Iron Furnace Problem
What is the porosity distribution in the iron ore hopper?

Problem presented by

Mansour Al-Harbi

Saudi Basic Industries Corporation (SABIC)

Executive Summary

In the processing of iron ore, pellets are funneled into a furnace and reduced by contact with a gas. We consider the problem of determining whether significant spatial segregation occurs within the furnace based on pellet size. Experiments and computation are used to test whether such segregation may happen at the inlet or near the sides. Experimental results indicate that segregation at the inlet may occur, while segregation due to friction at the sides probably does not. The computational results provided are too crude to base conclusions on, but indicate what could be done with further resources.

Version 1.0
July 13, 2011
iii+10 pages

Report coordinator

D. Ketcheson

Contributors

Aron Ahmadia (KAUST, Saudi Arabia)
Chris Breward (OCCAM, UK)
Conrad Curry (KAUST, Saudi Arabia)
Vu Nguyen (KAUST, Saudi Arabia)
David Ketcheson (KAUST, Saudi Arabia)
Maher Moakher (National Engineering School at Tunis, Tunisia)
Hussein Mosleh (KAUST, Saudi Arabia)
John Ockendon (University of Oxford, UK)
Philippe Vignal (KAUST, Saudi Arabia)

KSG 2011 was organised by
King Abdullah University of Science and Technology (KAUST)
In collaboration with
Oxford Centre for Collaborative Applied Mathematics (OCCAM)

Contents

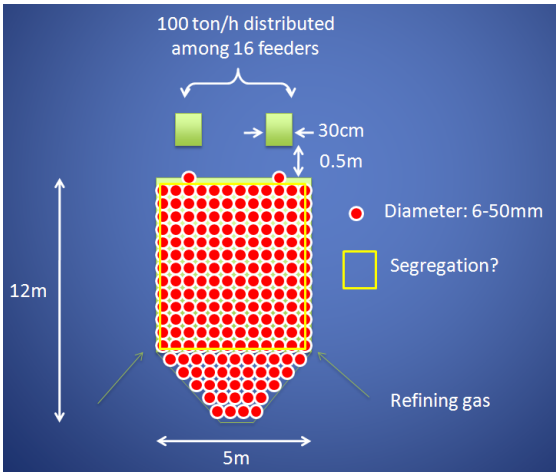


Figure 1: Simplified diagram of the shaft furnace.

1 Problem Description

Mixtures of solid particles can separate or segregate while handled or processed in an iron furnace, and this often results in costly quality control problems. In the iron-making process the iron ore pellets are in the size range 6-50 mm and are processed through a tube-like furnace called a "shaft furnace", which is depicted in a simplified manner in Figure 1.

Gas used for ore reduction is fed from the bottom and moves upwards against gravity in the opposite direction of the solid particle movement. The developing pellet size distribution, due to the solid segregation, significantly influences the gas flow pattern, where the gas would prefer to travel through sites occupied by coarser particles. This would then lead to an ore reduction quality variation due to the gas concentration and temperature variation.

SABIC asked the group to provide a mathematical formulation to help predict any possible solid segregation of ore particles. The group worked on the problem by addressing the question: What is the pellet size distribution in the iron ore hopper? They first worked out that the refining gas inputs and hopper vibrations could not cause any segregation (the gas moves only 1 m/min maximum, and the vibration acceleration is much smaller than gravitational acceleration). The potential causes were thought to be contact with rough walls of the hopper and ore input pile dynamics.

One problem with developing a mathematical model of pellet flow is determining how applicable the model is to the actual furnace. Ideally, data taken from the process would be used to compare against the mathematical model. However, in this case very little data is available. The furnace is opaque and there are no cameras on the inside, nor is it possible to see the input piles where the pellets are deposited into the furnace. A solution would be to wait until the furnace is shut down for maintenance; however this is not scheduled for years. Due to these difficulties, work followed two routes, experimental and computational, as described in the following sections.

Figure 2: Experimental Setup #1: Draining a cylinder with a conical outlet.

2 Experimental Work

In order to provide data to qualify a mathematical model to a rough degree, several bench-top granular flow experiments were conducted. The main goal was to determine the relative effect of segregation by the two most likely contributors, the interaction of the pellets with the furnace sidewalls and the deposition of pellets into the furnace. Participants at the Study Group visited the local supermarket and bought white cous cous and dried green beens to use as pellets. All the particles were roughly spherical and the beans had a radius about twice that of the cous cous. The two types of pellets were combined in proportions typical of the distribution of pellet size Hoppers were constructed from drinks bottles of various shapes and sizes.

