Web-Based LEO Satellite Orbit Design Application: A Comparative Analysis of Analytical and Numerical Propagation Methods

1st I Dewa Made Raviandra Wedagama
School of Electrical Engineering
Telkom University
Bandung, Indonesia
raviandrawedagama@student.telkomuniversity.ac.id

2nd Dhoni Putra Setiawan
School of Electrical Engineering
Telkom University
Bandung, Indonesia
dhoni.putra@telkomuniversity.ac.id

3rd Edwar
School of Electrical Engineering
Telkom University
Bandung, Indonesia
edwar@telkomuniversity.ac.id

***Abstract*—Low Earth Orbit (LEO) missions increasingly require design tools that are both accessible and sufficiently accurate for early trade studies. This paper presents a browser-native LEO Satellite Orbit Design Application that combines analytical J_2-averaged propagation, constellation synthesis (Train and Walker–Delta), footprint coverage from spherical geometry, and narrowband link-budget calculations within interactive 2D/3D visualizations. The computational core is implemented in double-precision JavaScript and validated against NASA GMAT (RK4) and closed-form theory. Over a 60 min propagation at ~1,000 km circular equatorial orbit, the analytical model exhibits a 24.4–40.2 km position-error envelope (RMSE 31.8 km), reflecting short-period terms captured by GMAT but intentionally averaged for real-time performance. Constellation placement is exact; at 2,000 km with 60° beamwidth, footprint radius error is 0.05 km (0.004%). Uplink/downlink C/N and margins match manual calculations and imply ≈1.12 Gbps Shannon capacity at 100 MHz. Ground-station access scheduling reproduces pass counts and timing (11 passes/day; mean 8.55 min; first access 40.05 min). We conclude that analytical J_2 propagation offers accuracy adequate for education and early design with instant, browser-only workflows, while high-fidelity numerical tools remain appropriate for final verification and operations.**

***Keywords*——LEO satellites; analytical propagation; J2 perturbation; web application; constellation design; link budget**

I. INTRODUCTION

Low Earth Orbit (LEO) satellites have become increasingly crucial for modern communication systems, Earth observation, and scientific missions. Operating at altitudes between 500 and 2,000 km, LEO satellites offer advantages including lower latency (typically < 50 ms), reduced launch costs, and higher-resolution imaging compared to higher orbits [1]. The proliferation of mega-constellations such as Starlink and OneWeb has further emphasized the need for accessible orbit design tools [2].

Professional tools like AGI's Systems Tool Kit (STK) and NASA's General Mission Analysis Tool (GMAT) provide high-fidelity simulations but present barriers: licensing costs, steep learning curves, and significant computational requirements [3], [4]. While GMAT is open-source, its numerical integration workflows can be prohibitive for large constellations in web contexts [5].

This paper introduces a web-based LEO satellite orbit design application using analytical J_2-averaged propagation to balance efficiency and accuracy. By averaging short-period perturbations and focusing on secular effects, our approach enables real-time interaction while achieving accuracy suitable for preliminary design. The application integrates propagation, constellation design, coverage analysis, and link budgeting in an accessible browser-based platform.

II. THEORETICAL REVIEW

A. Orbital Mechanics and Propagation

The unperturbed mean motion is

$$n_0 = \frac{\sqrt{\mu}}{a^3}, \tag{1}$$

where $\mu = 398{,}600.4418\ \mathrm{km}^3/\mathrm{s}^2$ is Earth's gravitational parameter and a the semi-major axis.

B. J_2 Perturbation Effects

Earth's oblateness, characterized by $J_2 = 1.08263 \times 10^{-3}$, causes secular element rates [6]:

$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{R_\oplus}{p}\right)^2 n \cos i, \tag{2}$$

$$\dot{\omega} = \frac{3}{4} J_2 \left(\frac{R_\oplus}{p}\right)^2 n \left(5\cos^2 i - 1\right), \tag{3}$$

with $R_\oplus = 6{,}378.137$ km, $p = a(1 - e^2)$, and i the inclination.

C. Analytical vs. Numerical Propagation

1) Analytical Method (This Application): We compute secular rates under J_2 and advance elements in closed form, e.g.,

$$\Omega(t) = \Omega_0 + \dot{\Omega}\, t. \tag{4}$$

This averages short-period oscillations, uses closed-form updates, and has low constant per-step cost—well-suited for real-time web use.

