# Testing a Quantum-inspired Evolutionary Algorithm by applying it to Non-linear Principal Component Analysis of the Implied Volatility Smile

Kai Fan[1,2], Conall O'Sullivan[2], Anthony Brabazon[1] and Michael O'Neill[1]

[1] Natural Computing Research and Applications Group,
University College Dublin, Ireland.
kai.fan@ucd.ie; anthony.brabazon@ucd.ie; m.oneill@ucd.ie

[2] School of Business, University College Dublin, Ireland.
conall.osullivan@ucd.ie

**Summary.** Principal Component Analysis (PCA) is a standard statistical technique that is frequently employed in the analysis of large correlated data sets. As it stands, PCA is limited to linear space. We examine a technique for non-linear PCA by transferring the data from the non-linear space to linear space, where the weights on the non-linear functions are optimised using a Quantum-inspired Evolutionary Algorithm. This non-linear principal component analysis is used to examine the linear and non-linear dynamics of the implied volatility smile derived from FTSE 100 stock index options over a sample period of 500 days.

**Key words:** Non-linear principal component analysis (NLPCA), quantum-inspired evolutionary algorithm (QIEA), implied volatility smile (IVS)

## 1 Introduction

This paper introduces a non-linear principal component analysis (NLPCA) methodology which uses a quantum-inspired real number encoding rather than neural network as in the traditional NLPCA approach. The NLPCA is used to determine the non-linear principal components that drive the variations in the implied volatility smile over time. The implied volatility smile (IVS) is how markets represent option prices. Option prices change from day to day to reflect changes in the asset price that the options are written on, the market conditions, such as volatility and risk aversion and general economic trends. This of use in the pricing and hedging of assets depend on the evolution of the IVS.

## 1.1 Quantum-inspired Evolutionary Algorithm

Quantum mechanics is an extension of classical mechanics which models behaviours of natural systems that are observed particularly at very short time or distance scales. An example of such a system is a sub-atomic particle, such as a free electron. A complex-valued (deterministic) function of time and space co-ordinates, called the *wave-function*, is associated with the system: it describes the *quantum state* the system is in. The standard interpretation of Quantum Mechanics is that this abstract wave-function allows us to calculate probabilities of outcomes of concrete experiments. The squared modulus of the wave-function is a probability density function (PDF): it describes the probability that an observation of, for example, a particle will find the particle at a given time in a given region of space. The wave-function satisfies the *Schrödinger equation.* This equation can be thought of as describing the time evolution of the wave-function — and so the PDF — at each point in space: as time goes on, the PDF becomes more "spread out" over space, and our knowledge of the position of the particle becomes less precise, until an observation is carried out; then, according to the usual interpretation, the wave-function "collapses" to a particular classical state (or *eigenstate*), in this case a particular position, and the spreading out of the PDF starts all over again.

Before the observation we may regard the system as being in a linear combination of all possible classical states (this is called *superposition of states*); then the act of observation causes one such classical state to be chosen, with probability given by the PDF. Note that the wave function may interfere with itself (for example, if a barrier with slits is placed in the "path" of a particle) and this interference may be constructive or destructive, that is, the probability of detecting a particle in a given position may go up or go down.

More generally, we may seek to observe properties of quantum systems other than position, e.g., energy, momentum, or the quantum *spin* of an electron, photon or other particle. [Spin is incorporated by necessity in Dirac's relativistic extension of the wave equation; in fact spin is one of the arguments of the wave-function.] Such properties are called *observables.* Observables may be either continuous (e.g., position of a particle) or discrete (e.g., the energy of an electron in a bound state in an atom). Some observables may only take finitely many values, e.g., there are only two possible values for a given particle's spin: "up" or "down". This last is an example of a *two-state system*: in such a system the quantum state $\psi$ is a linear superposition of just two eigenstates, say $|0\rangle$ and $|1\rangle$ in the standard Dirac bra-ket notation, that is,

$$\psi = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha$ and $\beta$ are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. Here $|0\rangle$ and $|1\rangle$ are basis vectors for a 2-dimensional complex Hilbert space. A two-state system where the states are normalised and orthogonal, as here, may be regarded as a *quantum bit* or *qubit*.[3] It is thought of as being in eigenstates $|0\rangle$ and $|1\rangle$ simultaneously, until an observation is made and the quantum state collapses to $|0\rangle$ (with probability $|\alpha|^2$) or $|1\rangle$ (with probability $|\beta|^2$). The relation $|\alpha|^2 + |\beta|^2 = 1$ captures the fact that precisely one of $|0\rangle$, $|1\rangle$ must be observed, so their probabilities of observation must sum to 1.

A *quantum computer* is one which works with qubits instead of the (classical) bits used by usual computers. Benioff [1] first considered a Turing machine which used a tape containing what we would call qubits. Feynman [6] developed examples of physical computing systems not equivalent to the standard model of deterministic computation, the Turing machine.

In recent years there has been a substantial interest in the theory and design of quantum computers, and the design of programs which could run on such computers, stimulated by Shor's discovery of a quantum factoring algorithm which would run faster than possible clasically. One interesting strand of research has been the use of natural computing (for example Genetic Programming (GP)) to generate quantum circuits or programs (algorithms) for quantum computers [18]. (Genetic programming is an evolutionary algorithm based methodology inspired by biological evolution to find computer programs that perform a user-defined task. Therefore it is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.) There has also been associated work in a reverse direction which draws inspiration from concepts in quantum mechanics in order to design novel natural computing algorithms. This is currently an area of active research interest. For example, quantum-inspired concepts have been applied to the domains of evolutionary algorithms [15, 8, 9, 20, 21], social computing [22], neuro-computing [13, 7, 19], and immuno-computing [14, 11]. A claimed benefit of these algorithms is that because they use a quantum representation, they can maintain a good balance between exploration and exploitation. It is also suggested that they offer computational efficiencies as use of a quantum representation can allow the use of smaller population sizes than typical evolutionary algorithms.

