Direct collocation을 활용한 최적제어

동적 시스템을 제어하기 위해 레퍼런스를 생성하는 부분은 최적제어를 수행하는 데 있어서 필수적인 요소이다. 본 기고문에서는 다양한 구속조건을 가지는 동적 시스템의 최적제어와 최적의 궤적을 구하기 위해 Direct collocation 방법을 리뷰한다. 목적함수, 연속동역학, 이산동역학, 그리고 기타 다양한 구속조건들을 전사(Transcription)의 과정을 배우고, 이를 바탕으로 원래의 무한차원 최적제어 문제를 비선형 계획법의 문제로 전환한 후, 일반적인 비선형 프로그래밍 Solver를 활용하여 근사해법을 찾는다. 하이브리드 시스템(이족보행로봇)에 대해서 Direct collocation 방법을 적용해볼 것이다. 또한, Direct collocation 외의 다른 유사한 방법들에 대해서 간단히 알아볼 것이다.

허필원(광주과학기술원 기계공학부) pilwonhur@gist.ac.kr, http://hurgroup.net

Ⅰ. 서론

본 기고문을 읽는 독자는 자신이 가지고 있는 시스템에 대한 최적제어나 최적의 궤적을 생성하는 것에 관심이 있을 것이다. 주어진 시스템에 일련의 제어신호를 입력으로 가하면, 그 시스템은 특정한 궤적을 따라 행동한다. 만약 이러한 제어신호와 궤적이 어떤 목적함수를 최소화한다면, 우리는 그 제어신호를 최적제어 신호라고 하고, 그 결과에 따라 발생하는 궤적은 최적의 궤적이라고 한다. 만약 같은 최적제어 신호가 초기조건의 변화 등과 같은 이유로 해서 원래의 최적 궤적과 다른 궤적을 초래하게 한다면, 목적함수는 최소가 될 수 없다. 최적제어나 최적의 궤적은 시스템의 동역학적 특성을 반드시 반영해야만 한다. 예를 들어, 이상적이며 직관적인 예로 사람의 보행을 생각해 보자. 사람은 보행을 하기 위해서 한쪽 다리를 지지대로 삼아 몸을 중력에 맡겨 몸의 무게중심을 이동시킨다. 이 과정에서 위치에너지가 감소한다. 다른 의미로는, 역진자의 움직임과 비슷하게 몸의 무게중심이 불안정하게 움직인다는 의미다. 순간적인 불안정성을 활용하여 몸의 무게중심을 이동시킨 후, 다시 반대쪽 발로 안정성을 확보하는 방법은 획기적이고 에너지 측면에서 효율적인 방법이지만, 제어의 관점에서는 그렇게 직관적이지 않다. 원론적인 이야기를 하자면, 사람의 중추신경계는 운동을 학습하는 과정을 통해서 몸의 생체역학적 특성을 최대한 활용하여 운동을 제어하는 법을 습득한다(1).

본 기고문에서 저자는 주어진 시스템의 동역학적(Dynamic) 특성을 고려하고, 복잡하고 다양한 구속조건(Constraints)을 만족시키면서, 원하는 목적함수(Objective function or cost function)를 수치적으로 최소화하는 방법 중 하나인 Direct collocation 방법을 최대한 쉽고 직관적으로 설명하고자 한다. 그리고 그 결과물로 최적제어(Optimal control)와 최적의 궤적(Trajectory)을 얻게될 것이다. 독자들이 이미 알고 있듯이, 동적인 시스템(Dynamic systems)의 움직임을 수학적으로 표현하면 미분방정식(Differential equations)으로 나타난다는 것을 알 것이다. 연속적인 시스템의 변형이나 움직임을 표현하기 위한 편미분방정식(Partial differential equation, PDE)이나 이를 단순화하여 몇 개의 집중된 시스템(Lumped systems)의 움직임으로 표현하는 상미분방정식(Ordinary differential equation, ODE)이 그 예이다. 참고로, 정적인 시스템(Static systems)의 움직임은 대수방정식(Algebraic equations)으로 표현된다. 최근 들어 소프트 로봇(Soft robotics)에 대한 연구가 활발히 진행되면서 PDE로 로봇의 움직임을 표현하고 제어하려는 시도가 많이 일어나고 있지만, 무한차원(Infinite dimensional)의 시스템을 유한한 갯수의 센서와 제어 신호를 통해서 완벽하게 관측 및 제어할 수 없다. 궁극적으로는, 정밀한 제어를 수

행하기 위해서 PDE로 표현된 시스템을 제어하는 방향으로 연구가 진행되어야 하겠지만, 현 상황에서는 ODE로 표현된 단순하고 근사화된 시스템의 제어가 훨씬 더 왕성하게 진행되고 있다. 이는 상당히 많은 공학분야에서 시스템을 ODE로 표현하는 것이 효율적일 뿐 아니라, 정밀도의 차이가 유의미하지 않은 경우가 많기 때문이다. 그러므로, 본 기고문에서는 ODE로 표현된 시스템에 대해서만 다루기로 하겠다.

앞에서 간접적으로 표현하였듯이, 동적인 시스템의 최적제어 및 최적의 궤적을 구하는 방법은 다양하다. 그 대표적인 방법들로는 Pontryagin의 Minimal principle (또는 Maximal principle), Bellman (그리고, Hamilton, Jacobi)의 Dynamic programming, Single shooting, Multi shooting, Operator method (Banach 공간) 등이 있다(2). 본 기고문의 마지막에 레퍼런스와 함께 이들에 대한 간략한 언급을 하였으므로, 관심이 있는 독자들은 참고하길 바란다. 설명을 돕기 위해서 2차원 공간에서 7개의 링크를 가지는 이족보행로봇의 보행을 예로 들 것이다. 본 기고문의 구성은 다음과 같다. II절에서 본 기고문에서 예로 들 이족보행 시스템에 대해서 언급하고, 특히 하이브리드 시스템의 특징에 대해서 간략히 살펴볼 것이다. III절에서 최적제어 및 최적 궤적을 구하기 위한 Direct collocation 방법에 대해서 구체적으로 알아본다. IV절에서는 7개의 링크를 가지는 이족보행시스템의 멀티컨택 보행에 대한 시뮬레이션을 수행하고, 그 결과에 대해서 살펴볼 것이다. 마지막으로, V절에서는 수치적으로 최적제어를 수행하기 위해서 생각해 보아야할 부분들에 대한 고찰에 대해서 언급하고, 결론을 내고자 한다. 본 기고문을 준비하면서 주로 참고했던 논문과 책은 다음과 같다(3-6).

II. 하이브리드 시스템: 이족보행

이족보행(Bipedal locomotion)은 하이브리드 시스템(Hybrid systems)으로 묘사할 수 있다. 하이브리드 시스템은 여러가지 동적인 시스템을 동시에 포함하는 시스템을 말한다. 예를 들어, 이족보행로봇은 지면과 로봇의 접촉상태에 따라 몇가지 연속동역학(Continuous dynamics)과 이산동역학(Discrete dynamics)를 가진다(그림 1). 이족보행로봇이 가지는 동역학 모델은 지면으로 부터 오는 외부힘, 구동기(Actuators)로 부터 오는 관절힘(Joint torques), 그리고 중력에 의한 무게(Weight)에 의해 운동이 결정되는 연속동역학 모델이다(식 1). 지면과의 접촉상태가 변하는 과정에는 이산사건(Discrete event)이 존재한다. 즉, 하나의 이산사건이 발생하면 하나의 연속동역학은 다른 연속동역학으로 변경된다. 일반적으로, 사람의 보행에서 관찰할 수 있는 이산사건은 Heel strike, Foot drop, Heel off, Toe off 등이 있다. 그러므로, 여러 개의 이산사건과 여러 개의 연속동역학을 가지는 이족보행로봇은 하이브리드 시스템이다. 참고로, 이족보행로봇의 이산동역학은 각 보행 주기마다 특정 관절각의 변화를 나타내는 Poincaré map 을 생각할 수 있으나, 이번 분석에서는 제외하도록 한다. 대신, 각 이산사건의 발생시 고려해야 하는 충격(Impact)과 관련된 충격동역학(Impact dynamics)은 이산동역학의 일부분으로 포함하고자 한다.

