

## Oscar Gonzalez' ERT skills assessment. August 5<sup>th</sup> 2022

I am an experienced **Scientific Programmer**, with a strong background and formal education in **Math** and **Physics**. I have used many programming languages, and even created my own *GLS Scripting Language* with *Operator Overloading* to process math on *LAS* petrophysical files. This ERT assessment allows me to show you all, some of my skills on these areas:

- **Algorithm + System** design (The boxes problem)
- Applied **Math** and optimization (The boxes problem), and of course basic **Physics** to understand the IRI statement.
- Programming Languages: (and I handle much more than those)
  1. *Java*
  2. *Groovy*
  3. *Bash Shell*
  4. *Fortran*
  5. *C*
  6. *PHP*
  7. *JavaScript*

Seven programming languages used for ERT
- Tools and Web:
  1. Linux *make*, *egrep*, *vim*, etc.
  2. *HTML* (Handwritten: without edition software)
  3. *CSS* (Handwritten: without edition software)

### Time line:

- Monday July 25: Ye Men **interview**
  - Tuesday 26: **Desktop computer still dead** ☹️
  - Wednesday 27: **Desktop computer still dead** ☹️
  - Thursday 28: **Start** with The boxes assessment
  - Friday 29: The boxes assessment
  - **Week-end, some rest, but less than I needed** 😊
  - Monday: **End** with the boxes assessment
  - Tuesday: **Start** with the IRI assessment
  - Wednesday: **End** with the IRI assessment
  - Thursday: *HTML*, *CSS*, *JavaScript* assessment
  - Friday August 5<sup>th</sup>: **This results seminar** 😊
- I could only fit an **effective allowance of 6 days**, and still I am working with JavaScript.
- Overall I fell OK with my results.

## Assessment 7: Pack random boxes into a large container

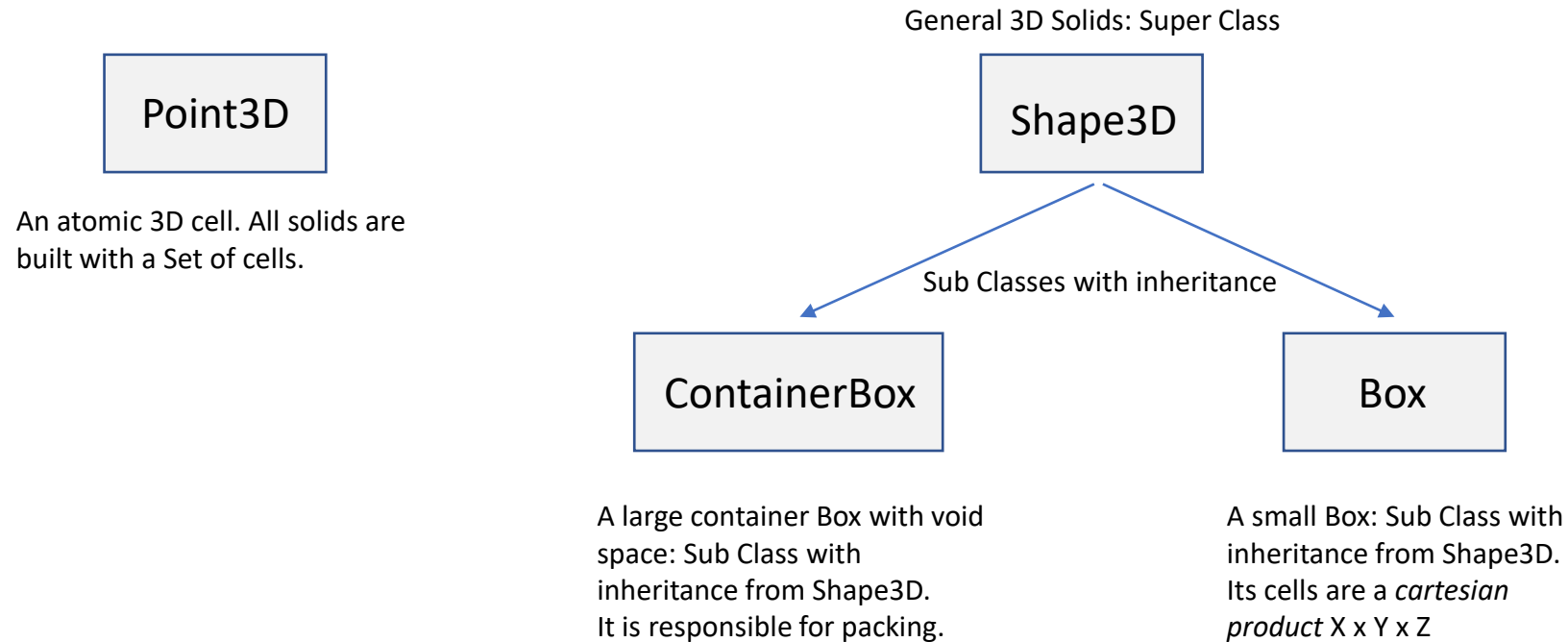
### Optimization (fun problem and open-ended solution)

*"You need a fairly optimal way to bunch of **small boxes** into a **big box**. It doesn't need to be perfect, as coming up with a provably optimal solution is likely a really hard problem. The input is a text file with a listing of **integer** volumes(**width, length, height**). The first line is the container volume, the rest are the volumes of boxes to be placed inside. Place as many as you can. Output the best configuration you find to optimize the number of boxes put into the container, and/or the configuration that most fully utilizes the space within the container. If you write in a low-level language, you can use a library for basic data structures"*

### Oscar's solution: Algorithm and Code Implementation: (it took me three days)

- I have moved so many times towing an U-Haul trailer with my house-holdings that I took the problem almost personal. The problem turns even more interesting if instead of different boxes, you pack **real irregular 3D solids**: You may start packing a *table* flipped down with its legs up, then put a *coach*, then the *chairs*, and below and above each chair seat put *boxes* with math books and so forth. And don't forget room for your *bike*.
- The problem statement mentions a word that immediately suggests an approach: "**Integer width, length, and heights**". It settles the framework to extend from boxes to full irregular 3D shapes. Dealing with integers (instead of real numbers), immediately allows us to model each irregular 3D body in a *3D grid of cells* (or *3D pixels*). So each body is just a **Set** or 3D points (or cells):  $\text{Body} = \{(ix, iy, iz)\}$
- Start randomly, or perhaps choosing a large *Body1* (Count of pixels  $\{(ix, iy, iz)\}$  is a large integer). Perform random body transforms (6 translations East, West, North, South, Up, Down), and  $4^3=64$  3D rotations along axis x, y, z. Collocate this body on the floor (z layer=0)
- Choose a next *Body2* and place it in the container or trailer. If the Set intersection between *Body1* and *Body2* 3D cells is not empty, it means that there is a collision between the two bodies: Reject and remove *Body2* from the container. If the intersection is empty accept *Body2* placement, and then continue to a next *Body<sub>i</sub>*, and so on.