2.1 Experimental Setup #1: Draining a Cylinder with a Conical Outlet

The first experimental setup was to determine the effect of the interaction of the pellets with the furnace sidewalls on segregation. A two litre soda bottle was drained and the bottom was removed. After inverting the bottle it represented the furnace. The bottle was clear making the beans in contact with the walls of the bottle and the beans on top visible. As the bottle was drained of beans the walls and top were observed. The beans moved downward in the cylinder toward the conical outlet as if they formed a rigid body. This experiment showed that the sidewall interaction most likely did not induce segregation in the pellets.

2.2 Experimental Setup #2: Three inlets pouring from the top of the cylinder

The second experimental setup involved three inlets depositing from a high distance to the piles. The beans sprayed from the inlets instead of dropping straight down making it difficult to characterize the distributions. The SABIC representatives later confirmed that the inlets deposit pellets while in contact with the piles.

Figure 3: Experimental Setup #3: Three inlets pouring from a variable height above the piles.

2.3 Experimental Setup #3: Three inlets pouring from a variable height above the piles

In order to more closely represent the furnace, the third experimental setup involved moving three draining bottles of beans upward while piles formed below them. This allows the distance that the pellets drop to be varied and see the resulting effect on segregation. The distance between the top of the pile and the inlet dramatically affected the segregation of the beans. At a great distance the segregation was indeterminable because the beans sprayed from the inlets. When the inlets were directly in contact with the tops of the piles little segregation occurred. When the inlet was near to the top of the pile, segregation was noticed with a higher concentration of large beans toward the outsides of the piles. A second key observation from this experiment was that the location where the piles interacted did not have any effect on the distributions of beans. This allowed data from experimental setup #4 to be used to qualify the mathematical model.

2.4 Experimental Setup #4: One inlet pouring from a variable height above the pile

This setup is similar to the experimental setup #3 except with only one input bottle. The simplified geometry allows for easier quantification of the segregation of the beans along the radius of the pile. As expected, the results from experimental setup #3 were reproduced: the distribution of beans was highly dependent upon the distance from the inlet to the pile.

Histograms for the radial distribution of the beans after the pile formed – for one such experiment – are shown in Figure 6. These histograms were computed by taking a photo of the bean pile from above, increasing the contrast to create a clear distinction between the large (dark-colored) and small (light-colored) beans, and then counting pixels as a function of radius. The high-contrast image used is shown in Figure 5.

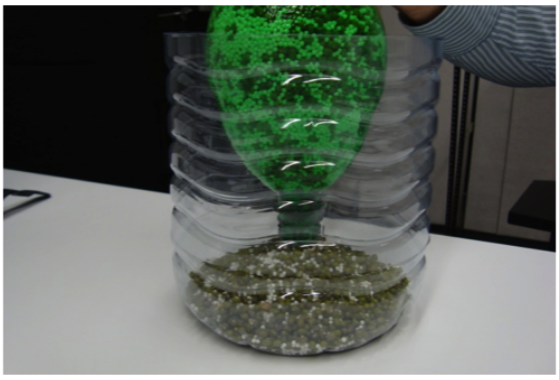


Figure 4: Experimental Setup #4: One inlet pouring from a variable height above the pile.

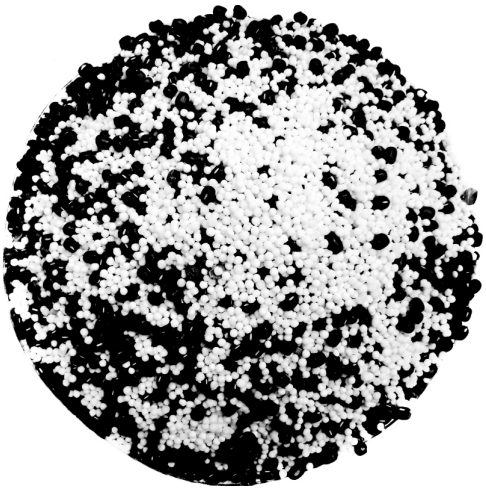
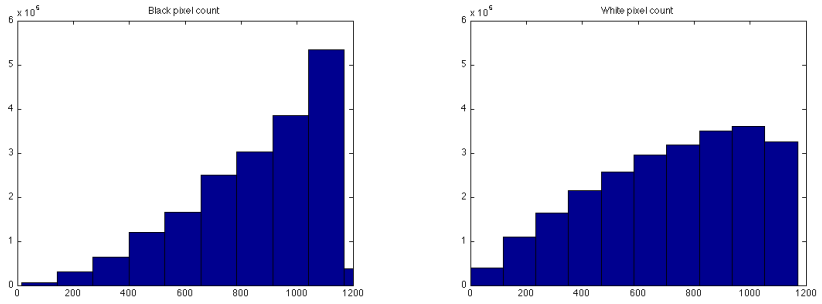


Figure 5: Overhead image of an experimental result.