a. 도메인과 가드. 도메인(Domain)은 특정한 접촉상태가 유지된 상태에서 하나의 연속동역학으로 물체의 움직임을 완전히 묘사할 수 있는 영역을 의미한다. 즉, 하나의 도메인 안에서는 운동방정식(식 1)의 구체적인 형태가 변형되지 않아야 한다. 예를 들어 그림 1에서 각 원은 하나의 도메인을 나타

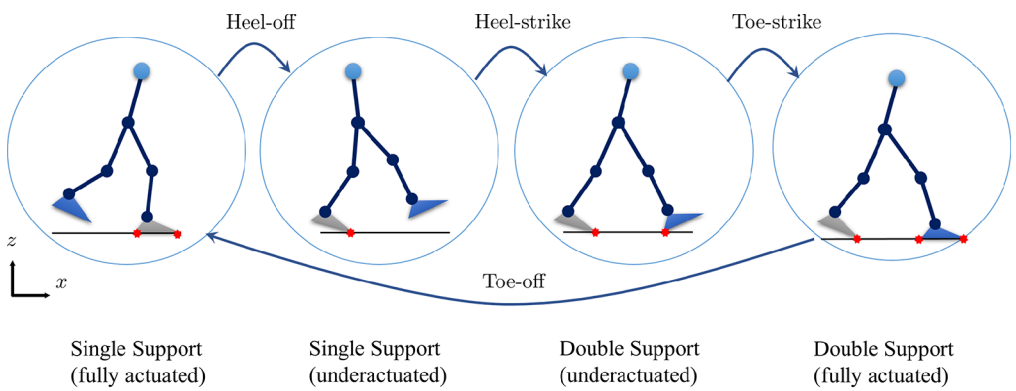

그림 1. 사람보행의 컨택순서.

낸다. 그러므로, 이족보행의 경우에는 도메인을 보행단계(Walking phase)라고 명명할 수도 있다. 하나의 도메인에서 다음 도메인으로 전환(Transition)을 하기 위해서는 특정 조건의 성립이 중요하다. 이 조건을 가드(Guard)라고 한다. 하나의 도메인 안에서 연속동역학에 의거해서 상태의 전이(State transition)가 일어나다가 특정 가드 조건을 만족시키는 순간, 도메인의 전환이 일어난다(Triggering).

b. 연속동역학. 일반적인 강체모델(Rigid body model)은 다음과 같은 운동방정식(식 1)으로 그 운동을 묘사할 수 있다.

$$M(\mathbf{q})\ddot{\mathbf{q}}+C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+G(\mathbf{q})=B\mathbf{u}+J^T\boldsymbol{\lambda} \quad (1)$$

여기서 $\mathbf{q}$는 로봇의 형태(Configuration)를 나타내는 일반화 좌표(Generalized coordinates)(그림 2), $M(\mathbf{q})$는 전체 로봇시스템의 관성행렬(Inertia matrix), $C(\mathbf{q},\dot{\mathbf{q}})$는 코리올리 행렬(Coriolis matrix), $G(\mathbf{q})$는 로봇의 무게를 나타내는 중력벡터(Gravity vector), J는 접촉지점($\phi(\mathbf{q})$)에 대한 자코비언 행렬(Jacobian matrix, $J \triangleq \partial\phi/\partial\mathbf{q}$), B는 토크 분배 행렬(Torque distribution matrix), $\mathbf{u}$ 는 제어입력, 그리고 $\boldsymbol{\lambda}$는 접촉힘을 나타낸다.

c. 이산동역학. 새로운 접촉이 발생하거나 기존의 접촉 상태가 무너지는 등, 접촉 상태에 변화가 생기면 시스템의 상태는 이산적인 변화를 겪게된다. 예를 들어, Heel strike 로 인한 충격이 발생하면 관절속도의 불연속적인 변화가 동반된다. 충격 전후로의 속도변화는 아래의 식에 의해서 기술된다.

$$\begin{bmatrix} M(\mathbf{q}) & -J^T \\ J & 0 \end{bmatrix}\begin{bmatrix} \dot{\mathbf{q}}^+ \\ \delta\mathbf{F}_{impact} \end{bmatrix}=\begin{bmatrix} M(\mathbf{q}^-)\dot{\mathbf{q}}^- \\ 0 \end{bmatrix} \quad (2)$$

여기서 위첨자 "+"는 충격 직후의 상태를 나타내고, 위첨자 "−"는 충격 직전의 상태를 나타낸다. $M(\cdot)$은 관성행렬(참고로, 형태의 연속성을 가정하면 $M(\mathbf{q}^-)=M(\mathbf{q}^+)$이고, J는 충격이 발생한 지점에서의 자코비언 행렬이다. 마지막으로 $\delta\mathbf{F}_{impact}$는 충격으로 인한 충격량(Impulse)을 나타낸다. 식 (2)는 식1의 양변에 아주 작은 시간인 Δt를 곱한 후, 비충격력(Nonimpulsive force)의 항을 제거하고 형태의 연속성을 가정($\mathbf{q}^-=\mathbf{q}^+$)하면 얻을 수 있으며, 이를 충격방정식(Impact dynamics)라고 부르기도 한다. 식 (2)를 살펴보면 두개의 행으로 구성되어 있는데, 첫 번째 행은 아주 짧은 순간(Infinitesimally short time)동안 충격으로 인한 일반화된 운동량(Generalized momentum)을 의미하고, 두 번째 행은 충격 직후 접촉점의 속도가 0이 됨을 의미한다.

d. 사람 보행의 접촉 순서. 건강한 사람의 정상상태(Steady state)에서의 보행은 여러 개의 도메인을 일정한 순서를 가지고 주기적으로 이동하는 패턴을 보인다. 그러므로, 이족보행 로봇의 궤적(Trajectory)을 생성하기 위해서는 미리 정해진(Predetermined) 접촉 순서(Contact sequence)를 따르면서 보행하도록 구속하는 것이 좋다. 그림 1은 사람의 보행에서 접촉 순서를 도메인 별로 구분해 놓은 것이다.

e. 구속조건을 갖는 연속동역학. 식 (1)은 일반적인 연속동역학 방정식의 표현이다. 각 도메인에서의 접촉 조건(그림 1)을 만족시키기 위한 구속힘(Constraining forces)은 접촉점의 갯수와 위치에 따라 다르다. 예를 들어, 그림 3는 각 도메인에서 사람의 보행에 의한 접촉점에서의 구속힘을 어떻게 정의할 수 있는지 보여준다. 또한, 각 접촉점의 속도는 0 이므로, $J\dot{\mathbf{q}}\equiv\mathbf{0}$이다. 그러므로, 식1과 $J\dot{\mathbf{q}}\equiv\mathbf{0}$의 미분값을 함께 표시하면, 아래와 같은 구속조건을 갖는 연속동역학의 방정식을 얻을 수 있다.

$$M(\mathbf{q})\ddot{\mathbf{q}}+C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+G(\mathbf{q})=B\mathbf{u}+J^T\boldsymbol{\lambda} \quad (3)$$

$$\dot{J}\ \dot{\mathbf{q}}+J\ddot{\mathbf{q}}=0 \quad (4)$$

여기서 J는 각 도메인에 포함된 모든 접촉점들에 대한 자

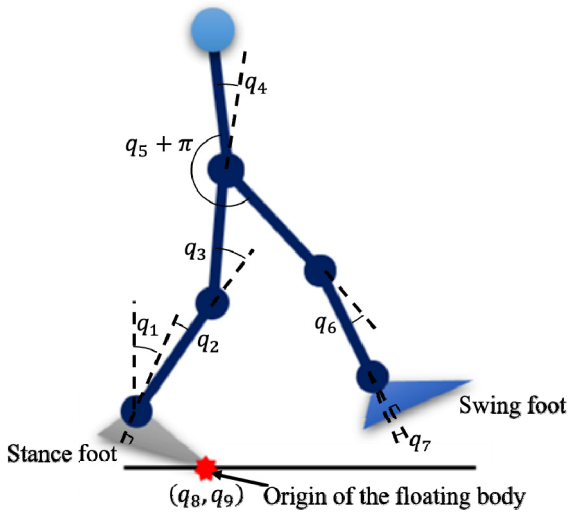

그림 2. 이족보행로봇 모델 및 일반화좌표의 정의. 2차원 공간에서의 7링크 모델. 7개의 관절각과 2개의 원점 좌표를 가지고 있는 9자유도 floating 모델이고, 모델의 원점은 Stance foot의 발가락 끝에 둠. Stance foot은 회색으로, Swing foot은 파란색으로 표시함.

