22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia, 3 to 8 December 2017
mssanz.org.au/modsim2017

Polynomial Chaos for sensitivity analysis in wildfire modelling

James E. Hilton [a], **Alec G. Stephenson** [a] **Carolyn Huston** [a] **and William Swedosh** [a]

[a]*Data61, CSIRO, Clayton South, VIC 3169, Australia*
Email: James.Hilton@csiro.au

Abstract: Computational models for wildfire propagation rely on a number of input variables such as fuel state, weather conditions and landscape features. These are typically forecast, estimated or measured and each has an associated degree of variability or uncertainty. This variability of these input variables affects the output variables, or predictions, of the wildfire model. However, the relation between these is currently not well quantified in operational wildfire models. With a move towards ensemble and probabilistic forecast approaches for wildfire predictions quantifying the sensitivity between the variation in the input variables and the resulting outputs in an efficient manner is becoming increasingly important.

A straightforward method of quantifying such sensitivity is through a basic Monte Carlo approach. Given a set of inputs with respective random distributions, an ensemble of simulations can be run with input conditions drawn from these distributions. The aggregation of the ensemble members is then used to determine the distribution of the output variables. However, convergence can be relatively slow and require a large set of ensemble members. A more sophisticated technique is Polynomial Chaos, in which the statistical distribution of the outputs are represented using orthogonal polynomial series expansions. The method allows the output distribution to be reconstructed by picking input values at specific points, corresponding to Gaussian quadrature points, rather than randomly. This allows output distributions to be estimated using fewer computational simulations, and hence much more rapidly, than using a Monte Carlo approach.

In this study we implemented a Polynomial Chaos method using the Python NumPy library. The method provided input values to 'black box' simulations carried out in Spark, a wildfire modelling framework (http://research.csiro.au/spark/). The modelling set up used in this study was a point ignition propagating under a constant wind direction but variable strength and a variable fuel load. The McArthur model was used for the head fire rate of spread and the simulated duration of the fire was six hours from an initial point source. As a comparison, we also used the NumPy library to generate and aggregate ensemble members for a Monte Carlo simulation. The only output variable considered in the study was the arrival time of the fire at points in the spatial domain.

Sensitivity analysis using Polynomial Chaos for one uncertain input variable, wind strength, required just four simulations to reconstruct the distribution of the output variables and to calculate the mean and standard deviation of the arrival time output variable. In comparison, the Monte Carlo method slowly converged to the distribution taking around one hundred times more simulations to approach a comparable value for the mean. For a sensitivity analysis of both wind strength and fuel load the Polynomial Chaos method required sixteen simulations to reconstruct the distribution. The Monte Carlo method showed similar convergence behaviour to the previous test, requiring several hundred simulations to approach a comparable value for the mean. The Polynomial Chaos method clearly provided a far more efficient method for sensitivity analysis in wildfire applications.

Other strengths of the method include the ability to find the distribution of any number of outputs without running any additional simulations. The sensitivity of an entire field in space, such as the distribution in arrival time over an entire spatial domain at each point, can easily be evaluated. The number of simulations scale as the polynomial order to the power of the number of input variables, making the technique computationally intractable for large numbers of uncertain variables. However, more advanced quadrature techniques, not covered in the present study, could resolve this difficulty. We believe the method may be suitable in applications such as operational fire predictions as uncertainty in predictions can be rapidly assessed. The method also does not depend on any particular fire simulation algorithms and can be built as a framework around a 'black box' simulator.

Keywords: *Wildfires, sensitivity, modelling, polynomial chaos*

1 INTRODUCTION

The behaviour of a fire perimeter can be computationally modelled using empirical relations derived from experimental measurements (Sullivan, 2009). These empirical relations depend on several input parameters representing factors such as fuel condition, fuel type, surface elevation and wind variables. However, each of these input factors are subject to variation or uncertainty which affects the predicted outputs from the simulation. In this study, we use a method known as Polynomial Chaos estimation to assess the uncertainty in the outputs of a wildfire simulation resulting from the uncertainty in the input variables.