Quantum-inspired evolutionary algorithms (QIEA) offer interesting potential. As yet, due to their novelty, only a small number of recent papers have implemented a QIEA, typically reporting good results [20, 21]. Consequently, we have a limited understanding of the performance of these algorithms and further testing is required in order to determine both their effectiveness and their efficiency. It is also noted that although a wide-variety of biologically-inspired algorithms have been applied for financial modelling [2], the QIEA methodology has not yet been applied to the finance domain. This study addresses both of these research gaps.

[3]Geometrically, a qubit is a compact 2-dimensional complex manifold, called the Bloch sphere.

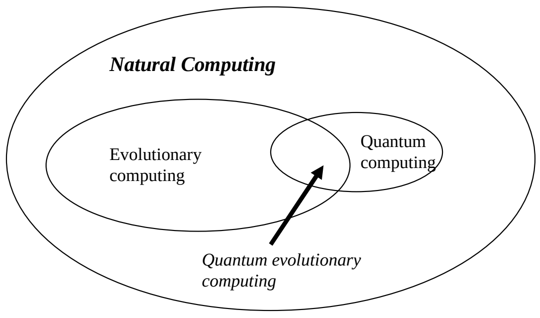


**Fig. 1.** Quantum-inspired evolutionary computing

### 1.2 Structure of Chapter

The rest of this chapter is organised as follows. The next section provides a concise overview of QIEA, concentrating on the quantum-inspired genetic algorithm (QIGA), and introduces NLPCA based on QIGA. We then outline the experimental methodology adopted. The remaining sections provide the results of these experiments followed by a number of conclusions.

## 2 The Quantum-inspired Genetic Algorithm

The best-known application of quantum-inspired concepts in evolutionary computing is the quantum-inspired genetic algorithm (QIGA) [15, 8, 9, 20, 21]. The QIGA is based on the concepts of a qubit and the superposition of states. In essence, in QIGAs the traditional representations used in evolutionary algorithms (binary, numeric and symbolic) are extended to include a quantum representation.

A crucial difference between a qubit and a (classical) bit is that multiple qubits can exhibit quantum entanglement. Entanglement is when the wave function of a system composed of many particles cannot be separated into independent wave functions, one for each particle. A measurement made on one particle can produce, through the collapse of the total wavefunction, an instantaneous effect on other particles with which it is entangled, even if they are far apart. Entanglement is a nonlocal property that allows a set of qubits to be highly correlated. Entanglement also allows many states to be acted on simultaneously, unlike bits that can only have one value at a time. The use of entanglement in quantum computers is sometimes called *quantum parallelism*, and gives a possible explanation for the power of quantum computing: because the state of the quantum computer (i.e., the state of the system considered as a whole) can be in a quantum superposition of many different classical computational states, these classical computations can all be carried out at the same time.

The quantum equivalent of a classical operator on bits is an *evolution* (not to be confused with the evolution of EAs). It transforms an input to an output, e.g., by rotation or Hadamard gate, and operates without measuring the value of the qubit(s). Thus it effectively does a parallel computation on all the qubits at once and gives rise to a new superposition.

In the language of evolutionary computation a system of $m$ qubits may be referred to as a *quantum chromosome* and can be written as a matrix with two rows:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}. \tag{1}$$

A key point when considering quantum systems is that they can compactly convey information on a large number of possible system states. In classical bit strings, a string of length $m$ can represent $2^m$ possible states. However, a quantum space of $m$ qubits has $2^m$ *dimensions* (as a complex manifold).[4] Thus, a single qubit register of length $m$ can simultaneously represent *all* possible bit strings of length $2^m$, e.g., an 8 qubit system can simultaneously encode 256 distinct strings. This implies that it is possible to modify standard evolutionary algorithms to work with very few, or even a single quantum individual, rather than having to use a large population of solution encodings. The qubit representation can also help to maintain diversity during the search process of an evolutionary algorithm, due to its capability to represent multiple system states simultaneously.

## 2.1 Representing a Quantum System

There are many ways that a quantum system could be defined in order to encode a set of binary (solution) strings. For example, in the following 3 qubit quantum system, the quantum chromosome is defined using the three pairs of amplitudes below

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \tag{2}$$

These numbers are the probabilities that a qubit (unit of information) will be observed in a particular eigenstate rather than another. Taking the first qubit, the occurrence of either state 0 or 1 is equally likely as both $\alpha_1$ and $\beta_1$ have the same amplitude. Following on from the definition of the 3 qubit system, the (quantum) state of the system is given by

$$\tfrac{\sqrt{3}}{4\sqrt{2}}|000\rangle+\tfrac{3}{4\sqrt{2}}|001\rangle+\tfrac{1}{4\sqrt{2}}|010\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|011\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|100\rangle+\tfrac{3}{4\sqrt{2}}|101\rangle+\tfrac{1}{4\sqrt{2}}|110\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|111\rangle \tag{3}$$

To provide intuition on this point, consider the system state $|000\rangle$. The associated probability amplitude for this state is $\frac{\sqrt{3}}{4\sqrt{2}}$ and this is derived

[4]It can be shown that, because of entanglement, an $m$-qubit physical system has $2^{m+1} - 2$ degrees of freedom, much larger than the $2m$ degrees a classical version would have.

from the probability amplitudes of the 0 state for each of the three individual qubits ($\frac{1}{\sqrt{2}} * \frac{\sqrt{3}}{2} * \frac{1}{2} = 0.25$). The associated probabilities of each of the individual states ($|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$) are $\frac{3}{32}, \frac{9}{32}, \frac{1}{32}, \frac{3}{32}, \frac{3}{32}, \frac{9}{32}, \frac{1}{32}, \frac{3}{32}$ respectively. Taking the first of these states as an example, $(\frac{\sqrt{3}}{4\sqrt{2}})^2 = \frac{3}{32}$.

