



سازمان صنایع دریایی
گروه صنایع شهید قربانی

عنوان فعالیت

نام طرح:	---	کد سند:	---
شماره ویرایش:	---	تاریخ صدور:	---
تعداد صفحات:	---	طبقه‌بندی:	---
نام و امضاء مجری فعالیت	نام و امضاء ناظر فعالیت	نام و امضاء مدیر گروه	نام و امضاء سرطراح / مدیر طرح
---	---	---	در فعالیت‌های طراحی سرطراح و در فعالیت‌های اجرایی مدیر طرح

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



بسمه تعالی

گروه صنایع شهید حسن قربانی

شناسنامه گزارش پروژه

دستورالعمل مستند سازی گزارش ها

عنوان (فارسی-لاتین):

Instruction of Reports Documentation

مسئول:	محمد محمدی	ویراست:	دوم
همکاران:	محمد محمدی	تعداد برگ:	52
مشاوران:	محمد محمدی	نوع گزارش:	دستورالعمل
مدیر گروه:	محمد محمدی	طبقه بندی:	محرمانه
مدیر طرح:	محمد محمدی	تاریخ ابلاغ:	91/1/20
طرح:		تاریخ اتمام:	91/5/28
شماره ثبت:		کد فعالیت:	

کلمات کلیدی: مستند سازی، استاندارد، شماره گذاری، روال.

چکیده:

در این گزارش، بررسی فنی فرایند پیاده سازی نرم افزاری بر پایه ی زبان پایتون برای تولید نقشه های گرادیان گرانش تشریح شده است. بدین منظور، ابتدا مفاهیم مورد نیاز در حوزه ی ژئودزی معرفی شده و سپس اهداف، کاربردها، معماری و فرایند پیاده سازی نرم افزار بررسی شده است. در ادامه، ساختار فنی پروژه، نحوه ی گسترش آن و وظایف هر دایرکتوری همراه با عملکردشان توضیح داده شده است. در پایان، جهت دسترسی به منابع، کدهای اصلی پروژه و دیتابیس های به کار رفته، لینک مخزن (ریپازیتوری) پروژه ارائه شده است.

فهرست مطالب

صفحه

عنوان

1- مقدمه.....	1
1-1- اهمیت مطالعات گرانس و گرادیان گرانس در ژئودزی.....	1
1-2- چالش‌های محاسباتی و داده‌ای.....	1
1-3- معرفی پروژهی <i>GeoGraphica</i>	2
1-4- اهداف و مزایای پروژه.....	2
1-5- مخاطبان و ذینفعان.....	3
2- معماری پروژه و نحوه پیکربندی.....	5
2-1- ساختار کلی پوشه‌ها و ماژول‌ها.....	5
2-2- الگوی ماژولار و مزایای آن.....	6
2-3- گردش کار کلی (کپسولی از معماری).....	7
3- ریز جزئیات پیاده سازی ماژول های هر دایرکتوری.....	9
3-1- جزئیات پیاده سازی ماژول های دایرکتوری <i>core</i>	9
3-1-1- <i>Constant</i> ماژول.....	9
3-1-2- <i>EGM96_data</i> ماژول.....	10
3-1-3- <i>functions</i> ماژول.....	10
3-1-4- <i>compute</i> ماژول.....	12
3-1-5- جمع بندی.....	13
3-2- جزئیات پیاده سازی ماژول های دایرکتوری <i>my_app</i>	14
3-2-1- پوشه‌ی <i>static</i>	14
3-2-2- پوشه‌ی <i>templates</i>	14
3-2-3- فایل <i>backend.py</i>	14
3-2-4- فایل <i>main.py</i>	15

15	3-2-5- گردش کلی کار در <i>my_app</i>
16	3-2-6- نکات برجسته در طراحی رابط کاربری (UI) و <i>Backend</i>
17	3-2-7- جمع‌بندی.....
18	3-3- جزئیات پیاده‌سازی ماژول‌های دایرکتوری <i>plotting</i>
21	3-4- جزئیات پیاده‌سازی ماژول‌های دایرکتوری <i>database</i>
22	3-5- جزئیات پیاده‌سازی ماژول‌های دایرکتوری <i>sources</i>
23	3-6- جزئیات پیاده‌سازی ماژول‌های دایرکتوری <i>tests</i>
25	3-7- جزئیات پیاده‌سازی ماژول‌های دایرکتوری <i>utils</i>
27	4- فایل <i>requirements.txt</i> و اهمیت آن در پروژه.....
29	5- ارزیابی و مقایسه نتایج محاسباتی <i>Python</i> و <i>Mathematica</i>

مقدمه



1- مقدمه

1-1- اهمیت مطالعات گرانش و گرادیان گرانش در ژئودزی

یکی از اساسی ترین مؤلفه ها در علوم زمین، شناخت و تحلیل میدان گرانش (*Gravity Field*) زمین است. این میدان، اطلاعات ارزشمندی را درباره ی توزیع جرم در پوسته ی زمین فراهم می کند. با پیشرفت فناوری های سنجش از راه دور و حسگرهای ماهواره ای، ابزارهای دقیق تری برای اندازه گیری و بررسی میدان گرانش در دسترس قرار گرفته اند. در این میان، **گرادیان گرانش (*Gravity Gradient*)** درک ما را از جزئیات ساختارهای زیرزمینی بالا می برد، زیرا به جای مقدار مطلق شتاب گرانی، تغییرات مکانی آن را بررسی می کند.

مطالعه ی گرادیان گرانش کاربردهای فراوانی دارد؛ از جمله:

- **اکتشافات معدنی و نفت و گاز** : مشاهده ی ناهنجاری های ریز در میدان گرانش و تطبیق آن با مدل های زیرزمینی.
- **زمین شناسی و مطالعات تکتونیکی** : تشخیص گسل ها ، دره های مدفون شده یا ساختارهای چین خورده.
- **هیدروژئولوژی** : ردیابی حفره ها یا سفره های زیرزمینی با مشاهده ی تغییرات جرمی.

1-2- چالش های محاسباتی و داده ای

از سوی دیگر، محاسبه ی گرادیان گرانش و ترسیم نقشه های مربوطه با مشکلاتی همراه است:

1. **حجم زیاد داده ها**: داده های ارتفاعی (*DEM*) یا مدل های دیجیتال زمین گاه به ده ها تا صدها مگابایت می رسند. در صورت افزایش دقت مکانی، پردازش این داده ها نیازمند روش های مؤثر برای مدیریت حافظه و توان محاسباتی بالاست.
2. **مدل های مختلف جاذبه** : مدل هایی نظیر *EGM96* ، *EGM2008* یا ماهواره های *Grace/GOCE* هر کدام ضرایب مختص خود را دارند. نگاشت درست این ضرایب به فرمول های محاسباتی و کنترل تصحیحات *C20*, *C40* و ... موضوعی تخصصی است.
3. **اینترپولاسیون و مصورسازی**: داده های گرانش اغلب در مختصات جغرافیایی مختلف مثلاً *Geocentric vs. Geodetic* ثبت شده اند. تبدیل این داده ها به نقشه هایی نرم و گویا، به تکنیک های اینترپولاسیون مانند *cubic griddata* و کتابخانه های گرافیکی نیاز دارد.