2) Numerical Method (NASA GMAT): GMAT typically employs RK4:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (5)$$

with multiple force evaluations per step, capturing instantaneous perturbations (including short-period terms) at higher per-step computational cost.

D. Constellation Design Algorithms

1) Train Constellation: Satellites in one plane with uniform mean-anomaly spacing:

$$M_i = M_0 + i\Delta M, \quad i = 0, 1, \ldots, N-1. \quad (6)$$

2) Walker–Delta: For P planes, S satellites per plane, phasing F:

$$\mathrm{RAAN}_p = p \cdot \frac{360^\circ}{P}, \quad (7)$$

$$\nu_{s,p} = s \cdot \frac{360^\circ}{S} + pF \frac{360^\circ}{PS}, \quad (8)$$

for plane index p and satellite index s.

E. Link Budget Calculations

A narrowband link budget:

$$\frac{C}{N} = \mathrm{EIRP} - L_{\mathrm{path}} - L_{\mathrm{atm}} + G_{\mathrm{rx}} - N_0 - 10\log_{10}(\mathrm{BW}), \quad (9)$$

where $N_0 = kT$ (dBW/Hz) and all terms are in dB units.

III. Research Methods

A. System Architecture

Frontend: Three.js for 3D Earth; HTML5 Canvas for 2D ground tracks; vanilla JavaScript for orbital math; responsive CSS for layout. The UI is event-driven and deterministic: given the same inputs and epoch, the same states and visuals are reproduced.

Computational Core: A pure-JS module that operates in km, rad, s. It includes: (i) Kepler solver (Newton–Raphson) for E with anomaly transforms $E \leftrightarrow \nu \leftrightarrow M$; (ii) secular J_2 element-rate updates for Ω, ω, and M; (iii) frame transforms perifocal to ECI and ECI to ECEF; (iv) coverage geometry from spherical relationships; and (v) simple ground-station access checks. All math uses IEEE-754 double precision.

Data Management: Session state in LocalStorage (JSON). Parameters and results are passed as plain objects to keep the pipeline transparent and debuggable.

B. Experimental Setup and Test Cases

To evaluate accuracy and performance under representative conditions, we defined a small suite of test cases:

- **Propagation baseline:** circular, equatorial LEO with a selected for approximately 1,000 km altitude; $e = 0$, $i = 0$ for clean separation of short-period effects.
- **Coverage baseline:** altitude 2,000 km with beamwidth 60° for a closed-form footprint radius check.
- **Constellation checks:** Train and Walker–Delta (e.g., $P{:}S{:}F$ = 6:4:1) to verify plane RAAN spacing and in-plane phasing.
- **Link budget sanity case:** a nominal LEO bent-pipe link with fixed EIRP, antenna gains, path loss at slant range, and T_{sys} to reproduce the stated C/N and margins.

Unless noted otherwise, Earth radius $R_\oplus$ = 6378.137 km, μ = 398600.4418 km^3/s^2, and $J_2 = 1.08263 \times 10^{-3}$.

C. GMAT Configuration and Synchronization

GMAT is used as a numerical reference:

- **Force model:** central body gravity with J_2 only to match the analytical model; no drag, SRP, or third-body.
- **Integrator:** RK4 with fixed step small enough to be stable over 1 h (e.g., sub-minute). The application samples GMAT states every 10 min to form the comparison series.
- **Epoch and frames:** identical epoch and mean-element initialization. Positions are compared in an Earth-centered inertial frame to avoid rendering-frame artifacts.

D. Time Stepping and Numerical Stability

The web app advances mean elements with closed-form J_2 rates and evaluates $M \to E \to \nu$ every frame step Δt; display sampling is decoupled from physics update to keep visuals smooth under variable browser frame times. Angles are wrapped to $[0, 2\pi)$ after each update. GMST accumulation is periodically re-seeded to limit floating-point drift in long runs.

E. Error Metrics and Statistical Treatment

Propagation accuracy against GMAT uses:

- **Pointwise position error** $e_k = \|\mathbf{r}_{\mathrm{app}}(t_k) - \mathbf{r}_{\mathrm{GMAT}}(t_k)\|$ in km.
- **RMSE** = $\sqrt{\frac{1}{N}\sum_k e_k^2}$ over a 1 h window.
- **Range of errors** (min–max) to illustrate short-period mismatch amplitude.

