J. Space Technol. Appl. 2(2), 67-80 (2022)
https://doi.org/10.52912/jsta.2022.2.2.67

우주기술과 응용
Journal of Space Technology and Applications
pISSN 2765-7469 eISSN 2799-3213

연구논문

정지궤도 기반 발사체 비행 궤적 추정시스템의 시뮬레이터 개발

명환춘†

한국항공우주연구원

Received: April 8, 2022
Revised: April 18, 2022
Accepted: April 24, 2022

†**Corresponding author :**
Hwan-Chun Myung
Tel : +82-42-860-2774
E-mail : mhc@kari.re.kr

ORCID
Hwan-Chun Myung
https://orcid.org/0000-0003-0848-6116

Simulator Development for GEO (Geostationary Orbit)-Based Launch Vehicle Flight Trajectory Prediction System

Hwan-Chun Myung

Korea Aerospace Research Institute, Daejeon 34133, Korea

요 약

최근의 우주개발기술 선진국들은 우주의 평화적 이용이라는 보편적 가치를 넘어서 자국의 이익을 극대화하기 위한 발판으로 우주의 전략적 활용에 더욱 더 집중하고 있으며, 조기경보 위성과 같이 지상에서 발사된 화염 정보를 이용하여 우주로 발사된 발사체의 실시간 감시와 궤적 추적 기능 등을 담당하는 위성들을 지속적으로 개발해 오고 있다. 본 연구에서는 이러한 조기경보 위성에서 발사체의 궤적을 실시간으로 추정할 수 있는 알고리즘을 진화연산이라는 인공지능 기법을 적용하여 제안하고, 이러한 비행 궤적 추정 알고리즘을 비행 궤적 추정 시스템의 시뮬레이터를 통하여 임의의 발사체 비행 궤적에 적용함으로써 제안된 방법의 성능과 특징을 입체적으로 확인하고자 한다.

Abstract

The missile early-warning satellite systems have been developed and upgraded by some space-developed nations, under the inevitable trend that the space is more strongly considered as another battle field than before. As the key function of such a satellite-based early warning system, the prediction algorithm of the missile flight trajectory is studied in the paper. In particular, the evolution computation, receiving broad attention in the artificial intelligence area, is applied to the proposed prediction method so that the global optimum-like solution is found avoiding disadvantage of the previous non-linear optimization search tools. Moreover, using the prediction simulator of the launch vehicle flight trajectory which is newly developed in C# and Python, the paper verifies the performance and the feature of the proposed algorithm.

핵심어 : 정지 궤도, 우주 발사체, 비행 궤적
Keywords : GEO (geostationary orbit), LV (launch vehicle), flight trajectory

1. 서론

1950년대부터 지상의 미사일 등과 같은 열원의 탐지를 우주에서 수행하기 위하여, 미국과 러시아(구 소련)를 중심으로 군사적 활용 관점에서 조기경보위성의 개발이 지속적으로 진행

되어 왔으며, 최근에는 중국을 포함한 여러 다른 우주 선진국들에서도 관련 위성과 기술 개발에 많은 관심을 보이고 있다[1]. 조기경보위성 시스템의 주된 임무는 지상에서 발생하는 열원의 적외선 정보를 실시간으로 탐지하는 것으로서, 정지궤도 및 타원궤도 등이 주로 이러한 목적을 위하여 활용되고 있다. 최근에는 이러한 열원의 탐지라는 단순한 임무를 넘어서 열원의 궤적을 추적하기 위한 임무도 함께 고려되면서, 저궤도 위성군에 의한 열원 궤적의 추적 기능도 조기경보위성 시스템의 중요한 임무로서 새롭게 포함되고 있다[2,3]. 일반적으로 정지궤도에 위치한 조기경보위성은 열원의 조기탐지와 함께 열원 정보의 지속 시간까지 획득된 적외선 정보를 이용하여 예상되는 비행 궤적과 탄착점을 추정할 수 있으며, 이러한 기능은 복수의 정지궤도 위성군을 통하여 비행 궤적과 탄착지점의 오차 성능을 보다 더 최적화할 수 있는 것으로 알려져 있다[4-6]. 그러나, 조기경보위성과 관련된 기술은 각국의 보안유지로 인하여 관련된 연구가 거의 알려져 있지 않으며, 단지 지상 레이더 기반의 발사체에 대한 궤적 연구가 일부분 공개되어 오고 있는 상황이다. 기존의 지상 레이더 또는 탄도미사일 추적센서 기반의 연구들에서는 정의된 우도함수(likelihood function)의 최대우도추정 방법(maximum likelihood estimation) 등을 통해 최적의 발사체 파라미터 벡터를 실시간으로 도출하는 방법을 이용하여 발사지점과 탄착지점, 비행 궤적 등을 추정하는 방법들에 대한 연구 등이 수행되어 왔다[7]. 본 연구에서는 이처럼 관련 연구가 거의 전무한 조기경보위성의 발사체 궤적 추정 방법을, 지상 추적센서 기반에서 널리 활용되고 있는 전형적인 비선형 최적화 방법과 진화연산으로 알려져 있는 인공지능 기법을 순차적으로 함께 적용하여 기존의 지상 추적센서 기반 연구결과들에 비하여 국부 최소값 문제를 보다 더 효과적으로 극복함으로써, 시뮬레이터의 정지궤도 환경에서 실시간으로 구현하였다는 점에서 연구결과의 의미를 확인할 수 있다.

본 연구는 전체적으로 정지궤도를 기반으로 하여 발사체의 비행 궤적을 추정하는 시스템의 시뮬레이터 개발을 주된 내용으로 하고 있다. 이러한 시뮬레이터는 적용될 비행 궤적 추정 알고리즘의 성능을 보다 더 입체적이고 종합적으로 확인하고 분석할 수 있도록 활용될 수 있다는 이점이 있으며, 실제 비행 궤적 추정 시스템의 개발을 위한 초기 모델의 역할도 수행할 수 있을 것으로 기대된다. 이러한 시뮬레이터는 크게 발사체 비행 궤적을 모사하는 3D 그래픽 부분과 모사된 비행 궤적을 실시간으로 추정하는 알고리즘 부분으로 나뉘어지는데, 각각의 그래픽 부분과 알고리즘 부분 구현에 가장 적합한 컴퓨터 언어들(C#, Python)을 함께 적용하여 개발되었다. 3D 그래픽은 C#의 OpenTK 라이브러리를 활용하여 사실적인 GUI(graphic user interface)를 구현함으로써, 비행 궤적을 다양한 각도에서 입체적으로 확인할 수 있도록 하였으며, 또한 Python의 Basemap 기능을 적용하여 발사체 궤적의 지리적 정보도 함께 제공할 수 있도록 하였다[8,9]. 비행 궤적 추정 알고리즘은 진화연산 방식과 BFGS(Broyden-Fletcher-Goldfarb-Shanno)의 준-뉴톤방식(Quasi-Newton method)을 순차적으로 결합하여 최적화 문제에서 발생하기 쉬운 국부 최소값 문제를 보완하면서 발사체의 최적화된 비행 궤적을 추정할 수 있는 방법을 적용하고, 발사체 관측 위성의 개수와 발사체 화염의 관측 방식에 따른 추정 성능의 차이점도 함께 분석한다.

