Homework #3 is due the beginning of class on Tuesday, May 31, although you may submit an electronic version before then. Each person should turn in his or her own write-up. No late homeworks will be accepted.

Problem 1. Concepts

The following will serve as the notes for the course on scattered interpolation with kriging/radial basis functions. There are two exercises in this section.

Consider a function $f(s)$ for $s \in \mathcal{S}$, where $\mathcal{S}$ is some parameter space. We assume that we have evaluated the function at n points $s_j \in \mathcal{S}$, and we let $f_j = f(s_j)$. The goal is to approximate $f(s)$ for some $s \in \mathcal{S}$.

Simple Kriging: The kriging model assumes that the function is the realization of a conditional random field with an assumed correlation function. To approximate $f(s)$, we seek the coefficients of the linear model

$$f(s) \approx \sum_{j=1}^{n} f_j a_j(s) = \mathbf{f}^T \mathbf{a}(s). \tag{1}$$

We assume that $f(s)$ comes from the space of zero-mean conditional random fields with two-point correlation function $\mathcal{C}(s_1, s_2)$. We consider correlation functions of the form

$$\langle f(s_1) f(s_2) \rangle = \mathcal{C}(s_1, s_2) = g(\|s_1 - s_2\|). \tag{2}$$

One example of such a function is $g(\|s_1 - s_2\|) = \exp(-\|s_1 - s_2\|^2/\rho)$, where ρ is the correlation length of the Gaussian correlation function. There are many other possible correlation models in the literature.

To compute the coefficients $a(s)$, we first compute the correlation matrix C associated with the sample points s_j,

$$C_{ij} = \mathcal{C}(s_i, s_j) = g(\|s_i - s_j\|). \tag{3}$$

Note that this matrix is symmetric and positive semidefinite. We can express this matrix as

$$C = \left\langle \mathbf{f}\mathbf{f}^T \right\rangle. \tag{4}$$

Notice that this matrix multiplied against a vector $\mathbf{a}$ produces

$$C\mathbf{a} = \left\langle \mathbf{f}\mathbf{f}^T \right\rangle \mathbf{a} = \left\langle \mathbf{f}(\mathbf{f}^T \mathbf{a}) \right\rangle, \tag{5}$$

which is precisely the vector of correlations between the approximation model (1) and the function at the design points s_j. Since we have the correlation function, we can compute the true correlation between the interpolation point and the design sites with

$$\mathcal{C}(s, s_j) = g(\|s - s_j\|) \equiv \phi_j(s). \tag{6}$$

Collecting the $\phi_j(s)$ into a vector $\boldsymbol{\phi}(s)$ and setting it equal to (5), we get the linear system

$$C\mathbf{a}(s) = \boldsymbol{\phi}(s). \tag{7}$$

The approximation model uses the result of this system as

$$f(s) \;\approx\; \mathbf{f}^T\mathbf{a}(s) \;=\; \mathbf{f}^T C^{-1}\boldsymbol{\phi}(s). \tag{8}$$

Notice that this approximation depends on the correlation parameter ρ through $C = C_\rho$ and $\boldsymbol{\phi}(s) = \boldsymbol{\phi}_\rho(s)$. In practice, one may use a maximum likelihood method to estimate ρ – or any other covariance parameter – from the data. See Owen & Koehler's *Computer Experiments* for more details.

EXERCISE 1: Derive the expression for the prediction variance of the kriging approximation.

Ordinary Kriging: One limitation of the simple kriging model is the zero-mean assumption on the random field model. We can relax this assumption by imposing an affine constraint on the coefficients of the linear model (1). In other words, we constrain the coefficients so that

$$1 \;=\; \sum_{j=1}^{n} a_j(s) \;=\; \mathbf{e}^T\mathbf{a}(s). \tag{9}$$

This constraint ensures that the resulting approximation will exactly interpolate constants, i.e. when all f_j are equal. This is equivalent to assuming an unknown constant mean in the random field.

With this constraint, the system to solve for the coefficients becomes

$$\begin{bmatrix} C & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a}(s) \\ \lambda(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(s) \\ 1 \end{bmatrix}. \tag{10}$$

The matrix in this system is full rank, and the $\lambda(s)$ is the Lagrange multiplier corresponding to the affine constraint (9).

EXERCISE 2: Show that the affine constraint implies that the appoximation model is unbiased for random fields with a constant but unknown mean.

Radial Basis Functions: Another approach to scattered interpolation uses radial basis functions. A basis function is radial if it comes from a function that depends only on the pairwise distance between points in the parameter space, such as $g(\|s_1 - s_2\|)$. One example is the Gaussian radial basis function

$$g(\|s_1 - s_2\|) = \exp(-\|s_1 - s_2\|^2/\rho), \tag{11}$$

for some parameter ρ. Given the design points s_i, the radial basis function $\phi_i(s) = g(\|s_i - s\|)$. We seek an approximation model that is a linear combination of the radial bases,

$$f(s) \;\approx\; \sum_{i=1}^{n} b_i\phi_i(s) \;=\; \mathbf{b}^T\boldsymbol{\phi}(s). \tag{12}$$

To compute the coefficients $\mathbf{b}$, we enforce the interpolation condition

$$f(s_j) = \sum_{i=1}^{n} b_i \phi_i(s_j), \qquad j = 1, \ldots, n. \tag{13}$$

If we define the matrix C whose elements are

$$C_{ij} = \phi_i(s_j), \tag{14}$$

then the interpolation condition yields the linear system of equations

$$C^T \mathbf{b} = \mathbf{f}. \tag{15}$$

Applying the results of this computation to the RBF approximation, we have

$$f(s) \;\approx\; \mathbf{b}^T \boldsymbol{\phi}(s) \;=\; \mathbf{f}^T C^{-T} \boldsymbol{\phi}(s). \tag{16}$$

Since C is symmetric, this is equivalent to simple kriging (8).

Problem 2. Programming

In this exercise, you will implement and apply a Markov chain Monte Carlo method for calibrating a computational model given measurement data corresponding to its output.

Inverse Problems & Bayes' Rule: Consider a computational model $f(s)$ that takes d input parameters $s = (s_1, \ldots, s_d)$ and produces a k-vector of outputs. (Note that the k outputs may be a derived quantity – e.g., a subset – of the full outputs.) Assume we have k measurements m corresponding to the outputs of the model, and assume these measurements have some noise. We will model the noise with a k-vector of independent Gaussians with variance σ. Loosely speaking, we want to find the parameters s that bring the model into agreement with the noisy measurements. We can write this as

$$f(s) = m + \eta, \qquad \eta \sim N(0, \sigma^2 I), \tag{17}$$

where I is the $k \times k$ identity matrix. This implies that $f(s) - m$ has the same distribution as η. We assume that the variability in $f(s) - m$ can be attributed to variability in the input parameters. Therefore, we interpret s as a random variable, and we seek a probability density function on the input parameter space that causes the random variable $f(s) - m$ to have the same distribution as η. This justifies the label *inverse problem.*

One common method for formalizing this process is to employ Bayes' rule, which we write as

$$p(s \,|\, m) = c\, p(m \,|\, s) p(s). \tag{18}$$

The quantities from (18) are defined as follows.

- $p(s \,|\, m)$ is the probability of the parameters s given the measurements m. This is commonly called the *posterior density.* This is the density we want to approximate.

- $p(m \mid s)$ is interpreted as the probability of the measurements given the input parameters s. Using relation (17) we see that, given s, $m = f(s) - \eta$ is distributed like a multivariate Gaussian with mean $f(s)$ and covariance σI.

- $p(s)$ is the *prior* probability density on the input parameter space. This density is meant to encode prior knowledge on the parameters. For example, if $s > 0$, then $p(s)$ will be zero for $s < 0$. For our purposes, we will assume that $p(s)$ is a uniform density on the input parameter space. By constraining the parameter space to some closed subspace, we essentially include the prior information in the problem.

- c is simply a normalizing constant that ensures that the product of the likelihood and the prior is a probability density function, i.e. integrates to one.

Since both $p(s)$ and c are constants over the parameter space, we can rewrite Bayes' rule (18) with a proportionality symbol as

$$p(s \mid m) \propto p(m \mid s). \tag{19}$$

This forms the foundation of the Markov chain Monte Carlo (MCMC) method. For notational convenience, we denote the likelihood $\mathcal{L}(s) = p(m \mid s)$.

In our case, the likelihood is a multivariate Gaussian density

$$\mathcal{L}(s) = (2\pi\sigma^2)^{-k/2} \exp\left(\frac{-\|f(s) - m\|^2}{2\sigma^2}\right). \tag{20}$$

This can be difficult to work with due to potential numerical overflow, so we prefer to use the log-likelihood

$$\ell(s) \;=\; \log(\mathcal{L}(s)) \;=\; \frac{-k}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|f(s) - m\|^2. \tag{21}$$

Markov chain Monte Carlo: Since we do not know the proportionality constant c, we cannot directly compute statistics from Bayes' rule. However, we can sample from the posterior density using the proportionality relationship (19) with a Metropolis-Hastings algorithm. This algorithm produces a Markov chain whose steady-state distribution is equal to the posterior. Thus, sample paths of the chain are samples from the posterior density. Here we give pseudocode to implement this procedure. Suppose that s_n is the state of the chain at time n.

- Let δ_n be drawn from a proposal distribution, which we assume is zero-mean multivariate Gaussian with d components and covariance $\epsilon^2 I$, where I is a $d \times d$ identity matrix. The parameter ϵ^2 is the step length.

- Compute $s' = s_n + \delta_n$ as the proposed step. If any component of s' falls outside the range of the parameter space, recompute δ_n and try again.

