Please note that this code presumes familiarity with my paper "Supply and Demand in a Two-Sector Matching Model". Any Equation, Definition or Proofs referenced here can be found in that paper.

This notebook produces Figures 3 and 5 in the main text, as well as figure OA.1 in the Online Appendix.

It is also worth noting that I approximate functions as pairs of vectors of equal length. The first is the vector of arguments and the second a vector of values. Denote the first as e and the second as z: then, in my comments, I will refer to the function as e:z.

In [1]:

```
using Distributions
```

In [2]:

```
using StatsFuns
include("vectors.jl")
```

Out[2]:

```
bvnuppercdf (generic function with 1 method)
```

In the next cell I write my own code to compute the conditional distribution functions for the Gaussian Copula. The function computing the values of the (unconitional) copula is based on code for evaluation of the cdf of bivariate normal distribution that was written originally by Alan Genz for Fortran and translated into Julia by Andreas Noack Jensen.

In [3]:

```
#This is the conditional distribution of U given V=v, for the Gaussian Copula
function conditional(a::Number, b::Number, rho::Number)
    if rho==0
        return a
    else
        if a>0 && a<1
        return cdf(Normal(0, 1), (normquant(a)-rho*normquant(b))/sqrt(1-rho^2))
        else
            if a==0
                return 0
            else
            return 1
            end
        end
        end
end
function conditional(a::Array{Float64, 1}, b::Array{Float64, 1}, rho::Number)
    if rho==0
        return a
    else
       if a>0 && a<1
        return cdf(Normal(0, 1), (normquant(a)-rho*normquant(b))/sqrt(1-rho^2))
        else
            if a==0
                return 0
            else
            return 1
            end
        end
        end
end

#This is a function that makes sure that the Gaussian Copula returns values also for ar
guments greater than 1.
function GaussCop(a::Number, b::Number, rho::Number)
    if a >= 1
return b
elseif b >= 1
return a
elseif a==0
return 0
elseif b==0
return 0
else
1-(1-a+1-b-bvnuppercdf(normquant(a), normquant(b), rho))
end
end
```

Out[3]:

```
GaussCop (generic function with 1 method)
```

The next cell defines the functions needed to compute the skill and productivity components of the surplus functions in the CDL specification (see Section 4). Note that "1" corresponds to "M" (manufacturing) in the main body, whereas "2" corresponds to "S" (services).

In [4]:

```
#the next 9 lines introduce all the parameters of the model. The values assigned to the
m are arbitrary at this stage.
#the correct values are assigned later.
cutoff=0.001 #this isthe truncation parameter a
A=1 #this is the shift constant A_i, which is the same across sectors in all computatio
ns
alpha1C, alpha1N, alpha2C, alpha2N=1, 0, 1, 0 #these are the parameters \alpha_{iC}, \a
lpha_{iS}
muC, muN=0, 0 #these are the means of the two basic skills
mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN #these are \mu_M, \mu_S from
the main text
sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5) #these are \sig
ma_M, \sigma_S from the main text
theta=alpha1C*alpha2C+alpha1N*alpha2N #this is parameter \rho from the main text
muZ1, muZ2, sigmaZ1, sigmaZ2=1, 1, 1, 1 # sigmaZ1, sigma Z2 correspond to \alpha_{MF},
 \alpha_{SF}. muZ1=\alpha_{MF} \mu_{MF}
#the remainning lines in this cell define the skill and productivity components of the
 surplus function, as well as their derivatives
m1(a)=normquant(a*(1-2*cutoff)+cutoff)*sigma1+alpha1C*muC+alpha1N*muN
m1prime(a)=(1-2*cutoff)*sigma1./normpdfalt(normquant(a*(1-2*cutoff)+cutoff))
m2(b)=normquant(b*(1-2*cutoff)+cutoff)*((alpha2C^2+alpha2N^2)^(0.5))+(alpha2C*muC+alpha
2N*muN)
m2prime(b)=(1-2*cutoff)*((alpha2C^2+alpha2N^2)^(0.5))./normpdfalt((m2(b)-(alpha2C*muC+a
lpha2N*muN))/((alpha2C^2+alpha2N^2)^(0.5)))
mf1(c)=normquant(c*(1-2*cutoff)+cutoff)*sigmaZ1+muZ1
mf1prime(c)=(1-2*cutoff)*sigmaZ1./normpdfalt(normquant(c*(1-2*cutoff)+cutoff))
mf2(c)=normquant(c*(1-2*cutoff)+cutoff)*sigmaZ2+muZ2
mfprime(c)=(1-2*cutoff)*sigmaZ2./normpdfalt(normquant(c*(1-2*cutoff)+cutoff))
L1(m)=exp(m)
l1(m)=exp(m)
L2(m)=exp(m)
l2(m)=exp(m)
Lf1(m)=exp(m)
lf1(m)=exp(m)
Lf2(m)=exp(m)
lf2(m)=exp(m)
L1(m1(0)), mu1, mu2
```

Out[4]:

```
(0.0454913852476533, 0, 0)
```

The next cell defines all the primitives of the model.

In [5]:

```
test=collect(0:0.00001:1)
R1=0.5   #mass of sector one firms
R2=0.5 #mass of sector two firms. The current version of the code can only handle R1+R2
<=1
factor=5   #precisions parameter
res=0  #reservation wage; normalised to 0 both here and in the paper
grid=10^factor #number of points for which any function is evaluated
mystep=1/grid
C(a, b)=GaussCop(a, b, theta) #copula
Cuvec(a, b)=conditional(b, a, theta)   #derivative of copula wrt to v_M
Cvvec(a, b)=conditional(a, b, theta)    #derivative of copula wrt to v_S
pi1vec(a, c)=L1(m1(a))*Lf1(mf1(c))+A  #surplus function in sector one
pi1uvec(a, c)=l1(m1(a))*m1prime(a)*Lf1(mf1(c))     #derivative of surplus function in s
ector one wrt v_M
pi1hvec(a, c)=L1(m1(a))*lf1(mf1(c))*mf1prime(c)           #derivative of sector one sur
plus wrt h
pi2vec(b, c)=L2(m2(b))*Lf2(mf2(c)) +A  #surplus function in sector two
pi2vvec(b, c)=l2(m2(b))*m2prime(b)*Lf2(mf2(c))     #derivative of surplus function in s
ector two wrt to v_S
pi2hvec(b, c)=L2(m2(b))*lf2(mf2(c))*mf2prime(c)   #derivative of sector two surplus wrt
h
function Heaviside(a)
    heav=0.5+0.5*sign(a)
end
#the remaining lines define the extended functions from the proof of Theorem OA.1
function Cuex(a::Array{Float64, 1}, b::Array{Float64, 1})
    vec=Array(Float64, length(a))
    for i=1:length(a)
        if a[i]>1
            vec[i]=Cuvec(1, b[i])
    else
            vec[i]=Cuvec(a[i], b[i])
    end
    end
    return vec
    end
function Cuex(a::Number, b::Number)
    if a>1
        return Cuvec(1, b)
    else
        return Cuvec(a, b)
    end

    end
function Cvex(a::Number, b::Number)
    if b>1
        return Cvvec(a, 1)
    else
        return Cvvec(a, b)
    end

    end
function pi2vex(b::Number, c::Number)
    if c>1
        return pi2vvec(b, 1)
    else
        return pi2vvec(b, c)
    end
end
```

```
function pi1uex(a::Number, c:: Number)
    if a>1 && c>1
        pi1uvec(1, 1)
        elseif a>1
            pi1uvec(1, c)
    elseif c>1
        pi1uvec(a, 1)
    else pi1uvec(a, c)
end
end
```