Polynomial Chaos uses an orthogonal polynomial series to represent the outputs of the model (Wiener, 1938; Sudret, 2008). The uncertainties in the model inputs are characterised using a probability density function (pdf) for each of the uncertain inputs. Although Monte Carlo estimation can be used for the same purpose, the usefulness of Polynomial Chaos lies in the ability to model a pdf of the outputs from sampling only a few possible input states at carefully selected positions (Gaussian quadrature points). As demonstrated in this study, Polynomial Chaos can give an estimation of the pdf of outputs using far fewer simulations than Monte Carlo, resulting in greater computational efficiency.

A further major advantage of the method is that it can be implemented over any spatial area for any number of output variables, allowing spatial maps of uncertainty to be constructed. This, as well as the ability for rapid calculation of uncertainty from only a small number of simulations, may make the method useful for operational wildfire prediction and planning purposes. In this study we outline the mathematics behind Polynomial Chaos, provide a basic guide to implementing the method and compare distributions calculated using the method to a Monte Carlo approach.

2 METHODOLOGY

Let the arrival time of a fire at a particular point in a two-dimensional domain be given by $\psi(X)$, where $X \in \mathbb{R}$ is an input into the model. The arrival time can be expanded as a polynomial:

$$\psi(X) \approx \sum_{i=0}^{\infty} \psi_i P_i(X), \tag{1}$$

where P are a sequence of orthogonal polynomials with respect to a chosen pdf $\rho(X)$ with support on some (possibly unbounded) interval $[p, q]$, satisfying:

$$\int_p^q P_i(X) P_j(X) \rho(X) dX = 0 \quad \text{for } i \neq j, \tag{2}$$

and $P_0(X) = 1$. Polynomials satisfying Eq. (2) include Hermite, Laguerre, Jacobi and Legendre polynomials, amongst others and the interval $[p, q]$ depends on the particular polynomial used. Integration of Eq. (1) and using Eq. (2) gives the coefficients ψ_j as:

$$\psi_j = \frac{\int_p^q \psi(X) P_j(X) \rho(X) dX}{\int_p^q P_j^2(X) \rho(X) dX}. \tag{3}$$

The numerator of Eq. (3) can be evaluated using Gauss quadrature:

$$\int_p^q \psi(X) P_j(X) \rho(X) dX \approx \sum_{k=0}^{N-1} \psi(X_k) P_j(X_k) \omega_k, \tag{4}$$

where the inputs are evaluated at N Gauss quadrature points with weights ω_k which depend on the choice of polynomial. The denominator of Eq. (3) is usually available in closed form. Once the coefficients ψ_j are estimated using Eq. (3), the mean arrival time can be estimated as:

Table 1. Two dimensional polynomial expansion up to third degree

Degree	j	$P_j(X^1, X^2)$	a	b
0	0	$P_0(X^1)P_0(X^2)$	0	0
1	1	$P_0(X^1)P_1(X^2)$	0	1
1	2	$P_1(X^1)P_0(X^2)$	1	0
2	3	$P_0(X^1)P_2(X^2)$	0	2
2	4	$P_1(X^1)P_1(X^2)$	1	1
2	5	$P_2(X^1)P_0(X^2)$	2	0
3	6	$P_0(X^1)P_3(X^2)$	0	3
3	7	$P_1(X^1)P_2(X^2)$	1	2
3	8	$P_2(X^1)P_1(X^2)$	2	1
3	9	$P_3(X^1)P_0(X^2)$	3	0

$$\mathbb{E}[\psi(X)] = \int_p^q \psi(X)\rho(X)dX = \sum_{j=0}^{\infty} \psi_j \int_p^q P_j(X)\rho(X)dX = \psi_0, \tag{5}$$

and the variance in arrival time can be estimated as:

$$\mathbb{V}[\psi(X)] = \int_p^q (\psi(X) - \mathbb{E}[\psi(X)])^2 \rho(X)dX = \sum_{j=1}^{\infty} \psi_j^2 \int_p^q P_j^2(X)\rho(X)dX. \tag{6}$$

It should be noted that the sum is taken from one in Eq. (6), rather than zero. For practical calculation the sums in Eq. (5) and Eq. (6) are truncated at a chosen maximum polynomial degree.

