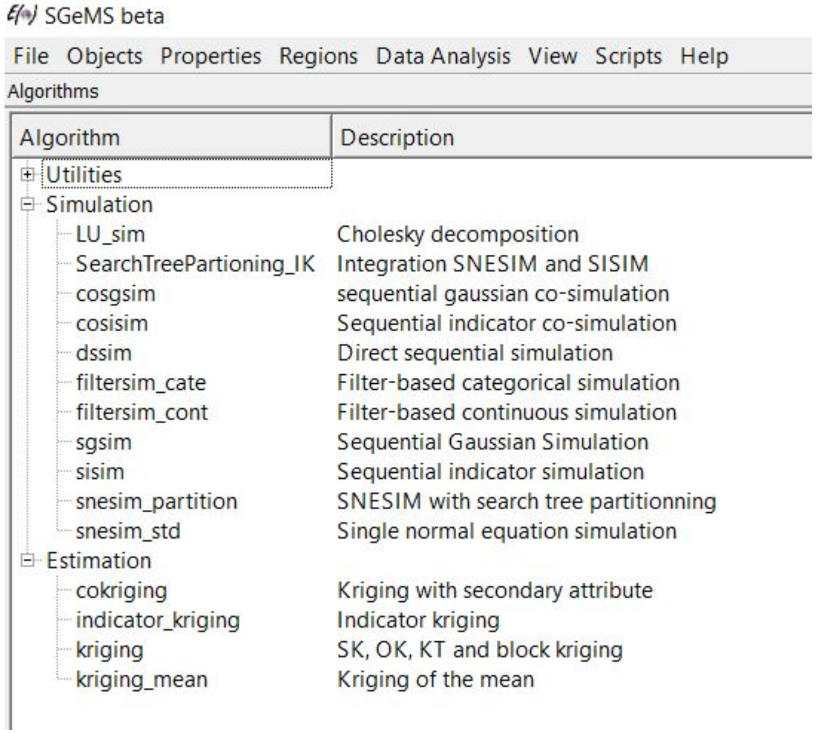
Finite Volume Method with Inhomogenous Parameters for Realistic Geological Models based on SPE10

Vladislav Trifonov
Semyon Polyansky
Dmitry Belousov

Tasks

- Write an API for SGeMS
- Implement FVM with inhomogenous parameters

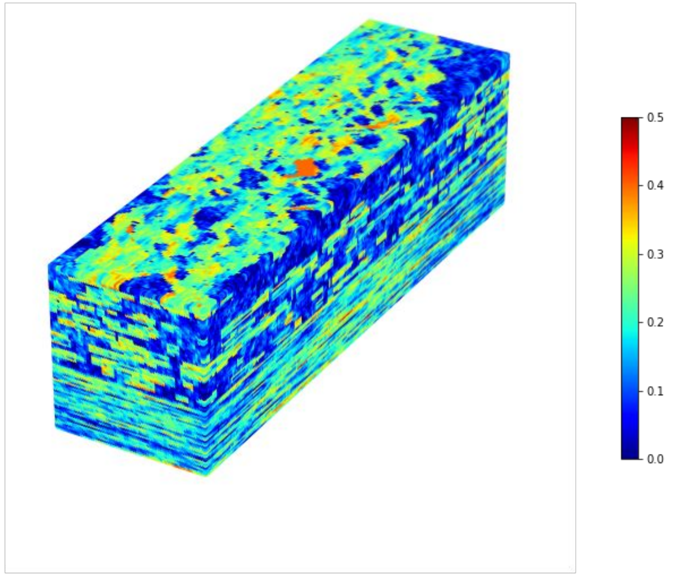
SPE10 model 2 and SGeMS



SGeMS beta

File Objects Properties Regions Data Analysis View Scripts Help
Algorithms

Algorithm	Description
Utilities	
Simulation	
LU_sim	Cholesky decomposition
SearchTreePartioning_IK	Integration SNESIM and SISIM
cosgsim	sequential gaussian co-simulation
cosisim	Sequential indicator co-simulation
dssim	Direct sequential simulation
filtersim_cate	Filter-based categorical simulation
filtersim_cont	Filter-based continuous simulation
sgsim	Sequential Gaussian Simulation
sisim	Sequential indicator simulation
snesim_partition	SNESIM with search tree partitionning
snesim_std	Single normal equation simulation
Estimation	
cokriging	Kriging with secondary attribute
indicator_kriging	Indicator kriging
kriging	SK, OK, KT and block kriging
kriging_mean	Kriging of the mean



- Model: 1200 x 2200 x 170 (ft)

  60 x 220 x 85 (cells)

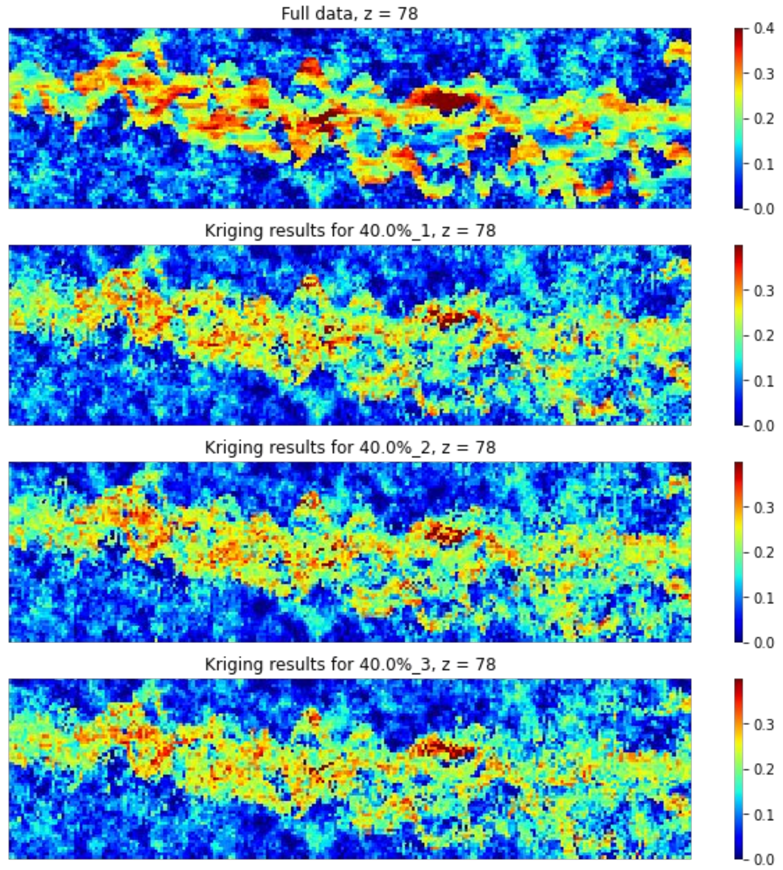
- Cell size: 20 x 10 x 2 (ft)

