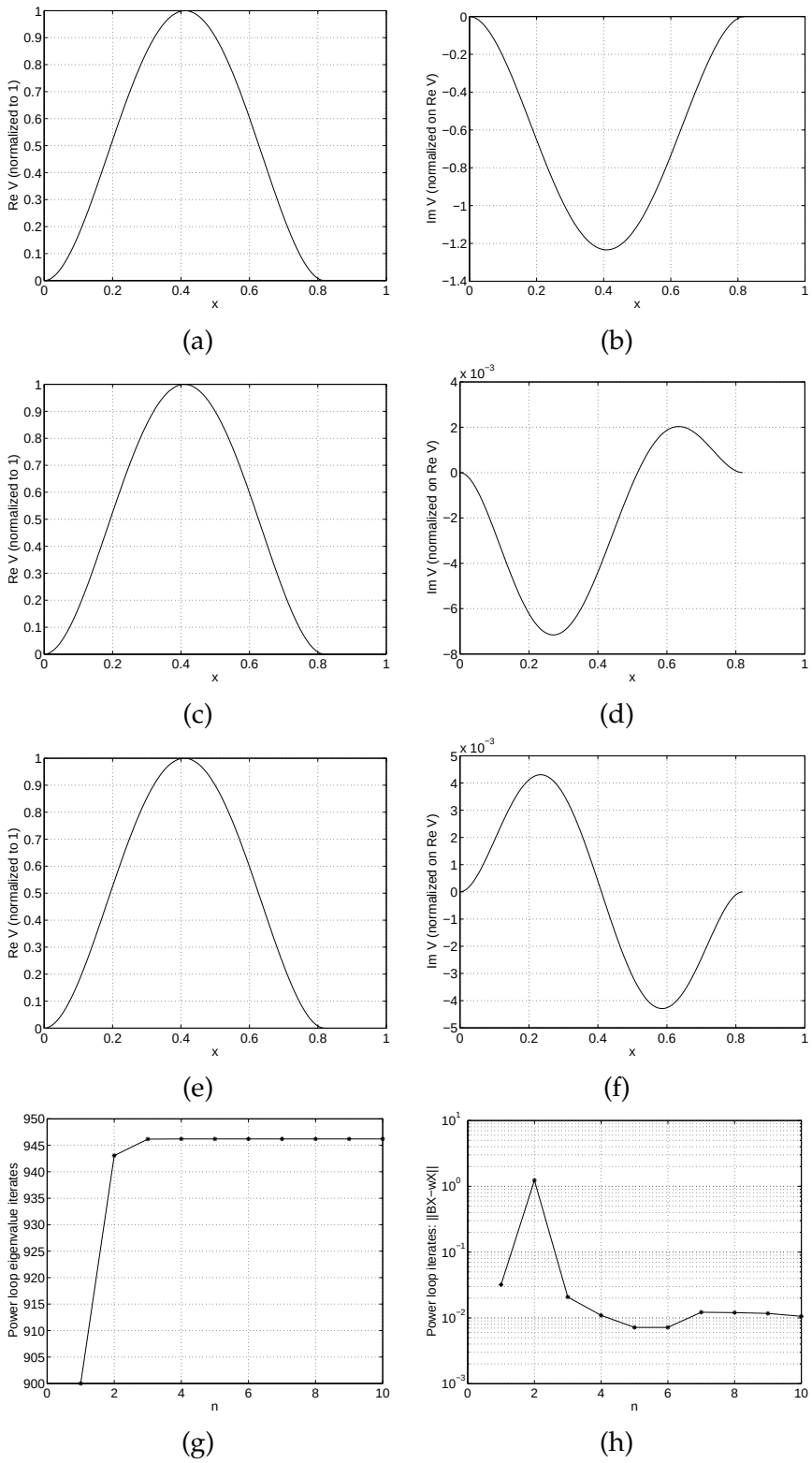

Figure 10: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 128 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

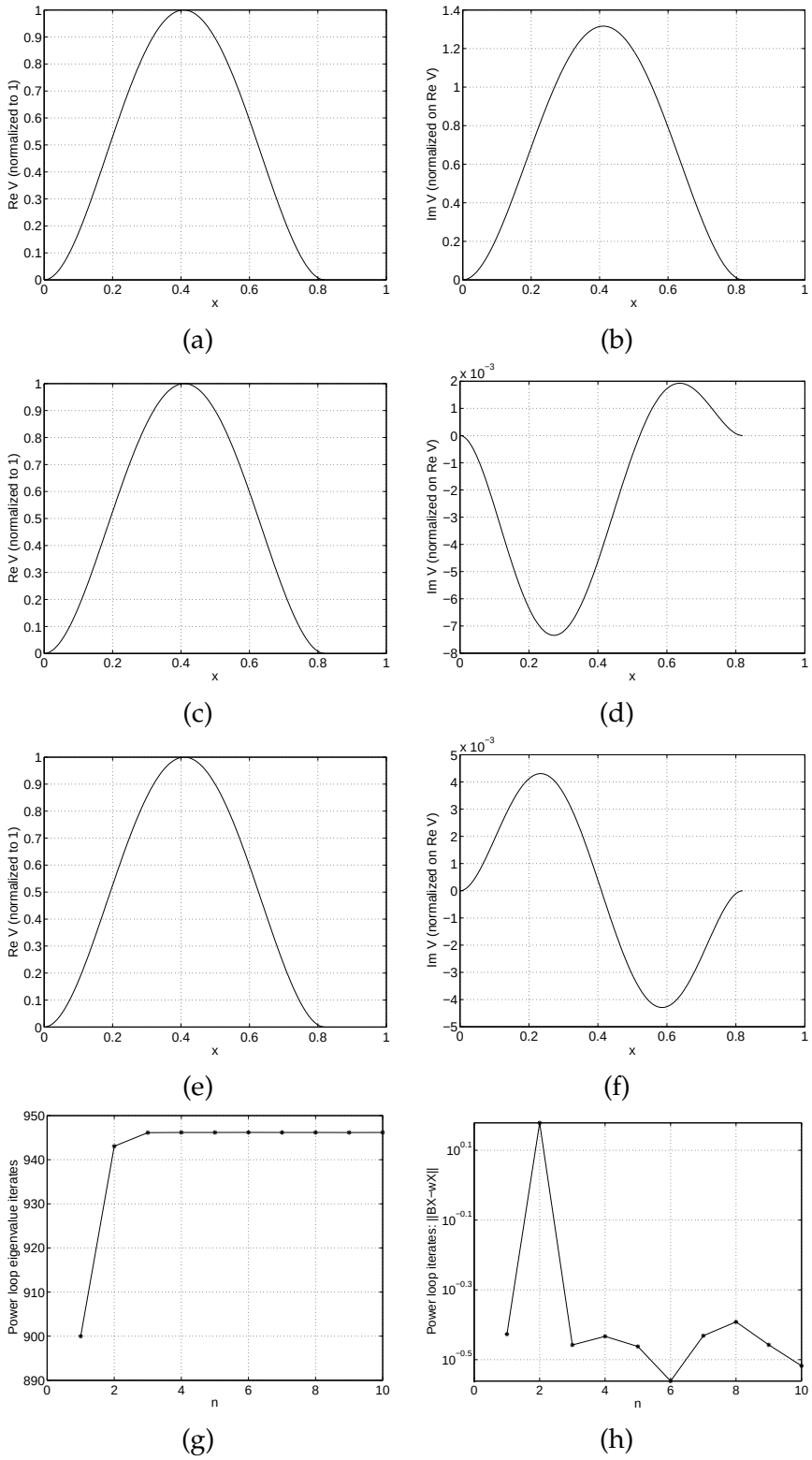

Figure 11: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 256 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

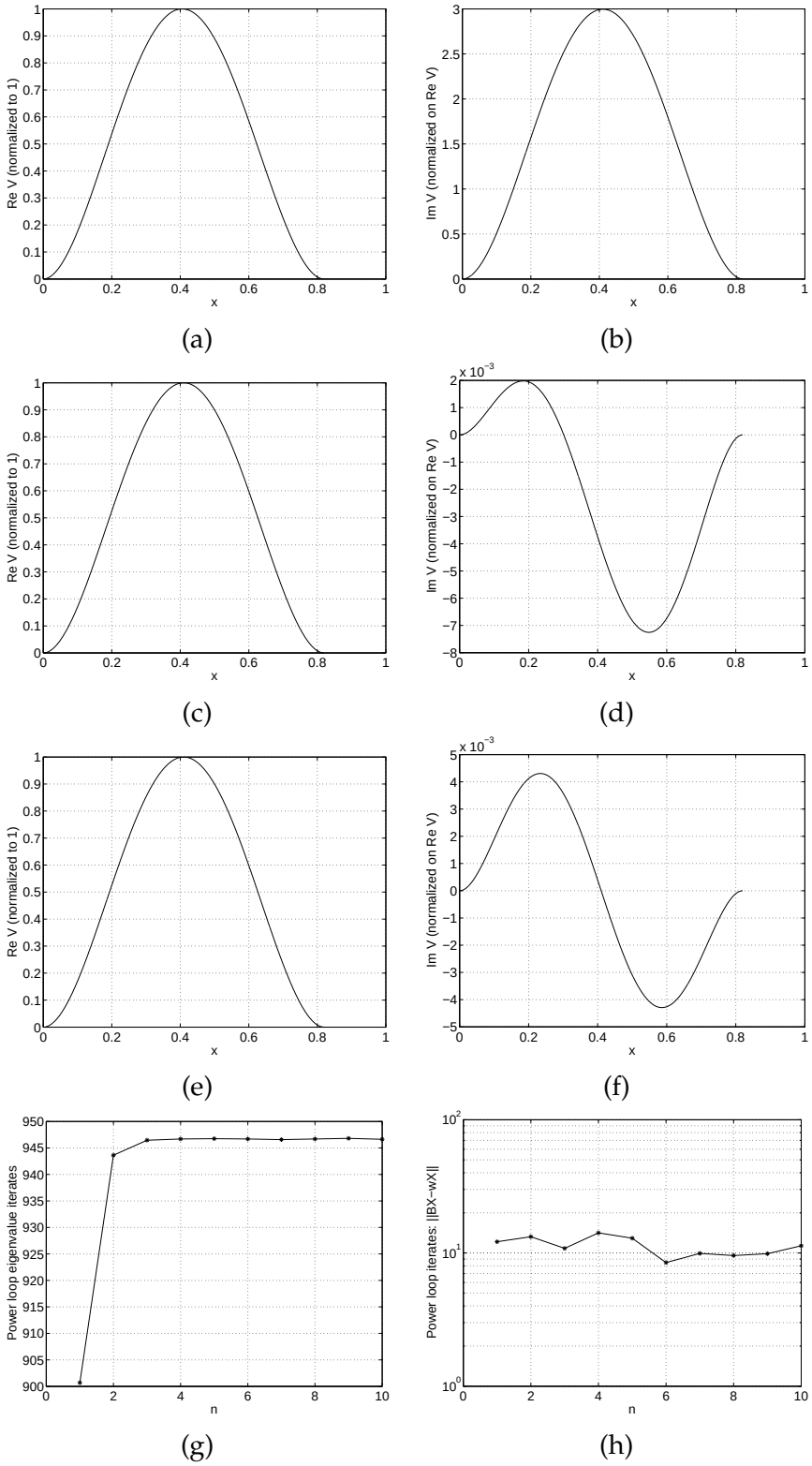

Figure 12: Computed eigen-results, with shift $p = 900$, for the Euler-Bernoulli beam with 512 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

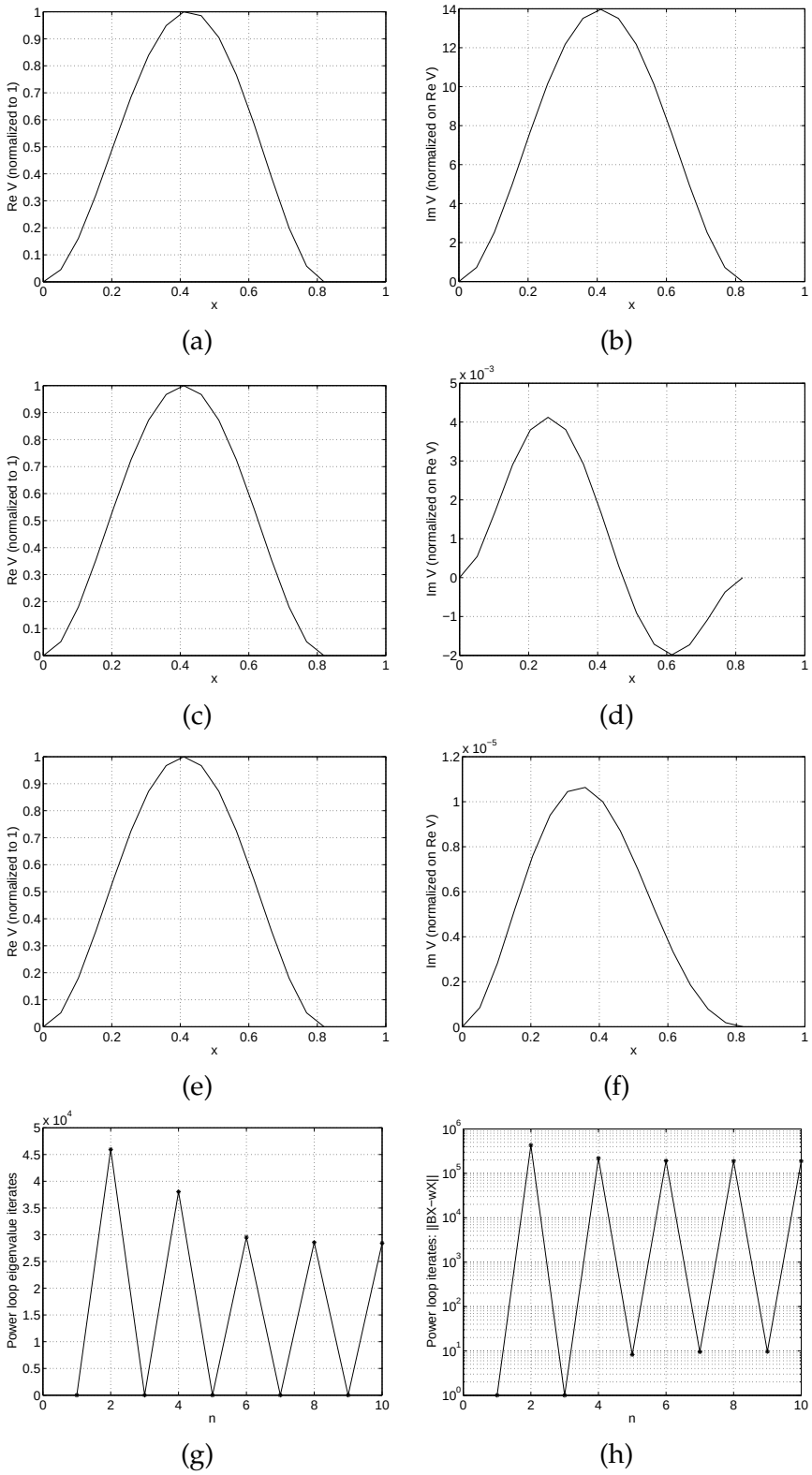

Figure 13: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 16 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