본 논문은 2장에서 시뮬레이터의 구성과 구현 방법을 설명하고, 3장에서는 제안된 비행 궤적 추정 알고리즘과 발사체의 모델을 제시한다. 마지막으로 4장에서는 제안된 알고리즘을 적용한 시뮬레이터의 구현 결과를 확인하고, 시뮬레이션 결과를 통하여 제안된 알고리즘의 성능 분석을 수행한다.

2. 시뮬레이터의 개요

2.1 시뮬레이터의 구성

본 연구에서 개발한 시뮬레이터는 크게 사용자의 조작 편의성을 위한 GUI 부분과 비행 궤적 추정 알고리즘 부분으로 나뉘어진다. GUI는 사용자의 명령을 전달받고 시뮬레이션의 상황을 종합적이고 입체적으로 보여주기 위한 통합 환경을 의미하며, 발사체의 비행 궤적과 관련한 실시간 정보 제공 및 발사체의 자세와 비행 궤적의 움직임을 2D 그래프와 3D 그래픽을 활용하여 구현함으로써 사용자에게 보다 더 실제적인 정보를 제공하는 기능을 수행한다. 비행 궤적 추정 알고리즘은 발사체의 모델을 바탕으로 비행 궤적을 생성하는 부분과 정지궤도에서 관측된 화염 정보를 바탕으로 비행 궤적을 추정하는 부분으로 구성된다. 일반적으로 C# 언어는 GUI 구성과 3D 그래픽 구현에 있어서 다양하고 유용한 라이브러리들을 활용할 수 있다는 이점이 있는 반면에, Python 언어는 인공 지능 기법을 포함한 폭넓은 수치해석 라이브러리들을 제공함으로써 복잡한 연산과정을 간결하게 구현할 수 있다는 점에서 매우 효과적이라고 알려져 있다. 본 연구에서는 이러한 C#과 Python 언어들의 장점들을 극대화하기 위하여 C#에서 Python 언어와의 통합을 위하여 제공하고 있는 Pythonnet 라이브러리를 활용함으로써, GUI 부분의 구현을 위해서는 주로 C#을 사용하고, 알고리즘과 발사체 동역학의 모사 등과 같은 연산과정에서는 Python을 사용하여 시뮬레이터를 구현하였다. 또한, Python에서 제공하고 있는 Basemap 기능을 활용하여 발사체 궤적의 실제 지리 정보를 제공할 수 있도록 GUI 기능에 추가함으로써, 인터넷 연결을 통한 원격 지도 데이터와의 연동 없이도 자체적으로 내장된 지도 데이터를 이용하여 지표상의 실제 위치에 발사체의 비행 궤적을 투사하여 보여줄 수 있도록 하였다(Fig. 1).

2.2 GUI 의 구현

GUI는 Fig. 2에서와 같이 사용자의 명령을 수행하는 부분과 발사체의 자세와 상태 정보를 제공하는 부분, 발사체 궤적을 제공하는 세 부분으로 크게 나눌 수 있으며, 실제로 시뮬레이션이 구현된 모습은 Fig. 3에서 확인할 수 있다.

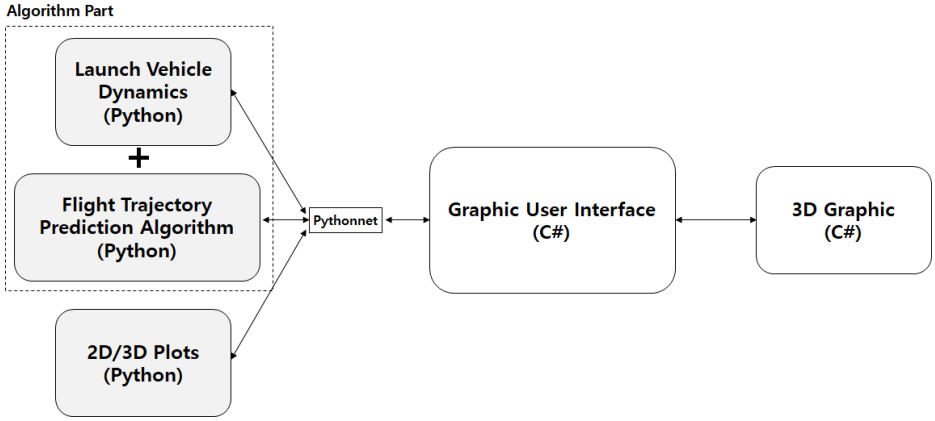


Fig. 1. Architecture of simulator software.

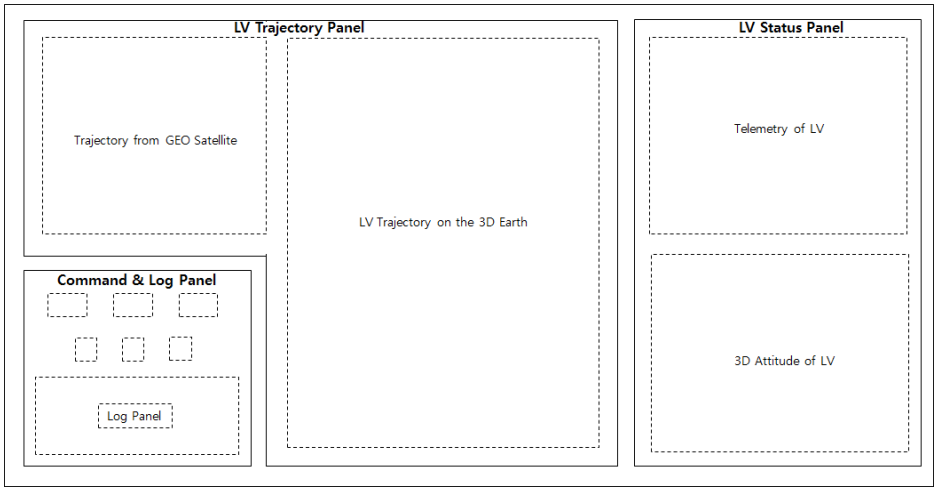


Fig. 2. GUI (graphic user interface) layout: GEO (geostationary orbit), LV (launch vehicle).

