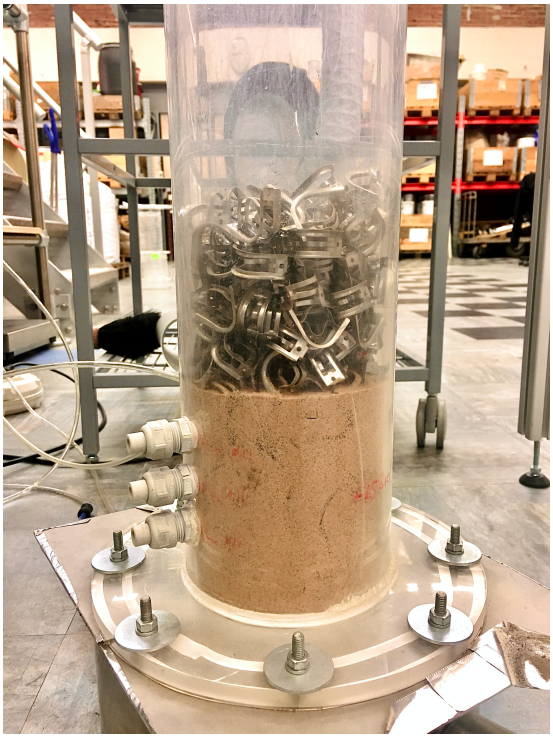
CHALMERS



Fluidiseringsegenskaper för packad-fluidiserad-bädd

En utvärdering av packningsmaterial

Kandidatarbete inom Energiteknik
SEEX15-20-1

MÅRTEN BENGTSSON
JULIA CRAMSTEDT
JESPER LARSSON
MAX WASSENIUS

INSTITUTIONEN FÖR
RYMD-, GEO- OCH MILJÖVETENSKAP

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

Abstract

When fluidised beds are used as Chemical Looping Combustion (CLC) reactors, they can be utilised to convert fuels into heat and power in a setup where CO_2 is easy to separate from the exhaust stream. This makes the process of collecting and storing CO_2 exhausts far more effective than the expensive methods of gas separation that are available today. A limiting factor with this setup is the gas-solid mass transfer in the bubbling regime, where the mass transfer is affected by phenomena such as slugging and channeling. To address this problem it has been hypothesised that introducing a packing material to the fluidised bed will decrease bubble size, resulting in improved gas-solid mass transfer and prevention of destructive fluidisation phenomena. The report aims to show the impact of adding packing materials to a fluidised bed, by showing results from experiments performed in a cylinder meant to replicate the environment of a fluidised bed reactor. Hiflow rings, RMSR 25-3, ASB packing and Raschig rings were tested for multiple ratios between sand and packing heights, and their performance was compared to each other through pressure drop calculations and visual observations for different superficial gas velocities.

RMSR 25-3 and Hiflow rings were determined to have the desired behaviour when put in a fluidised bed, as both packings were successful in reducing destructive fluidisation behaviour. They showed high potential in maintaining turbulent mixing and could handle two times the flow of air compared to Raschig rings and ASB packing. The reason for this is speculated to be the high voidage of RMSR 25-3 and Hiflow rings, along with their general geometry. It can be concluded that introducing RMSR 25-3 or Hiflow rings to a fluidised bed decreases bubble size in the emulsion phase, thus increasing the gas-solid mass transfer in the bubble regime.

Sammanfattning

Då en fluidiserad bädd används i kemiskt cirkulerande förbränning omvandlas bränsle till värme och elektricitet i en process där koldioxid i avgaserna enkelt går att separera, vilket är mycket gynnsamt i arbetet med lagring av koldioxid. En begränsande faktor med denna teknik är massöverföring mellan det syrebärande materialet och bränslet i den bubblande bädden. Massöverföringen påverkas negativt av fluidiseringsfenomen såsom sluggning och kanalbildning. Att addera packningsmaterial tillsammans med det fasta materialet i sanden har lagts fram som en eventuell lösning, i hopp om att packningsmaterialet kommer att reducera storleken på bubblorna och därmed förebygga störningar i fluidiseringen, samt bidra till en ökad masstransport i emulsionsfasen. Rapporten ämnar att visa effekten av att tillsätta packningsmaterial till en fluidiserad bäddreaktor, genom en kallflödesmodell i en plexiglascylinder. Hiflow-ringar, RMSR 25-3, ASB-packning och Raschigringar undersöktes olika förhållanden mellan sand- och packningshöjder och packningarnas prestationer jämfördes med varandra genom tryckfallsberäkningar och visuella observationer vid olika superficiella gashastigheter.

RMSR 25-3 och Hiflow-ringar var packningar som bedömdes ha eftertraktade beteenden när de adderades till den fluidiserade bädden, då båda packningarna lyckades minska destruktiva fluidiseringsfenomen. Båda packningarna uppvisade hög potential när det kom till att upprätthålla turbulent omblandning och de kunde hantera dubbelt så höga luftflöden som ASB-packning och Raschigringar utan att icke-gynnsamma fluidiseringsfenomen förekom. Anledningen till att de var överlägsna de andra packningarna föreföll bero på packningarnas höga tomrumsfraktioner. Rapporten drar slutsatsen att tillsats av RMSR 25-3 eller Hiflow-ringar till en fluidiserad bädd förebygger sluggning och minskar storleken på bubblor i emulsionsfasen, vilket leder till ökad masstransport i bubbelregionen.

Erkännande

Vi vill tacka examinator Magnus Rydén och handledare Nasrin Nemati som varit till stor hjälp i utformningen av arbetet och gett oss givande feedback längs vägen. Ett extra stort tack till Nasrin som har varit väldigt stöttande i experimentprocessen och alltid varit tillgänglig för att svara på frågor vilket vi uppskattat enormt. Vi hoppas att ni båda får användning av resultatet i rapporten i er fortsatta forskning.

English Title: Fluidisation Characteristics of a Confined Fluidised Bed - An Evaluation of Packing Materials

Innehållsförteckning

Nomenklatur

Symboler

A_{tv}	Inre tvärsnittsarea av cylindern	m^2
b	Bredd	mm
d	Diameter	
d_i	Diameter på hålen i sil i	μm
$\bar{d}_p$	Medelvärde av partiklarnas diameter	μm
$\bar{d}_{sand}$	Medelvärde av sandpartiklarnas diameter	μm
h	Höjd	
h_{sand}	Sandens höjd i cylindern	cm
$h_{packning}$	Packningens höjd i cylindern	cm
m	Massa	
m_i	Massa av sand i sil med sildiameter i	g
$m_{packning}$	Packningens massa	kg
m_{tot}	Totalmassa silad sand	g
n	Nominell storlek	mm
P	Tryck	
$P_{windbox}$	Relativa trycket i windbox	mbar
P_1	Relativa trycket närmast botten av bädden (h = 1cm)	mbar
P_2	Relativa trycket i mitten av bädden (h = 6.5 cm)	mbar
P_3	Relativa trycket i toppen av bädden (h = 12.1 cm)	mbar
ΔP	Tryckfall över hela bädden	mbar
Q	Gasens volymflöde	m^3/s
R	Förhållande mellan sandhöjd och packningshöjd	cm/cm
t	Tjocklek	mm
u	Superficiell gashastighet	m/s
u_{mf}	Minsta fluidiseringshastighet	m/s
V	Volym	
V_B	Bulkvolym	m^3
V_T	Volymen av tomrummet	m^3
ρ	Densitet	
$\bar{\rho}_p$	Partikeldensitet	g/cm^3
$\bar{\rho}_{packning}$	Packningens densitet	kg/m^3
Φ	Tomrumsfraktionen	m^3/m^3

Förkortningar

CLC	Chemical Looping Combustion
CCS	Carbon Capture and Storage

1 Introduktion och bakgrund

Med den konsensus kring effekterna av global uppvärmning som råder idag är det mer aktuellt än någonsin att hitta lösningar som kan minska människans koldioxidavtryck. Då en stor del av det globala koldioxidutsläppet kan kopplas till energiproduktion är detta område högst aktuellt för vidare utveckling. Som svar på detta har produktionen av energi från förnyelsebara källor som sol, vind och vatten ökat de senaste decennierna [1]. En energiproduktion som i huvudsak uppgörs av förnyelsebara energikällor möter dock ett problem i att energibehovet världen över har sett en stor ökning under kort tid. Att tillgodose det växande energibehovet med endast förnyelsebara energikällor år svårt på grund av höga kostnader, låg effektivitet och geografiskt läge. Därför uppförs fortfarande nya förbränningskraftverk och en stor majoritet av världens energiproduktion förblir inom den närmaste framtiden starkt beroende av förbränning [1].

Energiproduktionens behov av förbränning lägger stor vikt på innovation och utveckling av tekniker för att minska koldioxidutsläppen från förbränningskraftverk. Ett koncept som tagits fram som visar på god applicerbarhet på förbränningskraftverk, är Carbon Capture and Storage, CCS. Grundtanken bakom CCS är att fånga upp CO_2 från förbränningavgaser och transportera gasen till ett slutförvar. Dessa slutförvaringsplatser föreslås utgöras av exempelvis tömda oljereservoarer och naturgasfyndigheter, eller djupt belägna saltvattensakvifer där CO_2 kan injiceras in och förvaras utanför jordens yttre kolkretslopp [2]. För att effektivt utnyttja utrymme i slutförvaret krävs en hög renhet i avgaserna med avseende på CO_2.

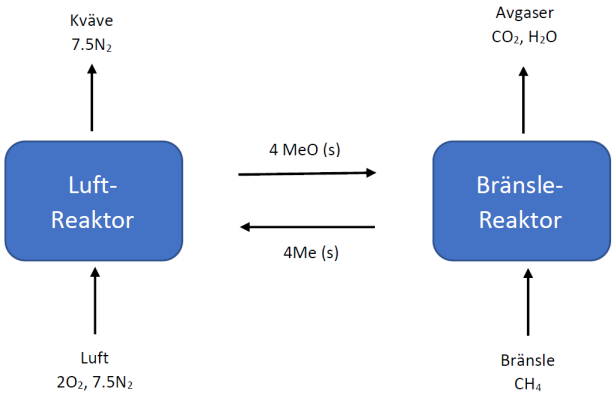
Då avgaserna från förbränningskraftverk i regel endast utgörs av 5-30% CO_2, krävs ett separationssteg för ökad renhet av CO_2 om målet är att samla in och lagra gasen. En sådan separation är kraftigt energikrävande då förbränningsavgaserna till en majoritet består av lättflyktig gas som kväve. Ett lovande alternativ, som i teorin tillåter absolut renhet med avseende på CO_2 i förbränningsavgaserna utan kostsamma separationssteg, är Chemical Looping Combustion, CLC. Uppsättningen av en CLC-reaktor, med in- och utflöden, kan ses i figur 1. Detta är en teknik som visats kunna appliceras på en cirkulerande fluidiserad bäddpanna och låter bränsle strömma igenom och fluidisera en bädd av syrebärande partiklar, oftast pulveriserade metalloxider [3]. Då bränslet pumpas igenom bädden underifrån oxideras det i en reaktion med metalloxiden, vilket bildar CO_2 och H_2O som slutprodukt enligt reaktionen:

$$CH_4 + 4\,MeO \longrightarrow CO_2 + 2\,H_2O + 4\,Me$$

där MeO betecknar en metalloxid som agerar syrebärare. Vanligast är oxider av Fe, Mn, Ni eller Cu [3]. Den bildade vattenångan från reaktionen kan enkelt separeras från koldioxiden genom kondensering, vilket ger ett rent utflöde av CO_2 då reaktionen endast sker med syre bundet till metalloxiden och i frånvaro av luft. För att åter oxidera bäddmaterialet slussas partiklarna vidare till en luftreaktor där de reagerar med syret i genomströmmande luft enligt reaktionen:

$$2\,O_2 + 4\,Me \longrightarrow 4\,MeO$$

innan de förs tillbaka till bränslereaktorn. En CLC-anläggning utgörs alltså av två separata, men sammankopplade reaktorer; en bränslereaktor där tillfört bränsle oxideras och en luftreaktor där syrebärande partiklar återoxideras.



Figur 1: Enkel illustration av CLC-processen.

En stor del av av tekniken bakom CLC har utvecklats nyligen och pilotanläggningar med naturgas och kol som

bränsle visar lovande resultat med avseende på omsättningsgrad av bränslet och koldioxidinsamling [4][5]. Utöver att användas vid förbränning av konventionella fossila bränslen kan CLC nyttjas vid förbränning av andra material, såsom avfall och biomassa. Användandet av biomassa som bränsle i en CLC-anläggning kan, på grund av växters förmåga att absorbera koldioxid, leda till en netto-minskning av CO_2 i atmosfären, om man kombinerar anläggningen med CCS. Flerpartssamarbetet *Negative CO_2 Emissions with Chemical Looping Combustion of Biomass* har med detta som bakgrund bildats mot målet att utveckla ny teknik som kan leda till förbränning med netto-negativa koldioxidutsläpp [3].

Då oxidationen av bränslet sker på ytan av metalloxidpartiklarna är massöverföring mellan gas och partikel en avgörande faktor i effektiviteten av en CLC-reaktor. Generellt gäller det för fluidiserade bäddar att en hastighetsökning av den genomströmmande gasen resulterar i större storlek på de bildade bubblorna i bädden. Vid vissa fall kan det leda till att fenomen som sluggning och kanalbildning uppstår. Förekomsten av dessa fenomen leder i olika grad till att kontaktytan mellan gas och partikel minskar. Då den största delen av massöverföringen sker i gränsskiktet mellan de två faserna [6]. har sluggning och kanalbildning negativa effektter på processens effektivitet.

För att kunna bibehålla mindre storlek av bildade bubblor och förebygga fenomen som sluggning vid högre gasgenomflöden har det föreslagits att addera packningsmaterial till bädden. Packningarna är tänkta att störa koaliseringen av bubblor som stiger genom bäddmaterialet, vilket förhindrar dem från att växa i storlek. När bubblorna hindras från att växa finns det mer utrymme för den fluidiserande gasen att existera i emulsionsfasen av bädden, där den har som störst kontaktyta med bäddmaterialet. Fluidiseringsegenskaperna av en packad-fluidiserad-bädd är ett relativt okänt område då det inte gjorts mycket forskning kring fluidiserade bäddar med adderad packning. Fortsatta tester kring packningsmaterials förmåga att påverka fluidiseringsfenomen i en fluidiserad bädd kan bidra till vidare förståelse av ämnet. Dessutom kan det visa sig ge värdefullt underlag för användning inom konventionella fluidiserade bäddar såväl som aktualiseringen av en fullskalig CLC-anläggning.

1.1 Syfte och problemformulering

Syftet med projektet är att undersöka om användandet av packning i en fluidiserad bädd ger gynnsamma fluidiseringsförhållanden. Dessa fluidiseringsförhållanden innefattar små och jämnt fördelade bubblor, vilket ger en bra kontakt mellan gas och det fluidiserade materialet, samt en frånvaro av sluggning, kanalbildning och vertikal segregation. De listade förhållandena jämförs med resultat från en icke-packad fluidiserad bädd för att avgöra huruvida packning är gynnsamt.

2 Teori

Följande del av rapporten avhandlar de teoretiska områden som krävs för att kunna ta till sig rapportens innehåll och slutsatser. Detta innefattar en beskrivning av en fluidiserad bädd, olika fludiseringsförhållanden, ingruppering av bäddmaterial, tryckfall och tryckvariationer över bädden samt packad fluidiserad bädd.

2.1 Fluidiserad bädd

En fluidiserad bädd består av en bädd av partiklar, oftast sand, som är placerad i en behållare där en gas strömmar upp underifrån genom bädden. Själva fluidiseringen uppstår när dragkraften på partiklarna är lika stor som partiklarnas vikt. Gashastigheten måste alltså vara tillräckligt hög för att lyfta partiklarna och kunna ta sig igenom bädden. Den punkt då fluidisation först uppstår kallas för minsta fluidiseringshastighet.

Det finns flera olika fenomen som kan förekomma i en fluidiserad bädd. Här fokuseras det på två fenomen för att beskriva en fluidiserad bädd på ett enkelt sett. Det första fenomenet, bubblande bädd, sker vid själva fluidiseringen av bädden när densitetsskillnader mellan gas och bäddmaterial gör att lokala hålrum i bädden skapas. Dessa bubblor av gas beror på att gashastigheten är högre än den som krävs för fluidisering, vilket skapar en typ av instabilitet i bädden.

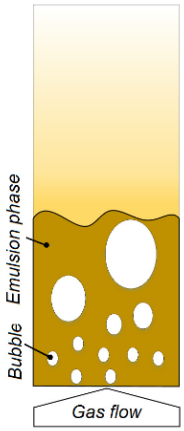
Nästa fenomen är vanligt förekommande i cirkulerande fluidiserade bäddpannor och innebär att gashastigheten

är så stor att partiklarna följer med gasen uppåt i reaktorn. I en cirkulerande fluidiserad bäddpanna så transporteras partiklarna tillbaka till bädden underifrån. Huruvida dessa fenomen uppstår och när de i så fall gör det beror på gasen och bäddmaterialets fysiska egenskaper, men även gasflödet och storleken på behållaren [7].

2.2 Gynnsamma fluidiseringsförhållanden

I kemiska reaktioner är massöverföringen mellan den reagerande gasen och aktiva partikeln vanligtvis en begränsande faktor [9]. I en fluidiserad bädd finns två områden; emulsionsfasen och bubbelfasen, se figur 2. Emulsionsfasen är det fluidiserade bäddmaterialet och bubbelfasen är bubblorna i bädden [8]. Det finns alltså gas i både emulsionsfasen och bubbelfasen, men i bubbelfasen är gasen mer samlad i en enhet.

För att få bra kontakt mellan gasen och bäddmaterialet behöver massöverföringsresistansen mellan faserna vara låg [9]. Fler bubblor med mindre diameter ökar kontaktytan mellan faserna och borde därför i teorin öka omsättningen, eftersom mer aktivt material kan komma i kontakt med den reagerande gasen.



Figur 2: Visualisering över emulsions- och bubbelfasen [8].

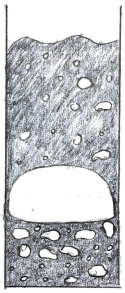
2.3 Icke gynnsamma fluidiseringsfenomen

De fluidiseringsfenomen som är vanligt förekommande i fluidiserade bädd-reaktorer är kanalbildning och sluggning, vilka båda är oönskade och effektivitetshämmande skeenden.

Kanalbildning innebär att en majoritet av gasflödet tar en specifik väg genom bädden. Ett exempel är att gasen flödar upp i en kant av reaktorn. Kanalbildning uppstår för att gasen naturligt strävar efter den väg med lägst tryckfall. Finns det en sådan väg i bädden kommer gasen välja den och bilda en kanal. Att flödet strömmar på en väldigt begränsad del av reaktorns totala bäddvolym ger partiklar och gas sämre möjlighet att interagera med varandra på ett tillfredsställande sätt. Se figur 3 för en enkel illustration.

Sluggning innebär att mängden bubblor i bubbelfasen ökar och växer i storlek så att de nästan fyller hela tvärsnittsarean av bädden [10]. Sluggning kan göra att sandbädden separerar och delen som är ovanför bubblan transporteras en sträcka vertikalt, för att sedan falla tillbaka ner när bubblan spricker. Se figur 4 för en enkel illustration av sluggning.

Figur 3: Kanalbildning.



Figur 4: Sluggning.

Huruvida sluggning uppstår eller inte beror på bäddens höjd, h, i förhållande till bäddens diameter, D . I en bädd med större diameter är det mindre sannolikt att sluggning uppstår. Partikelstorleken har också påverkan på om sluggning uppstår, där små partiklar är mindre benägna att uppvisa sluggning [11]. Förekomsten av sluggning är väldigt påfrestande för utrustningen och försämrar masstransporten i och med storleken på bubblorna.

2.4 Gruppering av partiklar

I en fluidiserad bädd används oftast en sand som bäddmaterial. För att skilja partiklar och deras fluidiseringsegenskaper åt presenterade D. Geldart år 1973 fyra grupper av partiklar; Grupp A, B, C och D, för att kunna skilja på deras egenskaper.

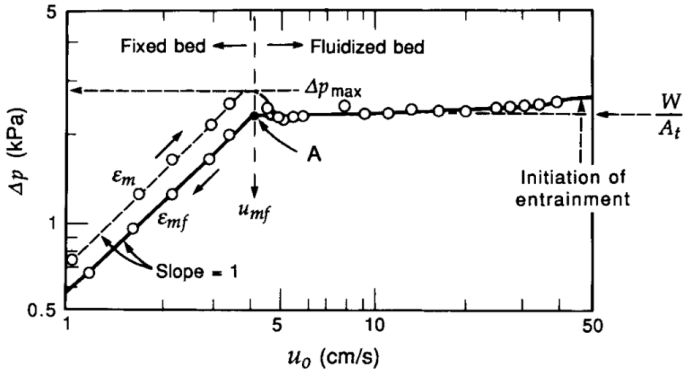
Grupp A innebär en medeldensitet på partiklarna, $\bar{\rho}_p$, på mindre än 1.4 g/cm^3 och med en medeldiameter, $\bar{d}_p$, mellan 20-100 μm [12] och kan beskrivas som ett fint pulver.

Geldart grupp B innefattar de flesta material som har en medeldiameter på partiklarna inom spannet 40 $\mu m < \bar{d}_p < 500$ μm, och en medeldensitet inom spannet 1.4 $g/cm^3 < \bar{\rho}_p < 4.0$ g/cm^3. Generellt skapas det bubblor väldigt nära den minimala fluidiseringshastigheten i bäddar med Grupp B-partiklar [13]. Geldart grupp A och B är vanligt att använda i fluidiserade bäddar med förbränningsreaktioner [14].

Grupp C innefattar väldigt fina, nästan puderaktiga, partiklar inom spannet $20 < \bar{d}_p < 30$ μm. Grupp D har en medeldiameter över 600 μm och en hög partikeldensitet [12].

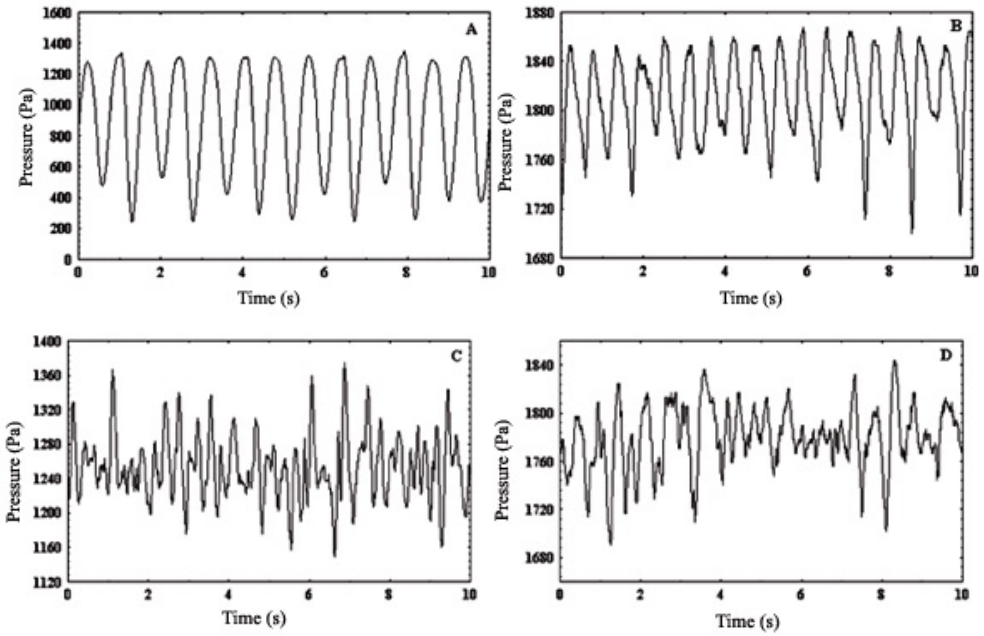
2.5 Tryckfall och tryckvariationer

I figur 5 visualiseras tryckfallet mot gasflödeshastigheten (volymhastighet genom tvärsnittarean). Här kan det noteras att tryckfallet bör vid minsta fluidiseringshastigheten, u_{mf} i figur 5, uppnå en högsta punkt och sedan plana ut. Det innebär alltså att när bädden befinner sig i bubbelområdet är tryckfallet över bädden konstant. Figuren nedan visar på en bra uppträdande bädd som befinner sig i bubbelområdet, förutsatt att tryckvariationerna i varje punkt inte är för stora [15].



Figur 5: Tryckfall, ΔP, mot gasflödeshastigheten, u_0, för en bädd med Geldart Grupp B-partiklar [15].

Stora tryckvariationer, figur 6A. visar på hur sluggning påverkar trycket i en fludiserad bädd. Här används partiklar tillhörande Geldart grupp B och ger en variation på cirka 600 *Pa*, eller 6 *mbar*. För en bädd med partiklar från Geldart grupp B som befinner sig i enbubbelfasen, där flera små bubblor slås ihop till en större, visualiseras tryckvariationerna i figur 6B. Den bildade bubblan i sig är inte tillräckligt stor för att kallas för sluggning. Amplituden varierar med ungefär 1.3 *mbar*.



Figur 6: Tryckvariation mot tiden för en bädd (A) med sluggning (Geldart grupp B), (B) i enbubbelfasen (Geldart grupp B), (C) i flerbubbelfasen (Geldart grupp A) och (D) i flerbubbelfasen (Geldart grupp B) [16].

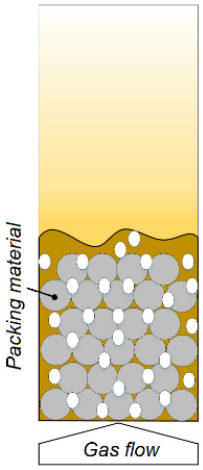
I bubbelregimet med flera bubblor visualiserar tryckvariationen i figur 6C för partiklar i Geldart grupp A och i figur 6D för partiklar i Geldart grupp B. Jämfört med figur 6A och 6B så är variationen i amplituden mycket mindre och mer irreguljära för 6C och 6D samt att det sker fler tryckvariationer per tidsenhset [16]. Värt att notera är det även att amplituden för 6A är markant större än 6B.

2.6 Packad fluidiserad bädd

Packad fluidiserad bädd innebär att solida material med betydligt större diameter än de solida partiklarna i emulsionsfasen finns i bädden, se figur 7. Dessa material benämns som packningsmaterial, och är vanligt förekommande i t.ex. destillationskolonner [17].

Idén med att ha packning i den fluidiserade bädden är bland annat att det skulle kunna minska bubbelstorleken och därmed öka massöverföringen mellan de två faserna. Det skulle också minska bubblornas maximala storlek.

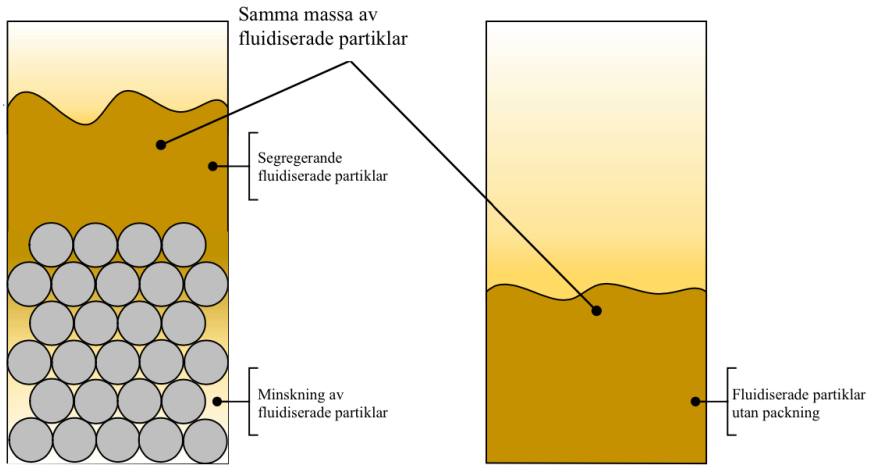
Tidigare studier visar på att metoden verkar vara effektiv. I en artikel av Jesper Aronsson så påvisar packning i en fluidiserad bädd bland annat på minskad bubbelstorlek, ökad massöverföring mellan gas och solid samt ett högre tryckfall över bädden [8][18]. I en tidig studie från 60-talet visar packad fluidiserad bädd på en större bäddexpansion i jämförelse med vanlig fluidiserad bädd, samt en minskning i tryckfall då den fludiserade bädden expanderar över packningen [19].



Figur 7: Visualisering av en fluidiserad bädd med packning [8].

2.6.1 Vertikal segregation

Vertikal segregation innebär att två separata faser bildas i den solida fasen, oftast att packningen och sanden separeras från varandra och att det då uppstår ett skikt med mycket liten andel bäddmaterial i botten av cylindern, se figur 8 [18].



Figur 8: Visualisering av vertikal segregation [18] / Orginaltext översatt till svenska.

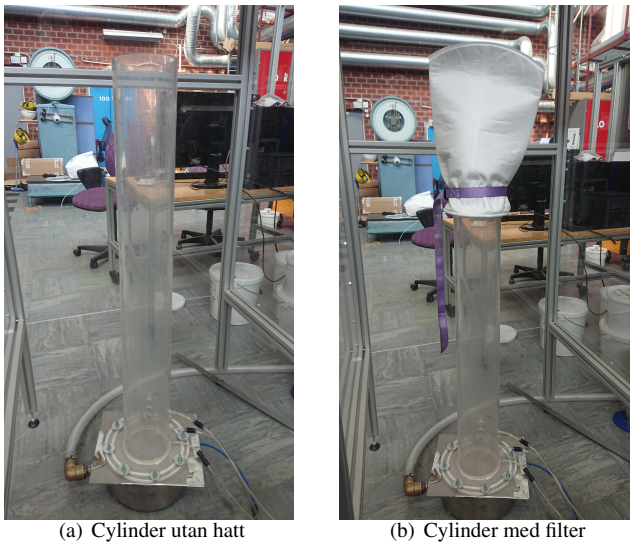
I figur 5 så kan det noteras att tryckfallet börjar gå uppåt igen vid högre gashastighet. Detta innebär att bäddpartiklar börjar dras med uppåt i bädden, vilket skulle kunna tyda på vertikal segregation alternativt att partiklarna dras med gasen, som i en cirkulerande fluidiserad bädd. Det kan även bero på bäddexpansion, det vill säga att bäddhöjden ökar i och med fluidiseringen.

3 Metod

Nedan följer en redogörelse för den apparatur som användes under experimenten, samt en beskrivning av den arbetsmetod som ligger till grund för projektets experiment och analys och projektets avgränsningar.

3.1 Apparatur

Experimenten som ligger till grund för rapporten genomfördes i en cylindrisk kallflödesreaktor i plexiglas, med en innerdiameter på 12 *cm* och en höjd på 1 *m*, enligt Figur 9(a). Figuren illustrerar cylindern utan sand eller packning närvarande.



(a) Cylinder utan hatt (b) Cylinder med filter

Figur 9: Bilder på cylindern som användes vid experimenten.

Anordningen bestod av en vindbox som var placerad under cylindern där luft från experimenthallens kompressorsystem pumpades in, se Figur 10. Experimenten använde två olika källor för luftflöde för att pumpa in luft; MFC FR, och MFC AR. Bild på massflödesswitch hittas i Bilaga C. MFC FR användes för lägre flöden, och MFC AR för högre. Bytet av massflödeskälla skedde runt 0.55 m/s. Luften transporterades från vindboxen till cylinderns botten genom en porös metallplatta. Fyra HUBA Control sensorer vars syfte var att mäta tryck fanns placerade i systemet. En positionerad i vindboxen, och de övriga tre högre upp i bädd-delen av cylindern på höjderna 1 cm, 6.5 cm respektive 12.1 cm ovanför vindboxen. Bilder på sensorernas placering finnes i Bilaga B. Värdena som gavs av sensorna digitaliserades genom en NiDAQ A/D omvandlare. Luftflödet styrde genom en Bronkhorst massflödesmätare av programmet LabView^{TM}. Försöken kördes kallt, det vill säga ingen förbränningsreaktion var aktuell under experimenten.

Figur 10: Bild på vindbox.

Användandet av utrustningen ledde till att små sandpartiklar frigjordes i luften, vilket kan ses som en hälsorisk vid utdragen exponering och långvarig inandning. För att förhindra inandning av partiklar placerades därför ett filter på cylinderns topp, se figur 9(b), och en extern dammsugare fanns i anslutning till cylindern i syfte att suga upp partiklar som trängde igenom filtret. Bilder på dessa finns att hitta i Bilaga C.

3.2 Sand

För experimenten användes två typer av sand med olika medeldiameter. Ett krav som ställdes på sanden var att den skulle tillhöra Geldart Group B, och för att få en mer enhetlig sand med kontrollerat intervall på diametern genomfördes silning av sanden i en silningsapparatur, se Bilaga B. Detta gjordes endast för Sand I. Medeldiametern för Sand I var 240.6 μm och dess densitet var 1.45 g/cm^3. Sand II silades endast för att kontrollera dess medeldiameter. Den mättes att ha en medeldiameter på 574.6 μm och en densitet på 1.40 g/cm^3 .

Följande ekvation användes för att beräkna medeldiametern hos sandpartiklarna:

$$\bar{d}_{sand} = \frac{\sum m_i d_i}{m_{tot}}$$

Där m_i är massan sand som fastnar i varje silsteg och d_i diametern på hålen i sil i. m_{tot} är totala massan silad sand.

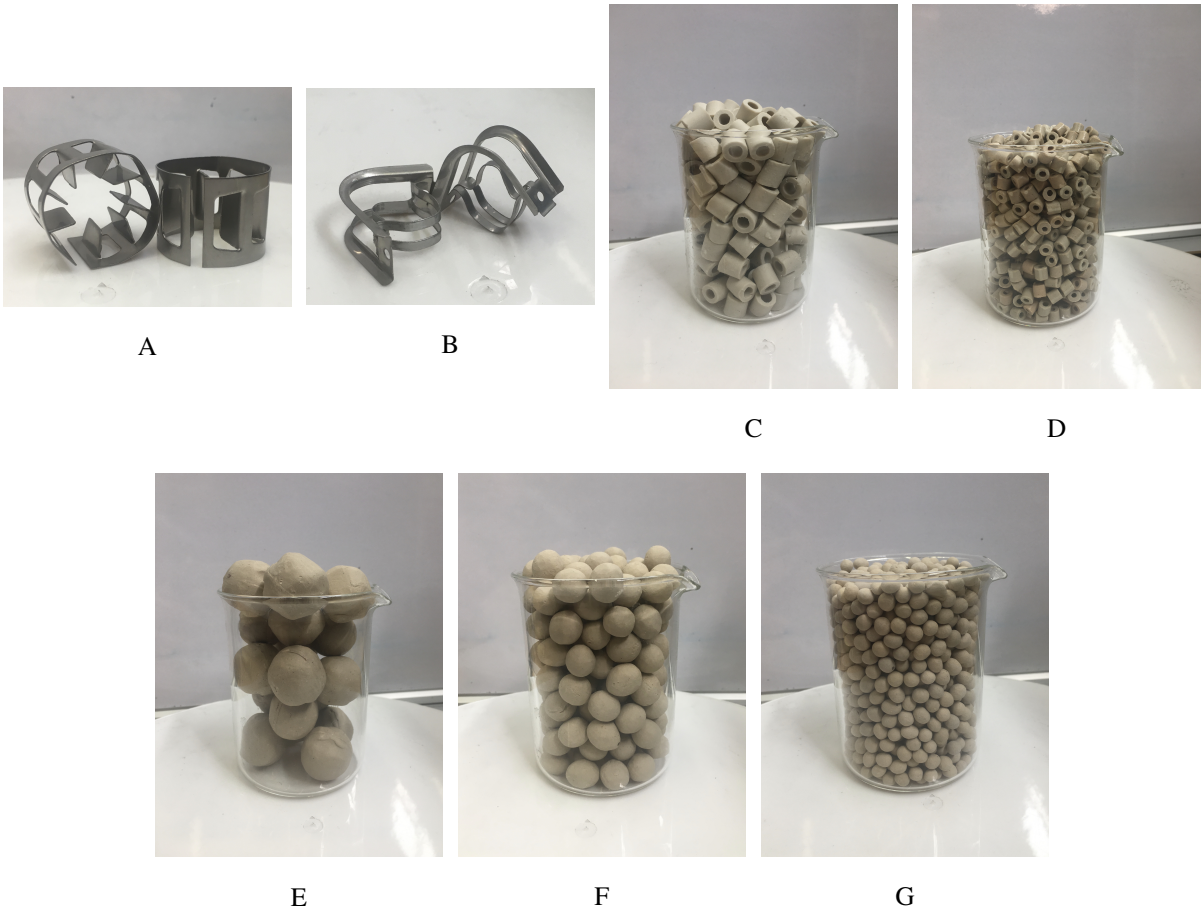
3.3 Packningar

I tabell 1 finnes information om alla packningar som fanns tillgängliga för experimenten. I kolumnen ”Storlek” används vissa förkortningar enligt följande: n = nominell storlek, d = diameter, t = tjocklek, b = bredd. Viktigt att notera är att all data om storlek är hämtad från tillverkare. För att beräkna tomrumsfraktionen användes ekvation 1 och för densiteten användes ekvation 2, båda vilka finnes i avsnitt 3.3.1.

Tabell 1: Information om tillgängliga packningar

Namn	Material	Storlek	Densitet	Tomrumsfraktion
Hiflow-ringar 25-5 [20]	Rostfritt stål	n25mm, t0.5mm	280 kg/m^3	95 %
RMSR 25-3 [21]	Rostfritt stål	n25mm, t0.3mm	204 kg/m^3	96 %
Raschigringar 10x10 [22]	Keramik	b10mm, d10mm	890 kg/m^3	58 %
Raschigringar 6x6 [22]	Keramik	b6mm, d6mm	1110 kg/m^3	50 %
ASB 1” [18][23]	Aluminiumsilikat	d25.4mm	1340 kg/m^3	44 %
ASB 1/2” [18][23]	Aluminiumsilikat	d12.7mm	1390 kg/m^3	42 %
ASB 1/4” [18][23]	Aluminiumsilikat	d6.35mm	1395 kg/m^3	41 %

Bilder på alla packningar finnes i figur 11.



Figur 11: Packningar som fanns tillgängliga för experimenten (A) Hiflow-ringar 25-5, (B) RMSR 25-3, (C) Raschigringar 10x10, (D) Raschigringar 6x6, (E) ASB 1”, (F) ASB 1/2”, (G) ASB 1/4”.

Av de packningar som fanns tillgängliga användes alla förutom Raschigringar 6x6 och ASB 1/4”, figur 11D respektive figur 11G, i experimenten.

3.3.1 Beräkning av tomrumsfraktion samt densitet

Tomrumsfraktionen, ϕ, definieras på som i ekvation 1

$$\phi = \frac{V_T}{V_B} \tag{1}$$

där V_T är volymen av tomrummet och V_B är bulkvolymen, båda med enheten m^3.

För att beräkna tomrumsfraktionen användes en cylindrisk bägare med liknande diameter som cylindern för experimenten. Bägaren vägdes tom, full med packning, full med packning och vatten samt endast med vatten. Från detta kunde volymen av tomrummet samt bulkvolymen, det vill säga volymen av bägaren, tas fram med hjälp av vattnets densitet som antogs vara 997 kg/m^3. Densiteten av packningen, $\rho_{packning}$, kunde också räknas fram genom att använda datan för bulkvolymen samt packningens vikt enligt ekvation 2

$$\rho_{packning} = \frac{m_{packning}}{V_B} \tag{2}$$

där packningens massa har enheten kg och bulkvolymen har enheten m^3.

3.4 Definition av förhållande mellan sand och packning

Under experimenten testades olika förhållanden mellan sand och packning. Förhållandet, R, definieras på följande sätt:

$$R = \frac{h_{sand}}{h_{packning}}$$

Detta förhållandet bestämdes genom att massan packningsmaterial $m_{packning}$ som krävdes för att uppnå ungefär 12 cm i höjd i kolonnen mättes. Sedan bestämdes m_{sand} då hålrummet mellan packningen fylldes upp till ungefär 12 cm. Dessa massor användes för förhållandet då följande antagande gjordes:

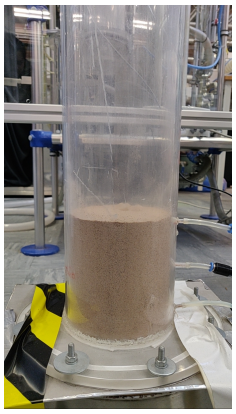
$$h_{packning} \sim m_{packning}$$

det vill säga att dubbel massa skulle ge dubbel packningshöjd. Massan av packning och sand bestämdes för varje individuell typ av packning.

