01 Jun 2008

A Quantum Calculus Formulation of Dynamic Programming and Ordered Derivatives

John E. Seiffertt IV
Missouri University of Science and Technology, jes0b4@mst.edu

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Recommended Citation

J. E. Seiffertt and D. C. Wunsch, "A Quantum Calculus Formulation of Dynamic Programming and Ordered Derivatives," *Proceedings of the IEEE International Joint conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)*, Institute of Electrical and Electronics Engineers (IEEE), Jun 2008.
The definitive version is available at https://doi.org/10.1109/IJCNN.2008.4634327

A Quantum Calculus Formulation of Dynamic Programming and Ordered Derivatives

John Seiffertt, *Student Member, IEEE* and Donald C. Wunsch II, *Fellow, IEEE*

Abstract— **Much recent research activity has focused on the theory and application of quantum calculus. This branch of mathematics continues to find new and useful applications and there is much promise left for investigation into this field. We present a formulation of dynamic programming grounded in the quantum calculus. Our results include the standard dynamic programming induction algorithm which can be interpreted as the Hamilton-Jacobi-Bellman equation in the quantum calculus. Furthermore, we show that approximate dynamic programming in quantum calculus is tenable by laying the groundwork for the backpropagation algorithm common in neural network training. In particular, we prove that the chain rule for ordered derivatives, fundamental to backpropagation, is valid in quantum calculus. In doing this we have connected two major fields of research.**

Index Terms— **dynamic programming, quantum calculus, time scales, backpropagation, dynamic equations**

I. Introduction

Quantum calculus is the modern name for the investigation of the calculus without limits which began with Euler and currently enjoys ties to abstract algebra and has found application in the quantum mechanics literature. The book by Kac and Cheung [20] covers many of the fundamental aspects of the quantum calculus. As this field becomes more widely researched, an increasing number of application areas are being discovered. For example, a recent study of financial derivative pricing realized a quantum calculus analog of the Black-Scholes equation [22]. Additionally, it has been shown that quantum calculus is a subfield of the more general mathematical field of time scales calculus. Time scales provide a unified framework for studying dynamic equations on both discrete and continuous domains. The texts by Bohner and Peterson ([11],[12]) collect much of the core theory in the calculus of time scales. Applications of this mathematics include population biology [7], geometric analysis [10], real time communications networks [18], intelligent robotic control [17], adaptive sampling [16], approximation theory [28], macroeconomics [1], and financial engineering [27], among others. Many of these are ideal areas for Approximate Dynamic Programming (ADP) ([23], [29]).

Dynamic programming itself has been extensively applied in computational economics ([21], [30], [32]) and asset price modeling [13], among a host of engineering topics with which we assume the reader is familiar. Thus, with the recent work on quantum calculus applications to finance, it is worthwhile to extend the investigation to other computational methods which have enjoyed success in economics and finance. Dynamic programming concerns itself with allocation of resources under uncertainty ([2],[3]) and we believe that time scales calculus, and quantum calculus in particular, may be the appropriate mathematical framework in which to cast an important class of decision problems.

This paper presupposes familiarity with the core ideas of dynamic programming. Quantum calculus, however, is introduced and reviewed in section II and the basic structure of a dynamic programming problem as well as derivation of the dynamic programming algorithm are presented in section III. Section IV provides the foundation for backpropagation on q-time scales by proving the chain rule for ordered derivatives originally introduced for the continuous case by Werbos. Section V concludes with notes on further research that can help leverage this newly developing area of mathematics into applications of engineering interest.

II. Overview of the Quantum Calculus

A time scale $\mathbb{T}$ is any nonempty closed subset of the real line $\mathbb{R}$. Examples include the integers $\mathbb{Z}$, the scaled integers $h\mathbb{Z}$ (where $h > 0$ is a scaling factor), as well as more mathematically adventurous entities such as the Cantor set. In studying quantum calculus we are concerned with a specific time scale, called the q-time scale, defined as follows:

$$\mathbb{T} = q^{\mathbb{Z}} = \{q^k : k \in \mathbb{Z}\}$$

such that $q > 1$. Dynamic equations in the quantum calculus, then, have domain $q^{\mathbb{Z}}$. It is worth noting that the quantum calculus converges to the classical continuous calculus in the limit as q approaches 1 from above.

Important in the study of any time scale are three characteristic functions. Let $\mathbb{T}$ be a time scale. The *forward jump operator* σ is defined as $\sigma(t) = \inf\{x \in \mathbb{T} : x > t\}$. The

Manuscript received November 30, 2007. This work was supported in part by the Mary K Finley Endowment at the Missouri University of Science and Technology*.

John Seiffertt is with the Applied Computational Intelligence Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology* (e-mail: jes0b4@mst.edu)

Donald C. Wunsch II is director of the Applied Computational Intelligence Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology* (e-mail: dwunsch@mst.edu)

*(Formerly known as University of Missouri-Rolla)

978-1-4244-1821-3/08/$25.00©2008 IEEE

purpose of this function is to step through to the successor element of the time scale, in the sense given in its definition. For the time scale $\mathbb{T} = \mathbb{R}$, $\sigma(t) = t$ for all $t \in \mathbb{R}$, and for the time scale $\mathbb{T} = \mathbb{Z}$, $\sigma(t) = t + 1$. The q-time scale analog of the forward jump operator is given by $\sigma(t) = q^{t+1}$ as the q-time scale is said to be *isolated*. In a similar fashion, the *backward jump operator* ρ is defined as $\rho(t) = \sup\{x \in \mathbb{T}: x < t\}$. Central to many formulas of the time scales calculus, the *graininess function* μ is given by $\mu(t) = \sigma(t) - t$. Note that when $\mathbb{T} = \mathbb{Z}$, $\mu(t) = 1$ and when $\mathbb{T} = \mathbb{R}$, $\mu(t) = 0$. The graininess of the q-time scale can be shown to be $\mu(t) = q^t(q-1)$ via application of the definition of the forward jump operator σ and some algebra.

To discuss calculus on the q-time scale we need to define a derivative. The *q-differential* of a function *f* is given by

$$d_q f(x) = f(qx) - f(x) \tag{1}$$

and the *q-derivative* of the function f is defined by the following expression:

$$D_q f(x) = \frac{d_q f(x)}{d_q x} = \frac{f(qx) - f(x)}{(q-1)x}. \tag{2}$$

Further derivatives can be defined in a manner analogous to their real counterparts. For example, the *second q-derivative* is defined as

$$D_q^2 f(x) = \frac{D_q f(qx) - D_q f(x)}{d_q x} \tag{3}$$

The standard rules for differentiation of products and quotients apply in quantum calculus:

$$\begin{aligned} d_q(f(x)g(x)) &= g(qx)d_q f(x) + f(x)d_q g(x) \\ &= f(qx)d_q g(x) + g(x)d_q f(x) \\ d_q \frac{f(x)}{g(x)} &= \frac{g(x)d_q f(x) - f(x)d_q g(x)}{g(qx)g(x)} \end{aligned} \tag{4}$$

To prove the dynamic programming algorithm, we will require the use of induction in the quantum calculus. Induction on time scales other than the integers must take into consideration the fact that successor and predecessors are not necessarily uniform. Therefore, use of the forward and backward jump operators is necessary. A form of induction holds true on time scales. If $t_0 \in \mathbb{T}$ and $S(t)$ is a statement for each $t \in [t_0, \infty)$ such that the following four conditions hold:

1. $S(t_0)$ is true
2. $S(t)$ being true at a right-scattered t forces $S(\sigma(t))$ to be true
3. $S(t)$ being true at a right-dense t forces $S(t')$ to be true for all t' in a right-neighborhood of t
4. $S(t')$ being true for all $t' \in (t, t_0]$ when t is right-dense forces $S(t)$ to be true

then it can be concluded that $S(t)$ is true for all $t \in [t_0, \infty)$.

Further details can be found in [11]. For our purposes this basic overview will suffice. There is also a dual version which involves left-scattered and left-dense intervals and uses the backwards jump operator $\rho(t)$ to perform backwards inductive proofs. It is actually this dual version that we use in the proof of the Dynamic Programming Algorithm. Note further that for the q-time scale only conditions 1 and 2 need to be met since this time scale lacks dense points.

For the proof of ordered derivatives in the quantum calculus, we will need to employ differentiation with respect to a function. The Stieltjes integral provides a means for this in the traditional calculus, where we have the relation $\int f dg = \int f g' dt$. We define a similar construction on the q-time scale where $\int f d_q g = \int f d_q g \, d_q t$. Now, as the q-time scale is a fundamentally discrete set the integrals become expressed as summations as shown at the end of this section. We present this notation in the more general case to maintain consistency with the relationship between the q-calculus and general time scales where the formula is given by $\int f \Delta g = \int f g^{\Delta} \Delta t$, where the symbol Δ denotes the idea of *delta differentiation*, which is the generalization of q-differentiation to any time scale. Deeper analysis of these concepts is beyond the scope of this paper; the interested reader is directed to [9], [12], and [19] for further information on the theory of integration on time scales.

Let $f: \mathbb{T} \to \mathbb{R}$ and $g: \mathbb{T} \to \mathbb{R}$ be functions on a q-time scale $\mathbb{T}$, and define the following q-derivative:

$$\frac{d_q f}{d_q g} = f^{d_q g} \tag{5}$$

This is simply the q-time scale analog of the expression $\frac{\partial f}{\partial g} = \frac{\partial f}{\partial t} / \frac{\partial g}{\partial t}$ from classical analysis. We will make use of the notation $f^{d_q g}$ in our proof of ordered derivatives on time scales.

Finally, a further note on the translation of integrals. Let $f: \mathbb{T} \to \mathbb{R}$ where $\mathbb{T} = \{t_0, t_1, \ldots\}$. Then $\int_{t_0}^{t} f(\tau)\Delta\tau = \sum_{\tau \in [t_0, t)} f(\tau)\mu(t)$, where μ is the graininess function of the time scale $\mathbb{T}$. In particular, for the q-time scale we arrive at the following:

$$\int_{t_0}^{t} f(\tau)\Delta\tau = \sum_{i=t_0}^{i=q^{t-1}} f(q^i)q^i(q-1) \tag{6}$$

This construction is important in our discussions of dynamic programming and backpropagation, as it is convenient and illuminating to first consider the general time scale formulation in some cases before delving into the particulars of the quantum calculus version.

III. Dynamic Programming in the Quantum Calculus

Dynamic programming concerns itself with the solution of multistage decision problems. Key elements include a set of *decision points* for making control choices, definitions of system *states* which capture all salient details of interest to the modeler, *policies* which map states to controls, a set of *costs/rewards* for each state/decision choice, a *cost-to-go function* measuring the costs of future action under a given policy, and a possible source of *random disturbance* to emulate the probability that the decision maker is operating under conditions of less than perfect information regarding the consequences of control selections upon state transitions. The reader unfamiliar with the above framework is directed to the texts [4], [5], [25], and [31] for further details.

We set out with the following definitions: decision points t contained in a q-time scale $\mathbb{T}$ with a terminal point T, a set of controls for each state given by $c(x(t),t)$, random disturbances modeled by a stochastic term $w(t)$, a cost/reward function denoted by $r(x(t),c(x(t),t),w(t),t)$ with terminal point T defined piecewise as $r_T(x(T))$, and a dynamic system where states $x(t)$ evolve according to the following rule:

$$d_q x(t) = f(x(t),c(x(t),t),w(t),t), \tag{7}$$

For policies π, we require each set of state-control pairs to represent a valid association for both the space and time dimensions. The *tail* of the policy π, denoted as π^t, chronicles the sets of state-action pairs starting at decision point t and ending at the terminal point T. This notion is critical for discussion of the optimality principle and the subsequent derivation of the dynamic programming algorithm. The cost-to-go function is given by

$$\begin{aligned} J_\pi(x(t_0),t_0) &= E\langle \int_{t_0}^{q^T} r(x(\tau),c(x(\tau),\tau),w(\tau),\tau)\Delta\tau\rangle \\ &= (q-1)E\langle \sum_{i=t_0}^{q^{T-1}} q^i r(x(q^i),c(x(q^i),q^i),w(q^i),q^i)\rangle \end{aligned} \tag{8}$$

where the expansion in terms of q-time scales makes use of equation (6).

Before continuing, a brief aside on the nature of the expectation in the cost-to-go function is in order. In service of the desire to balance the dueling needs of robust modeling and mathematical tractability, we force a requirement of countability on the random disturbance term $w(t)$. This decision limits us from considering stochastic terms such as Brownian motion or Gaussian noise while still permitting a wide array of useful probabilistic models. For example, the standard discrete-time interpretation of $w(t)$ takes the form of a transition probability matrix P whose entries (i,j,k) indicate the chance the system evolves from state i to state j under control choice k. While this is perhaps the most widely used form of the system, we will proceed with the more general equation given by (8) in the interest of maximal mathematical abstraction and utility for our theorems.

Regardless of the nature of our disturbance terms, we define an *optimal policy* π^* to be one which minimizes the cost-to-go functional J. The corresponding optimal cost-to-go functional is denoted

$$J^*(x(t_0),t_0) = \min_\pi J_\pi(\,x(t_0),t_0) \tag{9}$$

where the *min* is considered over all policies. The goal of the dynamic programming problem is to calculate an optimal policy π^*. The most basic process by which this is achieved is called the *Dynamic Programming Algorithm*.

This algorithm is a form of backwards induction. Starting from the terminal decision point T and following a schedule of recursively defined steps backwards in time towards the initial point t_0, the optimal policy can be calculated even in a stochastically rich environment. The algorithm begins with setting $J(x(T),T) = r_T(x(T))$ and proceeds via the following rule:

$$\begin{aligned} J(x(t),t) = \min_c E\langle r(x(t),c(x(t),t),w(t),t) \\ + J(f(x(t),c(t),w(t),t),q^{t+1})\rangle \end{aligned} \tag{10}$$

This recursion is a consequence of Bellman's Principle of Optimality, which states that any optimal policy must be still optimal when enacted on any tail of the system. That is, the solution which minimizes the cost-to-go function starting at any given point t,

$$(q-1)E\langle \sum_{i=t}^{q^{T-1}} q^i r(x(q^i),c(x(q^i),q^i),q^i)\rangle, \tag{11}$$

is simply the portion of the optimal policy π^* which coincides with the particular tail in question. The justification of this principle runs as a proof by contradiction: If the tail problem had a different solution than that given, then the cost-to-go function could be minimized further by changing out the optimal policy's tail with this alternate policy, thus prohibiting the optimal policy from being, in fact, optimal. This cannot be the case so the optimality principle must hold. This concept is used in the proof of the dynamic programming algorithm given below.

Our proof follows that of [4] and suffices to establish the viability of dynamic programming in quantum calculus.

Theorem 1: The policy which minimizes the dynamic programming recursion (10) for all states and all times is optimal.

Proof: Set $J^*(x(T),T) = r_T(x(T))$ and proceed, via quantum calculus induction, to show that following the dynamic programming algorithm's update rule yields the optimal policy each step of the way, i.e. that $J^*(x(t),t) = J(x(t),t))$ for all $t \in \mathbb{T}$. Since the nature of this algorithm is to proceed backwards in time, we will use the dual version of time scales induction as described in section II of this paper. (Recall that the backwards jump operator $\rho(t) = q^{t-1}$ for the q-time scale. However, we will maintain the use of the symbol $\rho(t)$ and trust no confusion will arise. This notation has the advantage of more closely mirroring the form of the version of the quantum calculus induction algorithm we invoke in the proof.)

Letting $t = T$ yields, by definition,

$$J^*(x(N),N) = r_T\big(x(T)\big) = J(x(N),N)). \tag{12}$$

Now assume $J^*(x(t),t) = J(x(t),t)$ for some time point $t \in \mathbb{T}$ and all states $x(t)$. Then we have

$$J^*\big(x(\rho(t)),\rho(t)\big) = r(x(t),c(x(t),t),w(t),\rho(t)) + \min_{c,\pi} E\,\langle (q-1)E\,\langle \sum_{i=t_0}^{q^{T-1}} q^i r(x(q^i),c(x(q^i),q^i),w(q^i),q^i)\rangle\rangle . \tag{14}$$

Note that the minimization is taken term by term over all controls and policies, respectively. We now use the principle of optimality to distribute the *min* through the expectation, as the tail problem is an optimal policy for the sub-problem contained within the tail. This yields the following:

$$J^*\Big(x\big(\rho(t)\big),\rho(t)\Big) = \min_{c} E\langle r\big(x(t),c(x(t),t),w(t),\rho(t)\big)\rangle + \min_{\pi} E\,\langle (q-1)E\,\langle \sum_{i=t_0}^{q^{T-1}} q^i r(x(q^i),c(x(q^i),q^i),w(q^i),q^i)\rangle\rangle \tag{15}$$

Using the definition of $J^*(x(t),t)$, which subsumes the term minimized over the policy, we can reduce this expression to

$$J^*\big(x(\rho(t)),\rho(t)\big) = \min_{c} E\langle r(x(t),c(x(t),t),w(t),\rho(t)) + J^*(x(t),t)\rangle$$

By the induction hypothesis, we know the optimal cost-to-go is equivalent to the approximation due to the dynamic programming algorithm. Thus, we write

$$J^*\big(x(\rho(t)),\rho(t)\big) = \min_{c} E\langle r(x(t),c(x(t),t),w(t),\rho(t)) + J(x(t),t)\rangle$$

which, by definition, is simply

$$J^*\big(x\big(\rho(t)\big),\rho(t)\big) = J\big(x\big(\rho(t)\big),\rho(t)\big) \tag{16}$$

Which, in turn, is our desired result.■

With this, the dynamic programming algorithm is shown to work in quantum calculus. In fact, this can be interpreted as the Hamilton-Jacobi-Bellman equation in the quantum calculus, as the q-time scale is isolated. It should be noted, however, that this algorithm is quite computationally expensive, particularly for industrial-scale problems [23]. To circumvent this failing, suboptimal methods are routinely employed. Collectively called Approximate Dynamic Programming (ADP), these algorithms seek to calculate sub-optimal policies to whatever degree of accuracy is required by a given application. These techniques are tied quite intimately to backpropagation, and section IV of this paper will provide a proof of the foundations of backpropagation in quantum calculus. In this way ADP as well as optimal dynamic programming is shown to have solid footing on q-time scales.

IV. Ordered Derivatives and Backpropagation

To generalize from dynamic programming in quantum calculus to approximate dynamic programming we need to investigate backpropagation. This derivative calculation engine is often utilized in neural network training algorithms and figures prominently in ADP techniques ([23], [24], [29]). It is therefore of interest to demonstrate its validity on q-time scales. In his PhD dissertation ([36], reprinted in [34]), Paul Werbos introduced the notion of an ordered derivative as separate from derivatives of typical application in physical systems. The use of these ordered derivatives is motivated by the particular character of analysis of social systems as compared to a system operating under strictly physical dynamics. Relying on a series of ordered variables, each a consequence of the previous variables in the system, Werbos notes that the traditional total derivative chain rule fails to provide adequate calculations for investigation of such an integrated social system as well as for a wide array of other mathematically related connectionist systems, the model application given in Werbos's dissertation being that of political forecasting. Werbos thereby establishes backpropagation as a viable manner in which to calculate derivatives not only in neural network training but for the manipulation of weights in a broad spectrum of adaptive systems. To complete the required calculations Werbos proved a special chain rule for these ordered derivatives that worked for the applications in which the traditional chain rule of continuous analysis broke down. Further discussion of ordered derivatives can be found in [33].

In this section we formulate the ordered derivative in quantum calculus and prove the chain rule Werbos derived obtains in this alternate environment. We conclude that the backpropagation approach to q-derivative calculation is as valid as the one for the classical derivative and, as such, neural network training on quantum calculus may follow its traditional counterpart.

Following [35], we define the following system variables: input $X(t)$, output $\bar{Y}(t)$ which approximates the target output $Y(t)$. We denote the dimension of the outputs by n. The relationship between inputs and outputs is determined via a series of adaptive weights W. It is the goal of backpropagation, and hence the ordered derivative calculations, to tune W to reduce an error measure between $\bar{Y}(t)$ and $Y(t)$. The implicit time scale is an isolated subset of $\mathbb{Z}$. Throughout this section we will consider time scales of higher generality and will assume $t \in \mathbb{T}$, where $\mathbb{T}$ is not restricted to a subset of $\mathbb{Z}$. We will let T denote the number of inputs or the time the system is allowed to compute.

We define the following measure of predictive system error:

$$\begin{aligned} E &= \int_1^{\sigma(T)} E(\tau)\Delta\tau \\ &= \int_1^{\sigma(T)} \sum_{i=1}^{n} \left(\frac{1}{2}\right)(\bar{Y}_i(\tau) - Y_i\,(\tau))^2 \Delta\tau \end{aligned} \tag{17}$$

This form represents the standard least-squares error measure in common use among statistical analysts. Note that the integral plays the part of the generalized summation operation and is appropriate for any time scale $\mathbb{T}$ free of restriction to an isolated or, more specifically, quantum case. The proper summation that remains in the equation runs over the dimension of the output vector: a fixed scalar value unrelated to our time scale $\mathbb{T}$ and thus not subsumed by an integral.

Our purposes require investigation of a specific time scale. Using equation (6), we can translate (17) into the terms of quantum calculus as follows:

$$\begin{aligned} E &= \int_{t_0}^{q^n} E(\tau)\Delta\tau \\ &= \sum_{i=0}^{n-1} E\,(q^i)q^i(q-1) \\ &= (q-1)\sum_{i=0}^{n-1}\frac{q^i}{2}(\bar{Y}_i(q^i)-Y_i(\,q^i))^2. \end{aligned} \tag{18}$$

Further analysis of the network equations requires the chain rule for ordered derivatives, as this is the tool used to transform the network equations into a calculator of the proper updates of a system's adaptive weights via backpropagation. As a preamble to the chain rule theorem, it is therefore necessary to define what we mean by an ordered derivative.

Let $x_1(t), x_2(t) \dots, x_n(t)$ be an ordered sequence of variables with $t \in \mathbb{T}$. These variables represent stages of a larger calculation (e.g., layers, in a sense, of a multi-layer perceptron network) and follow a recursion given by

$$x_i = f_i(x_{i-1}, x_{i-2}, \dots, x_1)$$

so that we can speak meaningfully of causation as a basis for the relationships among the x_i's. We are interested in determining the way the error E changes with respect to one of the x_i's, i.e. we want to calculate d_qE/d_qx_i, or, using the notation from the preliminaries section, we want to calculate $E^{d_qx_i}$. Following [34], we set up the error as a sequence of recursive functions such that

$$E_n(x_n, \dots, x_1) = x_n \tag{19}$$

and

$$E_{i-1}(x_{i-1}, \dots, x_1) = E_i(f_i(x_{i-1}, \dots, x_1), x_{i-1}, \dots, x_1). \tag{20}$$

Then, the ordered derivative of E, which equals x_n via our construction, is defined to be

$$E^{d_q^+x_i} = \frac{d_qE_i}{d_qx} = E_i^{d_qx_i}. \tag{21}$$

The backpropagation algorithm of derivative calculation hinges on sifting the network equations through the ordered differentiation operator. In particular, the chain rule for ordered derivatives plays a key role. The following theorem establishes this chain rule for quantum calculus.

Theorem 2: $E_j^{d_qx_i} = \sum_{k=j+1}^{n} x_n^{d_q^+x_k} f_k^{d_qx_i}$

Proof: As in [34], we proceed by induction on j. We will start with $j = n-1$ and end with $j = i$. With $j < i$, the recursive definitions of the f_i's and x_i's force the terms $f_k^{d_qx_i}$ to zero and, therefore, they do not contribute to the summation. So, it will suffice to consider j in the range $n-1$ to i.

Let $j = n-1$. Then our hypothesis becomes

$$E_{n-1}^{d_qx_i} = x_n^{d_q^+x_{n-1+1}} f_{n-1+1}^{d_qx_i} = x_n^{d_q^+x_n} f_n^{d_qx_i}. \tag{22}$$

Calling on the definition of the sequence of E_i's, we see that $E_{n-1}^{d_qx_i} = f_n^{d_qx_i}$ so that, since $x_n^{d_q^+x_n} = 1$, the claim is proven.

Now assume the hypothesis is true for some $j+1 < n$. Our task is to show the claim holds for $j > i$. Consider $d_qE_j/d_qx_i = E_j^{d_qx_i}$. Since E is defined from $\mathbb{R} \to \mathbb{R}$, the delta derivative construction reduces to the traditional case. Also, by definition, $E_j(x_j, \dots, x_1) = E_{j+1}(f_{j+1}, x_j, \dots, x_1)$.

Therefore,

$$\begin{aligned} E_j^{d_qx_i} &= E_{j+1}^{d_qx_i} \\ &= \frac{d_qE_{j+1}}{d_qx_1}\frac{d_qx_1}{d_qx_i} + \frac{d_qE_{j+1}}{d_qx_2}\frac{d_qx_2}{d_qx_i}\cdots \\ &\quad + \frac{d_qE_{j+1}}{d_qx_j}\frac{d_qx_j}{d_qx_i} \\ &\quad + \frac{d_qE_{j+1}}{d_qf_{j+1}}\frac{d_qf_{j+1}}{d_qx_i} \end{aligned} \tag{23}$$

for $i \le j$. From our definition from the preliminaries applied to our recursive definition of the ordered variables x_i, we have that $d_qx_k/d_qx_i = 0$ when $k < i$, as the preceeding variables in the order are unaffected by the later variables in the causation chain. This result allows us to reduce our equation to $E_j^{d_qx_i} = \frac{d_qE_{j+1}}{d_qx_i} + \frac{d_qE_{j+1}}{d_qf_{j+1}}\frac{d_qf_{j+1}}{d_qx_i}$. We collapse the first remaining term so that it matches the form of our induction hypothesis $E_{j+1}^{d_qx_i} = \sum_{k=j+2}^{n} x_n^{d_q^+x_k} f_k^{d_qx_i}$, giving us

$$\begin{aligned} E_j^{d_qx_i} &= \sum_{k=j+2}^{n} x_n^{d_q^+x_k} f_k^{d_qx_i} + \frac{d_qE_{j+1}}{d_qf_{j+1}}\frac{d_qf_{j+1}}{d_qx_i} \\ &= \sum_{k=j+1}^{n} x_n^{d_q^+x_k} f_k^{d_qx_i} \end{aligned} \tag{24}$$

which is our desired result.■

Thus, the chain rule for ordered derivatives in quantum calculus is established. With this result, we are able to construct neural network architectures in quantum calculus and train them via backpropagation. While the traditional chain rule of classical analysis fails to hold for ordered derivatives, the chain rule for ordered derivatives does hold on q-time scales. Since time scales in general, and q-time scales in particular, may be the appropriate mathematical framework to discuss a certain class of resource allocation problems, and dynamic programming concerns itself with optimization of multi-stage decision scenarios, a quantum calculus approach to the approximation of the optimal solution becomes an exciting new area of computational decision theory.

V. Conclusion and Extensions

The calculus of time scales in general, and the q-time scale studied in the quantum calculus in particular, is an increasingly relevant and emerging area of mathematics with wide-ranging opportunities for application. We have established the functionality of ordered derivatives in the quantum calculus, and by doing so extended backpropagation to this domain. Furthermore, we showed the dynamic programming algorithm, derived from Bellman's principle of optimality, also obtains on q-time scales. Together, backpropagation and the dynamic programming algorithm form the foundation of ADP. As such, we have connected two major fields of research.

It is important to note that many of the ideas in dynamic programming, including the Bellman equation itself, have their birth in the calculus of variations ([15], [26], [37]) which is just now beginning its translation into the language of time scales ([6], [8], [14]). While the calculus of variations is now considered to be a proper subset of functional analysis, a further investigation into dynamic programming in the quantum calculus may do well to take into account classical variational principles. This storied domain still has much to offer researchers in optimization.

In particular, and in keeping with the place held by the q-calculus in physics, it is interesting to investigate quantum mechanical extensions of the Hamilton-Jacobi-Bellman equation for use in dynamic programming. While there is a necessary limit to the similarities between the motion of a particle in a physical system and the control path of an agent in a multi-stage decision process, we believe that this limit has yet to be reached under the current Hamilton-Jacobi-Bellman theoretics. The impact of the quantum calculus on computational decision theory may not have yet reached its peak, and we believe further thought in these directions can yield great rewards.

References

[1] Atici, F.M., Biles, D.C., & Lebedinsky, A. (2006) An application of time scales to economics. *Mathematical and Computer Modelling.* 43(7-8). pp 718-726.

[2] Bellman, R. (1957) Dynamic Programming. Princeton University Press. Princeton, NJ.

[3] Bellman, R., & Dreyfus, S. (1962) Applied Dynamic Programming. Princeton University Press. Princeton, NJ.

[4] Bertsekas, D. (2000) Dynamic Programming and Optimal Control, Second Edition, Vols 1 and 2. Athena Scientific. Belmont, MA.

[5] Bertsekas, D., & Tsitsiklis, J. (1996) Neuro-Dynamic Programming. Athena Scientific. Belmont, MA.

[6] Bohner, M. (2004) Calculus of variations on time scales. *Dynamic Systems and Applications*. 13. pp 339-349.

[7] Bohner, M., Fan, M., & Zhang, J. (2007) Periodicity of scalar dynamic equations and applications to population models. *Journal of Mathematical Analysis and Applications,* 330(1): 1-9.

[8] Bohner, M., & Guseinov, G. (2007) Double integral calculus of variations on time scales. *Computers and Mathematics with Applications.* 54(1). pp 46-57.