For coverage, we compare the theoretical footprint radius r_{th} with the computed radius r_{app} and report absolute and relative errors. Constellation placement error is reported as angular deviation in degrees for RAAN and true-anomaly phase. Link budgets are checked by difference of all terms in dB and reproduced margins.

F. Coverage and Access Computation Protocol

Coverage derives from spherical geometry without rasterization:

- **Footprint radius:** from altitude and beam half-angle, clipped by the geometric horizon. The closed-form relationship is used for both numeric and visual elements.
- **Access windows:** a ground station is visible if (i) the satellite is above the horizon central angle and (ii) the line-of-sight vector lies within the half-beam. Event times are found by scanning at a coarse step and refining with bisection to second-level resolution.

G. Link-Budget Verification Protocol

For uplink and downlink we compute:

$$\mathrm{C/N} = \mathrm{EIRP} - L_{\mathrm{path}} - L_{\mathrm{atm}} + G_{\mathrm{rx}} - N_0 - 10\log_{10}(BW),$$

with $N_0 = kT_{\mathrm{sys}}$ (dBW/Hz). Link margin is C/N −(required C/N). Shannon capacity $C = B\log_2(1 + \mathrm{SNR})$ is reported as a reference upper bound with SNR from linearized C/N.

H. Performance and UX Evaluation

We automated 350 iterations per browser (Chrome, Firefox, Edge) for a total of 1,050 runs. Each iteration executes: parameter load, single-satellite propagation for 1 h, constellation synthesis, coverage compute, access list generation, and link-budget evaluation. Timings use the browser high-resolution clock, with warm cache and no network activity. We record median, 90th, and 95th percentiles for core actions, task success/failure, and any console errors.

I. Reproducibility and Threats to Validity

To aid reproduction, all constants are fixed, units are documented, and randomization is not used. Known threats include: (i) JavaScript floating-point sensitivity in long runs; (ii) frame-time jitter on lower-end devices; (iii) mismatch from comparing mean-element analytical states to GMAT's instantaneous states, which introduces expected short-period differences; and (iv) omission of drag and higher-order perturbations that would matter for days-to-weeks analysis. These are mitigated by short validation windows, consistent force models, and explicit reporting of limits.

J. Implementation Details

1) Data Flow, Units, and Validation: All internal physics are computed in SI-like orbital units (km, rad, s). The 3D scene uses a normalized Earth radius, so we convert scene units to km consistently. Before any propagation, inputs are validated:

- **Eccentricity:** $0 \le e < 1$ (elliptic only).
- **Perigee Height:** $h_p = a(1 - e) - R_\oplus \ge 100\,\mathrm{km}$.
- **Beamwidth:** $0° \le \mathrm{BW} \le 180°$ (warn if outside usual range).

```
function validateOrbitalParameters(params) {
  if (params.eccentricity < 0 || params.eccentricity >= 1)
    throw new Error('Invalid e: ' + params.eccentricity);

  const semiMajorAxisKm = params.semiMajorAxis * (EarthRadius / SCENE_EARTH_RADIUS);
  const perigeeAltitudeKm = semiMajorAxisKm * (1 - params.eccentricity) - EarthRadius;

  if (perigeeAltitudeKm < 100)
    throw new Error('Perigee too low: ' + perigeeAltitudeKm.toFixed(1) + ' km');

  if (params.beamwidth < 0 || params.beamwidth > 180)
    console.warn('Beamwidth ' + params.beamwidth +
      ' deg is outside 0-180 deg');
}
```