The analysis can be extended to multiple inputs. Let X^1 and X^2 be two independent inputs with pdfs $\rho(X^1)$ and $\rho(X^2)$, respectively. The two-dimensional orthogonal polynomials $P_j(X^1, X^2)$ are given by all possible products of one-dimensional polynomials $P_a(X^1)P_b(X^2)$, where a and b are the degree of the one-dimensional polynomials. If the two-dimensional polynomials are ordered by the maximum degree of the one-dimensional polynomials, $P_j(X^1, X^2)$ can be expressed using the combinations given in Table 1. Note that the index j in Table 1 and the following equations refer to the index of the particular combination, rather than the degree of the polynomial. The two-dimensional expansion of Eq. (1) becomes:

$$\psi(X^1, X^2) \approx \sum_{j=0}^{\infty} \psi_j P_j(X^1, X^2). \tag{7}$$

As the inputs are assumed independent the orthogonality rule still holds:

$$\int_p^q \int_p^q P_i(X^1, X^2)P_j(X^1, X^2)\rho(X^1, X^2)dX^1dX^2 = \int_p^q P_a(X^1)P_b(X^1)\rho(X^1)dX^1 \int_p^q P_c(X^2)P_d(X^2)\rho(X^2)dX^2 = 0 \quad \text{for } i \neq j, \tag{8}$$

where $P_i(X^1, X^2) = P_a(X^1)P_c(X^2)$ and $P_j(X^1, X^2) = P_b(X^1)P_d(X^2)$. Eq. (8) gives the two-dimensional polynomial coefficients ψ_j as:

$$\psi_j = \frac{\int_p^q \int_p^q \psi(X^1, X^2)P_j(X^1, X^2)\rho(X^1, X^2)dX^1dX^2}{\int_p^q \int_p^q P_j^2(X^1, X^2)\rho(X^1, X^2)dX^1dX^2}, \tag{9}$$

and the quadrature becomes the product of one-dimensional quadratures:

$$\int_p^q \int_p^q \psi(X^1, X^2) P_j(X^1, X^2) \rho(X^1, X^2) dX^1 dX^2 \approx \sum_{l=0}^{M} \left(\sum_{k=0}^{N-1} \psi(X_k^1, \psi_l^2) P_a(X_k^1) \omega_k \right) P_b(X_l^2) \omega_l. \quad (10)$$

The methodology can be extended to an arbitrary number of dimensions, but becomes computationally intractable for dimensions above three or four using standard Gaussian quadrature. Above this, specialised higher order quadrature methods must be used. These are not implemented in the current study, so the number of uncertain inputs here is restricted to below this number. However, this is usually suitable for the major sources of variation (wind characteristics, temperature and fuel load) in wildfire modelling.

3 APPLICATIONS

Application of the method to a simple wildfire scenario is demonstrated in the following examples. The uncertain inputs, X, used in the examples were the wind strength, s, and fuel load, f, and the output, ψ, was the arrival time of a fire. All other input variables were fixed. The fire was simulated using the McArthur empirical model (McArthur, 1967) with parameters given in Table 2. The shape of the fire was simulated using an elliptical model with a backing ratio of 2% of the head fire speed and a backing ratio of 15% of the head fire speed. The simulation used a cell resolution of 10 m and duration of 6 hours.

The simulation was run as a 'black box' in which values for the uncertain inputs were determined by an external script and fed to a fire propagation simulator (Spark, Hilton *et al*. (2015)). The simulator ran using the given inputs and produced an arrival time at each point of the domain. An example output from a simulation run is shown in Fig. 1a, where the isochrones show a plan view of the progression of a fire. The initial starting point was a circle of radius 30 m.