## Software Design Diagram: Use class objects



### Language implementation (fully coded, up and running):

- A detailed API was written in **Java** for the core four classes
- A builder was written in Bash shell under Linux. (Far more detailed and general than a Makefile)
- The access to the API is done through several *JVM* interpreted script languages:
  - **Groovy** (The one I choose)
  - Python (Jython)
  - Ruby (JRuby)
  - Lisp (Clojure)
  - Scala
  - Kotlin

**Next Step:** Let's check the implementation code and **run alive!**

## Optimization Algorithm

The assessment statement asks to **maximize the amount of boxes**. That is an over-simplification. You may fit in your trailer perhaps only two mattresses and one coach, but would you prefer to fit one thousand of cheap pencils or colored crayons instead? We may want to define many objective functions to **minimize**, like:

- O1: The amount of 3D shapes. To minimize, just use instead  $-1.0 * O1$
- O2: The void space volume of the container after packaging should be a minimum
- O3: Safety first!, compute the center of gravity of the filled container, you want its Z coordinate as close as possible to the floor layer  $Z=0$

Combine all desired optimizations in a single global Optimization, for instance:

- $O_{global} = w1 * O1 + w2 * O3 + \dots + wn * On$

### Optimization Strategy: Slow thermal cooling of “temperature”: $O_{global}$ (called *Annealing*)

It usually works nicely and emulates the real physical world: Perform small random arrangements of the objects, including translations and rotations, entering and removing of objects. For each scenario realization compute  $O_{global}$ . Iterate for a maximum allowance of CPU time, then just simply choose, the arrangement 3D solids realization that yields the minimum value for  $O_{global}$ , that is, let it cool down the “temperature”.

I have worked on Annealing when studying GeoStatistics at Stanford. I could not fully implement the code, that honestly would take weeks or even a couple of months, so it is beyond of the scope of the assessment.

## Assessment 1: IRI Electron Density Profile

Assessment IRI EDP: Create a C-based modeling and simulation program that drive IRI model Fortran code. The code should capture and generate vertical EDP (Electron Density Profile) for a given time and location of interest.

time of interest: Mar 3 2021 UT 11:00:00 and Mar 4, 2021 UT 23:00:00

location o interest: Lat 37.8N and Lon 75.4W

Assessment Criteria:

- 1) Create a simple Makefile that can compile iri2016 (<http://irimodel.org>) and generate a shared object/library
- 2) Write a C-program that links with the shared object created and create all data needed for step 3)
- 3) Use gnuplot ([www.gnuplot.info](http://www.gnuplot.info)) or other similar C-based plotting tools to generate plots of EDP parameters using the shared objective created in step 1.
- 4) alternatively, use F2PY (<https://www.numfys.net/howto/F2PY/>) and Python to create EDP plots using the shared object created in step 1. (Although C-based plotting is the preferred solution)
- 5) Furnish instructions/documentation, etc. on how to run the code and lesson/insights learned by doing this exercise.



37.8,75.4

37°48'00.0"N 75°24'00.0"W

37.800000, -75.400000

Directions

Save

North Atlantic Ocean

QJX2-X2R Chincoteague, Virginia, USA

Add a missing place

Add your business

Add a label

Photos

Layers

Restaurants

Hotels

Attractions

Transit

Parking

Pharmacies

ATMs

Edmonton

Saskatoon

Calgary

Regina

Winnipeg

Vancouver

Kelowna

Seattle

Portland

Minneapolis

Chicago

St. Louis

Indianapolis

Nashville

Atlanta

Jacksonville

Orlando

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Diego

Los Angeles

Las Vegas

San Jose

San Francisco

Sacramento

Salt Lake City

Denver

Albuquerque

Montreal

Ottawa

Quebec City

Philadelphia

New York

Columbus

Indianapolis

St. Louis

Chicago

Minneapolis

Winnipeg

Edmonton

Calgary

Saskatoon

Regina

Vancouver

Kelowna

Seattle

Portland

San Francisco

Sacramento

San Jose

Los Angeles

Las Vegas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Sexagesimal: degrees and minutes. Not decimal. Enough for the exercise. That's why it lays on the sea. My mistake.

Oscar is here now

Exercise Location.

Not quite. Why?

Map data ©2022 Google, INEGI

Canada

Terms

Privacy

Send feedback

200 km

37°48'00.0"N 75°24'00.0"W - Google Maps

37°48'00.0"N 75°24'00.0"W

37.8,75.4

37.800000, -75.400000

Directions

Save

North Atlantic Ocean

QJX2-X2R Chincoteague, Virginia, USA

Add a missing place

Add your business

Add a label

Photos

Layers

Restaurants

Hotels

Attractions

Transit

Parking

Pharmacies

ATMs

Edmonton

Saskatoon

Calgary

Regina

Winnipeg

Vancouver

Kelowna

Seattle

Portland

Minneapolis

Chicago

St. Louis

Indianapolis

Nashville

Atlanta

Jacksonville

Orlando

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Diego

Los Angeles

Las Vegas

San Jose

San Francisco

Sacramento

Salt Lake City

Denver

Albuquerque

Montreal

Ottawa

Quebec City

Philadelphia

New York

Columbus

Indianapolis

St. Louis

Chicago

Minneapolis

Winnipeg

Edmonton

Calgary

Saskatoon

Regina

Vancouver

Kelowna

Seattle

Portland

San Francisco

Sacramento

San Jose

Los Angeles

Las Vegas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Sexagesimal: degrees and minutes. Not decimal. Enough for the exercise. That's why it lays on the sea. My mistake.

Oscar is here now

Exercise Location.

Not quite. Why?

Map data ©2022 Google, INEGI

Canada

Terms

Privacy

Send feedback

200 km

37°48'00.0"N 75°24'00.0"W - Google Maps

37°48'00.0"N 75°24'00.0"W

37.8,75.4

37.800000, -75.400000

Directions

Save

North Atlantic Ocean

QJX2-X2R Chincoteague, Virginia, USA

Add a missing place

Add your business

Add a label

Photos

Layers

Restaurants

Hotels

Attractions

Transit

Parking

Pharmacies

ATMs

Edmonton

Saskatoon

Calgary

Regina

Winnipeg

Vancouver

Kelowna

Seattle

Portland

Minneapolis

Chicago

St. Louis

Indianapolis

Nashville

Atlanta

Jacksonville

Orlando

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Diego

Los Angeles

Las Vegas

San Jose

San Francisco

Sacramento

Salt Lake City

Denver

Albuquerque

Montreal

Ottawa

Quebec City

Philadelphia

New York

Columbus

Indianapolis

St. Louis

Chicago

Minneapolis

Winnipeg

Edmonton

Calgary

Saskatoon

Regina

Vancouver

Kelowna

Seattle

Portland

San Francisco

Sacramento

San Jose

Los Angeles

Las Vegas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Phoenix

Tucson

Ciudad Juárez

San Antonio

Houston

Austin

Dallas

Sexagesimal: degrees and minutes. Not decimal. Enough for the exercise. That's why it lays on the sea. My mistake.

Oscar is here now

Exercise Location.

Not quite. Why?

