High Precision Satellite Orbit Simulation: A Test Bench for Automatic Differentiation in MATLAB

M. Kalkuhl, W. Wiechert
{kalkuhl, wiechert}@simtec.mb.uni-siegen.de
Department of Simulation, Faculty 11/12
University of Siegen, D-57068 Siegen, Germany

H. M. Bücker, A. Vehreschild
{buecker, vehreschild}@sc.rwth-aachen.de
Institute for Scientific Computing
RWTH Aachen University, D-52056 Aachen, Germany

Abstract

A detailed understanding of the earth's gravitational field is crucial for a precise prediction of a satellite flight trajectory. The model of the CHAMP gravitational potential based on recent high accuracy measurements is taken as a starting point for a mechanically generated implementation of its gradient. For a more detailed mathematical model, the Jacobian of the right-hand side of the motion equation is also mechanically computed. This is achieved by automatic differentiation with the recent software tool ADiMat (Bischof et al., SCAM2002, pp. 65–72, IEEE Comp. Soc., 2002). Since an analytic form of the gradient and the Jacobian are available, the situation serves as a suitable test case for ADiMat implementing automatic differentiation in MATLAB.

1 Introduction

During the last some years, the development of high precision orbit models has become more and more important when satellites aviate in low altitudes of less than 1000 km. Moreover, a very precise tracking of the satellite flight trajectory is required for different simulation applications, e.g. SAR-data processing (SAR – **S**ynthetic **A**perture **R**adar).

Elementary models for celestial mechanics or satellite orbits are often based on simple models like Newton's law of gravitation. This law describes an ideal model of the gravitational field with spherical equipotential surfaces. To predict a flight trajectory of a satellite within a high precision simulation [1] of a real mission, it is necessary to use a realistic model of the earth's gravitational field instead of the idealized representation. Moreover, other physical influences on the satellite like atmospheric drag, celestial gravitation, solar radiation pressure, relativistic effects, or earth tides should be considered for such a simulation.

A potential model description for most of these phenomena is established. For the simulation of a flight trajectory, a motion equation of the satellite in the form

$$m \cdot \ddot{\vec{r}} = \sum_{\text{effect}} \vec{F}_{\text{effect}} = m \cdot \sum_{\text{effect}} \vec{a}_{\text{effect}} \tag{1}$$

is required. Here, the knowledge of the accelerations of each effect, $\vec{a}_{\text{effect}}$, acting on the satellite with mass m at position $\vec{r}$ is assumed. The acceleration is expressed in terms of the gradient of a scalar-valued function, the gravitational potential. That is, the conversion from potential to acceleration is based on differentiating the gravitational potential with respect to the spatial coordinates. In a more realistic model, the Jacobian of the right-hand side of (1) with respect to position and velocity is needed.

Rather than relying on numerical differentiation via divided differencing, the gradient and the Jacobian are evaluated by automatic differentiation (AD). As compared to numerical differentiation, the main advantage of this set of techniques is its accuracy. The derivative values obtained from AD are truncation error-free. From a computer science point of view, this technique is a program transformation changing the semantics of a given program. This means, a given piece of code evaluating some function is transformed into a new piece of code capable of evaluating not only the original function but also its derivatives. Software tools implementing the AD technology are available for different programming languages; see `www.autodiff.org`. AD-tools for MATLAB programs include [2, 3, 4]. All these tools are based on operator overloading. The goal of the present study is to report on the application of a recent AD-tool called ADiMat [5] which is unique in the sense of explicitly rewriting MATLAB programs for the computation of derivatives.

The structure of this contribution is as follows. In Sect. 2, a sketch of the computer model used to simulate the gravitational potential is given where, for the sake of simplicity, all other effects acting on the satellite are neglected. The computation of the gradient of the gravitational potential is described in Sect. 3. Here, a MATLAB program is transformed via automatic differentiation. Additionally, a more complex example, involving the computation of the Jacobian of the ODE (1), is presented in Sect. 4. Finally, in Sect. 5, results of numerical experiments are reported.