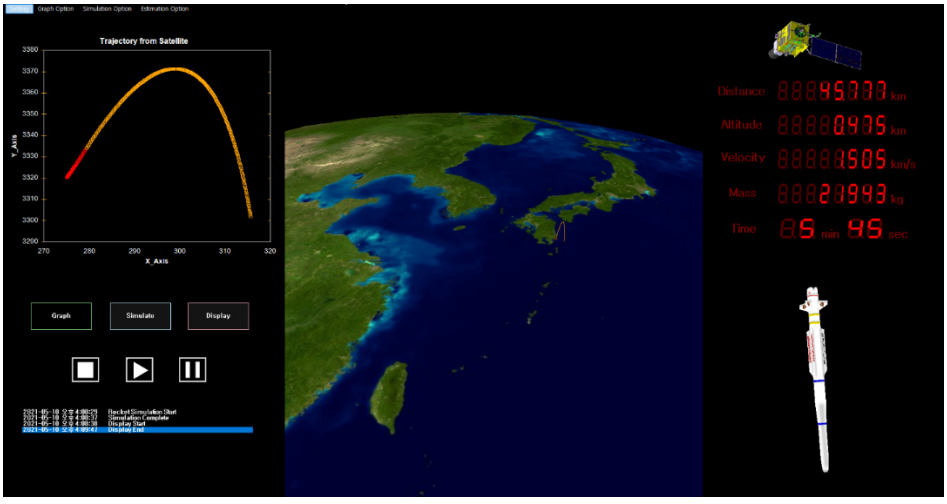
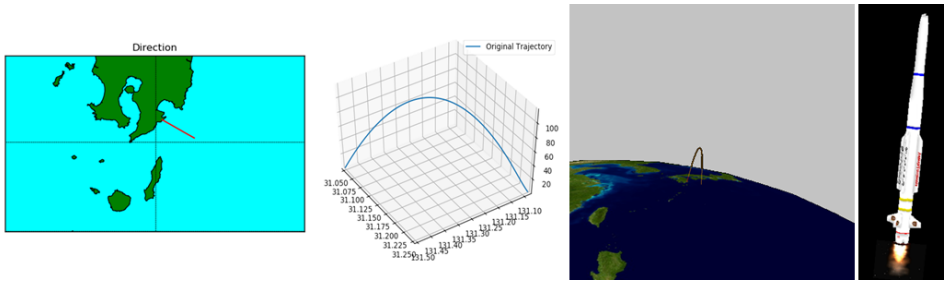


Fig. 3. Simulation example of GUI (graphic user interface).

명령을 수행하는 부분에서는 시뮬레이션의 수행과 결과 확인 등을 선택할 수 있으며, 추가 그래프 선택을 통해서 발사체의 상태정보와 궤적의 시간상의 변화를 별도의 2D/3D 그래프로 확인할 수 있다. Python에서 제공하는 Basemap을 적용한 2D 그래프에서는 발사체의 실제 궤적을 지도상에서 정확히 투영하여 보여줌으로써, 발사체의 이동방향과 발사 및 최종위치 등을 보다 더 실제적으로 확인할 수 있다. 발사체의 상태 정보는 거리, 고도, 속력, 질량, 비행 시간을 보여줄 수 있으며, 특히 발사체의 3D 그래픽을 활용하여 화염의 분사 유무와 자세 및 방향 등의 실시간 변화를 제공한다. 발사체 궤적 부분은 지구의 3D 그래픽 모델을 이용하여 입체적인 궤적의 움직임을 확인할 수 있도록 개발되었으며, 회전 및 확대/축소 등의 기능도 포함되었다. 이와 함께 정지궤도에서의 조기경보 탑재체 검출기에 투사되어 관측되는 발사체의 2D 궤적도 화염구간과 별도로 확인할 수 있게 함으로써, 발사체의 궤적 추적에 활용되는 투사된 2D 화염 데이터의 모습도 실시간 관측이 가능하도록 추가하였다(Fig. 3-4).



(a) (b) (c) (d)

Fig. 4. GUI (graphic user interface) examples: (a) basemap, (b) 3D trajectory with altitude, latitude, and longitude, (c) 3D trajectory on the Earth, and (d) 3D attitude of LV (launch vehicle) with flame.

3. 발사체 모델과 최적화 알고리즘

3.1 발사체 모델

본 연구에서 대상으로 하는 발사체 모델은 일본 우주과학연구소(ISAS)가 1980년대에 개발하여 과학 위성 발사 등에 활용한 3단식 고체 연료 로켓 M-3S이다. 전체 길이는 약 23.8 m 이고, 중량은 48.7 t, 탑재 가능한 위성의 무게는 최대 300 kg 정도로 알려져 있다(Fig. 5) [10].

발사체의 궤적을 결정하는 초기 조건들로는 발사체 자체의 설계 제원은 물론 발사 조건 등 다양한 요소들이 관련되어 있는데, 본 연구에서는 크게 9개의 초기 조건들을 발사체의 궤적을 추정하기 위한 주요 파라미터들로 고려하였으며, Table 1에서는 이러한 파라미터들과 설정값들을 보여주고 있다.

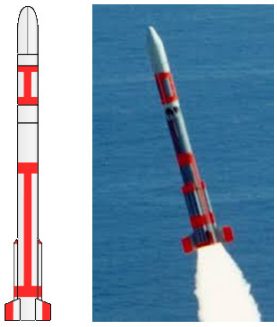


Fig. 5. M-3S rocket.

Table 1. Main parameters and values of launch vehicle [9]: S.I.(specific impulse), L.C.(lift coefficient)

Parameter	Initial value	Parameter	Initial value	Parameter	Initial value
S.I. (θ_1)	255 N-sec/kg	L.C. (θ_2)	3.5	Pitch (θ_3)	85 deg
Yaw (θ_4)	120 deg	Latitude (θ_5)	31.251 deg	Longitude (θ_6)	131.08 deg
Altitude (θ_7)	194 m	Mass (θ_8)	45,247.4 kg	Thrust force (θ_9)	1,147,000 N

발사체의 궤적을 결정하는 초기조건들로 구성된 파라미터 벡터는 다음과 같이 정의될 수 있으며,

$$\boldsymbol{\Theta} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9]'. \quad (1)$$

여기서 θ_i의 정의는 Table 1과 동일하다. 발사체 모델의 동역학을 구현하기 위하여 사용된 상태 변수는 모두 14개로서, 발사체의 질량(m)을 포함하여 위치($\boldsymbol{P}$)와 속도($\dot{\boldsymbol{P}}$), 쿼터니온($\boldsymbol{Q}$), 회전축(roll/pitch/yaw)을 중심으로 하는 각속도($\boldsymbol{\Omega}$) 등으로 구성되었다. 이러한 상태 벡터는 오일러 적분 방식을 이용하여 상태값의 변동량을 계산함으로써, 시간에 따른 발사체의 3차원 궤적을 생성하도록 사용되었다. 초기조건을 바탕으로 발사체의 궤적을 나타내는 상태 벡터는

$$\boldsymbol{\Psi}(t, \boldsymbol{\Theta}) = [\ \mathrm{m}(\mathrm{t}, \boldsymbol{\Theta}), \boldsymbol{P}(\mathrm{t}, \boldsymbol{\Theta}), \dot{\boldsymbol{P}}(\mathrm{t}, \boldsymbol{\Theta}), \boldsymbol{Q}(\mathrm{t}, \boldsymbol{\Theta}),\ \boldsymbol{\Omega}(\mathrm{t}, \boldsymbol{\Theta})]',$$
$$\mathrm{m} \in \boldsymbol{\mathcal{R}}, \boldsymbol{P} \in \boldsymbol{\mathcal{R}}^3, \dot{\boldsymbol{P}} \in \boldsymbol{\mathcal{R}}^3, \boldsymbol{Q} \in \boldsymbol{\mathcal{R}}^4,\ \boldsymbol{\Omega} \in \boldsymbol{\mathcal{R}}^3, \boldsymbol{\Psi} \in \boldsymbol{\mathcal{R}}^{14}, \quad (2)$$