In these examples the inputs were assumed to be normally distributed, so the probabilists' Hermite polynomials, H_{ej}, were used as the basis, where $[p, q] = [-\infty, \infty]$ and the pdf was given by:

$$\rho(X) = \frac{1}{2\pi} e^{-\frac{X^2}{2}}. \quad (11)$$

Using Eqs. (3) and (4) with N Gauss quadrature points $X_0, ..., X_{N-1}$ gives:

$$\psi_j \approx \frac{1}{j!} \sum_{k=0}^{N-1} \psi(X_k) H_{ej}(X_k) \omega_k, \quad (12)$$

where the following analytic relation for the denominator of Eq. (3) was used:

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} H_{ej}^2(X) e^{-\frac{X^2}{2}} dX = j! \,. \quad (13)$$

This gives the variance as:

$$\mathbb{V}[\psi(X)] = \sum_{j=1}^{N-1} \psi_j^2 j! \,. \quad (14)$$

The analysis can be repeated for two dimensions, giving the variance for two uncertain inputs as:

$$\mathbb{V}[\psi(X^1, X^2)] = \sum_{j=1}^{N-1} \psi_j^2 a! b! \,. \quad (15)$$

The general procedure for applying Polynomial Chaos in one dimension is:

Table 2. Parameter values for McArthur model

Parameter	Value
Wind direction	$180°$
Wind strength	s m s^{-1}
Temperature	$25°$
Relative humidity	30%
Drought factor	5
Fuel load	f t ha^{-1}

1. Decide the uncertain input variable X to be taken into account and choose the pdf $\rho(X)$ for the input.
2. Choose a maximum degree for the polynomial expansion, N, depending on the required accuracy of the solution. In this study $N = 4$ was chosen.
3. Calculate the quadrature points for the input pdf, $X_0, ..., X_{N-1}$. The number of terms in the quadrature can be restricted to the maximum degree for the polynomial expansion, resulting in a vector of quadrature points of length N.
4. Map X to the input distribution. For example, a normal distribution with a mean input value s_{mean} and standard deviation s_{stdev}, can be mapped as $s = s_{\text{mean}} + s_{\text{stdev}}X$, where $\mathbb{E}[X] = 0$ and $\mathbb{V}[X] = 1$.
5. Carry out N simulations for each value within the vector s resulting in a vector of output values $\psi(X_k)$.
6. Calculate ψ_j using the quadrature formula given in Eq. (12).
7. The mean value of the output is ψ_0, and the standard deviation can be calculated from Eq. (14).

To assess the efficiency of Polynomial Chaos, a separate external script was used for a Monte Carlo set-up in which simulations were run with randomised values. The Monte Carlo results were compared to the Polynomial Chaos results. All external scripting used Python libraries and the random numbers were generated using the standard NumPy Mersenne Twister implementation (Matsumoto *et al*., 1998). An example Python code implementation is given in the Appendix.

3.1 One dimensional results

In this example the wind strength was assumed to have an uncertain value and the one-dimensional formulation was used. The wind strength was assumed to have a normal distribution with a mean of 40 km h^{-1} and standard deviation of 10 km h^{-1}, such that $s = 40 + 10X$, where X is distributed as a standard normal. The output ψ was the time taken for a fire to reach a measurement location 1 km upwind of a starting location, shown schematically in Fig. 1a. The maximum polynomial degree was set to four, requiring four simulations for the Polynomial Chaos method.

A comparison between the Polynomial Chaos approximation to the arrival time dependency on the wind strength and a Monte Carlo estimate is shown in Fig. 1b. The four output values from the Polynomial Chaos method are shown as large circles on the main plot. The polynomial approximation to the dependency curve between arrival time and wind strength is plotted as a solid line, calculated from Eq. (1). The results from the Monte Carlo simulations are shown as individual crosses.