Out[5]:

```
pi1uex (generic function with 1 method)
```

The next cell loads a number of functions

In [6]:

```
include("functions.jl") #a number of functions used throughout this file. See the file
 for details
```

Out[6]:

```
makecont (generic function with 2 methods)
```

In [7]:

```
iterations=14
#the first four arrays store the separation function and its inverse
psivec=Array{Float64}(undef,  grid+1, iterations)
vvec=Array{Float64}(undef,  grid+1, iterations)
phivec=Array{Float64}(undef,  grid+1, iterations)
uvec=Array{Float64}(undef,  grid+1, iterations)
# the next two the supply functions
mass2vec=Array{Float64}(undef,  grid+1, iterations)
mass1vec=Array{Float64}(undef,  grid+1, iterations)
#the next two store the wage functions
w1=Array{Float64}(undef, grid+1, iterations)
w2=Array{Float64}(undef, grid+1, iterations)
#the next three store the inverse wage distribution, the composition effect and the wag
e effect respectively
wagerank=Array{Float64}(undef, grid+1, iterations)
compeff=Array{Float64}(undef, grid+1, iterations)
wageeff=Array{Float64}(undef, grid+1, iterations)
#the next two store the inverse distribution function for each sector
wagerank1=Array{Float64}(undef, grid+1, iterations)
wagerank2=Array{Float64}(undef, grid+1, iterations)
#the final line stores all the parameter values
parametersvec=Array{Float64}(undef, 15,  iterations)
```

Out[7]:

```
15×14 Array{Float64,2}:
 2.29568e-316  2.29568e-316  2.74005e-316  …  1.20742e-311  1.27108e-311
 2.29568e-316  2.29568e-316  2.29598e-316     1.20954e-311  1.31564e-311
 2.73972e-316  2.29568e-316  2.29568e-316     1.21803e-311  1.26047e-311
 2.29589e-316  2.73991e-316  2.29568e-316     1.18195e-311  1.31988e-311
 2.29568e-316  2.29594e-316  2.86109e-316     1.23925e-311  1.24986e-311
 2.29568e-316  2.29568e-316  2.7482e-316   …  1.23925e-311  1.32625e-311
 2.73977e-316  2.29568e-316  2.29599e-316     1.23288e-311  1.33474e-311
 2.29568e-316  2.73996e-316  2.29568e-316     1.235e-311    1.35808e-311
 2.29568e-316  2.29595e-316  2.29568e-316     1.22864e-311  1.35808e-311
 2.29568e-316  2.29568e-316  2.74825e-316     1.22439e-311  1.35171e-311
 2.73982e-316  2.29568e-316  2.29568e-316  …  1.24137e-311  1.35383e-311
 2.29591e-316  2.74001e-316  2.29568e-316     1.26471e-311  1.34747e-311
 2.29568e-316  2.29568e-316  2.29568e-316     1.25622e-311  1.34322e-311
 2.29568e-316  2.29568e-316  2.7483e-316      1.26895e-311  1.3602e-311
 2.73987e-316  2.29568e-316  2.29602e-316     1.2732e-311   3.11755e-321
```

The next cell solves the model and produces the inverse wage functions for Figures 3, 5 and OA.1.