와 같이 나타낼 수 있다. 고려된 발사체의 초기 파라미터들은 크게 발사 위치(latitude, longitude, altitude)와 발사 방향(pitch, yaw), 발사체의 특성(mass, specific impulse, lift coefficient, thrust force) 등으로 구분될 수 있으며, 이러한 초기 파라미터들의 차이는 각 파라미터의 특성에 따라서 다양한 정도로 비행 궤적의 변경을 초래하게 된다. 53초로 설정된 연소 기간동안 발사체는 추진제를 전부 소모하면서 21,943 kg까지 무게가 감소하게 되고, 발사체의 위치와 방향은 최종지점까지 지속적으로 수정되면서 발사체의 전체 비행 궤적을 형성한다. 5분 45초의 전체 비행 시간 동안 발사체는 최대 116 km의 고도까지 상승하게 되며, 최대 이동 거리는 약 45 km에 이른다(Fig. 6).

3.2 비행 궤적 추정 알고리즘

본 연구는 비행 궤적을 추정하는 데 있어서 인공지능 및 최적화 기법의 적용과 상대 정밀도 향상 방법에 중점을 두고 있기 때문에, 실제 비행 궤적 관측 상황에서 발생할 수 있는 관측 위성의 지향오차(pointing error) 및 검출기의 공간해상도 제약에 따른 양자화 오류(quantization

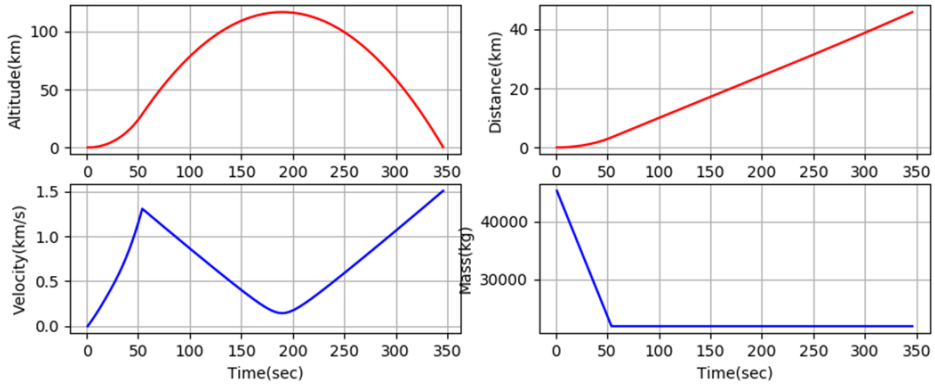


Fig. 6. Flight characteristics of LV (launch vehicle).

error) 등은 추후 과제에서 절대 정밀도 개선 관점에서 검토될 예정이다. 발사체의 관측 벡터는 이전의 상태 벡터와는 달리 발사체의 화염을 관측하는 정지궤도 위성(궤도: ϕ)의 광학 검출기 관점에서 다음과 같이 표현될 수 있다.

$$\boldsymbol{\Psi}_{\boldsymbol{D1}}(\boldsymbol{t},\boldsymbol{\theta},\boldsymbol{\phi}) = [\, \boldsymbol{x_D}(\mathbf{t},\boldsymbol{\theta},\boldsymbol{\phi}), \boldsymbol{y_D}(\mathbf{t},\boldsymbol{\theta},\boldsymbol{\phi})]', \quad \boldsymbol{\Psi}_{\boldsymbol{D1}} \in \boldsymbol{\mathcal{R}^2} \ . \tag{3}$$

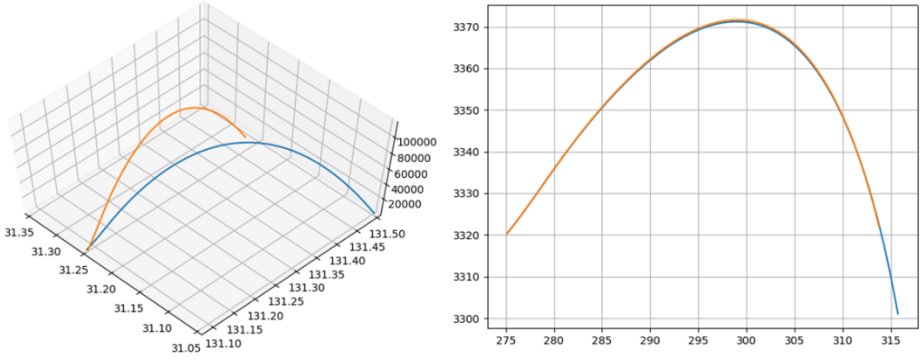
여기서, x_D와 y_D는 각각 2차원 검출기에서 관측된 화염의 위치를 나타낸다. 이 때, 추정된 초기조건들의 파라미터 벡터($\tilde{\boldsymbol{\theta}}$)를 이용한 성능평가 함수는 관측된 발사체의 실제 관측 벡터($\boldsymbol{\Psi}_{D1}(t,\boldsymbol{\theta},\phi)$)와 알고리즘에 의하여 추정되어 도출된 관측 벡터($\tilde{\boldsymbol{\Psi}}_{D1}(t,\tilde{\boldsymbol{\theta}},\phi)$)를 이용하여 다음과 같이 정의한다.

$$f_1(T_F,\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}},\phi) = \sqrt{\sum_{t=0}^{T_F}\{\boldsymbol{\Psi}_{D1}(t,\boldsymbol{\theta},\phi) \ - \ \tilde{\boldsymbol{\Psi}}_{D1}(t,\tilde{\boldsymbol{\theta}},\phi)\}^2}, t = 0, \Delta T, 2\Delta T, \cdots, T_F. \tag{4}$$

여기서 ΔT는 광학 검출기의 관측 주기를 나타내며, T_F는 발사체의 화염이 지속되는 시간을 나타낸다. 이 때, 정의된 성능평가 함수를 이용한 비행 궤적 추정의 최적화 문제는

$$\tilde{\boldsymbol{\theta}}_{opt} \ = \ \underset{\tilde{\boldsymbol{\theta}}}{\mathbf{arg\ min}}\ f_1(T_F,\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}},\phi) \ , \tag{5}$$

와 같이 표현될 수 있다. 일반적으로 한 대의 정지궤도 위성의 광학 탑재체에서 발사체의 화염 궤적을 관찰할 경우에는, 실제적인 3차원 궤적이 아닌 광학 탑재체의 검출기에 투사된 2차원 궤적만을 관측할 수밖에 없다. Fig. 7에서와 같이 이러한 정보 손실은 결과적으로 발사체 비행 궤적 추정의 오류를 초래할 수밖에 없게 되는데, 이러한 이유로 인하여 손실된 정보를 보정하기 위하여 두 대 이상의 정지궤도 위성 활용이 필수적인 것으로 인식되고 있다. 본 연구에서는 이러한 단일 위성관측의 한계를 개선하기 위해서 발사체의 화염 위치 정보와 함께 속도 정보도 함께 고려하는 방법을 적용하고자 한다. 속도 정보와 같은 추가적인 정보의 고려



(a) (b)

Fig. 7. Limitation of 3D trajectory prediction (case of a single GEO satellite): (a) two different 3D trajectories and (b) 3D trajectories similarly projected on an 2D optical detector in a GEO satellite.

