Numerical quadrature and Markov Chain Monte Carlo methods

Henry Lloyd-Laney

March 2020

1 Introduction

This is a companion piece to the supplementary information, describing some computational approaches that are not required for understanding of the paper. Here, we detail the ways that we approximate a currenting arising from an electrochemical system where some of the parameters are dispersed according to a probability distribution, and details for Markov-Chain Monte-Carlo (MCMC) methods.

2 Numerical quadrature

We describe both of these approaches here for a single integral, but they can be easily combined for multiple integrals. For each individual integral, we are calculating the expectation of the current with regards to a particular parameter $E[I(p,t)]$. This is expressed as

$$E[I(p,t)] = \int pdf(p)I(p,t)dp, \tag{1}$$

such that the pdf is the probability density function for the distribution chosen for that parameter.

2.1 Midpoint rule

The midpoint rule for an integral is

$$\int_b^a f(x)dx = (b-a)f\left(\frac{b+a}{2}\right). \tag{2}$$

In order to calculate the value of the expectation in equation 1 using this rule, we use the expression

$$E[I(p,t)] \approx \sum_{i=1}^{n} w_i I(p_i,t), \tag{3}$$

where the set of parameter values is $p_0, \ldots, p_n$ and the weight w_i is calculated using the midpoint rule such that

$$w_i = (p_i - p_{i-1}) pdf\left(\frac{p_i + p_{i-1}}{2}\right), \tag{4}$$

or alternatively

$$w_i = cdf(p_i) - cdf(p_{i-1}), \tag{5}$$

where *pdf* and *cdf* are the probability density and cumulative density functions respectively. As the support for many distributions is infinite, for numerical purposes we define the parameter range to integrate over (i.e. $p_0, \ldots, p_n$) using the inverse cumulative density function cdf^{-1}, such that $p_0 = cdf^{-1}(x)$ and $p_n = cdf^{-1}(1-x)$. The area defined by this range of values will be $1 - 2x$, and so we choose a value for x that is small enough such that $\sum_{i=1}^{n} w_i \approx 1$.

2.2 Gaussian quadrature

Gaussian quadrature is an alternative numerical integration technique that offers greater accuracy over the midpoint rule by making better choices about the location of the nodes (and appropriate weights) than the linearly spaced nodes of the midpoint rule. To do this, we use the Lagrange polynomials, which are derived by performing the Gram-Schmidt orthogonalisation procedure on a set of basis polynomials

$$\{1, x, x^2, x^3, \ldots x^n\}, \tag{6}$$

using an inner product defined such that

$$(p, q) = \int_{-1}^{1} p(x) q(x) dx \tag{7}$$

This operation results in a set of polynomials that are orthogonal to every lower-degree Lagrange polynomial (i.e. $(L_n, L_q) = 0$ for $q < n$), and also, because of the way they were constructed, the n^{th} Lagrange polynomial is also orthogonal to every linearly independent polynomial of degree less than n. The n^{th} Lagrange polynomial takes the form

$$L_n(x) = x^n - \sum_{i=0}^{n-1} = \frac{(L_i, x^k)}{(p_i, p_i)} L_i(x) \tag{8}$$

were $L_0(x) = 1$ We use Lagrange polynomials in numerical integration in the following way. Given a polynomial $p(x)$ of degree $2n - 1$ that we wish to integrate, we can represent it as the product of the Lagrange polynomial $L_n(x)$of degree n and a quotient $q(x)$ of degree $n - 1$ or less, with a remainder, $r(x)$ which is also of degree $n - 1$ or less

$$p(x) = q(x) L_n(x) + r(x). \tag{9}$$

The integral of $p(x)$ then becomes

$$\int_{-1}^{1} p(x)dx = \int_{-1}^{1} q(x)L_n(x)dx + \int_{-1}^{1} r(x)dx, \tag{10}$$

and because the first term on the RHS of the equation is an inner product of a Lagrangian polynomial with a polynomial of lower degree, it is equal to 0. The numerical integration takes the form of summing over appropriately weighted evaluations of the function $p(x)$ for N nodes, such that

$$\sum_{i=1}^{N} w_i p(x_i) = \int_{-1}^{1} p(x)dx, \tag{11}$$

where the weights and nodes (w_i and x_i respectively) are chosen so that the sum in equation 11 is exactly equal to the integral of the polynomial $p(x)$. We can expand the sum in equation 11 using the relation in equation 9, such that

$$\sum_{i=1}^{N} w_i p(x_i) = \sum_{i=1}^{N} w_i (q(x_i)L_n(x_i) + r(x_i)). \tag{12}$$