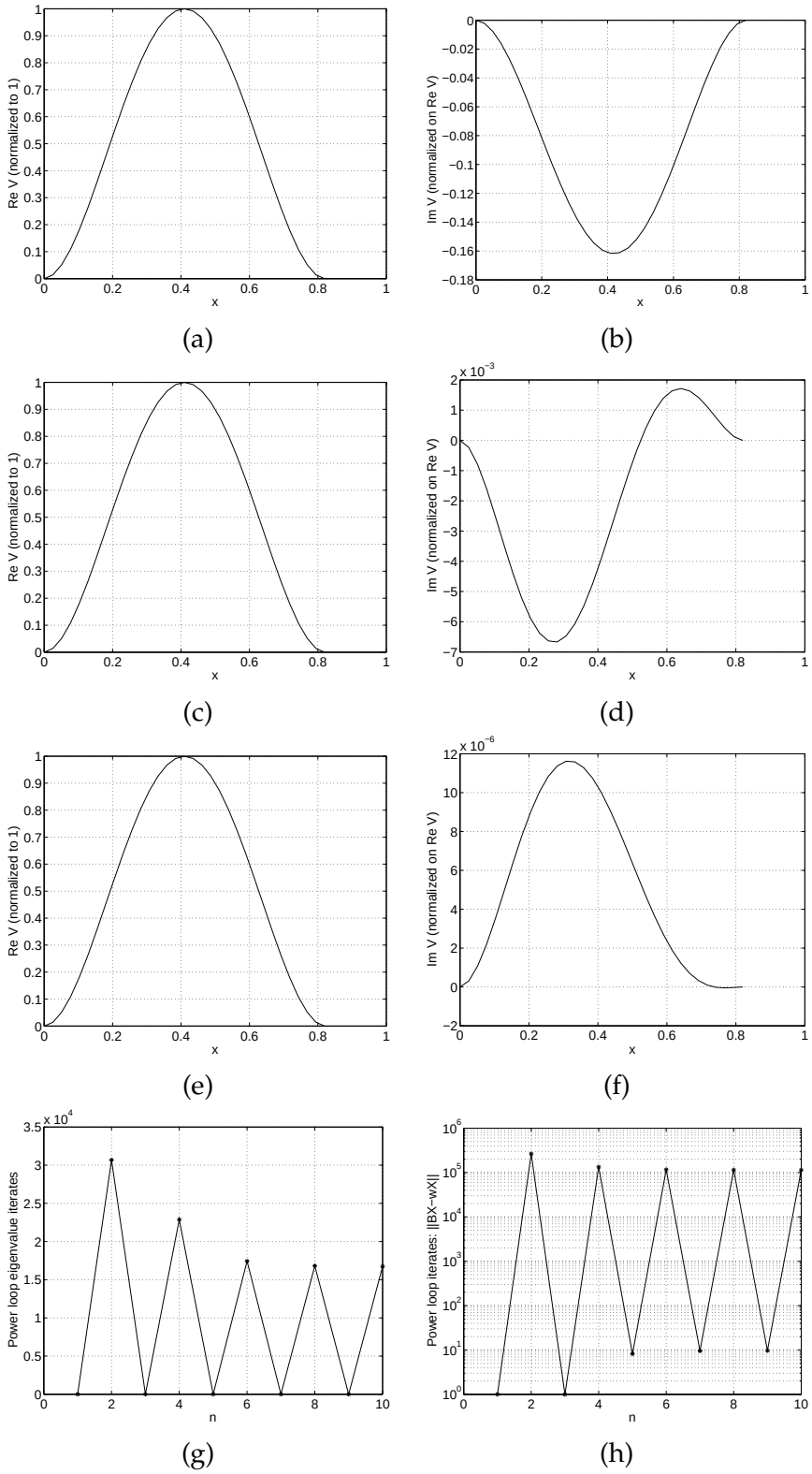

Figure 14: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 32 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

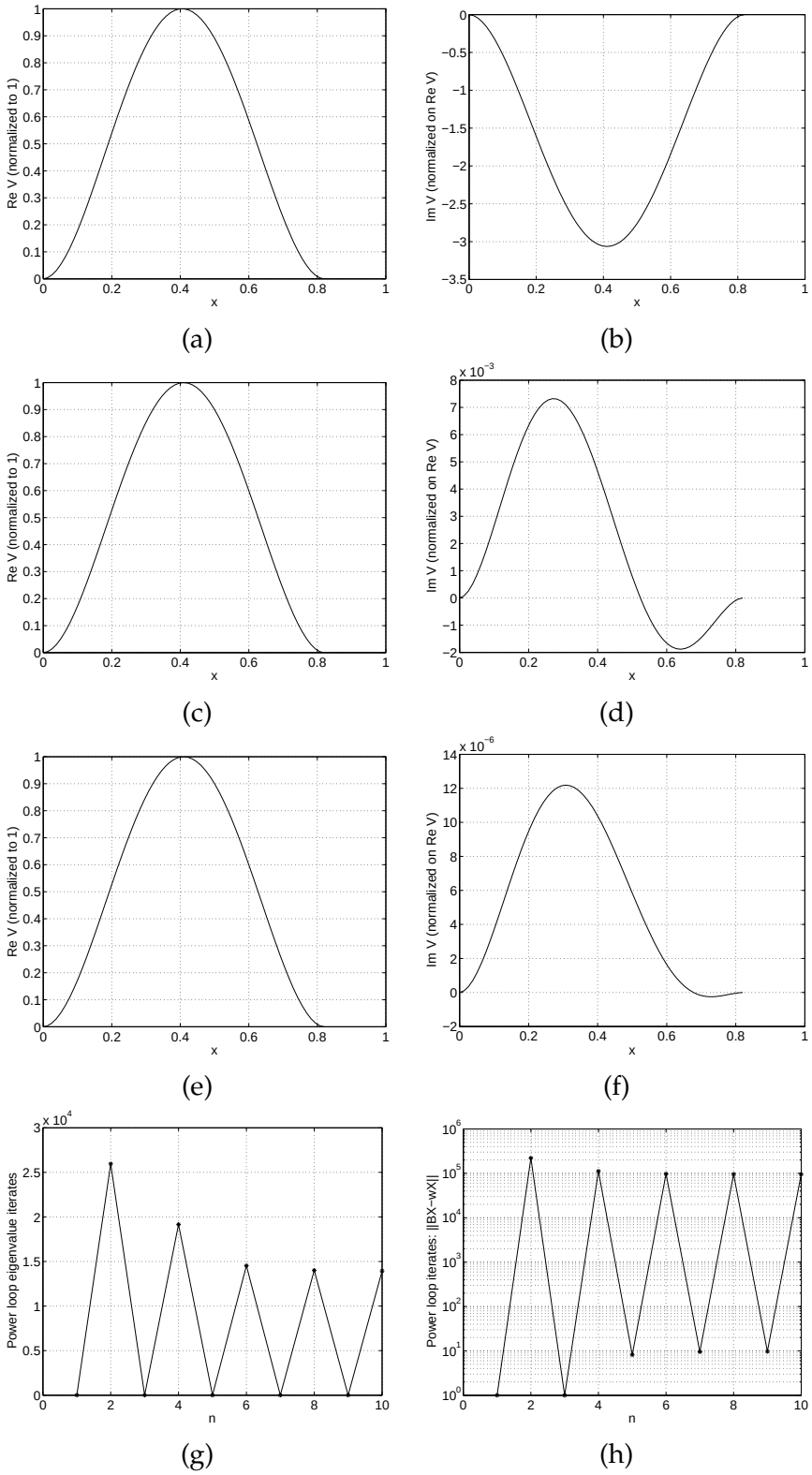

Figure 15: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 64 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

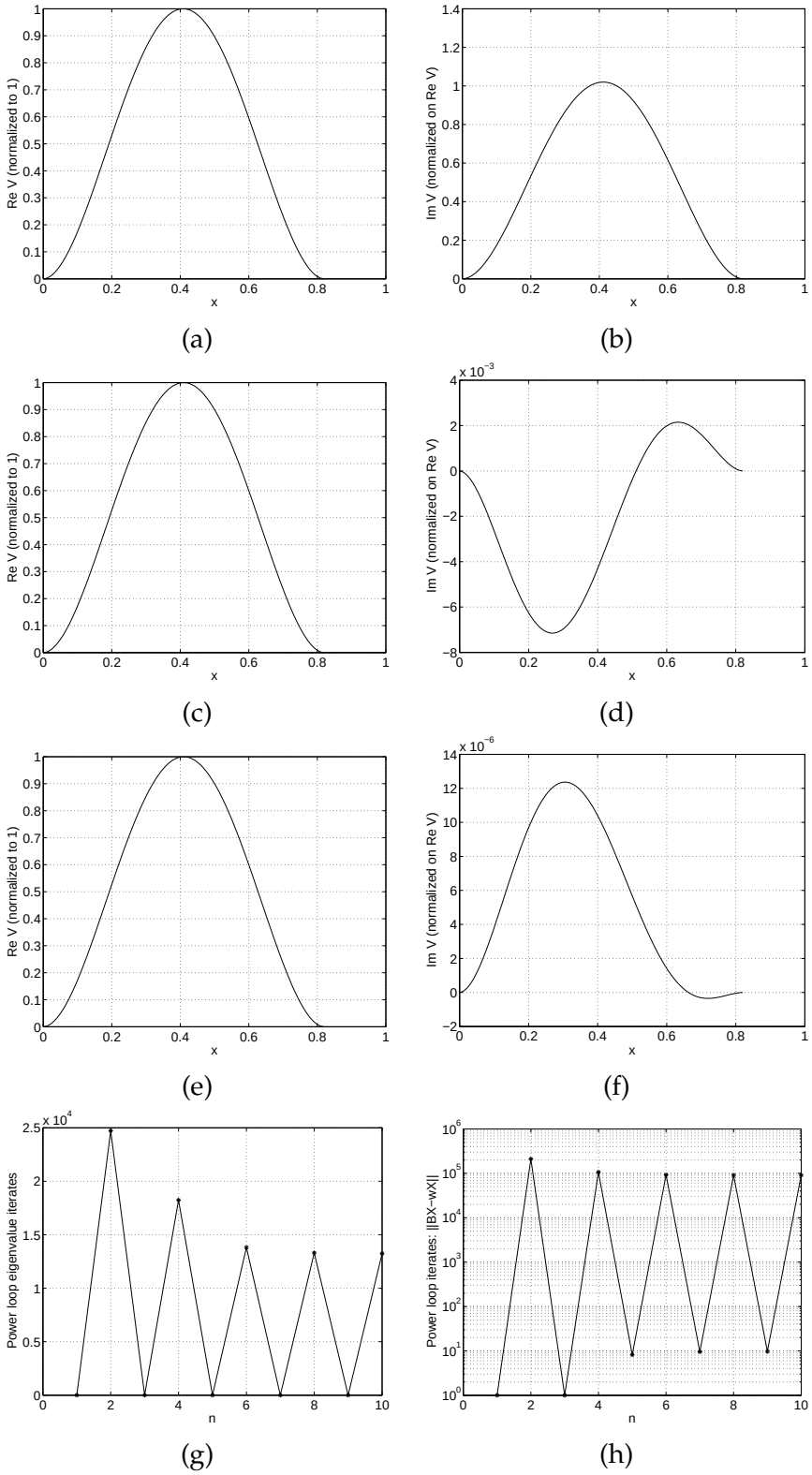

Figure 16: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 128 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

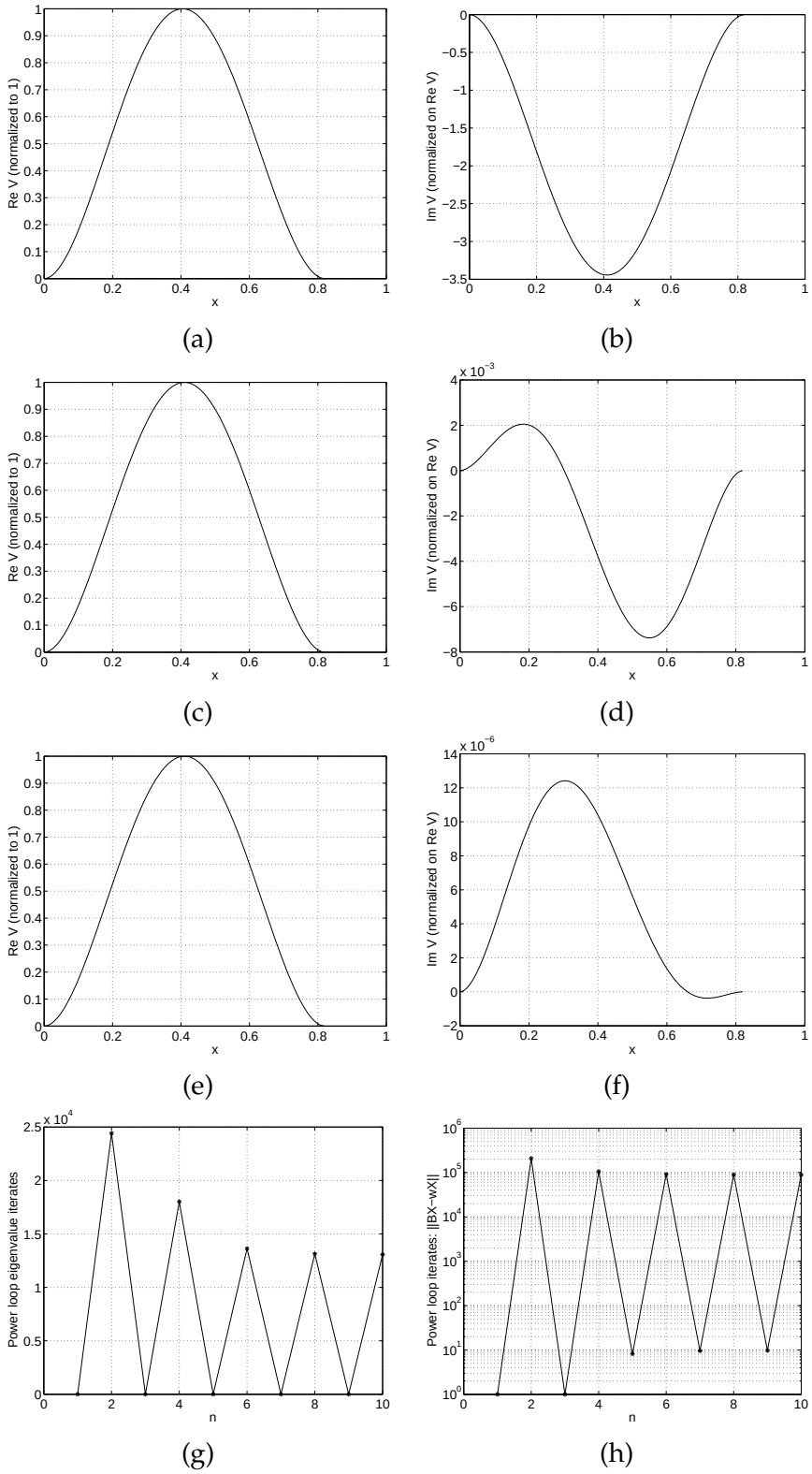

Figure 17: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 256 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

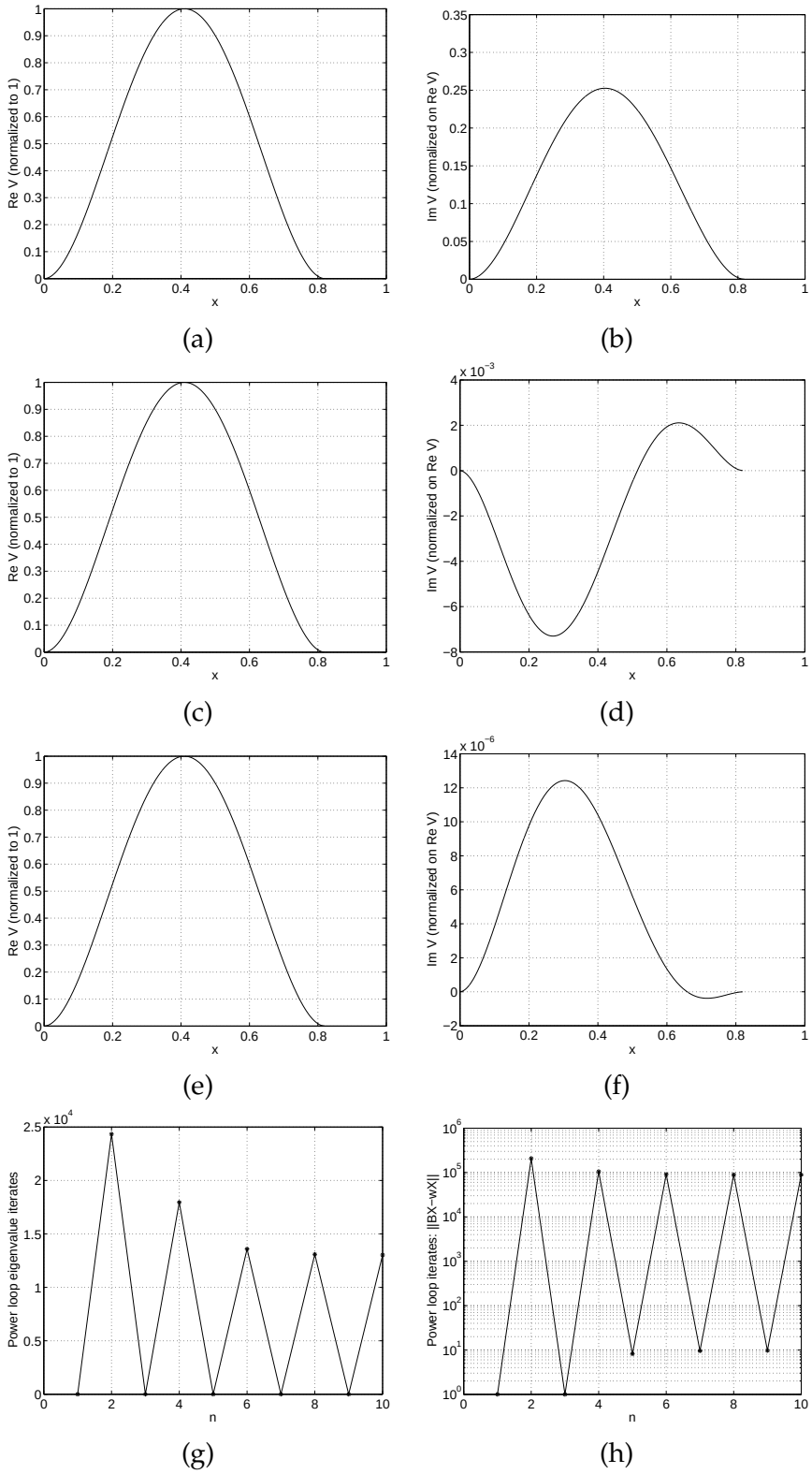

Figure 18: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 512 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