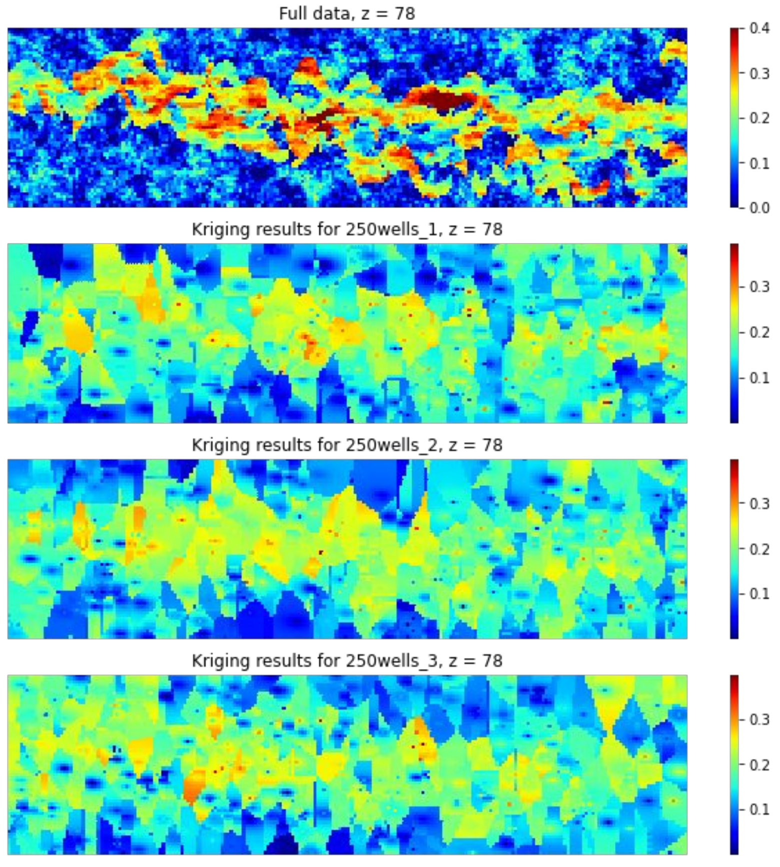
Kriging and Cokriging

- Weighted interpolation
- Variables are normally distributed
- Prior covariance
- Cross covariance for cokriging