3.5 Experiment - Packad fluidiserad bädd

Försöken undersökte för olika packningsmaterial och R hur den packade fluidiserade bädden svarade mot varierande luftflödeshastigheter.

Den initiala bäddhöjden, det vill säga höjden på sanden, som hålls som standard för försöken var 12 cm, se figur 12. Eftersom packningsmaterialen har varierande tomrumsfraktioner och geometrier kom därför vikten på sanden som användes inte att vara konstant i alla försök. Det var initiala bäddhöjden som kom att hållas konstant till 12 cm.



Figur 12: Bild på 12 cm bäddhöjd av sand.

Luftvolymflödet i försöken stegrades från 0.01 m/s upp till ett flöde där det visuellt bedömdes vara utan mening att fortsätta. Detta flöde varierade beroende på prestation hos packningen. Ett detaljerat stegringschema som efterföljdes i varje experiment hittas i Bilaga B. Under stegringen antecknades visuella betraktelser av bäddens beteende där mycket fokus låg på att identifiera störningar i fluidiseringen hos bädden. Framförallt sluggning och kanalbindning var av intresse. Höjden hos bädden antecknades också för att kunna jämföra sandbäddens förmåga att expandera med de olika packningarna.

För varje stegring av luftflödet genomfördes en tryckmätning för bädden där mätning gjordes under en minut för varje steg. Trycken som uppmättes av sensorerna var relativt atmosfärstrycket.

De olika packningarna provades på identiskt sätt, och tester gjordes med R = 1, R = 0.5 och R = 0.25. En detaljerad standardiserad laborationsmetod för försöken hittas i Bilaga A.

3.5.1 Icke-packad fluidiserad bädd

Experiment för bädd utan packning närvarande genomfördes för både Sand I och Sand ii. Testerna utfördes på identiskt sätt som experimenten med packning närvarande. Sand I testades för bäddhöjd 12 cm, och Sand II testades för 12 cm respektive 30 cm.

3.5.2 Djup bädd

Experiment med en ökad bäddhöjd, 30 cm, genomfördes för Sand II. Experimenten genomfördes på identiskt sätt som fallet med 12 cm, med skillnaden att endast RMSR 25-3 testades.

3.5.3 Definition av superficiell gashastighet

För experimenten användes en superficiell gashastighet som beräknas på följande sätt

$$u = \frac{Q}{A_{tv}}$$

där u är den superficiella hastigheten för gasen i m/s, Q är volymflödet av gasen i m^3/s och A_{tv} är tvärsnittsarean för insidan av cylindern i m^2.

3.6 Sammanställning av data

För att kunna analysera den data som gavs av programmet NI Labview^{TM} skapades programmet `BachelorThesisSEEX15-20-1.m` i `MATLAB`-sviten. Vid varje körning av gasgenomströmning, exempelvis 0.03 m/s, tog programmet Labview^{TM} fram trycket vid fyra olika punkter, $P_{windbox}$, P_1, P_2, och P_3 med en sampling rate på 50, d.v.s. att vid varje sekund som mätningen var igång togs det 50 datapunkter vid varje sensor. Programmet sparade 47 olika mätvektorer totalt. Detta gav i genomsnitt en matris med 47 kolumner och 2500-6000 rader. För att få datan sorterad i naturlig ordning och ta ut rätt tryck vid rätt punkt användes programmen `BachelorThesisSEEX15-20-1.m` och `natsortbib.m`[24], Bilaga E.1 och E.3.

All data sorterades i rätt ordning och medelvärdet för varje trycksensor under mätningen togs. Programmet gav sedan en excel-fil där all data kunde analyseras vidare. För vidare analys skrevs även kod som tog ut rätt typ av packning och rätt hastigheter. Denna delen av koden gjorde en ny matris som bara innehöll tryckvärden för en typ av packning vid ett visst packning/sand-förhållande. Analys av data gjordes med hjälp av figurer och en enklare approximation av minsta fluidiseringshastigheten.

3.7 Avgränsningar

Modellen som används är endast en kallflödesmodell, det vill säga att ingen värmetransport betraktas. Det är endast luft som strömmar genom sandbädden och ingen reaktion sker. Dessutom är projektet på en liten skala, vilket gör att en uppskalning eventuellt inte skulle ge samma resultat. Det bör tas i beaktning att analysen gjorts mycket baserat på observationer och inte i stor mängd på mätdata.

En annan avgränsning var att endast två stycken sandstorlekar användes, varav båda befann sig i Geldart Grupp B. Dessa ansågs vara speciellt lämpade då det är dessa som brukar se störst användning inom cirkulerande fluidiserade bäddpannor. Generellt lades inte så stor vikt på valet av sand, utan packningen var det som var av intresse. Ingen särskild hänsyn till bäddexpansion har tagits utöver enkla observationer.

Alla packningsmaterial testades inte med de båda sandstorlekarna. Som en utgångspunkt testades packningsmaterialen till en början endast med Sand I. Beroende på deras prestation i dessa tester valdes de mest lovande packningsmaterialen ut för att testas vidare med Sand II för att undersöka om deras prestation påverkades av sandvalet.

4 Resultat

Följande del av rapporten presenterar de resultat som den experimentella och analytiska delen av undersökningen genererat. Presentationen sker i form av grafer som relaterar parametrar mot varandra, och genom rapportering av de observationer som noterats i labbjournal under experimenten.

4.1 Sammanfattad observationsdata

Nedan presenteras vikter och hastighetsdata tagen från de experiment som utförts för Sand I, Sand II samt djup bädd-körning.

4.1.1 Sand I

I tabell 2 presenteras den minsta superficiella hastighet för vilket fluidisering observerades för Sand I, samt högst testade superficiella gashastigheten och mängden sand som krävdes för att fylla upp till 12 *cm* i cylindern. Ingen minsta fluidiseringshastighet angavs för Raschigringar med förhållande R = 0.5 då den aldrig observerades.

Tabell 2: Observerad minsta fluidiseringshastighet, sandvikt och högst testade superficiella gashastigheten för packningsmaterial vid olika sand/packningsförhållande för Sand I.

	Packnings-material	**Minsta fluidiserings-hastighet (m/s)**	**Högsta testade superficiell hastighet (m/s)**	**Vikt sand (g)**
$R = 1$	Icke-packad	0.04	0.35	2237.5
	RMSR 25-3	0.05	0.35	2100.4
	Hiflow-ringar	0.05	0.55	2225.8
	Raschigringar 10x10	0.04	0.55	1349.1
	ASB 1”	0.03	0.55	1101.4
	ASB 1/2”	0.04	0.95	841.9
$R = 0.5$	RMSR 25-3	0.05	0.90	2100.4
	Hiflow-ringar	0.06	1.00	2225.8
	Raschigringar 10x10	-	0.75	1349.1
	ASB 1”	0.03	1.00	1101.4
	ASB 1/2”	0.04	1.00	798.8
$R = 0.25$	RMSR 25-3	0.05	0.95	2100.4
	Hiflow-ringar	0.05	1.00	2225.7

Fluidiseringsfenomen som uppstod under experimenten noterades vid de hastigheter då de förekom. I tabell 3 visas hastighetsintervallen för eventuella fenomen som uppstod för de olika packningsmaterialen. Intervallet är angivet i enheten av den superficiella gashastigheten, m/s. Streck i tabellen indikerar att fenomenet inte observerades vid försöket.

Tabell 3: Observerade fluidiseringsfenomen vid uppmätt hastighetsintervall för alla packningar och packningsförhållanden för Sand I. Hastigheter är angivna i m/s.

	Packning	**Kanalbildning**	**Slugging**	**Vertikal segregation**	**Bubbelformation**
$R = 1$	Icke-packad	—	0.30-0.35	—	0-0.35
	RMSR 25-3	—	—	—	0-0.35
	Hiflow-ringar	—	0.55-0.55	—	0-0.55
	Raschigringar 10x10	0.07-0.55	0.45-0.55	0.10-0.55	0-0.55
	ASB 1”	—	0.40-0.55	0.40-0.55	0-0.34
	ASB 1/2”	0.50-0.75	—	0.50-0.75	0-0.50
$R = 0.5$	RMSR 25-3	0.90-0.90	—	0.80-0.90	0-0.90
	Hiflow-ringar	—	—	0.70-1.00	0-1.00
	Raschigringar 10x10	0.10-0.75	0.64-0.75	0.10-0.75	0-0.45
	ASB 1”	—	0.64-1.00	0.45-1.00	0-0.34
	ASB 1/2”	0.40-0.60	—	0.30-0.60	0-0.49
$R = 0.25$	RMSR 25-3	—	—	0.70-1.00	0-0.85
	Hiflow-ringar	—	—	0.92-1.0	0-0.60

4.1.2 Sand II

De experiment som genomfördes med Sand II utfördes med packningsförhållandena R = 0.5 och R = 0.25 för RSMR 25-3 och Hiflow-ringar. Minsta fluidiseringshastighet, den högsta testade superficiella hastigheten samt vikten sand som användes för olika sand/packningsförhållanden för Sand II presenteras i tabell 4. Starthöjden för Sand II i experimentet för Hiflow-ringar uppmättes till 12 *cm* och för RMSR 25-3 till 14 *cm*.

Tabell 4: Observerad minsta fluidiseringshastighet, sandvikt och högst testade superficiella gashastigheten för packningsmaterial vid olika sand/packningsförhållande för Sand II.

	Packnings-material	**Minsta fluidiserings-hastighet (m/s)**	**Högsta testade superficiell hastighet (m/s)**	**Vikt sand (g)**
–	Icke-packad	0.20	0.65	2309.0
$R = 0.5$	RMSR 25-3	0.25	1.47	2626.3
	Hiflow-ringar	0.20	1.47	2068.0
$R = 0.25$	RMSR 25-3	0.30	1.62	2752.0
	Hiflow-ringar	0.25	1.40	2068.0

Tabell 5 visar intervallen då fluidiseringsfenomen observerats för RMSR 25-3 och Hiflow-ringar vid försök med Sand II.

Tabell 5: Observerade fluidiseringsfenomen vid uppmätt hastighetsintervall för alla packningar och packningsförhållanden. Data för Sand II. Hastigheter är angivna i m/s.

	Packning	**Kanalbildning**	**Slugging**	**Vertikal segregation**	**Bubbelformation**
-	Icke-packad	—	0.45-0.65	—	0-0.65
$R = 0.5$	RMSR 25-3	—	—	0.20-1.47	0-1.47
	Hiflow-ringar	1.20-1.47	—	0.7-1.47	0-1.47
$R = 0.25$	RMSR 25-3	1.25-1.62	—	1.2-1.62	0-1.62
	Hiflow-ringar	1.40-1.40	—	0.70-1.4	0-1.40

4.1.3 Djup bädd

Tabell 6 visar minsta fluidiseringshastigheter, högst testade superficiella gashastigheten och sandvikt för experiment körda med Sand II under djup bädd-körning.

Tabell 6: Observerad minsta fluidiseringshastighet, sandvikt och högst testade superficiella gashastigheten för packningsmaterial vid djup bädd-körning för Sand II.

	Packnings-material	**Minsta fluidiserings-hastighet (m/s)**	**Högsta testade superficiell hastighet (m/s)**	**Vikt sand (g)**
$R = 1$	Icke-packad djup bädd	0.20	0.60	5177.1
	RMSR 25-3 djup bädd	0.25	0.80	4903.2
	Hiflow-ringar djup bädd	0.25	1.40	2626.3
$R = 0.5$	RMSR 25-3 djup bädd	0.25	1.00	4886.8

Tabell 7 visar hastighetsintervallen då eventuella fenomen uppstod för RMSR 25-3 djup bädd, då tester med Sand

II kördes. För jämförelse är även icke-packad djup bädd tabellerad.

Tabell 7: Observerade fluidiseringsfenomen för djup bädd-körning, Sand II. Hastigheter är angivna i m/s.

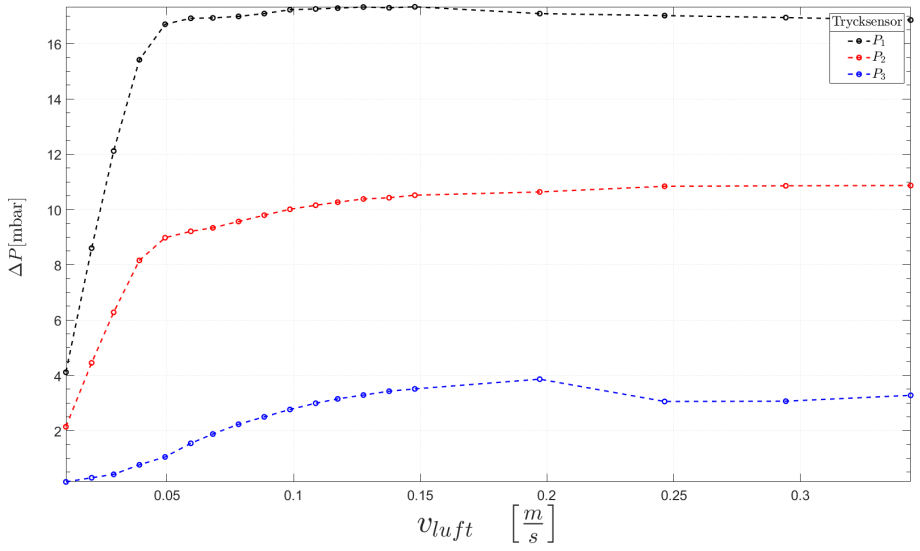
	Packning	**Kanalbildning**	**Slugging**	**Vertikal segregation**	**Bubbelformation**
$R = 1$	Icke-packad djup bädd	—	0.20-0.60	—	0-0.60
	RMSR 25-3 djup bädd	—	0.45-0.80	—	0-0.35
$R = 0.5$	RMSR 25-3 djup bädd	—	0.50-1.00	—	0-1.47

4.2 Resultat Sand I

Sand I var det bäddmaterial som användes till de flesta av försöken och testades för alla bäddmaterial vid förhållandena R = 1 och R = 0.5. Den kördes även utan packning i en icke-packad bädd och i försök vid förhållandet R = 0.25 för Hiflow-ringar och RMSR 25-3.

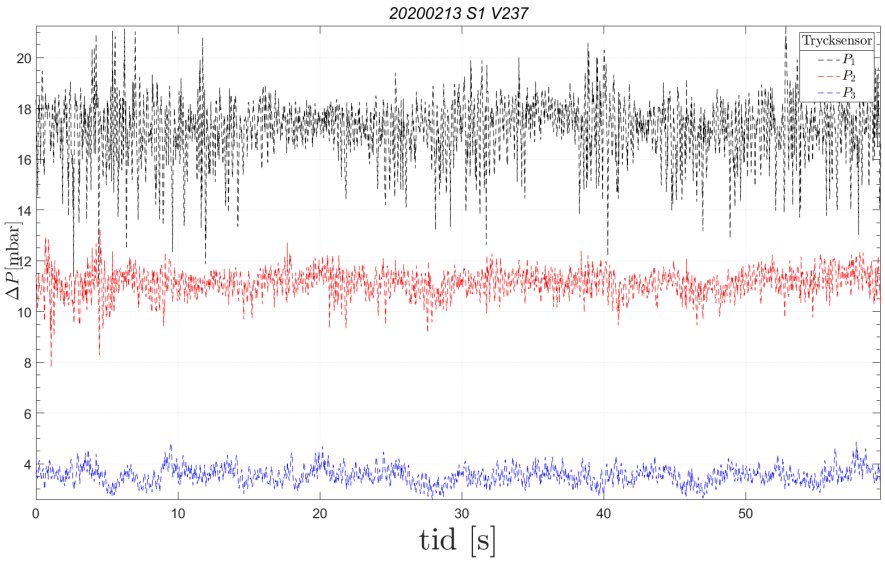
4.2.1 Icke-packad fluidiserad bädd

För utvärdering av packningsmaterialens påverkan på fluidiseringsfenomen utfördes försöken för en fluidiserad bädd utan packning med avsikt att uppföra standardvärden och fenomen för jämförelse. Figur 13 visar hur trycket beror på ökad superficiell gashastighet i den icke-packade fluidiserade bädden för Sand I. Grafen visar hur trycket ökar olika mycket vid olika avstånd från inströmning av luft där P_1 är trycket närmast botten, P_2 i mitten och P_3 överst. Trycket är relativt atmosfärstryck, övertryck.



Figur 13: Tryckskillnad vid ökad superficiell gashastighet för icke-packad bädd.

Den icke-packade bädden uppvisade mycket bubblor i sanden under experimentets gång och en turbulent yta med stora bubblor från hastigheter över 0.20 m/s. Kring 0.30 m/s började sluggning observeras. Figur 14 visar hur trycket varierar under samplingstiden för icke-packad bädd för Sand I.



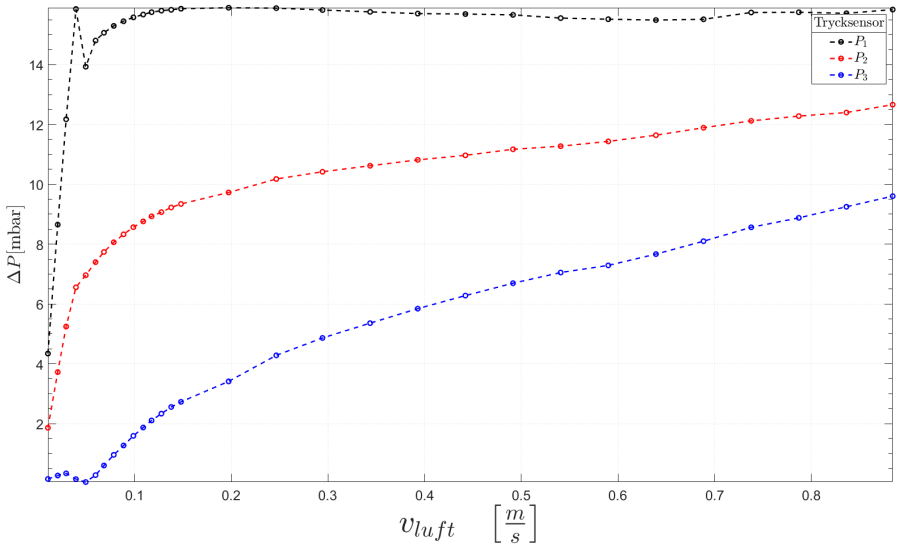
Figur 14: Tryckändring under samplingstiden för Sand I, v_{luft} = 0.35 m/s.

4.2.2 RMSR 25-3

Förhållandet **R = 1** fluidiserade jämnt vid lägre hastigheter och uppvisade något mindre bubblor än för icke-packad bädd. Vid runt 0.08 m/s kunde packningen inte längre synas vid sandytan på grund av bäddexpansion. Efter detta började sanden fluidisera över packningen likt en icke-packad bädd, men fortsatt jämnt. Vid 0.25 m/s var ytan väldigt turbulent, se figur 17. Vid högre hastighet växte bubblorna i storlek och kring 0.30 m/s observerades de förekomma irreguljärt med ökad intensitet.

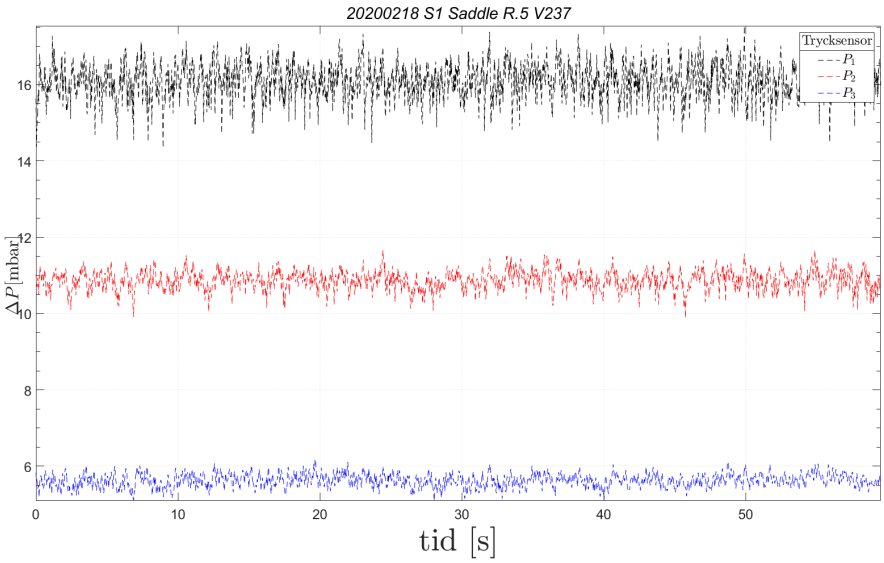
För **R = 0.5** fluidiserades bädden jämnt likt den gjorde för R=1. Fluidiseringen skedde med gradvis bäddexpansion fram till 0.40 m/s där sanden började stänka ovanför packningen och en turbulent yta kunde observeras. När 0.65 m/s nåddes hade sandbädden expanderat så att packningen inte längre var möjlig att se. Med expansion ovanför packningen var ytan inte längre lika turbulent, utan fortsatte med en jämn fluidisering. Vid 0.70 m/s började ytan bli turbulent. Flödet 0.90 m/s gav upphov till ett observerat luftflöde runt individuella packningar som liknades till någon lokal kanalbildning. Sandpartiklar vid ytan började bete sig gasformigt runt, se figur 18, och kunde observeras som en grumlig luft ovanför packningen.

Figur 15 visar hur trycket beror på ökad superficiell gashastighet för packningen RMSR 25-3, R = 0.5, Sand I.



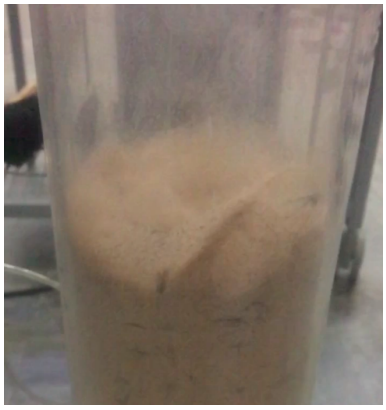
Figur 15: Tryck vid ökad superficiell gashastighet för RMSR 25-3, R=0.5.

Figur 16 visar hur trycket förändras under samplingstiden för RMSR 25-3 R=0.5 . Trycket hos sensorn närmast gasinflödet oscillerar mer än de längre ifrån gasinflödet.

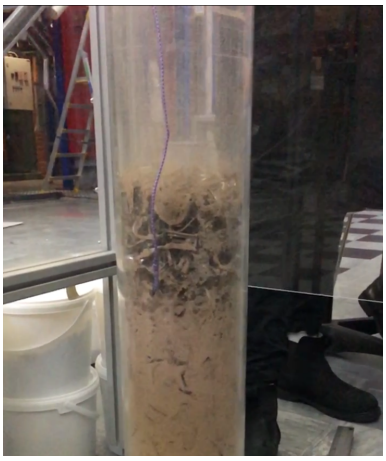


Figur 16: Tryckändring under samplingstiden för RMSR 25-3, R=0.5, $v_{luft} = 0.35\ m/s$.

Fluidiseringen vid förhållandet **R = 0.25** bettede sig likt försöket för R=0.5 i avseende på bäddexpansion och bubbelstorlek. Vid hastigheten 0.85 m/s observerades sandpartiklarna ha ett gasliknande beteende över packningen, se figur 18. I samma figur går det även att observera att lite sand hade börjat lägga sig som ett tunt lager ovanpå packningen. Själva bädden var fortfarande fluidiserad.



Figur 17: Bildbeskrivning av turbulent yta.



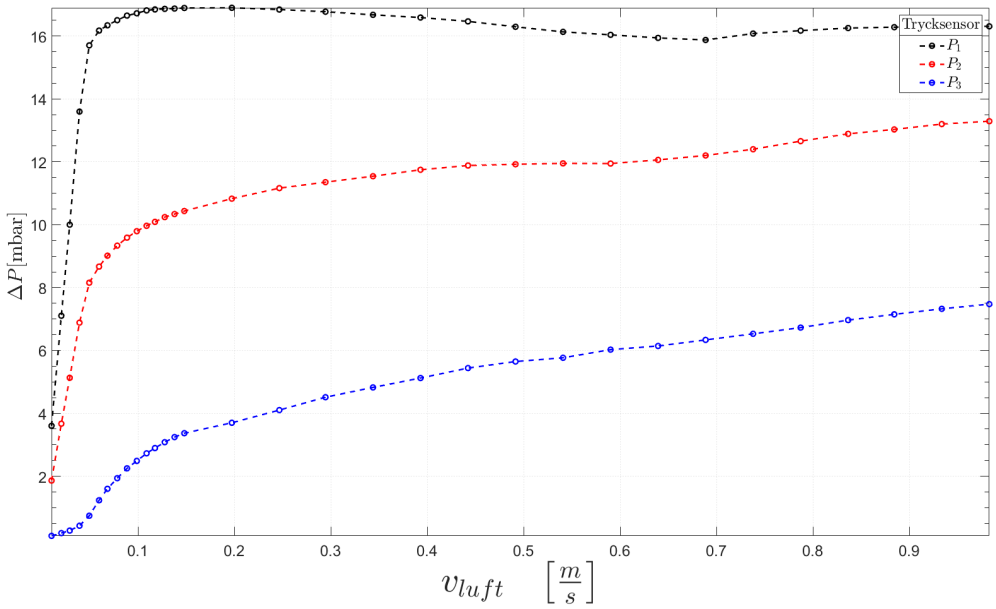
Figur 18: Gasformig sand ovanför packning för Sand I, RMSR 25-3, R = 0.25

4.2.3 Hiflow-ringar

För **R = 1** kunde en turbulent yta noteras vid 0.12 m/s. Turbulensen vid ytan observerades öka med högre luftflödeshastigheter. Vid 0.55 m/s kunde påbörjan till sluggning observeras i bädden.

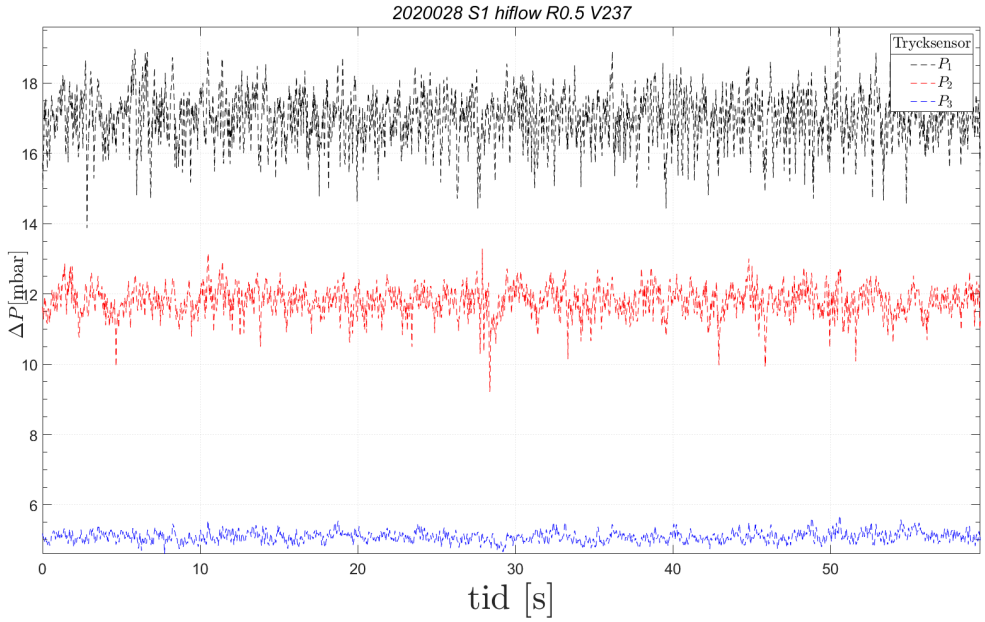
Förhållandet **R = 0.5** visade hur sanden började fastna på packningarna vid 0.09 m/s i form av att sanden skvätte vid sandytan och inte rann tillbaka. En turbulent sandyta oberverades vid 0.65 m/s. Vid 0.80 m/s hade sandbädden expanderat så att packningen täcktes helt. Hastigheten 0.95 m/s visade beteendet med gasformiga sandpartiklar, likt figur 18.

R = 0.25 gav att det vid 0.60 m/s kunde observeras hur sanden började bete sig gasformigt och sandpartiklar fastnade på packningarna högt upp. Vid 0.95 m/s hade så mycket sand fastnat på packningarna att den började rinna tillbaka till sandytan, se figur 40 i Bilaga D. Figur 19 visar hur trycket beror på ökad superficiell gashastighet för packningen Hiflow-ringar, R = 0.5.



Figur 19: Tryck vid ökad superficiell gashastighet för Hiflow-ringar, R = 0.5.

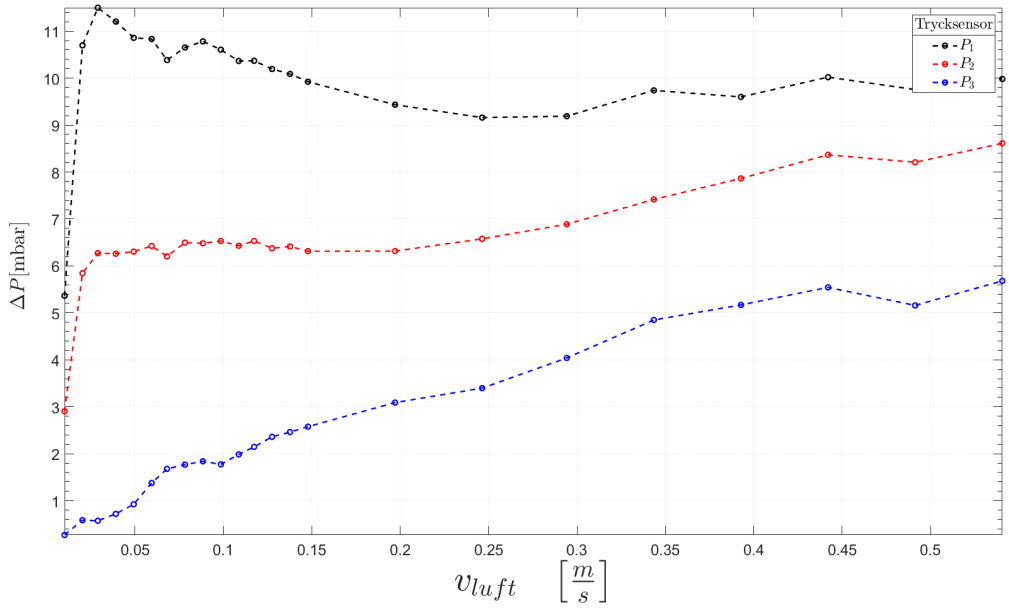
Figur 20 visar hur trycket förändras under samplingstiden för Hiflow-ringar.



Figur 20: Tryckändring under samplingstiden för Hiflow-ringar, R = 0.5, v_{luft} = 0.35 m/s.

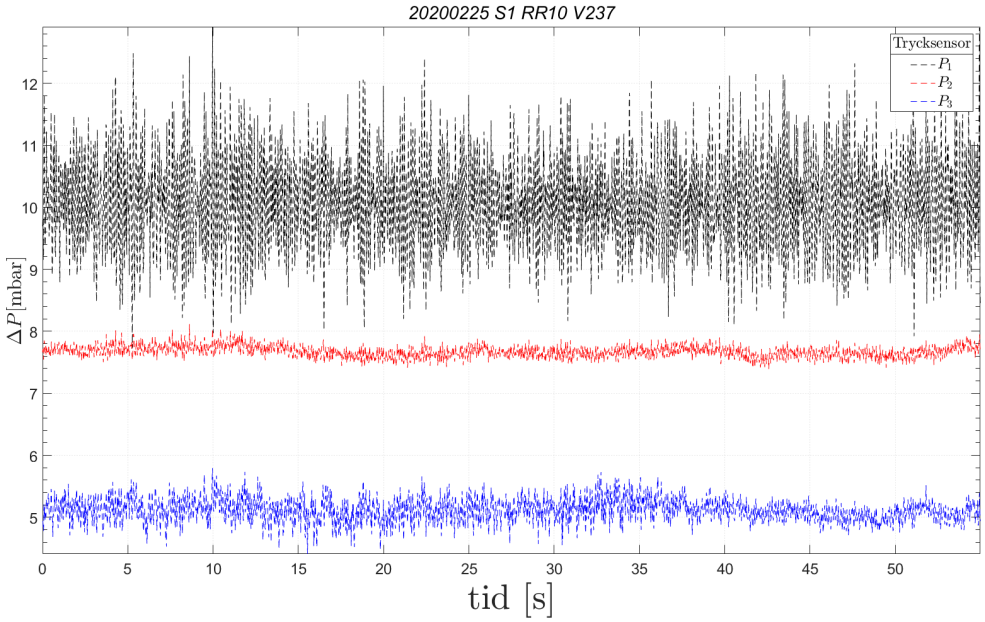
4.2.4 Raschigringar

Raschigringar 10x10mm med **R = 1** uppvisade tydlig kanalbildning vid låga hastigheter, först observerat runt 0.07 m/s. Vid högre hastigheter, med start från 0.35 m/s så observerades tydlig vertikal segregation av packningen och sanden. Vid 0.45 m/s uppkom till viss del sluggning, samt att sanden uppvisade dålig cirkulation i bädden. En stor andel av sanden låg för denna lufthastighet ovanpå packningen. Figur 21 visar hur trycket beror på ökad superficiell gashastighet för packningen Raschigringar 10x10 mm, R = 1.



Figur 21: Tryck vid ökad superficiell gashastighet för Raschigringar 10x10 mm, R = 1.

Figur 22 visar hur trycket förändras under samplingstiden för Raschigringar vid luftflödet 0.35 m/s och packningsförhållandet R = 1.

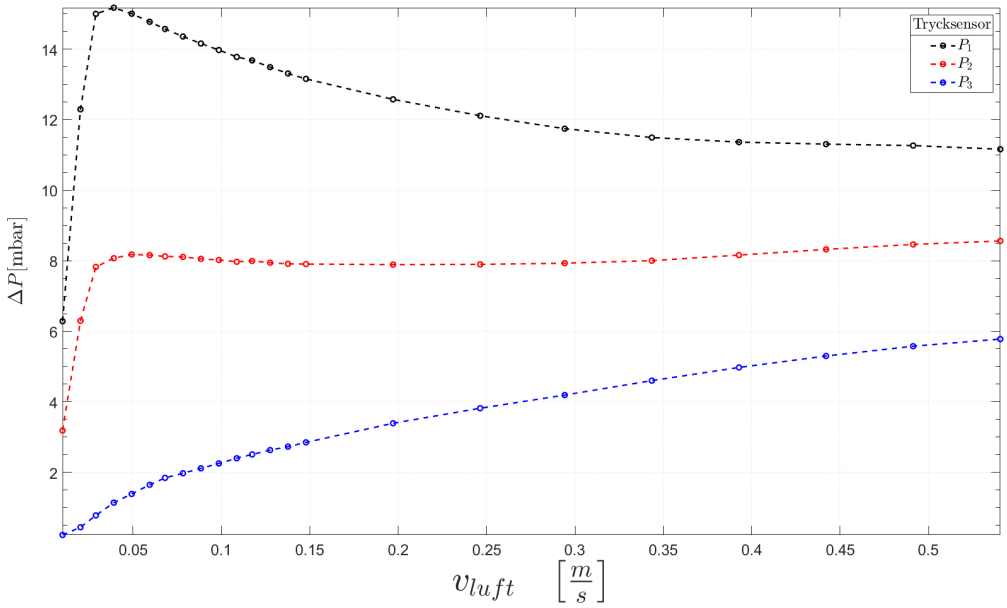


Figur 22: Tryckändring under samplingstiden för Raschigringar , R = 1, v_{luft} = 0.35 m/s.

Raschigringar med förhållandet **R = 0.5**, visade tidig kanalbildning nära väggarna på cylindern vid 0.10 m/s. Vid 0.20 m/s observerades det hur sanden blåsts upp och landat ovanpå packningen. När 0.35 m/s nåtts noterades det att det att sand blåste upp ur tvärsnittsareans mitt, i ett fontänliknande beteende, vilket tyder på kanalbildning. Vidare ökning av luftflöde gav mer deponering av sand ovanför packningen, och det kunde observeras att inga bubblor bildades på ytan vid 0.45 m/s. Härefter så noterades det att det var lite sand kvar bland packningen, då majoriteten migrerat uppåt i cylindern. Vid 0.60 m/s observerades tendenser till fluidisering för första gången i det lager av sand som skapats ovanpå packningen. Vid 0.75 m/s kunde sluggning noteras i sagda sandlager ovanpå packningen.

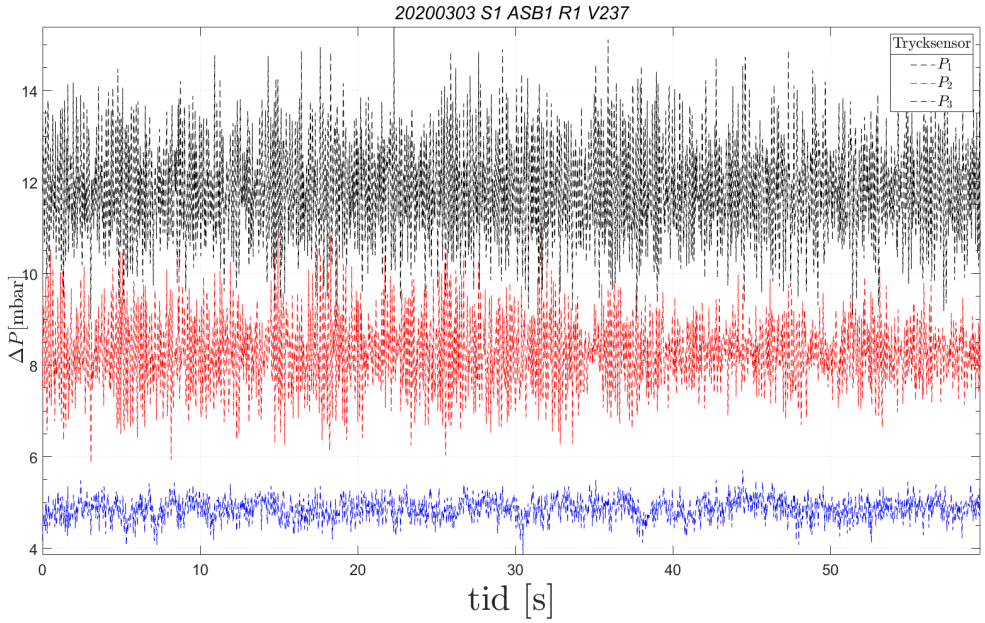
4.2.5 ASB-packning

För **ASB 1”** och **R = 1** expanderade bädden tidigt vid 0.08 m/s. Sanden fluidiserade ovanför packningen och vid 0.40 m/s kunde sluggning observeras i sanden. Sandytan var dessutom väldigt turbulent. Figur 23 visar hur trycket beror på ökad superficiell gashastighet för packningen ASB 1” vid R = 1.



Figur 23: Tryck vid ökad superficiell gashastighet för ASB 1", R = 1.

Figur 24 visar hur trycket förändras under samplingstiden för ASB 1" R = 1 vid luftflödet 0.35m/s.



Figur 24: Tryckändring under samplingstiden för ASB 1" , R = 1, v_{luft} = 0.35 m/s.

För **ASB 1"** vid **R = 0.5** började sanden stänka upp ovanpå packningen vid 0.10 m/s. Det noterades vid 0.45 m/s hur större luftfickor bildades runt packningarna. Vid 0.95 m/s noterades att packningen inte verkade ha tydlig effekt längre, då majoriteten av sanden befann sig ovanför packningen.

