Developing Automated Geospatial Procedures in Civil Engineering

Joseph Kaelin - Katharina Kaelin - Philippe Nater - Alan Hodgkinson

Civil Engineering

Motivation

low productivity in Civil Engineering

dominance of ad hoc procedures and lack of standard procedures

automation and **reuse**

coding is a good way to automate and push standard procedures

Automated Procedures for Civil Engineering

aim?

automate routine work

most tasks are routine

use standard design criteria & methodology

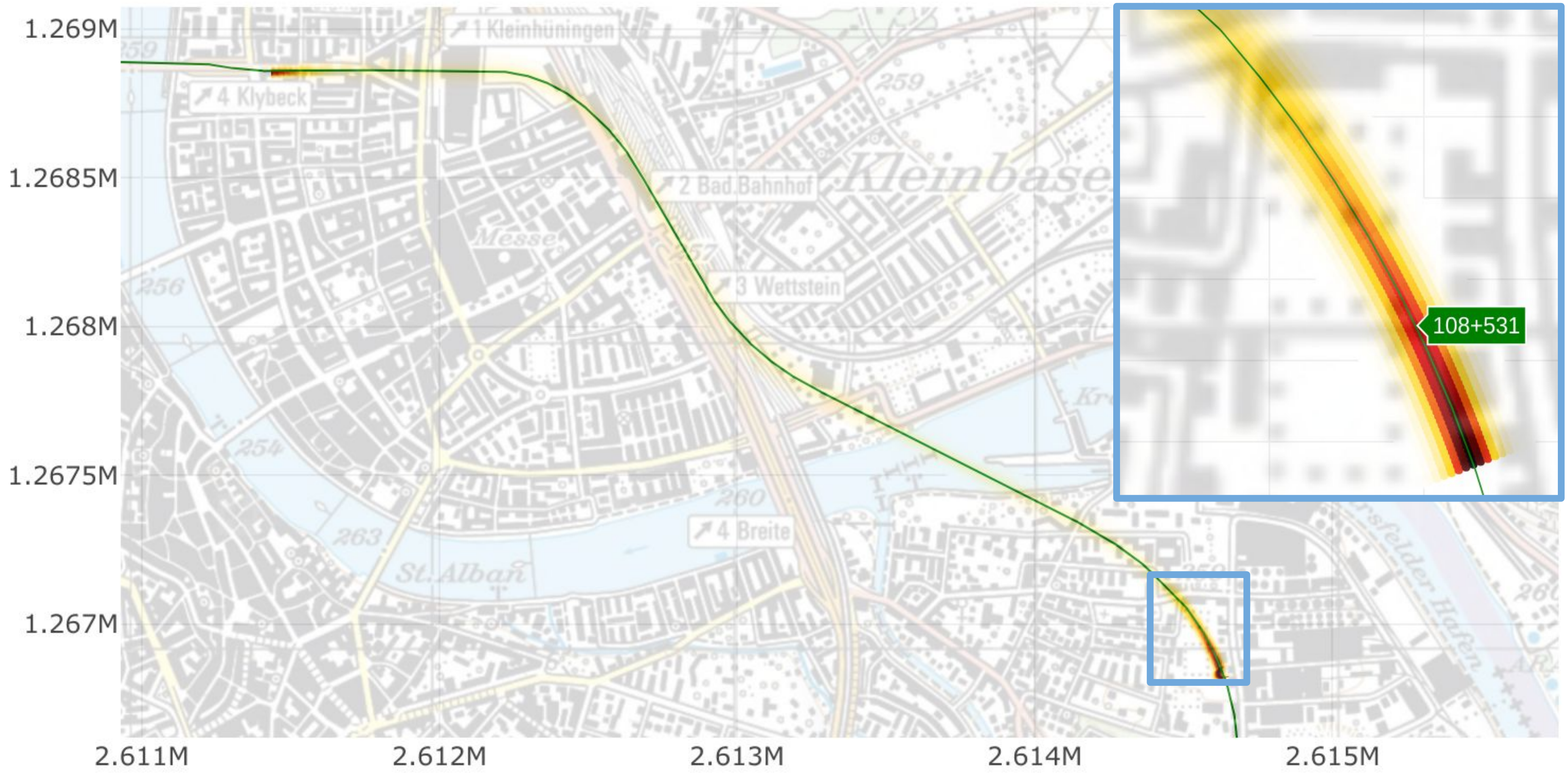
not aimed at replacing engineer

free engineers to add project-specific value

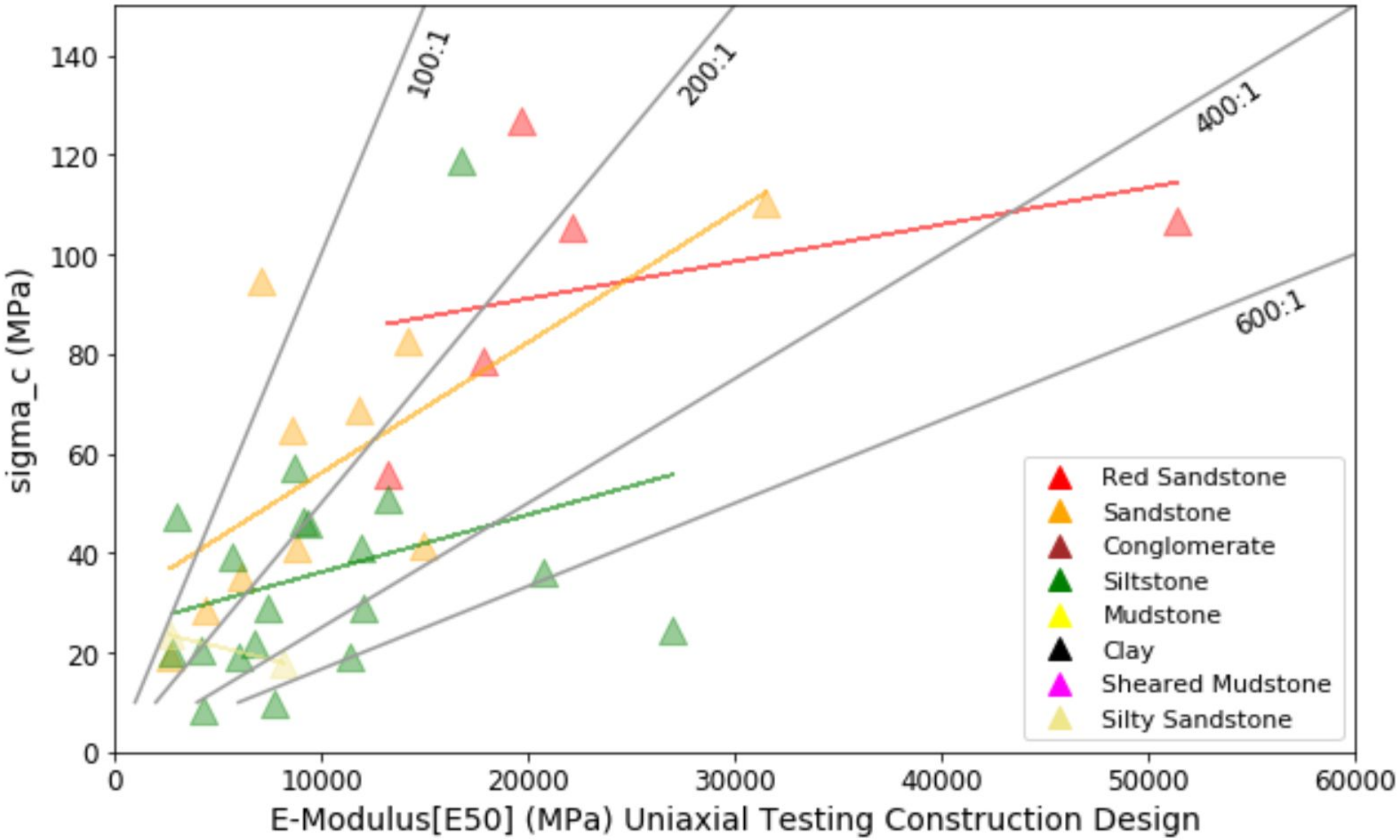
need?