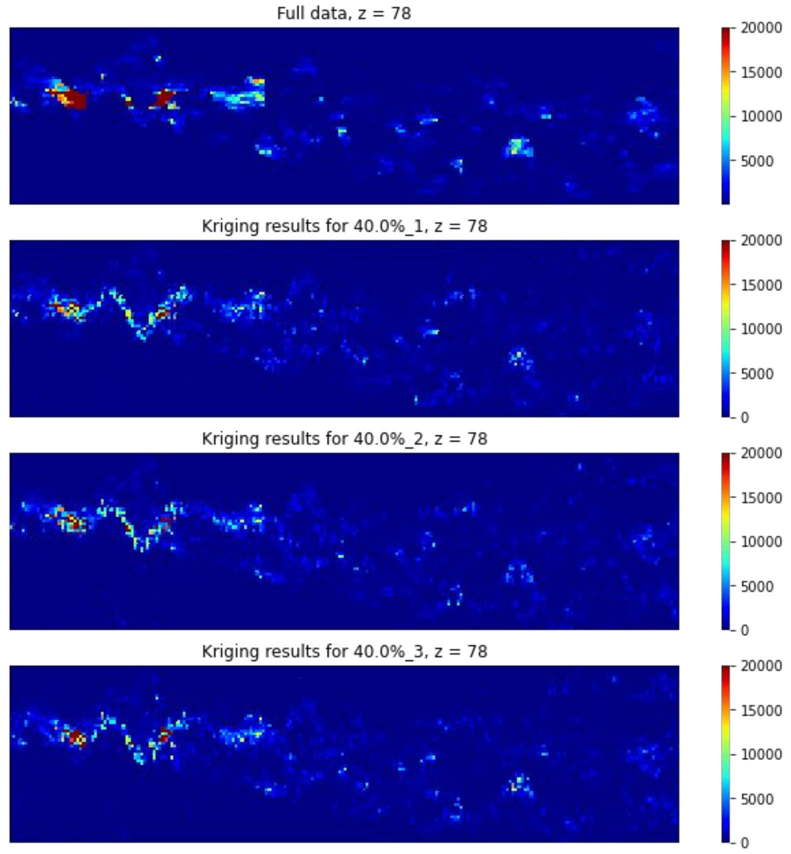
40% of initial data

250 wells of initial data

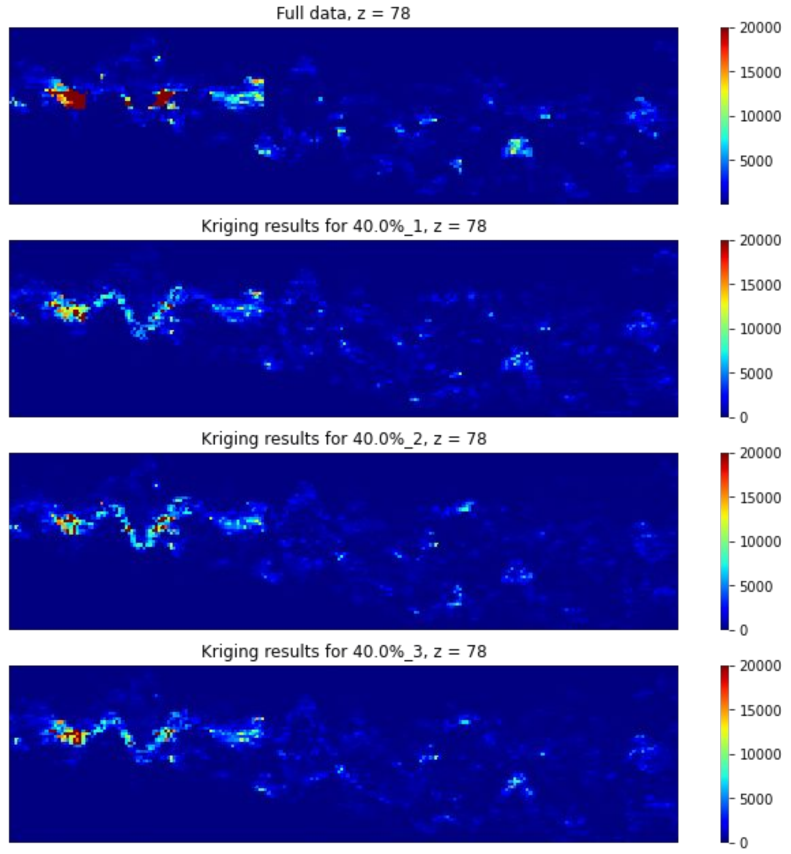




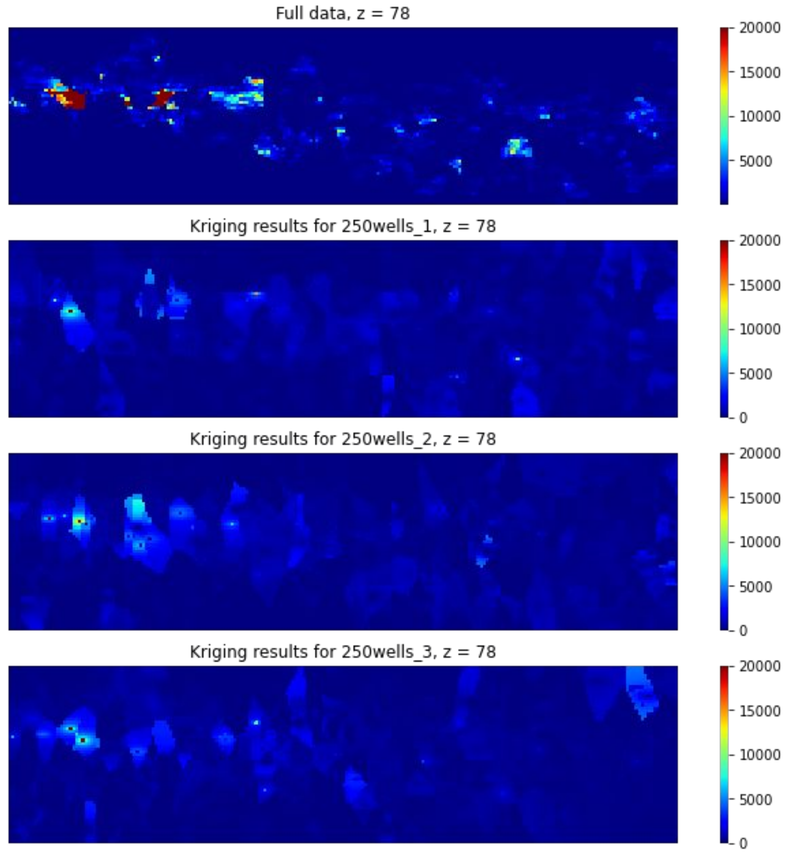
Kriging



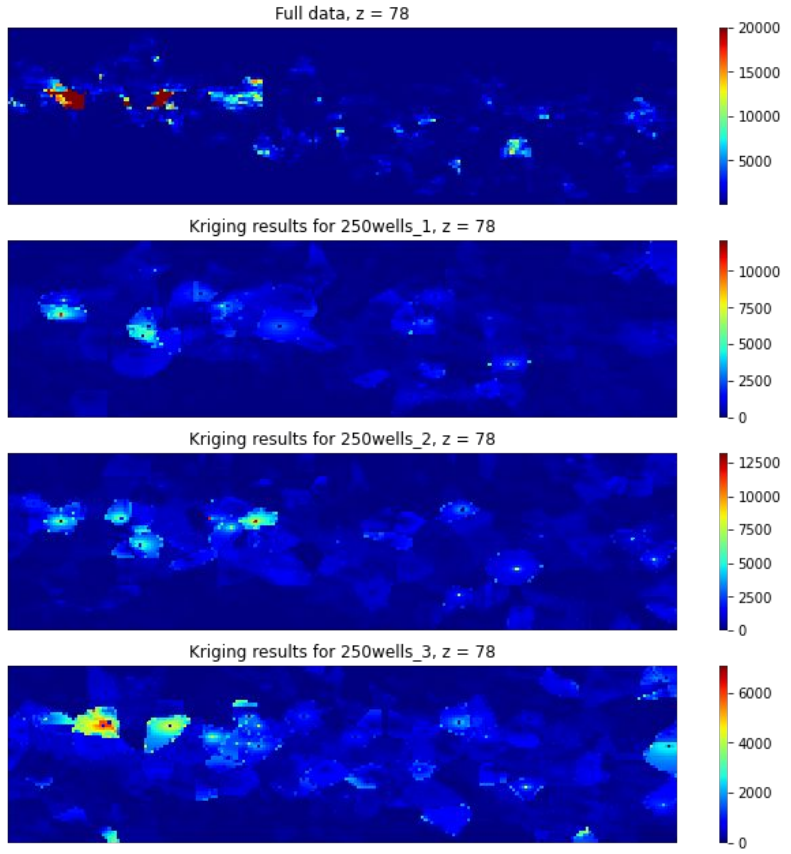
Cokriging



Kriging



Cokriging



```python
from generator import *
input_folder = 'D:/spe10_mod2/input_folder/'
output_folder = 'C:/Users/Vlad/Desktop/geostat_pr/'

gen = Generator(root=input_folder, path_to_save=output_folder, model='spe10m2', alg='ord_kriging')
```

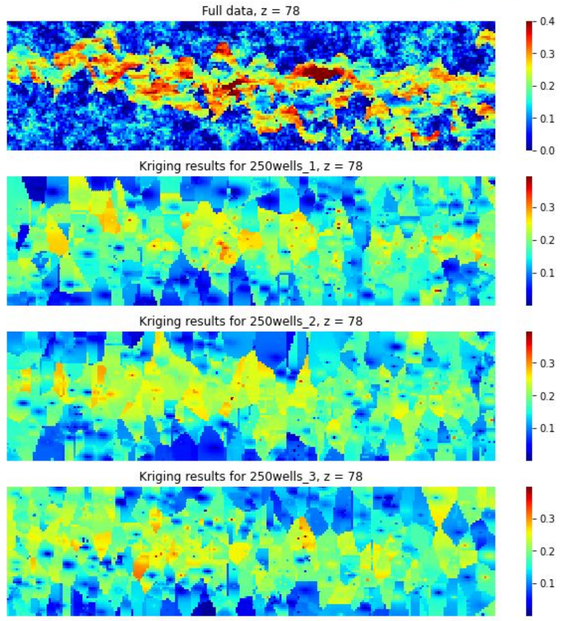
```python
gen(size=3, alpha=.4, search_ellipsoid='800 700 200  0 90 0')
```

•••

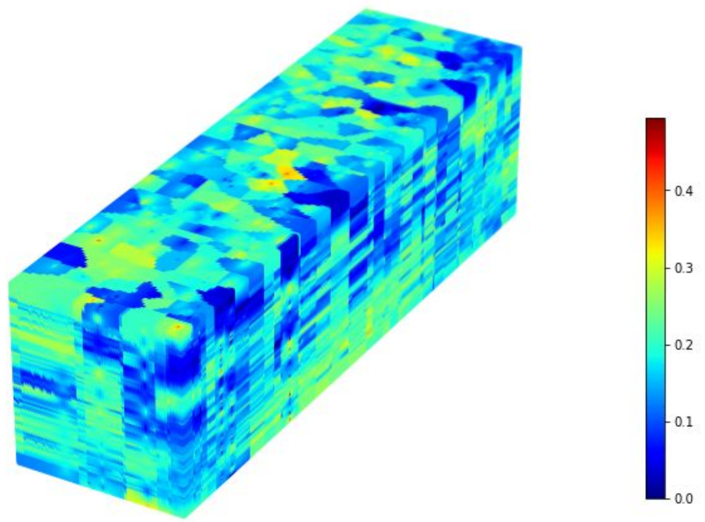
```python
gen(size=3, n_wells=250, search_ellipsoid='800 700 200  0 90 0')
```

•••

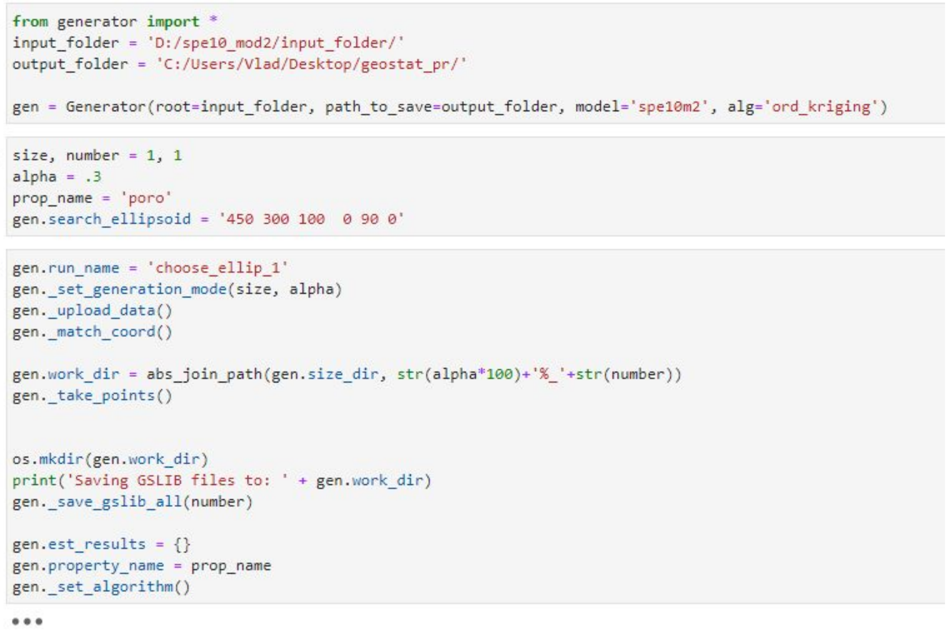
```python
gen.PlotResults(from_npz=True, npz_dir=r'C:\Users\Vlad\Desktop\geostat_pr\2022-05-25_20.17.00')
```



```python
gen.Plot3D(npz_dir=r'C:\Users\Vlad\Desktop\geostat_pr\2022-05-25_20.17.00\250wells_1.npz')
```



