American Journal of Mathematical and Computer Modelling
2026, Vol. 11, No. 3, pp. 112–119
https://doi.org/10.11648/j.ajmcm.20261103.11

SciencePG
Science Publishing Group

Research Article

Breaking the Curse of Dimensionality Using a Recursive Dynamic Programming Framework

Adebayo Kayode James[1, *], Alabi Taiye John[2], Omowaye Kehinde Solomon[1]

[1]Department of Mathematics, Ekiti State University, Ado Ekiti, Nigeria
[2]Department of Statistics, School of Applied Sciences, Lokoja, Nigeria

Abstract

The resolution of optimal control problems (OCPs) in real-world applications is frequently beset by the curse of dimensionality and the inherent nonlinearity of system dynamics. This paper presents a literature framework that combines recursive relationship formulations with the principles of dynamic programming (DP) to address these challenges. The research stresses on a perspective that geared towards bridging the gap between the theoretical approach of DP and the numerical computation that demands practical applications. Dynamic programming is a method that finds solutions to larger sub-problems after the problem has been reduced to smaller ones. The main idea is to embed recursive relationships directly within the DP framework. The hypothesis is that for a significant class of OCPs, particularly those characterized by certain structural properties or separable cost functions, the optimal decision at a given state can be expressed recursively as a function of decisions made in states or stages. The Recursive Dynamic Programming (RDP), embeds a state-dependent recursive structure directly within the DP iteration, enabling a more efficient traversal of the state space and the generation of near-optimal control policies. We demonstrate the efficacy of this approach and its application to the resource-constrained problem. The approach follows derivation of an analytical and a semi-analytical recursive expression for major decision variables or for the gradient function value, which are then applied iteratively within the DP. The RDP framework is shown to significantly reduce computational overhead compared to classical DP while maintaining a high degree of solution accuracy, offering a pragmatic and scalable pathway for tackling complex OCPs prevalent in engineering and economic systems.

Keywords

Optimal Control, Dynamic Programming, Recursive Relationships, Curse of Dimensionality

1. Introduction

The quest for optimality is a fundamental driver in the design, operation, and control of complex systems across a vast spectrum of human endeavor. From navigating the trajectory of a spacecraft to managing the intricate supply chains of a global corporation, the objective remains consistent: to steer a dynamic system from an initial state to a desired final state while minimizing or maximizing a specified performance criterion [1]. This pursuit is formalized mathematically within the domain of optimal control theory, a field that provides the intellectual and analytical tools to derive policies that are, in a

*Correspondence: Adebayo Kayode James (kayode.adebayo@eksu.edu.ng)

Received: 9 August 2026; **Accepted:** 17 August 2026; **Published:** 27 August 2026

defined sense, the best possible.

The formulation of a standard optimal control problem typically involves defining a set of state variables that describe the system's condition, a set of control variables that allow for external influence, and a dynamic model that governs the evolution of the states over time. The performance is quantified by a cost functional, and the task is to find the control history that maximizes this functional while respecting any constraints on the states or controls [2].

Consider a discrete-time optimal control problem defined over a finite horizon N:

$$\min_{\{u_k\}_{k=0}^{N-1}} J = \sum_{k=0}^{N-1} L(x_k, u_k, k) + \Phi(x_k) \tag{1}$$

subject to the system dynamics:

$$x_{k+1} = f(x_k, u_k, k),\ k = 0,1, \dots, N-1 \tag{2}$$

$$x_0 = x_{unit}$$

where $x_k \in \mathbb{R}^n$ denotes the state vector, $u_k \in \mathbb{R}^n$ denotes the control vector, $L(.)$ is the stage cost function, $\Phi(.)$ is the terminal cost function, and $f(.)$ represents the system dynamics.

The two principal pillars upon which the classical solution of these problems rests are the Calculus of Variations, leading to Pontryagin's Minimum Principle, and the dynamic programming approach, pioneered by Richard Bellman, [3, 4].

While the Minimum Principle offers an elegant, indirect method for solving OCPs through a set of necessary conditions, it can become analytically intractable for problems with complex state constraints or nonlinear dynamics. Dynamic programming, on the other hand, provides a direct and conceptually powerful framework by breaking down a multi-stage decision problem into a sequence of simpler, single-stage sub-problems [5]. Its core tenet, the Principle of Optimality, states that any tail of an optimal path must itself be optimal for the corresponding tail sub-problem.