- feasibility investigations
- alternative studies
- design & documentation
- optimization

TunnelGIS - surface settlements (GeoPython 2017)



Data Organizer - rock lab data (GeoPython 2017)



Decision Support System

integrated data

domain-specific procedures

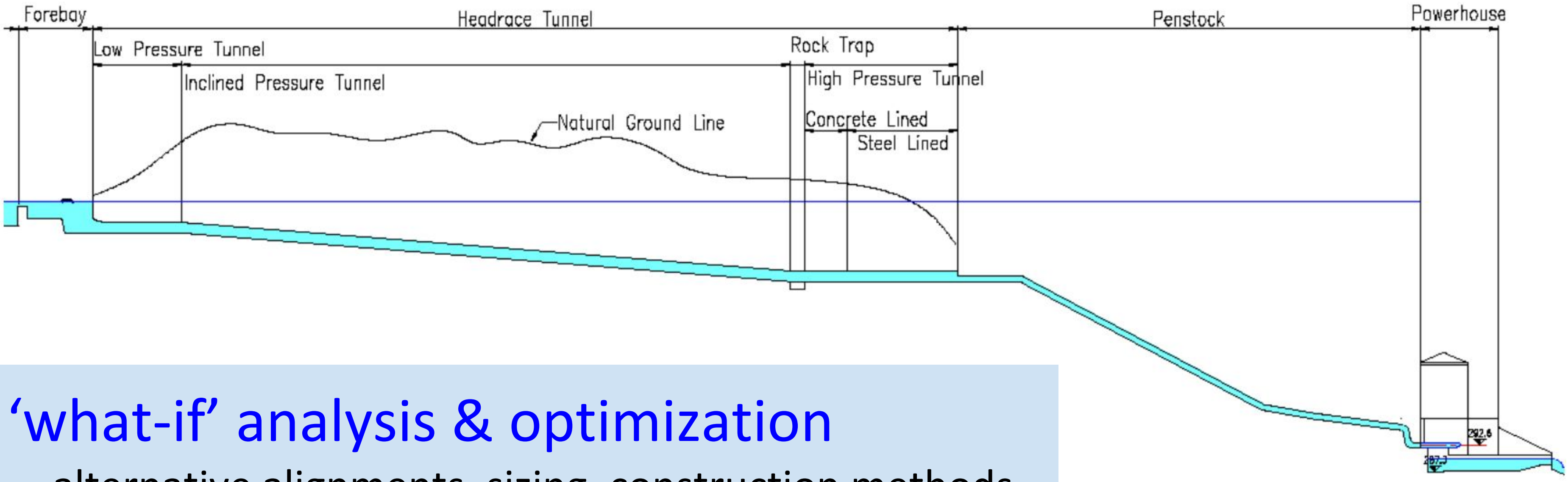
select, document, code engineering procedures

- provides substantial time-savings
- improves quality control

selection criteria

- ‘fitness for purpose’
- building codes
- industry standards
- cost

Initial Focus is on Hydropower Waterways



‘what-if’ analysis & optimization

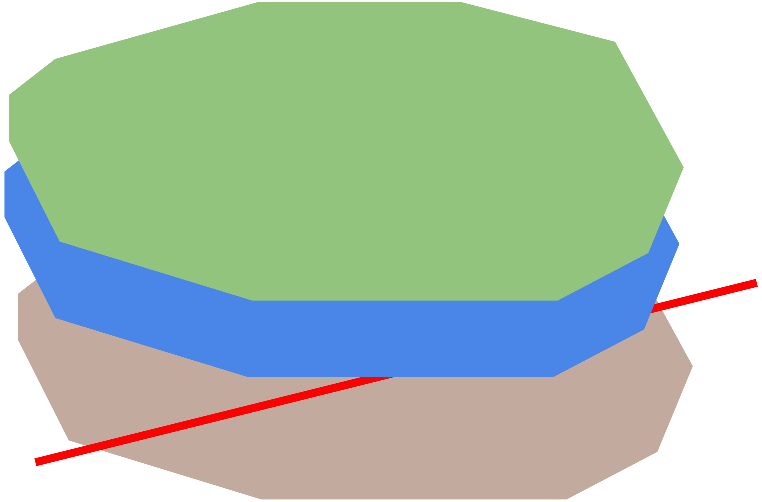
alternative alignments, sizing, construction methods