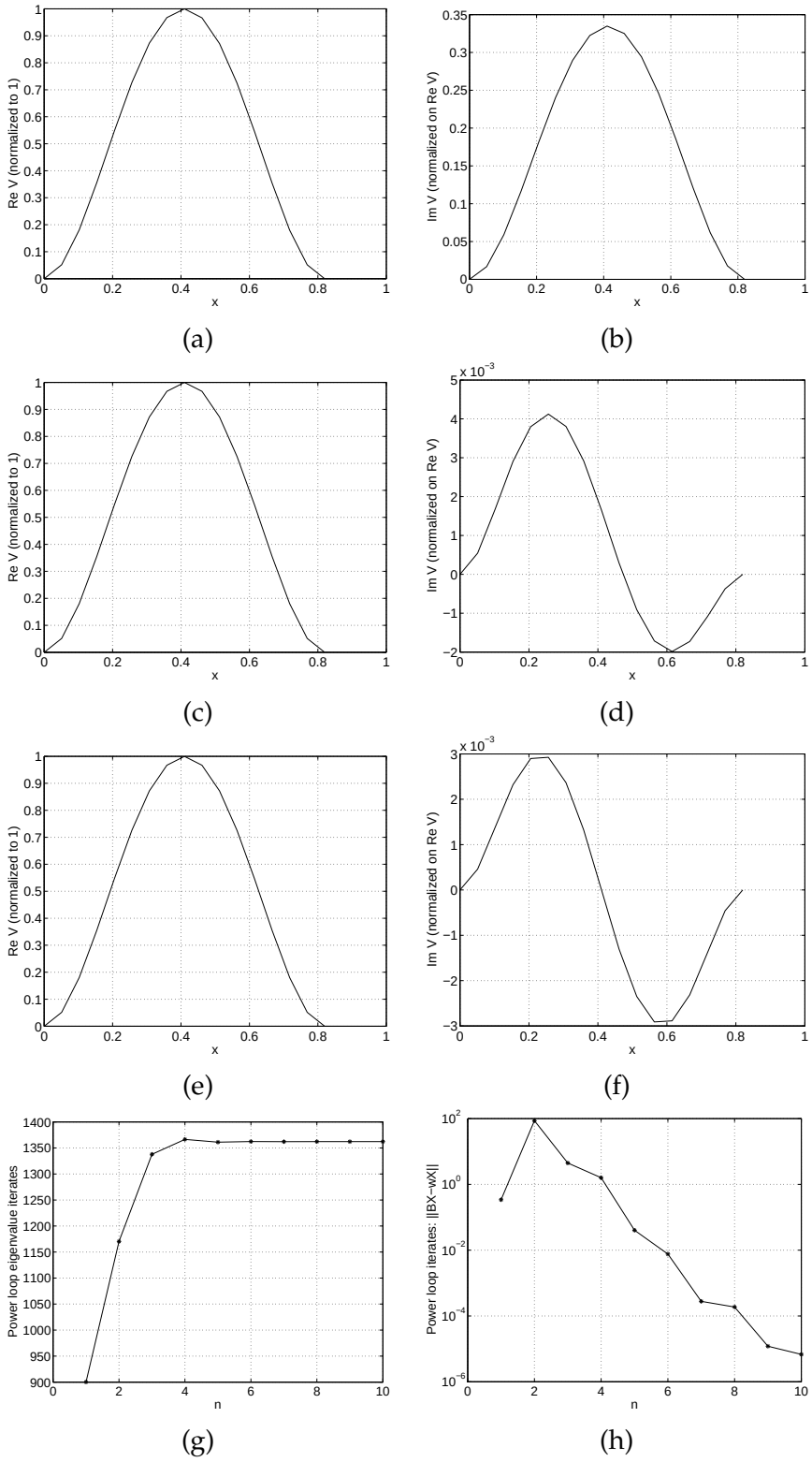

Figure 19: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 16 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

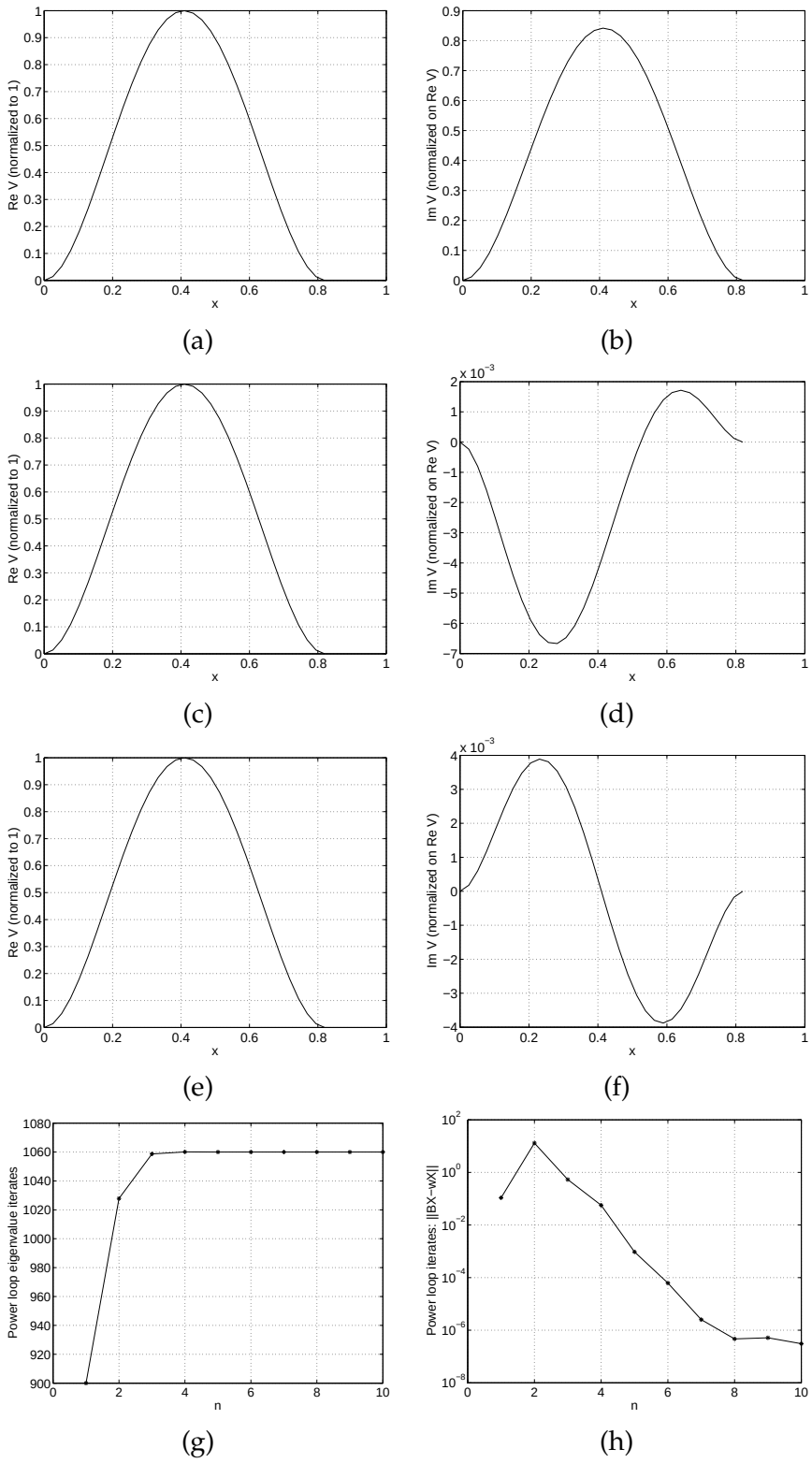

Figure 20: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 32 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

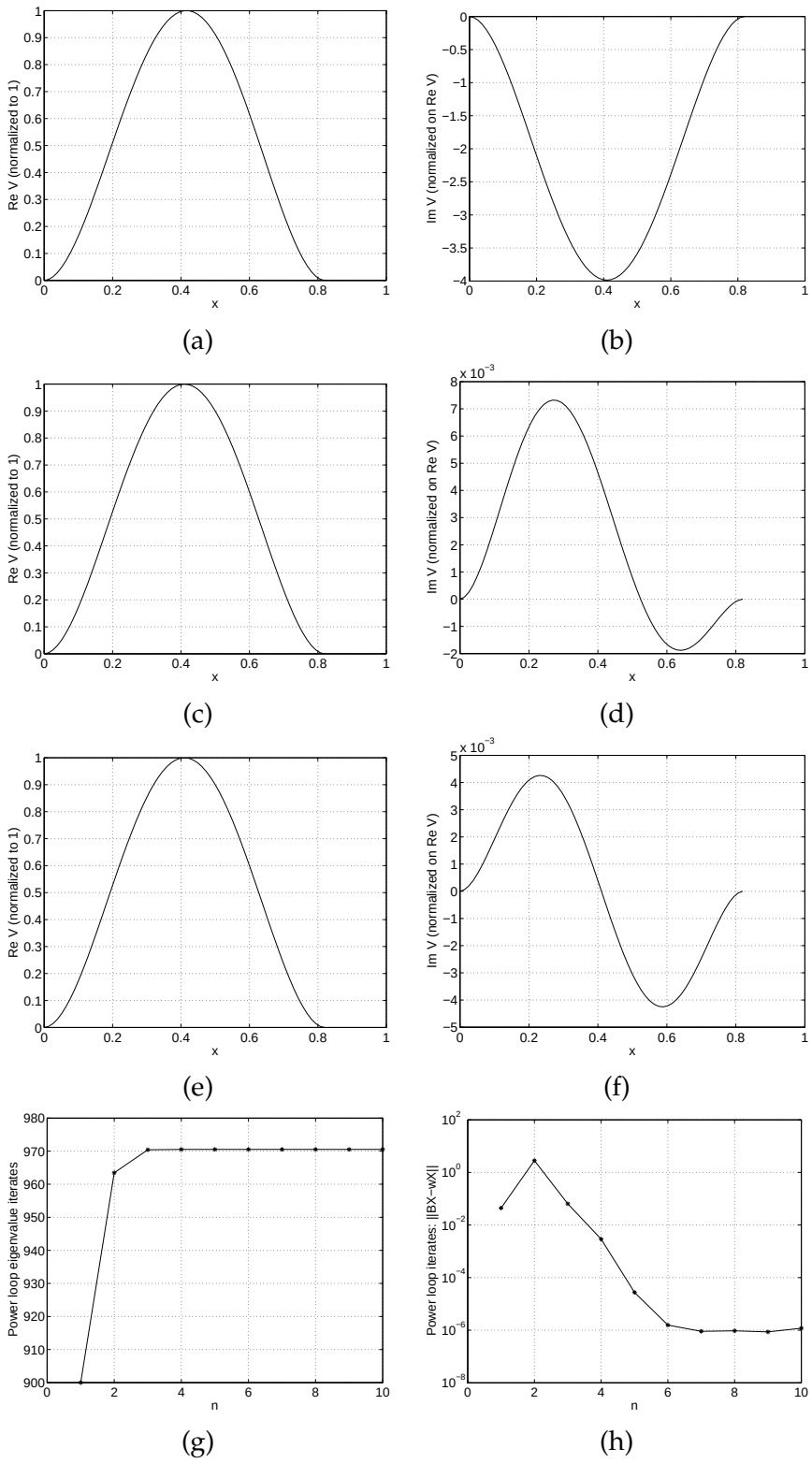

Figure 21: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 64 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

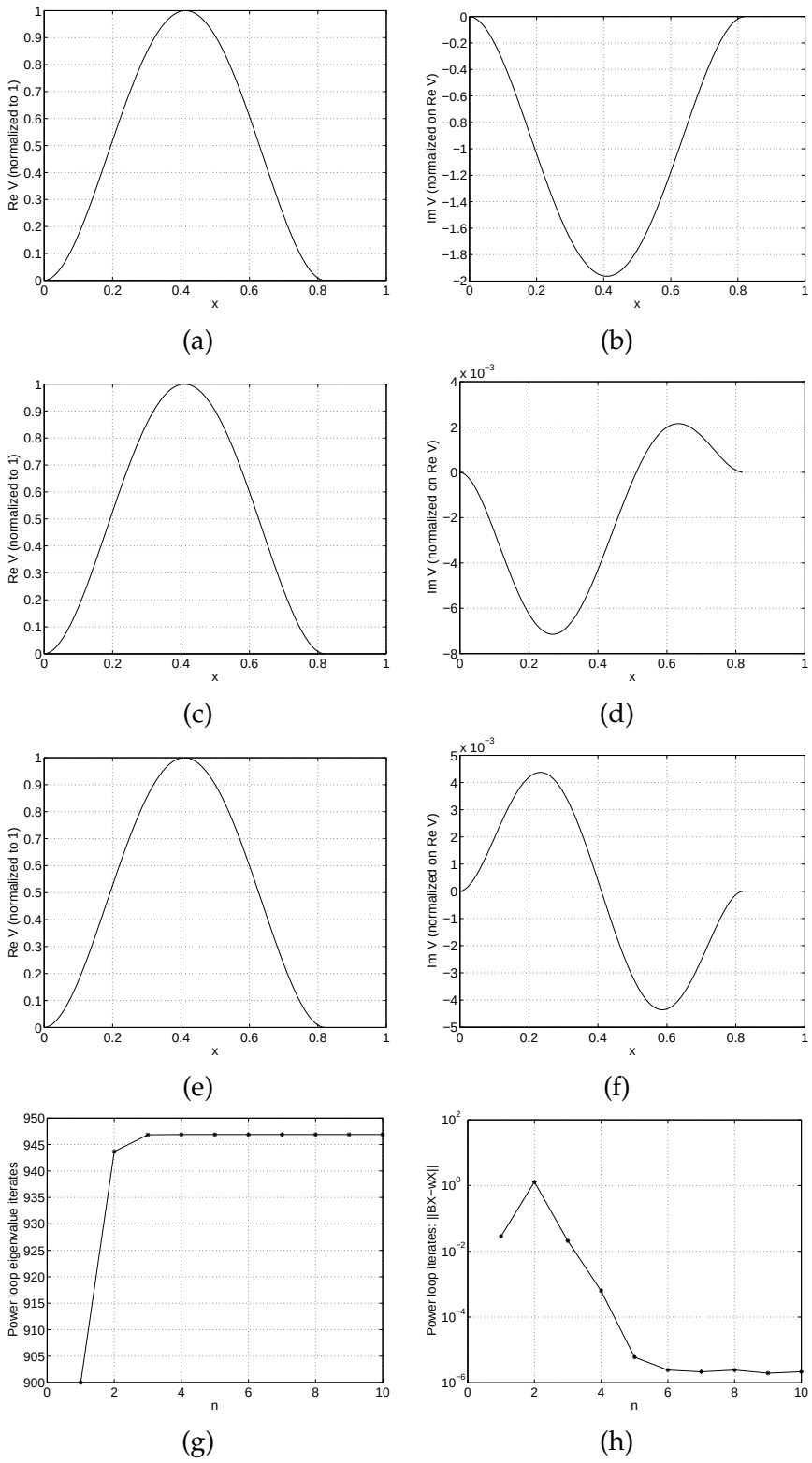

Figure 22: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 128 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