1-3- معرفی پروژهی *GeoGraphica*

با توجه به مسائل فوق، در پروژهی *GeoGraphica* تلاش شده است یک چارچوب نرم‌افزاری یکپارچه ایجاد شود تا متخصصان ژئودزی و دیگر رشته‌های مرتبط بتوانند در محیطی ساده و کاربردی، داده‌های گرانش و گرادیان گرانش را پردازش و مصورسازی کنند. این نرم‌افزار با زبان *Python* و بهره‌گیری از کتابخانه‌های *Numerical* نظیر *NumPy*، *SciPy* و *mpmath* توسعه یافته است. برخی از قابلیت‌های کلیدی عبارت‌اند از:

1. **پشتیبانی از مدل جهانی *EGM96*** : ضرایب این مدل در فایل *CSV* خوانده می‌شوند و کاربر می‌تواند تا درجه و مرتبه‌ی ۳۶۰ از آن استفاده کند.
2. **محاسبات موازی**: با بهره‌گیری از *joblib* و *Threading* در پایتون، روند محاسبات سنگین (*Fourier-based, Parker method*) سریع‌تر انجام می‌شود.
3. **رابط کاربری تحت وب و دسکتاپی**: با استفاده از *FastAPI* و کتابخانه‌ی *pywebview*، یک واسطه گرافیکی فراهم گردیده تا کاربر بتواند محدوده‌ی جغرافیایی، رزولوشن و سایر پارامترهای ورودی را از طریق یک فرم دریافت کند. نتیجه‌ی نهایی (نقشه‌های کانطور) نیز در قالب تصاویر یا *PDF* ارائه می‌شود.
4. **امکان انتخاب *Colormap*** : کاربران برای نمایش داده‌ها در قالب‌های رنگی مختلف، می‌توانند از *Colormap* های متعدد *Matplotlib* بهره‌برند.

1-4- اهداف و مزایای پروژه

- **سرعت و سهولت کار**: به‌جای استفاده از چندین نرم‌افزار پراکنده مثل *MATLAB*، *ArcGIS*، *GMT* و ...، تلاش شده که فرایند بارگذاری داده‌ی ارتفاعی، محاسبه‌ی گرادیان و رسم نقشه‌ها در یک بسته‌ی واحد قابل انجام باشد.
- **افزودگی و ماژولار بودن**: معماری پروژه به شکل ماژولار طراحی شده است تا در آینده بتوان مدل‌های جدیدتر (*EGM2008*) یا الگوریتم‌های موازی قدرتمندتر را نیز به آن اضافه نمود.
- **ظرفیت سفارشی‌سازی**: کاربران حرفه‌ای می‌توانند به صورت عمیق با بخش *Computations* کار کنند یا حتی توابع محاسباتی خود را در آن جایگزین نمایند.

5-1- مخاطبان و ذینفعان

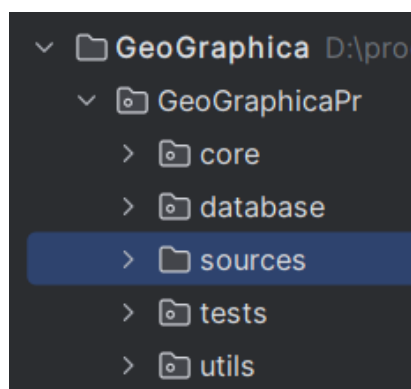
- **ژئودزیست‌ها و ژئوفیزیکدان‌ها:** که روزانه با داده‌های گرانی و مدل‌های ارتفاعی سروکار دارند.
- **مهندسين اکتشاف معدنی و نفتی:** برای ارزیابی و تحلیل منابع زیرزمینی و ذخایر اقتصادی.
- **مراکز پژوهشی و دانشگاهی:** برای آموزش و پروژه‌های تحقیقاتی در حوزه‌ی علوم زمین، مدل‌سازی اقیانوس و

معماری پروژه و نحوه پیکربندی

2- معماری پروژه و نحوه پیکربندی

2-1- ساختار کلی پوشه‌ها و ماژول‌ها

پروژه‌ی حاضر تحت عنوان موقت *GeoGraphica* در حالت کلی به صورت ماژولار طراحی شده و از چندین پوشه (*Directory*) اصلی تشکیل شده است. هدف از این ساختار ماژولار، تفکیک مسئولیت‌ها (*Separation of Concerns*) و نگهداری راحت‌تر کد در بخش‌های مرتبط است.



مهم‌ترین بخش‌های این پروژه عبارت‌اند از:

1. *core*

- *computations*: در این پوشه توابع و کلاس‌های اصلی محاسبات گرادیان گرانج قرار گرفته‌اند. برای نمونه، *compute*, *Constant*, *EGM96_data*, *functions*.
- *my_app*: ماژول مربوط به رابط کاربری و *Backend* در این پوشه قرار دارد. فایل‌های کلیدی شامل *main.py* (ایجاد *WebView* و مدیریت رابط دسکتاپی) و *backend.py* (راه‌اندازی سرور *FastAPI* و تعریف اندپوینت‌ها) هستند.
- *Plotting*: وظیفه‌ی رسم و مصورسازی نقشه‌های گرادیان گرانج، در اینجا پیاده‌سازی شده است. فایل *plotter.py* از کتابخانه‌ی *Matplotlib* بهره می‌گیرد تا داده‌های محاسبه‌شده را به صورت *Contour Plot* نمایش دهد و خروجی تصویری در قالب *PNG/PDF* ارائه کند.



2. database

در این پوشه داده‌های ضروری پروژه، از جمله فایل‌های مرتبط با مدل *EGM96* (مانند *EGM96.csv*، *EGM96.gfc*) و فایل‌های ارتفاعی (مثلاً *Elev_Matrix_Output.xlsx*) نگهداری می‌شوند. همچنین داده‌های خام (*DEM*) در زیرفولدرهای مربوطه (مانند *gt30e020n40_dem*) قرار گرفته‌اند.

3. sources

این فولدر شامل فایل‌ها و منابعی است که ممکن است در مراحل اولیه و میانی توسعه جمع‌آوری شده باشند (برای مثال داده‌های مرحله‌ی اول و دوم پروژه، یا اسناد مربوط به ارزیابی داده‌ها). در واقع، محلی برای نگهداری منابع اضافی یا کدهای نمونه‌ی فازهای قبلی است.

4. tests

فایل‌ها و اسکریپت‌های تست پروژه در اینجا نگهداری می‌شوند. برای نمونه، *test_plooter.py* شبیه‌سازی داده‌های محاسباتی و تست فرایند رسم نمودار را انجام می‌دهد. همچنین فایل‌های دیگری مانند *TxxLinearPlotter.py* برای مقایسه‌ی خروجی‌ها در یک محیط مجزای تحلیل استفاده شوند.

5. utils

این پوشه حاوی ابزارها و ماژول‌های کمکی (*Utility Modules*) است که در بخش‌های مختلف پروژه مورد استفاده قرار می‌گیرند. از جمله مهم‌ترین فایل‌های این بخش، *files_paths.py* است که مسیرهای پیش‌فرض فایل‌های ورودی/خروجی نظیر *CSV*، دیتاست‌های ورودی و ... را مدیریت می‌کند.

2-2- الگوی ماژولار و مزایای آن

با پیاده‌سازی این ساختار ماژولار، هر بخش از پروژه وظیفه‌ی خاص خود را بر عهده دارد:

- **جداکردن منطق محاسبات از رابط کاربری**: ماژول‌های محاسباتی در *computations* متمرکز شده‌اند تا توسعه‌دهندگان بتوانند الگوریتم‌های جدید (مانند مدل‌های دیگر یا روش‌های پردازشی متفاوت) را به راحتی اضافه کنند، بدون آن‌که نیاز باشد در کد رابط کاربری تغییری بدهند.
- **یکپارچگی ساده داده‌ها**: تمامی فایل‌ها و مدل‌های ارتفاعی در پوشه‌ی *database* قرار دارند و با استفاده از ماژول *files_paths.py* به‌طور متمرکز مدیریت می‌شوند؛ این مسئله نگهداری و به‌روزرسانی داده‌ها را آسان‌تر می‌کند.



- امکان تست و نگهداشت آسان : با تفکیک فایل های تست (*tests*) ، فرایند خطایابی و ارزیابی عملکرد بخش های مختلف ساده تر شده و می توان هر کدام از توابع اصلی را به صورت جداگانه تست یا پرو فایل کرد.
- قابلیت گسترش پذیری : ماژول های *plotting* و *my_app* را می توان برای اضافه کردن قابلیت های جدید (مثلاً رسم نمودارهای سه بعدی یا رابط کاربری پیشرفته تر) توسعه داد، بی آنکه ساختار کل پروژه دستخوش تغییر اساسی شود.