Experiment with a search ellipsoid



```
from generator import *
input_folder = 'D:/spe10_mod2/input_folder/'
output_folder = 'C:/Users/Vlad/Desktop/geostat_pr/'

gen = Generator(root=input_folder, path_to_save=output_folder, model='spe10m2', alg='ord_kriging')
```

```
size, number = 1, 1
alpha = .3
prop_name = 'poro'
gen.search_ellipsoid = '450 300 100  0 90 0'
```

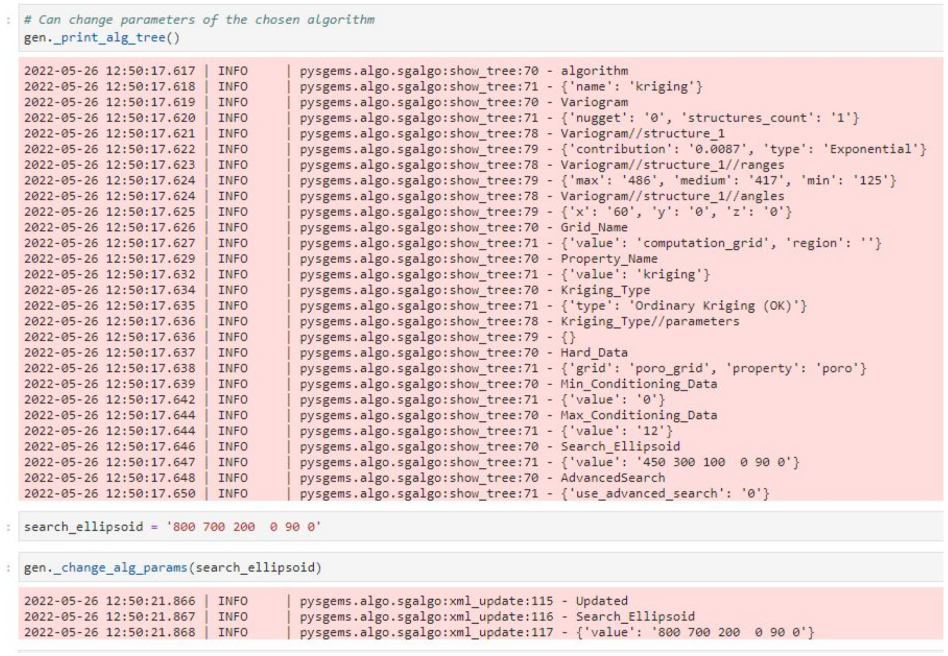
```
gen.run_name = 'choose_ellip_1'
gen._set_generation_mode(size, alpha)
gen._upload_data()
gen._match_coord()

gen.work_dir = abs_join_path(gen.size_dir, str(alpha*100)+'%_'+str(number))
gen._take_points()


os.mkdir(gen.work_dir)
print('Saving GSLIB files to: ' + gen.work_dir)
gen._save_gslib_all(number)

gen.est_results = {}
gen.property_name = prop_name
gen._set_algorithm()
```

•••



```
# Can change parameters of the chosen algorithm
gen._print_alg_tree()
```

```
2022-05-26 12:50:17.617 | INFO     | pysgems.algo.sgalgo:show_tree:70 - algorithm
2022-05-26 12:50:17.618 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'name': 'kriging'}
2022-05-26 12:50:17.619 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Variogram
2022-05-26 12:50:17.620 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'nugget': '0', 'structures_count': '1'}
2022-05-26 12:50:17.621 | INFO     | pysgems.algo.sgalgo:show_tree:78 - Variogram//structure_1
2022-05-26 12:50:17.622 | INFO     | pysgems.algo.sgalgo:show_tree:79 - {'contribution': '0.0087', 'type': 'Exponential'}
2022-05-26 12:50:17.623 | INFO     | pysgems.algo.sgalgo:show_tree:78 - Variogram//structure_1//ranges
2022-05-26 12:50:17.624 | INFO     | pysgems.algo.sgalgo:show_tree:79 - {'max': '486', 'medium': '417', 'min': '125'}
2022-05-26 12:50:17.624 | INFO     | pysgems.algo.sgalgo:show_tree:78 - Variogram//structure_1//angles
2022-05-26 12:50:17.625 | INFO     | pysgems.algo.sgalgo:show_tree:79 - {'x': '60', 'y': '0', 'z': '0'}
2022-05-26 12:50:17.626 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Grid_Name
2022-05-26 12:50:17.627 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'value': 'computation_grid', 'region': ''}
2022-05-26 12:50:17.629 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Property_Name
2022-05-26 12:50:17.632 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'value': 'kriging'}
2022-05-26 12:50:17.634 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Kriging_Type
2022-05-26 12:50:17.635 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'type': 'Ordinary Kriging (OK)'}
2022-05-26 12:50:17.636 | INFO     | pysgems.algo.sgalgo:show_tree:78 - Kriging_Type//parameters
2022-05-26 12:50:17.636 | INFO     | pysgems.algo.sgalgo:show_tree:79 - {}
2022-05-26 12:50:17.637 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Hard_Data
2022-05-26 12:50:17.638 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'grid': 'poro_grid', 'property': 'poro'}
2022-05-26 12:50:17.639 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Min_Conditioning_Data
2022-05-26 12:50:17.642 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'value': '0'}
2022-05-26 12:50:17.644 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Max_Conditioning_Data
2022-05-26 12:50:17.644 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'value': '12'}
2022-05-26 12:50:17.646 | INFO     | pysgems.algo.sgalgo:show_tree:70 - Search_Ellipsoid
2022-05-26 12:50:17.647 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'value': '450 300 100  0 90 0'}
2022-05-26 12:50:17.648 | INFO     | pysgems.algo.sgalgo:show_tree:70 - AdvancedSearch
2022-05-26 12:50:17.650 | INFO     | pysgems.algo.sgalgo:show_tree:71 - {'use_advanced_search': '0'}
```

```
search_ellipsoid = '800 700 200  0 90 0'
```

```
gen._change_alg_params(search_ellipsoid)
```

```
2022-05-26 12:50:21.866 | INFO     | pysgems.algo.sgalgo:xml_update:115 - Updated
2022-05-26 12:50:21.867 | INFO     | pysgems.algo.sgalgo:xml_update:116 - Search_Ellipsoid
2022-05-26 12:50:21.868 | INFO     | pysgems.algo.sgalgo:xml_update:117 - {'value': '800 700 200  0 90 0'}
```

General one-dimensional treatment of nonoscillatory central differencing finite volume scheme

Consider, initially, a one-dimensional problem expressible as a conservation law in vector form: $u_t + f(u)_x = 0$

Upon transferring to integer-centered cell averages in the spatial direction, obtain for a characteristic time step: $\bar{u}(x, t+\Delta t) = \bar{u}(x,t) - \frac{1}{\Delta x}\int_t^{t+\Delta t}\left[f\left(u\left(x+\frac{\Delta x}{2},\tau\right)\right) - f\left(u\left(x-\frac{\Delta x}{2},\tau\right)\right)\right]\mathrm{d}\tau$

Assuming that the solution is expressible as a certain polynomial supported on the integer-centered cells (of piecewise order n-1 for n-th order of convergence), sample on the staggered centres (i.e. cell breakpoints) to obtain our final expression:

$$\bar{w}^{n+1}_{j+\frac{1}{2}} = \frac{1}{\Delta x}\int_{I_{j+\frac{1}{2}}} w(x,t^n)\,\mathrm{d}x - \frac{1}{\Delta x}\left[\int_{t^n}^{t^{n+1}} f(w(x_{j+1},t))\,\mathrm{d}t - \int_{t^n}^{t^{n+1}} f(w(x_j,t))\,\mathrm{d}t\right]$$

General one-dimensional treatment of nonoscillatory central differencing finite volume scheme

The rest of the method consists in finding the quadraturae employed above.

