APPM 4720 / 5720 — HOMEWORK # 3 **Due: Feb, 2018.**

Instructions

Put away your cell-phone and read this document from start to finish. Discuss in the group what the different instructions and tasks mean. Make sure you have a plan how to achieve the main task (and write up the results!) within the allotted time (team-work is probably needed). How will you check that the subtasks are correct?

Main task

Consider a grid with $N+1$ grid-points $x_L = x_0 < x_1 < \ldots < x_N = x_R$, defining N elements $\Omega_i = \{x \in [x_{i-1}, x_i]\}$, $i = 1, \ldots, N$. You are to approximate a function $f(x)$ on $x \in [x_L, x_R]$ by L_2-projection onto the space of element-wise Legendre polynomials of degree q. That is you must find $c(k,i)$ so that

$$\int_{\Omega_i} P_l(r) \sum_{k=0}^{q} c(k,i) P_k(r) dx = \int_{\Omega_i} P_l(r) f(x) dx, \quad l = 0, \ldots, q. \tag{1}$$

Here $r \in [-1, 1]$ is a local variable such that on element Ω_i the affine map $x(r)$ satisfies $x(-1) = x_{i-1}$, $x(1) = x_i$ (I use fancy words here so that you don't have to be afraid of them the next time you see them, just find a, b so that $x(r) = ar + b$ satisfies the two conditions, it is easy!).

Subtasks

Feel free to use available open source code for the tasks below. Make sure you give appropriate credit and that you respect the licensing requirements (if any).

1. Write a module `type_defs.f90` that defines `sp`, `dp`, etc. Use this module in all of the code you write in this class.

2. Write a module `leg_funs.f90` that contains functions for evaluating Legendre polynomials (and eventually their derivatives) of degree k.

3. Write a module `quad_1d.f90` that defines a type `quad_1d` that holds the start and end coordinate of the element, the degree of the element q and a two dimensional allocatable array of size `(0:q,nvars)`. In this homework `nvars` is one as you are approximating a single function but in a later homework when you solve systems of PDE it will be greater than one. The module should contain internal subroutines that "allocates a quad" and "deallocates a quad". We will stick more information into this structure, can you think of some things that could be good to know once the elements starts to communicate with each other?

4. You should approximate the integrals by Gauss quadrature on Gauss-Legendre-Lobatto nodes and of sufficiently high degree (how high?). You may want to use the existing code `lglnodes.m` by Greg von Winckel (after translating it to Fortran). Note that the integrals are carried out on the reference domain $r \in [-1, 1]$ so you have to perform a (as discussed above) change of variables $dx = x_r(r)dr$

5. Write routines that can output the approximation on a given grid (not necessarily the same as the one above). This does not have to be super general but at least make sure you can (over)sample the solution on each element.

6. Note that (1) are a set of N decoupled linear systems of equations of size $(q+1)\times(q+1)$. Do you have to write a routine that performs Gauss elimination (or more precisely call `LAPACK`)? Why not?

7. Your approximation can be stored in a `quad_1d` array with N entries, say:
   ```
   type(quad_1d), dimension(:), allocatable :: qds_1d
   ```
   which can be ordered from left to right. Note that in multiple dimensions such an array may not have a natural ordering and for example element 1 and 2 may be far away from each other.

To be reported (minimum)

Write up a short report in LaTeXwhere you describe your findings.

1. For at least three different functions $f(x)$ provide evidence that for $q = 0, 1, \ldots$, your approximation is increasingly accurate with an increasing number of elements as measured in uniform and L_2-norm. For example you can provide log-log plots where the errors are displayed as functions of the (typical) element size.

2. Fix the number of elements and inspect how the error decreases as you increase q. Can you fit (by trial and error) a function of the type c^{-q} to the error?

3. Use a script to carry out the collection of the data required to produce the figures / tables in your report.

4. Be curious. What happens if $f(x)$ is not smooth? Suppose you have equidistant x_0, x_1 etc. how does the rates of convergence compare when you add (small, say 5%) random perturbations to the grid? Suppose you prefer Chebyshev or monomials, what would change?