For the discrete-time system described above, the Principle of Optimality leads to the Bellman equation:

$$V_k(x_k) = \min_{u_k \in U_k} [L(x_k, u_k, k) + V_{k+1}(f(x_k, u_k, k))] \tag{3}$$

with the terminal condition:

$$V_N(x_N) = \Phi(x_N) \tag{4}$$

where $V_k(x_k)$ represents the optimal cost-to-go function at time k from state x_k.

For continuous-time systems, this recursive relationship converges to the Hamilton-Jacobi-Bellman (HJB) equation, a partial differential equation whose solution yields the optimal cost-to-go function and, consequently, the optimal control policy:

$$0 = \min_{u \in U} \left[L(x, u, t) + \frac{\partial V}{\partial t} + \frac{\partial V}{\partial t} f(x, u, t) \right] \tag{5}$$

However, the theoretical elegance of DP is frequently overshadowed by its practical computational challenges, a phenomenon famously termed "the curse of dimensionality" [6]. For discrete-time systems, the DP algorithm requires a pointwise evaluation of the cost function over a discretized state space. As the number of state variables increases, the size of this state space grows exponentially, rendering the computational cost and memory requirements prohibitive.

To quantify this, let the state space be discretized with pp grid points per dimension. The number of states to be evaluate stands at:

$$|x| = p^n \tag{6}$$

where n is the number of state variables. As the number of state variables increases, the size of this state space grows exponentially, rendering the computational cost and memory requirements prohibitive. This fundamental limitation has historically confined the application of exact DP to problems with a small number of states, often referred to as 'toy problems' in [7].

Motivated by the need to overcome this barrier and to develop computationally tractable algorithms for real-world problems, researchers have explored a multitude of strategies. These include approximation techniques such as Approximate Dynamic Programming (ADP) or Neuro-Dynamic Programming [8], which use function approximation to represent the value function, and various decomposition and aggregation methods that reduce the effective size of the state space [9].

This paper introduces a perspective that aims to bridge the gap between the theoretical rigor of DP and the computational demands of practical applications. The central idea is to embed recursive relationships directly within the DP framework. The hypothesis is that for a significant class of OCPs, particularly those characterized by certain structural properties or separable cost functions, the optimal decision at a given state can be expressed recursively as a function of decisions made in states or stages.

Consider a problem structure where the optimal control can be expressed as:

$$u_k^* = g\left(x_k, \frac{\partial V_{k+1}}{\partial x_{k+1}}\right) \tag{7}$$

or more generally, where a recursive relationship of the form:

$$h_k(x_k, u_k, \lambda_{k+1}) = 0 \tag{8}$$

exists, with λ_{k+1} representing some function of future states.

This concept moves beyond the recursive nature of the Bellman equation itself. Instead, it proposes a hybrid framework, which we term Recursive Dynamic Programming (RDP). In

this approach, we derive analytical or semi-analytical recursive expressions for key decision variables or for the value function's gradient, which are then utilized within the DP iteration. This internal recursion serves to reduce the dimensionality of the search space at each decision epoch, as it effectively encodes a part of the optimization problem's structure, thereby guiding the search process and improving computational efficiency.

2. Literature Review

2.1. Foundations of Optimal Control

The formal study of optimal control, as a distinct discipline, crystallized in the mid-20th century, synthesizing concepts from classical calculus of variations with the burgeoning field of modern control theory. The foundational work of Lev Pontryagin and his colleagues in the Soviet Union led to the development of the Maximum Principle [10], a powerful set of necessary conditions for optimality.

For a continuous-time OCP with dynamics

$$\dot{x}(t) = f(x(t), u(t), t) \tag{9}$$

and cost functional:

$$J = \int_0^T L(x(t), u(t), t)dt + \Phi(x(T)) \tag{10}$$

Pontryagin's Minimum Principle states that the optimal control u∗(t) minimizes the Hamiltonian:

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T(t) f(x, u, t) \tag{11}$$

subject to the adjoint equations:

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - \left(\frac{\partial f}{\partial x}\right)\lambda^T(t) \tag{12}$$

and the terminal condition:

$$\lambda(T) = \frac{\partial \Phi}{\partial x}(x(T)) \tag{13}$$

The optimality condition is:

$$\frac{\partial H}{\partial u} = 0 \tag{14}$$

for unconstrained controls or:

$$H(x^*(t), u^*(t), \lambda^*(t), t) \leq H(x^*(t), u, \lambda^*(t), t)\ \forall u \in \tag{15}$$

for constrained controls.

The theorem, by introducing the concept of a co-state or adjoint variable, provides a framework for solving OCPs by converting them into a two-point boundary value problem (TPBVP). The Maximum Principle is exceptionally versatile and has been successfully applied to a wide range of problems, including time-optimal control [11], aerospace guidance [12], and economic growth models [13]. However, its application often requires significant analytical effort to solve the resulting TPBVP, which can be particularly challenging for systems with high-order dynamics, state inequalities, or discontinuities [14]. Also, the Maximum Principle provides only local necessary conditions, and finding a global optimum can be nontrivial.

Concurrently, Richard Bellman was developing a fundamentally different approach in the United States, rooted in the concept of dynamic programming [3, 4]. Bellman's Principle of Optimality asserts that for a multi-stage decision process, an optimal policy has the property that, whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [3]. This elegantly recursive principle leads to the Bellman equation as previously formulated.

For a deterministic discrete-time system, the optimal policy π∗ is defined as:

$$\pi_k^*(x_k) = \arg\ min_{u_k \in U_k}[L(x_k, u_k, k) + V_{k+1}(f(x_k, u_k, k))] \tag{16}$$

The resulting optimal cost satisfies the recursive relationship:

$$V_k(x_k) = min_{u_k}[L(x_k, u_k, k) + V_{k+1}(x_{k+1})] \tag{17}$$

with

$$x_{k+1} = f(x_k, u_k, k) \tag{18}$$

For continuous-time systems, this equation converges to the Hamilton-Jacobi-Bellman (HJB) partial differential equation, which provides a sufficient condition for optimality and, crucially, yields a globally optimal control law in feedback form [15]:

$$-\frac{\partial V}{\partial t}(x, t) = min_{u_k \in U_k}\left[L(x, u, t) + \frac{\partial V}{\partial x}(x, t) f(x, u, t)\right] \tag{19}$$

which provides a sufficient condition for optimality and, crucially, yields a globally optimal control law in feedback form [15]. The optimal control law is then:

$$u^*(x, t) = \arg\ min_{u \in U}\left[L(x, u, t) + \frac{\partial V}{\partial x}(x, t) f(x, u, t)\right] \tag{20}$$

The DP approach is particularly attractive because it provides a solution in closed-loop or feedback form. This means the optimal control at any state and time is determined as a function of the current state, which is immensely valuable for robustness against disturbances and modeling uncertainties. This fundamental advantage is highlighted by the work of Fleming and Rishel, [16], who rigorously established the connection between stochastic optimal control and the HJB equation.

For stochastic systems with dynamics:

$$dx = f(x,u,t)dt + \sigma(x,u,t)dW \tag{21}$$

where W is a Brownian motion process, the HJB equation becomes:

$$-\frac{\partial V}{\partial t}(x,t) = min_{u\in U}\left[L(x,u,t) + \frac{\partial V}{\partial x}f(x,u,t) + \frac{1}{2}Tr\left(\sigma^T\frac{\partial^2 V}{\partial x^2}\sigma\right)\right] \tag{22}$$

2.2. The Curse of Dimensionality and Its Implications

Despite its conceptual power, the practical application of classical dynamic programming is severely curtailed by what Bellman himself termed the 'curse of dimensionality' [6]. This issue arises from the computational demands of the algorithm, which necessitates a discretization of the state space. For a system with n state variables, if each dimension is discretized into k grid points, the total number of possible states to evaluate is k^n. As n grows, this number increases exponentially, leading to a computational and storage burden that quickly becomes insurmountable even for moderately high-dimensional systems [17].

The curse of dimensionality is in two forms [8]:

Computational Complexity: The number of operations required for each DP iteration is proportional to the size of the state space. For a single DP iteration, the computational cost is:

$$C_{DP} = O\left(N.\Pi_{i=1}^{n}p_i.\Pi_{j=1}^{m}q_j\right) \tag{23}$$

Where p_i is the number of grid points in the $i-th$ state dimension and q_j is the number of grid points in the j-th control dimension. The total cost over the entire horizon is:

$$C_{total} = O(N.p^n.q^m) \tag{24}$$

This exponential scaling makes direct DP unfeasible for problems with more than a few state variables.

Representational Complexity: Storing the value function, which is a table of values for each discretized state, also requires memory that scales exponentially with the number of state variables.

$$M_{DP} = O(\Pi_{i=1}^{n}p_i)O(P^N) \tag{25}$$

This fundamental limitation has spurred decades of research into developing methods that can circumvent or mitigate the curse of dimensionality. The literature on this topic is vast, and a comprehensive overview is beyond the scope of this section. However, the strategies can be broadly classified into two categories: exact methods for special problem structures, and approximate methods for general problems.

Exact methods often exploit special properties of the problem to reduce dimensionality. For instance, if the dynamics are linear and the cost is quadratic, the optimal control problem can be solved analytically using the Linear Quadratic Regulator (LQR) framework [18].

The LQR problem is defined as:

$$x_{k+1} = A_k x_k + B_k u_k \tag{26}$$

with quadratic cost:

$$J = \frac{1}{2}x_N^T S_N x_N + \sum_{k=0}^{N-1}\frac{1}{2}(x_k^T Q_k x_k + u_k^T R_k u_k) \tag{27}$$

The optimal control law for the LQR problem is linear feedback:

$$u_k^* = -K_k x_k \tag{28}$$

where the gain matrix K_k is computed backward in time using the Riccati equation:

$$K_k = (R_k + B_k^T S_{k+1}B_k)^{-1}B_k^T S_{k+1}A_k \tag{29}$$

$$S_k = Q_k + A_k^T S_{k+1}A_k - A_k^T S_{k+1}B_k(R_k + B_k^T S_{k+1}B_k)^{-1}B_k^T S_{k+1}A_k \tag{30}$$

with terminal condition $S_N = \Phi_x$, where Φ_x is the Hessian of the terminal cost.

Similarly, certain problems with separable cost functions or monotonic properties can sometimes be solved more efficiently using specialized decomposition techniques [19].

2.3. Approximate Dynamic Programming and Neuro-Dynamic Programming

The most prominent and successful approach for tackling large-scale OCPs is Approximate Dynamic Programming

(ADP), also referred to as Neuro-Dynamic Programming [8, 20]. The core idea of ADP is to use a parametric or non-parametric function approximation architecture to represent the value function or the Q-factor (the value of taking a particular action in a given state), rather than storing it in a large look-up table.

Let $\tilde{V}_k(x_k, \theta_k)$ be a parametric approximation of the true value function $V_k(x_k)$, where θ_k is a vector of parameters. The goal is to find θ_k such that:

$$\tilde{V}_k(x_k, \theta_k) \approx V_k(x_k) \; \forall x_k \in \chi \tag{31}$$

Common function approximation architectures include:

Linear Function Approximation:

$$\tilde{V}_k(x, \theta) = \theta^T \phi(x) = \sum_{i=1}^{d} \theta_i \phi_i(x) \tag{32}$$

where $\phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_d(x)]^T$ is a vector of basis functions.

Neural Network Approximation:

For a multi-layer neural network with l layers, the approximation can be expressed recursively as:

$$z^{(0)} = x \tag{33}$$

$$z^{(j)} = \sigma_j\left(W^{(j)} z^{(l-1)} + b^{(j)}\right) \text{ for } j = 1, \ldots, l-1 \tag{34}$$

$$\tilde{V}_k(x, \theta) = W^{(l)} z^{(l-1)} + b^{(l)} \tag{35}$$

where $\theta = \left\{W^{(j)}, b^{(j)}\right\}_{j=1}^{l}$ are the network parameters, and $\sigma_j(.)$ are activation functions.

The work of [8] provided a comprehensive and rigorous foundation for ADP, covering a range of algorithms such as policy iteration, value iteration, and Q-learning, all within the context of function approximation.

Policy Iteration alternates between:

1) Policy evaluation: Compute V^{π_k} for policy π_k

2) Policy improvement:

$$\pi_{k+1}(x) = \arg min_{u \epsilon U} \left[L(x, u) + \gamma V^{\pi_k}(f(x, u)\right]$$

Value Iteration directly updates the value function:

$$\tilde{V}_k(x, \theta) = min_{u \epsilon U}[L(x, u) + \gamma \tilde{V}_k(f(x, u)] \tag{36}$$

Q-learning learns the Q-function directly:

$$(x, u) = L(x, u) + \gamma V(f(x, u)) \tag{37}$$

The Q-learning update rule for a sample (x, u, x', r) is:

$$Q(x, u) \leftarrow Q(x, u) + \alpha[r + \gamma \min_u, Q(x', u') - Q(x, u)] \tag{38}$$

where α represents the rate of learning, and γ stands for the discount factor.

The use of artificial neural networks as function approximators has proven particularly successful due to their universal approximation capabilities and their ability to learn complex, non-linear mappings [21]. The key advantage of ADP is that it decouples the computational cost from the size of the state space. Instead of visiting every state, the algorithm samples a subset of states and uses the function approximator to generalize the value function to unvisited regions of the state space [8]. This makes it possible to solve OCPs with tens or even hundreds of state variables, a feat impossible for classical DP. Numerous applications have been reported, ranging from optimal power system control [22] and inventory management [23] to autonomous vehicle navigation [24].

However, ADP is not without its drawbacks. The performance of ADP algorithms is highly dependent on the choice of the function approximation architecture and the sampling strategy [20]. Poorly chosen approximators can lead to poor performance or even instability in the learning process [25].

Also, the theoretical approach ensured offered by ADP often tend to be weaker than those of exact DP, and the convergence of many ADP algorithms is to a local optimum rather than the global one [8]. Also, the behavior of these algorithms can be difficult to interpret and debug, which can be a significant barrier to adoption in safety-critical applications. The recent trend is the combination of DP with other computational intelligence techniques, such as evolutionary algorithms or particle swarm optimization [26]. These methods are used to search for optimal policies directly in the parameter space of a control policy, often guided by a DP-derived or heuristic value function. While these methods can be highly flexible and effective for complex problems, they do not offer the same degree of theoretical guarantees as DP or ADP.

2.4. Recursive and Decomposition-Based Methods in OCP

The concept of using recursive relationships in optimization is not new. Within the field of optimal control, several methods use decomposition and recursion to enhance computational efficiency.

Differential Dynamic Programming (DDP) is a local DP method that iteratively solves a series of linear-quadratic subproblems along a nominal trajectory [27, 28]. Starting from a nominal trajectory $\left\{x_k^{(0)}, u_k^{(0)}\right\}$, DDP expands the value function around the trajectory:

$$V_k(x_k + \delta x_k) \approx V_k(x_k) + V_{x,k}^T \delta x_k + \frac{1}{2} \delta x_k^T V_{xx.k} \delta x_k \tag{39}$$

The backward pass computes the optimal control modification:

$$\delta u_k^* = -K_k \delta x_k - \alpha_k \tag{40}$$

where K_k and α_k are computed from the derivatives of the dynamics and cost. The forward pass then updates the trajectory. The computational complexity of each DDP iteration is $O(Nn^3)$, which is independent of the state space discretization, making it particularly effective for high-dimensional continuous systems.

The DDP algorithm uses the following recursive equations during the backward pass:

$$Q_{xx,k} = L_{xx,k} + f_{x,k}^T V_{xx,k+1} f_{x,k} \tag{41}$$

$$Q_{uu,k} = L_{uu,k} + f_{u,k}^T V_{xx,k+1} f_{u,k} \tag{42}$$

$$Q_{ux,k} = L_{ux,k} + f_{u,k}^T V_{xx,k+1} f_{x,k} \tag{43}$$

$$K_k = Q_{uu,k}^{-1} Q_{ux,k} \tag{44}$$

$$\alpha_k = Q_{uu,k}^{-1}\left(L_{u,k} + f_{u,k}^T V_{x,k+1}\right) \tag{45}$$

$$V_{xx,k} = Q_{xx,k} - Q_{xu,k} Q_{uu,k}^{-1} Q_{ux,k} \tag{46}$$

$$V_{x,k} = Q_{x,k} - Q_{xu,k} Q_{uu,k}^{-1} Q_{u,k} \tag{47}$$

Another related area is the use of State-Dependent Riccati Equation (SDRE) methods for nonlinear control [29]. SDRE approaches treat the nonlinear dynamics as a set of linear dynamics with state-dependent coefficient matrices.

$$\dot{x}(t) = A\big(x(t)\big)x(t) + B\big(x(t)\big)u(t) \tag{48}$$

and then solve a Riccati equation at each time step:

$$A(x)^T P(x) + P(x)A(x) - P(x)B(x)R^{-1}B(x)^T P(x) + Q(x) = 0 \tag{49}$$

The control law is then:

$$u(t) = -R^{-1}B\big(x(t)\big)^T P\big(x(t)\big)x(t) \tag{50}$$

while not a DP method, SDRE leverages the structure of the problem to create a recursive-like computational scheme.

The idea of embedding recursive relationships within DP, as proposed in this paper, can be viewed as a more generalized and integrated version of these concepts. It aligns with the spirit of 'structured' DP, where the algorithm is tailored to exploit the specific mathematical structure of the problem, such as convexity [30], monotonicity [31], or separability [32] and [33].

For problems with convex cost functions and convex feasible sets, the optimization problem:

$$min_{u\epsilon U}[L(x,u) + V_{k+1}(f(x,u)] \tag{51}$$

becomes a convex optimization problem, which can be solved more efficiently. Similarly, for separable problems where:

$$L(x,u) = \sum_{i=1}^n L_i(x_i, u_i) \tag{52}$$

$$f(x,u) = \sum_{i=1}^n f_i(x_i, u_i) \tag{53}$$

The DP problem can be decomposed into independent subproblems, dramatically reducing computational complexity. The contribution of this paper is to formalize the process of "extracting" these recursive relationships from the problem structure and incorporating them into the DP algorithm itself, thereby creating a more efficient and targeted computational framework.

2.5. Research Gap

From the reviewed literature, several gaps are identified:

1) Existing dynamic programming frameworks are computationally expensive for high-dimensional systems.
2) Recursive structures are often treated as secondary computational tools rather than primary design components.
3) Limited research exists on explicitly embedding optimized recursive relationships into dynamic programming formulations for improved efficiency.
4) There is a lack of context-specific research addressing optimal control problems in developing real-life problems.

These gaps motivate the need for a structured development of recursive relationships that are directly embedded within dynamic programming techniques to enhance computational efficiency and applicability.

3. Conclusions

In conclusion, the existing literature reveals a clear dichotomy. On one hand, classical DP provides an elegant, theoretically sound method for global optimization but is computationally intractable for high-dimensional systems. On the other hand, ADP provides a pragmatic, scalable solution but often sacrifices theoretical guarantees and interpretability. The Recursive Dynamic Programming framework aims to carve out a middle ground, retaining the rigor and global perspective of DP while leveraging the problem's inherent structure (via recursive relationships) to significantly enhance computational tractability.

The literature demonstrates that while dynamic programming remains a powerful tool for solving optimal control problems, its practical implementation is hindered by computational limitations. Recursive relationships play a central role in the formulation of dynamic programming, yet they are often under-optimized in current approaches. This study positions

recursive embedding as a key strategy for improving efficiency in solving optimal control problems, particularly within computationally constrained environments.

Abbreviations

OCP	Optimal Control Problem
DP	Dynamic Programming
RDP	Recursive Dynamic Programming
ADP	Approximate Dynamic Programming
TPBVP	Two Point Boundary Value Problem
HJB	Hamilton Jacobi Bellman
LQR	Linear Quadratic Equation
SDRE	State Dependent Riccati Equation
DDP	Differential Dynamic Programming

Author Contributions

Adebayo Kayode James: Conceptualization, Methodology, Resources, Writing – review & editing

Alabi Taiye John: Data curation, Writing – original draft

Omowaye Kehinde Solomon: Formal Analysis, Investigation, Methodology

Data Availability Statement

The data supporting the outcome of this research work has been reported in this manuscript.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] D. E. Kirk, *Optimal Control Theory: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[2] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*. Washington, DC: Hemisphere Publishing, 1975.

[3] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[4] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503-515, 1954.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Belmont, MA: Athena Scientific, 2017, vol. I.

[6] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press, 1961.

[7] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2011.

[8] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[9] S. S. Rao, *Engineering Optimization: Theory and Practice*, 4th ed. Hoboken, NJ: John Wiley & Sons, 2009.

[10] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. New York: Interscience Publishers, 1962.

[11] M. Athans and P. L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications*. New York: McGraw-Hill, 1966.

[12] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Philadelphia, PA: SIAM, 2010.

[13] K. J. Arrow, "Applications of control theory to economic growth," in *Mathematics of the Decision Sciences, Part 2*, G. B. Dantzig and A. F. Veinott, Eds. Providence, RI: American Mathematical Society, 1968, pp. 85-120.

[14] M. A. Henson and D. E. Seborg, *Nonlinear Process Control*. Upper Saddle River, NJ: Prentice Hall, 1997.

[15] R. W. H. Sargent, "Optimal control," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 361-371, 2000.

[16] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*. New York: Springer-Verlag, 1975.

[17] P. R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Englewood Cliffs, NJ: Prentice Hall, 1986.

[18] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[19] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas*, 2nd ed. Princeton, NJ: Princeton University Press, 2009.

[20] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32-50, 2009.

[21] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2009.

[22] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement learning versus model predictive control: A comparison on a power system problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 517-529, 2009.

[23] W. B. Powell and J. M. Swiger, "A dynamic programming algorithm for the optimal control of a production-inventory system," *Naval Research Logistics*, vol. 36, no. 5, pp. 637-658, 1989.

[24] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2002, pp. 1398-1403.

[25] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computation*, vol. 6, no. 2, pp. 215-219, 1994.

[26] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: A survey," *Control Engineering Practice*, vol. 10, no. 11, pp. 1223-1241, 2002.

[27] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: American Elsevier, 1970.

[28] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005 American Control Conference*, vol. 1, 2005, pp. 300-306.

[29] Kayode James Adebayo, Taiye John Alabi, Olaosebikan Temitayo Emmanuel, Ademoroti Albert Olalekan, Ayinde Samuel Olukayode, Akinmuyise Mathew Folorunso, (2025), Dynamic Programming Approach to Solving Continuous-Time Linear Quadratic Regulator Problems, TWIST, 20(2), 273-281.

[30] J. R. Cloutier, "State-dependent Riccati equation techniques: An overview," in *Proceedings of the 1997 American Control Conference*, vol. 2, 1997, pp. 932-936.

[31] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton University Press, 1970.

[32] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons, 1994.

[33] R. E. Larson and J. L. Casti, *Principles of Dynamic Programming, Part I: Basic Analytic and Computational Methods*. New York: Marcel Dekker, 1978.

Biography

Adebayo Kayode James is an Associate Professor at Ekiti State University, Mathematics Department, Faculty of Physical Sciences. He completed his Ph. D. in Mathematics from Ekiti State University in 2016 and his Master of Mathematics in Control Theory from the same institution in 2010. In addition, he is a member of several Mathematics bodies in and outside Nigeria. He has participated in multiple international research collaboration projects in recent years. He currently serves as a reviewer for quite a lot of journals.

Alabi Taiye John is a Senior Lecturer at Adeyemi Federal University of Education, Ondo, Ondo State, in the Department of Mathematics. He completed his Ph. D. in Mathematics from Ekiti State University in 2020, and his Masters of Mathematics in Control Theory from Ekiti State University in the year 2010. He is a member of several Mathematics bodies with research covering Optimization, Control Theory, and applied Mathematics.

Omowaye Kehinde Solomon is an Assistant Lecturer in the Department of Mathematics, Faculty of Physical Sciences at Ekiti State University, Ado Ekiti, Ekiti State, Nigeria. He completed his M. Sc. in Mathematics from Ekiti State University in 2025, and his Ph. D. in view in Mathematics in Ekiti State University in Control Theory. His research interests include Optimization, Transportation Theory, Operations Research, Computational Mathematics, and Applied Mathematics. He is currently serving in several research groups in his institution. Presently, he is the working on Disruption in VRP. He is a member of some notable Mathematics organization.

Research Field

Adebayo Kayode James: Optimization, Control Theory, Mathematica Modelling.

Alabi Taiye John: Optimization; Numerical Analysis, Mathematical Modelling.

Omowaye Kehinde Solomon: Optimization; Mathematical Modelling, Operations Research, Computational mathematics, and Applied mathematics.