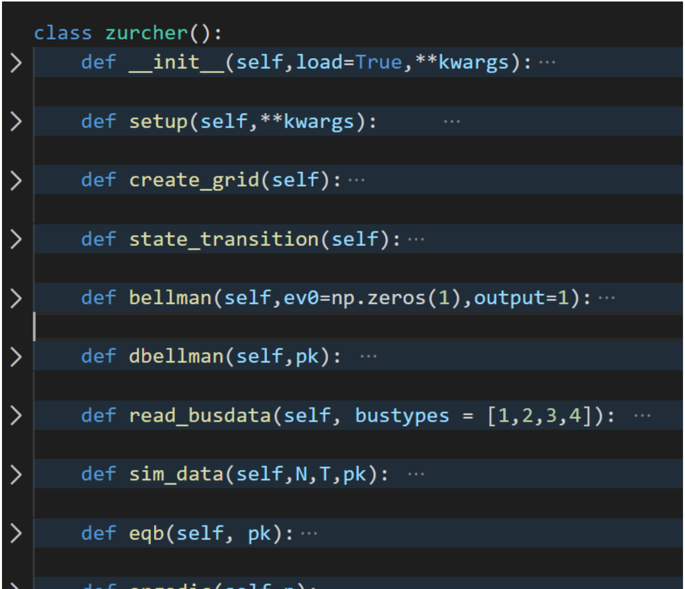
Exercise 3: NFXP

Malene C. Fuglsang

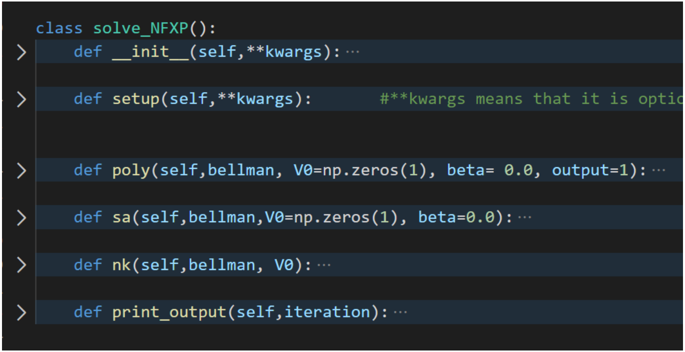
March 23, 2020

Zurcher



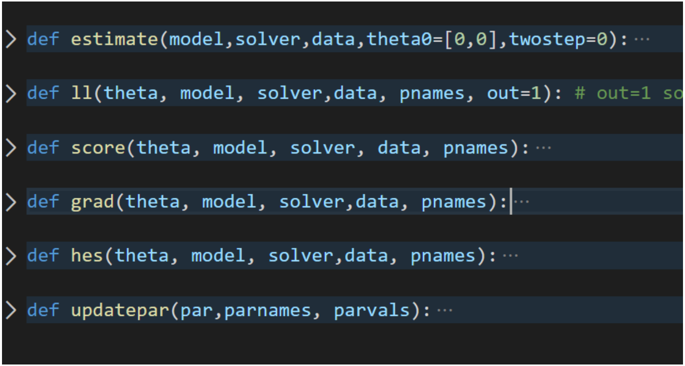
```
class zurcher():
    def __init__(self,load=True,**kwargs):···

    def setup(self,**kwargs):       ···

    def create_grid(self):···

    def state_transition(self):···

    def bellman(self,ev0=np.zeros(1),output=1):···

    def dbellman(self,pk): ···

    def read_busdata(self, bustypes = [1,2,3,4]): ···

    def sim_data(self,N,T,pk): ···

    def eqb(self, pk):···

    def ergodic(self,p):
```

Solve_NFXP



```
class solve_NFXP():
    def __init__(self,**kwargs):···

    def setup(self,**kwargs):       #**kwargs means that it is optic


    def poly(self,bellman, V0=np.zeros(1), beta= 0.0, output=1):···

    def sa(self,bellman,V0=np.zeros(1), beta=0.0):···

    def nk(self,bellman, V0):···

    def print_output(self,iteration):···
```

Estimate



```
def estimate(model,solver,data,theta0=[0,0],twostep=0):···

def ll(theta, model, solver,data, pnames, out=1): # out=1 so

def score(theta, model, solver, data, pnames):···

def grad(theta, model, solver,data, pnames):···

def hes(theta, model, solver,data, pnames):···

def updatepar(par,parnames, parvals):···
```

Solve NFXP

- In order to solve NFXP, you have to shift between SA and NK. SA has the advantage that it will always converge, however it is slow, especially as beta goes towards 1. Therefore you have to shift to NK after some iterations. NK convergence must faster (quadratic convergence), but does not in generally converges globally.
- So in order to solve the model you have to go through the following steps:
  1. Set parameters
  2. Contraction iteration (SA)
  3. Newton-Kantorovich iteration (NK)
  4. Check for convergence
  5. Repeat step 2 and 3 until convergence

Successive Approximations

You run the code: V1,iter_sa= sa(bellman, V0)

- Use V0 as input. Solve the bellman equation, and you get V1
- Use V1 as input. Solve the bellman equation, and you get V2
- You continuous this procedure until one of the following conditions holds:
    1. If the difference between V1-V0 is below the tolerance
    2. If the relative tolerance approaches beta
    3. If the number of iterations exceed the maximum.

    - The first implies convergence,
    - The second needs a shift to N-K,
    - The third needs a shift to N-K, but if you are too far away, the solution will not necessary converge

Newton-Kantorovich

You run the code: V0, _ = nk(bellman, V0)

- Use V0 as input. Solve the bellman equation, and you get V1 and dv
- Use V0, V1 and dv as an input, and use: $V = V0 - (1 - dv)^{-1}(V0 - V1)$
- Do a SA iteration for stability V0 = bellman_sa(V)
- You continuous this procedure until one of the following conditions holds:
  1. If the difference between V-V0 is below the tolerance
  2. If the number of iteration exceed the maximum

Estimate NFXP

In order to estimate the model, the code does the following steps:

1. Update parameters
2. Solve the model
3. Evaluate the likelihood function
4. Check for converges
5. Repeat these steps until convergence

(use information about the gradient and the hessian)