Map data ©2022 Google, INEGI

Canada

Terms

Privacy

Send feedback

200 km



# International Reference Ionosphere

[IRI VIA FTP](#)

[RUN ONLINE](#)

[REFERENCES](#)

[LINKS](#)

[CONTACT \(info@irimodel.org\)](mailto:info@irimodel.org)

## BRIEF DESCRIPTION:

The International Reference Ionosphere (IRI) is an international project sponsored by the Committee on Space Research (COSPAR) and the International Union of Radio Science (URSI). These organizations formed a Working Group ([members list](#)) in the late sixties to produce an empirical standard model of the ionosphere, based on all available data sources ([Charter](#)). Several steadily improved editions of the model have been released. For given location, time and date, IRI provides monthly averages of the electron density, electron temperature, ion temperature, and ion composition in the ionospheric altitude range (see details below).

The major data sources are the worldwide network of ionosondes, the powerful incoherent scatter radars (Jicamarca, Arecibo, Millstone Hill, Malvern, St. Santin), the ISIS and Alouette topside sounders, and in situ instruments flown on many satellites and rockets. IRI is updated yearly during special [IRI Workshops](#) (e.g., during COSPAR general assembly). More information can be found in the [workshop reports](#).

The IRI model and software is updated according to the decisions of the IRI Working Group. The software package includes the FORTRAN subroutines, model coefficients (CCIR, URSI, IGRF), indices files (IG\_RZ.DAT, APF107.DAT) and README and LICENSE files. The IRI build-up and formulas are described in detail in a 158-page NSSDC report by Bilitza (1990) (PDF accessible in REFERENCES).

An IRI listserv keeps the community informed about model updates, workshops, publication, and other IRI-related matters. To subscribe send a message to [info@irimodel.org](mailto:info@irimodel.org) with 'subscribe IRI your\_email\_address' in the SUBJECT line and your name, affiliation and mailing address in the body of the message.

## PARAMETERS:

Electron density, electron temperature, ion temperature, ion composition ( $O^+$ ,  $H^+$ ,  $He^+$ ,  $N^+$ ,  $NO^+$ ,  $O^+_{2+}$ , Cluster ions), equatorial vertical ion drift, vertical ionospheric electron content (vTEC; a user can select the ending height for the integration along the electron density profile), F1 probability, spread-F probability, auroral boundaries, effects of ionospheric storms on F and E peak densities

## INPUTS:

Required: solar indices (F10.7 daily, 81-day, and 12-month running mean; sunspot number 12-month running mean), ionospheric index (ionosonde-based IG index 12-month running mean), magnetic index (3-h ap, daily ap). The indices are found internally by IRI for the user-specified date and time. In the case of F10.7D, F10.7\_81, F10.7\_12, R\_12, and IG\_12 a user can input his/her own values.

Optional: The user can provide a number of input parameters and the IRI profiles will then be adjusted to these input parameters:

F2-peak height (hmF2) or propagation factor M3000F2, F2-peak plasmafrequency (foF2) or electron density (NmF2)

F1-ledge height (hmF1), plasmafrequency (foF1) or electron density (NmF1)

E-peak height (hmE), plasmafrequency (foE) or electron density (NmE)

D-ledge height (hmD), plasmafrequency (foD) or electron density (NmD)

## HEIGHT RANGE:

*Electron density:* daytime: 65-2000km, nighttime: 80-2000km

*Electron and ion temperature:* 60-2500km (IRI-95 option: 60-3000km)

*Ion composition:* 75-2000km (DS95/DY85 option: 80-2000km)

## AVAILABILITY:

\* Fortran source code: [IRI-2016](#) (10/13/2021), [IRI-2012](#), [IRI-2007](#), [IRI-2001](#), [COMMON FILES for all versions](#)

NOTE: Besides the files that come with each version you also need to download the COMMON FILES and INDICES FILES.



## International Reference Ionosphere - IRI (2016) with IGRF-13 coefficients

This page enables the computation and plotting of IRI parameters: electron and ion ( $O^+$ ,  $H^+$ ,  $He^+$ ,  $O_2^+$ ,  $NO^+$ ) densities, total electron content, electron, ion and neutral (CIRA-86) temperatures, equatorial vertical ion drift and others.

[Go to the IRI description](#)

Help

### Select Date and Time

Year(1958-2020):

Month:  Day(1-31):

**Note:**If date is outside the Ap index range (1958/02/14-2022/6/6),then STORM model will be turned off.

Time  Time (0. - 24.0 in decimal hours):

### Select Coordinates

Coord. Type  Latitude(-90. - 90. deg.):  Longitude(0. - 360. deg.)

Height (km, from 60. to 2000.):

### Select profile type and range:

Height [60. - 2000. km]  Start  Stop  Stepsize

Submit

Reset

# Compile the IRI 2016 Program

- Download all the *Fortran* source code files. Download also all the necessary non-embedded data files.
- Install a Linux Fortran compiler. GNU *gcc* and *gfortran* makes the work, the create a file soft link “f77” to *gfortran* in root */bin/*
- How to compile the code?

1. Write a *.sh Linux Bash script* with full control on dates, folders, and more (**My recommendation**)

```
1 # Normally prefer a full fresh compile:
2
3 rm *.o
4 rm iri
5
6 # Compile to .o object files using -c
7
8 f77 -c iriflip.for
9 f77 -c cira.for
10 f77 -c igrf.for
11 f77 -c iridreg.for
12 f77 -c iritec.for
13 f77 -c irifun.for
14 f77 -c irisub.for
15 f77 -c iritest.for
16
17 # The final Fortran iri executable:
18
19 f77 *.o -o iri
```

2. Straightforward command line: `gfortran -o iri iritest.for irisub.for irifun.for iritec.for iridreg.for igrf.for cira.for iriflip.for`

3. Write a *Makefile* to be run through Linux *make*:

**My opinion:** *make* is a legacy, very old program with primitive syntaxis, like using “TAB” character at start of rules.

```
1 allObjectFiles: iriflip.o cira.o igrf.o iridreg.o iritec.o irifun.o irisub.o iritest.o iri
2
3 iriflip.o: iriflip.for
4     f77 -c iriflip.for
5
6 cira.o: cira.for
7     f77 -c cira.for
8
9 igrf.o: igrf.for
10    f77 -c igrf.for
11
12 iridreg.o: iridreg.for
13    f77 -c iridreg.for
14
15 iritec.o: iritec.for
16    f77 -c iritec.for
17
18 irifun.o: irifun.for
19    f77 -c irifun.for
20
21 irisub.o: irisub.for
22    f77 -c irisub.for
23
24 iritest.o: iritest.for
25    f77 -c iritest.for
26
27 iri:
28    f77 iriflip.o cira.o igrf.o iridreg.o iritec.o irifun.o irisub.o iritest.o -o iri
```

4. Check alive the builder-compiler

5. Check alive the ASCII compiled executable

```
662 $ cat iri_keyboardInput.txt
0,37.8,-75.4
2021,0303,1,11
100
1
100,2000,50
0
0
0
0
```

Create a keyboard input text-file, instead of typing any time you need to run.