### 2.2 Real-Valued Quantum-inspired Evolutionary Algorithms

In the initial literature which introduced the QIGA, a binary representation was adopted, wherein each quantum chromosome was restricted to consist of a series of 0s and 1s. The methodology was modified to include real-valued vectors by da Cruz et al., [4]. As with binary-representation QIGA, real-valued QIGA maintains a distinction between a quantum population and an observed population of, in this case, real-valued solution vectors. However the quantum individuals have a different form to those in binary-representation QIGA. The quantum population $Q(t)$ is comprised of $N$ quantum individuals ($q_i : i = 1, 2, 3, \ldots, N$), where each individual $i$ is comprised of $G$ genes ($g_{ij} : j = 1, 2, 3, \ldots, G$). Each of these genes consist of a pair of values $q_{ij} = (p_{ij}, \sigma_{ij})$ where $p_{ij}, \sigma_{ij} \in \Re$ represent the mean and the width of a square pulse. Representing a gene in this manner has a parallel with the quantum concept of superposition of states as a gene is specified by a range of possible values, rather than by a single unique value.

The original QIGA algorithms, e.g., [8, 9] are based very closely on physical qubits, but the "quantum-inspired" algorithm of da Cruz et al. [4] used in this chapter draws less inspiration from quantum mechanics since it:

- does not use the idea of a quantum system (in particular, no qubits);
- only allows for constructive (not destructive) interference, and that interference is among "wave-functions" of *different* individuals;
- uses real numbers as weights, rather than the complex numbers which arise in superposition of states in physical systems;
- the PDFs used (uniform distributions) are not those arising in physical systems.

However, the da Cruz et al algorithm does periodically sample from a distribution to get a "classical" population, which can be regarded as a wave-function (quantum state) collapsing to a classical state upon observation.

#### Algorithm

The real-valued QIGA algorithm is as follows

```
Set t=0

Initialise Q(t) of N individuals with G genes

While (t < max t)
```

```
        Create the PDFs (and corresponding CDFs, which describe the probability
            distributions of real-valued random variables, see equation(6)) for
            each gene locus using the quantum individuals
        Create a temporary population, denoted E(T), of K real-valued solution
            vectors by observing Q(t) (via the CDFs)

        If (t=0) Then C(t)=E(t)
        (Note: the population C(T) is maintained between iterations of the algorithm)
        Else    E(t)=Outcome of crossover between E(t) and C(t)
                Evaluate E(t)
                C(t)= K best individuals from E(t) U C(t)
        End if

        With the N best individuals from C(t)
        Q(t+1)=Output of translate operation on Q(t)
        Q(t+1)=Output of resize operation on Q(t+1)
        t=t+1
    Endwhile
```

### Initialising the Quantum Population

A *quantum chromosome*, which is observed to give a specific solution vector of real-numbers, is made up of several quantum genes. The number of genes is determined by the required dimensionality of the solution vector. At the start of the algorithm, each quantum gene is initialised by randomly selecting a value from within the range of allowable values for that dimension. A gene's width value is set to the range of allowable values for the dimension. For example, if the known allowable values for dimension $j$ are $[-75, 75]$ then $q_{ij}$ (dimension $j$ in quantum chromosome $i$) is initially determined by randomly selecting a value from this range (say) -50. The corresponding width value will be 150. Hence, $q_{ij} = (-50, 150)$. The square pulse need not be entirely within the allowable range for a dimension when it is initially created as the algorithm will automatically adjust for this as it executes. The height of the pulse arising from a gene $j$ in chromosome $i$ is calculated using

$$h_{ij} = \frac{1/\sigma_{ij}}{N} \tag{4}$$

where $N$ is the number of individuals in the quantum population. This equation ensures that the probability density functions (PDFs) (see next subsection) used to generate the observed individual solution vectors will have a total area equal to one. Fig. 2 provides an illustration of a quantum gene where N=4.

### Observing the Quantum Chromosomes

In order to generate a population of real-valued solution vectors, a series of observations must be undertaken using the population of quantum chromosomes (individuals). A pseudo-interference process between the quantum individuals is simulated by summing up the square pulses for each individual gene across all members of the quantum population. This generates a separate PDF (just

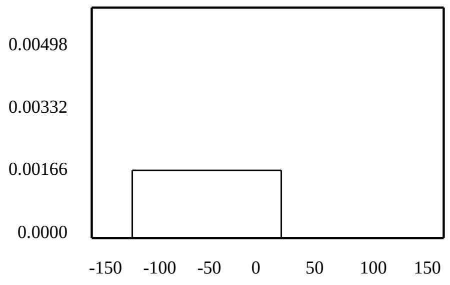


**Fig. 2.** A square pulse, representing a quantum gene, with a width of 150, centred on -50. The height of the gate is 0.005

the sum of the square pulses) for each gene and eq. 4 ensures that the area under this PDF is one. Hence, the PDF for gene $j$ on iteration $t$ is

$$PDF_j(t) = \sum_{i}^{j} g_{ij} \tag{5}$$

where $g_{ij}$ is the square pulse of the $j^{th}$ gene of the $i^{th}$ quantum individual (of $N$). To use this information to obtain an observation, the PDF is first converted into its corresponding Cumulative Distribution Function (CDF)

$$CDF_j(x) = \int_{L_j}^{U_j} PDF_j(x)dx \tag{6}$$

where $U_j$ and $L_j$ are the upper and lower limits of the probability distribution. By generating a random number $r$ from (0,1), the CDF can be used to obtain an observation of a real number $x$, where $x = CDF^{-1}(r)$. If the generated value $x$ is outside the allowable real valued range for that dimension, the generated value is limited to its allowable boundary value. A separate PDF and CDF is calculated for each of the $G$ gene positions. Once these have been calculated, the observation process is iterated to create a temporary population with $K$ members, denoted E(t).

## Crossover Mechanism

The crossover operation takes place between C(t) and the temporary population E(t). This step could be operationalised in a variety of ways with [4] choosing to adopt a variant of uniform crossover, without an explicit selection operator. After the $K$ crossover operations have been performed, with the resulting children being copied into E(t), the best $K$ individuals $\in C(t) \cup E(t)$ are copied into C(t).

### Updating the Quantum Chromosomes

The $N$ quantum chromosomes are updated using the $N$ best individuals from C(t) after performing the crossover step. Each quantum gene's mean value is altered using

$$p_{ij} = c_{ij} \tag{7}$$

so that the mean value of the $j^{th}$ gene of the $i^{th}$ quantum chromosome is given by the corresponding $j^{th}$ value of the $i^{th}$ ranked individual in $C(t)$.

The next step is to update the corresponding width value of the $j^{th}$ gene. The objective of this process is to vary the exploration / exploitation characteristics of the search algorithm, depending on the feedback from previous iterations. If the search process is continuing to uncover many new better solutions, then the exploration phase should be continued by keeping the widths relatively broad. However, if the search process is not uncovering many new better solutions, the widths are reduced in order to encourage finer-grained search around already discovered good regions of the solution space. There are multiple ways this general approach could be operationalised. For example, [4] suggests use of the 1/5th mutation rule from Evolutionary Strategies [16] whereby

$$if\ \phi < 1/5 \text{ then } \sigma_{ij} = \sigma_{ij} g$$

$$if\ \phi > 1/5 \text{ then } \sigma_{ij} = \sigma_{ij}/g$$

$$if\ \phi = 1/5 \text{ then } \sigma_{ij} = \sigma_{ij}$$

where $\sigma_{ij}$ is the width of the $i^{th}$ quantum chromosome's $j^{th}$ gene, $g$ is a constant in the range $[0,1]$ and $\phi$ is the proportion of individuals in the new population that have improved their fitness.

In this study we update the width of the $i^{th}$ quantum chromosome's $j^{th}$ gene by comparing each successive generation's best fitness function. If the best fitness function has improved (disimproved) we shrink (enlarge) the width in order to improve the local (global) search.

### QIGA vs Canonical Genetic Algorithm

A number of distinctions between the QIGA above and the canonical GA (CGA) can be noted. In the CGA, the population of solutions persists from generation to generation, albeit in a changing form. In contrast, in QIGA, the population of solutions in $P(t)$ are discarded at the end of each loop. The described QIGA, unlike CGA, does not have explicit concepts of crossover or mutation. However, the adaptation of the quantum chromosomes in each iteration does embed implicit selection as the best solution is selected and is used to adapt the quantum chromosome(s). The crossover and mutation steps are also implicitly present, as the adaptation of the quantum chromosome in effect creates diversity, as it makes different states of the system more or less likely over time. Another distinction between the QIGA and the CGA is that the CGA operates directly on representations of the solution (the members of the current population of solutions), whereas in QIGA the update step is performed on the probability amplitudes of the ground states for each qubit making up the quantum chromosome(s).

## Observing the Quantum Chromosomes

In order to generate a population of real-valued solution vectors, a series of observations must be undertaken using the population of quantum chromosomes (individuals). A pseudo-interference process between the quantum individuals is simulated by summing up the square pulses for each individual gene across all members of the quantum population. This generates a separate PDF (just the sum of the square pulses) for each gene and eq. 4 ensures that the area under this PDF is one. Hence, the PDF for gene $j$ on iteration $t$ is

$$PDF_j(t) = \sum_{i}^{j} g_{ij} \tag{8}$$

where $g_{ij}$ is the square pulse of the $j^{th}$ gene of the $i^{th}$ quantum individual (of $N$). To use this information to obtain an observation, the PDF is first converted into its corresponding Cumulative Distribution Function (CDF)

$$CDF_j(x) = \int_{L_j}^{U_j} PDF_j(x)dx \tag{9}$$

where $U_j$ and $L_j$ are the upper and lower limits of the probability distribution. By generating a random number $r$ from (0,1), the CDF can be used to obtain an observation of a real number $x$, where $x = CDF^{-1}(r)$. If the generated value $x$ is outside the allowable real valued range for that dimension, the generated value is limited to its allowable boundary value. A separate PDF and CDF is calculated for each of the $G$ gene positions. Once these have been calculated, the observation process is iterated to create a temporary population with $K$ members, denoted E(t).

In the next section we use non-linear principal component analysis (NLPCA) to decompose the variation of the implied volatility smile into a smaller number of non-linear principal components. To run NLPCA a set of weights on a number of non-linear mapping functions must be determined by optimising the proportion of variation explained by the principal components. Given the non-linearities inherent in options prices and in the NLPCA method QIEA is used to determine these weights in case the optimisation problem is not convex.

## 3 Non-Linear Principal Component Analysis

Suppose $X \in M^{m,n}$ is a panel data set that contains correlated data points along the columns, evaluated at different points in time along the rows. Given that $X$ consists of correlated data points, the variation in $X$ can be decomposed into a small number $r$ of orthogonal principal components with $r < n$, resulting in a reduction of the dimension of the problem with only a small loss in information. The principal components from standard PCA are linear combinations (along the rows) of the original data set. If it is suspected that the data set contains non-linearities, a common procedure is to "linearise" the data set using suitable transformations prior to analysis. This approach has the advantage that it retains the simplicity of the underlying principal component analysis (PCA) whilst gaining the ability to cope with non-linear data. To do this we construct a modified data set $X_{NL}$ from the original data set $X$:

$$X_{NL} = G(X), \tag{10}$$

where $G$ is a function consisting of $n$ individual mapping functions from linear to non-linear space:

$$G = w_1 g_1(X) + w_2 g_2(X) + \cdots + w_n g_n(X), \tag{11}$$

and where $g_i(X)$ is an individual non-linear mapping function of $X$ and $w_i$ is the weight on the function $g_i$. There are an infinite number of mapping functions $g_i(X)$ to choose from and in this paper we consider a small number of mapping functions we think are important given the domain knowledge of the problem under consideration (see next section). There are a total of four functions chosen in this study and they are given as follows:

- Logistic mapping:
$$g_1(X) = 4X \circ (1 - X), \tag{12}$$
where $\circ$ denotes element by element matrix multiplication.
- Exponential mapping:
$$g_2\left(X\right) = \exp\left(X\right), \tag{13}$$
where the exponential function is applied on an element by element basis.
- Hénon mapping:
$$g_3\left(X\left(t\right)\right) = 1 - 1.4\left(X(t)\right)^2 + 0.3X(t-1), \tag{14}$$
where $X(t)$ is a single row of the data set $X$.
- Auto regressive process:
$$g_4\left(X\left(t\right)\right) = 0.25X(t-1) + \varepsilon(t), \tag{15}$$
where $X(t)$ is as above and $\varepsilon(t)$ is a standard normal random variable.

The objective of this data mapping is to compensate for any non-linearities within $X$. That is to linearise the data before implementing PCA. Provided this is performed as an integral part of the analysis, a non-linear variant of PCA will result. The method is described as follows: we perform standard PCA on the non-linear transfromation of the original data set and optimise the weights on the different mapping functions with the objective of maximising the proportion of variation explained by the first principal component from standard PCA. A quantum-inspired evolutionary algorithm is used to find the weights on the non-linear mapping functions $g_i \in G$ given the potential for local minima. The next step in this research is to increase the number of functions considered and optimise not only the weights but the various parameters of the functions $g_i \in G$. This is work that is currently under investigation.

## 4 Implied Volatility Smiles (IVS)

In this section we explain the meaning of the implied volatility smile. A European call (put) option on an asset $S_t$ with maturity date $T$ and strike price $K$ is defined as a contingent claim with payoff at time $T$ given by $\max\left[S_T - K, 0\right]$ ($\max\left[K - S_T, 0\right]$). The well known Black- Scholes (BS) formula for the price of a call option on an underlying asset $S_t$ is given by

$$C_{BS}\left(S_t, K, r, q, \tau; \sigma\right) = S_t e^{-q\tau} N\left(d_1\right) - K e^{-r\tau} N\left(d_1\right)$$

$$d_1 = \frac{-\ln m + \left(r - q + \frac{1}{2}\sigma^2\right)\tau}{\sigma\sqrt{\tau}} \qquad d_2 = d_1 - \sigma\sqrt{\tau}$$

where $\tau = T - t$ is the time-to-maturity, $t$ is the current time, $m = K/S$ is the moneyness of the option, $r$ and $q$ are the continuously compounded risk-free rate and dividend yield and $N(\cdot)$ is the cumulative normal distribution function. Suppose a market option price, denoted by $C_M\left(S_t, K\right)$, is observed. The Black-Scholes implied volatility for this option price is that value of volatility which equates the BS model price to the market option price as follows

$$\begin{aligned} \sigma_{BS}\left(S_t, K\right) &> 0 \\ C_{BS}\left(S_t, K, r, q, \tau; \sigma_{BS}\left(S_t, K\right)\right) &= C_M\left(S_t, K\right) \end{aligned}$$

If the assumptions underlying the BS option pricing model were correct, the BS implied volatilities for options on the same underlying asset would be constant for different strike prices and maturities. However in reality the BS implied volatilities are varying over strike price and maturity. The variation of implied volatilities over strike price for a fixed maturity is known as the implied volatility smile. Given that the options are written on a single underlying asset this result seems at first paradoxical, i.e. we have a number of different implied volatilities for a single asset which should only have one measure for its volatility. The assumptions in the BS model can be relaxed, such as allowing the underlying asset price to follow a more complex data generating process than the log normal stochastic process (as assumed by BS), or allowing the underlying asset price to experience sudden discontinuous jumps etc. When the resulting complications of these more general assumptions are taken into account, the implied volatility smile begins to make sense and is simply highlighting the erroneous assumptions that underpin the BS model.

Implied volatilities are frequently used in the financial markets to quote the prices of options. The participants in the options markets do not believe that the BS model is correct, but use the model as a convenient way of quoting option prices. The reason is that implied volatilities usually have to be updated less frequently than option prices themselves and implied volatilities vary less dramatically than option prices with strike price and maturity. Option traders and brokers monitor movements in volatility smiles closely. As option prices change over time the implied volatility smile (for various maturities) also changes.

If we stack the implied volatility smile (for one particular maturity) according to the time the IVS data was recorded, that results is a time series of panel data with highly correlated entries. Implied volatilities at different strikes are highly correlated because as the volatility of the asset rises all implied volatilities rise yet some may rise more than others. However the economic forces of no-arbitrage (no free-lunches) ensures that the implied volatilities cannot get too detached from one another because if they did this represents a riskless trading opportunity for savvy investors, who sell the more expensive option (with the higher implied volatility) and hedge it with cheaper options (with lower implied volatilities).

PCA is an ideal tool to reduce the complexity of such a data set by explaining the variation of the IVS over time in terms of a small number of orthogonal principal factors. The approach has been applied and the dynamical properties of the implied volatility smile has been studied in recent years using increasingly advanced principal component approaches. See Heynen, Kemma and Vorst [10] and Fengler, Härdle and Schmidt [5] for PCA applied to the term structure of at-the-money implied volatilities (implied volatilities for different maturities and a fixed strike price) and see Skiadopoulos, Hodges and Clewlow [17] for PCA applied to implied volatility smiles. The evidence suggests that changes in the implied volatility smile are driven predominantly by three factors. The first factor is a level factor which controls the overall height of the implied volatility smile. The second and third factors are slope and curvature factors across the strike price axis. However options and the implied volatilities associated with options are multi-dimensional non-linear instruments and standard PCA may neglect some of non-linear subleties inherent in option implied volatilities. This is the reason NLPCA is applied to the IVS in this paper.

## 5 Results

### 5.1 Data

The data used in this study are option implied volatilities across 11 different strikes and a number of different maturities on the FTSE 100 index. The data consists of end-of-day settlement option implied volatilities from the 26th of March 2004 till the 17th of March 2006 consisting of 500 trading days. FTSE 100 index options are European style options and the underlying asset is the FTSE 100 performance index. To price options on this index one must adjust the index by extracting the present value of future cash dividend payments before each options expiration date. The annualised dividend yield of the FTSE 100 index was downloaded from Datastream. The one-month LIBOR rate was used as the risk-free rate where the LIBOR rate was converted into a continuously compounded rate. The forward price used in the option calculations is then simply $F_t = S_0 e^{(r-q)t}$ where $S_0$ is the current index price level, $F_t$ is the price for the forward contract maturing at time $t$, $r$ is the continuously compounded risk-free rate and $q$ is the continuously compounded dividend yield. Settlement prices of call and put option are calculated from the implied volatilities using the Black-Scholes formula.

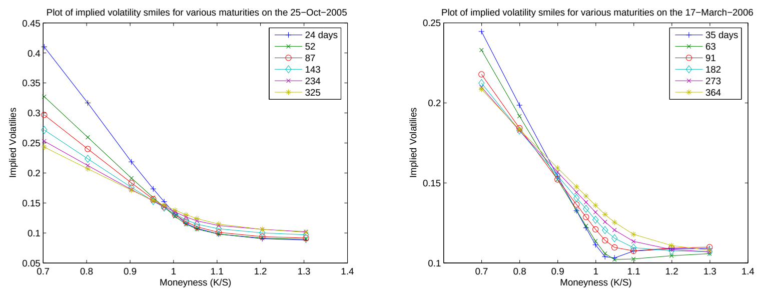
As calendar time passes the option contracts wind down towards maturity and the observed implied volatility surface (implied volatilities plotted across strike price and maturity) is constantly changing in terms of its moneyness and maturity values, see figure 3. To obtain implied volatilities on a fixed grid of moneyness and maturity the market implied volatilities were interpolated using a non-parametric Nadaraya-Watson estimator, see Cont and da Fonseca [3], so that we have interpolated implied volatilities on a fixed grid of moneyness and maturity for all the days in the data sample. On day $t$ an interpolated estimate for implied volatility at a moneyness $m$ and a time to maturity $\tau$ is given by

$$I_t(m,\tau) = \frac{\sum_i^{n_m}\sum_j^{n_\tau} I_t(m_i,\tau_j) f(m-m_i,\tau-\tau_j)}{\sum_i^{n_m}\sum_j^{n_\tau} f(m-m_i,\tau-\tau_j)},$$
$$f(x,y) = (2\pi)^{-1}\exp\left(-x^2/h_1\right)\exp\left(-y^2/h_2\right),$$

where $n_m$ and $n_\tau$ are the number of different option moneyness levels and maturities available on a particular day in the sample, $m_i$ is the moneyness and $\tau_j$ is the maturity of the $(i,j)^{th}$ observed option and $h_1$ and $h_2$ are the bandwidths of the estimator across moneyness and maturity. The bandwidths for the estimator were chosen using cross validation, where one implied volatility is removed and is then interpolated from all the other available implied volatilities on that date. The difference between the interpolated and the observed implied volatility is the cross validation error. This is calculated for all implied volatilities available on a particular day and this error is miminised by optimising over $h_1$ and $h_2$. For each day $t$ in the sample we define the implied volatility smile at a fixed maturity $\tau_j$ by

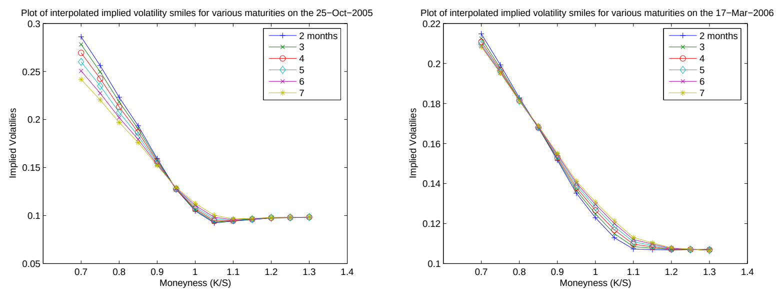
$$IVS(t) = \{I_t(1,\tau_j),\ldots,I_t(n_m,\tau_j)\}\,.$$

We then stack these implied volatility smiles over time to form the data matrix $X = \{IVS(1),\ldots,IVS(500)\}'$. Non-linear principal component analysis (NLPCA) is conducted on the implied volatility smile and logarithm of the implied volatility smile for maturities ranging from 2 to 6 months.



(a) 05-Oct-2005 (b) 17-Mar-2006

**Fig. 3.** Implied Volatility Smiles on two different dates.

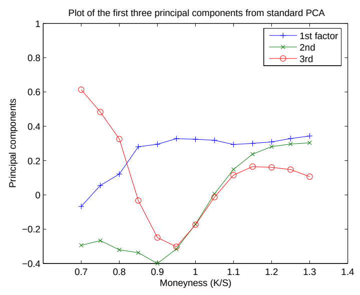


(a) 05-Oct-2005 (b) 17-Mar-2006

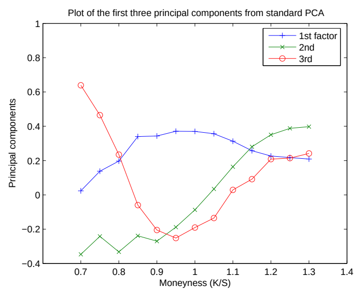
**Fig. 4.** Interpolated Implied Volatility Smiles on the same dates as above.

### 5.2 Result Analysis

The first three principal components from linear PCA explain up to approximately 96% of the variation in the level of the implied volatility smile, depending on the maturity of the IVS considered. As 96% in PCA analysis may be overfitting, we would rather target the first principal component than the



(a) 2 month IVS



(b) 5 month IVS

**Fig. 5.** Three linear principal components for an IVS with a maturity of 2 and 5 months.

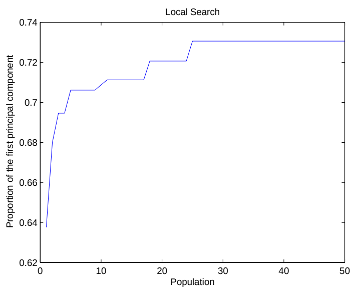
first three components and this why the objective function in the NLPCA was chosen to be proportion of variation explained by the first principal component.

The analysis of the eigenfactors from standard PCA for the implied volatility smiles of each maturity shows that the first factor has a positive effect on all implied volatilities. This eigenfactor can be interpreted as a level or a volatility factor. An increase in this factor causes the whole IVS to increase and causes all options to become more expensive since options are increasing functions of volatility. The second factor has a negative effect for implied volatilities with $K < S$, e.g. out-of-the-money puts, and a positive effect for implied volatilities with $K > S$, e.g. out-of-the-money calls. This factor can be interpreted as a skew factor and increase in this factor causes out-of-the money calls to become more expensive relative to out-of-the-money puts. The third factor has a positive effect for implied volatilities with $K < S$ and $K > S$ e.g. out-of-the-money calls and puts, and a negative effect for implied volatilities that are close to the money with $K \approx S$. This factor can be interpreted as a curvature factor an an increase in this factor causes out-of-the money calls and puts to become more expensive relative to near-the-money calls and puts.

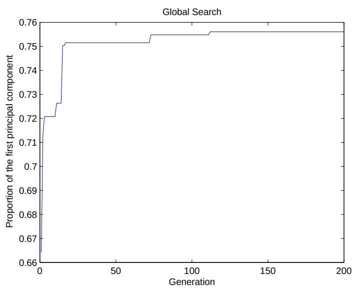
**Table 1.** Parameters setting in Quantum-inspired Genetic Algorithm

| Population size | Generation Number | Crossover rate | Shrinkage | Enlargement |
|---|---|---|---|---|
| 200 | 50 | 0.7 | 0.8 | 1.2 |

In our NLPCA-QIEA analysis, the weights on the mapping functions are optimised by using a quantum-inspired genetic algorithm to maximise the objective function which is the proportion of variation in the data explained



(a) Local Search



(b) Global Search

**Fig. 6.** Local and Global search

**Table 2.** Results of QIGA. The proportion explained by the first principal component (PC) from the last generation are averaged over 30 runs and compared with the parameter values from 30 runs of a Matlab optimiser.

| Maturity | Linear PCA (%) | Non-linear PCA(%) | Standard deviation(%) | Matlab optimiser(%) |
|---|---|---|---|---|
| 2 months | 64.15 | 80.47 | 0.04 | 80.97 |
| 3 months | 69.57 | 80.27 | 0.04 | 81.17 |
| 4 months | 72.90 | 80.87 | 0.03 | 81.30 |
| 5 months | 77.01 | 81.67 | 0.01 | 82.04 |
| 6 months | 80.27 | 84.35 | 0.01 | 84.38 |

**Table 3.** Results of QIGA. The optimal and average weights on mapping functions from the last generation are averaged over 30 runs.

| Maturity(months) | Logistic | Exp | Hénon | AR |
|---|---|---|---|---|
| 2 | 0.000 | 0.3652 | 0.9814 | 0.3669 |
| 3 | 0.000 | 0.3634 | 0.9791 | 0.3769 |
| 4 | 0.000 | 0.3614 | 0.9800 | 0.3885 |
| 5 | 0.000 | 0.3704 | 0.9800 | 0.3542 |
| 6 | 0.000 | 0.3653 | 0.9815 | 0.3798 |

by the first principal component. The weights are also optimised using the Matlab function *fminsearch*. *Fminsearch* uses the simplex search method of Lagarias et al [12]. This is a direct search method that does not use numerical or analytic gradients. Figure 5 depicts the evolution of the objective function versus the generation number. The parameter settings in the QIEA are given in Table 1. NLPCA is more efficient than linear PCA especially for the options with shorter times-to-maturity. For example, for the 2 month IVS the 1st principal component from NLPCA explains approximately 80% of the variation of the data versus only 64% for standard PCA. However the outperformance of NLPCA is to be expected given the extra degrees of freedom involved since it uses four non-linear functions that first operate on the data before PCA is applied. It is interesting to note that for the two month IVS the first component from NLPCA with four non-linear functions explains 80% of the variation whilst the first *three* components from linear PCA explain up to 96% of the variation in the data. Although 96% is a higher level of explanatory power this is more than likely overfitting historical data at the expense of poor out-of-sample performance. If we forecast the evolution of the IVS out-of-sample using the techniques in this paper, a parsimonious procedure would be to include a more general set of time series models in the set of non-linear functions and use these to forecast the first factor from NLPCA and reconstruct future IVS's from the weights derived from historical analysis. This would be more parsimonious than fitting a separate time series model to three linear principal components and then reconstructing the future IVS as would have to be done in linear PCA. Thus, at least for shorter term options, the NLPCA method can explain 80% of the variation in the data with one linear combination of non-linear functions of the data versus approximately 64% for linear PCA. Thus rather than increasing the number of principal components in the analysis we have shown that another route is to use non-linear principal components to achieve a statistical significant increase in explanatory power.

It is interesting to note the weights on the various functions that were derived in the NLPCA. The weight on the logistic function are always very close to zero thus this mapping had nothing to contribute to the NLPCA. The weights on the exponential function and the autogressive function were approximately the same at a little over $\frac{1}{3}$ and the weights on the Hénon function were close to one. The exponential function is capturing the skew effect mentioned earlier. The auto regressive function is capturing serial correlation in the daily movements of the IVS (something that cannot be done under linear PCA). The fixed value on the AR function was taken to be 0.25 thus when we multiply this by the weight of appoximately 0.35 this results in serial correlation coefficient of approximately 0.088. Thus there is positive serial correlation in the data and this represents a possible trading strategy. The Hénon function is capturing a combination of a curvature effect (mentioned earlier), due to the squaring of the data, and a time series effect due to the dependence on past values. The weight on this function is close to one meaning this function is relatively important in the analysis because the average values of the entries in the vector $1 - 1.4X(t)^2 + 0.3X(t-1)$ are close to one and the weigth multiplied by this average value are relatively large compared to the values from the other functions. This implies that the function depending on the curvature of the current IVS and the past level of the IVS is very important for explaining the variation in the IVS over time.

## 6 Conclusions

A non-linear principal component analysis was conducted on the implied volatility smile derived from FTSE 100 stock index options. It was shown, at least for shorter term options, that the NLPCA method can explain 80% of the variation in the data with one non-linear principal component versus approximately 64% for one linear principal component in linear PCA. Thus the non-linear functions used in the NLPCA captured some of the higher order non-linear factors that affect the data and effectly increased the explanatory power of the method.

The weights on these non-linear functions were optimised using a quantum-inspired evolutionary algorithm (QIEA). Although the problem considered in this paper was not high-dimensional, it has potential to be a highly non-linear non-convex optimisation problem due to the fact that the options data analysed are highly non-linear and method used to describe the variation in the options data is a non-linear method. Thus it was thought that this was a reasonable problem to test out the QIEA with a view to using it for more extensive analysis in future work. This future work consists of expanding the number of non-linear functions being considered with a focus on including a larger number of time series models. This would be very useful in predicting the IVS out-of-sample and in constructing options trading strategies. Future work could also look at mutli-objective NLPCA where the proportion of variation explained by the first factor is maximised followed by the proportion of variation explained by the second factor, etc. Also it would be useful to relax the restriction on the parameters of the non-linear functions used in NLPCA and allow the QIEA to find optimal values for these parameters. All of these extensions will result in very high-dimensional optimisation problems where the use of evolutionary algorithms such as the QIEA may be essential.

## References

1. Benioff P (1980). The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines. Journal of Statistical Physics 22: 563–591.
2. Brabazon A and O'Neill M (2006). Biologically-inspired Algorithms for Financial Modelling. Berlin Springer.
3. Cont R and da Fonseca J (2002). Dynamics of implied volatility surfaces. Journal of Quantitative finance 2: 45-60.
4. da Cruz A, Vellasco M and Pacheco M (2006). Quantum-inspired evolutionary algorithm for numerical optimization. Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006), 16-21 July, Vancouver, 9180–9187, IEEE Press.
5. Fengler M, Härdle W and Schmidt P (2002). Common factors governing VDAX movements and the maximum loss. Jounal of Financial Markets and Portfolio Management 1: 16-19.

6. Feynman R (1982). Simulating Physics with Computers. International Journal of Theoretical Physics 21 (6&7): 467–488.
7. Garavaglia S (2002). A quantum-inspired self-organizing map (QISOM). Proceedings of 2002 International Joint Conference on Neural Networks (IJCNN 2002), 12-17 May 2002, 1779–1784, IEEE Press.
8. Han K-H and Kim J-H (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Transactions on Evolutionary Computation 6(6): 580–593.
9. Han K-H and Kim J-H (2002). Quantum-inspired evolutionary algorithms with a new termination criterion, $H_\varepsilon$ gate and two-phase scheme. IEEE Transactions on Evolutionary Computation 8(3): 156–169.
10. Heynen R and Kemma K and Vorst T. (1994). Analysis of the term structure of implied volatilities. Journal of Financial and Quantitative Analysis 29: 31-56.
11. Jiao L and Li Y (2005). Quantum-inspired immune clonal optimization. Proceedings of 2005 International Conference on Neural Networks and Brain (ICNN&B 2005) 13-15 Oct. 2005, 461–466, IEEE Press.
12. Lagarias J C, Reeds J A, Wright M H and Wright P E (1998). Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. SIAM Journal of Optimization, Vol. 9, Number 1: 112–147, 1998.
13. Lee C-D, Chen Y-J, Huang H-C, Hwang R-C and Yu G-R (2004). The non-stationary signal prediction by using quantum NN. Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics, 10-13 Oct. 2002, 3291–3295, IEEE Press.
14. Li Y, Zhang Y, Zhao R and Jiao L (2004). The immune quantum-inspired evolutionary algorithm. Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics, 10-13 Oct. 2002, 3301–3305, IEEE Press.
15. Narayanan A and Moore M (1996). Quantum-inspired genetic algorithms. Proceedings of IEEE International Conference on Evolutionary Computation, May 1996, 61–66, IEEE Press.
16. Rechenberg I (1973). Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution. Fromman-Holzboog Verlag Stuggart.
17. Skiadopoulos G, Hodges S and Clewlow L (1999). The Dynamics of the S&P 500 Implied Volatility Surface. Review of Derivatives Research 3: 263-282.
18. Spector L (2004). Automatic Quantum Computer Programming: A Genetic Programming Approach. Kluwer Academic Publishers, Boston, MA.
19. Tsai X-Y, Chen Y-J, Huang H-C, Chuang S-J and Hwang R-C (2005). Quantum NN vs NN in Signal Recognition. Proceedings of the Third International Conference on Information Technology and Applications (ICITA 05), 4-7 July 2005, 308–312, IEEE Press.
20. Yang S, Wang M and Jiao L (2004). A genetic algorithm based on quantum chromosome. Proceedings of IEEE International Conference on Signal Processing (ICSP 04), 31 Aug- 4 Sept. 2004, 1622–1625, IEEE Press.
21. Yang S, Wang M and Jiao L (2004). A novel quantum evolutionary algorithm and its application. Proceedings of IEEE Congress on Evolutionary Computation 2004 (CEC 2004), 19-23 June 2004, 820–826, IEEE Press.
22. Yang S, Wang M and Jiao L (2004). A Quantum Particle Swarm Optimization. Proceedings of the Congress on Evolutionary Computation 2004 1:320–324, New Jersey: IEEE Press.