Project B2: Solving Quantum Systems Numerically

Haow Jern, Tee & 01059846

December 18, 2017

Abstract

Numerical Integration was performed for an exponential gaussian function using Trapezoidal & Simpson's rules, Monte Carlo methods with flat and important sampling. Monte Carlo with adaptive sampling was attempted but not successful in its implementation. Estimates of the integral, its errors, and number of evaluations were obtained and compared and Simpson's rule was found to be the most efficient in providing results of the same accuracy.

1 Introduction

The time-independent, 1D wavefunction of a particle is given by

$$\phi = \frac{1}{\sqrt[4]{\pi}} \exp^{ia(z)} \exp^{-\frac{z^2}{2}} \tag{1}$$

where a(z) is a phase function.

The probability of this particle being between z = a, and z = b is the integral of the square of absolute value of Equation 1, henceforth referred as I. As a result of the absolute operator, the e^{iaz} term disappears. Hence, the probabilistic equation is

$$P(z) = \int_a^b \frac{1}{\sqrt[2]{\pi}} \exp^{-z^2} dz \tag{2}$$

where for this paper, $a = 0$ and $b = 2$.

However, this function cannot be integrated analytically. Hence, numerical methods must be used to approximate the solution. The purpose of this paper is to explore and compare two main ideas of integration - Newton-Coates Rules and Monte Carlo Integration in 1 dimension.

The general idea for the algorithms are to continuously compute an estimate for the integral until the estimate is within a user-supplied relative accuracy. To avoid confusion, this stopping condition will be referred to as the stopping accuracy, ϵ_s and the relative accuracy, ϵ_r will now be defined as

$$\epsilon_r = \frac{\epsilon_I}{I} \tag{3}$$

where ϵ_I is the error of the estimate I.

Validation was performed using $f(z) = z^2$ from 1 to 2, and were verified analytically to be true. There were some numbers that were not fully accurate, e.g. 3.99999 instead of 4.0 but this was attributed to arithmetic floating point error.

2 Methods

2.1 Quadrature Methods - Newton-Coates

The simplest way of approximating an integrand is to perform a Riemann sum, i.e. divide the integrand into rectangles of the same width, and sum it up together. Newton-Coates methods utilises the same idea, but with different ways of interpolating between the top points of the rectangles. The two main methods that are explored here is the Trapezoidal and the Simpson's rule, using linear and parabolic interpolation respectively.

2.1.1 Trapezoidal Rule

Trapezoidal rule describes the integral between two points, z_i and z_{i+1}. To include the whole integral, the Trapezoidal rule is combined for all points, leading to the Extended Trapezoidal Rule:

$$\int_{z_0}^{z_{n-1}} f(z)dx = h(\frac{1}{2}f(z_0) + f(z_1) + ...f(z_{n-2}) + \frac{1}{2}f(z_{n-1})) + \epsilon_I \tag{4}$$

where h is the smallest width between successive points, ϵ_I is the error of the estimate and is the difference between the approximated and true value of the integral.

For each additional point that is used, the estimates are improved in accuracy. Hence the stopping condition for one function call is to compute two estimates, return the better estimate if they are within ϵ_s, and recursively call itself again if they are not. The full algorithm is provided in the Lecture Notes, Chapter 10, page 79 for reference. The stopping condition is given by

$$\epsilon_s = \frac{I_2 - I_1}{I_1} \tag{5}$$

where I_2 is more accurate than I_1 through the use of an additional point.

2.1.2 Simpson's Rule

Simpson's rule differs from Trapezoidal rule by the addition of one point to form its first estimate. The tops of the equal width samples are interpolated parabolically instead of linearly. The extended Simpson's Rule is given by

$$\int_{z_0}^{z_{n-1}} f(z)dx = h(\frac{1}{3}f(z_0) + \frac{4}{3}f(z_1) + \frac{2}{3}f(z_2) + \frac{4}{3}f(z_3)... + \frac{2}{3}(f(z_{n-3}) + \frac{4}{3}f(z_{n-2}) + \frac{1}{3}f(z_{n-1})) + \epsilon_I \tag{6}$$

However, the Simpson's rule is designed so that it can use the existing Trapezoidal Rule to implement the algorithm, given in the Lecture Notes. Hence, subsequent error analysis and stopping rules are similar with Trapezoidal Rule.

