SAULO DA COSTA E SILVA — saulocpp@gmail.com, ssilva@lgc.com
Geophysicist – Halliburton, Landmark Graphics Corporation. Mobile: +47 40033534

1 Description

This document describes how the function *Vrms2Vint* present in the library *FileOps_GPU_Operations* bundled with FileOps, which reads an input AVF file of RMS velocities and writes an output AVF file of interval velocities calculated with the Dix Equation. For a given layer n, its interval velocity is determined by:

$$Vint_n = \sqrt{\frac{T_n Vrms_n^2 - T_{n-1} Vrms_{n-1}^2}{T_n - T_{n-1}}}$$

Legacy implementations employ serial processing, where a single thread is responsible for retrieving data from memory, calculating the new velocity and writing it back to memory in a long loop. To address this inneficiency, the CUDA® API was used to take advantage of NVidia cards for parallel processing. With the velocity conversion out of the way, disk work becomes the limiting factor of the process (GPUs can't directly access file systems), so effort was also put into efficient parsing.

2 Requirements

The function has the following requirements for system and input AVF:

• 64bit Windows, Linux or MacOSX: tested on Windows 8.1/10, Ubuntu 18.04 (both x86-64 and ARM64 on a Jetson Nano) and RHEL 6.10, MacOS 10.13.6;
• NVidia card with Compute Capability 3.0 or newer and free memory that is at least 1/3 the size of the input file;
• NVidia video driver of versions described in the installation guides for CUDA 10.2: https://docs.nvidia.com/cuda/archive/10.2/;
• Glibc version 2.27 (compiled in Ubuntu 18.04) for both PC and Jetson Nano;
• Header **must** contain 6 lines beginning with #, for example:

```
#FUNCTION_TYPE = TVrms
#LINEAR_UNITS = METERS
#DATUM = 0
#X_OFFSET = 0
#Y_OFFSET = 0
#
```

or

```
#
#FIELDS = Function ID, X, Y, Time, Vrms
#FUNCTION_TYPE = TVrms
#LINEAR_UNITS = METERS
#DATUM = 0.000000
#
```

• Each function must begin with time 0, such as:

```
Function1 495917.00 6782900.00    0.0000 1479.9832  ←
Function1 495917.00 6782900.00  190.8400 1479.9832
Function1 495917.00 6782900.00 1874.2000 1989.1157
Function1 495917.00 6782900.00 1918.5800 1995.6210
Function1 495917.00 6782900.00 1960.9399 2004.2107
Function1 495917.00 6782900.00 1996.6400 2011.8517
#
Function2 495917.00 6782887.00    0.0000 1479.9475  ←
Function2 495917.00 6782887.00  190.7500 1479.9475
Function2 495917.00 6782887.00 1873.7500 1989.1759
Function2 495917.00 6782887.00 1918.1300 1995.6813
Function2 495917.00 6782887.00 1960.5000 2004.2590
Function2 495917.00 6782887.00 1996.5300 2011.9640
```

3 Results

To see how *Vrms2Vint* performs against other velocity tools in this operation, the total time was measured as well as the difference between the calculated interval velocities, shown on Appendix B. Differences due to rounding doesn't mean that any of the applications is wrong, as the compilers can have different rules depending on the optimization or platform, being fully or partly compliant to the IEEE754 standard. RMS velocities also accumulate error in each subsequent layer.

TDQ version 5000.10.0.0.201501271138 was one of the applications tested, running on Linux RHEL 6.10 and on an Intel Xeon E5-2643 v2, 128GB of DDR3 RAM and NVidia P40-8Q. Vrms2Vint was tested on the same machine (column *Vrms2Vint A*). As for Petrel, the version tested was 2016 running on an HP ZBook 17" G2 with a Core i7-4910MQ 2900MHz, 32GB of DDR3L 1600MHz RAM and Windows 7.1. The AVF file used has 42239339 Vrms samples distributed in 334551 functions with a total size of 2,7 GB.

For completeness, column *Vrms2Vint B* refers to a test on the development machine: Intel Core i5-6500 3200MHz, 16GB DDR4 2133MHz RAM, NVidia GeForce GTX 1080 Ti, Ubuntu 16.04 with R390.30 video driver and HyperX 256GB Savage SSD. The table below shows the time taken by each application at the different stages of the operation. To roughly time TDQ, the command *date* was used as soon as the import and export started and finished. Petrel was also manually timed with a stopwatch from the moment the conversion started until the progress bar reached 100%. Vrms2Vint exposes the time taken to compute the velocities as well as the file operations, so in order to compare it to the other applications it is needed to analyze the table to infer how they perform:

	TDQ	Petrel	Vrms2Vint A	Vrms2Vint B
Reading input AVF	840 / **1980** sec	N/A	29,81 sec	30,97 sec
Calculating velocities	N/A	**1320** sec	0,0028 sec	0,0023 sec
Writing output AVF	80 sec	N/A	68,86 sec	70 sec
Aprox. total time	920 / **2060** sec	N/A	98,6728 sec	100,9723 sec

TDQ works internally with interval velocities, so when a velocity function file is loaded, it will be converted. The results also comfirm this, reading the input takes much longer than writing the output. The first run took 840 seconds and had the *Optimization filter* activated, which discards redundant samples "that lie within 1 (foot/meter) of the linearly interpolated value at the location", according to TDQ's manual.