6. Run it. The source code has unstable **numerical underflow** IEE floating point number exceptions!, but managed to produce results. Poor math coding?

```
650 $ iri < iri_keyboardInput.txt
jmag(=0/1,geog/geom),lati/deg,long/deg
year(yyyy),mmdd(or -ddd),iut(=0/1,LT/UT),hour
height/km
variable? (1/2/.../8 for height/lat/long/year/month/day/day of year/hour)
begin, end, and stepsize for the selected variable
output-option (if variable=height then choose 0, 3,4, or 5)
(enter 0 for standard table of IRI parameters)
(enter 1 for list of peak heights and densities)
(enter 2 for plasma frequencies, B0, M3000, valley, width and depth,)
(enter 3 for 6 parameters of your choice)
(enter 4 for D-region models at 60,65,...,110 km)
(enter 5 special test output)
upper height [km] for TEC integration (0 for no TEC)
Options: t(rue) or f(false)
Enter 0 to use standard or 1 to enter your own
```

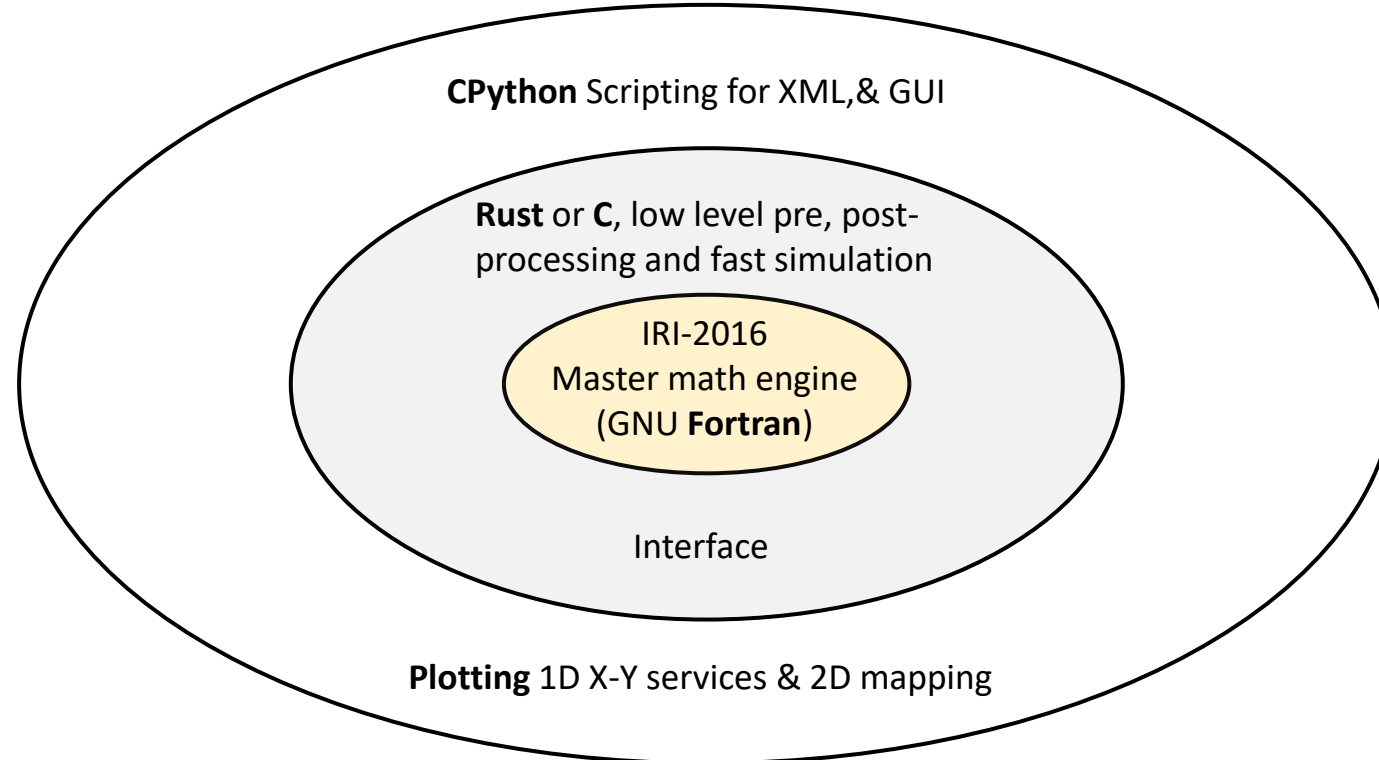
```
*** IRI parameters are being calculated ***
Ne: NeQuick for Topside
Ne, foF2: URSI model is used.
Ne: B0,B1-ABT-2009
Ne, foF1: probability function used.
Ne, D: IRI1990
Ne, foF2: storm model included
Ion Com.: RBV-10 & TBT-15
Te: TBT-2012 model
Auroral boundary model off
Ne, foE: storm model off
Enter 0 to exit or 1 to generate another profile?
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG IEEE_DENORMAL
```