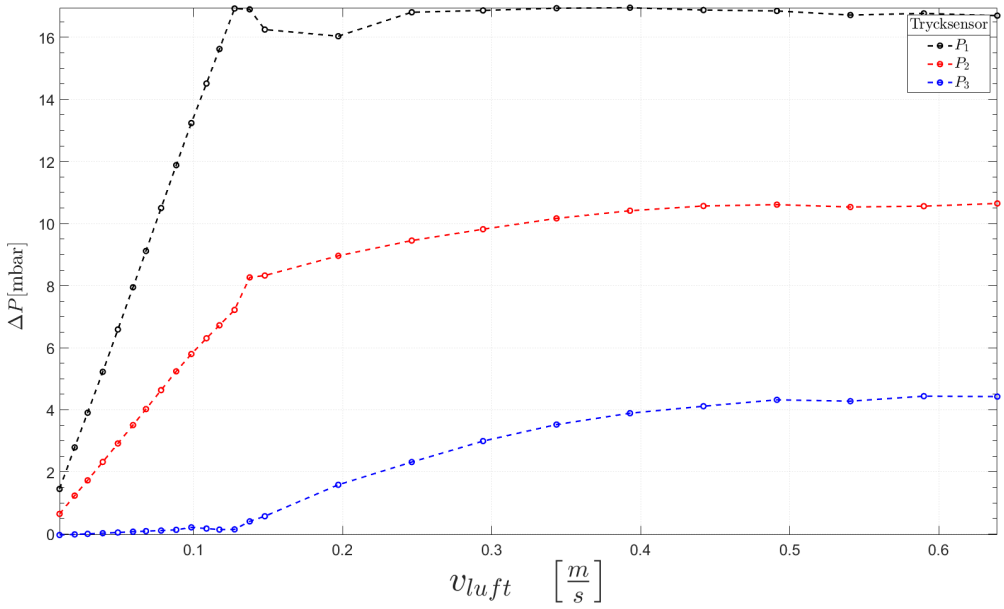
För **ASB 1/2"** med **R = 1** så noterades större luftrum och en separation av sanden från botten av cylindern vid 0.25 m/s. Vid 0.35 m/s ökade luftrummet kring packningen och sandytan betedde sig mer turbulent. Vid 0.50 m/s observerades det gasformiga sandpartiklar ovanför sandbädden. Vid 0.60 m/s kunde kanalbildning observeras kring kanterna på cylindern. Efter dessa högre hastigheter hade någon form av vertikal segregation bildats.

För **ASB 1/2”** och **R = 0.5** kunde det vid 0.30 m/s observeras hur sanden började lägga sig på toppen av bädden. Vid 0.40 m/s kunde viss kanalbildning observeras genom att ena sidan av sandbädden var mer turbulent än andra sidan. Ju mer hastigheten ökade desto mer minskade sanden i botten av cylindern och ökade ovanför packningen, det vill säga vertikal segregation. Vid 0.70 m/s började sandpartiklarna överst bete sig gasformigt och vid 0.95 m/s noterades en vertikal segregation på 5 cm från botten.

4.3 Resultat Sand II

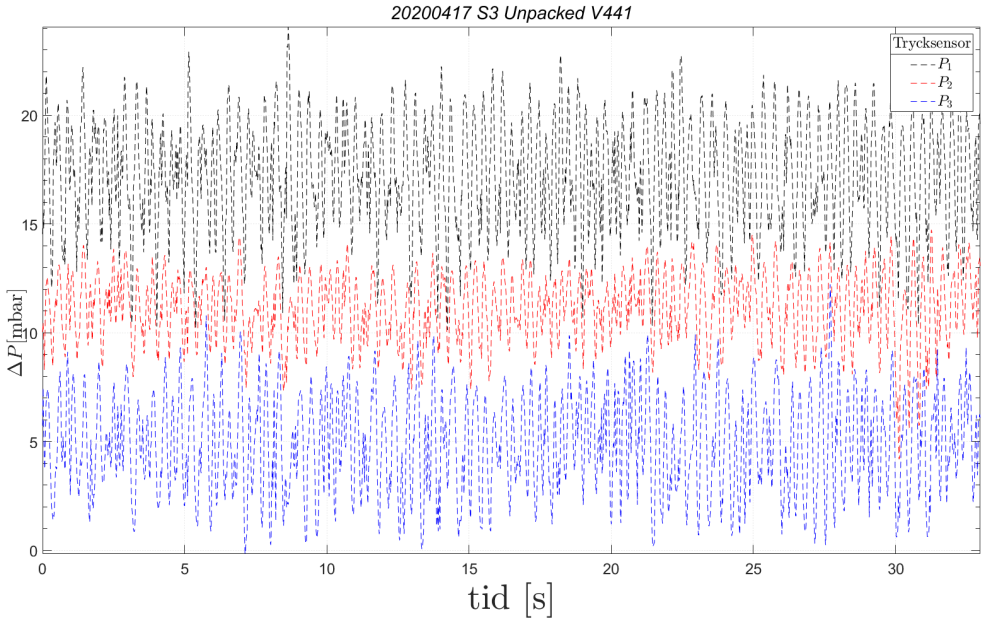
4.3.1 Icke-packad fluidiserad bädd

Figur 25 visar hur trycket beror på ökad superficiell gashastighet för den fluidiserade bädden för Sand II. Grafen visar hur trycket ökar olika mycket vid olika avstånd från inströmmningen av luft, där P_1 är närmast cylinderns botten, P_2 i mitten och P_3 överst.



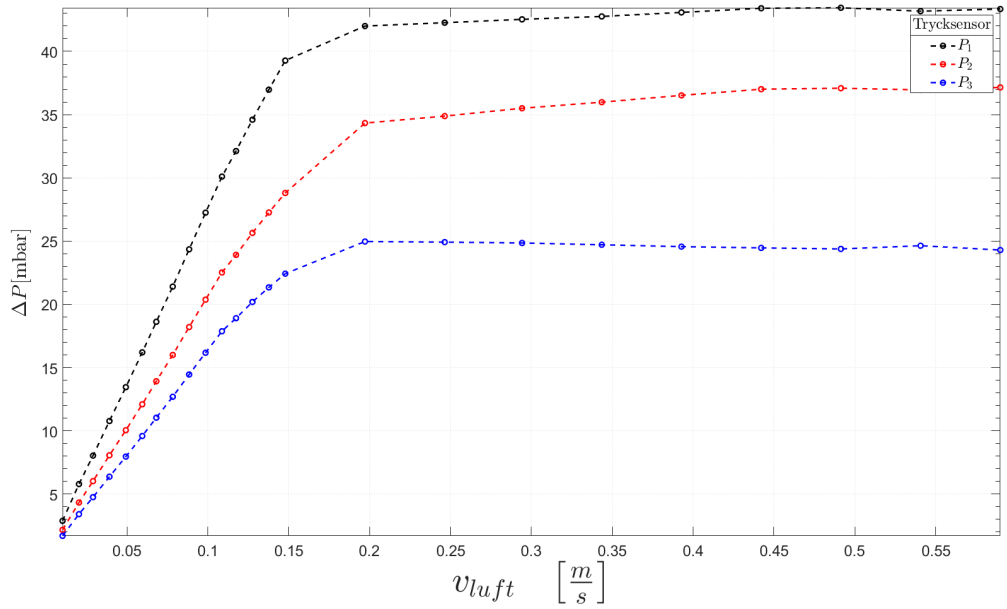
Figur 25: Tryckskillnad vid ökad superficiell gashastighet för icke-packad bädd Sand II.

Stora bubblor bildades vid 0.25 m/s och vid 0.45 m/s förekom viss sluggning. Vid 0.55 m/s vad sluggning tydligt förekommande. Figur 26 visar hur trycket förändras under samplingstiden för en icke-packad bädd.



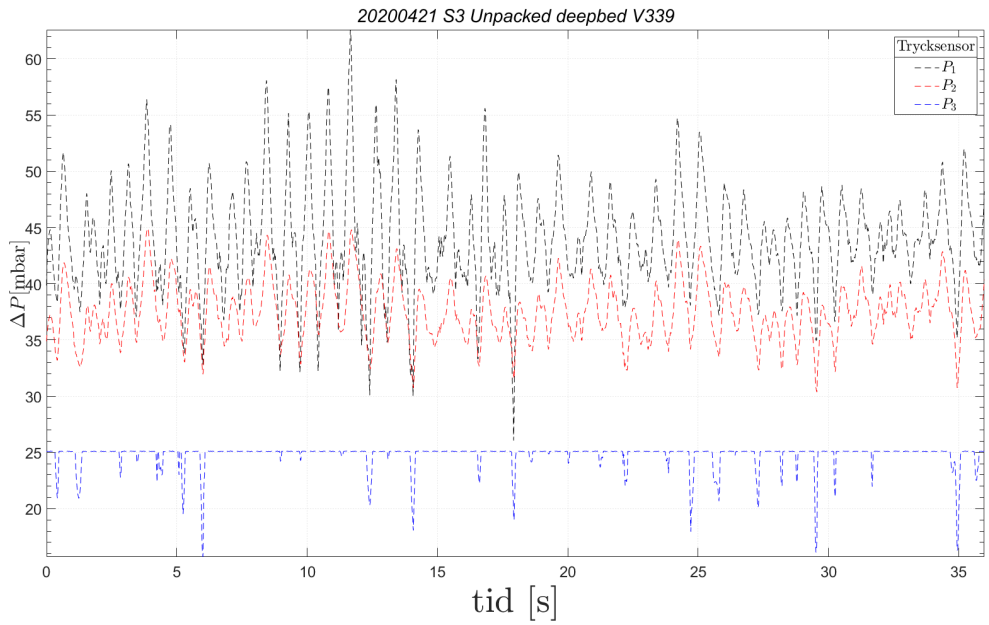
Figur 26: Tryckändring under samplingstiden för Sand II, v_{luft} = 0.65 m/s.

Figur 27 visar hur trycket ändras med ökad gasgenomströmning för Sand II, djup bädd.



Figur 27: Tryckskillnad vid ökad superficiell gashastighet för icke-packad bädd Sand II djup bädd.

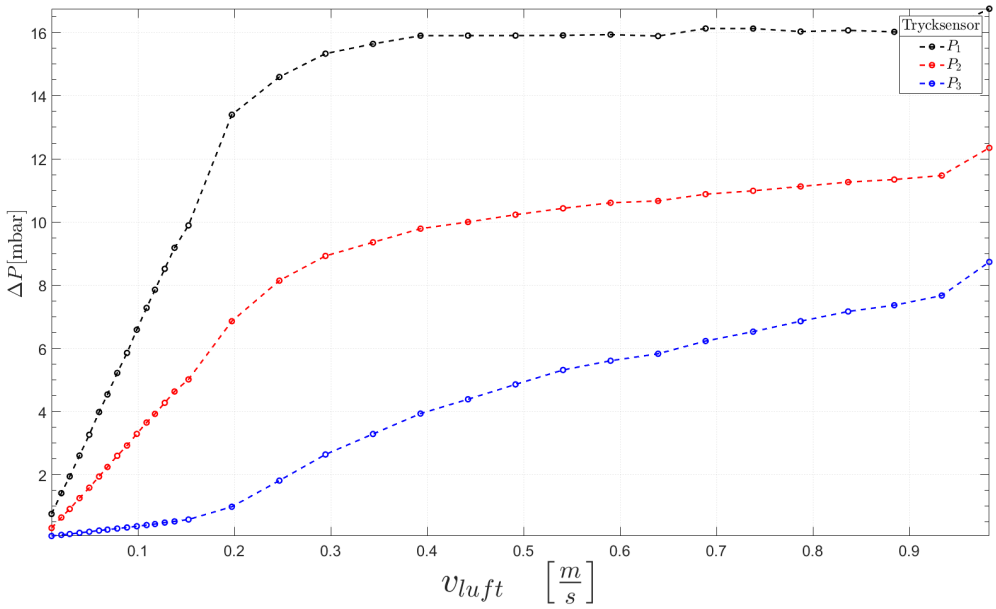
För **djup bädd** kunde sluggning observeras strax efter fluidisation vid 0.20 m/s och fortsatte slugga upp tills att experimentet avslutades vid 0.60 m/s då sanden nästan nådde toppen av cylindern. Se figur 41 i Bilaga D. Figur 28 visar hur trycket förändras under samplingstiden för djup bädd, Sand II.



Figur 28: Tryckändring under samplingstiden för Sand II djup bädd, v_{luft} = 0.50 m/s.

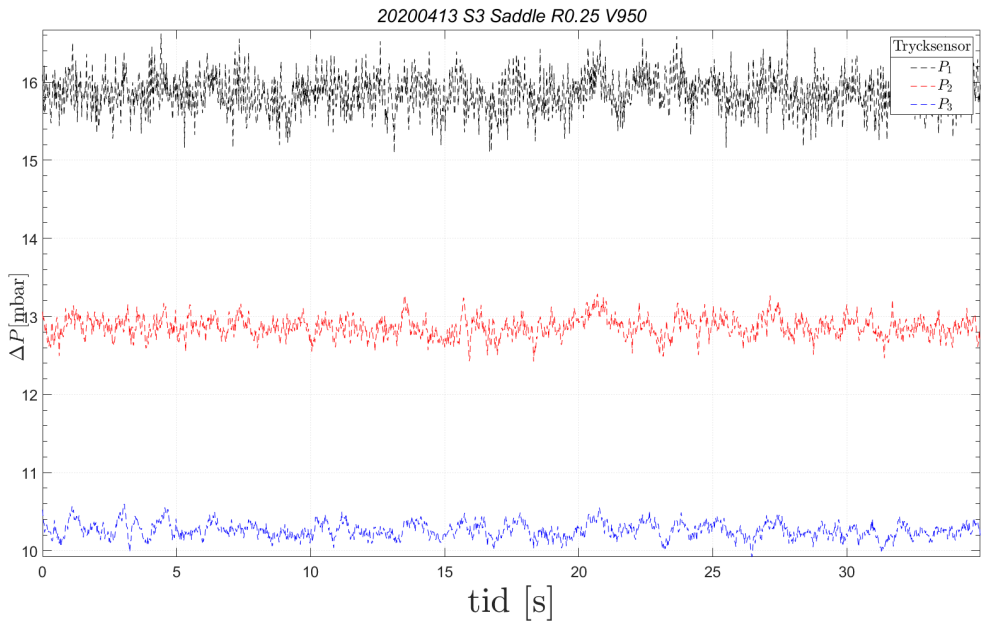
4.3.2 RMSR 25-3

För **R = 0.5** observerades det att sanden skvätte ovanför packningen först vid 0.61 m/s. Vid 0.95 m/s gjordes en notering om att bubblorna var jämnt fördelade i bädden. Figur 29 visar tryck mot luftflöde för packningen RMSR 25-3 med packningsförhållandet R = 0.25, Sand II.



Figur 29: Tryck vid ökad superficiell gashastighet för RMSR 25-3 Sand II, R=0,5.

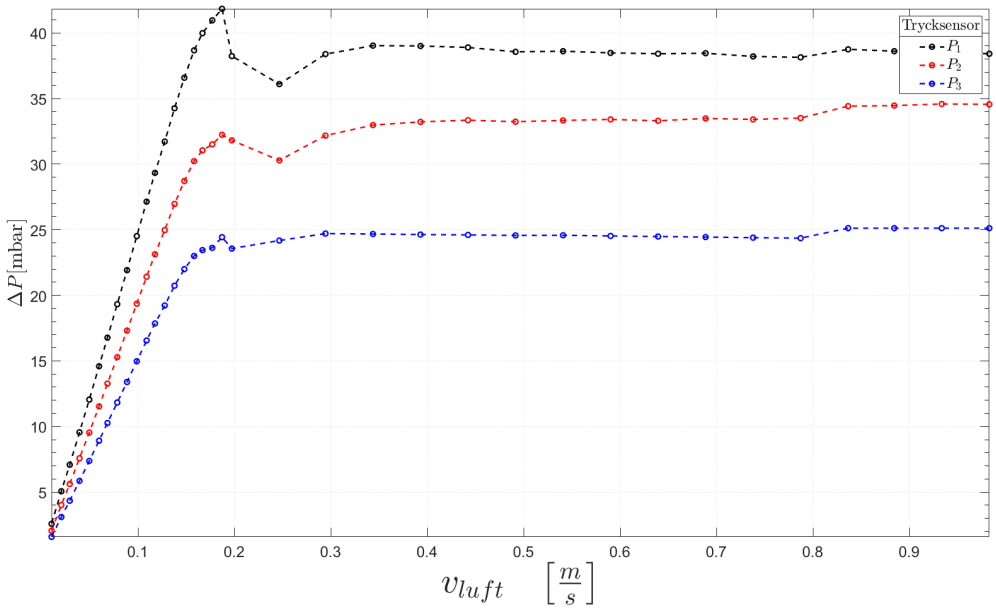
Figur 30 visar tryckförändring under samplingtiden för packningen RMSR 25-3 med packingsförhållandet $R =$ 0.25, Sand II och flödeshastigheten v_{luft} = 1.40 m/s. För **R = 0.25** observerades lite kanalbildning vid 1.25 m/s och vid 1.62 m/s så började sanden vid ytan bete sig gasformigt.



Figur 30: Tryckändring under samplingstiden för Sand II, RMSR 25-3, v_{luft} = 1.40 m/s.

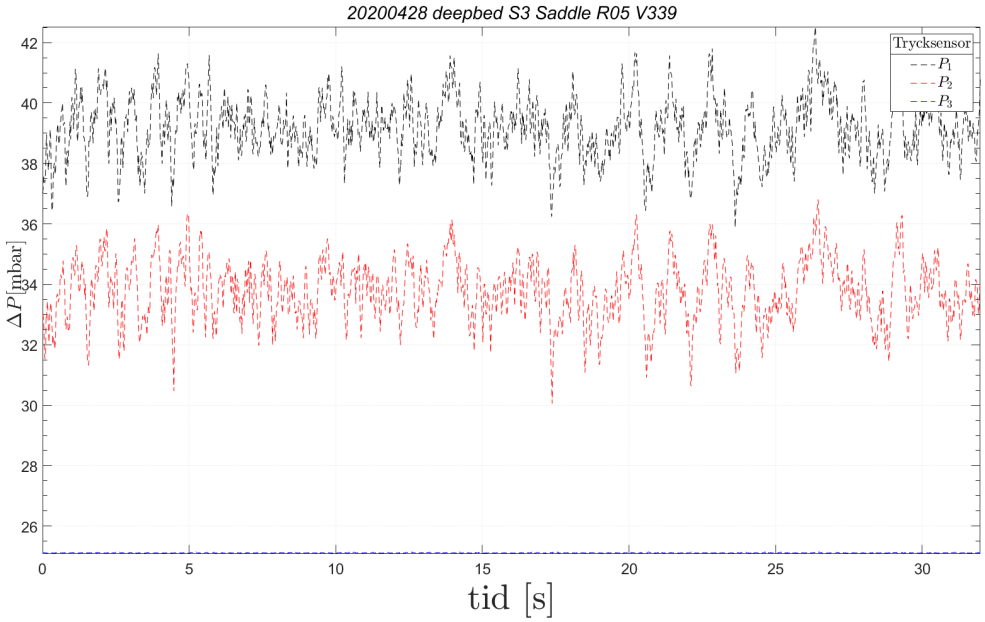
För att se hur konfigurationen presterade med mer sand kördes även experiment med ökad sandhöjd, cirka 30 cm. Med förhållandet **R = 1** vid 0.3 m/s, efter fluidisering, förekom stora bubblor i bädden och vid ytan. Vid 0.45 m/s observerad sluggning i den sanden som befann sig ovanför packningen. I den del av bädden där packningen fanns förekom en bra fördelning av bubblor.

Efter fluidisering för **R = 0.5**, vid 0.25 m/sn, noterades en jämn fördelning av bubblor i bädden. Vid 0.30 m/s noterades en lite större andel bubblor på ena sidan cylindern jämfört med den andra sidan. Vid 0.45 m/s noterades ett visst pulserande i sanden, men inte så att det kunde kallas för sluggning. Figur 31 visar hur trycket ändras med ökad gasgenomströmning för RMSR 25-3 i fallet Sand II, för djup bädd.



Figur 31: Tryckskillnad vid ökad superficiell gashastighet för RMSR 25-3, Sand II djup bädd. Grafen visar hur trycket ökar olika mycket vid olika avstånd från inströmmningen av luft.

Figur 32 visar hur trycket förändras under samplingstiden för djup bädd, Sand II.

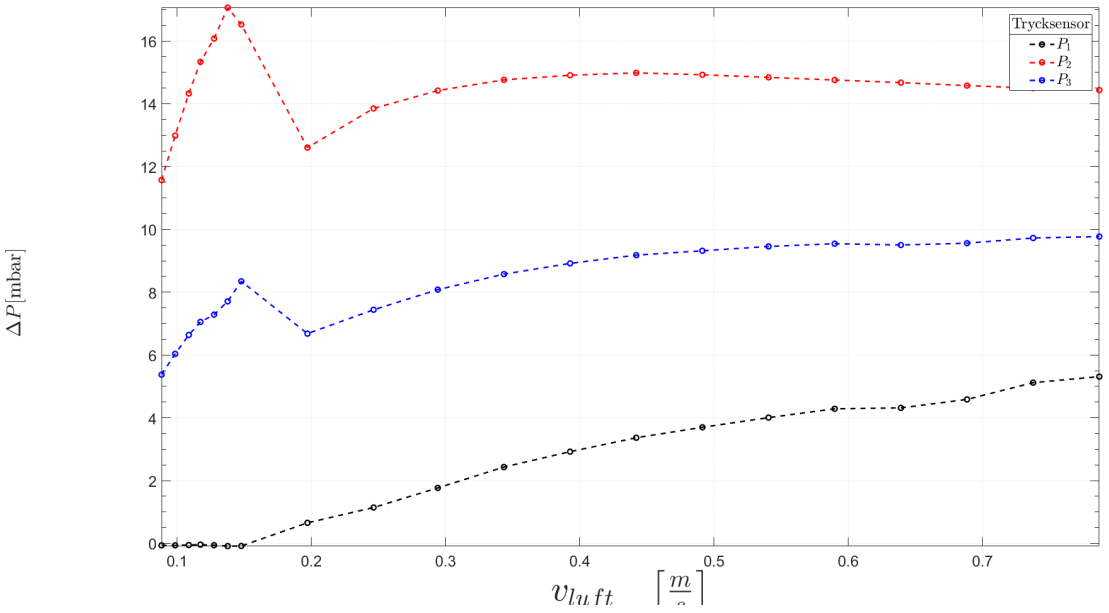


Figur 32: Tryckändring under samplingstiden för RMSR 25-3, Sand II djup bädd, v_{luft} = 0.50 m/s.

4.3.3 Hiflow-ringar

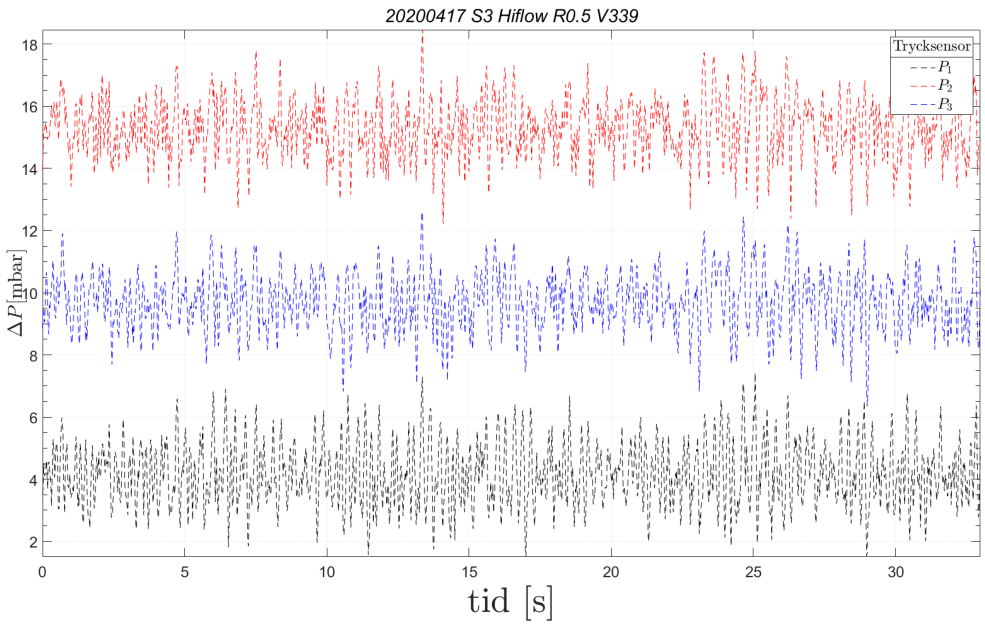
För **R = 0.5** vid 0.45 m/s noterades stora bubblor i bädden och vid 0.75 m/s stänktes sand ovanför packningen. Vid 0.90 m/s var bädden turbulent och vid 1.2 m/s noterades viss kanalbildning vid kanterna på cylinderna.

Figur 33 visar tryck mot luftflöde för packningen RMSR 25-3 med packingsförhållandet $R = 0.5$, Sand II.



Figur 33: Tryck vid ökad superficiell gashastighet för Hiflow-ringar Sand II, R = 0.5.

För **R = 0.25** förekom kanalbildning först vid 1.40 m/s. Figur 34 visar tryckförändring under samplingtiden för packningen Hiflow-ringar med packingsförhållandet $R = 0.25$, Sand II och flödet v_{luft} = 1.40 m/s.



Figur 34: Tryckändring under samplingstiden för Sand II, Hiflow-ringar, $v_{luft} = 0.50\ m/s$.

5 Diskussion

Följande del av rapporten avhandlar de tankar och resonemang kring arbetets resultat som ligger till grund för de slutsatser som dragits.

5.1 Packningsmaterialens prestation

De packningsmaterial som testades skiljde sig mycket i beteenden. Hur de presterade i relation till varandra beskrivs i följande stycken.

5.1.1 RMSR 25-3

Bland alla använda packningsmaterial hade RMSR 25-3 högst tomrumsfraktion med 96%. Den höga tomrumsfraktionen ledde till att vikten sand som användes var ungefär densamma som krävdes till en icke-packad bädd, vilket kan ses i tabell 1. Något som noterades är att den observerade minsta fluidiseringshastigheten, 0.05 m/s, för Sand I från tabell 2, är högre än den för icke-packad bädd, vilket kan vara en indikation på att packningen gör det något svårare för sanden att fluidisera.

Det kan ses från tabell 3 att packningsmaterialet RMSR 25-3 tillåter fluidisering utan förekomsten av ogynsamma fluidiseringsfenomen, som kanalbildning och sluggning, åtminstone upp till de hastigheter som var testade. Vid jämförelse mellan figur 14 och 16 för **Sand I** så blir det tydligt att packningen har haft inverkan på trycksvängningarna. Från figur 16 ses det att amplituden av svängningarna är mindre och hålls mer konsekventa då packning har introducerats till bädden. Detta stämmer med observationerna gjorda vid de olika körningarna som visade på jämnare och mindre bubbelstorlek. Vid en superficiell gashastighet av 0.35 m/s förekom sluggning i den icke-packade bädden medan den packade bädden hade expanderat något men bibehållit jämn bubbelformation i R-förhållanden.

Vid förhållandena R = 0.5 och R = 0.25 vid ökande superficiell gashastighet tycktes bädden expandera jämnt över hela cylindern utan förekomst av någon tydlig vertikal segregering, förrän vid väldigt hög gashastighet.

Icke-packad bädd med **Sand II** fluidiserades vid ett högre gasgenomflöde än Sand I och de bildade bubblorna var

av betydligt större storlek. Vid ökat gasgenomflöde svängde trycket väldigt mycket och med jämn period vilket kan ses i figur 26. Detta tolkades som förekomsten av sluggning, med stöd av observationer från experimenttillfället. Packningsmaterialet RMSR 25-3 testades vid förhållandena R = 0.5 och R = 0.25 med Sand II då det var dessa förhållanden som hade uppvisat störst skillnad gentemot icke-packad bädd för Sand I. Vid tester med tillsatt packningsmaterial kunde höga hastigheter, över 1.33 m/s, uppnås utan förekomsten av de icke gynnsamma fluidiseringsfenomen som förekom för Sand II i en icke-packad bädd vid samma hastighet. Från figur 30 ses hur amplituden av tryckförändringarna minskat avsevärt även vid mer än dubbla gasgenomflödet än för icke-packad bädd i figur 26 vid förhållandet R = 0.25. Bubbelstorleken observerades även att vara liten under försöken med tillsatt packningsmaterial, vilket är gynnsamt för hög massöverföring i bubbelregimen.

RMSR 25-3 testades även vid förhållandet R = 0.5 för djup bädd. Detta visade sig bidra till att starkt begränsa uppkomsten av sluggningsfenomen, som hade varit tydliga och uppkommit tidigt vid försök med djup bädd utan packning.

RMSR 25-3 klarade av höga flöden och varierande partikeldiameter på sanden. Dess diskontinuerliga form tillsammans med höga tomrumsfaktor antas vara en anledning till att den presterar bra, då detta leder till mer turbulens, vilket leder till ökad massöverföring i en reaktor.

5.1.2 Hiflow-ringar

Hiflow-ringarna hade en tomrumsfraktion väldigt lik den av RMSR 25-3 men hade en mer cylindrisk form. Den minsta observerade fluidiseringshastigheten var också högre än den för icke-packad bädd. Packningsmaterialets prestation kan liknas till den av RMSR 25-3 då de båda lyckades bra med hålla oönskade fluidiseringfenomen till ett minimum vid höga gashastigheter, se tabell 3. Hiflow-ringar tillät en jämn expansion av bädden med bibehållen fluidisering. Jämför man figur 16 med figur 20 så ses Hiflow-ringar uppvisa större trycksvängingar vilket tyder på att RMSR 25-3 uppvisade en stabilare fluidisering. Då packningsmaterialen är lika i både tomrumsfraktion och densitet antogs detta bero på formen hos Hiflow-ringar. Formen hos Hiflow-ringarna är mer ordnad och tillät den att lägga sig mer strukterat än RMSR 25-3. Den ordnade strukturen kan leda till att genomflödande gas inte tillåts många flödesvägar, som i sin tur skulle kunna bidra till ett ökat tryckfall. Bädden observerades även ha en starkare tendens att segregera uppåt jämfört med RMSR 25-3.

För Sand II klarade Hiflow-ringar (R = 0.25) ett flöde på strax över 1.77 m/s utan att oönskade fluidiseringfenomen uppstod, men med fortsatt god blandning. Hiflow-ringar visade sig prestera bättre än RMSR 25-3 vid extrema gasgenomströmningar för Sand II. Detta kan indikera på att Hiflow-ringar är bättre när höga flöden av gas är eftersträvat, om man utgår från data för Sand II.

5.1.3 Raschigringar

Raschigringarnas tomrumsfraktion finns angiven i tabell 1 och 59% för 10x10 mm. Tomrumsfraktionen ligger grund till att mängden sand som använts var mindre än den använd för de övriga packningsmaterialen som ses i tabell 2, med undantag för ASB som hade lägre tomrumsfraktion.

Vid förhållandet R = 1 för Raschigringar 10x10 kunde bädden observeras att fluidisera men gav upphov till kanalbildning redan vid låga hastigheter. Det förekom även sluggning och vertikal segregation av bädden vid högre hastigheter vilket ses i tabell 3. Raschigringar 10x10 med förhållandet R = 0.5 hade tendenser till fluidisering men det förekom aldrig jämnt över hela bädden. Vid högre hastigheter hade den största delen av sanden segregerat upp ovanför packningsmaterialet. Även då förekom icke gynnsamma fluidiseringsfenomen i form av sluggning i det bildade sandlagret. Packningsmaterialet hade överlag problem att klara av jämn fluidisering och höga flöden då kanalbildning och vertikal segregering förekom tidigt, för båda testade R-förhållanden. Från figur 22 ses Raschigringar med förhållandet R = 1 visa stora trycksvängningar vid hastigheten 0.35 m/s. Framförallt förekommer svängningarna från sensorn P_1 närmast botten av bädden. Dessa svängningar antas kunna relateras till de ogynnsamma fluidiseringsfenomen som Raschigringarna gav upphov till vid den hastigheten.

Då testen med Raschigringar 10x10 inte gav fördelaktiga resultat med avseende på fluidiseringsegenskaper i jämförelse med en icke-packad bädd så testades inte Raschigringar 6x6. Dessa antogs ha relativt liknande egenskaper som den större varianten av packningsmaterialet och därmed leda till liknande resultat.

5.1.4 ASB

De använda packningarna av ASB beräknades ha liknande tomrumsfraktion, oavsett deras skillnad i storlek. Detta borde leda till att massan sand som används för att nå till 12 *cm* är densamma för ASB 1" som för ASB 1/2". Tabell 2 visar dock att så inte var fallet och mängden sand skiljde sig med ca 300 gram för de två olika packningarna. Anledningen till detta kan antas komma från att ASB 1/2" på grund av dess mindre storlek kunde placeras mer ordnat i cylindern och därmed ge mindre plats för sanden. Då den använda cylindern till kallflödesexperimenten endast var 12 *cm* i diameter kan packningsformationen av ASB 1" skilja sig mellan olika körningar och leda till att den givna tomrumsfaktorn inte längre stämmer för tillämpningen. De båda packningsmaterialen avvek från den angivna tomrumsfraktionen i tabell 2 med runt 5%.

Under testen så fluidiserade bäddmaterialet vid relativt låga hastigheter och bubblorna höll sig jämnt fördelade över ytan. Vid jämförande av minsta fluidiseringshastigheterna i tabell 2 fluidiserade både ASB 1" och 1/2" vid liknande hastighet av en icke-packad bädd. Med stegande ökning av hastigheten expanderade bädden konsekvent tills sanden nått toppen av packningen vid vilken punkt det började förekomma vertikal segregation från cylinderns botten för båda packningsstorlekar, se figur 3. Sanden fortsatte att ackumuleras tills all sand nått över packningen och fluidiserades ovanför packningsmaterialet. Den låga tomrumsfraktionen av packningarna ledde till att mindre massa av sand användes i bädden för att uppnå en höjd av 12 *cm*. Då mängden sand skiljer sig såpass mycket från den icke-packade bädden antas det även ha legat grund till packningens egenskaper i bädden.

Resultaten för ASB 1" och ASB 1/2" uppvisade lika expansionsbeteende av bädden vid R = 1 och R = 0.5 med avseende på vertikal segregation. Bildandet av en bädd ovanför packningen ledde till slut att packningsmaterialet inte hade någon vidare betydelse för fluidiseringen och bubbelbildningen. De stora variationer i tryck som kan ses i figur 24 påvisar i sig ett sluggningsbeteende för ASB 1". Vid denna superficiella gashastighet, 0.35 m/s, hade bädden expanderat till den punkt att en stor del av sanden befann sig ovanför packningen. Enligt tabell 3 påvisade inte ASB 1/2" några sluggningsfenomen. Anledningen till detta tros vara att mängden sand i detta försök var för liten för att kunna ge upphov till bubblor tillräckligt stora för att bilda sluggning. Bädden bildade fortfarande stora bubblor då den expanderat ovanför packningen. Då de två testade storlekarna av ASB gav liknande resultat och hade runt samma tomrumsfraktion som ASB 1/4" så antogs denna storlek på packningsmaterialet ha liknande egenskaper. Då testen inte gav lika positiva resultat som för andra testade packningsmaterial så gjordes inga vidare experiment för att testa ASB 1/4".

5.2 Sandskillnader

För majoriteten av experimenten användes Sand I, som hade en lägre medeldiameter på sandkornen gentemot Sand II. Direkta skillnader i prestation var hur en lägre medeldiameter resulterade i en betydligt lägre minsta fludisering-shastighet i samtliga av försöken som genomfördes. Det är generellt enklare för den strömmande luften att passera igenom sand med mindre diameter på kornen. Att Sand II kräver högre flöden av luft för att fluidisera medför att sanden också klarar av högre luftflöden innan det uppkommer störningar i sandens prestation. Det går allstå att argumentera för att sand med lägre medeldiameter på kornen är mer gynnsamt för lägre flöden av luft eftersom man erhåller fluidisering med bra omblandning tidigt, i jämförelse med sand som har högre medeldiameter. För processer som vill hålla ett högt luftflöde, vilket är ett vanligt krav, pekar resultaten från experimenten på att det är gynnsamt att använda sand med högre medeldiameter.

5.3 Potentiella packningsproblem

För de packningar som var mest lovande, RMSR 25-3 och Hiflow-ringar, har det inte under testerna funnits underlag för att det skulle ge några problem att använda dem i en fluidiserad bädd. Det verkar som att Hiflow-ringar är något bättre presterande för Sand II än RMSR 25-3 medan verkar RMSR 25-3 bättre för Sand I. För de mindre lovande packningarna, ASB och Raschigringar, så fanns det problem med att sanden migrerade uppåt vid höga gashastigheter och placerade sig ovanpå packningen. Det är ofördelaktigt då det försämrar packningens inverkan på bubbelstorleken samt att sanden istället fluidiserar ovanför packningen.

Huruvida packningsmaterialet skulle vara ett problem i en reaktor är svårt att formulera en argumentation för, men det framstår inte som omöjligt att valet av packning skulle kunna påverka en potentiell reaktion i en reaktor av

fluidiserad bäddtyp. En metall skulle kunna reagera med syret i reaktionen, något som sker även om packningen består av rostfritt stål. Packningen bör även klara av de höga temperaturer som skulle kunna uppstå i en reaktion.

5.4 Förhållandet mellan mängden packning och sand

Det visade sig tidigt vara fördelaktigt att ha mer packning än sand i cylindern, då detta verkade leda till ökad räckvidd för sanden att befinna sig i samma volym som packningen. Ökat gasflöde gör att bädden expanderar, mer packning leder till att kontaktytan mellan packning och sand bibehålls. Raschigringar och ASB 1" hade inte samma fördel av att ha mer packning än sand då de uppvisade fler oönskade fluidisieringsfenomen. ASB 1/2" presterade aningen bättre. Ett stort problem med Raschigringar och ASB var att den fluidiserade bädden hade en stor tendens att vandra uppåt i cylindern vid högre gasflöden. Detta ledde till att mer packning i cylindern bara bidrog till en försenad segregering för dessa typer av packningar. För RMSR 25-3 och Hiflow-ringar visade det sig vara gynnsamt att ha mer packning än sand i cylindern.

5.5 Potentiella problem med metoden

Den laborationsmetod som har efterföljts, och som finns beskriven under metod, innehåller moment som gör det svårt att få kvantitativa resultat. En stor del av den slutsats som formulerats utifrån resultaten bygger på observationer där en laborant med ord beskrivit vilka fenomen och beteenden som personen i fråga anser uppkommit i cylindern. Metoden ger inte resultat som nödvändigtvis ger en identisk upprepning om man skulle genomföra samma experiment igen. I tidiga experiment stod det inte helt klart hur alla olika fenomen som uppstod såg ut vilket gjorde bedömningen av det visuella svårare för laboranten.

Utöver detta kunde en noggrannare studie av bäddhöjd ha genomförts, för att se ifall detta hade någon inverkan. Det borde också ha diskuterats hurvida massa borde använts istället för höjd för att framställa noggrannare experiment.

6 Slutsats

Projektet visar hur olika packningsmaterial har olika förmåga att motverka icke-gynnsamma fluidiseringsfenomen. Två av de testade packningsmaterialen, RMSR 25-3 och Hiflow-ringar, uppvisade positiva effekter och lyckades båda att reducera oönskade fluidiseringsfenomen för alla packningsförhållanden och för båda sandtyperna. De två packningsmaterialen presterade bra även i en djup bädd, och lyckades eliminera flera av de oönskade fluidiseringsfenomen . Raschigringar presterade inte på en acceptabel nivå, då det förekom problem med att få bädden att fluidisera redan vid låga luftflöden, där icke-gynnsamma fluidiseringfenomen var närvarande. ASB-packning uppvisade en svag förmåga att hämma icke-gynnsamma fluidiseringsfenomen, då packningens lägre tomrumsfaktor i kombination med formen på packningen, ledde till vertikal segregation.

De egenskaper som var bärande för god prestation hos de packningsmaterial som bedömdes mest lämpliga, Hiflow-ringar och RMSR 25-3, verkar vara deras höga tomrumsfraktioner samt förmågan att placera sig i ostrukturerade mönster i sanden. Den höga tomrumsfraktionen har effekten att mängden sand i bädden inte påverkas speciellt mycket av packningens närvaro. Ostrukturerat packningsmönster motverkar strömningsluftens möjligheter att hitta enklare vägar genom bädden, och motverkar på så sätt kanalbildning. Hiflow presterade bäst för Sand II och RMSR 25-3 bäst för Sand I utav alla packningar, detta tillsammans med packningsförhållanderna R = 0.5 och R = 0.25 gav mest positiv inverkan på den fluidiserade bädden och ökade troligtvis masstransporten betydande mellan gas och sanden .