3-2- گردش کار کلی (کپسولی از معماری)

1. دریافت محدوده های جغرافیایی و سایر پارامترها (از طریق رابط وب یا دسکتاپ)
 2. فراخوانی ماژول محاسبات *compute_gradients.py* یا توابع موجود در *functions.py* برای محاسبه ی ماتریس های گرادیان گرانش با استفاده از داده های ارتفاعی و مدل *EGM96*
 3. ذخیره ی خروجی در قالب *CSV* یا سایر فرمت های واسط
 4. رسم نقشه ها با استفاده از ماژول *plotter.py* و ارائه ی نتیجه به کاربر (داخلی یا تحت وب)
- این الگو امکان جداسازی وظایف (*Separation of Concerns*) را فراهم کرده و به ارتقای خوانایی و نگهداری کد کمک شایانی کرده است.

ریز جزئیات پیاده سازی ماژول های هر دایرکتوری



3- ریز جزئیات پیاده سازی ماژول های هر دایرکتوری

3-1- جزئیات پیاده سازی ماژول های دایرکتوری *core*

3-1-1- *Constant* ماژول

این ماژول مسئول نگهداری و مدیریت ثابت های فیزیکی و هندسی مورد نیاز در محاسبات است. ساختار آن به صورت یک کلاس به نام *Constants* طراحی شده که در سازنده (*Constructor*) خود، متغیرهای زیر را مقداردهی می کند:

- *EOTVOS* : مقدار ثابت برای تبدیل گرادیان به واحد *Eötvös* تقریباً 10^{-9} .
- *GM* : ثابت جهانی *G* ضربدر جرم زمین (حدود $14^{10} \times 3.986004418$).
- *a* : شعاع نیم قطر استوا (*Earth's semi-major axis*) در بیضوی مرجع *WGS84* (حدود متر 6378137).
- *Nmax* : بالاترین درجه و مرتبه برای مدل *EGM96* (در اینجا ۳۶۰).
- *PRECISION* : دقت محاسباتی برای کتابخانه *mpmath*.
- *f* : مقدار تختی بیضوی زمین (*Flattening*)؛ در *WGS84* برابر با $1/298.257223563$.
- *e2* : مجذور خروج از مرکز بیضوی (*eccentricity squared*).
- *h* : ارتفاع پیش فرض (مثلاً 100 متر) برای محاسبات بالاتر از بیضوی.
- *G* : ثابت جهانی گرانش (6.6742×10^{-11}).
- *p* : چگالی متوسط یا دلخواه در اینجا 2670 kg/m^3 .

کلاس *Constants* برای دسترسی به این ثابت ها، یکسری متدها و *Property* ها ارائه می کند (نظیر *get_a()*، *get_h()*، *EOTVOS*، و ...) تا در سایر بخش های پروژه (ماژول های محاسباتی و *plotting*) قابل فراخوانی باشند. این روش باعث می شود در صورت نیاز به تغییر ثابت ها یا به روزرسانی به مدل جدیدتر، تنها همین فایل اصلاح شود.



3-1-2- *EGM96_data* ماژول

در این ماژول، ضرایب مدل جاذبه‌ی زمین *EGM96* (تا درجه و مرتبه‌ی ۳۶۰) از یک فایل CSV بارگذاری می‌شود. مراحل اصلی عبارت‌اند از:

1. تعریف مسیر *csv_file_path*: مسیر فایل *EGM96.csv* در پوشه‌ی *database*
2. ایجاد یک دیکشنری *global* با نام *data* که به صورت $data[n][m] = [C_{nm}, S_{nm}]$ مقداردهی می‌شود (n درجه، m مرتبه)

3. اصلاح برخی ضرایب نظیر *C20, C40, C60* و... با توجه به تصحیحات لازم (همانند تفکیک چرخش یا مسائل ژئودتیک).

در پایان، دیکشنری *data* حاوی تمام ضرایب ضروری برای محاسبه‌ی پتانسیل و گرادیان گرانش در مدل *EGM96* می‌شود. دیگر ماژول‌ها مانند *functions.py* از این دیکشنری برای فراخوانی ضرایب استفاده می‌کنند.

3-1-3- *functions* ماژول

این ماژول قلب محاسبات گرادیان گرانش به روش *EGM96* است. ساختار کلی آن عبارت است از:

1. بارگذاری داده‌ها و ثابت‌ها:
در ابتدای فایل، ماژول *Constant.py* و *EGM96_data.py* ایمپورت شده و از آن‌ها مقادیر *EOTVOS, Gm, Nmax* و... دریافت می‌شود.
2. مقادیر و متغیرهای سراسری:
 - *legendre_data*: برای کش ذخیره‌ی چندجمله‌ای‌های لژاندر (*Legendre Polynomials*)
 - *part_two*: یک متغیر سراسری برای تجمع نتایج موقت در حین محاسبات چندبخشی (*Threading*)
 - متغیر *chunk_size* و یک *Lock* از کتابخانه‌ی *threading* جهت ایمنی در دسترسی

همزمان.

3. توابع کمکی:

- `convert_seconds()`: تبدیل ثانیه به روز، ساعت، دقیقه، ثانیه (برای گزارش زمان).
- `legendre_data_existence()` و `retrieve_legendre_data()`: مدیریت کش
- چندجمله‌ای‌های لژاندر محاسبه شده.
- `C_nm()` و `S_nm()`: بازیابی ضرایب C و S از دیکشنری `EGM96_data`.
- مجموعه توابع `a_nm`, `b_nm`, `c_nm`, `d_nm`, ... : ضرایب مختلف مورد نیاز در محاسبات سری لژاندر نرمال شده.
- `normal_pnm()`: محاسبه‌ی چندجمله‌ای‌های *Associated Legendre* به صورت نرمال شده. (Normalized)
- ...

4. توابع مربوط به محاسبه‌ی Txx , Txy , Txz , ...

هر مؤلفه‌ی تسور یک تابع اصلی مثلاً `Txx_function` دارد که داده‌های ژئودتیک r , φ , λ را گرفته، چندنخی `MultiThreading` اجرا می‌شود و در نهایت مقداری از جنس `mpf` (دقت بالای `mpmath`) برمی گرداند.

5. توابع `compute_Txx_chunk`, `compute_Txy_chunk`, ...

این توابع درون حلقه‌ای برای محاسبات موازی طراحی شده‌اند. هرکدام در بازه‌ای از درجات n و مراتب m محاسبه را انجام می‌دهند، سپس در متغیر سراسری `part_two` ذخیره می‌شوند (قفل `threading Lock` برای جلوگیری از رخ دادن `race condition` استفاده شده است)

6. خروجی نهایی

توابعی مانند `Txx_function` از تجمیع $(part_one * part_two)$ و پاک کردن کش لژاندر (در صورت نیاز) مقدار نهایی را برمی گردانند.

3-1-4- *compute* ماژول

این فایل (با متد *compute_gradients*) ترکیبی از منطق محاسبات پارکر (Fourier-based) و فراخوانی توابع EGM96 است. دو بخش اصلی در آن قابل تفکیک است:

1. محاسبات *Parker* :

- دریافت محدوده‌ی طول و عرض جغرافیایی و رزولوشن.
- تبدیل این محدوده‌ها به نقاط مشبک (*Grids*) با استفاده از *NumPy*.
- بستن سری فوریه (*Fourier Series*) برای ماتریس ارتفاعی و محاسبه‌ی مؤلفه‌های گرادیان مانند *TxxParker*، *TxyParker* و

2. محاسبات *EGM96* :

- استفاده از توابع موازی (*Parallel*) کتابخانه‌ی *joblib* برای فراخوانی *Txx_function* و *Txy_function* در نقاط مختلف.

