External Features from RHINO Pipeline v2.0

20181220

Overview

The RHINO v2 External Features are derived from the 'Direct Extracted Features', which are in turn calculated from processed and filtered seismic 'traces'.

These features are generated from the axial and tangential components. For each of these sensor components there are two versions of the features -- the uncalibrated and the calibrated versions.

The physical properties we intend to monitor are :

Property	**Note**
delay	Time interval between wavelet arrivals. Appears to be different for axial and tangential components, with tangential delay approximately twice the axial delay. This should theoretically be proportional to 1/velocity[1]
Density	Mass per unit volume
bulk modulus	The 'coefficient of stiffness' or 'bulk modulus' is proportional to density and the square of the compressional or 'p-wave' velocity
Reflection coefficient	(1 -R) / (1 + R) where R is the ratio of amplitude of the first multiple to the primary[2].
shear modulus	It is proportional to density and the square of the shear or 's-wave' velocity[3]
uniaxial compressive strength[4] (ucs)	the maximum axial compressive stress that a right-cylindrical sample of material can withstand before failing. Aka:

[1] Reference needed

[2] This is sometimes cited as (Z1-Z2)/(Z1+Z2) were Z1, Z2 are the 'impedances' of the media at the interface. Multiply through by 1 = (1/Z1) / (1/Z1) to see that R = Z2/Z1; http://users.physics.harvard.edu/~schwartz/15cFiles/Lecture9-Impedance.pdf

[3] https://en.wikipedia.org/wiki/Shear_modulus Or *modulus of rigidity* (G or μ) describes an object's tendency to shear (the deformation of shape at constant volume) when acted upon by opposing forces.

[4] https://www.glossary.oilfield.slb.com/en/Terms/u/uniaxial_compressive_strength.aspx

	unconfined compressive strength. We expect this proportional to the sqrt of the primary wavelet
velocity	For axial component this is the compressional velocity and for tangential component this is the 'shear' velocity[5]

External features which are derived from processed traces are designated by a leading letter and then a text string identifying the feature.

The following designators are used:

Designator	Component	Calibrated	Note
a	axial	No	
t	tangential	No	
c	axial	Yes	compressive
s	tangential	Yes	shear

By "*calibrated*" we mean that we have mapped the feature so that the values can be interpreted in terms of physical properties.

External Feature Labels

So the external feature labels are as listed in the Table below:

Feature Name	Component/Property	Calibrated	Note
a_delay	Axial / delay	No	
a_dens	Axial / density	No	
a_mod	Axial / modulus	No	

[5] Note that in the pipe the wave travels at 'pipe velocity', the velocity in the medium will be incorporated into the processed trace in the form of a phase shift or apparent delay

a_reflection_coef	Axial / reflection_coef	Yes ?	
a_str	Axial / ucs	No	
a_vel	Axial / velocity	No	
c_density	Axial / density	Yes	
c_modulus	Axial / modulus	Yes	
c_strength	Axial / UCS	Yes	
c_velocity	Axial / velocity	Yes	
t_delay	Tangential / delay	Yes	Calibrated yes, but not a rock property
t_mod	Tangential / modulus	No	
t_reflection_coefficient	Tangential / reflection_coef	No	
t_vel	tangential / velocity	No	
s_modulus	Tangential / modulus	yes	
s_velocity	tangential / velocity	yes	

Recipes for Practical Calculation of Features:

a_delay: a_delay = axial_multiple_peak_time_sample - axial_primary_peak_time_sample

This formula resulted in a 'blocky' looking log because the sample-granularity that was applied resulted in only a few discrete values being observed. Variation over a blasthole was often only observed to have three or four distinct values. When smoothing was applied to these logs they looked somewhat plausible. We can reduce the blockyness by using axial_multiple_peak_time_poly, and axial_primary_peak_time_poly, i.e. the polyfit critical points rather than those at sample granularity.

a_dens: $a\_reflection\_coef / (a\_vel)^2$

Previously known as "pseudo_density", we have been calculating it as:
a_dens = reflection_coefficient_sample / primary_pseudo_velocity_sample**2
Where the primary_pseudo_velocity_sample = 1. / self.primary_wavelet_width_sample

Which can be expressed as
a_dens = reflection_coefficient_sample * self.primary_wavelet_width_sample**2

N.B.: this used primary_wavelet_width_sample which has been suggested is not a valid measurement[6]: For the record:
primary_wavelet_width_sample = axial_primary_zero_crossing_after_sample - axial_primary_left_trough_time[7]

a_mod: k1 * a_reflection_coef
Not previously calculated, just plotted as reflection_coef, i.e. k1=1.0;
k1 a constant to be determined; may depend on MWD

a_reflection_coef: (1-R) / (1+R)
where R is the multiple-to-primary-amplitude ratio (axial)
Need to confirm whether we have been / want to use amplitude_ratio or amplitude_ratio_sample

```
def reflection_coefficient(self):
    return (1.0 - self.amplitude_ratio) / (1.0 + self.amplitude_ratio)
def reflection_coefficient_sample(self):
    return (1.0 - self.amplitude_ratio_sample) / (1.0 + self.amplitude_ratio_sample)
```

where

```
def amplitude_ratio(self):
    return self.df['axial_multiple_peak_amplitude'] / self.axial_primary_peak_amplitude

def amplitude_ratio_sample(self):
        return self.df['axial_multiple_peak_sample'] / self.df['axial_primary_peak_sample']
```

a_str: sqrt(self.axial_primary_peak_sample)
Previously known as pseudo_ucs;

We looked at logs of this quantity with _sample resolution vs _poly (polynomail fit) and found little difference to first order behaviour.

a_vel: 1 / a_delay

This is the “delay velocity”. It is the one that has been ‘blocky’ in the past

c_density: This will be some scaled version of **a_dens**, for now we set equal to **a_dens**
c_modulus: This will be some scaled version of **a_mod**, for now we set equal to **a_mod**
c_strength: This will be some scaled version of **a_str**, for now we set equal to **a_str**

---

[6] Jamie can you comment on this?
[7] From dcrhino/process_pipeline/qc_log_plotter.py:L471

c_velocity: This will be some scaled version of **a_vel**, for now we set equal to **a_vel**

t_delay: tangential_delay = tangential_multiple1_time_poly - tangential_primary_time_poly
Here we are using polynomial fits so we are not so 'blocky'[8]. The label for this feature is 1810_tangential_delay

t_mod: k2 * t_reflection_coef
Not previously calculated, k2 a constant to be determined; may depend on MWD

t_reflection_coef: (1-R) / (1+R) where R is the multiple-to-primary-amplitude ratio (tangential)
In the code this is referred to as *1810_tangential_impedance*
Tangential_impedance = (1 - 'tangential_amplitude_ratio') / (1 + 'tangential_amplitude_ratio')
Where
tangential_amplitude_ratio' = features['tangential_multiple1_amplitude_poly'] / 'tangential_primary_amplitude_poly']

t_vel: 1 / t_delay

s_modulus: This will be some scaled version of **t_mod**, for now we set equal to **t_mod**

s_velocity : This will be some scaled version of **s_vel**, for now we set equal to **s_vel**

[8] dcrhino/analysis/unstable/feature_extraction/feature_extraction_tangential_bob_20181031.py