Ett förslag för vidare forskning hade varit att testa de packningarna som gav gynnsamma förhållande i en reaktion för att utvärdera deras kapacitet i att klara höga temperaturer. Det hade också varit intressant att skapa packningar av andra material än metall med en högre tomrumsfraktion för att säkerställa att detta är den faktor som har mest inverkan.

Källförteckning

[1] bp p.l.c. BP Statistical Review of World Energy, 2019. URL https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/bp-stats-review-2019-full-report.pdf.

[2] Brian J McPherson. Carbon Capture and Storage. *Access Science*, 2010. doi: https://doi.org/10.1036/1097-8542.YB100132. URL https://www.accessscience.com/content/carbon-capture-and-storage/YB100132#YB100132s009.

[3] Magnus Rydén, Anders Lyngfelt, Øyvind Langørgen, Yngve Larring, Anders Brink, Sebastian Teire, Hallstein Havåg, and Per Karmhagen. Negative CO_2 Emissions with Chemical-Looping Combustion of Biomass – A Nordic Energy Research Flagship Project. *Energy Procedia*, 114:6074–6082, 2017. doi: https://doi.org/10.1016/j.egypro.2017.03.1744. URL https://www.sciencedirect.com/science/article/pii/S187661021731946X?via%3Dihub.

[4] Tianxu Shena, Shen Wanga, Jingchun Yan, Laihong Shen, and Hanjing Tian. Performance improvement of chemical looping combustion with coal by optimizing operational strategies in a 3 kwth interconnected fluidized bed. *International Journal of Greenhouse Gas Control*, 84(103060), 7 2020.

[5] Malin Hanning, Magnus Rydén, Christina Dueso, Tobias Mattisson, and Anders Lyngfelt. Camn0.9mg0.1o3-δ as oxygen carrier in a gas-fired 10 kwth chemical-looping combustion unit. *Industrial & Engineering Chemistry Research*, 52(21):6923–6932, 10 2017.

[6] Grace J.R. and Clift R. On the two-phase theory of fluidization. *Chemical Engineering Science*, 29(2): 327–334, 2 1974.

[7] Donald E. Beasley. Fluidized bed. *Access Science*, 1999. doi: https://doi.org/10.1036/1097-8542.YB990415. URL https://www.accessscience.com/content/fluidized-bed/YB990415.

[8] Magnus Rydén. The application of confined fluidization in energy conversion. Technical report, Chalmers, 2018.

[9] Tobias Mattisson, Erik Jerndal, Carl Linderholm, and Anders Lyngfelt. Reactivity of a spray-dried NiO/NiAl_2O_4 oxygen carrier for chemical-looping combustion. *Chemical Engineering Science*, 66(20): 4636–4644, 2011. doi: https://doi.org/10.1016/j.ces.2011.06.025.

[10] Hans H. Bruun, Falk Hamad, and Barbara K. Pierscionek. Two-Phase Flow. *AccessScience*, 2014. doi: https://doi.org/10.1036/1097-8542.757752. URL https://www.accessscience.com/content/two-phase-flow/757752.

[11] Cornelius Emeka Agu, Lars-André Tokheim, Marianne Eikeland, and Britt M.E. Moldestad. Determination of onset of bubbling and slugging in a fluidized bed using a dual-plane electrical capacitance tomography system. *Chemical Engineering Journal*, 328:997–1008, 2017. doi: https://doi.org/10.1016/j.cej.2017.07.098. URL https://www.sciencedirect.com/science/article/pii/S1385894717312445.

[12] Hassan Abba Khawaja and Mojtaba Moatamedi. *Multiphysics Modelling of Fluid-Particulate Systems*. Academic Press, 2020. ISBN 978-0-12-818345-8. doi: https://doi.org/10.1016/B978-0-12-818345-8.00001-9. URL https://www.sciencedirect.com/science/article/pii/B9780128183458000019.

[13] D. Geldart. Types of Gas Fluidization. *Powder Technology*, 7(5):285–292, 1973. doi: https://doi.org/10.1016/0032-5910(73)80037-3.

[14] Ronald W. Breault. *Handbook of Chemical Looping Technology*. Wiley-VCH Verlag GmbH & Co. KGaA, 2018. ISBN 978-3-52-780933-2. doi: https://doi.org/10.1002/9783527809332.

[15] Daizo Kunii and Octave Levenspiel. *Fluidization Engineering*. Butterworth-Heinemann, 2 edition, 1991. ISBN 978-0-08-050664-7. doi: https://doi.org/10.1016/C2009-0-24190-0.

[16] C. Alberto, S. Felipe, and S. C. S. Rocha. Time series analysis of pressure fluctuation in gas-solid fluidized beds. *Brazilian Journal of Chemical Engineering*, 21, 2004. doi: https://doi.org/10.1590/S0104-66322004000300014. URL https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-66322004000300014&lng=en&tlng=en.

[17] Fyllkropp, 2020. URL https://www.ne.se/uppslagsverk/encyklopedi/ltext=%C3%A5ng/fyllkropp. [May 10, 2020].

[18] Jesper Aronsson, David Pallarès, Magnus Rydén, and Anders Lyngfelt. Increasing Gas-Solids Mass Transfer in Fluidized Beds by Application of Confined Fluidization — A Feasibility Study. *Applied Sciences*, 9(4):634, 2019. doi: 10.3390.

[19] J. P. Sutherland, George Vassilatos, Hiroshi Kubota, and G. L. Osberg. The Effect of Packing on a Fluidized Bed. *A.I.Ch.E. Journal*, 9(4):437–441, 1963.

[20] Hansa Engineering AB. Hiflow, 2020. URL http://hansa-engineering.se/miljoteknik-processteknik/fyllkroppar/hiflow/.

[21] Hansa Engineering AB. RMSR, 2020.

[22] Hembryggning Gert Strand AB. Rashigringar & Fyllkroppar, 2020. URL https://hembryggning.se/utrustning-for-jasning/tillbehor/rashigringar-fyllkroppar.

[23] Vereinigte Füllkörper-Farbriken GmbH & Co. KG. DURANIT, 2020. URL http://www.vff.com/phocadownloadpap/120521{_}duranit{_}e{_}download.pdf.

[24] Stephen Cobeldic. natsortbib.m, 2019. URL https://se.mathworks.com/matlabcentral/fileexchange/47434-natural-order-filename-sort.

[25] Yair Altman. screencapture.m, 2016. URL https://www.mathworks.com/matlabcentral/fileexchange/24323-screencapture-get-a-screen-capture-of-a-figure-frame-or-component.

Bilagor

A Standardiserad laborationsmetod

Laborationsmetod

Datorprogram

1. Sätt på och logga in på datorn i experimenthallen
2. Starta LabVIEW
3. Starta rätt program i LabVIEW, i vårt fall Jespers “Cold Flow-model”

Sanden

4. Sila sanden (vid behov). Gå annars till steg 6
5. Beräkna medeldiameter för sanden och märk sandens behållare med namn och diameter

Unpacked sand

6. Beräkna medeldensitet på sanden som används
7. Beräkna hur mycket sand som krävs för att få en bäddhöjd på 12 centimeter
    a. Notera vikten på sanden
8. Häll i uppmätt sand i tuben
    a. Notera höjden
9. Notera att MFC FR är öppen
10. Kalibrera LabVIEW med “calibrate”-verktyget.
11. Ange volymflöde för MFC
    a. se excel för vilket volymflöde
    b. kontrollera att volymflödet inte överstiger 400l/min för MFC FR. Om det gör det, genomför steg 13.
    c. Spara med rätt filnamn (DATUM_SAND_PACKNING_VF_R) och i rätt folder
    d. spara tryckdata i 1 minut med “Save”-verktyget
    e. anteckna observationer i lab-boken
    f. bedöm om det är värt att fortsätta experimentet (se steg 14).
12. Upprepa steg 11 med nytt volymflöde.
13. Vid ~400 L/min ändra MFC från FR till AR
    a. Sänk först FR-flödet successivt till noll
    b. Öka AR-flödet successivt till önskat volymflöde
14. Avbryt då slugging/splashing är för överdriven. Se avsnitt “Avsluta Försök”.

Packed sand

15. Notera vilken ratio R försöket ska hålla
16. Fyll med packning till ca ~12 cm för R=1
    a. notera massan av packningen som används
    b. använd packningsmassan för R=1 för att beräkna massan packning som ska användas vid R=0,5 och R=0,25.
17. För R=0,5 eller 0,25, fyll på med beräknad packningsmassa.
18. Fyll sand till ~12 cm
    a. notera massan av sanden
19. Notera att MFC FR är öppen

20. Kalibrera LabVIEW med “calibrate”-verktyget.
21. Ange volymflöde för MFC
    a. se excel för vilket volymflöde
    b. kontrollera att volymflödet inte överstiger 400l/min för MFC FR. Om det gör det, genomför steg 13.
    c. Spara med rätt filnamn (DATUM_SAND_PACKNING_VF_R) och i rätt folder
    d. spara tryckdata i 1 minut med “Save”-verktyget
    e. anteckna observationer i lab-boken
    f. bedöm om det är värt att fortsätta experimentet (se steg 14).
22. Upprepa steg 11 med nytt volymflöde.
23. Vid ~400 L/min ändra MFC från FR till AR
    a. Sänk först FR-flödet successivt till noll
    b. Öka AR-flödet successivt till önskat volymflöde
24. Avbryt då slugging/splashing är för överdriven. Se avsnitt “Avsluta Försök”.

Avsluta försök

25. Töm cylindern på så mycket sand som är möjligt
26. Separera sand från packning med en sil
27. Rengör cylindern med dammsugare
28. Skicka datafilerna till Google Drive-mapp

Notera

- Det kan hamna sand i tuberna för tryckfallsmätning
    - om så är fallet, stanna experimentet och rengör.

B Stegringschema hastigheter

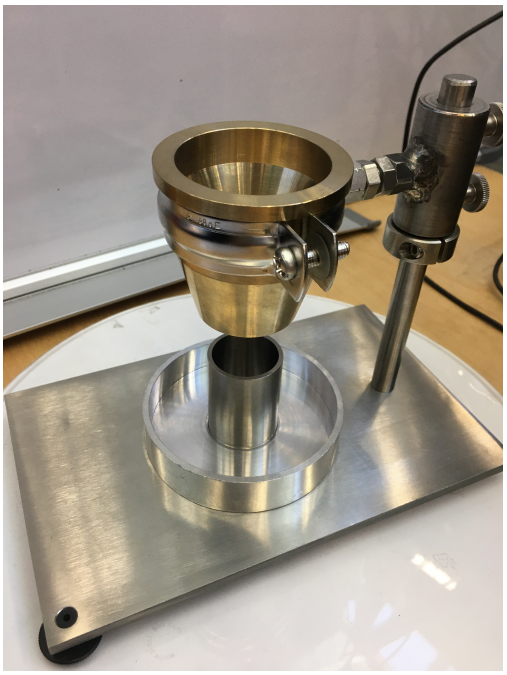
Gas Velocity [m/s]	Input Volumetric air flow [Nl/min]
0,01	7
0,02	14
0,03	20
0,04	27
0,05	34
0,06	41
0,07	47
0,08	54
0,09	61
0,1	68
0,11	75
0,12	81
0,13	88
0,14	95
0,15	102
0,2	136
0,25	170
0,3	203
0,35	237
0,4	271
0,45	305
0,5	339
0,55	373
0,6	407
0,65	441
0,7	475
0,75	509
0,8	543
0,85	577
0,9	610
0,95	644
1	678

C Figurer - Apparatur


Figur 35: Bilder på dammuppsamling (vänster) och massflödesreglering (höger)


Figur 36: Bilder på silar (vänster) och silmaskin (höger)

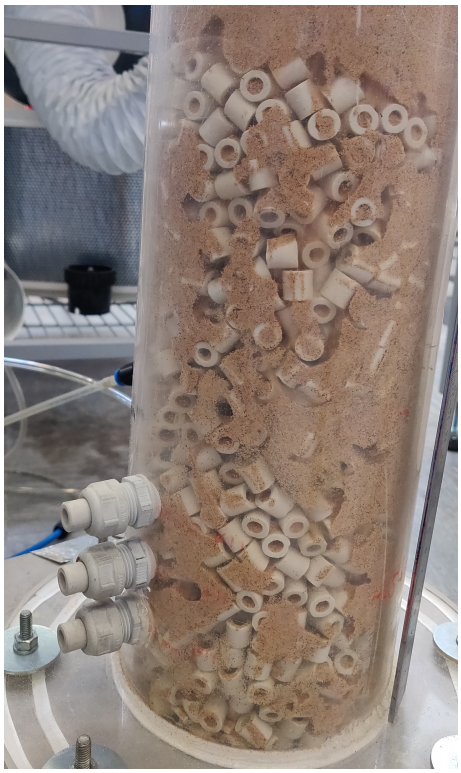
Figur 37: Bilder på bäddsensorer (vänster) och kammarsensor (höger)



Figur 38: Bild på instrument för att bestämma sandens densitet.

D Figurer - Resultat

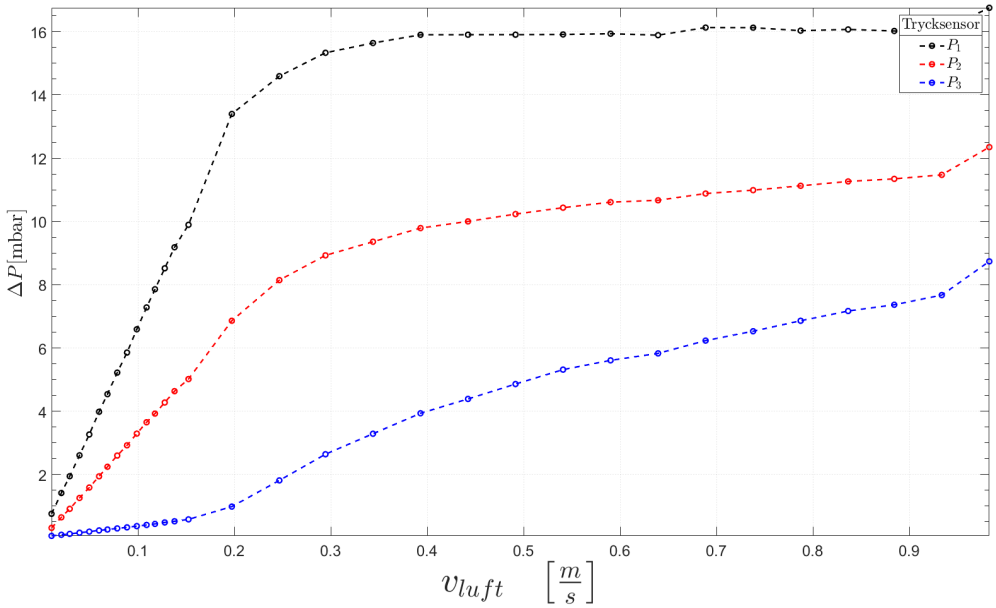
I huvudtexten gjordes ett urval av de resultat som var mest intressanta att diskutera. Nedan finnes tryckfall för övriga experiment som inte presenterades i huvudtexten.



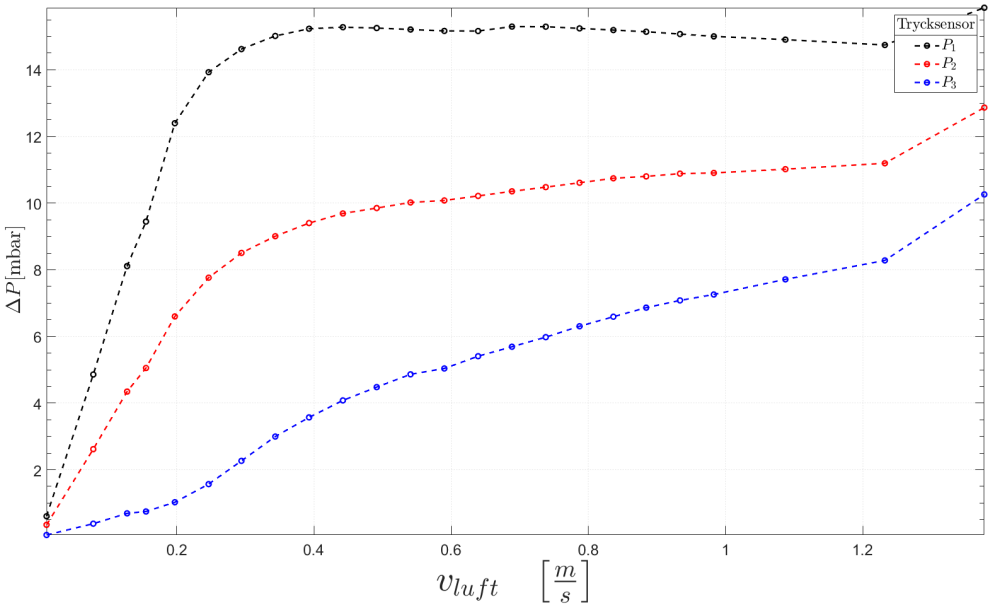
Figur 39: Vertikal segregation av sand för Raschigringar, R = 0.5.

Figur 40: Sand som rinner ner efter att ha hamnat på packningen.

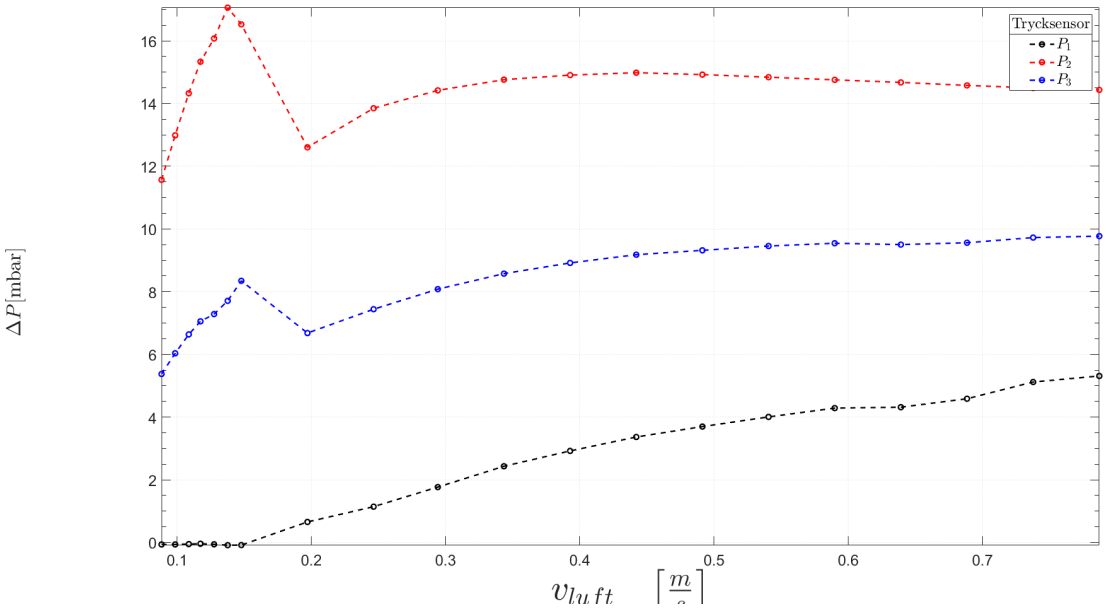
Figur 41: Sluggning i Sand II, djup bädd.



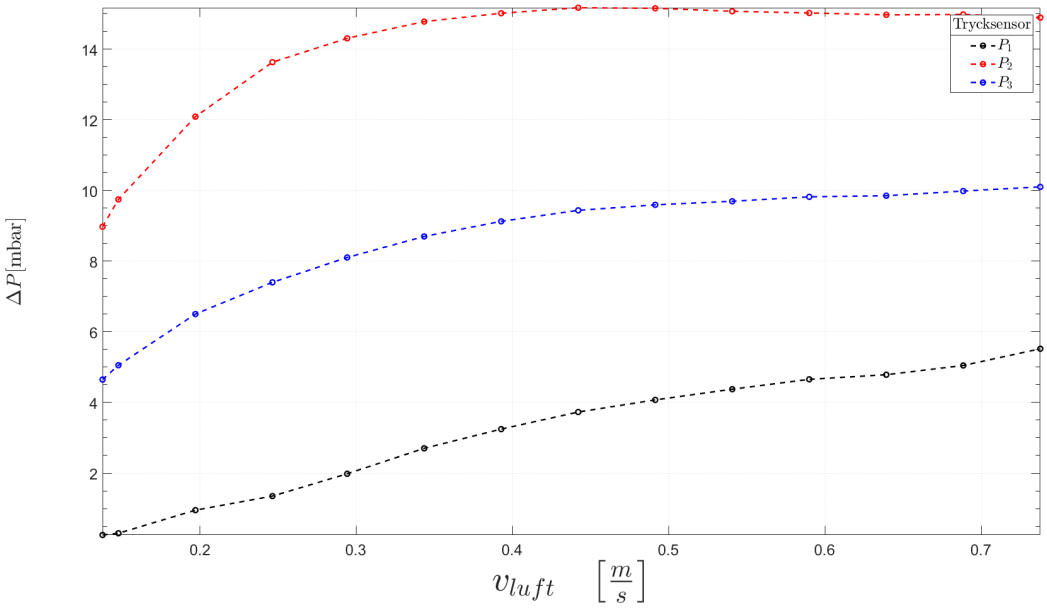
Figur 42: Tryck vid ökad gasgenomströmmning för RMSR 25-3 Sand II, R = 0.5.



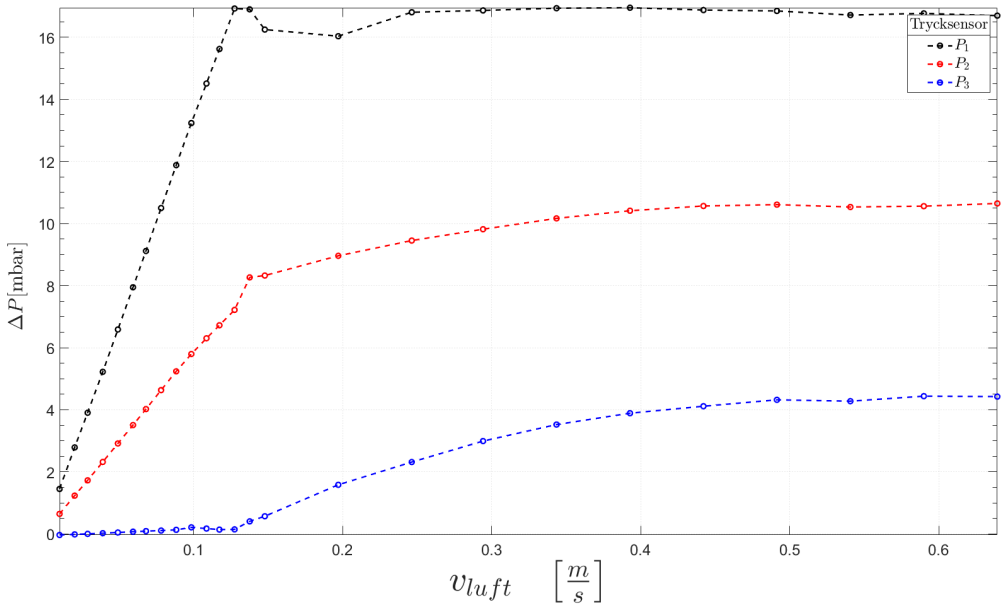
Figur 43: Tryck vid ökad gasgenomströmmning för RMSR 25-3 Sand II, R = 0.25.



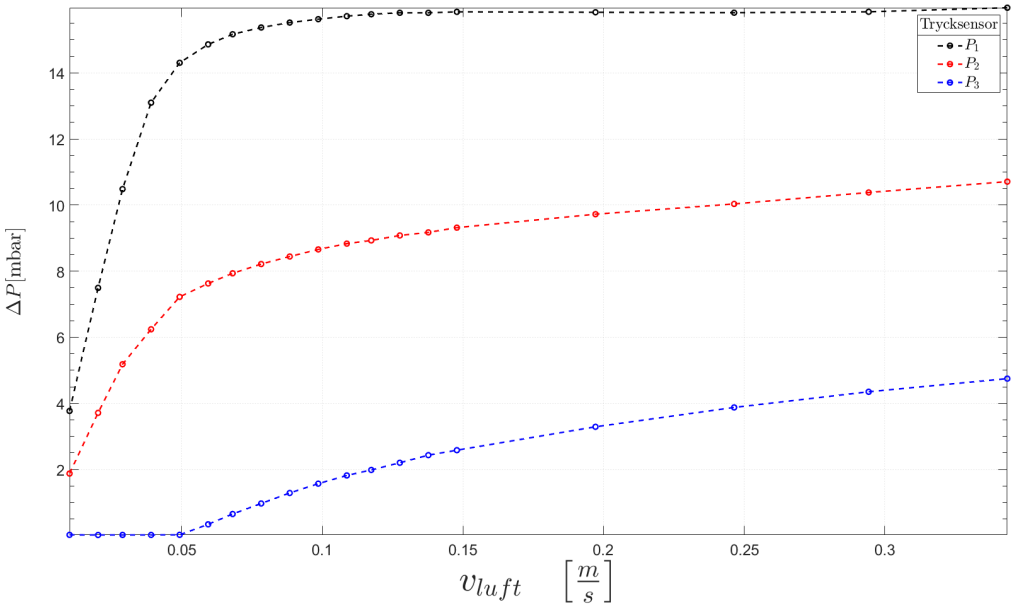
Figur 44: Tryck vid ökad gasgenomströmmning för Hiflow-ringar Sand II, R = 0.5.



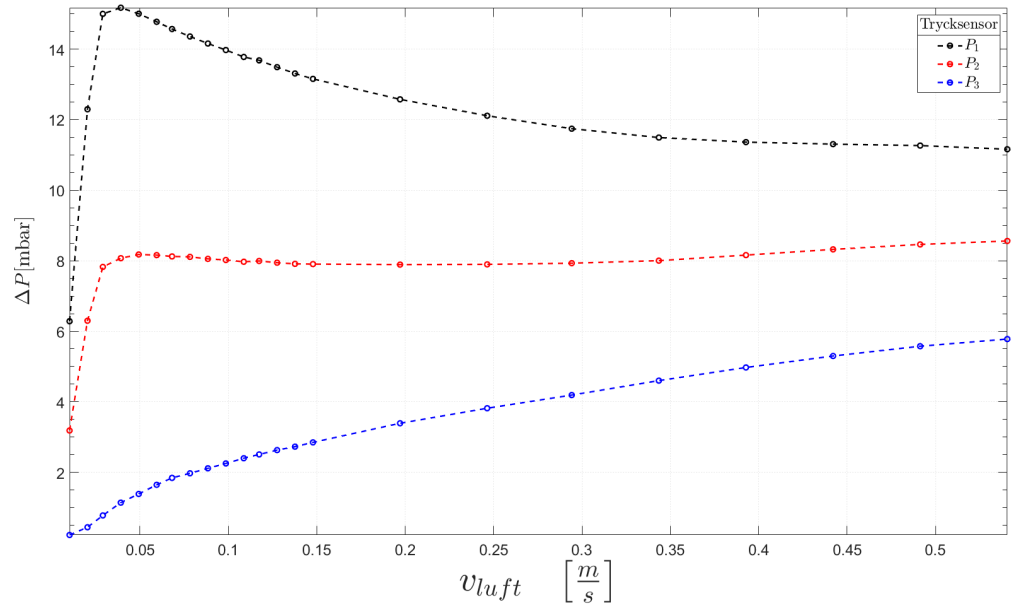
Figur 45: Tryck vid ökad gasgenomströmmning för Hiflow-ringar Sand II, R = 0.25.



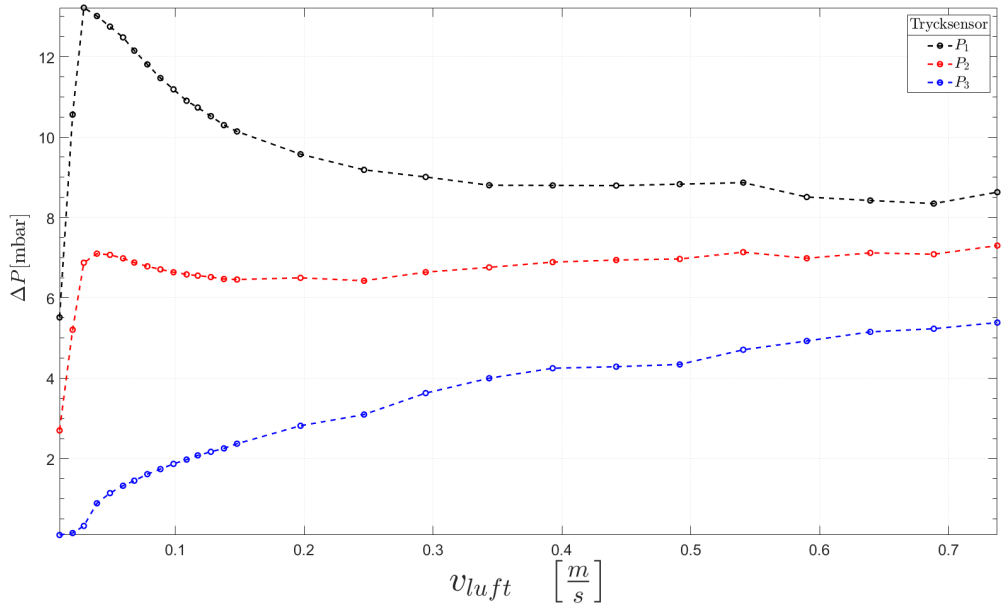
Figur 46: Tryck vid ökad gasgenomströmmning för icke packad bädd, Sand I.



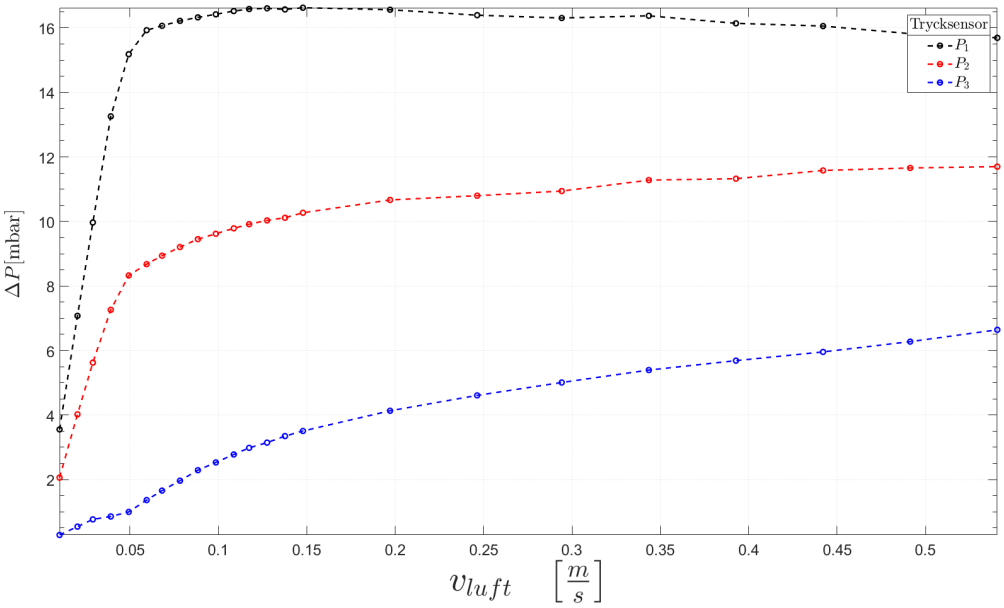
Figur 47: Tryck vid ökad gasgenomströmmning för RMSR 25-3, Sand I, R = 1



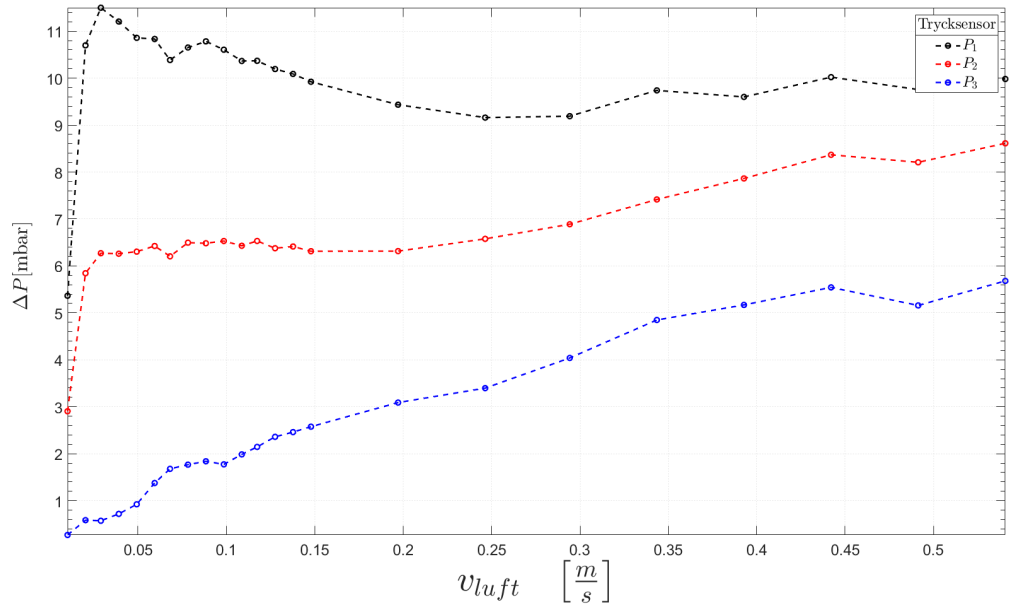
Figur 48: Tryck vid ökad gasgenomströmmning för ASB 1”, Sand I, R = 1



Figur 49: Tryck vid ökad gasgenomströmmning för ASB 1/2”, Sand I, R = 1



Figur 50: Tryck vid ökad gasgenomströmmning för Hiflow-ringar, Sand I, R = 1



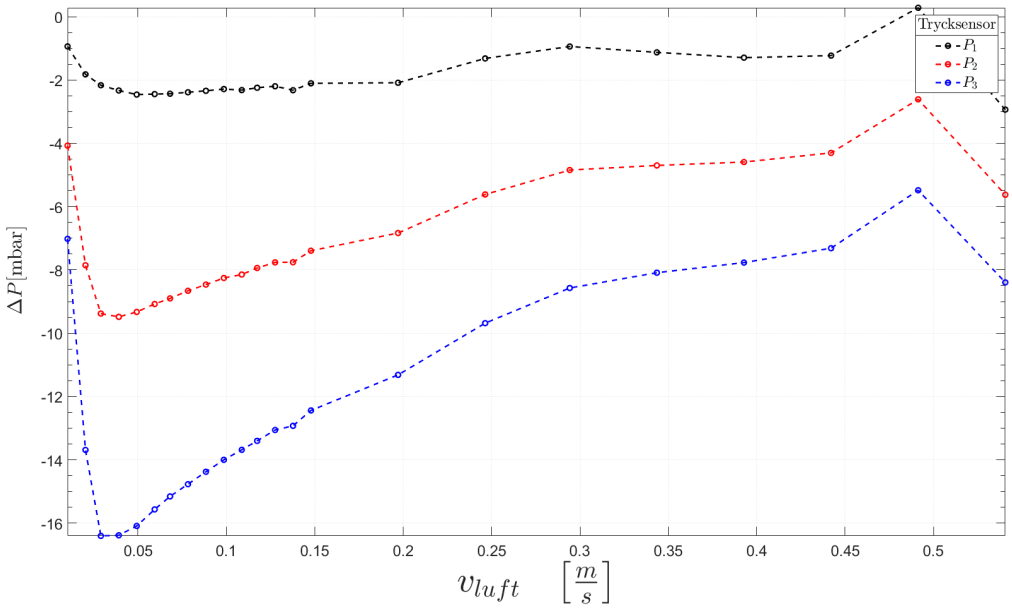
Figur 51: Tryck vid ökad gasgenomströmmning för Raschigringar, Sand I, R = 1

D.1 Tryckfall över hela bädden

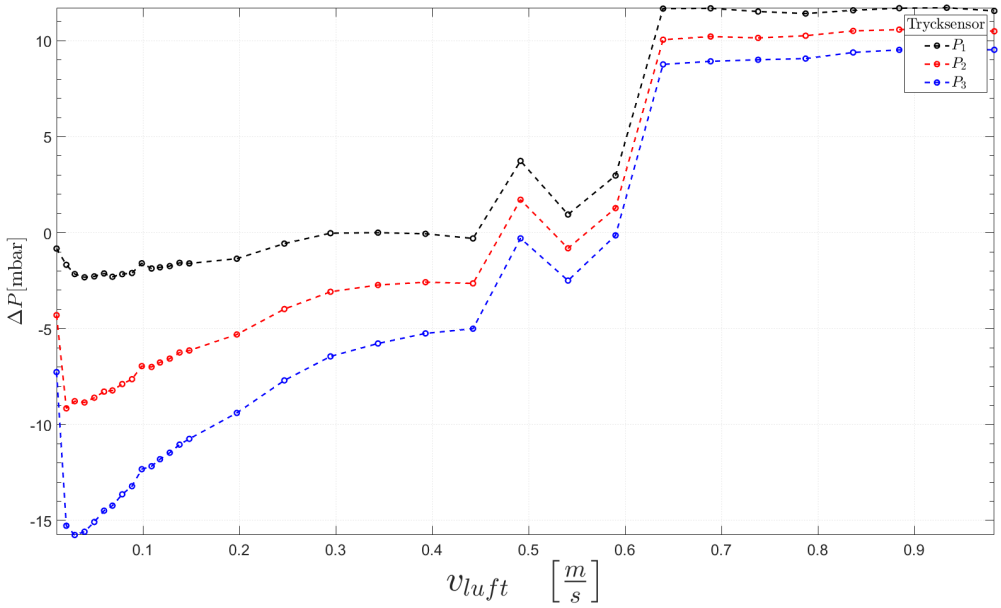
Tryckfallet över hela bädden beräknas på följande vis

$$\Delta P = P_{\text{windbox(Bädd)}} - P_{\text{windbox(Tom bädd)}} - P_{atm}$$

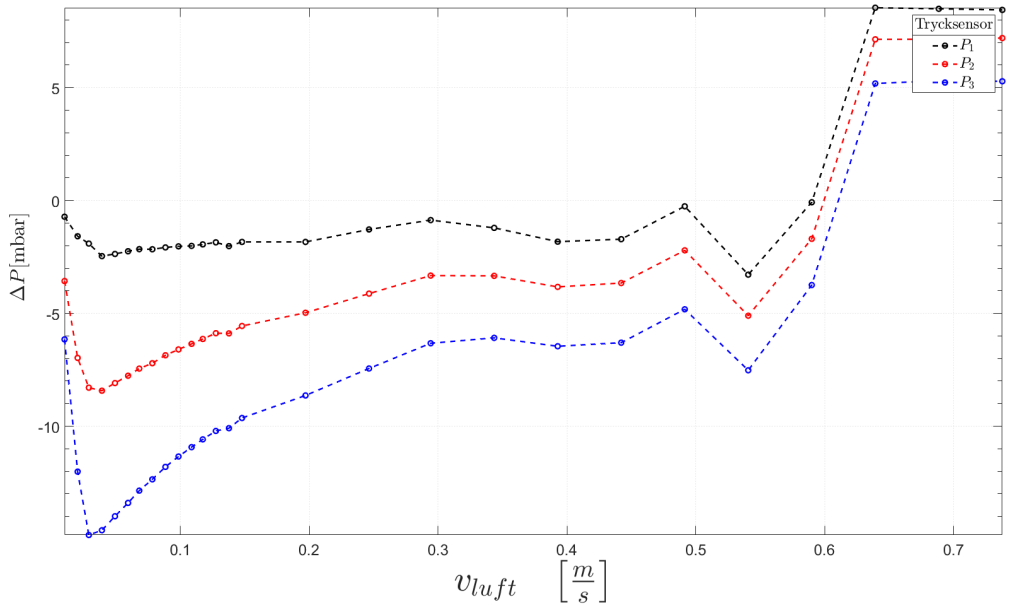
Notera att de fall då trycket planar ut i slutet innebär att trycksensorn för $P_{windbox}$ nått sitt maxtryck.



