Montestigliano Workshop | **Problem 4 Differentiation matrices** | **Level 1 Grady Wright**

A natural thing to do with an interpolant of samples of a function is to differentiate it to approximate derivatives of the underlying function. This procedure is the central idea behind spectral and pseudospectral collocation methods (see, for example, [3, 4]). In these methods it is convenient to form differentiation matrices for a node set $X = \{\mathbf{x}_j\}_{j=1}^N$ that can be applied directly to a vector of samples of a function f at X to give an approximation to some derivative at X. In this problem, you will construct differentiation matrices from RBF interpolants on the unit sphere for the surface gradient and the surface Laplacian (Laplace-Beltrami operator) following the procedure outlined in the lecture slides. Further background material can be found in [1, 2, 5], where the ideas were first proposed and used in applications.

While it is possible to work in spherical coordinates for constructing the differentiation matrices using RBFs and avoid the "pole problem" (see [1]), you will be working in Cartesian coordinates since it will be useful for applications later. As discussed in the lecture slides, the surface gradient $\nabla_{\mathcal{S}}$ on the sphere can be expressed in Cartesian coordinates using the projection matrix $P = (I - \mathbf{x}\mathbf{x}^T)$ and the standard gradient in $\mathbb{R}^3$, $\nabla = [\partial_x\ \partial_y\ \partial_z]^T$, as follows:

$$\nabla_{\mathcal{S}} := P\nabla = (\mathbf{I} - \mathbf{x}\mathbf{x}^T)\nabla = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla \\ \mathbf{p}_z \cdot \nabla \end{bmatrix}. \tag{1}$$

The surface Laplacian $\Delta_{\mathcal{S}}$ on the sphere can also be expressed in Cartesian coordinates from the relationship that it is simply the divergence of the gradient:

$$\Delta_{\mathcal{S}} := \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}} = (P\nabla) \cdot (P\nabla),$$

Plugging in P and simplifying gives the expression

$$\Delta_{\mathcal{S}} := \Delta - \mathbf{x}^T(2 + \mathbf{x}^T\nabla)\nabla, \tag{2}$$

where $\Delta = \partial_{xx} + \partial_{yy} + \partial_{zz}$ is the standard Laplacian in $\mathbb{R}^3$. When the surface Laplacian is applied to a radial kernel ϕ the resulting expression can be simplified so that it is expressed entirely in terms of distance as shown in [5]. Letting $\mathbf{x}, \mathbf{x}_j \in \mathbb{S}^2$, ϕ a C^2 radial kernel, and $r_k(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_k\|$, we can write

$$\Delta_{\mathcal{S}}\phi(r_k(\mathbf{x})) = r_k(\mathbf{x})^2 \left(1 - \frac{r_k(\mathbf{x})^2}{4}\right) \psi(r_k(\mathbf{x})) + (2 - r_k(\mathbf{x})^2)\chi(r_k(\mathbf{x})), \tag{3}$$

where

$$\chi(r_k(\mathbf{x})) = \frac{1}{r_k(\mathbf{x})}\phi'(r_k(\mathbf{x})), \tag{4}$$

$$\psi(r_k(\mathbf{x})) = \frac{1}{r_k(\mathbf{x})}\chi'(r_k(\mathbf{x})). \tag{5}$$

We note that the surface gradient of $\phi(r_k(\mathbf{x}))$ can also be written in terms of the derivative term $\chi(r_k(\mathbf{x}))$; for example,

$$\mathbf{p}_x \cdot \nabla\phi(r_k(\mathbf{x})) = \left[x_j\, \mathbf{x}_j^T\mathbf{x}_k - x_k\right] \chi(r_k(\mathbf{x})). \tag{6}$$

Doing the division by $r_k(\mathbf{x})$ analytically in the derivative terms (4) and (5) is preferable to handle the case when $r_k(\mathbf{x}) = 0$, which happens when $\mathbf{x} = \mathbf{x}_k$. This is not a problem since the kernel is radial and assumed to be twice continuously differentiable, so that the $r_k(\mathbf{x})$ term in the denominator will cancel.