Since Vrms2Vint doesn't employ any resampling and, instead, processes the entire input, a second run of TDQ with the filter deactivated was done in order to have a comparison in the same scenario, now resulting in 1980 seconds. Saving the output file took roughly 80 seconds in both cases, so the conversion is responsible for most of these 840 and 1980 seconds.

Petrel doesn't import AVF, so the velocity was imported from the OpenWorks database (which was saved as a seismic volume) and then the Dix Equation was applied. As a sidenote, the target velocity is named Instantaneous instead of Interval, according to the supplier, because the operation is done on sample pairs and not on geologic intervals (as is done when using the velocity modeling). In addition to this, the velocity is only effectively converted when the option "Realize" is toggled on.

It also doesn't export to AVF, requiring intermediate exports to various formats until an AVF is available, which could adversely affect the time measurement due to added clicks. Also important to note is that, by reading the velocity from a seismic volume in a Landmark format into a non-Landmark application, there was no guarantee that the data was not resampled in any way.

Vrms2Vint A isolates the time taken for each operation and the results are clear: its performance is vastly superior. However, this is not surprising, as a single-threaded application working on a SIMD task is slower than a massively parallel device designed for this kind of load. *Vrms2Vint B* refers to a budget home PC with both the operating system and Vrms2Vint running from a SSD disk on a USB 3.0 port. Once the data is in memory, the massively parallel implementation turns into disk-bound what was originally a CPU-bound operation. The computation is down to a few milliseconds.

The total speedup is unavailable for Petrel since the constant manual intervention to get the velocity in and out prevents reasonable timing. But for the conversion time Vrms2Vint is roughly **471428x faster** (yes, you read it correctly): 1320 seconds VS 2,8 milliseconds. Comparing to the time taken by TDQ to fully process the file we observe a performance increase of 20,87x, against 9,32x with the optimization filter on. Important to note that Vrms2Vint doesn't discard any sample, so more performance could be obtained either when reading or writing the file if that was the case.

As for TDQ's velocity conversion, it is not possible to isolate the operation but it is possible to make some assumptions since we have surrouding numbers. TDQ and Vrms2Vint (column A) were run on the same PC and we see that the time to output the AVF has a performance delta of ~13%. If we assume that TDQ has a file input and parser as fast as that on Vrms2Vint, we have file handling accounting for 110 seconds out of the 920 and 2060 seconds (optimization filter on and off).

This leaves 810 and 1950 seconds for the velocity computation in respective filter scenarios (no OpenWorks project selection or other Oracle tasks impairing the operation, it was already done upon program startup), which makes Vrms2Vint **289285x** and **696428x faster** in the respective cases for pure velocity computation.

4 Appendix A

The application was compiled and tested on different platforms, each with their respective CUDA-supported toolchain, and the runs used the same input file as before. Below is a description of the platforms of this complementary test:

• PC: The one used for column *Vrms2Vint B*. On Windows 8.1, MS Visual Studio 2017 Community Edition. On Ubuntu 16.04, g++ 5.4.0;
• Macbook Pro 15" Retina 2012: Intel Core i7 3600Mhz, 16GB DDR3 1600MHz RAM, NVidia GeForce GT650M, 512GB SSD, Xcode 9.4 (clang++).

The results in the following table are in seconds:

	PC – Linux	PC – Windows	Mac OS X
Reading input AVF	30,97	51,30	44,55
Calculating velocities	0,0023	0,0025	0,05
Writing output AVF	70	68,86	40,42
Aprox. total time	100,9723	120,1625	85,02

Interestingly the Macbook Pro outperformed a PC with newer hardware in total time. With an almost 15 seconds increase in the read stage, the cut on almost 30 seconds while writing provided 15 seconds in total time saved. Comparing to the results of the Windows and Linux runs on the PC, it is possible to infer that the compiler played an important role in this particular task. Windows has file handling routines in its API, such as FileRead(Ex)/FileWrite(Ex), but in this initial release it employs standard C/C++ I/O functions for easier cross-platform compilation.

