# Code Assignment 2

Maximilian Huber

In [1]:
```
# setup functions
```

Out[1]:
```
E_z (generic function with 1 method)
```

In [2]:
```
# define X, Y, Z, N, n_x, m_y
```

## Coordinate Decent - in parallel

I started up Julia with 4 threads. Threads in Julia do not have dedicated memory!

In [3]:
```
Threads.nthreads()
```

Out[3]:
```
4
```

In [4]:
```
# define coordinate_decent()
function coordinate_decent(;tol::Float64 = 1e-5, max_iter::Int64 = 50, p_init::Vector{Float64} = zeros(N))
    tic()
    iteration = 0
    residual = 1.
    p_old = p_init
    p_new = zeros(N)

    U = zeros(N, N)
    C = zeros(N, N)

    for i in 1:N # passanger/car locations
        for j in 1:N #pickup locations
            U[i, j] = u(X[i], Z[j])
            C[i, j] = c(Y[i], Z[j])
        end
    end

    while (residual > tol) && (iteration < max_iter)
        iteration += 1

        Threads.@threads for i in 1:N
            p_new[i] = find_zero(p_z -> E_z(U, C, vcat(p_old[1:i-1], p_z, p_old[i+1:end]), i), (0., 3.), Order1())
        end

        residual = maximum(abs.(S(p_new, C) .- D(p_new, U)) ./ (S(p_new, C) .+ D(p_new, U)))
        p_old .= p_new
    end

    return (residual, sum(S(p_new, C)),
        (S(p_new, C) * p_new)[1]/sum(S(p_new, C)), iteration, toq(), p_new)
end
```

Out[4]:
```
coordinate_decent (generic function with 1 method)
```

In [5]:
```
(residual, rides, avg_price, iter, t, p) = coordinate_decent()

@show (residual, rides, avg_price, iter, t);
```

```
(residual, rides, avg_price, iter, t) = (6.527430780612634e-6, 30.19670964825541, 2.0414353028463283, 27, 3.981476557)
```

## Matching

Let x be passengers (men, `i` in the slides) and y be cars (women, `j` in the slides).

In [64]:
```
α_xy(x, y) = -1 * convert(Float64, norm(x - y) >= 0.5)
γ_xy(x, y) = -norm(x - y)^2
```

Out[64]:
```
γ_xy (generic function with 1 method)
```

In [107]:
```
# run the iterative procedure↔
```

```
  0.061872 seconds (80.12 k allocations: 2.503 MiB)
```

It took only one real iteration, because of horizontal preferences. Drivers pick up where they are, hence:

- (i) welfare is 0
- (ii) welfare is 0
- (iii) one iteration
- (iv) 0.06 seconds, but I did not even have compiled the function

### Adachi

The Adachi algorithm assumes that all preferences are strict (slide 38, slide set 05).

The results would be the same.

## Tiebreaking

**Unfinished!**

In [108]:
```
max_iter = 100

μA_xy_old = repmat(n_x', N)
μP_xy = zeros(N, N)
μE_xy = zeros(N, N)

μA_xy_new = zeros(N, N)


A = zeros(N, N) # preference of x
Γ = zeros(N, N) # preference of y

for i in 1:N # passanger/car locations
    for j in 1:N #pickup locations
        A[i, j] = α_xy(X[i], Y[j]) + 0.01 * rand()
        Γ[i, j] = γ_xy(X[i], Y[j]) + 0.01 * rand()
    end
end
```

In [110]:
```
done = false
iteration = 0

while !done && (iteration < max_iter)
    iteration += 1

    μP_xy = zeros(N, N)
    μE_xy =zeros(N, N)

    for i in 1:N
        J = collect(1:N)
        while (sum(μP_xy[i, :]) < sum(μA_xy_old[i, :])) && (length(J) != 0)
            # propose the highest available car
            j = J[indmax(A[i, J] .* (μA_xy_old[i, J] .> 0))]
            μP_xy[i, j] = min(μA_xy_old[i, j], n_x[i] - sum(μP_xy[i, :]))
            deleteat!(J, findin(J, j))
        end
    end

    for j in 1:N
        I = collect(1:N)
        while (sum(μE_xy[:, j]) < m_y[j]) && (length(I) != 0)
            # accept the highest available passenger
            i = I[indmax(Γ[I, j] .* (μP_xy[I, j] .> 0))]
            μE_xy[i, j] = min(μP_xy[i, j], m_y[j] - sum(μE_xy[:, j]))
            deleteat!(I, findin(I, i))
        end
    end

    μA_xy_new .= μA_xy_old .- (μP_xy .- μE_xy)
    μA_xy_new[μA_xy_new .< 0] = 0
    done = all(μE_xy .≈ μP_xy)
    μA_xy_old .= μA_xy_new
end
```

In [111]:
```
iteration
```

Out[111]:
```
100
```

Did not converge. Don't know what is wrong!