If we choose the n nodes to be the zeroes of $L_n(x)$, then we get the relation

$$\sum_{i=1}^{n} w_i p(x_i) = \sum_{i=1}^{n} w_i r(x_i) = \int_{-1}^{1} r(x)dx = \int_{-1}^{1} p(x)dx. \tag{13}$$

Because we have said that the weighted sum over the polynomial $\sum_{i=1}^{N} w_i p(x_i)$ is exactly equal to the integral $\int_{-1}^{1} p(x)dx$, and in addition that this is equal to the weighted sum of $r(x)$ which is of degree n, it follows that we can exactly integrate a polynomial $p(x)$ of degree $2n-1$ by evaluating it with only n nodes, as long as these nodes are the values of the zeroes of the n^{th} degree Lagrange polynomial, and that the evaluations are weighted correctly.

The final step is to choose the weights such that equation 11 is exact. For example, for Lagrange polynomial of degree 4, we construct a matrix equation

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_2 & x_4 \\ x_1^2 & x_2^2 & x_2^2 & x_4^2 \\ x_1^3 & x_2^3 & x_2^3 & x_4^3 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \frac{2}{3} \\ 0 \end{bmatrix},$$

where the values of the weights (w_1, w_2, w_3, w_4) needed for the 4^{th} order Lagrange polynomial is the solution to this equation, where the values of x_i are the zeros of the polynomial in question. The intuition behind this is that if the function is (for example) $f(x) = x$, or the second element of the basis defined in equation 6, then the resulting function

$$\sum_{i=1}^{4} w_i f(x_i) = 0 = \int_{-1}^{1} x dx \tag{14}$$

is the exact solution to the integral, and this is valid up to cubic functions. The brilliance of Gaussian quadrature is that, using the relation in 13, this is actually exact for polynomials of degree 7.

2.3 Gauss-Hermite quadrature

We do not actually use Gaussian Legendre quadrature as described above, as it is only valid for integrals over finite intervals. For the expectation we want to calculate in equation 1 for a Normal distribution, we need to calculate the integrals over infinite intervals. For this, we use Gauss-Hermite quadrature, which uses a similar principle as described above, but uses the Hermite polynomials (another set of orthogonal polynomials) to evaluate integrals of the form

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i). \tag{15}$$

To make this appropriate for the calculation of the expectation in equation 1, we write it in the form

$$E[I(p,t)] = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(p-\mu)^2}{2\sigma^2}\right) I(p,t)dp \tag{16}$$

where μ and σ are mean and standard deviation of the normal distribution. If we transform p such that $p = \sigma\sqrt{2x} + \mu$, then

$$E[I(p,t)] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} \exp\left(- x^2\right) I(\sigma\sqrt{2x} + \mu, t)dx \tag{17}$$

consequently, for a Hermite polynomial of degree n

$$E[I(p,t)] \approx \sum_{i=1}^{n} \frac{1}{\sqrt{\pi}} w_i I(\sigma\sqrt{2x_i} + \mu, t)dx \tag{18}$$

where x_i and w_i are the nodes and appropriate weights respectively for the Hermite polynomial. Both of these integration methods (Gauss-Legendre and Gauss-Hermite) are valid as long as the function is well-approximated by a linearly independent polynomial. Information in this section was predominantly found in [1].

3 Markov-chain Monte-Carlo

The quantity we want to determine is the posterior probability distribution, or the probability of observing a set of parameter values $\boldsymbol{\theta}$ given our data $\mathbf{y_D}$, expressed as $p(\boldsymbol{\theta}|\mathbf{y_D})$. The formula for the posterior is given by Bayes rule

$$p(\boldsymbol{\theta}|\mathbf{y_D}) = \frac{p(\mathbf{y_D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y_D})}. \tag{19}$$

From this expression, we can see that the posterior is proportional to the likelihood (the probability of observing the data given a set of parameters) multiplied by the prior (which is the probability of observing the parameters used to calculate the likelihood). The normalising factor is the probability of the data $p(\mathbf{y_D})$, which requires the computation of multiple multidimensional integrals.