2.2 Monte-Carlo Methods

Monte-Carlo methods refer to algorithms that use random sampling to obtain numerical results. The most basic sampling is when samples are randomly drawn from a uniform distribution, otherwise known as Flat Sampling. Drawing from non-uniform distributions is known as Importance Sampling.

2.2.1 Flat Sampling

Following a similar treatment as provided by the lecture notes, I is given by

$$I = \frac{V}{N}\sum_{i=1}^{N} f(z_i) \tag{7}$$

where V is the integration volume, N is the number of samples, z_i are samples drawn from a uniform distribution.

In this case, the uniform distribution ranges from 0 to 2. Hence, the volume of integration is 2, as we are only using values from one dimension.

The error for one estimate, is the standard deviation, σ_I given by

$$\sigma_I = \frac{V}{\sqrt{N}}\sigma_{f_i} \tag{8}$$

$$\sigma_{f_i}^2 = \frac{1}{N-1}\sum_{i=1} N(f(x_i) - < f >)^2 \tag{9}$$

where $< f >$ is $I \times V$.

The stopping condition is when the relative accuracy,

$$\epsilon_r = \frac{\sigma_I}{I} \tag{10}$$

is smaller than the user-supplied accuracy. The general algorithm will be to continuously increment N, calculate I, and return I if this condition is satisfied.

However, a problem lies as I will vary according to the samples chosen, and there will be a distribution of I due to this variation. Hence, obtaining one estimate is not sufficient. The distribution in obtaining M estimates will be a Gaussian as M gets large due to the Central Limit Theorem[1]. This is somewhat observed on Figure 1. Notice also how the width of this Gaussian, σ_M decreases as the number of samples, N increases.

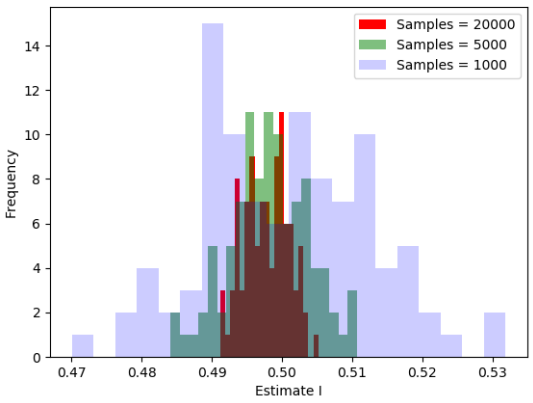


Figure 1: Demonstration of the Gaussian Distribution of Estimate I due to the Central Limit Theorem. As number of samples N increases, the width of the Gaussian decreases.

Imagine that it is possible to obtain large M number of estimates, I_j. These will then follow a Gaussian distribution, and hence for the best estimate of I, it can be obtained by using the mean:

$$I = \frac{1}{M}\sum_{j=1}^{M} I_j^2 \tag{11}$$

The variance is then given as

$$\sigma_M^2 = \frac{1}{M}\sum_{j=1}^{M} \sigma_{I_j}^2 \tag{12}$$

where σ_{I_j} is the error in each estimate I_j.

In practice however, obtaining large M is not practical as Monte Carlo methods are very computationally heavy in general. However, if successive values of σ_{I_j} are close to each other, Equation 12 can be written as

$$\sigma_M^2 \approx \frac{M \times \sigma_I^2}{M} \tag{13}$$

The $M's$ can be canceled out, and now the variance in multiple estimates is approximately the same as the variance in just one estimate. From Figure 2, it is shown that multiple σ_{I_j}'s converge together when N increases, proving that it is possible to use the above approximation at large enough N.

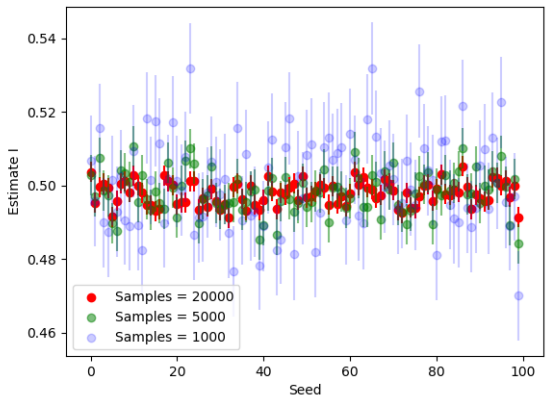


Figure 2: Relationship of Estimate I with independent set of samples. Each estimate I, represented by a point has an error given by its standard deviation σ_I, shown by the error bars. However, due to Central Limit Theorem, the estimates follow a Gaussian Distribution, where the width, σ_m decreases as number of samples used increases. Each seed represents a different set of sample.