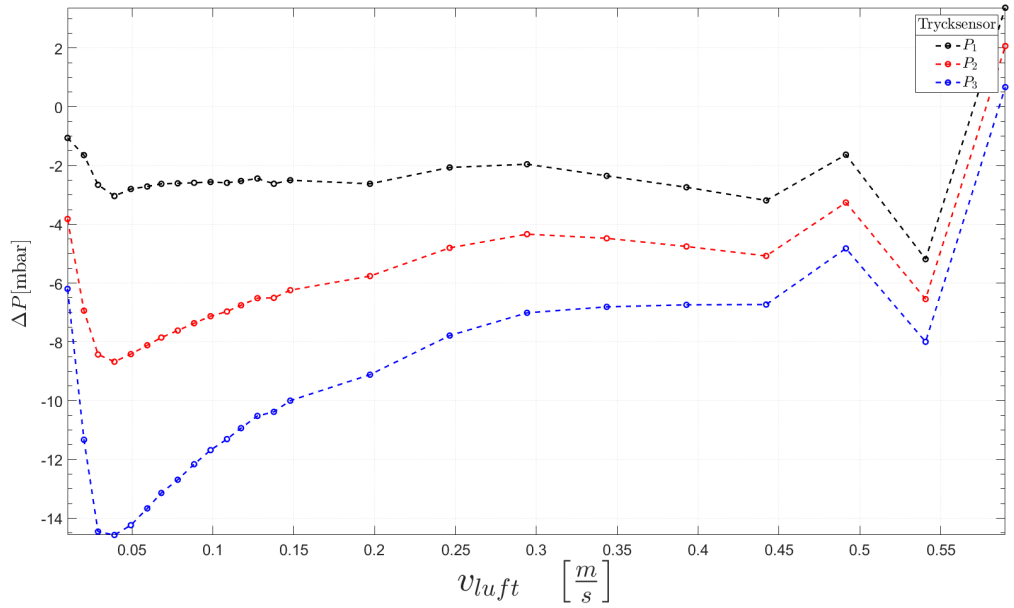
Figur 52: Tryckfall över hela bädden vid ökad gasgenomströmmning för ASB 1", Sand I, R = 1



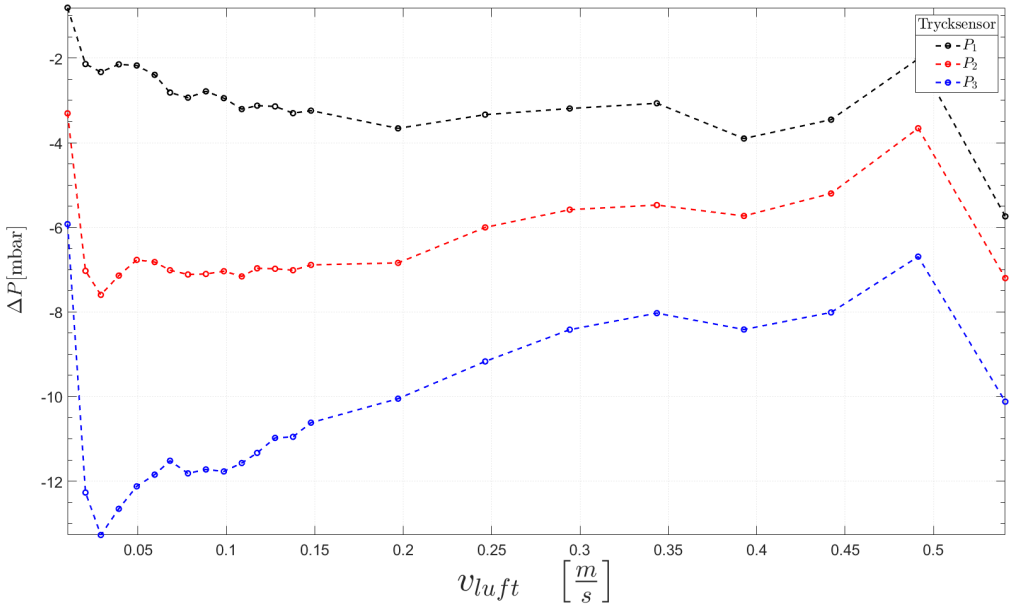
Figur 53: Tryckfall över hela bädden vid ökad gasgenomströmmning för ASB 1”, Sand I, R = 0.5



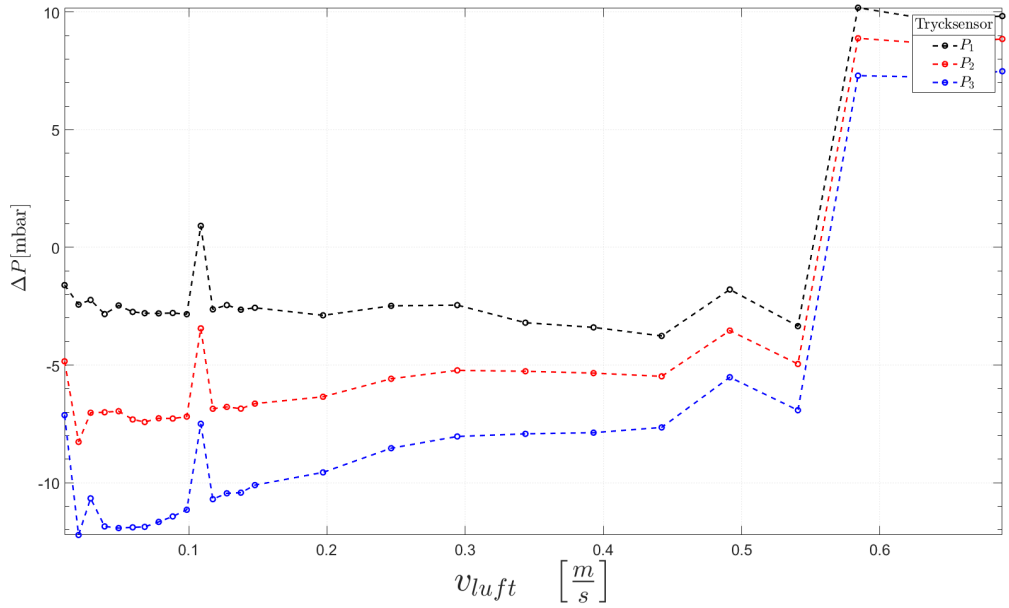
Figur 54: Tryckfall över hela bädden vid ökad gasgenomströmmning för ASB 1/2”, Sand I, R = 1



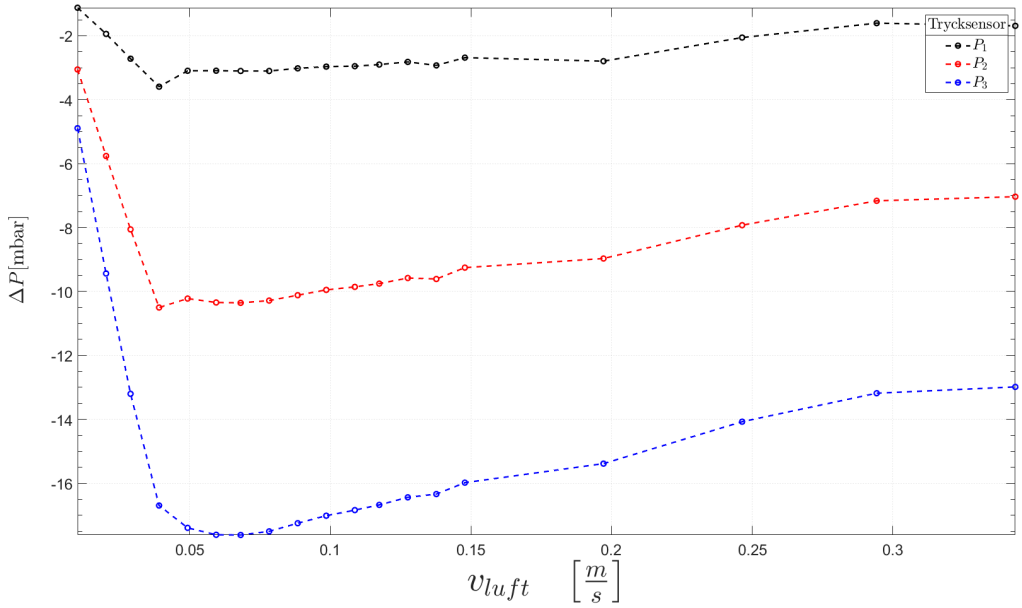
Figur 55: Tryckfall över hela bädden vid ökad gasgenomströmmning för ASB 1/2”, Sand I, R = 0.5



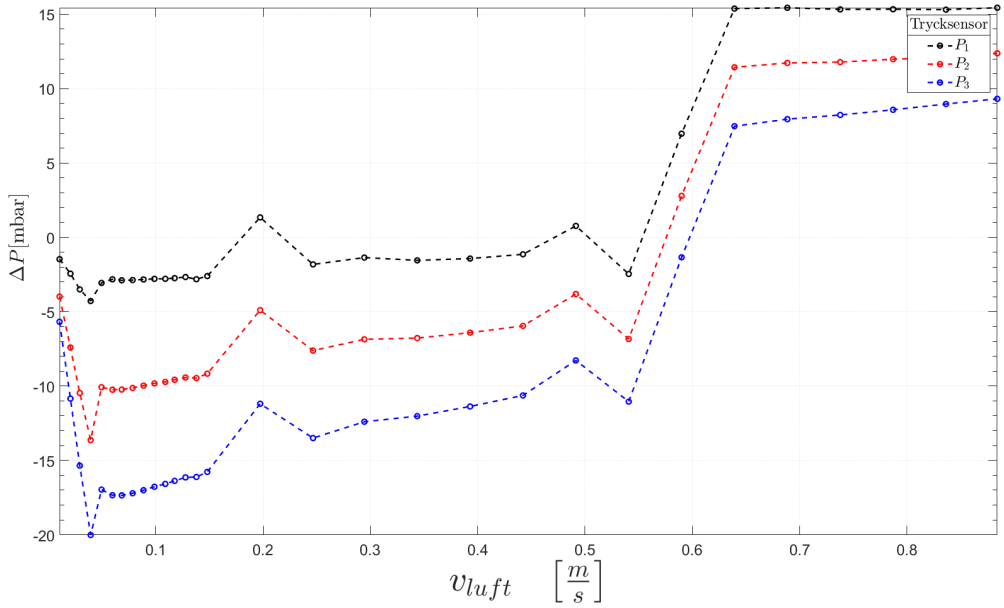
Figur 56: Tryckfall över hela bädden vid ökad gasgenomströmmning för Raschigringar, Sand I, R = 1



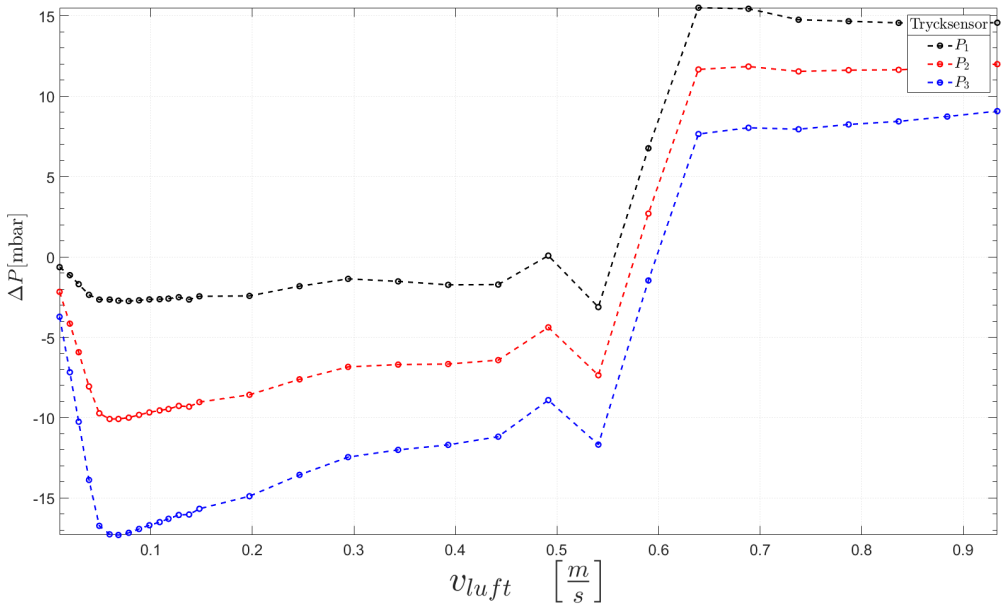
Figur 57: Tryckfall över hela bädden vid ökad gasgenomströmmning för Raschigringar, Sand I, R = 0.5



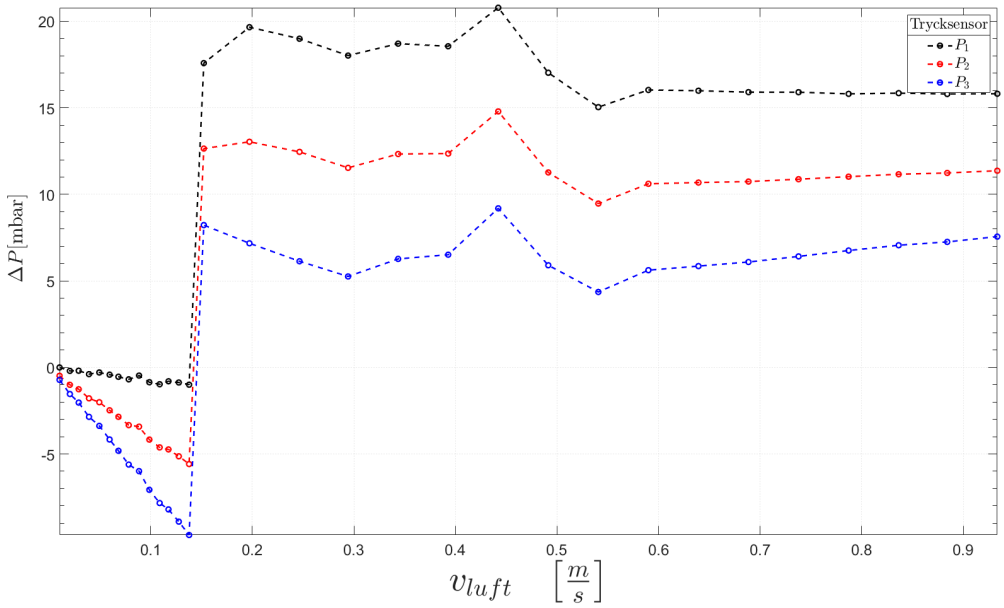
Figur 58: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3, Sand I, R = 1



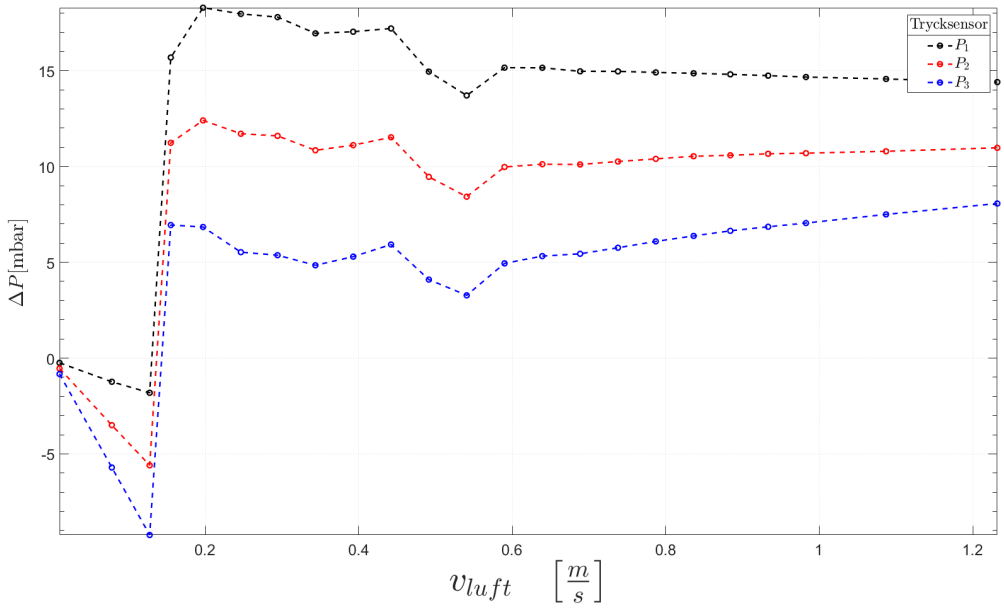
Figur 59: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3 Sand I, R = 0.5



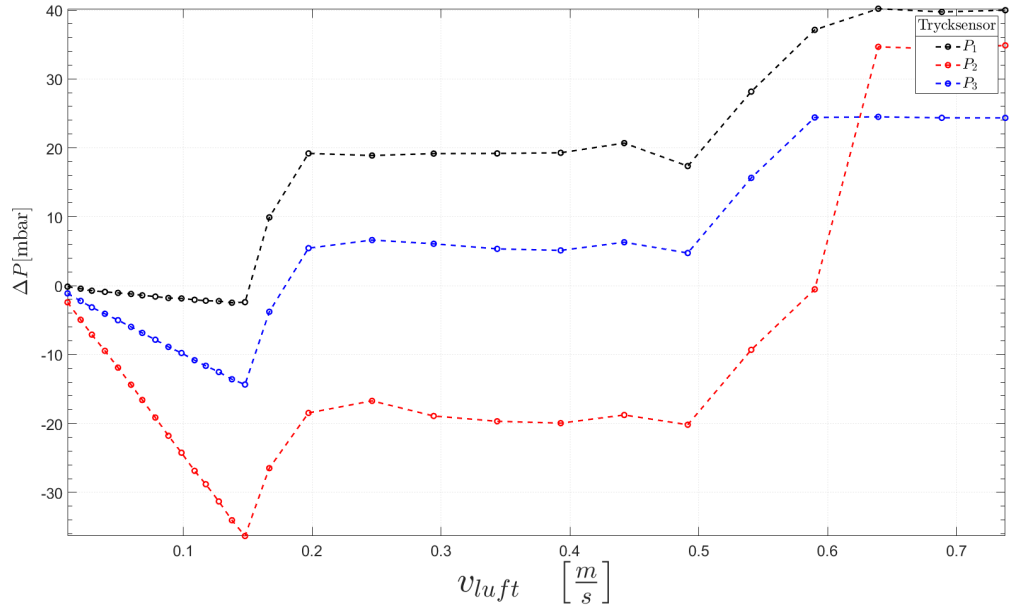
Figur 60: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3, Sand I, R = 0.25



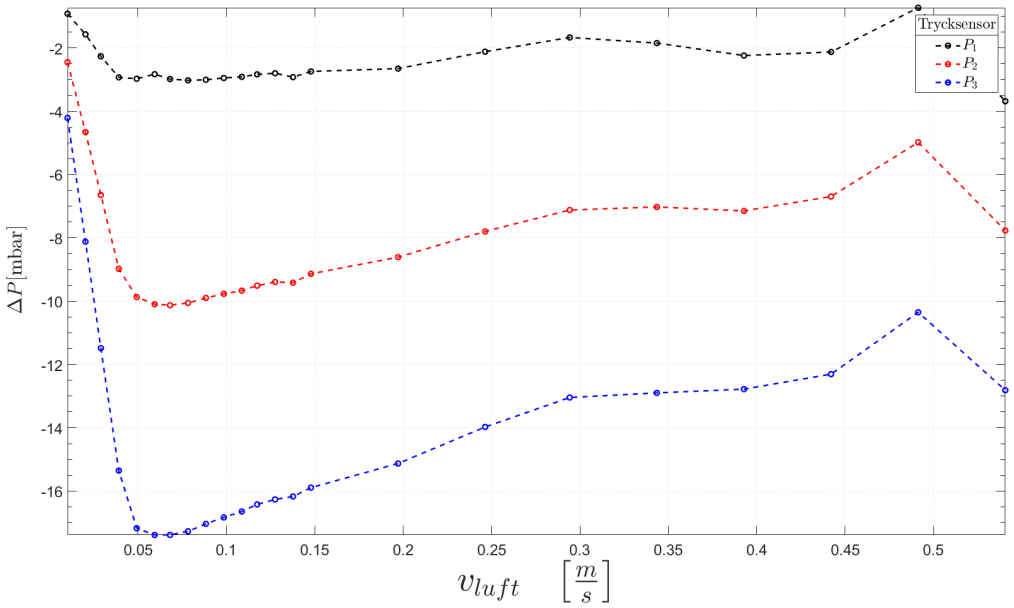
Figur 61: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3, Sand II, R = 0.5



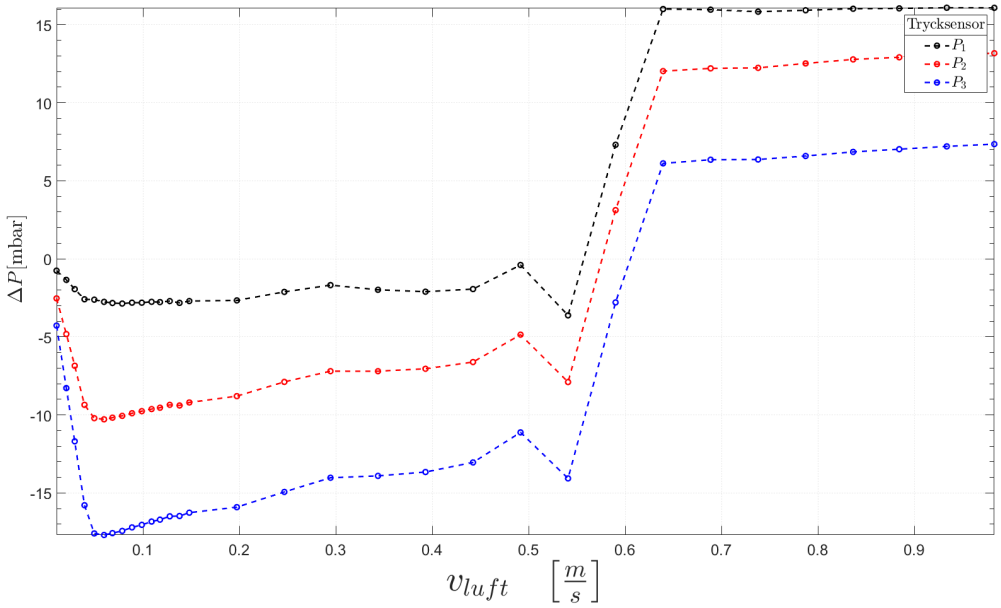
Figur 62: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3, Sand II, R = 0.25



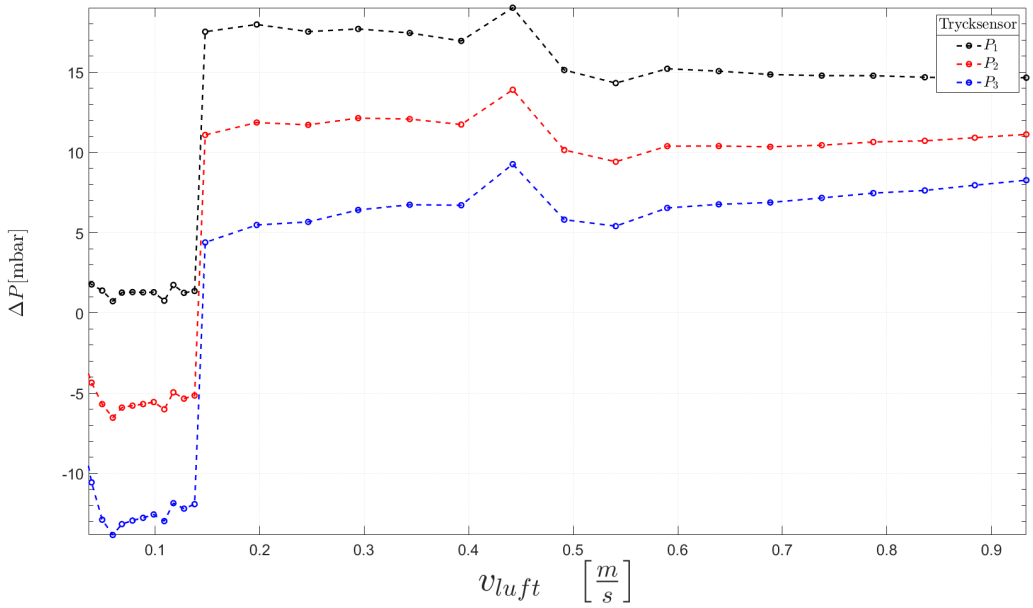
Figur 63: Tryckfall över hela bädden vid ökad gasgenomströmmning för RMSR 25-3 djup bädd, Sand II, R = 1



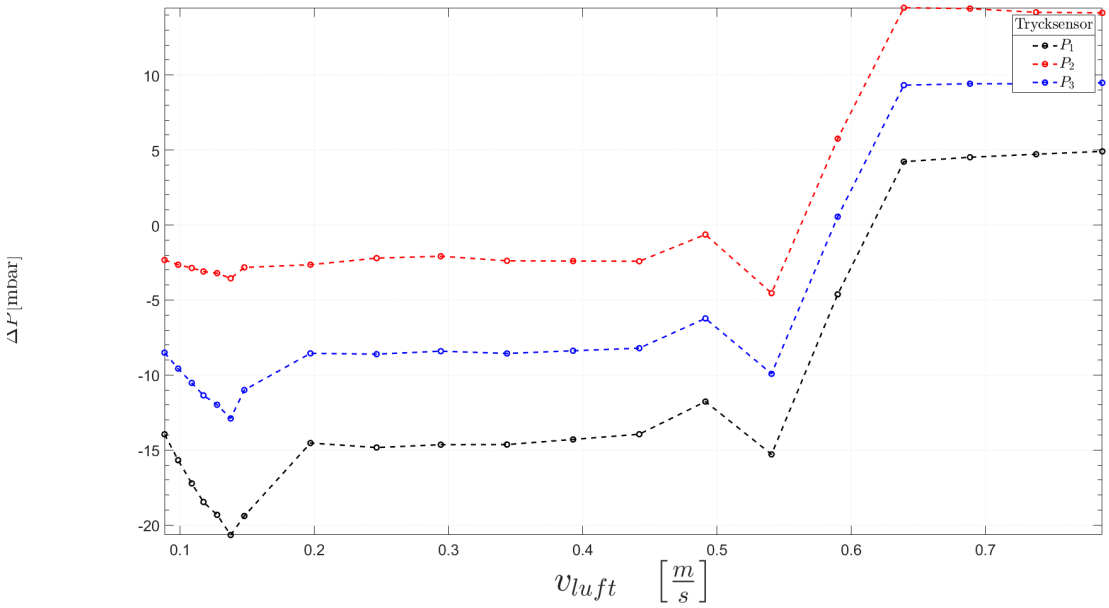
Figur 64: Tryckfall över hela bädden vid ökad gasgenomströmmning för Hiflow-ringar, Sand I, R = 1



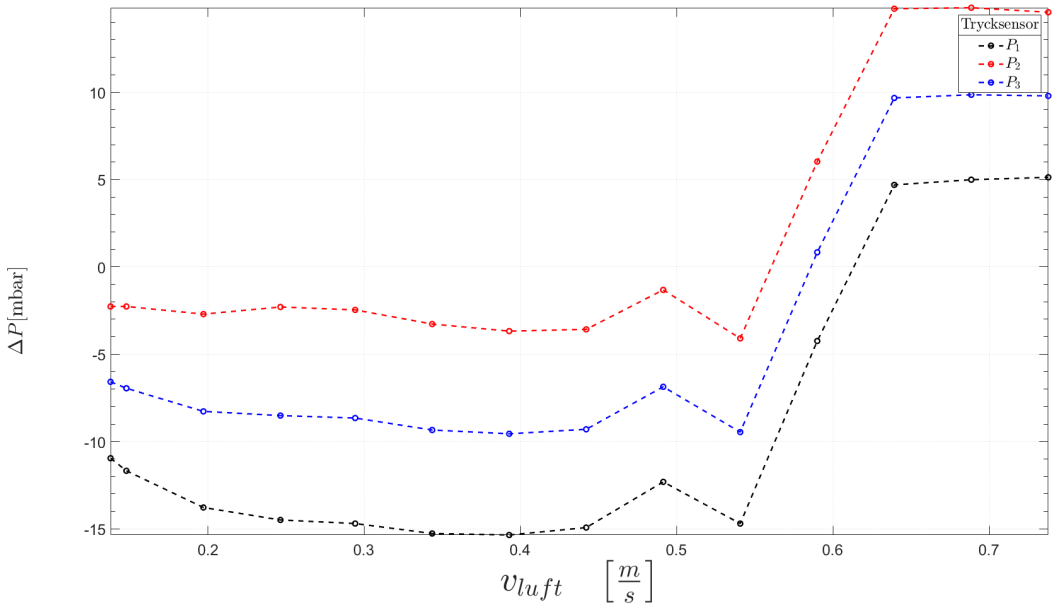
Figur 65: Tryckfall över hela bädden vid ökad gasgenomströmmning för Hiflow-ringar, Sand I, R = 0.5



Figur 66: Tryckfall över hela bädden vid ökad gasgenomströmmning för Hiflow-ringar, Sand I, R = 0.25



Figur 67: Tryckfall över hela bädden vid ökad gasgenomströmmning för Hiflow-ringar, Sand II, R = 0.5



Figur 68: Tryckfall över hela bädden vid ökad gasgenomströmmning för Hiflow-ringar, Sand II, R = 0.25

E MATLAB

E.1 BachelorThesisSEEX15-20-1.m

```
% Bachelor Thesis SEEX15-20-1
% ================================================================================
%
% Copyright (c) 2019--2020, Bengtsson Maarten
% All rights reserved.
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are
% met:
%
%     * Redistributions of source code must retain the above copyright
%       notice, this list of conditions and the following disclaimer.
%     * Redistributions in binary form must reproduce the above copyright
%       notice, this list of conditions and the following disclaimer in
%       the documentation and/or other materials provided with the distribution
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
%
% ================================================================================
% ================================================================================
tic;
clc
workspace
% ================================================================================
% ------------------------------ GENERAL COLUMN INDEXING ---------------------------
% 1 = X-Value       COMMENT: Number of sec passed.
% 2 = AR1-AR3       COMMENT: 2nd above distributor.
% 3 = AR3-AR5       COMMENT: Nonsense.
% 4 = AR5-AR8       COMMENT: Nonsense.
% 5 = LS1-CY1       COMMENT: Nonsense
% 6 = FR7-LS1       COMMENT: Nonsense
% 7 = FR1-FR4       COMMENT: Nonsense
% 8 = FR4-FR7       COMMENT: Nonsense
% 9 = FR7-FR8       COMMENT: Nonsense
% 10 = LS2-FR1      COMMENT: Nonsense
% 11 = LS2-DC1      COMMENT: Nonsense
% 12 = DC1-CY2      COMMENT: Nonsense
% 13 = LS2F-LS2     COMMENT: Nonsense
% 14 = LS3-FR4      COMMENT: P_window
% 15 = FR7-CY3      COMMENT: 3rd above distributor.
% 16 = LS3-CR3      COMMENT: 1st above distributor.
% 17 = CR1-CR3      COMMENT: Value.
% 18 = CR3-CY3      COMMENT: Nonsense
% 19 = CY3-CS1      COMMENT: Nonsense
% 20 = CS4-CSfB     COMMENT: Nonsense
% 21 = CSfB-FR7     COMMENT: Nonsense
% 22 = LS4-CSfB     COMMENT: Nonsense
% 23 = LS4-AR5      COMMENT: Nonsense
% 24 = ARwb-atm     COMMENT: Nonsense
% 25 = FRsb-atm     COMMENT: Nonsense
% 26 = LS1wb-atm    COMMENT: Nonsense
% 27 = LS2wb-atm    COMMENT: Nonsense
% 28 = LS3wb-atm    COMMENT: Nonsense
% 29 = LS4wb-atm    COMMENT: Nonsense
% 30 = CRwb-atm     COMMENT: Nonsense
% 31 = CSwb-atm     COMMENT: Nonsense
```

```
% 32 = Bag height   COMMENT: Nonsense
% 33 = CSfb-CS1     COMMENT: Nonsense
% 34 = LS1wb-LS!    COMMENT: Nonsense
% 35 = FR1-atm      COMMENT: Nonsense
% 36 = FR4-atm      COMMENT: Nonsense
% 37 = FR7-atm      COMMENT: Nonsense
% 38 = Optical_1    COMMENT: Nonsense
% 39 = Optical_2    COMMENT: Value.
% 40 = Optical_3    COMMENT: Value.
% 41 = Optical_4    COMMENT: Value.
% 42 = Optical_5    COMMENT: Value.
% 43 = Optical_6    COMMENT: Value.
% 44 = Weight       COMMENT: Value.
% 45 = Resampled    COMMENT:
% 46 = Resampled 1  COMMENT: Value.
% 47 = Resampled 2  COMMENT: 0
% 48 = Resampled 3  COMMENT: 0

% Old code for dropBox download.
% % Get directory form downloads. This will vary from user to user.
% %cd 'C:\Users\Tinki\Downloads\drive-download-20200218T171112Z-001'
% data = dir('20200213_S1_Saddle_V*'); % order and sort the files
% N = length(data);
% startRow = 25;
% endRow = 3000;
% filename = data.name;
% fid = SEEX15_20_1_importfile(filename,startRow,endRow);
%
% result = mean(table2array(fid))
%
% result(result < 0.05) = []
% dropboxAccessToken = 'jx-OodfX_v0AAAAAAAAEohQQ5isUFke2dop56Kee8KnWWhq09FreXVWQnSRsLwQs'
% fileNames = {'https://www.dropbox.com/home/Appar/BachelorThesisSEEX15'}
% downloadFromDropbox(dropboxAccessToken,fileNames)

% Get correct path.
% cd C:\Users\Tinki\Desktop\BachelorThesis\
% addpath(genpath('C:\Users\Tinki\Desktop\BachelorThesis\'))

% If on school computer change directory to thsi and comment out the section
% above.


if isfolder('C:\Users\Tinki\Desktop\BachelorThesis\')
    cd('C:\Users\Tinki\Desktop\BachelorThesis')
    addpath(genpath('C:\Users\Tinki\Desktop\BachelorThesis\'))
else
    cd('\\file00.chalmers.se\home\benmaar\.win\Desktop\BachelorThesisSchoolComputer')
    addpath(genpath('\\file00.chalmers.se\home\benmaar\.win\...
    Desktop\BachelorThesisSchoolComputer'))
end

% mkdir('C:\Users\Tinki\Desktop\BachelorThesis\')
% We need to come up with a way to programatically add folders and
% subfolders to current path.

% Determine where your m-file's folder is.
% folder = fileparts(which('BachelorThesisSEEX15_20_1.m'));

% Add that folder plus all subfolders to the path.
% addpath(genpath(folder));

% get filename, this will change for each anlaysis
data = dir(fullfile('**/*.txt'));
sortedData = natsortfiles({data.name});

% Initialize start and endpoint for data analysis.
startRow = 25;
endRow = 3000;

% Preallocation
N = length(data);
processedData = cell(1, N);
```

```
rawData = cell(1, N);
pressure = cell(1, N);
rawName = cell(1, N);

% ================================================================================
%====================== SORTING AND ANALYZE ======================================
% ================================================================================

% for-loop just to extract neccessary data.
for i = 1:N

    % Get correct filename
    filename = char(sortedData{i});

    % Get data
    fid = importfile(filename, startRow, endRow);
    X = mean(fid, 1);

    % Allocate data.
    pressure{i} = X;
    rawData{i} = fid;
    rawName{i} = filename;
end

% Get valid valid variable names for table.
validVariableName = matlab.lang.makeValidName(sortedData);

validRowNames = validVariableName;
validVarNames = {'P_window', 'P_1', 'P_2', 'P_3'};

% Preallocation.
P_window = cell(1, N);
P_1 = cell(1, N);
P_2 = cell(1, N);
P_3 = cell(1, N);

% Index for pressure taken from master column index.
idx = [14, 16, 2, 15];

% for-loop to make pressure values easier to access.
for i = 1:N
    P_window{i} = pressure{1, i}(idx(1));
    P_1{i} = pressure{1, i}(idx(2));
    P_2{i} = pressure{1, i}(idx(3));
    P_3{i} = pressure{1, i}(idx(4));
end

% Generate table to clearly see each data point.
T = table(P_window', P_1', P_2', P_3', 'VariableNames', validVarNames, ...
    'RowNames', validRowNames);

% Save table as an Excel-file.
writetable(T, 'BcsMatlab.xlsx', 'WriteRowNames', true);
% T(:, contains(T.Properties.RowNames, 'x20200225_S1_RR10_R05_'))

% ================================================================================
saveTime(1) = toc;

%% Get Data For Sand 1
%============================ PLOTTING DATA ======================================
% ================================================================================

% Match files with table to extract correct values. This will be tedious.
% Excract data from the table. Make it easy MxN- matrices that we can plot.
% Add more for more experiments.

% ================ Everything belowe this is Sand I ============================
% ================================================================================
%--------------------------------------------------------------------------------
%--------------------------------------------------------------------------------
% Extract values for plotting and further research.
%--------------------------------------------------------------------------------