Altitude of ionosphere electron density is the classic plot

H	ELECTRON	DENSITY	TEMPERATURES				ION PERCENTAGES[%]*10						IF16m-2	
km	Ne/cm <sup>3</sup>	Ne/Nm <sup>2</sup>	Tn/K	Ti <sub>1</sub> /K	Ti <sub>2</sub> /K	O <sup>+</sup>	N <sup>+</sup>	H <sup>+</sup>	He <sup>+</sup>	O <sup>2+</sup>	N <sup>0+</sup>	Clust	TEC	LAY
100,0	6360	0,052	181	181	181	0	0	0	0	80	920	-1	-1,0	-1
150,0	10331	0,085	600	600	775	8	0	0	0	228	763	-1	-1,0	-1
200,0	83393	0,685	722	722	1354	286	2	0	0	291	410	-1	-1,0	-1
250,0	121756	0,999	752	821	1573	905	4	0	0	39	48	-1	-1,0	-1
300,0	102526	0,842	760	920	1784	970	23	3	4	0	0	-1	-1,0	-1
350,0	73028	0,599	763	1019	1991	963	24	7	6	0	0	-1	-1,0	-1
400,0	49816	0,409	763	1119	2146	946	24	19	11	0	0	-1	-1,0	-1
450,0	34312	0,282	763	1223	2297	911	24	46	19	0	0	-1	-1,0	-1
500,0	24306	0,200	763	1336	2447	898	23	109	30	0	0	-1	-1,0	-1
550,0	17781	0,146	763	1448	2595	715	22	218	45	0	0	-1	-1,0	-1
600,0	13417	0,110	763	1561	2728	581	23	337	59	0	0	-1	-1,0	-1
650,0	10412	0,085	763	1674	2858	449	25	454	72	0	0	-1	-1,0	-1
700,0	8283	0,068	763	1786	2988	327	26	565	82	0	0	-1	-1,0	-1
750,0	6733	0,055	763	1899	3117	240	26	646	89	0	0	-1	-1,0	-1
800,0	5577	0,046	763	2011	3243	190	24	691	95	0	0	-1	-1,0	-1
850,0	4696	0,039	763	2124	3340	154	23	723	101	0	0	-1	-1,0	-1
900,0	4011	0,033	763	2237	3369	124	21	750	105	0	0	-1	-1,0	-1
950,0	3470	0,028	763	2349	3368	98	20	772	110	0	0	-1	-1,0	-1
1000,0	3035	0,025	763	2462	3364	77	18	792	113	0	0	-1	-1,0	-1
1050,0	2681	0,022	763	2575	3359	60	17	809	115	0	0	-1	-1,0	-1
1100,0	2389	0,020	763	2687	3355	46	15	822	117	0	0	-1	-1,0	-1
1150,0	2146	0,018	763	2800	3350	35	14	833	118	0	0	-1	-1,0	-1
1200,0	1940	0,016	763	2913	3346	27	12	841	120	0	0	-1	-1,0	-1
1250,0	1766	0,014	763	3025	3341	21	11	848	121	0	0	-1	-1,0	-1
1300,0	1616	0,013	763	3138	3337	16	10	853	121	0	0	-1	-1,0	-1
1350,0	1487	0,012	763	3251	3333	12	9	857	122	0	0	-1	-1,0	-1
1400,0	1374	0,011	763	3335	3336	9	8	860	123	0	0	-1	-1,0	-1
1450,0	1275	0,010	763	3357	3357	7	7	863	123	0	0	-1	-1,0	-1
1500,0	1188	0,010	763	3380	3380	5	7	865	123	0	0	-1	-1,0	-1
1550,0	1111	0,009	763	3403	3403	4	6	867	124	0	0	-1	-1,0	-1
1600,0	1042	0,009	763	3426	3426	3	5	868	124	0	0	-1	-1,0	-1
1650,0	980	0,008	763	3449	3449	2	5	869	124	0	0	-1	-1,0	-1
1700,0	925	0,008	763	3472	3472	2	4	870	124	0	0	-1	-1,0	-1
1750,0	874	0,007	763	3495	3495	1	4	871	124	0	0	-1	-1,0	-1
1800,0	829	0,007	763	3519	3519	1	3	871	124	0	0	-1	-1,0	-1
1850,0	788	0,006	763	3542	3542	1	3	872	124	0	0	-1	-1,0	-1
1900,0	750	0,006	763	3565	3565	1	3	872	124	0	0	-1	-1,0	-1
1950,0	715	0,006	763	3588	3588	0	2	873	124	0	0	-1	-1,0	-1
2000,0	683	0,006	763	3611	3611	0	2	873	124	0	0	-1	-1,0	-1

## Write a C program that links for *Fortran* \*.o libraries and drive pre and post processing

**Motivation:** Certainly we don't plan to reinvent the wheel. If there is a reliable, reputable, top quality, and fast program already available, don't touch it unless really necessary. Just write or adapt interfaces in other languages. For a Linux GNU Fortran environment, the natural *glue language* choices are **C** and **Rust**. For quick prototyping then interface the C layer with another layer high level Scripting language, like **CPython** that may use GNU Fortran and C libraries:





- **Step 1:** Explore the eight Fortran source code files and understand the architecture. Start by searching with *Linux egrep* the collection of all subroutines and functions:

```

1 File 1: ./irisub.for
2   Line 224: SUBROUTINE IRI_SUB(JF,JMAG,ALATI,ALONG,IYYYY,MDD,DHOUR,
3   Line 2343: subroutine iri_web(jmag,jf,alati,along,iyyy,mdd,iut,dhour,
4
5 File 3: ./iritec.for
6   Line 36: subroutine IRIT13(ALATI,ALONG,jmag,jf,iy,md,hour,hbeg,hend,
7   Line 95: subroutine iri_tec (hstart,hend,istep,tectot,tectop,tecbot)
8
9 File 4: ./iridreg.for
10  Line 28: SUBROUTINE F00(HGT,GLAT1,IDAY,ZANG,F107T,EDENS,IERROR)
11
12 File 5: ./cira.for
13  Line 24: SUBROUTINE GTD7(IYD,SEC,ALT,GLAT,GLONG,STL,F107A,F107,AP,MASS,D,T)
14  Line 323: SUBROUTINE GTD7D(IYD,SEC,ALT,GLAT,GLONG,STL,F107A,F107,AP,MASS,
15  Line 401: SUBROUTINE GHP7(IYD,SEC,ALT,GLAT,GLONG,STL,F107A,F107,AP,
16  Line 497: SUBROUTINE GLATF(LAT,GM,REFF)
17  Line 561: SUBROUTINE GTS7(IYD,SEC,ALT,GLAT,GLONG,STL,F107A,F107,AP,MASS,D,T)
18  Line 1016: SUBROUTINE METERS(METER)
19  Line 1266: SUBROUTINE TSELEC(SV)
20  Line 1575: SUBROUTINE SPLINEM(X,Y,N,YP1,YPN,Y2)
21  Line 1617: SUBROUTINE SPLINTM(XA,YA,Y2A,N,X,Y)
22  Line 1653: SUBROUTINE SPLINI(XA,YA,Y2A,N,X,YI)
23
24 File 6: ./irirtam.for
25  Line 16: SUBROUTINE READIRTAMCOF(ISEL,IDATE,IHMM,MFF,FF)
26
27 File 7: ./irifun.for
28  Line 259: subroutine tops_cor2(xh,vmod,a01)
29  Line 525: SUBROUTINE ELTEIK(PF107Y,INVDIP,MLT,ALT,DDD,PF107,TE,SIGTE)
30  Line 905: SUBROUTINE KODERR(MIRREQ,DOUT)
31  Line 1210: SUBROUTINE KOEFD(MIRREQ,DOUT)
32  Line 1515: SUBROUTINE KOF107(MIRREQ,DOUT)

```

... and much more ...

A Linux Bash Shell script that uses egrep

Both subroutines **IRI\_SUB** and **iri\_web** have suitable entry points to interface with C code driver. Notice how different typing and case styles might reflect the hands of different programmers perhaps along years.

```

2342 c
2343 subroutine iri_web(jmag,jf,alati,along,iyyy,mdd,iut,dhour,
2344 & height,h_tec_max,ivar,vbeg,vend,vstp,a,b)
2345 c-----
2346 c changes:
2347 c 11/16/99 jf(30) instead of jf(17)
2348 c 10/31/08 outf, a, b (100 -> 500)
2349 c-----
2350 c
2351 c input:  jmag,alati,along,iyyy,mdd,dhour see IRI_SUB
2352 c         height height in km
2353 c         h_tec_max =0 no TEC otherwise upper boundary for integral
2354 c         iut      =1 for UT      =0 for LT
2355 c         ivar     =1 altitude
2356 c               =2,3 latitude,longitude
2357 c               =4,5,6 year,month,day
2358 c               =7 day of year
2359 c               =8 hour (UT or LT)
2360 c         vbeg,vend,vstp variable range (begin,end,step)
2361 c output: a similar to outf in IRI_SUB
2362 c         b similar to oarr in IRI_SUB
2363 c
2364 c         numstp number of steps; maximal 1000
2365 c-----
2366 dimension outf(20,1000),oar(100),oarr(100),a(20,1000)
2367 dimension xvar(8),b(100,1000)
2368 logical jf(50)
2369
2370 nummax=1000
2371 numstp=int((vend-vbeg)/vstp)+1
2372 if(numstp.gt.nummax) numstp=nummax
2373
2374 do 6249 i=1,100
2375 6249 oar(i)=b(i,1)
2376
2376

```

- **Step 2:** Write C interface program. Make sure the Fortran subroutines has an appended underscore “\_”. Carefully declare equivalent variable types, and document the source code.