Now, the algorithm will be the same as before but with an extra condition. That is if the stopping condition is satisfied, compute a second estimate using an independent set of samples with the same N. Find the difference between those two variances, and check if it is within a user-specified accuracy, ϵ_m. ϵ_m is usually chosen to be 10^{-2}, as this is deemed to be small enough for the approximation in Equation 13 to hold, and not too small that it may take too much computational time due to the inverse relationship from Figure 3 and Equation 8.

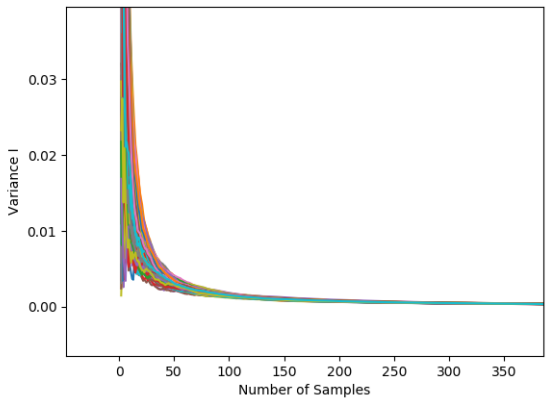


Figure 3: Relationship between Variance σ_I^2 and Number of Samples, N. Each coloured line represents different independent set of samples used, which was set by using different random seeds. In total, 100 independent sets were used. σ_I^2 varies wider for different estimates at smaller N, and quickly converges after 100 samples used.

2.2.2 Importance Sampling

Importance Sampling is one way to reduce the variance of the estimate[2], and simultaneously improve computation time. When sampling from a uniform distribution, there may be many samples that are chosen where the integrand is small. A better method is to sample from a distribution, $g(x)$, where it is ideally proportional to the integrand, so more samples can be chosen where there is larger contribution to the integrand. This is known as Importance Sampling.

Sampling from the distribution was implemented using the Rejection method instead of the Transformation method which would have been faster. This is because it was not possible to invert the Cumulative Distribution Function that is required.

The importance sampling equation for the estimate I is

$$I = \frac{1}{N}\sum_{i=0}^{N}\frac{f(z)}{g(z)} \tag{14}$$

where z is sampled from the $g(x)$ distribution. $g(x)$ was chosen to be

$$g(x) = -0.48x + 0.98 \tag{15}$$

For comparison purposes later, a non-proportional sampling function is also implemented, given by

$$g(x) = +0.48x + 0.02 \tag{16}$$

2.2.3 Adaptive Importance Sampling

From Importance Sampling, the challenge lies in choosing a correct distribution that mimics the integrand. Adaptive Importance Sampling solves this by improving $g(x)$ in each iteration by using a previous set of samples. There are many methods, notably Vegas, Miser and Mixtures of Products of Beta Distributions [3],[4],[5], but in this project, the Vegas algorithm was implemented.

The idea is to divide the integrand into n rectangles of width δz_i and samples M times uniformly within each rectangle. As initially each rectangle is the same width, they have the same initial probability distribution. After each iteration, these widths can decrease or increase in size, changing the probability in each rectangle. The estimate in one iteration can be calculated by summing the estimates in each rectangle using Equation 14, where $g(z) = \frac{1}{\delta z_i}$, which reduces to a flat Monte Carlo Integration.

To obtain the final estimate after N iterations, each I_i value is weighed by its respective inverse variance, $w_i = \frac{1}{\sigma_{I_i}}$, so estimates with lower variance will contribute more to the final result.

This division into multiple rectangles, is known as stratified sampling[3]. By dividing in this way, the resulting variance is lower. This is because the variance in each region depends on the mean estimate of the region, and because samples drawn are closer to each other in one region than drawing the same number of

samples throughout the whole domain, the variance in every region is reduced, and subsequently the final estimate's variance is also reduced.