In [8]:

```
for n=1:iterations
    #the first two iterations compute the data used to draw Figure 3
     if n==1
global delta=2/3
global alpha1N0=2/3
global alpha1C0, alpha2C0, alpha2N0=sqrt(1-alpha1N0^2), 0, 1
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global muC=0
global muN=muC*sqrt(5)
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
global muZ1, sigmaZ1, muZ2, sigmaZ2=0, 0.4, 0, 0.4
        end
    if n==2
global alpha1N0=0.99
global alpha1C0=sqrt(1-alpha1N0^2)
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
    end
#The next three iterations produce the data used to draw Figure 5 (a)
     if n==3
global delta=0.45
global alpha1N0=2/3
global alpha1C0, alpha2C0, alpha2N0=sqrt(1-alpha1N0^2), 0, 1
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global muC=0.05
global muN=muC*sqrt(5)
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
global muZ1, sigmaZ1, muZ2, sigmaZ2=0, 1.2, 0, 1.2
    end
    if n==4
global theta=(alpha1C0*0.1+alpha1N0*alpha2N0)/((((alpha1C0^2+alpha1N0^2)^(0.5)))*(((0.1
)^2+alpha2N0^2)^(0.5)))
    end
    if n==5
      global  alpha2C0=0.1
    global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alp
ha2C0, delta*alpha2N0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
    end
#The next three iterations produce the data used to draw Figure 5 (b)
    if n==6
    global delta=0.55
global alpha1N0=0
global alpha1C0=1
global alpha2C0=99999/100000
global alpha2N0=sqrt(1-alpha2C0^2)
global alpha1N, alpha1C, alpha2N, alpha2C=delta*alpha1N0, delta*alpha1C0, delta*alpha2N
0, delta*alpha2C0
global muN=0
```

```
global muC=sqrt((1+alpha2C0)/(1-alpha2C0))*muN
global mu1, mu2= alpha1N*muN+alpha1C*muC, alpha2N*muN+alpha2C*muC
global sigma1, sigma2=(alpha1N^2+alpha1C^2)^(0.5), (alpha2N^2+alpha2C^2)^(0.5)
global theta=(alpha1N*alpha2N+alpha1C*alpha2C)/(sigma1*sigma2)
global muZ1, sigmaZ1, muZ2, sigmaZ2=0 ,0.2 ,0 ,0.2
    end
    if n==7
      global  x=sqrt(1-alpha2C0^2)+0.1
      global   theta=(alpha1N0*x+alpha1C0*alpha2C0)/(((alpha1N0^2+alpha1C0^2)^(0.5))*((
x^2+alpha2C0^2)^(0.5)))
    end
   if n==8
     global  alpha2N0=x
global alpha1N, alpha1C, alpha2N, alpha2C=delta*alpha1N0, delta*alpha1C0, delta*alpha2N
0, delta*alpha2C0
global mu1, mu2= alpha1N*muN+alpha1C*muC, alpha2N*muN+alpha2C*muC
global sigma1, sigma2=(alpha1N^2+alpha1C^2)^(0.5), (alpha2N^2+alpha2C^2)^(0.5)
global theta=(alpha1N*alpha2N+alpha1C*alpha2C)/(sigma1*sigma2)
        end
#The next three iterations produce the data used to draw Figure 5 (c)
     if n==9
global delta=1
global alpha2C0=0
global alpha1N0=9.5/10
global alpha2N0=1
global alpha1C0=sqrt(1+alpha2C0^2-alpha1N0^2)
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global muC=0
global muN=0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
global muZ1, sigmaZ1, muZ2, sigmaZ2=0, 0.4, 0, 0.4
    end
    if n==10
      global  alpha2C0=0.3
global alpha1C0=sqrt(1+alpha2C0^2-alpha1N0^2)
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
    end
   if n==11
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
    end
#The final three iterations produce the data used to draw Figure OA.1
    if n==12
   global delta=1
global delta2=4
global alpha2C0=0*delta2
global alpha2N0=1*delta2
global alpha1N0=(2/3)
global alpha1C0=sqrt(1-alpha1N0^2)
global alpha1C, alpha1N, alpha2C, alpha2N=delta*alpha1C0, delta*alpha1N0, delta*alpha2C
0, delta*alpha2N0
global muC=0
global muN=0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2)
```

```
global muZ2, sigmaZ2=-5.6861914063172945, 2
global muZ1, sigmaZ1=muZ2+sqrt(1-alpha1N0^2)*7+1, sigmaZ2
    end
    if n==13
     global alpha2C=0.1
     global alpha1N, alpha1C, alpha2N=delta*alpha1N0, delta*alpha1C0, delta*alpha2N0
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(((alpha1C^2+alpha1N^2)^(0.5))*((alpha2C
^2+alpha2N^2)^(0.5)))
    end
    if n==14
      global alpha2C=0.1
     global alpha1N, alpha1C, alpha2N=delta*alpha1N0, delta*alpha1C0, delta*alpha2N0
global mu1, mu2= alpha1C*muC+alpha1N*muN, alpha2C*muC+alpha2N*muN
global sigma1, sigma2=(alpha1C^2+alpha1N^2)^(0.5), (alpha2C^2+alpha2N^2)^(0.5)
global theta=(alpha1C*alpha2C+alpha1N*alpha2N)/(sigma1*sigma2) #a parameter determining
skill interdependence. theta=0 implies independence
    end
    #This following line sets the starting values of vc and uc, depending on whether jo
bs are strictly or weakly scarce
    vcmax, vcdash, vcstart, ucstart=1, -1, 0, 0
    print("iteration=$n ") #This reports the iteration computed, to track progress
    #the next line uses the function stablematch that is defined in to compute the equi
librium
    psi, v, mass2= equilibriumbase(C, Cuex, Cvex, pi1uex, pi2vex, R2, R1, grid, 100, 10
0, vcstart, ucstart, vcmax, vcdash)
    #The line above calls the function stablematch from twosectormatchingfunctions11.j
l, which returns the functions psi and mass2
    #The next handful of lines uses the calculated information to compute the inverse s
eparation function and then stores all the functions in the arrays defined above
    uc=psi[1]
    u=collect(uc:(1-uc)/grid:1)
    phi=inverse(u, psi, v)
    vvec[:, n]=v
    psivec[:, n]=psi
    mass2vec[:, n]=mass2
    uvec[:,n]=u
    phivec[:, n]=phi

    vc=v[1]
    uc=u[1]
   #The code below uses the data about equilibrium to calculate the economic variables
 of interest. The precise formulas follow either from the paper (like the wage equatio
n, Proposition 1), common sense or well known formulae (variance).
G2=mass2/R2 #skill distribution in services
mass1psi=C.(psivec, vvec).-mass2vec #S_M(\psi(v_S))=C(\psi(v_S), v_S)-S_S(v_S)
mass1fun(a)=makecont(a, vvec[:, n], mass1psi[:, n]) #a function that provides a linear
 approximation of the values of S_M(\psi(v))
        #for points for which it has not been computed
mass1=mass1fun(phivec[:, n]) #S_M(\psi(\phi(v_M)))=S_M(v_M)
        mass1vec[:, n]=mass1 #stores the above
G1=mass1/R1 #skill distribution in manufacturing
#The next two loops "calibrate" the model, to make sure that the difference between hig
hest and lowest wages is somewhat realistic
        #specifically, it makes sure that W(1)=152891 initially, whereas W(0)=2140 init
ially.
     if n==1 || n==3 || n==6 || n==9
 global  muZ1, muZ2, A=0, 0, 0
global t=collect(0:mystep:1)
global Gsym=C.(t, t)
global Afun(x)=2140-exp(sigma1*normquant(cutoff)+mu1+sigmaZ*normquant(cutoff)+x)
global pi1usym=pi1uex.(t, Gsym)
```

```
global w1sym=inttrapvec(pi1usym, 0)
global muZ1=log((152891-2140)/w1sym[end])
            global muZ2=muZ1
global A=2140-pi1vec(0, 0)
    end
 if n==12
 global   muZ1, A=0, 0
      pi1uG1=pi1uex.(u, G1)
w1=inttrapvec(pi1uG1, uc)
  global      rec=w1[end]
global        muZ1=log((152891-2140)/w1[end])
global A=2140-pi1vec(uc, 0)
        global muZ2=muZ1-(sqrt(1-alpha1N0^2)*7+1)
    end
   #The next four lines store the parameter values
          parametersvec[1, n], parametersvec[2, n], parametersvec[3, n], parametersvec
[4, n]=alpha1C, alpha1N, alpha2C, alpha2N
   parametersvec[5, n], parametersvec[6, n], parametersvec[7, n], parametersvec[8, n]=m
uC, muN, mu1, mu2
   parametersvec[9, n], parametersvec[10, n], parametersvec[11, n], parametersvec[12, n
]=muZ1, sigmaZ1, muZ2, sigmaZ2
   parametersvec[13, n], parametersvec[14, n], parametersvec[15, n]=sigma1, sigma2, the
ta

      F=C.(psivec[:, n], vvec[:, n])  #workers wage rank as a function of skill in serv
ices
    Fcomp=C.(psivec[:, 1], vvec[:, 1]) #workers wage rank as a function of skill in ser
vices, composition effect only
    global theta=parametersvec[15,1]
    Fwage=C.(psivec[:, n], vvec[:, n])  #workers wage rank as a function of skill in se
rvices, wage effect only
    global theta=parametersvec[15,n]

    h=collect(0:mystep:1)
    Finv=inverse(h, F, vvec[:, n]) #inverse wage distribution
    Finvcomp=inverse(h, Fcomp, vvec[:, 1]) #inverse wage distr, composition effect
    Finvwage=inverse(h, Fwage, vvec[:, n]) #inverse wage distr, wage effect

    wmin=min(pi1vec(uc, 0), pi2vec(vc, 0))#the lowest wage (under Assumption 5)
    pi1uG1=pi1uex.(u, G1) #wage gradient in manufacturing
w1=inttrapvec(pi1uG1, uc).+wmin #wage function in manufacturing
pi2vG2=pi2vex.(v, G2) #wage gradient in services
    w2[:, n]=inttrapvec(pi2vG2, vc).+wmin #Wage, as function of skill, in services
    w1fun(a)=makecont(a, uvec[:, n], w1) #makes the wage function continuous by connect
ing the points in the array with linear segments
w2fun(a)=makecont(a, vvec[:, n], w2[:, n]) #makes the wage function continuous by conne
cting the points in the array with linear segments
    w2funold(a)=makecont(a, vvec[:, 1], w2[:, 1]) #same as above but for old wage funct
ions
G2inv=inverse(h, G2, v) #The inverse of the relative skill distribution in services, so
a function that determines what agent is firm h matched with
    G1inv=inverse(h, G1, u) #The inverse of the relative skill distribution in service
s, so a function that determines what agent is firm h matched with

    wagerank[:, n]=w2fun(Finv) #inverse distribution of wages (overall)
    wagerank1[:, n]=w1fun(G1inv) #inverse distribution of wages (manufacturing)
    wagerank2[:, n]=w2fun(G2inv) #inverse distribution of wages (services)
    #the next three lines make sure that the top wages do not suffer from an approximat
ion bias
        wagerank[end, n]=max(w2[end, n], w1[end])
        wagerank1[end, n]=w1[end]
```

```
        wagerank2[end, n]=w2[end, n]
    compeff[:, n]=w2funold(Finvcomp) #inverse distribution of wages (overall, compositi
on effect only)
    wageeff[:, n]=w2fun(Finvwage) #inverse distribution of wages (overall, wage effect
 only)
end
```

```
iteration=1 iteration=2 iteration=3 iteration=4 iteration=5 iteration=6 it
eration=7 iteration=8 iteration=9 iteration=10 iteration=11 iteration=12 i
teration=13 iteration=14
```