In order to approximate the posterior probability distribution for each parameter without calculating the multidimensional integrals required for the normalising factor, we use MCMC, a well-understood technique first described by Metropolis [2]. The Markov chain is defined such that its stationary distribution is the posterior probability density, and so by drawing a sufficient number of samples, the Markov chain will converge to the posterior. The Metropolis-Hastings algorithm performs MCMC by sampling parameter space, and moving accordingly towards areas of higher posterior density, where each parameter vector is a "state" in the Markov chain. Given a vector of parameter values, $\boldsymbol{\theta}$, which can include the noise standard deviation parameter σ, we wish to obtain the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y_D})$ that best describes our data. We use MCMC methods to approximate some desired distribution $P(x)$ (for our case, this is the posterior, $p(\boldsymbol{\theta}|\mathbf{y_D})$). The algorithm uses an ergodic Markov chain, where the stationary distribution $h(x) = P(x)$). As the algorithm runs, the distribution of the samples should become an increasingly good approximation of $P(x)$. The transition probabilities between states of the Markov chain are calculated as the ratio of the current posterior distribution and a proposal posterior distribution, which cancels the $p(\mathbf{y_D})$ term, meaning we do not have to calculate $p(\boldsymbol{y_D})$, the term which requires the calculation of multidimensional integrals. For this process to work, the Markov chain must be reversible, so the probability of a transition $\boldsymbol{\theta} \to \boldsymbol{\theta}'$ is equal to $\boldsymbol{\theta}' \to \boldsymbol{\theta}$. This is the condition of detailed balance, such that:

$$P(\boldsymbol{\theta}'|\boldsymbol{\theta})P(\boldsymbol{\theta}) = P(\boldsymbol{\theta}|\boldsymbol{\theta}')P(\boldsymbol{\theta}') \tag{20}$$

And therefore:

$$\frac{P(\boldsymbol{\theta}'|\boldsymbol{\theta})}{P(\boldsymbol{\theta}|\boldsymbol{\theta}')} = \frac{P(\boldsymbol{\theta}')}{P(\boldsymbol{\theta})} \tag{21}$$

For each step of the algorithm, a new state of the Markov chain is proposed according to a proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta})$. Whether this transition happens or not is a function of the acceptance probability $A(\boldsymbol{\theta}', \boldsymbol{\theta})$. Therefore, the probability of the $\boldsymbol{\theta} \to \boldsymbol{\theta}'$ transition occurring is:

$$P(\boldsymbol{\theta}'|\boldsymbol{\theta}) = q(\boldsymbol{\theta}'|\boldsymbol{\theta})A(\boldsymbol{\theta}', \boldsymbol{\theta}) \tag{22}$$

If we insert this relationship into equation 22, we obtain:

$$\frac{A(\boldsymbol{\theta}', \boldsymbol{\theta})}{A(\boldsymbol{\theta}, \boldsymbol{\theta}')} = \frac{P(\boldsymbol{\theta}')}{P(\boldsymbol{\theta})}\frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}')}{q(\boldsymbol{\theta}'|\boldsymbol{\theta})} \tag{23}$$

If the proposal distribution is symmetric (i.e. $q(\boldsymbol{\theta}|\boldsymbol{\theta}') = q(\boldsymbol{\theta}'|\boldsymbol{\theta})$), equation 23 reduces down to $\frac{A(\boldsymbol{\theta}', \boldsymbol{\theta})}{A(\boldsymbol{\theta}, \boldsymbol{\theta}')} = \frac{P(\boldsymbol{\theta}')}{P(\boldsymbol{\theta})}$. For the algorithm, we define a probability of acceptance α such that:

$$\alpha = \min\left\{\frac{P(\boldsymbol{\theta}')}{P(\boldsymbol{\theta})}, 1\right\} \tag{24}$$

According to Bayes rule (equation 19), the acceptance probability becomes:

$$\alpha = \min\left\{\frac{p(\mathbf{y_D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathbf{y_D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}, 1\right\} \tag{25}$$