2 Gravitational Potential

One of the latest and most detailed descriptions of the gravitational field is the CHAMP model of the GeoForschungsZentrum Potsdam [6, 7] which is based on high accuracy measurements during the CHAMP campaign. Common descriptions of the gravitational field use the potential definition. The CHAMP gravitational potential U is given by a series expansion of the normalized Legendre functions $\bar{P}_{n,m}$, namely

$$U = \sum_{n=0}^{\infty} \frac{\mu \cdot a_e^n}{r^{n+1}} \cdot \sum_{m=0}^{n} \bar{P}_{n,m} \cdot \sin(\delta) \cdot \Big(c_{nm} \cdot \cos(m \cdot \lambda) + s_{nm} \cdot \sin(m \cdot \lambda) \Big). \tag{2}$$

Here, μ is the gravitational constant, a_e denotes the earth semi-major axis, and c_{nm} as well as s_{nm} are harmonic coefficients. Furthermore, λ, δ, and r denote the longitude, latitude, and distance between the point of origin and the emission point, respectively.

By adding up more terms to the series expansion of the gravitational model (2), its accuracy can be improved. Currently, the parameters are available up to the 120th degree. That is, $n = 0, 1, \ldots, 120$. For example there are 6642 harmonic coefficients at level $n = 80$ and 14762 for $n = 120$.

For the simulation of the flight trajectory the dynamics of the satellite can now be described by the motion equation (nonlinear ODE system of dimension 3)

$$m \cdot \ddot{\vec{r}} = \vec{F}_{\text{grav}} = m \cdot \nabla U, \qquad (3)$$

in which the gravitational potential

$$\nabla U = \left(\frac{\partial U}{\partial \lambda}, \frac{\partial U}{\partial \delta}, \frac{\partial U}{\partial r} \right) \qquad (4)$$

is needed for the gravitational acceleration acting on the satellite. The plot of the deviations from the zonal mean potential is depicted in Figure 1 for the CHAMP model of degree $n = 40$.

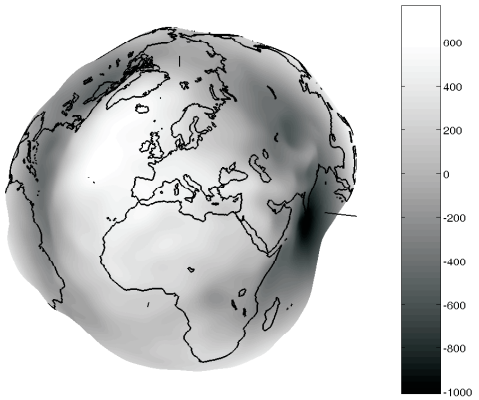

Figure 1: Deviation from the zonal mean potential of the earth in exaggerated representation.

3 Derivative Computation Using ADiMat

In order to compute the gradient ∇U given by (4) for a high precision orbit simulation, numerical differentiation by divided differencing cannot satisfy the required accuracy because of its truncation error. Therefore, automatic differentiation (AD) [8] is used to achive a trunction error-free evaluation of the gradient (4). In AD, a program is conceptually treated as the composition of a (huge) number of elementary operations such as binary addition or multiplication. The known derivatives of these elementary functions are mechanically accumulated by the chain rule.

The program written in MATLAB to compute the earth's gravitational potential is given in the form

```
[U] = potential(lambda, delta, r, ...)
```

where the variables `lambda`, `delta` and `r`, represent the coordinates of the operating point for the evaluation of the potential `U`. With the recent AD tool ADiMat [5] the code for the derivatives of the variable `U` with respect to `lambda`, `delta` and `r` is generated. The result is a new MATLAB program

```
[g_U, U] = g_potential(g_lambda, lambda, g_delta, delta,
                       g_r, r, ...)
```

capable of computing the gradient (4) in the variable `g_U` after initializing `g_lambda`, `g_delta` and `g_r` to the unit vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, respectively. This initialization of derivative variables is called seeding. In the so-called forward mode of AD on which ADiMat is based, the control flow of the derivative computations follows the control flow of the original computation. Therefore, the derivative variables `g_lambda`, `g_delta` and `g_r` have to be initialized because they are associated with input variables of the original code. When initializing a derivative variable `g_x`, a unity vector with a 1 at position i indicates that the directional derivative with respect to the variable `x` is stored in the ith column of each derivative variable. As a consequence, the derivatives of `U` with respect to `lambda` are available in the first column of `g_U` after the execution of `g_potential`.

4 A More Complex Example: Jacobian Matrix Computation

During a spaceborn mission of a satellite it is necessary to consider incoming sensor data such as GPS- or radar-signals. Often the GPS-signals are only available in unsatisfied frequency and precision with respect to the other sensor data frequencies to afford a data interpretation. Therefore, all missing positions between two GPS-points should be taken from a simulated trajectory.

For the simulation of the flight trajectory, the ODE (1) is transformed in a first order ODE system to be solved by standard ODE solvers. Thus, the dimension is doubled. To update the simulation of the orbit with incoming GPS-data, a discrete extended Kalman-Filter is used. For its implementation the Jacobian of the right-hand side of the first order ODE system is required. Here the important part is given by

$$J\left(\vec{r},\dot{\vec{r}}\right)=\sum_{\text{effect}}\frac{\partial\,\vec{a}_{\text{effect}}}{\partial[\vec{r},\dot{\vec{r}}]}\left(\vec{r},\dot{\vec{r}}\right). \tag{5}$$

In addition to the gravitational model some of the effects acting on the satellite mentioned in Sect. 1 are added to the simulation to increase the complexity of the model. In this example atmospheric drag, celestial gravitation, and the earth tides are considered. For the atmospheric drag model a constant air density is used instead of an enhanced atmospheric density model. Additionally, two coordinate transformations not mentioned before have to be considered for the Jacobian. The mathematical description of the effects is as follows:

$$\vec{a}^{\,\text{cart}}_{\text{drag}}=-\tfrac{1}{2}\cdot c_D\cdot\tfrac{A}{m}\cdot\rho\cdot\|\dot{\vec{r}}_a\|\cdot\dot{\vec{r}}_a$$

$\dot{\vec{r}}_a$ – speed vector relative to atmosphere dependent on $\vec{r}$, A – satellite cross sectional area, ρ – air density, c_D – drag coefficient,

$$\vec{a}^{\,\text{cart}}_{\text{cel}}=-\mu_{\text{cel}}\left(\frac{\vec{r}-\vec{r}_{\,\text{cel}}}{\left\|\vec{r}-\vec{r}_{\,\text{cel}}\right\|^3}+\frac{\vec{r}_{\,\text{cel}}}{\left\|\vec{r}_{\,\text{cel}}\right\|^3}\right)$$

$\vec{r}_{\,\text{cel}}$ – position of the attracting celestial body, μ_{cel} – gravitational constant of the attracting celestial body

$$\vec{a}^{\,\text{sph}}_{\text{ind}}=\tfrac{\mu_{\text{cel}}}{r_{\text{cel}}}\sum_{n=2}^{\infty}\tfrac{k_n\cdot a_e^{2n+1}}{r_{\text{cel}}\cdot r^{n+1}}\begin{pmatrix}\frac{\partial P_n(\Psi)}{\partial\Psi}\frac{\partial\Psi}{\partial\lambda}\\ \frac{\partial P_n(\Psi)}{\partial\Psi}\frac{\partial\Psi}{\partial\delta}\\ -\frac{n+1}{r}P_n\left(\Psi\right)\end{pmatrix}$$

r_{cel} – geocentric distance of the attracting celestial body, k_n – Love's number, $P_n\left(\Psi\right)$ – non normalized Legendre function of degree n

The computation of the Jacobian by applying ADiMat is conceptually done in the same way as described in Sect. 3. A corresponding description is therefore omitted for the sake of brevity. However, notice that the MATLAB code for the computation of the right-hand side of the ODE is much more complex than the one for the computation of the gravitational potential.

5 Numerical Results

In contrast to most other situations where the analytic derivatives of functions implemented by computer codes are extremely hard or even impossible to obtain, hand-coded implementations of the analytic derivatives are available for both, the gradient ∇U and the Jacobian J. In addition, all derivatives were compared with divided differencing. In all numerical experiments, the degree of the gravitational model is set to $n = 40$, representing a series expansion where the number of harmonic coefficients, namely 1722, is quite large.

The acceleration resulting from the automatically generated MATLAB code computing the gradient of the gravitational potential is depicted in Figure 2. The AD-generated code for ∇U_{AD} and the hand-coded program with the analytic derivatives for $\nabla U_{\text{analytic}}$ produce almost the same results. More precisely, the absolute error of the difference satisfies $||\nabla U_{\text{AD}} - \nabla U_{\text{analytic}}||_2 < 3 \cdot 10^{-15}$. Thus, the errors' magnitude lies in the range of the machine precision.

For the computation of the Jacobian J, the model complexity of the other considered effects is as mentioned in Sect. 4. Again, the AD approach provides absolute errors in the magnitude of the machine precision in the difference of automatically generated code and the analytic implementation. For each entry of the Jacobian, the error is less than 10^{-13}.

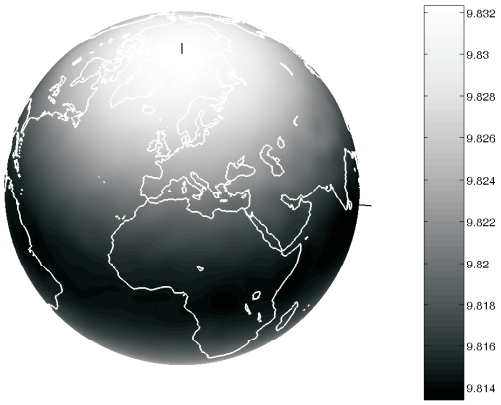

Figure 2: Gravitational acceleration on the earth's surface in m/s^2 of the CHAMP gravitational model at degree $n = 40$ calculated by automatic differentiation.

6 Concluding Remarks

Given a computer program to calculate the CHAMP gravitational potential of the earth and another one to calculate the Jacobian of the right-hand side of an ODE, new programs for the evaluation of the derivative are automatically generated. This is done by the new software tool ADiMat, implementing the forward mode of automatic differentiation for programs

written in MATLAB. The approach taken by ADiMat is based on source transformation and is unique in the sense of explicitly rewriting MATLAB code rather than making use of the existing language feature of operator overloading. The primary aim of this work is to demonstrate the feasibility of a source transformation approach in MATLAB. Another objective is to verify available hand-coded versions of the analytic gradient and Jacobian, which is typically cumbersome to implement and also prone to programming errors.

A further investigation – currently under work – is the use of the approach sketched in the present work for an even more complex model of the atmospheric density for which the analytic derivatives are not available.

References

[1] *Montenbruck, O. and Gill, E.:* Satellite Orbits: Models, Methods and Applications. Berlin: Springer, 2000.

[2] *Coleman, T. F. and Verma, A.:* ADMIT-1: Automatic differentiation and MATLAB interface toolbox. ACM Transactions on Mathematical Software, 26(1) (2000), pp. 150–175.

[3] *Rump, S. M.:* INTLAB – INTerval LABoratory. Developments in Reliable Computing (1999), pp. 77–104.

[4] *Shampine, L. F., Ketzscher, R. and Forth, S. A.:* Using AD to solve BVPs in Matlab. ACM Transactions on Mathematical Software, 31(1) (2005), pp. 79–94.

[5] *Bischof, C. H., Bücker, H. M., Lang, B., Rasch, A. and Vehreschild, A.:* Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs. Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002), pp. 65–72.

[6] *GeoForschungsZentrum Potsdam (GFZ):* Sektion 1.3: Gravitationsfeld und Erdmodelle. URL: http://www.gfz-potsdam.de/pb1/pg3/index_S13d.html.

[7] *Reigber, C., Lühr, H. and Schwintzer, P.:* First CHAMP Mission Results for Gravity, Magnetic and Atmospheric Studies. Berlin: Springer, 2003.

[8] *Griewank, A.:* Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Philadelphia: SIAM, 2000.