The next cell prints all the parameters for each iteration. This print out has been used to create Section OA.5 in the Online Appendix.

In [9]:

```
for n=1:iterations
   p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15=round(parametersvec
[1, n]; digits=3),round(parametersvec[2, n]; digits=3),round(parametersvec[3, n]; digit
s=3),round(parametersvec[4, n]; digits=3),round(parametersvec[5, n]; digits=3),round(pa
rametersvec[6, n]; digits=3),round(parametersvec[7, n]; digits=3),round(parametersvec[8
, n]; digits=3),round(parametersvec[9, n]; digits=3),round(parametersvec[10, n]; digits
=3),round(parametersvec[11, n]; digits=3),round(parametersvec[12, n]; digits=3),round(p
arametersvec[13, n]; digits=3),round(parametersvec[14, n]; digits=3),round(parametersve
c[15, n]; digits=3)
    muMF, muSF=round(p9/p10; digits=3), round(p11/p12; digits=3)
    println("$n alpha_{MC}=$p1, alpha_{MN}=$p2, alpha_{SC}=$p3, alpha_{SN}=$p4, mu_{xC}
=$p5, mu_{xN}=$p6, mu_M=$p7, mu_S=$p8, mu_{MF}=$muMF, alpha_{MF}=$p10, mu_{SF}=$muSF, a
lpha_{SF}=$p12, sigma_M=$p13, sigma_S=$p14, rho=$p15")
println("")
end
```

1 alpha_{MC}=0.497, alpha_{MN}=0.444, alpha_{SC}=0.0, alpha_{SN}=0.667, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=22.932, alpha_{MF}=0.4, mu_{SF}=22.932, alpha_{SF}=0.4, sigma_M=0.667, sigma_S=0.667, rho=0.667

2 alpha_{MC}=0.094, alpha_{MN}=0.66, alpha_{SC}=0.0, alpha_{SN}=0.667, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=22.932, alpha_{MF}=0.4, mu_{SF}=22.932, alpha_{SF}=0.4, sigma_M=0.667, sigma_S=0.667, rho=0.99

3 alpha_{MC}=0.335, alpha_{MN}=0.3, alpha_{SC}=0.0, alpha_{SN}=0.45, mu_{xC}=0.05, mu_{xN}=0.112, mu_M=0.05, mu_S=0.05, mu_{MF}=6.881, alpha_{MF}=1.2, mu_{SF}=6.881, alpha_{SF}=1.2, sigma_M=0.45, sigma_S=0.45, rho=0.667

4 alpha_{MC}=0.335, alpha_{MN}=0.3, alpha_{SC}=0.0, alpha_{SN}=0.45, mu_{xC}=0.05, mu_{xN}=0.112, mu_M=0.05, mu_S=0.05, mu_{MF}=6.881, alpha_{MF}=1.2, mu_{SF}=6.881, alpha_{SF}=1.2, sigma_M=0.45, sigma_S=0.45, rho=0.738

5 alpha_{MC}=0.335, alpha_{MN}=0.3, alpha_{SC}=0.045, alpha_{SN}=0.45, mu_{xC}=0.05, mu_{xN}=0.112, mu_M=0.05, mu_S=0.053, mu_{MF}=6.881, alpha_{MF}=1.2, mu_{SF}=6.881, alpha_{SF}=1.2, sigma_M=0.45, sigma_S=0.452, rho=0.738

6 alpha_{MC}=0.55, alpha_{MN}=0.0, alpha_{SC}=0.55, alpha_{SN}=0.002, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=49.63, alpha_{MF}=0.2, mu_{SF}=49.63, alpha_{SF}=0.2, sigma_M=0.55, sigma_S=0.55, rho=1.0

7 alpha_{MC}=0.55, alpha_{MN}=0.0, alpha_{SC}=0.55, alpha_{SN}=0.002, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=49.63, alpha_{MF}=0.2, mu_{SF}=49.63, alpha_{SF}=0.2, sigma_M=0.55, sigma_S=0.55, rho=0.995

8 alpha_{MC}=0.55, alpha_{MN}=0.0, alpha_{SC}=0.55, alpha_{SN}=0.057, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=49.63, alpha_{MF}=0.2, mu_{SF}=49.63, alpha_{SF}=0.2, sigma_M=0.55, sigma_S=0.553, rho=0.995

9 alpha_{MC}=0.312, alpha_{MN}=0.95, alpha_{SC}=0.0, alpha_{SN}=1.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=19.915, alpha_{MF}=0.4, mu_{SF}=19.915, alpha_{SF}=0.4, sigma_M=1.0, sigma_S=1.0, rho=0.95

10 alpha_{MC}=0.433, alpha_{MN}=0.95, alpha_{SC}=0.3, alpha_{SN}=1.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=19.915, alpha_{MF}=0.4, mu_{SF}=19.915, alpha_{SF}=0.4, sigma_M=1.044, sigma_S=1.044, rho=0.95

11 alpha_{MC}=0.433, alpha_{MN}=0.95, alpha_{SC}=0.3, alpha_{SN}=1.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=19.915, alpha_{MF}=0.4, mu_{SF}=19.915, alpha_{SF}=0.4, sigma_M=1.044, sigma_S=1.044, rho=0.991

12 alpha_{MC}=0.745, alpha_{MN}=0.667, alpha_{SC}=0.0, alpha_{SN}=4.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=1.933, alpha_{MF}=2.0, mu_{SF}=-1.176, alpha_{SF}=2.0, sigma_M=1.0, sigma_S=4.0, rho=0.667

13 alpha_{MC}=0.745, alpha_{MN}=0.667, alpha_{SC}=0.1, alpha_{SN}=4.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=1.933, alpha_{MF}=2.0, mu_{SF}=-1.176, alpha_{SF}=2.0, sigma_M=1.0, sigma_S=4.0, rho=0.685

14 alpha_{MC}=0.745, alpha_{MN}=0.667, alpha_{SC}=0.1, alpha_{SN}=4.0, mu_{xC}=0.0, mu_{xN}=0.0, mu_M=0.0, mu_S=0.0, mu_{MF}=1.933, alpha_{MF}=2.0, mu_{SF}=-1.176, alpha_{SF}=2.0, sigma_M=1.0, sigma_S=4.001, rho=0.685

In [10]:

```
using PyPlot
```

```
WARNING: using PyPlot.grid in module Main conflicts with an existing ident
ifier.
```

The next 5 cells create the two panels of Figure 3.

