Numerical Programming 2 (CSE) 2015

Worksheet 6
Updated

Exercise 1 (Spectral method)

Solve the one dimensional PDE

$$y'' + a(x)y' + b(x)y = f(x) \quad \text{on} \quad (-1, 1)$$

with $a(x) = f(x) = \sin(2\pi x)$, $b(x) = \cos(\pi x)$ and periodic boundary conditions $y(-1) = y(1)$ using the spectral method presented in class.

Compute the numerical solution for different values of N and estimate the order of convergence of the method by plotting the error as a function of N in a log-log plot.

Exercise 2 (Spectral method using FFT)

The quadrature rule $Q(f) := \frac{1}{d}\sum_{j=0}^{d-1} f(\frac{j}{d})$ provides a very accurate approximation of $\int_0^1 f(x)\,\mathrm{d}x$ if f is smooth and periodic on $[0, 1]$ (the error decays exponentially with d if f is analytic).

Given a vector v, $v_j := f(\frac{j-1}{2N+1}), j = 1, \ldots, 2N+1$, the *Fast Fourier Transform (FFT)* computes simultaneously all the (complex) scalar products

$$a_j := Q(\overline{\exp(2ij\pi\cdot)}f) \approx \int_0^1 \overline{\exp(2ij\pi x)} f(x)\,\mathrm{d}x, \qquad j = -N, \ldots, N,$$

where $i = \sqrt{-1}$, using only $\mathcal{O}(N \log N)$ operations (whereas $\mathcal{O}(N^2)$ operations would be necessary to compute them all individually).

The MATLAB commands `fft` and `ifft` compute the FFT resp. its inverse and it holds that

$$\texttt{fft}(v) = \frac{1}{2N+1} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \\ a_{-N} \\ \vdots \\ a_{-1} \end{pmatrix},$$

$$\texttt{ifft}(\texttt{fft}(v)) = v.$$

Solve numerically the one dimensional PDE

$$y'' + y = \exp(\sin(2\pi x)) \quad \text{on} \quad (0, 1)$$

with periodic boundary conditions using the FFT.

Exercise 3 (Stiff ODE)

Use the forward Euler, backward Euler and the trapezoidal method to compute the solution of the following ODE:

$$\begin{cases} y' = -\dfrac{1}{\epsilon}(y - \sin(t)), & 0 \le t \le \pi \\ y(0) = 1. \end{cases} \tag{1}$$

Verify that the exact solution of (1) is given by[1]

$$y(t) = \frac{\sin(t) - \epsilon\cos(t)}{1+\epsilon^2} + \left(1 + \frac{\epsilon}{1+\epsilon^2}\right) e^{-t/\epsilon}.$$

Solve the problem for $\epsilon = 1, 0.1, 0.01, 0.001$ and compare the performances of the three methods by computing the error between the exact and the numerical solution. You can use `fsolve` for implementing the implicit methods.

[1] Bonus problem: How would you compute the exact solution of (1) ?