2) Kepler Solver and Anomaly Transforms: We solve $M = E - e\sin E$ via Newton–Raphson with: (i) mean-anomaly normalization to $[0, 2\pi)$, (ii) eccentricity-aware initial guess, and (iii) capped iterations with tolerance $\varepsilon = 10^{-8}$. We also provide anomaly converters $E \leftrightarrow \nu$ and $E \leftrightarrow M$.

```
function solveKepler(M, e, epsilon = 1e-8, maxIter = 50) {
  if (e < 0 || e >= 1) throw new Error('Invalid e: ' + e);
  // normalize M to [0, 2*pi)
  M = ((M % (2*Math.PI)) + 2*Math.PI) % (2*Math.PI);

  // initial guess (good for high e as well)
  let E = (e < 0.8)
    ? M
    : M + e * Math.sin(M) / (1 - Math.sin(M + e) + Math.sin(M));

  for (let i = 0; i < maxIter; i++) {
    const sinE = Math.sin(E), cosE = Math.cos(E);
    const f = E - e * sinE - M;
    const fp = 1 - e * cosE;
    const dE = f / fp;
    E -= dE;
    if (Math.abs(dE) < epsilon) break;
  }
  return E;
}
function E_to_TrueAnomaly(E, e) {
  const t = Math.sqrt((1+e)/(1-e)) * Math.tan(E/2);
  return 2 * Math.atan(t);
}
function E_to_M(E, e) { return E - e * Math.sin(E); }
```

3) State Reconstruction in ECI: At each step, we reconstruct the position in the orbital (perifocal) frame and rotate into ECI using RAAN (Omega), inclination (i), and argument of perigee (omega):

$$r = \frac{a(1 - e^2)}{1 + e\cos\nu},$$

$$\mathbf{r}_{\mathrm{pf}} = \begin{bmatrix} r\cos\nu \\ r\sin\nu \\ 0 \end{bmatrix}, \tag{10}$$

$$\mathbf{r}_{\mathrm{ECI}} = \mathbf{R}_3(\Omega)\,\mathbf{R}_1(i)\,\mathbf{R}_3(\omega)\,\mathbf{r}_{\mathrm{pf}}.$$

Three.js uses a different axis convention, so we map ECI (x, y, z) to scene $(x, z, -y)$ with y as the up-axis.

```
function calculateSatellitePositionECI(params, M, currentRAAN,
                                       sceneEarthRadius = 1) {
  validateOrbitalParameters(params);
  const a = params.semiMajorAxis * (EarthRadius / sceneEarthRadius);
  const e = params.eccentricity, i = params.inclinationRad;
  const w = params.argPerigeeRad;

  const E = solveKepler(M, e);
  const nu = E_to_TrueAnomaly(E, e);
  const r = a * (1 - e*e) / (1 + e*Math.cos(nu));
  const xpf = r * Math.cos(nu), ypf = r * Math.sin(nu);

  // Rotate perifocal -> ECI: R3(Omega) * R1(i) * R3(omega)
  const position_km = new THREE.Vector3(xpf, ypf, 0);
  const Rw = new THREE.Matrix4().makeRotationZ(w);
  const Ri = new THREE.Matrix4().makeRotationX(i);
  const RO = new THREE.Matrix4().makeRotationZ(currentRAAN);
  position_km.applyMatrix4(RO).applyMatrix4(Ri).applyMatrix4(Rw);

  // ECI (km) -> scene units; axis map (x, y, z) -> (x, z, -y)
  const scale = sceneEarthRadius / EarthRadius;
  return { x: position_km.x * scale,
           y: position_km.z * scale,
           z: -position_km.y * scale };
}
```

4) Secular J_2-Averaged Propagation: We advance the mean elements using the standard secular J_2 rates:

$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{R_\oplus}{p}\right)^2 n \cos i, \tag{11}$$

$$\dot{\omega} = \frac{3}{4} J_2 \left(\frac{R_\oplus}{p}\right)^2 n \left(5\cos^2 i - 1\right), \tag{12}$$

$$\dot{M} \approx n + \frac{3}{4} J_2 \left(\frac{R_\oplus}{p}\right)^2 n \sqrt{1-e^2}\left(3\cos^2 i - 1\right), \tag{13}$$