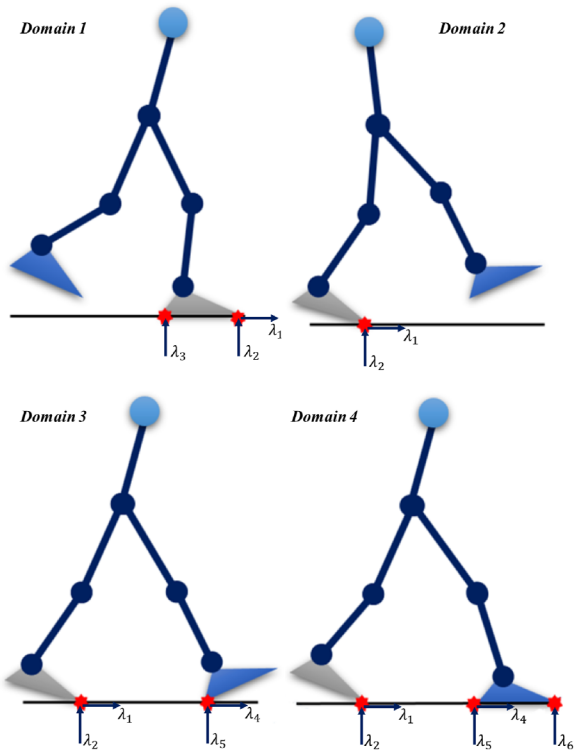

그림 3. 각 도메인별 사람 보행의 컨택점에서의 구속힘.

코비언 행렬을 수직으로 모아둔(Vertically concatenated) 블록 자코비언 행렬(Block Jacobian matrix)이고, λ는 각 도메인에 포함된 모든 접촉점들에 작용하는 구속힘들의 칼럼벡터(Column vector) $\lambda = [\lambda_1, \lambda_2, ...]^T$이다.

f. 추가적인 구속조건. 식 (2-4)는 연속동역학과 이산동역학에 대한 구속조건이다. 이외에도 주기적인 이족보행을 위해 필요한 구속조건은 다음과 같다. 양(+)의 수직지면반력(Vertical ground reaction force), 쿨롱 마찰모델(Coulomb friction model)에 의한 마찰콘(Friction cone) 조건, 압력중심점(Center of pressure)의 제약조건, 보행의 시작과 끝 시점에서의 연속조건(Periodicity), 무릎관절의 과신전(Hyperextension) 방지, 몸통(Torso)의 직립조건, 발의 최소높이 조건(Foot clearance) 등이 필요하다. 구체적인 수치적 표현 및 기타 구속조건은 다음의 레퍼런스(5,6)를 참고하기 바란다.

g. 목적함수. 앞에서와 같이 로봇의 동역학적 운동방정식과 구속조건들을 구하면, 목적함수(Objective function or cost function)를 정의해야만 최적제어의 문제를 풀 수 있다. 원하는 임무(Task)를 수행하기 위해서 적당한 목적함수를 정의하는 것은 설계자의 몫이다. 예를 들어, 제어신호의 노력치(Effort)(식 5), 절대 일(Absolute work)(식 6), 운동의 편안함을 나타내는 저크(Jerk)(식 7), 수행 시간(Time)(식 8) 등이 대표적인 목적함수이고, 아래와 같은 수식으로 표현될 수 있다.

$$\min_{u(t),q(t),\dot{q}(t)} \int_0^T u^2(\tau) d\tau \tag{5}$$

$$\min_{u(t),q(t),\dot{q}(t)} \int_0^T |u(\tau)\dot{q}(\tau)| d\tau \tag{6}$$

$$\min_{u(t),q(t),\dot{q}(t)} \int_0^T |\ddot{q}(\tau)|^2 d\tau \tag{7}$$

$$\min_{u(t),q(t),\dot{q}(t)} \int_0^T 1 d\tau \tag{8}$$

h. 가속도에 대한 고려사항. 로봇의 동역학적 운동방정식을 수치적으로 적분을 수행하기 위해서는 1차미분방정식의 형태(First order form)로 표현하는 것이 편리하다. 상태변수(State variable)를 $\mathbf{x} \equiv [\mathbf{q}, \dot{\mathbf{q}}]^T$로 정의하면, 다음과 같은 1차미분방정식 형태의 시스템 동역학 식을 얻을 수 있다.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \dot{\mathbf{q}} \\ M^{-1}(-C\dot{\mathbf{q}} - G + B\mathbf{u} + J^T\boldsymbol{\lambda}) \end{bmatrix} \tag{9}$$

여기서 $\ddot{\mathbf{q}}$은 식 (1)과 식 (9)에서 알 수 있듯이 M^{-1}를 포함하고 있다. 최적화를 수행하는 과정에서 $\ddot{\mathbf{q}}$의 미분(Gradient)이 필수적이지만, M^{-1}의 복잡도로 인해서 해석적(Analytically or symbolically) 미분이 매우 힘들다(Intractable). 물론, 수치적으로 M^{-1}를 계산하거나 $\ddot{\mathbf{q}}$를 미분하는 것은 가능하나, 계산 오차가 누적될 수 있고, $\ddot{\mathbf{q}}$의 값을 평가(Function evaluation)해야 하는 횟수가 크게 증가하므로, 계산의 효율성이 떨어진다. 이러한 문제를 해결하기 위해서 $\ddot{\mathbf{q}}$를 변수(Decision variable)로 정의하면 $\ddot{\mathbf{q}}$를 직접적으로 계산할 필요가 없이, 식1을 직접적인 구속조건으로 사용할 수 있기 때문에, M^{-1}를 계산하지 않아도 된다(식 10). 이로 인해, 최적화의 과정에서 식 (1)의 미분만 수행하면 되므로, M^{-1}을 포함하는 식 (9)의 미분보다 훨씬 더 효율적이다. 또한, 자동미분(Automatic differentiation)을 수행할 경우 오차 없이 식1의 수치미분을 계산할 수 있을 뿐 아니라, 식 (1)의 평가(Function evaluation) 횟수도 증가하지 않으므로, 효율적인 연산이 가능하다.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix}$$
$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = B\mathbf{u} + J^T\boldsymbol{\lambda} \quad (10)$$

III. Direct collocation

a. 표기법. 본 기고문에서 Direct Collocation 방법을 효율적으로 설명하기 위해서 다음과 같은 수학적 표기법(Notation)을 사용할 예정이다.

표기법	설명
t_k	time at knot point k
N	number of trajectory (spline) segments
$h_k = t_{k+1} - t_k$	duration of spline segment k
$\mathbf{x}_k = \mathbf{x}(t_k)$	state at knot point k
$\mathbf{u}_k = \mathbf{u}(t_k)$	control at knot point k
$w_k = w(\mathbf{x}_k,\mathbf{u}_k)$	integrand of objective function at knot point k
$\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)$	system dynamics at knot point k
$\dot{\mathbf{q}} = \frac{d}{dt}\mathbf{q}$, $\ddot{\mathbf{q}} = \frac{d^2}{dt^2}\mathbf{q}$	first and second time-derivative of $\mathbf{q}$

b. 전사(Transcription). 앞 절에서와 같이 로봇의 동역학적 운동방정식과 구속조건들을 구하면, 목적함수를 정하여 일반적인 최적화 문제로 나타낼 수 있다. 아래의 식 (11)은 그 예시로써, 이족보행의 최적 제어와 최적의 궤적을 구하기 위한 문제를 수치적으로 표현한 것이다.