The MBP has a GPU that is 2 generations older than that on the PC, with a smaller memory bus (critical on a memory-bound operation such as this Vint computation) and lower clocks, clearly reflecting on the increased time for the conversion: 50ms vs 2,5ms. Though the extra milliseconds are negligible in the bigger picture, in GPU computing a 20x performance difference between cards is huge.

5 Appendix B

The next pages show a table with the input time and Vrms of a randomly chosen function of the file, and the interval velocities calculated by different applications to compare the rounding due to floating-point arithmetics. It can be observed that the differences are negligible and the results are consistent among the tested applications. Petrel not included because, as explained before, it doesn't output AVF.

Input Time	Input Vrms	TDQ R5000	Vrms2Vint
0.0000	1479.9821	1479.9821	1479.9821
100.0000	1479.9821	1479.9821	1479.9821
150.0000	1525.0651	1611.4517	1611.4517
200.0000	1586.2969	1757.2369	1757.2366
250.0000	1643.7615	1855.9114	1855.9116
300.0000	1689.1344	1899.8134	1899.8135
350.0000	1736.4149	1996.7238	1996.7241
500.0000	1835.2573	2047.4161	2047.4160
550.0000	1851.0563	2002.2004	2002.2006
600.0000	1861.4159	1971.7830	1971.7834
750.0000	1875.5948	1931.2699	1931.2698
799.9999	1880.6302	1954.6062	1954.6057
850.0000	1887.8074	1999.1403	1999.1401
900.0000	1896.3151	2035.5122	2035.5123
950.0000	1905.2843	2060.0630	2060.0642
1100.0000	1928.8872	2072.1382	2072.1379
1150.0000	1936.9436	2106.4033	2106.4031
1200.0000	1946.2579	2149.3770	2149.3772
1249.9999	1957.2107	2203.8062	2203.8059
1300.0000	1968.9176	2241.8079	2241.8076
1450.0000	2000.9548	2259.6633	2259.6638
1650.0000	2030.5986	2233.7825	2233.7820
1700.0000	2037.4598	2252.1848	2252.1826
1750.0000	2044.8862	2283.0535	2283.0571
1800.0000	2053.0693	2321.3792	2321.3726
1850.0000	2062.3853	2373.5269	2373.5322
1900.0000	2072.5066	2417.3899	2417.3848
1949.9999	2083.2148	2455.7700	2455.7693
1999.9999	2094.3689	2490.7092	2490.7161
2050.0000	2105.8906	2523.9944	2523.9944
2100.0000	2118.2261	2573.5669	2573.5618
2150.0000	2131.7427	2637.6196	2637.6216
2200.0000	2146.8115	2716.8191	2716.8196
2249.9998	2163.6538	2806.3735	2806.3750
2300.0000	2181.8228	2883.2664	2883.2605
2350.0000	2201.0276	2952.3564	2952.3591
2400.0000	2220.7488	3005.0420	3005.0352
2450.0000	2241.1348	3064.1045	3064.1106
2499.9998	2262.2605	3127.2346	3127.2334
2550.0000	2284.7871	3215.8513	3215.8503
2600.0000	2308.9395	3315.0618	3315.0605
2650.0000	2334.9419	3425.0618	3425.0627
2700.0000	2362.4182	3524.4175	3524.4221
2750.0000	2391.0039	3613.0918	3613.0864
2800.0000	2420.2510	3687.4058	3687.4055
2850.0000	2449.8503	3751.5662	3751.5664
2900.0000	2479.3582	3799.7205	3799.7239

Input Time	Input Vrms	TDQ R5000	Vrms2Vint
2949.9998	2508.5208	3837.8931	3837.8875
3000.0000	2537.0642	3864.4102	3864.4219
3050.0000	2564.7344	3879.1638	3879.1663
3150.0000	2618.2361	3913.1531	3913.1492
3199.9998	2644.4182	3958.9875	3958.9910
3350.0000	2719.5457	3999.1057	3999.1045
3400.0000	2742.3137	3981.4688	3981.4756
3449.9998	2763.5061	3946.0576	3946.0562
3500.0000	2783.3799	3918.3647	3918.3625
3550.0000	2801.9451	3887.0740	3887.0762
3750.0000	2869.1936	3873.2021	3873.2021
3800.0000	2885.0229	3892.9138	3892.9141
4200.0000	2999.6055	3924.6963	3924.6973
4300.0000	3025.2812	3955.9412	3955.9392
4350.0000	3038.3101	4003.1877	4003.1733
4400.0000	3051.6265	4045.7068	4045.7058
4450.0000	3065.3250	4095.1211	4095.1238
4499.9995	3079.5049	4151.8994	4151.9028
4550.0000	3093.9707	4196.4282	4196.4336
4600.0000	3108.8228	4248.3042	4248.2974
4650.0000	3123.8628	4287.4854	4287.4917
4700.0000	3139.1926	4334.0068	4333.9980
4749.9995	3154.6792	4372.0005	4372.0000
4800.0000	3170.2454	4404.9414	4404.9468
4850.0000	3185.9192	4440.0610	4440.0552
4900.0000	3201.6931	4474.8496	4474.8550
4950.0000	3217.5774	4510.5884	4510.5806
4999.9995	3233.6689	4553.4678	4553.4624
5050.0000	3249.7634	4582.0874	4582.0942
5100.0000	3265.9614	4618.0293	4618.0376
5150.0000	3282.1594	4646.6670	4646.6484
5200.0000	3298.3574	4675.2554	4675.2700
5250.0000	3314.4487	4695.9585	4695.9570
5300.0000	3330.5398	4724.3628	4724.3613
5350.0000	3346.6348	4753.0679	4753.0693
5400.0000	3362.7261	4781.1992	4781.2119
5450.0000	3379.0308	4825.9189	4825.9082
5500.0000	3395.4397	4862.6768	4862.6836
5550.0000	3412.1621	4915.8096	4915.8027
5600.0000	3429.0962	4961.6606	4961.6724
5649.9995	3446.4519	5024.2563	5024.2534
5700.0000	3464.0352	5072.4712	5072.4639
5750.0000	3481.9688	5130.7617	5130.7500
5800.0000	3500.0151	5171.0845	5171.0850
5850.0000	3518.2986	5221.4253	5221.4355
5899.9995	3536.5779	5253.1660	5253.1455
5950.0000	3555.0896	5303.7183	5303.7212
6000.0000	3573.5977	5335.8389	5335.8467

Input Time	Input Vrms	TDQ R5000	Vrms2Vint
6050.0000	3592.1099	5368.5854	5368.5732
6149.9995	3629.0176	5412.3940	5412.3989
6200.0000	3647.4131	5455.9429	5455.9458
6300.0000	3683.8542	5489.5171	5489.5098
6399.9995	3719.8374	5533.6255	5533.6260
6450.0000	3737.6558	5565.8242	5565.8467
6550.0000	3773.1724	5607.3882	5607.3833
6600.0000	3790.9912	5659.3535	5659.3506
6649.9995	3808.7812	5688.0166	5688.0146
6700.0000	3826.6199	5723.4780	5723.4727
6750.0000	3844.4551	5754.3921	5754.3901
6850.0000	3880.5142	5819.0093	5819.0059
6899.9995	3898.7300	5883.3711	5883.3589
6950.0000	3916.9497	5915.5068	5915.5371
7000.0000	3935.2937	5958.8833	5958.8896
7050.0000	3953.7656	6002.8467	6002.8232
7100.0000	3972.3621	6046.6958	6046.7100
7150.0000	3990.9587	6079.1958	6079.2119
7200.0000	4009.8088	6135.5820	6135.5679
7250.0000	4028.7834	6180.2109	6180.2329
7299.9995	4047.8889	6225.7983	6225.7935
7350.0000	4067.1191	6270.9712	6270.9634
7400.0000	4086.6011	6328.6089	6328.6060
7450.0000	4106.0830	6362.5757	6362.5679
7500.0000	4125.6904	6408.6084	6408.6304
7549.9995	4145.4292	6455.5674	6455.5464
7600.0000	4165.1353	6486.6714	6486.6831
7650.0000	4184.7612	6513.1392	6513.1523
7700.0000	4204.5303	6561.4829	6561.4673
7750.0000	4224.2998	6595.9121	6595.9307
7799.9995	4244.0654	6629.9839	6629.9897
7850.0000	4263.9683	6678.1528	6678.1211
7900.0000	4284.0117	6726.9272	6726.9404
7950.0000	4304.0552	6761.8354	6761.8164
7999.9995	4324.3760	6824.8438	6824.8975
8050.0000	4344.6963	6860.1689	6860.1343
8100.0000	4365.0161	6895.4424	6895.4326
8150.0000	4385.4756	6945.1299	6945.1577
8200.0000	4406.0698	6994.6606	6994.6870
8249.9990	4426.6650	7030.6318	7030.6177
8350.0000	4467.9966	7091.7021	7091.6992
8400.0000	4488.5908	7138.0195	7138.0210
8499.9990	4529.3711	7169.8750	7169.8579
8550.0000	4549.5415	7199.2104	7199.2349
8650.0000	4589.4712	7229.3477	7229.3613
8700.0000	4609.2202	7257.6284	7257.6045
8850.0000	4667.1641	7277.2319	7277.2222
8999.9990	4723.0918	7300.2212	7300.2153