Technical Documentation - Space Engine

Overview

The Space Engine is a comprehensive earthquake prediction system that correlates space weather phenomena with seismic activity using empirical, physics-based calculations.

Architecture

Core Components

```
┌─────────────────────────────────────────────────────────────┐
│                      Space Engine Core                       │
├─────────────────────────────────────────────────────────────┤
│                                                               │
│  ┌──────────────────┐      ┌──────────────────┐             │
│  │  Atmospheric     │      │  Angle of        │             │
│  │  Boundary        │      │  Incidence       │             │
│  │  Refraction      │      │  Tracking        │             │
│  └──────────────────┘      └──────────────────┘             │
│                                                               │
│  ┌──────────────────┐      ┌──────────────────┐             │
│  │  Sun Path        │      │  Dynamic Lag     │             │
│  │  Prediction      │      │  Calculation     │             │
│  └──────────────────┘      └──────────────────┘             │
│                                                               │
│  ┌──────────────────┐      ┌──────────────────┐             │
│  │  RGB Resonance   │      │  Data            │             │
│  │  Calculations    │      │  Integration     │             │
│  └──────────────────┘      └──────────────────┘             │
│                                                               │
│  ┌──────────────────┐      ┌──────────────────┐             │
│  │  Resultant       │      │  Equatorial      │             │
│  │  Resonance       │      │  Enhancement     │             │
│  └──────────────────┘      └──────────────────┘             │
│                                                               │
└─────────────────────────────────────────────────────────────┘
                            │
                            ▼
                  ┌─────────────────┐
                  │  FastAPI REST   │
                  │      API        │
                  └─────────────────┘
                            │
                            ▼
                  ┌─────────────────┐
                  │    Frontend     │
                  │  Visualization  │
                  └─────────────────┘
```

Feature Details

1. Atmospheric Boundary Refraction

Physics Background

Electromagnetic signals traveling from space through Earth's atmosphere experience refraction at specific altitude boundaries. The 80-85km range represents the critical mesosphere-thermosphere transition.

Implementation

```python
def calculate_atmospheric_refraction(altitude_km: float, raw_value: float) -> float:
    if altitude_km <= 80:
        refraction_factor = 1.15
    elif altitude_km <= 85:
        ratio = (altitude_km - 80) / 5.0
        refraction_factor = 1.15 * (1 - ratio) + 1.12 * ratio
    else:
        refraction_factor = 1.12

    return raw_value * refraction_factor
```

Calibration Factors

- **80km Boundary**: 1.15 (stronger refraction in denser atmosphere)
- **85km Boundary**: 1.12 (weaker refraction in thinner atmosphere)
- **Interpolation**: Linear between boundaries

---

2. Angle of Incidence Tracking

Solar Elevation Calculation

Uses spherical trigonometry to calculate sun position:

```
sin(elevation) = sin(latitude) × sin(declination) +
                 cos(latitude) × cos(declination) × cos(hour_angle)
```

Where:
- **Declination**: `23.45° × sin(360° × (284 + day_of_year) / 365.25)`
- **Hour Angle**: `15° × (hour - 12)`

Tetrahedral Angles

Based on geometric relationships in Earth's structure:

- **Volcanic**: 54.74° (tetrahedral face angle)
- **Seismic**: 26.52° (complementary tetrahedral angle)

Magnetic Latitude Conversion

```
cos(magnetic_colatitude) = sin(lat) × sin(pole_lat) +
                           cos(lat) × cos(pole_lat) × cos(lon - pole_lon)

magnetic_latitude = 90° - magnetic_colatitude
```

Magnetic pole coordinates:
- Latitude: 80.65°N
- Longitude: 72.68°W

---

3. Sun Path Prediction

Stationary Earth Reference Frame

Calculates sun position relative to fixed Earth coordinates over time.

Ray Path Distance

```
def _calculate_ray_path_distance(elevation_angle: float) -> float:
    atmospheric_thickness = 100  # km

    if elevation_angle >= 85:
        return atmospheric_thickness
    else:
        return atmospheric_thickness / sin(elevation_angle)
```

---

4. Dynamic Lag Time Calculation

Physics-Based Delays

Light Travel Base Delay

```
delay = distance / speed_of_light
      = 149,597,870.7 km / 299,792.458 km/s
      ≈ 499 seconds
      ≈ 8.3 minutes
```

Solar Lag (4-12 hours)
- Seasonal variation based on Earth's orbit
- Formula: `lag = 4 + 4 × (1 + sin(day_of_year × 2π/365))`

Geomagnetic Lag (4-8 hours)
- Diurnal variation based on Earth's rotation
- Formula: `lag = 4 + 2 × (1 + sin(hour × 2π/24))`

Ionospheric Lag (1-7 hours)
- Semi-diurnal variation (twice daily)
- Formula: `lag = 1 + 3 × (1 + sin(hour × 4π/24))`

Angle Correction

```
if abs(elevation) < 85:
    angle_factor = 1.0 / cos(elevation)
else:
    angle_factor = 10.0  # Large correction for extreme angles
```

---

5. RGB Resonance Calculations

Formula

```
RGB_resonance = sqrt((R² + G² + B²) / 3.0)
```

Component Mapping

R Component (Solar Wind):
- Solar Activity
- Solar Flare Intensity
- Cosmic Ray Intensity
- Magnetosphere Compression