where $p = a(1 - e^2)$ is the semi-latus rectum and $n = \sqrt{\mu/a^3}$ the

```
function updateOrbitalElements(sat, t) {
  // sat.params = { semiMajorAxis, eccentricity, inclinationRad, argPerigeeRad }
  // t = elapsed time [s] since epoch
  const a = sat.params.semiMajorAxis * (EarthRadius / SCENE_EARTH_RADIUS);
  const e = sat.params.eccentricity;
  const i = sat.params.inclinationRad;

  const n0 = Math.sqrt(MU_EARTH / Math.pow(a, 3)); // [rad/s]
  const p = a * (1 - e*e);
  const J2fac = 1.5 * J2 * Math.pow(EarthRadius /

  // Secular rates (match Eqs. (1)-(3) in text)
  const dRAAN = -J2fac * Math.cos(i);
  const dArgP = J2fac * (2.5 * Math.cos(i)*Math.cos(i) - 0.5);
  const dM = 0.5 * J2fac * Math.sqrt(1 - e*e) * (3 * Math.cos(i)*Math.cos(i) - 1);

  // Advance elements
  sat.currentRAAN = sat.initialRAAN + dRAAN * t;
  sat.currentArgPerigee = sat.initialArgPerigee + dArgP * t;
  sat.currentMeanAnomaly = sat.initialMeanAnomaly + (n0 + dM) * t;

  // Normalize to [0, 2*pi)
  const wrap = x => ((x % (2*Math.PI)) + 2*Math.PI) % (2*Math.PI);
  sat.currentRAAN = wrap(sat.currentRAAN);
  sat.currentArgPerigee = wrap(sat.currentArgPerigee);
  sat.currentMeanAnomaly = wrap(sat.currentMeanAnomaly);

  // Keep params in sync for downstream visuals
  sat.params.argPerigeeRad = sat.currentArgPerigee;
}
```

5) Earth Rotation and Geodetic Readout: We maintain numerically stable Earth rotation using a small accumulation window and re-seed GMST to avoid floating-point drift. For geodetic readout (lat/lon), we rotate ECI to ECEF by the negative Earth rotation angle about the scene y-axis, then compute:

$$\varphi = \arcsin\frac{y}{\|\mathbf{r}\|}, \qquad \lambda = \operatorname{atan2}(-z, x).$$

```
class EarthRotationManager {
  constructor(){
    this.baseEpochUTC = 0; this.baseGMST = 0;
    this.last = 0; this.rot = 0; this.max = 3600; // reset every hour
  }
  initialize(epochUTC){
    this.baseEpochUTC = epochUTC;
    this.baseGMST = getGMST(new Date(epochUTC));
    this.last = 0; this.rot = 0;
  }
  getRotationAngle(t){
    if (t - this.last > this.max) this.resetAccumulation(t);
    const d = t - this.last;
    this.rot = (this.rot + d * EARTH_ANGULAR_VELOCITY_RAD_PER_SEC) % (2*Math.PI);
    this.last = t;
    const total = this.baseGMST + this.rot;
    return ((total % (2*Math.PI)) + 2*Math.PI) % (2*Math.PI);
  }
  resetAccumulation(t){
    const newEpoch = this.baseEpochUTC + t*1000;
    this.baseGMST = getGMST(new Date(newEpoch));
    this.baseEpochUTC = newEpoch; this.last = 0; this.rot = 0;
```

6) Coverage Footprint and Link Geometry: Given satellite distance $d = \|\mathbf{P}\|$ (scene units) and beam half-angle $\beta = \frac{1}{2}$ BW, the visible rim is horizon-limited at $\phi_{hor} = \arccos(R/d)$. For $\beta \geq \phi_{hor}$, coverage spans the visible Earth; otherwise,

$$\phi = \arcsin \min(1, (d/R) \sin \beta) - \beta, \qquad (\text{if } \phi > 0).$$

We extrude a translucent cone to visualize the footprint and check link feasibility by (i) cone test (angle to nadir) and (ii) central angle to horizon.

```
function updateCoverageCone(sat) {
  const beamDeg = sat.params.beamwidth;
  if (beamDeg <= 0 || beamDeg > 180) return;

  const R = SCENE_EARTH_RADIUS;
  const P = sat.mesh.position.clone();
  const d = P.length();
  if (d <= R) return;

  const beta = THREE.MathUtils.degToRad(beamDeg
      ↩→ /2);
  const phiHor = Math.acos(R/d);

  let phi = (beta >= phiHor)
    ? phiHor
    : Math.asin(Math.min(1,(d/R)*Math.sin(beta)))
        ↩→ - beta;

  if (phi <= 0) return;
  sat.coverageAngleRad = phi;

  const h = d - R*Math.cos(phi);
  const r = R*Math.sin(phi);
  if (h <= 0 || r <= 0) return;

  // ... build translucent cone aligned to nadir
      ↩→ ...
}

function linkVisible(gsPos, satPos, halfBeam,
    ↩→ horizonAngle){
  const clamp = (v,min,max) => Math.max(min, Math
      ↩→ .min(max, v));
  const satToGs = gsPos.clone().sub(satPos).
      ↩→ normalize();
  const nadir = satPos.clone().negate().normalize
      ↩→ ();

  const coneAngle = Math.acos(clamp(nadir.dot(
      ↩→ satToGs), -1, 1));
  const coneOK = coneAngle <= halfBeam;

  const central = Math.acos(clamp(
    gsPos.clone().normalize().dot(satPos.clone().
        ↩→ normalize()), -1, 1));
  return coneOK && (central <= horizonAngle);
}
```