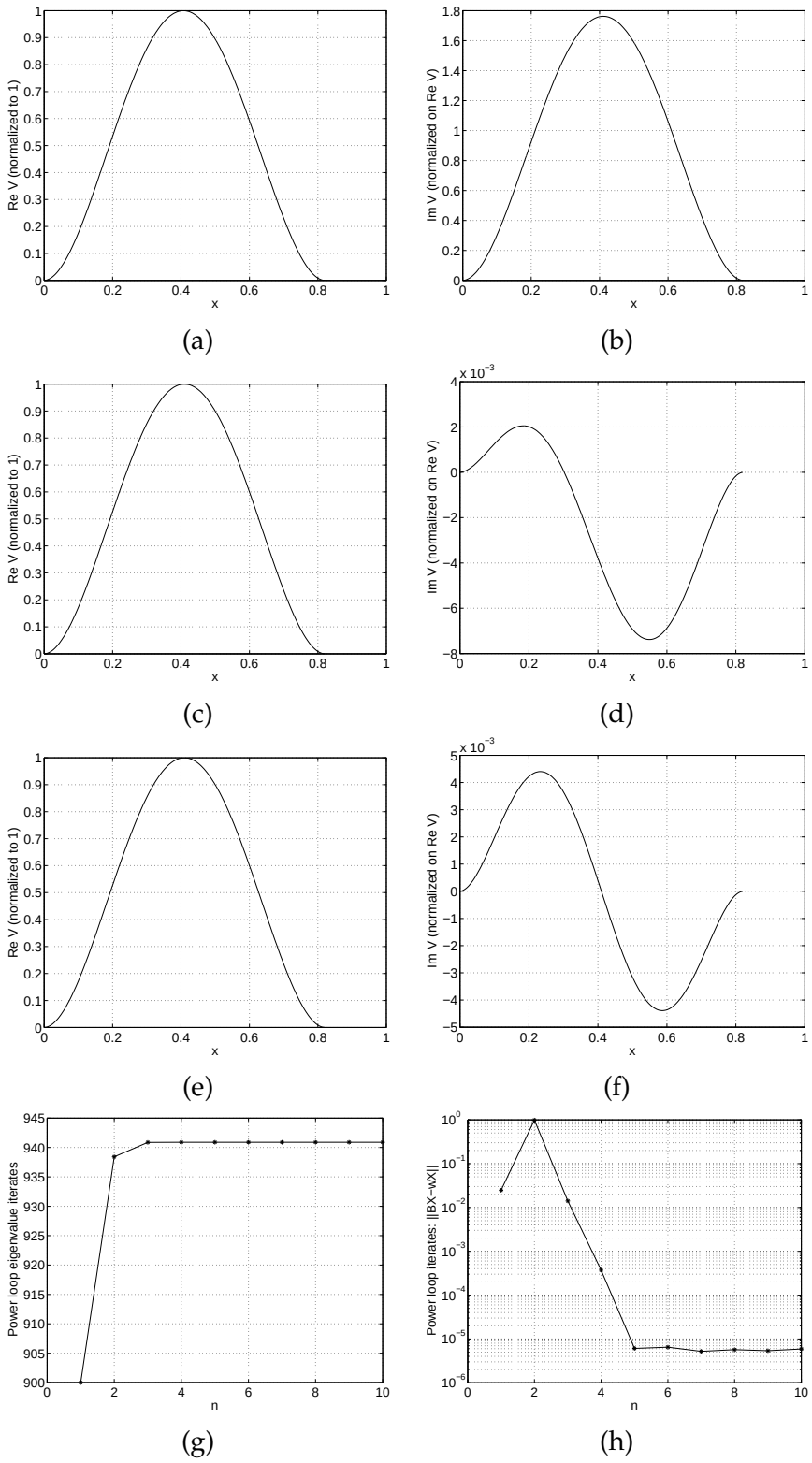

Figure 23: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 256 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

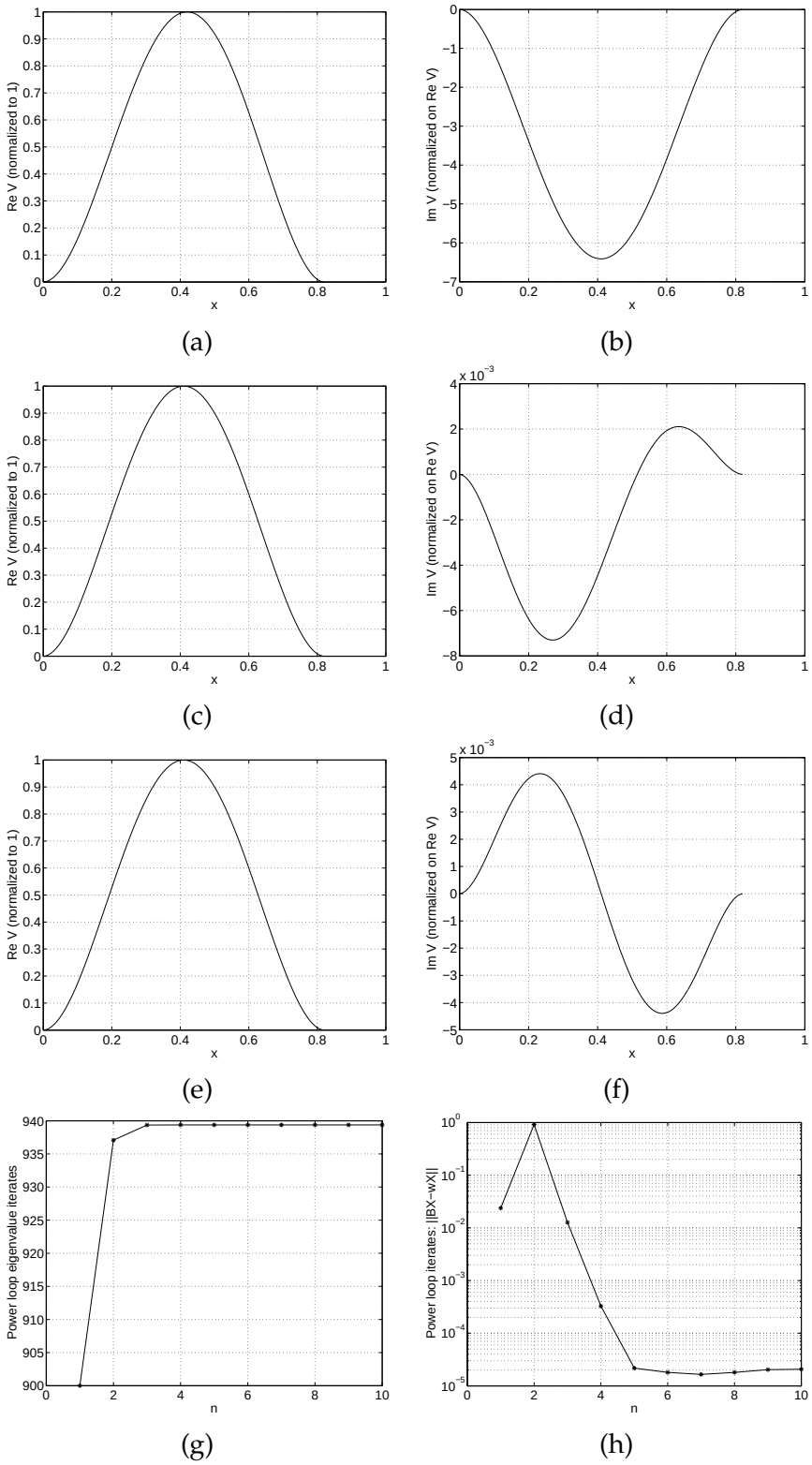

Figure 24: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 512 linear elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

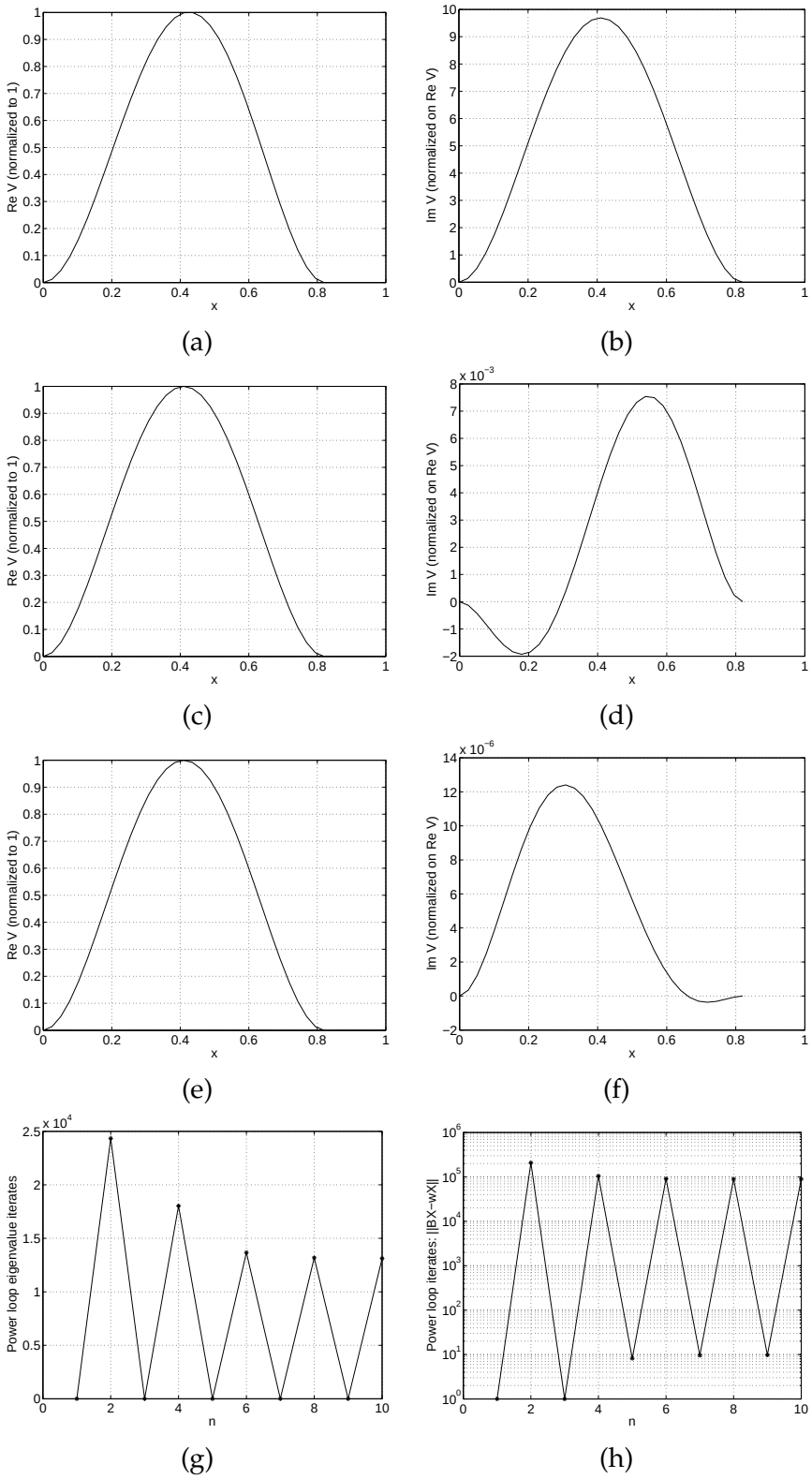

Figure 25: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 16 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

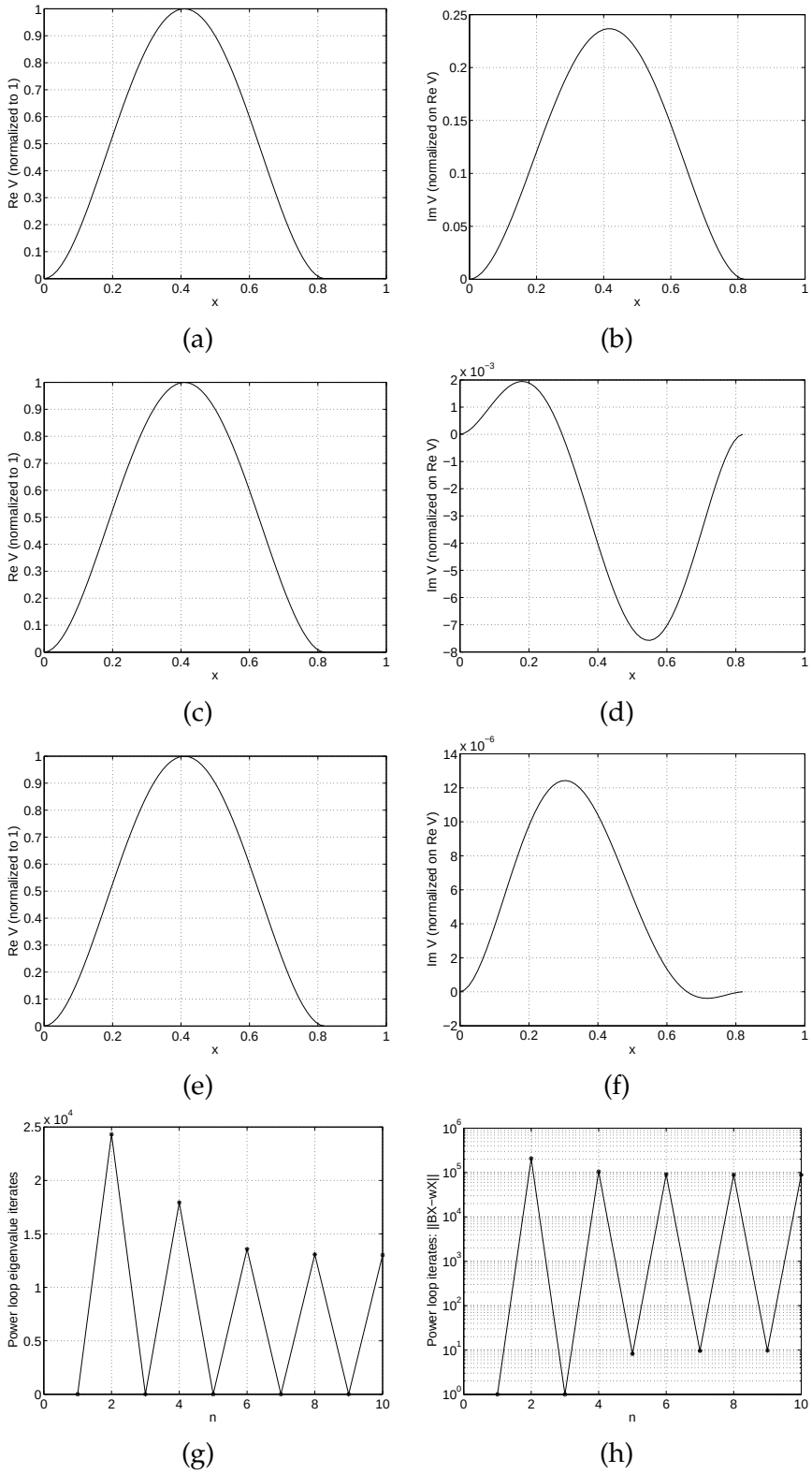

Figure 26: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 32 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

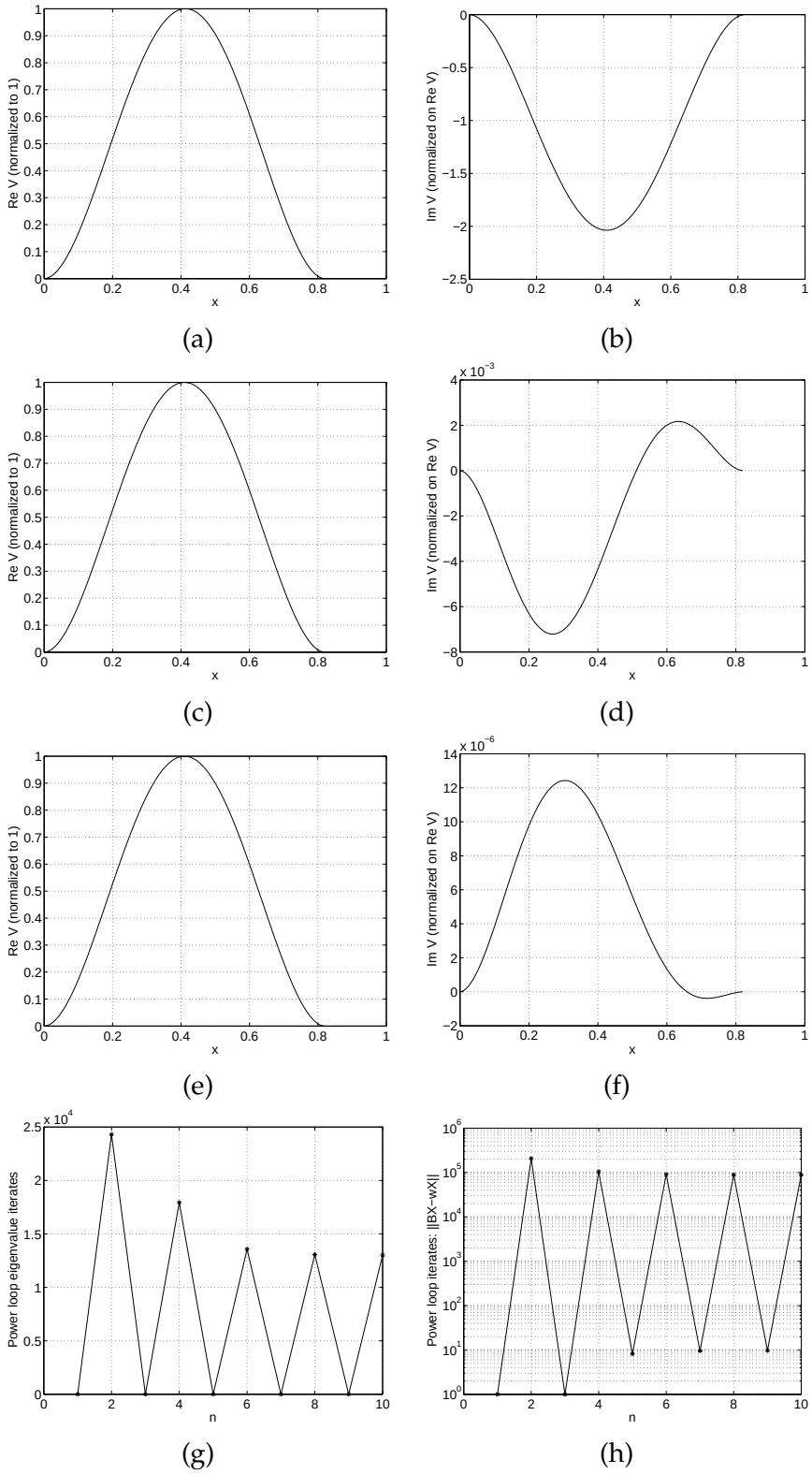

Figure 27: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 64 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