**Standard \*.h header C files by GNU**

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 /* jmag jf alati along iyyyy madd iut dhour height h_tec_max ivar vbeg vend vstp a b */
6 void iri_web_(int *, int[50], double *, double *, int *, int *, int *, int *, double *, int *, int *, double *, double *, double *, double[20][1000], double[90]);
7
8 int main()
9 {
10     int jmag; /* Switch for 0:geographic coordinates, 1:geo-magnetic coordinates */
11     int jf[50]; /* A collection of up to 50 0-1 logical switches */
12     double alati; /* Latitude */
13     double along; /* Longitude */
14     int iyyyy; /* Year */
15     int madd; /* Month and date */
16     int iut; /* Switch for time: 0=Universal Time, 1= Local Time */
17     int dhour; /* Hour */
18     double height; /* Height */
19     int h_tec_max; /* Switch for TEC integration, use 0 for none, otherwise a real value */
20     int ivar; /* Indicator variable for input: 1=altitude, 2=latitude, 3=longitude, etc. */
21     double vbeg; /* Beginning of variable */
22     double vend; /* End of variable */
23     double vstp; /* Step of variable */
24     double a[20][1000]; /* Output 2D tables, this output does not need to be initialized */
25     double b[90]; /* Additional output parameters, this output does not need to be initialized */
26
27     jmag = 0;
28
29     for (int i=0; i<50; ++i) {jf[i] = 0;} /* Default switches as 1=true */
30
31     jf[4-1] = 2; /* B0,B1 - Bil-2000 */
32     jf[5-1] = 2; /* foF2 - CCIR */
33     jf[6-1] = 2; /* Ni - DS-1995 & DY-1985 */
34     jf[21-1] = 2; /* ion drift computed */
35     jf[23-1] = 2; /* Te_tops (Bil-1985) */
36     jf[28-1] = 2; /* spread-F probability */
37     jf[29-1] = 2; /* IRI01-topside */
38     jf[30-1] = 2; /* IRI01-topside correction */
39     jf[31-1] = 3; /* C indexes starts from 0, but Fortran from 1 */
40     jf[33-1] = 2; /* Auroral boundary model */
41     jf[35-1] = 2; /* E-peak auroral storm model */
42     jf[39-1] = 2; /* hmF2 (M3000F2) */
43     jf[40-1] = 2; /* hmF2 AMTB-model */
44     jf[47-1] = 2; /* CGM computation on */
45
46     alati = 37.8;
47     along = -75.4;
48     iyyyy = 2021;
49     madd = 0303; /* From Fortran source mm=303/3=3 OK, and dd=303-3*100=3 OK */
50     iut = 1;
51     dhour = 11.0;
52     height = 100;
53     h_tec_max = 0;
54     ivar = 1;
55     vbeg = 100;
56     vend = 2000;
57     vstp = 50;
58
59     iri_web_(&jmag, jf, &alati, &along, &iyyyy, &madd, &iut, &dhour, &height, &h_tec_max, &ivar, &vbeg, &vend, &vstp, a, b);
60
61     /* 17 lines: if (1==1) Debug Switch ----- */
62     int index;
63     for (index = 0; index < 40; ++index)
64     {
65         printf ("%F %F %F %F %F %F\n", a[0][index], a[1][index], a[2][index], a[3][index], a[4][index], a[5][index]);
66     }
67 }

```

**Identify and document Fortran variable names, one to one**

**External C function prototypes equivalent to Fortran**

**Formally declare variable types, make sure that 1D and 2D arrays have the same dimensions as Fortran source code, even if you don't like static arrays**

**Nice: a[][] is the Fortran output subroutine. We have full control for post-processing without reading Fortran text output files.**

**Override non-default switches. Even better from an outside text file, or an XML file digested by CPython**

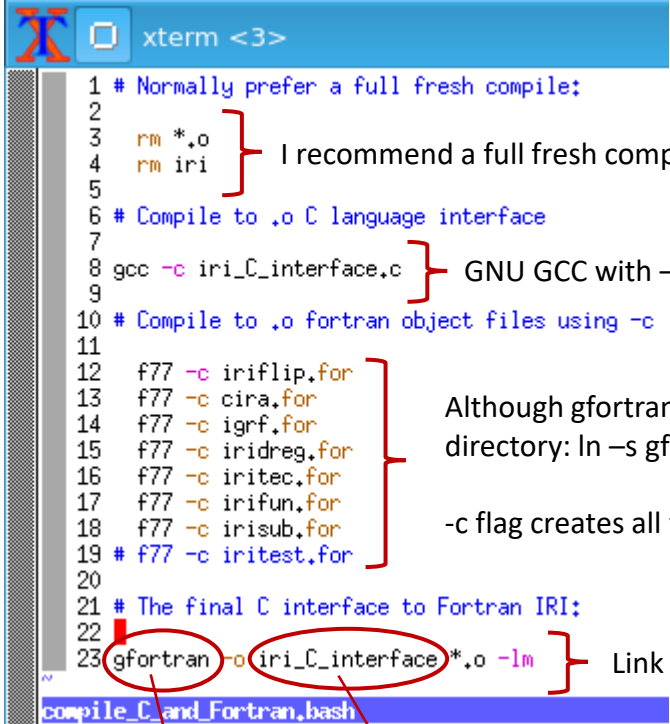
**The exercise geographic location and parameters.**

**Call to the Fortran engine subroutine**

**The 2D a[][] output array is captured. You are free to make use of it, pos-processing and crunch numbers.**

iri\_C\_interface.c 1.1 All

- **Step 4: Joint Fortran-C compilation.** Write a *Linux Bash* script instead of a *MakeFile* for full control



```
1 # Normally prefer a full fresh compile:
2
3 rm *.o
4 rm iri } I recommend a full fresh compilation
5
6 # Compile to .o C language interface
7
8 gcc -c iri_C_interface.c } GNU GCC with -c flag creates .o object file
9
10 # Compile to .o fortran object files using -c
11
12 f77 -c iriflip.for
13 f77 -c cira.for
14 f77 -c igrf.for
15 f77 -c iridreg.for
16 f77 -c iritec.for
17 f77 -c irifun.for
18 f77 -c irisub.for
19 # f77 -c iritest.for
20
21 # The final C interface to Fortran IRI:
22
23 gfortran -o iri_C_interface *.o -lm } Link all *.o object files and -lm default libraries
```

compile\_C\_and\_Fortran.bash

Final "-o" output is the executable file in Linux

Make sure to use gfortran, not gcc here. The reason is that gfortran brings specific fortran defaults and libraries that GCC does not have in C neither C++