는 광학 탑재체 검출기에서 궤적상으로 동일하게 투사되는 3차원 비행 궤적의 구분을 좀 더 세밀하게 수행하는 데 있어서 도움을 줄 수 있을 것으로 예상할 수 있다.

속도정보를 고려하는 경우에는 검출기 관점의 관측벡터와 성능평가 함수를 다음과 같이 각각 재정의할 수 있다.

$$\boldsymbol{\Psi}_{D2}(t,\boldsymbol{\theta},\phi) = [x_D(\mathrm{t},\boldsymbol{\theta},\phi), y_D(\mathrm{t},\boldsymbol{\theta},\phi), \dot{x}_D(\mathrm{t},\boldsymbol{\theta},\phi), \dot{y}_D(\mathrm{t},\boldsymbol{\theta},\phi)]' \ , \ \boldsymbol{\Psi}_{D2} \in \boldsymbol{\mathcal{R}}^4, \quad (6)$$

$$f_2(T_F,\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}},\phi) = \sqrt{\sum_{t=0}^{T_F}\{\boldsymbol{\Psi}_{D2}(t,\boldsymbol{\theta},\phi\ ) \ - \ \widetilde{\boldsymbol{\Psi}}_{D2}(t,\widetilde{\boldsymbol{\theta}},\phi)\}^2}. \quad (7)$$

여기서, $\dot{x}_D$와 $\dot{y}_D$는 각각 2차원 검출기에서 관측된 화염의 속도를 나타낸다. 그러나, 이러한 추가적인 정보의 고려도 단일위성에 의한 정보손실의 한계를 완벽히 극복하는 것은 쉽지 않다. 따라서, 요구되는 비행 궤적의 정확도가 매우 정밀한 경우에는 복수위성에 의한 발사체 궤적 추정 방법을 고려할 수밖에 없으며, 이 경우에도 속도 정보와 같은 추가적인 정보의 고려가 비행 궤적을 추정하는 데 있어서 정확도 향상에 동일한 기여를 할 수 있을 것으로 쉽게 예상할 수 있다. 그러나, 복수 위성에 의한 비행 궤적 추정 방법은 정확도 측면에서는 매우 유리한 측면이 있지만, 단일 위성 관측에 비하여 처리해야 할 데이터 양의 증가가 최적화 과정의 소요시간 증가로 이어지게 된다는 문제점이 있을 수 있다. 이러한 추정 소요시간의 증가는 발사체 비행 궤적의 추정은 물론 최종지점의 신속한 예측의 필요성 관점에서도 또 다른 제약 사항으로 고려될 수 있다. 그러므로, 요구되는 비행 궤적 추정의 정확도와 함께 추정에 소요되는 처리시간 등의 주요 요구사항들을 동시에 고려하여 적절한 관측 방식을 선정하는 것이 중요하다. 복수 위성(n개 위성의 궤도: $\phi_1,\cdots,\phi_n$)에 의한 궤적 추정 방법 중에서 속도 정보 활용의 유무에 따라서 다음과 같이 검출기 관점의 발사체 관측 벡터와 성능평가 함수들을 이전과 유사하게 정의할 수 있다. 속도 정보를 고려하지 않는 경우,

$$\boldsymbol{\Psi}_{D3,i}(t,\boldsymbol{\theta},\phi_i) = [x_D(\mathrm{t},\boldsymbol{\theta},\phi_i), y_D(\mathrm{t},\boldsymbol{\theta},\phi_i)]', \quad \mathbf{i} = 1,\cdots,n, \quad \boldsymbol{\Psi}_{D3} \in \boldsymbol{\mathcal{R}}^2, \quad (8)$$

$$f_3(T_F,\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}},\boldsymbol{\Phi}) = \sqrt{\sum_{i=1}^{n}\sum_{t=0}^{T_F}\{\boldsymbol{\Psi}_{D3,i}(t,\boldsymbol{\theta},\phi_i\ ) \ - \ \widetilde{\boldsymbol{\Psi}}_{D3,i}(t,\widetilde{\boldsymbol{\theta}},\phi_i)\}^2}, \quad \boldsymbol{\Phi} = [\phi_1,\cdots,\phi_n]', \quad (9)$$

그리고, 속도 정보를 고려하는 경우,

$$\Psi_{D4,i}(t,\boldsymbol{\theta},\phi_i) = [x_D(\mathrm{t},\boldsymbol{\theta},\phi_i), y_D(\mathrm{t},\boldsymbol{\theta},\phi_i), \dot{x}_D(\mathrm{t},\boldsymbol{\theta},\phi_i), \dot{y}_D(\mathrm{t},\boldsymbol{\theta},\phi_i)]', \quad \boldsymbol{\Psi}_{D4} \in \boldsymbol{\mathcal{R}}^4, \quad (10)$$

$$f_4(T_F,\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}},\boldsymbol{\Phi}) = \sqrt{\sum_{i=1}^{n}\sum_{t=0}^{T_F}\{\boldsymbol{\Psi}_{D4,i}(t,\boldsymbol{\theta},\phi_i\ ) \ - \ \widetilde{\boldsymbol{\Psi}}_{D4,i}(t,\widetilde{\boldsymbol{\theta}},\phi_i)\}^2}, \quad (11)$$

와 같이 나타낼 수 있다.

비행 궤적 추정 알고리즘의 핵심은 발사체 모델에서 고려되고 있는 주요 파라미터들의 초기 조건을 화염 정보만으로 추정하는 것으로서, 본 연구에서는 기존의 최적화 기법과 함께 진화연산이라는 인공지능 기법을 병용함으로써 비행 궤적 추정 알고리즘의 초기 설정값에 대한

지나친 의존도를 완화하고자 한다. Fig. 8에서 확인할 수 있는 바와 같이, Phase I에서는 진화 연산 중에서 ES(evolution strategy) 방법을 적용하여, 9개의 파라미터들로 구성된 유전자 공간의 다중(multi-wide) 검색을 수행하게 되며, 이러한 다중검색을 통하여 최적의 파라미터 설정값들의 존재 가능성이 가장 높은 영역을 발견하게 된다. 본 연구에서는 최적의 파라미터 값들을 기준으로 각 파라미터 값의 30%-170%로 확장된 검색영역을 유전자 공간대상으로 선정하였으며, 50개의 부모 유전자로부터 300개의 자손 유전자를 매 세대마다 생성하였고, 60%의 변이확률과 30%의 교배확률을 적용하여 10세대 이후의 최적 결과를 Phase II의 초기값으로 사용하였다. Phase II에서는 비선형 최적화 문제에서 널리 사용되고 있는 BFGS 알고리즘을 적용하여 Phase I에서 발견된 최적화 대상 영역에서 9개의 최적화된 파라미터 값들을 빠르게 도출할 수 있도록 하였다. 이러한 인공지능 기법과 기존의 최적화 방법의 결합 방식은 인공지는 분야에서 오랫동안 폭넓게 연구를 수행해 오고 있으며[11-13], Table 2와 3은 본 연구에서 적용한 알고리즘의 구체적인 개요를 간략히 보여주고 있다.

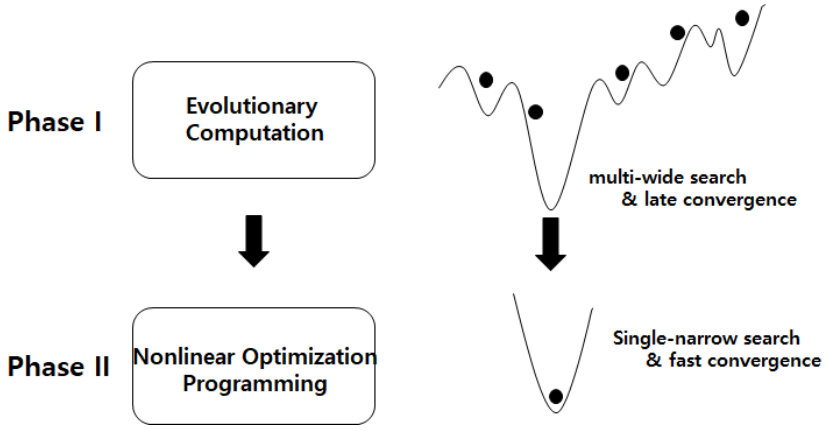


Fig. 8. Multi-phase approach for parameter optimization.

Table 2. Phase I algorithm

Phase I algorithm: *Evolutionary computation*
Step 1: Initialize population → $\tilde{\boldsymbol{\theta}}_{j,0} \in \boldsymbol{\mathcal{R}}^9,\ j(parent\ inidicator) = 1, \cdots, 50,$
Step 2: Evaluate population → $\mathbf{Eval}_{\forall j}(f_{\#}(T_F, \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{j,0}, \boldsymbol{\Phi})),\ \# \in \{1,2,3,4\}$
Step 3: Evolve generation (generation limit: g ≤ 10)
For each generation ($\tilde{\boldsymbol{\theta}}_{j,g}$)
- Select parents
- Produce offspring ($P_{crossover} = 0.3, P_{mutation} = 0.6,\ number\ of\ offspring\ =\ 300$)
- Evaluate parents and offspring
- Select the next generation
Step 4: Find the best solution for the optimal region → $\tilde{\boldsymbol{\theta}}_{p1,10}$
Step 5: Implement Phase II Algorithm for the optical solution, as an initial value of $\tilde{\boldsymbol{\theta}}_{p1,10}$

Table 3. Phase II algorithm

Phase II algorithm: *BFGS (Broyden-Fletcher-Goldfarb-Shanno)*
Step 1: Set the initial value($\tilde{\boldsymbol{\theta}}_0$) with $\tilde{\boldsymbol{\theta}}_{p1,10}$ (without Phase I, randomize the initial value)
Step 2: Iterate nonlinear optimization procedure (Convergence limit: Tolerance ≤ 0.00001)
– Obtain the search direction ($\boldsymbol{q}_k$) with Hessian matrix($\boldsymbol{H}_k$, k: iteration number): $\boldsymbol{q}_k = -\boldsymbol{H}_{\mathrm{k}} \nabla f_{\#}(T_F, \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_k, \boldsymbol{\Phi}),\ \ \boldsymbol{q}_k \in \mathcal{R}^9, \boldsymbol{H}_k \in \mathcal{R}^{9x9},\ \nabla f_{\#} \in \mathcal{R}^9$
– Calculate the step size(α_k) by using golden section search technique: $\alpha_k = \underset{\alpha}{\mathbf{arg\ min}}\ f_{\#}\ (T_F, \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_k + \alpha \boldsymbol{q}_k, \boldsymbol{\Phi})$
– Update new solution: $\tilde{\boldsymbol{\theta}}_{k+1} = \tilde{\boldsymbol{\theta}}_k + \alpha_k \boldsymbol{q}_k$
– Update Hessian matrix with the difference of gradients: $\boldsymbol{H}_{\mathrm{k+1}} \leftarrow \boldsymbol{H}_{\mathrm{k}},\ \alpha_k \boldsymbol{q}_k,\ \nabla f_{\#}(T_F, \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{k+1}, \boldsymbol{\Phi}) - \nabla f_{\#}(T_F, \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_k, \boldsymbol{\Phi})$
Step 3: Find the optimal solution → $\tilde{\boldsymbol{\theta}}_{opt} = \tilde{\boldsymbol{\theta}}_{p2}$

4. 시뮬레이션 결과와 분석

발사체 비행 궤적의 추정 시뮬레이션은 Fig. 3에서와 같이 본 연구에서 개발한 시뮬레이터를 통해서 수행되었으며, 위성 개수 및 관측 방법의 선택 등은 GUI의 상단 메뉴에서 선택하여 변경할 수 있도록 하였다. 발사체의 화염 구간 이후부터 추정 알고리즘이 적용되어 추정이 완료된 이후에, 추정된 비행 궤적이 현재 진행 중인 발사체의 비행 궤적과 함께 GUI에 나타냄으로써, 발사체의 향후 비행 궤적은 물론 최종지점의 위치를 사전에 확인할 수 있도록 하였다. 정지궤도 위성들은 단수위성의 경우에 GK2-A/B 위성들이 위치하고 있는 128.2도(ϕ)를 고려하였고, 복수위성의 경우에는 동경 128.2도(ϕ_1)와 함께 미국의 SBIRS 조기경보위성(GEO-4)이 위치하고 있는 것으로 알려져 있는 동경 159.08도(ϕ_2)에 위치하는 것으로 가정하였다. 검출기의 관측주기는 일반적으로 알려진 조기경보위성의 성능을 고려하여 1초(ΔT)로 설정하였으며, 화염의 지속주기(T_F)는 M-3S 발사체를 고려하여 53초로 시뮬레이션을 수행하였다. 또한, 궤적 추정 정확도의 성능평가 지표로서 최종지점의 예측 정확도를 분석하였다.

4.1 최적화 알고리즘과 성능 비교

진화연산 기법을 적용한 최적화 알고리즘의 효과를 확인하기 위하여 진화연산 방식의 적용 유무(w/o Evolution Computation: Phase II, w/ Evolution Computation: Phase I + Phase II)에 따른 비행 궤적 추정시험을 수행하였으며, Fig. 9-10은 이러한 비교 성능시험의 결과를 보여주고 있다. 모든 시뮬레이션은 10번씩 임의의 초기 변수값들을 적용하여 반복적으로 수행되었으며, 한 개의 정지궤도 위성과 위치 정보만을 사용하는 것을 가정하였다. 9개 변수들의 값은 편의를 위하여 물리적 특성에 상관없이 모두 '3'이라는 동일한 값으로 정규화되도록 조정되었다. Fig. 9의 왼쪽과 오른쪽은 진화연산 방식을 적용하지 않은 경우와 적용한 경우에 각각 최적화된 9개 변수값들의 분포도를 보여주고 있다. 진화연산에 의한 최적화 방법이 고려되지 않는 경우에는 초기값에 따라서 최적화된 변수값들의 변이가 매우 크다는 사실을 알

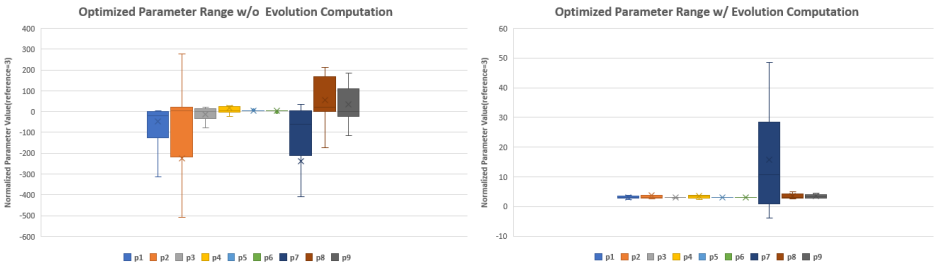


Fig. 9. Comparison of optimized parameter values (w/o and w/ evolution computation, average of 10 iterations): p1 = specific impulse, p2 = lift coefficient, p3 = pitch, p4 = yaw, p5 = latitude, p6 = longitude, p7 = altitude, p8 = mass, p9 = thrust force.

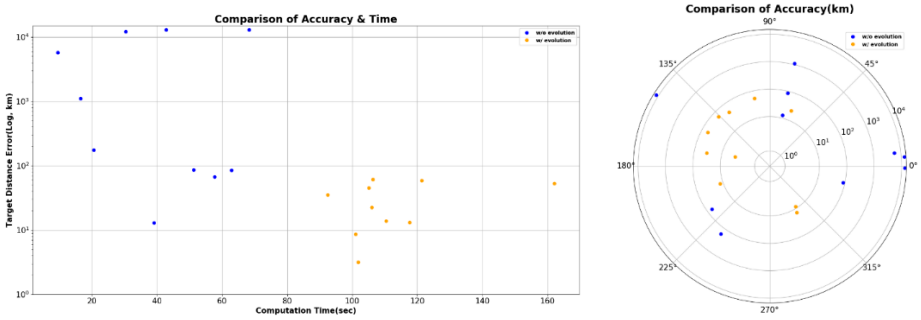


Fig. 10. Comparison of accuracy & time (w/o: blue and w/: orange).

수 있으며, 반면에 진화연산을 적용한 방법에서는 대부분의 변수들이 상대적으로 작은 변이 분포도 안에서 추정되고 있음을 확인할 수 있다. Fig. 10에서는 이러한 두 가지 방법들에 의하여 추정된 최종지점의 위치를 시간과 거리 관점에서 비교하고 있다. 진화연산을 적용하지 않은 경우에는 상대적으로 매우 빠른 추정시간 이내에서 결과를 얻을 수 있지만, 추정된 최종지점의 정확도 분포가 매우 크다는 특징이 있는 반면에, 진화연산을 적용한 방법에서는 더 오랜 추정시간이 요구되지만 정확도의 분포가 상대적으로 일정한 범위 안(<100 km)에서 안정적으로 도출될 수 있음을 알 수 있다.

4.2 관측 방식과 성능 비교

관측 방식에 따른 시뮬레이션은 이전의 4.1에서와 동일한 방식과 절차를 통해서 수행되었으며, 본 연구에서는 4가지(Case 1: 단일위성 + 위치정보 → f_1 적용, Case 2: 단일위성 + 위치/속도정보 → f_2 적용, Case 3: 복수위성 + 위치정보 → f_3 적용, Case 4: 복수위성 + 위치/속도정보 → f_4 적용)의 서로 다른 관측 방식을 시뮬레이션에 적용하였다. Case 1의 경우에는 특별히 발사체 엔진의 주요 특성인 양력계수(lift coefficient, p2)와 발사 환경의 주요 요소인 고도(altitude, p5) 변수의 추정에서 상대적으로 큰 변수값 오차분포를 보이고 있음을 Fig. 11에서 알 수 있으며, 이러한 변수들의 오차가 약 3.17-61.06 km 정도의 최종지점 오차를 유도한 것으로 분석된다. 이와 달리, 위치정보와 함께 속도정보도 활용하는 Case 2의 경우에는 양력계수의 추정에서 향상된 결과를 보여준 반면에, 고도 변수의 추정에서는 약간 더 큰 오차분포를 보여주고 있으며, 결과적으로는 Case 1의 경우보다 더 우수한 0.21-1.06 km 정도의 최

종지점 오차를 보여주고 있다. 이러한 결과를 바탕으로 고도 변수보다는 발사체 자체 특성인 양력계수의 특성이 최종지점의 정확도에 더 많은 영향을 미치는 것으로 판단된다. Case 3의 경우는 발사 방향(yaw, p4)을 결정하는 요소의 오차분포가 다른 관측방식의 오차분포보다 상대적으로 큰 결과를 보이고 있으며, 이러한 큰 변수값 오차의 경우에는 최종지점 오차 분포에서도 다른 추정값들에 비하여 큰 최종지점 오차를 보이고 있음을 Fig. 12에서 확인할 수 있다(왼쪽 그림, 초록색 오른쪽 끝 두 지점). 그럼에도 불구하고, Case 3은 5.07–455.11 m 정도의 최종지점 정확도를 보임으로써 복수위성에 의한 궤적 추정 정확도의 향상된 특성을 보여주고 있음을 알 수 있다. 마지막으로 Case 4는 변수값 추정에서 가장 우수한 결과를 보여주고 있음을 Fig. 11에서 확인할 수 있지만, 다른 변수들의 추정 정확도에 비해서 발사 방향(yaw, p4) 변수의 최적화에는 여전히 상대적으로 큰 추정 오차가 존재하고 있음을 알 수 있다. Case 4는 고려된 여러 관측 방식 중에서 가장 우수한 최종지점 관측 정확도를 보여주고 있으며, 그 범위는 약 0.09–78.75 m 정도이다. 결과적으로 Fig. 11과 Fig. 12에서 확인할 수 있는 바와 같이, 궤적의 추적 성능은 단일위성보다는 2D 검출기에 의한 정보손실을 보완할 수 있는 복수

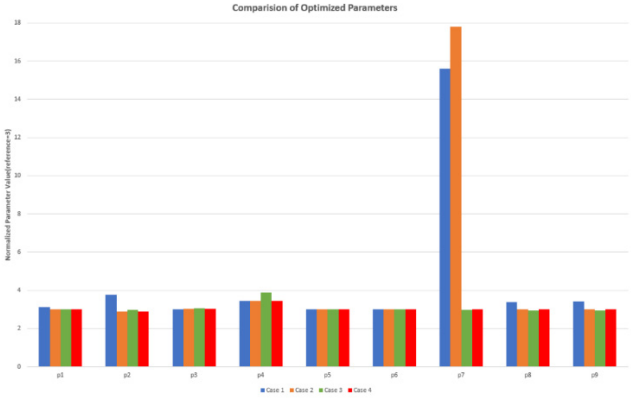


Fig. 11. Comparison of optimized parameter values: Case 1 (blue) = single satellite ($\phi = 128.2°E$) with position, Case 2 (orange) = single satellite with position + velocity, Case 3 (green) = two satellites ($\phi_1 = 128.2°E, \phi_2 = 159.08°E$) with position, Case 4 (red) = two satellites ($\phi_1 = 128.2°E, \phi_2 = 159.08°E$) with position + velocity.