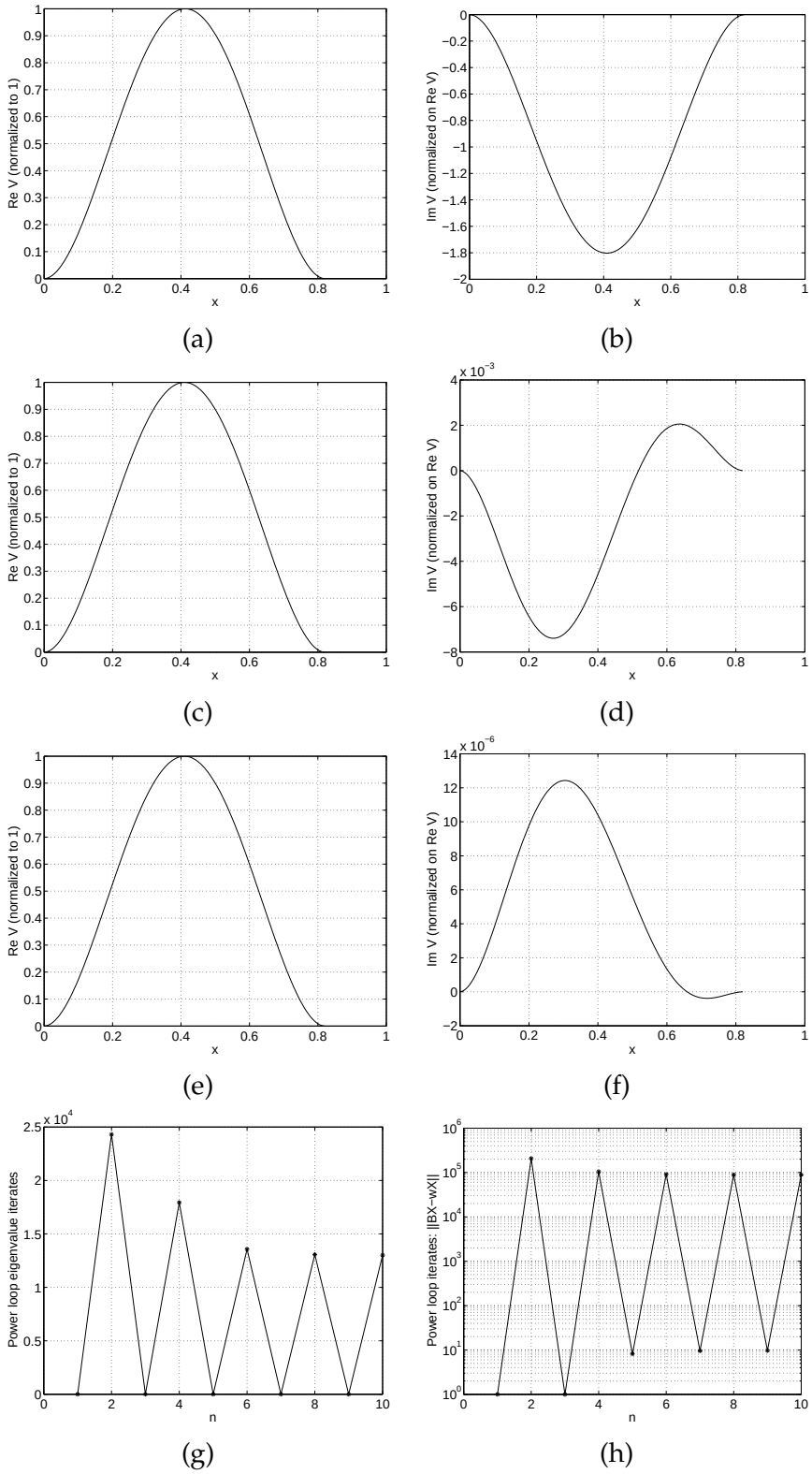

Figure 28: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 128 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

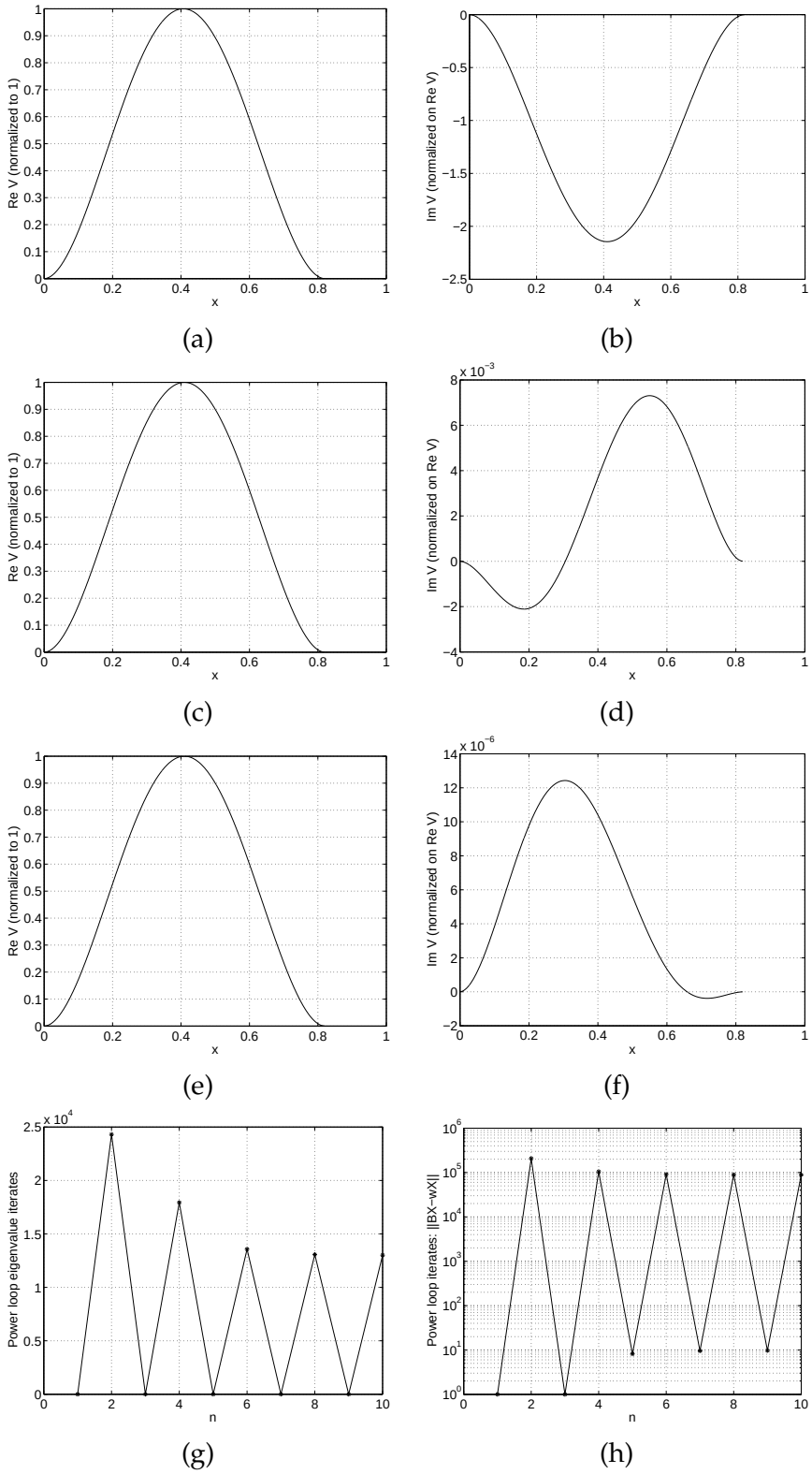

Figure 29: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 256 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

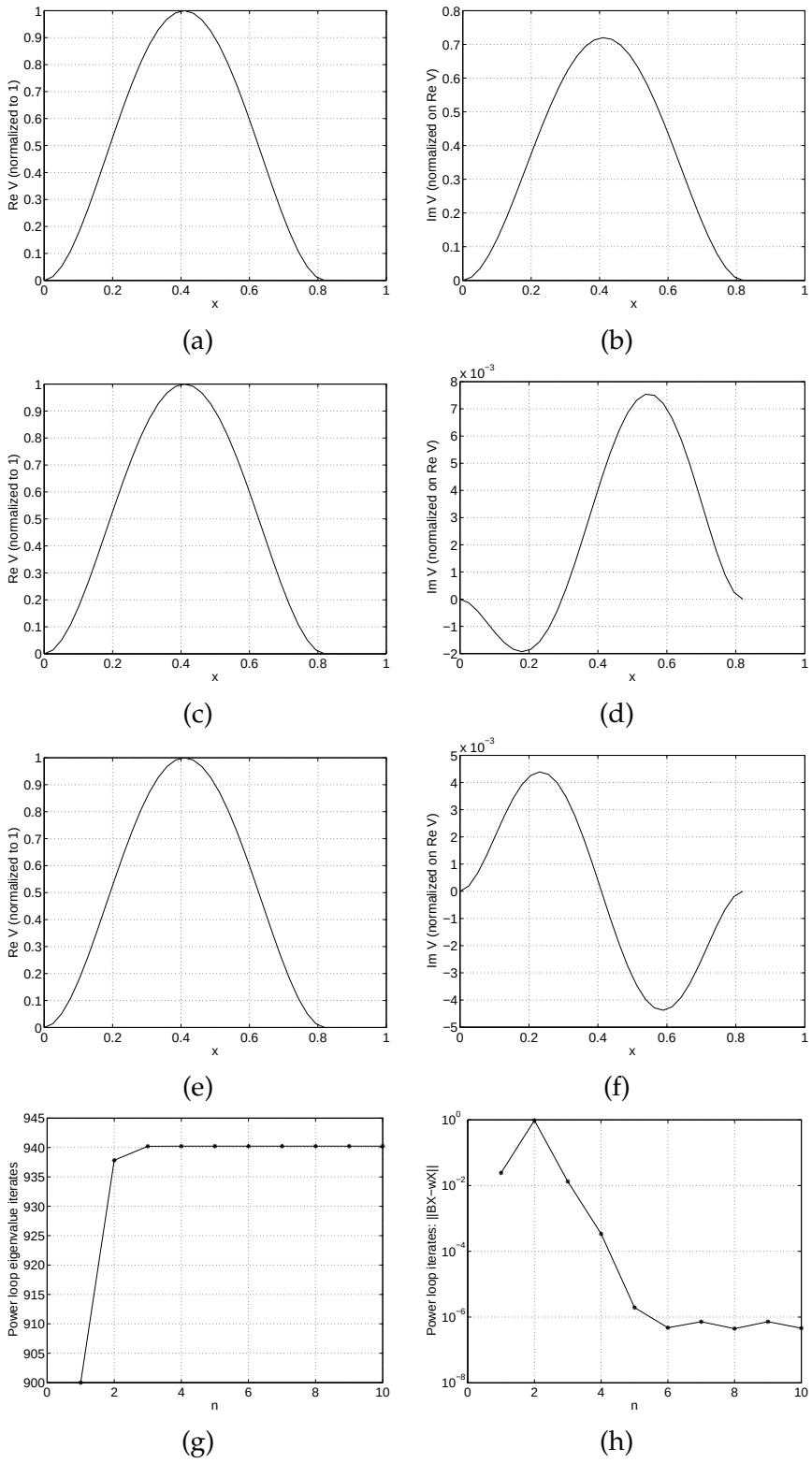

Figure 30: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 16 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

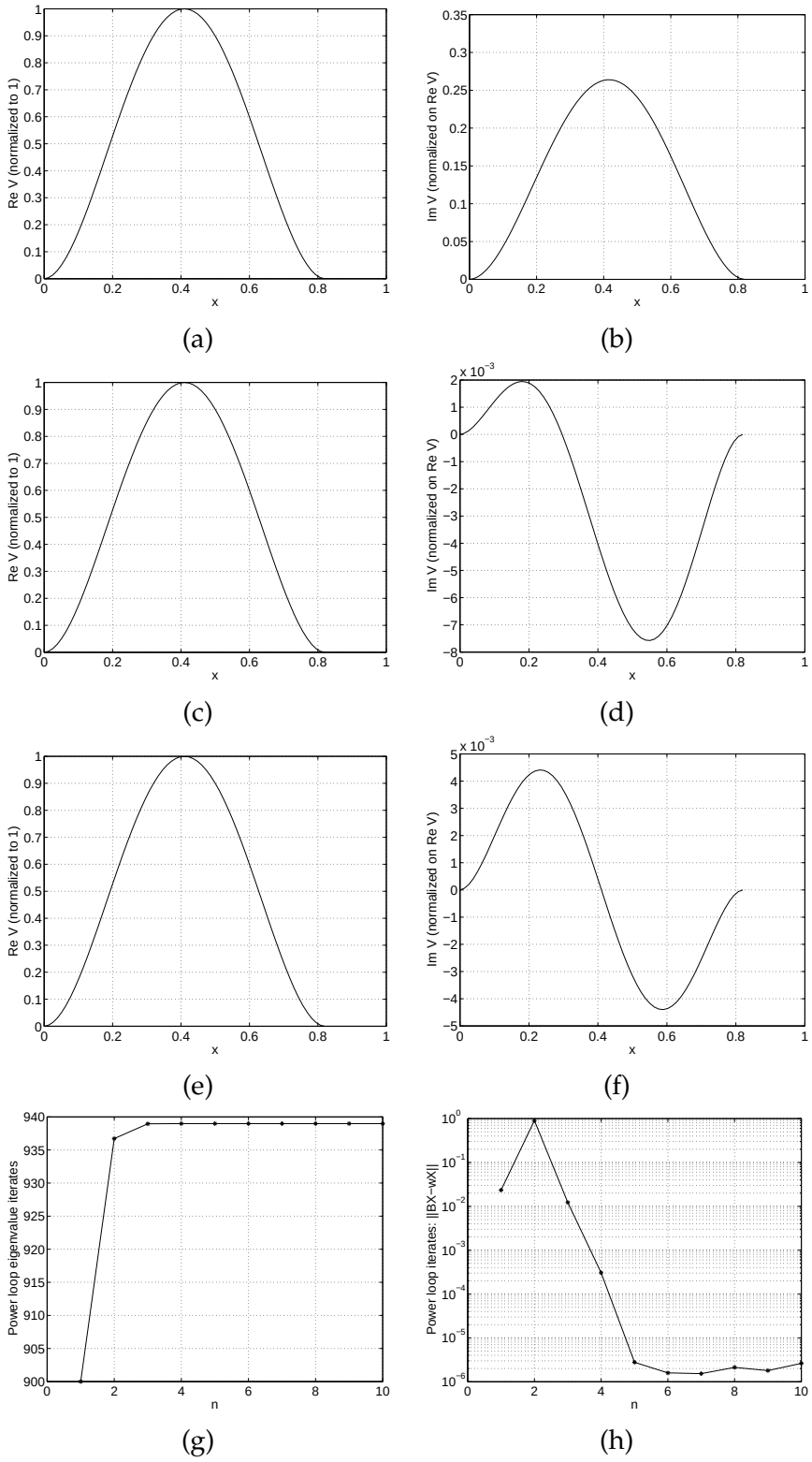

Figure 31: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 32 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