(a) Large bean distribution, computed using black pixels as a proxy.

(b) Small bean distribution, computed using white pixels as a proxy.

Figure 6: Approximate radial distribution of large and small beans, using thresholding and pixel counts.

3 Computational Work

Modelling the collisions of three sizes of pellets through one inlet was carried out using Yade, an open source code for the Discrete Element Method (DEM). The software is written in C++ with a Python interface.

The DEM is a general computational method for the simulation of the dynamics of a large collection of discrete particles. It was first introduced by Cundall and Strack in 1979 for the simulation of circular disks with applications in geotechnics. The DEM, which is also called 'particle dynamics', is largely motivated by the success of molecular dynamics in predicting the dynamics of gas and liquid systems. Nowadays, the DEM has become a general framework for dealing with the discrete nature of granular flows with applications in particle mixing, solid milling, soil mechanics, etc. In the DEM, the granular material is modeled as a large collection of discrete (spherical in our case) particles which interact with each other and move according to Newton's laws of classical mechanics. The collision between particles is assumed to be binary and of finite duration. Furthermore, the overlap of particles, which models their deformation, is assumed to be small compared with their sizes. The DEM algorithm starts by assigning initial positions and velocities of the particles. Then a contact detection algorithm is used to find out which pairs of particles are in contact. Normal and tangential contact laws are used to compute the interparticle forces. Finally, Newton's equations of motion are integrated using the leap-frog algorithm to obtain the positions of particles at one time step ahead. Time steps are chosen to be small enough so that stability criteria are met. The major drawback of the DEM is its large CPU time.

The computational simulation using DEM started by defining its three main elements: contact detection algorithm, contact force model, and numerical integration. The granular material was modelled as a collection of soft and rough spherical particles, and it was assumed that deformation of a particle is small compared to its size, and collisions were assumed to be only binary with finite duration.

We used the freely available YADE (Yet Another Dynamic Engine) framework described at `https://yade-dem.org/wiki/Yade` for the Discrete Element Method to investigate some computational approaches to the problem.

As we were working from tutorial slides that were based off the development version of yade[1], it was necessary to build the development release from source, following the directions available here: `https://yade-dem.org/wiki/Installation_of_yade_on_debian_or_kubuntu`.

More up to date information on the installation process is usually available here: `https://launchpad.net/yade`.

Appendix A contains the two scripts we used for generating the data shown in the presentation. The script `our_gravity.py` is an adaptation of `gravity.py`, with some preliminary setup attempting to create the scenario described to us. The initial setup of the simulation created by this script is depicted in Figure 7.

The script `our_post.py` is a very simple postprocessing step to gather informa-

[1] `https://www.yade-dem.org/doc/tutorial-examples.html#gravity-deposition`

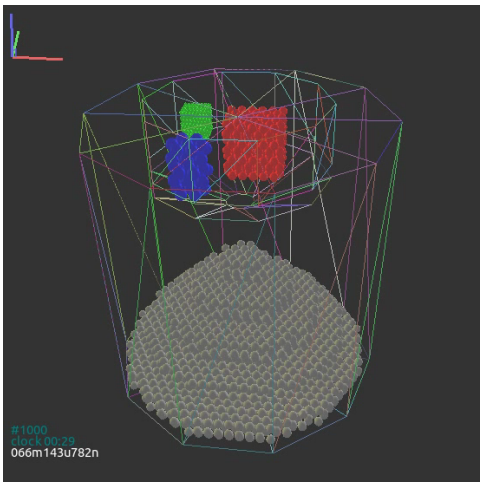


Figure 7: Initial setup for YADE simulation.

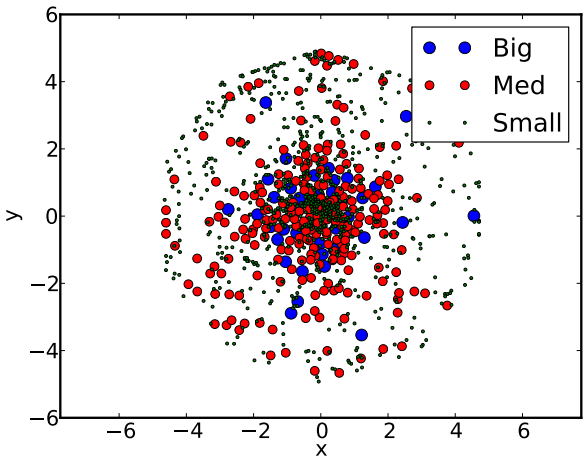


Figure 8: Final locations of particles at the end of the simulation.

tion about the radial distance of the different pellet sizes from the center. Results of the simulation are shown in Figures 8 and 9.

We believe that further modification to the source code could yield a more accurate simulation capable of properly modeling the pellet deposition process.

4 Conclusion

The group has demonstrated how simple experimentation and software simulation can be used to gain insight into the mechanisms of the problem. **This preliminary work indicates that segregation due to sidewall friction is unlikely, whereas segregation due to pouring dynamics is possible.** To gain more detailed conclusions, further work can easily be done by extending these approaches.

In order to corroborate this negative but reassuring conclusion, the experiments

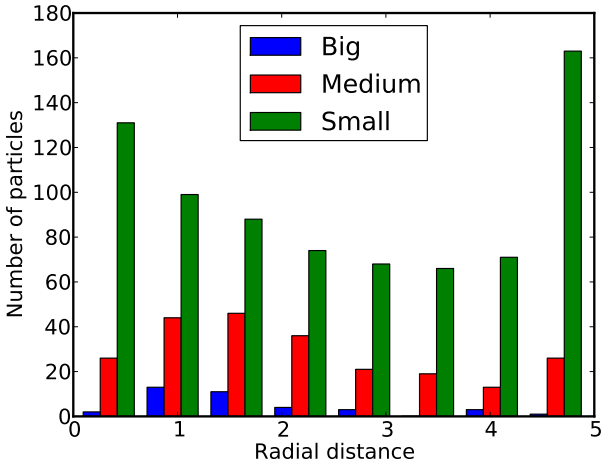


Figure 9: Histogram of particle distribution versus horizontal distance from the inlet location.

conducted should be used as a guide for designing more realistic experiments with actual iron ore samples. These experiments should employ the correct physical parameters for the pouring of the iron pellets, which were not available at the Study Group. In this way it should be possible to quickly establish the general degree of segregation occurring in the actual loading of the hopper.

Based on the scientific literature, this segregation likely depends in a significant way on the distribution of pellet sizes, which varies substantially from sample to sample. As it may not be feasible to conduct and analyze experiments with many different samples, it would be desirable to further develop the computational work started here. Once the physically appropriate computational setup and data post-processing is established, it would be straightforward to conduct and analyze a large number of simulations with varying distributions of grain sizes. However, significant work is needed in setting up the simulations with more realistic pouring conditions and pre-mixing of the pellets. Also, in order to simulate a sufficiently large number of pellets, it may be necessary to modify the algorithms used here, e.g. by freezing particles that are no longer in motion in order to save computational effort.

After the characteristics of the pellet pile in the furnace are established, the next step in this work would be to simulate the flow of gas through the pile with given characteristics. This will allow determination of the typical degree of variation in gas concentration and temperature.