We state for the sake of completeness the second-order piecewise linear polynomial's constituent term and note that higher order is trivially obtainable by considering an appropriate Taylor expansion: $p_j(x) = \bar{w}_j^n + w'_j\left(\frac{x - x_j}{\Delta x}\right)$

The principal demand we place upon the reconstruction of point values and derivatives of *w* in the above is (aside from, naturally, preservation of cell averages) the demand of non-oscillatory nature. A natural and intuitive demand of such kind is the non-increasing behavior of the function's supremum, which naturally translates into the satisfaction by our polynomial the condition imposed by the Weierstrass theorem, thereby eliminating the spurious oscillation: $\sup_x \mid \sum_j p_j(x)\chi_j(x) \mid \leqslant \sup_x \mid \sum_j \bar{w}_j^n \chi_j(x) \mid$

General one-dimensional treatment of nonoscillatory central differencing finite volume scheme

A second-degree accurate method, proposed by the source paper, employs the van Leer MinMod limiter:

$$w'_j = \mathrm{MinMod}(\alpha\Delta_+\bar{w}_j, \Delta_0\bar{w}_j, \alpha\Delta_-\bar{w}_j), \quad 1 \leqslant \alpha < 4.$$

With $\Delta_\pm w_j = \pm(w_{j\pm1} - w_j)$, and $\Delta_0 = \frac{1}{2}(\Delta_+ + \Delta_-)$. and $\mathrm{MinMod}(a,b,c)=\mathrm{sign}(a)\min(|a|,|b|,|c|)$

It ought to be noted that the MinMod assumes a zero value wherever the signs are distinct and that this particular reconstruction enjoys an additional TVD (total-variation diminishing) property, meaning that it preserves the monotonicity of functions, which is an upgrade if we were to consider shocks and rarefaction waves (spoiler: we don't)

It is also noted that we are at liberty to choose the parameter alpha, which we mostly keep at 1.4-1.5 as is customary.

General one-dimensional treatment of nonoscillatory central differencing finite volume scheme

Having produced the reconstruction outlined above (implicitly, since we never write the polynomials down explicitly: we evaluate only specific parts of them), we obtained the following for the first integral in the evolution equation:

$$\frac{1}{\Delta x}\int_{I_{j+\frac{1}{2}}} w(x, t^n)\,\mathrm{d}x = \frac{1}{\Delta x}\left[\int_{x_j}^{x_{j+\frac{1}{2}}} p_j(x)\,\mathrm{d}x + \int_{x_{j+\frac{1}{2}}}^{x_{j+1}} p_{j+1}(x)\,\mathrm{d}x\right] = \frac{1}{2}[\bar{w}_j^n + \bar{w}_{j+1}^n] + \frac{1}{8}[w_j' - w_{j+1}'].$$

Now considering the integral with flux terms, we can see that it is integrable in numerical quadraturae to arbitrary accuracy provided the insides of cells have no discontinuities of the target fields. For simplicity, take the midpoint rule $\int_{t^n}^{t^{n+1}} f(w(x_j, \tau))\,\mathrm{d}\tau \approx \Delta t f(w_j^{n+\frac{1}{2}}),$.

And obtain the following predictor with lambda a mesh ratio: $w_j^{n+\frac{1}{2}} = \bar{w}_j^n - \frac{\lambda}{2} f_j', \quad f_j = f(w_j^n)$

In this case we are well able to avoid evaluation of the Jacobian: $f_j' = \mathrm{MinMod}(\alpha\Delta_+ f_j, \Delta_0 f_j, \alpha\Delta_- f_j), \quad f_j = f(w_j^n).$

General one-dimensional treatment of nonoscillatory central differencing finite volume scheme

Then:

$$\int_{t^n}^{t^{n+1}} f(w(x_j,\tau))\,\mathrm{d}\tau \approx \Delta t f\left(w_j^{n+\frac{1}{2}}\right) =: \Delta t f_j^{n+\frac{1}{2}},$$

And we also have corrector:

$$\bar{w}_{j+\frac{1}{2}}^{n+1} = \frac{1}{2}[\bar{w}_j^n + \bar{w}_{j+1}^n] + \frac{1}{8}[w'_j - w'_{j+1}] - \lambda\left[f_{j+1}^{n+\frac{1}{2}} - f_j^{n+\frac{1}{2}}\right]$$

General two-dimensional treatment of nonoscillatory central differencing finite volume scheme

Now consider a two-dimensional problem of same kind: $u_t + f(u)_x + g(u)_z = 0.$

Similarly, take cell averages:

$$\bar{u}_t(x,z,t) + \frac{1}{\Delta x \Delta z}\int_{z-\frac{\Delta z}{2}}^{z+\frac{\Delta z}{2}} \left[f\left(u\left(x+\frac{\Delta x}{2},\eta,t\right)\right) - f\left(u\left(x-\frac{\Delta x}{2},\eta,t\right)\right)\right] \mathrm{d}\eta$$
$$+ \frac{1}{\Delta x \Delta z}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} \left[g\left(u\left(\xi,z+\frac{\Delta z}{2},t\right)\right) - g\left(u\left(\xi,z-\frac{\Delta z}{2},t\right)\right)\right] \mathrm{d}\xi = 0.$$

Proceeding analogously to what we did before except with upscaling of everything in dimension, we obtain the following general evolution relation which we now set to evaluate:

$$\bar{w}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} = \bar{w}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} - \frac{1}{\Delta x \Delta z}\int_{t^n}^{t^{n+1}}\int_{z_k}^{z_{k+1}} [f(w(x_{j+1},z,t)) - f(w(x_j,z,t))]\,\mathrm{d}z\,\mathrm{d}t$$
$$- \frac{1}{\Delta x \Delta z}\int_{t^n}^{t^{n+1}}\int_{x_j}^{x_{j+1}} [g(w(x,z_{k+1},t)) - g(w(x,z_k,t))]\,\mathrm{d}x\,\mathrm{d}t.$$

General two-dimensional treatment of nonoscillatory central differencing finite volume scheme

How do we approximate with polynomials in 2D? We elect to follow the multilinear approach (in our case bilinear): we simply interpolate in 1D along each spatial axis: $p_{jk}(x,z) = \bar{w}^n_{jk} + w'_{jk}\left(\frac{x-x_j}{\Delta x}\right) + \grave{w}_{jk}\left(\frac{z-z_k}{\Delta z}\right)$

The right-slanted primes are a feature and not a bug: they denote z-derivative as opposed to x-derivative:

$$w'_{jk} = \mathrm{MinMod}(\alpha\Delta_{+x}\bar{w}^n_{jk}, \Delta_{0x}\bar{w}^n_{jk}, \alpha\Delta_{-x}\bar{w}^n_{jk})$$

$$\grave{w}_{jk} = \mathrm{MinMod}(\alpha\Delta_{+z}\bar{w}^n_{jk}, \Delta_{0z}\bar{w}^n_{jk}, \alpha\Delta_{-z}\bar{w}^n_{jk}).$$