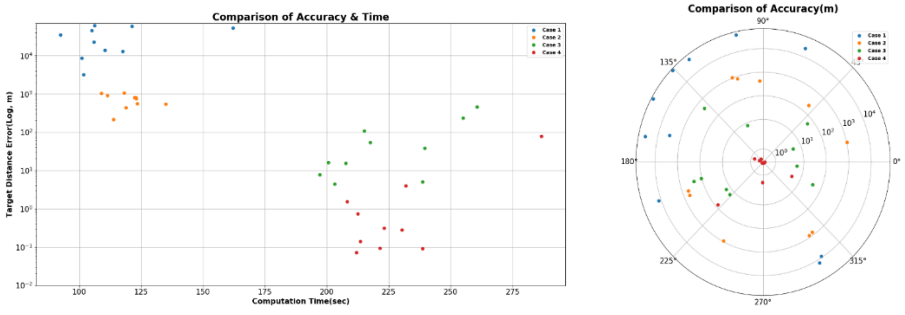


Fig. 12. Comparison of accuracy & time (Case 1:blue, Case 2: orange, Case 3: green, Case 4: red).

위성이 더 우수하고, 위치정보만을 사용하는 경우보다는 위치와 속도정보를 함께 사용하는 경우가 추정된 변수값들과 최종지점의 정확도 측면에서 모두 우위에 있음을 확인할 수 있다. 다만, 복수위성의 경우 2배 이상의 계산 복잡도 증가로 인한 추정 시간의 상대적인 지연(92-162 sec → 197-286 sec)은 불가피한 특성으로 고려될 수밖에 없으며, 구축하고자 하는 시스템의 요구특성(정확도 vs 관측시간)을 파악하여 위성의 개수를 결정하는 것이 필요하다. 속도정보의 활용과 관련해서는 추정시간의 증가와 같은 별도의 성능 저하 없이 추정 정확도의 증가를 구현할 수 있는 방법으로 고려될 수 있음을 알 수 있다.

5. 결론

본 연구를 통하여 발사체의 궤적을 실시간으로 추정하기 위한 정지궤도위성 기반의 발사체 비행 궤적 추정 시스템의 시뮬레이터를 개발하였다. 시뮬레이터는 적용된 궤적 추정 알고리즘의 입체적이고 효과적인 분석을 위하여 C#의 2D/3D 그래픽 라이브러리 등을 적극적으로 활용하였으며, 궤도 추정 알고리즘의 효율적인 성능을 위하여 Python에서 제공하는 진화연산 알고리즘을 단계적으로 적용하였다. 본 연구에서는 위성의 개수와 사용 정보의 종류에 따라서 크게 4가지 방식의 궤적 추정 알고리즘을 제안하였으며, 각각의 시뮬레이션을 통하여 성능의 장단점을 비교하였다. 개발된 시뮬레이터와 알고리즘은 실제 정지궤도위성에서 발생할 수 있는 여러 성능 저하 요인들을 동시에 고려하지 않았으며, 따라서 제안된 알고리즘의 성능비교는 상대적인 비교 관점에서만 의미를 가지는 한계가 있다. 이후 연구에서는 실제 정지궤도위성의 자세 제어에서 유발될 수 있는 지향 오차와 검출기의 공간해상도의 한계로부터 발생하는 양자 오차 등을 함께 고려하여, 제안된 알고리즘의 성능에 대한 실제적 활용가능성을 검토할 예정이다.

감사의 글

이 논문에 대하여 중요한 지적과 코멘트를 하여 주신 익명의 심사위원님들께 감사드립니다.

References

1. Myung H, Development trend of early warning satellite, Curr. Ind. Technol. Trends Aerosp. 18, 86-97 (2020).
2. Teague R, SBIRS transformational capability, Space and Missile Systems Center (2006).
3. Andreas NS, Space-based infrared system (SBIRS) system of systems, in 1997 IEEE Aerospace Conference, Snowmass, CO, 13 Feb 1997, 429-438.
4. Danis NJ, Space-based tactical ballistic missile launch parameter estimation, IEEE Trans. Aerosp. Electron. Syst. 29, 412-424 (1993). https://doi.org/10.1109/7.210079
5. An W, Zhang T, Zhou Y, The track capture of ballistic missile from satellite observations, J. Electron. Inf. Technol. 29, 2676-2678 (2007).

6. Dun L, et al., Tactical parameter estimation of ballistic missile launch with space early warning system, J. Astronaut. 22, 84-101 (2001).
7. Yuan T, Bar-Shalom Y, Willett P, Ben-Dov R, Pollak S, Estimation of thrusting trajectories in 3D from a single fixed passive sensor, IEEE Trans. Aerosp. Electron. Syst. 50, 2096-2108 (2014). https://doi.org/10.1109/TAES.2014.130418
8. Myung H, Simulator development of launch vehicle flight trajectory, Bull. Korean Space Sci. Soc. 30 (2021).
9. Kenji N, Python for Engineer (Jpub, Paju, Korea, 2017).
10. GlobalSecurity, M-3S (2022) [Internet], viewed viewed 2021 April 1, available from: https://www.globalsecurity.org/space/world/japan/m-3s.htm
11. Yoon YS, Moon Co, Lee SY, A hybrid genetic algorithm for solving nonlinear optimization problems, J. Expert Syst. 3, 11-22 (1997).
12. Nasr SM, El-Shorbagy MA, El-Desoky IM, Hendawy ZM, Mousa AA, Hybrid genetic algorithm for constrained nonlinear optimization problems, Br. J. Math. Comput. Sci. 7, 466-480 (2015). https://doi.org/10.9734/BJMCS/2015/16193
13. Asim M, Khan WM, Yeniay Ö, Jan MA, Tairan N, et al., Hybrid genetic algorithms for global optimization problems, Hacettepe J. Math. Stat. 47, 539-551 (2018). https://doi.org/10.15672/HJMS.2017.473

Author Information

명 환 춘 mhc@kari.re.kr

한국과학기술원 전기전자공학과에서 2002년 박사학위를 취득하였고, LG 전자 중앙연구소의 디지털 미디어 사업부를 거쳐서, 2005년부터 현재까지 한국항공우주연구원에서 정지궤도 위성의 탑재체와 관련된 업무를 수행하고 있다.