A Scripts for running and processing YADE simulations

our_gravity.py:

```
# gravity deposition in box, showing how to plot and save history of
    data,
# and how to control the simulation while it is running by calling
# python functions from within the simulation loop

# import yade modules that we will use below
from yade import pack, plot, geom
import qt

# create rectangular box from facets
#O.bodies.append(utils.facetBox((2.5,2.5,2.5),(5,5,5),wallMask=31))


realH  = 80
realOR = 30

# boundary cylinder height
H = 10

# bunker height
bH = 5

# sub-bunker height
bHb = 2
# output height
bHo = 1

O.materials.append(yade.wrapper.RpmMat(density=7000))

b = geom.facetBunker((2.5,2.5,7.5),6,1,bHb,bHo)
O.bodies.append(b)

f = geom.facetCylinder((2.5,2.5,H/2),5,H)
O.bodies.append(f)

fpred = pack.inCylinder((2.5,2.5,0),(2.5,2.5,1),4.8)
O.bodies.append(pack.regularHexa(fpred,radius=0.15,gap=0.15/4,color
    =(0.5,0.5,0.5)))

fpred = pack.inCylinder((2.5,2.5,1),(2.5,2.5,2),4)
O.bodies.append(pack.regularHexa(fpred,radius=0.15,gap=0.15/4,color
    =(0.5,0.5,0.5)))
fpred = pack.inCylinder((2.5,2.5,2),(2.5,2.5,3),3)
O.bodies.append(pack.regularHexa(fpred,radius=0.15,gap=0.15/4,color
    =(0.5,0.5,0.5)))

fpred = pack.inCylinder((2.5,2.5,3),(2.5,2.5,4),2)
O.bodies.append(pack.regularHexa(fpred,radius=0.15,gap=0.15/4,color
    =(0.5,0.5,0.5)))
```

```
fpred = pack.inCylinder((2.5,2.5,4),(2.5,2.5,5),1)
O.bodies.append(pack.regularHexa(fpred,radius=0.15,gap=0.15/4,color
    =(0.5,0.5,0.5)))

pred1 = pack.inAlignedBox((0.5,0.5,H-2),(1.5,1.5,H))
radius1=0.2
O.bodies.append(pack.regularHexa(pred1,radius=radius1,gap=radius1/4,
    color=(0,0,1)))

pred2 = pack.inAlignedBox((3,3,H-2),(4,4,H))
radius2=0.15
O.bodies.append(pack.regularHexa(pred2,radius=radius2,gap=radius2/4,
    color=(1,0,0)))

pred3 = pack.inAlignedBox((0.5,3,H-0.5),(1.5,4,H))
radius3=0.05
O.bodies.append(pack.regularHexa(pred3,radius=radius3,gap=radius3/4,
    color=(0,1,0)))

O.engines=[
  qt.SnapshotEngine(iterPeriod=100,fileBase='./snap-',label='
      snapshooter'),
  ForceResetter(),
  InsertionSortCollider([Bo1_Sphere_Aabb(),Bo1_Facet_Aabb()]),
  InteractionLoop(
    # handle sphere+sphere and facet+sphere collisions
    [Ig2_Sphere_Sphere_L3Geom(),Ig2_Facet_Sphere_L3Geom()],
    [Ip2_FrictMat_FrictMat_FrictPhys()],
    [Law2_L3Geom_FrictPhys_ElPerfPl()]
  ),
  GravityEngine(gravity=(0,0,-9.81)),
  NewtonIntegrator(damping=0.4),
  # call the checkUnbalanced function (defined below) every 2 seconds
  PyRunner(command='checkUnbalanced()',realPeriod=2),
  # call the addPlotData function every 200 steps
  PyRunner(command='addPlotData()',iterPeriod=200)
]
O.dt=.5*utils.PWaveTimeStep()

# enable energy tracking; any simulation parts supporting it
# can create and update arbitrary energy types, which can be
# accessed as O.energy['energyName'] subsequently
O.trackEnergy=True

# if the unbalanced forces goes below .05, the packing
# is considered stabilized, therefore we stop collected
# data history and stop
def checkUnbalanced():
  if utils.unbalancedForce()<.05:
#   O.pause()
    plot.saveDataTxt('bbb.txt.bz2')
    # plot.saveGnuplot('bbb') is also possible
```

```
# collect history of data which will be plotted
def addPlotData():
  # each item is given a names, by which it can be the unsed in plot.plots
  # the **O.energy converts dictionary-like O.energy to plot.addData arguments
  plot.addData(i=O.iter,unbalanced=utils.unbalancedForce(),**O.energy)

# define how to plot data: 'i' (step number) on the x-axis, unbalanced force
# on the left y-axis, all energies on the right y-axis
# (O.energy.keys is function which will be called to get all defined energies)
# None separates left and right y-axis
plot.plots={'i':('unbalanced',None,O.energy.keys)}

# show the plot on the screen, and update while the simulation runs
plot.plot()

O.saveTmp()
```

Now the post-processing code, `our_post.py`:

```
from yade import post2d
import pickle

def extractMass(b):
    return b.state.mass

flattener = post2d.AxisFlatten(useRef=False,axis=2)

massData = post2d.data(extractMass,flattener)

r,x,y = massData['radii'],massData['x'],massData['y']

pickle.dump((r,x,y), open("rad_x_y_dump3.pickle","wb"))
```