```

```
% X-values to plot against.
% valuesX = [7, 14, 20, 27, 34, 41, 47, 54, 61, 68, 75, 81, 88, 95, 102, ...
%     136, 170, 203, 237, 271, 305, 339, 373, 407, 441, 475, 509, 543, ...
%     577, 610, 644, 678];

% Display correct name for value matching each packing witch each run of gas
% flow. i.e 7,14-20 L/min Rashing Ring matching 20200225_S1_RR10_R05_V7,
% 0200225_S1_RR10_R05_V14 etc.
strMatchingY = {'x20200213_S1_V', 'x20200225_S1_RR10_V', ...
    'x20200225_S1_RR10_R05_', 'x20200213_S1_Saddle_V', ...
    'x20200218_S1_Saddle_R_5_', 'x20200218_S1_Saddle_R_25_', ...
    'x20200228_S1_Hiflow_R1_', 'x2020028_S1_hiflow_R0_5', ...
    'x20200303_S1_hiflow_R0_25_', 'x20200303_S1_ASB1_R1_', ...
    'x20200305_S1_ASB1_R0_5_', 'x20200306_S1_ASB0_5_R1_', ...
    'x20200403_S1_ASB0_5_R0_5_', 'x20200414_empty_bed_V'};

% Note strMatchingY is kind off the master cell that controls the rest of
% the code. Very important to have everytthing in it fro running plots,
% minimal fluidization velocity etc etc.


% Preallocation.
numTest = length(strMatchingY);
valuesY = cell(1, numTest);
newVar = cell(1, numTest);
valuesX = cell(1, numTest);
Pavg = cell(1, numTest-1);
Psensor = cell(1, numTest-1);
Pdistributor = cell(1, 1);

for i = 1:numTest

    % Extract correct series of experiment.
    idx = ~cellfun('isempty', regexp(T.Properties.RowNames, ...
        strMatchingY{i}, 'once'));
    name = T(T.Properties.RowNames(idx), :);
    % Get numeric values from string to proper X-values.
    X = (regexp(name.Properties.RowNames, '\d+[\.]?\d*', 'match'));

    % Preallocation.
    x = zeros(1, length(X));

    % Access all valuees using for-loop.
    for k = 1:length(X)
        x(k) = str2double(cell2mat(X{k}(end)));
    end

    % Get numeric values from string to proper X and Y values.
    valuesX{i} = x;
    valuesY{i} = cell2mat(table2array(name(:, 1:end)));
end

% Assign pressure values to calculate pressure drop over ditributor.
for i = 1:numTest - 1
    Pavg{i} = valuesY{i};
end


% Assing pressure values.
valuesXsubstract{1} = x';
valuesYsubstract{1} = valuesY{end}; % Must be empy bed index 14.

% % Substract empy bed values from pressure measures values.
% numRow = size(valuesY{1}, 1);
% numCol = size(valuesYsubstract{1}, 2);
% for i = 1:numTest
%     for k = 1:numRow
%         for p = 1:numCol
%             y = valuesY{i}(k, p) - valuesYsubstract{1}(k, p);
%             newVar{i}(k, p) = y;
%         end
%     end
% end
```

```
286
287 % To calculate pressure drop, first we need to calculate the pressure drop
288 % over distributor plate as below:
289
290 % Pressure drop over distributor (at a certain velocity)= average pressure
291 % of one of the 3 sensors (optional) above the distributor plate at empty
292 % conditions (empty bed with no packings and no sand, at a certain velocity)
293 % average pressure of windbox (empty bed with no packings and no sand,
294 % at the same velocity).
295
296 % Get pressure drop empty bed for 1 sensor, lets use P_2.
297 PemptySensor = valuesYsubstract{1}(:, 2);
298 PemptyWindbox = valuesYsubstract{1}(:, 1);
299 % Get Pressure drop over distributor.
300 Pdistributor{1} = PemptySensor - PemptyWindbox;
301
302
303 % Try on Sadlle R =0.5. idx = 5.
304
305 % Psadel = Pavg{3}(1:length(Pdistributor{1}), :);
306
307 % P = Psadel - Psadel(:, 1) - Pdistributor{1};
308
309 fun = @(x, y, z) x - y - z;
310
311 % Substract values of empty from values given by sand III.
312 for i = 1:numTest
313     % Get intersect.
314     [C, idxRow, idxCol] = intersect(valuesXsubstract{1}, ...
315         valuesX{i}, 'stable');
316
317     % Substract empty bed from regular bed.
318     ii = length(idxRow);
319     jj = length(valuesY{i});
320     numIter = min(ii, jj);
321
322     for k = 1:numIter
323         valuesY{i}(k, :) = bsxfun(@minus, valuesY{i}(k, :), ...
324             valuesYsubstract{1}(idxRow(k), :));
325         % Take advantage of the for loop. Psensor size == vlauesXsubstract.
326         if i < numTest
327             x = bsxfun(@minus, Pavg{i}(k, :), ...
328                 Pdistributor{1}(idxRow(k), :));
329
330             % Get pressure drop over entire bed.
331             Psensor{i}(k, :) = bsxfun(@minus, x, Pavg{i}(k, 1));
332         else
333         end
334     end
335 end
336
337
338 % valuesY = newVar;
339 % valuesY  = newVar;
340 saveTime(2) = toc;
341
342 P = valuesY;
343
344 % Preallocation.
345 r = 0.0605;
346 plotDataX = cell(1, numTest);
347
348 for i = 1:numTest
349     % Convert volumtric flow from L/min to [m^3/s]
350     x = valuesX{i} / (1000 * 60);
351
352     % Change plotting data to veolcity instead of volumtetric flow.
353     y = x / (pi * r^2);
354     plotDataX{i} = y;
355 end
356
357 % =========================================================================
358
```

```
%% Plot Calculated Pressure Drop Sand I.

clc
% Names for plotting. Add more for more experiments.
strPressureDrop = {'PD Unpacked Bed', 'PD Rashing Ring 10x10 mmR = 1', ...
    'PD Rashing Ring 10x10 mm, R = 0,5', 'PD Saddle Rings, R = 1', ...
    'PD Saddle Rings, R = 0,5', 'PD Saddle Rings, R = 0,25', ...
    'PD HiFlow Pall Rings, R = 1', 'PD HiFlow Pall Rings, R = 0,5', ...
    'PD HiFlow Pall Rings, R = 0,25', 'PD ASB 1 inch, R = 1', ...
    'PD ASB 1 inch, R = 0,5', 'PD ASB 0.5 inch, R = 1', ...
    'PD ASB 0.5 inch, R = 0,5'};


strLegend = {'$P_1$', '$P_2$', '$P_3$'};
colourString = {'green', 'black', 'red', 'blue'};
%

% Plotting and saving data. Come up with easy managable names for the
% graphs.

% Create Subfolder to save data in. Much neater than before.
if isfolder('C:\Users\Tinki\Desktop\BachelorThesis\Pressure Drop All Figures')
    fPath = 'C:\Users\Tinki\Desktop\BachelorThesis\Pressure Drop All Figures';
else
    cd('\\file00.chalmers.se\home\benmaar\.win\Desktop\BachelorThesisSchoolComputer')
    fPath = 'Z:\.win\Desktop\BachelorThesisSchoolComputer\Pressure Drop All Figures';
end


% Values to change fontsizes nad apperance of figures.
legendFontsize = 18;
axisFontSize = 1.2;
lineWidth = 1.8;
XaxisFontSize = 42;
YaxisFontSize = 24;
masterFontSize = 18;

% Strings for axis,legends,Xlabel and Ylabel.
Xstring = {'$v_{luft} \quad \big[ \frac{m}{s} \big] $'};
Ystring = {'$\Delta P$[mbar]'};
legendTitle = {'Trycksensor'};

% Preallocation for graphical object.
h = gobjects(size(Psensor{1}, 2));
% resolution = 300; % —DPI use double of standard DPI for PNG—file saving.

% Normalise system to use Fullscreen figures, remember to change back since
% you are altering the default settings.
% set(groot, 'defaultFigureUnits','normalized')
% set(groot, 'defaultFigurePosition',[0 0 1 1])

% Get actual screensize.
screenSize = get(0, 'ScreenSize');

% Cut away some of the screen for better grpahs to save. Offset = 70 seems
% to work absolutly best.
offset = 70;
embeddedPosition = [screenSize(1), screenSize(2), ...
    screenSize(3), screenSize(4) - offset];

% screenSize(1),screenSize(2)

for i = 1:numTest - 1 % So we don't include empty bed.

    %------------------------------------------------------------------------
    % Set correct maximazing fo figures in MATLAB

    %         fig = gcf; % only works for  MATLAB R2018a or alter.
    %         fig.WindowState = 'maximized'; % Maximize figure on screen.
    %     % Good for great resolution in plotting.
    % Speed up saving process using set(gcf,'Postiions',...etc...etc...).
    %     set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    set(gcf, 'WindowState', 'fullscreen')
```

```
    % Plot current data.
    for k = 2:size(valuesY{1}, 2)
        h(k) = plot(plotDataX{i}(1:length(Psensor{i})), Psensor{i}(:, k), '--o', ...
            'LineWidth', lineWidth, 'Color', colourString{k});
        hold on
    end
    hold off
    % Initalize axis, figure properties legend properties.
    ax = gca;
    ax.FontSize = masterFontSize;
    set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    fig = gcf;
    set(gcf, 'color', 'w');
    setappdata(gca, 'LegendColorbarManualSpace', 1);
    setappdata(gca, 'LegendColorbarReclaimSpace', 1);
    %     set(ax, 'nextplot', 'replacechildren'); % Forces new plot but keeps
    % current format on legends and axis. Also remove anyoing childrens ;)
    % feature('DefaultCharacterSet', 'UTF8')
    % Set X-axis
    xlabel(Xstring, 'Interpreter', 'latex', 'FontSize', ...
        XaxisFontSize);
    ax.XAxisLocation = 'bottom';
    set(get(gca, 'XLabel'), 'Rotation', 0); % Horizontal text.
    ax.XLimMode = 'auto';
    % Set Y-axis.
    ylabel(Ystring, 'Interpreter', 'latex', ...
        'FontSize', YaxisFontSize);
    ax.YAxisLocation = 'origin';
    ax.YLimMode = 'auto';
    %     set(get(gca, 'YLabel'), 'Rotation', 180); % Horizontal text.
    ylh = get(gca, 'ylabel');
    ylp = get(ylh, 'Position');
    set(ylh, 'Rotation', 90, 'Position', ylp, 'VerticalAlignment', ...
        'middle', 'HorizontalAlignment', 'center')
    ax.YAxis.MinorTick = 'on';

    % Set title.
    %     ax.Title.String = str{i};

    % Set axis properties.
    ax.Title.FontWeight = 'normal';
    ax.Title.FontAngle = 'italic';
    ax.LabelFontSizeMultiplier = axisFontSize;
    ax.TitleFontSizeMultiplier = axisFontSize;
    ax.FontSmoothing = 'on';
    ax.GridLineStyle = ':';
    ax.GridAlpha = 0.2;
    ax.YAxis.Limits = [min(min(Psensor{i}(:, 2:end))), ...
        max(max(Psensor{i}(:, 2:end)))];

    ax.XAxis.Limits = [plotDataX{i}(1), ...
        max(plotDataX{i}(1:length(Psensor{i}(:, 2:end))))];
    ax.TickDirMode = 'auto';

    % Set X and Y-tick properties, mainly changing their FontSize


    % Set legends.
    l = legend('show');
    l.TextColor = 'black';
    l.FontAngle = 'italic';
    l.FontName = 'Comic Sans MS';
    l.FontSize = legendFontsize;
    l.Title.String = legendTitle;
    l.Title.Color = 'black';
    legend(strLegend, 'FontSize', legendFontsize, 'FontAngle', 'italic', ...
        'Location', 'northeast', 'Interpreter', 'latex');
    box on, grid on
    set(gca, 'FontName', 'AvantGarde')

    % Save figures as high-resolution PNG-files.
    newStr = [strPressureDrop{i}, '.png']; % add .png to names.
    saveFile = join(newStr);
```

```
505       %     drawnow;
506       % pause to let the figure fill the whole screen, screencapture is way
507       % to fast for this. Pause = 0.39 seems to work fine for now.
508       pause(0.39)
509
510
511       % NEVER EVER USE PRINTER AS AN OPTION!
512       screencapture(0, 'position', embeddedPosition, ...
513           'target', fullfile(fPath, saveFile))
514       % This is the fastest way to save figures, extremely fast.
515       %     if verLessThan('matlab', '9.8.0')
516       %         % Use only if older than R2020a
517       %         saveas(gcf, fullfile(fPath, saveFile), 'png')
518       %     else
519       %         % Only for matlab R2020a
520       %         exportgraphics(gcf, fullfile(fPath, saveFile), 'Resolution', resolution)
521       %     end
522       %     saveas(gcf, fullfile(fPath, saveFile),'epsc')
523       %     print(gcf,fullfile(fPath, saveFile),'-dpng','-r900');
524       %     F = getframe(gcf);
525       %     imwrite(F.cdata, saveFile);
526       close gcf
527   end
528
529   %% Plot All Data For Sand 1
530
531   clc
532   % Names for plotting. Add more for more experiments.
533   str = {'Unpacked Bed', 'Rashing Ring 10x10 mmR = 1', ...
534       'Rashing Ring 10x10 mm, R = 0,5', 'Saddle Rings, R = 1', ...
535       'Saddle Rings, R = 0,5', 'Saddle Rings, R = 0,25', ...
536       'HiFlow Pall Rings, R = 1', 'HiFlow Pall Rings, R = 0,5', ...
537       'HiFlow Pall Rings, R = 0,25', 'ASB 1 inch, R = 1', ...
538       'ASB 1 inch, R = 0,5', 'ASB 0.5 inch, R = 1', 'ASB 0.5 inch, R = 0,5'};
539
540   strLegend = {'$P_1$', '$P_2$', '$P_3$'};
541   colourString = {'green', 'black', 'red', 'blue'};
542   %
543
544   % Plotting and saving data. Come up with easy managable names for the
545   % graphs.
546
547   % Create Subfolder to save data in. Much neater than before.
548   if isfolder('C:\Users\Tinki\Desktop\BachelorThesis\Sand1Graphs\')
549       fPath = 'C:\Users\Tinki\Desktop\BachelorThesis\Sand1Graphs\';
550   else
551       cd('\\file00.chalmers.se\home\benmaar\.win\Desktop\BachelorThesisSchoolComputer')
552       fPath = 'Z:\.win\Desktop\BachelorThesisSchoolComputer\Sand1Graphs';
553   end
554
555
556   % Values to change fontsizes nad apperance of figures.
557   legendFontsize = 18;
558   axisFontSize = 1.2;
559   lineWidth = 1.8;
560   XaxisFontSize = 42;
561   YaxisFontSize = 24;
562   masterFontSize = 18;
563
564   % Strings for axis,legends,Xlabel and Ylabel.
565   Xstring = {'$v_{luft} \quad \big[ \frac{m}{s} \big] $'};
566   Ystring = {'$\Delta P$[mbar]'};
567   legendTitle = {'Trycksensor'};
568
569   % Preallocation for graphical object.
570   h = gobjects(size(valuesY{1}, 2));
571   % resolution = 300; % -DPI use double of standard DPI for PNG-file saving.
572
573   % Normalise system to use Fullscreen figures, remember to change back since
574   % you are altering the default settings.
575   % set(groot, 'defaultFigureUnits','normalized')
576   % set(groot, 'defaultFigurePosition',[0 0 1 1])
577
```

```
% Get actual screensize.
screenSize = get(0, 'ScreenSize');

% Cut away some of the screen for better grpahs to save. Offset = 70 seems
% to work absolutly best.
offset = 70;
embeddedPosition = [screenSize(1), screenSize(2), ...
    screenSize(3), screenSize(4) - offset];

% screenSize(1),screenSize(2)


% Using for-loop to plot every set of pressure data and velocites
for i = 1:numTest - 1 % So we don't include empty bed.

    %-------------------------------------------------------------------------
    % Set correct maximazing fo figures in MATLAB

    %         fig = gcf; % only works for  MATLAB R2018a or alter.
    %         fig.WindowState = 'maximized'; % Maximize figure on screen.
    %     % Good for great resolution in plotting.
    % Speed up saving process using set(gcf,'Postiions',...etc...etc...).
    %     set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    set(gcf, 'WindowState', 'fullscreen')
    % Plot current data.
    for k = 2:size(valuesY{1}, 2)
        h(k) = plot(plotDataX{i}(1:length(valuesY{i})), valuesY{i}(:, k), '--o', ...
            'LineWidth', lineWidth, 'Color', colourString{k});
        hold on
    end
    hold off
    % Initalize axis, figure properties legend properties.
    ax = gca;
    ax.FontSize = masterFontSize;
    set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    fig = gcf;
    set(gcf, 'color', 'w');
    setappdata(gca, 'LegendColorbarManualSpace', 1);
    setappdata(gca, 'LegendColorbarReclaimSpace', 1);
    %     set(ax, 'nextplot', 'replacechildren'); % Forces new plot but keeps
    % current format on legends and axis. Also remove anyoing childrens ;)
    % feature('DefaultCharacterSet', 'UTF8')
    % Set X-axis
    xlabel(Xstring, 'Interpreter', 'latex', 'FontSize', ...
        XaxisFontSize);
    ax.XAxisLocation = 'bottom';
    set(get(gca, 'XLabel'), 'Rotation', 0); % Horizontal text.
    ax.XLimMode = 'auto';
    % Set Y-axis.
    ylabel(Ystring, 'Interpreter', 'latex', ...
        'FontSize', YaxisFontSize);
    ax.YAxisLocation = 'origin';
    ax.YLimMode = 'auto';
    %     set(get(gca, 'YLabel'), 'Rotation', 180); % Horizontal text.
    ylh = get(gca, 'ylabel');
    ylp = get(ylh, 'Position');
    set(ylh, 'Rotation', 90, 'Position', ylp, 'VerticalAlignment', ...
        'middle', 'HorizontalAlignment', 'center')
    ax.YAxis.MinorTick = 'on';

    % Set title.
    %     ax.Title.String = str{i};

    % Set axis properties.
    ax.Title.FontWeight = 'normal';
    ax.Title.FontAngle = 'italic';
    ax.LabelFontSizeMultiplier = axisFontSize;
    ax.TitleFontSizeMultiplier = axisFontSize;
    ax.FontSmoothing = 'on';
    ax.GridLineStyle = ':';
    ax.GridAlpha = 0.2;
    ax.YAxis.Limits = [min(min(valuesY{i}(:, 2:end))), ...
        max(max(valuesY{i}(:, 2:end)))];
```

```
651
652     ax.XAxis.Limits = [plotDataX{i}(1), ...
653         max(plotDataX{i}(1:length(valuesY{i}(:, 2:end))))];
654     ax.TickDirMode = 'auto';
655
656     % Set X and Y-tick properties, mainly changing their FontSize
657
658
659     % Set legends.
660     l = legend('show');
661     l.TextColor = 'black';
662     l.FontAngle = 'italic';
663     l.FontName = 'Comic Sans MS';
664     l.FontSize = legendFontsize;
665     l.Title.String = legendTitle;
666     l.Title.Color = 'black';
667     legend(strLegend, 'FontSize', legendFontsize, 'FontAngle', 'italic', ...
668         'Location', 'northeast', 'Interpreter', 'latex');
669     box on, grid on
670     set(gca, 'FontName', 'AvantGarde')
671
672     % Save figures as high-resolution PNG-files.
673     newStr = [str{i}, '.png']; % add .png to names.
674     saveFile = join(newStr);
675     %     drawnow;
676     % pause to let the figure fill the whole screen, screencapture is way
677     % to fast for this. Pause = 0.39 seems to work fine for now.
678     pause(0.39)
679
680
681     % NEVER EVER USE PRINTER AS AN OPTION!
682     screencapture(0, 'position', embeddedPosition, ...
683         'target', fullfile(fPath, saveFile))
684     % This is the fastest way to save figures, extremely fast.
685     %     if verLessThan('matlab', '9.8.0')
686     %         % Use only if older than R2020a
687     %         saveas(gcf, fullfile(fPath, saveFile), 'png')
688     %     else
689     %         % Only for matlab R2020a
690     %         exportgraphics(gcf, fullfile(fPath, saveFile), 'Resolution', resolution)
691     %     end
692     %     saveas(gcf, fullfile(fPath, saveFile),'epsc')
693     %     print(gcf,fullfile(fPath, saveFile),'-dpng','-r900');
694     %     F = getframe(gcf);
695     %     imwrite(F.cdata, saveFile);
696     close gcf
697 end
698
699 %-------------------------------------------------------------------------
700
701 %% Get Data For Sand III
702 % ================== Everything belowe this is Sand III ======================
703 % ============================================================================
704 clc
705 strMatchingSand3Y = { ...
706     'x20200417_S3_Hiflow_R0_5_', 'x20200417_S3_Hiflow_R0_25_', ...
707     'x20200417_S3_Unpacked_', 'x20200413_S3_Saddle0_5_R0_5_', ...
708     'x20200413_S3_Saddle_R0_25_', 'x20200421_deepbed_S3_Saddle_R1_', ...
709     'x20200421_S3_Unpacked_deepbed_', 'x20200428_deepbed_S3_Saddle_R05_'};
710
711
712 % Preallocation.
713 numTest = length(strMatchingSand3Y);
714 name = cell(1, numTest);
715 valuesYsand3 = cell(1, numTest);
716 valuesXsand3 = cell(1, numTest);
717 Pavg = cell(1, numTest);
718 Psensor = cell(1, numTest);
719 Pdistributor = cell(1, 1);
720
721
722 for i = 1:numTest
723
```

```
    % Extract correct series of experiment.
    idx = ¬cellfun('isempty', regexp(T.Properties.RowNames, ...
        strMatchingSand3Y{i}, 'once'));

    % Name is current meta-data row. Maybe get index for empty bed from
    % this?
    name = T(T.Properties.RowNames(idx), :);

    % Get numeric values from string to proper X-values.
    X = (regexp(name.Properties.RowNames, '\d+[\.]?\d*', 'match'));

    % Preallocation.
    x = zeros(1, length(X));

    % Access all valuees using for-loop.
    for k = 1:length(X)
        x(k) = str2double(cell2mat(X{k}(end)));
    end

    % Get numeric values from string to proper X and Y values.
    valuesXsand3{i} = x;
    valuesYsand3{i} = cell2mat(table2array(name(:, 1:end)));

    % Assign pressure values to calculate pressure drop over ditributor.
    Pavg{i} = valuesYsand3{i};
end

% To calculate pressure drop, first we need to calculate the pressure drop
% over distributor plate as below:

% Pressure drop over distributor (at a certain velocity)= average pressure
% of one of the 3 sensors (optional) above the distributor plate at empty
% conditions (empty bed with no packings and no sand, at a certain velocity)
% average pressure of windbox (empty bed with no packings and no sand,
% at the same velocity).

% Get pressure drop empty bed for 1 sensor, lets use P_2.
PemptySensor = valuesYsubstract{1}(:, 2);
PemptyWindbox = valuesYsubstract{1}(:, 1);
% Get Pressure drop over distributor.
Pdistributor{1} = PemptySensor - PemptyWindbox;

% Substract values of empty from values given by sand III.
for i = 1:numTest
    [C, idxRow, idxCol] = intersect(valuesXsubstract{1}, ...
        valuesXsand3{i}, 'stable');

    % Substract empty bed from regular bed.
    ii = length(idxRow);
    jj = length(valuesYsand3{i});
    numIter = min(ii, jj);

    for k = 1:numIter
        valuesYsand3{i}(k, :) = bsxfun(@minus, valuesYsand3{i}(k, :), ...
            valuesYsubstract{1}(idxRow(k), :));

        if i < numTest
            x = bsxfun(@minus, Pavg{i}(k, :), ...
                Pdistributor{1}(idxRow(k), :));

            % Get pressure drop over entire bed.
            Psensor{i}(k, :) = bsxfun(@minus, x, Pavg{i}(k, 1));
        else
        end

    end
end

% Preallocation.
r = 0.0605;
plotDataX = cell(1, numTest);

for i = 1:numTest
```

```
    % Convert volumtric flow from L/min to [m^3/s]
    x = valuesXsand3{i} / (1000 * 60);

    % Change plotting data to veolcity instead of volumtetric flow.
    y = x / (pi * r^2);
    plotDataX{i} = y;
end


%% Plot Calculated Pressure Drop Sand III.

clc
% Names for plotting. Add more for more experiments.
strPressureDrop = {'PD S3 HiFlow R = 0.5', 'PD S3 HiFlow R = 0.25', 'PD S3 Unpacked ...
    Bed', ...
    'PD S3 Saddle Rings R = 0.5', 'PD S3 Saddle Rings R = 0.25', ...
    'PD S3 DeepBed Saddle R = 1', 'PD S3 DeepBEd Unpacked', ...
    'PD S3 DeepBed Saddle R = 0.5'};


strLegend = {'$P_1$', '$P_2$', '$P_3$'};
colourString = {'green', 'black', 'red', 'blue'};
%

% Plotting and saving data. Come up with easy managable names for the
% graphs.

% Create Subfolder to save data in. Much neater than before.
if isfolder('C:\Users\Tinki\Desktop\BachelorThesis\Pressure Drop All Figures')
    fPath = 'C:\Users\Tinki\Desktop\BachelorThesis\Pressure Drop All Figures';
else
    cd('\\file00.chalmers.se\home\benmaar\.win\Desktop\BachelorThesisSchoolComputer')
    fPath = 'Z:\.win\Desktop\BachelorThesisSchoolComputer\Pressure Drop All Figures';
end


% Values to change fontsizes nad apperance of figures.
legendFontsize = 18;
axisFontSize = 1.2;
lineWidth = 1.8;
XaxisFontSize = 42;
YaxisFontSize = 24;
masterFontSize = 18;

% Strings for axis,legends,Xlabel and Ylabel.
Xstring = {'$v_{luft} \quad \big[ \frac{m}{s} \big] $'};
Ystring = {'$\Delta P$[mbar]'};
legendTitle = {'Trycksensor'};

% Preallocation for graphical object.
h = gobjects(size(Psensor{1}, 2));
% resolution = 300; % -DPI use double of standard DPI for PNG-file saving.

% Normalise system to use Fullscreen figures, remember to change back since
% you are altering the default settings.
% set(groot, 'defaultFigureUnits','normalized')
% set(groot, 'defaultFigurePosition',[0 0 1 1])

% Get actual screensize.
screenSize = get(0, 'ScreenSize');

% Cut away some of the screen for better grpahs to save. Offset = 70 seems
% to work absolutly best.
offset = 70;
embeddedPosition = [screenSize(1), screenSize(2), ...
    screenSize(3), screenSize(4) - offset];

% screenSize(1),screenSize(2)

for i = 1:numTest - 1 % So we don't include empty bed.

    %------------------------------------------------------------------------
    % Set correct maximazing fo figures in MATLAB
```

```
869
870     %         fig = gcf; % only works for  MATLAB R2018a or alter.
871     %         fig.WindowState = 'maximized'; % Maximize figure on screen.
872     %     % Good for great resolution in plotting.
873     % Speed up saving process using set(gcf,'Postiions',...etc...etc...).
874     %     set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
875     set(gcf, 'WindowState', 'fullscreen')
876     % Plot current data.
877     for k = 2:size(valuesY{1}, 2)
878         h(k) = plot(plotDataX{i}(1:length(Psensor{i})), Psensor{i}(:, k), '--o', ...
879             'LineWidth', lineWidth, 'Color', colourString{k});
880         hold on
881     end
882     hold off
883     % Initalize axis, figure properties legend properties.
884     ax = gca;
885     ax.FontSize = masterFontSize;
886     set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
887     fig = gcf;
888     set(gcf, 'color', 'w');
889     setappdata(gca, 'LegendColorbarManualSpace', 1);
890     setappdata(gca, 'LegendColorbarReclaimSpace', 1);
891     %     set(ax, 'nextplot', 'replacechildren'); % Forces new plot but keeps
892     % current format on legends and axis. Also remove anyoing childrens ;)
893     % feature('DefaultCharacterSet', 'UTF8')
894     % Set X-axis
895     xlabel(Xstring, 'Interpreter', 'latex', 'FontSize', ...
896         XaxisFontSize);
897     ax.XAxisLocation = 'bottom';
898     set(get(gca, 'XLabel'), 'Rotation', 0); % Horizontal text.
899     ax.XLimMode = 'auto';
900     % Set Y-axis.
901     ylabel(Ystring, 'Interpreter', 'latex', ...
902         'FontSize', YaxisFontSize);
903     ax.YAxisLocation = 'origin';
904     ax.YLimMode = 'auto';
905     %     set(get(gca, 'YLabel'), 'Rotation', 180); % Horizontal text.
906     ylh = get(gca, 'ylabel');
907     ylp = get(ylh, 'Position');
908     set(ylh, 'Rotation', 90, 'Position', ylp, 'VerticalAlignment', ...
909         'middle', 'HorizontalAlignment', 'center')
910     ax.YAxis.MinorTick = 'on';
911
912     % Set title.
913     %     ax.Title.String = str{i};
914
915     % Set axis properties.
916     ax.Title.FontWeight = 'normal';
917     ax.Title.FontAngle = 'italic';
918     ax.LabelFontSizeMultiplier = axisFontSize;
919     ax.TitleFontSizeMultiplier = axisFontSize;
920     ax.FontSmoothing = 'on';
921     ax.GridLineStyle = ':';
922     ax.GridAlpha = 0.2;
923     ax.YAxis.Limits = [min(min(Psensor{i}(:, 2:end))), ...
924         max(max(Psensor{i}(:, 2:end)))];
925
926     ax.XAxis.Limits = [plotDataX{i}(1), ...
927         max(plotDataX{i}(1:length(Psensor{i}(:, 2:end))))];
928     ax.TickDirMode = 'auto';
929
930     % Set X and Y-tick properties, mainly changing their FontSize
931
932
933     % Set legends.
934     l = legend('show');
935     l.TextColor = 'black';
936     l.FontAngle = 'italic';
937     l.FontName = 'Comic Sans MS';
938     l.FontSize = legendFontsize;
939     l.Title.String = legendTitle;
940     l.Title.Color = 'black';
941     legend(strLegend, 'FontSize', legendFontsize, 'FontAngle', 'italic', ...
```

```
        'Location', 'northeast', 'Interpreter', 'latex');
    box on, grid on
    set(gca, 'FontName', 'AvantGarde')

    % Save figures as high-resolution PNG-files.
    newStr = [strPressureDrop{i}, '.png']; % add .png to names.
    saveFile = join(newStr);
    %     drawnow;
    % pause to let the figure fill the whole screen, screencapture is way
    % to fast for this. Pause = 0.39 seems to work fine for now.
    pause(0.39)


    % NEVER EVER USE PRINTER AS AN OPTION!
    screencapture(0, 'position', embeddedPosition, ...
        'target', fullfile(fPath, saveFile))
    % This is the fastest way to save figures, extremely fast.
    %     if verLessThan('matlab', '9.8.0')
    %         % Use only if older than R2020a
    %         saveas(gcf, fullfile(fPath, saveFile), 'png')
    %     else
    %         % Only for matlab R2020a
    %         exportgraphics(gcf, fullfile(fPath, saveFile), 'Resolution', resolution)
    %     end
    %     saveas(gcf, fullfile(fPath, saveFile),'epsc')
    %     print(gcf,fullfile(fPath, saveFile),'-dpng','-r900');
    %     F = getframe(gcf);
    %     imwrite(F.cdata, saveFile);
    close gcf
end
```

```
%% Plot Data For Sand III
% Save String.
% str = {'Sand III HiFlow \n Pall rings, R = 0.5', ...
%     'Sand III HiFlow \n Pall rings, R = 0.25', 'Sand III Unpacked bed', ...
%     'Sand III Saddle rings \n, R = 0.5', 'Sand III Saddle rings \n, R = 0.25'};

strSave = {'S3 HiFlow R = 0.5', 'S3 HiFlow R = 0.25', 'S3 Unpacked Bed', ...
    'S3 Saddle Rings R = 0.5', 'S3 Saddle Rings R = 0.25', ...
    'S3 DeepBed Saddle R = 1', 'S3 DeepBEd Unpacked', ...
    'S3 DeepBed Saddle R = 0.5'};

fPath = 'C:\Users\Tinki\Desktop\BachelorThesis\Sand3Graphs';
% h = gobjects(size(valuesXsand3{1}, 2));

% Values to change fontsizes nad apperance of figures.
legendFontsize = 18;
axisFontSize = 1.2;
lineWidth = 1.8;
XaxisFontSize = 42;
YaxisFontSize = 24;
masterFontSize = 18;

% Strings for axis,legends,Xlabel and Ylabel.
Xstring = {'$v_{luft} \quad \big[ \frac{m}{s} \big] $'};
Ystring = {'$\Delta P$[mbar]'};
legendTitle = {'Trycksensor'};
```

```
% Preallocation for graphical object.
h = gobjects(size(valuesYsand3{1}, 2));
% resolution = 300; % -DPI use double of standard DPI for PNG-file saving.

% Normalise system to use Fullscreen figures, remember to change back since
% you are altering the default settings.
% set(groot, 'defaultFigureUnits','normalized')
% set(groot, 'defaultFigurePosition',[0 0 1 1])

% Get acutal screensize.
screenSize = get(0, 'ScreenSize');

% Cut away some of the screen for better grpahs to save. Offset = 70 seems
% to work absolutly best.
offset = 70;
embeddedPosition = [screenSize(1), screenSize(2), ...
    screenSize(3), screenSize(4) - offset];

% screenSize(1),screenSize(2)


% Using for-loop to plot every set of pressure data and velocites
for i = 1:numTest

    %-------------------------------------------------------------------------
    % Set correct maximazing fo figures in MATLAB

    %         fig = gcf; % only works for  MATLAB R2018a or alter.
    %         fig.WindowState = 'maximized'; % Maximize figure on screen.
    %     % Good for great resolution in plotting.
    % Speed up saving process using set(gcf,'Postiions',...etc...etc...).
    %     set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    set(gcf, 'WindowState', 'fullscreen')
    % Plot current data.
    for k = 2:size(valuesY{1}, 2)
        h(k) = plot(plotDataX{i}(1:length(valuesYsand3{i})), ...
            valuesYsand3{i}(:, k), '--o', ...
            'LineWidth', lineWidth, 'Color', colourString{k});
        hold on
    end
    % Initalize axis, figure properties legend properties.
    ax = gca;
    ax.FontSize = masterFontSize;
    set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
    fig = gcf;
    set(gcf, 'color', 'w');
    setappdata(gca, 'LegendColorbarManualSpace', 1);
    setappdata(gca, 'LegendColorbarReclaimSpace', 1);
    %     set(ax, 'nextplot', 'replacechildren'); % Forces new plot but keeps
    % current format on legends and axis. Also remove anyoing childrens ;)
    % feature('DefaultCharacterSet', 'UTF8')
    % Set X-axis
    xlabel(Xstring, 'Interpreter', 'latex', 'FontSize', ...
        XaxisFontSize);
    ax.XAxisLocation = 'bottom';
    set(get(gca, 'XLabel'), 'Rotation', 0); % Horizontal text.
    ax.XLimMode = 'auto';
    % Set Y-axis.
    ylabel(Ystring, 'Interpreter', 'latex', ...
        'FontSize', YaxisFontSize);
    ax.YAxisLocation = 'origin';
    ax.YLimMode = 'auto';
    %     set(get(gca, 'YLabel'), 'Rotation', 180); % Horizontal text.
    ylh = get(gca, 'ylabel');
    ylp = get(ylh, 'Position');
    set(ylh, 'Rotation', 90, 'Position', ylp, 'VerticalAlignment', ...
        'middle', 'HorizontalAlignment', 'center')
    ax.YAxis.MinorTick = 'on';

    % Set title.
    %     ax.Title.String = str{i};

    % Set axis properties.
```

```
    ax.Title.FontWeight = 'normal';
    ax.Title.FontAngle = 'italic';
    ax.LabelFontSizeMultiplier = axisFontSize;
    ax.TitleFontSizeMultiplier = axisFontSize;
    ax.FontSmoothing = 'on';
    ax.GridLineStyle = ':';
    ax.GridAlpha = 0.2;

    % Set axis limits, could be auto but more control with manual.
    ax.YAxis.Limits = [min(min(valuesYsand3{i}(:, 2:end))), ...
        max(max(valuesYsand3{i}(:, 2:end)))];

    ax.XAxis.Limits = [plotDataX{i}(1), ...
        max(plotDataX{i}(1:length(valuesYsand3{i}(:, 2:end))))];
    ax.TickDirMode = 'auto';

    % Set X and Y-tick properties, mainly changing their FontSize


    % Set legends.
    l = legend('show');
    l.TextColor = 'black';
    l.FontAngle = 'italic';
    l.FontName = 'Comic Sans MS';
    l.FontSize = legendFontsize;
    l.Title.String = legendTitle;
    l.Title.Color = 'black';
    legend(strLegend, 'FontSize', legendFontsize, 'FontAngle', 'italic', ...
        'Location', 'northeast', 'Interpreter', 'latex');
    box on, grid on
    set(gca, 'FontName', 'AvantGarde')

    % Save figures as high-resolution PNG-files.
    newStr = [strSave{i}, '.png']; % add .png to names.
    saveFile = join(newStr);
    %     drawnow;
    % pause to let the figure fill the whole screen, screencapture is way
    % to fast for this. Pause = 0.39 seems to work fine for now.
    pause(0.39)


    % NEVER EVER USE PRINTER AS AN OPTION!
    screencapture(0, 'position', embeddedPosition, ...
        'target', fullfile(fPath, saveFile))
    % This is the fastest way to save figures, extremely fast.
    %     if verLessThan('matlab', '9.8.0')
    %         % Use only if older than R2020a
    %         saveas(gcf, fullfile(fPath, saveFile), 'png')
    %     else
    %         % Only for matlab R2020a
    %         exportgraphics(gcf, fullfile(fPath, saveFile), 'Resolution', resolution)
    %     end
    %     saveas(gcf, fullfile(fPath, saveFile),'epsc')
    %     print(gcf,fullfile(fPath, saveFile),'-dpng','-r900');
    %     F = getframe(gcf);
    %     imwrite(F.cdata, saveFile);
    close gcf
end

%% Plot Pressure During Sample Time
% ------------------------------------------------------------------------------
% ========================== Plot Sampling Data ================================
% ------------------------------------------------------------------------------
% SAMPLING DATA SAMPLING DATA SAMPLING DATA SAMPLING DATA SAMPLING DATA !!!!!!!!
% SAMPLING DATA SAMPLING DATA SAMPLING DATA SAMPLING DATA SAMPLING DATA !!!!!!!!
clc
% Plot over the sample interval, Time V.S Pressure.
strLegend = {'$P_1$', '$P_2$', '$P_3$'};
% h = gobjects(size(rawName, 2));
colourString = {'black', 'red', 'blue', 'green'};
saveName = cell(1, length(rawName));
titleString = cell(1, length(rawName));

```

```
1161
1162 % Declare path to save file, use fullfile!!
1163 fPath = 'C:\Users\Tinki\Desktop\BachelorThesis\SamplingDataGraphs';
1164
1165 % Manipulate save string.
1166 for i = 1:length(rawName)
1167     x = rawName{i}(1:end - 4);
1168     saveName{i} = [x, '_graph'];
1169     titleString{i} = strrep(x, '_', ' ');
1170 end
1171
1172 % Values to change fontsizes nad apperance of figures.
1173 legendFontsize = 18;
1174 axisFontSize = 1.2;
1175 lineWidth = 1.8;
1176 XaxisFontSize = 42;
1177 YaxisFontSize = 24;
1178 masterFontSize = 18;
1179
1180 % Strings for axis,legends,Xlabel and Ylabel.
1181 Xstring = {'tid [s]'};
1182 Ystring = {'$\Delta P$[mbar]'};
1183 legendTitle = {'Trycksensor'};
1184
1185 % Preallocation for graphical object.
1186 h = gobjects(size(rawName{1}, 2));
1187 resolution = 300; % -DPI use double of standard DPI for PNG-file saving.
1188
1189 % Normalise system to use Fullscreen figures, remember to change back since
1190 % you are altering the default settings.
1191 % set(groot, 'defaultFigureUnits','normalized')
1192 % set(groot, 'defaultFigurePosition',[0 0 1 1])
1193
1194 % Get acutal screensize.
1195 screenSize = get(0, 'ScreenSize');
1196
1197 % Cut away some of the screen for better grpahs to save. Offset = 70 seems
1198 % to work absolutly best.
1199 offset = 70;
1200 embeddedPosition = [screenSize(1), screenSize(2), ...
1201     screenSize(3), screenSize(4) - offset];
1202
1203 % Plot the data using useful tricks to get it to the right subfolders and
1204 % name.
1205
1206
1207 idx = [14, 16, 2, 15];
1208
1209 for i = 1:length(rawName)
1210     %--------------------------------------------------------------------------
1211     % Set correct maximazing fo figures in MATLAB
1212     set(gcf, 'WindowState', 'fullscreen')
1213
1214     if ~isempty(rawData{i})
1215
1216         % Plot current data.
1217         for k = 2:size(idx, 2)
1218
1219             h(k) = plot(rawData{i}(:, 1), rawData{i}(:, idx(k)), '--', ...
1220                 'LineWidth', 0.7, 'Color', colourString{k-1});
1221             hold on
1222
1223         end
1224
1225         % Initalize axis, figure properties legend properties.
1226         ax = gca;
1227         ax.FontSize = masterFontSize;
1228         set(gcf, 'unit', 'norm', 'position', [0, 0, 1, 1])
1229         fig = gcf;
1230         set(gcf, 'color', 'w');
1231         setappdata(gca, 'LegendColorbarManualSpace', 1);
1232         setappdata(gca, 'LegendColorbarReclaimSpace', 1);
1233         %     set(ax, 'nextplot', 'replacechildren'); % Forces new plot but keeps
```

```
1234        % current format on legends and axis. Also remove anyoing childrens ;)
1235        % feature('DefaultCharacterSet', 'UTF8')
1236        % Set X-axis
1237        xlabel(Xstring, 'Interpreter', 'latex', 'FontSize', ...
1238            XaxisFontSize);
1239        ax.XAxisLocation = 'bottom';
1240        set(get(gca, 'XLabel'), 'Rotation', 0); % Horizontal text.
1241        ax.XLimMode = 'auto';
1242        % Set Y-axis.
1243        ylabel(Ystring, 'Interpreter', 'latex', ...
1244            'FontSize', YaxisFontSize);
1245        ax.YAxisLocation = 'origin';
1246        ax.YLimMode = 'auto';
1247        %     set(get(gca, 'YLabel'), 'Rotation', 180); % Horizontal text.
1248        ylh = get(gca, 'ylabel');
1249        ylp = get(ylh, 'Position');
1250        set(ylh, 'Rotation', 90, 'Position', ylp, 'VerticalAlignment', ...
1251            'middle', 'HorizontalAlignment', 'center')
1252        ax.YAxis.MinorTick = 'on';
1253
1254        % Set title.
1255        ax.Title.String = titleString{i};
1256
1257        % Set axis properties.
1258        ax.Title.FontWeight = 'normal';
1259        ax.Title.FontAngle = 'italic';
1260        ax.LabelFontSizeMultiplier = axisFontSize;
1261        ax.TitleFontSizeMultiplier = axisFontSize;
1262        ax.FontSmoothing = 'on';
1263        ax.GridLineStyle = ':';
1264        ax.GridAlpha = 0.2;
1265
1266        % Set axis limits, could be auto but more control with manual.
1267        ax.YAxis.Limits = [min(min(rawData{i}(:, idx(2:end)))), ...
1268            max(max(rawData{i}(:, idx(2:end))))];
1269
1270        ax.XAxis.Limits = [0, max(rawData{i}(:, 1))];
1271        ax.TickDirMode = 'auto';
1272        % Set legends.
1273        l = legend('show');
1274        l.TextColor = 'black';
1275        l.FontAngle = 'italic';
1276        l.FontName = 'Comic Sans MS';
1277        l.FontSize = legendFontsize;
1278        l.Title.String = legendTitle;
1279        l.Title.Color = 'black';
1280        legend(strLegend, 'FontSize', legendFontsize, 'FontAngle', 'italic', ...
1281            'Location', 'northeast', 'Interpreter', 'latex');
1282        box on, grid on
1283        set(gca, 'FontName', 'AvantGarde')
1284
1285        % Save figures as high-resolution PNG-files.
1286        newStr = [saveName{i}, '.png']; % add .png to names.
1287        saveFile = join(newStr);
1288        %     drawnow;
1289        % pause to let the figure fill the whole screen, screencapture is way
1290        % to fast for this. Pause = 0.39 seems to work fine for now.
1291        pause(0.39)
1292
1293
1294        % NEVER EVER USE PRINTER AS AN OPTION!
1295        screencapture(0, 'position', embeddedPosition, ...
1296            'target', fullfile(fPath, saveFile))
1297        % This is the fastest way to save figures, extremely fast.
1298        %     if verLessThan('matlab', '9.8.0')
1299        %         % Use only if older than R2020a
1300        %         saveas(gcf, fullfile(fPath, saveFile), 'png')
1301        %     else
1302        %         % Only for matlab R2020a
1303        %         exportgraphics(gcf, fullfile(fPath, saveFile), 'Resolution', resolution)
1304        %     end
1305        %     saveas(gcf, fullfile(fPath, saveFile),'epsc')
1306        %     print(gcf,fullfile(fPath, saveFile),'-dpng','-r900');
```

```
1307          %     F = getframe(gcf);
1308          %     imwrite(F.cdata, saveFile);
1309          close gcf
1310      else
1311          fprintf(1, '  \n');
1312          fprintf(1, '  Current Datafile is Empty, number: %3.0f.\n', i);
1313      end
1314
1315  end
1316
1317  %% Windbox—empty bed—atmospheric pressure.
1318
1319  %% Get Minimal Fluidization Velocity For Sand 1.
1320  % ————————————————————————————————————————————————————————————————————
1321  % ========================== More Data Analysis ==========================
1322  % ————————————————————————————————————————————————————————————————————
1323
1324  % Find minimal fluidization velocity u_mf.
1325
1326  % We can make more with this loop.
1327  % Try looping over each P_1—P_3 and try to get minimal fluidization
1328  % velocity on each level of the bed.
1329
1330  % n = 15;
1331  id = 'MATLAB:polyfit:RepeatedPointsOrRescale';
1332  warning('off', id);
1333
1334  r = 0.0605;
1335
1336  for i = 1:numTest — 1
1337      x = valuesX{i}(1:length(valuesY{i}));
1338      y = valuesY{i}(:, 2)';
1339
1340      idx = islocalmax(y);
1341      localMaxX = x(idx);
1342      % Using modelfun
1343      % fcn = @(b,x) b(1).*exp(b(2).*x) + b(3).*log(b(4).*x);
1344      % [B,fval] = fminsearch(@(b) norm(y — fcn(b,x)), ones(4,1));
1345
1346
1347      % plot(x, y, 'p')
1348      % hold on
1349      % plot(x, fcn(B,x), '—')0,2 136
1350      % hold off
1351      % grid
1352
1353      % Using polyfit. Works for now.
1354
1355      %     X = 0:0.001:700;
1356      %     p = polyfit(x, y, n);
1357      %     fun = @(x)polyval(p, X);
1358      %     A = fun(X);
1359      %     TF = islocalmax(A);
1360      %     plot(x,A,x(TF),A(TF),'r*');
1361      %     hold on
1362      %     plot(x,y,'——')
1363      %     hold off
1364      %
1365      %     localMaxX = X(TF);
1366
1367      % convert to minimal fluidization velocity and not volumetric flow.
1368      B = localMaxX ./ 1000 ./ 60;
1369      localMaxX = B ./ (pi * r.^2);
1370      % localMaxY = y(TF);
1371
1372      if isempty(localMaxX)
1373          fprintf(1, '  \n');
1374          fprintf(1, '  No local Maxima was found for case\n');
1375          fprintf(1, '  %s \n', str{i});
1376      else
1377
1378          fprintf(1, '  \n');
1379          fprintf(1, '  Minimal fluidization velocity for case\n');
```

```
1380            fprintf(1, '  %s is:%3.9f.\n', str{i}, localMaxX(1));
1381        end
1382    end
1383    % ================================================================================
```