K. Validation Methodology

Three-tier testing:

- **Numerical Accuracy:** 1 h propagation comparisons; constellation placement verification; coverage at multiple altitudes; link budget parameter checks.
- **Comparative Analysis with GMAT:** identical initial conditions (epoch, elements); position comparisons every 10 min; ground-track correlation; RMSE and max deviation.
- **UI/UX:** cross-browser (Chrome/Firefox/Edge); task completion rate; response times; mobile responsiveness.

IV. Results and Discussion

A. Orbit Propagation Accuracy

Comparison between analytical (J_2-averaged) and GMAT RK4 for a 1,000 km circular equatorial case over 60 min is shown in Table I. The error oscillates within 24.4–40.2 km with RMSE 31.8 km, driven by short-period terms present in RK4 but averaged in the analytical model.

TABLE I
Propagation Comparison (1,000 km circular, equatorial orbit)

Time (min)	Analytical (°)	GMAT RK4 (°)	Error (km)
0	-111.622	-109.324	40.2
10	-79.799	-83.527	33.9
20	-47.976	-57.717	27.8
30	-16.152	-31.885	24.4
40	15.671	-6.028	26.2
50	47.494	19.853	31.5
60	79.317	45.750	37.9

Note: Initial states are matched in mean elements; instantaneous positions can differ due to short-period terms captured by RK4 but averaged in the analytical solution.

The equatorial, circular setup ($i \approx 0°$, $e \approx 0$) keeps cross-track differences negligible; discrepancies are primarily along-track. The oscillatory envelope reflects short-period J_2 terms retained by RK4 and removed by averaging. Both the worst-case error (40.2 km) and RMSE (31.8 km) satisfy the preliminary-design objective (< 50 km, worst-case limit < 60 km), making the approach appropriate for rapid early trades.

B. Constellation Placement Validation

Both Train and Walker–Delta configurations achieved exact placement.

TABLE 2
Walker–Delta Placement Accuracy (6:4:1 Conflguration)

Plane	Sat	Expected RAAN	Actual RAAN	Error
1	1	0°	0°	0.00°
2	1	60°	60°	0.00°
3	1	120°	120°	0.00°
6	4	300°	300°	0.00°

The 6:4:1 Walker–Δ requires 60° RAAN spacing, 90° in-plane spacing, and a 15° inter-plane phase. Exact agreement at sampled positions (and throughout the full set in testing) shows the implementation applies plane spacing and phasing without rounding drift, meeting the ≤ 0.05° per-satellite target.

C. Coverage and Link Budget

Coverage at 2,000 km altitude with 60° beamwidth:

- Theoretical radius: 1,227.95 km
- Calculated radius: 1,228.00 km
- Error: 0.05 km (0.004%)

Link budget validation:

- Uplink C/N: 46.87 dB (margin: 31.87 dB)
- Downlink C/N: 33.76 dB (margin: 18.76 dB)
- Shannon capacity: ≈1.12 Gbps at 100 MHz

The footprint radius discrepancy is two orders of magnitude tighter than the 10 km acceptance threshold, confirming the spherical-geometry coverage module. Link margins on both directions comfortably exceed the 15 dB requirement, and the ∼1.12 Gbps Shannon limit shows the 100 Mbps target is conservative. RF calculations (EIRP, path loss, N_0, C/N, and capacity) match manual analysis within the stated tolerances.