[9] Bohner, M., & Guseinov, G. (2005) Multiple Integration on time scales. Dynamic Systems and Applications, 14(3-4): 579-606

[10] Bohner, M., & Hudson, T. (2007) Euler-type boundary value problems in quantum calculus. *International Journal of Applied Mathematics and Statistics* 9(J07). pp 19-23.

[11] Bohner, M. & Peterson, A. (2001) Dynamic Equations on Time Scales: An Introduction with Applications. Birkhauser, Boston.

[12] Bohner, M. & Peterson, A. (2003) Advances in Dynamic Equations on Time Scales. Birkhauser, Boston.

[13] Duffie, D. (2001) Dynamic Asset Pricing Theory: Third Edition. Princeton University Press.

[14] Ferreira, R., & Torres, D. (2007) Remarks on the calculus of variations on time scales. *Int J Econ Ecol Stat.* 9. pp 65-73.

[15] Gelfand, I.M., & Fomin, S.V. (2000) Calculus of Variations. Dover. Mineola, NY.

[16] Gravagne, I., Davis, J., DaCunha, J., & Marks II, R. (2004) Bandwidth reduction for controller area networks using adaptive sampling. *Proc. Int. Conf. Robotics and Automation* New Orleans, LA, April 2004, pp. 5250 – 5255

[17] Gravagne, I., Davis, J., Marks II, R. (2005) How deterministic must a real-time controller be? *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Alberta, Canada.Aug. 2-6, 2005, pp. 3856 – 3861

[18] Gravagne, I., Marks II, R., Davis, J., Dacunha, J. (2004) Application of time scales to real time communications networks. *American Mathematical Society Western Section meeting, special session on Time Scales*. University of Southern California.

[19] Guseinov, G. (2003) Integration on time scales. J. Math. Anal. Appl., Vol. 285, pp.107-127.

[20] Kac, V., & Pokman, C. (2001) Quantum Calculus. Springer. New York, NY.

[21] Miranda, M, & Fackler, P. (2002) Applied Computational Economics and Finance. MIT Press.

[22] Muttel, C. (2007) The Black Scholes Equation in Quantum Calculus. Master's Thesis. University of Missouri-Rolla.

[23] Powell, W. (2007) Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley Series in Probability and Statistics. Hoboken, NJ.

[24] Prokhorov, D., & Wunsch, D (1997) Adaptive Critic Designs. *IEEE Transactions on Neural Networks.* 8(5) 997-1007.

[25] Puterman, M. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons. New York, NY.

[26] Rund, H. (1966) The Hamilton-Jacobi Theory in the Calculus of Variations. Its Role in Mathematics and Physics. D. Van Nostrand. London, New York.

[27] Sanyal, Suman. (2008) Stochastic Dynamic Equations. *PhD Dissertation.* Missouri University of Science and Technology.

[28] Sheng, Q., Fadag, M., Henderson, J., & Davis, J. (2006) An Exploration of Combined Dynamic Derivatives on Time Scales and Their Applications. *Nonlinear Analysis:Real World Applications* 7, pp.395–413

[29] Si, J., Barto, A., Powell, W., & Wunsch, D. (2004). Handbook of Learning and Approximate Dynamic Programming. IEEE Press Series on Computational Intelligence. 2004.

[30] Stokey, N., Lucas, R., & Prescott, E. (1989) Recursive Methods in Economic Dynamics. Harvard University Press.

[31] Sutton, R., & Barto, A. (1998) Reinforcement Learning: An Introduction. MIT Press. Cambridge, MA.

[32] Testafasion, L, & Judd, K., eds (2006) Handbook of Computational Economics: Agent Based Computational Economics. North-Holland.

[33] Werbos, P. (2005) Backwards differentiation in AD and neural nets: past links and new opportunities. in Bucker, M., Corliss, G., Hovland, P., Naumann, U., & Boyana, N. (eds) *Automatic differentiation: applications, theory, and implementations.* Springer. New York, NY.

[34] Werbos, P. (1994) The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting.. Wiley. New York, NY

[35] Werbos, P. (1990) Backpropagation through time: What it does and how to do it. Proceedings of the IEEE. 78(10).

[36] Werbos, P. (1974) Beyond Regression. *PhD Dissertation.* Harvard University.

[37] Young, LC. (1969) Lectures on the calculus of variations and optimal control theory. WB Saunders Company. Philadelphia, PA.