E.2 importfile.m

```
1   function S1SaddleR = importfile(filename, startRow, endRow)
2   %IMPORTFILE Import numeric data from a text file as a matrix.
3   %   S1SADDLER = IMPORTFILE(FILENAME) Reads data from text file FILENAME for
4   %   the default selection.
5   %
6   %   S1SADDLER = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
7   %   STARTROW through ENDROW of text file FILENAME.
8   %
9   % Example:
10  %   S1SaddleR = importfile('20200218_S1_Saddle_R.5_V54.txt', 25, 75);
11  %
12  %    See also TEXTSCAN.
13
14  % Auto-generated by MATLAB on 2020/02/19 21:07:57
15
16  %% Initialize variables.
17  delimiter = '\t';
18  if nargin<=2
19      startRow = 25;
20      endRow = 75;
21  end
22
23  %% Format for each line of text:
24  %   column1: double (%f)
25  %   column2: double (%f)
26  %   column3: double (%f)
27  %   column4: double (%f)
28  %   column5: double (%f)
29  %   column6: double (%f)
30  %   column7: double (%f)
31  %   column8: double (%f)
32  %   column9: double (%f)
33  %   column10: double (%f)
34  %   column11: double (%f)
35  %   column12: double (%f)
36  %   column13: double (%f)
37  %   column14: double (%f)
38  %   column15: double (%f)
39  %   column16: double (%f)
40  %   column17: double (%f)
41  %   column18: double (%f)
42  %   column19: double (%f)
43  %   column20: double (%f)
44  %   column21: double (%f)
45  %   column22: double (%f)
46  %   column23: double (%f)
47  %   column24: double (%f)
48  %   column25: double (%f)
49  %   column26: double (%f)
50  %   column27: double (%f)
51  %   column28: double (%f)
52  %   column29: double (%f)
53  %   column30: double (%f)
54  %   column31: double (%f)
55  %   column32: double (%f)
56  %   column33: double (%f)
57  %   column34: double (%f)
58  %   column35: double (%f)
59  %   column36: double (%f)
60  %   column37: double (%f)
61  %   column38: double (%f)
62  %   column39: double (%f)
```

```
%   column40: double (%f)
%   column41: double (%f)
%   column42: double (%f)
%   column43: double (%f)
%   column44: double (%f)
%   column45: double (%f)
%   column46: double (%f)
%   column47: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = ...
    '%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f...
%f%f%f%f%f%f%f%*s%*s%*s%*s%*s%*s%[^\n\r]';

%% Open the text file.
[fileID, msg] = fopen(filename,'r');
if fileID < 0
  error('Failed to open file "%s" because: "%s"', filename, msg);
end

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'TextType', 'string', 'HeaderLines', startRow(1)-1, 'ReturnOnError', ...
    false, 'EndOfLine', '\r\n');
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'TextType', 'string', 'HeaderLines', startRow(block)-1, ...
        'ReturnOnError', false, 'EndOfLine', '\r\n');
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Create output variable
S1SaddleR = [dataArray{1:end-1}];
```

E.3 natsort.m

Följande program är gjort av Stephen Cobeldick [24]

```
function [X,ndx,dbg] = natsort(X,rgx,varargin)
% Alphanumeric / Natural-Order sort the strings in a cell array of strings (1xN char).
%
% (c) 2012-2019 Stephen Cobeldick
%
% Alphanumeric sort a cell array of strings: sorts by character order and
% also by the values of any number substrings. Default: match all integer
% number substrings and perform a case-insensitive ascending sort.
%
%%% Example:
% >> X = {'x2', 'x10', 'x1'};
% >> sort(X)
% ans =   'x1'  'x10'  'x2'
% >> natsort(X)
% ans =   'x1'  'x2'  'x10'
%
%%% Syntax:
```

```
%  Y = natsort(X)
%  Y = natsort(X,rgx)
%  Y = natsort(X,rgx,<options>)
% [Y,ndx,dbg] = natsort(X,...)
%
% To sort filenames or filepaths use NATSORTFILES (FEX 47434).
% To sort the rows of a cell array of strings use NATSORTROWS (FEX 47433).
%
%% Number Substrings %%
%
% By default consecutive digit characters are interpreted as an integer.
% Specifying the optional regular expression pattern allows the numbers to
% include a +/- sign, decimal digits, exponent E-notation, quantifiers,
% or look-around matching. For information on defining regular expressions:
% http://www.mathworks.com/help/matlab/matlab_prog/regular-expressions.html
%
% The number substrings are parsed by SSCANF into numeric values, using
% either the *default format '%f' or the user-supplied format specifier.
%
% This table shows examples of regular expression patterns for some common
% notations and ways of writing numbers, with suitable SSCANF formats:
%
% Regular       | Number Substring | Number Substring               | SSCANF
% Expression:   | Match Examples:  | Match Description:             | Format Specifier:
% ==============|==================|================================|==================
% *         \d+ | 0, 123, 4, 56789 | unsigned integer               | %f  %i  %u  %lu
% --------------|------------------|--------------------------------|------------------
%      [-+]?\d+ | +1, 23, -45, 678 | integer with optional +/- sign | %f  %i  %d  %ld
% --------------|------------------|--------------------------------|------------------
%     \d+\.?\d* | 012, 3.45, 678.9 | integer or decimal             | %f
% (\d+|Inf|NaN) | 123, 4, NaN, Inf | integer, Inf, or NaN           | %f
%  \d+\.\d+e\d+ | 0.123e4, 5.67e08 | exponential notation           | %f
% --------------|------------------|--------------------------------|------------------
%  0[0-7]+      | 012, 03456, 0700 | octal notation & prefix        | %o  %i
%   [0-7]+      |  12,  3456,  700 | octal notation                 | %o
% --------------|------------------|--------------------------------|------------------
%  0X[0-9A-F]+  | 0X0, 0X3E7, 0XFF | hexadecimal notation & prefix  | %x  %i
%    [0-9A-F]+  |   0,   3E7,   FF | hexadecimal notation           | %x
% --------------|------------------|--------------------------------|------------------
%  0B[01]+      | 0B1, 0B101, 0B10 | binary notation & prefix       | %b   (not SSCANF)
%    [01]+      |   1,   101,   10 | binary notation                | %b   (not SSCANF)
% --------------|------------------|--------------------------------|------------------
%
%% Debugging Output Array %%
%
% The third output is a cell array <dbg>, to check if the numbers have
% been matched by the regular expression <rgx> and converted to numeric
% by the SSCANF format. The rows of <dbg> are linearly indexed from <X>,
% even columns contain numbers, odd columns contain split substrings:
%
% >> [~,~,dbg] = natsort(X)
% dbg =
%    'x'    [ 2]
%    'x'    [10]
%    'x'    [ 1]
%
%% Examples %%
%
%%% Multiple integers (e.g. release version numbers):
% >> A = {'v10.6', 'v9.10', 'v9.5', 'v10.10', 'v9.10.20', 'v9.10.8'};
% >> sort(A)
% ans =   'v10.10'  'v10.6'  'v9.10'  'v9.10.20'  'v9.10.8'  'v9.5'
% >> natsort(A)
% ans =   'v9.5'  'v9.10'  'v9.10.8'  'v9.10.20'  'v10.6'  'v10.10'
%
%%% Integer, decimal, NaN, or Inf numbers, possibly with +/- signs:
% >> B = {'test+NaN', 'test11.5', 'test-1.4', 'test', 'test-Inf', 'test+0.3'};
% >> sort(B)
% ans =   'test' 'test+0.3' 'test+NaN' 'test-1.4' 'test-Inf' 'test11.5'
% >> natsort(B, '[-+]?(NaN|Inf|\d+\.?\d*)')
% ans =   'test' 'test-Inf' 'test-1.4' 'test+0.3' 'test11.5' 'test+NaN'
%
%%% Integer or decimal numbers, possibly with an exponent:
```

```
% >> C = {'0.56e007', '', '43E-2', '10000', '9.8'};
% >> sort(C)
% ans =   ''  '0.56e007'  '10000'  '43E-2'  '9.8'
% >> natsort(C, '\d+\.?\d*([eE][-+]?\d+)?')
% ans =   ''  '43E-2'  '9.8'  '10000'  '0.56e007'
%
%%% Hexadecimal numbers (with '0X' prefix):
% >> D = {'a0X7C4z', 'a0X5z', 'a0X18z', 'a0XFz'};
% >> sort(D)
% ans =   'a0X18z'  'a0X5z'  'a0X7C4z'  'a0XFz'
% >> natsort(D, '0X[0-9A-F]+', '%i')
% ans =   'a0X5z'  'a0XFz'  'a0X18z'  'a0X7C4z'
%
%%% Binary numbers:
% >> E = {'a11111000100z', 'a101z', 'a000000000011000z', 'a1111z'};
% >> sort(E)
% ans =   'a000000000011000z'  'a101z'  'a11111000100z'  'a1111z'
% >> natsort(E, '[01]+', '%b')
% ans =   'a101z'  'a1111z'  'a000000000011000z'  'a11111000100z'
%
%%% Case sensitivity:
% >> F = {'a2', 'A20', 'A1', 'a10', 'A2', 'a1'};
% >> natsort(F, [], 'ignorecase') % default
% ans =   'A1'  'a1'  'a2'  'A2'  'a10'  'A20'
% >> natsort(F, [], 'matchcase')
% ans =   'A1'  'A2'  'A20'  'a1'  'a2'  'a10'
%
%%% Sort order:
% >> G = {'2', 'a', '', '3', 'B', '1'};
% >> natsort(G, [], 'ascend') % default
% ans =   ''   '1'  '2'  '3'  'a'  'B'
% >> natsort(G, [], 'descend')
% ans =   'B'  'a'  '3'  '2'  '1'  ''
% >> natsort(G, [], 'num<char') % default
% ans =   ''   '1'  '2'  '3'  'a'  'B'
% >> natsort(G, [], 'char<num')
% ans =   ''   'a'  'B'  '1'  '2'  '3'
%
%%% UINT64 numbers (with full precision):
% >> natsort({'a18446744073709551615z', 'a18446744073709551614z'}, [], '%lu')
% ans =       'a18446744073709551614z'  'a18446744073709551615z'
%
%% Input and Output Arguments %%
%
%%% Inputs (*==default):
% X   = CellArrayOfCharRowVectors, to be sorted into natural-order.
% rgx = Regular expression to match number substrings, '\d+'*
%     = [] uses the default regular expression, which matches integers.
% <options> can be entered in any order, as many as required:
%     = Sort direction: 'descend'/'ascend'*
%     = NaN/number order: 'NaN<num'/'num<NaN'*
%     = Character/number order: 'char<num'/'num<char'*
%     = Character case handling: 'matchcase'/'ignorecase'*
%     = SSCANF number conversion format, e.g.: '%f'*, '%x', '%li', '%b', etc.
%
%%% Outputs:
% Y   = CellArrayOfCharRowVectors, <X> sorted into natural-order.
% ndx = NumericArray, such that Y = X(ndx). The same size as <X>.
% dbg = CellArray of the parsed characters and number values.
%       Each row is one input char vector, linear-indexed from <X>.
%
% See also SORT NATSORTFILES NATSORTROWS CELLSTR REGEXP IREGEXP SSCANF

%% Input Wrangling %%
%
assert(iscell(X),'First input <X> must be a cell array.')
tmp = cellfun('isclass',X,'char') & cellfun('size',X,1)<2 & cellfun('ndims',X)<3;
assert(all(tmp(:)),'First input <X> must be a cell array of char row vectors (1xN char).')
%
if nargin<2 || isnumeric(rgx)&&isempty(rgx)
    rgx = '\d+';
else
    assert(ischar(rgx)&&ndims(rgx)<3&&size(rgx,1)==1,...
```

```
        'Second input <rgx> must be a regular expression (char row vector).') %#ok<ISMAT>
end
%
% Optional arguments:
tmp = cellfun('isclass',varargin,'char') & cellfun('size',varargin,1)<2 & ...
    cellfun('ndims',varargin)<3;
assert(all(tmp(:)),'All optional arguments must be char row vectors (1xN char).')
% Character case:
ccm = strcmpi(varargin,'matchcase');
ccx = strcmpi(varargin,'ignorecase')|ccm;
% Sort direction:
sdd = strcmpi(varargin,'descend');
sdx = strcmpi(varargin,'ascend')|sdd;
% Char/num order:
chb = strcmpi(varargin,'char<num');
chx = strcmpi(varargin,'num<char')|chb;
% NaN/num order:
nab = strcmpi(varargin,'NaN<num');
nax = strcmpi(varargin,'num<NaN')|nab;
% SSCANF format:
sfx = ¬cellfun('isempty',regexp(varargin,'^%([bdiuoxfeg]|l[diuox])$'));
%
nsAssert(1,varargin,sdx,'Sort direction')
nsAssert(1,varargin,chx,'Char<->num')
nsAssert(1,varargin,nax,'NaN<->num')
nsAssert(1,varargin,sfx,'SSCANF format')
nsAssert(0,varargin,¬(ccx|sdx|chx|nax|sfx))
%
% SSCANF format:
if nnz(sfx)
    fmt = varargin{sfx};
    if strcmpi(fmt,'%b')
        cls = 'double';
    else
        cls = class(sscanf('0',fmt));
    end
else
    fmt = '%f';
    cls = 'double';
end
%
%% Identify Numbers %%
%
[mat,spl] = regexpi(X(:),rgx,'match','split',varargin{ccx});
%
% Determine lengths:
nmx = numel(X);
nmn = cellfun('length',mat);
nms = cellfun('length',spl);
mxs = max(nms);
%
% Preallocate arrays:
bon = bsxfun(@le,1:mxs,nmn).';
bos = bsxfun(@le,1:mxs,nms).';
arn = zeros(mxs,nmx,cls);
ars =  cell(mxs,nmx);
ars(:) = {''};
ars(bos) = [spl{:}];
%
%% Convert Numbers to Numeric %%
%
if nmx
    tmp = [mat{:}];
    if strcmp(fmt,'%b')
        tmp = regexprep(tmp,'^0[Bb]','');
        vec = cellfun(@(s)sum(pow2(s-'0',numel(s)-1:-1:0)),tmp);
    else
        vec = sscanf(sprintf(' %s',tmp{:}),fmt);
    end
    assert(numel(vec)==numel(tmp),'The %s format must return one value for each input ...
        number.',fmt)
else
    vec = [];
```

```
end
%
%% Debugging Array %%
%
if nmx && nargout>2
    dbg = cell(mxs,nmx);
    dbg(:) = {''};
    dbg(bon) = num2cell(vec);
    dbg = reshape(permute(cat(3,ars,dbg),[3,1,2]),[],nmx).';
    idf = [find(¬all(cellfun('isempty',dbg),1),1,'last'),1];
    dbg = dbg(:,1:idf(1));
else
    dbg = {};
end
%
%% Sort Columns %%
%
if ¬any(ccm) % ignorecase
    ars = lower(ars);
end
%
if nmx && any(chb) % char<num
    boe = ¬cellfun('isempty',ars(bon));
    for k = reshape(find(bon),1,[])
        ars{k}(end+1) = char(65535);
    end
    [idr,idc] = find(bon);
    idn = sub2ind(size(bon),boe(:)+idr(:),idc(:));
    bon(:) = false;
    bon(idn) = true;
    arn(idn) = vec;
    bon(isnan(arn)) = ¬any(nab);
    ndx = 1:nmx;
    if any(sdd) % descending
        for k = mxs:-1:1
            [¬,idx] = sort(nsGroup(ars(k,ndx)),'descend');
            ndx = ndx(idx);
            [¬,idx] = sort(arn(k,ndx),'descend');
            ndx = ndx(idx);
            [¬,idx] = sort(bon(k,ndx),'descend');
            ndx = ndx(idx);
        end
    else % ascending
        for k = mxs:-1:1
            [¬,idx] = sort(ars(k,ndx));
            ndx = ndx(idx);
            [¬,idx] = sort(arn(k,ndx),'ascend');
            ndx = ndx(idx);
            [¬,idx] = sort(bon(k,ndx),'ascend');
            ndx = ndx(idx);
        end
    end
else % num<char
    arn(bon) = vec;
    bon(isnan(arn)) = ¬any(nab);
    if any(sdd) % descending
        [¬,ndx] = sort(nsGroup(ars(mxs,:)),'descend');
        for k = mxs-1:-1:1
            [¬,idx] = sort(arn(k,ndx),'descend');
            ndx = ndx(idx);
            [¬,idx] = sort(bon(k,ndx),'descend');
            ndx = ndx(idx);
            [¬,idx] = sort(nsGroup(ars(k,ndx)),'descend');
            ndx = ndx(idx);
        end
    else % ascending
        [¬,ndx] = sort(ars(mxs,:));
        for k = mxs-1:-1:1
            [¬,idx] = sort(arn(k,ndx),'ascend');
            ndx = ndx(idx);
            [¬,idx] = sort(bon(k,ndx),'ascend');
            ndx = ndx(idx);
            [¬,idx] = sort(ars(k,ndx));
```

```
            ndx = ndx(idx);
        end
    end
end
%
ndx  = reshape(ndx,size(X));
X = X(ndx);
%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%natsort
function nsAssert(val,inp,idx,varargin)
% Throw an error if an option is overspecified.
if nnz(idx)>val
    tmp = {'Unknown input arguments',' option may only be specified once. Provided ...
        inputs'};
    error('%s:%s',[varargin{:},tmp{1+val}],sprintf('\n''%s''',inp{idx}))
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%nsAssert
function grp = nsGroup(vec)
% Groups of a cell array of strings, equivalent to [¬,¬,grp]=unique(vec);
[vec,idx] = sort(vec);
grp = cumsum([true,¬strcmp(vec(1:end-1),vec(2:end))]);
grp(idx) = grp;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%nsGroup
```

E.4 natsortfiles.m

```
function [X,ndx,dbg] = natsortfiles(X,rgx,varargin)
% Alphanumeric / Natural-Order sort of a cell array of filename/filepath strings (1xN ...
    char).
%
% (c) 2014-2019 Stephen Cobeldick
%
% Alphanumeric sort of a cell array of filenames or filepaths: sorts by
% character order and also by the values of any numbers that are within
% the names. Filenames, file-extensions, and directories (if supplied)
% are split apart and are sorted separately: this ensures that shorter
% filenames sort before longer ones (i.e. thus giving a dictionary sort).
%
%%% Example:
% P = 'C:\SomeDir\SubDir';
% S = dir(fullfile(P,'*.txt'));
% C = natsortfiles({S.name});
% for k = 1:numel(C)
%     fullfile(P,C{k})
% end
%
%%% Syntax:
%  Y = natsortfiles(X)
%  Y = natsortfiles(X,rgx)
%  Y = natsortfiles(X,rgx,<options>)
% [Y,ndx,dbg] = natsortfiles(X,...)
%
% To sort all of the strings in a cell array use NATSORT (File Exchange 34464).
% To sort the rows of a cell array of strings use NATSORTROWS (File Exchange 47433).
%
%% File Dependency %%
%
% NATSORTFILES requires the function NATSORT (File Exchange 34464). The optional
% arguments <options> are passed directly to NATSORT. See NATSORT for case
% sensitivity, sort direction, numeric substring matching, and other options.
%
%% Explanation %%
%
% Using SORT on filenames will sort any of char(0:45), including the printing
% characters ' !"#$%&''()*+,-', before the file extension separator character '.'.
% Therefore this function splits the name and extension and sorts them separately.
```

```
%
% Similarly the file separator character within filepaths can cause longer
% directory names to sort before shorter ones, as char(0:46)<'/' and
% char(0:91)<'\'. Check this example to see why this matters:
%
% >> X = {'A1\B', 'A+/B', 'A\B', 'A=/B', 'A/B'};
% >> sort(X)
% ans =   'A+/B'  'A/B'  'A1\B'  'A=/B'  'A\B'
% >> natsortfiles(X)
% ans =   'A\B'  'A/B'  'A1\B'  'A+/B'  'A=/B'
%
% NATSORTFILES splits filepaths at each file separator character and sorts
% every level of the directory hierarchy separately, ensuring that shorter
% directory names sort before longer, regardless of the characters in the names.
%
%% Examples %%
%
% >> A = {'a2.txt', 'a10.txt', 'a1.txt'};
% >> sort(A)
% ans = 'a1.txt'  'a10.txt'  'a2.txt'
% >> natsortfiles(A)
% ans = 'a1.txt'  'a2.txt'  'a10.txt'
%
% >> B = {'test_new.m'; 'test-old.m'; 'test.m'};
% >> sort(B) % Note '-' sorts before '.':
% ans =
%    'test-old.m'
%    'test.m'
%    'test_new.m'
% >> natsortfiles(B) % Shorter names before longer (dictionary sort):
% ans =
%    'test.m'
%    'test-old.m'
%    'test_new.m'
%
% >> C = {'test2.m'; 'test10-old.m'; 'test.m'; 'test10.m'; 'test1.m'};
% >> sort(C) % Wrong numeric order:
% ans =
%    'test.m'
%    'test1.m'
%    'test10-old.m'
%    'test10.m'
%    'test2.m'
% >> natsortfiles(C) % Shorter names before longer:
% ans =
%    'test.m'
%    'test1.m'
%    'test2.m'
%    'test10.m'
%    'test10-old.m'
%
%%% Directory Names:
% >> D = {'A2-old\test.m';'A10\test.m';'A2\test.m';'A1archive.zip';'A1\test.m'};
% >> sort(D) % Wrong numeric order, and '-' sorts before '\':
% ans =
%    'A10\test.m'
%    'A1\test.m'
%    'A1archive.zip'
%    'A2-old\test.m'
%    'A2\test.m'
% >> natsortfiles(D) % Shorter names before longer (dictionary sort):
% ans =
%    'A1archive.zip'
%    'A1\test.m'
%    'A2\test.m'
%    'A2-old\test.m'
%    'A10\test.m'
%
%% Input and Output Arguments %%
%
%%% Inputs (*=default):
% X   = CellArrayOfCharRowVectors, with filenames or filepaths to be sorted.
% rgx = Regular expression to match number substrings, '\d+'*
```

```
%     = [] uses the default regular expression, which matches integers.
% <options> can be supplied in any order and are passed directly to NATSORT.
%
%%% Outputs:
% Y   = CellArrayOfCharRowVectors, filenames of <X> sorted into natural-order.
% ndx = NumericMatrix, same size as <X>. Indices such that Y = X(ndx).
% dbg = CellVectorOfCellArrays, size 1xMAX(2+NumberOfDirectoryLevels).
%       Each cell contains the debug cell array for directory names, filenames,
%       and file extensions. Helps debug the regular expression. See NATSORT.
%
% See also SORT NATSORT NATSORTROWS DIR FILEPARTS FULLFILE NEXTNAME CELLSTR REGEXP ...
    IREGEXP SSCANF

%% Input Wrangling %%
%
assert(iscell(X),'First input <X> must be a cell array.')
tmp = cellfun('isclass',X,'char') & cellfun('size',X,1)<2 & cellfun('ndims',X)<3;
assert(all(tmp(:)),'First input <X> must be a cell array of strings (1xN character).')
%
if nargin>1
    varargin = [{rgx},varargin];
end
%
%% Split and Sort File Names/Paths %%
%
% Split full filepaths into file [path,name,extension]:
[pth,fnm,ext] = cellfun(@fileparts,X(:),'UniformOutput',false);
% Split path into {dir,subdir,subsubdir,...}:
pth = regexp(pth,'[^/\\]+','match'); % either / or \ as filesep.
len = cellfun('length',pth);
num = max(len);
vec = cell(numel(len),1);
%
% Natural-order sort of the file extensions and filenames:
if isempty(num)
    ndx = [];
    ids = [];
    dbg = {};
elseif nargout<3 % faster:
    [~,ndx] = natsort(ext,varargin{:});
    [~,ids] = natsort(fnm(ndx),varargin{:});
else % for debugging:
    [~,ndx,dbg{num+2}] = natsort(ext,varargin{:});
    [~,ids,tmp] = natsort(fnm(ndx),varargin{:});
    [~,idd] = sort(ndx);
    dbg{num+1} = tmp(idd,:);
end
ndx = ndx(ids);
%
% Natural-order sort of the directory names:
for k = num:-1:1
    idx = len>=k;
    vec(:) = {''};
    vec(idx) = cellfun(@(c)c(k),pth(idx));
    if nargout<3 % faster:
        [~,ids] = natsort(vec(ndx),varargin{:});
    else % for debugging:
        [~,ids,tmp] = natsort(vec(ndx),varargin{:});
        [~,idd] = sort(ndx);
        dbg{k} = tmp(idd,:);
    end
    ndx = ndx(ids);
end
%
% Return the sorted array and indices:
ndx = reshape(ndx,size(X));
X = X(ndx);
%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%natsortfiles
```

E.5 natsortfiles-doc.m

```
%% NATSORTFILES Examples
% The function <https://www.mathworks.com/matlabcentral/fileexchange/47434
% |NATSORTFILES|> sorts a cell array of filenames or filepaths (1xN char),
% taking into account any number values within the strings. This is known
% as a ‗natural order sort‗ or an ‗alphanumeric sort‗. Note that MATLAB's
% inbuilt <https://www.mathworks.com/help/matlab/ref/sort.html |SORT|> function
% sorts the character codes only (as does |SORT| in most programming languages).
%
% |NATSORTFILES| is not a naive natural-order sort, but splits and sorts
% filenames and file extensions separately, which means that |NATSORTFILES|
% sorts shorter filenames before longer ones: this is known as a ‗dictionary
% sort‗. For the same reason filepaths are split at every path-separator
% character (either |'\'| or |'/'|), and each directory level is sorted
% separately. See the "Explanation" sections below for more details.
%
% For sorting the rows of a cell array of strings use
% <https://www.mathworks.com/matlabcentral/fileexchange/47433 |NATSORTROWS|>.
%
% For sorting a cell array of strings use
% <https://www.mathworks.com/matlabcentral/fileexchange/34464 |NATSORT|>.
%
%% Basic Usage
% By default |NATSORTFILES| interprets consecutive digits as being part of
% a single integer, any remaining substring/s are treated as characters:
A = {'a2.txt', 'a10.txt', 'a1.txt'};
sort(A)
natsortfiles(A)
%% Output 2: Sort Index
% The second output argument is a numeric array of the sort indices |ndx|,
% such that |Y = X(ndx)| where |Y = natsortfiles(X)|:
[~,ndx] = natsortfiles(A)
%% Output 3: Debugging Array
% The third output is a cell vector of cell arrays, where the cell arrays
% correspond to the directory hierarchy, filenames, and file extensions.
% The cell arrays contain all matched numbers (after converting to
% numeric using the specified |SSCANF| format) and all split character
% substrings. These cell arrays are useful for confirming that the
% numbers are being correctly identified by the regular expression.
% Note that the even columns contain any matched number values,
% while the odd columns contain any split substrings:
[~,~,dbg] = natsortfiles(A);
dbg{:}
%% Input 2: Regular Expression
% The optional second input argument is a regular expression which
% specifies the number matching:
B = {'1.3.txt','1.10.txt','1.2.txt'};
natsortfiles(B)   % by default match integers
natsortfiles(B, '\d+\.?\d*') % match decimal fractions
%% Inputs 3+: Optional Arguments
% Further inputs are passed directly to |NATSORT|, thus giving control over
% the case sensitivity, sort direction, and other options. See the
% |NATSORT| help for explanations and examples of the supported options:
C = {'B.txt','10.txt','1.txt','A.txt','2.txt'};
natsortfiles(C, [], 'descend')
natsortfiles(C, [], 'char<num')
%% Example with DIR and a Cell Array
% One common situation is to use <https://www.mathworks.com/help/matlab/ref/dir.html
% |DIR|> to identify files in a folder, sort them into the correct order,
% and then loop over them: below is an example of how to do this.
% Remember to <https://www.mathworks.com/help/matlab/matlab_prog/preallocating-arrays.html
% preallocate> all output arrays before the loop!
P = 'natsortfiles_test'; % directory path
S = dir(fullfile(P,'*.txt')); % get list of files in directory
C = natsortfiles({S.name}); % sort file names into order
for k = 1:numel(C)
    disp(fullfile(P,C{k}))
end
%% Example with DIR and a Structure
% Users who need to access the |DIR| structure fields can use |NATSORTFILE|'s
% second output to sort |DIR|'s output structure into the correct order:
```

```
P = 'natsortfiles_test'; % directory path
S = dir(fullfile(P,'*.txt')); % get list of files in directory
[¬,ndx] = natsortfiles({S.name}); % indices of correct order
S = S(ndx); % sort structure using indices
for k = 1:numel(S)
    fprintf('%-13s%s\n',S(k).name,S(k).date)
end
%% Explanation: Dictionary Sort
% Filenames and file extensions are separated by the extension separator:
% the period character |'.'|. Using a normal |SORT| the period gets sorted
% _after_ all of the characters from 0 to 45 (including |!"#$%&'()*+,-|,
% the space character, and all of the control characters, e.g. newlines,
% tabs, etc). This means that a naive natural-order sort will
% sort some short filenames after longer filenames. In order to provide
% the correct dictionary sort, with shorter filenames first, |NATSORTFILES|
% splits filenames from file extensions and sorts them separately:
D = {'test_ccc.m'; 'test-aaa.m'; 'test.m'; 'test.bbb.m'};
sort(D) % '-' sorts before '.'
natsort(D) % '-' sorts before '.'
natsortfiles(D) % correct dictionary sort
%% Explanation: Filenames
% |NATSORTFILES| combines a dictionary sort with a natural-order sort, so
% that the number values within the filenames are taken into consideration:
E = {'test2.m'; 'test10-old.m'; 'test.m'; 'test10.m'; 'test1.m'};
sort(E) % Wrong numeric order.
natsort(E) % Correct numeric order, but longer before shorter.
natsortfiles(E) % Correct numeric order and dictionary sort.
%% Explanation: Filepaths
% For the same reason, filepaths are split at each file path separator
% character (both |'/'| and |'\'| are treated as file path separators)
% and every level of the directory structure are sorted separately. This
% ensures that the directory names are sorted with a dictionary sort and
% that any numbers are taken into consideration:
F = {'A2-old\test.m';'A10\test.m';'A2\test.m';'AXarchive.zip';'A1\test.m'};
sort(F) % Wrong numeric order, and '-' sorts before '\':
natsort(F) % correct numeric order, but longer before shorter.
natsortfiles(F) % correct numeric order and dictionary sort.
%% Regular Expression: Decimal Numbers, E-notation, +/- Sign
% |NATSORTFILES| number matching can be customized to detect numbers with
% a decimal fraction, E-notation, a +/- sign, binary/hexadecimal, or other
% required features. The number matching is specified using an
% appropriate regular expression: see |NATSORT| for details and examples.
G = {'1.23V.csv','-1V.csv','+1.csv','+NaNV.csv','1.200V.csv'};
natsortfiles(G) % by default match integers
natsortfiles(G,'[-+]?(NaN|Inf|\d+\.?\d*)')
%% Bonus: Interactive Regular Expression Tool
% Regular expressions are powerful and compact, but getting them right is
% not always easy. One assistance is to download my interactive tool
% <https://www.mathworks.com/matlabcentral/fileexchange/48930 |IREGEXP|>,
% which lets you quickly try different regular expressions and see all of
% <https://www.mathworks.com/help/matlab/ref/regexp.html |REGEXP|>'s
% outputs displayed and updated as you type.
```

E.6 natsortfiles-test.m

```
function natsortfiles_test()
% Test function for NATSORTFILES.
%
% (c) 2014-2019 Stephen Cobeldick
%
% See also NATSORTFILES TESTFUN NATSORT_TEST NATSORTROWS_TEST
fun = @natsortfiles;
chk = testfun(fun);
%
%% Examples HTML %%
%
A = {'a2.txt', 'a10.txt', 'a1.txt'};
chk(A, fun, {'a1.txt', 'a2.txt', 'a10.txt'})
chk(A, fun, [], [3,1,2])
```

```
chk(A, fun, [], [], {{'a',2;'a',10;'a',1},{'.txt';'.txt';'.txt'}})
%
B = {'1.3.txt','1.10.txt','1.2.txt'};
chk(B, fun, {'1.2.txt', '1.3.txt', '1.10.txt'}, [3,1,2])
chk(B, '\d+\.?\d*', fun, {'1.10.txt', '1.2.txt', '1.3.txt'}, [2,3,1])
%
C = {'B.txt','10.txt','1.txt','A.txt','2.txt'};
chk(C, [],  'descend', fun, {'B.txt', 'A.txt', '10.txt', '2.txt', '1.txt'}, [])
chk(C, [], 'char<num', fun, {'A.txt', 'B.txt', '1.txt', '2.txt', '10.txt'})
%
P = 'natsortfiles_test';
S = dir(fullfile(P,'*.txt'));
chk({S.name}, fun, {'A_1.txt','A_1-new.txt','A_1_new.txt','A_2.txt',...
'A_3.txt','A_10.txt','A_100.txt','A_200.txt'})
%
chk({'test_ccc.m'; 'test-aaa.m'; 'test.m'; 'test.bbb.m'}, fun,... D
    {'test.m'; 'test-aaa.m'; 'test.bbb.m'; 'test_ccc.m'}, [3;2;4;1])
chk({'test2.m'; 'test10-old.m'; 'test.m'; 'test10.m'; 'test1.m'}, fun,... E
    {'test.m'; 'test1.m'; 'test2.m'; 'test10.m'; 'test10-old.m'}, [3;5;1;4;2])
chk({'A2-old\test.m';'A10\test.m';'A2\test.m';'AXarchive.zip';'A1\test.m'}, fun,... F
    {'AXarchive.zip';'A1\test.m';'A2\test.m';'A2-old\test.m';'A10\test.m'}, [4;5;3;1;2])
%
G = {'1.23V.csv','-1V.csv','+1.csv','+NaNV.csv','1.200V.csv'};
chk(G, fun,...
    {'1.23V.csv', '1.200V.csv', '+1.csv', '+NaNV.csv', '-1V.csv'}, [1,5,3,4,2])
chk(G, '[-+]?(NaN|Inf|\d+\.?\d*)', fun,...
    {'-1V.csv', '+1.csv', '1.200V.csv', '1.23V.csv', '+NaNV.csv'}, [2,3,5,1,4])
%
%% Examples Mfile Help
%
chk({'A1\B', 'A+/B', 'A\B', 'A=/B', 'A/B'}, fun,... X
    {'A\B', 'A/B', 'A1\B', 'A+/B', 'A=/B'}, [3,5,1,2,4])
chk({'a2.txt', 'a10.txt', 'a1.txt'}, fun,... A
    {'a1.txt', 'a2.txt', 'a10.txt'}, [3,1,2])
chk({'test_new.m'; 'test-old.m'; 'test.m'}, fun,... B
    {'test.m'; 'test-old.m'; 'test_new.m'}, [3;2;1])
chk({'test2.m'; 'test10-old.m'; 'test.m'; 'test10.m'; 'test1.m'}, fun,... C
    {'test.m'; 'test1.m'; 'test2.m'; 'test10.m'; 'test10-old.m'}, [3;5;1;4;2])
chk({'A2-old\test.m';'A10\test.m';'A2\test.m';'A1archive.zip';'A1\test.m'}, fun,... D
    {'A1archive.zip';'A1\test.m';'A2\test.m';'A2-old\test.m';'A10\test.m'}, [4;5;3;1;2])
%
%% Orientation
%
chk({}, fun, {}, [], {}) % empty!
chk(cell(0,2,3), fun, cell(0,2,3), nan(0,2,3)) % empty!
chk({'1';'10';'20';'2'}, fun,...
    {'1';'2';'10';'20'}, [1;4;2;3])
chk({'2','10','8';'#','a',' '}, fun,...
    {'2','10','#';'8',' ','a'}, [1,3,2;5,6,4])
%
%% Stability
%
chk({'';'';''}, fun, {'';'';''}, [1;2;3], {{'';'';''},{'';'';''}})
%
U = {'2';'3';'2';'1';'2'};
chk(U, [], 'ascend', fun,...
    {'1';'2';'2';'2';'3'}, [4;1;3;5;2])
chk(U, [], 'descend', fun,...
    {'3';'2';'2';'2';'1'}, [2;1;3;5;4])
%
V = {'x';'z';'y';'';'z';'';'x';'y'};
chk(V, [], 'ascend', fun,...
    {'';'';'x';'x';'y';'y';'z';'z'},[4;6;1;7;3;8;2;5])
chk(V, [], 'descend', fun,...
    {'z';'z';'y';'y';'x';'x';'';''},[2;5;3;8;1;7;4;6])
%
W = {'2x';'2z';'2y';'2';'2z';'2';'2x';'2y'};
chk(W, [], 'ascend', fun,...
    {'2';'2';'2x';'2x';'2y';'2y';'2z';'2z'},[4;6;1;7;3;8;2;5])
chk(W, [], 'descend', fun,...
    {'2z';'2z';'2y';'2y';'2x';'2x';'2';'2'},[2;5;3;8;1;7;4;6])
%
%% Extension and Separator Characters
```

```
%
chk({'A.x3','A.x20','A.x','A.x1'}, fun,...
    {'A.x','A.x1','A.x3','A.x20'}, [3,4,1,2])
chk({'A=.z','A.z','A..z','A-.z','A#.z'}, fun,...
    {'A.z','A#.z','A-.z','A..z','A=.z'}, [2,5,4,3,1])
chk({'A¬/B','A/B','A#/B','A=/B','A-/B'}, fun,...
    {'A/B','A#/B','A-/B','A=/B','A¬/B'}, [2,3,5,4,1])
%
%% Other Implementation Examples
%
% <https://blog.codinghorror.com/sorting-for-humans-natural-sort-order/>
chk({'z1.txt','z10.txt','z100.txt','z101.txt','z102.txt',...
'z11.txt','z12.txt','z13.txt','z14.txt','z15.txt','z16.txt','z17.txt',...
'z18.txt','z19.txt','z2.txt','z20.txt','z3.txt','z4.txt. ...
','z5.txt','z6.txt','z7.txt','z8.txt','z9.txt'}, fun,...
    {'z1.txt','z2.txt','z3.txt','z4.txt','z5.txt','...
    z6.txt','z7.txt','z8.txt','z9.txt','z10.txt','z11.txt',...
    'z12.txt','z13.txt','z14.txt','z15.txt','z16.txt','z17.txt',...
    'z18.txt','z19.txt','z20.txt','z100.txt','z101.txt','z102.txt'})
%
% <https://blog.jooq.org/2018/02/23/how-to-order-file-names-semantically-in-java/>
chk({'C:\temp\version-1.sql','C:\temp\version-10.1.sql','C:\temp\version-10.sql',...
'C:\temp\version-2.sql','C:\temp\version-21.sql'}, fun,...
    {'C:\temp\version-1.sql','C:\temp\version-2.sql','C:\temp\version-10.sql',...
    'C:\temp\version-10.1.sql','C:\temp\version-21.sql'})
%
% <http://www.davekoelle.com/alphanum.html>
chk({'z1.doc','z10.doc','z100.doc','z101.doc',...
'z102.doc','z11.doc','z12.doc','z13.doc','z14.doc',...
'z15.doc','z16.doc','z17.doc','z18.doc','z19.doc',...
'z2.doc','z20.doc','z3.doc','z4.doc','z5.doc','z6.doc',...
'z7.doc','z8.doc','z9.doc'}, fun, ...
    {'z1.doc','z2.doc','z3.doc','z4.doc','z5.doc','z6.doc',...
    'z7.doc','z8.doc','z9.doc','z10.doc','z11.doc','z12.doc',...
    'z13.doc','z14.doc','z15.doc','z16.doc','z17.doc','z18.doc',...
    'z19.doc','z20.doc','z100.doc','z101.doc','z102.doc'})
%
% <https://sourcefrog.net/projects/natsort/>
chk({'rfc1.txt';'rfc2086.txt';'rfc822.txt'}, fun,...
    {'rfc1.txt';'rfc822.txt';'rfc2086.txt'})
%
% <https://www.strchr.com/natural_sorting>
chk({'picture 1.png','picture 10.png','picture 100.png','picture 11.png','picture ...
    2.png','picture 21.png','picture 2_10.png','picture 2_9.png','picture ...
    3.png','picture 3b.png','picture A.png'}, fun,...
    {'picture 1.png','picture 2.png','picture 2_9.png','picture 2_10.png','picture ...
        3.png','picture 3b.png','picture 10.png','picture 11.png','picture ...
        21.png','picture 100.png','picture A.png'})
%
chk() % display summary
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

E.7 `screencapture.m`

Följande program är skrivet av Yair Altman [25].