It can be seen that the Monte Carlo points lie directly on the polynomial approximation to the dependency curve showing the two methods match, as expected. The convergence of the Monte Carlo simulations to the mean arrival time value is shown in the inset of Fig. 1b. The dashed line on the plot shows the mean arrival time from the Polynomial Chaos method, calculated using Eq. (5). As expected, the Monte Carlo converges to this value but takes several hundred simulations to approach this result. In comparison the Polynomial Chaos approximation only requires four simulations.

3.2 Two dimensional results

In this example the wind strength and fuel load were both assumed to have uncertain values with normal distributions. The wind strength was assumed to have the form $s = 40 + 10X^1$, and the fuel load was assumed to have the form $f = 25 + 5X^2$ where X^1 and X^2 were distributed as standard normals. The

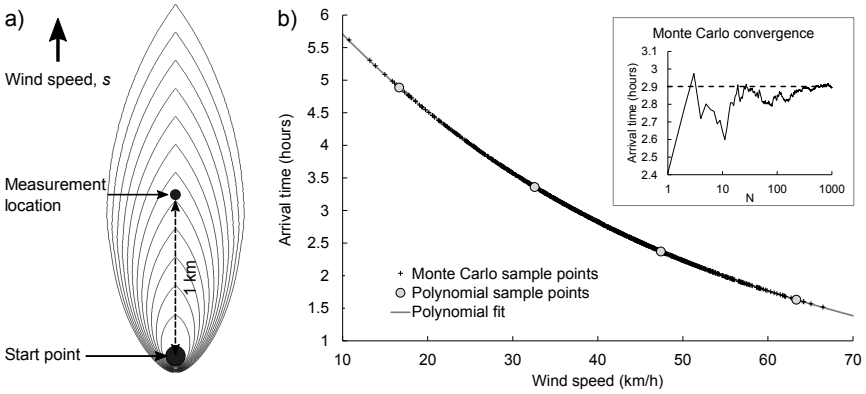

Figure 1. a) McArthur simulation, each curved line is an hourly isochrone showing the fire perimeter. b) Comparison of Monte Carlo method with Polynomial Chaos approach for wind strength variation. Inset: convergence of mean value of arrival time at measurement location from Monte Carlo method. Dashed line shows the mean value calculated from the Polynomial Chaos approach.

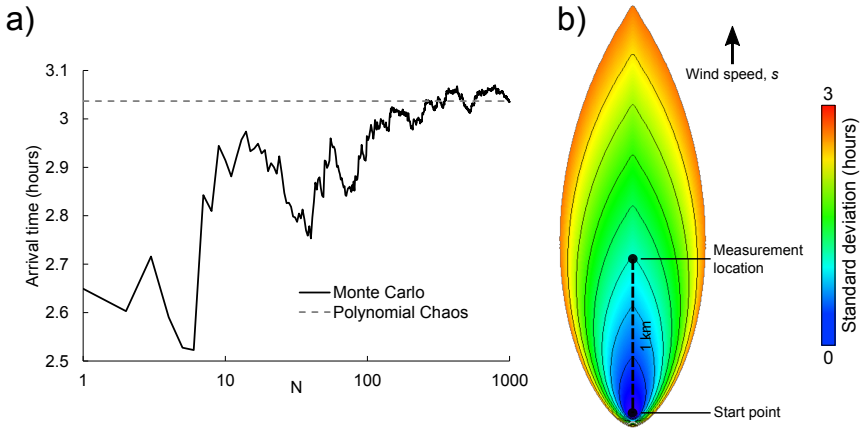

Figure 2. a) Convergence of mean value of arrival time at measurement location from Monte Carlo method for variation in wind strength and fuel load. Dashed line shows the mean value calculated from the Polynomial Chaos approach. b) Colour shaded map of the standard deviation of arrival time with superimposed hourly isochrones of the mean arrival time.

maximum polynomial degree was set to four, requiring sixteen simulations for the Polynomial Chaos method. Convergence of the mean value from the Monte Carlo simulation is shown in Fig. 2a, where the mean value from the Polynomial Chaos method is shown as the dashed horizontal line. As with the one-dimensional example the Polynomial Chaos method only requires sixteen simulation for calculation of the mean arrival time, whereas the Monte Carlo method requires several hundred simulation to converge to an estimated mean value.