- Compute the likelihood ratio $r = \mathcal{L}(s')/\mathcal{L}(s_n)$. Notice that this ratio cancels the effects of the proportionality constant and thus yields a ratio of posteriors.
- Draw U form a uniform distribution on $[0, 1]$.
- If $U < \min(r, 1)$, then $s_{n+1} = s'$. (In other words, we accept the proposed step.) Otherwise, $s_{n+1} = s_n$, and the chain does not move.

Notice the similarities between this method at the acceptance/rejection method for sampling from an unknown density function.

After the process is run N times, the set of states $\{s_n\}$ constitutes samples from the posterior. In practice, we typically remove the first $N_1 < N$ samples from the set; these correspond to the so-called *burn-in* period. Determining the proper burn-in period is more art than science. From the samples of the posterior, one can compute a sample mean and covariance for the posterior density on the parameter space.

Using Surrogate Models: The primary drawback of the MCMC method is that each evaluation of the likelihood function requires an evaluation of the model $f(s)$. Chains in high dimensional parameter spaces may need millions of steps to find a region where the likelihood is large. For expensive function evaluations, this can be prohibitively expensive. In this case, one can use a few expensive function evaluations to tune a surrogate model $\tilde{f}(s) \approx f(s)$. Then the likelihood can be evaluated using the surrogate model within the MCMC.

Download `hw3.tgz` from the website. Inside the tarball, you will find a function `complex_solver2.m` whose function call is

```
[Z,X,Y] = complex_solver2(s,m);
```

The input `s` are the input parameters, and `m` controls the grid resolution. The outputs contain the surface `Z` and an (x, y) grid in the matrices `X` and `Y`. You can visualize the surface by calling

```
surf(X,Y,Z);
```

The code solves an advection-diffusion problem, where the parameters $s = (s_1, s_2)$ control the location and height of the initial disturbance. The parameter ranges are $s_1 \in [1, 3]$ and $s_2 \in [0, 1]$, distributed uniformly.

Your assignment, should you choose to accept it, is to determine the parameters that were used to create the images in the `truth.mat` file. In other words, these are the measurements. You may use at most ten (10) sensors placed throughout the grid to determine the parameters. You must choose the location of the sensors. See the example script for more details.

You will need some interpolation software for part C. Download and install the following MATLAB packages:

1. DACE – `http://www2.imm.dtu.dk/~hbn/dace/`.

2. PMPack – `http://www.stanford.edu/class/me470/docs/pmpack.tgz`

You will use the packages to construct kriging and tensor product pseudospectral interpolation surrogates for the complex solver at high resolution. The tarball from the website contains a script to get you started with these.

A. Implement the MCMC algorithm described above. Test it on synthetic data that you generate. In other words, pick a point in the parameter space, run the solver, and pretend that the output is the measurements.

B. Using grid resolution `m=20` and your favorite choice of sensor placement, apply the MCMC method to estimate the posterior density. Take the sensor measurements from `Z_20` in `truth.mat`. What are the parameters that were most likely used to generate the image? How certain are you of this? What is the smallest number of sensors you need to determine the parameters? Try it with 10, 6, and 2 sensors. How did you choose where to place them? How many steps does your MCMC method need to reach the steady-state, or what is a good burn-in period? Make sure to report what step size you used (ϵ^2 from above).

C. Plot the log-likelihood function over the parameter space. Plot the states of your Markov chain in the parameter space. Did your chain find the region of largest log-likelihood?

D. Change the grid resolution to `m=152` – this will take much longer to run for a given input. Use DACE and PMPack to construct surrogates for the expensive simulation as a function of s. Use the MCMC method with the surrogate evalution in the likelihood, where the measurements are taken from `Z_152` in `truth.mat`. Again, what are the parameters that are most likely to produce the image? Which surrogate worked better? Can you think of a way to include the error estimate from the surrogate in this process?

Problem 3. Reading

Read the following papers.

A. H. P. Flath, L. C. Wilcox, V. Akcelik, J. Hill, B. van Bloemen Waanders, and O. Ghattas. *Fast Algorithms for Bayesian Uncertainty Quantification in Large-Scale Linear Inverse Problems Based on Low-Rank Partial Hessian Approximations.* SIAM Journal of Scientific Computing, 2011.

C. T. Bui-Thanh, K. Willcox, and O. Ghattas. *Model Reduction for Large-Scale Systems with High-Dimensional Parametric Input Space.* SIAM Journal of Scientific Computing, 2008.

B. D. R. Jones. *A Taxonomy of Global Optimization Methods Based on Response Surfaces.* Journal of Global Optimization, 2001.

Write a one-page critical response to one of the papers. You should summarize the work in a paragraph and attempt to answer questions such as: How might you apply these ideas to your own research? What are the primary advantages and disadvantages of the methods? How might the methods or concepts be improved?