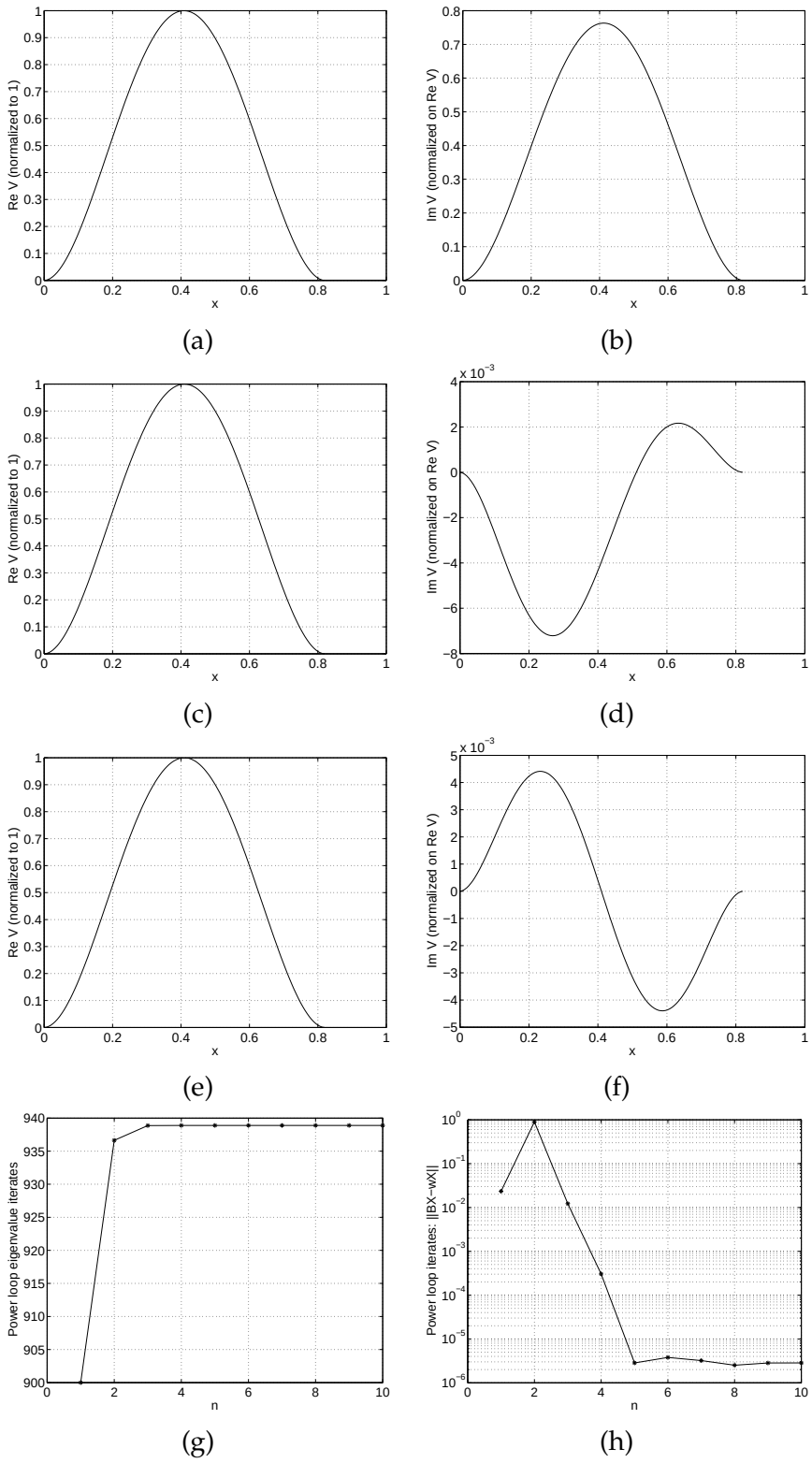

Figure 32: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 64 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

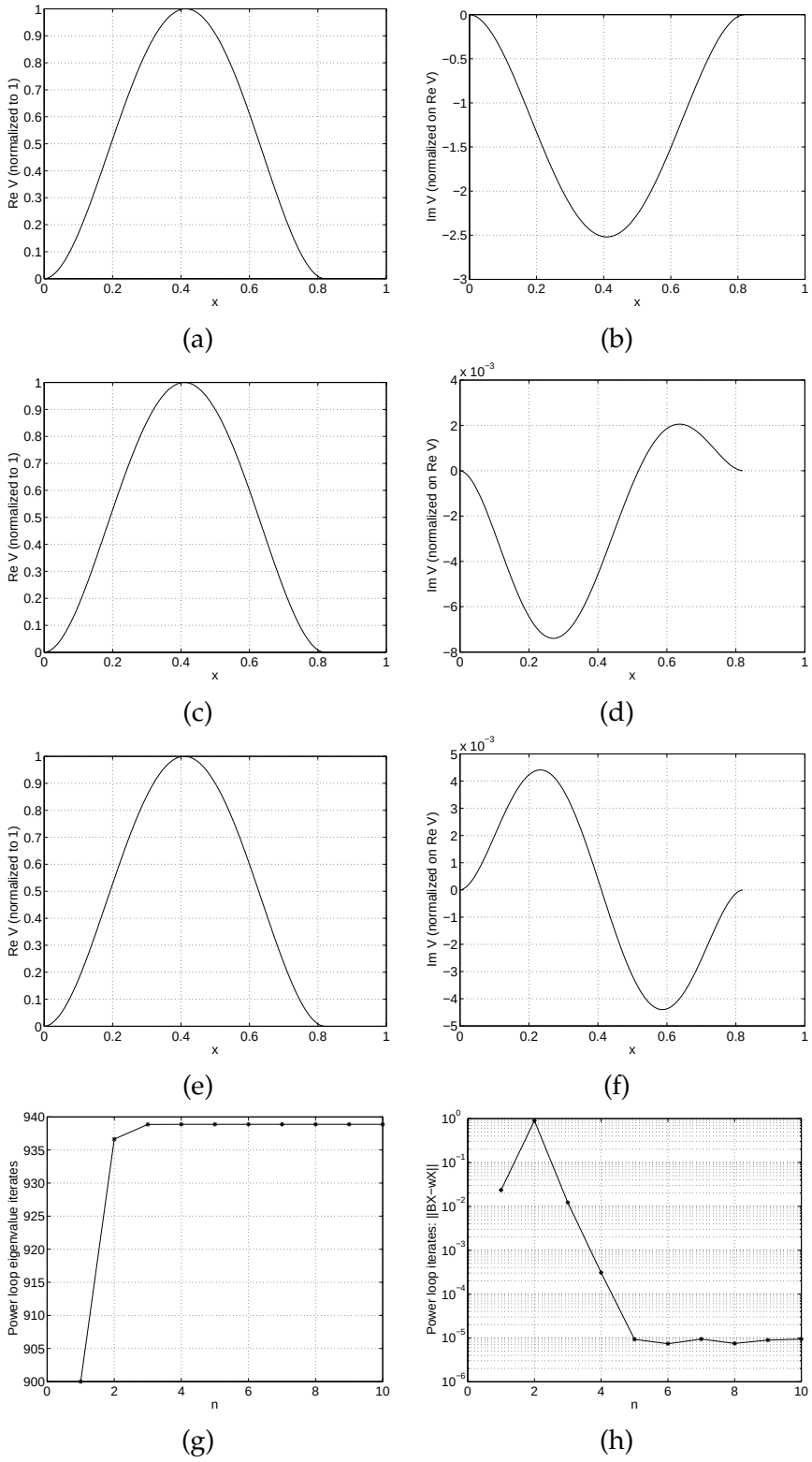

Figure 33: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 128 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

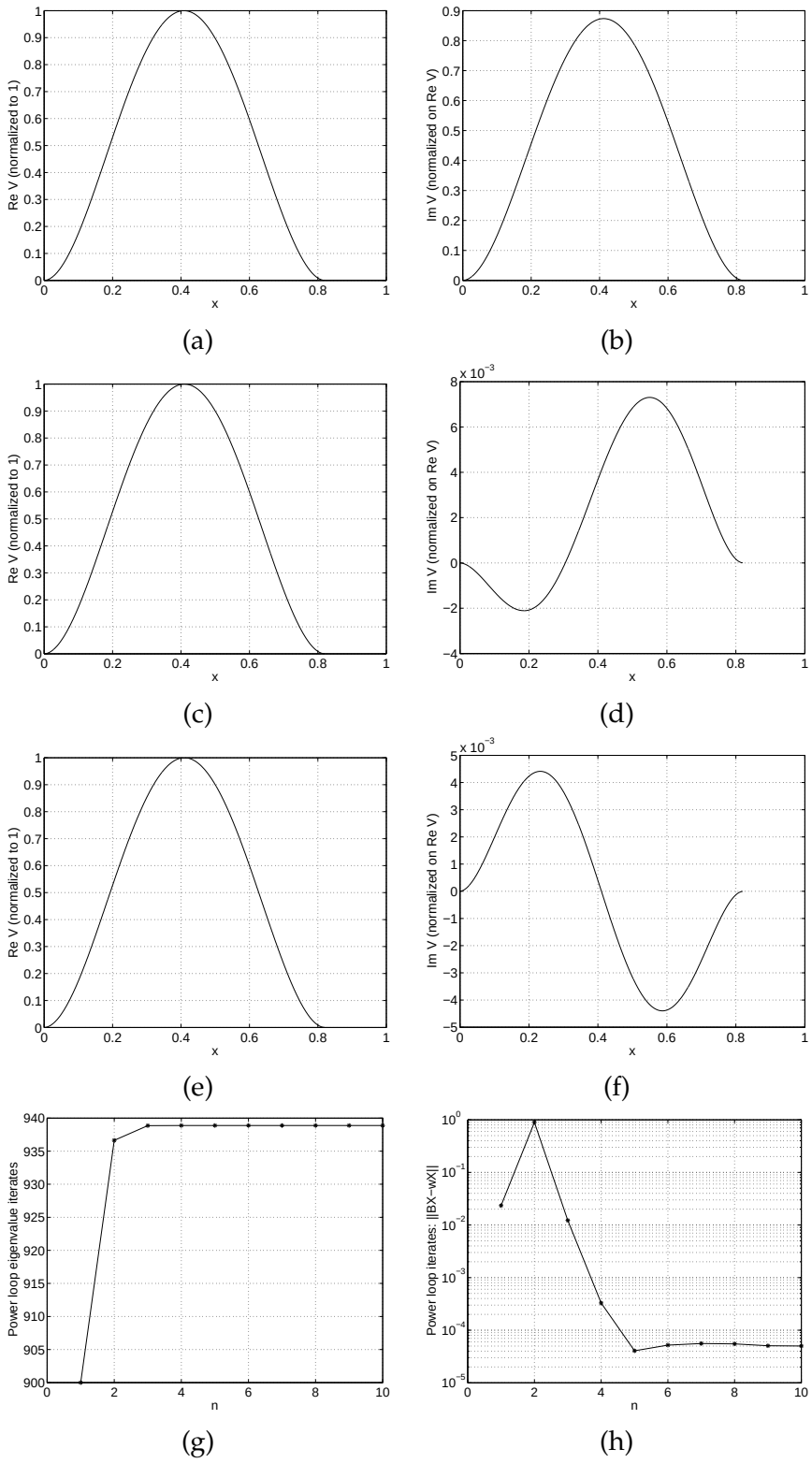

Figure 34: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 256 quadratic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

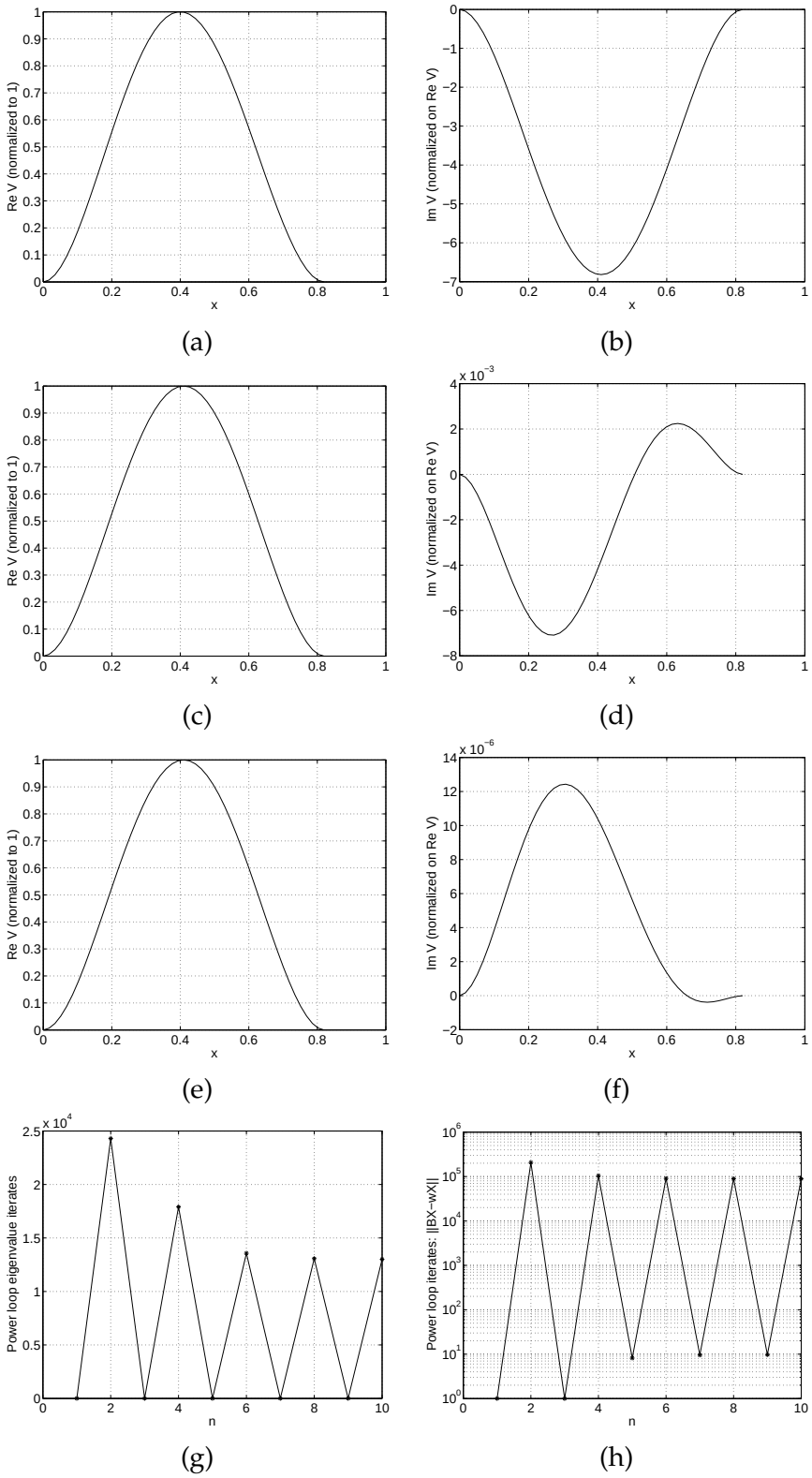

Figure 35: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 16 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