4.a) Write a MATLAB function for computing differentiation matrices for the three components of the surface gradient listed in (1). In the lecture slides the construction of the differentiation matrices for the x-component of the surface gradient $\mathbf{p}_x \cdot \nabla$ is given, which can be simplified using (6) above. The the construction for the remaining components should follow immediately from this. Your function should take as input a set of nodes $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$ and a function handle for the radial kernel ϕ and function handle for the related derivative term χ given in (4). Try to vectorize the code so that at most only one loop is used. Use Gaussian elimination (the `mrdivide` or "forward slash" function in MATLAB) to compute BA^{-1}. In the case where the interpolation matrix is positive definite (i.e. ϕ is a positive definite kernel) you may consider instead using Cholesky factorization. The output of your code should be 3 N-by-N differentiation matrices `Dx`, `Dy`, and `Dz` for the three respective components of the surface gradient (1). For an N-by-1 vector `f` containing samples of a function f at the nodes in X, computing the product `Dx*f` in MATLAB should give an approximation to $\mathbf{p}_x \cdot \nabla f$ at X.

(i) Test your code on the smooth function

$$f(\mathbf{x}) = f(x, y, z) = \cos(3x^2 + 5y^2 + 8(z-1)^2),$$

for which the surface gradient is

$$\nabla_{\mathcal{S}} f(\mathbf{x}) = \begin{bmatrix} 2x(3x^2 + 5y^2 + 8z(z-1) - 3) \\ 2y(3x^2 + 5y^2 + 8z(z-1) - 5) \\ 2(z(3x^2 + 5y^2) + 8(z-1)^2(z+1)) \end{bmatrix} \sin(3x^2 + 5y^2 + 8(z-1)^2).$$

Use both the IMQ and GA kernels in your tests and try some different values for the shape parameter ε, values of N, and different node sets to see how the errors behave.

4.b) Write a MATLAB function for also computing a differentiation matrix for the surface Laplacian. The construction of this matrix should be similar to that described for $\mathbf{p}_x \cdot \nabla$ in the lecture slides, but the entries of the B matrix in this formula should be replaced by (3) evaluated at $\mathbf{x}_j$. Your function should take as input a set of nodes $X = \{\mathbf{x}\}_{j=1}^N \subset \mathbb{S}^2$ and function handles for the radial kernel ϕ, and the related derivative terms χ and ψ in (4) and (5). As with 4.a), avoid loops and use Gaussian elimination. The output of your code should be an N-by-N differentiation matrices `L` for the surface Laplacian (2). For an N-by-1 vector `f` containing samples of a function f at the nodes in X, computing the product `L*f` in MATLAB should give an approximation to $\Delta_{\mathcal{S}} f$ at X.

(i) Test your code on various spherical harmonics $Y_\ell^m(\mathbf{x})$, which, of course, have the property that $\Delta_{\mathcal{S}} Y_\ell^m(\mathbf{x}) = -\ell(\ell+1) Y_\ell^m(\mathbf{x})$. The function `sphHarm` in the `rbfsphere` package can be used to compute spherical harmonics; see `help sphHarm` for instructions on using this file. As with 4.a) (i), use both the IMQ and GA kernels in your tests and try some different values for the shape parameter ε, values of N, and different node sets to see how the errors behave.

References

[1] N. FLYER AND G. B. WRIGHT, *Transport schemes on a sphere using radial basis functions*, J. Comput. Phys., 226 (2007), pp. 1059–1084.

[2] ——, *A radial basis function method for the shallow water equations on a sphere*, Proc. Roy. Soc. A, 465 (2009), pp. 1949–1976.

[3] B. FORNBERG, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, 1996.

[4] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

[5] G. B. Wright, N. Flyer, and D. A. Yuen, *A hybrid radial basis function - pseudospectral method for thermal convection in a 3D spherical shell*, Geophysics, Geochemistry, Geosystems, 11 (2010), p. Q07003.