Equipped with this, we can now evaluate the first (average) term in our formula:

$$\begin{aligned}
\bar{w}^n_{j+\frac{1}{2}k+\frac{1}{2}} &= \frac{1}{\Delta x\Delta z}\int_{x_j}^{x_{j+\frac{1}{2}}}\int_{z_k}^{z_{k+\frac{1}{2}}} p_{jk}(x,z)\,\mathrm{d}z\,\mathrm{d}x + \int_{x_{j+\frac{1}{2}}}^{x_{j+1}}\int_{z_k}^{z_{k+\frac{1}{2}}} p_{j+1,k}(x,z)\,\mathrm{d}z\,\mathrm{d}x \\
&\quad + \frac{1}{\Delta x\Delta z}\int_{x_j}^{x_{j+\frac{1}{2}}}\int_{z_{k+\frac{1}{2}}}^{z_{k+1}} p_{j,k+1}(x,z)\,\mathrm{d}z\,\mathrm{d}x + \int_{x_{j+\frac{1}{2}}}^{x_{j+1}}\int_{z_{k+\frac{1}{2}}}^{z_{k+1}} p_{j+1,k+1}(x,z)\,\mathrm{d}z\,\mathrm{d}x \\
&= \frac{1}{4}\left(\bar{w}^n_{jk} + \bar{w}^n_{j+1,k} + \bar{w}^n_{j,k+1} + \bar{w}^n_{j+1,k+1}\right) \\
&\quad + \frac{1}{16}\left[\left(w'_{jk} - w'_{j+1,k}\right) + \left(w'_{j,k+1} - w'_{j+1,k+1}\right) + \left(\grave{w}_{jk} - \grave{w}_{j,k+1}\right) + \left(\grave{w}_{j+1,k} - \grave{w}_{j+1,k+1}\right)\right]
\end{aligned}$$

General two-dimensional treatment of nonoscillatory central differencing finite volume scheme

Then, like before, we do the predictor step with lambda and mu being mesh ratios (of temporal to spatial step): $w_{jk}^{n+\frac{1}{2}} = \bar{w}_{jk}^{n} - \frac{\lambda}{2} f'_{jk} - \frac{\mu}{2} g'_{jk}$

Employ a direct Jacobian-free approximation:

$$f'_{jk} = \mathrm{MinMod}(\alpha \Delta_{+x} f_{jk}, \Delta_{0x} f_{jk}, \alpha \Delta_{-x} f_{jk}),$$
$$g'_{jk} = \mathrm{MinMod}(\alpha \Delta_{+z} g_{jk}, \Delta_{0z} g_{jk}, \alpha \Delta_{-z} g_{jk}).$$

And approximate the flux integrals:

$$\int_{t^n}^{t^{n+1}} \int_{z_k}^{z_{k+1}} f(w(x_{j+1}, z, t))\, \mathrm{d}z\, \mathrm{d}t \sim \frac{\Delta z \Delta t}{2} \left[f(w_{j+1,k}^{n+\frac{1}{2}}) + f(w_{j+1,k+1}^{n+\frac{1}{2}}) \right].$$

$$\int_{t^n}^{t^{n+1}} \int_{x_j}^{x_{j+1}} g(w(x, z_{k+1}, t))\, \mathrm{d}x\, \mathrm{d}t \sim \frac{\Delta x \Delta t}{2} \left[g(w_{j+1,k+1}^{n+\frac{1}{2}}) + g(w_{j,k+1}^{n+\frac{1}{2}}) \right].$$

General two-dimensional treatment of nonoscillatory central differencing finite volume scheme

In the end, an evolution formula is obtained!

$$
\begin{aligned}
\bar{w}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} = {} & \frac{1}{4}\left(\bar{w}^{n}_{jk} + \bar{w}^{n}_{j+1,k} + \bar{w}^{n}_{j,k+1} + \bar{w}^{n}_{j+1,k+1}\right) + \frac{1}{16}\left(w'_{jk} - w'_{j+1,k}\right) - \frac{\lambda}{2}\left[f\left(w^{n+\frac{1}{2}}_{j+1,k}\right) - f\left(w^{n+\frac{1}{2}}_{jk}\right)\right] \\
& + \frac{1}{16}\left(w'_{j,k+1} - w'_{j+1,k+1}\right) - \frac{\lambda}{2}\left[f\left(w^{n+\frac{1}{2}}_{j+1,k+1}\right) - f\left(w^{n+\frac{1}{2}}_{j,k+1}\right)\right] + \frac{1}{16}\left(w^{\backprime}_{jk} - w^{\backprime}_{j,k+1}\right) \\
& - \frac{\mu}{2}\left[g\left(w^{n+\frac{1}{2}}_{j,k+1}\right) - g\left(w^{n+\frac{1}{2}}_{jk}\right)\right] + \frac{1}{16}\left(w^{\backprime}_{j+1,k} - w^{\backprime}_{j+1,k+1}\right) - \frac{\mu}{2}\left[g\left(w^{n+\frac{1}{2}}_{j+1,k+1}\right) - g\left(w^{n+\frac{1}{2}}_{j+1,k}\right)\right].
\end{aligned}
$$

On preservation of dimensionality and stagger recovery

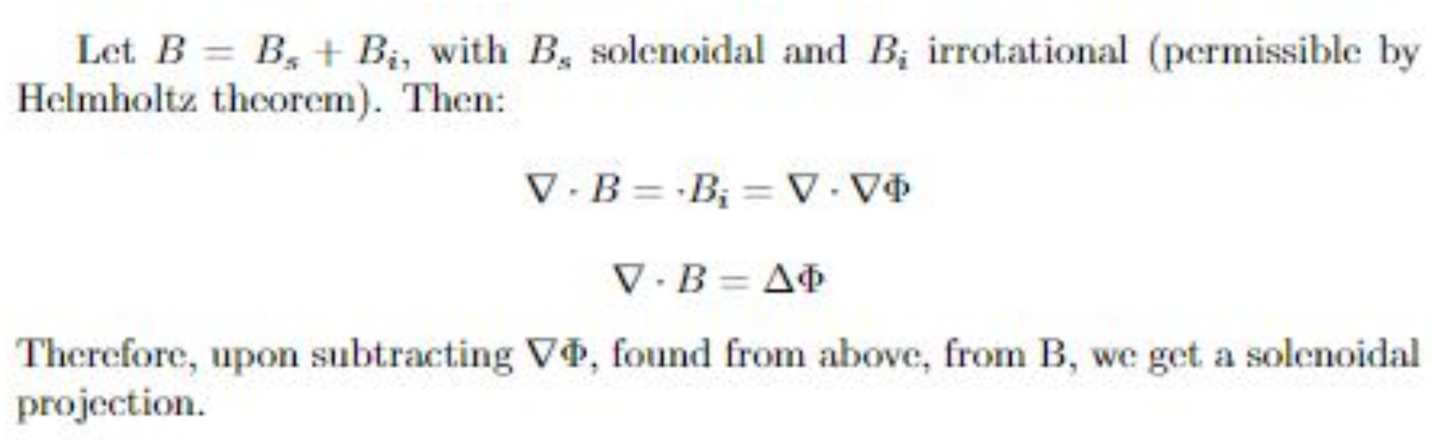
It is observed readily that upon transition to cells from endpoints the vector dimensionality diminishes by 1. Additionally, the evolution step yields the forward-staggered by a halfstep values. They need to be brought under control. One option is to explicitly reconstruct the polynomials and evaluate them at endpoints of staggered cells, incurring further computational costs. However, it is more attractive to alternative forward- and backstaggering by means of alternating padding from the right and the left endpoints. In our code the alternating padding is employed implicitly by using modified formulae for each of the cases.

On solenoidal projection

The density field obtained by staggered evolution is not necessarily solenoidal, which may be undesirable if we e.g. deal with incompressible fluids. Additionally, if we wish to make our solver capable of treating magnetic fields, this problem is to be solved.

A so-called Leray projection is then in order at the end of each step. It can be done, of course, by projecting onto a basis of solenoidal functions, e.g. solenoidal RBFs (Daniel A. Cervantes Cabrera, Pedro Gonzalez-Casanova, Christian Gout, L. Juárez Héctor, Rafael Resendiz. Vector field approximation using radial basis functions). However, I employed a different idea.

My idea (formulated for B)



Let $B = B_s + B_i$, with B_s solenoidal and B_i irrotational (permissible by Helmholtz theorem). Then:

$$\nabla \cdot B = \cdot B_i = \nabla \cdot \nabla \Phi$$

$$\nabla \cdot B = \Delta \Phi$$

Therefore, upon subtracting $\nabla \Phi$, found from above, from B, we get a solenoidal projection.

Showcases

Kelvin-Helmholtz instability w/ an external magnetic field

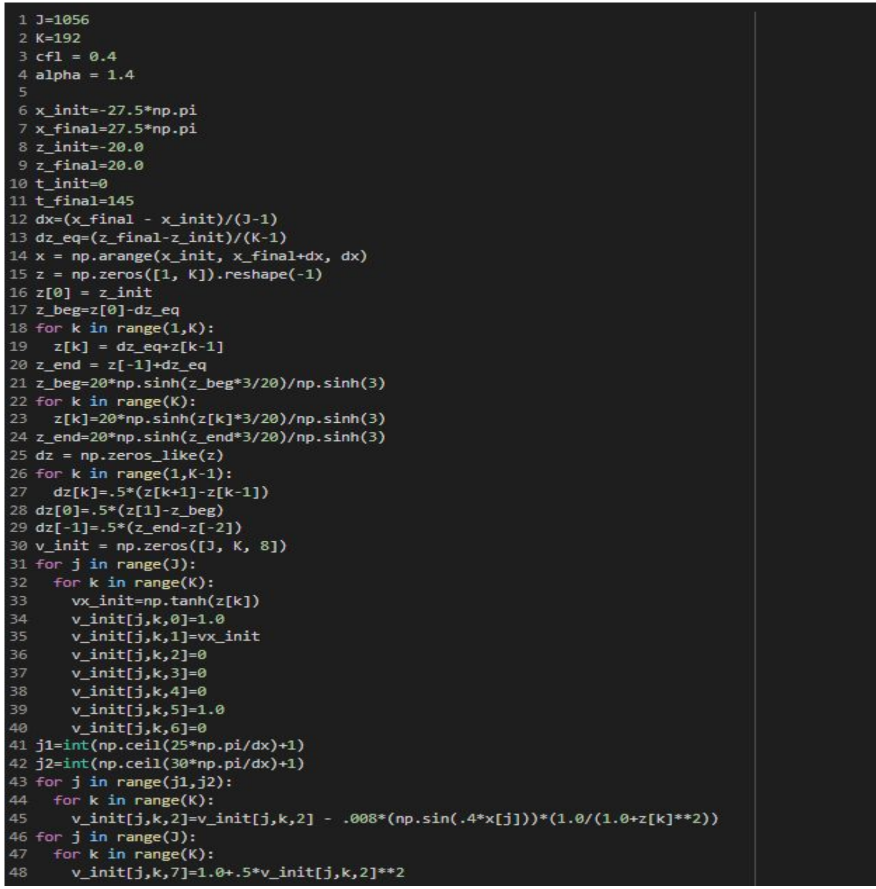
To make points denser @ region of greater influence of small perturbation, employ Roberts transform:

$$z \leftarrow \frac{H \sinh(\tau z/2H)}{\sinh(\tau/2)}, \quad \tau = 6,$$

(stole the idea from G.-S. Jiang, C.C. Wu,

A high-order WENO finite difference scheme

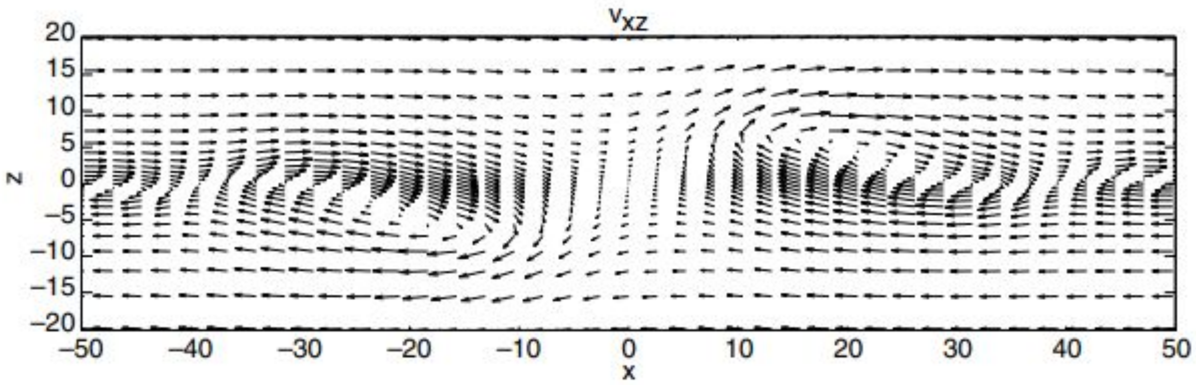
for the equations of ideal magnetohydrodynamics)



```
J=1056
K=192
cfl = 0.4
alpha = 1.4

x_init=-27.5*np.pi
x_final=27.5*np.pi
z_init=-20.0
z_final=20.0
t_init=0
t_final=145
dx=(x_final - x_init)/(J-1)
dz_eq=(z_final-z_init)/(K-1)
x = np.arange(x_init, x_final+dx, dx)
z = np.zeros([1, K]).reshape(-1)
z[0] = z_init
z_beg=z[0]-dz_eq
for k in range(1,K):
  z[k] = dz_eq+z[k-1]
z_end = z[-1]+dz_eq
z_beg=20*np.sinh(z_beg*3/20)/np.sinh(3)
for k in range(K):
  z[k]=20*np.sinh(z[k]*3/20)/np.sinh(3)
z_end=20*np.sinh(z_end*3/20)/np.sinh(3)
dz = np.zeros_like(z)
for k in range(1,K-1):
  dz[k]=.5*(z[k+1]-z[k-1])
dz[0]=.5*(z[1]-z_beg)
dz[-1]=.5*(z_end-z[-2])
v_init = np.zeros([J, K, 8])
for j in range(J):
  for k in range(K):
    vx_init=np.tanh(z[k])
    v_init[j,k,0]=1.0
    v_init[j,k,1]=vx_init
    v_init[j,k,2]=0
    v_init[j,k,3]=0
    v_init[j,k,4]=0
    v_init[j,k,5]=1.0
    v_init[j,k,6]=0
j1=int(np.ceil(25*np.pi/dx)+1)
j2=int(np.ceil(30*np.pi/dx)+1)
for j in range(j1,j2):
  for k in range(K):
    v_init[j,k,2]=v_init[j,k,2] - .008*(np.sin(.4*x[j]))*(1.0/(1.0+z[k]**2))
for j in range(J):
  for k in range(K):
    v_init[j,k,7]=1.0+.5*v_init[j,k,2]**2
```

Showcases



Showcases

Given an initial density distribution as an exponential function of a scaled sum of X,Y and arrays of K, phi, compute the evolution.

Consider the equation of Darcy single-phase flow.

$$\nabla \cdot \left[\frac{\alpha\rho K}{\mu}(\nabla p + \rho g \nabla D)\right] + \alpha q = \alpha \frac{\partial(\phi\rho)}{\partial t}$$

We use the following simplifications: thickness is uniform, depth is uniform, fluid viscosity and compressibility are unity (induces a simple multiplicative scaling factor that is not very important qualitatively)

Showcases

We get (in absence of wells): $\nabla \cdot [\frac{K}{\mu c}\nabla\rho] = \frac{\partial\phi\rho}{\partial t}$

And transform into a conservation law:

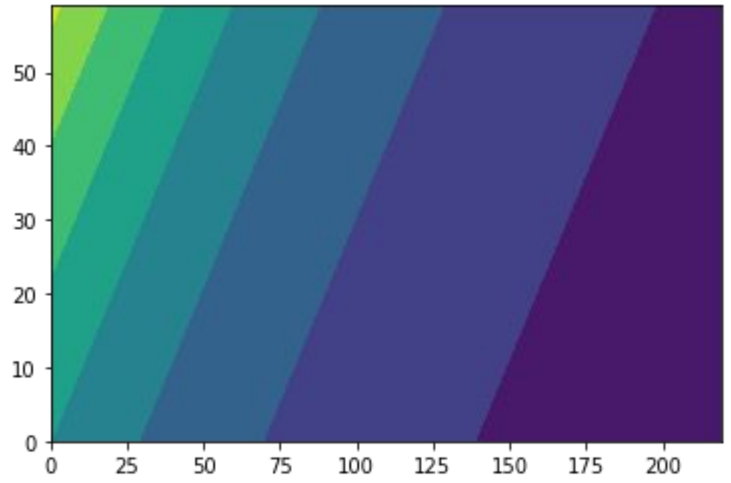
$$u + f(u)_x + g(u)_y = 0$$

$$u = (\rho, r, p)$$

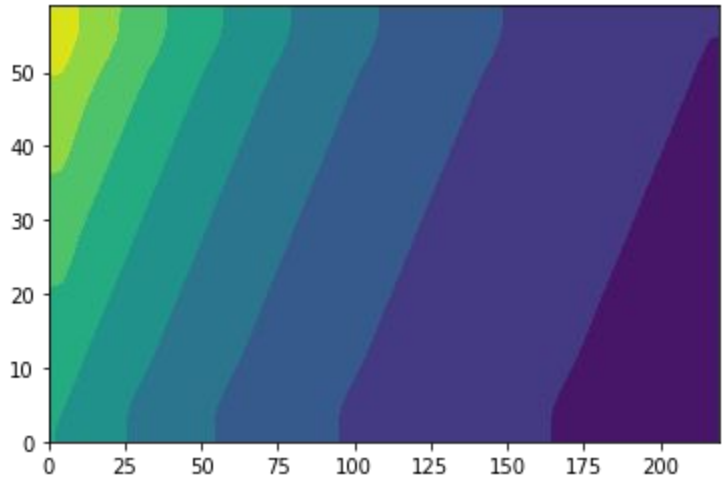
$$-f = (\frac{K_{xx}r}{\phi}, \rho, 0)$$

$$-g = (\frac{K_{yy}r}{\phi}, 0, \rho)$$

Showcases

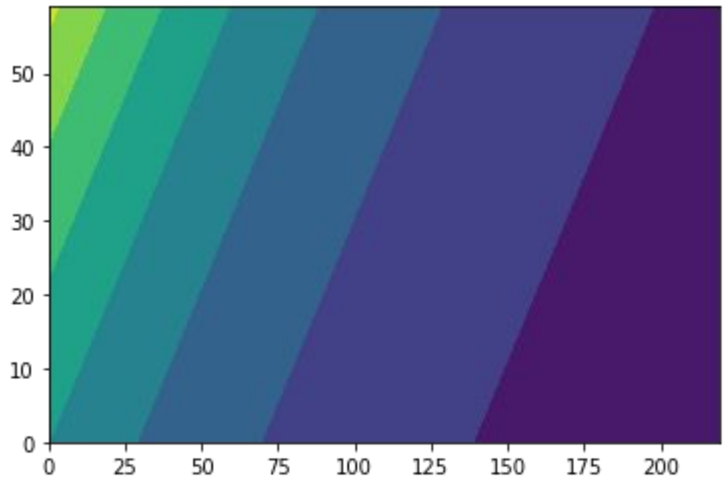


0
10
20
30
40
50
0
25
50
75
100
125
150
175
200

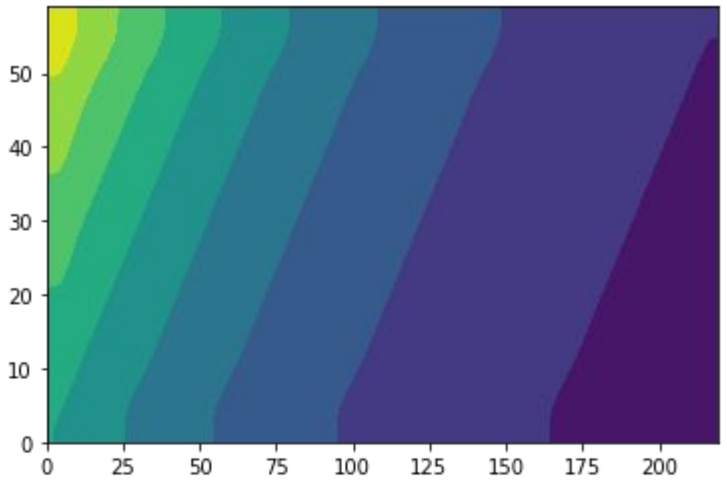


0
10
20
30
40
50
0
25
50
75
100
125
150
175
200

Showcases

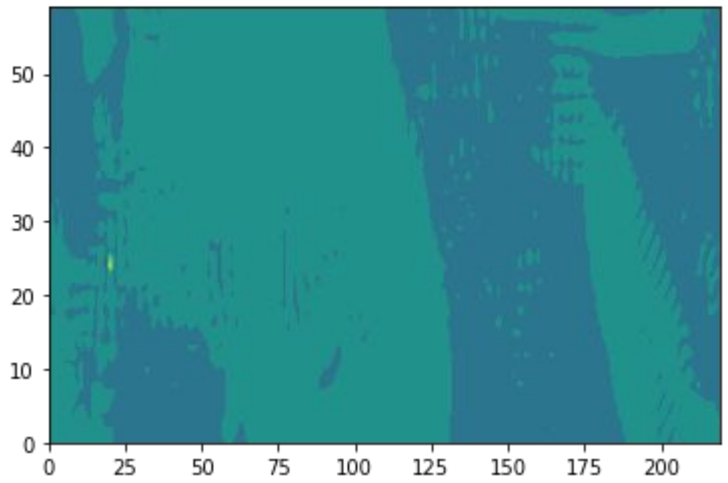


0
10
20
30
40
50
0
25
50
75
100
125
150
175
200



0
10
20
30
40
50
0
25
50
75
100
125
150
175
200

At much greater permeabilities we get



On adding wells

Adding wells is a nontrivial task because they do not easily conform to the conservation law form. In order to add them, one may well introduce such a field Q that the sink-source term q is given as a divergence of Q. This is the only way I see.