```
function imageData = screencapture(varargin)
% screencapture - get a screen-capture of a figure frame, component handle, or screen ...
    area rectangle
%
% ScreenCapture gets a screen-capture of any Matlab GUI handle (including desktop,
% figure, axes, image or uicontrol), or a specified area rectangle located relative to
% the specified handle. Screen area capture is possible by specifying the root (desktop)
% handle (=0). The output can be either to an image file or to a Matlab matrix (useful
% for displaying via imshow() or for further processing) or to the system clipboard.
% This utility also enables adding a toolbar button for easy interactive screen-capture.
%
% Syntax:
```

```
%    imageData = screencapture(handle, position, target, 'PropName',PropValue, ...)
%
% Input Parameters:
%    handle   - optional handle to be used for screen-capture origin.
%               If empty/unsupplied then current figure (gcf) will be used.
%    position - optional position array in pixels: [x,y,width,height].
%               If empty/unsupplied then the handle's position vector will be used.
%               If both handle and position are empty/unsupplied then the position
%                 will be retrieved via interactive mouse-selection.
%               If handle is an image, then position is in data (not pixel) units, so the
%                 captured region remains the same after figure/axes resize (like imcrop)
%    target   - optional filename for storing the screen-capture, or the
%              'clipboard'/'printer' strings.
%               If empty/unsupplied then no output to file will be done.
%               The file format will be determined from the extension (JPG/PNG/...).
%               Supported formats are those supported by the imwrite function.
%    'PropName',PropValue -
%               optional list of property pairs (e.g., ...
    'target','myImage.png','pos',[10,20,30,40],'handle',gca)
%              PropNames may be abbreviated and are case-insensitive.
%              PropNames may also be given in whichever order.
%              Supported PropNames are:
%                - 'handle'    (default: gcf handle)
%                - 'position'  (default: gcf position array)
%                - 'target'    (default: '')
%                - 'toolbar'   (figure handle; default: gcf)
%                     this adds a screen-capture button to the figure's toolbar
%                     If this parameter is specified, then no screen-capture
%                       will take place and the returned imageData will be [].
%
% Output parameters:
%    imageData - image data in a format acceptable by the imshow function
%                 If neither target nor imageData were specified, the user will be
%                   asked to interactively specify the output file.
%
% Examples:
%    imageData = screencapture;  % interactively select screen-capture rectangle
%    imageData = screencapture(hListbox);  % capture image of a uicontrol
%    imageData = screencapture(0,  [20,30,40,50]);  % capture a small desktop region
%    imageData = screencapture(gcf,[20,30,40,50]);  % capture a small figure region
%    imageData = screencapture(gca,[10,20,30,40]);  % capture a small axes region
%      imshow(imageData);  % display the captured image in a matlab figure
%      imwrite(imageData,'myImage.png');  % save the captured image to file
%    img = imread('cameraman.tif');
%      hImg = imshow(img);
%      screencapture(hImg,[60,35,140,80]);  % capture a region of an image
%    screencapture(gcf,[],'myFigure.jpg');  % capture the entire figure into file
%    screencapture(gcf,[],'clipboard');     % capture the entire figure into clipboard
%    screencapture(gcf,[],'printer');       % print the entire figure
%    screencapture('handle',gcf,'target','myFigure.jpg'); % same as previous, save to file
%    screencapture('handle',gcf,'target','clipboard');    % same as previous, copy to ...
    clipboard
%    screencapture('handle',gcf,'target','printer');      % same as previous, send to ...
    printer
%    screencapture('toolbar',gcf);  % adds a screen-capture button to gcf's toolbar
%    screencapture('toolbar',[],'target','sc.bmp'); % same with default output filename
%
% Technical description:
%    http://UndocumentedMatlab.com/blog/screencapture-utility/
%
% Bugs and suggestions:
%    Please send to Yair Altman (altmany at gmail dot com)
%
% See also:
%    imshow, imwrite, print
%
% Release history:
%    1.17 2016-05-16: Fix annoying warning about JavaFrame property becoming obsolete ...
    someday (yes, we know...)
%    1.16 2016-04-19: Fix for deployed application suggested by Dwight Bartholomew
%    1.10 2014-11-25: Added the 'print' target
%    1.9  2014-11-25: Fix for saving GIF files
%    1.8  2014-11-16: Fixes for R2014b
```

```
%    1.7  2014-04-28: Fixed bug when capturing interactive selection
%    1.6  2014-04-22: Only enable image formats when saving to an unspecified file via ...
    uiputfile
%    1.5  2013-04-18: Fixed bug in capture of non-square image; fixes for Win64
%    1.4  2013-01-27: Fixed capture of Desktop (root); enabled rbbox anywhere on ...
    desktop (not necesarily in a Matlab figure); enabled output to clipboard (based on ...
    Jiro Doke's imclipboard utility); edge-case fixes; added Java compatibility check
%    1.3  2012-07-23: Capture current object (uicontrol/axes/figure) if w=h=0 (e.g., by ...
    clicking a single point); extra input args sanity checks; fix for docked windows ...
    and image axes; include axes labels & ticks by default when capturing axes; use ...
    data-units position vector when capturing images; many edge-case fixes
%    1.2  2011-01-16: another performance boost (thanks to Jan Simon); some ...
    compatibility fixes for Matlab 6.5 (untested)
%    1.1  2009-06-03: Handle missing output format; performance boost (thanks to Urs); ...
    fix minor root-handle bug; added toolbar button option
%    1.0  2009-06-02: First version posted on <a ...
    href="http://www.mathworks.com/matlabcentral/fileexchange/authors/27420">MathWorks ...
    File Exchange</a>

% License to use and modify this code is granted freely to all interested, as long as ...
    the original author is
% referenced and attributed as such. The original author maintains the right to be ...
    solely associated with this work.

% Programmed and Copyright by Yair M. Altman: altmany(at)gmail.com
% $Revision: 1.17 $  $Date: 2016/05/16 17:59:36 $

    % Ensure that java awt is enabled...
    if ~usejava('awt')
        error('YMA:screencapture:NeedAwt','ScreenCapture requires Java to run.');
    end

    % Ensure that our Java version supports the Robot class (requires JVM 1.3+)
    try
        robot = java.awt.Robot; %#ok<NASGU>
    catch
        uiwait(msgbox({['Your Matlab installation is so old that its Java engine (' ...
            version('-java') ...
                        ') does not have a java.awt.Robot class. '], ' ', ...
                        'Without this class, taking a screen-capture is impossible.', ' ...
                            ', ...
                        'So, either install JVM 1.3 or higher, or use a newer Matlab ...
                            release.'}, ...
                        'ScreenCapture', 'warn'));
        if nargout, imageData = [];  end
        return;
    end

    % Process optional arguments
    paramsStruct = processArgs(varargin{:});

    % If toolbar button requested, add it and exit
    if ~isempty(paramsStruct.toolbar)

        % Add the toolbar button
        addToolbarButton(paramsStruct);

        % Return the figure to its pre-undocked state (when relevant)
        redockFigureIfRelevant(paramsStruct);

        % Exit immediately (do NOT take a screen-capture)
        if nargout,  imageData = [];  end
        return;
    end

    % Convert position from handle-relative to desktop Java-based pixels
    [paramsStruct, msgStr] = convertPos(paramsStruct);

    % Capture the requested screen rectangle using java.awt.Robot
    imgData = getScreenCaptureImageData(paramsStruct.position);

    % Return the figure to its pre-undocked state (when relevant)
    redockFigureIfRelevant(paramsStruct);
```

```
139
140     % Save image data in file or clipboard, if specified
141     if ¬isempty(paramsStruct.target)
142         if strcmpi(paramsStruct.target,'clipboard')
143             if ¬isempty(imgData)
144                 imclipboard(imgData);
145             else
146                 msgbox('No image area selected — not copying image to ...
                        clipboard','ScreenCapture','warn');
147             end
148         elseif strncmpi(paramsStruct.target,'print',5)  % 'print' or 'printer'
149             if ¬isempty(imgData)
150                 hNewFig = figure('visible','off');
151                 imshow(imgData);
152                 print(hNewFig);
153                 delete(hNewFig);
154             else
155                 msgbox('No image area selected — not printing ...
                        screenshot','ScreenCapture','warn');
156             end
157         else  % real filename
158             if ¬isempty(imgData)
159                 imwrite(imgData,paramsStruct.target);
160             else
161                 msgbox(['No image area selected — not saving image file ' ...
                        paramsStruct.target],'ScreenCapture','warn');
162             end
163         end
164     end
165
166     % Return image raster data to user, if requested
167     if nargout
168         imageData = imgData;
169
170     % If neither output formats was specified (neither target nor output data)
171     elseif isempty(paramsStruct.target) & ¬isempty(imgData)  %#ok ML6
172         % Ask the user to specify a file
173         %error('YMA:screencapture:noOutput','No output specified for ScreenCapture: ...
                specify the output filename and/or output data');
174         %format = '*.*';
175         formats = imformats;
176         for idx = 1 : numel(formats)
177             ext = sprintf('*.%s;',formats(idx).ext{:});
178             format(idx,1:2) = {ext(1:end-1), formats(idx).description}; %#ok<AGROW>
179         end
180         [filename,pathname] = uiputfile(format,'Save screen capture as');
181         if ¬isequal(filename,0) & ¬isequal(pathname,0)  %#ok Matlab6 compatibility
182             try
183                 filename = fullfile(pathname,filename);
184                 imwrite(imgData,filename);
185             catch  % possibly a GIF file that requires indexed colors
186                 [imgData,map] = rgb2ind(imgData,256);
187                 imwrite(imgData,map,filename);
188             end
189         else
190             % TODO — copy to clipboard
191         end
192     end
193
194     % Display msgStr, if relevant
195     if ¬isempty(msgStr)
196         uiwait(msgbox(msgStr,'ScreenCapture'));
197         drawnow; pause(0.05);  % time for the msgbox to disappear
198     end
199
200     return;  % debug breakpoint
201
202 %% Process optional arguments
203 function paramsStruct = processArgs(varargin)
204
205     % Get the properties in either direct or P-V format
206     [regParams, pvPairs] = parseparams(varargin);
207
```

```
208     % Now process the optional P-V params
209     try
210         % Initialize
211         paramName = [];
212         paramsStruct = [];
213         paramsStruct.handle = [];
214         paramsStruct.position = [];
215         paramsStruct.target = '';
216         paramsStruct.toolbar = [];
217         paramsStruct.wasDocked = 0;       % no false available in ML6
218         paramsStruct.wasInteractive = 0;  % no false available in ML6
219
220         % Parse the regular (non-named) params in recption order
221         if ~isempty(regParams) & (isempty(regParams{1}) | ishandle(regParams{1}(1)))  ...
                %#ok ML6
222             paramsStruct.handle = regParams{1};
223             regParams(1) = [];
224         end
225         if ~isempty(regParams) & isnumeric(regParams{1}) & (length(regParams{1}) == 4)  ...
                %#ok ML6
226             paramsStruct.position = regParams{1};
227             regParams(1) = [];
228         end
229         if ~isempty(regParams) & ischar(regParams{1})  %#ok ML6
230             paramsStruct.target = regParams{1};
231         end
232
233         % Parse the optional param PV pairs
234         supportedArgs = {'handle','position','target','toolbar'};
235         while ~isempty(pvPairs)
236
237             % Disregard empty propNames (may be due to users mis-interpretting the ...
                    syntax help)
238             while ~isempty(pvPairs) & isempty(pvPairs{1})  %#ok ML6
239                 pvPairs(1) = [];
240             end
241             if isempty(pvPairs)
242                 break;
243             end
244
245             % Ensure basic format is valid
246             paramName = '';
247             if ~ischar(pvPairs{1})
248                 error('YMA:screencapture:invalidProperty','Invalid property passed to ...
                        ScreenCapture');
249             elseif length(pvPairs) == 1
250                 if isempty(paramsStruct.target)
251                     paramsStruct.target = pvPairs{1};
252                     break;
253                 else
254                     error('YMA:screencapture:noPropertyValue',['No value specified for ...
                            property ''' pvPairs{1} '''']);
255                 end
256             end
257
258             % Process parameter values
259             paramName  = pvPairs{1};
260             if strcmpi(paramName,'filename')  % backward compatibility
261                 paramName = 'target';
262             end
263             paramValue = pvPairs{2};
264             pvPairs(1:2) = [];
265             idx = find(strncmpi(paramName,supportedArgs,length(paramName)));
266             if ~isempty(idx)
267                 %paramsStruct.(lower(supportedArgs{idx(1)})) = paramValue;  % ...
                        incompatible with ML6
268                 paramsStruct = setfield(paramsStruct, lower(supportedArgs{idx(1)}), ...
                        paramValue);  %#ok ML6
269
270                 % If 'toolbar' param specified, then it cannot be left empty - use gcf
271                 if strncmpi(paramName,'toolbar',length(paramName)) & ...
                        isempty(paramsStruct.toolbar)  %#ok ML6
272                     paramsStruct.toolbar = getCurrentFig;
```

```
273                 end
274
275             elseif isempty(paramsStruct.target)
276                 paramsStruct.target = paramName;
277                 pvPairs = {paramValue, pvPairs{:}};  %#ok (more readable this way, ...
                        although a bit less efficient...)
278
279             else
280                 supportedArgsStr = sprintf('''%s'',',supportedArgs{:});
281                 error('YMA:screencapture:invalidProperty','%s \n%s', ...
282                       'Invalid property passed to ScreenCapture', ...
283                       ['Supported property names are: ' supportedArgsStr(1:end-1)]);
284             end
285         end  % loop pvPairs
286
287     catch
288         if ~isempty(paramName),  paramName = [' ''' paramName ''''];  end
289         error('YMA:screencapture:invalidProperty','Error setting ScreenCapture property ...
                %s:\n%s',paramName,lasterr); %#ok<LERR>
290     end
291 %end  % processArgs
292
293 %% Convert position from handle-relative to desktop Java-based pixels
294 function [paramsStruct, msgStr] = convertPos(paramsStruct)
295     msgStr = '';
296     try
297         % Get the screen-size for later use
298         screenSize = get(0,'ScreenSize');
299
300         % Get the containing figure's handle
301         hParent = paramsStruct.handle;
302         if isempty(paramsStruct.handle)
303             paramsStruct.hFigure = getCurrentFig;
304             hParent = paramsStruct.hFigure;
305         else
306             paramsStruct.hFigure = ancestor(paramsStruct.handle,'figure');
307         end
308
309         % To get the acurate pixel position, the figure window must be undocked
310         try
311             if strcmpi(get(paramsStruct.hFigure,'WindowStyle'),'docked')
312                 set(paramsStruct.hFigure,'WindowStyle','normal');
313                 drawnow; pause(0.25);
314                 paramsStruct.wasDocked = 1;  % no true available in ML6
315             end
316         catch
317             % never mind - ignore...
318         end
319
320         % The figure (if specified) must be in focus
321         if ~isempty(paramsStruct.hFigure) & ishandle(paramsStruct.hFigure)  %#ok ML6
322             isFigureValid = 1;  % no true available in ML6
323             figure(paramsStruct.hFigure);
324         else
325             isFigureValid = 0;  % no false available in ML6
326         end
327
328         % Flush all graphic events to ensure correct rendering
329         drawnow; pause(0.01);
330
331         % No handle specified
332         wasPositionGiven = 1;  % no true available in ML6
333         if isempty(paramsStruct.handle)
334
335             % Set default handle, if not supplied
336             paramsStruct.handle = paramsStruct.hFigure;
337
338             % If position was not specified, get it interactively using RBBOX
339             if isempty(paramsStruct.position)
340                 [paramsStruct.position, jFrameUsed, msgStr] = ...
                        getInteractivePosition(paramsStruct.hFigure); %#ok<ASGLU> ...
                        jFrameUsed is unused
341                 paramsStruct.wasInteractive = 1;  % no true available in ML6
```

```
342                 wasPositionGiven = 0;  % no false available in ML6
343             end
344
345         elseif ¬ishandle(paramsStruct.handle)
346             % Handle was supplied - ensure it is a valid handle
347             error('YMA:screencapture:invalidHandle','Invalid handle passed to ...
                    ScreenCapture');
348
349         elseif isempty(paramsStruct.position)
350             % Handle was supplied but position was not, so use the handle's position
351             paramsStruct.position = getPixelPos(paramsStruct.handle);
352             paramsStruct.position(1:2) = 0;
353             wasPositionGiven = 0;  % no false available in ML6
354
355         elseif ¬isnumeric(paramsStruct.position) | (length(paramsStruct.position) ≠ 4)  ...
                %#ok ML6
356             % Both handle & position were supplied - ensure a valid pixel position vector
357             error('YMA:screencapture:invalidPosition','Invalid position vector passed ...
                    to ScreenCapture: \nMust be a [x,y,w,h] numeric pixel array');
358         end
359
360         % Capture current object (uicontrol/axes/figure) if w=h=0 (single-click in ...
                interactive mode)
361         if paramsStruct.position(3)≤0 | paramsStruct.position(4)≤0  %#ok ML6
362             %TODO - find a way to single-click another Matlab figure (the following ...
                    does not work)
363             %paramsStruct.position = getPixelPos(ancestor(hittest,'figure'));
364             paramsStruct.position = getPixelPos(paramsStruct.handle);
365             paramsStruct.position(1:2) = 0;
366             paramsStruct.wasInteractive = 0;  % no false available in ML6
367             wasPositionGiven = 0;  % no false available in ML6
368         end
369
370         % First get the parent handle's desktop-based Matlab pixel position
371         parentPos = [0,0,0,0];
372         dX = 0;
373         dY = 0;
374         dW = 0;
375         dH = 0;
376         if ¬isFigure(hParent)
377             % Get the reguested component's pixel position
378             parentPos = getPixelPos(hParent, 1);  % no true available in ML6
379
380             % Axes position inaccuracy estimation
381             ΔX = 3;
382             ΔY = -1;
383
384             % Fix for images
385             if isImage(hParent)  % | (isAxes(hParent) & ...
                    strcmpi(get(hParent,'YDir'),'reverse'))  %#ok ML6
386
387                 % Compensate for resized image axes
388                 hAxes = get(hParent,'Parent');
389                 if all(get(hAxes,'DataAspectRatio')==1)  % sanity check: this is the ...
                        normal behavior
390                     % Note 18/4/2013: the following fails for non-square images
391                     %actualImgSize = min(parentPos(3:4));
392                     %dX = (parentPos(3) - actualImgSize) / 2;
393                     %dY = (parentPos(4) - actualImgSize) / 2;
394                     %parentPos(3:4) = actualImgSize;
395
396                     % The following should work for all types of images
397                     actualImgSize = size(get(hParent,'CData'));
398                     dX = (parentPos(3) - min(parentPos(3),actualImgSize(2))) / 2;
399                     dY = (parentPos(4) - min(parentPos(4),actualImgSize(1))) / 2;
400                     parentPos(3:4) = actualImgSize([2,1]);
401                     %parentPos(3) = max(parentPos(3),actualImgSize(2));
402                     %parentPos(4) = max(parentPos(4),actualImgSize(1));
403                 end
404
405                 % Fix user-specified img positions (but not auto-inferred ones)
406                 if wasPositionGiven
407
```

```
408                     % In images, use data units rather than pixel units
409                     % Reverse the YDir
410                     ymax = max(get(hParent,'YData'));
411                     paramsStruct.position(2) = ymax - paramsStruct.position(2) - ...
                            paramsStruct.position(4);
412
413                     % Note: it would be best to use hgconvertunits, but:
414                     % ^^^^  (1) it fails on Matlab 6, and (2) it doesn't accept Data units
415                     %paramsStruct.position = hgconvertunits(hFig, ...
                            paramsStruct.position, 'Data', 'pixel', hParent);  % fails!
416                     xLims = get(hParent,'XData');
417                     yLims = get(hParent,'YData');
418                     xPixelsPerData = parentPos(3) / (diff(xLims) + 1);
419                     yPixelsPerData = parentPos(4) / (diff(yLims) + 1);
420                     paramsStruct.position(1) = ...
                            round((paramsStruct.position(1)-xLims(1)) * xPixelsPerData);
421                     paramsStruct.position(2) = ...
                            round((paramsStruct.position(2)-yLims(1)) * yPixelsPerData + 2*dY);
422                     paramsStruct.position(3) = round( paramsStruct.position(3) * ...
                            xPixelsPerData);
423                     paramsStruct.position(4) = round( paramsStruct.position(4) * ...
                            yPixelsPerData);
424
425                     % Axes position inaccuracy estimation
426                     if strcmpi(computer('arch'),'win64')
427                         dX = 7;
428                         dY = -7;
429                     else
430                         dX = 3;
431                         dY = -3;
432                     end
433
434                 else  % axes/image position was auto-infered (entire image)
435                     % Axes position inaccuracy estimation
436                     if strcmpi(computer('arch'),'win64')
437                         dX = 6;
438                         dY = -6;
439                     else
440                         dX = 2;
441                         dY = -2;
442                     end
443                     dW = -2*dX;
444                     dH = -2*dY;
445                 end
446             end
447
448             %hFig = ancestor(hParent,'figure');
449             hParent = paramsStruct.hFigure;
450
451         elseif paramsStruct.wasInteractive  % interactive figure rectangle
452
453             % Compensate for 1px rbbox inaccuracies
454             dX = 2;
455             dY = -2;
456
457         else  % non-interactive figure
458
459             % Compensate 4px figure boundaries = difference betweeen OuterPosition and ...
                    Position
460             dX = -1;
461             dY = 1;
462         end
463         %disp(paramsStruct.position)  % for debugging
464
465         % Now get the pixel position relative to the monitor
466         figurePos = getPixelPos(hParent);
467         desktopPos = figurePos + parentPos;
468
469         % Now convert to Java-based pixels based on screen size
470         % Note: multiple monitors are automatically handled correctly, since all
471         % ^^^^  Java positions are relative to the main monitor's top-left corner
472         javaX  = desktopPos(1) + paramsStruct.position(1) + dX + dX;
473         javaY  = screenSize(4) - desktopPos(2) - paramsStruct.position(2) - ...
```

```
            paramsStruct.position(4) + ΔY + dY;
        width  = paramsStruct.position(3) + dW;
        height = paramsStruct.position(4) + dH;
        paramsStruct.position = round([javaX, javaY, width, height]);
        %paramsStruct.position

        % Ensure the figure is at the front so it can be screen-captured
        if isFigureValid
            figure(hParent);
            drawnow;
            pause(0.02);
        end
    catch
        % Maybe root/desktop handle (root does not have a 'Position' prop so ...
            getPixelPos croaks
        if isequal(double(hParent),0)  % =root/desktop handle;  handles case of hParent=[]
            javaX = paramsStruct.position(1) - 1;
            javaY = screenSize(4) - paramsStruct.position(2) - paramsStruct.position(4) ...
                - 1;
            paramsStruct.position = [javaX, javaY, paramsStruct.position(3:4)];
        end
    end
%end  % convertPos

%% Interactively get the requested capture rectangle
function [positionRect, jFrameUsed, msgStr] = getInteractivePosition(hFig)
    msgStr = '';
    try
        % First try the invisible-figure approach, in order to
        % enable rbbox outside any existing figure boundaries
        f = figure('units','pixel','pos',[-100,-100,10,10],'HitTest','off');
        drawnow; pause(0.01);
        oldWarn = warning('off','MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');
        jf = get(handle(f),'JavaFrame');
        warning(oldWarn);
        try
            jWindow = jf.fFigureClient.getWindow;
        catch
            try
                jWindow = jf.fHG1Client.getWindow;
            catch
                jWindow = jf.getFigurePanelContainer.getParent.getTopLevelAncestor;
            end
        end
        com.sun.awt.AWTUtilities.setWindowOpacity(jWindow,0.05);  %=nearly transparent ...
            (not fully so that mouse clicks are captured)
        jWindow.setMaximized(1);  % no true available in ML6
        jFrameUsed = 1;  % no true available in ML6
        msg = {'Mouse-click and drag a bounding rectangle for screen-capture ' ...
               ... %'or single-click any Matlab figure to capture the entire figure.' ...
              };
    catch
        % Something failed, so revert to a simple rbbox on a visible figure
        try delete(f); drawnow; catch, end  %Cleanup...
        jFrameUsed = 0;  % no false available in ML6
        msg = {'Mouse-click within any Matlab figure and then', ...
               'drag a bounding rectangle for screen-capture,', ...
               'or single-click to capture the entire figure'};
    end
    uiwait(msgbox(msg,'ScreenCapture'));

    k = waitforbuttonpress;  %#ok k is unused
    %hFig = getCurrentFig;
    %p1 = get(hFig,'CurrentPoint');
    positionRect = rbbox;
    %p2 = get(hFig,'CurrentPoint');

    if jFrameUsed
        jFrameOrigin = getPixelPos(f);
        delete(f); drawnow;
        try
            figOrigin = getPixelPos(hFig);
        catch  % empty/invalid hFig handle
```

```
543             figOrigin = [0,0,0,0];
544         end
545     else
546         if isempty(hFig)
547             jFrameOrigin = getPixelPos(gcf);
548         else
549             jFrameOrigin = [0,0,0,0];
550         end
551         figOrigin = [0,0,0,0];
552     end
553     positionRect(1:2) = positionRect(1:2) + jFrameOrigin(1:2) - figOrigin(1:2);
554
555     if prod(positionRect(3:4)) > 0
556         msgStr = sprintf('%dx%d area captured',positionRect(3),positionRect(4));
557     end
558 %end  % getInteractivePosition
559
560 %% Get current figure (even if its handle is hidden)
561 function hFig = getCurrentFig
562     oldState = get(0,'showHiddenHandles');
563     set(0,'showHiddenHandles','on');
564     hFig = get(0,'CurrentFigure');
565     set(0,'showHiddenHandles',oldState);
566 %end  % getCurrentFig
567
568 %% Get ancestor figure - used for old Matlab versions that don't have a built-in ancestor()
569 function hObj = ancestor(hObj,type)
570     if ~isempty(hObj) & ishandle(hObj)  %#ok for Matlab 6 compatibility
571         try
572             hObj = get(hObj,'Ancestor');
573         catch
574             % never mind...
575         end
576         try
577             %if ~isa(handle(hObj),type)  % this is best but always returns 0 in Matlab 6!
578             %if ~isprop(hObj,'type') | ~strcmpi(get(hObj,'type'),type)  % no isprop() ...
                    in ML6!
579             try
580                 objType = get(hObj,'type');
581             catch
582                 objType = '';
583             end
584             if ~strcmpi(objType,type)
585                 try
586                     parent = get(handle(hObj),'parent');
587                 catch
588                     parent = hObj.getParent;  % some objs have no 'Parent' prop, just ...
                            this method...
589                 end
590                 if ~isempty(parent)  % empty parent means root ancestor, so exit
591                     hObj = ancestor(parent,type);
592                 end
593             end
594         catch
595             % never mind...
596         end
597     end
598 %end  % ancestor
599
600 %% Get position of an HG object in specified units
601 function pos = getPos(hObj,field,units)
602     % Matlab 6 did not have hgconvertunits so use the old way...
603     oldUnits = get(hObj,'units');
604     if strcmpi(oldUnits,units)  % don't modify units unless we must!
605         pos = get(hObj,field);
606     else
607         set(hObj,'units',units);
608         pos = get(hObj,field);
609         set(hObj,'units',oldUnits);
610     end
611 %end  % getPos
612
613 %% Get pixel position of an HG object - for Matlab 6 compatibility
```

```
614 function pos = getPixelPos(hObj,varargin)
615     persistent originalObj
616     try
617         stk = dbstack;
618         if ¬strcmp(stk(2).name,'getPixelPos')
619             originalObj = hObj;
620         end
621 
622         if isFigure(hObj) %| isAxes(hObj)
623         %try
624             pos = getPos(hObj,'OuterPosition','pixels');
625         else  %catch
626             % getpixelposition is unvectorized unfortunately!
627             pos = getpixelposition(hObj,varargin{:});
628 
629             % add the axes labels/ticks if relevant (plus a tiny margin to fix 2px ...
                    label/title inconsistencies)
630             if isAxes(hObj) & ¬isImage(originalObj)  %#ok ML6
631                 tightInsets = getPos(hObj,'TightInset','pixel');
632                 pos = pos + tightInsets.*[-1,-1,1,1] + [-1,1,1+tightInsets(1:2)];
633             end
634         end
635     catch
636         try
637             % Matlab 6 did not have getpixelposition nor hgconvertunits so use the old ...
                    way...
638             pos = getPos(hObj,'Position','pixels');
639         catch
640             % Maybe the handle does not have a 'Position' prop (e.g., text/line/plot) - ...
                    use its parent
641             pos = getPixelPos(get(hObj,'parent'),varargin{:});
642         end
643     end
644 
645     % Handle the case of missing/invalid/empty HG handle
646     if isempty(pos)
647         pos = [0,0,0,0];
648     end
649 %end  % getPixelPos
650 
651 %% Adds a ScreenCapture toolbar button
652 function addToolbarButton(paramsStruct)
653     % Ensure we have a valid toolbar handle
654     hFig = ancestor(paramsStruct.toolbar,'figure');
655     if isempty(hFig)
656         error('YMA:screencapture:badToolbar','the ''Toolbar'' parameter must contain a ...
                valid GUI handle');
657     end
658     set(hFig,'ToolBar','figure');
659     hToolbar = findall(hFig,'type','uitoolbar');
660     if isempty(hToolbar)
661         error('YMA:screencapture:noToolbar','the ''Toolbar'' parameter must contain a ...
                figure handle possessing a valid toolbar');
662     end
663     hToolbar = hToolbar(1);  % just in case there are several toolbars... - use only ...
            the first
664 
665     % Prepare the camera icon
666     icon = ['3333333333333333'; ...
667             '3333333333333333'; ...
668             '3333300000333333'; ...
669             '3333065556033333'; ...
670             '3000000000000033'; ...
671             '3022222222222033'; ...
672             '3022220002222033'; ...
673             '3022203110222033'; ...
674             '3022201110222033'; ...
675             '3022204440222033'; ...
676             '3022220002222033'; ...
677             '3022222222222033'; ...
678             '3000000000000033'; ...
679             '3333333333333333'; ...
680             '3333333333333333'; ...
```

```
681             '3333333333333333'];
682     cm = [   0      0      0; ...  % black
683              0   0.60      1; ...  % light blue
684           0.53   0.53   0.53; ...  % light gray
685            NaN    NaN    NaN; ...  % transparent
686              0   0.73      0; ...  % light green
687           0.27   0.27   0.27; ...  % gray
688           0.13   0.13   0.13];     % dark gray
689     cdata = ind2rgb(uint8(icon-'0'),cm);
690
691     % If the button does not already exit
692     hButton = findall(hToolbar,'Tag','ScreenCaptureButton');
693     tooltip = 'Screen capture';
694     if ~isempty(paramsStruct.target)
695         tooltip = [tooltip ' to ' paramsStruct.target];
696     end
697     if isempty(hButton)
698         % Add the button with the icon to the figure's toolbar
699         hButton = uipushtool(hToolbar, 'CData',cdata, 'Tag','ScreenCaptureButton', ...
                'TooltipString',tooltip, 'ClickedCallback',['screencapture(''' ...
                paramsStruct.target ''')']);  %#ok unused
700     else
701         % Otherwise, simply update the existing button
702         set(hButton, 'CData',cdata, 'Tag','ScreenCaptureButton', ...
                'TooltipString',tooltip, 'ClickedCallback',['screencapture(''' ...
                paramsStruct.target ''')']);
703     end
704 %end  % addToolbarButton
705
706 %% Java-get the actual screen-capture image data
707 function imgData = getScreenCaptureImageData(positionRect)
708     if isempty(positionRect) | all(positionRect==0) | positionRect(3)<=0 | ...
            positionRect(4)<=0  %#ok ML6
709         imgData = [];
710     else
711         % Use java.awt.Robot to take a screen-capture of the specified screen area
712         rect = java.awt.Rectangle(positionRect(1), positionRect(2), positionRect(3), ...
                positionRect(4));
713         robot = java.awt.Robot;
714         jImage = robot.createScreenCapture(rect);
715
716         % Convert the resulting Java image to a Matlab image
717         % Adapted for a much-improved performance from:
718         % http://www.mathworks.com/support/solutions/data/1-2WPAYR.html
719         h = jImage.getHeight;
720         w = jImage.getWidth;
721         %imgData = zeros([h,w,3],'uint8');
722         %pixelsData = uint8(jImage.getData.getPixels(0,0,w,h,[]));
723         %for i = 1 : h
724         %    base = (i-1)*w*3+1;
725         %    imgData(i,1:w,:) = deal(reshape(pixelsData(base:(base+3*w-1)),3,w)');
726         %end
727
728         % Performance further improved based on feedback from Urs Schwartz:
729         %pixelsData = reshape(typecast(jImage.getData.getDataStorage,'uint32'),w,h).';
730         %imgData(:,:,3) = bitshift(bitand(pixelsData,256^1-1),-8*0);
731         %imgData(:,:,2) = bitshift(bitand(pixelsData,256^2-1),-8*1);
732         %imgData(:,:,1) = bitshift(bitand(pixelsData,256^3-1),-8*2);
733
734         % Performance even further improved based on feedback from Jan Simon:
735         pixelsData = reshape(typecast(jImage.getData.getDataStorage, 'uint8'), 4, w, h);
736         imgData = cat(3, ...
737             transpose(reshape(pixelsData(3, :, :), w, h)), ...
738             transpose(reshape(pixelsData(2, :, :), w, h)), ...
739             transpose(reshape(pixelsData(1, :, :), w, h)));
740     end
741 %end  % getInteractivePosition
742
743 %% Return the figure to its pre-undocked state (when relevant)
744 function redockFigureIfRelevant(paramsStruct)
745   if paramsStruct.wasDocked
746       try
747           set(paramsStruct.hFigure,'WindowStyle','docked');
```

```
            %drawnow;
        catch
            % never mind - ignore...
        end
    end
%end  % redockFigureIfRelevant

%% Copy screen-capture to the system clipboard
% Adapted from
% http://www.mathworks.com/matlabcentral/fileexchange/28708...
% -imclipboard/content/imclipboard.m
function imclipboard(imgData)
    % Import necessary Java classes
    import java.awt.Toolkit.*
    import java.awt.image.BufferedImage
    import java.awt.datatransfer.DataFlavor

    % Add the necessary Java class (ImageSelection) to the Java classpath
    if ~exist('ImageSelection', 'class')
        % Obtain the directory of the executable (or of the M-file if not deployed)
        %javaaddpath(fileparts(which(mfilename)), '-end');
        if isdeployed % Stand-alone mode.
            [status, result] = system('path');  %#ok<ASGLU>
            MatLabFilePath = char(regexpi(result, 'Path=(.*?);', 'tokens', 'once'));
        else % MATLAB mode.
            MatLabFilePath = fileparts(mfilename('fullpath'));
        end
        javaaddpath(MatLabFilePath, '-end');
    end

    % Get System Clipboard object (java.awt.Toolkit)
    cb = getDefaultToolkit.getSystemClipboard;  % can't use () in ML6!

    % Get image size
    ht = size(imgData, 1);
    wd = size(imgData, 2);

    % Convert to Blue-Green-Red format
    imgData = imgData(:, :, [3 2 1]);

    % Convert to 3xWxH format
    imgData = permute(imgData, [3, 2, 1]);

    % Append Alpha data (not used)
    imgData = cat(1, imgData, 255*ones(1, wd, ht, 'uint8'));

    % Create image buffer
    imBuffer = BufferedImage(wd, ht, BufferedImage.TYPE_INT_RGB);
    imBuffer.setRGB(0, 0, wd, ht, typecast(imgData(:), 'int32'), 0, wd);

    % Create ImageSelection object
    %    % custom java class
    imSelection = ImageSelection(imBuffer);

    % Set clipboard content to the image
    cb.setContents(imSelection, []);
%end  %imclipboard

%% Is the provided handle a figure?
function flag = isFigure(hObj)
    flag = isa(handle(hObj),'figure') | isa(hObj,'matlab.ui.Figure');
%end  %isFigure

%% Is the provided handle an axes?
function flag = isAxes(hObj)
    flag = isa(handle(hObj),'axes') | isa(hObj,'matlab.graphics.axis.Axes');
%end  %isFigure

%% Is the provided handle an image?
function flag = isImage(hObj)
    flag = isa(handle(hObj),'image') | isa(hObj,'matlab.graphics.primitive.Image');
%end  %isFigure
```

INSTITUTIONEN FÖR
RYMD-, GEO- OCH MILJÖVETENSKAP
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020
www.chalmers.se

CHALMERS