variations in geological forecast

selection criteria

hydraulics, geology, leakage, constructibility, cost, construction time

Input Data



Terrain DTM
- hillslopes, watersheds, stream courses

Groundwater surface

Rock surface
- geological parameters

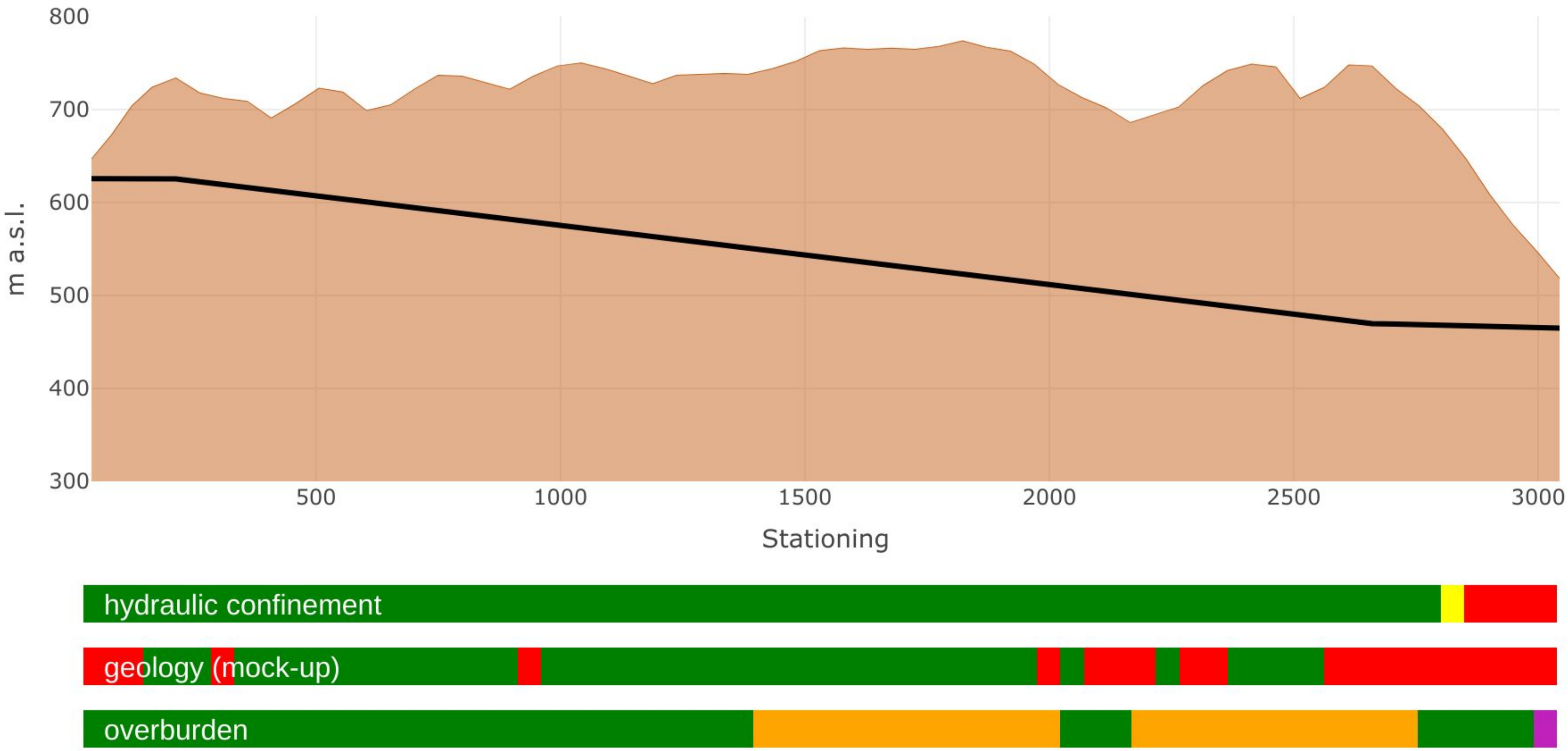
Alignments
- layout of project features

better organisation and visualization of input data

Results as Dashboard & Design Report

tunnel layout	plan views (map) longitudinal section (profile)	
tunnel cross-sections	excavation rock support quantities	python CAD?
result plots & tables		
work schedules	time-distance diagrams	

Example Results with Selection Criteria



Minimum Viable Product

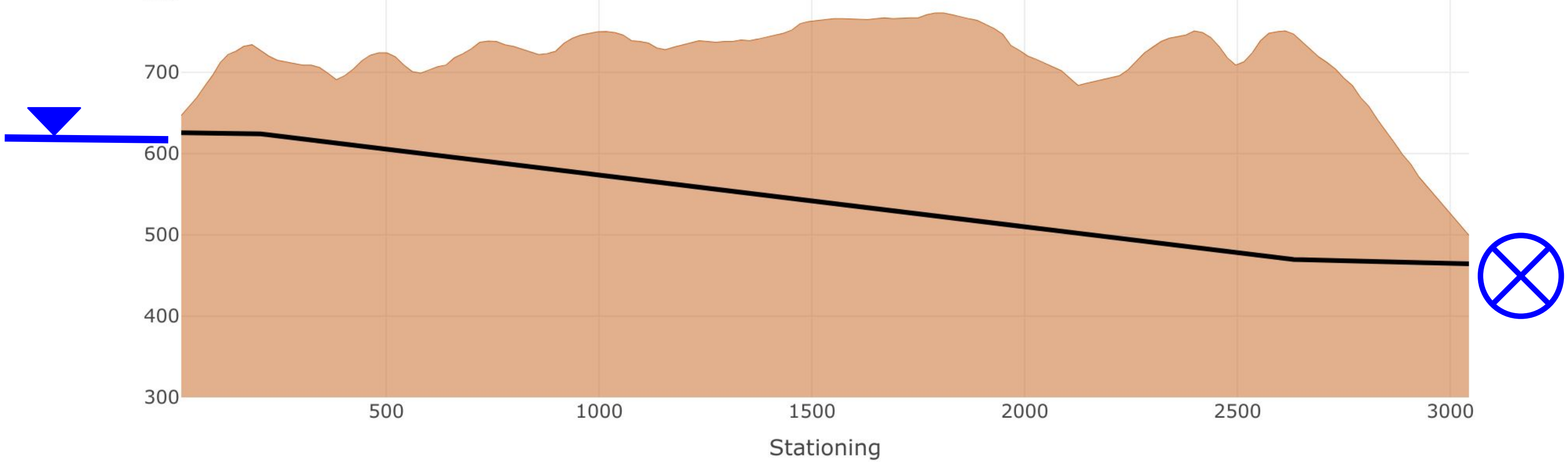
proof of concept

test user acceptance

feedback for future work

just enough to see if it works

Focus on Hydropower Waterways



unlined pressures tunnels

- widespread use due to economics
- requires rock mass to be tight against leakage
- concrete lining, steel lining in short sections