```

717 $ ls *.bash
cleanCompiled,bash  compile_C_and_Fortran,bash  compile_IRI,bash  manual_compile_IRI,bash

geoloil@localhost ~/Projects/ERT_Employment/SandBox_IRI_2016_Program
718 $ bash compile_C_and_Fortran,bash
rm: cannot remove 'iri': No such file or directory
iriflip,for:769:72:

    769 |   898   RTS(ITJS)=0,0
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 898 at (1)
iriflip,for:1546:72:

    1546 |   9     SIGEX(I)=0,0
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 9 at (1)
iriflip,for:1636:72:

    1636 |       DO 10 IK=1,12
        |
Warning: Fortran 2018 deleted feature: Shared DO termination label 10 at (1)
iriflip,for:1643:72:

    1643 |   687   OTHPR1(I)=1,0E-15
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 687 at (1)
iriflip,for:1649:72:

    1649 |   786   COLUM(I)=1,0E+25
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 786 at (1)
iriflip,for:1987:72:

    1987 |   20   IF(SUM,NE,0,0) PROB(1,I,L)=Y0(LL,I)/SUM
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 20 at (1)
iriflip,for:2002:72:

    2002 |   50   WRITE(17,90) ZLAM(L),((PROB(IS,J,L),J=1,6),IS=1,3)
        |
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with label 50 at (1)
iriflip,for:2059:72:

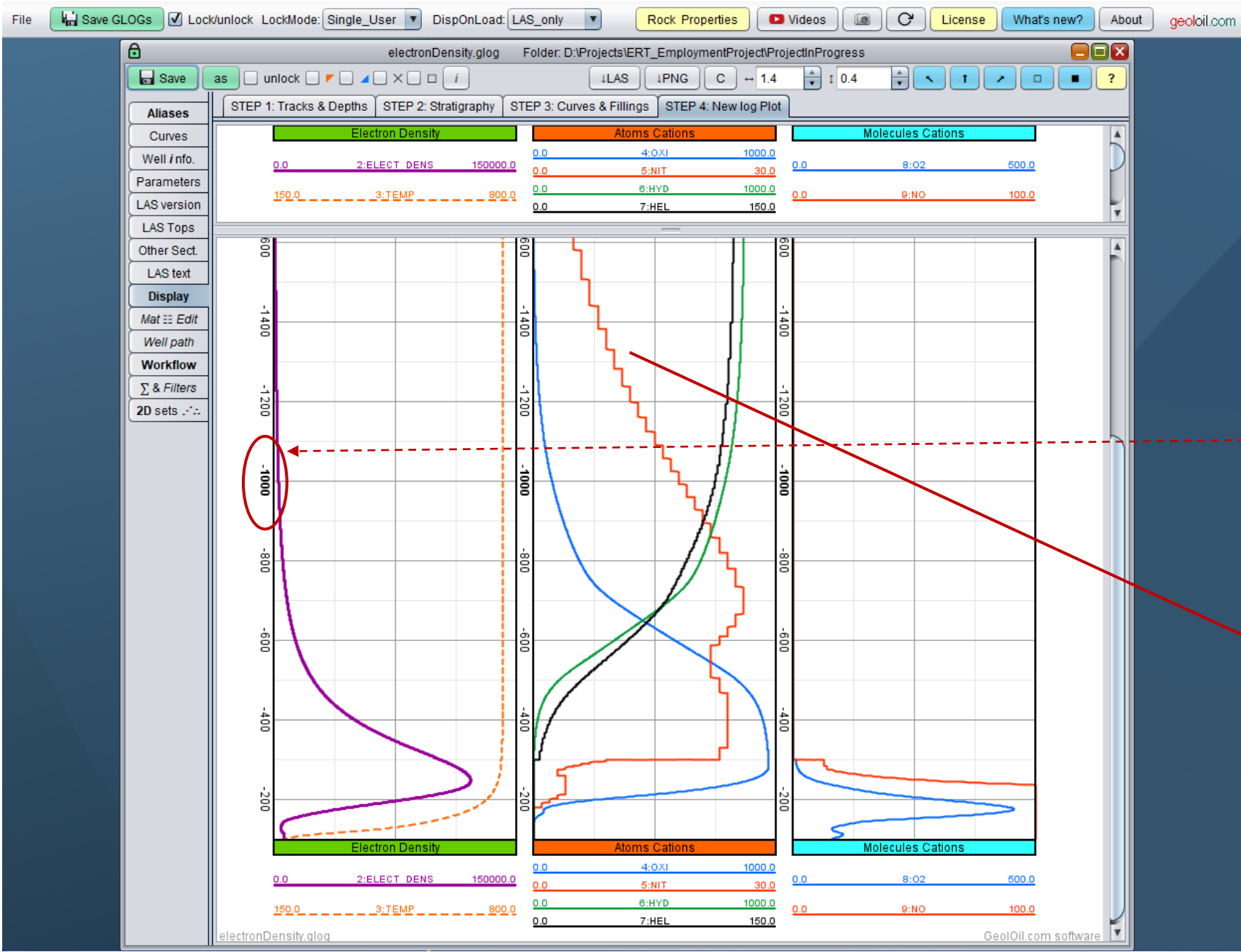
```

Many **Warning** compilation messages for the **Fortran** source codes. The C program interface had no warnings at all. Ignore most warnings for legacy old Fortran sources, but make sure that your C code is seamless.



# Height Plots of Properties

GeolOil Petrophysics, (c) 2012-2022



I used my commercial *petrophysics* software for the exercise. It is a huge Java package of 150,000+ lines of source code and 200+ classes.

It is used in the Oil & Gas industry for depths, not heights or elevations. That's why I had to enforce a **negative sign** on the Z vertical axis.

Staircase type numerical artifacts present in the Fortran library. The C interface post-processing program could easily smooth this behaviour using *Kernel* functions.

## Assessment 5: HTML5/CSS/JavaScript

Assessment Javascript: Basic HTML5/CSS/Javascript development

- 1). Build a page with a text input, a button, and a div.
- 2). Validate the text input on button click to do two things:
  - a). Cannot be null or empty
  - b). Must be numeric
- 3). Make a start array to hold 10 numbers that start with the current textbox value
  - a) For each value, multiply your current iterator value by a random number and add that to the start value.
  - b) Add the new value to the array
- 4). Pass the array to a function that will take each value, double it, and add it back to a new array to return to the original button click code
- 5). Create a table through JavaScript to hold the results. The first column will be the original values, the second column will be the returned values.

I wrote basic illustrative code on *HTML5*, *CCS*, *JavaScript*, and even *PHP*. All of then **ASCII Hand-Written**, not using Web edition software.

- **HTML** controls the basic *structural block of the web page*. I put head, example images, input and output boxes, and internal and external links.
- **CSS** control the font types, colors, but more interesting and valuable, I programmed a “**Responsive Design**”. This is not only nice, but necessary to ensure that the page is **readable on any device**, from a large *Desktop Screen* to a tiny *Mobile Phone Screen*.
- **JavaScript** programmatically validates HTML inputs from the **client side**, without connecting to the remote server.
- **PHP** receives *HTTP POST* data from HTML, it is processed (before JavaScript previously validated boxes inputs) on the server side and send results . I used a minimal of PHP features to include common files. Something HTML can't do.

Geoloil

PETROPHYSICS

4.6 ★★★★★ on Google

PRODUCTS

All Products

1. Log learning set

2. LAS file Editor

3. Alias Editor

4. LAS log Displayer

5. Functions Workflow

6. Multi-well Work-flow

7. Summaries & Upse

8. Ionic Water Analysis

9. Log Scripting

Scripting Manual

CONSULTING

All Services

1. Assembling

2. Processing

3. Interpretation

TRAINING

Custom Training

VIDEOS

All Videos

1. How to plot a LAS file

2. The log-plot display

3. Flatten MD to TVDSS

4. Function Workflows

5. Petrophysical Aliases

6. Multiwell Workflows

7. Summaries & Upse.

8. Water Salinity

9. Mineral Solvers

10. Multizone Parameters

RECIPES

All Recipes

1. Merging LAS files

2. Demo files data set

3. Core Data Import

4. Computing Net-Pay

5. Petrophysical Cutoffs

6. VSH from GR index

7. VSH from Neut-Dens

8. Water Saturations

9. Indonesia Water Sat.

10. Mod. Simandoux Eq

11. Fracture Porosity

12. Geomechanic Logs

13. Split Rock Volume

14. Mineral Solvers

15. Pay Indicator Index

16. Water Salinity & Rw

17. Estimating RHOM

18. Estimating Rw & m

PAPERS

1. Water Sat. from Rxo

2. Porosity from Rxo

3. Vshale or Vclay ?

4. Gas cap detection

5. Biased Beta Distr

6. Shaly Pickett Plot Eq.

REFERENCE

TechLog vs GeoOil

LAS Mnemonics list

Software releases

Questions & Ans.

Testimonials

ERT Assement JavaScript: Basic HTML5/CSS/JavaScript development

- 1). Build a page with a text input, a button, and a div.
- 2). Validate the text input on button click to do two things:

a). Cannot be null or empty

b). Must be numeric
- 3). Make a start array to hold 10 numbers that start with the current textbox value

a) For each value, multiply your current iterator value by a random number and add that to the start value.

b) Add the new value to the array
- 4). Pass the array to a function that will take each value, double it, and add it back to a new array to return to the original button click code
- 5). Create a table through JavaScript to hold the results. The first column will be the original values, the second column will be the returned values.

This is the exercise: Enter a real number from 0 to 100

This is a test

x=

Process

orig 1	<input type="text"/>	ret 1	<input type="text"/>
orig 2	<input type="text"/>	ret 2	<input type="text"/>
orig 3	<input type="text"/>	ret 3	<input type="text"/>
orig 4	<input type="text"/>	ret 4	<input type="text"/>
orig 5	<input type="text"/>	ret 5	<input type="text"/>
orig 6	<input type="text"/>	ret 6	<input type="text"/>
orig 7	<input type="text"/>	ret 7	<input type="text"/>
orig 8	<input type="text"/>	ret 8	<input type="text"/>
orig 9	<input type="text"/>	ret 9	<input type="text"/>
orig 10	<input type="text"/>	ret 10	<input type="text"/>

4.6 ★★★★★ on Google

Downloads

Products

Consulting

Windows

MacOS

Linux

Latest stable version March 2022

GeoOil runs on Windows, MacOS, and Linux. Compare it to Schlumberger's TechLog

Training

HandBook

Videos

TESTIMONIALS

All reviews—

Google reviews—

2022 April: "I'm a huge fan of GeoOil. It is user friendly, the help buttons are awesome, the videos are an excellent resource. Some of the stuff sort of comes naturally to me as I've used several types of software throughout my career in the O&G. Overall, this software is GREAT!"

Hailey Smith, Texas A&M University, USA. ■

2021 August: "I am sincerely grateful that GeoOil is at my fingertips."

Matthew Gerard, Expert Petrophysicist Consultant with 35 years of experience, Texas, USA. ■

2021 March: "I must say, the GeoOil software has been an absolute god send to me as it's enabled me to pick up extra work, work from home, and be self sufficient, so I greatly appreciate the support and the program."

Joel Corcoran, Senior Geoscientist, Consultant, England, UK. ■

2020 May: "I do enjoy the detailed workflow for determining various attributes".

Jason Currie, PG., MS, CEO and President of Point Bar Energy LLC, Oklahoma City, OK, USA. ■

2019 September: "I have been using GeoOil for over 3 years. It is a great tool for doing petrophysical analysis with the ability to do several complex work-flows in a sequential format that is easy to setup and runs very quickly. The log displays are very good and it is easy to edit, shift, manipulate, and manage log curves..." Read more.

Logo images are gracefully and automatically resized for Desktop, Tablet, or Phone screens

ERT Assement JavaScript: Basic HTML5/CSS/JavaScript development

- 1). Build a page with a text input, a button, and a div.
- 2). Validate the text input on button click to do two things:

a). Cannot be null or empty

b). Must be numeric
- 3). Make a start array to hold 10 numbers that start with the current textbox value

a) For each value, multiply your current iterator value by a random number and add that to the start value.

b) Add the new value to the array
- 4). Pass the array to a function that will take each value, double it, and add it back to a new array to return to the original button click code
- 5). Create a table through JavaScript to hold the results. The first column will be the original values, the second column will be the returned values.

This is the exercise: Enter a real number from 0 to 100

x=

orig 1	<input type="text"/>	ret 1	<input type="text"/>
orig 2	<input type="text"/>	ret 2	<input type="text"/>
orig 3	<input type="text"/>	ret 3	<input type="text"/>
orig 4	<input type="text"/>	ret 4	<input type="text"/>
orig 5	<input type="text"/>	ret 5	<input type="text"/>
orig 6	<input type="text"/>	ret 6	<input type="text"/>
orig 7	<input type="text"/>	ret 7	<input type="text"/>
orig 8	<input type="text"/>	ret 8	<input type="text"/>
orig 9	<input type="text"/>	ret 9	<input type="text"/>
orig 10	<input type="text"/>	ret 10	<input type="text"/>

Left: Desktop Screen/Cell Phone Screen: Right

Menu navigation bar on left Desktop does not fit in the phone screen, so CCS relocates it at the bottom of the mobile, instead of disappear.

CSS responsive design ensures an automatic adjustment of font to be readable even on a tiny phone

Clickable image links

Font types & colors controlled by CCS

**Next Step:** Let's check the implementation code **alive!**



That's all. **Thanks for watching!**

*Oscar Gonzalez*

Phone: 303-931.0333

Email: [Oscar@geoloil.com](mailto:Oscar@geoloil.com)

Web-Site: <https://geoloil.com>