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \int_0^T w(\mathbf{x}(\tau),\mathbf{u}(\tau))d\tau$$
$$\text{s.t.} \quad \text{equality constraints} \quad (11)$$
$$\text{inequality constraints}$$

여기서 등식 구속조건(equality constraints)은 시스템 동역학(System dynamics)인 $\dot{\mathbf{x}}(t) = \mathbf{f}(t,\mathbf{x}(t),\mathbf{u}(t))$, 충격 동역학(Impact dynamics)(식 2), 연속조건(Periodicity) 등이 있다. 부등식 구속조건(Inequality constraints)은 수직지면반력, 마찰콘 조건, 압력중심점의 조건, 상태변수 조건, 토크의 조건등이 있다. 연속동역학 및 다양한 구속조건을 만족시키면서 목적함수를 최소화하기 위한 방법은 다양하게 존재한다(see e). 본 기고문의 주제인 Direct Collocation 방법이 바로 이중에 하나이다. 본 방법은 전사(Transcription)이라는 과정을 통해서 이산화(Discretization)를 수행한다. 이러한 전사과정을 거치면 원래의 최적화 문제(식11)는 비선형 계획법(Nonlinear programming)으로 변경된다.

전사의 구체적인 방법을 소개하기 전에, Direct collocation 방법의 개념을 조금 더 구체적으로 설명하고자 한다. 식 (11)에서 목적함수는 일반적으로 적분의 형태로 나타나 있고, 이를 이산화하여 근사화하기 위해서는 구적법(Quadrature)을 사용할 수 있다. 또한, 식 (9)나 식 (10)의 시스템 동역학 방정식을 이산화하기 위해서 전체 시간을 N등분한 후, 각 구간(Time interval)에 대해서 다음과 같이 근사화할 수 있다.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$$
$$\downarrow$$
$$\int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x},\mathbf{u})dt$$
$$\downarrow \quad (12)$$
$$\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x},\mathbf{u})dt$$
$$\approx \sum_{i=0}^{n} A_i(t_k,t_{k+1})\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i)dt$$

여기서 $\mathbf{x}_i = \mathbf{x}(t_i), \mathbf{u}_i = \mathbf{u}(t_i), t_i \in [t_k, t_{k+1}]$이고, $A_i(t_k,t_{k+1})$는 해당 구적법(Quadrature)에 대한 t_i에서의 계수(Coefficient)이다. 전체 시간 T를 N등분했을 때, $t_k\,(\equiv \frac{T}{N}k, k = \{0,2,...,N\})$를 Collocation point, 노드(Node), 또는 매듭점(Knot point) 등으로 부른다. 참고로, N은 통상적으로 설계자가 지정하고, T는 문제에서 주어지거나 혹은 설계변수(Decision variable)로 둔다.

$\mathbf{x}$와 $\mathbf{u}$는 최적화 과정을 통해서 구해야 하는 최적궤적 및 최적제어 신호이다. 문제를 설정하는 과정인 식(11)이나 식(12)에서

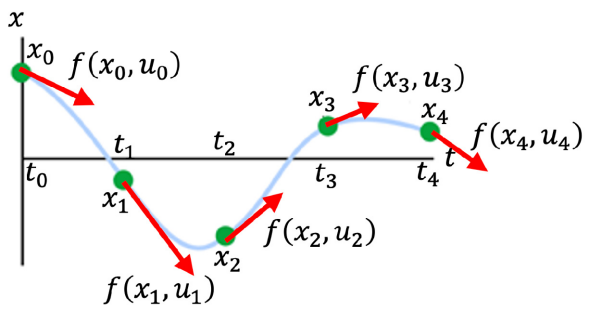

그림 4. Direct collocation 개념. 최적의 연속 신호인 x와 각 Collocation point t_k에서 이산화된 x_k와 u_k (생략) 그리고 시스템 동역학 방정식으로부터 알 수 있는 기울기가 서로 모순이 발생하기 않도록 최적화 문제를 설정해야 함.

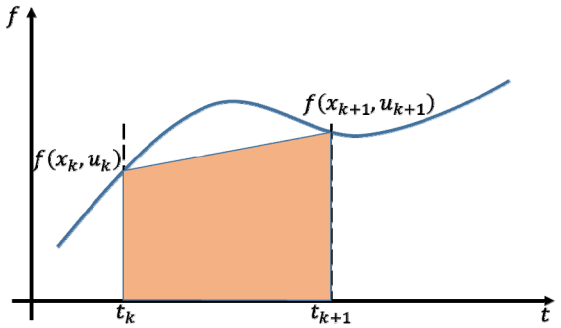

그림 5. Trapezoidal quadrature 방법의 개념도.

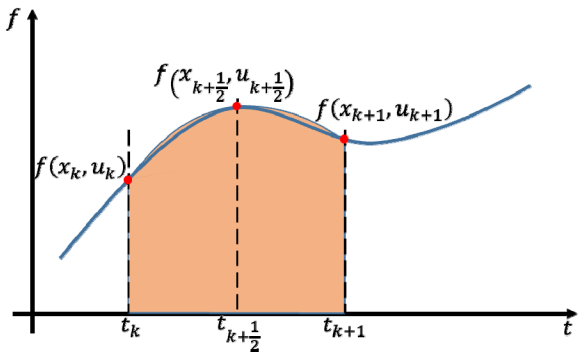

그림 6. Hermite-Simpson quadrature 방법의 개념도.

는 정확한 x(최적 궤적)과 u(최적 제어)의 해를 알 수는 없다. 하지만 그림4에서 보이는 것처럼 x_k, u_k, 그리고 기울기(Gradient) $f(x_k, u_k)$ 가 모두 모순이 없도록(Compatible) 최적화 문제를 설정해 주어야 한다. 적분이나 미분의 형태가 아닌 대수방정식의 형태를 가지는 구속조건들은 전사의 과정이 단순하다($\mathbf{x}\rightarrow\mathbf{x}_k$, $\mathbf{u}\rightarrow\mathbf{u}_k$). 반면, 적분의 형태로 표현되는 구속조건인 시스템 동역학 방정식(식 12)이나 목적함수는 적절한 구적법(Quadrature)를 통하여 대수방정식으로 근사화해야 한다. 본 기고문에서는 전사의 방법 중 대표적인 두 가지 방법을 설명하고자 한다.

c. 사다리꼴 방법(Trapezoidal method). 전사(Transcription)의 첫 번째 방법은 사다리꼴 방법(Trapezoidal method)이다. 그림 5에서 볼 수 있듯이, 사다리꼴을 통해서 식 (13)의 적분을 근사화한다.

$$\begin{aligned}\mathbf{x}(t_{k+1})-\mathbf{x}(t_k) &= \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x},\mathbf{u})dt \\ &\approx \sum_{i=0}^{n} A_i(t_k,t_{k+1})\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i)dt \\ &= \frac{1}{2}h(f(\mathbf{x}_k,\mathbf{u}_k)+f(\mathbf{x}_{k+1},\mathbf{u}_{k+1}))\end{aligned} \tag{13}$$

여기서 등간격의 시간구간을 가정하여 $h=t_{k+1}-t_k=\frac{T}{N}$ 이다. 사다리꼴 방법을 사용하면 시스템 동역학 방정식, 궤적, 그리고 제어 신호를 선형 스플라인(Linear splines)으로 근사화하는 것과 같다. 이 방법은 계산이 단순하고 구현이 쉬울 뿐 아니라 직사각형 형태의 구적법보다 더 높은 정확도를 보이므로, 많이 쓰이는 방법이다. 하지만, 그림 5에서 확인할 수 있듯이, 시스템 동역학 방정식 $\mathbf{f}(\mathbf{x},\mathbf{u})$ 가 다변수 1차방정식에서 벗어날수록 실제 적분과 사다리꼴의 면적은 차이가 커진다. 이러한 문제를 해결하기 위하여 시간구간을 절반으로 줄여서 사용하기도 한다. 하지만, 그림 5에서 볼 수 있듯이, 사다리꼴 방법은 계산의 편리성에도 불구하고 적분오차가 크게 발생할 수 있는 단점이 있다. 이를 해결하기 위해서 허마이트-심슨 방법을 소개한다.

d. 허마이트-심슨 방법(Hermite-Simpson method). 사다리꼴 방법의 오차 문제를 더 향상시킬 수 있는 구적법의 방법으로 허마이트-심슨 방법(Hermite-Simpson method)이 있다. 사다리꼴 방법은 시간구간의 양 끝점만을 사용하여 선형 스플라인으로 함수를 근사화 했지만, 허마이트-심슨 방법은, 시간구간(Time interval)의 양 끝점(Collocation point) 뿐만 아니라 Collocation point의 중간지점(Mid point, $t_{k+\frac{1}{2}}$)을 포함한 세점을 보간(Interpolate)하는 다항식(Polynomial)으로 근사화한다(그림6 참조). 심슨의 보간공식(Simpson's rule)을 적용하거나 혹은 라그랑지 보간(Lagrange interpolation)방식을 활용하여 직접 공식을 구하면 식 (14)와 같이 적분을 근사화한다.

$$\begin{aligned}\mathbf{x}(t_{k+1})-\mathbf{x}(t_k) &= \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x},\mathbf{u})dt \\ &\approx \sum_{i=0}^{n} A_i(t_k,t_{k+1})\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i)dt \\ &= \frac{1}{6}h\left(f(\mathbf{x}_k,\mathbf{u}_k)+4f\left(\mathbf{x}_{k+\frac{1}{2}},\mathbf{u}_{k+\frac{1}{2}}\right)\right. \\ &\quad \left.+f(\mathbf{x}_{k+1},\mathbf{u}_{k+1})\right)\end{aligned} \tag{14}$$

여기서 등간격의 시간구간을 가정하면 $h=t_{k+1}-t_k=\frac{T}{N}$ 이다. 허마이트-심슨 방법을 사용하면 시스템 동역학 방정식과 제어 신호는 2차 스플라인(Quadratic splines)으로 근사화하고, 궤적은 3차 스플라인(Cubic splines)으로 근사화하는 것과 같다. 그런데, 식 (14)에서 볼 수 있듯이, 중간지점($t_{k+\frac{1}{2}}$)에 대한 궤적과 제

어의 정보가 사용된다. $\mathbf{u}_{k+\frac{1}{2}}$는 제어신호이므로 임의로 생성이 가능하지만, $\mathbf{x}_{k+\frac{1}{2}}$는 동역학적 구속조건을 반드시 만족시켜야 하므로, 임의 생성이 불가능하다. 우리는 식 (14)에서 시스템 동역학 방정식을 $[t_k, t_{k+1}]$ 구간에 대해서 적분하였다. $[t_k, t_{k+\frac{1}{2}}]$ 구간에 대해서 적분을 수행하면 $\mathbf{x}_{k+\frac{1}{2}} - \mathbf{x}_k$의 표현을 얻을 수 있고, 이것과 식14의 $\mathbf{x}_{k+1} - \mathbf{x}_k$와 연립방정식을 풀면 다음의 식 (15)을 얻을 수 있다.

$$\mathbf{x}_{k+\frac{1}{2}} = \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{h}{8}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})) \quad (15)$$

그러므로, 식 (14)와 식 (15)를 동시에 적용하면 중간지점을 포함한 모든 Collocation point에 대해서 궤적의 정보를 구할 수 있게 된다. 또한, 식 (15)을 식 (14)에 대입하여 좀 더 압축된 형태(Compressed form)로도 최적화를 수행할 수 있다. 압축된 형태와 분리된 형태(Separate form)는 같은 결과를 주지만, 연산과 결과의 활용 면에서는 조금 차이가 있으므로, 필요에 맞게 사용하길 추천한다. 더 궁금한 독자는 (4)를 참고하면 도움이 될 것이다.

허마이트-심슨 방법은 오차의 측면에서 사다리꼴 방법보다 더 높은 정확도를 주지만, 더 많은 계산량을 필요로 한다. 앞에서도 언급을 하였지만, 사다리꼴 방법의 시간구간의 크기(즉, h)를 절반으로 줄여서 사용하면 허마이트-심슨 방법과 비슷하게 정확도가 높아지지만, 허마이트-심슨 방법은 리차드슨의 외삽법(Richardson's extrapolation)을 통해 오차를 최소화한 것이기 때문에, 시간구간을 절반으로 줄인 사다리꼴 방법보다도 더 효율적인 방법이다. 물론 허마이트-심슨 방법을 코드로 구현하는 것은 사다리꼴 방법보다 조금 더 까다롭기는 하지만, 정확도의 측면에서는 훨씬 더 좋은 결과를 주므로, 허마이트-심슨 방법을 사용하는 것이 더 유리하다. 만약 계산량이 부담스러운 경우, 사다리꼴 방법으로 최적제어와 최적궤적을 찾은 후, 이것을 허마이트-심슨 방법의 초기값으로 사용하면 좋은 성능을 보장할 수 있을 것이다.

e. 비선형 계획. 식 (16)의 최적제어 문제를 전사(Transcription)의 과정을 거치게 되면 다음과 같은 비선형 계획(Nonlinear programming)의 문제로 전환된다. 이러한 문제를 풀기 위해서 고려해야 할 사항들이 몇 가지 있다. 먼저 Solver의 선택이 중요하다. 대표적인 비선형 계획법 Solver는 IPOPT(7, 8), SNOPT(9), FMINCON(10) 등이 있다. 대부분의 비선형 계획법 Solver는 목적함수와 구속조건식들이 일관적(Consistent)이어야 한다. 즉, 매번 함수호출(Function calls)과 평가(Function evaluation)에 대해서 같은 순차적 연산을 수행해야 한다(4). 쉽게 말하면, 논리적인 분기(Logical branches)가 없어야 하고, 입력값에 대해서 출력이 충분히 부드럽게 변해야 한다. 대표적인 비일관적(Inconsistent) 함수들은 abs(), min(), max() 이다. 이런 함수들을 목적함수나 구속조건식에서 사용하게 되면 최적의 해를 찾지 못 하거나, 일관적이지 못한 결과를 얻게 되는 경우가 많다. 다행히도, 이런 문제들을 해결할 수 있는 트릭이 존재한다. 예를 들어 abs()의 경우 slack 변수(Slack variable)를 도입하여 쉽게 구현이 가능하고, 뾰족한 부분을 부드럽게 근사화하는 함수의 구현을 통해서도 해결이 가능하다(4, 8).

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1}\sum_{i=0}^{n} A_i(t_k, t_{k+1}) w(\mathbf{x}_i, \mathbf{u}_i) \\ \text{s.t.} \quad & \text{equality constraints} \\ & \text{inequality constraints} \end{aligned} \quad (16)$$

목적함수나 구속조건에서 사용자 정의함수를 사용하는 경우, 내부에 동적으로 변하는 루프나 알고리즘이 있으면 비일관적인 상황이 될 수 있다. 수치적으로 해를 찾는 알고리즘(Root finding algorithms)을 포함하는 경우가 대표적인 예이다. 이런 경우, 내부적으로 수행해야 하는 알고리즘의 횟수(Iteration)를 고정하거나 스텝크기(Step size)를 고정함으로써 일부 해결이 가능하다. 룩업 테이블(Look Up Table)이나 데이터를 사용해야 하는 경우에는 미분이 가능하도록 3차 스플라인(Cubic Spline)과 같은 방법으로 보간함수(Interpolation Function)를 만들어 사용하는 것이 좋다. 이족보행과 같은 하이브리드 시스템의 경우 컨택 순서에 따라서 동역학적 운동방정식의 표현이 달라질 수 있다(그림 1,3). 이 경우, 모든 도메인에서 사용되는 방정식의 구조는 일치시키면서, 방정식의 내용은 각 도메인의 내용에 모순이 없도록 구속방정식을 일관적으로 표현하는 것이 중요하다. 컨택 순서를 명확히 아는 경우에는 컨택 순서를 지정하여서 코딩을 하는 것이 중요하다. 만약 컨택 순서를 사전에(a priori) 알지 못하는 경우 Complementarity constraint를 활용한 Through-contact 최적화의 기법을 적용하는 것이 효율적이다(5, 11).

Ⅳ. 실험결과 및 토의

앞에서 언급한 일곱 개의 링크와 아홉 개의 자유도를 가지는 2차원 공간에서의 이족보행로봇에 대한 시뮬레이션을 수행하였고, 그 결과를 살펴보고자 한다.

a. 목적함수. 관절토크의 제곱을 시간에 대해서 적분한 값을 목적함수로 사용하였고, 허마이트-심슨 방법으로 전사하였다.

$$J=\int_0^T\left(\sum_{i=1}^{7}u_i^2(\tau)\right)d\tau$$

$$\approx\sum_{i=1}^{7}\left(\sum_{k=0}^{N-1}\frac{1}{6}\left(u_i^2(t_k)+4u_i^2(t_{k+\frac{1}{2}})+u_i^2(t_{k+1})\right)\right) \quad (17)$$

b. 구속조건. 컨택 순서(Contact sequence), 초기조건(Initial condition) 등에 대해서 다음과 같이 설정하였다.