design criteria for hydraulic confinement

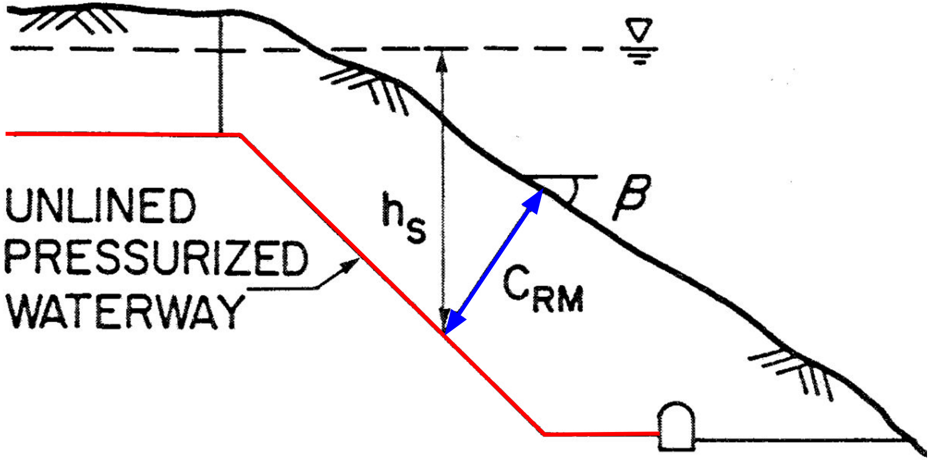
- prevent opening of joints

Civil Engineering

Design Criteria Hydraulic Confinement

Norwegian criteria selected as best practice

- confinement assumed related to overburden stress
- design validation requires hydraulic jacking tests



$$C_{RM} = \frac{h_S \gamma_W F}{\gamma_R \cos \beta}$$

C_{RM} = MINIMUM ROCK COVER

h_S = STATIC HEAD

γ_W = UNIT WEIGHT OF WATER

γ_R = UNIT WEIGHT OF ROCK

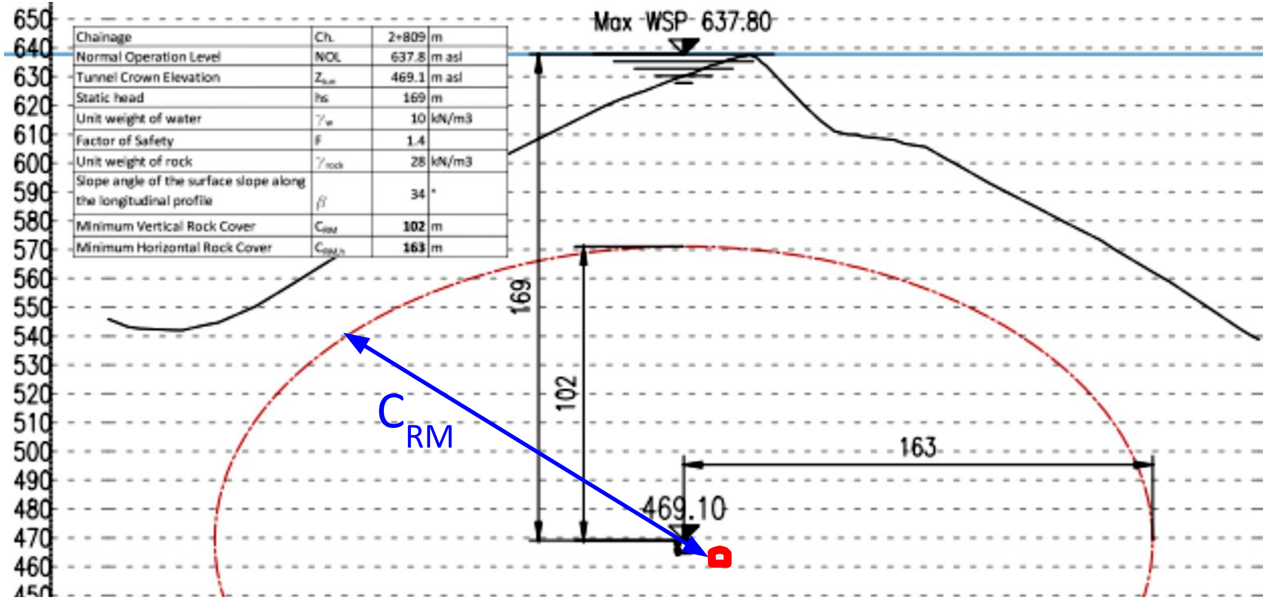
β = SLOPE ANGLE (VARIES ALONG SLOPE)