G Component (Magnetic Field):
- Geomagnetic Field
- Solar Wind Pressure
- Auroral Activity
- Interplanetary Magnetic Field

B Component (Particle Flux):
- Planetary Alignment
- Ionospheric Density
- Coronal Mass Ejection
- Galactic Cosmic Radiation

---

6. Data Integration

NASA OMNI2 API

Endpoint: `https://omniweb.gsfc.nasa.gov/cgi/nx1.cgi`

Parameters:

```
{
    'activity': 'retrieve',
    'res': 'hour',
    'spacecraft': 'omni2',
    'start_date': 'YYYYMMDD',
    'end_date': 'YYYYMMDD',
    'vars': [1, 6, 8, 22, 40, 38]  # Specific variables
}
```

Variables Retrieved:
1. Scalar B (magnetic field strength, nT)
6. Flow speed (solar wind velocity, km/s)
8. Proton density (n/cm³)
22. AE index (auroral electrojet)
40. DST index (geomagnetic storm)
38. Kp index (geomagnetic activity)

NOAA SWPC API

Endpoints:
- Solar flares: `/json/goes/xrs-1-day.json`
- Geomagnetic indices: `/json/planetary_k_index_1m.json`
- Solar wind: `/json/ace/swepam_1m.json`

Graceful Failure

When APIs are unavailable:

```
return {
    'data_available': False,
    'error': 'Unable to retrieve real-time space weather data',
    'fallback': 'Using historical baseline values'
}
```

---

7. Resultant Resonance Calculations

12D Correlation Matrix

Matrix dimensions: 12×12 (one row/column per space variable)

Diagonal Elements: 1.0 (self-correlation)

Off-Diagonal Elements: Cross-correlations calculated as:

```
correlation = rgb_factor × weight_factor × correlation_factor ×
              spatial_factor × temporal_factor
```

Where:
- `rgb_factor`: 1.0 if same RGB component, 0.5 otherwise
- `weight_factor`: sqrt(weight1 × weight2)
- `correlation_factor`: correlation1 × correlation2
- `spatial_factor`: 1.0 - |magnetic_lat| / 180
- `temporal_factor`: 0.5 + 0.5 × cos(hour × 2π/24)

Eigenvalue Analysis

Dominant eigenvalue extracted using NumPy:

```
eigenvalues = np.linalg.eigvals(correlation_matrix)
dominant_eigenvalue = np.max(np.real(eigenvalues))
```

Resultant Resonance

```
resultant = rgb_resonance × dominant_eigenvalue × matrix_mean
```

---

8. Equatorial Enhancement

Factor Application

Full Enhancement at Equator:

```
if |latitude| <= 23.5°:
    enhancement_ratio = 1.0 - |latitude| / 23.5°
    factor = 1.0 + (1.25 - 1.0) × enhancement_ratio
else:
    factor = 1.0
```

Enhanced Value:

```
enhanced_value = base_value × factor
```

Earthquake Correlation Score

Calculation

```
correlation = (
    resonance × 0.5 +           # 50% weight
    solar_factor × 0.2 +         # 20% weight
    lag_factor × 0.2 +           # 20% weight
    magnetic_factor × 0.1        # 10% weight
)
```

Where:
- `solar_factor = |solar_elevation| / 90`
- `lag_factor = 1.0 - |lag_hours - 8| / 8` (optimal at 8 hours)
- `magnetic_factor = |magnetic_lat| / 90`

Performance Considerations

Optimization Strategies

1. **Async Operations**: All API calls use `asyncio`
2. **Caching**: Store last calculation to avoid redundant computation
3. **Batch Processing**: Multiple calculations in single request
4. **Timeout Handling**: 10-second timeout on external APIs

Memory Usage

- Space Engine instance: ~1 MB
- Correlation matrix: ~1 KB (12×12 float64)
- Prediction result: ~10 KB JSON

Error Handling

API Failures

```python
try:
    response = await fetch_api_data()
except Exception as e:
    logger.error(f"API failed: {str(e)}")
    return graceful_failure_response()
```

Invalid Input

- Latitude: -90 to 90
- Longitude: -180 to 180
- Timestamp: Valid ISO format

Edge Cases

- Polar regions (lat = ±90°)
- Date boundaries (Dec 31 ↔ Jan 1)
- Empty space readings
- Extreme solar angles (near horizon)

Testing Strategy

Unit Tests

- Individual feature tests
- Edge case validation
- Error handling verification

Integration Tests

- Full prediction calculation
- API endpoint testing
- Frontend-backend integration

Performance Tests

- Response time < 500ms
- Concurrent request handling
- Memory leak detection

Future Enhancements

1. **Database Integration**: PostgreSQL for historical data
2. **Machine Learning**: Pattern recognition in space-seismic correlations
3. **Real-time Monitoring**: WebSocket streaming
4. **Advanced Visualization**: 3D globe, interactive charts
5. **Mobile App**: iOS/Android native applications

6. **Alert System**: Automated notifications for high-risk predictions

---

References

Scientific Papers

1. NASA OMNI2 Dataset Documentation
2. NOAA Space Weather Prediction Center API Documentation
3. Spherical Trigonometry in Solar Position Calculations
4. Earth's Magnetic Field Models

Code References

- Original BRETT system algorithms
- GEO_EARTH space correlation engine
- QuakePredictionTestSystem space_v_engine
- EarthQuake_historical_Test implementations

Document Version: 1.0
Last Updated: October 2024