(1) 컨택순서. 컨택 순서는 그림 1에서 정의한 순서를 사용하였다. 건강한 사람의 보행 데이터로부터 각 도메인별로 소요되는 시간의 비율을 고려해 단순화하여 각 도메인별로 11개의 Collocation point를 사용하였고, 마지막 도메인은 3개의 Collocation point를 사용하였다. 그러므로, 중간지점(Mid point)을 모두 포함하면 처음 3개의 도메인에는 총 21개의 노드가 있고, 마지막 도메인은 5개의 노드가 있다.미리 Collocation point의 갯수를 지정해 줌으로써 최적화의 일관성을 유지할 수 있도록 하였다.

(2) 연속동역학. 식(1) 또는 식 (3)을 연속동역학의 구속조건으로 사용하였다. 상태변수(State variable)는 식 (10)과 같이 정의하였다. 그리고 관성행열의 역함수 $M^{-1}(\mathbf{q})$ 의 복잡성과 계산의 비효율성을 피하기 위해 관절가속도인 $\ddot{\mathbf{q}}$를 설계변수(Decision variable)로 설정하였고, 연속동역학의 속성을 보장하기 위해서 식 (1) 또는 식 (3)을 직접 구속조건에 추가하였다. 연속동역학의 방정식을 구하기 위한 모델링은 그림3에서 정의한 일반화좌표를 따랐으며, 키와 몸무게로 파라미터화되어 있는 Winter 모델(12)을 사용하였다. 동역학 방정식은 Mathematica(v12)의 HurTool box(v2.0.5)를 사용하여 Euler-Lagrange 형태의 수식을 Julia 파일로 저장하여 사용하였다. 참고로, HurToolbox는 저자가 개발한 Mathematica Tool box로, 간단한 모델링 정보의 입력을 통해 해석적(Analytic) 강체동역학 수식을 효율적으로 계산하고, Matlab, Julia, Python등의 인터페이스를 가능하게 해 준다. Newton-Euler, Euler-Largange, Hamiltonian 그리고 Kane의 방법을 선택할 수 있다.

(3) 충격 이벤트 및 이산동역학. 도메인과 도메인 사이를 이동하기 위해서는 이산이벤트가 발생해야만 한다(가드 조건). 도메인1에서 도메인2로 넘어가는 순간에서는 Stance Heel이 떨어지는 순간이므로, 이로 인해서 λ_3가 0이 된다. 하지만, 이 과정에서 충격(Impact)은 발생하지 않으므로, 형태(Configuration)과 관절속도는 연속을 유지한다. 도메인2에서 도메인3로 넘어가는 순간에서는 Swing Heel Strike가 발생하므로, 이로 인해서 $\lambda_4 \neq 0$ $\lambda_5 > 0$이 된다. 그 결과 발생한 충격(Impact)으로 인해, 다시 한 번 형태(Configuration)는 연속을 유지하지만, 관절속도가 불연속적으로 점프하는 지점이 발생한다. 충격 직후의 관절속도를 계산하기 위해서 식 (2)를 사용한다. 도메인3에서 도메인4로 넘어가는 순간에서는 Swing Toe Strike가 발생하므로, 이로 인해서 $\lambda_6 > 0$이 된다. 이 과정에서 발생하는 충격(Impact)으로 인해 형태(Configuration)는 연속을 유지하지만, 관절속도는 불연속적으로 점프한다. 충격 직후의 관절속도를 계산하기 위해서 식 (2)를 사용한다. 도메인4에서 도메인1로 넘어가는 순간에서는 Stance Toe가 떨어지는 순간이므로, 이로 인해서 $\lambda_1 = \lambda_2 = 0$이 된다. 하지만, 이 과정에서 충격(Impact)은 발생하지 않으므로, 형태(Configuration)과 관절속도는 연속을 유지한다. 그리고, 이 순간 Stance leg와 Swing leg의 변경이 일어나므로, 관절각과 관절속도의 연속성에 대한 구속조건을 기술할 때 반드시 상태변수의 리라벨링(Relabeling)을 수행해야 한다.

(4) 형태(Configuration). 형태를 결정짓는 구속조건들은 상당히 다양하면서도 구체적으로 설정될 필요가 있다. 그 이유는 설계자가 사람의 보행에 시각적으로 너무 익숙해져 있어서 조금이라도 보행패턴이 다르면 이상하게 느끼기 때문이기도 하며, 사람의 보행이 그만큼 최적의 궤적을 따르는 것이기도 하다. 형태와 관련된 대표적인 구속조건들을 나열해 보면 다음과 같다. 몸의 모든 부위는 반드시 땅 위에 있어야 한다. 몸의 각 링크들이 컨벡스(Convex)의 형태를 가진다고 하면, 발가락 끝(Toe), 힐(Heel), 발목(Ankle), 무릎(Knee), 힙(Hip)의 높이가 반드시 땅보다 위쪽에 존재하도록 구속조건을 추가할 필요가 있다. 머리, 팔, 몸통을 하나의 HAT 링크로 가정하면[1], HAT 링크는 직립(Upright)을 하면서 약간의 변화(예: $\pm 10^\circ$)만 허용하도록 구속조건을 추가해야 한다. 만약 HAT 링크의 직립조건을 추가하지 않을 경우, HAT 링크는 당연히 아래로 향하여 위치에너지에서 이득을 얻으려 할 것이다. 또한, 각 관절의 활동범위를 제한하여

[1] 이러한 모델을 HAT (Head, Arm, Torso) 모델이라고 부르고, Sagittal 평면에서의 보행을 잘 설명한다.

야 한다. 예를 들어, 무릎의 과신전(Hyperextension)을 방지하고, 힙과 발목의 관절각도를 적절하게 제한해야 한다. 특히 발목의 경우 아킬레스건과 같은 조직들이 스프링과 유사한 역할을 하기 때문에 관절각의 범위가 상당히 좁다. 특히 발등굽힘(Dorsiflexion)의 경우 10° 이상 움직이는 경우는 지극히 비정상적이다. 적당한 최소 보행속도와 최소 보폭을 지정해 주는 것도 합리적인 보행패턴을 얻는데 상당히 도움이 된다.

(5) 관절토크. 관절토크는 생리학적으로 힘을 생성하는데 있어 한계가 있으므로, 적당한 값으로 상한과 하한값을 정해준다. 미끄러짐을 허용하지 않기 위해서 지면반력벡터가 반드시 마찰콘(Friction cone) 내부에 존재하도록 설정한다. 마찰콘 조건은 관절토크에 대해 간접적으로 구속조건을 부과한다. 또한, 도메인1과 도메인4에서는 Full actuation이 보장되어야 하므로(그림 1), 압력중심점(Center of pressure)가 반드시 발바닥(Base of support 또는 Support polygon) 내부에 존재하도록 구속조건을 부과한다.

(6) 미분 및 자코비언. 최적화를 수행하기 위해서 미분(Gradient) 및 자코비언(Jacobian) 정보가 필요하다. 해석적으로 미분을 수행한 후, Gradient와 Jacobian을 심볼릭하게 사용할 수도 있고, 수치 미분을 통하여 Gradient와 Jacobian을 근사화하여 사용할 수도 있다. 여기서는 따로 Gradient와Jacobian 식을 저장하지 않고, 자동미분(Automatic differentiation)을 사용하여 정확도와 연산의 효율을 높이도록 하였다.

(7) 초기조건. 초기조건은 Julia의 rand() 함수를 사용하여 랜덤으로 주었다.

c. 최적화 및 시뮬레이션 환경. 프로그래밍 언어는 Julia (v1.6.7)을 사용하였고, 최적화 문제를 기술하기 위한 모델링 언어로는 JuMP (Julia for Mathematical Programming)를 사용하였다. 비선형 프로그래밍 Solver는 IPOPT (with lin-ear solver MUMPS 5.4.1)를 사용하였다. Macbook Pro M1 Max 칩 환경에서 최적화 및 시뮬레이션을 수행하였다.