Each rectangle region is subdivided into m_i subregions given by

$$m_i = K \frac{< I_i > \delta_{x_i}}{\sum_{i=1}^{N_b ins} < I_i > \delta_{x_i}} \tag{17}$$

where K is a constant that changes the convergence speed of the algorithm. m_i is larger in the region where its estimate contributes the most to the total estimate.

These subregions are then regrouped into n rectangles again. Through this way, the probabilities of each region will change depending on its δ_{x_i} and the overall distribution will tend to follow the integrand. However, subdividing and regrouping can lead to oscillations in the rectangles, leading to unwanted results. A damping term[5] is included to avoid this, and Equation 17 is now written as

$$m_i = K[(\frac{< I_i > \delta_{x_i}}{\sum_{i=1}^{n\_regions} < I_i > \delta_{x_i}} - 1) \times (\log \frac{< I_i > \delta_{x_i}}{\sum_{i=1}^{n\_regions} < I_i > \delta_{x_i}})^{-1}]^{\alpha} \tag{18}$$

where α is usually between $0.2 - 2$.

3 Results & Discussion

3.1 Trapezoidal and Simpson's Rules

Trapezoidal and Simpson's rules were implemented successfully for different errors and their estimates were obtained in Table 1. Simpson's rule is shown to be significantly better with at least a 1000 times more accurate than using Trapezoidal rule.

From testing the code, the program ran smoothly but too quickly to explicitly measure the time taken for both methods. However, this can be inferred by measuring the number of functions calls, as both methods are recursive. From Table 1, Simpson's rule outperforms by using about half of the function calls needed by the Trapezoidal rule to obtain the same stopping accuracy. Therefore, Simpson's rule is a significantly better algorithm in terms of accuracy and efficiency.

From Figure 4, as lower relative accuracies are specified, the total function calls required grows exponentially, This suggests that both algorithms will perform exponentially slower as smaller accuracies are required.

Using scipy's curve_optimize function, the following fitting equations are obtained.

$$trapezoidalfit = \frac{a}{x^b} + c \tag{19}$$

where a, b and c are found to be 2.966879, 0.490854 & -7.055248 respectively.

$$simpsonfit = \frac{a}{x^b} + c \tag{20}$$

where a, b and c are found to be 1.380569, 0.509304 & 1.708078 respectively.

$\epsilon_s = 1e-6$

Rules	Integral Estimate	Error	Total Function Calls
Trapezoidal Rule	0.49766111	$1e-8$	2599
Simpson's Rule	0.49766113247930	$5e-15$	1477

$\epsilon_s = 1e-5$

Trapezoidal Rule	0.49766074	$4e-8$	833
Simpson's Rule	0.497661131494	$1e-12$	493

$\epsilon_s = 1e-4$

Trapezoidal Rule	0.4976584	$6e-7$	277
Simpson's Rule	0.4976611037	$3e-10$	155

$\epsilon_s = 1e-3$

Trapezoidal Rule	0.49763	$1e-5$	81
Simpson's Rule	0.497660641	$1e-9$	47

Table 1: Results of Trapezoidal and Simpson's Rule when subsequent integral estimates are within specified relative accuracy, ϵ, as defined in Equation 5.

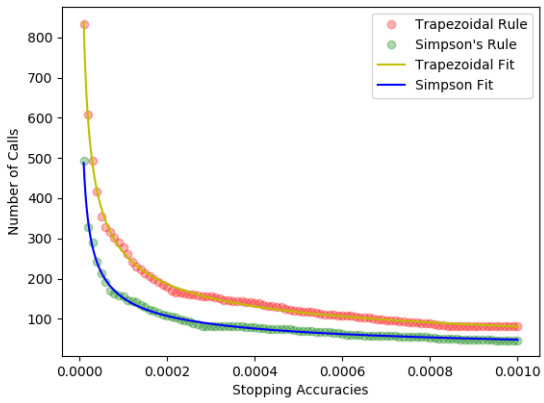


Figure 4: Relationship between number of function calls and different stopping accuracies. 300 different values ranging from $1e-5$ to $1e-3$ were used. Simpson's rule is more accurate than Trapezoidal Rule for all of the points. The $\frac{1}{\sqrt{x}}$ curve suggests that better estimates will require an exponentially increasing time to calculate.

