동적 이산 경사 하강법을 이용한 직진하는 우주선의 Dyson Slingshot 궤적 결정법

윤명, 전지호, 김한준, 김찬우, 홍진목
과학영재학교 경기과학고등학교

The Method Determining Dyson Slingshot Trajectory of a Spacecraft Following a Rectilinear Path Using the Dynamic Discrete Gradient Descent

Myeong Yoon, Jiho Jeon, Hanjoon Kim, Chanwoo Kim, Jinmok Hong
Gyeonggi Science High School for the Gifted

국문 초록

본 연구에서는 우주선이 충분히 멀리 떨어진 목표 항성계를 향해 날아가는 상황을 상정하고, 가속을 위해 경로 도처에 존재하는 쌍성계를 스윙바이하는 시나리오를 설계하였다. 보다 구체적으로 이러한 방식의 스윙바이는, 진행하던 경로에서 벗어나 항성계 방향으로 이동하고 이후 Dyson slingshot으로 알려진 쌍성계 스윙바이를 수행하는 일련의 과정을 거쳐야 함을 제시하였고, 최적화 문제를 구성하여 수치적으로 접근할 수 있도록 설계하였다. 이에 중력장이 없는 공간에서 우주선이 최소의 연료를 사용하며 속도를 전환하는 궤적을 제시하였다. 또한 Dyson slingshot의 궤적 결정을 위하여, 기존의 경사 하강법을 개선한 이산 경사 하강법을 고안하여 연료 절약량이 최대가 되는 궤적을 결정할 수 있는 방법을 제시하였다.

Keywords: binary star system, dyson slingshot, swing-by, spacecraft flight dyanimcs, spacecraft acceleration, gradient descent

I. 서론

가까운 미래에 실용할 수 있을 것으로 기대하기는 어렵지만, 장거리를 여행하는 우주선은 다른 항성계를 스윙바이할 수 있을 것이다. 다른 항성계를 스윙바이 하고자 할 때, 다중성계를 이용한 스윙바이 연구의 필요성은 우주에 다중성계가 얼마나 흔하게 나타나는지가 뒷받침해 준다. 항성계의 multiplicity 빈도는 항성의 질량이 증가함에 따라 증가하는 경향을 보이는데, 종족 I 주계열성의 경우 1.5 ~ 5 solar mass 범위에서는 절반 이상, 16 solar mass 이상에서는 80% 이상의 항성계가 다중성계인 것으로 알려져 있다.[1] 이렇듯 intragalactic 이상의 스케일의 우주 비행에서는 스윙바이 대상으로 단일성보다 쌍성계를 빈번하게 조우하게 될 것이다. 이에 본 연구에서는 장거리 우주 비행에서 쌍성계를 스윙바이하는, Dyson slingshot으로 알려진 항법의 구사를 구체적으로 설계하고자 한다. 보다 단기적으로, 본 연구가 SF 영화를 비롯한 장거리 우주 비행 또는 스윙바이를 주제로 하는 여러 창작물의 고증에도 도움이 될 것으로 기대한다.

II. 이론적 배경

1. Kepler 문제

Kepler 문제란 다음과 같은 상호작용이 주어진 two-body problem을 말한다.

$$\mathbf{F} = \frac{k}{r^2}\hat{\mathbf{r}} \tag{II.1}$$

질량이 M인 항성 주위를 도는 질량 m의 물체를 생각하자. 표준 중력 변수 $\mu = G(M+m) \approx GM$와 비각운동

량 $h = \mathbf{r} \cdot \dot{\mathbf{r}}$을 이용하여 항성이 원점에 놓인 관성계에서 Kepler 문제의 해를

$$r = \frac{h^2}{\mu(1 + e\cos\theta)} \tag{II.2}$$

와 같이 극좌표에서 표현할 수 있다. 여기서 e를 궤도 또는 궤적의 이심률이라 한다.

2. Kepler 방정식

Kepler 방정식은 평균근점이각 M, 이점근점이각 E, 이심률 e 사이에 성립하는 다음의 관계식을 일컫는다.[2]

$$M = E - e\sin E \tag{II.3}$$

$e \leq 1$일 때, 제1종 베셀 함수 J_α를 이용하여 E를 M에 대한 식으로

$$E = M + \sum_{m=1}^{\infty} \frac{2}{m} J_m(me)\sin(mM) \tag{II.4}$$

와 같이 전개할 수 있다.(단, $M \in [-\pi, \pi]$) 이를 급수 꼴로 쓰면 다음과 같다.[3]

$$E = \begin{cases} s + \dfrac{1}{60}s^3 + \dfrac{1}{1400}s^5 + \cdots \text{ where } s = (6M)^{\frac{1}{3}} & (e = 1) \\ \dfrac{1}{1-e}M - \dfrac{e}{(1-e)^4}\dfrac{M^3}{3!} + \dfrac{9e^2+e}{(1-e)^7}\dfrac{M^5}{5!} + \cdots & (e \neq 1) \end{cases} \tag{II.5}$$

본 연구에서는 II.5의 5차항 이상을 무시하였다.

3. 쌍성계

두 별의 질량이 m_1, m_2인 쌍성계의 질량중심을 원점으로 하는 좌표계를 도입하면, 두 별의 운동량 합이 0이므로 $i \in \{1, 2\}$에 대해

$$m_i\ddot{\mathbf{r}}_i = -\frac{Gm_1m_2{m_i}^2}{(m_1+m_2)^3}\frac{\mathbf{r}_i}{\|\mathbf{r}_i\|^3} \tag{II.6}$$

이 성립한다. 이는 Kepler 문제와 동치이며, 따라서 각 쌍성의 궤도는 이심률이 동일한 타원임을 알 수 있다. 쌍성 사이 거리가 최소인 때를 $t = 0$, 이때 거리를 r로 하면, 평균근점이각은 정의에 따라 다음과 같이 주어진다.

$$M = \sqrt{\frac{Gr^3}{e^3}(m_1+m_2)t} \tag{II.7}$$

4. 델타-V

우주선 비행 역학에서 델타-V $\Delta\mathfrak{v}$는 행성이나 달에서 발사, 착륙, 또는 우주선의 궤도 천이 같은 작업을 수행하는 데에 필요한 우주선의 연료량에 비례하는 량이다. 이는 속도 단위의 스칼라 값으로, 외력이 작용하지 않을 때 가속도를 시간에 대해 적분하여 얻는다. 우주선 속도 변화량의 절댓값과는 구분된다.

5. 변분법

변분법은 범함수의 극값을 찾는 수학적인 방법 중 하나이다. f가 $i = 0, 1, \cdots, n$에 대해 경계조건 $f^{(i)}(x_1) = f_{1,i}$, $f^{(i)}(x_2) = f_{2,i}$을 만족할 때, Lagrange의 접근을 이용하여 다음과 같은 범함수 J가 극값을 가질 조건을 찾을 것이다.

$$J[f] = \int_{x_1}^{x_2} L\left(f(x), f'(x), \cdots, f^{(n)}(x); x\right) dx \tag{II.8}$$

J가 극솟값을 가진다면 f의 작은 섭동 하에 J는 증가해야 한다. n번 미분 가능한 함수 $\eta(x)$를 도입하여 작은 ϵ에 대해 f를 $f_\epsilon = f + \epsilon\eta (= f + \delta f)$로 쓰기로 하자. 여기서, f의 경계조건은 여전히 충족되어야 하므로 $i = 0, 1, \cdots, n$에 대해 $\eta^{(i)}(x_1) = \eta^{(i)}(x_2) = 0$를 만족하는 함수로 한다. 같은 맥락에서, 섭동 후의 J와 L에 대하여 다음과 같은 표기를 사용하자.

$$\begin{aligned} J_\epsilon[f_\epsilon] &= \int_{x_1}^{x_2} L\left(f_\epsilon(x), {f_\epsilon}'(x), \cdots, {f_\epsilon}^{(n)}(x); x\right) dx \\ &= \int_{x_1}^{x_2} L_\epsilon dx \end{aligned}$$

f에서 J가 극솟값을 가지려면 $\epsilon = 0$ 근처에서 J가 증가해야 한다. 연쇄 법칙과 부분적분법을 적용하면,

$$\begin{aligned} \frac{dJ_\epsilon}{d\epsilon} &= \frac{d}{d\epsilon}\int_{x_1}^{x_2} L_\epsilon dx \\ &= \int_{x_1}^{x_2} \left(\frac{\partial L_\epsilon}{\partial x}\frac{dx}{d\epsilon} + \sum_{i=0}^{n} \frac{\partial L_\epsilon}{\partial {f_\epsilon}^{(i)}} \frac{d{f_\epsilon}^{(i)}}{d\epsilon} \right) dx \\ &= \int_{x_1}^{x_2} \sum_{i=0}^{n} \left((-1)^i \frac{d^i}{dx^i} \frac{\partial L_\epsilon}{\partial {f_\epsilon}^{(i)}} \right) \eta dx \end{aligned} \tag{II.9}$$

변분법의 기본 정리에 따라, II.8가 극값을 가질 조건으로 다음과 같은 Euler-Poisson 방정식을 얻는다.[4]

$$\sum_{m=0}^{n} (-1)^m \frac{d^m}{dx^m} \frac{\partial f}{\partial y^{(m)}} = 0 \tag{II.10}$$

6. 경사 하강법

경사 하강법이란 다변수 함수의 극솟값을 찾기 위한 1차 반복 최적화 알고리즘으로, 무작위적인 점에서 시작하여 해당 위치에서의 기울기 벡터의 반대 방향으로 이동하여 극값에 이를 때까지 반복하는 방법이다. 즉, 함수 $f : \mathbb{R}^n \to \mathbb{R}$에 대하여 초기 추측값 $\mathbf{x}_0$가 주어졌을 때, 양수 γ_i에 관하여 다음과 같은 관계식을 반복한다.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla f(\mathbf{x}_i) \tag{II.11}$$

상수 γ_i의 선택에 따라 극솟값으로의 수렴 여부가 결정되며, Barzilai–Borwein method에서는 수렴성을 보장하기 위하여 다음과 같은 값을 제시하고 있다.[5]

$$\gamma_i = \frac{\left| (\mathbf{x}_i - \mathbf{x}_{i-1})^T [\nabla F(\mathbf{x}_i) - \nabla F(\mathbf{x}_{i-1})] \right|}{\|\nabla F(\mathbf{x}_i) - \nabla F(\mathbf{x}_{i-1})\|^2} \tag{II.12}$$

III. 평행 스윙바이 문제

1. 단일성 스윙바이

스윙바이(Swing-by), 혹은 중력 슬링샷(Gravitational Slingshot)이란 우주선 항법 중 하나로 행성 등 다른 천체의 중력을 이용하여 궤도를 조정하고 속도를 내는 방법을 말한다. 우주선이 목성과 같이 중력이 큰 행성의 궤도를 지날 때 행성의 중력에 끌려 들어가다 바깥으로 튕겨 나가듯 속력을 얻는 것이 그 예이다.

다른 항성계를 이용하여 스윙바이한다는 연구 주제의 특성 상 우주선이 충분히 멀리 떨어진 항성계를 목적지로 여행하는 상황을 생각한다. 경로의 명확성을 위하여, 본 연구에서는 태양계를 원점으로, 진행 방향을 $+x$방향으로 하는 관성계 $\mathfrak{R}_0$를 기준으로 하였고, 필요에 따라 다른 좌표계를 도입하는 방식을 채택했다.

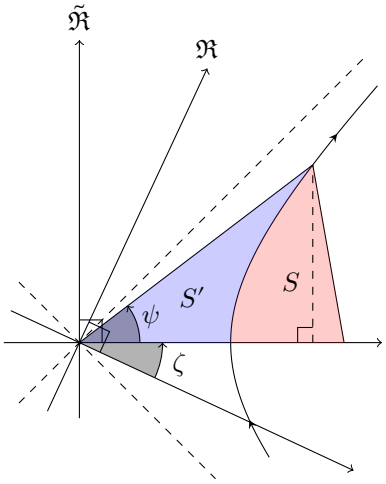

Figure 1: 관성계 $\tilde{\mathfrak{R}}$에서 본 우주선의 쌍곡선 궤적

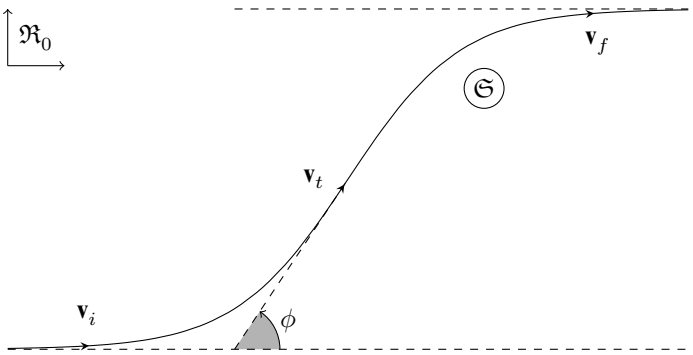

Figure 2: 평행 스윙바이 문제의 모식도

단일성을 이용한 스윙바이를 고려하기 위하여, $\mathfrak{R}_0$의 원점을 $\mathfrak{S}$로 옮긴 관성계 $\mathfrak{R}$과, $\mathfrak{R}$에서 우주선이 따르는 쌍곡선 궤도를 해석하기 위한 관성계 $\tilde{\mathfrak{R}}$를 도입할 것이다. 관성계 $\tilde{\mathfrak{R}}$에서, 우주선은 Figure 1과 같은 쌍곡선 궤적을 따르며 이 궤적 $(\tilde{x}/a)^2 - (\tilde{y}/b)^2 = 1$은 쌍곡각 ψ를 이용해 다음과 같이 매개화할 수 있다.

$$\begin{cases} \tilde{x} = -a\cosh\psi \\ \tilde{y} = b\sinh\psi \end{cases} \tag{III.1}$$

이러한 매개화 하에 쌍곡선 궤도에 대한 다음의 Kepler 방정식이 성립한다.

$$b\sqrt{a^2+b^2}\sinh\psi + ab\psi = ht \tag{III.2}$$

따라서 관성계 $\mathfrak{R}$에서의 스윙바이 궤적은 다음과 같이 얻어진다.

$$\begin{pmatrix} x_S \\ y_S \end{pmatrix} = \begin{pmatrix} \cos\zeta & -\sin\zeta \\ \sin\zeta & \cos\zeta \end{pmatrix} \begin{pmatrix} -a\cosh\psi \\ b\sinh\psi \end{pmatrix} + \begin{pmatrix} V_x \\ V_y \end{pmatrix} t \tag{III.3}$$

2. 평행 스윙바이 문제

직선 항로를 따르던 우주선의 도처에 다른 항성계 $\mathfrak{S}$가 존재할 때, 우주선은 그 항성계 방향으로 방향을 바꾼 후 $\mathfrak{S}$를 스윙바이하며 속도를 증가시킬 수 있을 것이다. 이때, 목적지 항성계는 충분히 멀리 떨어져 있으므로 가속 보조 전후 속도의 방향은 바뀌어선 안 된다. 모식적으로 Figure 2와 같이 나타나는 이 일련의 과정을 평행 스윙바이 문제라고 명명한다.

초기 우주선의 속도를 $\mathbf{v}_i$라 하자. 우주선은 항성계 $\mathfrak{S}$를 스윙바이하기 위하여 방향을 틀어야 한다. 여기서 방향 전환 이후의 속도를 $\mathbf{v}_t$라 하자. 방향 전환 궤적에 관한 구체적 논의는 추후에 이루어질 것이다. 스윙바이 후, 우주선의 속도는 $\mathbf{v}_i$와 같은 방향의 $\mathbf{v}_f$가 된다.

$\mathfrak{S}$가 단일성인 경우, 방향 전환을 거친 뒤 스윙바이는 III.2와 같은 매개화 하에 시간 원점을 우주선과 $\mathfrak{S}$가 가장 가까운 시점으로 하는 $\mathfrak{R}$에서의 Kepler 문제로 환원된다. 여기서 $\mathbf{v}_f$의 방향에 관한 조건이 주어져 있으므로, 우주선의 경로는 독립 변수 2개에 의해 유일하게 결정되며 평행 스윙바이 문제의 자유도는 2가 된다. 실질적인 우주선의 제어에서 가장 유의미하게 사용될 수 있는 독립 변수 쌍으로는 v_t와 ϕ, 즉 $\mathbf{v}_t$를 들 수 있을 것이다. 우주선은 $\mathbf{v}_t$ 벡터 하나를 결정함으로써 유일한 스윙바이 경로를 택할 수 있다.

반면 $\mathfrak{S}$가 쌍성계인 경우, 위와 같이 스윙바이 궤적이 coplanar하다는, 즉 방향 전환 궤적과 $\mathfrak{S}$의 공전궤도면이 coplanar하다는 가정을 추가하더라도 $\mathfrak{S}$에 의한 중력장이 시간에 따라 변화하고 있으므로 시간 원점 선택에 대한 자유도를 배제할 수 없다. 이 경우 평행 스윙바이 문제에서는 3의 자유도를 갖게 된다. 그럼에도 불구하고 후술할 수치해석 전략에서는 평행 스윙바이 문제의 자유도를 2로 간주할 것인데, 이는 시간 원점이 우주선의 항법 수준에서 선택할 수 있는 변수가 아니기 때문이며, 이와 관련된 부가적인 설명은 해당 단락에서 이어질 것이다.

IV. 방향 전환

1. 최소 델타-V 궤적

$\Delta\mathfrak{v}$를 직교 좌표에서 전개하면 다음과 같이 II.8의 형태로 쓸 수 있다.

$$\begin{aligned}\Delta\mathfrak{v} &= \int_{t_1}^{t_2} \|\vec{a}\| dt \\ &= \int_{t_1}^{t_2} \sqrt{\ddot{x}^2+\ddot{y}^2}\, dt\end{aligned} \tag{IV.1}$$

Euler-Poisson 방정식 II.10에서

$$\begin{cases} \dfrac{\ddot{x}}{\sqrt{\ddot{x}^2+\ddot{y}^2}} = \alpha t + \beta \\ \dfrac{\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2}} = \gamma t + \delta \end{cases}$$

를 얻는다. 두 식을 제곱해서 더하면

$$1 = \left(\frac{\ddot{x}}{\sqrt{\ddot{x}^2+\ddot{y}^2}}\right)^2 + \left(\frac{\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2}}\right)^2 = (\alpha t+\beta)^2 + (\gamma t+\delta)^2$$

이므로, $\alpha = \gamma = 0$임을 알 수 있다. 따라서 $\ddot{x}$와 $\ddot{y}$ 사이의 비는 일정하며, 비례상수 ξ를 통해 x와 y 사이에 일반적으로 성립하는 관계식

$$y(t) = \xi x(t) + c_2 t + c_3 \tag{IV.2}$$

을 도출할 수 있다. Euler-Poisson 방정식은$\Delta\mathfrak{v}$가 극값을 가질 조건이므로, 최솟값을 가질 조건을 확인할 필요가 있다. IV.1에 IV.2를 대입하면 $\Delta\mathfrak{v}$의 값의 최솟값은 다음과 같이 나타난다.

$$\begin{aligned}\Delta\mathfrak{v} &= \int_{t_1}^{t_2} \sqrt{\ddot{x}^2+\ddot{y}^2}\, dt \\ &= \sqrt{1+\xi^2}\int_{t_1}^{t_2} |\ddot{x}|\, dt \\ &\geq \sqrt{1+\xi^2}\,|\Delta\dot{x}|\end{aligned} \tag{IV.3}$$

자명하게, 등호의 성립은 $\ddot{x}$의 부호가 변하지 않음과 동치이다.

2. 방향 전환 궤적

앞서, $\mathbf{v}_t$ 벡터 하나를 결정함으로써 우주선이 유일한 스윙바이 경로를 택할 수 있다는 것에 관하여 논하였다. 이 단락에서는 실제로 우주선이 $\mathbf{v}_i$의 속도로 진행하다가 $\mathbf{v}_t$의 속도를 가지도록 운동 상태를 변화시킬 때 최소한의 연료를 사용하기 위해서 우주선은 어떤 경로를 따라야 하는가에 관하여 논할 것이다.

$x=0$의 경로를 따라 속도 $v_i\hat{x}$로 진행하던 우주선이 $\mathbf{v}_t$로 속도를 바꾸는 상황을 상정하자. 우주선이 델타-V를 최소로 하며 속도를 바꿀 궤적은 다음의 점근선 조건

$$\begin{cases} \lim_{t\to-\infty} y = 0 \\ \lim_{t\to\infty} (x\tan\phi - y) = 0 \end{cases}$$

과 속도 경계조건

$$\begin{cases} \lim_{t\to-\infty} \dot{x} = v_i \iff \lim_{t\to-\infty} \dot{y} = 0 \\ \lim_{t\to\infty} \dot{x} = v_t\cos\phi \iff \lim_{t\to\infty} \dot{y} = v_t\sin\phi \end{cases}$$

을 만족해야 한다. 먼저 $t\to-\infty$에서의 속도 경계조건을 사용하면

$$\lim_{t\to-\infty} \dot{y} = \lim_{t\to-\infty} \xi\dot{x} + c_2 = 0$$

이므로 $c_2 = -\xi v_i$이다. 다음으로 $t\to+\infty$에서의 속도 경계조건을 사용하면

$$\lim_{t\to\infty} \frac{\dot{y}}{\dot{x}} = \lim_{t\to\infty} \xi\left(1-\frac{v_i}{\dot{x}}\right) = \tan\phi$$

이고, 비례상수 ξ를 $\xi = \dfrac{\sin\phi}{\cos\phi - v_i/v_t}$로 결정할 수 있다. 따라서 IV.2는 아래와 같이 다시 쓸 수 있다.

$$y(t) = \xi x(t) - \xi v_i t + c_3 \tag{IV.4}$$

우주선의 가속은 연료 분사를 통해 이루어지므로, $\dot{x}(t)$를 시그모이드 함수 $\sigma(t)$로 가정하자. 이제 $x(t)$는

$$x(t) = \int_0^t \sigma(t)dt + C$$

의 꼴로 쓸 수 있다. $t\to-\infty$에서의 점근선 조건에 의해 $\lim_{t\to-\infty}(x(t)-v_it)=0$이고, 이를 통해 적분상수 C를 결정하면 다음과 같다.

$$x(t) = \int_0^t \sigma(t)dt + \int_{-\infty}^0 (\sigma(t)-v_i)\,dt \tag{IV.5}$$

V. 평행 스윙바이 문제의 수치적 해법

1. 최적화 문제 설계

본 연구에서는 가장 '효율적인' 스윙바이 궤적를 결정하기 위한 objective function을 $f=(v_f-v_i)-\Delta\mathfrak{v}$로 정의한다. 이는 동일한 속력 증가를 단순 가속으로만 얻었을 때에 대비해 절약한 연료량에 비례하는 값이다. 우주선의 운동에서 자연스럽게 택해지는 세 변수는 v_t, v_f, ϕ이다. 그러나 이 세 변수로부터 스윙바이 궤적을 결정하기 위해서는, $t\to-\infty$와 $t\to\infty$에서의 속도 벡터가 주어진 경곗값 문제를 해결해야 한다. 이는 수치해석에 의존해야 한다. 본 연구의 목적은 가능한 스윙바이 궤적 중 objective function을 최대로 하는 궤적을 찾는 것이므로

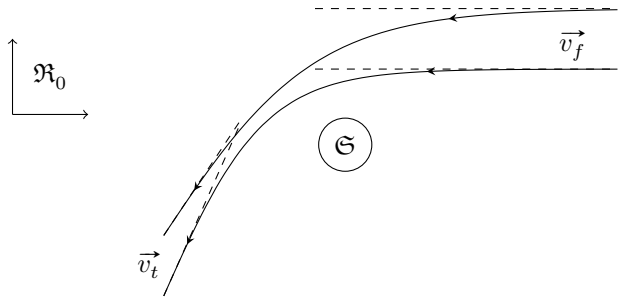

Figure 3: 시간 반전을 적용한 평행 스윙바이 문제의 스윙바이 부분

이를 위해선 함숫값을 가능한 모든 변수들에 대해 계산해 보아야 한다. 더군다나 함숫값을 구한 후 그 중 최대인 점을 찾는 과정 역시 번거롭다.

이에, 본 연구에서는 평행 스윙바이 문제에 시간 반전을 적용하여 3변수 경곗값 문제를 2변수 초깃값 문제로 바꾸어 해결하고자 하였다. $t = 0$에서 $\mathfrak{S}$의 질량중심을 원점에 위치시키고, 우주선은 (L, d)에서 속력 v_f로 $-x$ 방향으로 운동하고 있다고 설정한다. 원 문제의 경계조건 중 $t \to \infty$에서의 속도 방향이 x축과 평행하다는 것을 고려하면 L은 충분히 커야 한다. 이후 시간 반전을 고려한 $\mathfrak{S}$의 중력장 위로 초기 속도를 가진 물체가 운동하는 것을 계산하면, 우주선의 속도는 $-\mathbf{v}_t$로 수렴한다.

이는 마치 변수의 수를 줄인 것처럼 보이지만, 쌍성계의 configuration은 선택할 수 있는 변수가 아님을 고려하면 타당한 방법이다. 기존의 3변수 경곗값 문제에서, $t = 0$이 어떻게 선택되었는지 명시하지 않았다. 앞서 언급한 경곗값 문제의 해로서 $t = 0$에서의 configuration이 도출되는데, 이는 이미 비행하고 있는 우주선이 결정 가능한 변수가 아니다. 따라서 평행 스윙바이 문제의 자유도는 2라고 보는 것이 합당하다. 궤도면이 결정된 Kepler 궤도는 자유도가 4이고, 여기에 $\mathbf{v}_f$의 방향과 시간 원점이 결정되었기 때문이다. 이 과정에서는 항성계의 configuration이 결정되어 있다는 사실 외에는 고려하고 있지 않다. 실제로 스윙바이를 시도할 때는 결정된 쌍성계의 configuration이 t에 대해 어떻게 나타나는지, 본 연구의 방법에서 이용할 수 있는 형태로 역산하는 과정을 거쳐야 올바른 계산을 수행할 수 있을 것이다. 이 구체적인 방법에 관한 것은 후속 연구 과제로 남겨두기로 한다.

2. 동적 이산 경사 하강법

본 연구에서 논하고 있는 문제 상황과 같이 objective function이 변수에 대해 닫힌 꼴로 표현되지 않는 경우, 곧바로 함수의 기울기 벡터를 구하여 경사 하강법을 적용하는 것은 불가능하다. 이같은 경우에도 경사 하강법을 적용하기 위해, 본 연구에서는 이산적인 함수의 극솟값을 경사 하강법을 이용하여 찾는 방법을 고안하였다.

정의역에서 특정한 직사각형 또는 그에 상응하는 고차원의 영역을 선택한 뒤 일정한 간격의 격자로 나누어 각 격자점에 대하여 objective function의 값을 구하면 격자점에서의 symmetric diffrence quotient를 그 점에서의 기울기 벡터 성분으로 간주할 수 있다. 이렇게 각 점에 함숫값과 기울기 벡터를 대응시킨 뒤, 기존 경사 하강법과 동일한 방법으로 최적화를 수행할 수 있다. 이때 $\mathbf{x}_{i+1}$은 $\mathbf{x}_i - \gamma_i \nabla f(\mathbf{x}_i)$에서 가장 가까운 격자점으로 택하기로 한다.

문제 상황의 objective function은 실제로는 연속 함수이다. 이를 충분히 세밀하지 못한 간격의 격자로 쪼갠 후 이산 경사 하강법을 수행하면 실제 극솟값과의 오차가 커지게 되고, 격자의 간격을 줄이면서 정의역을 유지하면 격자의 수의 제곱에 비례하여 수치해석에 소요되는 시간이 증가하게 된다. 이에 따라 실제 문제 상황에 이산 경사 하강법을 적용하려고 할 때에는 크게 다음의 두가지 문제가 대두된다. 첫째, 격자 사이의 간격이 너무 크면 경사 하강법의 정확도가 높지 않은 경우($\mathbf{x}_{i+1} = \mathbf{x}_i$), 그리고 둘째, 정의역에서 선택한 영역 D 내에 극소인 점이 존재하지 않아 경계를 넘어서는 경우($\mathbf{x}_{i+1} \notin D \backslash \partial D$)이다. 따라서 수치해석에 사용하는 정의역의 특정 영역 D을 동적으로 조정하며 경사 하강을 수행하면, 전술한 두 문제를 동시에 해결할 수 있다. 처음 제시한 방법과 동일하게 경사 하강을 시작하나, 전술한 두 상황에 도달한 경우 해당 위치를 중심으로 새로운 영역 D'을 지정하고

이산 경사 하강법을 수행하는 것이다. 이러한 수치해석법을 동적 이산 경사 하강법(Dynamic Discrete Gradient Descent, DDGD)로 명명한다.

본 연구에서는 DDGD를 이용했을 때 기존의 경사 하강법을 이용하여 구한 극솟값보다 참값에 훨씬 근접하는 값을 제공한다는 것을 확인하였다. 이는 비단 objective function이 이산적인 함수이거나 closed-form으로 쓰이지 않아 곧바로 경사 하강법을 적용할 수 있는 경우가 아니라 하더라도 DDGD가 기존의 방식보다 개선된 방식임을 뜻한다.

3. MATLAB 코드

전술한 일련의 수치해석 과정에 필요한 MATLAB 코드는 아래와 같다. v_i, $\mathfrak{S}$의 정보, DDGD의 대상이 되는 함수 등은 모두 임의로 입력되어 있다. v_i는 상수이므로 objective function의 반환값은 $v_f - \Delta\mathfrak{v}$로 하였다.

```
function ob = obj_f(vf, d)
m1 = 10 ^ 30.5;
m2 = 10 ^ 31;
r = 10 ^ 8;
f = 0.3; %eccentricity
G = 6.6743 * 10 ^ -11;
L = 10 ^ 8;
vi = 10 ^ 6;

ax = zeros(2001,1);
ay = zeros(2001,1);
vx = zeros(2001,1);
vy = zeros(2001,1);
x = zeros(2001,1);
y = zeros(2001,1);

x(1,1) = L;
y(1,1) = d;
vx(1,1) = -vf;
vy(1,1) = 0;

k = sqrt(G * (m1 + m2)) * (f / r) ^ (1.5);
a = r * m1 / (f * (m1 + m2));
b = r * m2 / (f * (m1 + m2));
P = f * (m1 + m2) *G / r;

for i = 1:2000
t = i;
ax(i,1) = - P * (b * (x(i,1) - a * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) / ((x(i,1) - a * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) ^ 2 + (y(i,1) - a * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) ^ 2) ^ (1.5) + a * (x(i,1) + b * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) / ((x(i,1) + b * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) ^ 2 + (y(i,1) + b * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) ^ 2) ^ (1.5));
vx(i + 1,1) = vx(i,1) + ax(i,1) * 1;
x(i + 1,1) = x(i,1) + vx(i,1) * 1;
ay(i,1) = - P * (b * (y(i,1) - a * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) / ((x(i,1) - a * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) ^ 2 + (y(i,1) - a * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) ^ 2) ^ (1.5) + a * (y(i,1) + b * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) / ((x(i,1) + b * (cos(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) - f)) ^ 2
```

```
    + (y(i,1) + b * sin(k * t / (1 - f) - f * (k * t) ^ 3 / 6 * (1 - f) ^ 4) * sqrt(1 - f ^2)) ^ 2)
    ^ (1.5));
     vy(i + 1,1) = vy(i,1) + ay(i,1) * 1;
     y(i + 1,1) = y(i,1) + vy(i,1) * 1;
     end

     phi = atan(vy(2001,1) / vx(2001,1));
     xi = sin(phi) / (cos(phi) - vi / sqrt(vx(2001,1) ^ 2 + vy(2001,1) ^ 2));

     deltaV = sqrt(1 + xi ^ 2) * abs(vx(2001,1) + vi);
     ob = vf - deltaV;
     end

```

Listing 1: 초깃값 문제를 수행하고 Objective function을 계산하는 MATLAB 코드

```
function [x, y, err] = DDGD(obj_func, x0, y0, gg1, gam, xmin1, xmax1,    ymin1, ymax1)
global gg xmax xmin ymax ymin iter;
gg = gg1; %grid gap
xmax = xmax1;
xmin = xmin1;
ymax = ymax1;
ymin = ymin1;

%Discretize
for x = xmin:gg:xmax
for y0 = ymin:gg:ymax
[xi, yi] = ind(x, y0);
data(xi, yi) = obj_func(x, y0);
end
end
for x = xmin + gg:gg:xmax - gg
for y0 = ymin + gg:gg:ymax - gg
[xi, yi] = ind(x, y0);
grad_x(xi, yi) = (data(xi + 1, yi) - data(xi - 1, yi)) / (2 * gg);
grad_y(xi, yi) = (data(xi, yi + 1) - data(xi, yi - 1)) / (2 * gg);
end
end

p(1, 1) = x0;
p(1, 2) = y0;

%Gradient Descent
err = 1;
i = 1;
while 1
if i > 1000 || gg < 10^(-6)
break;
end
[xi, yi] = ind(p(i, 1), p(i, 2));
p(i + 1, 1) = p(i, 1) - round(gam * grad_x(xi, yi) / gg) * gg;
p(i + 1, 2) = p(i, 2) - round(gam * grad_y(xi, yi) / gg) * gg;
err = sqrt((p(i + 1, 1) - p(i , 1))^2 + (p(i + 1, 2) - p(i , 2))^2);

 i = i + 1;
```

```
    %Make new grid
    if (p(i, 1) <= xmin || p(i, 1) >= xmax || p(i, 2) <= ymin || p(i, 2) >= ymax || err == 0)
    ngg = abs(min([grad_x(xi, yi), grad_y(xi, yi)])) * gam / 2;
    fprintf("New grid gap = %.7f (%d iterations)\n", ngg, i);
    [x, y, err] = DDGD(obj_func, p(i, 1), p(i, 2), ngg, gam, p(i, 1) - 100 * ngg, p(i, 1) + 100 *
ngg, p(i, 2) - 100 * ngg, p(i, 2) + 100 * ngg);
    iter = iter + i;
    return;
    end
    end

    x = p(i, 1);
    y = p(i, 2);
    fprintf("Reached target in %d iterations\n", iter);
    fprintf("Final point is (x=%.5f, y=%.5f, f(x, y)=%.5f) with error %.7f\n",x , y0, obj_func(x,
y0), err);
    end

    function [i, j] = ind(x, y) %return index
    global gg xmax xmin ymax ymin;
    i = round((x - xmin) / gg + 1);
    j = round((y - ymin) / gg + 1);
    end

```

Listing 2: DDGD MATLAB 코드

■ 참고문헌

[1] Duchêne, Gaspard; Kraus, Adam (2013). Stellar Multiplicity. University of California Berkeley.

[2] Curtis, Howard D. (2020). Orbital Mechanics for Engineering Students (4th Edition).

[3] Boyd, John P. (2007). Rootfinding for a transcendental equation without a first guess: Polynomialization of Kepler's equation through Chebyshev polynomial equation of the sine. Applied Numerical Mathematics. 57 (1): 12–18.

[4] Kot, Mark (2014). A First Course in the Calculus of Variations.

[5] Barzilai, Jonathan; Borwein, Jonathan M. (1988). Two-Point Step Size Gradient Methods. IMA Journal of Numerical Analysis. 8 (1): 141–148.