➤ موازی‌سازی (*Parallel Computing*) :

استفاده از *Threading* قفل (*Lock*) برای ایمنی دسترسی به متغیر

سراسری (*part_two*) .

استفاده از *joblib* در مرحله‌ی محاسبه‌ی *TxxEGM96* و بقیه‌ی مؤلفه‌ها.

این روش مخصوصاً برای *Data Parallelism* مناسب است و سرعت

اجرای پروژه را بالا می‌برد.

- تولید ماتریس‌های خروجی *TxxEGM96*، *TxyEGM96* و

3. خروجی *CSV* :

- تمام مؤلفه‌های گرادیان (چه پارکر و چه EGM96) جداگانه در فایل‌های *CSV* صادر می‌شوند.

- جمع مؤلفه‌های پارکر و EGM96 به‌عنوان مؤلفه‌ی نهایی (*Total*) نیز در *CSV* های مجزا ذخیره می‌شود.

- ساختار فایل‌های *CSV* طوریست که سطر اول، مقادیر طول جغرافیایی را نشان می‌دهد و ستون اول، مقادیر عرض جغرافیایی .



4. مدیریت مسیرها:

○ از *utils* ماژول *files_paths.py* برای تعیین یا ایجاد پوشه‌ی مقصد (*base_path*) و نام فایل‌های خروجی *TxxTOTAL.csv* و ... استفاده می‌شود.

5. نحوه فراخوانی:

این تابع کلیدی از طریق اندپوینت */plot* در *backend* بخش (*my_app*) فراخوانی می‌شود که در آن کاربر مقادیر ورودی‌اش را مشخص می‌کند. در نهایت، نتایج *CSV* تولیدشده یا نقشه‌های آماده‌ی *Plotting* خواهند بود.

5-1-3- جمع‌بندی

پوشه‌ی *core* در حقیقت ستون فقرات (*Backbone*) محاسباتی و منطقی پروژه‌ی *GeoGraphica* است. هرآنچه مربوط به تولید داده‌های نهایی گرادیان گرانش می‌شود، اعم از پردازش سری‌های لژاندر، مدل *EGM96*، روش پارکر، تبدیل مختصات، و خروجی فایل‌های *CSV* در این پوشه قرار دارد. این تمایز بین ماژول‌های محاسباتی (*computations*) و ماژول رابط کاربری (*my_app*) باعث شده کدها ساختارمند، قابل فهم و نگهداری پذیر باشند.



3-2- جزئیات پیاده‌سازی ماژول‌های دایرکتوری *my_app*

هدف کلی این پوشه فراهم کردن رابط کاربری (UI) و کنترل‌کننده‌ی (Controller) برنامه است؛ یعنی بخش‌هایی که کاربر با آن‌ها تعامل می‌کند و همچنین بخشی که درخواست‌های وی را پردازش می‌کند. ساختار اصلی این فولدر به شکل زیر است:

3-2-1 پوشه‌ی *static*

حاوی فایل‌های استاتیک (Front-end) شامل:

- اسکریپت اصلی جاوااسکریپت برای ارسال درخواست به سرور با استفاده از متد *fetch*، فراخوانی اندپوینت‌ها (مانند */plot* و */select_directory*)، بارگذاری فهرست *Colormap* و ...
- *styles.css*: استایل‌ها و قالب‌بندی ظاهری (رنگ، پس‌زمینه، اندازه المان‌ها و ...) که برای صفحه‌ی HTML اعمال می‌شود.

3-2-2 پوشه‌ی *templates*

شامل فایل‌های قالب HTML، به‌ویژه *index.html* که رابط وب اصلی پروژه را فراهم می‌کند. در این فایل، ورودی‌ها (محدوده‌ی طول و عرض جغرافیایی، رزولوشن و ...) از کاربر دریافت می‌شوند و سپس با JavaScript به سرور ارسال می‌گردند.

3-2-3 فایل *backend.py*

- یک سرور *FastAPI* ایجاد می‌کند که اندپوینت‌های مختلف را در اختیار می‌گذارد. مهم‌ترین بخش‌ها عبارت‌اند از:
 - *@app.get("/")*: بازگشت فایل *index.html* برای نمایش صفحه‌ی اصلی برنامه.
 - *@app.post("/plot")*: دریافت *JSON* پارامترهای کاربر (طول و عرض جغرافیایی، رزولوشن، مسیر ذخیره‌سازی و ...) و اجرای محاسبات با متد *compute_gradients*.
 - *@app.get("/select_directory")*: فراخوانی یک دیالوگ انتخاب پوشه به کمک *tkinter* و برگرداندن مسیر انتخاب‌شده به فرانت‌اند.

- اندپوینت‌های دیگر مانند *colormaps* / برای ارائه‌ی فهرست رنگ‌ها در *Matplotlib* و *plots* / برای نمایش تب‌بندی‌شده‌ی تصاویر خروجی.
- با استفاده از *FileResponse* و اندپوینت‌هایی مانند *@app.get("/plots/image/{plot_type}")*، فایل‌های تصویری تولیدشده *Total*، *EGM96* و *Parker* برای نمایش در مرورگر بازگردانده می‌شوند.
- **مدیریت CORS** : از *fastapi.middleware.cors.CORSMiddleware* برای دسترسی همه‌ی منابع (*allow_origins=["*"]*) و جلوگیری از خطاهای *Cross-Origin* استفاده شده است.
- **پیکربندی StaticFiles** : با دستور *app.mount("/static", ...)* پوشه‌ی *static* را در مسیر */static* منتشر (مونت – *mount*) می‌کند تا فایل‌های جاوااسکریپت و *CSS* در مرورگر قابل دسترسی باشند.

3-2-4- فایل *main.py*

- در این فایل، برنامه با استفاده از کتابخانه‌ی *webview* یک پنجره‌ی دسکتاپی باز می‌کند که درون آن صفحه‌ی وب سرور *FastAPI* در حال اجرا بارگذاری می‌شود. این رویکرد اجازه می‌دهد تا کاربر برنامه را مشابه یک نرم‌افزار دسکتاپ با پنجره‌ی گرافیکی تجربه کند، درحالی‌که هسته‌ی اصلی آن همچنان از مزایای معماری تحت وب بهره می‌برد.
- تابع *start_server()* از *backend.py* در یک *Thread* جداگانه اجرا می‌شود. سپس دستور *webview.create_window(...)* پنجره‌ی جدیدی با *URL* پیش‌فرض (مثلاً <http://127.0.0.1:5000>) باز کرده و با فراخوانی *webview.start()* آن را نمایش می‌دهد.

3-2-5- گردش کلی کار در *my_app*

1. اجرای *main.py* :

- سرور *FastAPI* تعریف‌شده در *backend.py* در پورت 5000 راه‌اندازی می‌شود.
- یک پنجره وب (*WebView*) در اندازه‌ی ۸۰۰×۱۲۰۰ پیکسل باز می‌گردد و به آدرس <http://127.0.0.1:5000> متصل می‌شود.

2. بارگذاری صفحه‌ی اصلی:

- فایل `index.html` از پوشه‌ی `templates` دریافت می‌شود و المان‌هایی مانند `input` برای `Latitude` و `Longitude` و دکمه‌ی `Plot` را در اختیار کاربر قرار می‌دهد.
- فایل‌های `CSS` و `JS` در پوشه‌ی `static` نیز بارگذاری می‌شوند (مانند `styles.css`) و (`script.js`)

3. تکمیل اطلاعات توسط کاربر:

- محدوده‌های طول و عرض جغرافیایی، رزولوشن، مسیر ذخیره‌سازی و نوع `Colormap` انتخاب می‌شود.
- با کلیک روی دکمه‌ی `PLOT`، تابع `plotGraph()` در `script.js` اجرا و یک درخواست `POST` به اندپوینت `/plot` ارسال می‌شود.

4. دریافت درخواست در `backend.py`:

- در متد `@app.post("/plot")`، داده‌های دریافتی مدل `InputData` را تشکیل می‌دهند.
- بر اساس این اطلاعات، محاسبات انجام می‌شود.
- خروجی نهایی (ماتریس‌ها یا فایل‌های `CSV` و تصاویر رسم‌شده) تولید شده و مسیر فایل‌های تصویری در متغیرهای سراسری `total_map_path`، `egm96_maps_path`، `parker_maps_path` ذخیره می‌شود.

5. نمایش نقشه‌ها:

- کاربر به آدرس `/plots` منتقل می‌شود، صفحه‌ای تب‌بندی‌شده را مشاهده می‌کند که با کلیک هر تب، تصویر مربوط (`/plots/image/<plot_type>`) از سرور دریافت و نمایش داده می‌شود.

3-2-6- نکات برجسته در طراحی رابط کاربری (UI) و Backend

- تلفیق رابط وب و دسکتاپ: با استفاده از کتابخانه‌ی `pywebview`، برنامه به کاربر جلوه‌ی یک نرم‌افزار دسکتاپ می‌دهد، در حالی که هسته و منطق آن به شکل وب‌سرویس `FastAPI` پیاده‌سازی شده است.

- مدیریت ساده فایل‌های استاتیک با `app.mount("/static", ...)` و فایل‌هایی مانند `script.js` نگهداری و نسخه‌بندی `JS/CSS` ساده‌تر شده است.
- اتصال سریع به ماژول‌های محاسباتی: اندپوینت `/plot` مستقیماً می‌تواند با تابع `compute_gradients()` در `compute.py` ارتباط بگیرد و نتیجه را به شکل تصویر نهایی (PDF/PNG) یا CSV برگشت دهد.
- قابلیت انتخاب **Colormap**: از اندپوینت `/colormaps` برای بارگیری لیست رنگ‌های `Matplotlib` و نمایش آن در منوی کشویی `HTML` استفاده می‌شود. این قابلیت به کاربر آزادی در انتخاب پالت رنگ برای نقشه‌ها می‌دهد.
- دیالوگ انتخاب پوشه: اندپوینت `/select_directory` از کتابخانه‌ی استاندارد `tkinter` برای باز کردن دیالوگ فایل استفاده می‌کند و به کاربر اجازه می‌دهد مسیر ذخیره‌سازی فایل‌های خروجی را مستقیماً از طریق `GUI` انتخاب کند (به جای تایپ دستی).

7-2-3- جمع‌بندی

دایرکتوری `my_app` در نقش لایه‌ی ارائه (`Presentation Layer`) و کنترل‌کننده (`Controller`) است. از یک سو با فایل‌های `HTML/JS/CSS` برای نمایش و دریافت پارامترها سروکار دارد و از سوی دیگر با اندپوینت‌های `FastAPI` در `backend.py` محاسبات یا فراخوانی‌های لازم را انجام می‌دهد. این معماری ماژولار و جدا از منطق محاسباتی، نگهداری و توسعه‌ی نرم‌افزار را بسیار آسان کرده است.



3-3- جزئیات پیاده‌سازی ماژول‌های دایرکتوری *plotting*

این دایرکتوری محل قرارگیری کدهای مرتبط با ترسیم (*Plotting*) و مصورسازی خروجی محاسبات پروژه است. ماژول اصلی (*plotter.py*) وظیفه‌ی رسم داده‌های محاسبه‌شده (نظیر مؤلفه‌های گرادین گرانی T_{xx} , T_{xy} , T_{xz} , T_{yy} , T_{yz} , T_{zz}) را به صورت کانتورپلات دوبعدی بر عهده دارد. در ادامه ساختار کلی و عملکرد این ماژول تشریح شده است.

۱. کتابخانه‌های مورد استفاده

- **Matplotlib** : اصلی‌ترین ابزار پایتون برای ترسیم نمودار. در ابتدای ماژول با دستور `matplotlib.use("Agg")`، *mode* رندر را روی حالت بدون واسطه گرافیکی قرار می‌دهیم. این کار به ما اجازه می‌دهد حتی روی سرور یا محیطی بدون مانیتور نیز خروجی تصویری بگیریم.
- **scipy.interpolate.griddata** : از این تابع برای اینترپوله کردن داده‌ها استفاده می‌شود تا نمودارهای کانتور، ظاهری نرم‌تر (*Smooth*) داشته باشند.
- **GeoGraphicaPr.core.computations.functions.convert_seconds** : تابعی که برای گزارش مدت زمان اجرای ترسیم استفاده می‌شود (تبدیل ثانیه به روز، ساعت، دقیقه و ثانیه).

۲. تابع کلیدی *plot_matrices(...)*

امضای تابع:

```
3 usages new *
def plot_matrices(
    header_title,
    matrices,
    titles,
    phi_range_deg,
    landa_range_deg,
    selected_colormap,
    start_time,
    saved_path,
    contour_levels=10,
    axis_resolution=0.5,
    grid_size=200
):
    """
    Plots multiple 2D matrices (e.g., total gravity gradients) using contour plots,
    with optional interpolation for smoother visualization.

    ...
    """
```




- در ادامه پارامترهای ورودی مهم توضیح داده شده است:
1. **header_title** : عنوان کلی نمودار مثلاً "Gravity Gradient Total Maps" که روی شکل و همچنین در نام فایل خروجی درج می‌شود.
 2. **Matrices** : یک لیست از ماتریس‌های دوبعدی مثل $[TxxTOTAL, TxyTOTAL, \dots]$ ، که هر کدام نشان‌دهنده‌ی مقادیر گرادیان در نقاط مختلف جغرافیایی هستند.
 3. **Titles** : عناوین متناظر هر ماتریس مثل $["Txx Total", "Txy Total", \dots]$.
 4. **phi_range_deg, landa_range_deg** : بردارهای مختصات جغرافیایی (عرض و طول جغرافیایی) به درجه .
 5. **selected_colormap** : نوع **Colormap** انتخاب‌شده توسط کاربر (مثلاً "viridis", "plasma" و ...). در صورت عدم ارسال مقدار، یک کولورمپ سفارشی ساخته می‌شود.
 6. **start_time** : زمان شروع (مثلاً $time.time()$) برای اندازه‌گیری مدت کل ترسیم.
 7. **saved_path** : مسیری که خروجی نمودارها بصورت **PNG** و **PDF** در آن ذخیره می‌شود.
 8. **contour_levels** : تعداد محدوده‌ها در ترسیم کانتور.
 9. **axis_resolution** : فواصل پیش‌فرض برای قرارگیری **tick** روی محور طول و عرض جغرافیایی.
 10. **grid_size** : تعداد نقاط برای اینترپوله کردن داده‌ها (اگر عدد بیشتر باشد، نمودار نرم‌تر اما زمان اجرا طولانی‌تر خواهد شد).

۳. فرایند درون تابع

1. **تابع کمکی $interpolate_data(...)$**
 - با استفاده از **griddata** داده‌های ورودی (ماتریس) را روی یک شبکه‌ی گسسته‌ی ریزتر با ابعاد **grid_size** اینترپوله می‌کند. این کار به ایجاد نمودارهای کانتور نرم‌تر و زیباتر منجر می‌شود.
2. **تعریف $Colormap$ سفارشی (در صورت نبود ورودی)**
 - در صورت خالی بودن **selected_colormap** ، یک لیست رنگ ساده تعریف و با **LinearSegmentedColormap.from_list** کولورمپ می‌سازد.



3. ایجاد زیر نمودارها (Subplots)

- دستور `plt.subplots(2, 3, ...)` شش محور (*Axes*) در قالب دو ردیف و سه ستون ایجاد می کند.
- در یک حلقه ی `for idx, ax in enumerate(axes.flat)` : هر ماتریس به همراه عنوان مربوط رسم می گردد.

4. رسم کانتور

- با دستور `ax.contourf(...)` داده های اینترپوله شده ی هر ماتریس به صورت نقشه ی کانتور رنگی پر شده ترسیم می شود.
- با `ax.contour(...)` خطوط کانتور نیز به صورت مشکی اضافه می شود (خود این خطوط، تقسیم بندی مقادیر را مشخص می کنند).

5. تنظیمات ظاهری

- محورهای x (طول جغرافیایی) و y (عرض جغرافیایی) بر اساس `landa_range_deg` و `phi_range_deg` مقداردهی می شوند.
- برای هر نمودار عنوان، برجسب محور، شبکه ی (*Grid*) نازک با رنگ سیاه و شفافیت نسبی تعیین می شود.
- یک *Colorbar* در کنار هر محور قرار داده می شود که مقیاس رنگ را نشان می دهد.

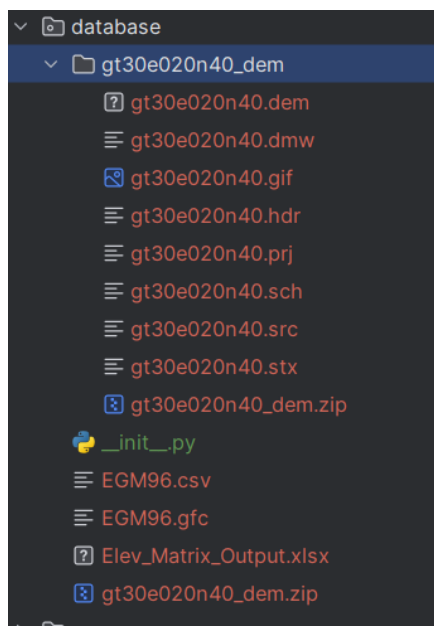
6. عنوان کلی و ذخیره نمودار

- دستور `plt.suptitle(...)` یک عنوان اصلی برای کل شکل (*Figure*) تعیین می کند.
- زمان سپری شده از `start_time` محاسبه و در خروجی چاپ می شود.
- فایل نهایی در دو قالب *PDF* و *PNG* در مسیر تعیین شده (`saved_path`) ذخیره می گردد.
- در پایان تابع، با `plt.close(fig)` شکل آزاد می شود تا از مصرف حافظه ی غیر ضروری جلوگیری شود.

7. خروجی

- در نهایت مسیر فایل *PNG* مثلاً `Gravity Gradient Total Maps.png` بازگردانده می شود تا در صورتی که مازول دیگری بخواهد فایل تصویر را مستقیماً استفاده کند، به آن دسترسی داشته باشد.

3-4- جزئیات پیاده سازی ماژول های دایرکتوری *database*



در این پوشه داده های ضروری پروژه، از جمله فایل های مرتبط با مدل *EGM96* (مانند *EGM96.csv*، *EGM96.gfc*) و فایل های ارتفاعی (مثلاً *Elev_Matrix_Output.xlsx*) نگهداری می شوند. همچنین داده های خام (*DEM*) در زیر فولدرهای مربوطه (مانند *gt30e020n40_dem*) قرار گرفته اند.

افزایش راندمان پروژه و بهینه کردن پروسه جنریت کردن نقشه های گرادیان گرانش

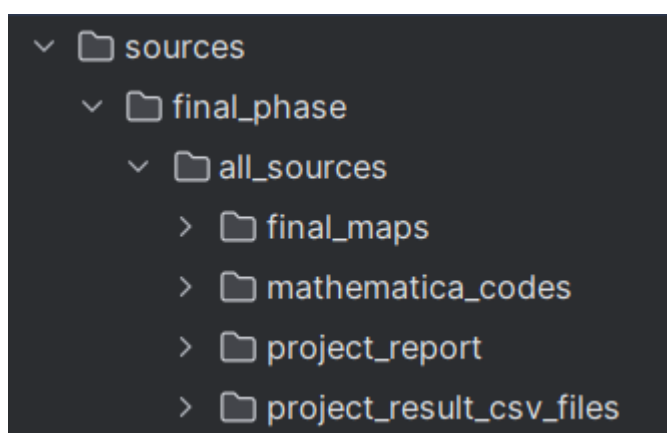
یک سری داده های مورد نیاز در محاسبه (مانند داده های مدل *EGM96*) از قبل از فایل اصلی با فرمت *gfc* استخراج شده و سطر و ستون های مدنظر در یک فایل جدید با فرمت *csv* ذخیره شدند.

این فایل پیش محاسبه و استخراج شده *EGM96.csv* در دایرکتوری *core/computations* در ماژول *EGM96_data.py* مورد استفاده قرار میگیرد.

همه داده ها و اطلاعات ذخیره شده در این فایل در یک دیکشنری ذخیره میشود که هرگاه نیاز داشتیم به ضرایب *Cnm*، *Snm* (داده های مدل *EGM96*) بتوانیم با مرتبه زمانی $O(1)$ از طریق دیکشنری تعریف شده در پایتون، به آنها دسترسی داشته باشیم.

همچنین اطلاعات داده های ارتفاعی فایل *gt30e020n40_dem.zip* از قبل استخراج شده و در فایل با عنوان *Elev_Matrix_Output.xlsx* ذخیره کردیم، تا هر دفعه برای جنریت کردن نقشه های گرادیان گرانش نیاز به استخراج و تحلیل اطلاعات فایل زیپ شده مدل ارتفاعی *gt30e020n40_dem* نباشد.

3-5- جزئیات پیاده‌سازی ماژول‌های دایرکتوری *sources*



دایرکتوری *sources* یکی از بخش‌های کلیدی پروژه است که شامل داده‌های مورد استفاده، کدهای مرتبط با محاسبات و نتایج پردازش شده است.

این دایرکتوری در مسیر

sources/final_phase/all_sources سازماندهی

شده و شامل چهار زیر دایرکتوری اصلی است که در ادامه توضیح داده می‌شوند.

1. *final_maps*

این دایرکتوری شامل نقشه‌های نهایی تولید شده در پروژه است. این نقشه‌ها از طریق پردازش داده‌های جاذبه‌ای و روش‌های تحلیلی مختلف مانند روش *Parker* و داده‌های مدل *EGM96* به دست آمده‌اند. این خروجی‌ها برای نمایش تغییرات گرانشی در ناحیه مورد مطالعه استفاده می‌شوند.

2. *mathematica_codes*

این بخش شامل کدهای *Mathematica* است که برای انجام پردازش‌های عددی در نسخه اولیه پروژه مورد استفاده قرار گرفته‌اند. از آنجایی که در مراحل نهایی پروژه پردازش به زبان *Python* منتقل شده، این کدها به عنوان مرجع یا برای مقایسه نتایج همچنان در ساختار پروژه باقی مانده‌اند.

3. *project_report*

دایرکتوری مربوط به مستندات و گزارشات پروژه است. این بخش شامل اسناد و توضیحات فنی درباره نحوه عملکرد الگوریتم‌ها، نتایج و تحلیل‌های انجام شده است. مستندات موجود در این پوشه به درک بهتر خروجی‌های تولید شده و استدلال‌های پشت روش‌های مورد استفاده کمک می‌کنند.

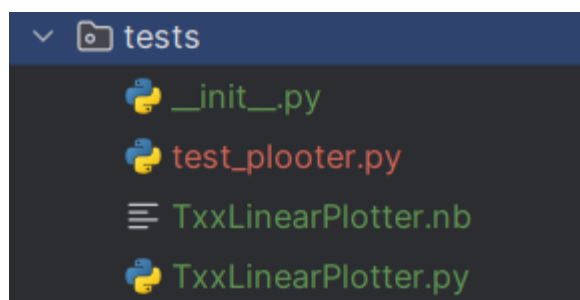
4. *project_result_csv_files*

این بخش شامل فایل‌های خروجی پردازش شده به فرمت *CSV* است. این فایل‌ها داده‌های عددی نهایی مربوط به ماتریس‌های جاذبه‌ای و گرادیان گرانشی را شامل می‌شوند. برخی از این فایل‌ها نتایج مربوط به محاسبات *Parker* و برخی دیگر مربوط به مدل *EGM96* هستند. خروجی‌های این دایرکتوری به عنوان ورودی برای تولید نقشه‌های نهایی در دایرکتوری *final_maps* مورد استفاده قرار گرفته‌اند.

3-6- جزئیات پیاده‌سازی ماژول‌های دایرکتوری *tests*

دایرکتوری *tests* مسئول انجام تست‌های مختلف بر روی

داده‌های خروجی و الگوریتم‌های پردازشی پروژه است. این دایرکتوری شامل اسکریپت‌های تستی است که دقت، صحت و کیفیت داده‌های پردازش شده را بررسی می‌کنند. تست‌ها معمولاً روی داده‌های نهایی، روش‌های ترسیم نمودار و پردازش‌های محاسباتی انجام می‌شوند.



1. *test_plooter.py*

این فایل شامل تست‌هایی برای بررسی دقت پردازش داده‌های گرادیان گرانشی و ترکیب نتایج محاسبات روش‌های مختلف مانند *Parker* و *EGM96* است. مهم‌ترین بخش‌های این فایل عبارتند از:

- بارگذاری داده‌ها از فایل‌های *CSV* خروجی *TxxParker.csv*، *TxxEGM96.csv* و غیره.
- محاسبه مجموع ماتریس‌ها برای تأیید صحت جمع داده‌های پردازش شده از روش‌های مختلف.
- تولید محدوده طول و عرض جغرافیایی بر اساس مقادیر مشخص شده مثلاً 32-50 تا 34,0.5
- بررسی ساختار داده‌ها و مطمئن شدن از همخوانی مقادیر ورودی و خروجی.

نکته: این فایل می‌تواند در تست‌های مقایسه‌ای و کنترل صحت داده‌های پردازش شده قبل از ترسیم

نهایی استفاده شود.



2. *TxxLinearPlotter.py*

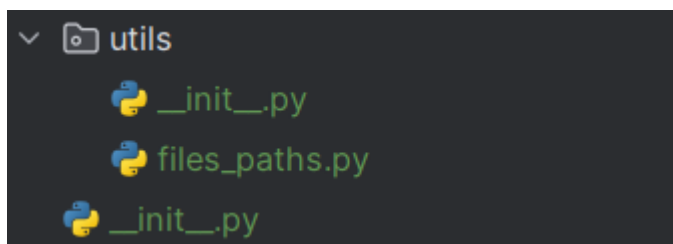
این فایل مربوط به بررسی نمودارهای خطی گرادیان گرانشی *Txx* در طول و عرض جغرافیایی خاص است. این تست شامل:

- بارگذاری داده‌های *TxxTOTAL.csv* از مسیر
`sources/final_phase/all_sources/project_result_csv_files/`
 - ترسیم *Txx* در برابر طول جغرافیایی برای یک عرض جغرافیایی ثابت $\varphi = 33^\circ$.
 - ترسیم *Txx* در برابر عرض جغرافیایی برای یک طول جغرافیایی ثابت $\lambda = 51^\circ$.
- این فایل به تحلیل نحوه تغییرات گرادیان گرانشی در نقاط خاص و بررسی کیفیت داده‌های خروجی کمک می‌کند.

3. *TxxLinearPlotter.nb*

این فایل یک نوت‌بوک (*.nb*) *Mathematica* است که شامل نسخه تعاملی *TxxLinearPlotter.py* می‌باشد. این نوت‌بوک از زبان *Wolfram Language* متعلق به *Mathematica* استفاده می‌کند و به کاربران امکان بررسی بصری و تحلیل نتایج را در یک محیط محاسباتی تعاملی می‌دهد.

3-7- جزئیات پیاده‌سازی ماژول‌های دایرکتوری *utils*



دایرکتوری *utils* مخفف *Utilities* شامل توابع و متغیرهای کمکی است که در بخش‌های مختلف پروژه استفاده می‌شوند. این دایرکتوری معمولاً شامل توابع عمومی مانند مدیریت مسیر فایل‌ها، تبدیل داده‌ها، و سایر ابزارهای مورد نیاز برای اجرای پردازش‌های اصلی پروژه است.

files_paths.py

این ماژول شامل مسیرهای فایل‌های داده‌ای است که در پردازش‌های مختلف پروژه استفاده می‌شوند. ساختار اصلی این فایل به شکل زیر است:

1. مسیر فایل‌های ورودی:

○ مسیر فایل *Elev_Matrix_Output.xlsx* که داده‌های ارتفاع را در خود ذخیره دارد :

○ `Elev_Matrix_Output_file_path = '../database/Elev_Matrix_Output.xlsx'`

2. مسیر پایه (*base_path*) برای فایل‌های خروجی:

○ خروجی‌های پروژه در دایرکتوری :

○ `../sources/final_phase/all_sources/project_result_csv_files/new/python`

ذخیره می‌شوند :

`base_path = '../sources/final_phase/all_sources/project_result_csv_files/new/python'`

3. نام فایل‌های CSV مرتبط با روش‌های مختلف پردازشی:

○ فایل‌های مربوط به روش *Parker*:

○ `TxxParker_file_name = "TxxParker.csv"`

○ `TxyParker_file_name = "TxyParker.csv"`

○ `TxzParker_file_name = "TxzParker.csv"`

○ `TyyParker_file_name = "TyyParker.csv"`

○ `TyzParker_file_name = "TyzParker.csv"`

○ `TzzParker_file_name = "TzzParker.csv"`

○ `allTparker_file_name = "Gravity Gradient Parker.csv"`

فایل‌های مربوط به مدل EGM96:

`TxxEGM96_file_name = "TxxEGM96.csv"` ○
`TxyEGM96_file_name = "TxyEGM96.csv"` ○
`TxzEGM96_file_name = "TxzEGM96.csv"` ○
`TyyEGM96_file_name = "TyyEGM96.csv"` ○
`TyzEGM96_file_name = "TyzEGM96.csv"` ○
`TzzEGM96_file_name = "TzzEGM96.csv"` ○
`allTEGM96_file_name = "Gravity Gradient EGM96.csv"` ○

فایل‌های نهایی که ترکیب شده‌اند :

`TxxTOTAL_file_name = "TxxTOTAL.csv"` ○
`TxyTOTAL_file_name = "TxyTOTAL.csv"` ○
`TxzTOTAL_file_name = "TxzTOTAL.csv"` ○
`TyyTOTAL_file_name = "TyyTOTAL.csv"` ○
`TyzTOTAL_file_name = "TyzTOTAL.csv"` ○
`TzzTOTAL_file_name = "TzzTOTAL.csv"` ○
`allTTOTAL_file_name = "Gravity Gradient TOTAL.csv"` ○

این ساختار مدیریت مسیر فایل‌ها را ساده‌تر می‌کند و به کدهای اصلی اجازه می‌دهد بدون سخت‌کد کردن مسیرها، از داده‌های پردازش شده استفاده کنند.

جمع‌بندی

دایرکتوری `utils` شامل ماژول‌هایی برای مدیریت مسیر فایل‌ها و متغیرهای کمکی است. این دایرکتوری نقش مهمی در ساختار پروژه دارد زیرا به جای هاردکد کردن مسیرها در اسکریپت‌های مختلف، از متغیرهای تعریف‌شده در `files_paths.py` برای خواندن و نوشتن داده‌ها استفاده می‌شود.



4- فایل *requirements.txt* و اهمیت آن در پروژه

فایل *requirements.txt* شامل لیستی از پکیج‌های مورد نیاز برای اجرای پروژه است. این فایل به عنوان یک مستند فنی مهم در هر پروژه پایتونی استفاده می‌شود و به دیگران کمک می‌کند که محیط لازم برای اجرای پروژه را دقیقاً مشابه محیط توسعه ایجاد کنند.

1. توضیح پکیج‌های استفاده‌شده در پروژه

این فایل شامل پکیج‌های اصلی و وابستگی‌های آن‌ها است که در فرآیند محاسبات، پردازش داده، ترسیم نمودار و مدیریت سرور نقش دارند. برخی از مهم‌ترین کتابخانه‌های موجود در این فایل عبارتند از:

پکیج‌های مربوط به توسعه وب و API

- *fastapi==0.115.8* → فریم‌ورک اصلی برای توسعه API
- *fastapi-cli==0.0.7* → ابزاری کمکی برای CLI برای FastAPI
- *httpx==0.28.1* → برای ارسال درخواست‌های HTTP در FastAPI
- *orjson==3.10.15* → یک جایگزین سریع‌تر برای json در FastAPI
- *python-multipart==0.0.20* → برای مدیریت درخواست‌های آپلود فایل در FastAPI
- *watchfiles==1.0.4* → برای ریلود خودکار در هنگام تغییر کد

پکیج‌های مرتبط با پردازش داده و محاسبات عددی

- *numpy==1.x.x* → محاسبات عددی و ماتریسی
- *scipy==1.13.1* → ابزارهای تحلیل داده و محاسبات علمی
- *pandas==2.2.2* → پردازش و تحلیل داده‌های جدولی

- *openpyxl==3.1.5* → برای کار با فایل‌های اکسل (*.xlsx*).



پکیج‌های مرتبط با گرافیک و ترسیم نمودار

- matplotlib==3.9.2 → برای ترسیم نمودارهای علمی
- kivymd==1.2.0 → برای طراحی رابط کاربری گرافیکی در پایتون
- pywebview==5.4 → برای نمایش رابط وب در یک پنجره GUI
- pyside6==6.8.2 → برای توسعه رابط کاربری دسکتاپ بر پایه Qt

پکیج‌های مرتبط با بهینه‌سازی و پروفایلینگ

- numba==0.60.0 → برای بهینه‌سازی محاسبات سنگین و افزایش سرعت پردازش
- line-profiler==4.1.3 → برای تحلیل عملکرد کد و شناسایی بخش‌های کند
- py-spy==0.3.14 → برای پروفایل کردن مصرف پردازنده در کد پایتون

پکیج‌های مرتبط با پردازش داده‌های مکانی

- rasterio==1.4.2 → برای پردازش داده‌های GIS و تصاویر ماهواره‌ای

نحوه استفاده از requirements.txt برای نصب پکیج‌ها

برای بازسازی محیط توسعه در سیستم‌های دیگر، می‌توان از دستور زیر استفاده کرد :

```
pip install -r requirements.txt
```

این دستور تمامی پکیج‌های مورد نیاز را نصب کرده و محیطی مشابه محیط توسعه را برای اجرا فراهم

می‌کند.

5- ارزیابی و مقایسه نتایج محاسباتی *Mathematica* و *Python*

در این بخش، نتایج محاسباتی حاصل از اجرای الگوریتم در محیط *Python* با نتایج متناظر در *Mathematica* مورد ارزیابی و مقایسه قرار می‌گیرد. هدف از این تحلیل، بررسی تفاوت‌های احتمالی در مقادیر خروجی، کارایی محاسباتی و دقت مدل‌های مورد استفاده در هر دو محیط است.

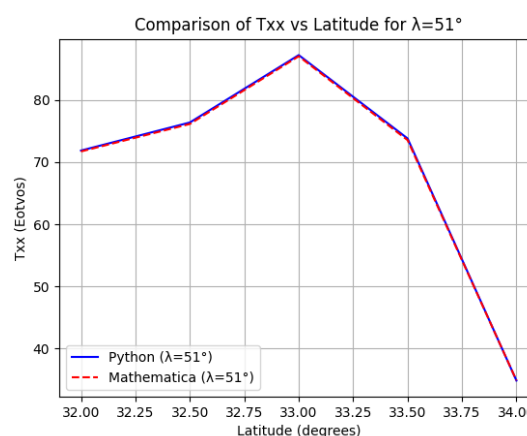
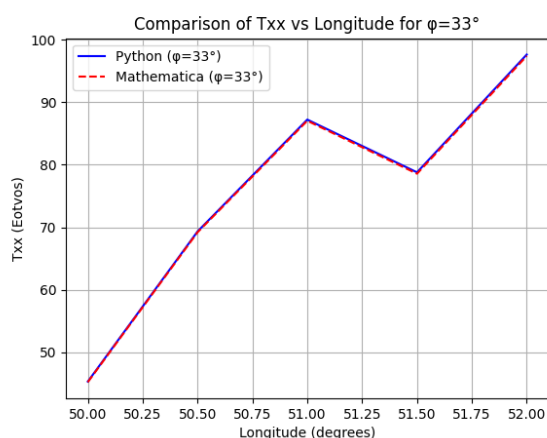
روش مقایسه

برای بررسی تطابق داده‌ها، از دو مجموعه خروجی محاسباتی استفاده شده است:

- خروجی محاسبات *Python* از فایل *TxxTOTAL.csv*
- خروجی محاسبات *Mathematica* از فایل *MTxxTOTAL.csv*

در این مقایسه، دو نوع نمودار بررسی شده است:

1. نمودار تغییرات مقدار T_{xx} بر حسب طول جغرافیایی در عرض جغرافیایی ثابت ($\phi = 33^\circ$)
2. نمودار تغییرات مقدار T_{xx} بر حسب عرض جغرافیایی در طول جغرافیایی ثابت ($\lambda = 51^\circ$)



اطلاعات مربوط به بازه‌های جغرافیایی و رزولوشن:

- بازه طول جغرافیایی: 50-52
- بازه عرض جغرافیایی: 32-34
- رزولوشن طول و عرض: 0.5

نتایج و تحلیل

1. مقایسه Txx بر حسب طول جغرافیایی در عرض ثابت $phi = 33$

در این تحلیل، مقدار Txx از هر دو فایل استخراج شده و در یک نمودار نمایش داده شده است. خطوط نشان‌دهنده مقادیر محاسبه شده در هر دو نرم‌افزار به شرح زیر است:

• *Python* : نمایش داده شده با خط آبی ممتد

• *Mathematica* : نمایش داده شده با خط قرمز چین‌دار

نتایج نشان می‌دهند که تغییرات کلی داده‌ها در هر دو روش مشابه است، اما در برخی نقاط تفاوت‌های جزئی در مقدار Txx مشاهده می‌شود. این تفاوت می‌تواند ناشی از تفاوت در روش‌های عددی مورد استفاده در هر دو پلتفرم باشد.

2. مقایسه Txx بر حسب عرض جغرافیایی در طول ثابت $lambda = 51$

در این حالت نیز مقدار Txx در هر دو فایل استخراج و رسم شده است. نتایج این مقایسه نشان می‌دهند که:

- الگوی تغییرات در هر دو نرم‌افزار مشابه است.
- در برخی نواحی اختلاف مقادیر خروجی بین دو روش وجود دارد که احتمالاً به دلیل دقت عددی محاسبات در هر دو روش است.
- تحلیل اختلافات و بررسی عوامل مؤثر
- دقت عددی: یکی از مهم‌ترین عوامل ایجاد اختلاف، دقت عددی در محاسبات است. روش‌های عددی مورد استفاده در *Python* و *Mathematica* می‌توانند متفاوت باشند و این موضوع بر روی خروجی نهایی تأثیرگذار است.
- روش‌های انتگرال‌گیری و تقریب: در برخی مراحل، استفاده از توابع ویژه و روش‌های انتگرال‌گیری عددی می‌تواند نتایج متفاوتی ایجاد کند.
- ساختار داده‌ها و نحوه پردازش آن‌ها: تفاوت در ساختار داده‌های ورودی، روش‌های درونیابی و فیلترهای به کاررفته نیز می‌تواند از دیگر عوامل ایجاد اختلاف باشد.