3.2 Monte Carlo Methods

Monte Carlo with Flat and Importance Sampling has been successfully implemented and the table of results for different specified accuracies are shown in Table 2. Importance sampling with a non-proportional PDF is also implemented for the error $1e-2$, it takes much longer than flat sampling, and highlights the problems of choosing a non-proportional PDF. Some smaller accuracies were not shown there as they required too long to obtain. Hence, a graph was plotted to show the relationship between number of evaluations and the stopping accuracy required. This is shown in Figure 5

Using scipy's curve_optimize function, the following fitting equations were obtained.

$$flatfit = \frac{a}{x^b} + c \tag{21}$$

where a, b and c are found to be 0.615912, 1.999049 & -86.534024 respectively.

$$importancefit = \frac{a}{x^b} + c \tag{22}$$

where a, b and c are found to be 0.615912, 2.027688 & -86.534024 respectively.

From the fitting equations 21 and 22, estimations of the number of evaluations required for smaller accuracies can be obtained, as shown in Table 3, and the number of calls is proportional to $\frac{1}{x^2}$, where x is the relative accuracy desired.

As Importance Sampling takes almost an order of magnitude smaller number of evaluations compared to Flat Sampling, it can be concluded that Importance Sampling is more efficient method. However, as the number of evaluations scale proportional to $\frac{1}{\sqrt{x}}$ for the Newton-Coates methods, Monte-Carlo methods are not as efficient as Newton-Coates methods for the problem at hand.

$\epsilon_s = 1e-3$

Sampling Method	Integral Estimate	Error	Total Evaluations
Flat	0.497814	0.000498	610230
Importance	0.498639	0.000499	72550

$\epsilon_s = 1e-2$

Flat	0.49391	0.00494	6160
Importance	0.50404	0.00418	1000
Importance (Non-Proportional)	0.502089	0.00502	63670

Table 2: Results of using Monte Carlo Integration with Flat and Importance Sampling when they are within the specified stopping accuracy. In the Monte Carlo methods, stopping accuracy is the relative accuracy, as defined in Equation 3. Smaller errors were not practical to obtain as it takes too much computation time.

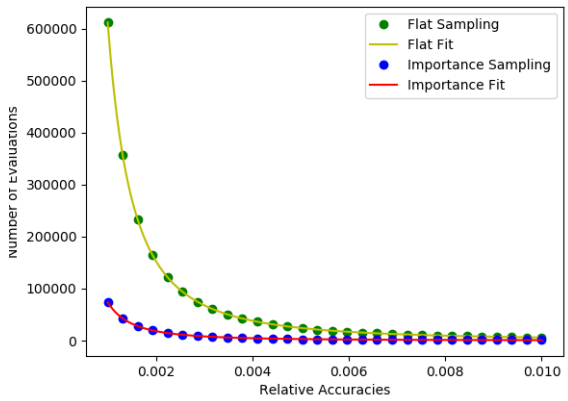


Figure 5: Relationship between number of evaluations and different stopping accuracies. 30 different values ranging from $1e-2$ to $1e-3$ were used. Importance Sampling performs much better than flat sampling at smaller accuracies. Equations of fitting were implemented using Scipy's Optimize_Curve function, and number of evaluations at smaller accuracies can be extrapolated through this.

$\epsilon_s = 1e-6$

Sampling Method	Total Evaluations
Flat	6.08×10^{11}
Importance	8.93×10^{10}

$\epsilon_s = 1e-5$

Flat	6.09×10^{9}
Importance	8.38×10^{8}

$\epsilon_s = 1e-4$

Flat	6.11×10^{7}
Importance	7.86×10^{6}

Table 3: Estimation of number of evaluations required at specified stopping accuracies using Monte Carlo Integration with Flat and Importance Sampling when they are within the specified stopping accuracy.

Adaptive Sampling has been attempted, however the results obtained do not seem to be consistent with previous results. Using $\alpha = 0.5$, total iterations allowed = 10, number of samples = 2000, n_regions = 100 $K = 10000$, Table 4 was obtained. Compared with the results from Simpson's rules, it seems that the estimates using this adaptive method is is not as accurate, even though the errors reduce drastically for each iteration. This could just be due to the randomness of the numbers used.

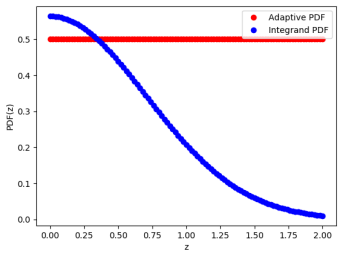
Besides that, it seems that it is possible to obtain a high accuracy from using very little iterations, due to the use of stratified sampling. Even so, this is merely speculation and no conclusions can be made.