In [27]:

```
high=2
logmedianoverall=log(wagerank[50000, high]).-log(wagerank[50000, 1]) #the change in the
log of median wage
logmediancompeff=log(compeff[50000, high]).-log(compeff[50000, 1]) #same but for the di
stribution effect
logmedianwageeff=log(wageeff[50000, high]).-log(wageeff[50000, 1]) #same but for the wa
ge effect
logoverall=log.(wagerank[:, high]).-log.(wagerank[:, 1]) #the change in the log of the
 inverse wage distribution
logcompeff=log.(compeff[:, high]).-log.(compeff[:, 1]) #the change in the log of the in
verse wage distribution if the wage function is held constant
logwageeff=log.(wageeff[:, high]).-log.(wageeff[:, 1]) #the change in the log of the in
verse wage distribution if the skill dsitribution is held constant
logoverall, logcompeff, logwageeff
```

Out[27]:

```
([0.0, -0.0034181, -0.00570044, -0.00760375, -0.00926921, -0.0107631, -0.0
121231, -0.0133751, -0.0145374, -0.0156231  …  0.0497334, 0.0505164, 0.051
3025, 0.0521002, 0.0529013, 0.0537179, 0.0545344, 0.0553607, 0.0561838, 0.
0572173], [0.0, -0.00341826, -0.00570112, -0.00760534, -0.00927218, -0.010
7678, -0.0121299, -0.0133844, -0.0145496, -0.0156385  …  -0.00854868, -0.0
0763099, -0.00670836, -0.00577229, -0.00483075, -0.00387134, -0.00291023,
-0.00193757, -0.000966687, 0.0], [0.0, 1.72942e-5, 5.15714e-5, 9.52333e-5,
0.000145297, 0.000200012, 0.00025826, 0.000319225, 0.000382326, 0.00044710
3  …  0.0578338, 0.0577447, 0.0576543, 0.0575637, 0.0574716, 0.0573794, 0.
0572856, 0.0571919, 0.0570969, 0.057002])
```

In [28]:

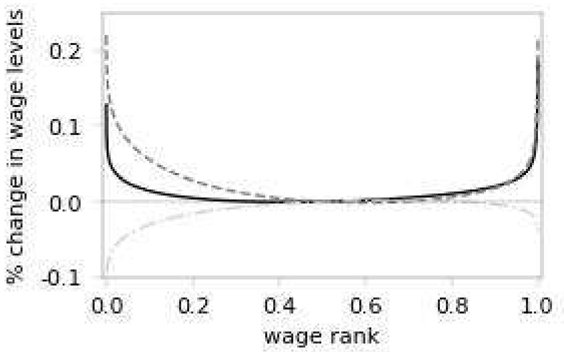
```
#This is used that the graphs are roughly symmetrically placed inthe pdf
xleftabss=0.16*0.5
xrightabsv=0.03*0.5
xleftabsv=0.16*0.5
xrightabss=0.05*0.5
ratio=(1-xrightabsv-xleftabsv-(-xrightabss-xleftabss))*0.5
```

Out[28]:

```
0.505
```

In [29]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 452.9679*(5/10)  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*golden_mean#*(1-xleftabsv/(1-ratio)-xleftabsv/(1-ratio))/(1-0.2-
0.03)       # height in inches
fig = figure(figsize=(fig_width,fig_height))
t=collect(0:0.00001:1)
plot(t, logoverall.-logmedianoverall, color="black"#(tableau20rbc[1,1], tableau20rbc[1,
2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall effect")
plot(t, logcompeff.-logmediancompeff, color="grey"#(tableau20rbc[5,1], tableau20rbc[5,
2], tableau20rbc[5,3])
    , linewidth=1.0,  ls="--", label="Composition effect")
plot(t, logwageeff.-logmedianwageeff, color="lightgrey"#(tableau20rbc[7,1], tableau20rb
c[7,2], tableau20rbc[7,3])
    , linewidth=1.0, ls="-.", label="Wage effect")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
ylim((-0.10, 0.25))
xlim(-0.01, 1.01)
xlabel("wage rank")
ylabel("% change in wage levels")
#legend(loc="best")
subplots_adjust(left=0.2, bottom=0.2)
savefig("polarisationnormal.pgf") #The figure issaved as a pgf, so that all the fonts i
n the pdf will be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.
```



In [14]:

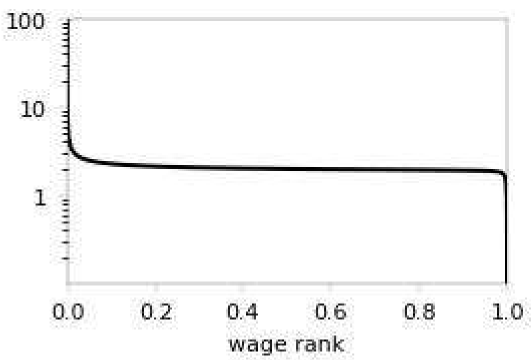
```
compoeff=compeff[:, high].-compeff[:, 1] #the change in the log of the inverse wage com
pibution if the wage function is held constant
wageseff=wageeff[:, high].-wageeff[:, 1] #the change in the log of the inverse wage com
pibution if the skill dsitribution is held constant
ratioeff=(-compoeff[2:end])./(wageseff[2:end])
logratioeff=log.(-compoeff[2:end])-log.(wageseff[2:end])
logcompoeff1=log.(compeff[2:end, high].-compeff[1, high]).-log.(compeff[2:end, 1].-comp
eff[1, 1]) #the change in the log of the inverse wage compibution if the wage function
 is held constant
logwageeff1=log.(wageeff[2:end, high].-wageeff[1, high]).-log.(wageeff[2:end, 1].-wagee
ff[1, 1]) #the change in the log of the inverse wage compibution if the skill dsitribut
ion is held constant
logcompoeff1, logwageeff1
```

Out[14]:

```
([-2.18555, -2.03988, -1.94869, -1.88109, -1.82668, -1.78076, -1.74092, -
1.70586, -1.67436, -1.64574  …  -0.00867247, -0.00774125, -0.00680508, -0.
00585533, -0.00490009, -0.00392679, -0.00295182, -0.0019652, -0.00098044,
0.0], [0.00448824, 0.00786115, 0.0107217, 0.0132561, 0.0155495, 0.0176565,
0.0196096, 0.0214346, 0.0231487, 0.0247665  …  0.0586437, 0.058552, 0.0584
59, 0.0583658, 0.058271, 0.0581762, 0.0580798, 0.0579835, 0.0578857, 0.057
7883])
```

In [15]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 452.9679*(5/10)  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*golden_mean#*(1-xleftabsv/(1-ratio)-xleftabsv/(1-ratio))/(1-0.2-
0.03)
fig = figure(figsize=(fig_width,fig_height))
t=collect(0.00001:0.00001:1)
plot(t, ratioeff[1:100000], color="black"#(tableau20rbc[9,1], tableau20rbc[9,2], tablea
u20rbc[9,3]), linewidth=2.0
    , label="Ratio: Composition to Wage Effect")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
yscale("log")
yticks([1, 10^1, 10^2], [1, 10, 100])
ylim(10^(-1), 10^2)
xlim(0.0, 1)
xlabel("wage rank")
subplots_adjust(left=0.2, bottom=0.2)
savefig("effectratio.pgf") #The figure issaved as a pgf, so that all the fonts in the p
df will be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.
```



The next two cells produce Figure 5 (a)

In [16]:

```
#This is used that the graphs are roughly symmetrically placed inthe pdf
xleftabss=0.16*0.5
xrightabsv=0.03*0.5
xleftabsv=0.16*0.5
xrightabss=0.05*0.5
ratio=(1-xrightabsv-xleftabsv-(-xrightabss-xleftabss))*0.5
```

Out[16]:

```
0.505
```

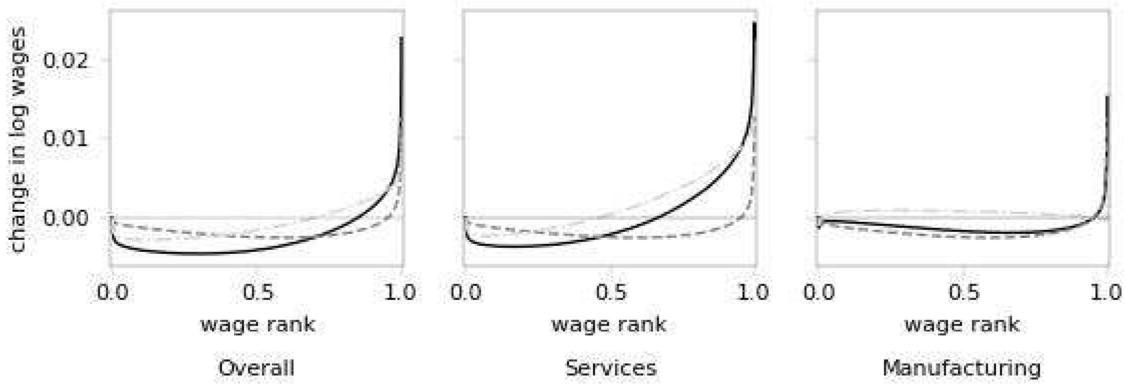
In [17]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 1*452.9679  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*0.35#*(1-0.15)/(1-0.1) #*golden_mean*(1-xleftabsv/(1-ratio)-xleft
absv/(1-ratio))/(1-0.2-0.03)       # height in inches
fig = figure(figsize=(fig_width,fig_height))
ax=fig[:add_subplot](1,3,1)#, sharey="True")
t=collect(0:0.00001:1)
plot(t, log.(wagerank[:, 5]./wagerank[:, 3]), color="black"#(tableau20rbc[1,1], tableau
20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall effect")
plot(t, log.(wagerank[:, 4]./wagerank[:, 3]), color="grey"#(tableau20rbc[5,1], tableau2
0rbc[5,2], tableau20rbc[5,3])
    , linewidth=1.0,  ls="--", label=L"Change in $\rho$")
plot(t, log.(wagerank[:, 5]./wagerank[:, 4]), color="lightgrey"#(tableau20rbc[7,1], tab
leau20rbc[7,2], tableau20rbc[7,3])
    , linewidth=1.0, ls="-.", label=L"Change in $\pi_S$")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
#ylim((-0.01, 0.03))
xlim(-0.01, 1.01)
xlabel("wage rank")
ylabel("change in log wages")
#legend(loc="best")
title("Overall", y=-0.5)
subplots_adjust(left=0.1, bottom=0.3)

ay=fig[:add_subplot](1,3,2, sharey=ax)#, sharey="True")
t=collect(0:0.00001:1)
plot(t, log.(wagerank2[:, 5]./wagerank2[:, 3]), color="black"#(tableau20rbc[1,1], table
au20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0
    , label="Overall")
plot(t, log.(wagerank2[:, 4]./wagerank2[:, 3]), color="grey"#(tableau20rbc[5,1], tablea
u20rbc[5,2], tableau20rbc[5,3])
    , linewidth=1.0
    ,  ls="--", label="Concordance")
plot(t, log.(wagerank2[:, 5]./wagerank2[:, 4]), color="lightgrey"#(tableau20rbc[7,1], t
ableau20rbc[7,2], tableau20rbc[7,3])
    ,linewidth=1.0
    , ls="-.", label="Wage effect")
axhline(0, 0, 1, lw=0.2, color="black", ls="-")
ax1=gca()
yticks(visible=false)
xlim(-0.01, 1.01)
xlabel("wage rank")
```

```
subplots_adjust(hspace=0.0)
title("Services", y=-0.5)
savefig("tbtcservices.pgf") #The figure issaved as a pgf, so that all the fonts in the
 pdf will be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.ax1

ay=fig[:add_subplot](1,3,3, sharey=ax)#, sharey="True")
t=collect(0:0.00001:1)
plot(t, log.(wagerank1[:, 5]./wagerank1[:, 3]), color="black"#(tableau20rbc[1,1], table
au20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall effect")
plot(t, log.(wagerank1[:, 4]./wagerank1[:, 3]), color="grey"#(tableau20rbc[5,1], tablea
u20rbc[5,2], tableau20rbc[5,3])
    , linewidth=1.0,  ls="--", label="Concordance")
plot(t, log.(wagerank1[:, 5]./wagerank1[:, 4]), color="lightgrey"#(tableau20rbc[7,1], t
ableau20rbc[7,2], tableau20rbc[7,3])
    ,linewidth=1.0, ls="-.", label="Wage effect")
axhline(0, 0, 1, lw=0.2, color="black", ls="-")
ax1=gca()
yticks(visible=false)
xlim(-0.01, 1.01)
xlabel("wage rank")
title("Manufacturing", y=-0.5)
savefig("tbtc.pgf") #The figure issaved as a pgf, so that all the fonts in the pdf will
be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.ax1
```



The next 3 cells produe Figure 5 (b)

In [18]:

```
high=8 #the figure is plotted for a change from theta=0.66 to theta=0.99
logmedianoverall=log(wagerank[50000, high]).-log(wagerank[50000, 6]) #the change in the
log of median wage
logmedianconceff=log(wagerank[50000, 7]).-log(wagerank[50000, 6]) #same but for the dis
tribution effect
logmediansigmaeff=log(wagerank[50000, 8]).-log(wagerank[50000, 7]) #same but for the wa
ge effect
logoverall=log.(wagerank[:, high]).-log.(wagerank[:, 6]) #the change in the log of the
 inverse wage distribution
logconceff=log.(wagerank[:, 7]).-log.(wagerank[:, 6]) #the change in the log of the inv
erse wage distribution if the wage function is held constant
logsigmaeff=log.(wagerank[:, 8]).-log.(wagerank[:, 7]) #the change in the log of the in
verse wage distribution if the skill dsitribution is held constant
logoverall, logconceff, logsigmaeff
```

Out[18]:

```
([-0.00869021, -0.00845213, -0.00815192, -0.00786224, -0.00758291, -0.0073
1272, -0.0070523, -0.00679981, -0.00655401, -0.00631444  …  0.00195462, 0.
00176506, 0.00158165, 0.00140965, 0.00125814, 0.00114369, 0.00109613, 0.00
112624, 0.0011715, 0.0010928], [0.0, 0.000325593, 0.000620395, 0.00089994
9, 0.00116806, 0.00142652, 0.00167459, 0.00191543, 0.00214962, 0.00237766
…  -0.00376379, -0.00400177, -0.00424577, -0.00449363, -0.00474823, -0.005
0104, -0.00528108, -0.00556172, -0.00585581, -0.00619833], [-0.00869021, -
0.00877773, -0.00877231, -0.00876219, -0.00875097, -0.00873924, -0.0087268
9, -0.00871524, -0.00870363, -0.0086921  …  0.00571841, 0.00576683, 0.0058
2743, 0.00590328, 0.00600636, 0.00615409, 0.00637721, 0.00668797, 0.007027
3, 0.00729112])
```

In [19]:

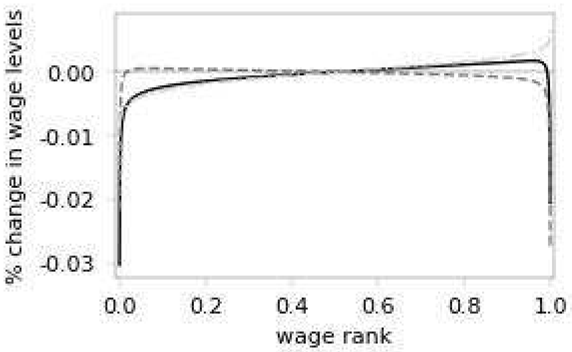
```
#This is used that the graphs are roughly symmetrically placed inthe pdf
xleftabss=0.16*0.5
xrightabsv=0.03*0.5
xleftabsv=0.16*0.5
xrightabss=0.05*0.5
ratio=(1-xrightabsv-xleftabsv-(-xrightabss-xleftabss))*0.5
```

Out[19]:

```
0.505
```

In [20]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 452.9679*(5/10)  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*golden_mean#*(1-xleftabsv/(1-ratio)-xleftabsv/(1-ratio))/(1-0.2-
0.03)       # height in inches
fig = figure(figsize=(fig_width,fig_height))
t=collect(0:0.00001:1)
plot(t, logoverall.-logmedianoverall, color="black"#(tableau20rbc[1,1], tableau20rbc[1,
2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall change")
plot(t, logconceff.-logmedianconceff, color="grey"#(tableau20rbc[5,1], tableau20rbc[5,
2], tableau20rbc[5,3])
    , linewidth=1.0,  ls="--", label=L"Change in $\rho$")
plot(t, logsigmaeff.-logmediansigmaeff, color="lightgrey"#(tableau20rbc[7,1], tableau20
rbc[7,2], tableau20rbc[7,3])
    ,linewidth=1.0, ls="-.", label=L"Change in $\sigma_i$")
axhline(0, 0, 1, lw=0.2, color="black", ls="-")
xlim(-0.01, 1.01)
xlabel("wage rank")
ylabel("% change in wage levels")
#legend(loc="best")
subplots_adjust(left=0.2, bottom=0.2)
savefig("tbtcfc.pgf") #The figure issaved as a pgf, so that all the fonts in the pdf wi
ll be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.
```



The following 3 cells produe Figure 5(c)

In [21]:

```
high=11 #the figure is plotted for a change from theta=0.66 to theta=0.99
logmedianoverall=log(wagerank[50000, high]).-log(wagerank[50000, 9]) #the change in the log of median wage
logmedianconceff=log(wagerank[50000, 11]).-log(wagerank[50000, 10]) #same but for the distribution effect
logmediansigmaeff=log(wagerank[50000, 10]).-log(wagerank[50000, 9]) #same but for the wage effect
logoverall=log.(wagerank[:, high]).-log.(wagerank[:, 9]) #the change in the log of the inverse wage distribution
logconceff=log.(wagerank[:, 11]).-log.(wagerank[:, 10]) #the change in the log of the inverse wage distribution if the wage function is held constant
logsigmaeff=log.(wagerank[:, 10]).-log.(wagerank[:, 9]) #the change in the log of the inverse wage distribution if the skill dsitribution is held constant
logoverall, logconceff, logsigmaeff
```

Out[21]:

```
([-0.00226529, -0.00230964, -0.00234811, -0.00238406, -0.00241835, -0.0024
513, -0.0024831, -0.0025139, -0.00254382, -0.00257291  …  0.15752, 0.15812
2, 0.158736, 0.159362, 0.159992, 0.160639, 0.161298, 0.161962, 0.162647,
0.163461], [0.0, -3.50734e-5, -6.49349e-5, -9.25952e-5, -0.000118813, -0.0
00143892, -0.000167976, -0.000191203, -0.000213687, -0.00023547  …  0.0117
67, 0.0122786, 0.0128018, 0.0133361, 0.0138743, 0.0144294, 0.0149961, 0.01
55679, 0.016161, 0.016881], [-0.00226529, -0.00227457, -0.00228318, -0.002
29147, -0.00229953, -0.00230741, -0.00231513, -0.0023227, -0.00233013, -0.
00233744  …  0.145753, 0.145844, 0.145935, 0.146026, 0.146118, 0.14621, 0.
146302, 0.146394, 0.146486, 0.14658])
```

In [22]:

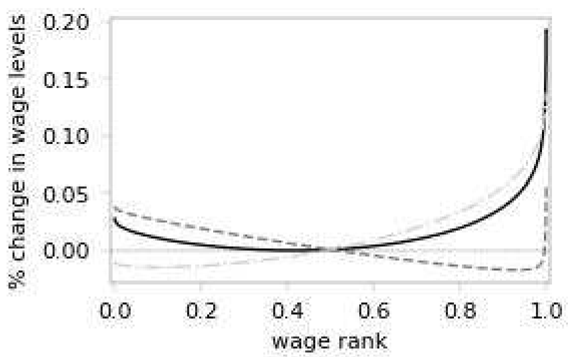
```
#This is used that the graphs are roughly symmetrically placed inthe pdf
xleftabss=0.16*0.5
xrightabsv=0.03*0.5
xleftabsv=0.16*0.5
xrightabss=0.05*0.5
ratio=(1-xrightabsv-xleftabsv-(-xrightabss-xleftabss))*0.5
```