Table 1. 최적화 수행 결과

항목	결과
최적화 시간	593초
목적함수 값	$9.653e5$
보폭	0.37m
보행속도	0.29초

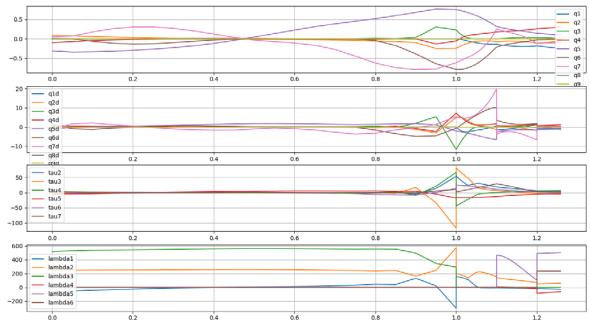

그림 7. 이족보행로봇의 최적화결과. 관절각($\mathbf{q}$), 관절 속도($\dot{\mathbf{q}}$), 관절토크($\mathbf{u}$), and 지면반력($\boldsymbol{\lambda}$) 결과가 주어져 있음.

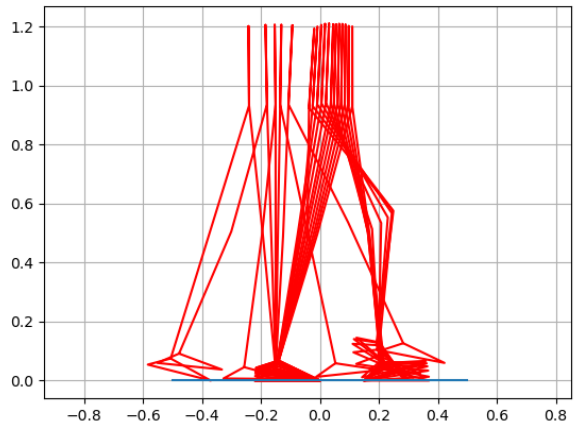

그림 8. 이족보행로봇의 최적화결과. 보행의 궤적을 효과적으로 볼수 있도록 Walking tile을 겹쳐서(Overlay) 보여줌.

d. 결과. 그림 7과 그림 8은 최적화 및 시뮬레이션 결과를 보여준다. 표 1는 최적화에 걸린 시간, 목적함수의 값, 보폭, 보행속도를 보여준다. 그림 8의 Walking tile을 보면 보행패턴이 정상적인 사람의 보행과는 다른 것을 확인할 수 있다. 또한, 표 1에서 알 수 있듯이, 보행속도가 비정상적으로 느린 것을 확인할 수 있다. 또한, 보행의 중간 단계(Mid stance)에서 롤오버(Rollover)가 일어난 후 푸시오프(Pushoff)가 명확하게 일어나지 않는다. 보행의 마지막 단계(Late stance)에서는 무릎의 저크(Knee jerk)가 심하게 일어난다. 원인은 여러가지가 있을 수 있으나, 결론적으로 보면 본 시뮬레이션에서 사용한 모델이 너무 단순하여 사람의 생체역학적(Biomechanical) 특성과 맞지않기 때문이다. 예를 들어, 모델의 발목(Ankle)은 기본적으로 자유롭게 회전이 가능하지만, 사람의 발목은 아킬레스건과 같은 조직들로 인해 움직임의 범위가 상당히 제한된다. 그러므로, 모델에서는 에너지를 추가적으로 사용하면서 푸시오프를 수행할 필요가 없다. 보행속도가 비정상적으로 느린 것은 목적함수인 제어노력치를 최소화하기 위한

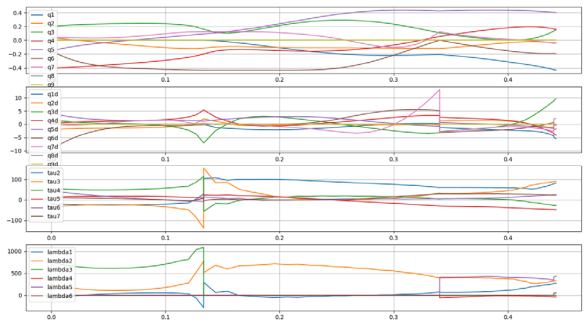

그림 9. 구속조건을 수정한 후의 이족보행로봇의 최적화결과. 관절각($\mathbf{q}$), 관절 속도($\dot{\mathbf{q}}$), 관절토크($\mathbf{u}$), and 지면반력($\boldsymbol{\lambda}$) 결과가 주어져 있음.

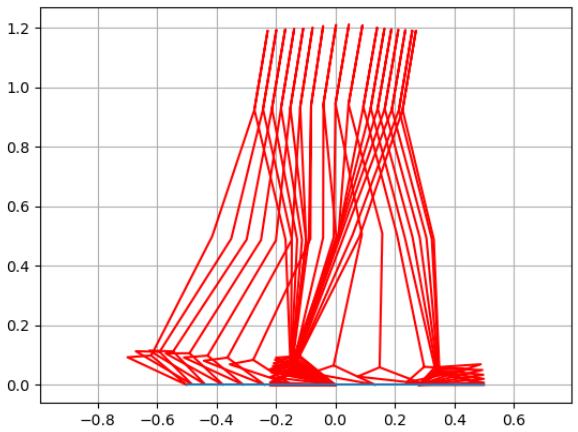

그림 10. 구속조건을 수정한 후의 이족보행로봇의 최적화결과. 보행의 궤적을 효과적으로 볼수 있도록 Walking tile을 겹쳐서(Overlay) 보여줌.

당연한 결과이다. 무릎의 저크가 심하게 발생하는 이유는 사람의 보행패턴에 따라 컨택순서를 강제로 고정시켰기 때문이다. 즉, 로봇의 보행에서는 사람의 보행패턴을 따라 하는 것이 제어 노력치의 측면에서 보면 불필요한 것이다.

그러면, 이러한 문제를 어떻게 해결할 수 있을까? 즉, 어떻게 구속조건을 수정하면 로봇의 보행을 사람의 보행과 비슷하게 만들 수 있을까? 첫 번째, 발목의 운동범위를 제한하여 아킬레스건이 존재하는 것과 유사한 효과를 구현해 줄 수 있다. 예를 들어 발목의 발등굽힘(Dorsiflexion)각도를 10도이하로 줄여줄 수 있다. 둘째, 과도한 무릎 저크는 사람의컨택 순서를 강제했기 때문이다. 특히 도메인2와 도메인3에서 푸시오프가 반드시 일어나야만 하지만, 보행속도가 느려서 자연스러운 푸시오프가 일어날 수가 없다. 그러므로, 푸시오프를 발생시킬 수 있도록 무릎 저크를 통해서 추가적인 운동에너지를 발생시킨 것이다. 그러면, 푸시오프를 발생시키기 위해서 무릎 저크를 반드시 일으켜야 하는가? Stance ankle의 관절토크를 직접 생성시킬 수도 있었을 것이다. 하지만, Stance ankle의 관절토크를 직접 생성하는 것보다 Swing knee의 저크를 발생시키는 것이 더 효율적(목적함수 값을 더 적게 증가시킴)이기 때문이다. 이를 해결하기 위해서는 반드시 충분한 운동에너지를 가지고 있어야 만한다. 그러므로, 최소 보행속도를 증가시켜주면 이러한 문제를 해결할 수 있다.