F = SAFETY FACTOR

Design Procedure Hydraulic Confinement

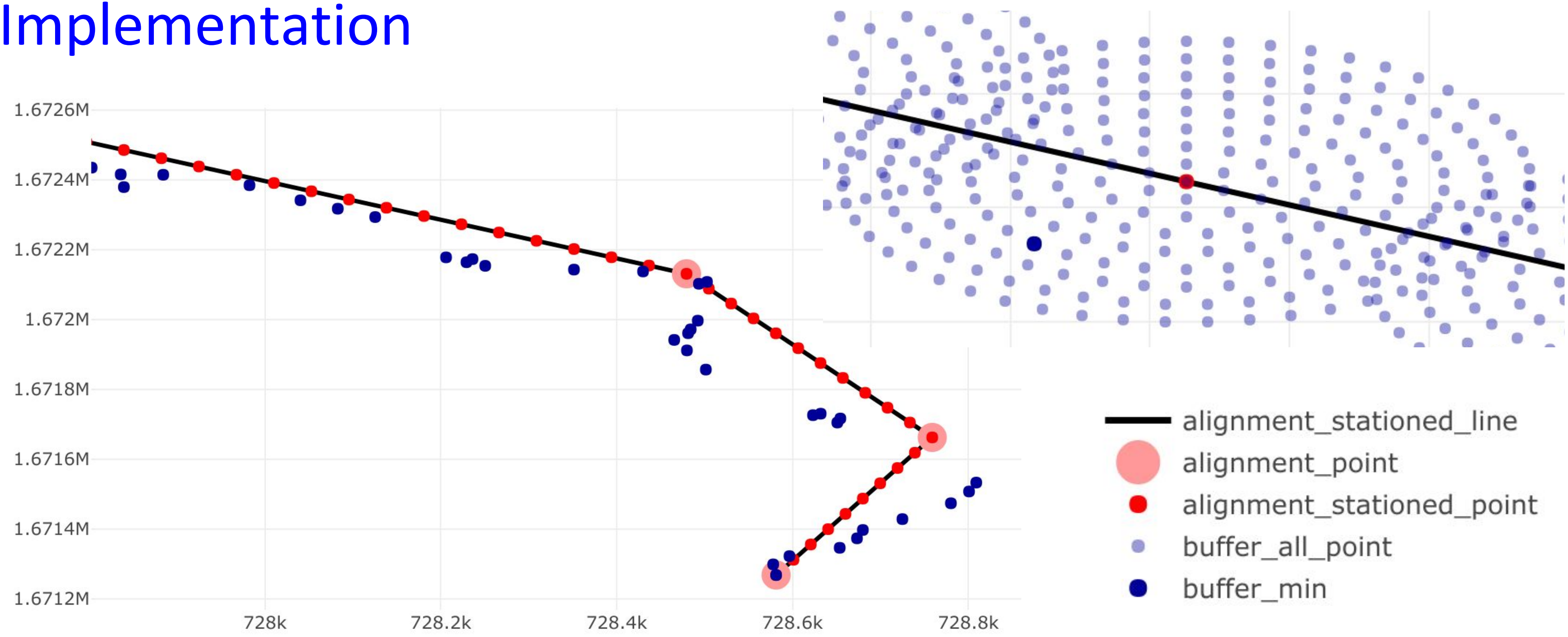
current practice

- draw cross-sections, longitudinal sections at representative tunnel locations
- determine C_{RM}, β at point on surface closest to tunnel
- calculate safety factor F in spreadsheet



Chainage	Ch.	2+809	m
Normal Operation Level	NOL	637.8	m asl
Tunnel Crown Elevation	Z_{tun}	469.1	m asl
Static head	hs	169	m
Unit weight of water	γ_w	10	kN/m3
Factor of Safety	F	1.4	
Unit weight of rock	γ_{rock}	28	kN/m3
Slope angle of the surface slope along the longitudinal profile	β	34	°
Minimum Vertical Rock Cover	C_{RM}	**102**	m
Minimum Horizontal Rock Cover	$C_{RM,h}$	**163**	m

Implementation



python at buffer points along stationed alignment

- select buffer point with least rock cover C_R
- calculate safety factor F

Python Ecosystem for Implementation

pyqgis processing.runalg with grass

using runalg vs grass alone?