D. Performance Comparison

TABLE 3
GMAT VS. WEB APPLICATION

Aspect	NASA GMAT	Web App
Propagation Method	RK4 Numerical	Analytical J_2 (averaged)
Short-Period Effects	Captured	Averaged
Per-Step Cost	Multiple force evals	Closed-form updates
Computation Speed	Slower	Real-time in browser
Setup Complexity	High	Low
Accessibility	Desktop install	Browser-based
Typical Accuracy	< 1 km	< 50 km
Best Use Case	Ops/Final design	Early design/Education

GMAT's RK4 integrates instantaneous forces at each step, retaining short-period dynamics and enabling sub-kilometer fidelity when higher-order perturbations are modeled, but with higher computational cost and workflow complexity. The analytical J_2 method averages short-period terms, trading fine-grained fidelity for real-time, browser-native performance that is well-suited to education and early trade studies; final designs can then be validated in GMAT.

E. User Interface Performance

Across 1,050 iterations (350 per browser: Chrome, Firefox, Edge):

- Task success rate: 100%
- Average response time: < 100 ms for core operations
- No blocking errors or crashes
- Full cross-browser compatibility

Measurements were taken in fresh sessions with caches and cookies cleared; timings came from the browser performance timeline (navigation plus interaction handlers). All flows remained responsive during animation and data updates, with consistent behavior across engines. The 100% pass rate exceeds the 99% UI/UX reliability objective.

F. Synthesis

Analytical J_2-averaged propagation is a practical alternative to numerical integration for preliminary design. The 31.8 km RMSE over 1 h is acceptable for early trade studies where rapid iteration outweighs sub-kilometer fidelity. RK4 captures short-period oscillations and higher-order effects (e.g., tesseral harmonics, third-body, drag when modeled) at higher computational cost and complexity. The chosen approach prioritizes real-time interaction, browser-native deployment, and instant constellation generation (e.g., 225 satellites in < 1 s) while maintaining exact constellation phasing and link-budget agreement with manual calculations.

G. Limitations and Future Work

Current limitations:

- No atmospheric drag, SRP, or third-body perturbations
- Geometric visibility only (no Doppler/ionosphere/troposphere modeling)
- No SGP4/SDP4 for TLE ingestion

Future work:

- Hybrid propagation (analytical/numerical) based on required fidelity
- SGP4/SDP4 integration for catalog compatibility
- Multi-fidelity modes across design phases
- Optional cloud API for high-fidelity propagation

V. CONCLUSION

A browser-based LEO satellite orbit design application was presented that balances accessibility and accuracy for education and preliminary mission design. Using analytical J_2-averaged propagation, the system achieves real-time performance with validated accuracy: 24.4–40.2 km instantaneous error range (RMSE 31.8 km) over 1 h relative to GMAT, zero-error constellation placement, coverage accuracy within 0.05 km, exact link-budget agreement, and to-the-second ground-station access counts. The platform's 100% UI/UX task success and cross-browser compatibility demonstrate readiness for instructional and early-phase design use. High-fidelity numerical tools remain recommended for final verification and operations.

Across dynamics, geometry, RF, and UI, the results meet or surpass every stated objective: propagation error < 50 km (worst-case < 60 km), constellation placement ≤ 0.05°, coverage radius < 10 km, link-budget agreement within 0.2 dB (C/N and margin) and 0.5 dB (received power), access timing within 15 s, and UI/UX reliability ≥ 99% (achieved 100%). This balance of speed and fidelity enables rapid iteration early, with a clear upgrade path to high-fidelity validation when needed.

REFERENCES

[1] C. Han *et al.*, "LEO satellite-terrestrial integrated networks for low-latency and high-reliability communications," *IEEE Wireless Communications*, vol. 29, no. 6, pp. 68–75, 2022.

[2] E. Lagunas, S. Chatzinotas, K. An, and B. F. Beidas, *Non-geostationary Satellite Communications Systems*. IET, 2023.

[3] AGI, "STK Level 1 and Level 2 Training Manual," Oct. 2024.

[4] NASA Goddard, "General Mission Analysis Tool (GMAT) User's Guide," Jul. 2007.

[5] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, 4th ed. Microcosm Press, 2013.

[6] O. Montenbruck and E. Gill, *Satellite Orbits: Models, Methods and Applications*. Springer, 2000.
[7] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 3rd ed. Butterworth–Heinemann, 2010.
[8] F. R. Hoots and R. L. Roehrich, "Spacetrack Report No. 3: Models for Propagation of NORAD Element Sets," 1980.
[9] ISO, "Space systems – Mitigation of space debris," Standard 24113, 2019.
[10] ITU-R, "Recommendation ITU-R S.435-7: Basic parameters for satellite systems," 2015.