Table 2. 최적화 수행 결과

항목	결과
최적화 시간	390초
목적함수 값	$7.6723e6$
보폭	0.5m
보행속도	1.13초

그림 9과 그림 10은 이러한 구속조건 수정사항을 적용한 후의 최적화 및 시뮬레이션 결과를 보여준다. 표 2는 이에 상응하는 최적화 시간, 목적함수의 값, 보폭, 보행속도를 보여준다. 그림 10의 Walking tile을 확인해 보면 무릎 저크가 사라졌고, 푸시오프가 두드러지게 나타났음을 확인할 수 있다. 보행속도와 보폭도 정상인의 보행속도가 비슷한 것을 확인할 수 있다(표 2). 흥미로운 것은 목적함수의 값이다. 정상인의 보행패턴과 유사하도록 구속조건을 수정한 결과, 목적함수의 값이 대략 8배 가까이 증가한 것을 확인할 수 있다. 즉, 훨씬 더 많은 에너지를 사용하여야만 사람의 보행패턴을 흉내낼 수 있다는 것이다. 여기서 두 가지 포인트를 생각해 볼 수 있다. 첫 번째는, 로봇의 보행이 반드시 사람의 보행패턴을 따라 해야만 할 필요는 없다는 것이다. 로봇은 사람의 조직과 같은 생체역학적 구조를 가지고 있지 않기 때문에, 사람의 최적운동이 반드시 로봇에게도 최적일 이유는 없다. 오히려 로봇에게 최적인 움직임은 사람의 그것과는 전혀 다른 결과를 보여준다. 둘째, 목적함수의 값만 보면 사람의 보행이 훨씬 더 비효율적인 것으로 보일 수도 있으나, 이것은 옳지 않은 해석이다. 왜냐하면 아킬레스건의 존재는 단순히 발목의 운동범위를 제한하는 것에 있지 않고 에너지를 재활용하는 것에 있다. 즉, 롤오버가 일어날 때 운동에너지가 탄성의 위치에너지로 저장되었다가 푸시오프가 일어날 때 저장된 에너지를 사용함으로써 추가적인 에너지를 소모하지 않기 때문이다. 오히려 사람의 보행이 로봇의 보행보다 더 효율적일 가능성이 많다.

e. 초기조건 및 Local minimum 문제. 본 실험에서는 임의의 초기조건을 사용하여 비선형 계획법의 최적화를 수행하였다. Nonconvex 최적화 문제의 특성상, 초기조건에 따른 Local

minimum 문제가 발생할 수 있다. 본 실험에서는 대략 95% 이상의 확률로 동일한 최적의 해를 찾을 수 있었다. 임의의 초기조건을 적용하는 것이 항상 옳지 않을 수도 있다. 이는 사람의 보행과 비슷한 보행을 원하는 사회/문화적 통념에 의한 경우가 많다. 그렇다 하더라도 사람의 보행과 비슷한 최적의 해가 Global minimum이 되는 경우가 대부분이다. 그러므로, 초기조건을 정할 때 임의의 값을 지정할 수도 있지만, 사람의 보행패턴을 초기조건으로 지정하는 것도 충분한 의미가 있을 뿐 아니라 계산상의 효율성도 보장할 수 있다.

f. Direct collocation 이외 다른 방법. 앞의 실험에서 볼 수 있듯이, Direct collocation은 주어진 동적 시스템에 대해 특정 목적함수를 최소화하는 최적제어와 최적의 궤적을 구하는데 좋은 도구임을 확인할 수 있었다. 하지만, Direct collocation이 유일한 방법은 아니다. 다른 가능한 방법들에 대해서 간단히 알아보도록 하자.

(1) Single shooting. Direct collocation은 목적함수와 구속조건을 전사하는 과정을 거친다. Single shooting도 마찬가지다. 한 가지 차이점은 Single shooting의 경우 초기값으로부터 Marching 방법을 통해 시뮬레이션을 직접 수행하면서 최적의 궤적을 찾는다는 점이다. 직접 시뮬레이션을 수행해야 하므로, Forward Euler 방법이나 Explicit Runge Kutta 방법을 사용한다(주로 후자). Single shooting은 직접 시뮬레이션을 하므로 정확도는 더 높을 수 있으나, 구속조건이나 제어가 간단한 경우가 아니면 해를 찾는 것이 쉽지 않다.

(2) Multiple shooting. Multiple shooting은 Single shooting과 비슷하다. 차이점은 한 번의 시뮬레이션만 수행하는 Single shooting과는 달리, 전체 시간을 여러 개의 구간으로 나누어서 각 구간 별로 따로 시뮬레이션을 수행하고, 각 노드 별로 오차를 줄이기 위해 초기조건을 찾는 비선형 프로그래밍을 수행한다. Multiple shooting은 Single shooting에 비해서 더 강인한 결과를 주고, 더 확장성이 크다(Scalable). Direct collocation에 비해서 설계변수(Decision variable)의 갯수가 적다는 장점이 있지만, 궤적에 대한 구속조건을 설정하는 것이 용이하지 못하다.

(3) Orthogonal collocation. Orthogonal collocation은 Direct collocation과 비슷하다. 큰 차이점은 Orthogonal collocation은 고차의 Chebyshev polynomial 이나 Legendre polynomial과 같은 Orthogonal polynomial 을 사용하여 Collocation point를 구한다는 점이다. 최적화하려는 함수가 충분히 부드럽다면 (Smooth) 최적해를 찾는 수렴속도(Convergence rate)가 지수적(Exponential)으로 빠르다.

(4) Differential dynamic programming. Differential dynamic programming은 시뮬레이션을 수행한 후 최적화를 수행한다는 점에서 shooting 방법과 유사하다. 차이점은 최적화를 수행하는 방법이다. Shooting은 일반적인 비선형 계획법을 사용하여 최적화를 수행하지만, Differential dynamic programming은 Dynamic programming의 방식을 적용하여 최적제어를 역전파(Backward propagation)함으로써 최적제어 및 궤적을 구한다.

VI. Conclusion

본 기고문에서는 다양한 구속조건을 가지는 동적 시스템의 최적제어와 최적의 궤적을 구하기 위해 Direct collocation 방법을 리뷰하였다. 목적함수, 연속동역학, 이산동역학, 그리고 기타 다양한 구속조건들을 전사(Transcription)의 과정을 통해 비선형 계획법의 문제로 전환한 후, 일반적인 비선형 프로그래밍 Solver를 활용하여 근사해법을 찾는다. 이족보행의 예를 들어, 하이브리드 시스템에 대해서 Direct collocation 방법을 적용하는 법을 살펴보았다. 특히 구속조건을 어떻게 해석하고 적용하는 것이 합리적인지에 대해 고찰하였다. 또한, Direct collocation 외의 다른 유사한 방법들에 대해서 간단히 알아보았다. 본 기고문을 통해서 동적 시스템을 제어하기 위해 레퍼런스를 생성하는 부분에서 조금이나마 도움이 되었기를 바란다.

REFERENCES

[1] K. Friston, “What is optimal about motor control?,” *Neuron*, vol. 72, pp. 488-498, 2011.

[2] J. T. Betts, “Survey of numerical methods for trajectory optimization,” *J. Guidance Control Dynam*, vol. 21, pp. 193-207, 1998.

[3] M. Kelly, “An introduction to trajectory optimization: How to do your own direct collocation,” *SIAM Review*, vol. 59, pp. 849-904, 2017.

[4] J. T. Betts, “Practical Methods for Optimal Control and Estimation Using Nonlinear Programming,” *SIAM,* 2010.

[5] K. Chao and P. Hur, “A step towards generating human-like walking gait via trajectory optimization through contact for a bipedal robot with one-sided springs on toes,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4848-4853, Sep. 2017.

[6] K. Chao and P. Hur, "Generalized contact constraints of hybrid trajectory optimization for different terrains and analysis of sensitivity to randomized initial guesses," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1435-1440, Nov. 2019.

[7] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Engrg.*, vol. 33, pp. 575-582, 2009.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization,* Cambridge University Press, 2004.

[9] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, 2006.

[10] Mathworks, MATLAB Optimization Toolbox, 2022.

[11] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *International Journal of Robotics Research*, vol. 33, pp. 69-81, 2014.

[12] D. Winter, *Biomechanics and Motor Control of Human Movement*, 4th Ed. Wiley, 2009.

저자약력

허 필 원

- 2010년 University of Illinois at Urbana-Champaign 응용수학(최적화 및 알고리즘) 석사학위 및 기계공학부 박사학위
- 2014년-2020년 Texas A&M University 기계공학부 조교수
- 2020년 12월-현재 광주과학기술원 기계공학부 부교수
- Frontiers in Human Neuroscience, International Journal of Control, Automation and Systems, ICRA 부편집장
- 연구분야: Human-Robot Interaction, Free Energy Principle, Bipedal Gait/Balance, Robot Dynamics and Control, Trust Estimation, Biomechanics, Neuromechanics, Sensorimotor Control and Rehabilitation