Once the simulations have been run the method can easily be extended to calculate the mean and variance in arrival time at each point in the domain, rather than just at a single point. This requires evaluation of the ψ_j coefficients at each point which has a significant computational overhead. However, each point can be calculated in parallel making the computation suitable for the GPU-based GeoStack module within the Spark framework. Results for the test case are shown in Fig 2b, where hourly isochrones of the mean arrival time are shown superimposed over a colour shaded image of the standard deviation in arrival time at each point. In this simple example the uncertainty in arrival time increases as the fire progresses.

Table 3. Mean and standard deviation of the fire arrival time (in hours) from the Monte Carlo and Polynomial Chaos approaches for one and two input variable examples.

	One input variable		Two input variables	
	$\mathbb{E}[\psi(s)]$	$\sqrt{\mathbb{V}[\psi(s)]}$	$\mathbb{E}[\psi(s,f)]$	$\sqrt{\mathbb{V}[\psi(s,f)]}$
Monte Carlo (hrs) ($N = 1000$)	2.895	0.660	3.035	0.971
Polynomial Chaos (hrs)	2.901	0.691	3.036	1.026

4 CONCLUSIONS

Polynomial Chaos is a promising technique for incorporating variation of input variables into wildfire models. The technique is more precise than the Monte Carlo approach, and requires far fewer simulation runs to arrive at an accurate estimate of mean arrival times and the variation in the arrival times. The examples in this study demonstrated applicability to assessing the dependence of arrival time of a fire to variation in wind strength and fuel load. The method can also be easily extended to a two-dimensional map of uncertainty in arrival times, and will work as a 'black box' input to any type of wildfire simulator. Although only normal distribution were used within this study, the method allows any distributions with appropriate weighting functions. The scaling behaviour of the Polynomial Chaos method can also be improved at high dimensions using more advanced schemes such as Smolyak quadrature requiring far fewer evaluation points. This avenue will be investigated in future work. The rapid results from the method may make it suitable to incorporate input variation into operational wildfire predictions at low computational cost.

APPENDIX

Python example code for Polynomial Chaos implementation using the NumPy libraries.

```
from numpy.polynomial.hermite_e import hermegauss, hermeval

# Get Gauss quadrature points
chi = hermegauss(N)[0]

# Carry out simulations using mapped values of chi
# ...

# Calculate mean and standard deviation
psi = []
G = hermegauss(N)
c = numpy.identity(N)
for j in range (0, N):
    psi.append(sum(hermeval(G[0][i], c[j])*psi_s[i]*G[1][i] for i in range (0, N))/(sqrt(2*
        pi)*factorial(j)))

mean = psi[0]
stdev = sqrt(sum((psi[i]**2)*factorial(i) for i in range (1, N)))
```

REFERENCES

Sullivan, A.L. (2009). Wildland surface fire spread modelling, 1990-2007. 2: Empirical and quasi-empirical models, *International Journal of Wildland Fire*, **18**, 369-386.

Wiener, N. (1938). The homogeneous chaos, *American Journal of Mathematics*, **60**, 897-936.

Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions, *Reliability Engineering and System Safety*, **93**, 964-979.

McArthur, A.G. (1967). Fire behaviour in eucalypt forest. Commonwealth Department of National Development. Forestry Timber Bureau, Leaflet 107, Canberra, ACT.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation*, **8**, 3-30.

Hilton J.E., Miller C., Sullivan, A.L. and Rucinski C. (2015). Effects of spatial and temporal variation in environmental conditions on simulation of wildfire spread, *Environmental Modelling and Software*, **67**, 118-127.