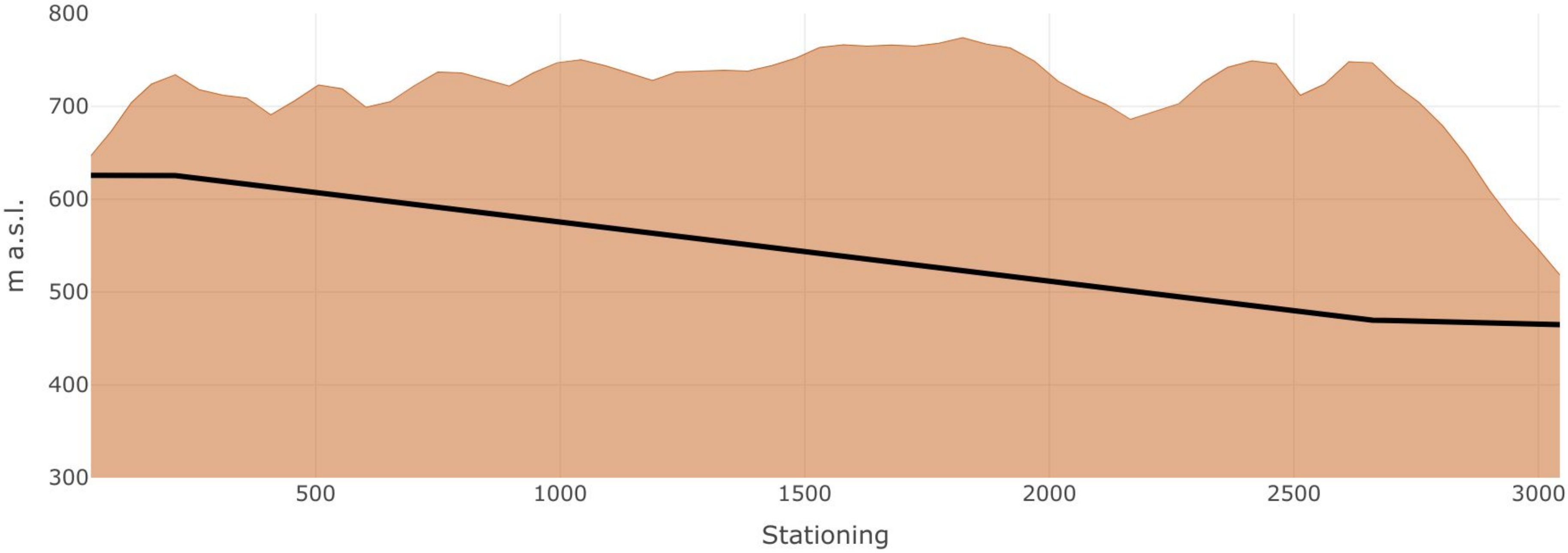
pandas, geopandas, numpy

plotly

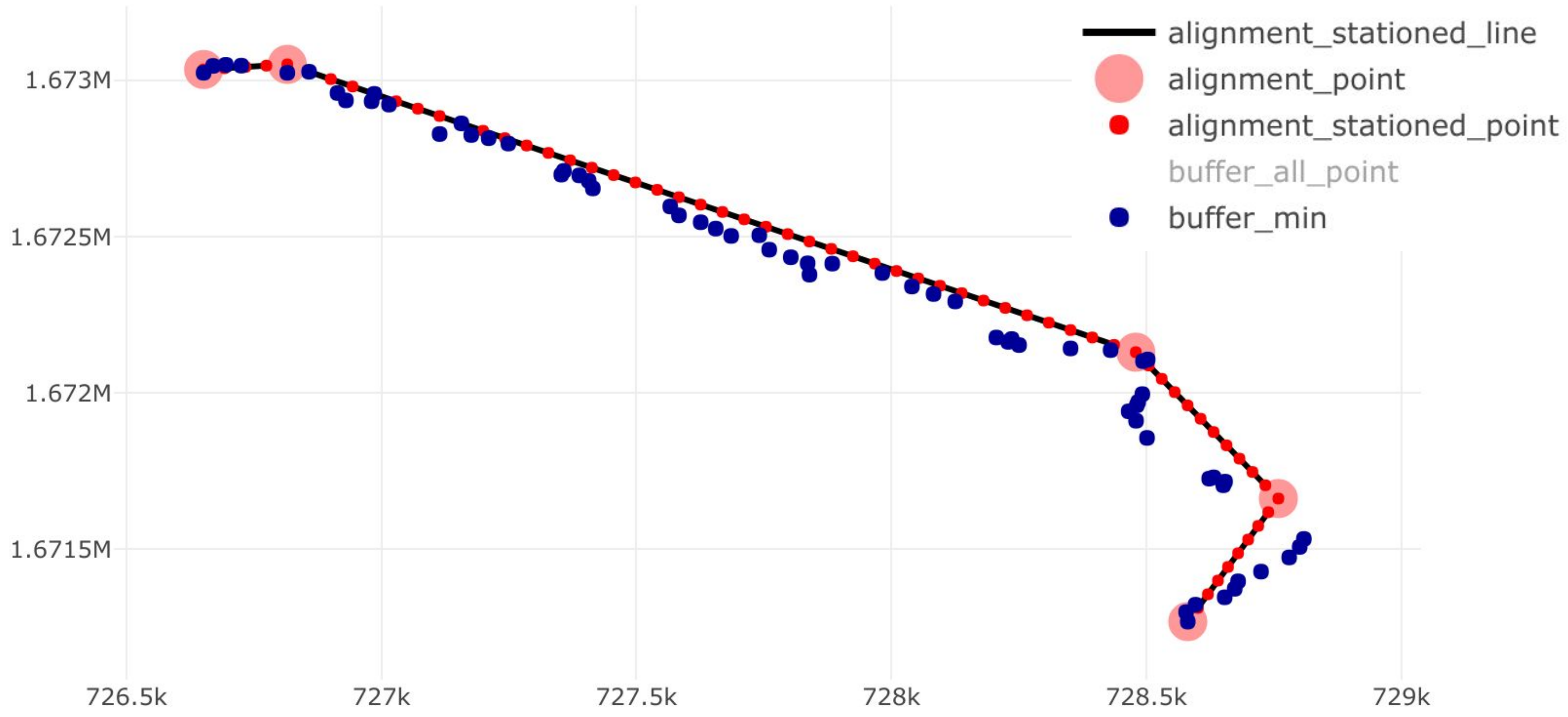
(couchbase)

rock density

Results for Hydraulic Confinement - plan view



Results for Hydraulic Confinement - profile



Improving User Experience

browser dashboard

practical way to make project spatial data available

- for use at construction sites
- for engineers who won’t use Jupyter, python
- visualization for discussions, client presentations

Browser Dashboard - Implementation

progressive web app (PWA) using Polymer web components

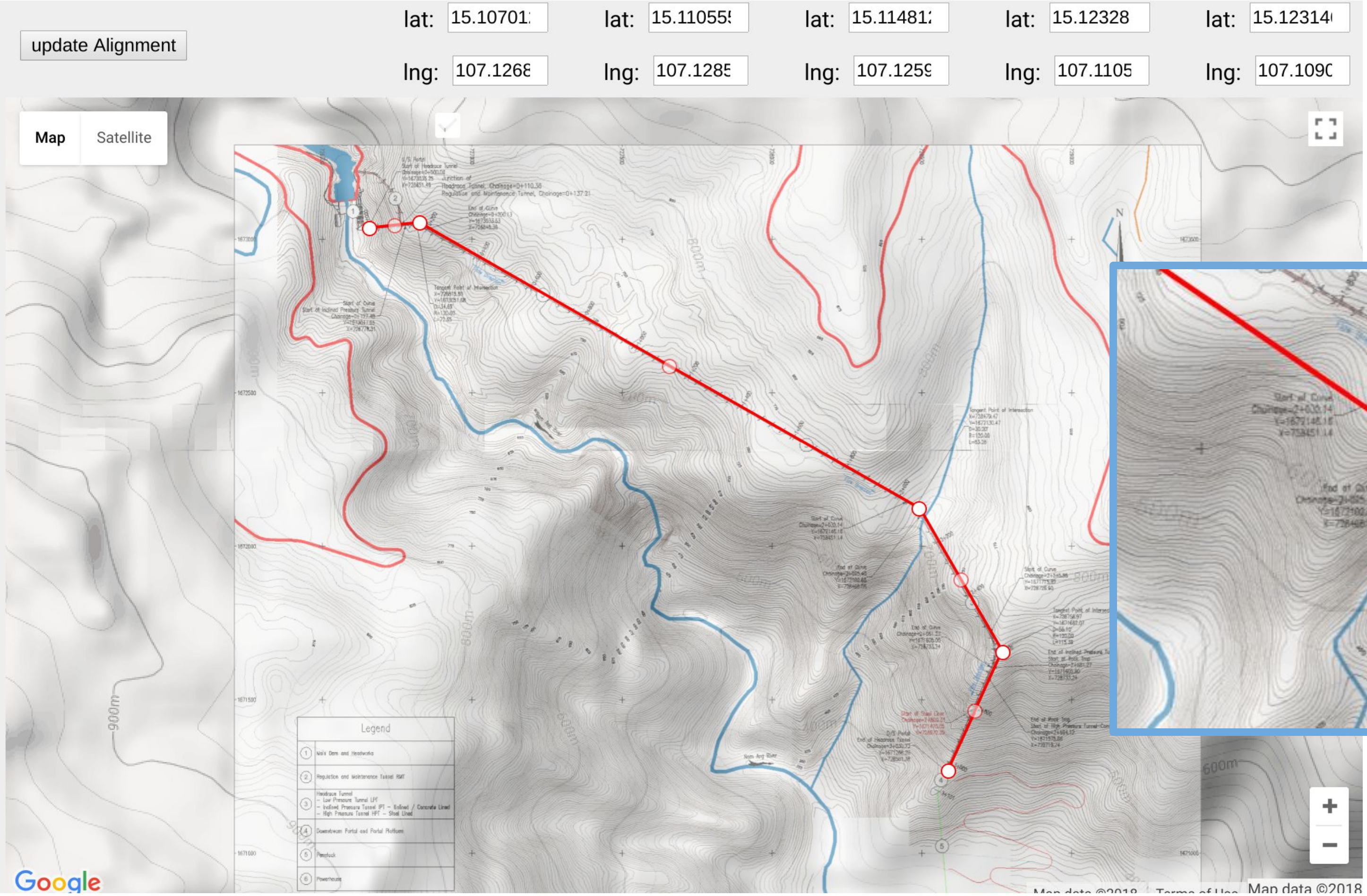
- plan view (map): Google Maps API
- profiles: plotly
- results plots: plotly

build API's from procedures in Jupyter Notebook

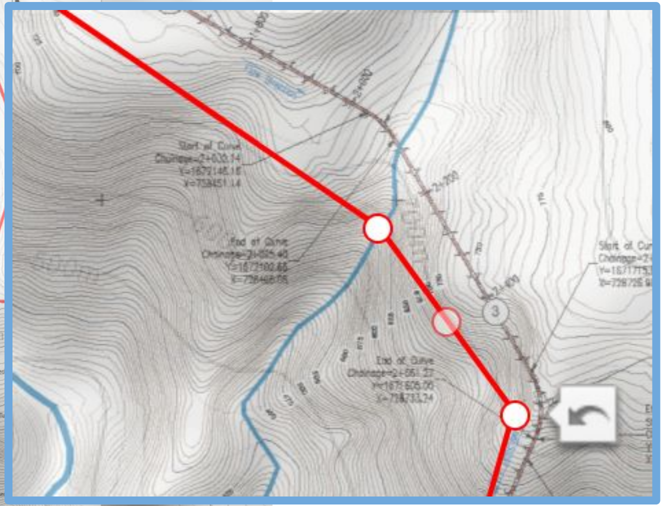
- moving procedures to API requires robust error checking
- using API for common procedures bolsters using Notebook as design report

python procedures accessed by Polymer from Flask Server

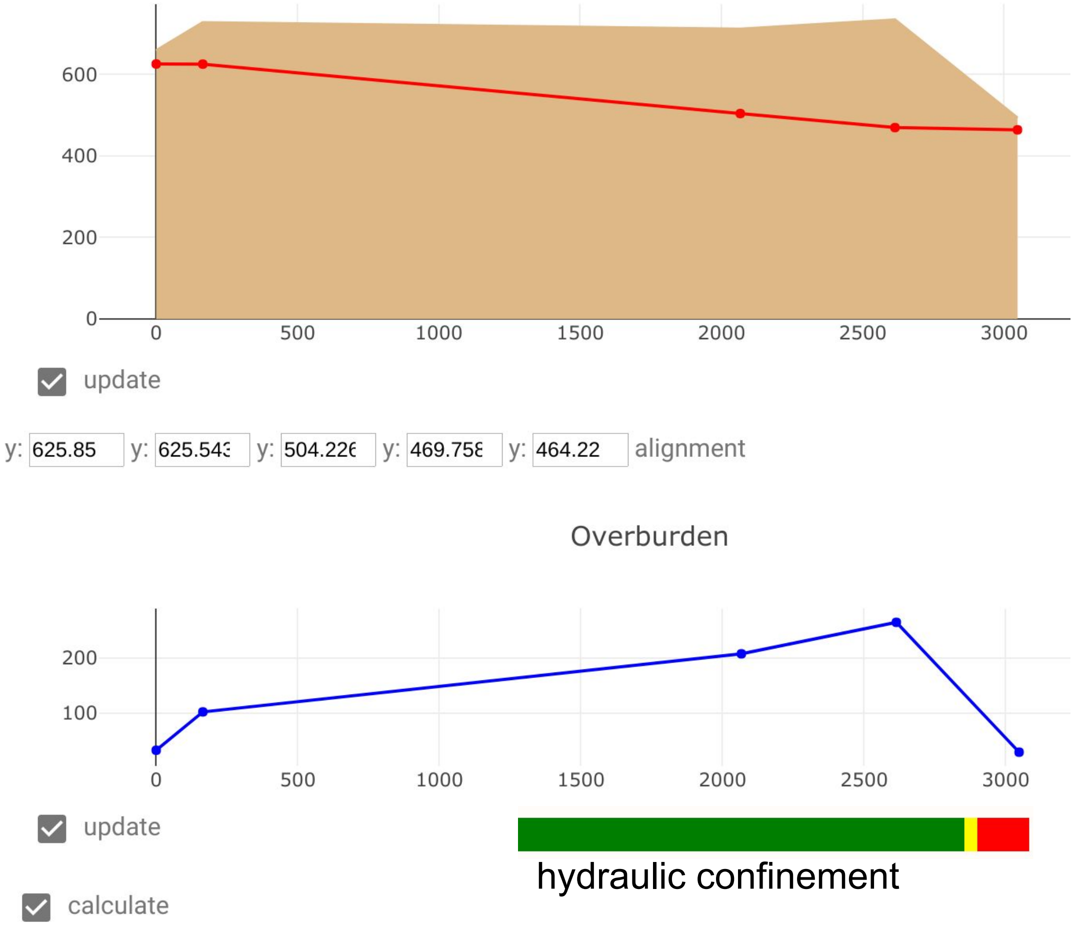
Plan View - Google Maps API

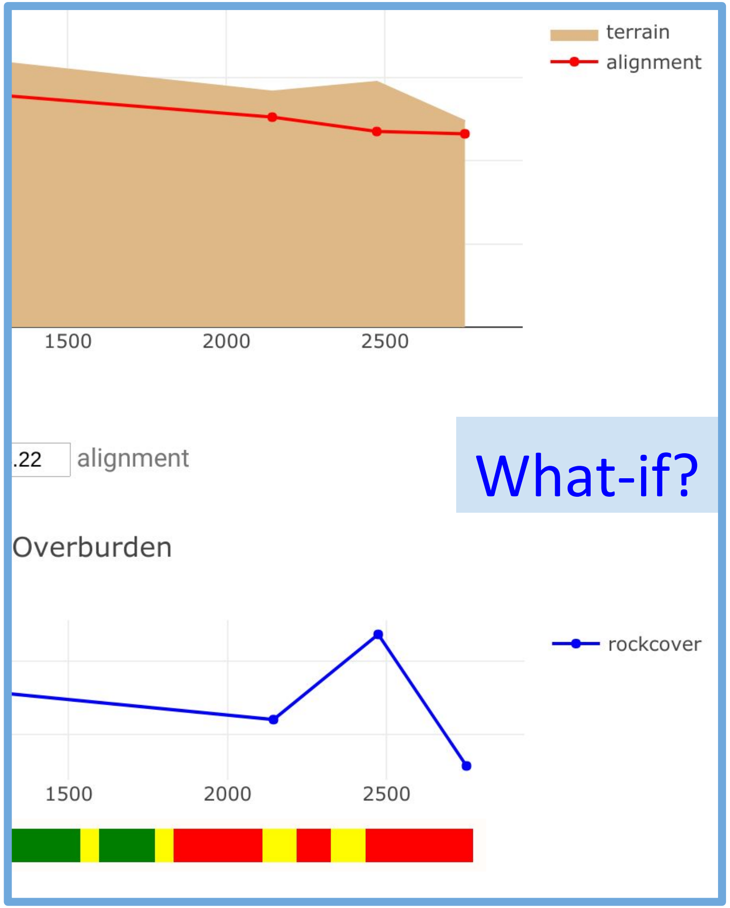


What-if?

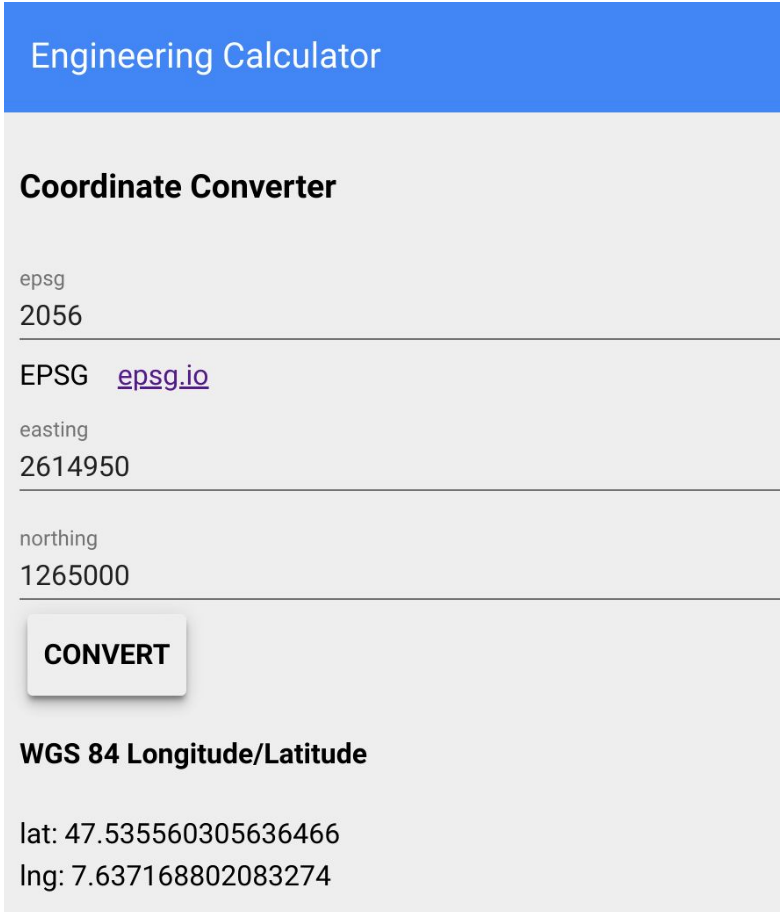


Profile - plotly





Polymer and Flask Server Implementation



Polymer app-drawer component

Polymer Web Components

```
 <iron-ajax
      url="http://gis.jkaelin.com:nnnn/coordinates"
      id="postExample"
      handle-as="json"
      content-type="application/json"
      method="POST"
      params='{{postBody}}'
      on-response="serverSuccess"
      last-response="{{serverSuccessObj}}"
      on-error="serverError"
      last-error="{{serverErrorObj}}">
      </iron-ajax>
</template>
```

web component

```
<script>
  class MyView1 extends Polymer.Element {
      static get is() { return 'my-view1'; }

      static get properties() {
          return {
              epsg: {
                  type: String,
                  notify: true,
                  value:'2683418',
              },
```

javascript class

Flask Server

```
@app.route('/coordinates', methods=['POST', 'GET'])
def pyproj_test():
    # ..... variable initialization deleted
    easting = request.values['easting']
    northing = request.values['northing']
    epsg = request.values['epsg']
    easting = request.values['easting'].split(',')
    northing = request.values['northing'].split(',')

    inProj = pp.Proj(init='epsg:'+str(epsg))
    outProj = pp.Proj(init='epsg:4326')  #WGS84

    try:
        for i in range(len(easting)):
            x1.append(float(easting[i]))
            y1.append(float(northing[i]))
            lnglat.append(pp.transform(inProj, outProj, float(easting[i]), float(northing[i])))

        return jsonify({'lng': [item[0] for item in lnglat], 'lat': [item[1] for item in lnglat]})

    except:
        pass

    return jsonify({'entry': 'invalid'})
```

Thank you for listening!

Thanks to Python and Polymer open source communities
Thanks also to Pöyry colleagues for support and feedback