For each step of the algorithm, a parameter vector is proposed, and then accepted or rejected. If the proposed vector is better than the current value, such that the acceptance ratio is greater than 1 (i.e. it has a greater posterior density), then the new parameter vector is accepted. If it is lower than 1, then it may be accepted, with a probability of less than 1 [3]. The adaptive Metropolis-Hastings algorithm improves upon this by scaling the proposal distribution by a factor a such that a specific quantity of samples are accepted — usually 25%. This speeds up convergence to the entire distribution of $p(\boldsymbol{\theta}|\mathbf{y_D})$ because it allows the algorithm to escape areas of low-scoring parameter space (when the acceptance rate is too low), but also increases the ability of the algorithm to escape local minima (when the acceptance rate is too high, so the algorithm samples parameter space more coarsely) [4]. The algorithm is shown below without burn-in, although in practice the first few thousand samples will be discarded. Exponential smoothing, as introduced in the CMAES section, is again introduced in the updates to the mean μ_t, covariance matrix $\mathbf{\Sigma}_t$ and step size a_t, using the learning parameter γ_t to incorporate information from previous generations when the number of iterations is low.

```
$\log(a_0) \leftarrow 0$;
$\boldsymbol{\theta}_0 \leftarrow$ initial parameter values;
$\boldsymbol{\Sigma}_0 \leftarrow$ a diagonal matrix;
$\boldsymbol{\mu}_0 \leftarrow \boldsymbol{\theta}_0$ $t \leftarrow 0$;
while $t < t_{final}$ do
    $\boldsymbol{\theta}' \sim \mathcal{N}(\boldsymbol{\mu}_t, a_t\boldsymbol{\Sigma}_t)$;
    if $p(\boldsymbol{\theta}') \neq 0$ then
        $\alpha \leftarrow \min\left\{\frac{p(\mathbf{y_D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathbf{y_D}|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t)}, 1\right\}$;
        $u \sim \mathcal{U}(0, 1)$;
        if $u < \alpha$ then
            $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}'$;
            accepted $\leftarrow 1$;
        else
            $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$;
            accepted $\leftarrow 0$;
        end
    else
        $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$;
        accepted $\leftarrow 0$;
    end
    $\gamma_t \leftarrow t^{-0.6}$;
    $\boldsymbol{\Sigma}_{t+1} \leftarrow (1 - \gamma_t) \times \boldsymbol{\Sigma}_t + \gamma_t \times (\boldsymbol{\theta}_{t+1} - \boldsymbol{\mu}_t)^T(\boldsymbol{\theta}_{t+1} - \boldsymbol{\mu}_t)$;
    $\boldsymbol{\mu}_{t+1} \leftarrow (1 - \gamma_t) \times \boldsymbol{\mu}_t + \gamma_t \times \boldsymbol{\theta}_{t+1}$;
    $\log(a_{t+1}) \leftarrow \log(a_t) + \gamma_t \times (\text{accepted} - 0.234)$
end
```

Algorithm 1: The adaptive Metropolis-Hastings algorithm

3.0.1 Intuition for adaptive MCMC

The purpose of algorithm 1 is to obtain an approximation of the "true" distribution of parameters that describe our data. Each parameter that we fit has its own distribution (that we assume is normal), with their a mean μ and variance σ. These are the marginal distributions of the multivariate normal distribution, defined by a vector of means $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. For each iteration of the algorithm:

1. A proposal vector of new means ($\boldsymbol{\theta}'$) is drawn from the current multivariate normal distribution

2. If this proposal is within the prior distribution (i.e. $p(\boldsymbol{\theta}) \neq 0$), then we compute the acceptance probability (which is the ratio of the likelihood multiplied by the prior for the proposal and current vector of means).

    - Because the quantity $p(\mathbf{y_D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')$ is proportional to the posterior probability $p(\boldsymbol{\theta}'|\mathbf{y_D})$, the ratio of this quantity for $\boldsymbol{\theta}'$ and $\boldsymbol{\theta}$ is a measure of which parameter set has a higher posterior probability, or in other words, which set better describes the data

3. If this ratio is greater than 1 then the candidate vector $\boldsymbol{\theta}'$ is a better explanation for the data than the current vector and is accepted.
4. If not, then a random, variable is drawn from a uniform distribution between 0 and 1.
5. If this random number is smaller than the ratio value α then the candidate vector $\boldsymbol{\theta}'$ is accepted, if not it is rejected.
6. A new covariance matrix is estimated from the difference between the accepted parameter vector and the mean of the previous iteration, and then added to the covariance matrix of the previous generation. The two matrices are scale by γ_t, so when t is small, so previous generations are weighted more highly. The contribution from previous iterations is reduced as t increases.
7. The accepted parameter is combined with the previous mean in a similar fashion to generate the new mean.
8. Finally, a_t is updated. This quantity is used to ensure the rate of acceptance of proposed candidate vectors is 0.234.
    - When a parameter vector is accepted, the quantity of a_t increases. Because this quantity scales the covariance matrix during the "drawing" of new parameter vectors, the size of steps taken in parameter space will increase, which will increase the number of low-scoring parameter vectors drawn.
    - Consequently, the value will decrease with rejections of the parameter vector, shrinking the covariance matrix.

The purpose of this algorithm is to take steps towards higher-likelihood areas of parameter space, and to maintain a particular rate of acceptance. By retaining a history of the accepted parameter vectors, we can infer information about these high-likelihood areas, which become our inferred distributions.

References

[1] W. J. Den Haan. *Numerical Integration.* 2011.

[2] W. Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[3] Christian P. Robert and George Casella. "The Metropolis—Hastings Algorithm". In: Monte Carlo Statistical Methods. Springer, 1999, pp. 231–283.

[4] Heikki Haario, Eero Saksman, and Johanna Tamminen. "An adaptive Metropolis algorithm". In: *Bernoulli* 7.2 (2001), pp. 223–242.