Methods of solving eigenvalue problems

Anna Lina P. Sjur and Jan-Adrian H. Kallmyr

September 23, 2018

Abstract

1 Introduction

A great variety of problems in the physical sciences can be represented as eigenvalue problems, which generally takes the form:

$$O\mathbf{v} = \lambda\mathbf{v}, \tag{1}$$

where O is an operator, λ is an eigenvalue, and $\mathbf{v}$ is an eigenvector. Such problems can easily be solved in terms of linear algebra, and is therefore of great use in simplifying complicated problems, as well as providing a framework for creating efficient algorithms. In particular, we will look at the motion of a fixed buckling beam, and show that this is numerically identical to the motion of quantum dots. Specifically, we will solve following eigenvalue problem

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}v(x) = \lambda v(x). \tag{2}$$

Starting with the methods section, we will present the buckling beam problem, as well as the quantum dot problem, scaling and generalising them to the form of eq. 1. Using different numerical methods, we will use unit testing to make sure that each algorithm is implemented correctly. Moving on to the results section, we will present the efficiency and error of each algorithm in terms of CPU-time and relative error respectively. Then finally, in the discussion section, we will compare the different methods, and look at more possiblities for problems that can be solved using the same general algorithms.

2 Methods

2.1 The buckling beam

Considering first the buckling beam problem, we have

$$\gamma\frac{\mathrm{d}^2u}{\mathrm{d}x^2} = -Fu(x), \quad x \in [0, L] \tag{3}$$

where γ is a property constant, $u(x)$ the vertical displacement, and F the force applied at (L, 0) towards the origin. We can scale this equation by defining a parameter $\rho = \frac{x}{L}$. Inserting, we get

$$-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}u(\rho) = \lambda u(\rho), \quad \rho \in [0, 1] \tag{4}$$

where $\lambda = \frac{FL^2}{\gamma}$. Now we see that this equation is on the form of eq. 2. However, enforcing Dirichlet boundary conditions $u(0) = u(1) = 0$ and using a 2nd order central approximation for n integration steps:

$$\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} + \mathcal{O}\big(h^2\big) = \lambda_i v_i, \tag{5}$$

where $h = \frac{1}{n+1}$, we obtain the eigenvalue equations

$$A\mathbf{v} = \lambda_i\mathbf{v}. \tag{6}$$

Here A is a tridiagonal matrix, λ_i is an eigenvalue, and $\mathbf{v} \in (0, 1)$ is a vector.

2.2 Quantum dots

2.3 Algorithm: Jacobi's determinant

3 Results

$\mathbf{n}$	$\mathbf{t_g/t_s}$	$\mathbf{t_{LU}/t_s}$
10	2.08	3.70
10^2	1.89	$1.00 \cdot 10^2$
10^3	1.48	$1.05 \cdot 10^4$
10^4	1.43	$1.18 \cdot 10^6$
10^5	1.39	-
10^6	1.41	-
10^7	1.39	-

Table 1: Ratio between CPU time for the general algorithm ($\mathbf{t_g}$), the special algorithm ($\mathbf{t_g}$) and the LU decomposition algorithm ($\mathbf{t_{LU}}$) for different matrix sizes ($\mathbf{n}$). The LU decomposition crashed for $\mathbf{n}$ greater than 10^4.

4 Discussion

A

$$examplematrix = \begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & 0 \\ a_1 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_2 & b_3 & c_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \ddots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & \dots & \dots & 0 & a_{n-1} & b_n \end{bmatrix},$$

Figure 1: The numeric solution using different solving algorithms. The graphs for n=100 and n=1000 are so similar that they are not distinguishable.