However, the 'adaptive method' seems to be successfully implemented as shown in Figure 6. The adaptive PDF changes to be more proportional to the integrand's distribution after each iteration.

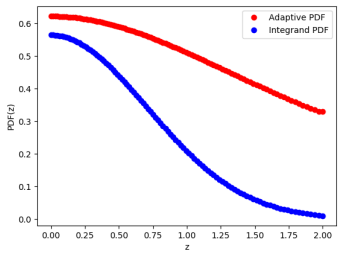
Adaptive

ϵ_s	Estimate	Error	Number of Evaluations
$1e-3$	0.49791	$4e-5$	100
$1e-4$	0.49791	$4e-5$	100
$1e-5$	0.497908	$6e-6$	1000
$1e-6$	0.497911	$6e-6$	1000

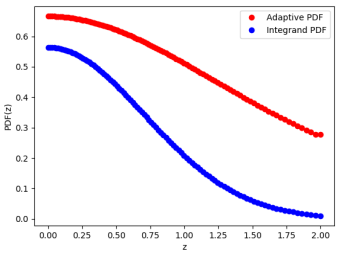
Table 4: Estimation of number of evaluations required at specified stopping accuracies using Monte Carlo Integration with Adaptive Sampling when they are within the specified stopping accuracy, ϵ_s



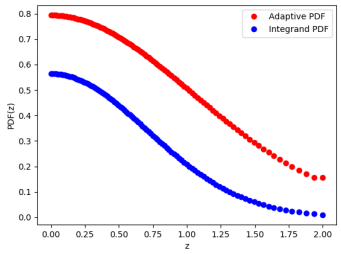
(a) 1st Iteration



(b) 2nd Iteration



(c) 3rd Iteration



(d) Final Iteration

Figure 6: Adaptive Method for $a = 0, b = 2$, number of updates = 10, number of samples in each region = 2000, number of regions = 100, $K = 10000$, alpha = 0.5. The Adaptive distribution changes every iteration to be more similar to the integrand's distribution.

4 Conclusion

In conclusion, Simpson's rule performs the best in using the least number of calls to obtain a result of a specified accuracy for the 1 dimension integrand. From exploring Newton-Coates methods, Trapezoidal Rule performs worse and offers no advantages compared to using Simpson's Rule; From exploring Monte Carlo Methods, Importance sampling is a more efficient method compared to using Flat sampling, but it requires a function with a distribution close to the integrand, hence the integrand should be known. It can however perform worse compared to using Flat Sampling when the function chosen is very different. Adaptive Sampling was not successfully implemented, but the adaptiveness of a distribution was. It solves the problem of Importance Sampling, where it is not possible identify the function, and has the advantages of Importance Sampling. However its drawback is that it requires a few initialisations to evolve the distribution, and this may end up taking more time that it is needed. In general, Monte Carlo methods performed worse compared to Newton-Coates methods, and Simpson's rule is the most efficient algorithm.

[Word Count = 2313]

References

[1] *Second Year Statistics of Measurement - Lecture 4 - Central Limit Theorem.* Paul Dauncey. Imperial College London. 13 October 2016.

[2] Art Owen. *Importance Sampling.* Stanford University. Available from: https://statweb.stanford.edu/~owen/mc/Ch-var-is.pdf [Accessed on 3 December 2017]

[3] Yi Zhou. *Adaptive Importance Sampling for Integration*. PhD Thesis. Stanford University. 1998.

[4] Man Suk Oh & James O. Berger. *Adaptive Importance Sampling in Monte Carlo Integration.* Purdue University. Vol.41, 143 - 168. 1989. Available from: http://www.tandfonline.com/doi/pdf/10.1080/00949659208810398?needAccess=true [Accessed on 5 December 2017.]

[5] G. Peter Lepage. *A New Algorithm Adaptive Multidimensional Integration* Stanford University. Journal of Computational Physics, 27, 92-203. 1978. Available from : https://ac.els-cdn.com/0021999178900049/1-s2.0-0021999178900049-main.pdf?_tid=cb492246-deb0-11e7-a4b4-00000aab0f6b&acdnat=1513023837_cfac6670d06843504197c603a22683f2 [Accessed on 6 December 2017]