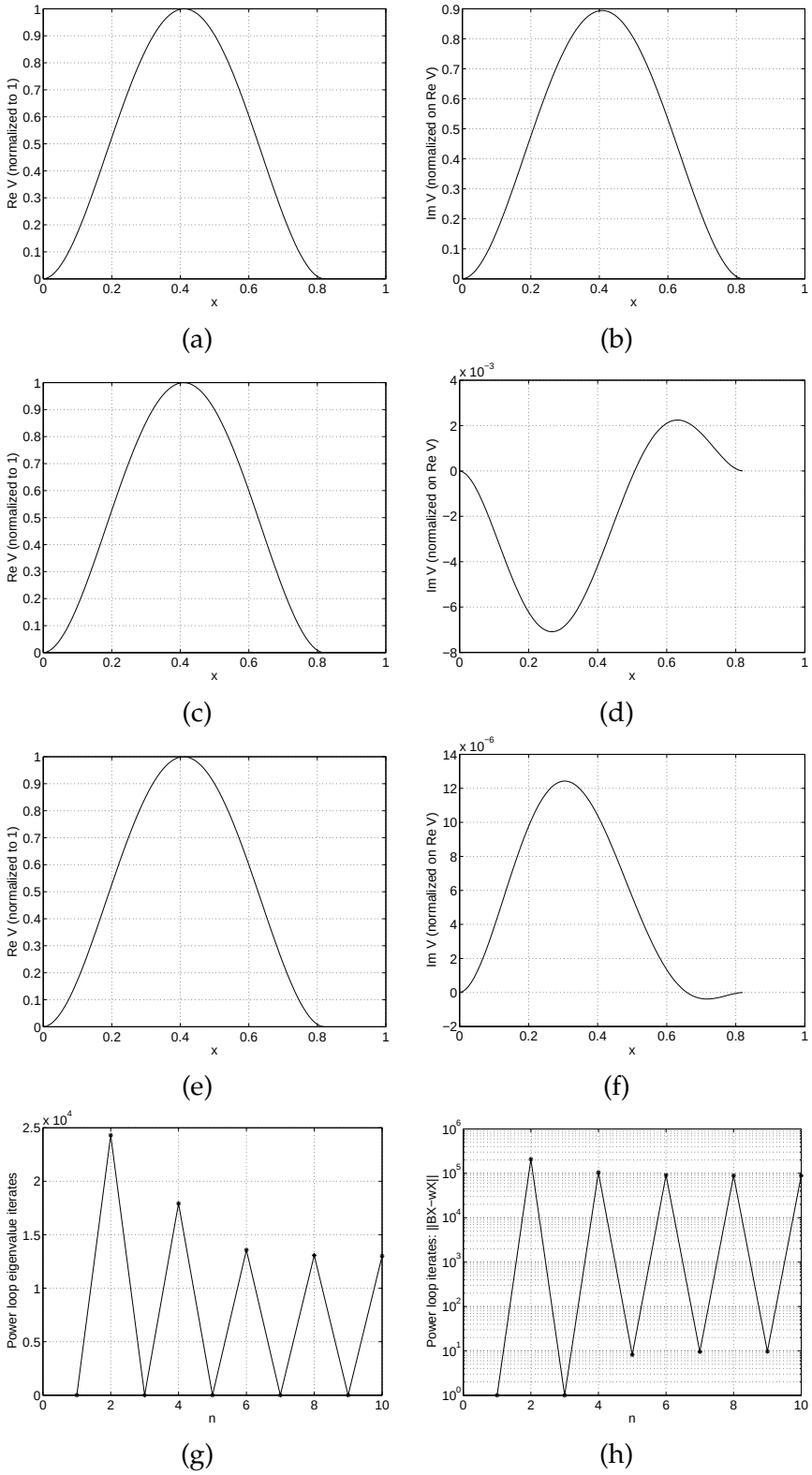

Figure 36: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 32 cubic elements: (a) and (b) show $\mathrm{Re}\,V$ and $\mathrm{Im}\,V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

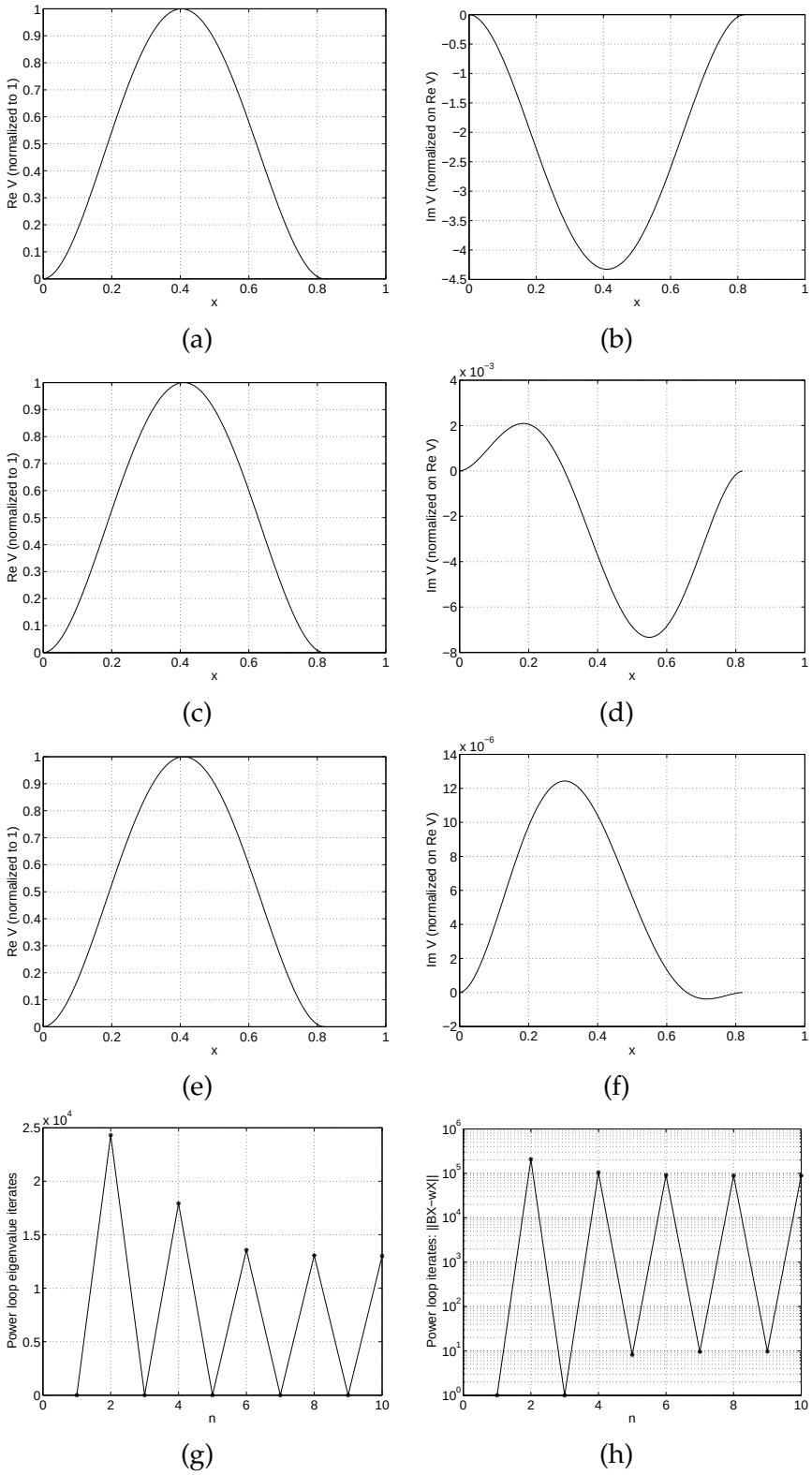

Figure 37: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 64 cubic elements: (a) and (b) show $\mathrm{Re}\,V$ and $\mathrm{Im}\,V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

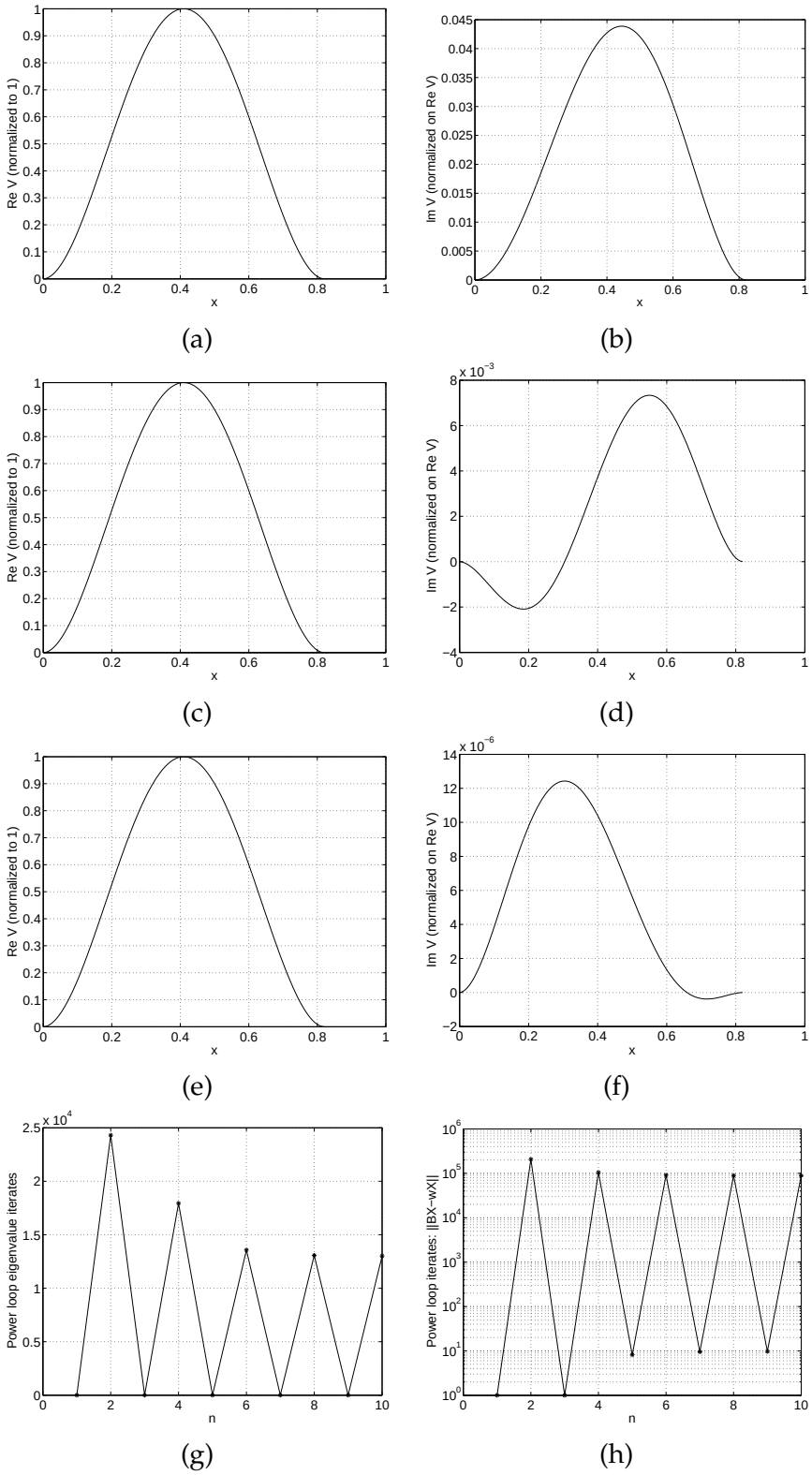

Figure 38: Computed eigen-results, without shift ($p = 0$), for the Timoshenko beam with 128 cubic elements: (a) and (b) show $\mathrm{Re}\,V$ and $\mathrm{Im}\,V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

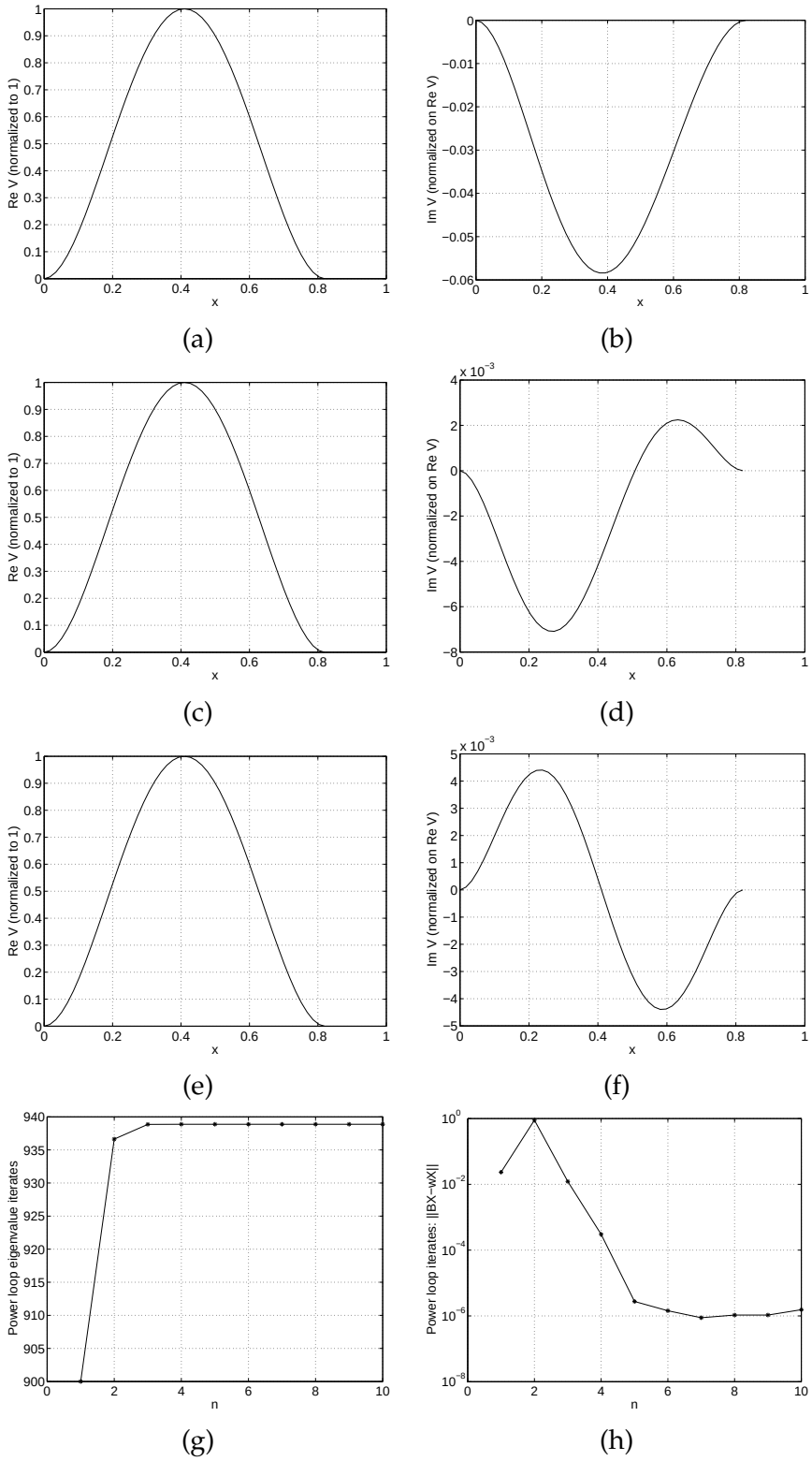

Figure 39: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 16 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

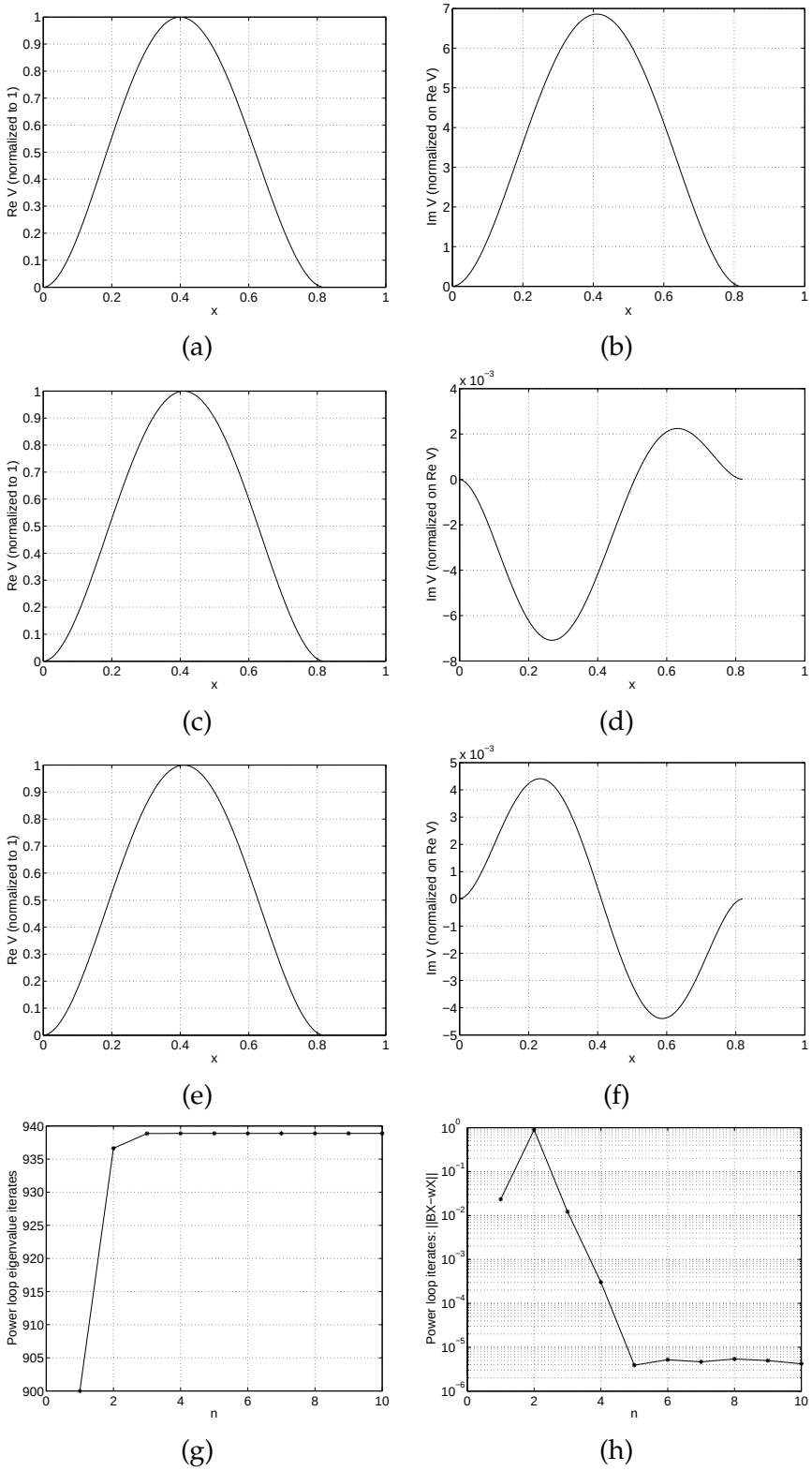

Figure 40: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 32 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{BX} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

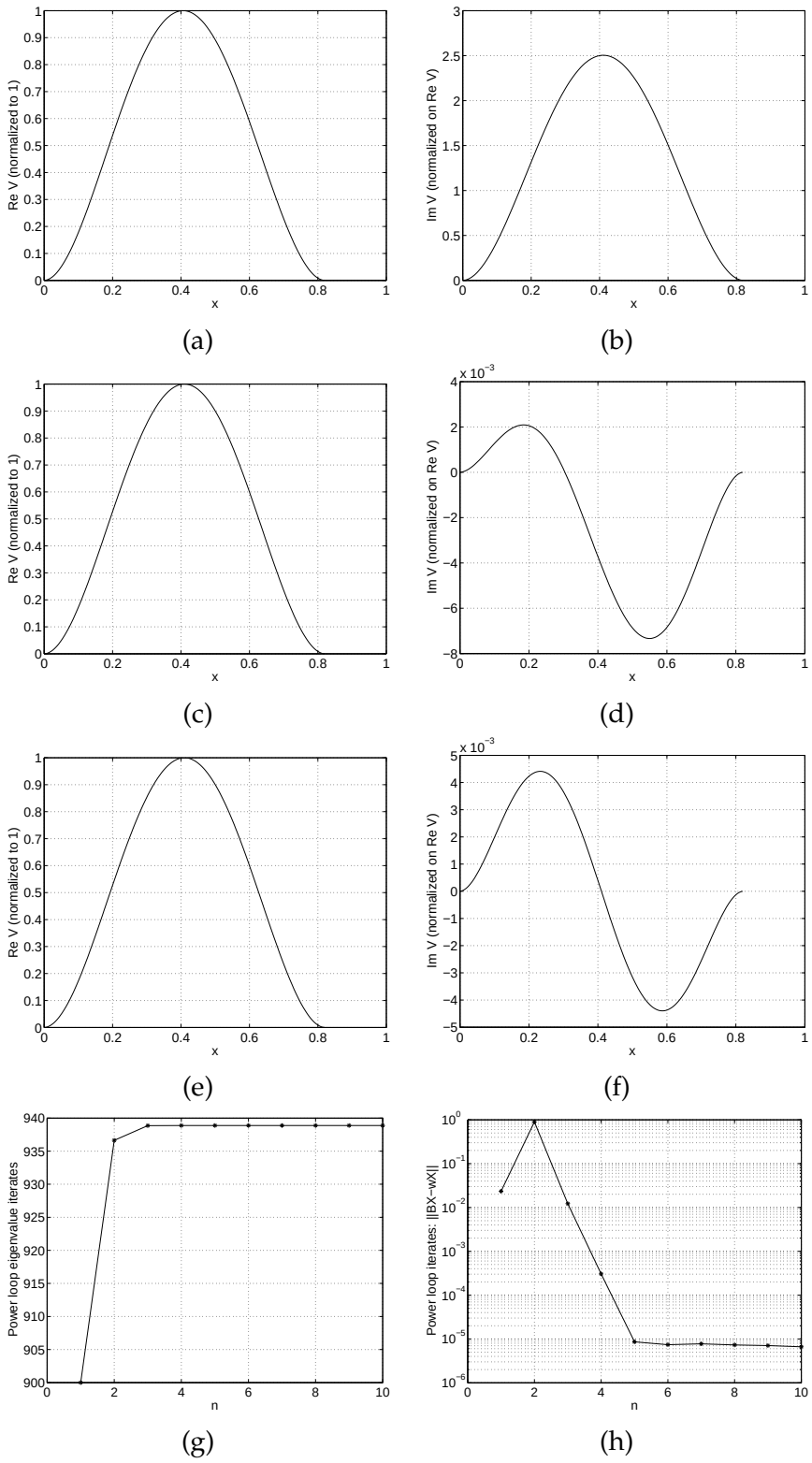

Figure 41: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 64 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega\boldsymbol{X}\|_\infty$, behaves during the iteration.

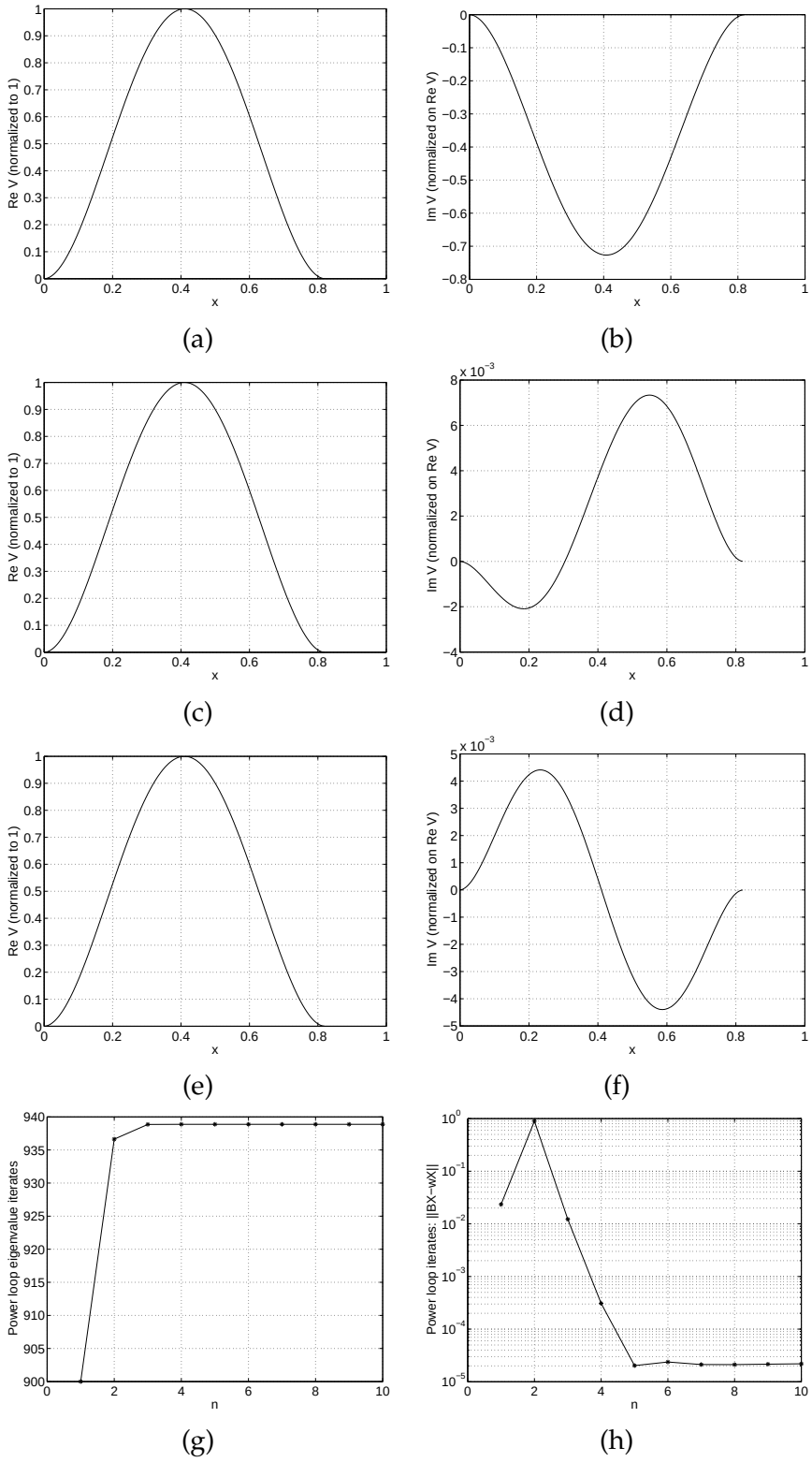

Figure 42: Computed eigen-results, with shift $p = 900$, for the Timoshenko beam with 128 cubic elements: (a) and (b) show $\mathrm{Re}\, V$ and $\mathrm{Im}\, V$ from Matlab's `polyeig` routine; (c) and (d) show the same but from Matlab's `eig` routine; (e) and (f) again show the same but from the inverse iteration; (g) shows the inverse iteration's eigenvalue iterates; and, (h) shows how the norm of the residual, $\|\boldsymbol{B}\boldsymbol{X} - \omega \boldsymbol{X}\|_\infty$, behaves during the iteration.

6 Conclusions

There are several points to be made. First, when no shift is used:

- Matlab's `eig` and `polyeig` give high quality approximations to the eigenvalue, but the inverse iteration cycles between two incorrect iterates. See Tables 1, 3, 5 and 7.
- On the other hand, none of the methods give a useful approximation to the imaginary part of the eigen mode. See Figures 1 to 6 and Figures 13 to 18; 25 to 29; and, 35 to 38. In particular, `polyeig` seems incapable of even picking up the inflection at the midpoint of the beam.

However, when we introduce a shift of $900\,\mathrm{rad/sec}$:

- The performance of both `eig` and `polyeig` neither improves nor deteriorates in any noticeable way in terms of the eigenvalue approximations. See Tables 2, 4, 6 and 8.
- The inverse iteration now provides both high quality eigenvalues, see Tables 2, 4, 6 and 8, and Coriolis distortion modes. See plot (f) in Figures 7 to 12; Figures 19 to 24; Figures 30 to 34 and Figures 39 to 42.

It seems clear that a shift is necessary in order to obtain useful mode shapes. For these results we were able to choose a 'good' value for this shift because we knew in advance the approximate value of $|\omega|_{\min}$. The question then arises as to how to choose this shift for a completely new prolem in which no approximate values are known in advance.

We can suggest an answer to this as follows: use either `eig` or `polyeig` to compute $|\omega|_{\min}$, our results suggest that a reliable value will be generated even though the mode shapes are unreliable. Use this value to derive a shift for the inverse iteration which, our results suggest, should be able to estimate reliable mode shapes as well.

Another observation relates to how the norm of the residual, $\|\boldsymbol{BX} - \omega\boldsymbol{X}\|_\infty$, behaves during the inverse iteration (plot (h) in Figures 7 to 12 and Figures 19 to 24; 30 to 34; and, 39 to 42).

For the Euler-Bernoulli beam this quantity (which should be zero) seems to bottom-out at a questionably large value when more than 64 elements are used (and for 512 elements it doesn't seem to even become consistently 'small'). On the other hand, for the Timoshenko beam the converged value seems to be consistently and reassuringly small across all the results. It would seem that this is a matrix conditioning issue: since the Timoshenko equations are of second order while the Euler-Bernoulli equation is of fourth order, it is not unreasonable to expect that the Timoshenko matrices are better conditioned, although further investigation is certainly warranted.

The results presented above convincly demonstrate that shifted inverse iteration gives consistently high quality estimates of the eigenvalue, the mode shapes and the meter sensitivites in all the cases under study, and also that the results are consistent with the approximate analytical results given in [4]. Hence, in the absence of any other analytical solution, we are forced to accept that these values are effectively 'exact'.

With that in mind, and given that `polyeig` gives poor mode shapes, it is surprising that the meter sensitivites computed from the `polyeig` output in Tables 9 to 16 agree so well with the 'exact' values computed from the shifted inverse iteration.

We are not able to give a definitive answer as to why `polyeig` and `eig` (and, presumably, other software products) are not able to produce accurate Coriolis distortion modes,

but our computational experience strongly suggests that it is due to computer rounding error. If this is the case then the estimation of meter sensitivites in the absence of a high quality mode shape should not be regarded as reliable. Notice that for very small radian angles $\tan x = \sin x / \cos x \approx x$, we have,

$$\begin{aligned} q_1 - q_2 &= \tan^{-1} \frac{\operatorname{Im} V(L/4)}{\operatorname{Re} V(L/4)} \\ &\quad - \tan^{-1} \frac{\operatorname{Im} V(3L/4)}{\operatorname{Re} V(3L/4)} \\ &\approx \frac{\operatorname{Im} V(L/4) - \operatorname{Im} V(3L/4)}{\operatorname{Re} V(L/4)}, \end{aligned}$$

because from plot (a) in Figures 1 to 42, we can see that $\operatorname{Re} V(L/4) \approx \operatorname{Re} V(3L/4)$. Hence, in order to arrive at a reliable sensitivity, all that is required of the computed imaginary mode shape is that the vertical distance between the quarter point values be accurate. In our `eig` results it is (but not for the `polyeig` results), but if the shape inaccuracy is in fact due to rounding error, then we cannot possibly expect this always to be the case across all software platforms, programming languages and eigen-algorithms.

Lastly, we note that no 'locking' can be observed observed for the Timoshenko beam with linear elements. It is not clear whether the fluid damping removes the danger, or whether (for these data) it will not appear until a larger number of elements are used.

7 The time discretisation

After finite element discretisation, each PDE problem can be written as a system of ordinary differential equations:

$$\boldsymbol{M}\boldsymbol{U}_{tt} + \boldsymbol{E}\boldsymbol{U}_t + \boldsymbol{A}\boldsymbol{U} = \boldsymbol{F}$$

This is then discretised in time as follows.

Set $\boldsymbol{W} = \boldsymbol{U}_t$ so that

$$\boldsymbol{M}\boldsymbol{W}_t + \boldsymbol{E}\boldsymbol{W} + \boldsymbol{A}\boldsymbol{U} = \boldsymbol{F}.$$

Now break the time interval up into discrete equally spaced time levels, $0 = t_0, t_1, \ldots$, and define the time step $k := t_i - t_{i-1}$.

A standard centered approximation then results in the equations,

$$\boldsymbol{M}\left(\frac{\boldsymbol{W}_i - \boldsymbol{W}_{i-1}}{k}\right) + \boldsymbol{A}\left(\frac{\boldsymbol{U}_i + \boldsymbol{U}_{i-1}}{2}\right) + \boldsymbol{E}\left(\frac{\boldsymbol{W}_i + \boldsymbol{W}_{i-1}}{2}\right) = \boldsymbol{F}$$

and

$$\frac{\boldsymbol{W}_i + \boldsymbol{W}_{i-1}}{2} = \frac{\boldsymbol{U}_i - \boldsymbol{U}_{i-1}}{k}.$$

Substituting the expression for $\boldsymbol{W}_i$ from the second into the first then gives a linear system for $\boldsymbol{U}_i$. Solving this system and back-substituting then gives $\boldsymbol{W}_i$ and we can move onto the next time step.

References

[1] Jochen Alberty, Carsten Carstensen, and Stefan A. Funken. Remarks around 50 lines of matlab: short finite element implementation. *Numerical Algorithms*, 20:117—137, 1999.

[2] Dietrich Braess. *Finite Elements: theory, fast solvers and applications in solid mechanics*. Cambridge, 1997.

[3] R. Cheesewright, C. Clark, A. Belhadj, and Y.Y. Hou. The dynamic response of Coriolis mass flow meters. *Journal of Fluids and Structures*, 18:165—178, 2003.

[4] R. Cheesewright and Simon Shaw. Uncertainties associated with finite element modelling of Coriolis mass flow meters. In preparation for a special issue of *Flow Measurement & Instrumentation* and Technical Report 06/3, `www.brunel.ac.uk/bicom`, due Sept. 2006.