Out[22]:

```
0.505
```

In [23]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 452.9679*(5/10)  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*golden_mean#*(1-xleftabsv/(1-ratio)-xleftabsv/(1-ratio))/(1-0.2-
0.03)       # height in inches
fig = figure(figsize=(fig_width,fig_height))
t=collect(0:0.00001:1)
plot(t, logoverall.-logmedianoverall, color="black"#(tableau20rbc[1,1], tableau20rbc[1,
2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall change")
plot(t, logconceff.-logmedianconceff, color="grey"#(tableau20rbc[5,1], tableau20rbc[5,
2], tableau20rbc[5,3])
    , linewidth=1.0,  ls="--", label=L"Change in $\rho$")
plot(t, logsigmaeff.-logmediansigmaeff, color="lightgrey"#(tableau20rbc[7,1], tableau20
rbc[7,2], tableau20rbc[7,3])
    , linewidth=1.0, ls="-.", label=L"Change in $\sigma_i$")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
#ylim((-0.10, 0.25))
xlim(-0.01, 1.01)
xlabel("wage rank")
ylabel("% change in wage levels")
#legend(loc="best")
subplots_adjust(left=0.2, bottom=0.2)
savefig("sbtc.pgf") #The figure issaved as a pgf, so that all the fonts in the pdf will
be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.
```



The next 3 cells produce Figure OA.1 in the Online Appendix.

In [24]:

```
#This is used that the graphs are roughly symmetrically placed inthe pdf
xleftabss=0.16*0.5
xrightabsv=0.03*0.5
xleftabsv=0.16*0.5
xrightabss=0.05*0.5
ratio=(1-xrightabsv-xleftabsv-(-xrightabss-xleftabss))*0.5
```

Out[24]:

```
0.505
```

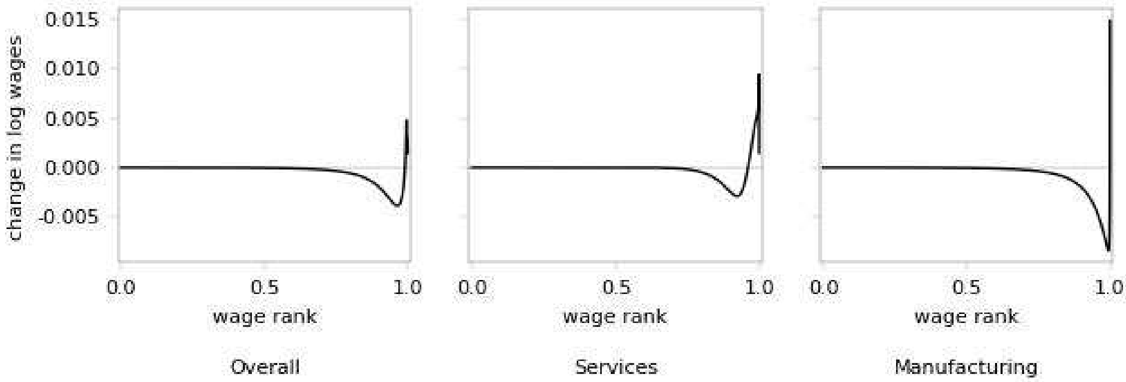
In [25]:

```
xm=1
linewidth = 0.2
si=8
si2=si
rc("axes", linewidth=linewidth)
#rc("font", family="")
rc("axes", titlesize="$si", labelsize="$si")
rc("xtick", labelsize="$si2")
rc("xtick.major", width=linewidth/2)
rc("ytick", labelsize="$si2")
rc("ytick.major", width=linewidth/2)
rc("legend", fontsize="$si")
rc("axes", unicode_minus="False")
fig_width_pt = 1.1*452.9679  # Get this from LaTeX using \showthe\columnwidth
inches_per_pt = 1.0/72.27               # Convert pt to inches
golden_mean = (sqrt(5)-1.0)/2.0         # Aesthetic ratio
fig_width = fig_width_pt*inches_per_pt  # width in inches
fig_height =fig_width*0.35#*(1-0.15)/(1-0.1) #*golden_mean*(1-xleftabsv/(1-ratio)-xleft
absv/(1-ratio))/(1-0.2-0.03)       # height in inches
fig = figure(figsize=(fig_width,fig_height))
ax=fig[:add_subplot](1,3,1)#, sharey="True")
t=collect(0:0.00001:1)
plot(t, log.(wagerank[:, 14]./wagerank[:, 12]), color="black"#(tableau20rbc[1,1], table
au20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall")
#plot(t, log.(wagerank[:, 2]./wagerank[:, 1]), color=(tableau20rbc[5,1], tableau20rbc
[5,2], tableau20rbc[5,3]), linewidth=1.0,  ls="--", label="Concordance")
#plot(t, logwageeff.-logmedianwageeff, color=(tableau20rbc[7,1], tableau20rbc[7,2], tab
leau20rbc[7,3]), ls="-.", label="Wage effect")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
#ylim((-0.005, 0.03))
xlim(-0.01, 1.01)
xlabel("wage rank")
ylabel("change in log wages")
#legend(loc="best")
title("Overall", y=-0.5)
subplots_adjust(left=0.1, bottom=0.3)

ay=fig[:add_subplot](1,3,2, sharey=ax)
t=collect(0:0.00001:1)
plot(t, log.(wagerank2[:, 14]./wagerank2[:, 12]), color="black"#(tableau20rbc[1,1], tab
leau20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
ax1=gca()
yticks(visible=false)
xlim(-0.01, 1.01)
xlabel("wage rank")
subplots_adjust(hspace=0.0)
title("Services", y=-0.5)
savefig("tbtcservices.pgf") #The figure issaved as a pgf, so that all the fonts in the
 pdf will be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not here

ay=fig[:add_subplot](1,3,3, sharey=ax)
t=collect(0:0.00001:1)
plot(t, log.(wagerank1[:, 14]./wagerank1[:, 12]), color="black"#(tableau20rbc[1,1], tab
```

```
leau20rbc[1,2], tableau20rbc[1,3])
    , linewidth=1.0, label="Overall")
axhline(0, 0, 1, lw=0.2, color="black", ls="--")
ax1=gca()
yticks(visible=false)
xlim(-0.01, 1.01)
xlabel("wage rank")
title("Manufacturing", y=-0.5)
savefig("tbtcinequality.pgf") #The figure issaved as a pgf, so that all the fonts in th
e pdf will be the same
#as those used elsewhere in the paper. This is done with all graphs. Note that this eff
ect is visible only in
#the pdf, not there.ax1
```



In [26]: