DELIVERABLE

D5.10

EOR-PNP Workshop

Globally the overall oil recovery factors for primary and secondary recovery range from 35% to 45% and a tertiary recovery method that can enhance the recovery factor by 10-30% could contribute to energy supply. The use of nanoparticles in enhanced oil recovery (EOR) processes comprise an emerging and well-promising approach. While surfactants injection into geological sites has been a commonly practiced EOR method, the injection of aqueous nanoparticle suspensions is still at its early stages. In the most active period between mid-1970 and mid-1980, there was a significant gap between what was planned in the EOR laboratory and what was achieved in field application, since the reservoir characterization was elementary with no digital geological modeling capabilities, and there were no concepts such as monitoring systems or intelligent completions. Over the years, irrespective of the inclement times in upstream oil industry, fundamental sciences, EOR physics, numerical modeling, monitoring and controlling capabilities, and EOR chemicals all demonstrated advancements. EOR potential for conventional and unconventional reservoirs is still a challenge, and particularly the development of nanotechnology-based new chemicals for the application of more efficient and cost-effective EOR projects. ***The methodologies and scientific results of EOR-PNP could contribute not only to the development of cost-effective and high-performance EOR methods, but also on the generation of innovative knowledge that might be helpful in other applications, such as the remediation of soils and aquifers contaminated by hydrocarbons.***

Communication

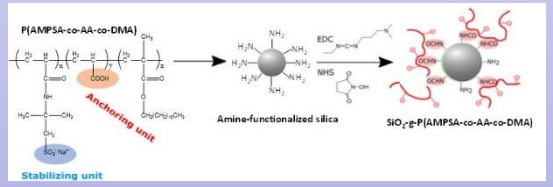
FORTH/ICE-HT, Stadiou str., Platani, 26504 Rio-Patras
Dr. Christos Tsakiroglou
Phone.: 2610 965212, *E-mail*: ctsakir@iceht.forth.gr

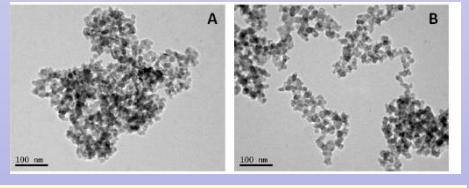
Registration
Ms. Kleanthi Zacharopoulou
Phone: 2610 965278, *E-mail*: kleanthi@iceht.forth.gr

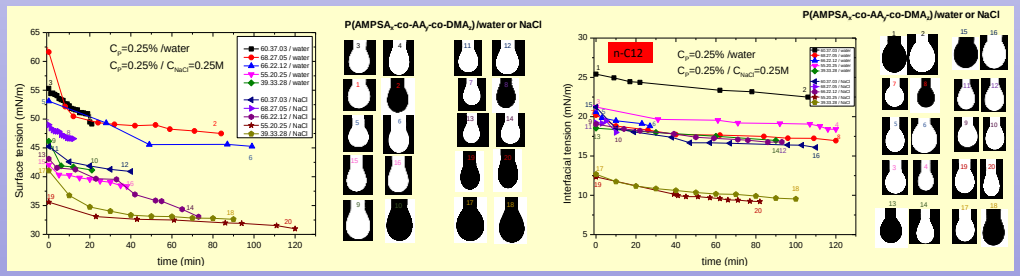
H.F.R.I.
Hellenic Foundation for Research & Innovation

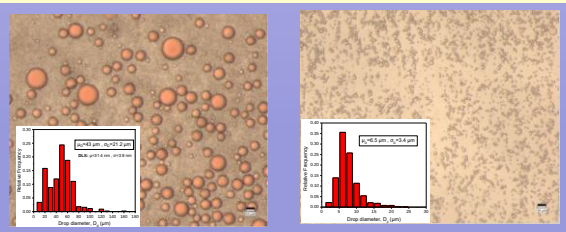
Workshop, 12-13 December 2022
FORTH/ICE-HT, Patras

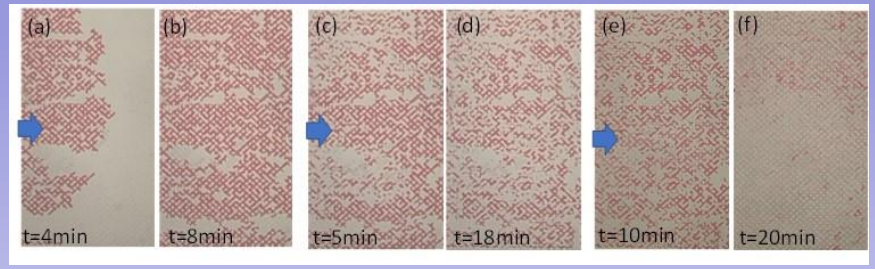
Advances Toward the Transport of Nanoparticles in Porous Media and Applications to Residual Oil Recovery











FOUNDATION FOR RESERACH AND TECHNOLOGY HELLAS – INSTITUTE OF CHEMICAL ENGINEERING SCIENCES (FORTH / ICE-HT)

12-13 December 2022

Auditorium of FORTH/ICE-HT Conference Center

The Workshop is organized within the framework of the Research Project:

«Enhanced oil recovery by polymer-coated nanoparticles (EOR-PNP)»

http://eorpnp.iceht.forth.gr

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment"

Workshop Program

Monday, 12 December 2022

14:30-15:00: Registration

Workshop I.

15:00-15:20. Christos Tsakiroglou, Research Director, FORTH/ICE-HT, ***«EOR-PNP – Introduction - Perspectives»***

15:20-15:50. Georgios Bokias, Professor, Department of Chemistry, University of Patras, ***«Synthesis and characterization of polymer-coated nanoparticles»***

15:50-16:10. Christina Ntente, Graduate Student, FORTH/ICE-HT & University of Patras, ***«Nano-colloid interfacial properties, and emulsion stabilization»***

16:10-16:30. Anastasia Strekla, Graduate Student, FORTH/ICE-HT & University of Patras, ***«Visualization EOR studies on glass-etched pore networks»***

16:30-17:00. Coffee break

17:00-18:30. Invited Speaker: Marios Ioannidis, Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Nanoparticle interactions with interfaces in porous media: a multiscale perspective»***

Tuesday, 13 December 2022

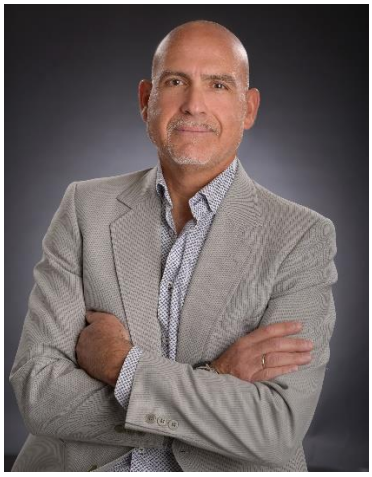
Workshop II.

10:00-10:30. Nadia Bali, Post-Doctoral Fellow, FORTH/ICE-HT, ***«Computational Fluid Dynamic models inside 3D digitally represented rocks and soils»***

10:30-11:30. Invited Speaker: Jeff Gostick, Associate Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Network Modeling and Quantitative Analysis of Volumetric Images: Introduction to OpenPNM and PoreSpy - Part I»***

11:30-12:00. ***Coffee break***

12:00-13:00. Invited Speaker: Jeff Gostick, Associate Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Network Modeling and Quantitative Analysis of Volumetric Images: Introduction to OpenPNM and PoreSpy - Part II»***



Marios Ioannidis is a Professor of Chemical Engineering at the University of Waterloo, Canada, where he currently serves as Chair of the Department of Chemical Engineering. Marios first studied chemical engineering at the University of Patras, obtaining the Diploma in Chemical Engineering in 1988, and continued his studies at the University of Waterloo, obtaining a PhD degree in Chemical Engineering in 1993. Marios and his students research fundamental aspects of multiphase flow and transport of solutes and nanoparticles in porous materials of different kinds (fibrous or granular, man-made or natural, etc.). The goal of their research is to enable engineering applications ranging from subsurface remediation and petroleum production to environmental sampling, fuel cell performance optimization and oil spill response. Professor Ioannidis has held numerous administrative positions within the University of Waterloo, including as Director of the Nanotechnology Engineering Program, and is a Fellow of Engineers Canada.

Jeff Gostick, PEng, PhD
Department Chemical Engineering
University of Waterloo

Title: Pore Network Modeling and Quantitative Analysis of Volumetric Images: Introduction to OpenPNM and PoreSpy

Abstract/Description: In this workshop we will cover the use of volumetric or 3D images of porous materials, typically obtained from X-ray tomograph, to study structural and transport properties of computationally. The first half of this workshop will focus on image analysis using tools such as ImageJ and PoreSpy, the latter of which is a Python package we develop and distribute freely. Topics in this segment will span the entire workflow from discussing typical data formats, processing greyscale images, applying simple filters like dilation/erosion, extracting higher level information such as pore size distributions, to finally performing simulations on images to obtain basic transport properties. The second segment will shift to looking at pore network modeling using OpenPNM, a Python package that is also developed by our group and distributed freely. Topics covered here will include the fundamentals of the pore network framework including network representation and determination of conductance values, generating and importing networks, performing multiphase simulations to estimate complex transport properties such as relative permeability curves, and finally ending with a simulation of dispersion in porous media. The program will be interactive, so attendees should bring a laptop and have ImageJ and the Anaconda distribution of Python installed, which includes precompiled and optimized versions of the main numerical packages. Familiarity the Python would be helpful, but attendees with knowledge of other languages such as Matlab or C/C++ should be able to grasp all the concepts with ease.

Biography: Jeff Gostick is the Azzam-Dullien Professor in Chemical Engineering at the University of Waterloo where he runs the Porous Materials Engineering & Analysis Lab. His research is centered around understanding the structure-performance relationship in porous electrodes used in hydrogen fuel cell, redox flow systems, zinc-air cells, Li-ion batteries, and super-capacitors. His group uses a combination experimental characterization, novel production methods, and advanced custom computational tools. He is the lead developer of the open-source pore network modeling project OpenPNM (openpnm.org), as well as PoreSpy, a tool for porous media image analysis (porespy.org). Prof Gostick is a licensed professional engineer, has published over 90 journal articles, and was recently named an Emerging Leader by the Canadian Society for Chemical Engineering.

H.F.R.I.
Hellenic Foundation for Research & Innovation

FORTH/ICE-HT
www.iceht.forth.gr

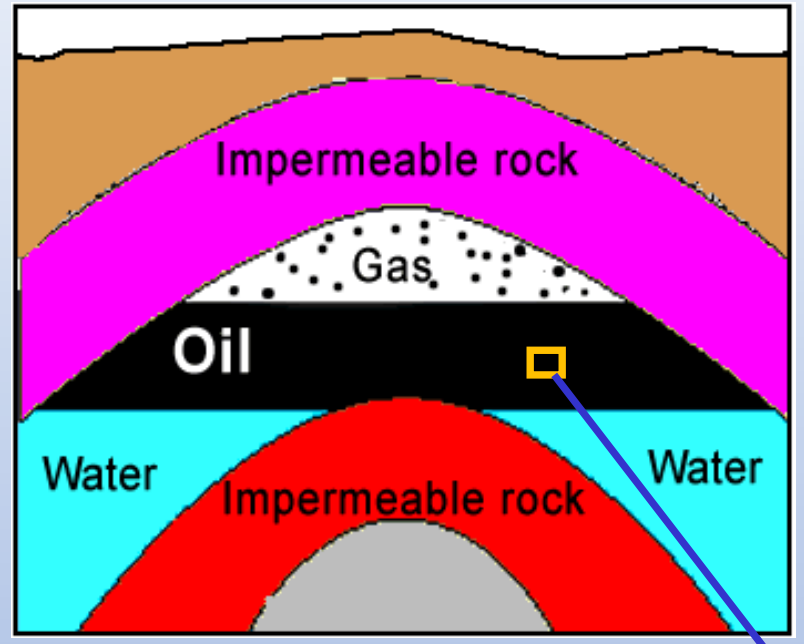
Advances Toward the Transport of Nanoparticles in Porous Media and Applications to Residual Oil Recovery

The Workshop is organized within the framework of the Research Project:

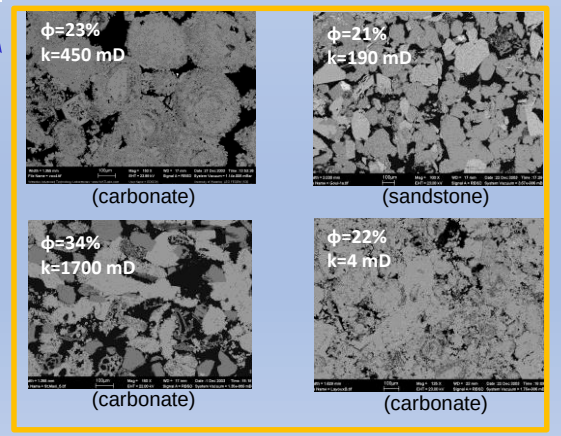
«**Enhanced oil recovery by polymer-coated nanoparticles (EOR-PNP)**»
(Duration: 1/1/2020-31/12/2023)
http://eorpnp.iceht.forth.gr

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment

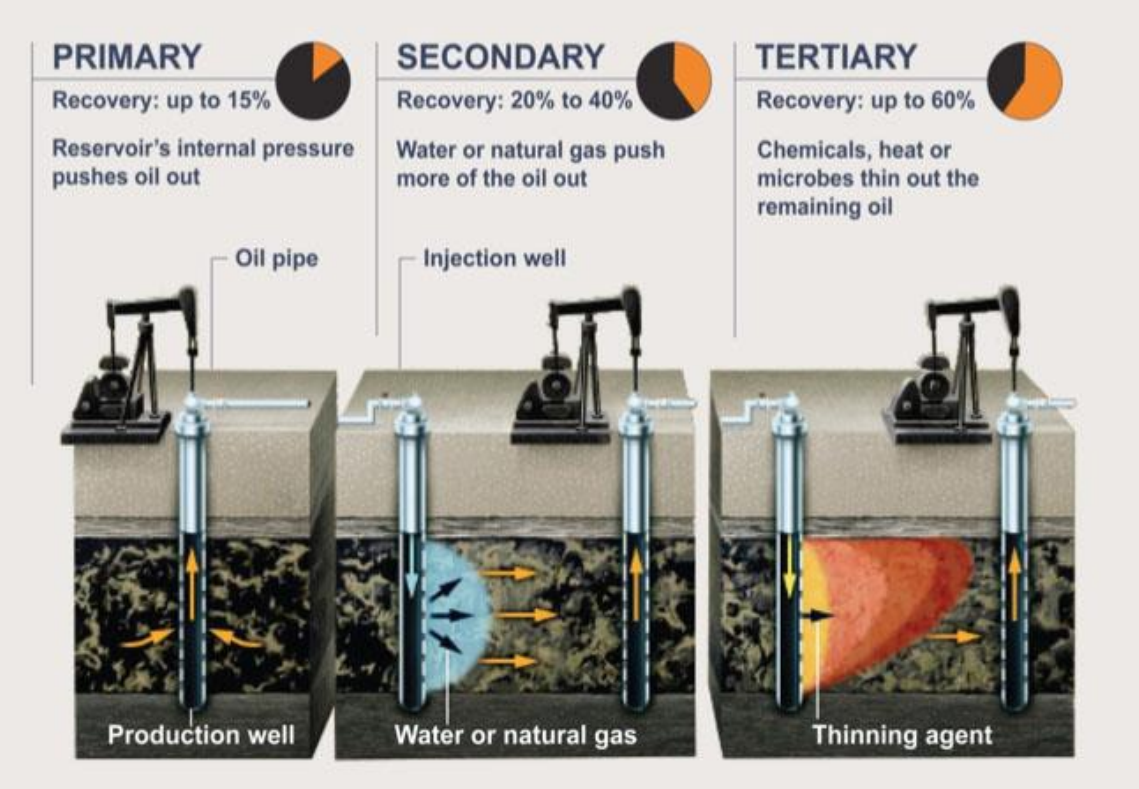
OIL AND GAS RESERVOIRS

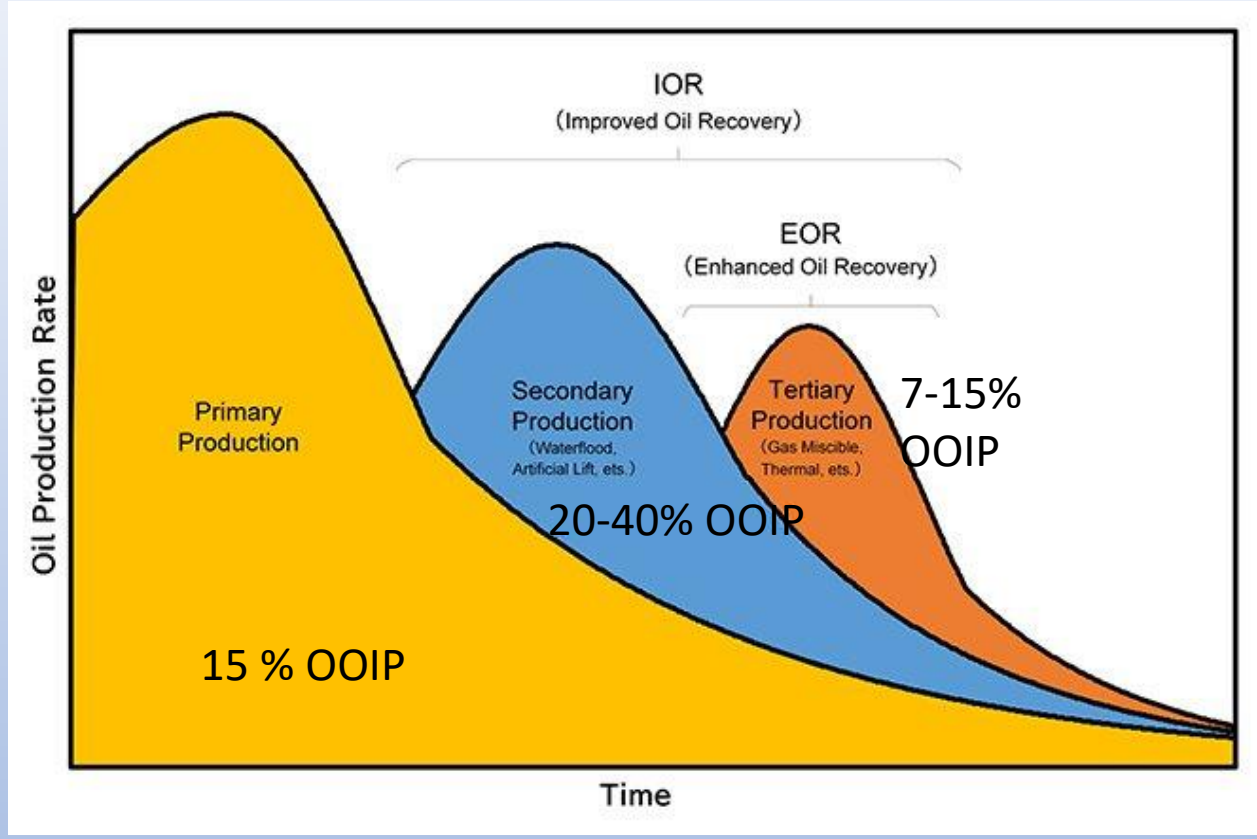


- A good reservoir is:
  - **Porous:** holes of size < 1 mm
  - **Permeable:** the holes are interconnected …..
  - …so that the fluids are able to be **produced** (removed from the reservoirs)



ENHANCED OIL RECOVERY (EOR)

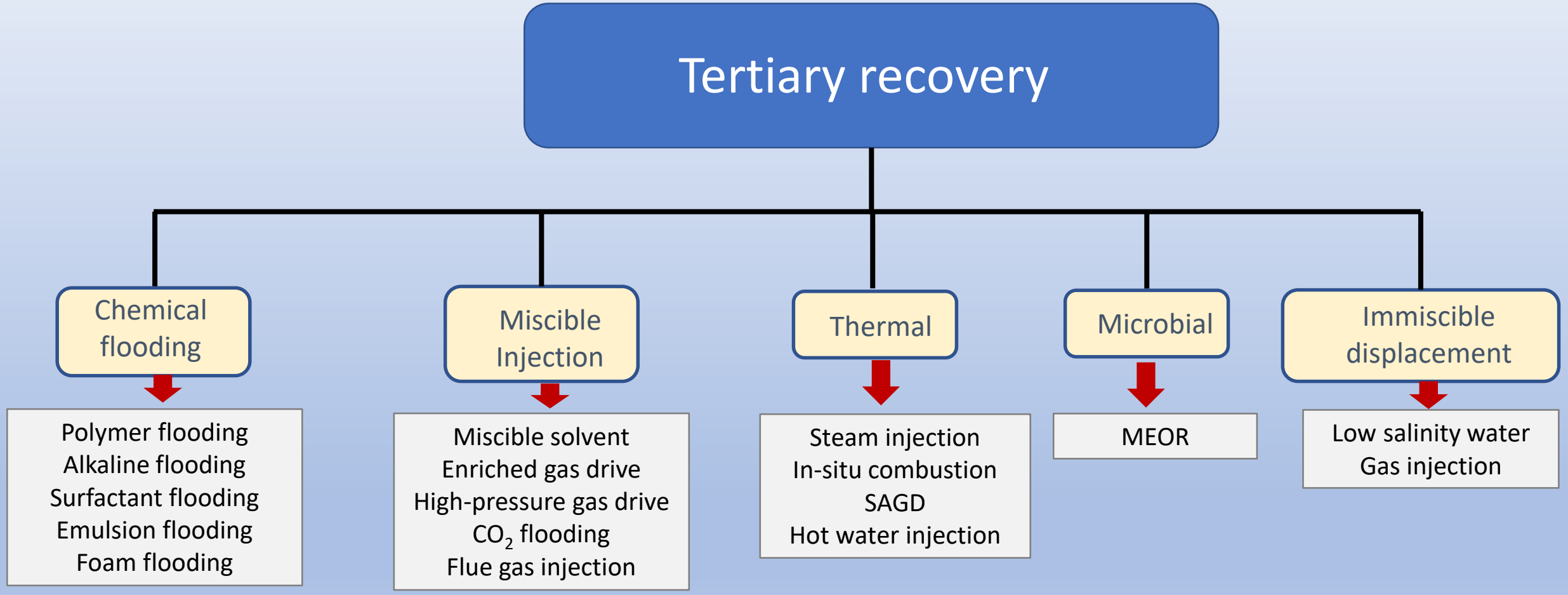




- Depletion of oil resources
- Increasing global energy demand
- Current high oil prices
- Increasing maturity of major oil fields

EOR can significantly impact oil production, as increase in the recovery rate of oil by even a small margin could bring significant revenues without developing unconventional resources

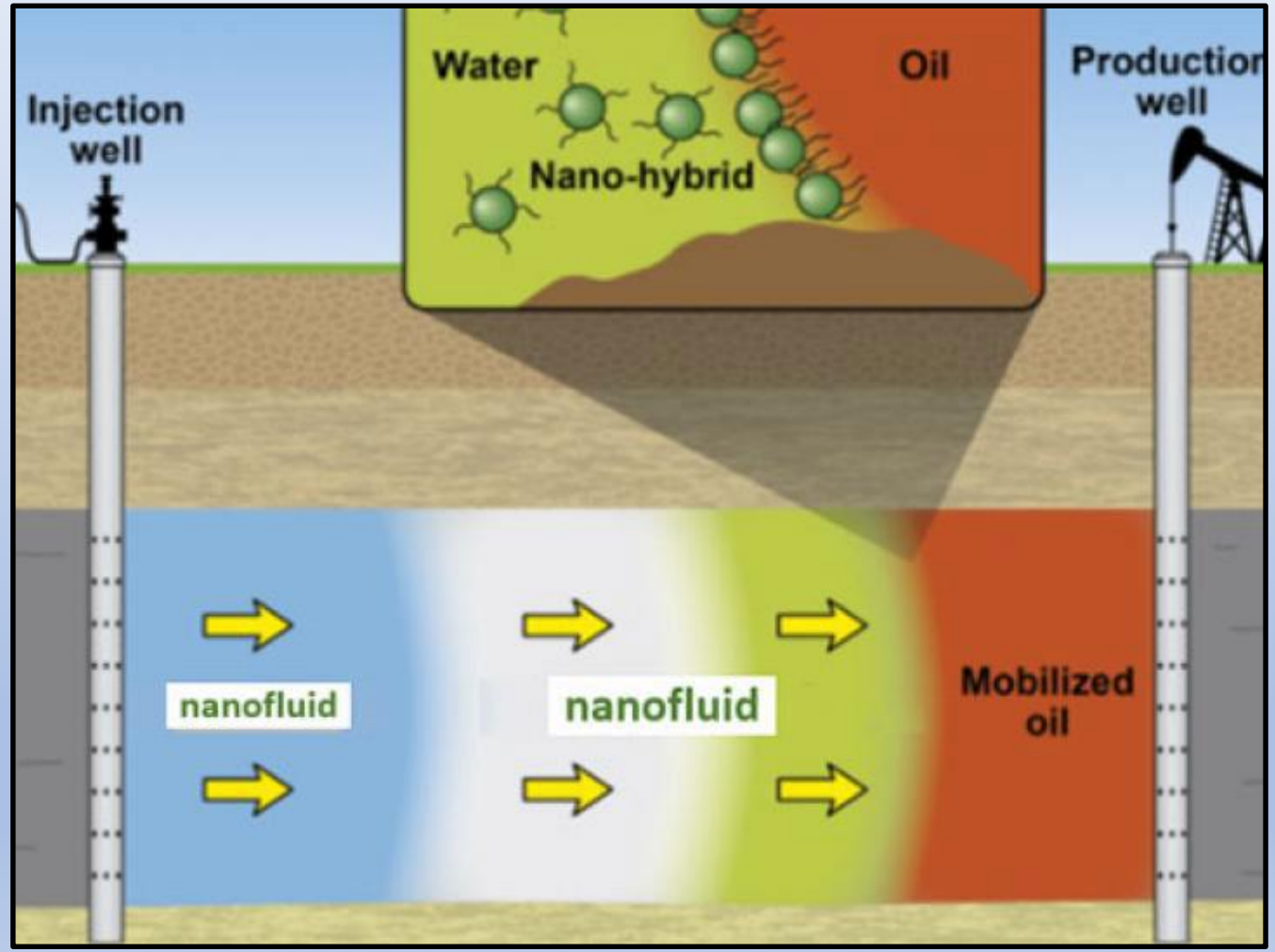
EOR METHODS


Tertiary recovery
Chemical flooding
Polymer flooding
Alkaline flooding
Surfactant flooding
Emulsion flooding
Foam flooding
Miscible Injection
Miscible solvent
Enriched gas drive
High-pressure gas drive
CO_2 flooding
Flue gas injection
Thermal
Steam injection
In-situ combustion
SAGD
Hot water injection
Microbial
MEOR
Immiscible displacement
Low salinity water
Gas injection

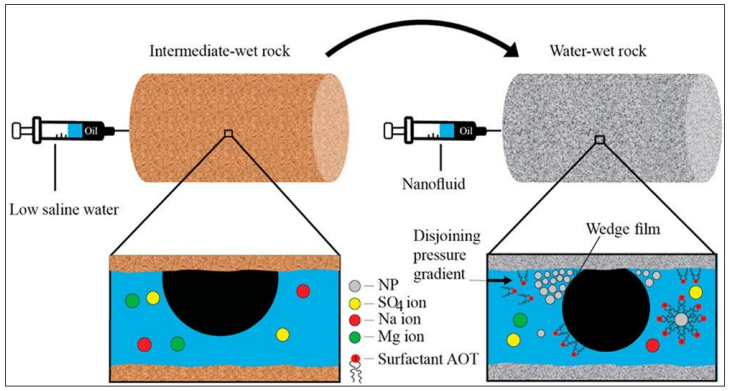
EOR-PNP - INTRODUCTION - PERSPECTIVES

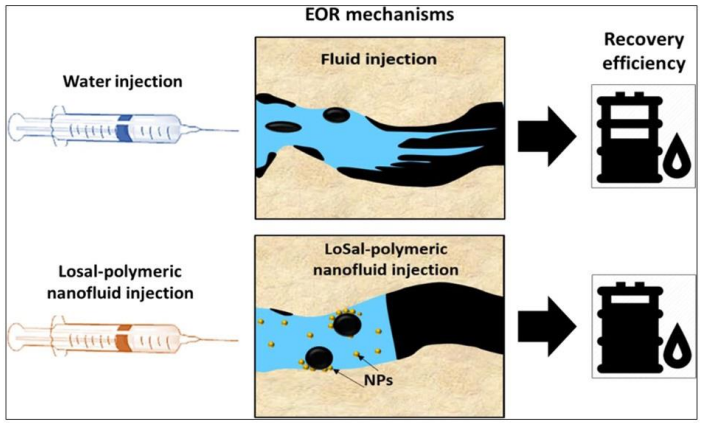
GOAL OF THE PROJECT

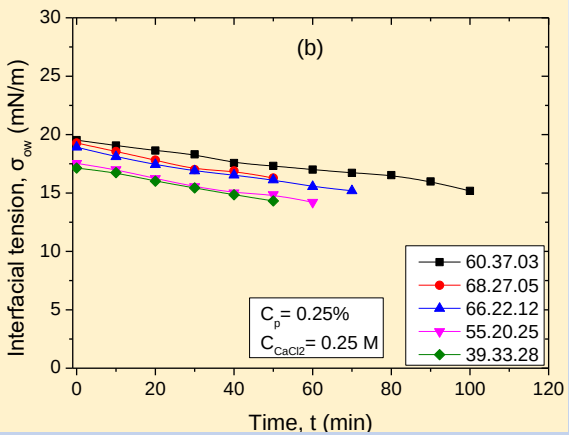
- Nanoparticle stabilization in aqueous solutions with polymer coatings (Synthesis /characterization)
- Stabilization of emulsions and foams
- Enhancement of the oil recovery by injecting nano-colloid solutions (nanofluid) in porous media models and cores
- Simulation of the displacement process in 3-D reconstructed pore networks
- Selection of the most efficient fluid systems

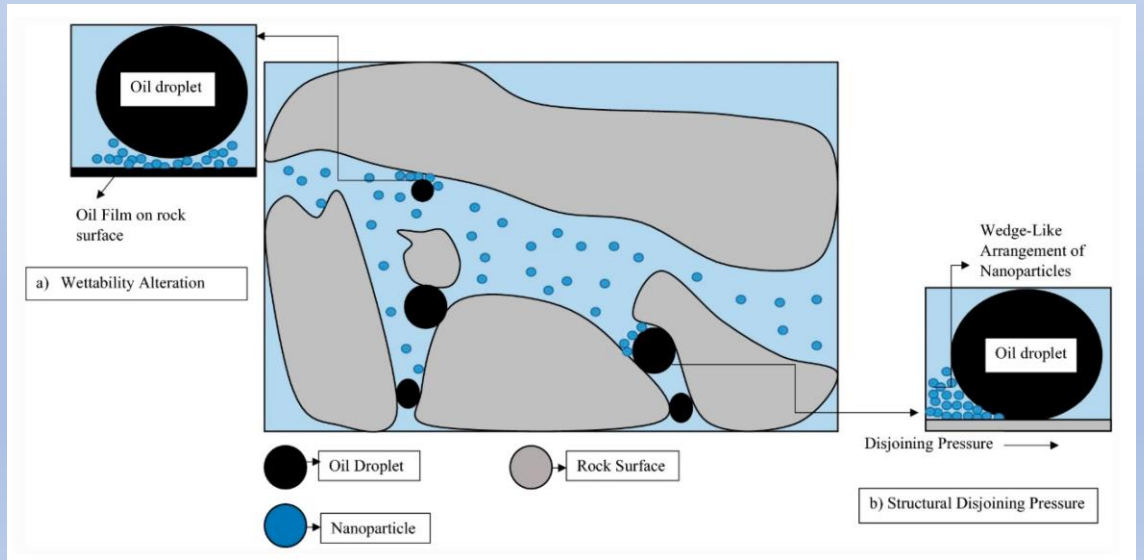


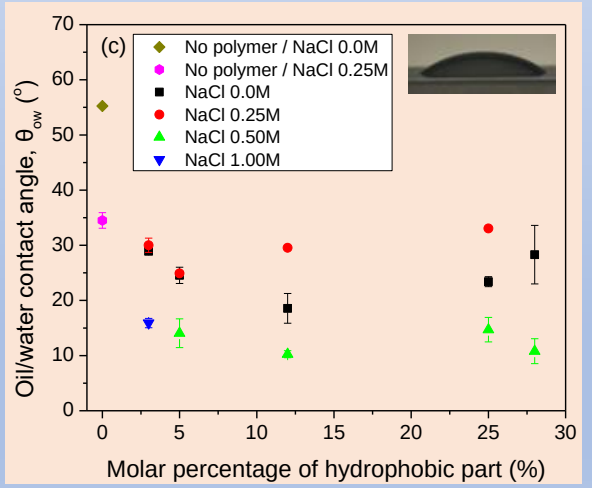
NANOFLUIDS AS EOR AGENTS





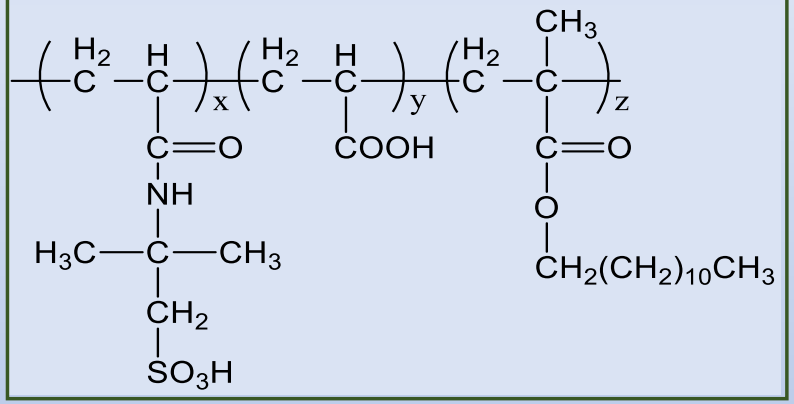


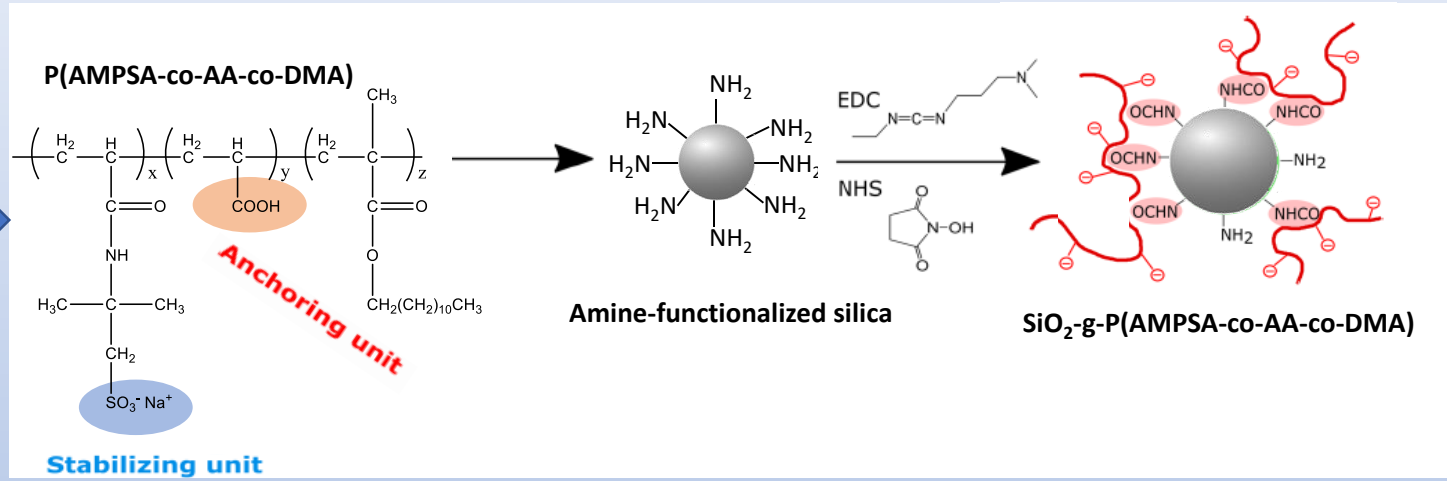




SYNTHESIS OF POLYMER-COATED NANOPARTICLES (PNP)

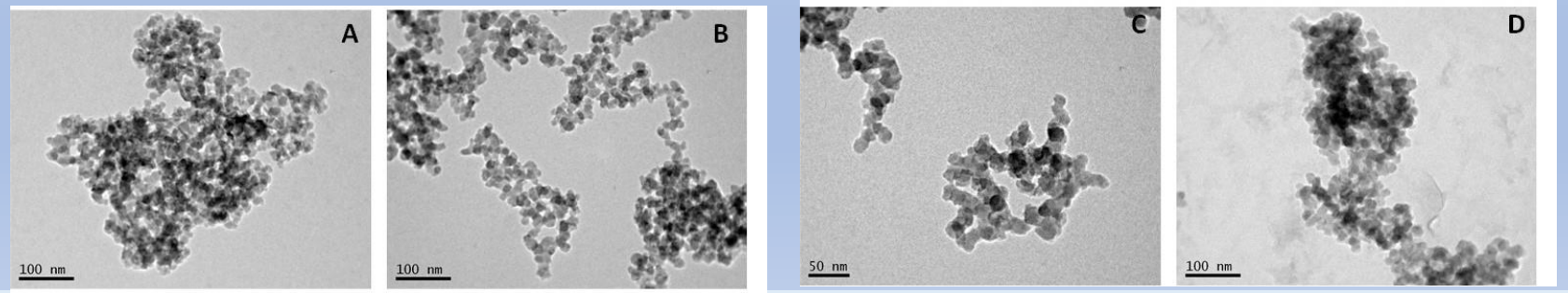
P(AMPSA$_x$-co-AA$_y$-co-DMA$_z$)





2-acrylamido-2-methyl-1-propanesulfonic acid (AMPSA) & acrylic acid (AA) & dodecyl methacrylate (DMA)

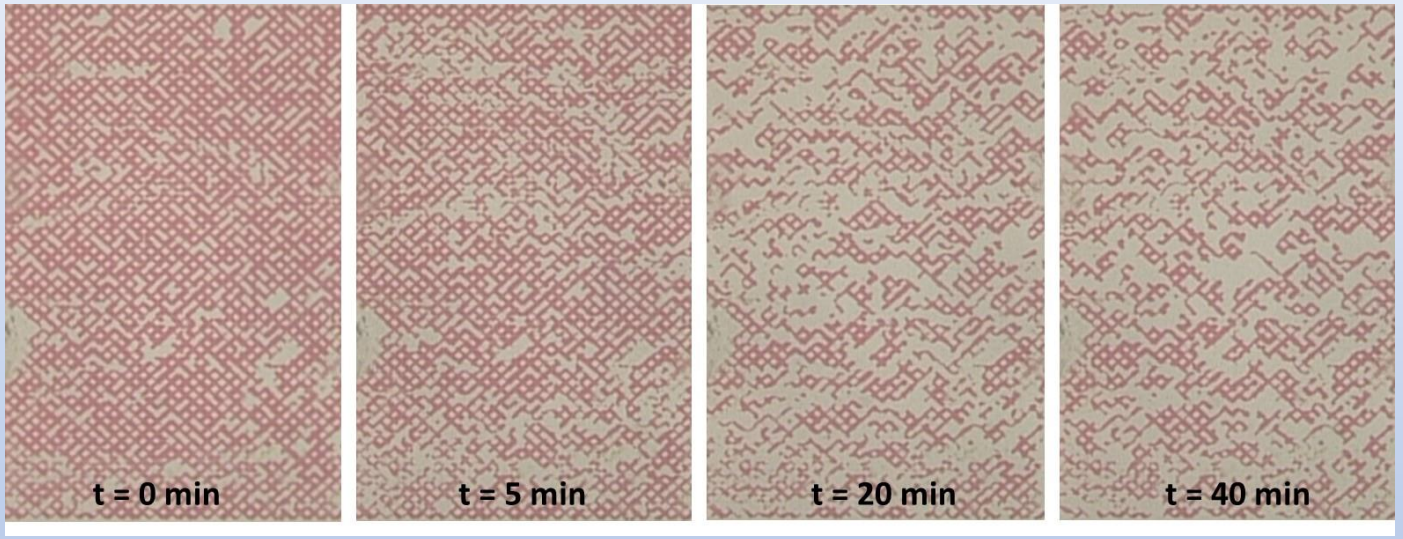
TEM images of (A) 2_SiO$_2$-PAMPSA; (B) 3_SiO$_2$-PAMPSA; (C) 2_SiO$_2$-g-P(AMPSA60-co-AA20-co-DMA20); (D) 3_SiO$_2$-g-P(AMPSA67-co-AA22-co-DMA11) NPs.



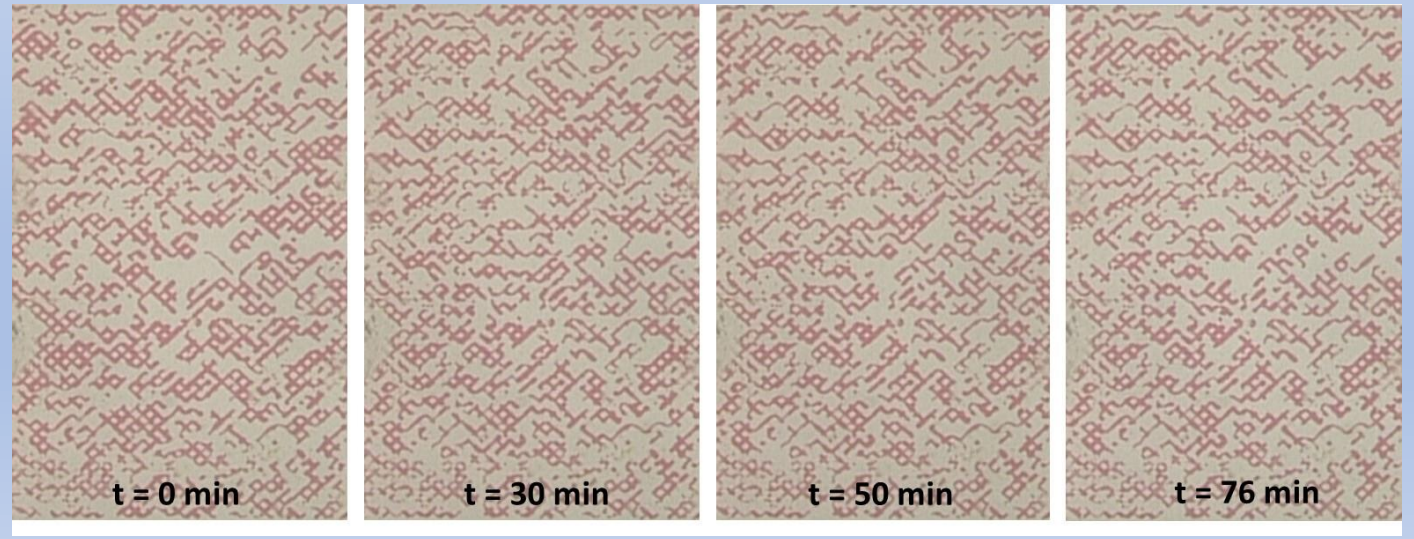
Main barrier:

The small quantities synthesized are insufficient to extend the studies to sand columns and rock plugs

OIL DISPLACEMENT BY PNP SUSPENSIONS

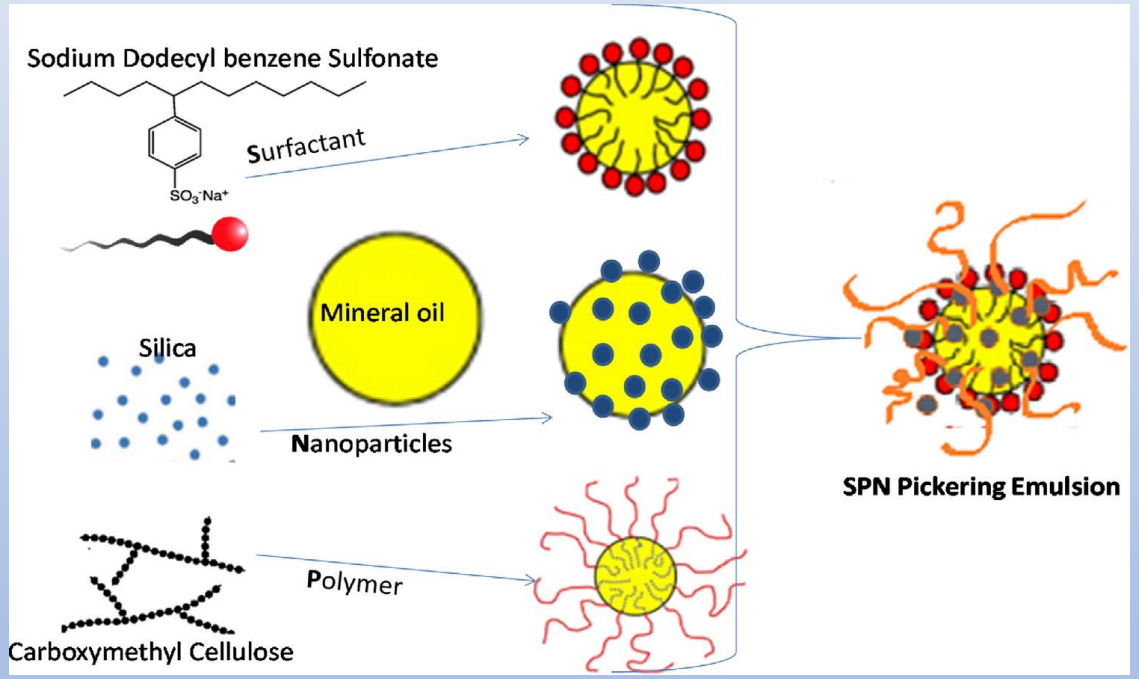


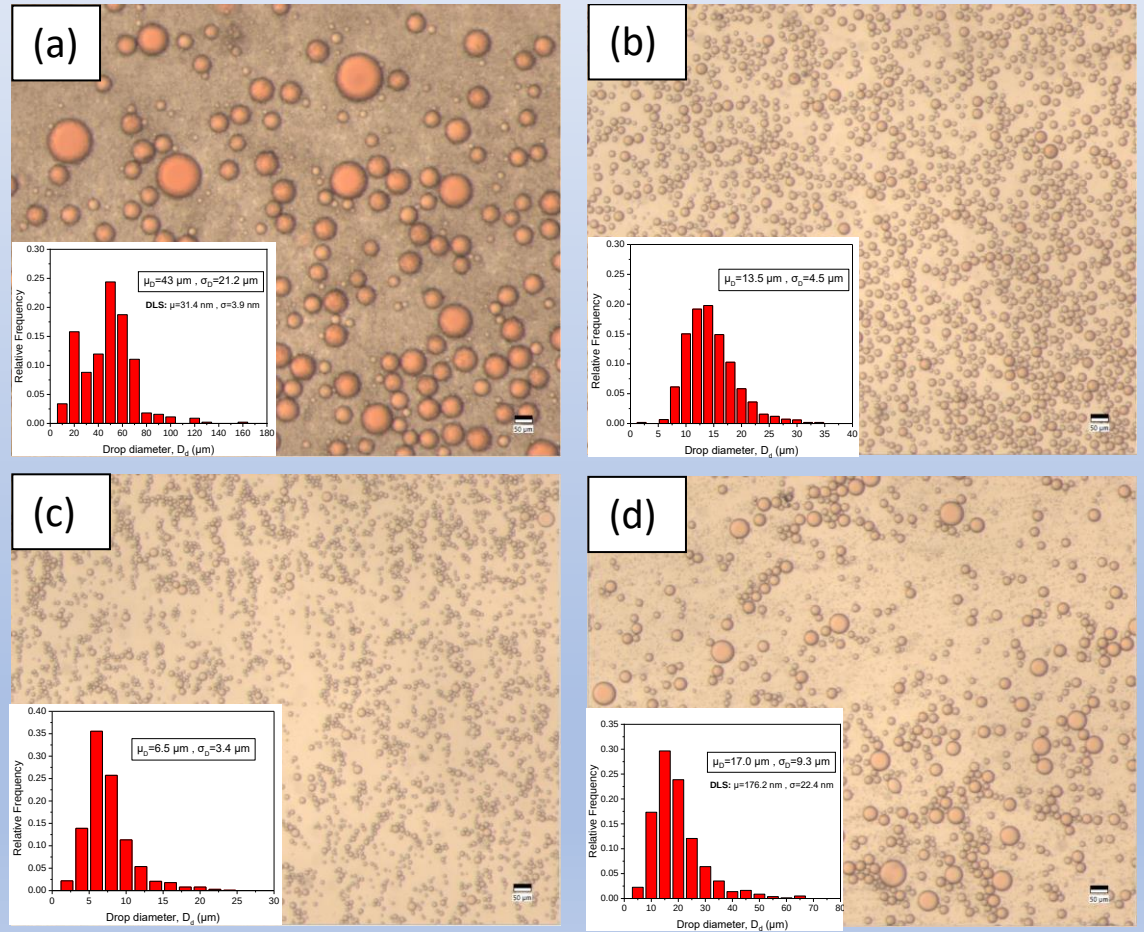
Displacement of paraffin oil by brine (NaCl 1M)
S_{or}=0.457



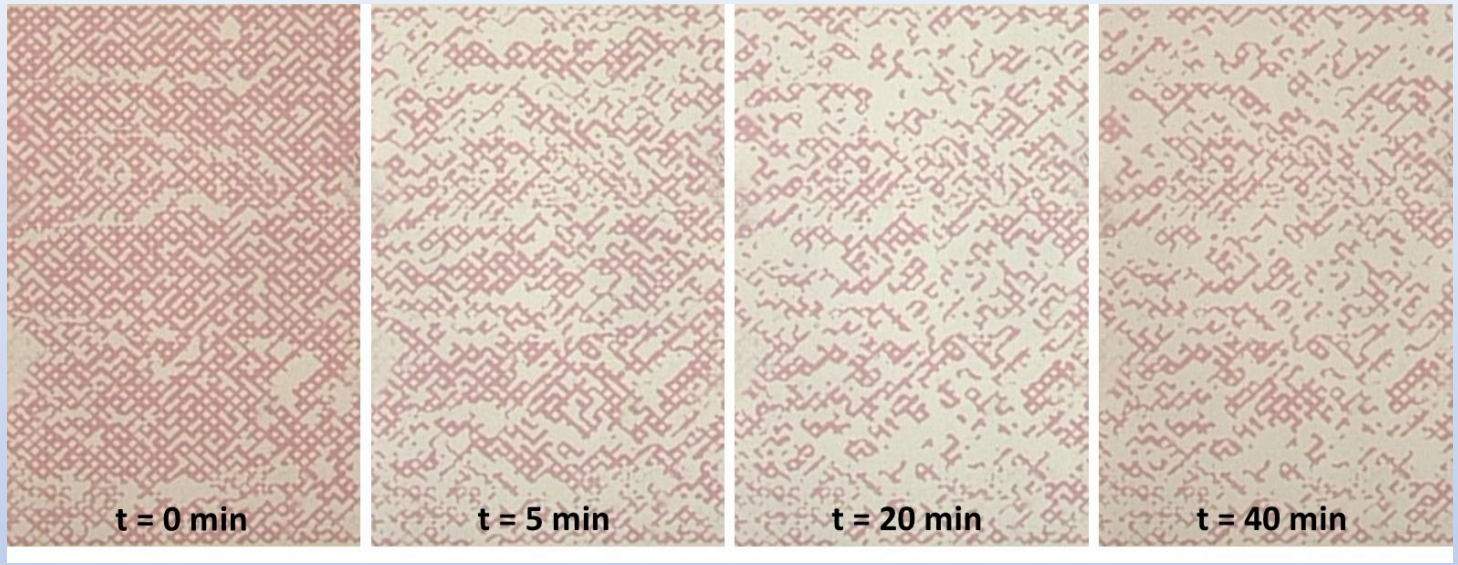
Displacement of residual paraffin oil by AMPSA 75-25 0.25% - NaCl 1M
S_{or}=0.403

PREPARATION OF PNP-PICKERING EMULSIONS

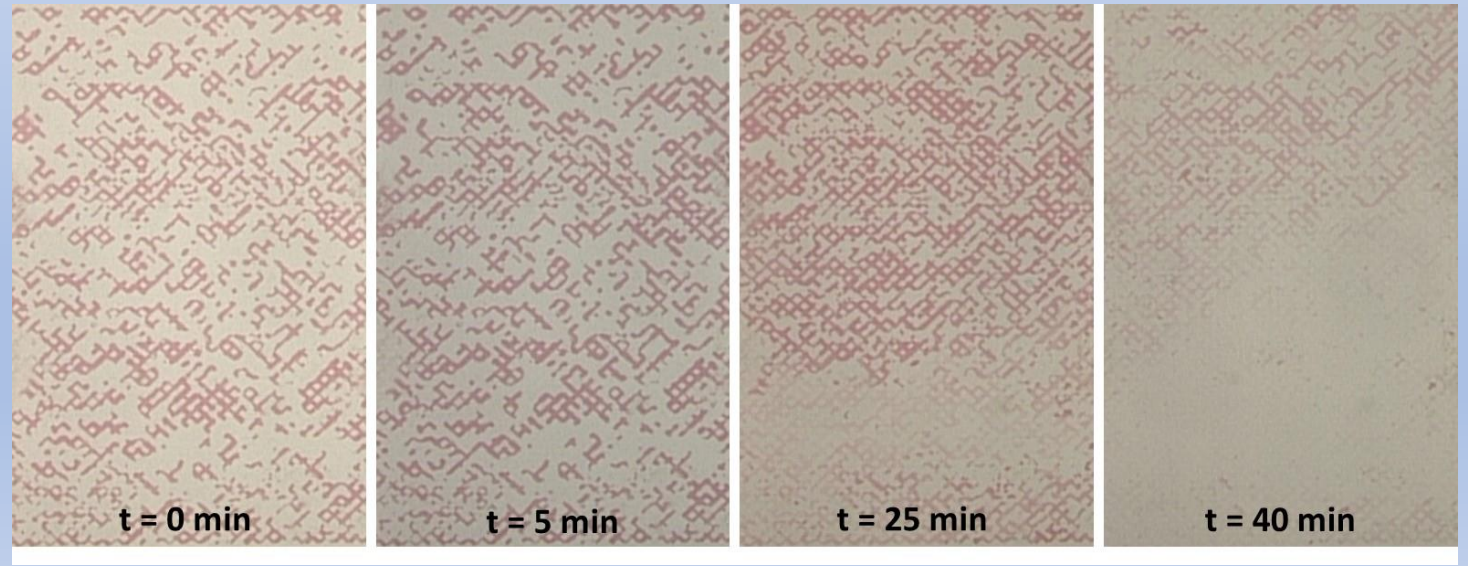




OIL DISPLACEMENT BY PNP-STABILIZED EMULSIONS



Displacement of paraffin oil by brine (NaCl 1M)
S_{or}=0.447

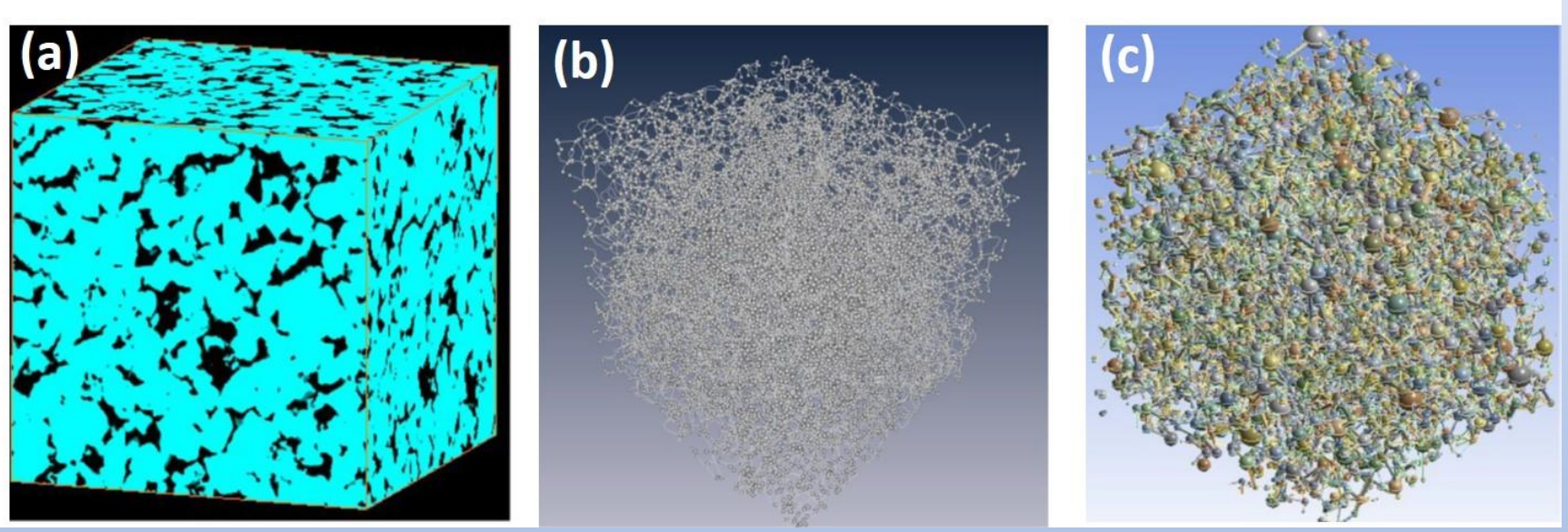


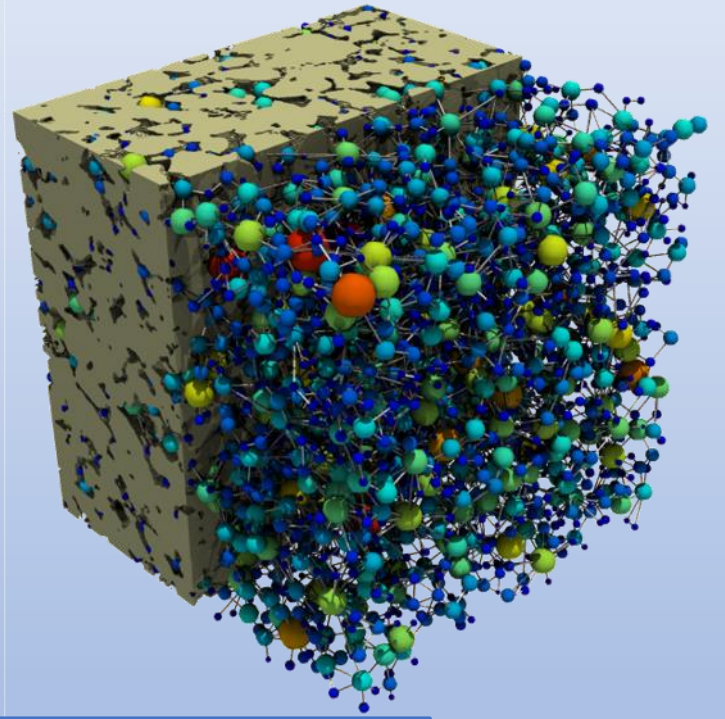
Displacement of residual paraffin oil by (SiO_2-AMPSA-DMA-0.25% + NaCl 1M) Pickering emulsion
S_{or}=0.157

H.F.R.I.
Hellenic Foundation for Research & Innovation

FORTH/ICE-HT
www.iceht.forth.gr

DIGITAL ROCK PHYSICS

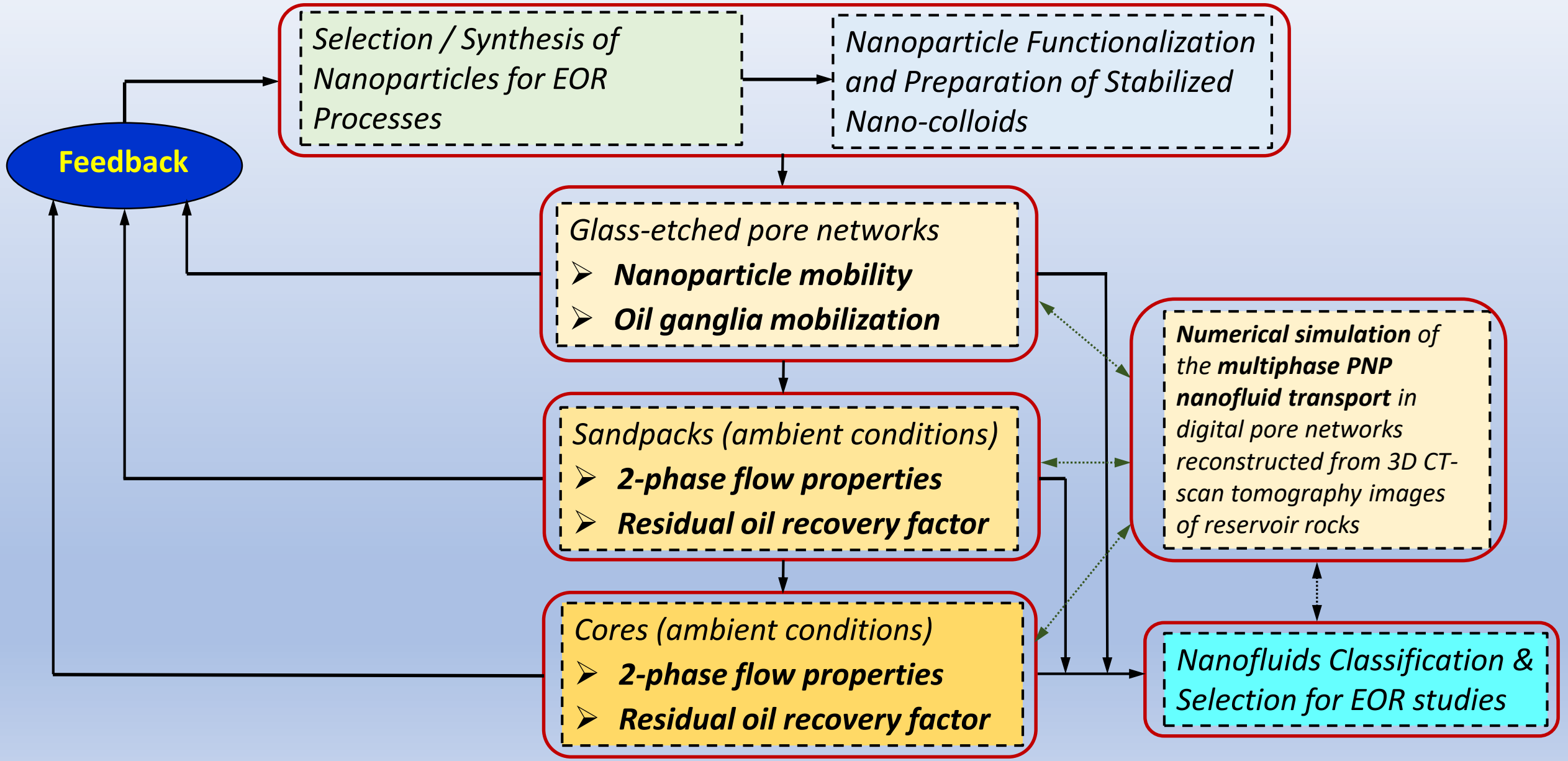




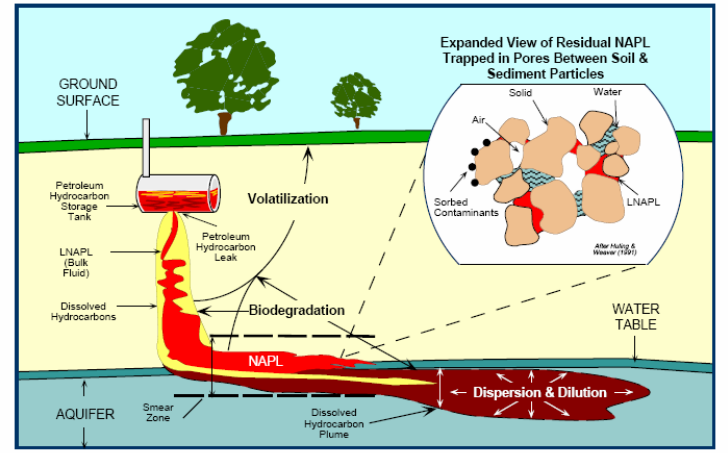
OpenPNM and PoreSpy software:

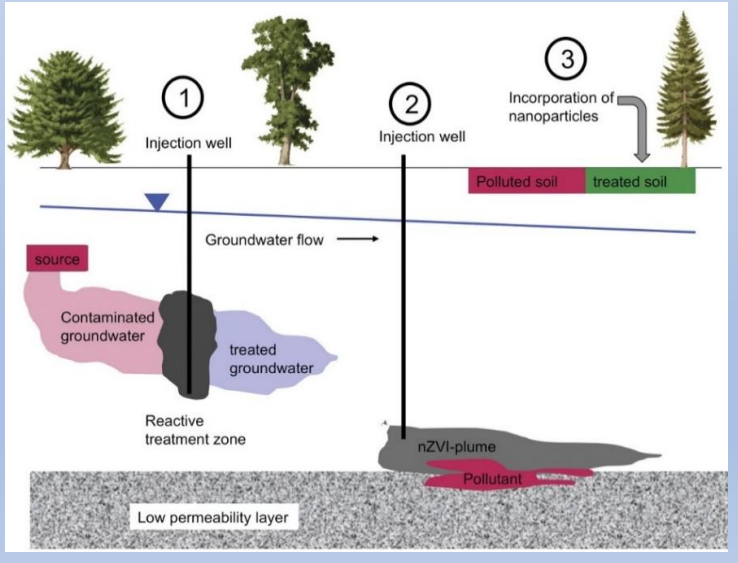
Tomorrow morning, you can bring your Laptop for training course on the use of software

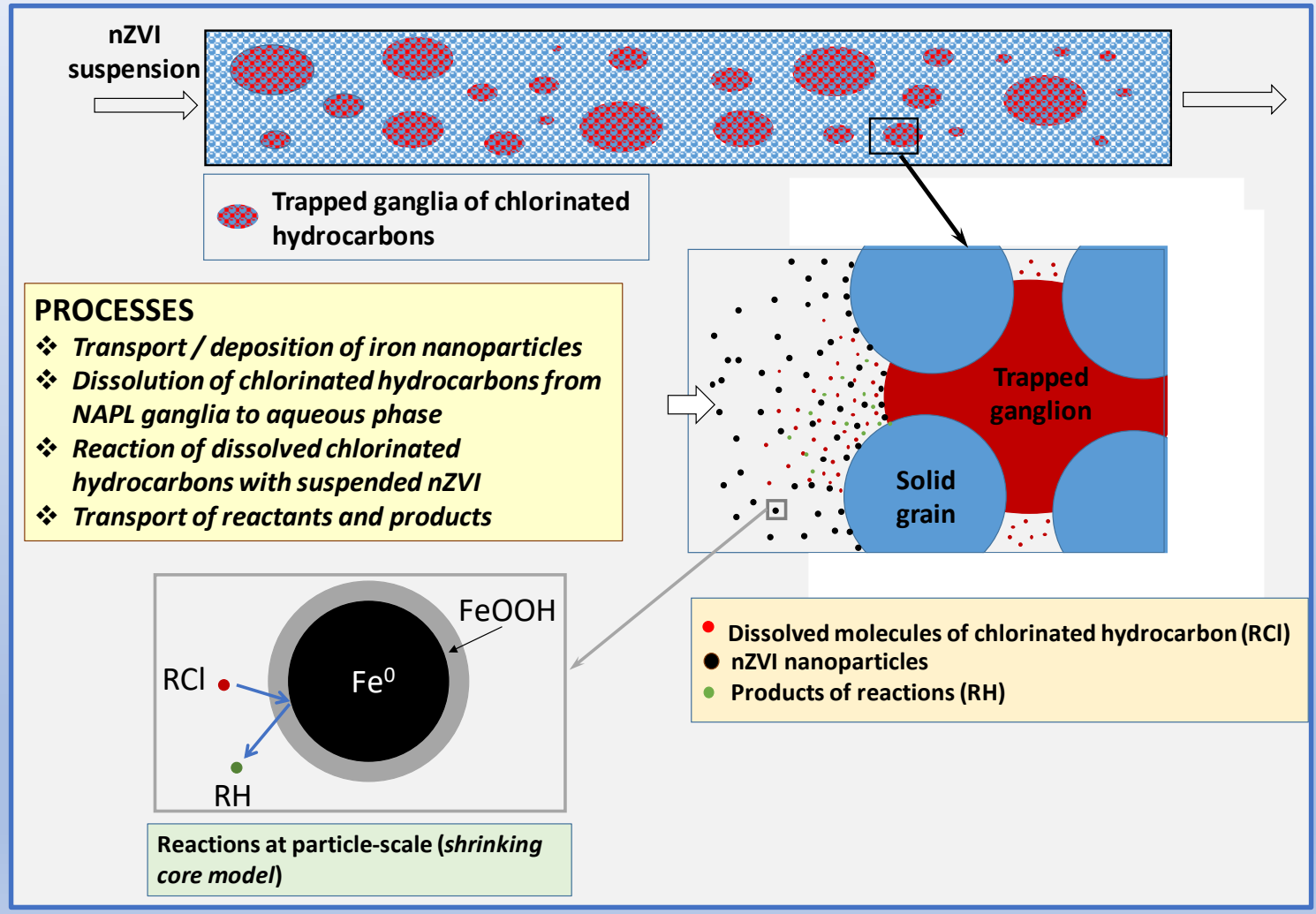
METHODOLOGY



ENVIRONMENTAL APPLICATIONS: NANOREMEDIATION



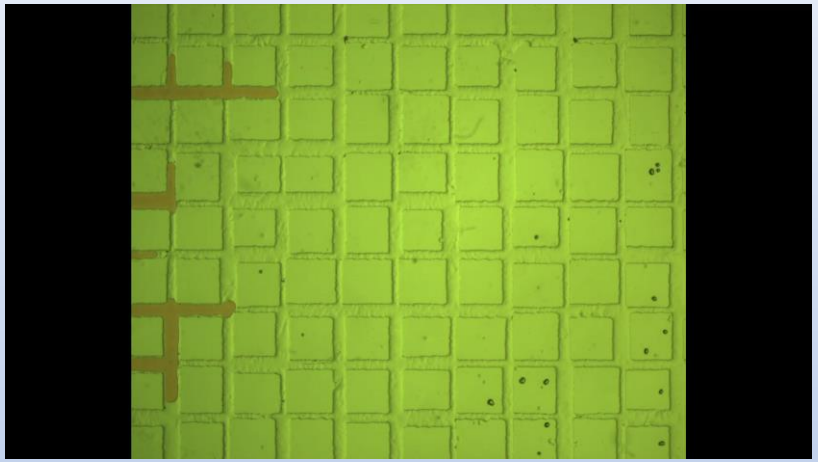




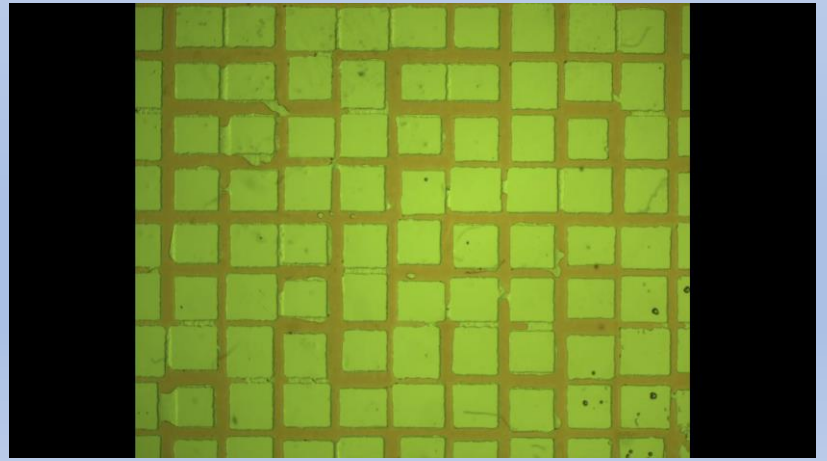
H.F.R.I.
Hellenic Foundation for Research & Innovation

FORTH/ICE-HT
www.iceht.forth.gr

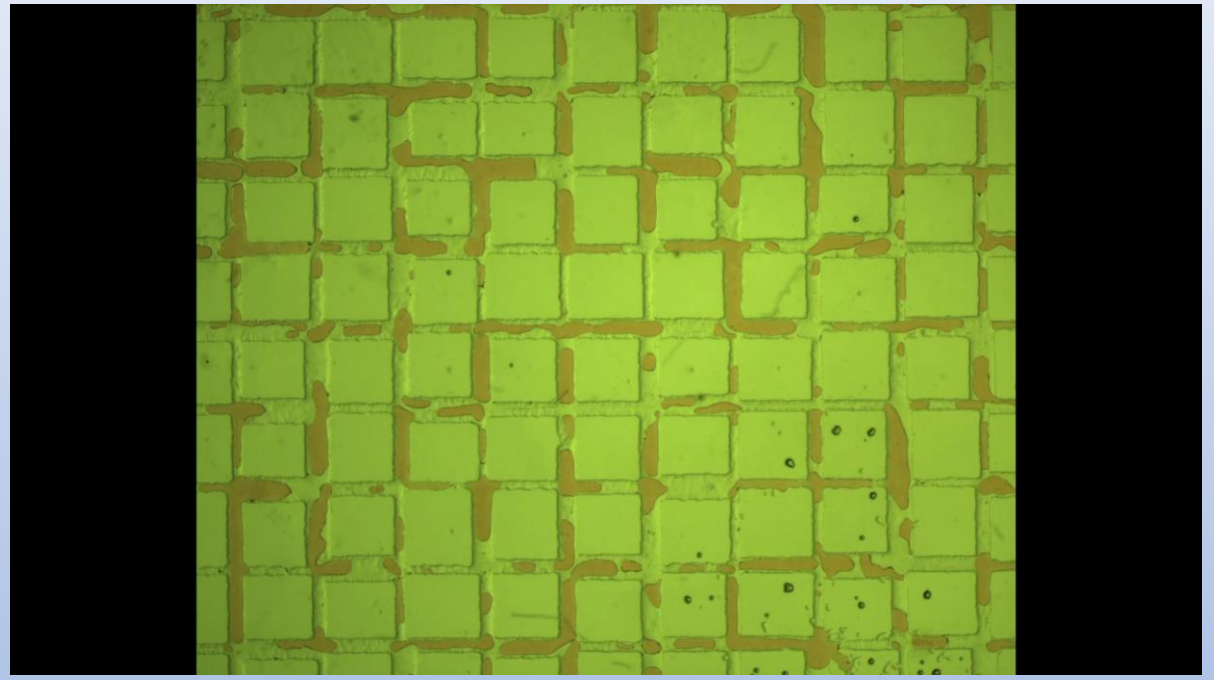
Drainage



Imbibition (secondary oil recovery)



Imbibition (tertiary oil recovery)



The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment (Project title: “Enhanced Oil Recovery by Polymer-coated Nano Particles - EOR-PNP», code: HFRI-FM17-361)

H.F.R.I.
Hellenic Foundation for Research & Innovation

FORTH/ICE-HT
www.iceht.forth.gr

WORKSHOP AGENDA

Monday, 12 December 2022

14:30-15:00: Registration

Workshop I.

15:00-15:20. Christos Tsakiroglou, Research Director, FORTH/ICE-HT, ***«EOR-PNP – Introduction - Perspectives»***

15:20-15:50. Georgios Bokias, Professor, Department of Chemistry, University of Patras, ***«Synthesis and characterization of polymer-coated nanoparticles»***

15:50-16:10. Christina Ntente, Graduate Student, FORTH/ICE-HT & University of Patras, ***«Nano-colloid interfacial properties, and emulsion stabilization»***

16:10-16:30. Anastasia Strekla, Graduate Student, FORTH/ICE-HT & University of Patras, ***«Visualization EOR studies on glass-etched pore networks»***

16:30-17:00. Coffee break

17:00-18:30. Invited Speaker: Marios Ioannidis, Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Nanoparticle interactions with interfaces in porous media: a multiscale perspective»***

Tuesday, 13 December 2022

Workshop II.

10:00-10:30. Nadia Bali, Post-Doctoral Fellow, FORTH/ICE-HT, ***«Computational Fluid Dynamics models inside 3D digitally represented rocks and soils»***

10:30-11:30. Invited Speaker: Jeff Gostick, Associate Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Network Modeling and Quantitative Analysis of Volumetric Images: Introduction to OpenPNM and PoreSpy - Part I»***

11:30-12:00. ***Coffee break***

12:00-13:00. Invited Speaker: Jeff Gostick, Associate Professor, Department of Chemical Engineering, University of Waterloo, Canada, ***«Network Modeling and Quantitative Analysis of Volumetric Images: Introduction to OpenPNM and PoreSpy - Part II»***

Workshop, 12-13 December 2022
FORTH/ICE-HT, Patras

Advances toward the transport of nanoparticles in porous media and residual oil recovery applications

Synthesis and characterization of polymer-coated nanoparticles

Georgios Bokias, Professor
Zacharoula Iatridi, Post-doctoral Fellow

Patras, 12 December 2022

Advanced Polymers & Hybrid Nanomaterials Research Laboratory

http://www.aphnrl.chem.upatras.gr/

Department of Chemistry, University of Patras, Greece
http://www.chem.upatras.gr/en

Patras, 12 December 2022

Advanced Polymers & Hybrid Nanomaterials Research Laboratory!

Department of Chemistry,
University of Patras, Rio, Greece
https://www.upatras.gr/en

Foundation for Research and Technology Hellas, Institute of Chemical Engineering Science (FORTH/ICE-HT), Patras, Greece
http://www.iceht.forth.gr/

Advanced Polymers & Hybrid Nanomaterials Research Laboratory

Faculty Members

- Joannis Kallitsis, Professor

- Georgios Bokias, Professor

- Chrysovalanto Deimede, Associate Professor

- Aikaterini Andreopoulou, Assistant Professor

Postdoctoral Fellows

- Dr. Charalampos Anastassopoulos

- Dr. Zacharoula Iatridi

- Dr. Georgia Lainioti

PhD Candidates

- Mrs. Dionisia Druvari, MSc

- Mr. Konstantinos Andrikopoulos, MSc

- Mrs. Ioanna Tzoumani, MSc

- Mrs. Efterpi Avdeliodi, MSc

Master students

- Mrs. Paraskevi Loukopoulou

- Mrs. Gjoshi Sara

- Mrs. Koutsougera Maria-Filomeni

- Mrs. Roufi Konstantina

- Mrs. Tsolka Eftychia

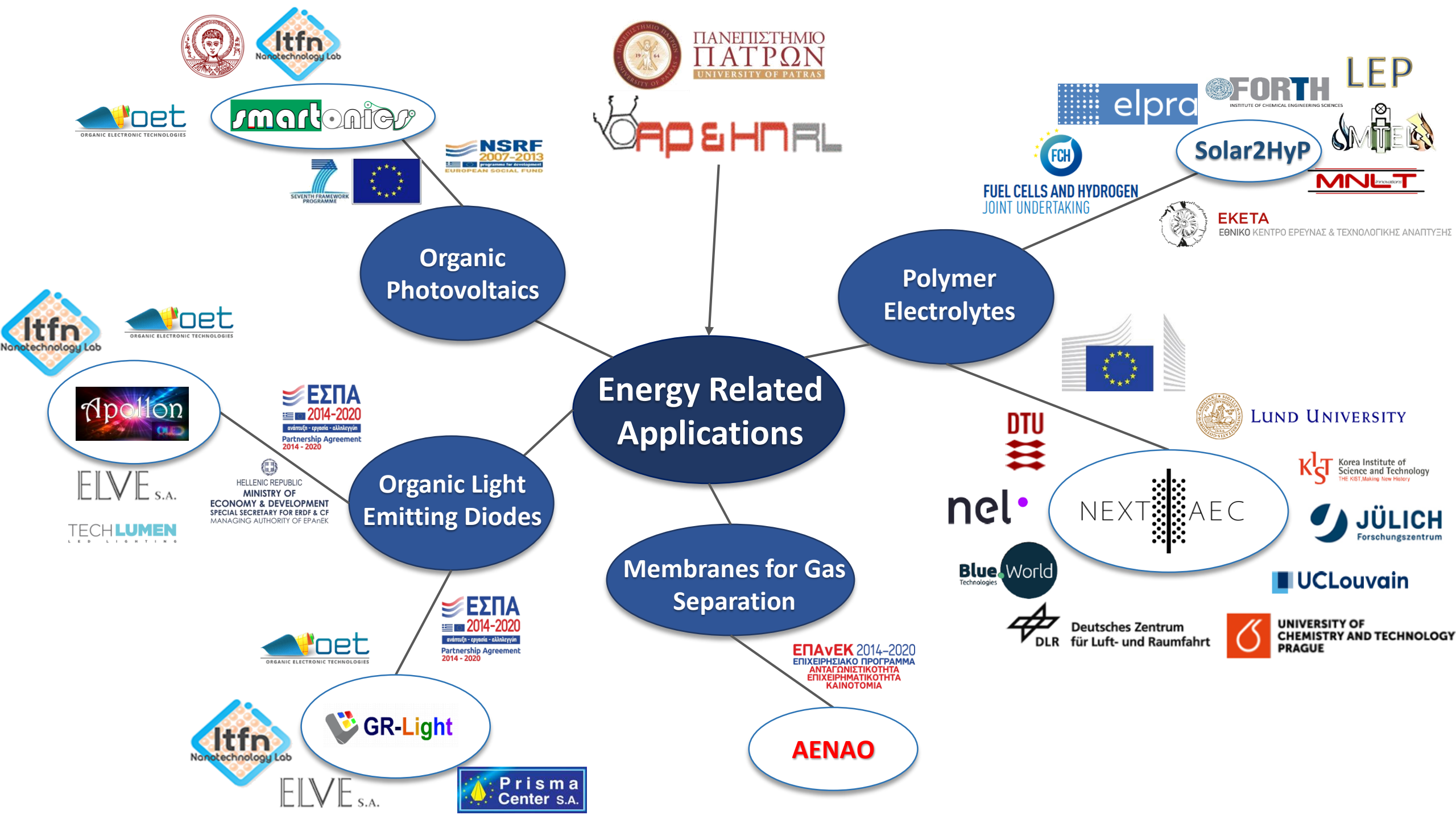
- Mrs. Stamatopoulou Eleftheria-Danae

- Mr. Fotis Panagiotou

Pre-graduate students *>15*

Secretary: Mrs. Evaggelia Kapota

Technical Support: Mr. Panagiotis Adamopoulos


Energy Related Applications
Organic Photovoltaics
Organic Light Emitting Diodes
Membranes for Gas Separation
Polymer Electrolytes
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS
AP & HNRL
ltfn Nanotechnology Lab
oet ORGANIC ELECTRONIC TECHNOLOGIES
smartonics
NSRF 2007-2013
EUROPEAN SOCIAL FUND
SEVENTH FRAMEWORK PROGRAMME
Apollon OLED
ELVE S.A.
TECHLUMEN
ΕΣΠΑ 2014-2020
Partnership Agreement 2014 - 2020
HELLENIC REPUBLIC
MINISTRY OF ECONOMY & DEVELOPMENT
SPECIAL SECRETARY FOR ERDF & CF
MANAGING AUTHORITY OF ΕΠΑνΕΚ
GR-Light
Prisma Center S.A.
ΕΠΑνΕΚ 2014-2020
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ
ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ
ΚΑΙΝΟΤΟΜΙΑ
ΑΕΝΑΟ
elpra
FORTH
INSTITUTE OF CHEMICAL ENGINEERING SCIENCES
LEP
Solar2HyP
FCH
FUEL CELLS AND HYDROGEN JOINT UNDERTAKING
MNLT
EKETA
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ & ΤΕΧΝΟΛΟΓΙΚΗΣ ΑΝΑΠΤΥΞΗΣ
DTU
LUND UNIVERSITY
nel
NEXT AEC
KIST Korea Institute of Science and Technology
THE KIST,Making New History
JÜLICH Forschungszentrum
Blue World Technologies
UCLouvain
DLR Deutsches Zentrum für Luft- und Raumfahrt
UNIVERSITY OF CHEMISTRY AND TECHNOLOGY PRAGUE


Life Quality
Antimicrobial Applications
Bionet
Sterial
Functional Materials for Optimization of Industrial Products
Chromasurf
SELFNANOPUD
Thorax
Environmental Applications
INCOMERA
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS
ΑΡ & ΗΠΑΛ
Τμήμα Αλιείας και Υδατοκαλλιεργειών
Τμήμα Γεωλογίας
SELONDA
ΕΠΑΛΘ 2014-2020
METRICON Digital Systems
Alumil
FILTREX FILTER SOLUTIONS
RINGEL Electronics
GLM
ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ ΙΚΥ
ITE ΙΔΡΥΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΡΕΥΝΑΣ
Systems Lab
incomera.eu
ΠΕΡΙΦΕΡΕΙΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
SIRMET ENGINEERING & MANAGEMENT
LAMS
ELVIFA - ΚΑΛΟΓΕΡΑΚΗΣ Η. ΜΙΧΑΗΛ ΑΕ
SOULIS KUEHNIS EMBROIDERIES
CHROMA surf
colora s.a.
ΕΠΑνΕΚ 2014-2020 ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ ΚΑΙΝΟΤΟΜΙΑ
ΕΣΠΑ 2014-2020
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
MEGARA RESINS
SIAMIDIS
ER-LAC
ADAMANT
Τμήμα Μηχανολόγων & Αεροναυπηγών Μηχανικών

Amphiphilic Polymers

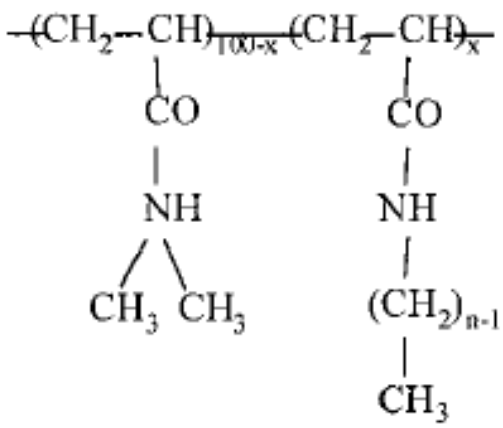
Hydrophobically modified water-soluble polymers

Main collaborators: Dr. Ilias Iliopoulos, Prof. Dominique Hourdet
ESPCI, France

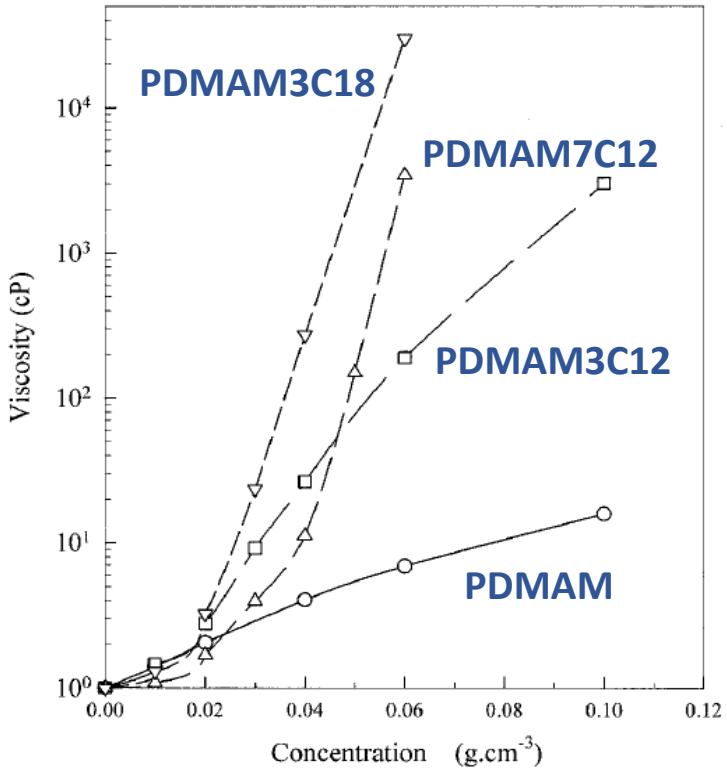
Water-soluble backbone

Examples

i) Hydrophobically modified PDMAM



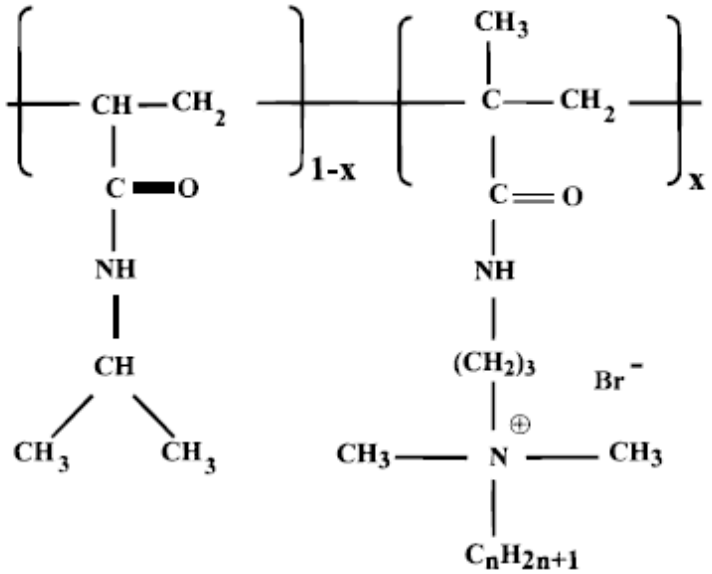
PDMAMxCn

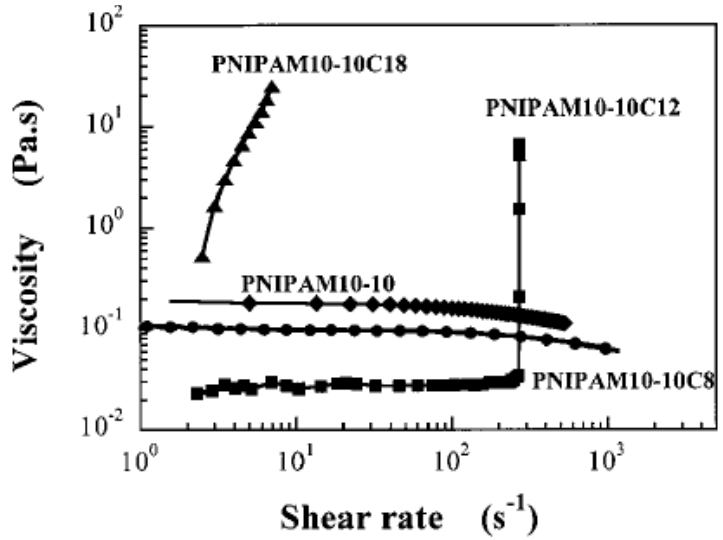


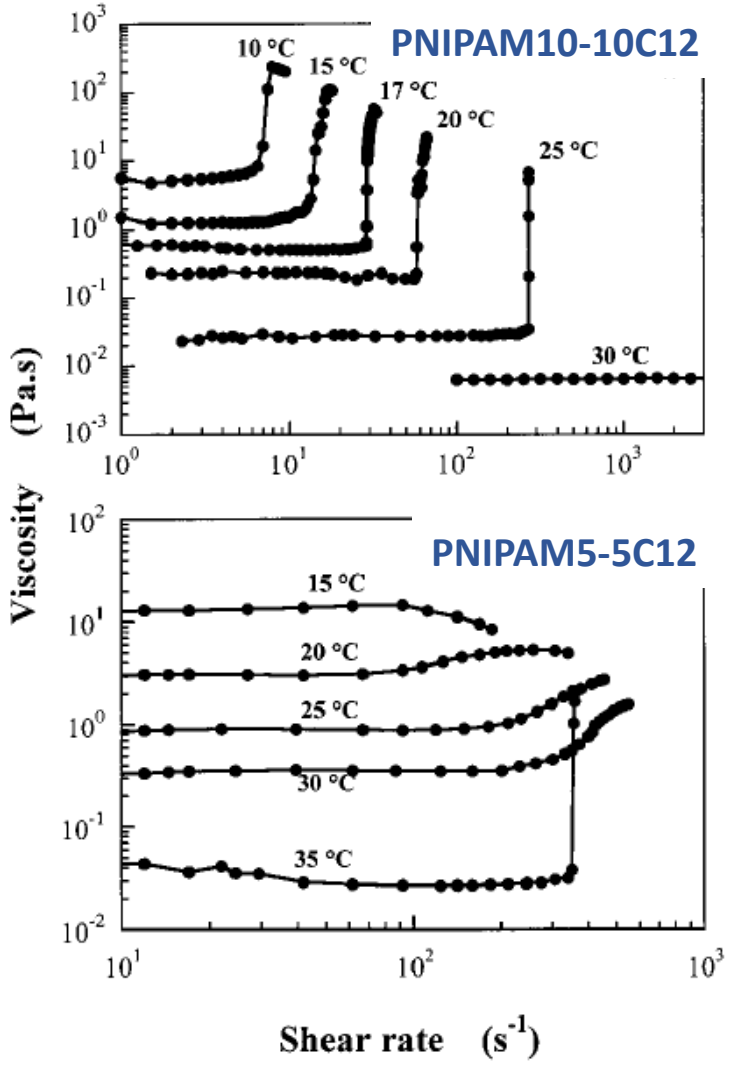
Macromol. Chem. Phys. 2000, *201*, 251-260.

Amphiphilic Polymers

ii) Cationic hydrophobically modified PNIPAM





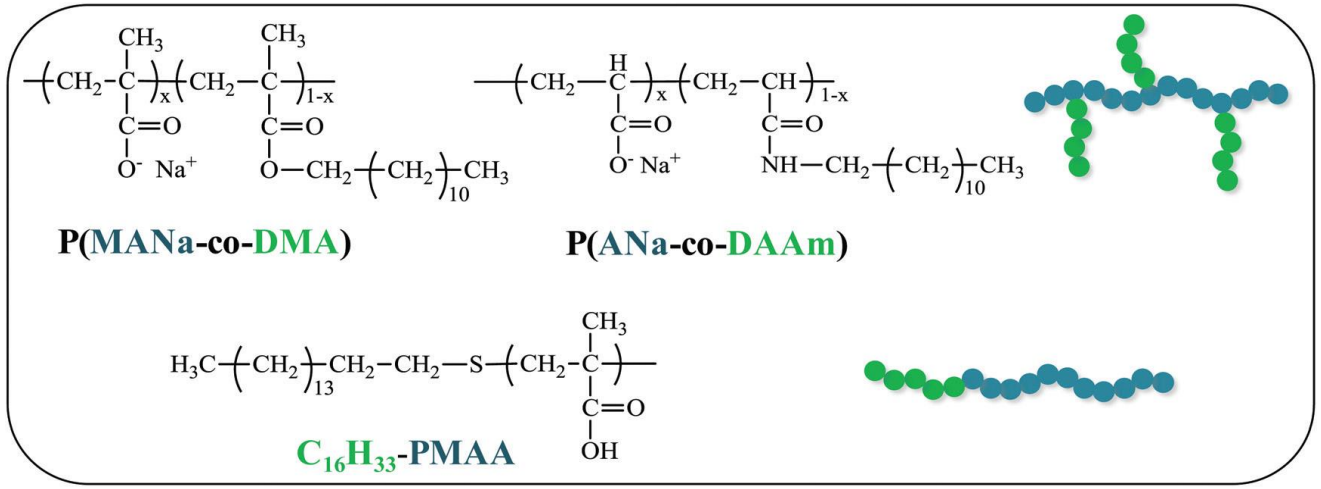


Macromolecules 2000, *33*, 2929-2935.

Amphiphilic Polymers - Transfer of Hydrophobic Magnetic NPs in Water

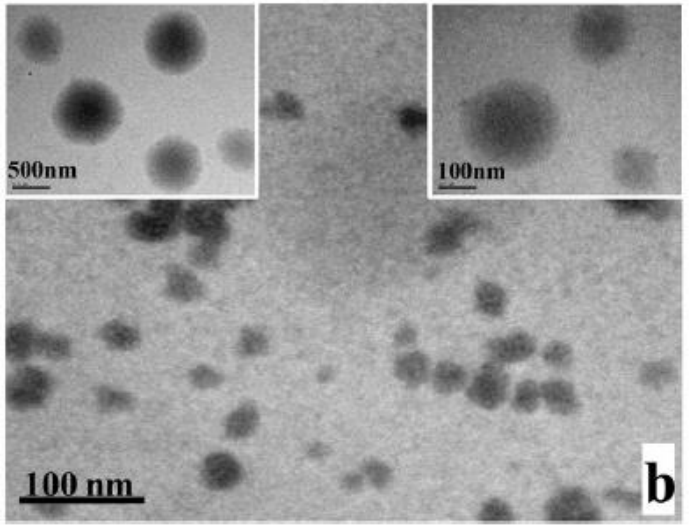
Main collaborator: Prof. Katherine Dendrinou-Samara, Chemistry Department, Univ. of Thessaloniki

i. Use of hydrophobically modified polyelectrolytes

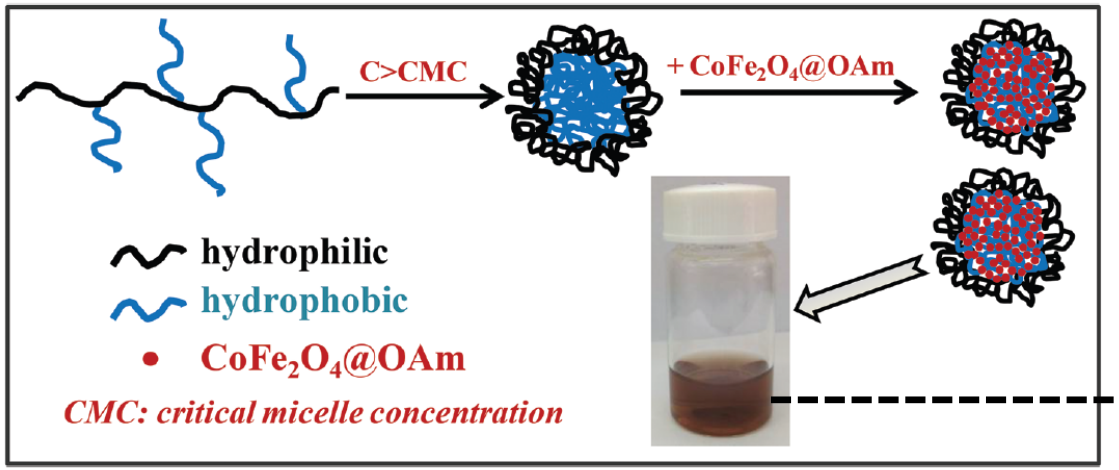


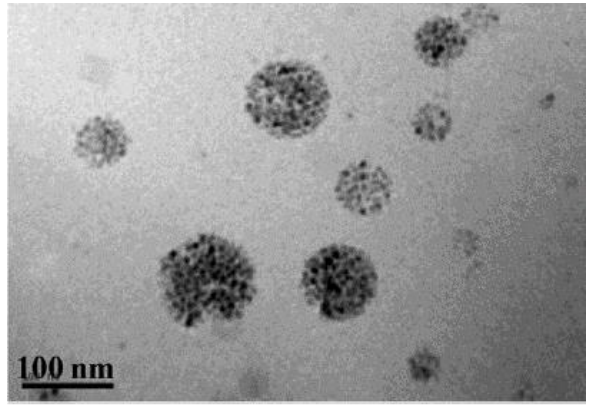
Self-assembly in aq. media

Cpol > CMC



✓ **dispersion** of hydrophobic MNPs in aqueous media

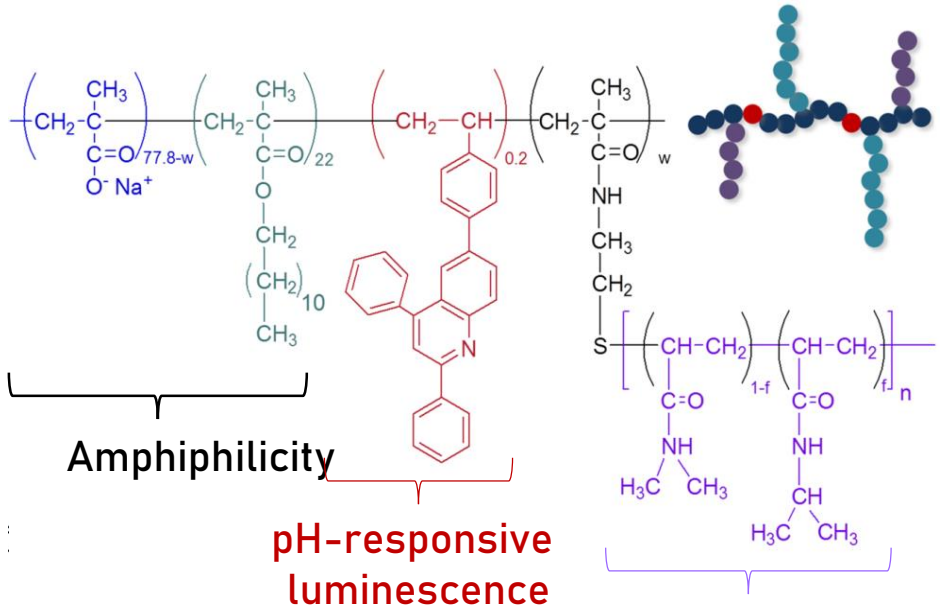




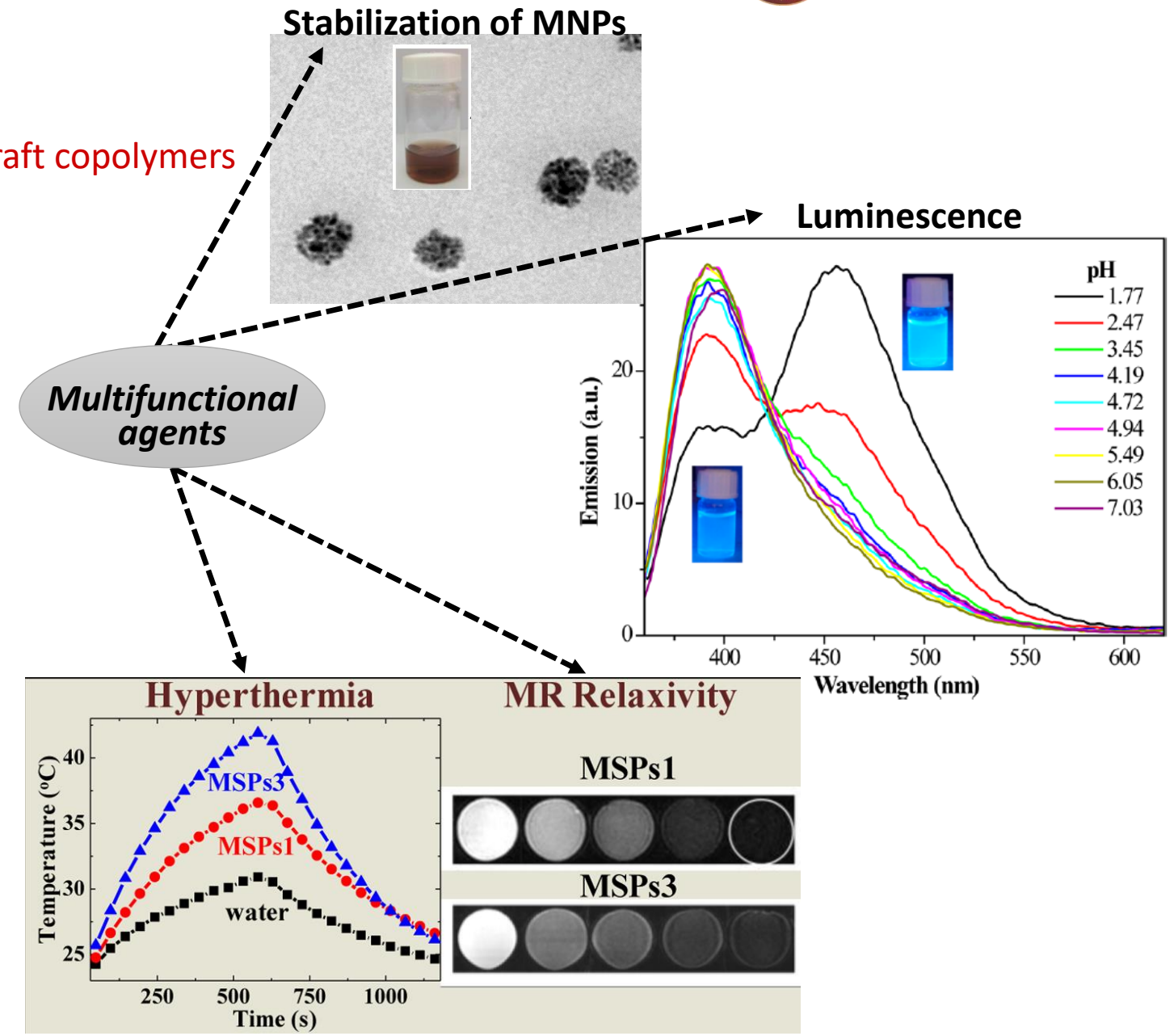
Dalton Trans. 2014, *43*, 8633-8643. *Dalton Trans*. 2015, *44*, 10980–10990.

Multifunctional Polymeric Platform of Magnetic Colloidal Superparticles

ii. Use of hydrophobically modified thermosensitive graft copolymers



P(MANa-co-DMA-co-SDPQ)-g-P(DMAM-co-NIPAM)

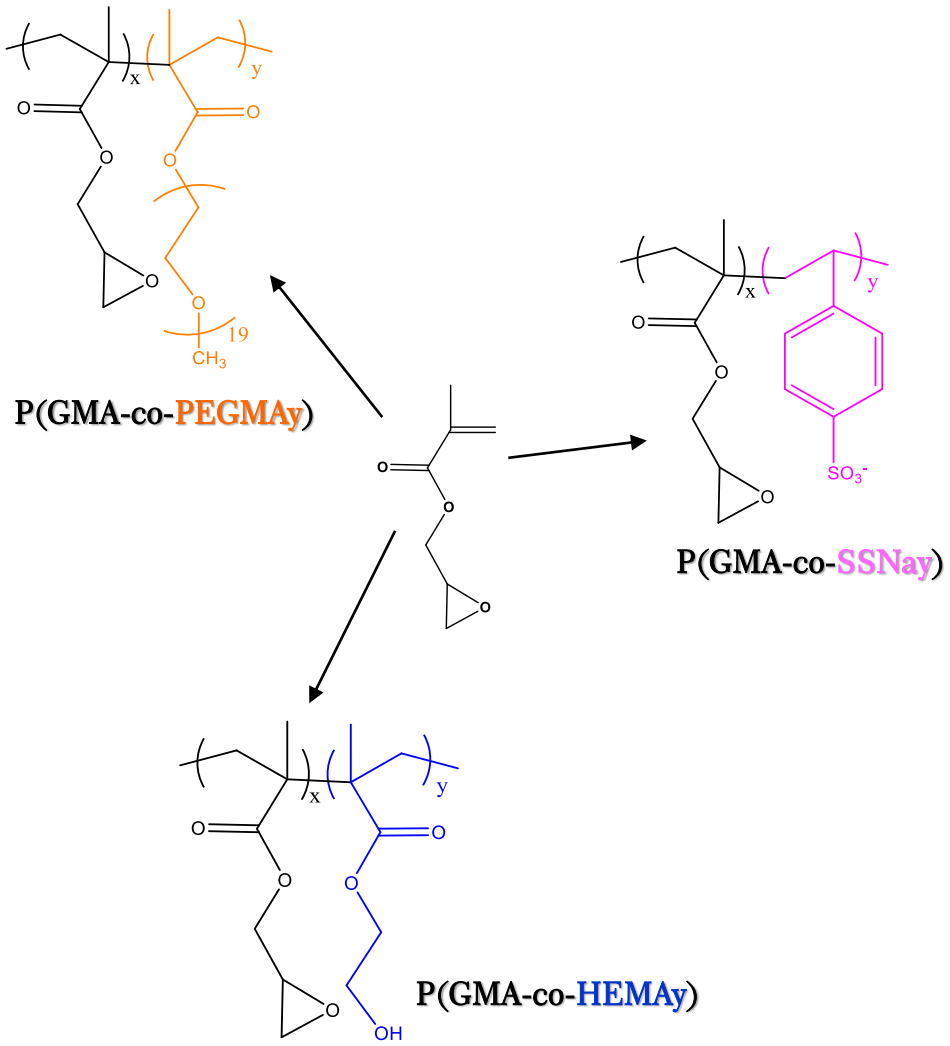


ACS Appl. Mater. Interfaces 2016, *8*, 35059.

Amphiphilic/Functional Copolymers

GMA-copolymers with reactive functional groups - Self-healing properties

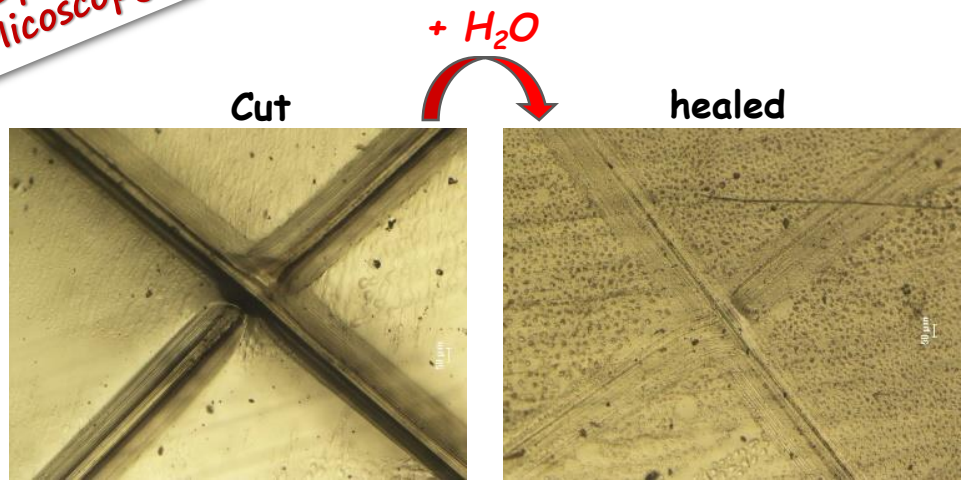
Main collaborator: Dr. P. Krassa, MEGARA Resins



Self-Healing Process

Polyurethane film containing
10 wt.% copolymer *P(GMA-co-PEGMA)*

Optical Micoscopy



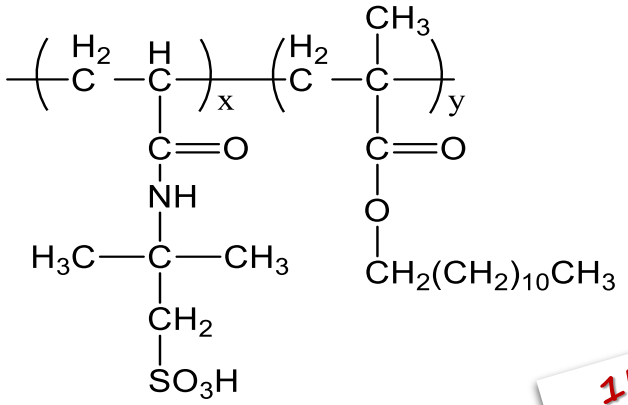
Int. J. Mol. Sci. 2022, *23*, 8118.

«Enhanced oil recovery by polymer-coated nanoparticles»

(Project Leader: Research Director Christos Tsakiroglou).

Amphiphilic copolymers

Water soluble amphiphilic copolymers



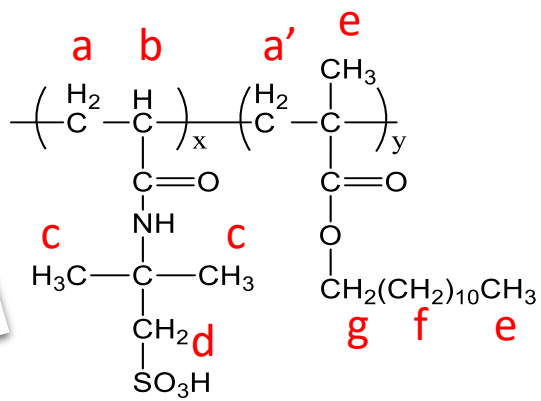
P(AMPSA-co-DMAy)

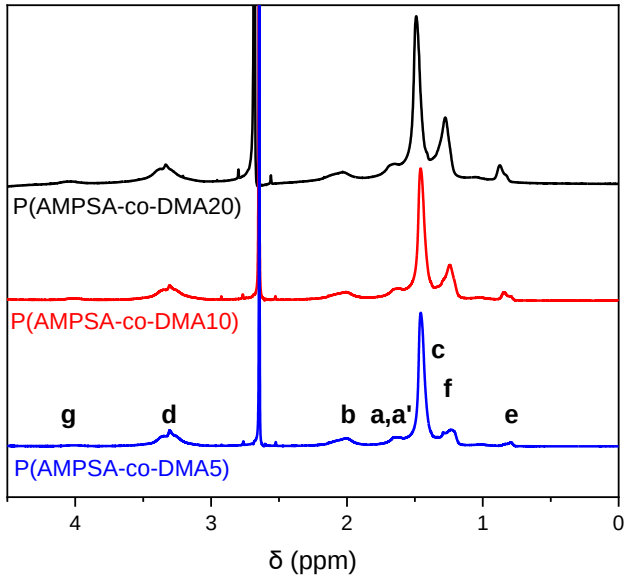
x=95, y=5 %mol

x=90, y=10 %mol

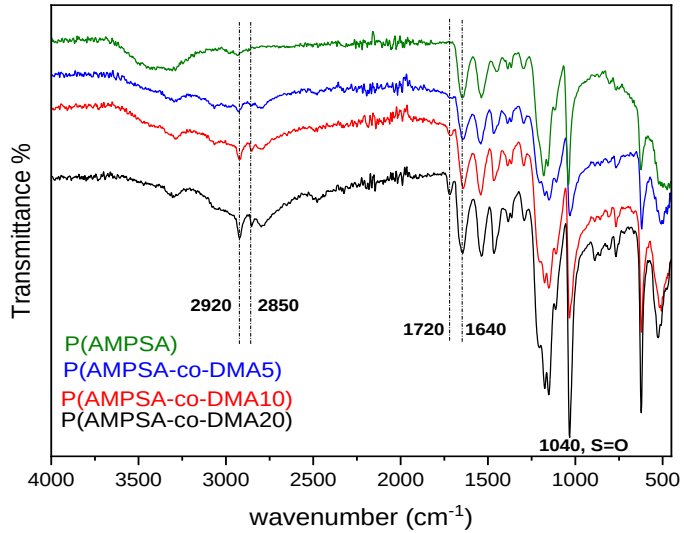
x=80, y=20 %mol

1H NMR





ATR-FTIR



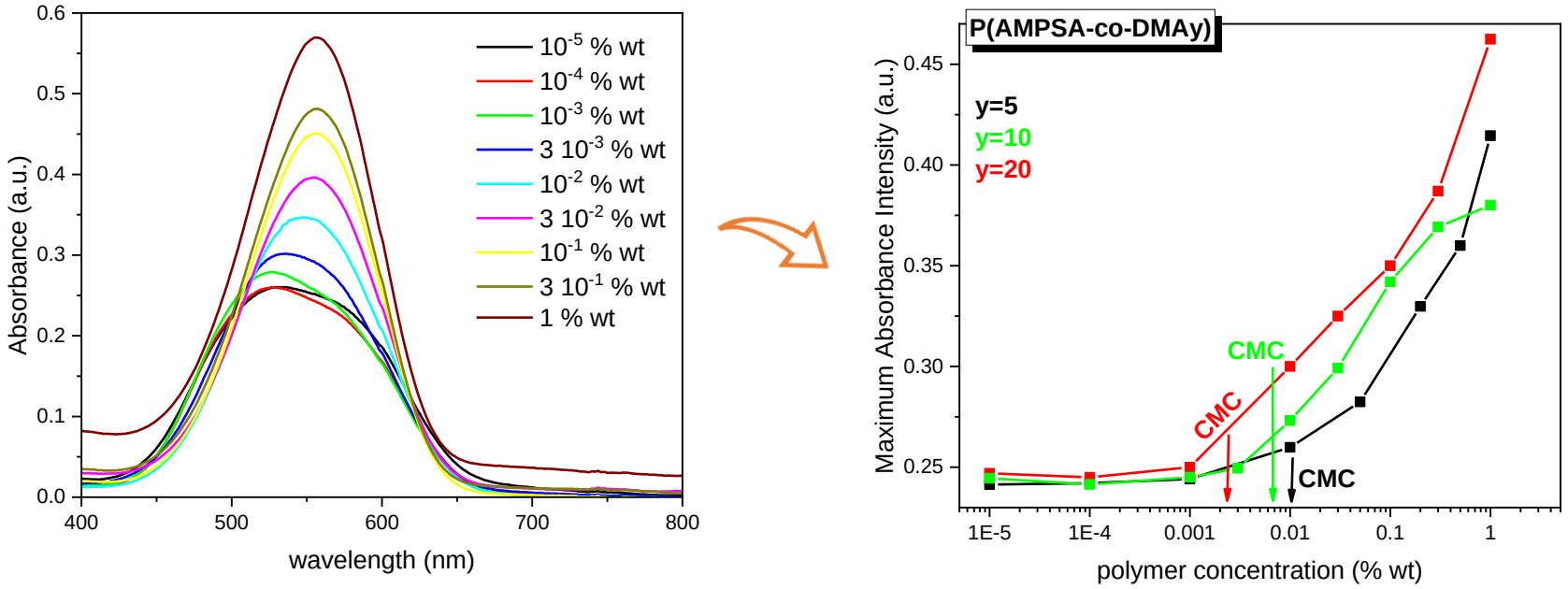
GPC (aq. $LiNo_3$)

polymer	Mn, g/mol	Mw, g/mol	PDI
$P(AMPSA_{90}$-co-$DMA_{10})$	13700	22500	1.64
$P(AMPSA_{95}$-co-$DMA_5)$	16800	34200	2.04

Amphiphilic copolymers

Self-organization in aqueous media

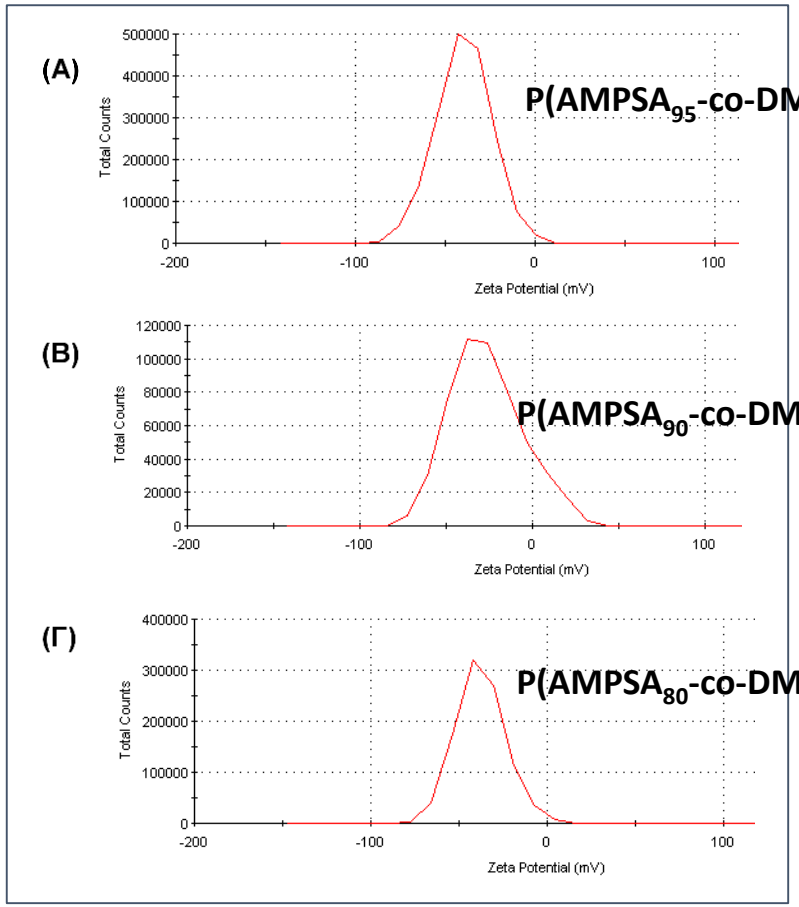
***Nile Red* probing** *for the determination of the Critical Micelle Concentration (CMC)*



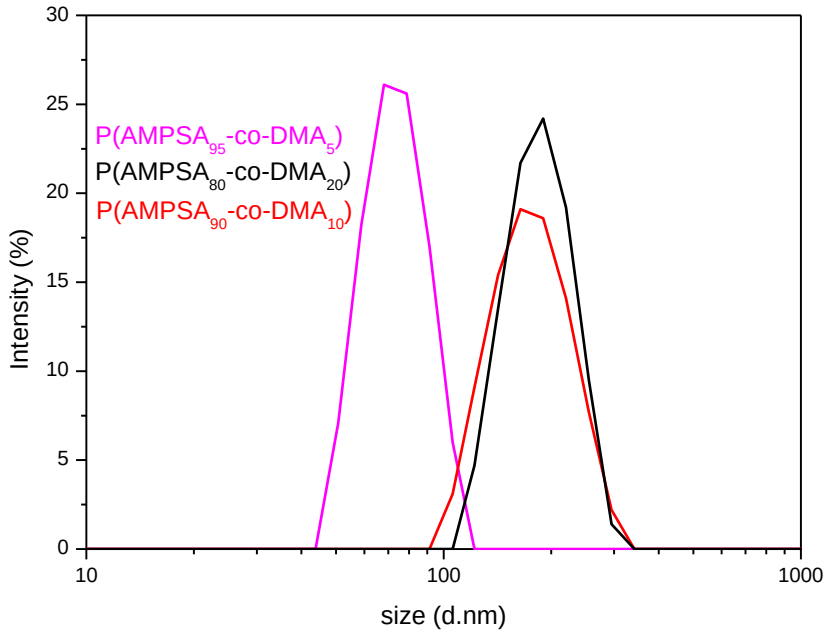
Amphiphilic copolymers

Self-organization in aqueous media

Zeta-potential



DLS

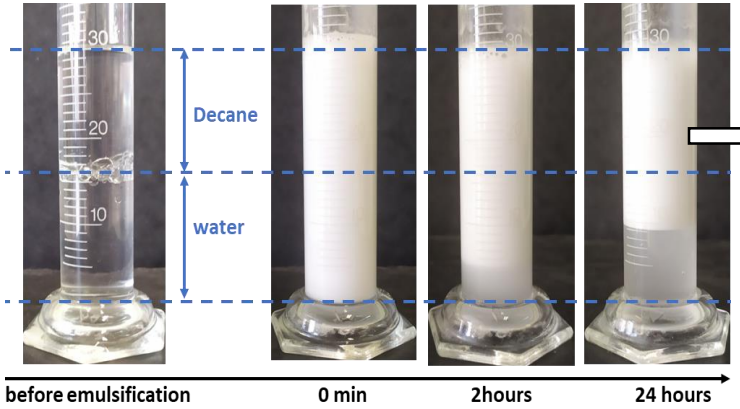


	zeta(mV)	size: diameter (nm)		
polymer		by intensity	by volume	by number
P(AMPSA$_{95}$-co-DMA$_5$)	-39.8	74.1	67.7	61.5
P(AMPSA$_{90}$-co-DMA$_{10}$)	-26.8	179.3	176.6	146.6
P(AMPSA$_{80}$-co-DMA$_{20}$)	-37.8	188.1	187.9	78.8

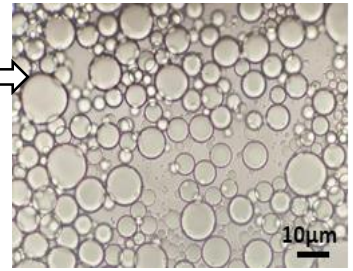
Amphiphilic copolymers

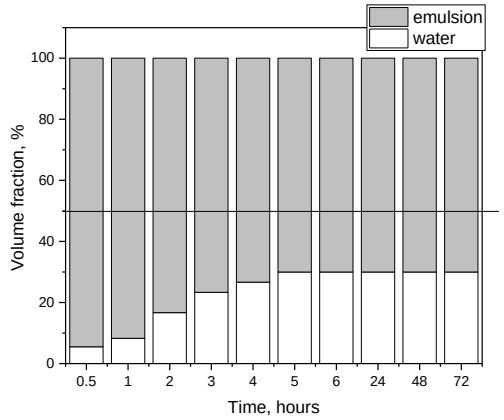
Emulsifying Properties

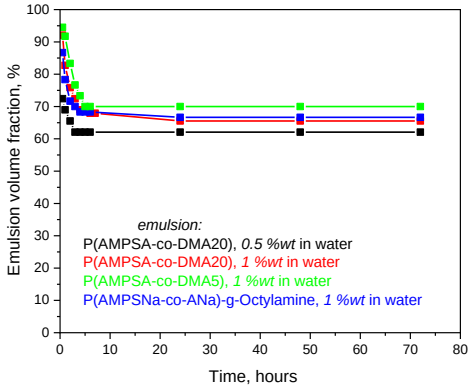
Visual appearance of P(AMPSA-co-DMA) emulsions



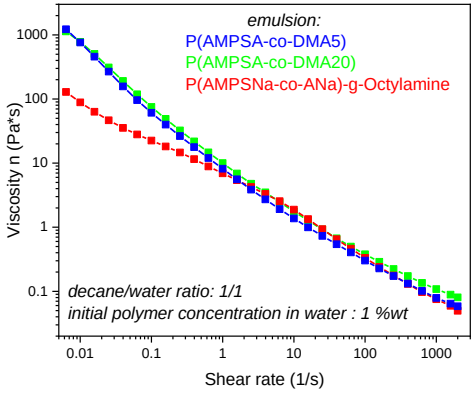
Microscope image

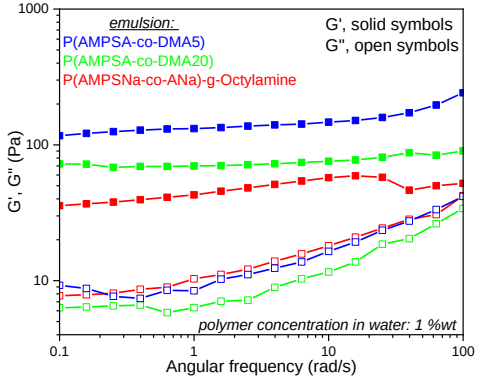






Rheology

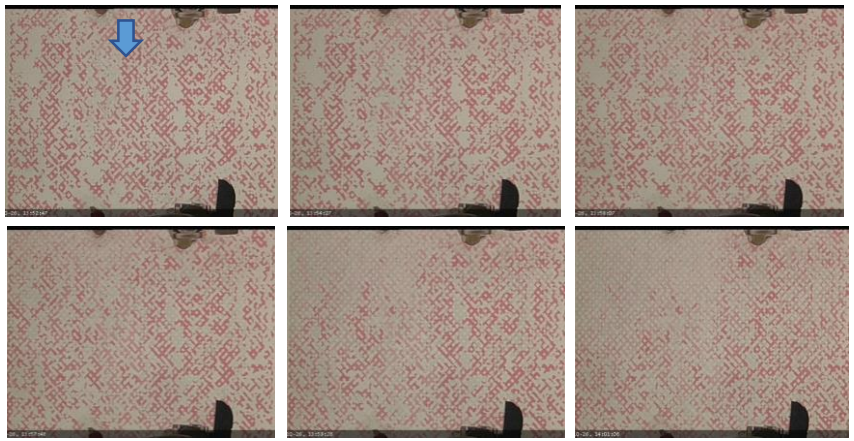




Non-Newtonian fluid type
Shear-thinning

✓ G' > G'' : solid-like behavior

Displacement (secondary imbibition) of oil by emulsion



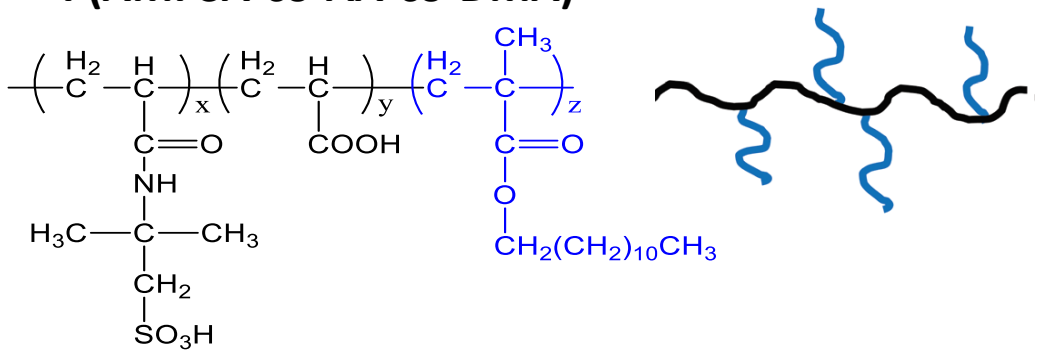
Successive snap-shots of the displacement of dodecane by emulsion prepared by mixing equal volumes (20mL:20mL) of decane with aq. dispersion of 1%wt P(AMPSA-co-DMA20)

Gradual creation of a frontal drive that progressively mobilizes the majority of residual oil due to the high viscosity ratio

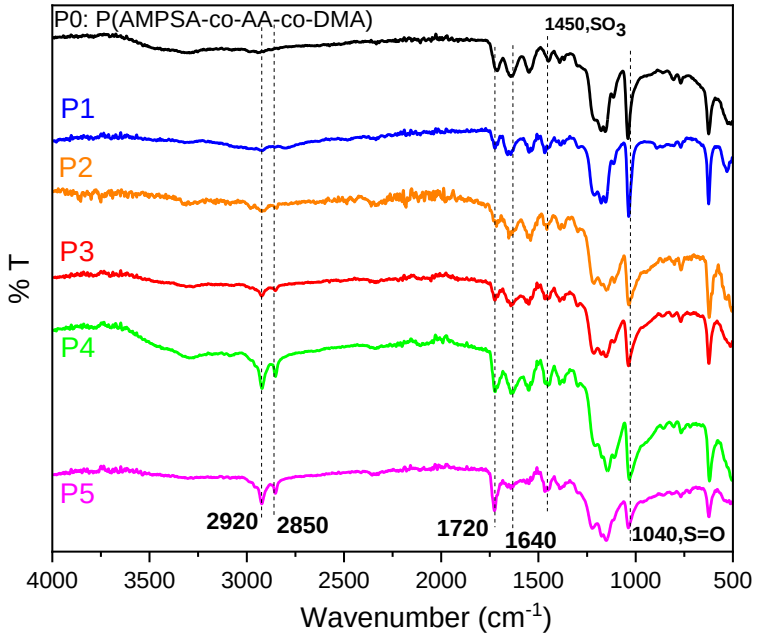
Amphiphilic terpolymers

Water soluble amphiphilic terpolymers

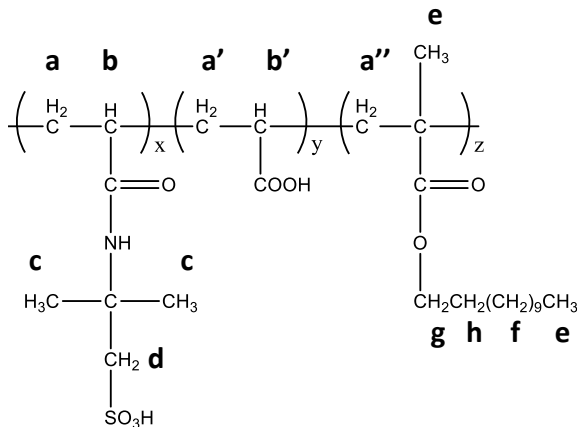
P(AMPSA-co-AA-co-DMA)

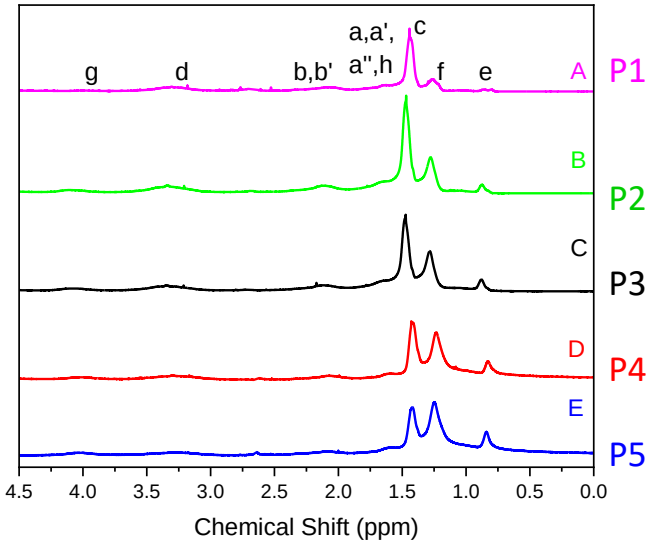


ATR-FTIR



1H NMR





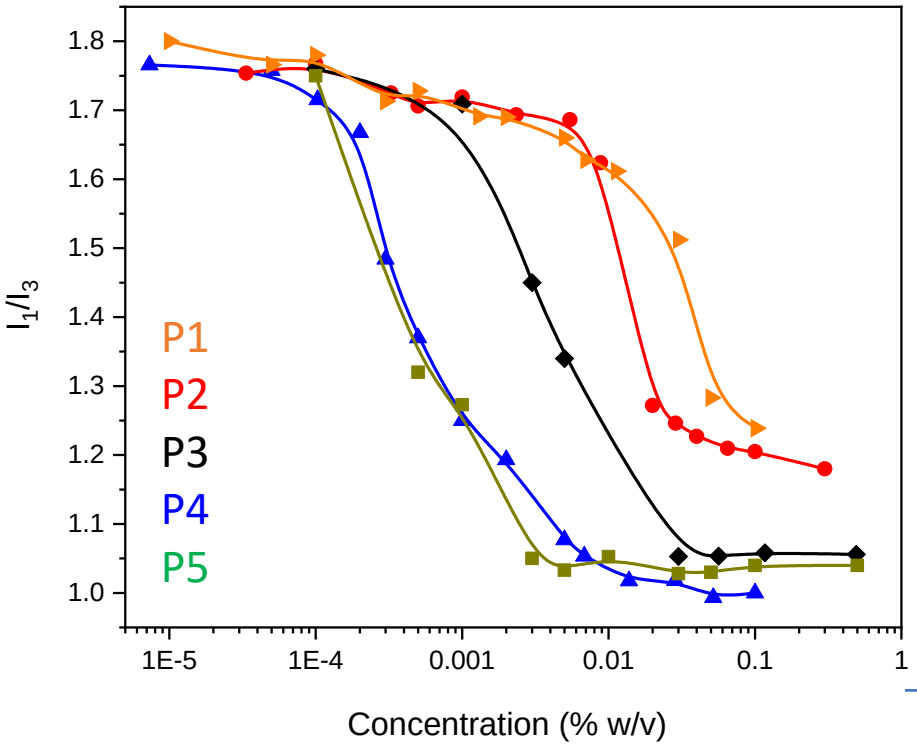
GPC

Polymer	Theoretical composition AMPSA/AA/DMA (% mol)	Composition from 1H NMR AMPSA/AA/DMA (% mol)	Mn (g/mol)	Mw (g/mol)	PDI
P1	65/30/05	60/37/3	17000	23400	1.38
P2	60/20/20	68/27/5	11310	15460	1.37
P3	68/22/10	66/22/12	12300	14450	1.17
P4	50/25/25	55/20/25	-	-	-
P5	40/30/30	39/33/28	-	-	-

Amphiphilic terpolymers

Self-organization in aqueous media

***Pyrene probing** for the determination of the Critical Micelle Concentration (CMC)*



Polymer	CMC (% w/v)
P1: P(AMPSA60-co-AA37-co-DMA3)	0.04
P2: P(AMPSA68-co-AA27-co-DMA5)	0.02
P3: P(AMPSA66-co-AA22-co-DMA12)	0.008
P4: P(AMPSA55-co-AA20-co-DMA25)	0.002
P5: P(AMPSA39-co-AA33-co-DMA28)	0.001

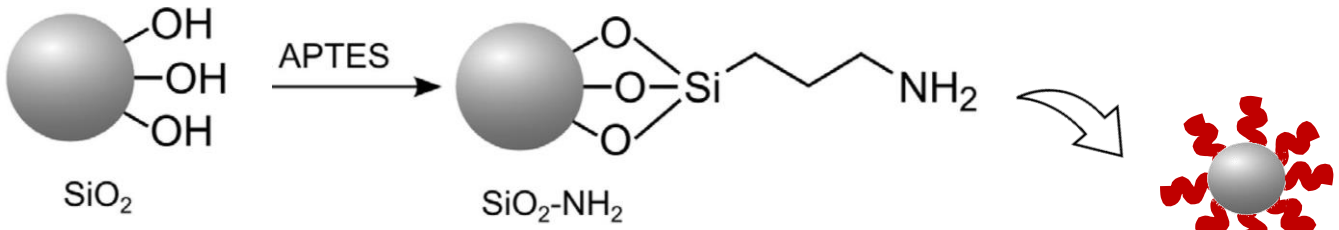
Zeta-potential & DLS

polymer	Zeta mV)	size:diameter (nm)		
		by intensity	by volume	by number
$P(AMPSA_{55}\text{-co-}DMA_{20}\text{-co-}AA_{25})$	-30.4	117.6	-	-
$P(AMPSA_{68}\text{-co-}DMA_{27}\text{-co-}AA_{5})$	-32.9	164.0	122.0	78.8
$P(AMPSA_{60}\text{-co-}DMA_{37}\text{-co-}AA_{3})$	-31.8	93.7	107.3	91.2

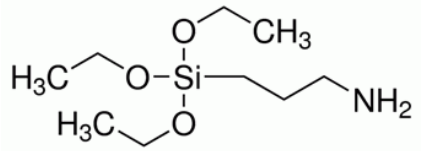
Polymer-coated nanoparticles

Polymer-coated nanoparticles (PNPs)

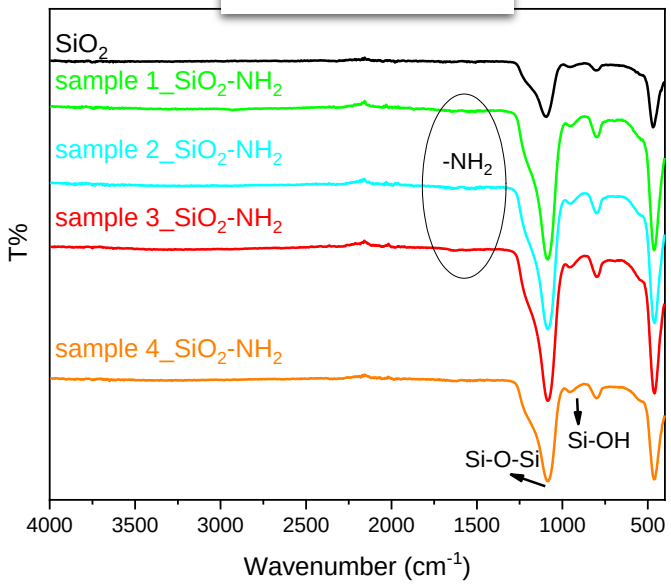
***Step 1) Amine**-functionalization of SiO_2 NPs with aminopropyltriethoxysilane (**SiO_2-NH_2**)*



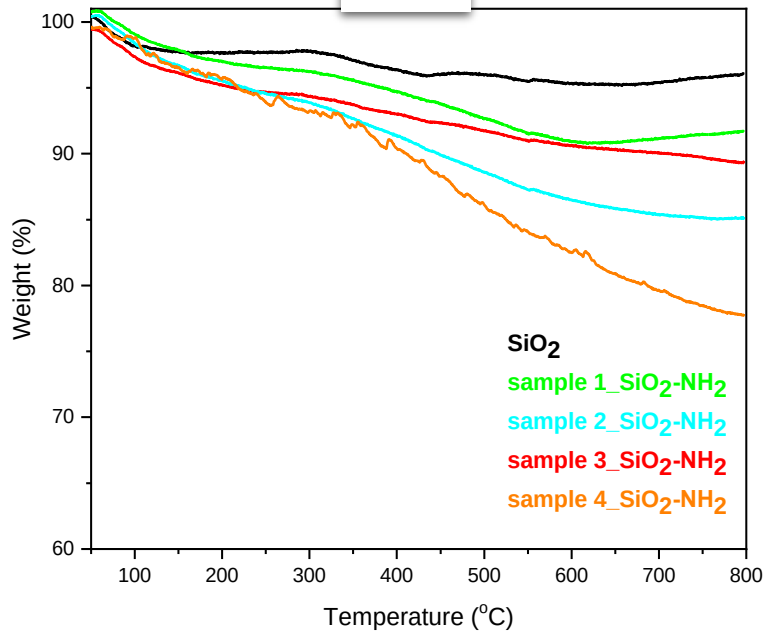
APTES:



ATR-FTIR



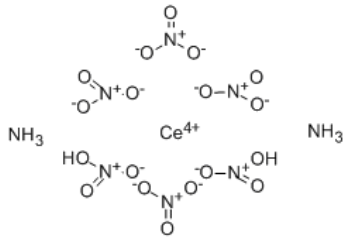
TGA



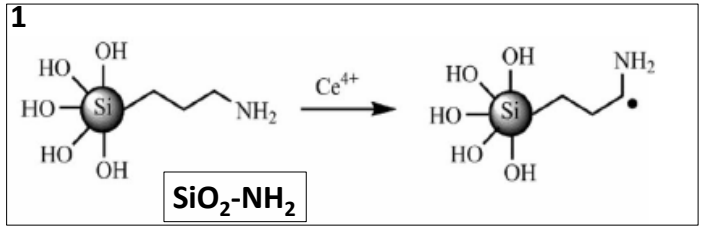
Polymer-coated nanoparticles

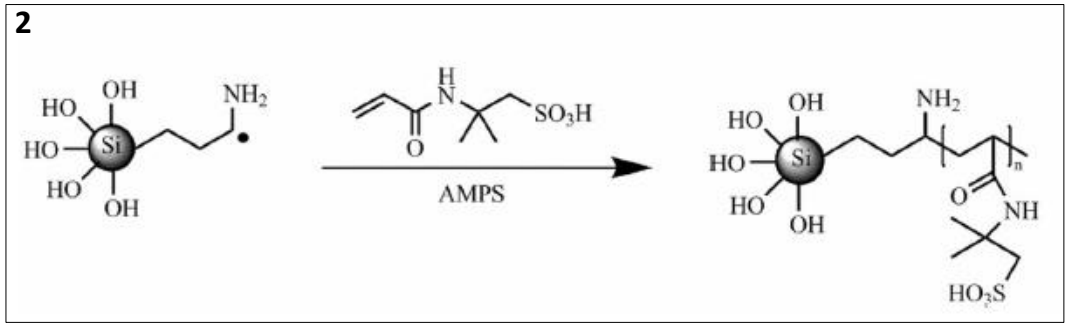
Polymer-coated nanoparticles (PNPs)

Methodology I) *Polymerization of AMPSA onto the functionalized* SiO_2-NH_2 *NPs through* ***Cerium (IV) ammonium nitrate*** *oxidizing agent (****SiO_2-PAMPSA****)*

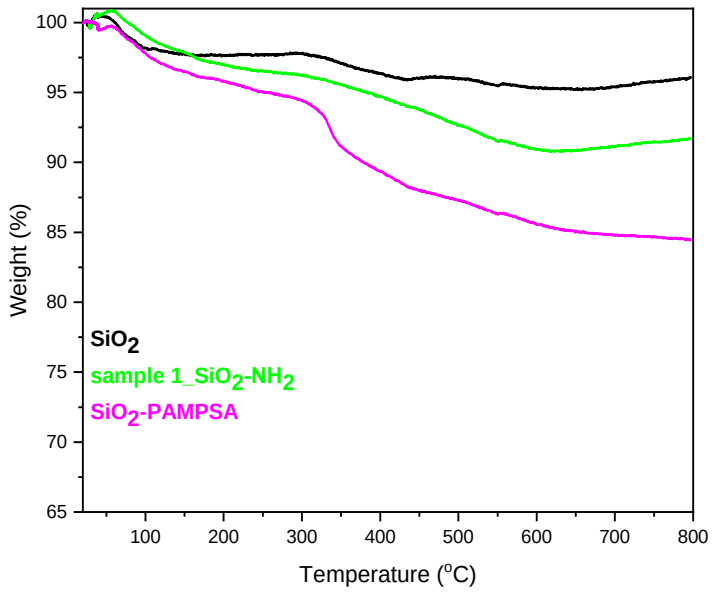


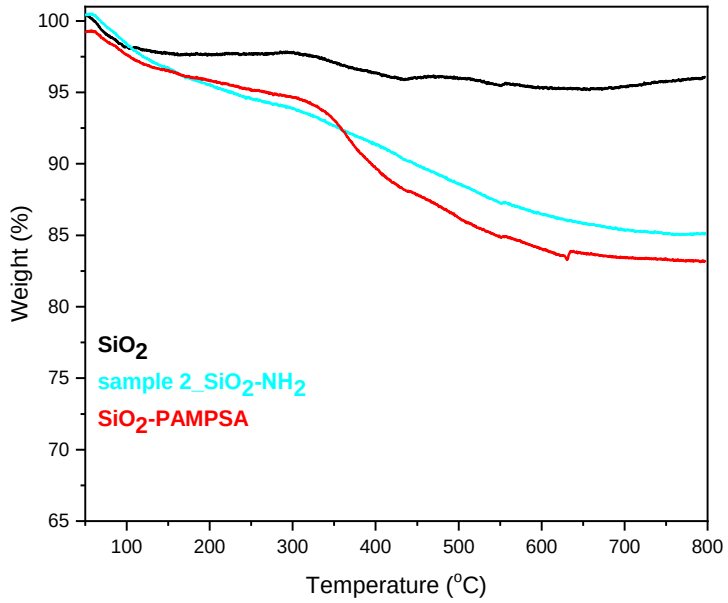
cerium (iv) ammonium nitrate





TGA

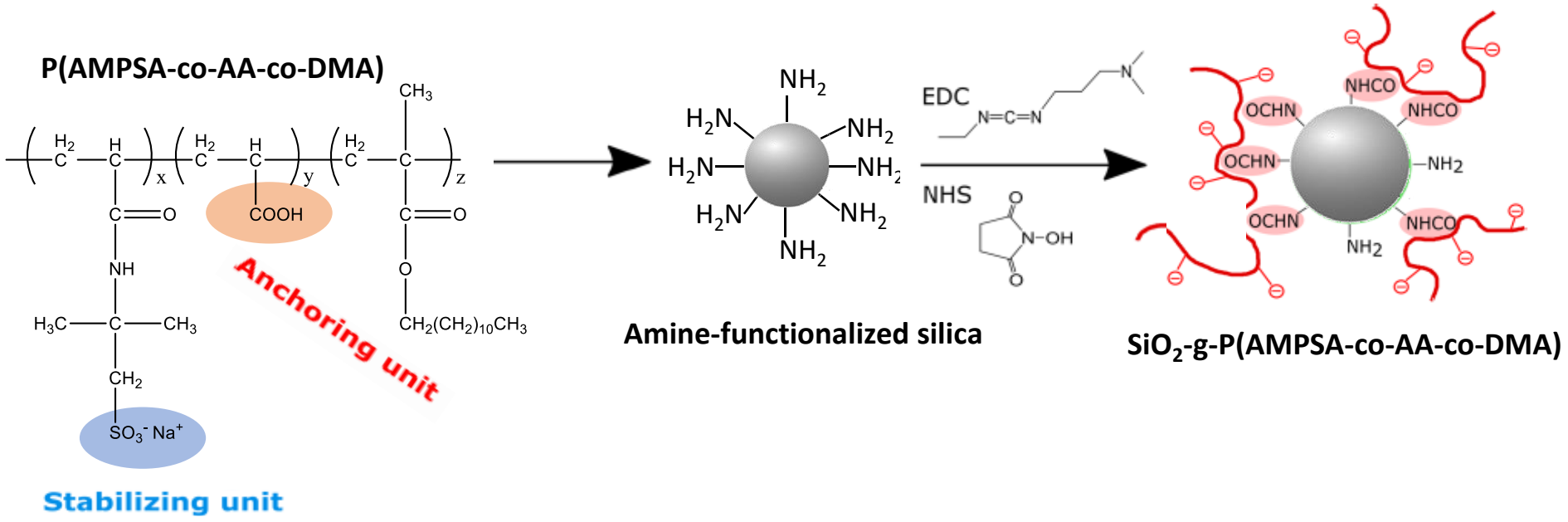




Polymer-coated nanoparticles

Polymer-coated nanoparticles (PNPs)

Methodology II) Post-grafting *of terpolymers on SiO_2-NH_2 NPs by* ***carbodiimide chemistry***

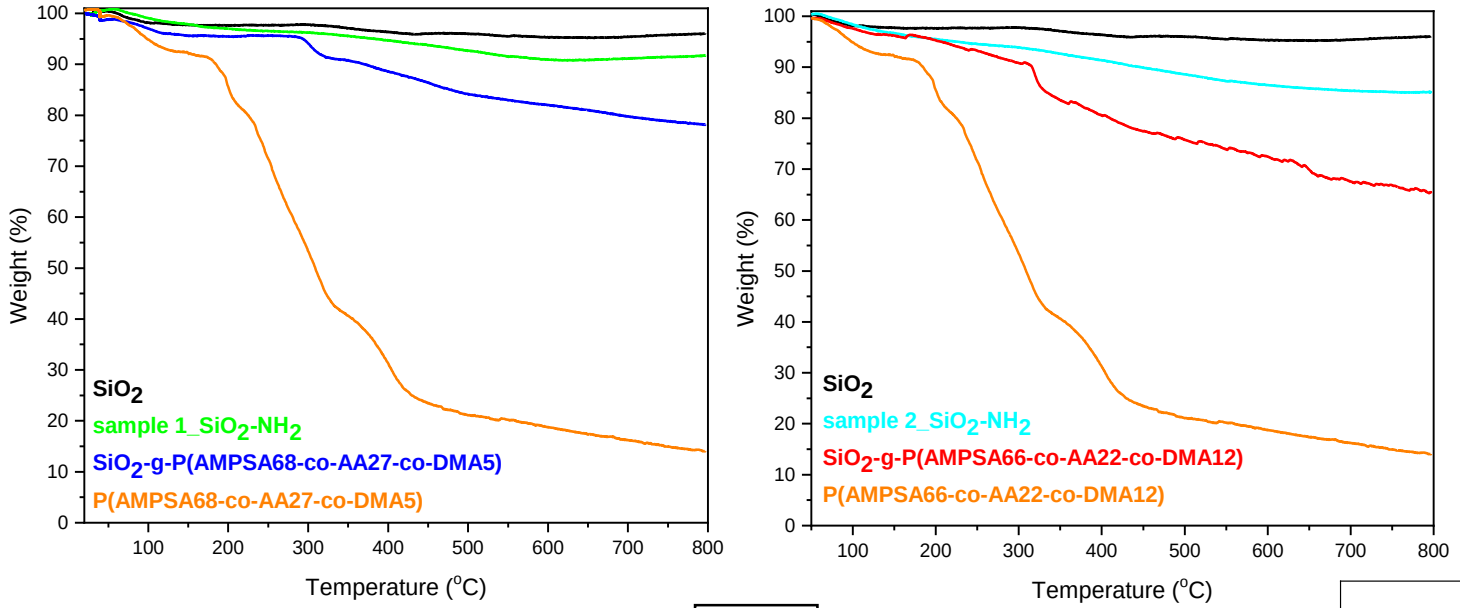


SiO_2-g-P(AMPSA-co-AA-co-DMA)

Polymer-coated nanoparticles

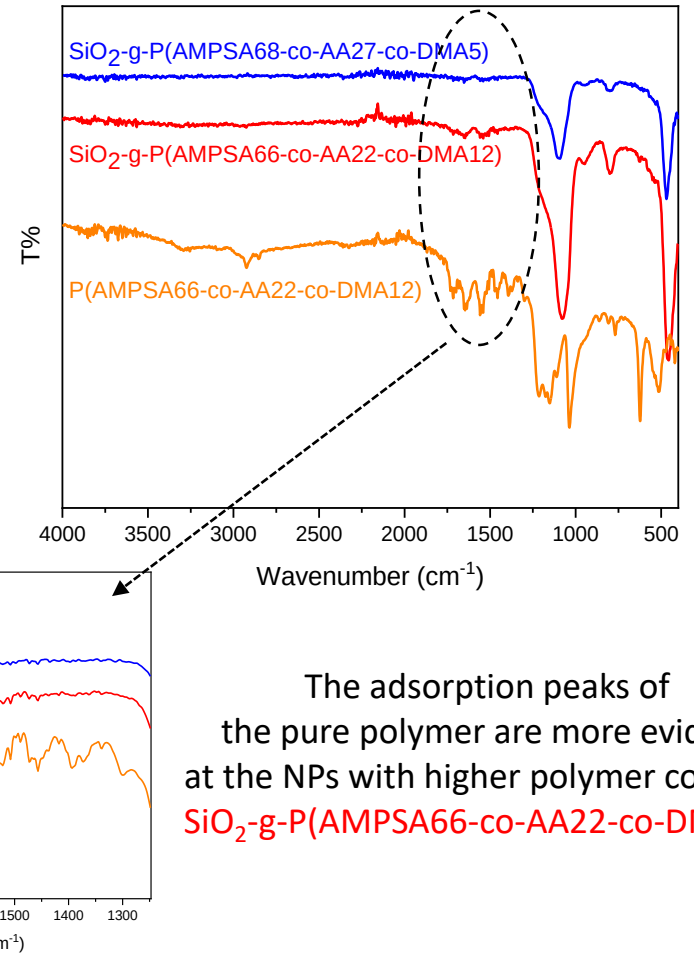
PNPs Characterization

TGA



SiO_2-g-P(AMPSA66-co-AA22-co-DMA12) NPs have more polymer content than SiO_2-g-P(AMPSA68-co-AA27-co-DMA5)

ATR-FTIR



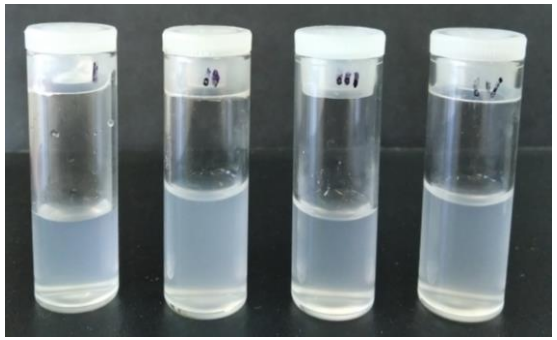
The adsorption peaks of the pure polymer are more evident at the NPs with higher polymer content, SiO_2-g-P(AMPSA66-co-AA22-co-DMA12)

Polymer-coated nanoparticles

Stability of polymer-coated SiO_2 particles in aqueous media

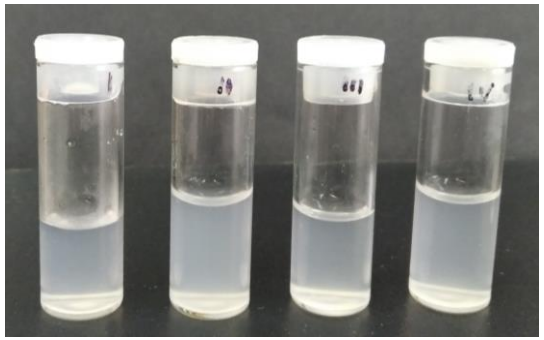
0.1 % w/v SiO_2-g-P(AMPSA66-co-AA22-co-DMA12) NPs

0 hours



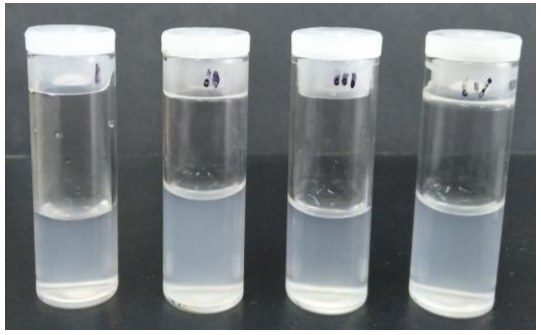
H_2O NaCl 1%wt NaCl 4 %wt NaCl 10 %wt

2 hours

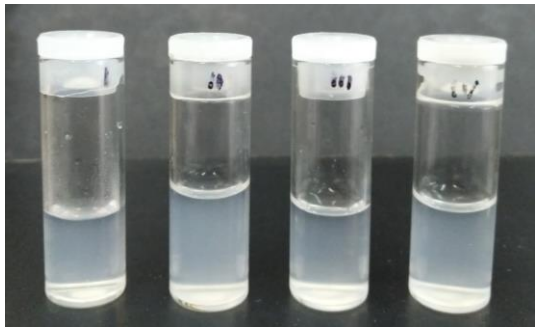


H_2O NaCl 1%wt NaCl 4 %wt NaCl 10 %wt

6 hours



24 hours

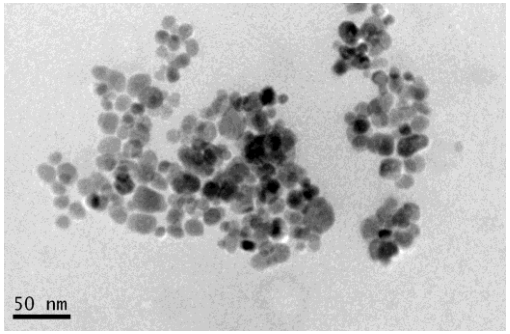


Polymer-coated nanoparticles

Iron oxide magnetic nanoparticles

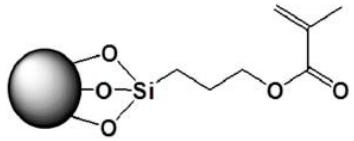
Synthesis of iron oxide NPs (co-precipitation method)

$$1\ Fe^{2+} + 2\ Fe^{3+} + 8\ OH^- \rightarrow Fe_3O_4 + 4H_2O$$



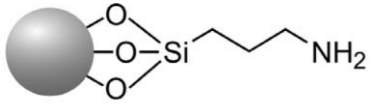
➢ Surface functionalization of Fe_3O_4 NPs with

- double bonds



Fe_3O_4-MPS

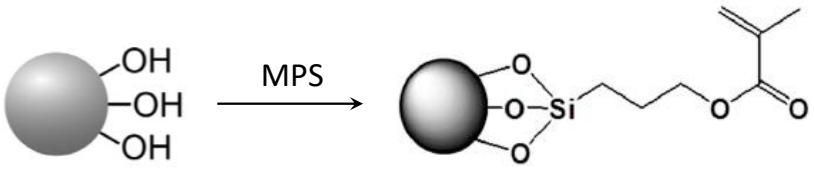
- amine groups



Fe_3O_4-NH_2

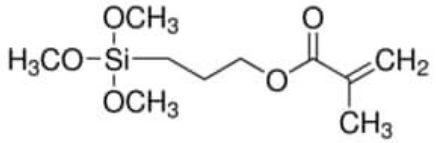
Polymer-coated nanoparticles

***Functionalization** with MPS*

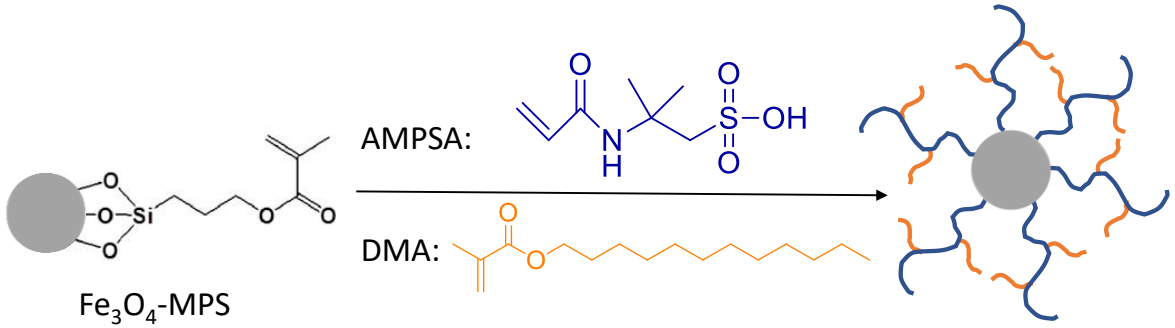


Fe_3O_4 Fe_3O_4-MPS

MPS :



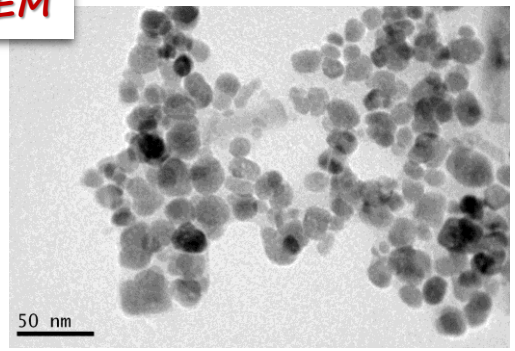
***Polymerization** of AMPSA and DMA monomers onto the functionalized Fe_3O_4-MPS NPs*



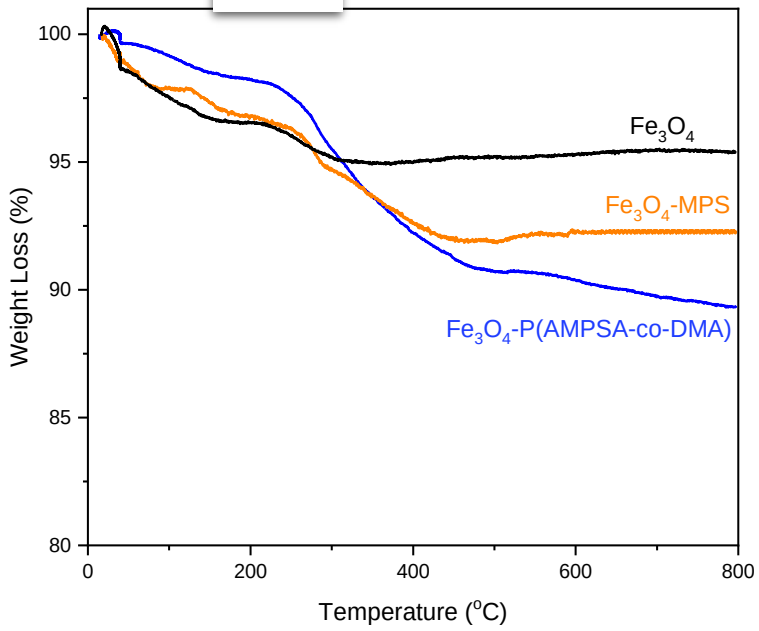
Fe_3O_4-MPS

Fe_3O_4-P(AMPSA-co-DMA)

TEM

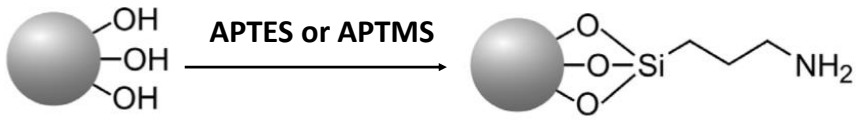


TGA

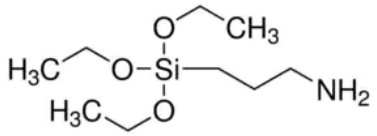


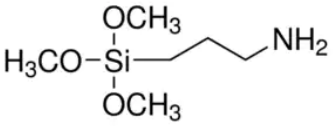
Polymer-coated nanoparticles

***Amine**-functionalization with APTES or APTMS*

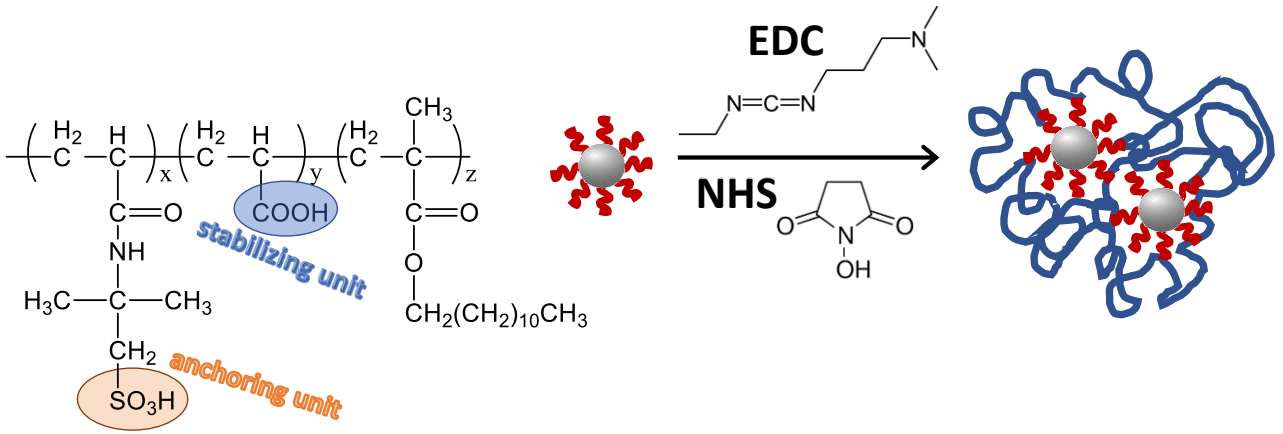


Fe_3O_4 Fe_3O_4-NH_2

APTES : 

APTMS : 

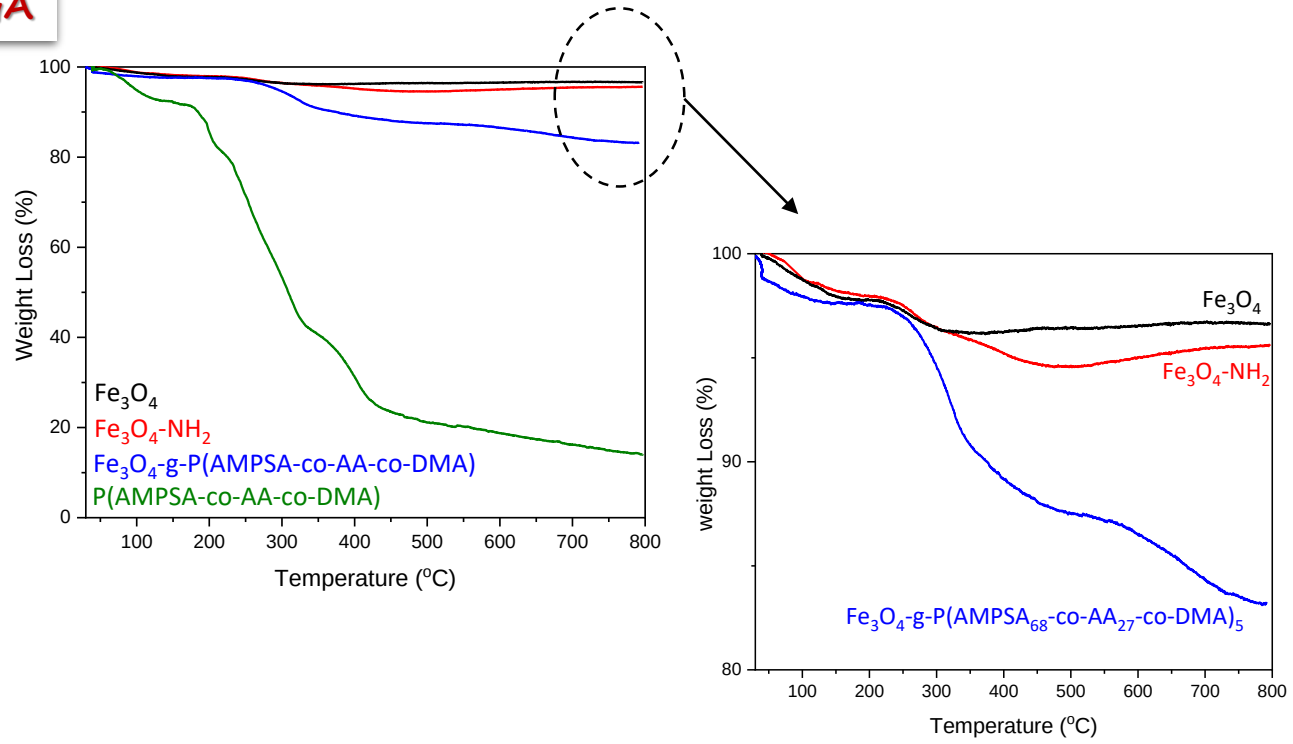
***Post-grafting** of terpolymers on Fe_3O_4-NH_2 NPs*
*by **carbodiimide chemistry***



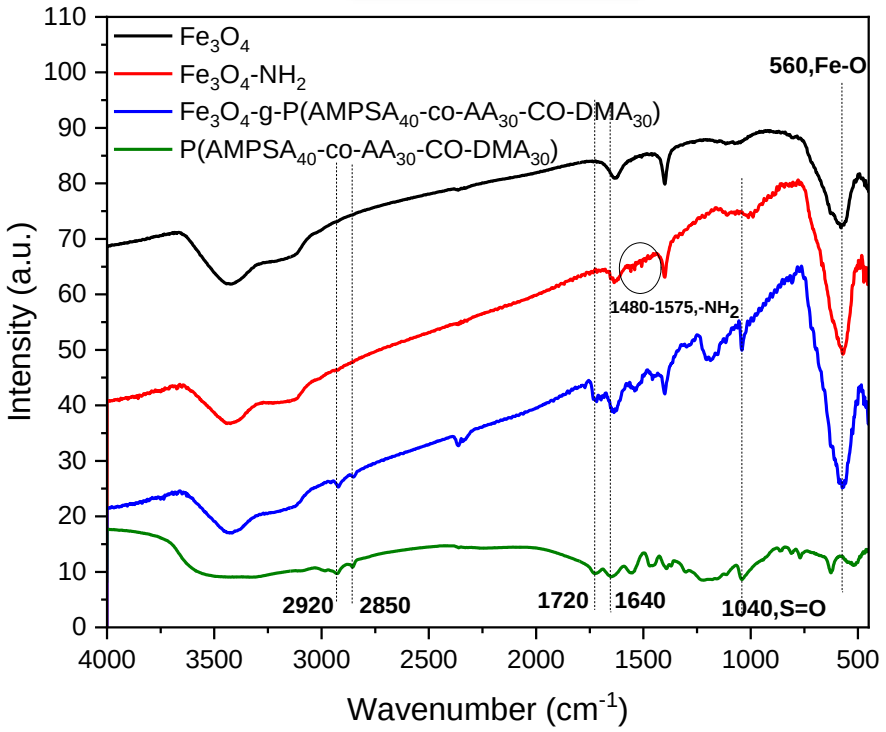
Polymer-coated nanoparticles

Polymer-coated nanoparticles (PNPs)

TGA



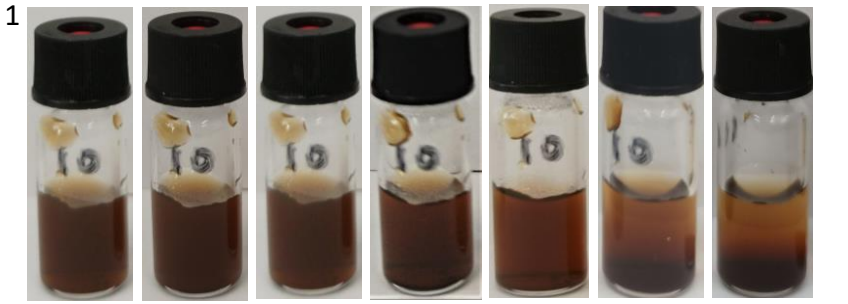
ATR-FTIR



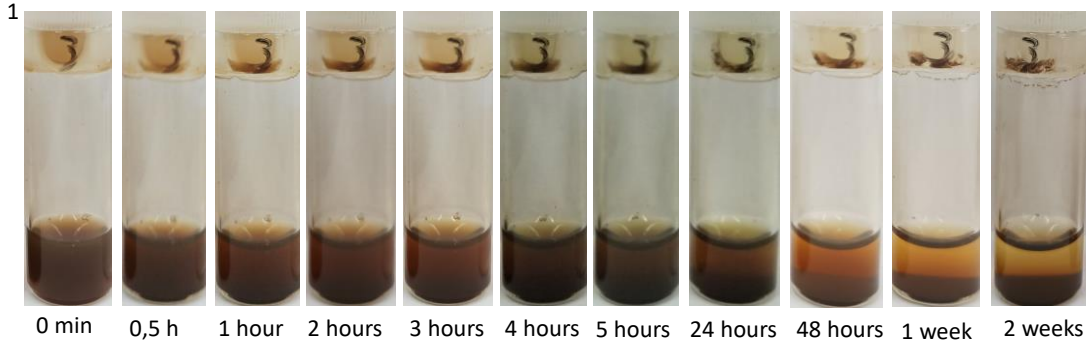
Polymer-coated nanoparticles

Stability of magnetic iron oxide particles in aqueous media

Fe_3O_4-g-P($AMPSA_{68}$-co-AA_{27}-co-DMA_5) in H_2O (C = 0,1 % w/v)



Fe_3O_4-g-P($AMPSA_{68}$-co-AA_{27}-co-DMA_5) in NaCl (C = 0,1 % w/v)



Conclusions

- Successful synthesis of amphiphilic copolymers
- Surface functionalization of SiO_2 or Fe_3O_4 NPs with double bonds or amine groups
- Synthesis of polymer-coated NPs by three methodologies:
  - Free radical polymerization of monomers on the acrylic-modified NPs surface
  - Post-grafting of terpolymers on amine-functionalized NPs, by carbodiimide chemistry
  - Polymerization of monomers through Cerium (IV) ammonium nitrate oxidizing agent
- Some polymer-coated NPs form stable aqueous dispersions

Acknowledgements

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment (Project title: Enhanced Oil Recovery by Polymer-coated Nano Particles, acronym: «EOR-PNP», code: HFRI-FM17-361)

Thank you for your attention!

Nano-colloid interfacial properties, and emulsion stabilization

Christina Ntente

Foundation for Research and Technology Hellas, Institute of Chemical Engineering Sciences (FORTH/ICE-HT), 26504 Patras, Greece

University of Patras, Department of Chemistry, 26504 Patras, Greece

UNIVERSITY OF PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

FORTH
FOUNDATION FOR RESEARCH AND TECHNOLOGY - HELLAS

H.F.R.I.
Hellenic Foundation for Research & Innovation

EORPNP

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Objectives

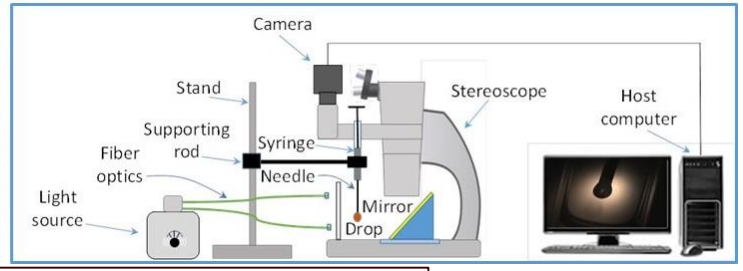
- Measurements of the ***surface and interfacial tensions*** of aqueous polymer and PNP – dispersions.
- Measurements of the ***contact angles*** *(CA)* on glass and PMMA surfaces.
- Preparation of the stabilized ***Pickering emulsions***.
- ***Rheology*** measurements of the Pickering emulsions.

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

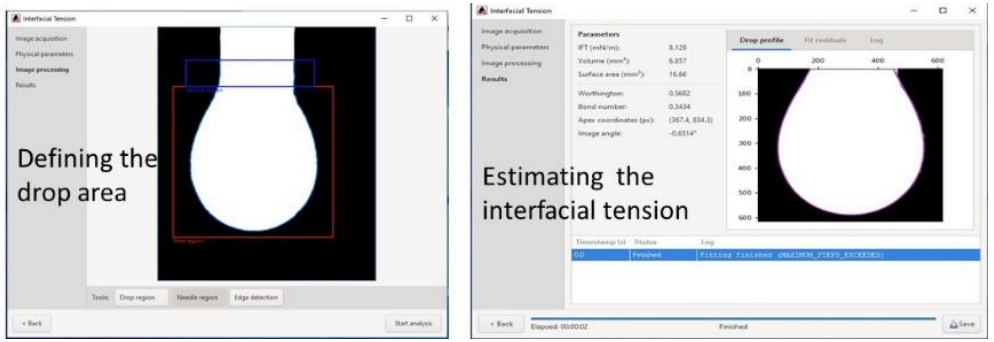
Surface Tension Characterization

Pendant Drop Method:

Dynamic Surface and Interfacial Tension measurements of nano-colloid suspensions.



*Experimental setup of **pendant drop method***



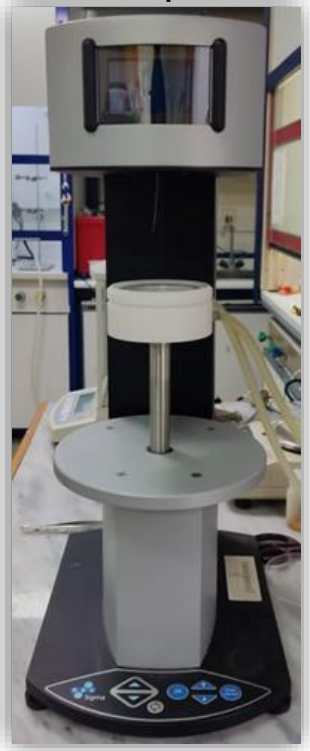
*Estimation of surface/interfacial tension by **OpenDrop software***

"Analyzing the recorded interfacial configurations with the open access OpenDrop software of inverse modeling of Young-Laplace equation"

Berry, J.D., et al. (2015). *J. Colloid Interface Sci.* 454: 226-237.

Du Nouy ring Method:

Static Surface and Interfacial Tension measurements of nano-colloid suspensions.

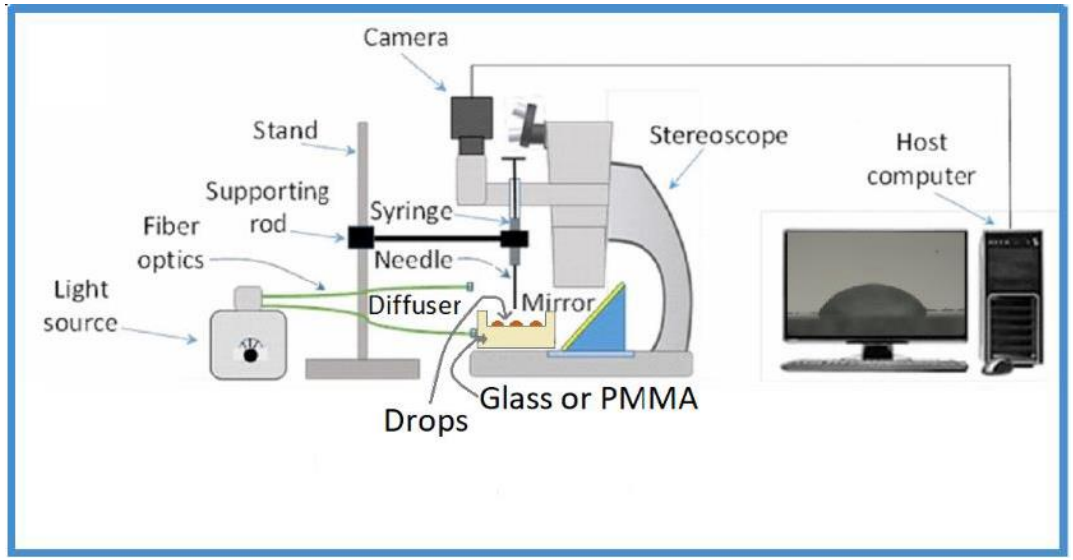


Tensiometer Sigma -702

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

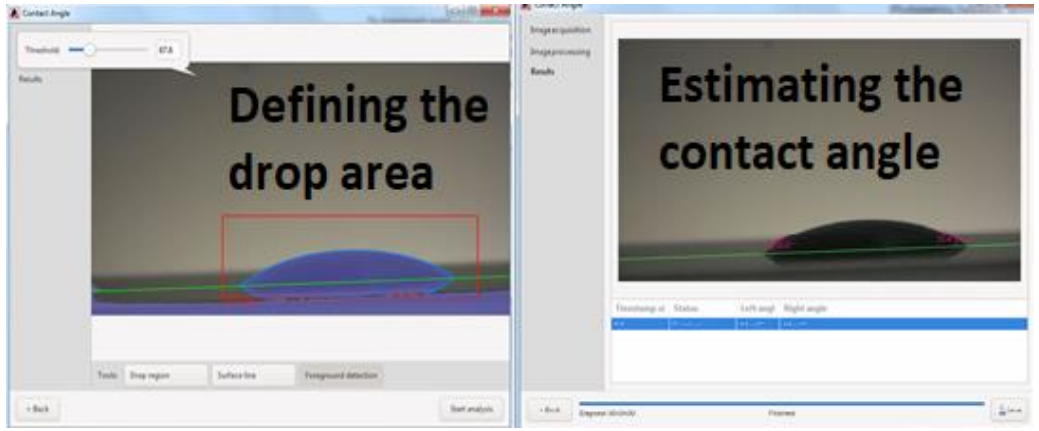
Wettability Characterization

Contact angles measurements of nano-colloid suspensions



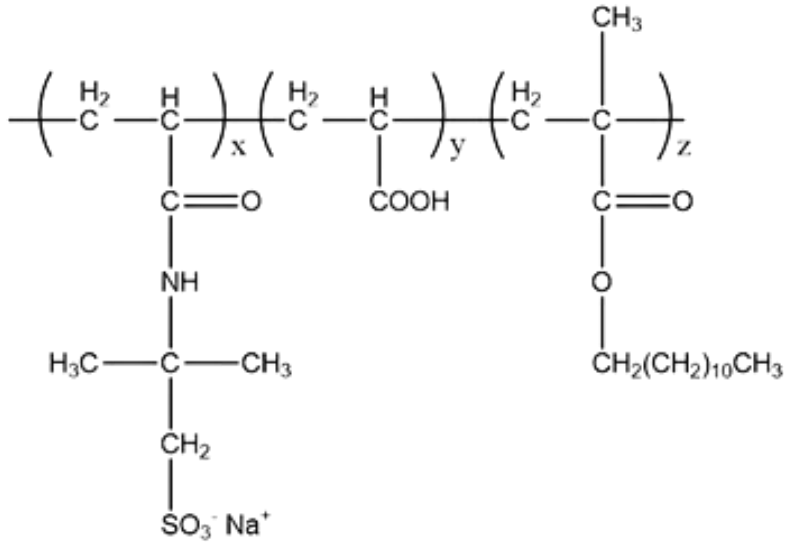
*Experimental setup of **contact angles measurements***

"Analyzing the recorded contact angles configurations with the open access OpenDrop software"



*Estimation of contact angle by **OpenDrop software***

Polymer Characterization

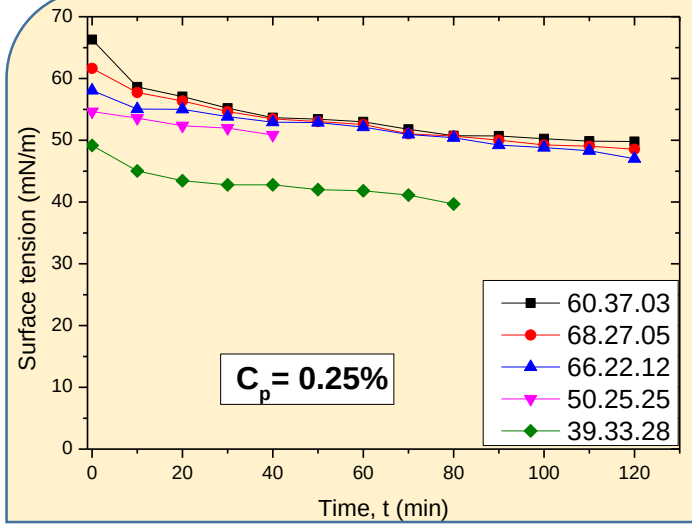


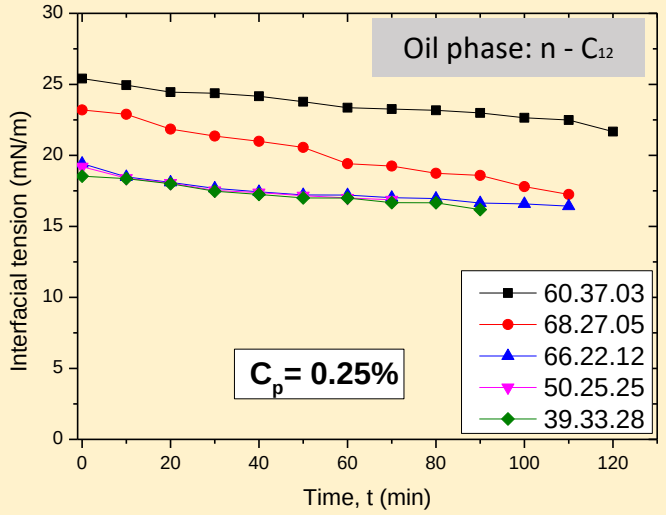
P(AMPSAx-co-AAy-co-DMAz):
Poly(2-Acrylamido-2-methylpropane sulfonic acid-co-acrylic acid-co-dodecyl methacrylate)

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Surface and Interfacial Tension Characterization

Dynamic surface (ST) and interfacial tension (IT) with n-C_{12} as function of time for various types of polymers (Cp = 0.25% w/v) in water and salt solutions.



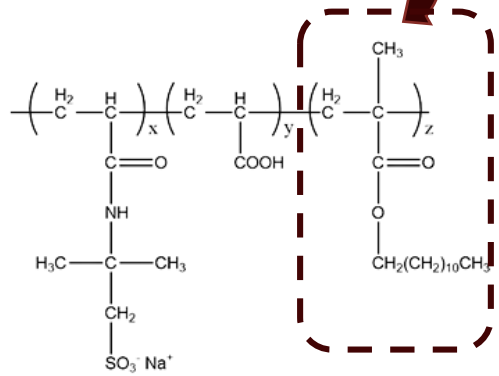


Maximized reduction rate of ST, IT for polymer types

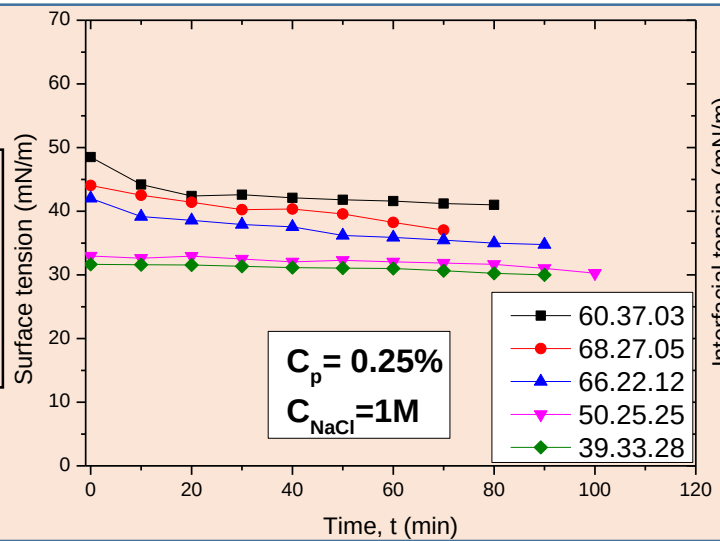
$P(AMPSA_{55} – co – AA_{20} - co – DMA_{25})$

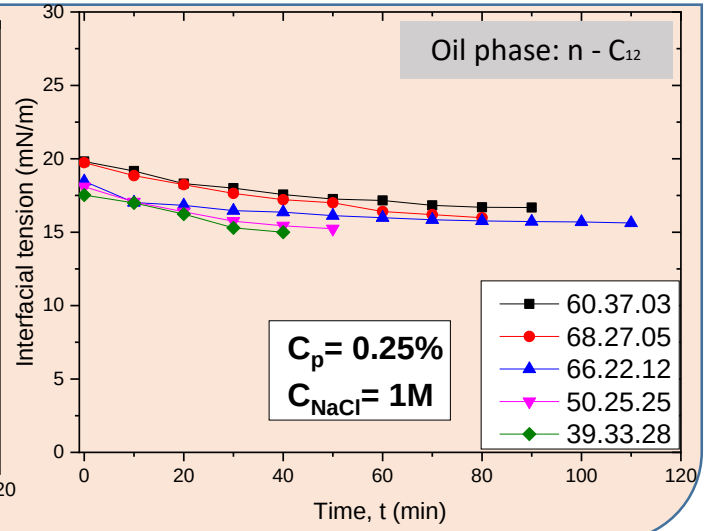
$P(AMPSA_{39} – co – AA_{33} - co – DMA_{28})$

Significant reduction at ST, IT as the ***percentage of DMA increases***



With the addition of salt ***NaCl 1.0M***, the ***ST, IT drop significantly***, due to the increased ionic strength.

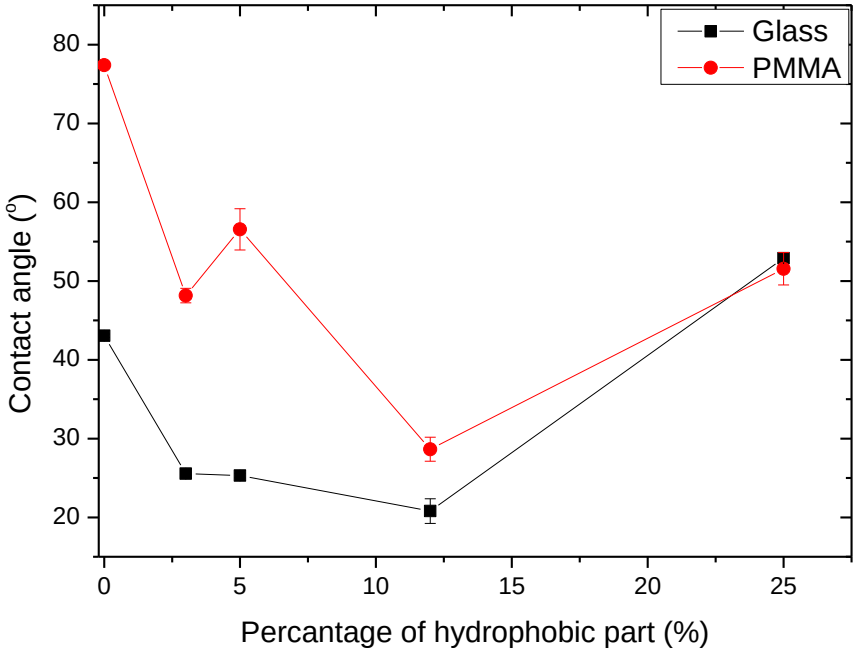




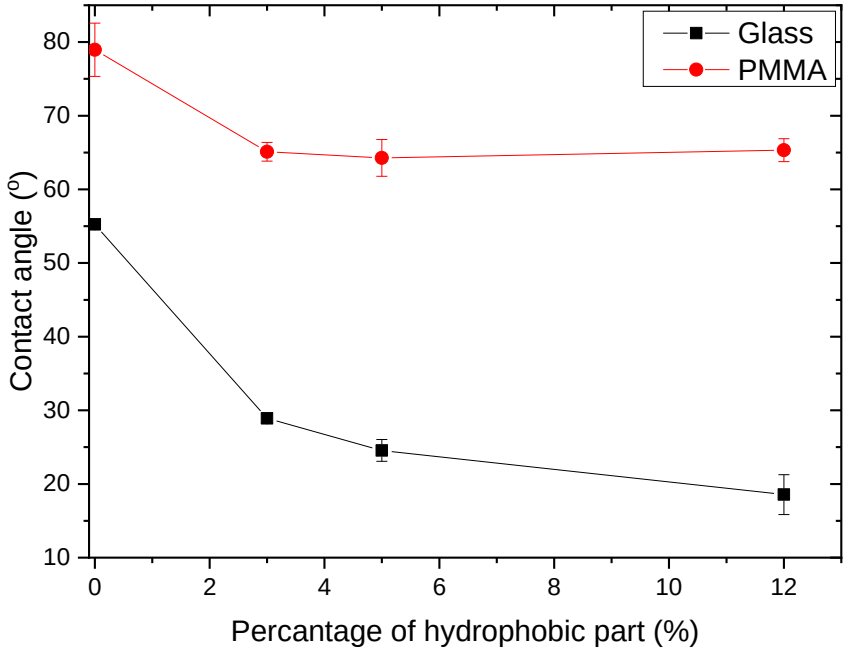
Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Wettability Characterization

CA as function of various types of polymers (Cp = 0.25% w/v), in air, on glass and PMMA surface.



CA as function of various types of polymers (Cp = 0.25% w/v), in oil (n-C_{12}), on glass and PMMA surface.



For all cases, CA < 90° , so ***polymers have moderate wettability***

- **Glass surface** shows further **hydrophilicity** for polymers with smaller percentage of DMA, in air.
- **PMMA surface** displays further **hydrophobicity** in the presence of higher percentage of DMA, in air and oil.

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Pickering emulsion

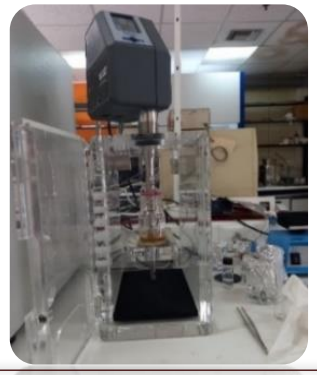
Formation of Pickering emulsions from different types of polymers in water and salt solutions

Static surface (ST) and interfacial tension (IT) with oil red n-C_{12} for different types of polymer solutions

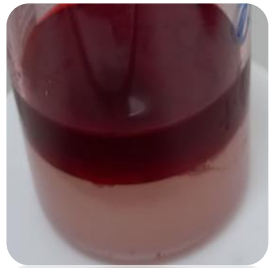
Types of polymers	ST ± STD (mN/m)	IFT with oil red n-C_{12} (mN/m)
66.22.12 (0.25% w/v – NaCl 1.0M)	47.46 ± 1.28	20.50
55.20.25 (0.25% w/v – water)	51.50 ± 1.45	24.46
55.20.25 (0.25% w/v – NaCl 1.0M)	40.77 ± 1.32	20.00

Preparation Process

Homogenizing dispersion with oil red n-C_{12} (c=1500 ppm) (volume rate: 2/1) in an ultrasound probe for 10min.



Ultrasound probe UP400St (Hielscher Ultrasonics GmbH)

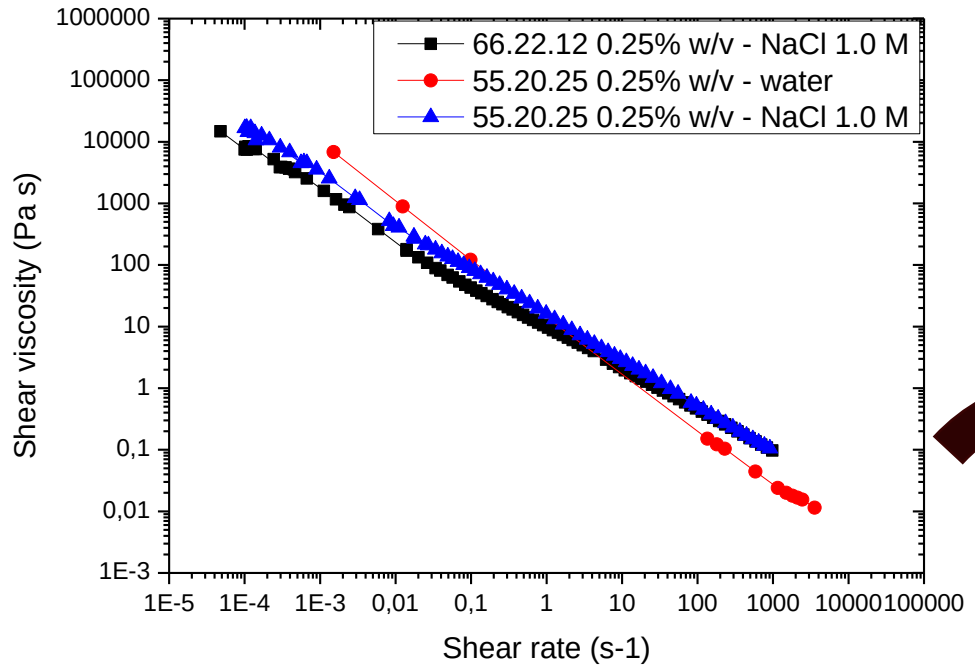


Immiscible phases

Pickering emulsion

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Emulsion Rheology



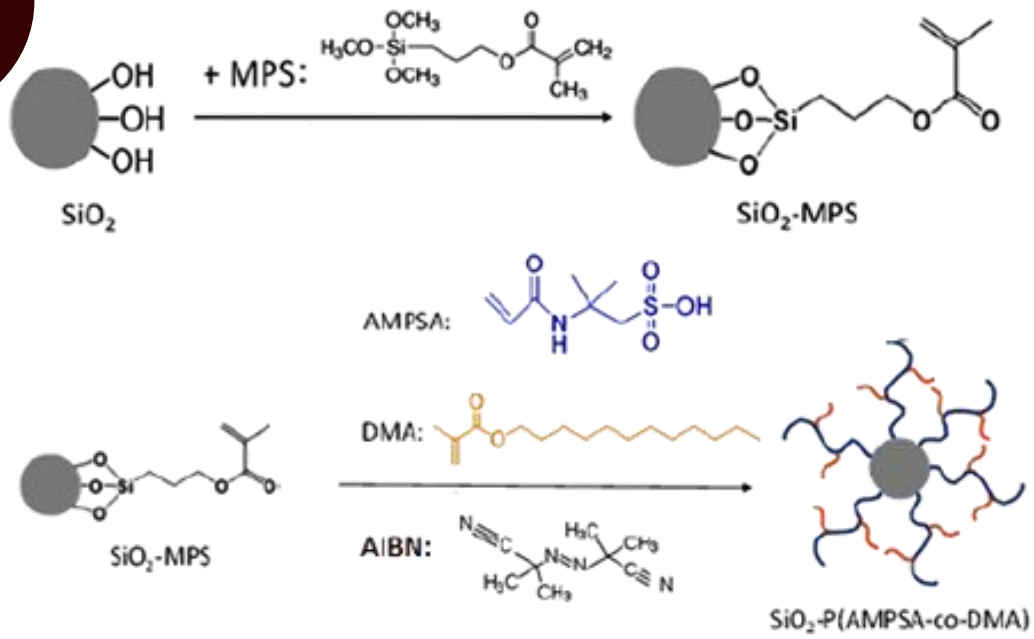
Power law model

$\mu = \mu_0 \gamma^{n-1}$

Emulsion	μ_0 (Pa s)	n
	17.44	0.20
	15.00	0.09
	16.00	0.25

The ***shear thinning rheology*** of the Pickering emulsions was fitted satisfactorily with the ***Power law model***

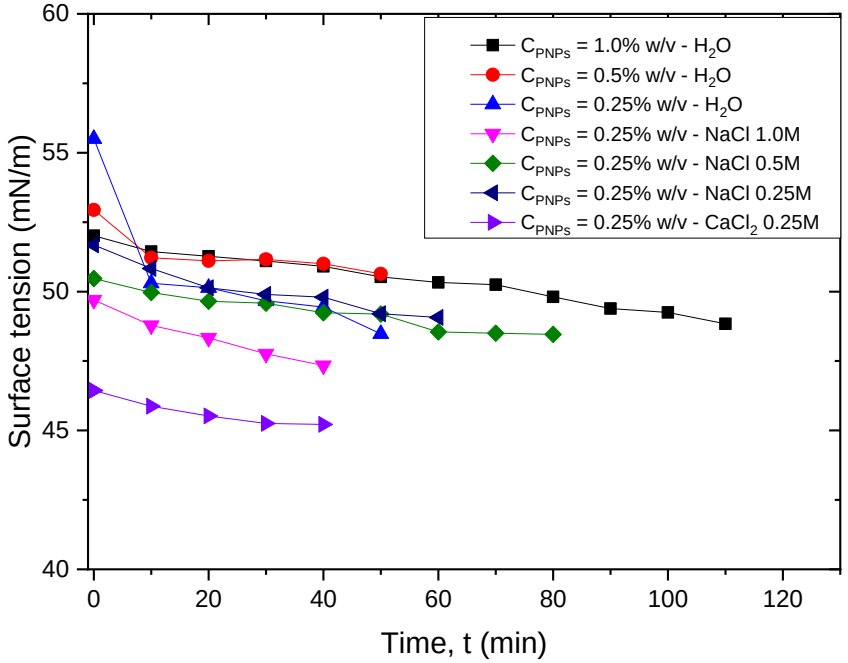
Polymer – coated nanoparticles (PNP) Characterization

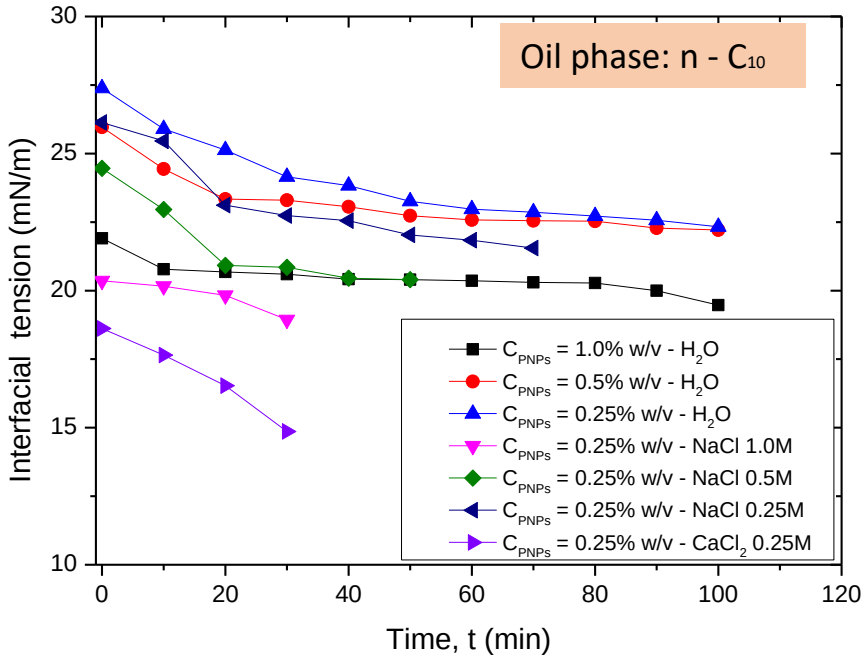


Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Surface and Interfacial Tension Characterization

Dynamic surface (ST) and interfacial tension (IT) with n-C_{10} as function of time for various concentrations of PNPs in water and salt solutions.



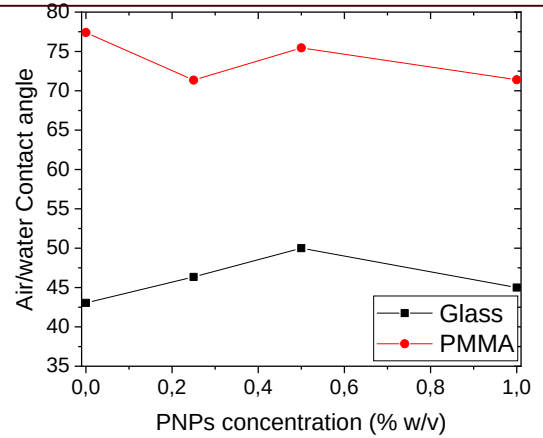


- ***Maximized reduction*** rate of ST, IT of SiO_2-P(AMPSA-co-DMA) NPs at PNP ***concentration equal to 0.25%.***
- with the addition of ***NaCl***, the ***ST, IT change weakly***.
- with the addition of ***$CaCl_2$***, the ***ST, IT drop significantly***, due to the **stronger electrostatic interactions** of the divalent $CaCl_2$ with the P(AMPSA-co-DMA) polyelectrolyte and **increased ionic strength**.

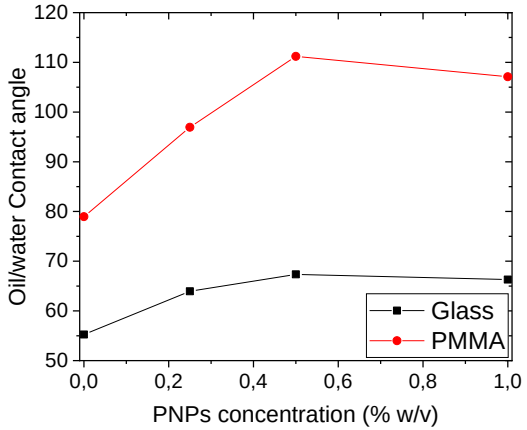
Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Wettability Characterization

CA as function of various concentrations of PNP (- water), in air and oil, on glass and PMMA surface.

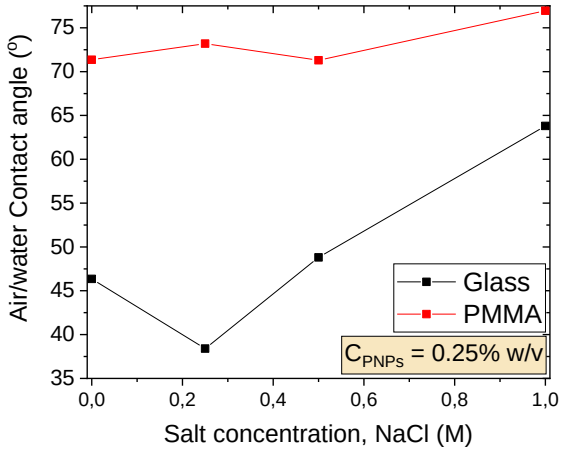


In the presence of PNP, ***increasing PNP*** concentration ***decreases CA on glass*** surfaces and ***increases on PMMA*** surfaces.

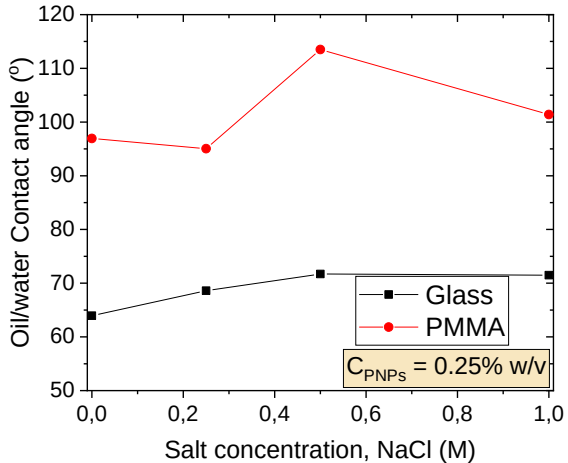


PNP have *moderate wetting on glass* surfaces (in the presence of air, oil) while they have *even less wetting on PMMA* surfaces.

CA as function of various concentrations of salt solutions (NaCl) for PNP (0.25% w/v), in air and oil, on glass and PMMA surface.



The presence of ***salts*** does not particularly affect on CA.



Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Pickering emulsion

Formation of Pickering emulsions from nano-colloid suspensions and salt solutions

Static surface (ST) and interfacial tension (IT) with n-C_{10} for PNP concentration 0.25% w/v at various salt concentrations

C_{PNPs} = 0.25% w/v – salt	ST ± STD (mN/m)	IFT PNPs με n-C_{10} (mN/m)
NaCl 1.0M	53.07 ± 0.46	26.88
NaCl 0.5M/$CaCl_2$ 0.25M	55.40 ± 0.21	33.25

Preparation Process

Homogenizing PNP – dispersion (*SiO_2-P(AMPSA-co-DMA 0.25%/salts solutions*) with n-C_{10} (1:1) in an ultrasound probe for 10min.

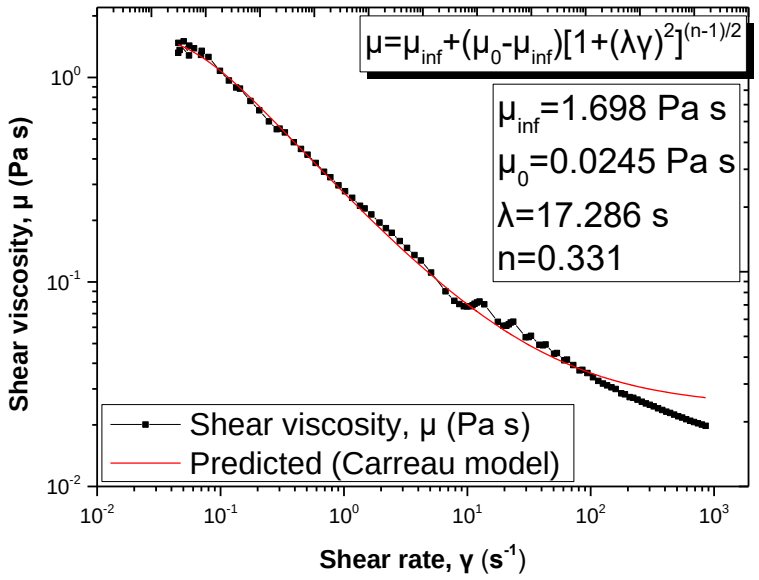
Immiscible phases

PNP – stabilized Pickering emulsion

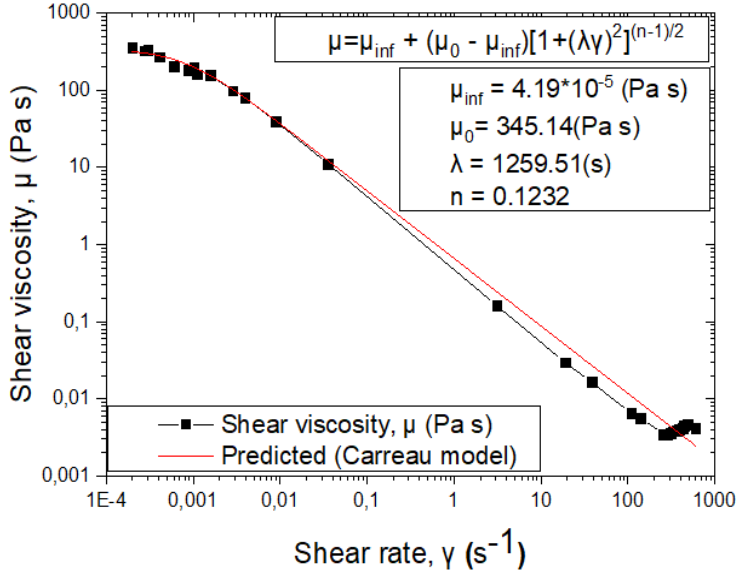
Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Emulsion Rheology

SiO_2-P(AMPSA-co-DMA) 0.25% -NaCl 0.5M / $CaCl_2$ 0.25M



SiO_2-P(AMPSA-co-DMA) 0.25% - NaCl 1.0M



The ***shear thinning rheology*** of the Pickering emulsion was fitted satisfactorily with the ***Carreau model***

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

Conclusions

Polymers

- ✓ **Surface and Interfacial tension** depend on:
  - the *percentage of* hydrophobic part *DMA*.
  - the *presence of salt*.
- ✓ **Polymers** have
  - *good wettability* on *glass* surface.
  - *moderate wettability* on *PMMA* surface.
- ✓ Formation of **stable Pickering emulsions** was fitted satisfactorily with *the Power law model*.

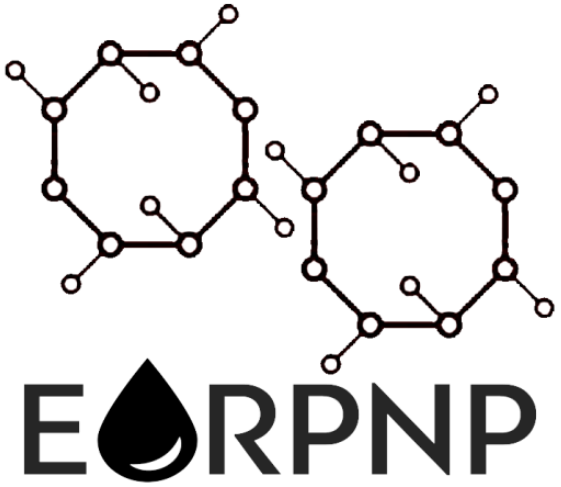
Polymer coated – nanoparticles (PNP)

- ✓ **Surface and Interfacial tension** depend on the *concentration of* SiO_2-P(AMPSA-co-DMA) *PNP*
  - with the addition of *NaCl*, the *ST, IT change weakly*.
  - with the addition of *$CaCl_2$*, the *ST, IT drop significantly*.
- ✓ **PNP** have
  - *moderate wettability* on *glass* surface
  - *even less wettability* on *PMMA* surface.
- ✓ Formation of **stable Pickering emulsions** was fitted satisfactorily with *the Carreau model*.

Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)

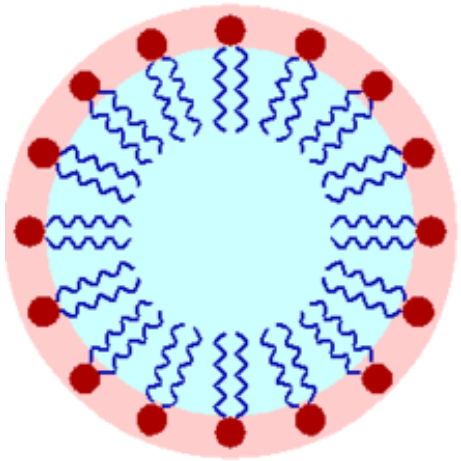
Acknowledgments

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment" (Project Number: HFRI-FM17-361, Title: Enhanced oil recovery by polymer-coated nanoparticles, Acronym: EOR-PNP).

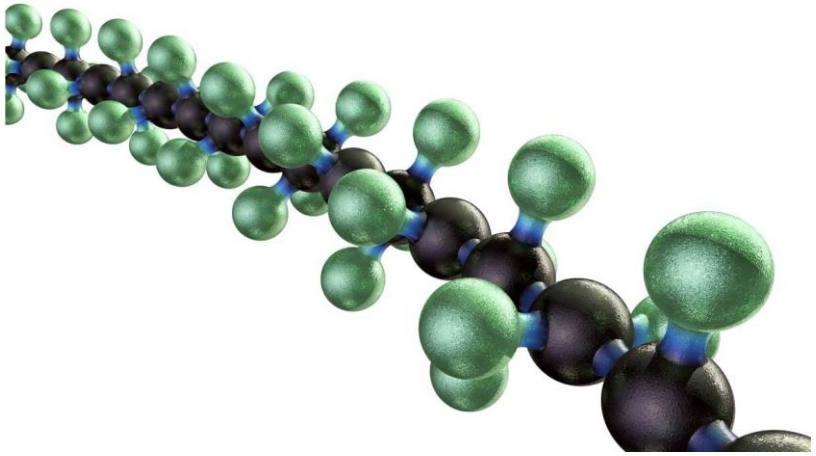


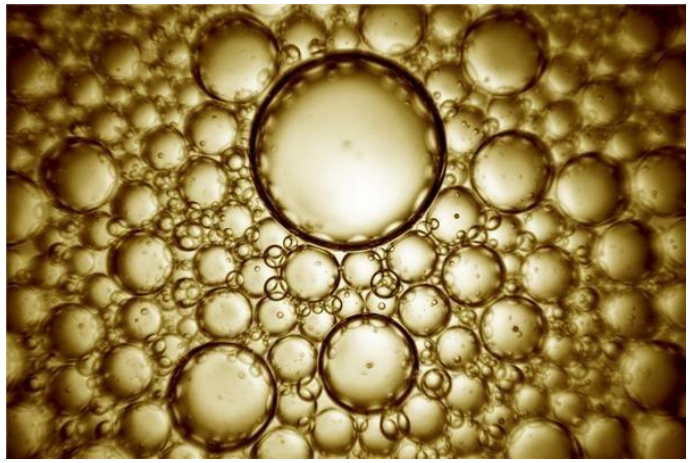
H.F.R.I.
Hellenic Foundation for Research & Innovation

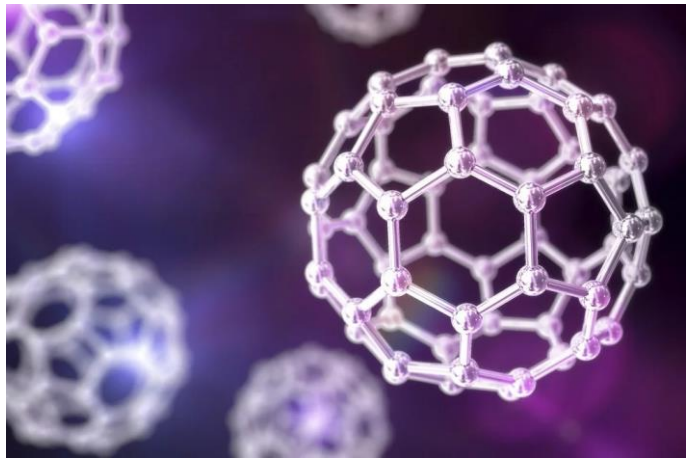
Enhanced oil recovery by polymer – coated nanoparticles (EOR-PNP)



Thank you for your attention!!!







INSTITUTE OF CHEMICAL ENGINEERING SCIENCES

Workshop: "Advances Toward the Transport of Nanoparticles in Porous Media and Applications to Residual Oil Recovery"

Visualization EOR studies on glass-etched pore networks

A. Strekla [1,2], Ch. Ntente [1,3], M. Theodoropoulou [1] and Ch. Tsakiroglou[1,*]

[1] Foundation for Research and Technology Hellas, Institute of Chemical Engineering Sciences (FORTH/ICE-HT), 26504 Patras, Greece

[2] University of Patras, Department of Physics, 26504 Patras, Greece

[3] University of Patras, Department of Chemistry, 26504 Patras, Greece

* ctsakir@iceht.forth.gr

Introduction

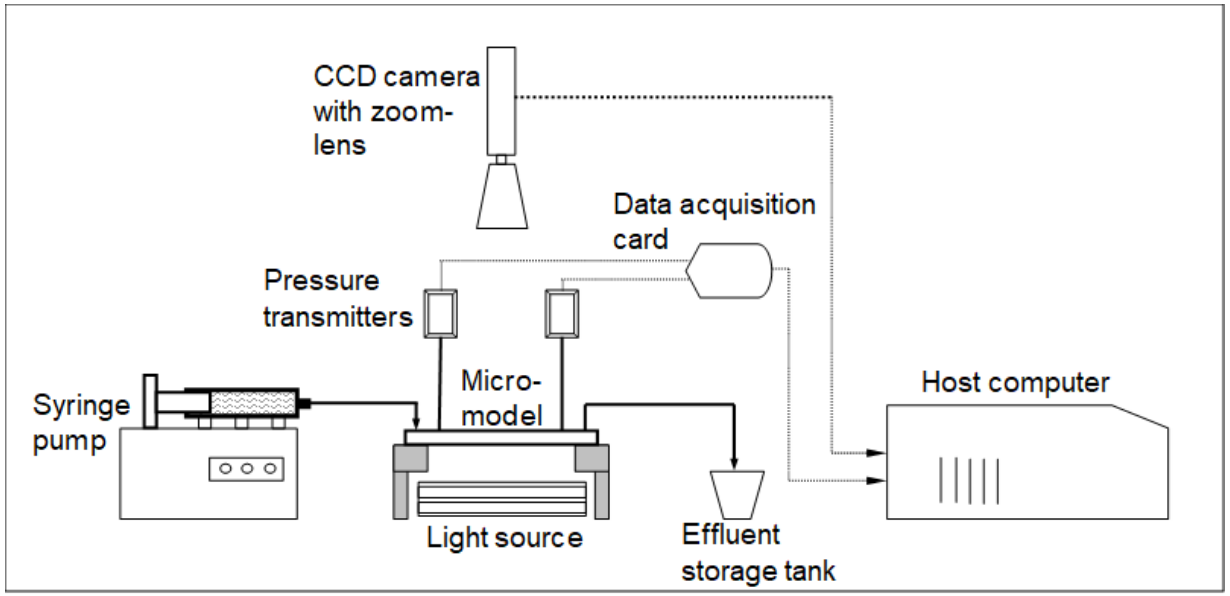
Problem

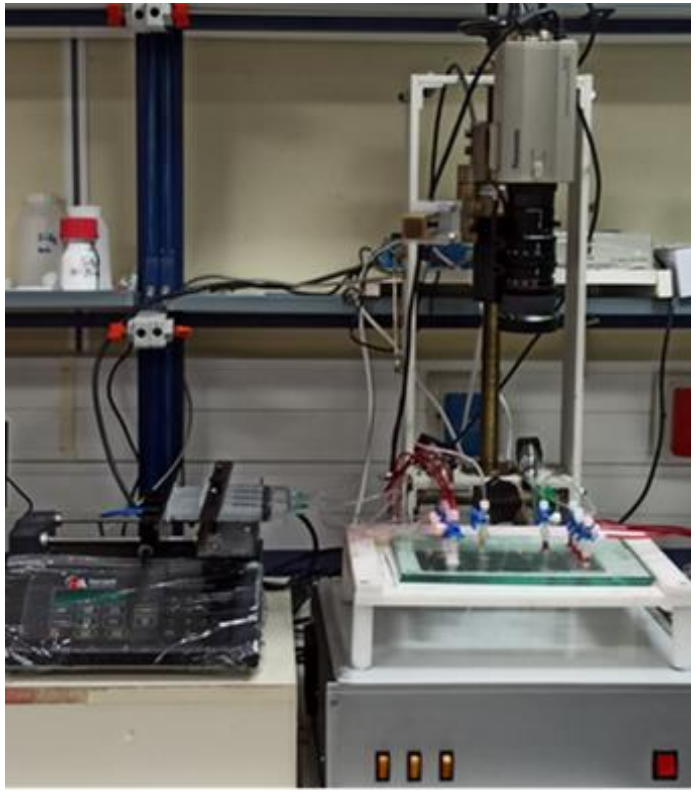
- Energy consumption worldwide is expected to increase by 50% relative to current levels by the end of 2030.
- This growth is unlikely to be met by renewable resources, and thus there is a strong and growing demand for oil as a predominant energy resource.

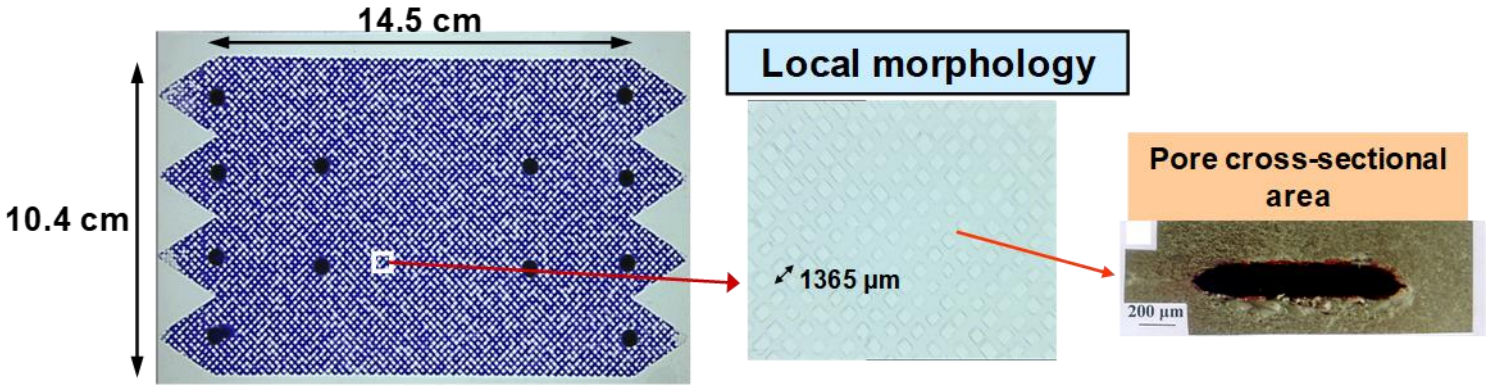
Objectives

- Globally the overall oil recovery factors for primary and secondary recovery range from 35% to 45%.
- A tertiary recovery method can enhance the recovery factor by 10-30%, which could contribute to energy supply.
- The use of nanoparticles in enhanced oil recovery (EOR) processes comprises an emerging and well-promising approach.

Experimental setup of visualization tests on transparent pore networks







Values of Ca and κ

Capillary number $Ca = \frac{\mu_{displacing}\, U}{IT}$

Ca pore scale -> $\mathbf{Ca_{L1}=Ca*factor}$

$$\mathbf{factor} = \frac{L_P r_H}{k}$$

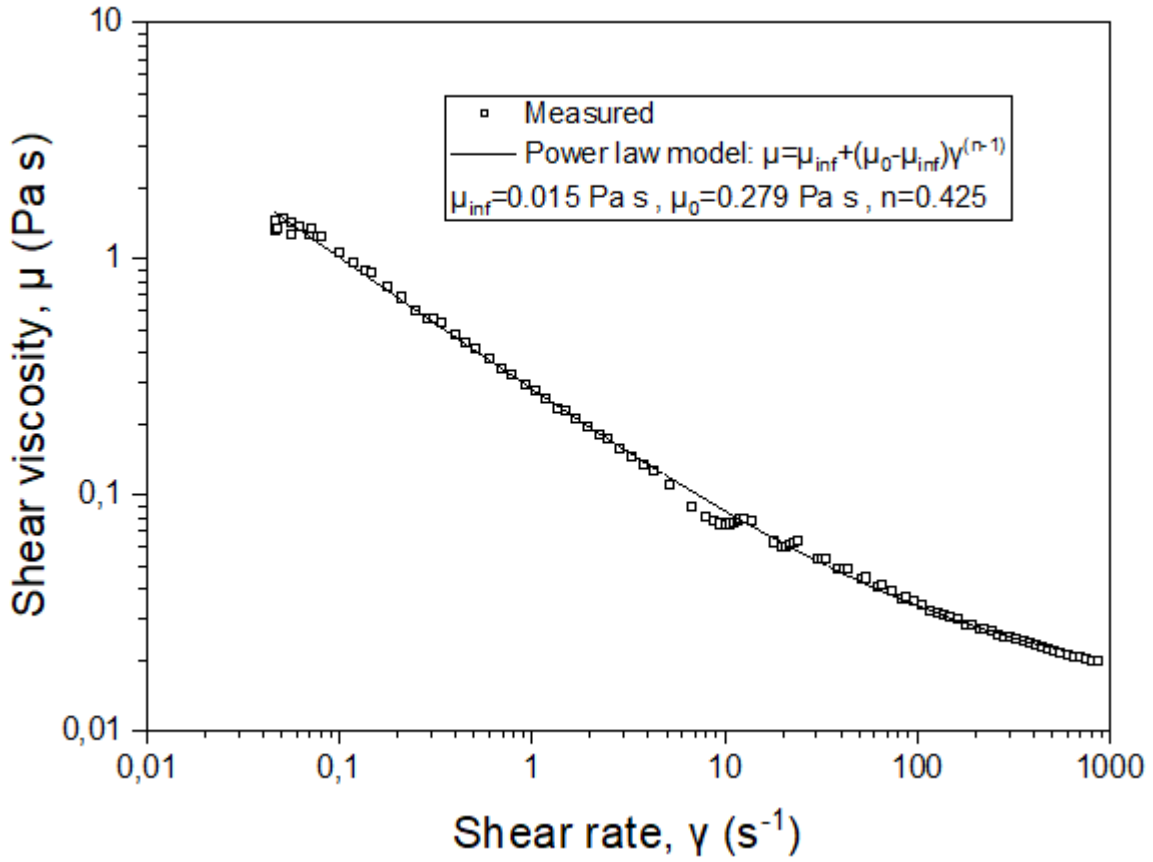
Ca network scale -> $\mathbf{Ca_{LN}=Ca_{L1}*L_N/L_p}$

Tsakiroglou et al., AIChE J. 49, 2472 (2003)

Viscosity ratio $\kappa = \frac{\mu_{displacing}}{\mu_{displaced}}$

Shear viscosity of emulsions

$$\mu = \mu_{inf} + (\mu_1 - \mu_{inf})\dot{\gamma}^{n-1}$$



Rheology of **SiO2=AMPSA-DMA-0.25%** + **NaCl 0.5M-CaCl2 0.25M** emulsion

Pore level to network scale:

$$\langle\mu\rangle = \mu_{inf} + \left(\frac{\mu_1 - \mu_{inf}}{n}\right)\gamma_w{}^{n-1}$$

$$\gamma_w = \left(\frac{8u_p}{4r_H}\right)\left(\frac{3n+1}{4n}\right)$$

$$u_p = \frac{u_0}{\varphi_V}$$

$$\varphi_V = \frac{\pi\langle W_p\rangle\langle D_p\rangle}{4L_p{}^2}$$

Tsakiroglou., J. Non-Newtonian Fluid Mechanics 105 (2002)

Values of Ca and κ

fluid	Flow rate(10^{-6}/60) m3/s	U (10^{-5} m/s)	μ (Pa s)	IT (mN/m)	Ca (10^{-6})	κ
Old paraffin oil	0.08	0.94	0.02	30.5	6.1	20.0
New paraffin oil	0.08	0.94	0.1	40.76	23.0	100.0
NaCl 1M-old oil	0.2	2.35	0.001	34.96	0.67	0.05
NaCl 1M-new oil	0.2	2.35	0.001	27.76	0.84	0.01
DISPERSIONS						
SiO2=AMPSA-DMA-0.25%+NaCl 1M	0.2	2.35	0.001	20.89	1.12	0.05
SiO2=AMPSA-DMA-0.25% + NaCl 0.5M-CaCl2 0.25M	0.2	2.35	0.001	30.51	0.77	0.05
AMPSA-55.20.25-0.25%-water	0.2	2.35	0.001	27.05	0.87	0.01
AMPSA-55.20.25-0.25% + NaCl 1M	0.2	2.35	0.001	12.65	1.85	0.05

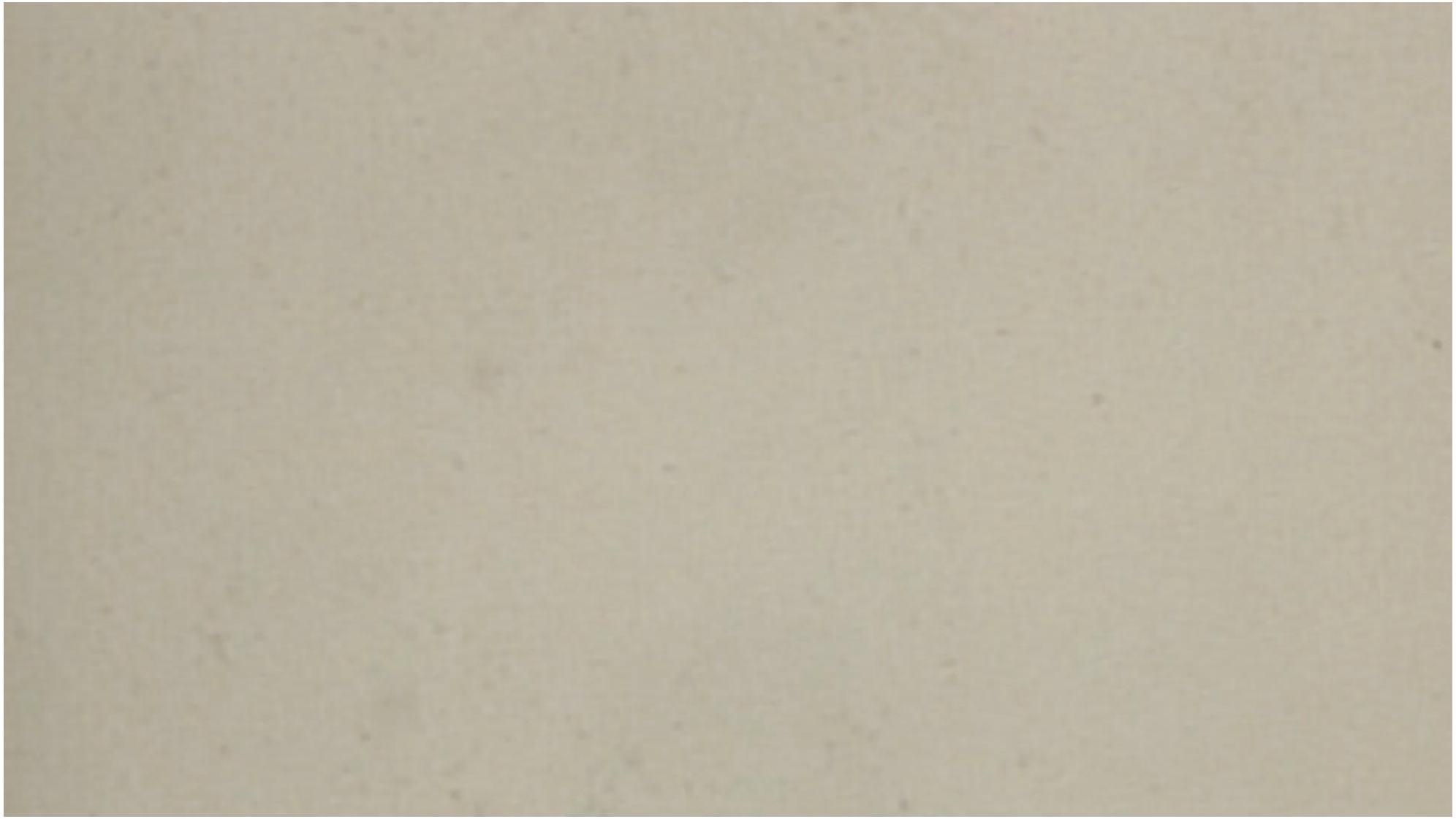
Values of Ca and κ

fluid	Flow rate(10^{-6}/60) m3/s	U (10^{-5} m/s)	μ (Pa s)	IT (mN/m)	Ca (10^{-5})	κ
Old paraffin oil	0.08	0.94	0.02	30.5	0.61	20.0
New paraffin oil	0.08	0.94	0.1	40.76	2.30	100.0
EMULSIONS						
SiO2=AMPSA-DMA-0.25%+NaCl 1M	0.2	2.35	0.051	20.89	5.73	2.55
SiO2=AMPSA-DMA-0.25% + NaCl 0.5M-CaCl2 0.25M	0.2	2.35	0.077	30.51	5.93	3.83
AMPSA-55.20.25-0.25%-water	0.2	2.35	0.014	27.05	1.21	0.14
AMPSA-55.20.25-0.25% + NaCl 1M	0.2	2.35	0.014	12.65	2.60	0.7

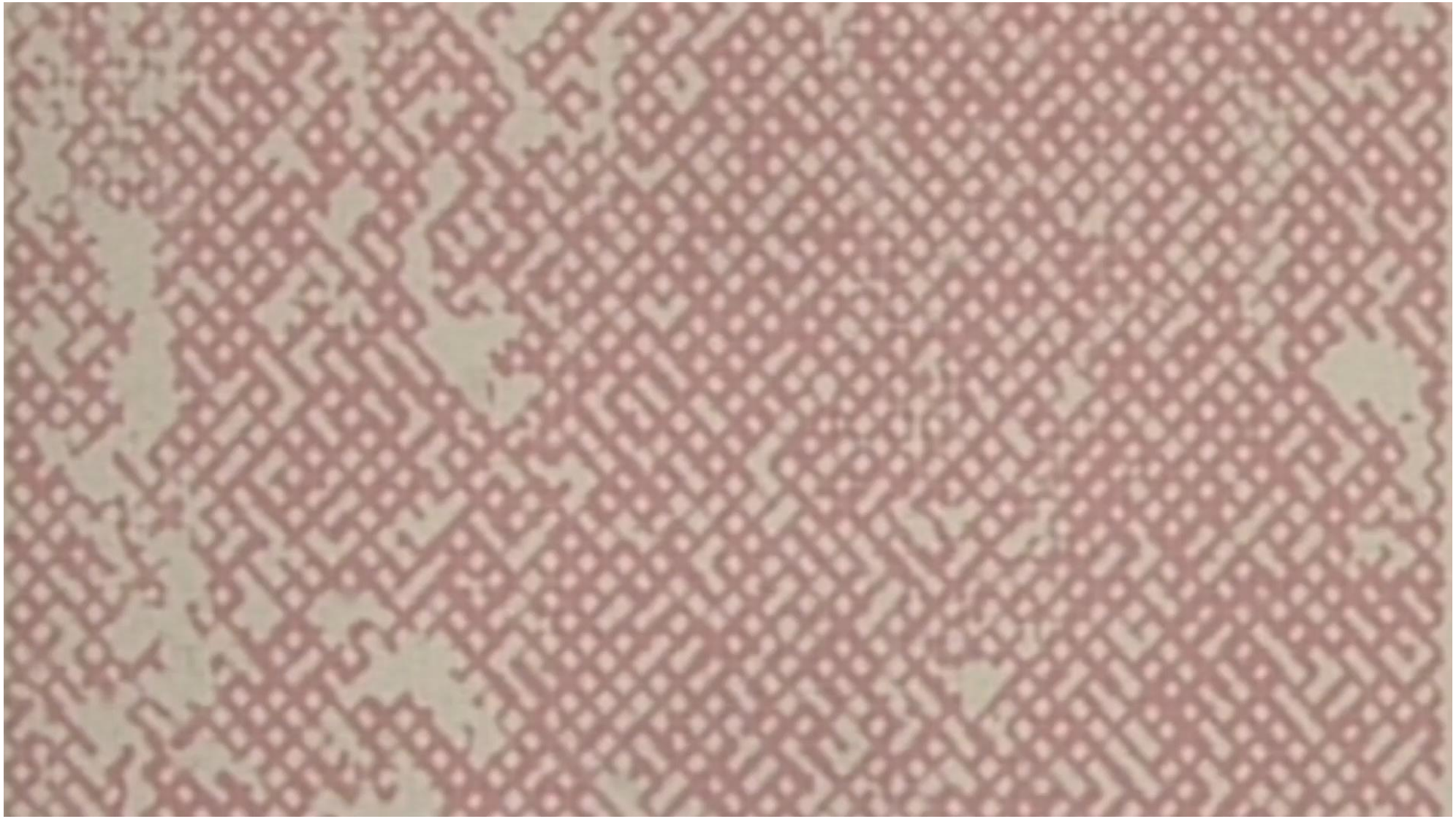
Ca to pore and network scale

fluid		Ca	pore scale Ca	network scale Ca
Old paraffin		$6.1x10^{-6}$	0.018	1.8
New paraffin		$23.0x10^{-6}$	0.069	6.9
NaCl 1M – old oil		$0.67x10^{-6}$	0.0020	0.2
NaCl 1M- new oil		$0.84x10^{-6}$	0.0025	0.25
SiO2=AMPSA-DMA-0.25%+NaCl 1M	dispersion	$1.12x10^{-6}$	0.0033	0.33
	emulsion	$5.73x10^{-5}$	0.172	17.2
SiO2=AMPSA-DMA-0.25% + NaCl 0.5M-CaCl2 0.25M	dispersion	$0.77x10^{-6}$	0.0023	0.23
	emulsion	$5.93x10^{-5}$	0.178	17.8
AMPSA-55.20.25-0.25%-water	dispersion	$0.87x10^{-6}$	0.0026	0.26
	emulsion	$1.21x10^{-5}$	0.036	3.6
AMPSA-55.20.25-0.25% + NaCl 1M	dispersion	$1.85x10^{-6}$	0.0055	0.55
	emulsion	$2.60x10^{-5}$	0.078	7.8

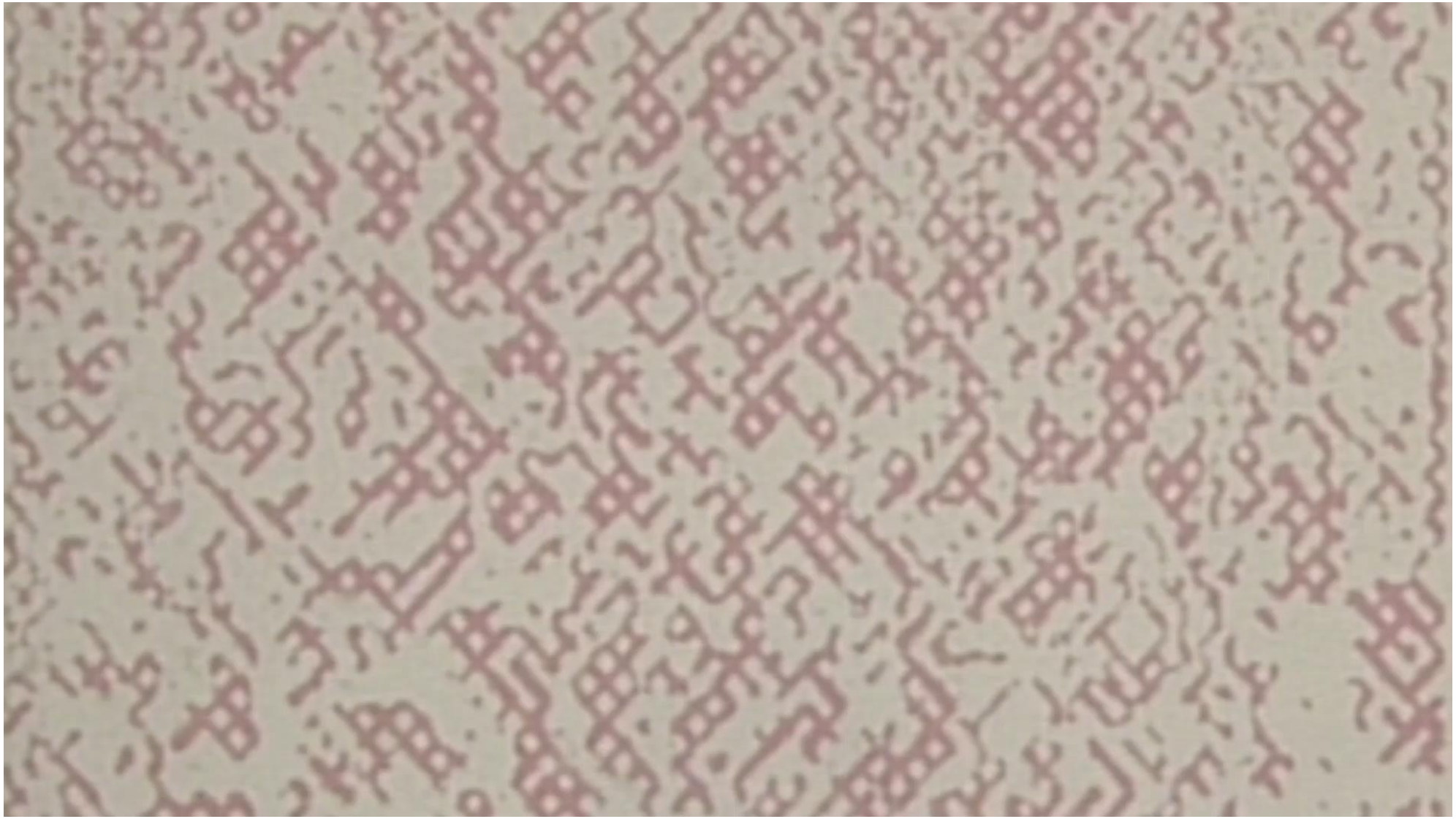
Primary Drainage - Displacement with paraffin oil



Primary Imbibition – Displacement with NaCl 0.5M + CaCl2 0.25M

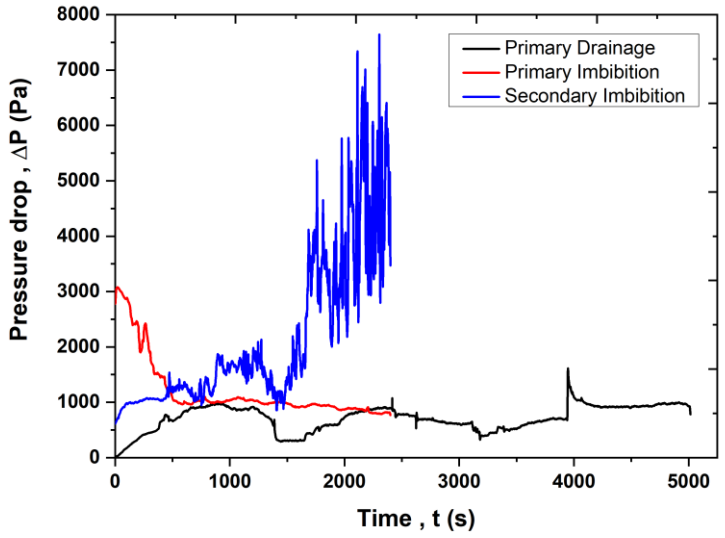


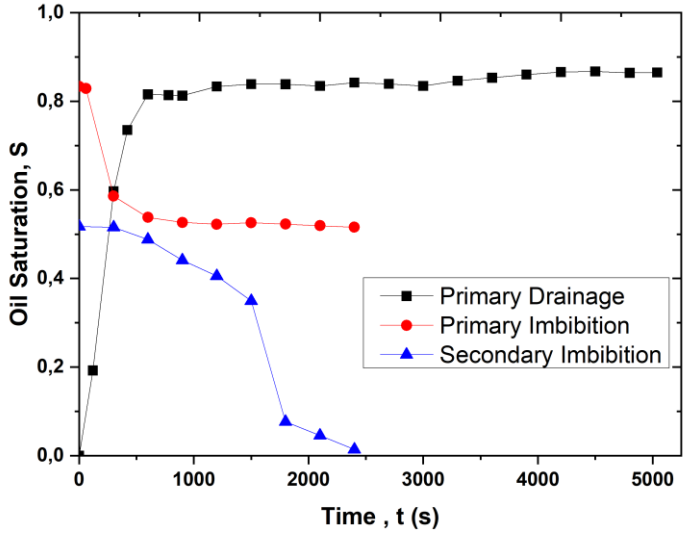
Secondary Imbibition – Displacement with emulsion of SiO2-AMPSA=DMA 0.25% - NaCl 0.5M + CaCl2 0.25M



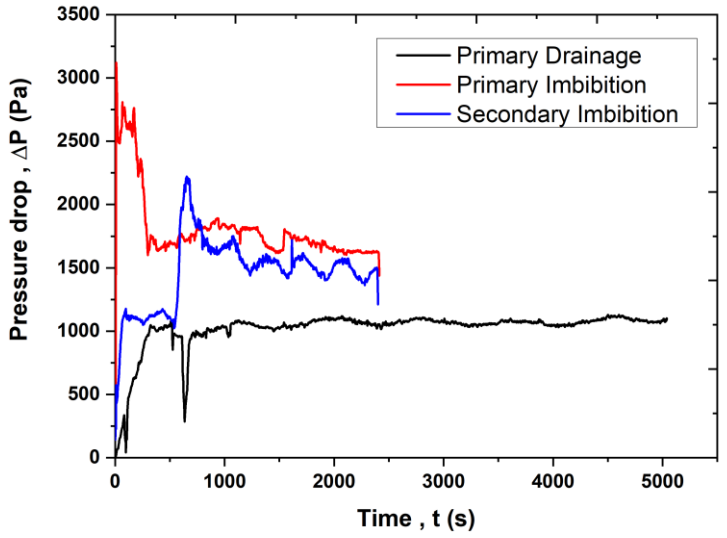
SiO2-AMPSA=DMA 0.25% -NaCl 0.5M+ CaCl2 0.25M

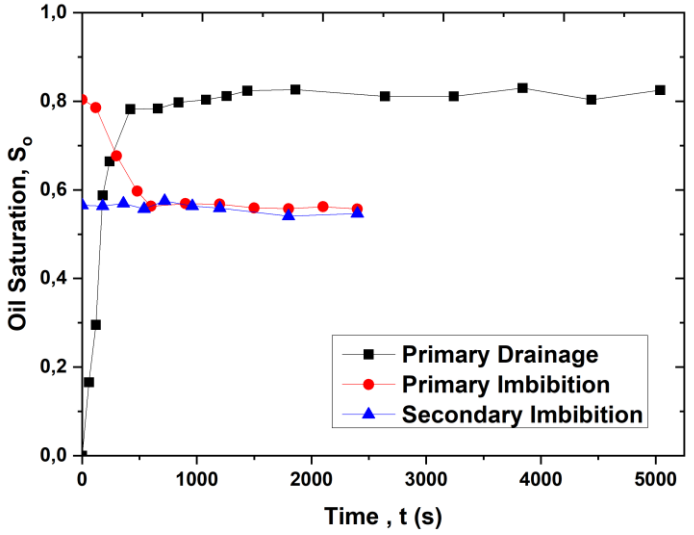
emulsion





dispersion



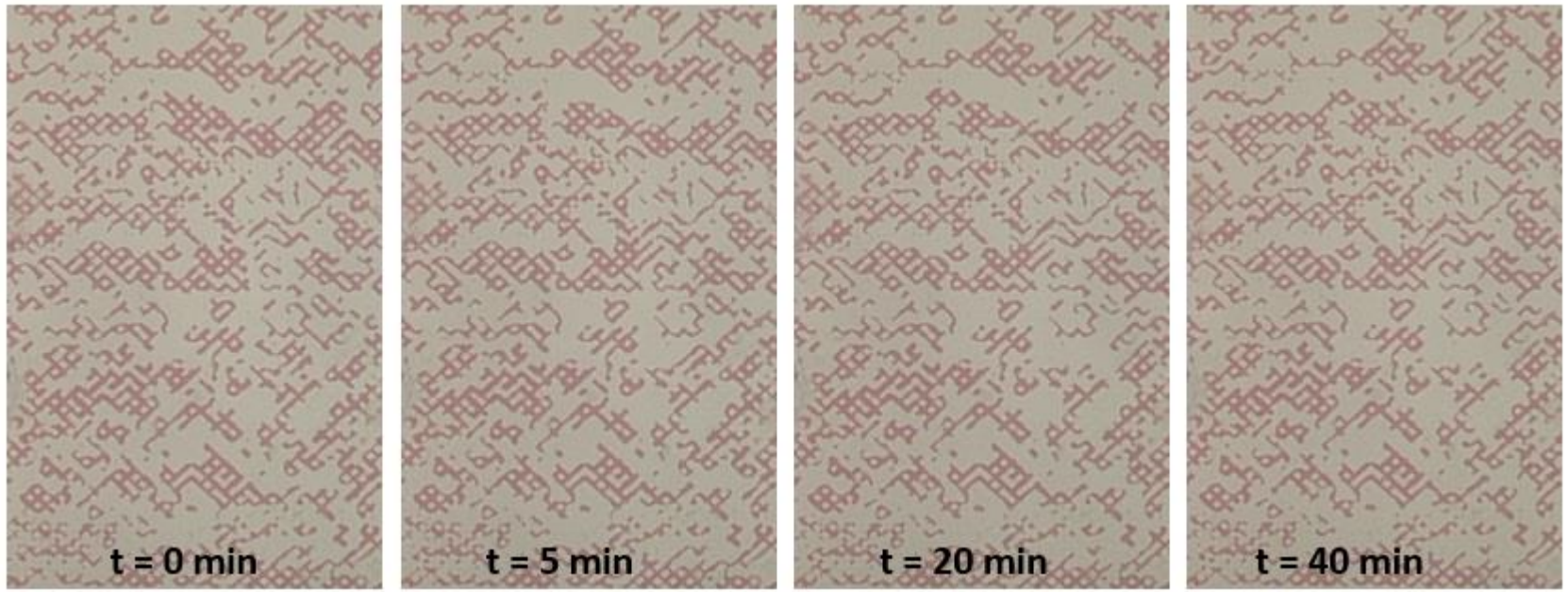


Results 1

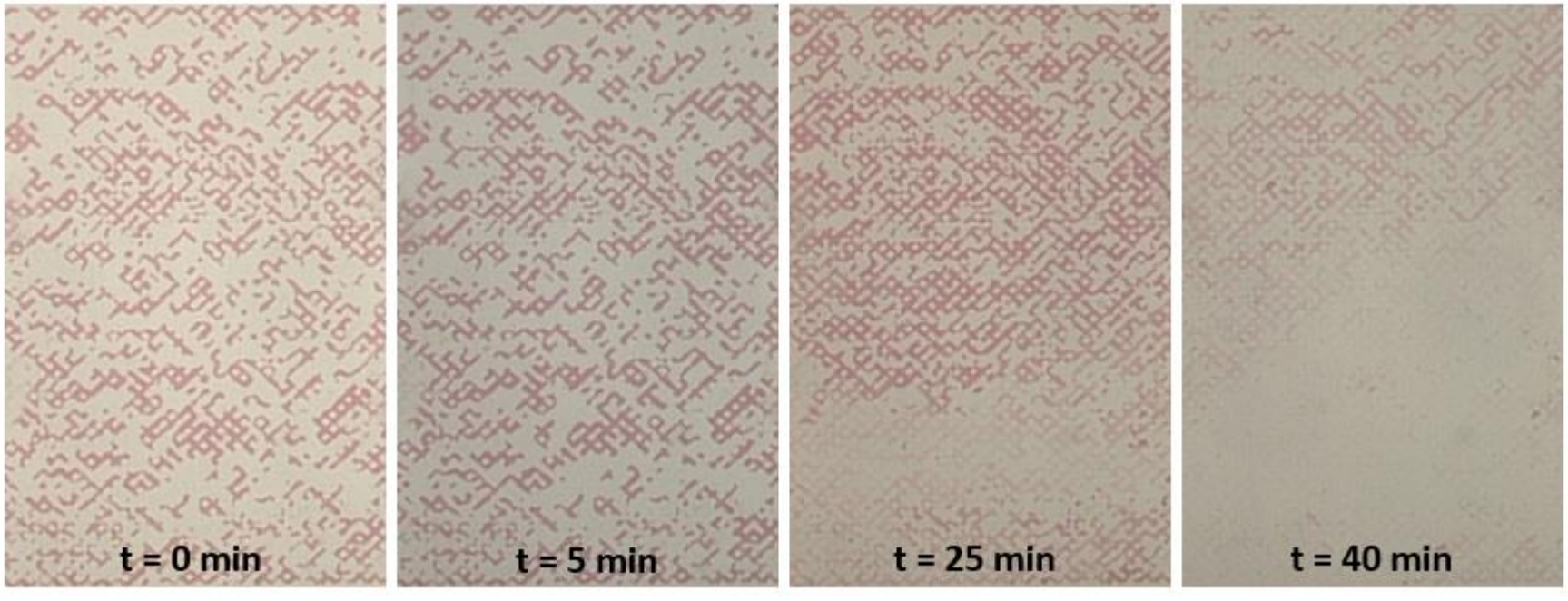
SiO2=AMPSA-DMA-0.25% + NaCl 0.5M - CaCl2 0.25M						
Type of displacement	Displaced fluid	Displacing fluid	Flow rate (mL/min)	Injected volume (mL)	Oil saturation	Oil removal efficiency (%)
Drainage	NaCl 0,5M - CaCl2 0,25M	-	0.08	8	0.82	-
Prim. Imbib.	Resid. paraffin oil	NaCl 0,5M - CaCl2 0,25M	0.2	8	0.56	30.2
Sec. Imbib.	Resid. paraffin oil	Dispersion	0.2	8	0.55	1.7
Drainage	NaCl 0,5M - CaCl2 0,25M	-	0.08	8	0.86	-
Prim. Imbib.	Resid. paraffin oil	NaCl 0,5M - CaCl2 0,25M	0.2	8	0.52	39.5
Sec. Imbib.	Resid. paraffin oil	Emulsion	0.2	8	0.014	97.3

Secondary Imbibition **SiO2-AMPSA=DMA 0.25% - NaCl 1M**

Dispersion

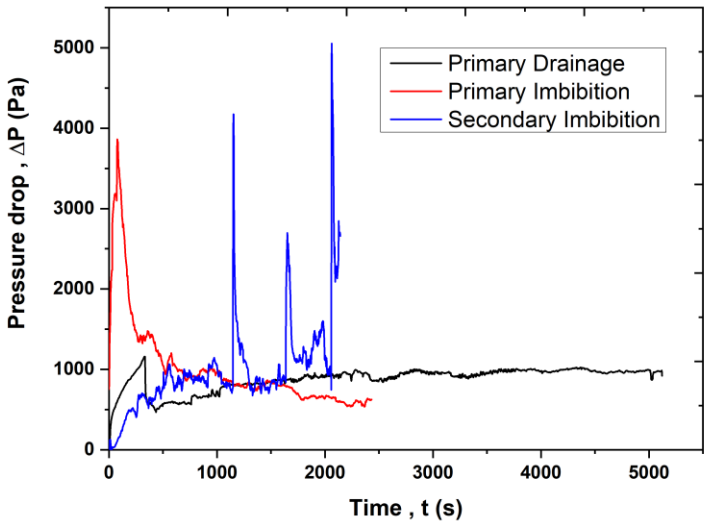


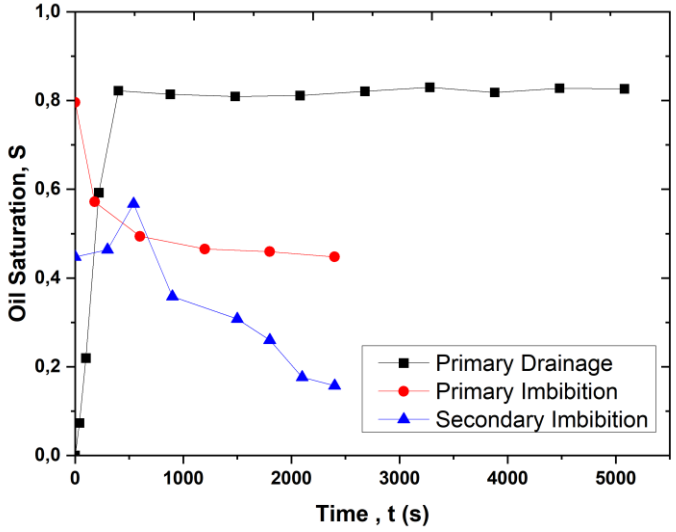
Emulsion



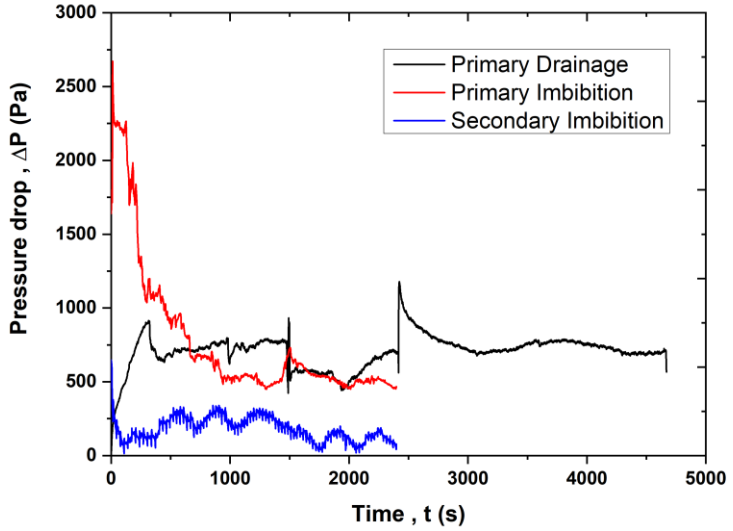
SiO2-AMPSA=DMA 0.25% - NaCl 1M

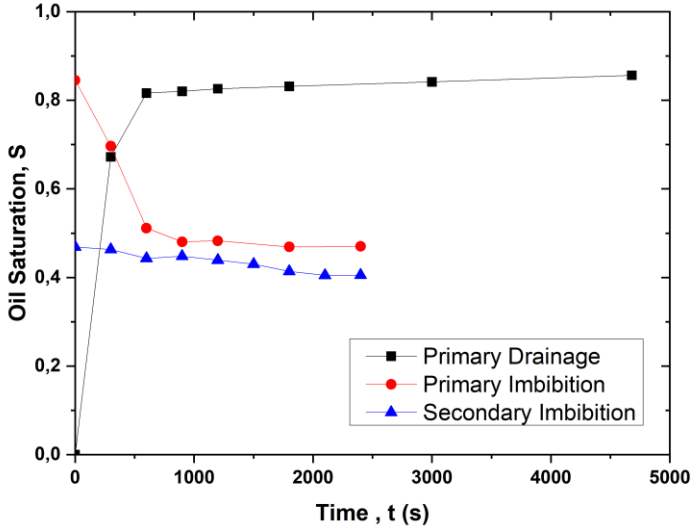
emulsion





dispersion





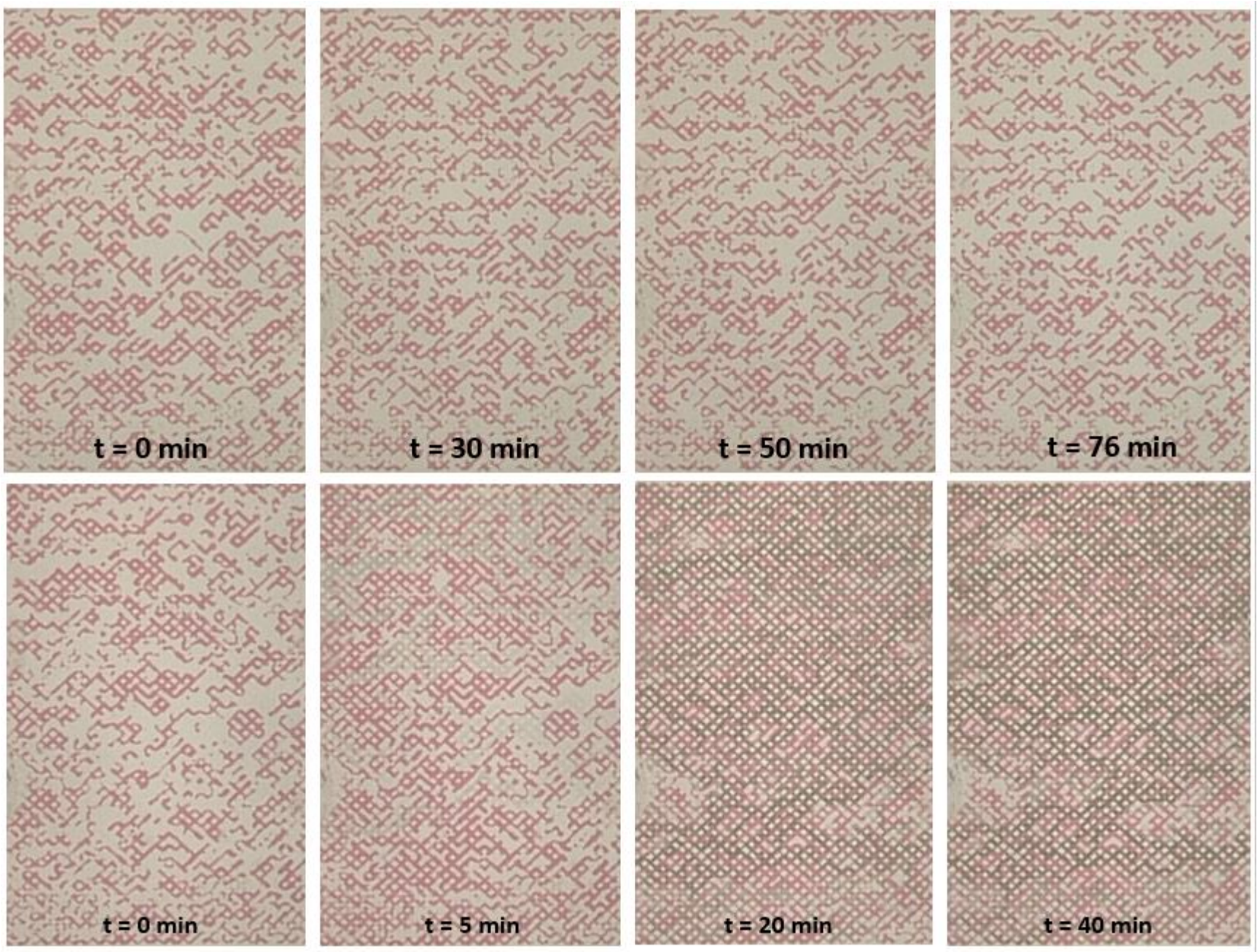
Results 2

SiO2=AMPSA-DMA-0.25% + NaCl 1M						
Type of displacement	Displaced fluid	Displacing fluid	Flow rate (mL/min)	Injected volume (mL)	Oil saturation	Oil removal efficiency (%)
Drainage	NaCl 1M	-	0.08	7.6	0.86	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.47	45.3
Sec. Imbib.	Resid. paraffin oil	Dispersion	0.2	8	0.40	14.9
Drainage	NaCl 1M	-	0.08	8	0.83	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.45	45.8
Sec. Imbib.	Resid. paraffin oil	Emulsion	0.2	8	0.16	64.4

Secondary Imbibition **P(AMPSA-co-AA-co-DMA) 0.25%-water**

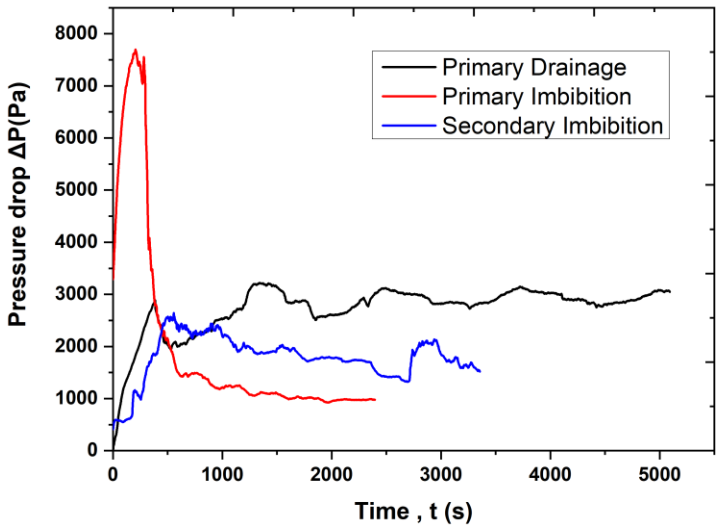
Dispersion

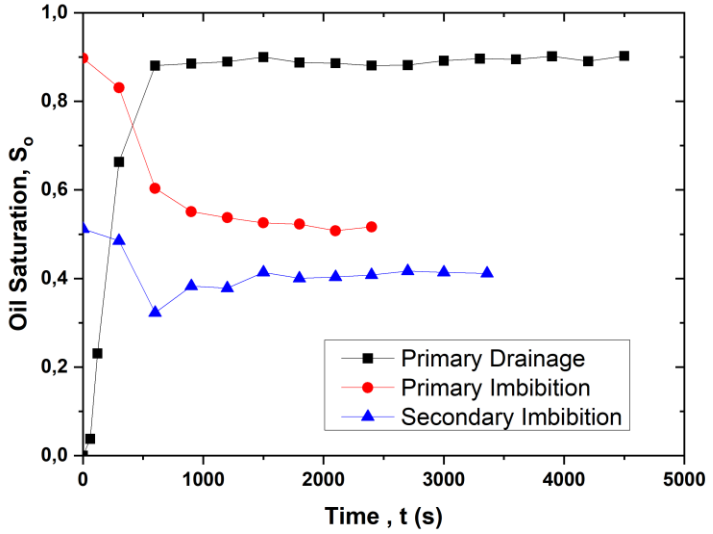
Emulsion



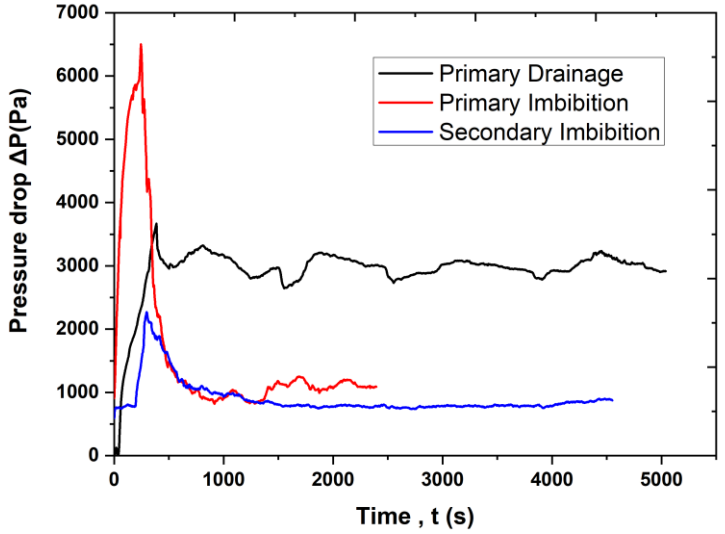
P(AMPSA-co-AA-co-DMA) 0.25%-water

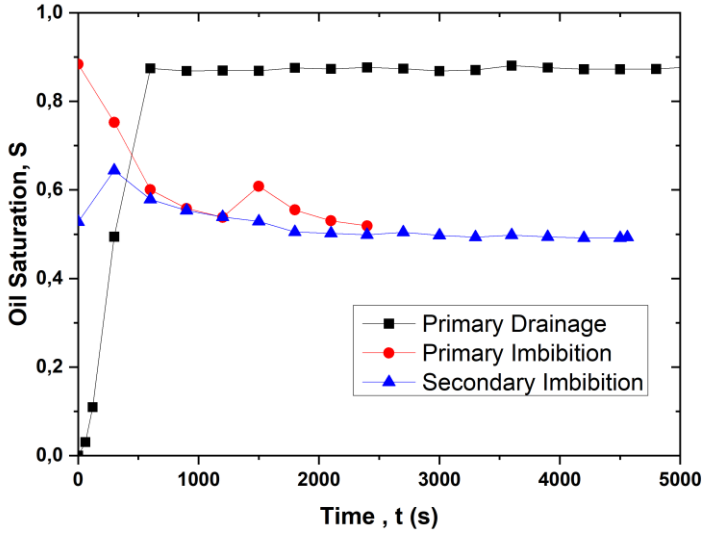
emulsion





dispersion



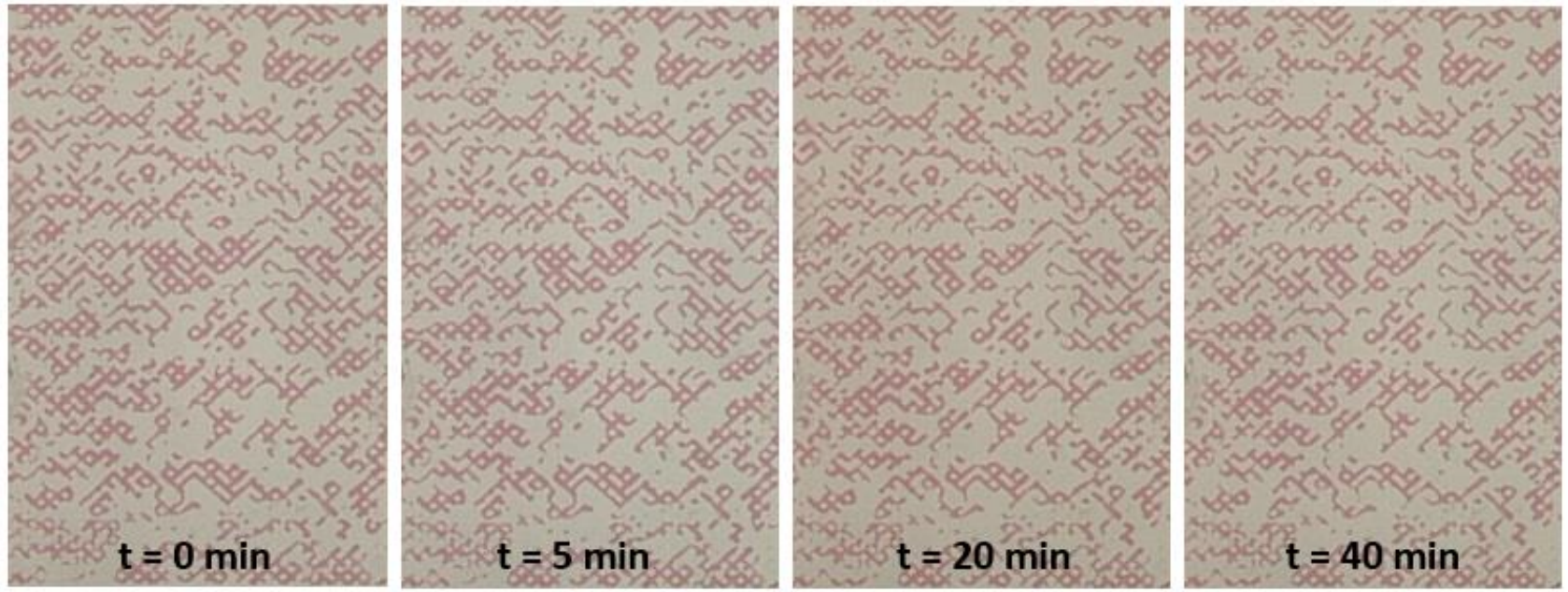


Results 3

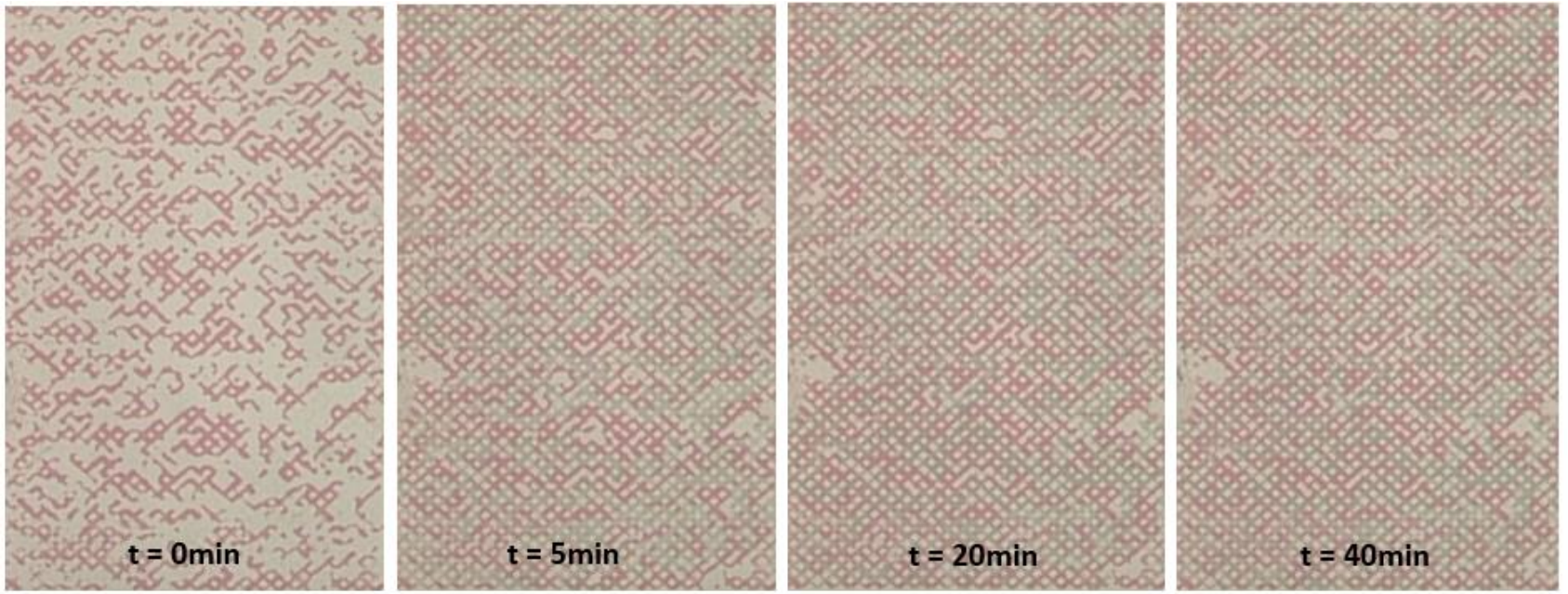
AMPSA 55.20.25 – 0.25% water						
Type of displacement	Displaced fluid	Displacing fluid	Flow rate (mL/min)	Injected volume (mL)	Oil saturation	Oil removal efficiency (%)
Drainage	NaCl 1M	-	0.08	8	0.88	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.53	39.8
Sec. Imbib.	Resid. paraffin oil	Dispersion	0.2	8	0.50	5.7
Drainage	NaCl 1M	-	0.08	8	0.90	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.51	42.6
Sec. Imbib.	Resid. paraffin oil	Emulsion	0.2	11.2	0.41	20.5

Secondary Imbibition **P(AMPSA-co-AA-co-DMA) 0.25%- NaCl 1M**

Dispersion

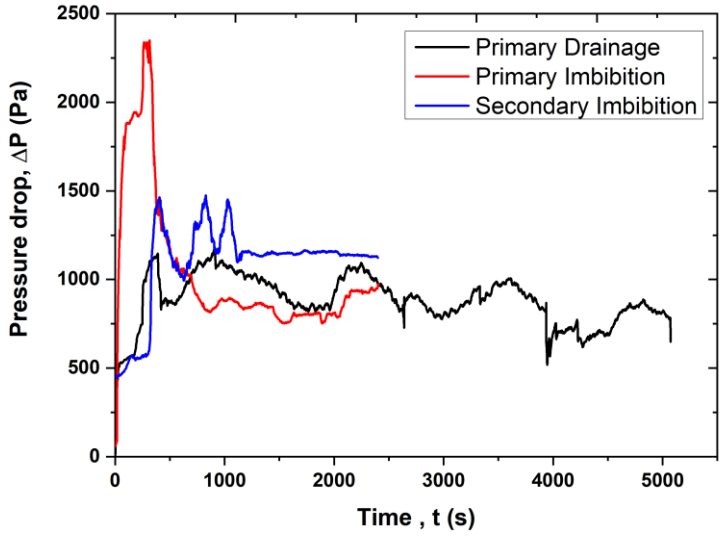


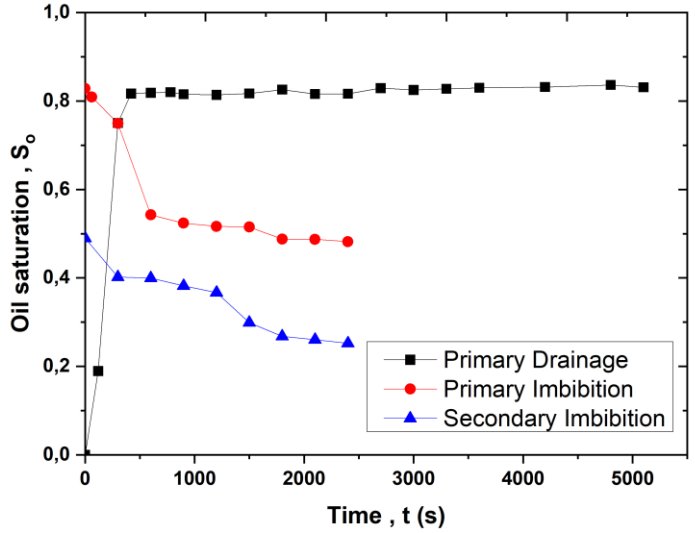
Emulsion



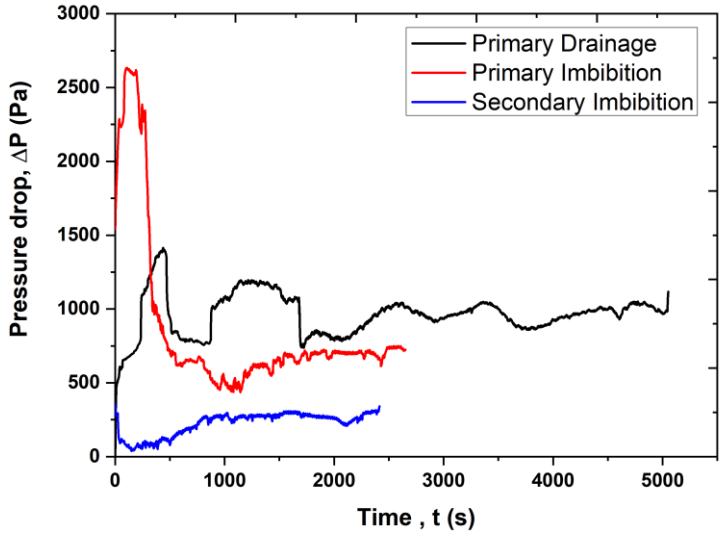
P(AMPSA-co-AA-co-DMA) 0.25% - NaCl 1M

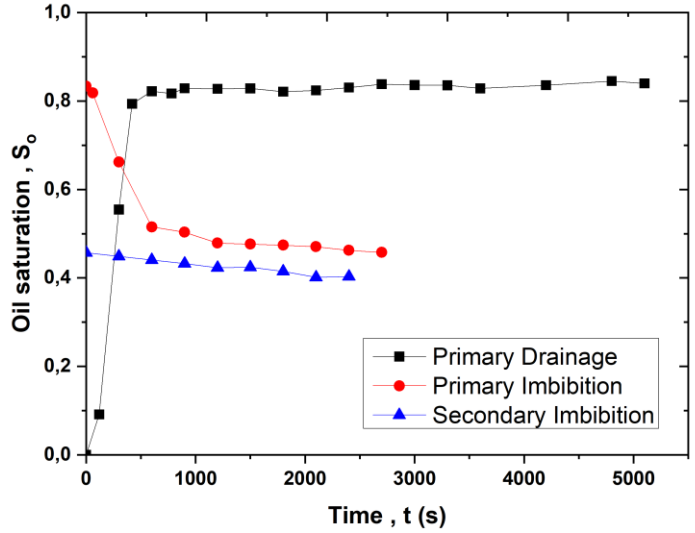
emulsion





dispersion





Results 4

AMPSA 55.20.25 - 0,25% +NaCl 1M						
Type of displacement	Displaced fluid	Displacing fluid	Flow rate (mL/min)	Injected volume (mL)	Oil saturation	Oil removal efficiency (%)
Drainage	NaCl 1M	-	0.08	8	0.84	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.46	45.2
Sec. Imbib.	Resid. paraffin oil	Dispersion	0.2	8	0.40	2.8
Drainage	NaCl 1M	-	0.08	8	0.83	-
Prim. Imbib.	Resid. paraffin oil	NaCl 1M	0.2	8	0.48	45.8
Sec. Imbib.	Resid. paraffin oil	Emulsion	0.2	8	0.25	47.9

Conclusions

- The potential to increase the residual oil recovery efficiency by injecting suspensions of polymer coated nanoparticles is investigated with visualization tests on a glass-etched pore network.
- The nanoparticles mobilize trapped oil by transferring it from upstream to downstream through a mechanism of successive steps of drainage (local increase of oil saturation) / imbibition (local decrease of oil saturation).
- It seems that the oil recovery efficiency of secondary imbibition tests is favored when using the SiO2-AMPSA=DMA 0.25% suspension with the presence of NaCl 1M . On the other hand, the oil recovery efficiency increases respectably when using emulsions stabilized by SiO2-AMPSA=DMA 0.25% with NaCl and CaCl2.

Thanks for your attention

Acknowledgements

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment (Project title: Enhanced Oil Recovery by Polymer-coated Nano Particles, acronym: «EOR-PNP», code: HFRI-FM17-361)

UNIVERSITY OF WATERLOO

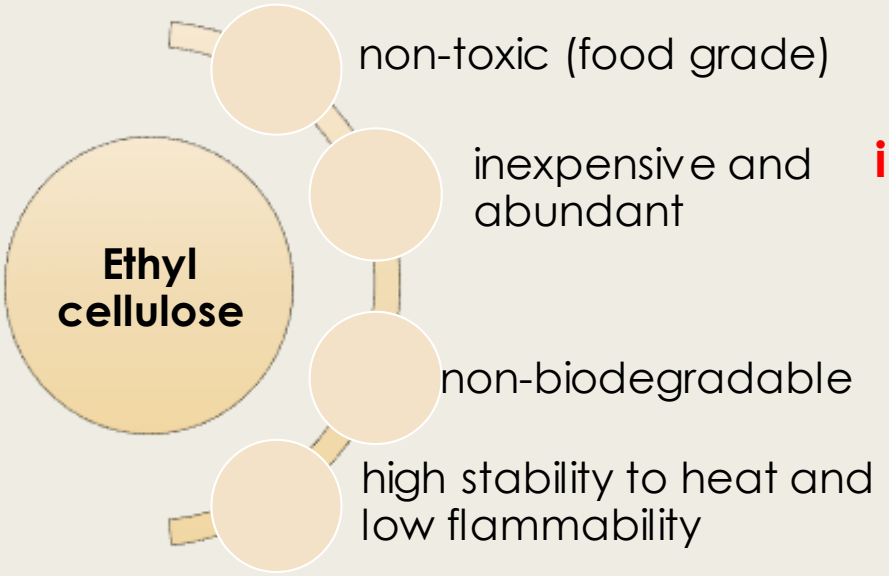
NANOPARTICLE INTERACTIONS WITH INTERFACES IN POROUS MEDIA: A MULTISCALE PERSPECTIVE

Marios Ioannidis, University of Waterloo

Outline

- Need for a multiscale approach
  - *Surface forces*
  - *Diffusion and hydrodynamic dispersion*
- A case for using a model system
  - *Ethyl cellulose nanoparticles: Synthesis and characterization*
- Interactions between
  - *Nanoparticles in bulk: suspension stability*
  - *Adsorbing nanoparticles: dynamics of attachment*
  - *Adsorbed nanoparticles: maximum coverage*
- Nanoparticle transport in porous media
  - *Modeling hydrodynamic dispersion*
  - *Modeling retention at fluid-fluid interfaces*

A multiscale approach is required to establish applications.



multiscale approach

single particle scale

particle-particle and particle-interface interactions and Interactions between particles adsorbed at the interface

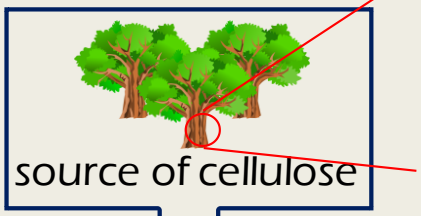
single interface scale

adsorption kinetics, Surface/interfacial tension, contact angle, extent and stability of np coverage

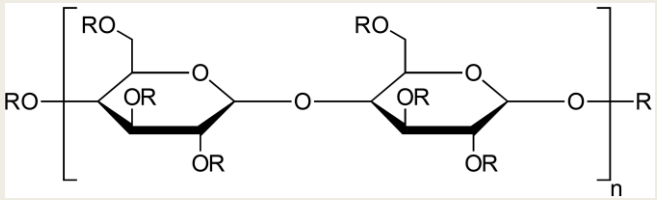
multiple interface scale

foam, emulsion, oil-water separation, transport in porous media,…

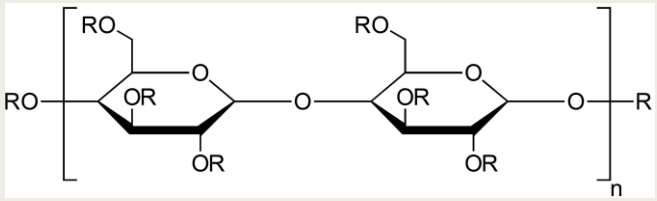
EC Nanoparticles are easily synthesized.



etherification process

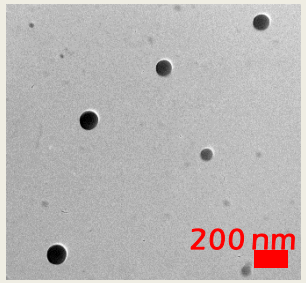


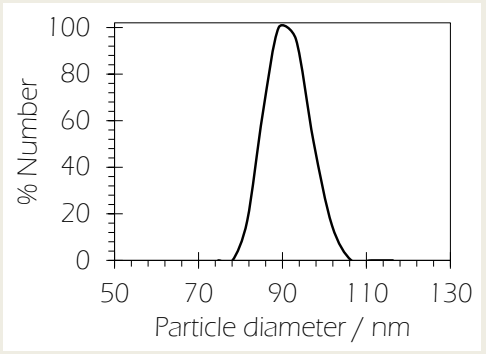
R=H

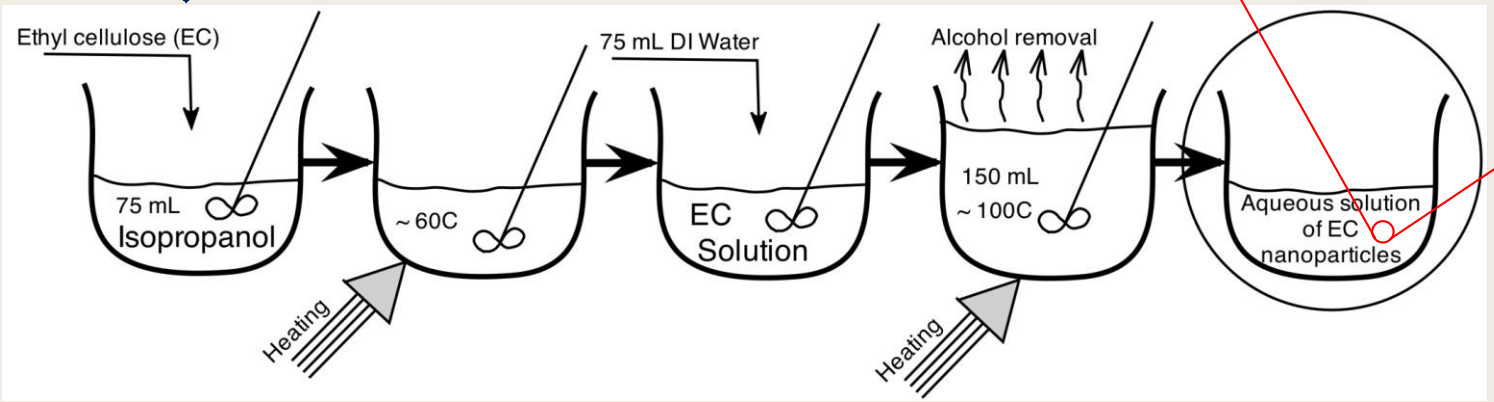


R=H or CH_2CH_3

DLS results and TEM images of 89.1 nm EC nanoparticles.*

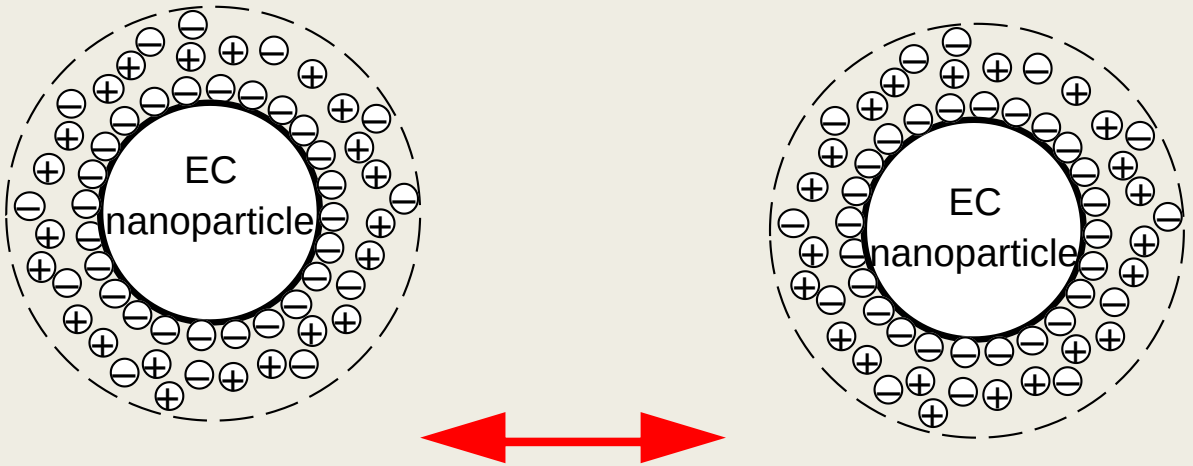




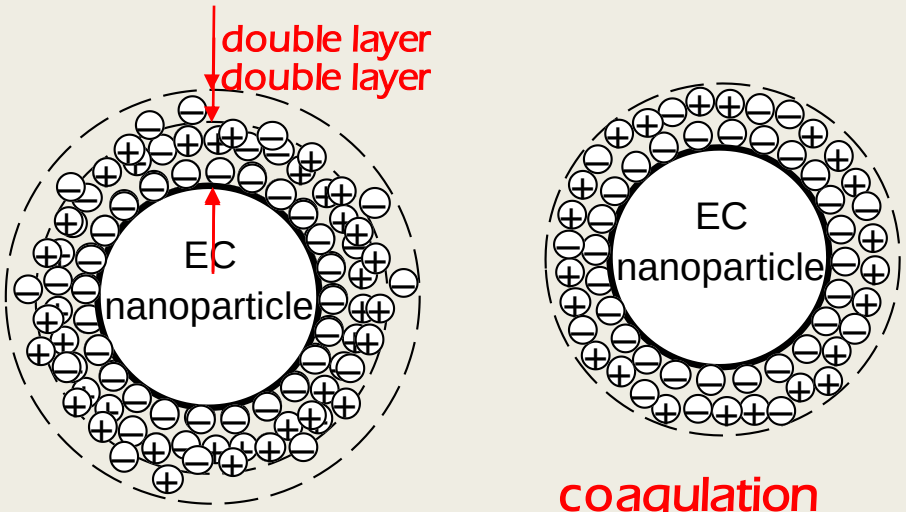


[*] N. Bizmark *et al. Langmuir* **2014**, 30, 710.

Electrostatic repulsion stabilizes EC nanoparticle suspended in water.



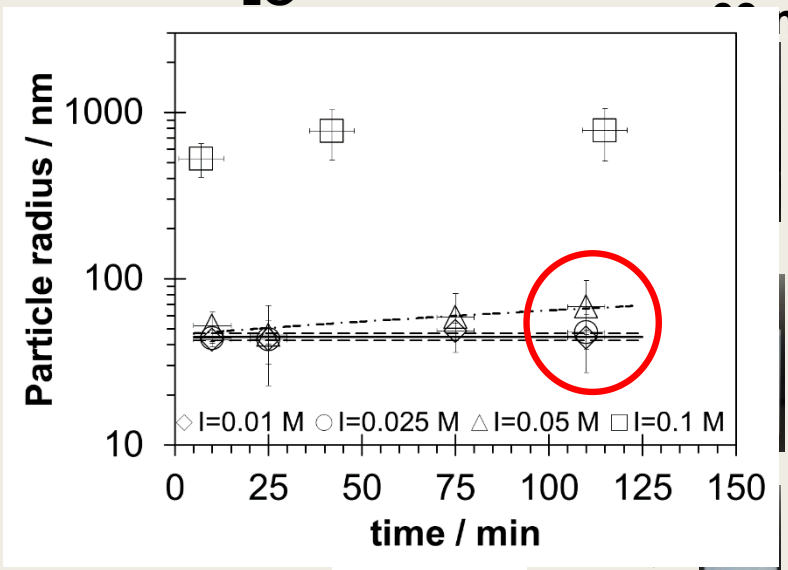
no salt



coagulation

in the presence of sufficiently large amount of ions in solution

Salt destabilizes the EC nanoparticle colloids.



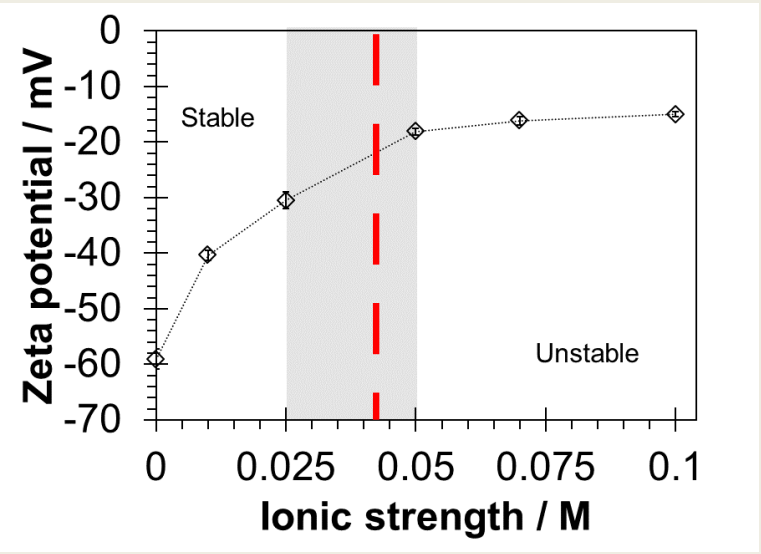
Changes in the hydrodynamic radius of EC nanoparticles over two hours at various ionic strengths and at EC nanoparticle concentration of 0.5 g L^{-1}.

Comparison pair EC 0.5 gL^{-1}	$t_{obs.}$	$T_{critical}$ (from t-distribution table)	Conclusion
0.01 M vs. 0.025 M	0.83	2.78	1
0.01 M vs. 0.05 M	2.93		2
0.025 M vs. 0.05 M	2.90		

1. $t_{obs.} < T_{critical}$ ➔ Fail to reject the null hypothesis
➔ There is no significant difference between the nanoparticle size.
2. $t_{obs.} > T_{critical}$ ➔ Reject the null hypothesis
➔ There is a significant difference between the nanoparticle size.

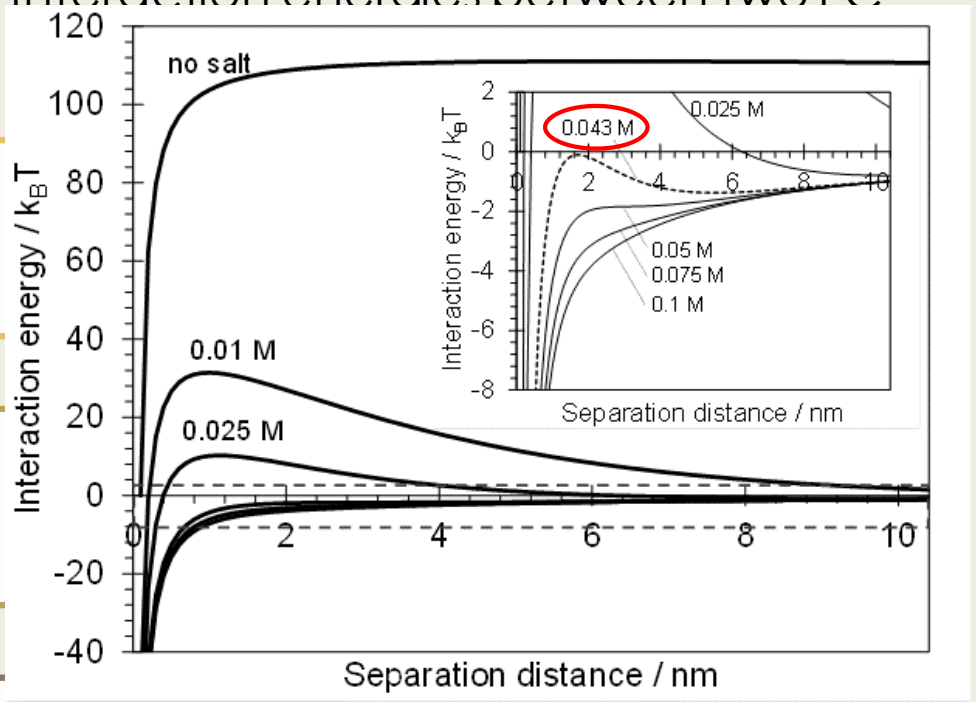
N. Bizmark and M. A. Ioannidis, *Langmuir*, **2015**, 31, 9282.

Extended-DLVO calculations predict the critical coagulation concentration.



Zeta potential of EC nanoparticles in deionized water (neutral pH) as a function of ionic strength. The grey zone separates the conditions that result in a stable and unstable colloidal solution.

Interaction energies between two EC

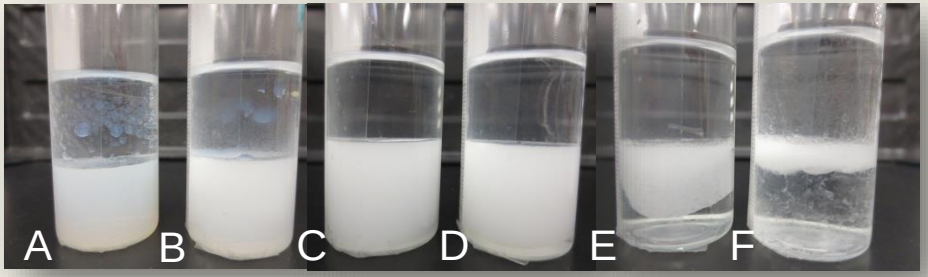


Total interaction energy between two EC nanoparticles of the same size (radius 43.96±4.92 nm) which is computed from extended-DLVO theory at various ionic strengths. The inset magnifies the dashed area.

N. Bizmark and M. A. Ioannidis, *Langmuir*, **2015**, 31, 9282.

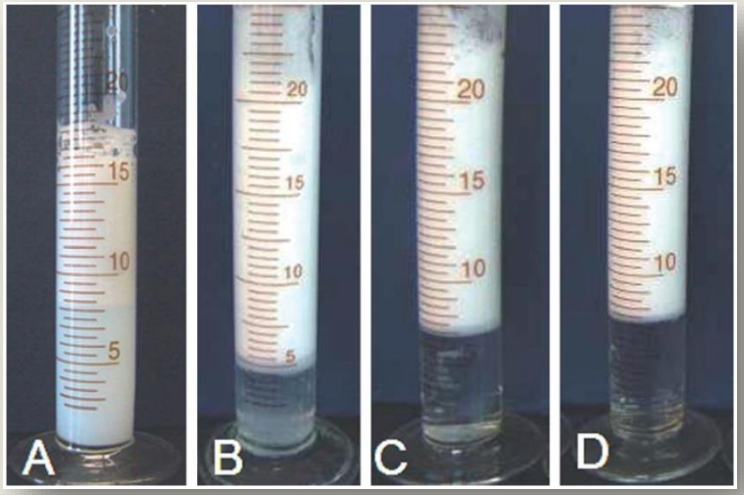
Stable emulsions and foams are made by EC nanoparticles.

EC nanoparticle-stabilized emulsion



Pickering decane-in-water emulsions stabilized by EC nanoparticles at a concentration of 1 wt.% and ionic strength (I) of (A) 0 M, (B) 0.01 M, (C) 0.025 M, and (D) 0.05 M. (E) and (F) are emulsions made at EC 0.4 wt.% and I=0.025M and 0.05 wt.% and I=0.1M, respectively.

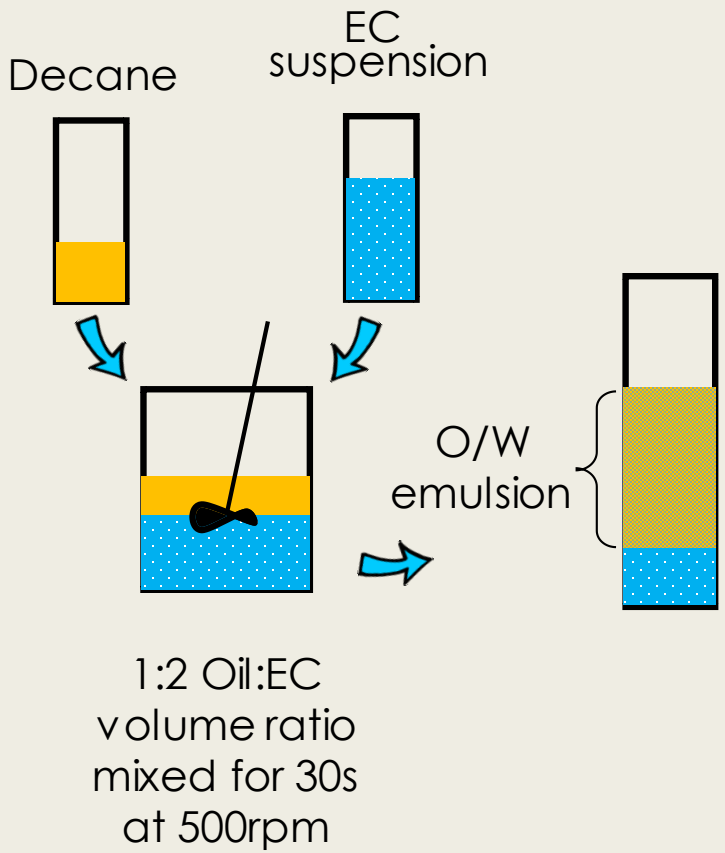
EC nanoparticle-stabilized foam

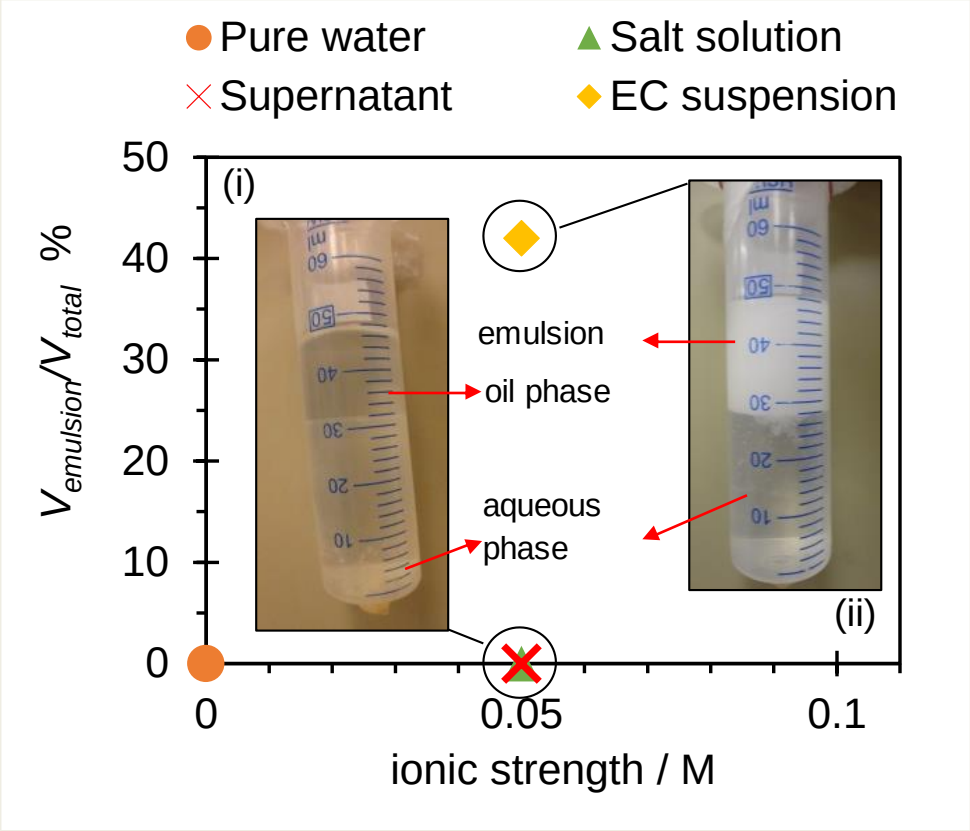


EC foam produced by hand-shaking for 2 min at different concentration of NaCl (A) 0M, (B) 0.01 M, (C) 0.05 M, and (D) 0.1 M. (pH of all samples being neutral).[*]

[*] H. Jin *et al. Soft Matter* **2012**, 8, 2194.

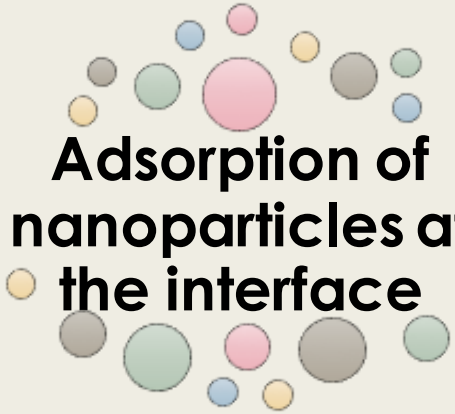
Pickering emulsions are generated only in the presence of EC nanoparticles.





N. Bizmark and M. A. Ioannidis, Langmuir **2017**, 33, 10568.

Adsorption is a necessary condition for emulsion/foam stabilization.



Adsorption of nanoparticles at the interface

Emulsion/Foam stabilization

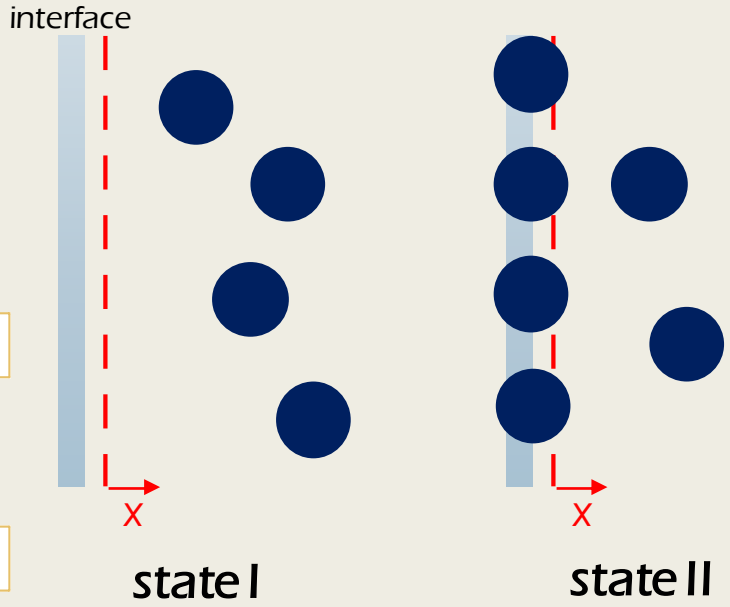
The adsorption of colloidal nanoparticles at fluid interfaces is irreversible – the process may be probed experimentally by measurements of the **dynamic surface or interfacial tension**.

Thermodynamic arguments enable application of RSA theory to dynamic surface/interfacial tension data.

$$\frac{dN_i}{dt} = -j\Big|_{x\to 0^+}$$

$$\frac{d\Theta}{dt} = \pi r^2 \frac{dN_i}{dt}$$

$$\Rightarrow \quad \frac{d\Theta}{dt} = -\pi r^2 \, j\Big|_{x\to 0^+}$$



adsorption flux during the early stages of adsorption:

$$j\Big|_{x\to 0^+} = -D\frac{\partial C}{\partial x}\Big|_{x\to 0^+} = -C_0\sqrt{\frac{D}{\pi t}}$$

adsorption flux during the late stages of adsorption:

$$j\Big|_{x\to 0^+} = -k_a N_A C_0 B(\Theta)$$

$$\Theta = 2\pi r^2 N_A C_0 \left(\frac{Dt}{\pi}\right)^{0.5}$$

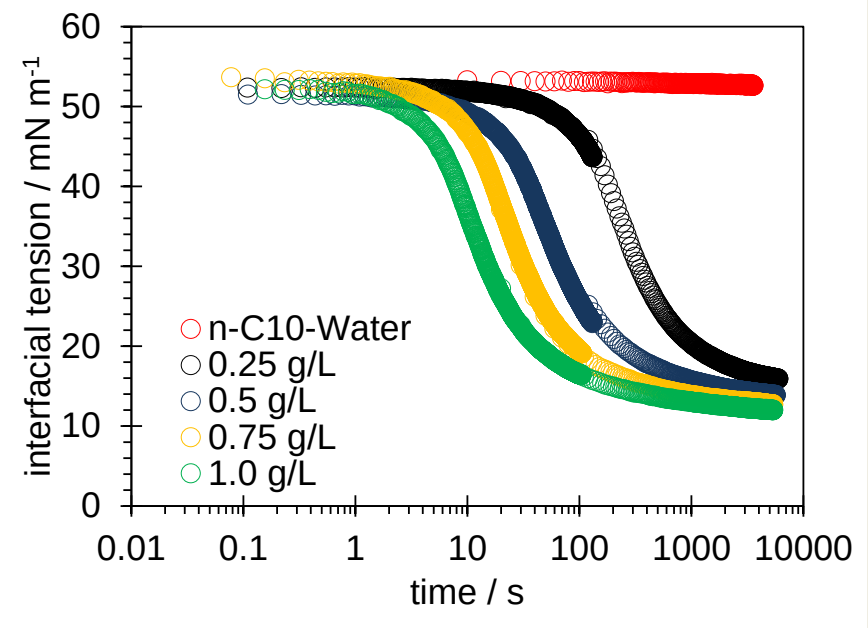
$$\Theta = \Theta_{\max} - \frac{K_l}{\pi r^2 N_A C_0}\left(\frac{1}{Dt}\right)^{0.5}$$

$$\Theta(t) = -\frac{(\gamma_0 - \gamma(t))\pi r^2}{\Delta E}$$

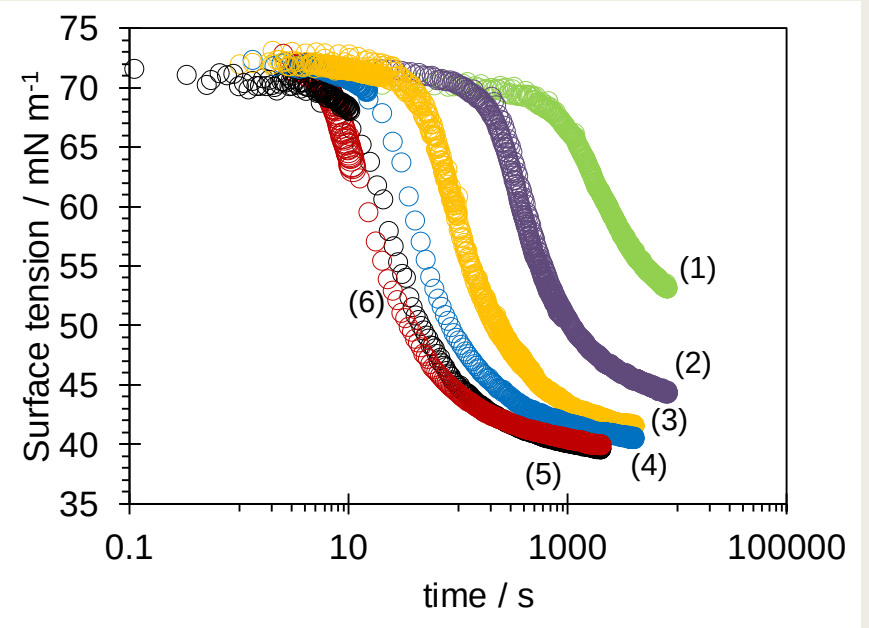
$$\gamma = \gamma_0 + 2N_A|\Delta E|C_0\left(\frac{Dt}{\pi}\right)^{0.5} \qquad t \to 0$$

$$\gamma = \gamma_\infty + \frac{K_l|\Delta E|}{(\pi r^2)^2 N_A C_0}\left(\frac{1}{Dt}\right)^{0.5} \qquad t \to \infty$$

EC nanoparticles adsorb readily at fluid interfaces.



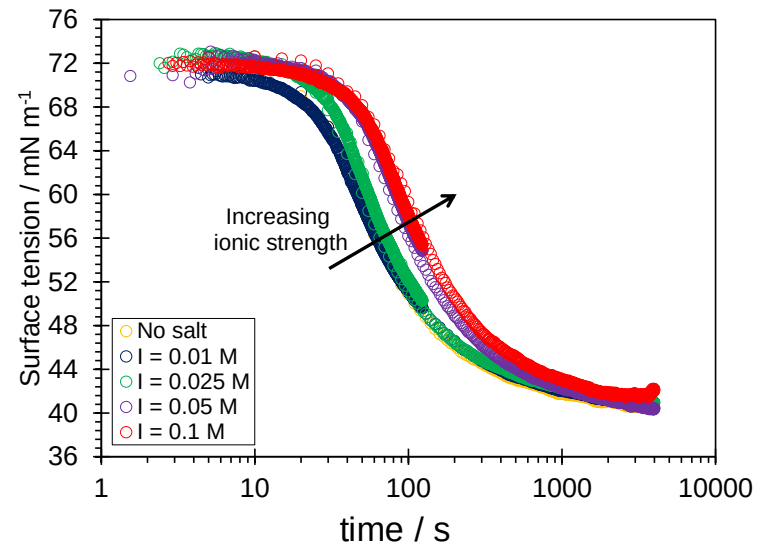
IFT measurements at different ionic strength for n-C10-water interface at the EC nanoparticle concentration of 0.5 g L^{-1}.



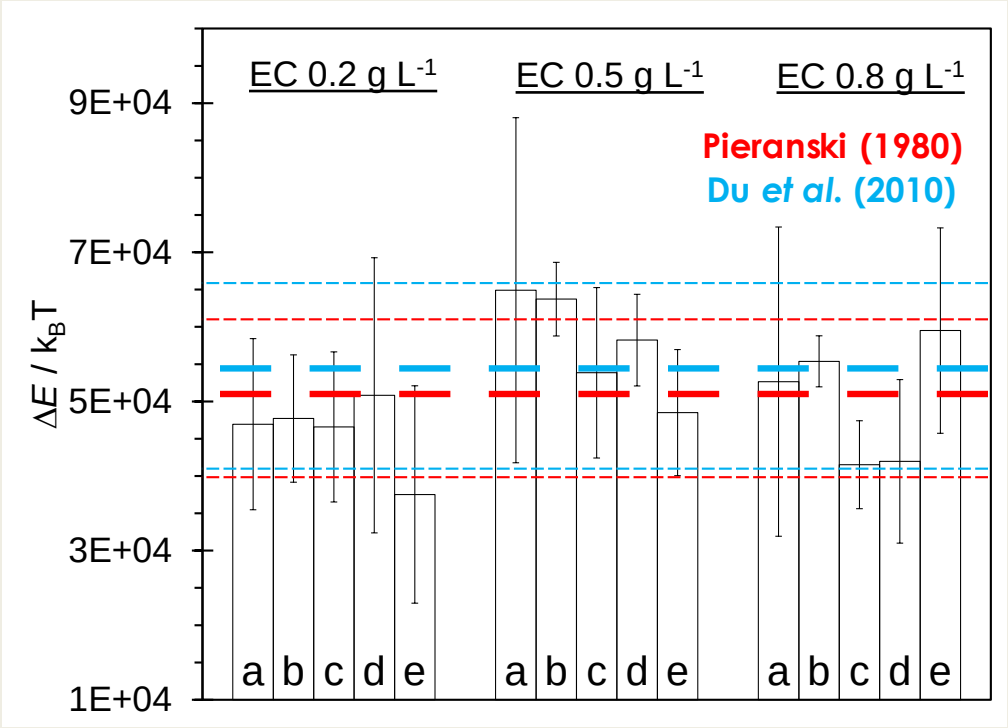
IFT measurements at different ionic strength for air-water interface at the EC nanoparticle concentration of (1) 0.1 g/L, (2) 0.2 g/L, (3) 0.4 g/L, (4) 0.6 g/L, (5) 0.8 g/L, and (6) 1.0 g/L.

N. Bizmark *et al. Langmuir* **2014**, 30, 710.
N. Bizmark and M. A. Ioannidis, Langmuir **2017**, 33, 10568.

Dynamic surface (interfacial) tension reveals irreversible adsorption of nanoparticles regardless of ionic strength.



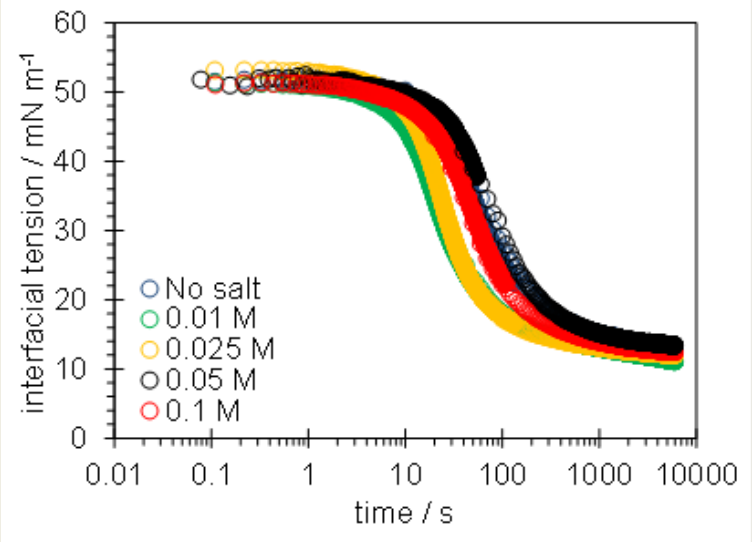
DST measurements of colloidal suspension of EC nanoparticles at the EC concentration of 0.5 g L^{-1} and various ionic strength.



Adsorption energy at three levels of EC nanoparticle concentrations and at ionic strengths of (a) 0 M, (b) 0.01 M, (c) 0.025 M, (d) 0.05 M, and (e) 0.1 M.

N. Bizmark and M. A. Ioannidis, *Langmuir*, **2015**, 31, 9282.
P. Pieranski. *Physical Review Letters*, **1980**, 45, 569.
K. Du et al. *Langmuir*, **2010**, 26, 12518.

Dynamic surface (interfacial) tension reveals irreversible adsorption of nanoparticles regardless of ionic strength.



IFT measurements at different ionic strength for n-C10-water interface at the EC nanoparticle concentration of 0.5 g L^{-1}.

Ionic strength (M)	iso-C8-water	n-C10-water
0.01	(10±2)x10^4 k_BT	(8±3)x10^4 k_BT
0.025	(9±2)x10^4 k_BT	(7±3)x10^4 k_BT
0.05	(10±4)x10^4 k_BT	(8±3)x10^4 k_BT
0.1	(9±1)x10^4 k_BT	(6±2)x10^4 k_BT
0.2	(8±1)x10^4 k_BT	–

Compare with (7±1)x10^4 k_BT from Du *et al.*

Adsorption energy of EC nanoparticle at alkane-water interfaces at various ionic strengths. EC nanoparticle concentration is 0.5 g L^{-1}.

N. Bizmark and M. A. Ioannidis, Langmuir **2017**, 33, 10568.
K. Du et al. *Langmuir*, **2010**, 26, 12518.

Adsorption flux decreases during the late stages of adsorption beyond the CCC.

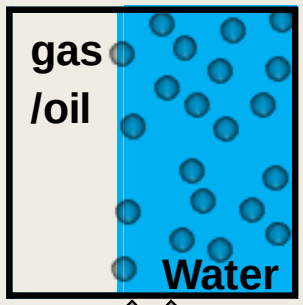
ionic strength [M]	EC nanoparticle concentration [g L^{-1}]	iso-C8-water	n-C10-water	air-water
0	0.5	85±21	98±12	90±10
0.01		62±11	83±5	100±10
0.025		54±4	57±1	90±20
0.05		54±5	107±7	120±70
0.1		61±2	84±11	140±40
0.2		64±4	–	-

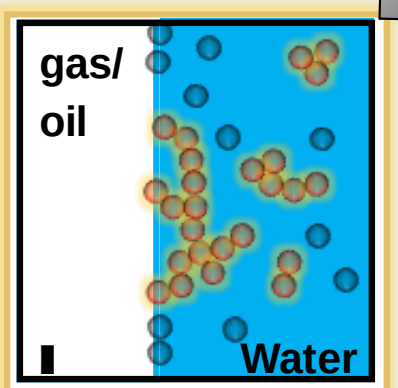
$$\gamma = \gamma_\infty + \frac{K_l |\Delta E|}{\left(\pi r^2\right)^2 N_A C_0} \left(\frac{1}{Dt}\right)^{0.5} \Rightarrow \left.\frac{d\gamma}{dt^{-0.5}}\right|_{t\to\infty} = \frac{K_l |\Delta E|}{\left(\pi r^2\right)^2 N_A C_0 D^{0.5}}$$

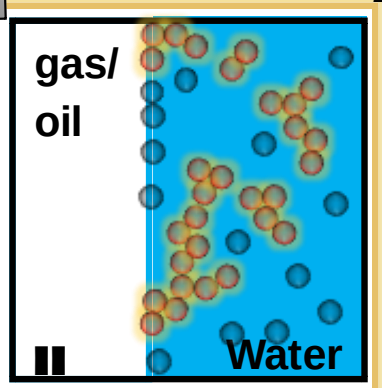
Two possible scenarios for the adsorption of EC nanoparticles at fluid interfaces:

early stages of adsorption ($t \rightarrow 0$)

late stages of adsorption ($t \rightarrow \infty$) beyond CCC







Nanoparticle flux at the interface

$$j\big|_{x\to 0^+} = -k_a C_0 B(\Theta)$$

$$B^{RSA}(\Theta) \cong 2.32\left(1-\frac{\Theta}{\Theta_\infty}\right)^3$$

nanoparticle aggregates formed in the bulk are adsorbed at the fluid interface alongside single nanoparticles.

only single nanoparticles are adsorbed at the fluid interface; nanoparticle aggregates at the interface are formed by the attachment on nanoparticles on already adsorbed ones

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Chemical Engineering

N. Bizmark and M. A. Ioannidis, *Langmuir*, **2015**, 31, 9282.

Extended-DLVO calculations are consistent with maximum surface coverage.

In Oil

$$\phi_{vdW}^{O} = -r\theta\frac{H_{121}}{12\pi h}$$

$$\phi_{elec}^{O} = \frac{\left(A_{po}\sigma_{po}\right)^2}{8\pi r\varepsilon_o\varepsilon_0}\left\{\frac{1}{\left(1+\frac{h}{2r}\right)^2} - \frac{1+\frac{h}{2r}}{(3+\cos\theta)^2+\left(2+\frac{h}{r}\right)^2}\right\}$$

$$\phi_{d-d}^{O} = \frac{\left(\pi\sigma_d r^2\sin^2\theta\right)^2}{32\pi\varepsilon\varepsilon_0\left(r+\frac{h}{2}\right)^3}$$

In Water

$$\phi_{vdW}^{W} = -r\left(\pi-\theta\right)\frac{H_{131}}{12\pi h}$$

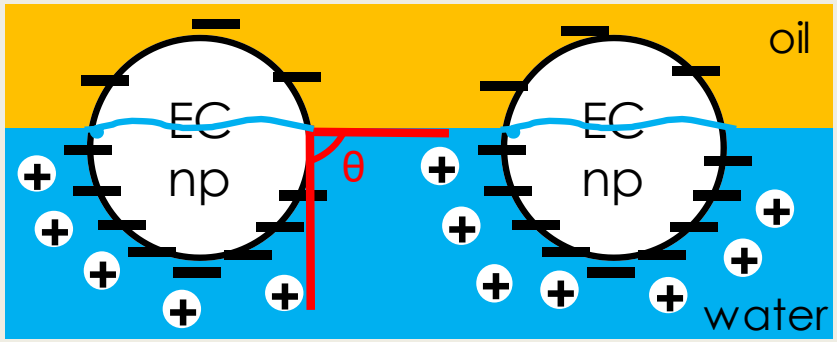
$$\phi_{hydro}^{W} = -r\left(\pi-\theta\right)\frac{K_{131}}{2\pi h}$$

$$\phi_{elec}^{W} = 4\left(\pi-\theta\right)\varepsilon_0\varepsilon_r r\psi_{pw}^2\left(1-\frac{r}{h+2r}\right)\ln\left[1+\frac{\exp\left(-\kappa h\right)}{1+h/r}\right]$$

$$\phi_{d-d}^{w} = \frac{\left(A_{pw}\sigma_{pw}\right)^2}{16\pi\varepsilon_0\varepsilon_w^2\kappa^2\left(r+\frac{h}{2}\right)^3}$$

Electrostatic repulsion through oil
Dipole-dipole repulsion through oil

Van der Waals attraction through oil



Van der Waals attraction through water
Hydrophobic attraction through water
Capillary attraction due to undulation

Electrostatic repulsion through water
Dipole-dipole repulsion through water

Horozov *et al. Langmuir* **2005**, *21*, 7405.
Horozov *et al. Langmuir* **2003**, *19*, 2822.
N. Bizmark and M. A. Ioannidis, Langmuir **2017**, 33, 10568.

Capillary attraction a negligible impact on the net interaction energy between particles adsorbed at the interface.

monopole capillary

$$Bo = \frac{\Delta\rho g r^2}{\gamma} = O(10^{-10})$$

negligible

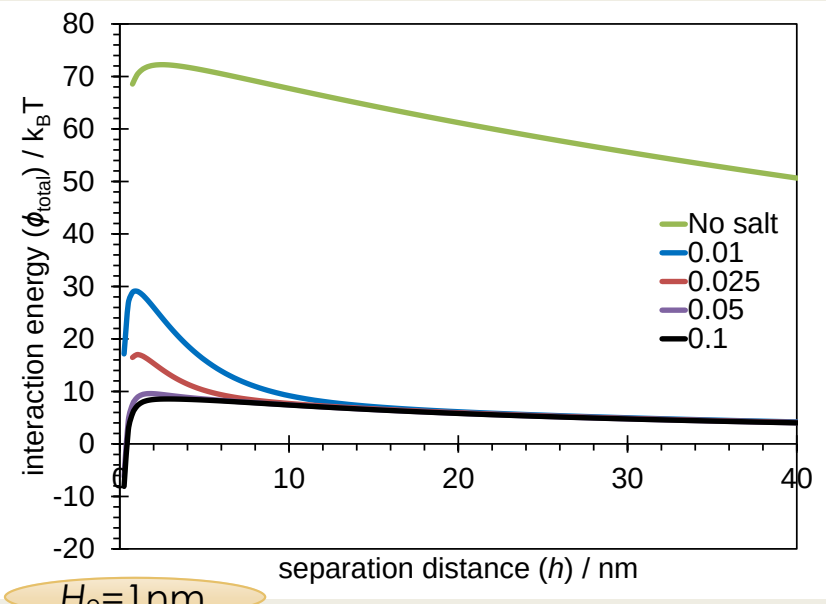
quadrupole capillary

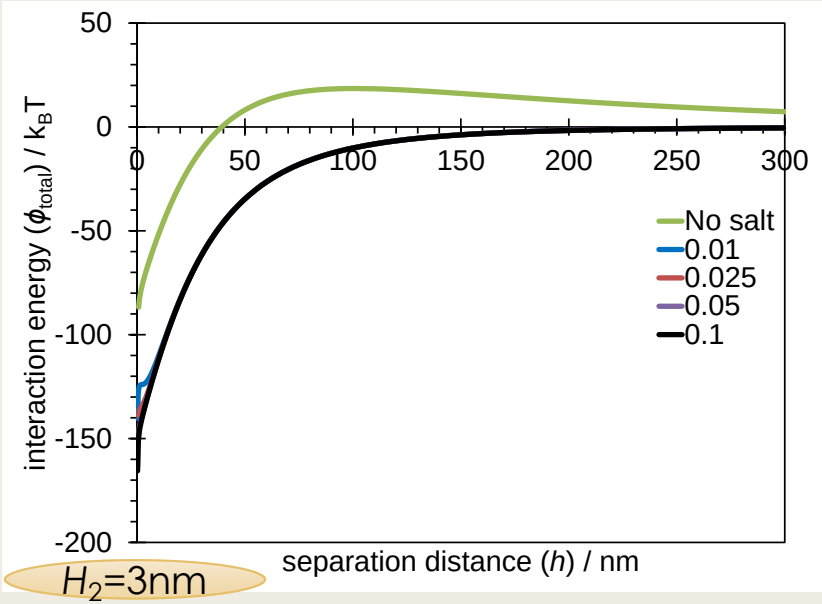
$$\Delta\phi_{Qcap}(h) = \phi_{Qcap}(h) - \phi_{Qcap}(\infty)$$

$$\phi_{Qcap}(h) = \pi\gamma H_2^2 (2S(\delta) - G(\delta))$$

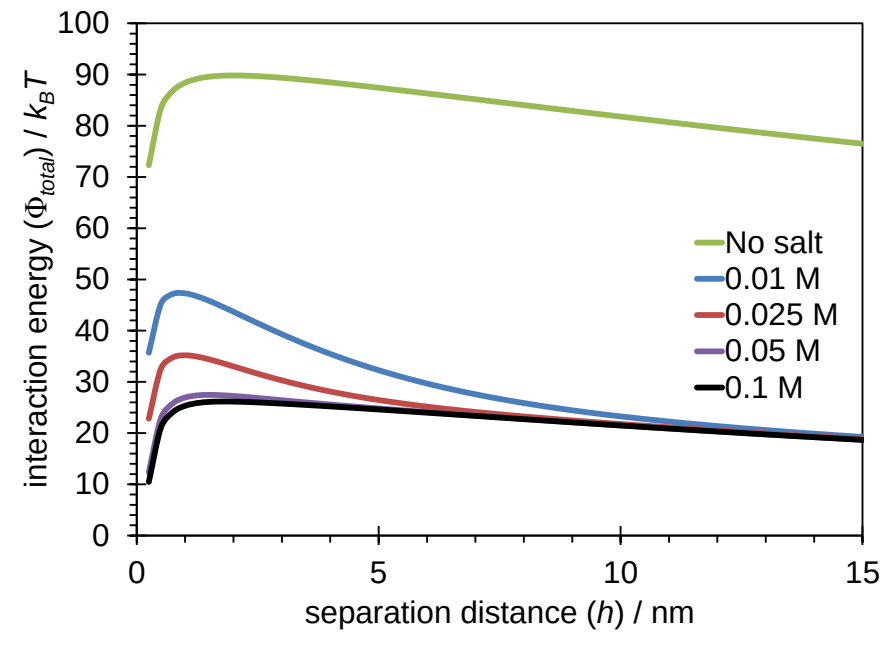
$$\phi_{Qcap}(\infty) = 2\pi\gamma H_2^2$$

Extended DLVO calculations for adsorbed EC nanoparticles at n-decane-water interface considering quadrupolar capillary attraction.

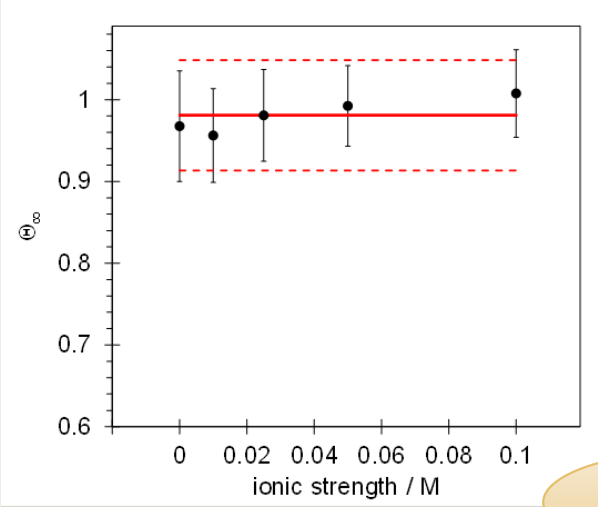




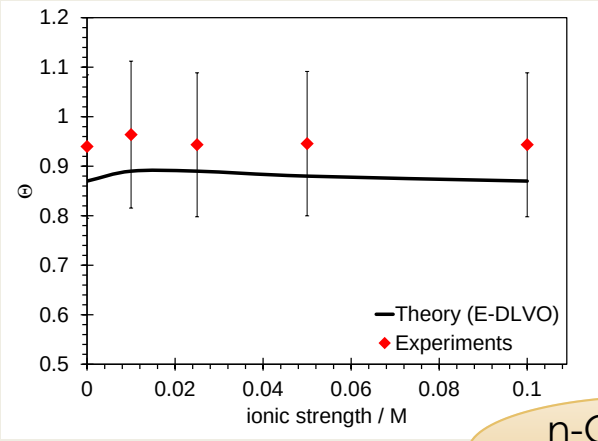
Extended-DLVO calculations predict maximum surface coverage at the interface.



DST measurements of colloidal suspension of EC nanoparticles at the EC concentration of 0.5 g L^{-1} and various ionic strength.



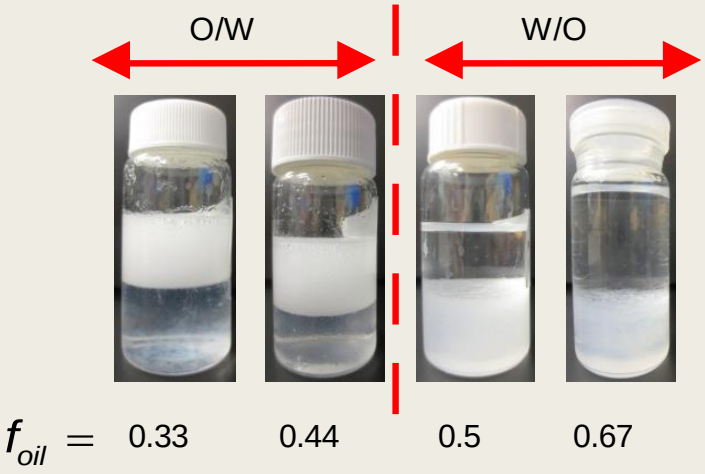
air-water interface

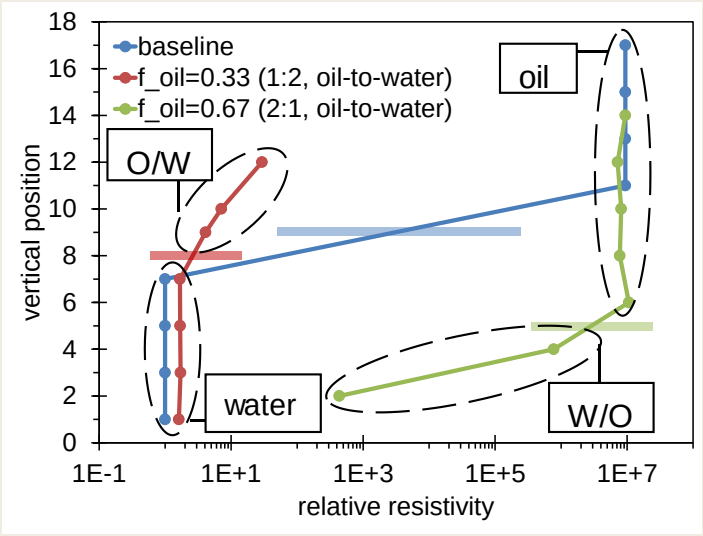


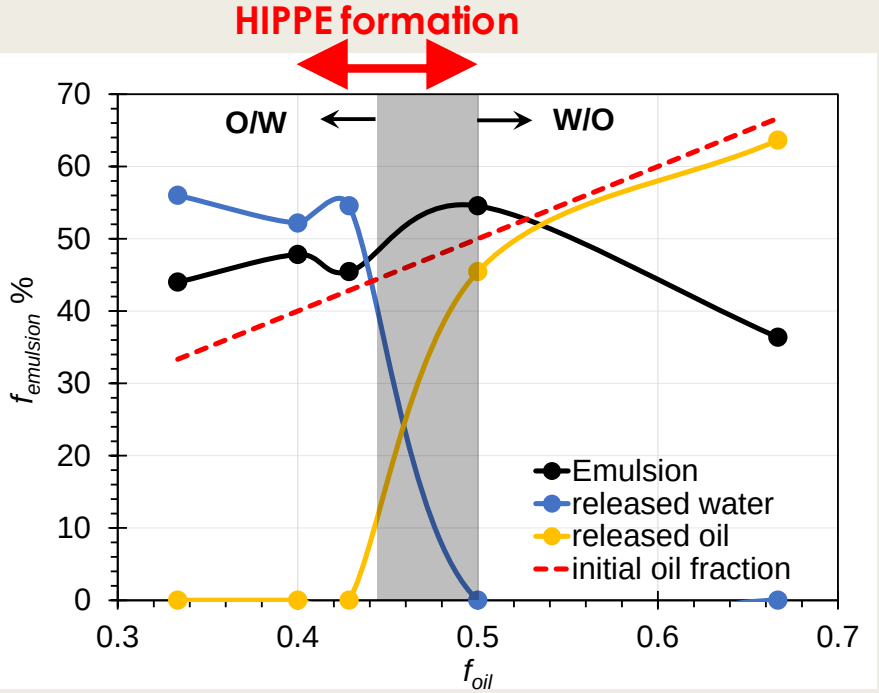
n-C10-water interface

Horozov *et al. Langmuir* **2005**, *21*, 7405.
Horozov *et al. Langmuir* **2003**, *19*, 2822.
N. Bizmark and M. A. Ioannidis, Langmuir **2017**, 33, 10568.

Oil volume fraction decides the type of Pickering emulsion generated with EC nanoparticles.



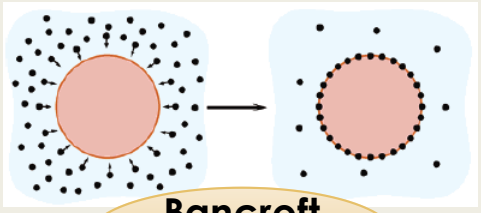




The O/W Pickering emulsions are enriched in oil as f_{oil} as increased and at $f_{oil} \geq 0.4$, where the dispersed phase occupied at least 74% of the volume of emulsion, **high internal phase Pickering emulsions** (HIPPEs) formed.

(anti-)Bancroft-type Pickering emulsions are generated depending on the oil volume fraction.

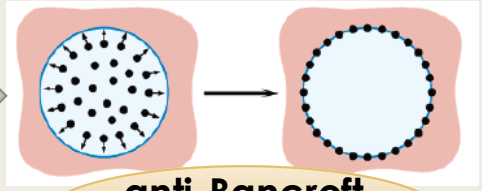
Emulsifier dispersed in continuous phase



Bancroft emulsion

$$D_{32}\big|_{f_{oil}<0.5} = \frac{8r\rho_p\Theta}{\rho_0}\frac{f_{oil}}{1-f_{oil}}$$

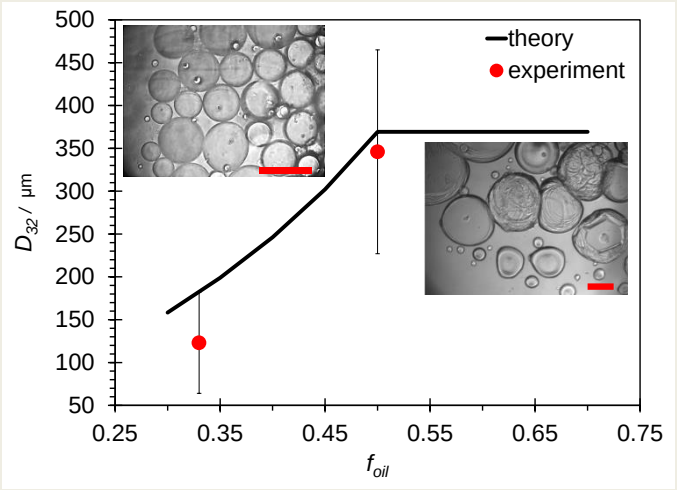
Emulsifier dispersed in dispersed phase



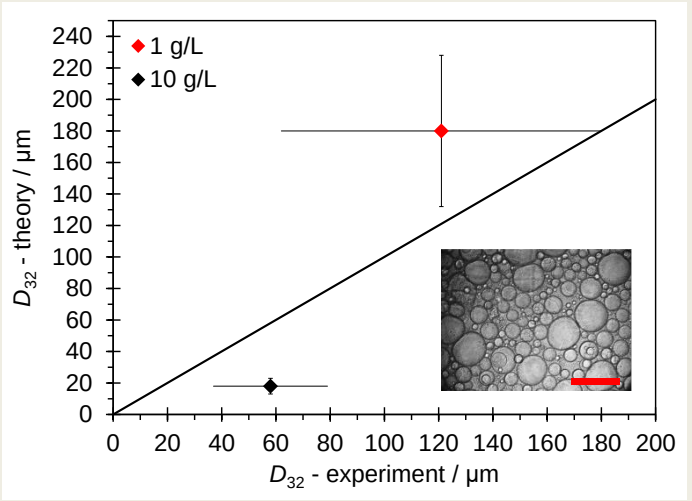
anti-Bancroft emulsion

$$D_{32}\big|_{f_{oil}\geq 0.5} = \frac{8r\rho_p\Theta}{\rho_0}$$

effect of f_{oil}



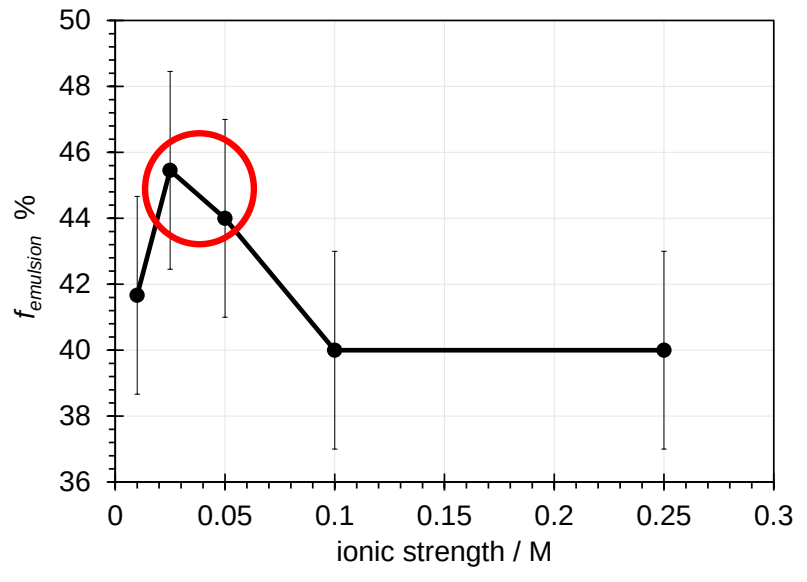
effect of ρ_0

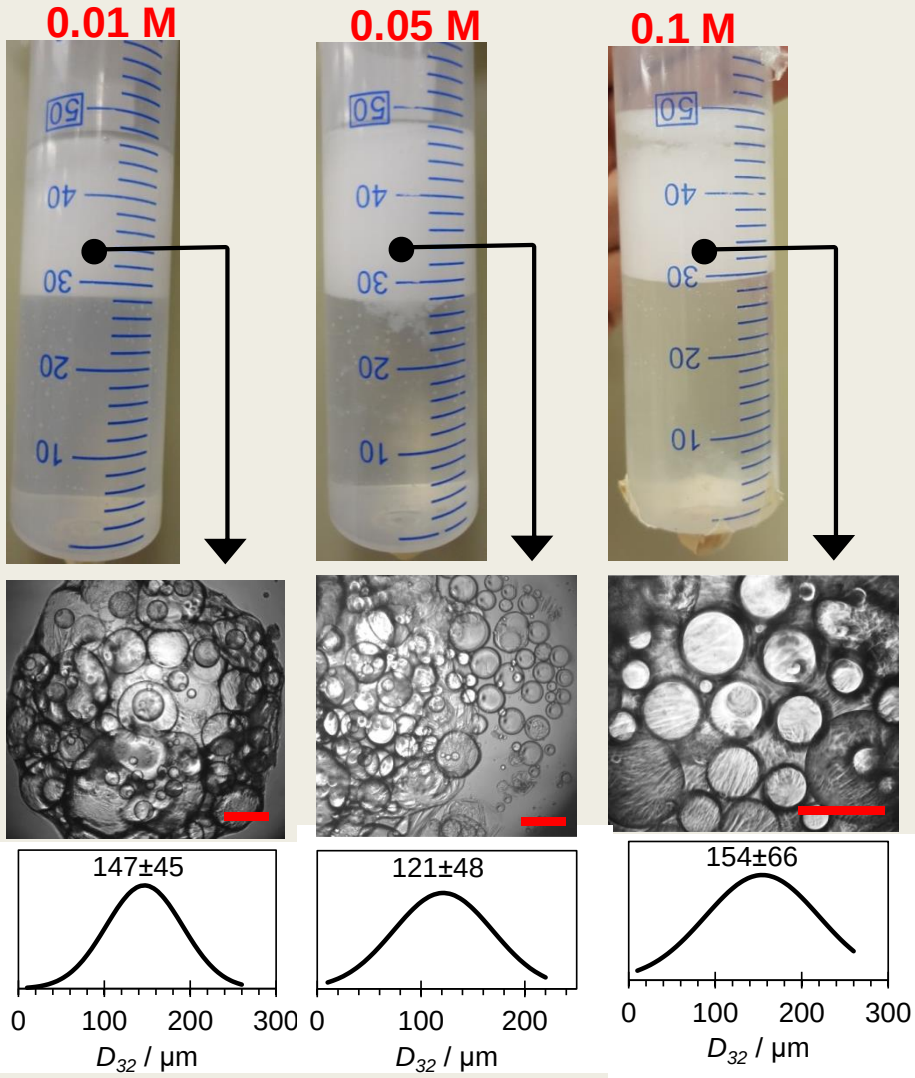


Analyses on the drop size distribution further support the conclusion that the fluid interface is densely covered to its maximum limit (i.e. 91%).

Cartoons from K. Golemanov *et al*. *Langmuir* **2006**, *22*, 4968.

The most stable Pickering emulsion could be obtained when the ionic strength is close to the CCC.

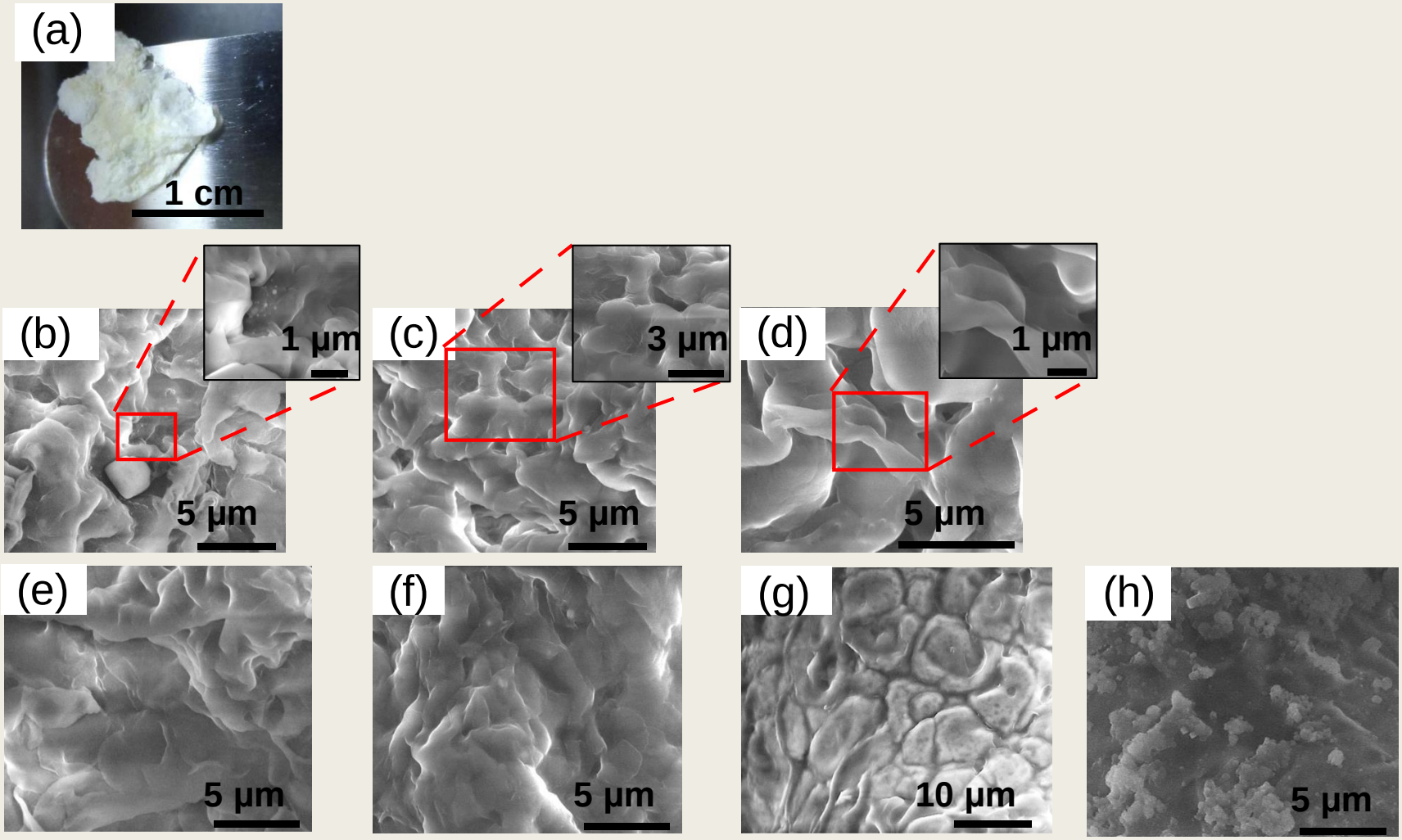




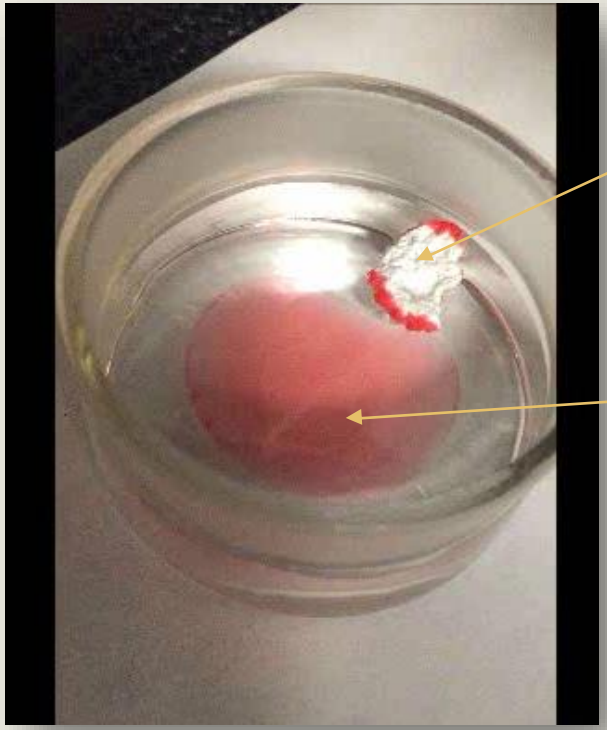
Salt has no significant effect on the volume fraction of emulsion or mean size of oil drops.

However, the largest volume for Pickering emulsions could be obtained when ionic strength is between 0.025 M and 0.05 M.

High internal phase Pickering emulsions serve as a template for porous material fabrication.



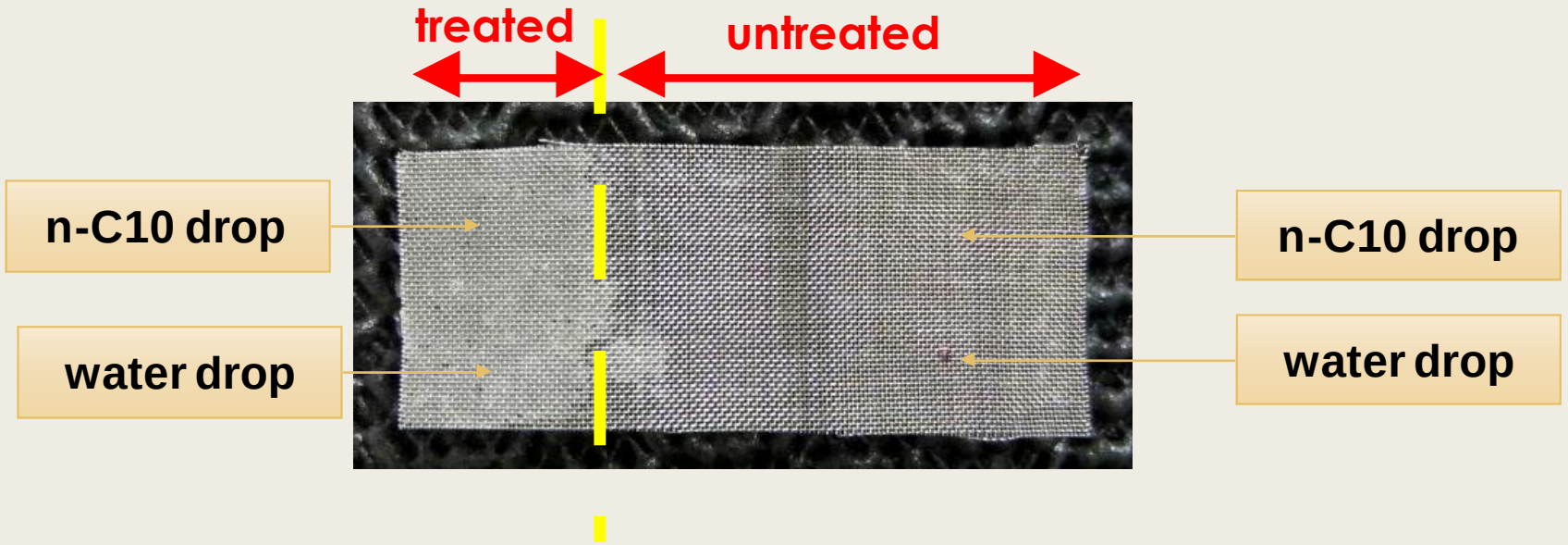
Oil spill is successfully removed.



dried Pickering emulsion

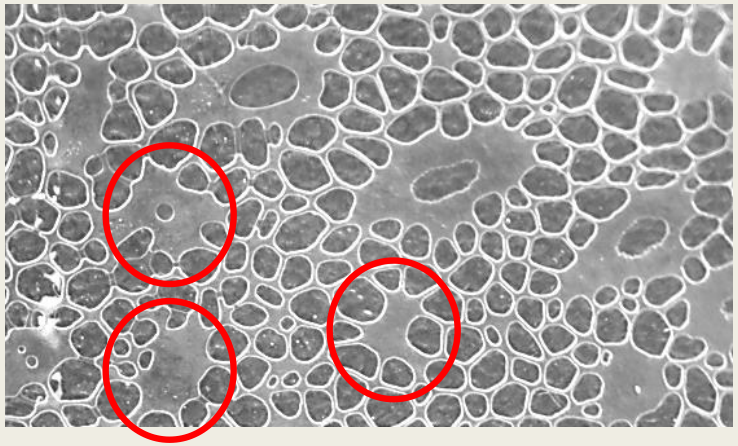
Red-dyed decane spilled on the surface of water

Wettability is controlled.



Foam can be generated *in situ* by co-injecting gas and np suspension.

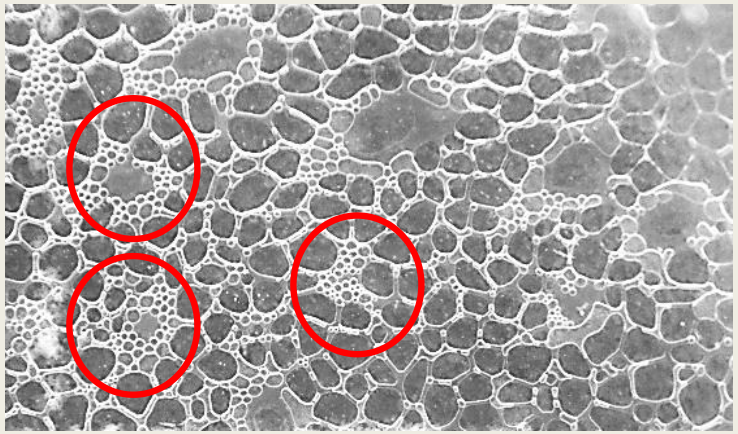
Injecting water at ionic strength of 0.05 M



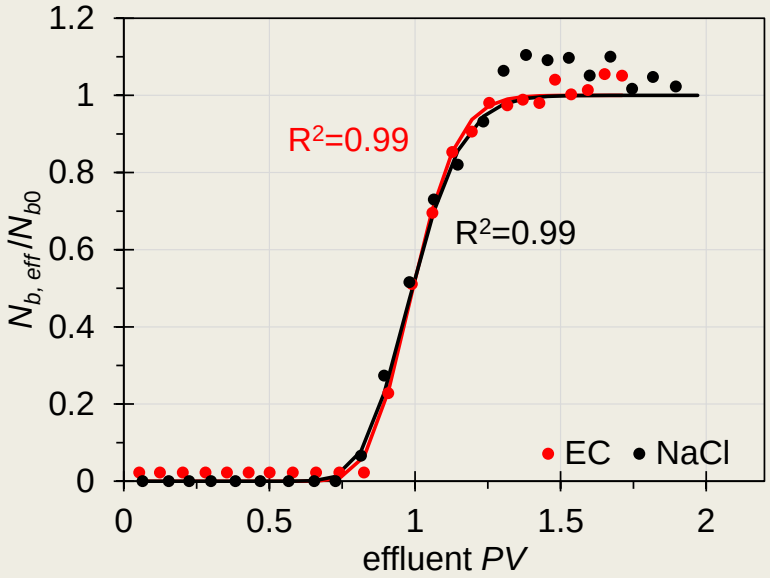
100% water-saturated (ionic strength = 0.05 M)

nitrogen stream

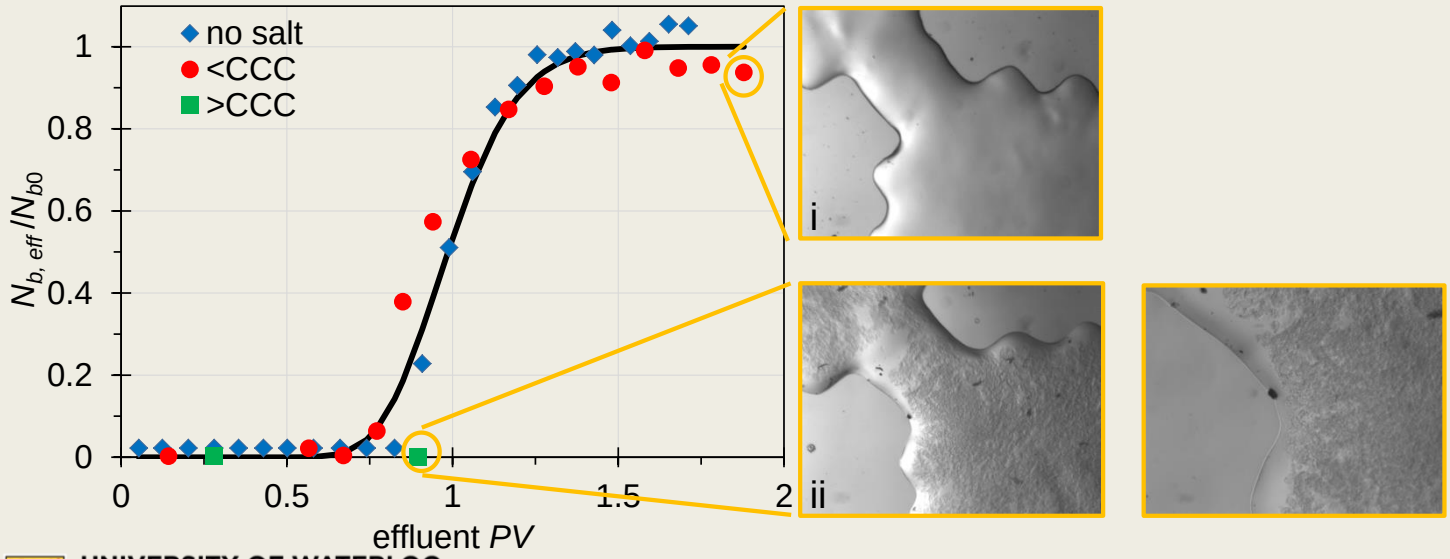
EC nanoparticle suspension at 0.05 M



EC nanoparticles are transported as a conservative tracer in porous media.

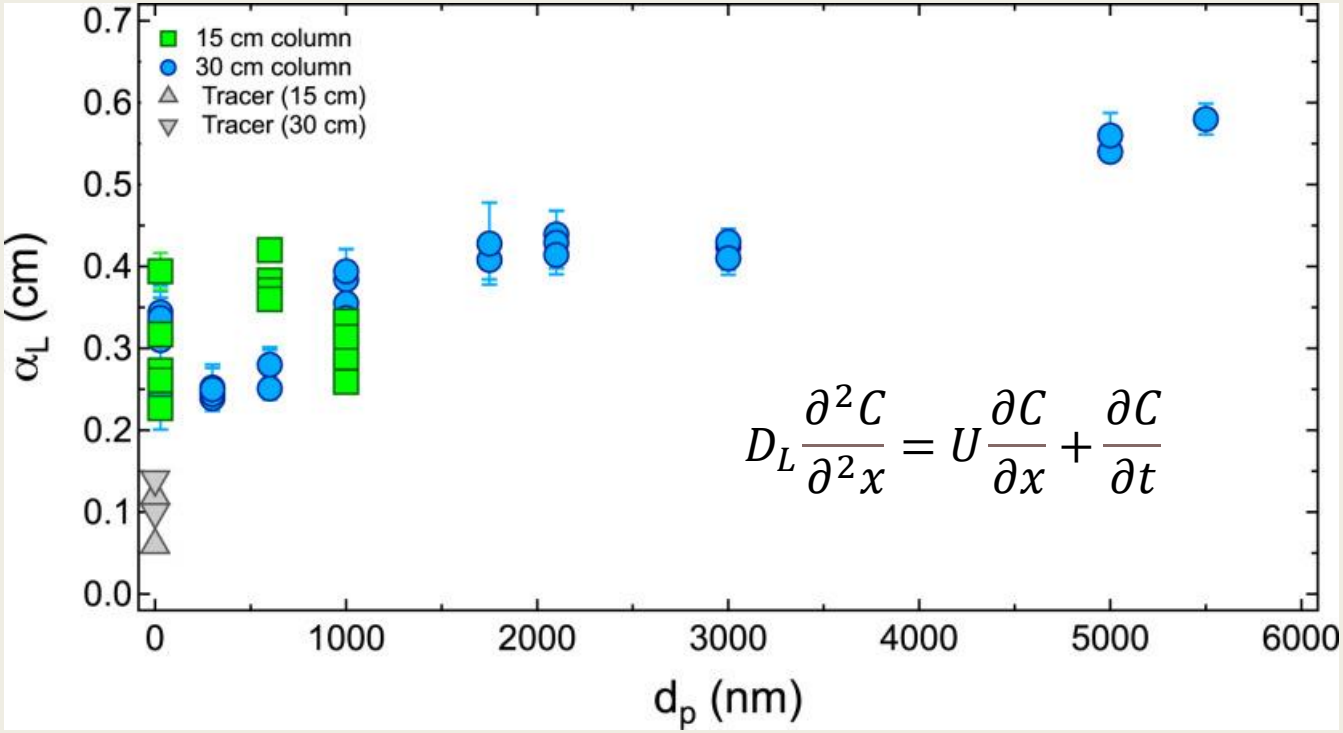


Transported species	Ionic strength (M)	α_L (cm)	D_h (m²/s)	Pe
NaCl	1.52	0.3±0.1	(4±2)x10⁷	104±61
EC np	0	0.2±0.1	(3±1)x10⁷	135±44
EC np	0.01	0.4±0.1	(6±1)x10⁷	70±19
EC np	0.1	-		



Pore network modeling of nanoparticle transport in porous media

- A testbed for multiphysics
  - *Is dispersivity particle size-dependent?*



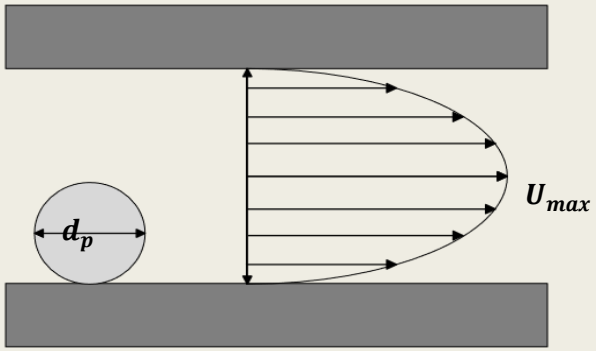
Chrysikopoulos and Katzourakis, Water Resources Research, Volume: 51, Issue: 6, Pages: 4668-4683, First published: 21 May 2015, DOI: (10.1002/2014WR016094)

Hydrodynamic dispersion of nanoparticles in porous media: is there a size effect?

- Velocity profile exclusion:

$$U_{EPV} = \bar{U}\left[1 + \frac{d_P}{R} - \frac{1}{4}\left(\frac{d_P}{R}\right)^2\right]$$

$$D_{ETD} = D + \frac{1}{192}\frac{U_{max}^2 R^2}{D}\left(1 - \frac{d_P}{R}\right)^6$$



- Hindered diffusion:

$$D_{HD} = D_{AB}F_1(\varphi)F_2(\varphi)$$

$$F_1(\varphi) = \frac{Flux\ area\ available\ to\ solute}{total\ flux\ area} = \frac{\pi\left(d_{pore} - d_p\right)^2}{\pi {d_{pore}}^2} = (1-\varphi)^2$$

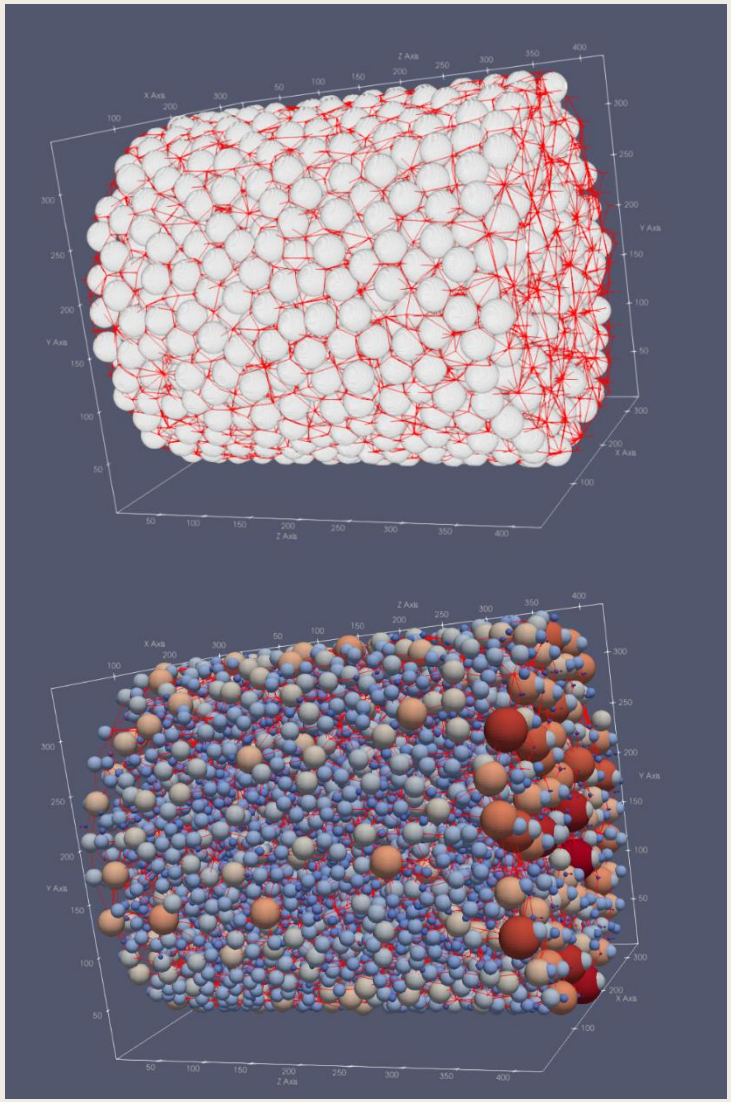
$$F_2(\varphi) = 1 - 2.104\varphi + 2.09\varphi^3 - 0.95\varphi^5$$

Pore network model extraction

- Replicate the experimental system of Chrysikopoulos & Katzourakis (2015)

Extracted pore networks with Lc= 15 cm, Dc = 2.5 cm, and D_b = 2 mm

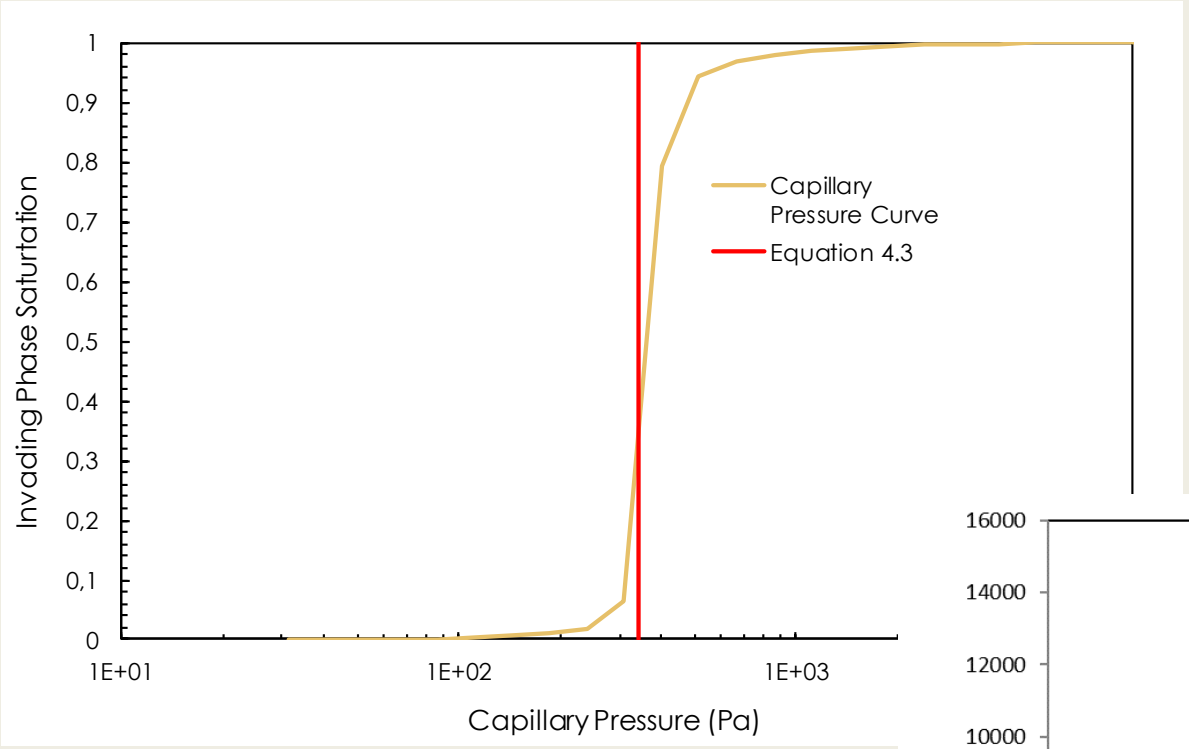
	Column #1	Column #2
Bead Count	10916	10918
ε	0.390	0.390
k_S (D)	3598	3607
k_{KC} (D)	4072	4173
τ	1.43	1.42



Pore network models at different scales extracted from column #1.

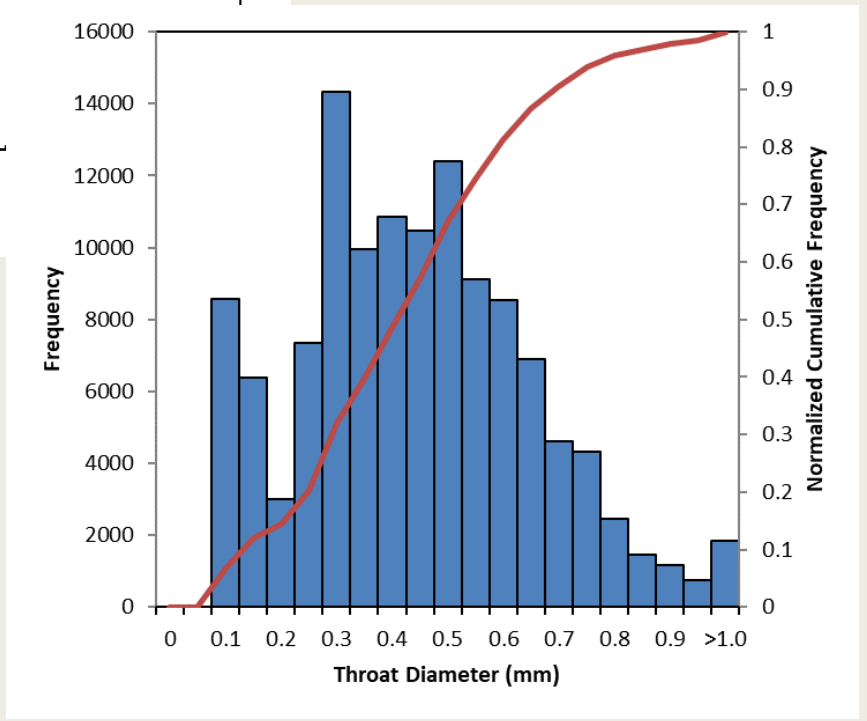
	Ratio to Column #1	d_b (mm)	k_s (D)
Column #1	1	2	4072
Column #3	0.1	0.2	40.72
Column #4	0.04	0.08	6.51
Column #5	0.02	0.04	1.63
Column #6	0.01	0.02	0.407
Column #7	0.002	0.004	0.0163

Pore network model validation



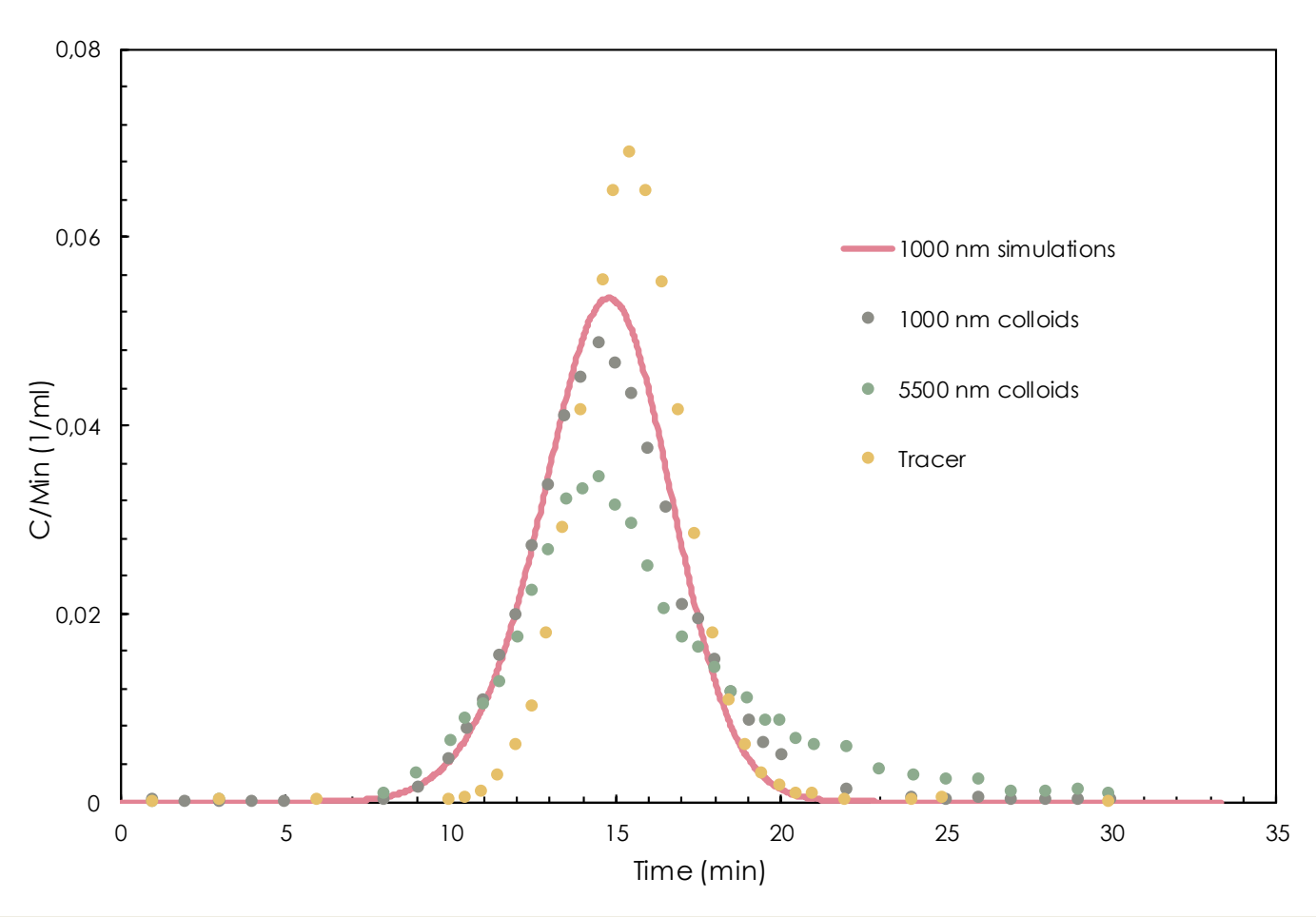
$$p_c^{\circ} = \frac{2\gamma_{air-water}}{R_t^o}$$

$$R_t^o = 0.21 d_b$$

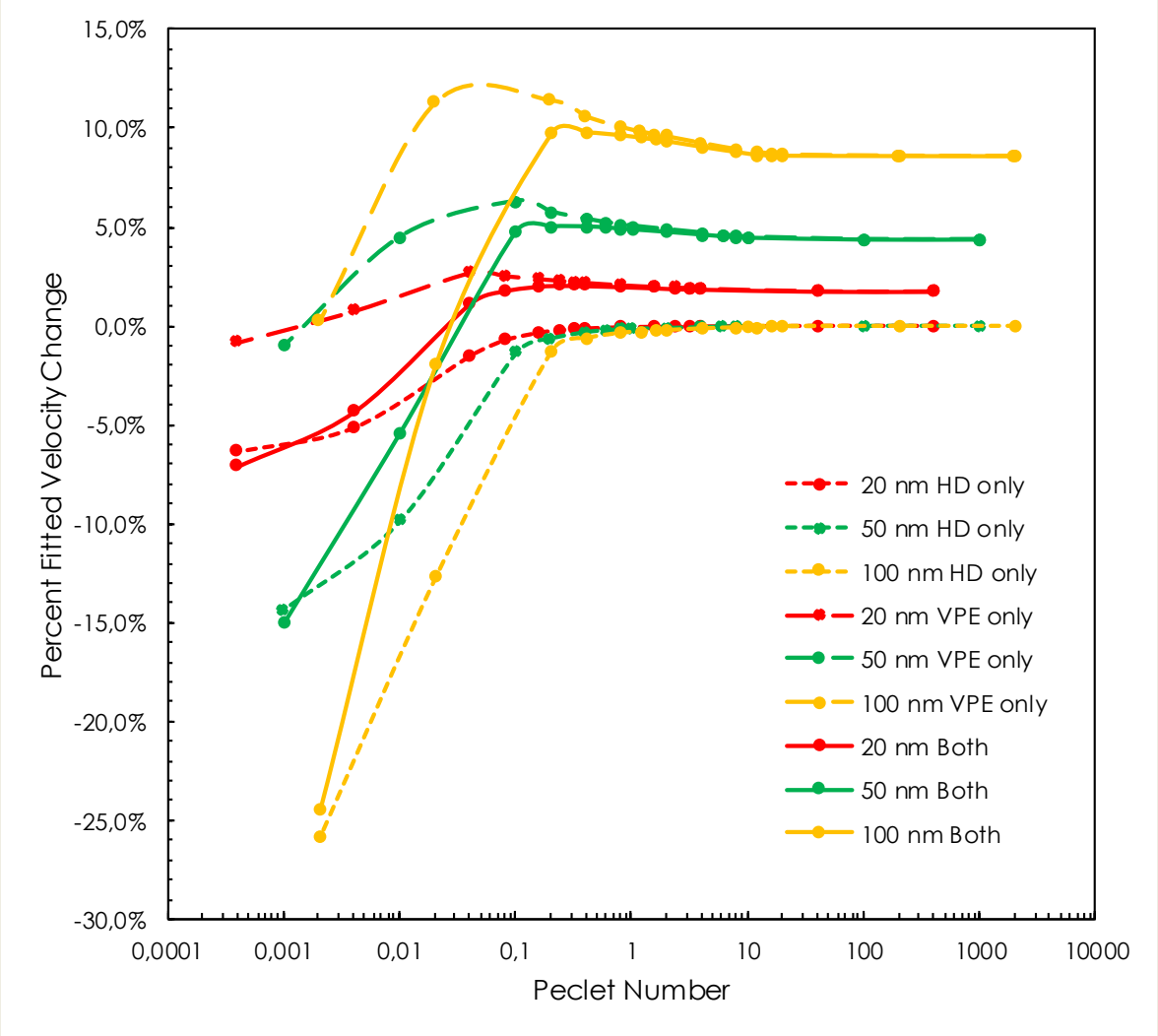


Comparison to experiments

- PNM challenges the conclusions of Chrysikopoulos & Katzourakis (2015)

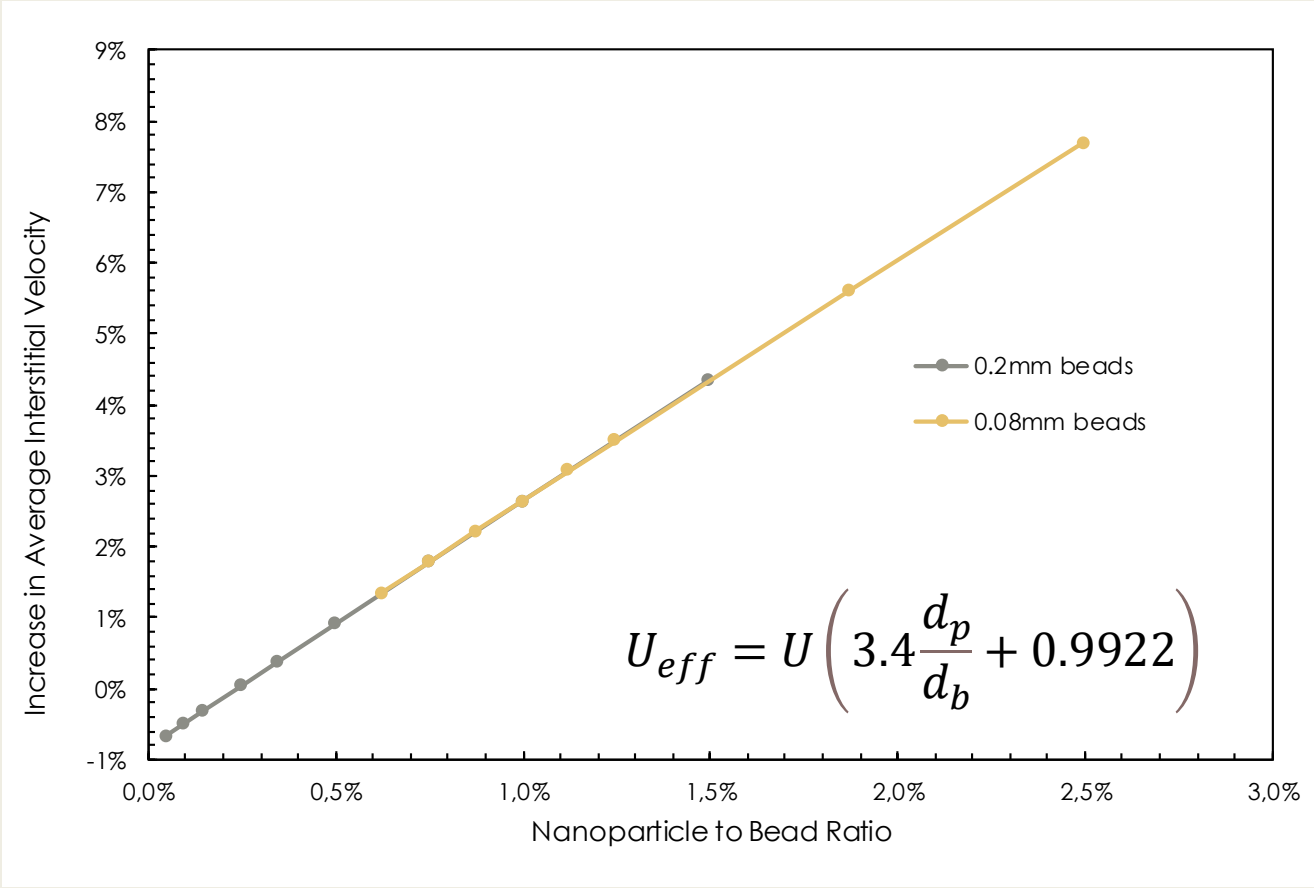


Anticipated effect is real but small

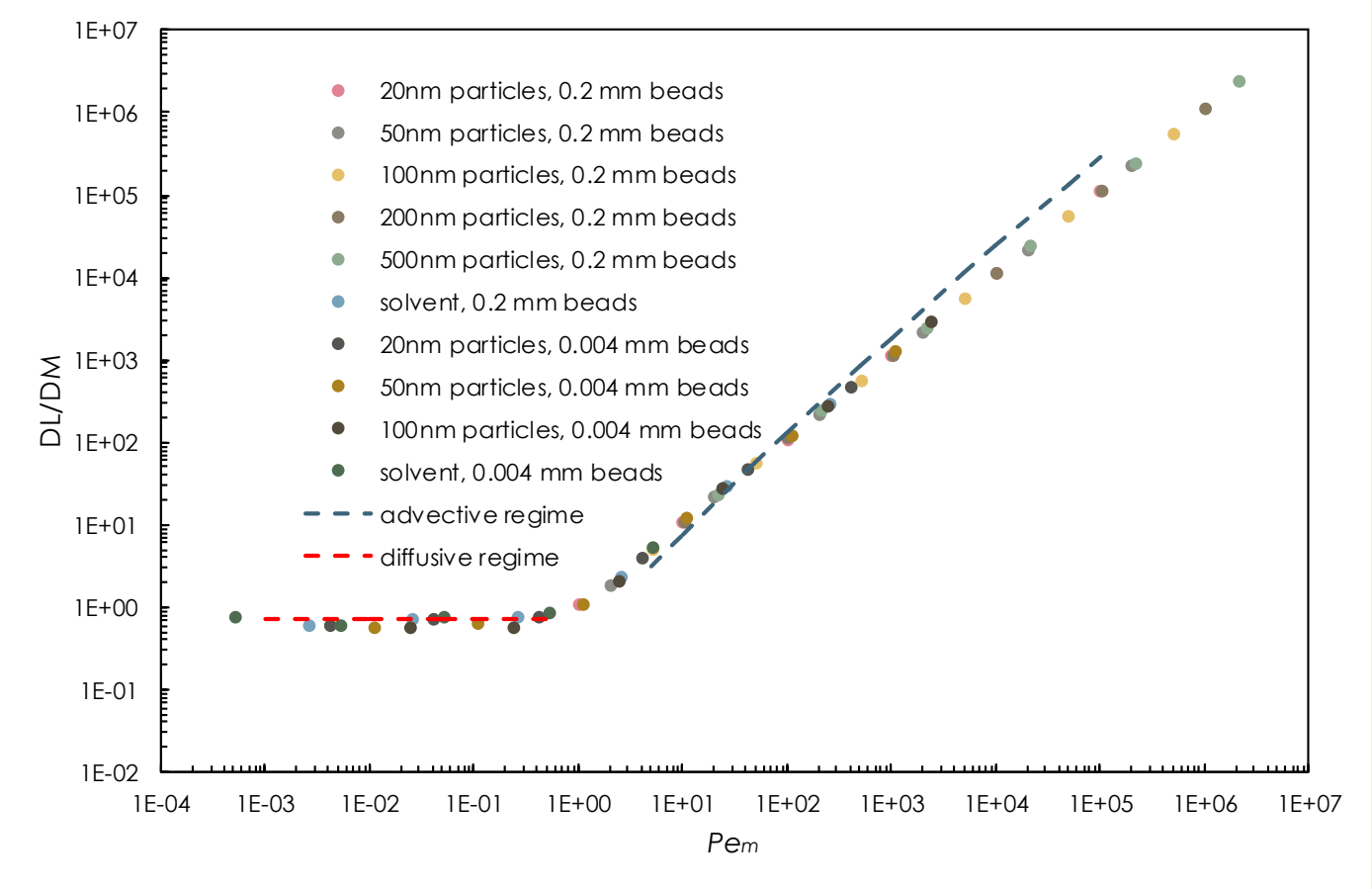


$$d_b = 0.004 \text{ mm}$$

Anticipated effect is real but small



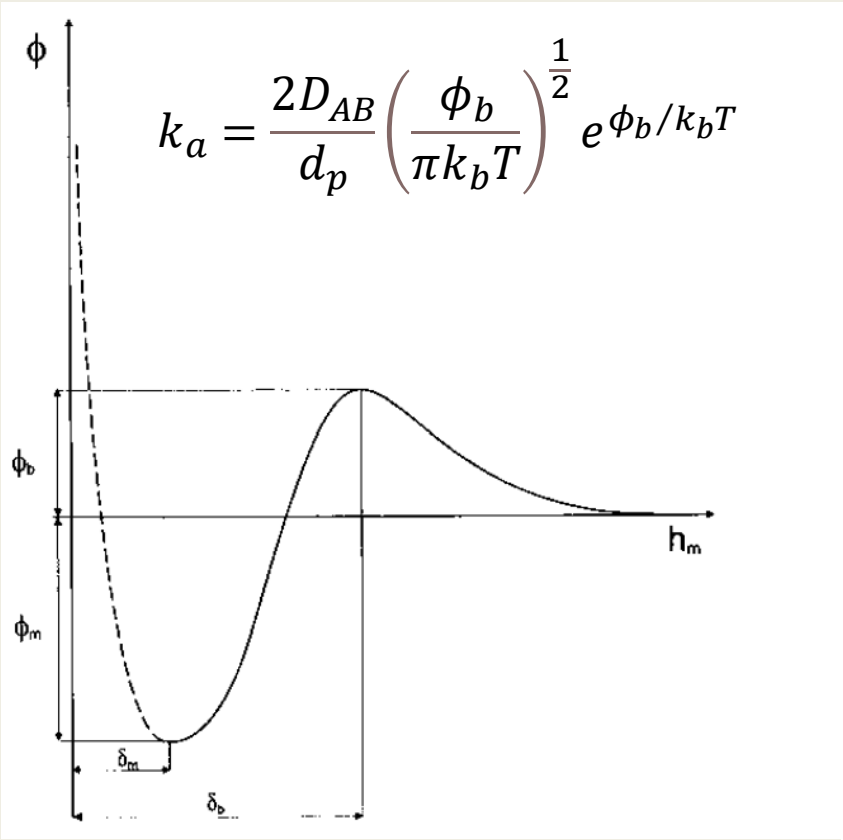
PNM predictions of the dispersion coefficient follow established trends



$$\frac{D_L}{D_m} = \frac{Pe_m}{6\tau}\left[ln\left(\frac{3}{2}Pe_m\right) - \frac{1}{4}\right], \qquad \text{valid for } \frac{Pe_m}{\tau} \gg 1$$

Pore network modeling of nanoparticle transport in porous media: attachment at interfaces

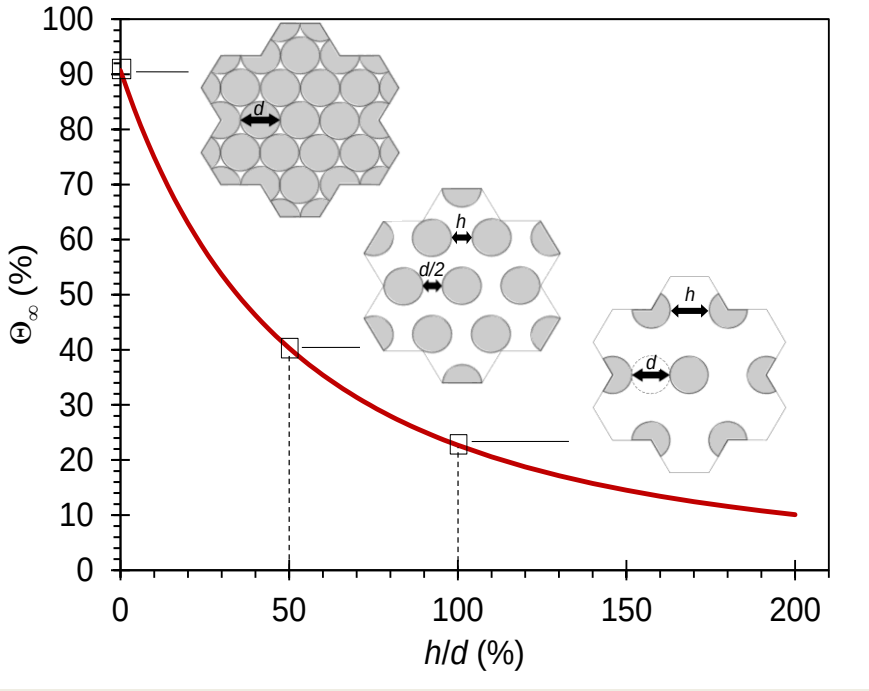
$$k_a = \frac{2D_{AB}}{d_p}\left(\frac{\phi_b}{\pi k_b T}\right)^{\frac{1}{2}} e^{\phi_b/k_b T}$$



$$-\vec{j} = \frac{1}{S}\frac{d\theta}{dt} = k_a C \bar{B}(\theta)$$

$$\theta = \frac{S n_{ads}}{A_{ads}}$$

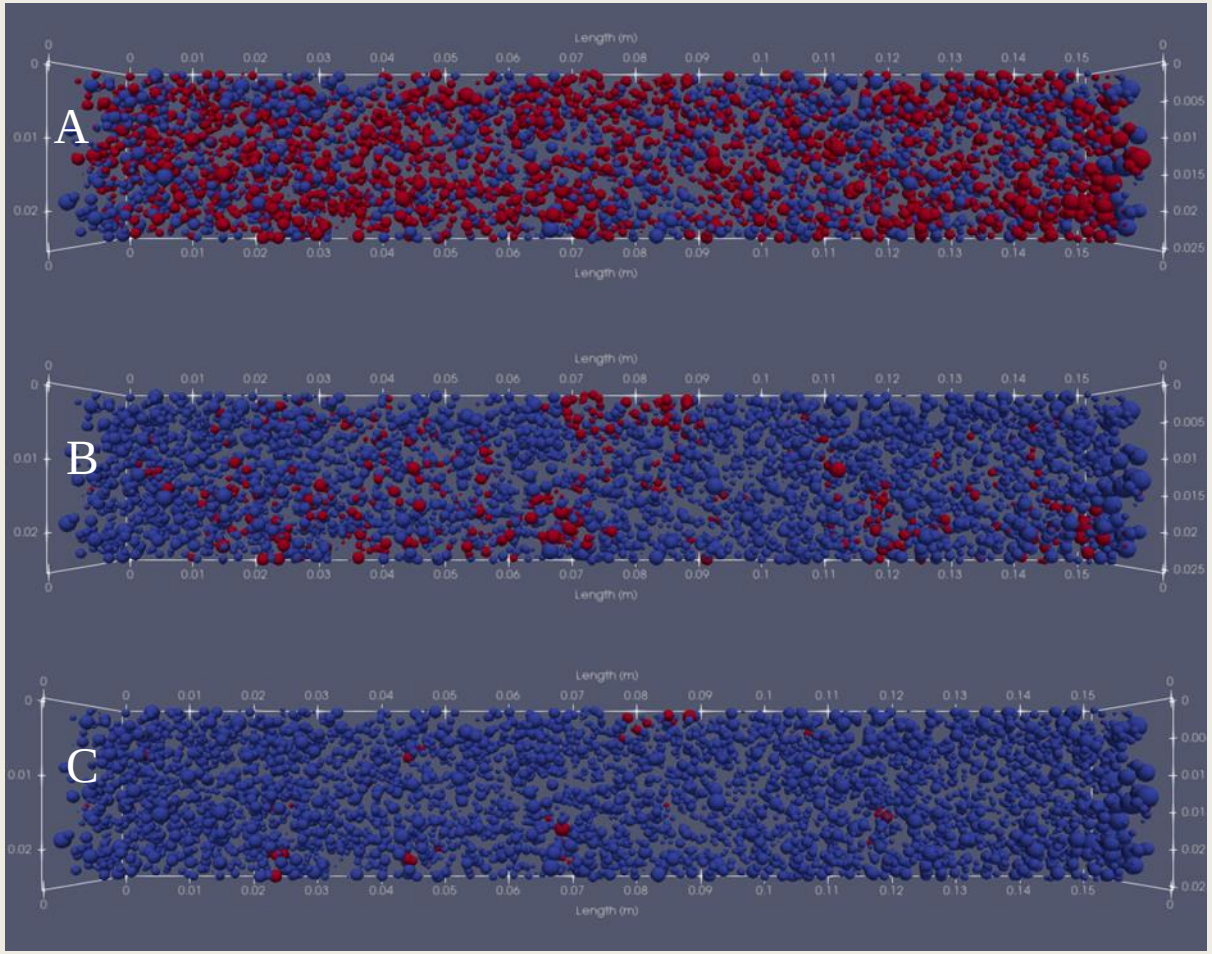
$$\bar{B}(\theta) = 2.32\left(1 - \frac{\theta}{\theta_{max}}\right)^{m}$$



UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Chemical Engineering

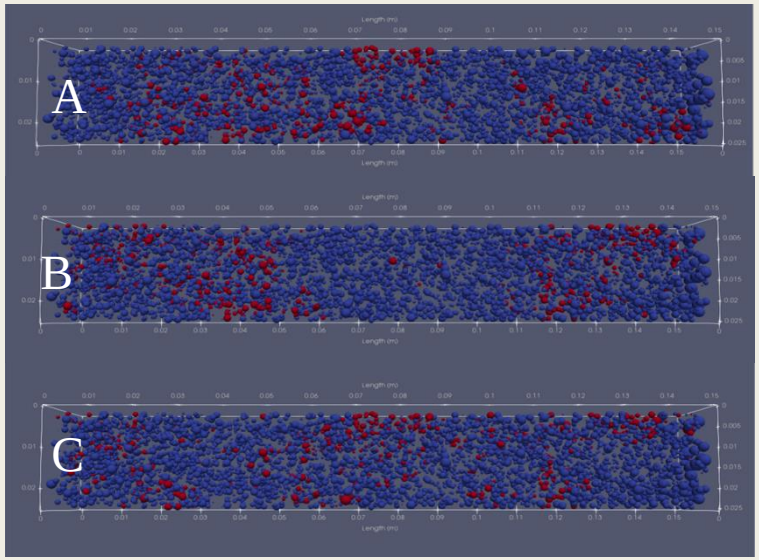
Pore network modeling of nanoparticle transport in porous media: attachment at interfaces

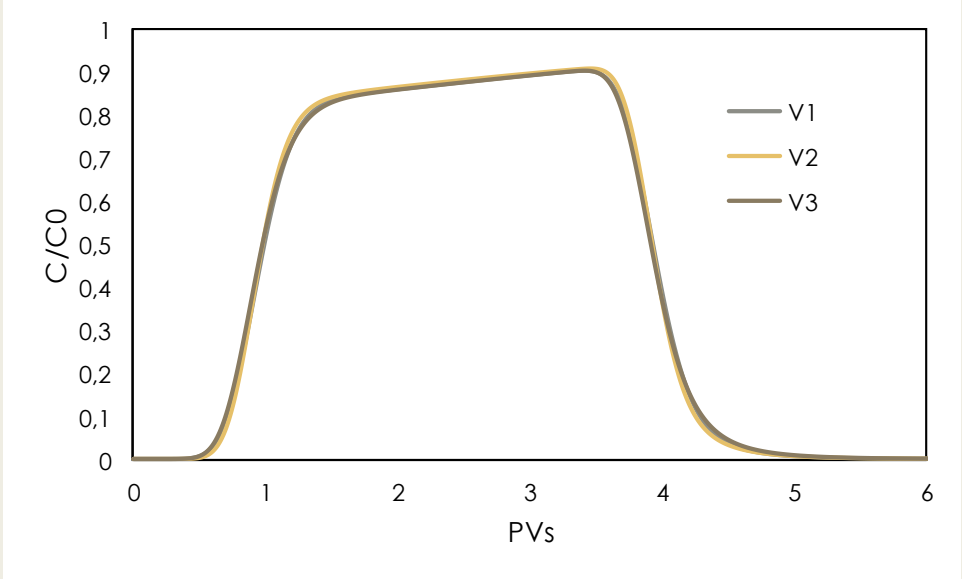
Pore network extracted from a 15-cm packed column with water phase shown in blue and NWP shown in red where $\boldsymbol{S_{NWP} = 0.5}$ in (a), $\boldsymbol{S_{NWP} = 0.1}$ in (b), and $\boldsymbol{S_{NWP} = 0.01}$ in (c).



Pore network modeling of nanoparticle transport in porous media: Reproducibility

Pore Network	L	M1	M2	M3	H
S_{NWP}	0.01	0.1	0.1	0.1	0.5
$S_{NWP,calc}$	0.013	0.115	0.115	0.115	0.588
ε_{eff}	0.391	0.350	0.350	0.350	0.163
$A_o\ (m^2)$	0.0018	0.0140	0.0139	0.0140	0.0561
$a_o(m^{-1})$	62.4	535.4	531.3	534.6	3850.6

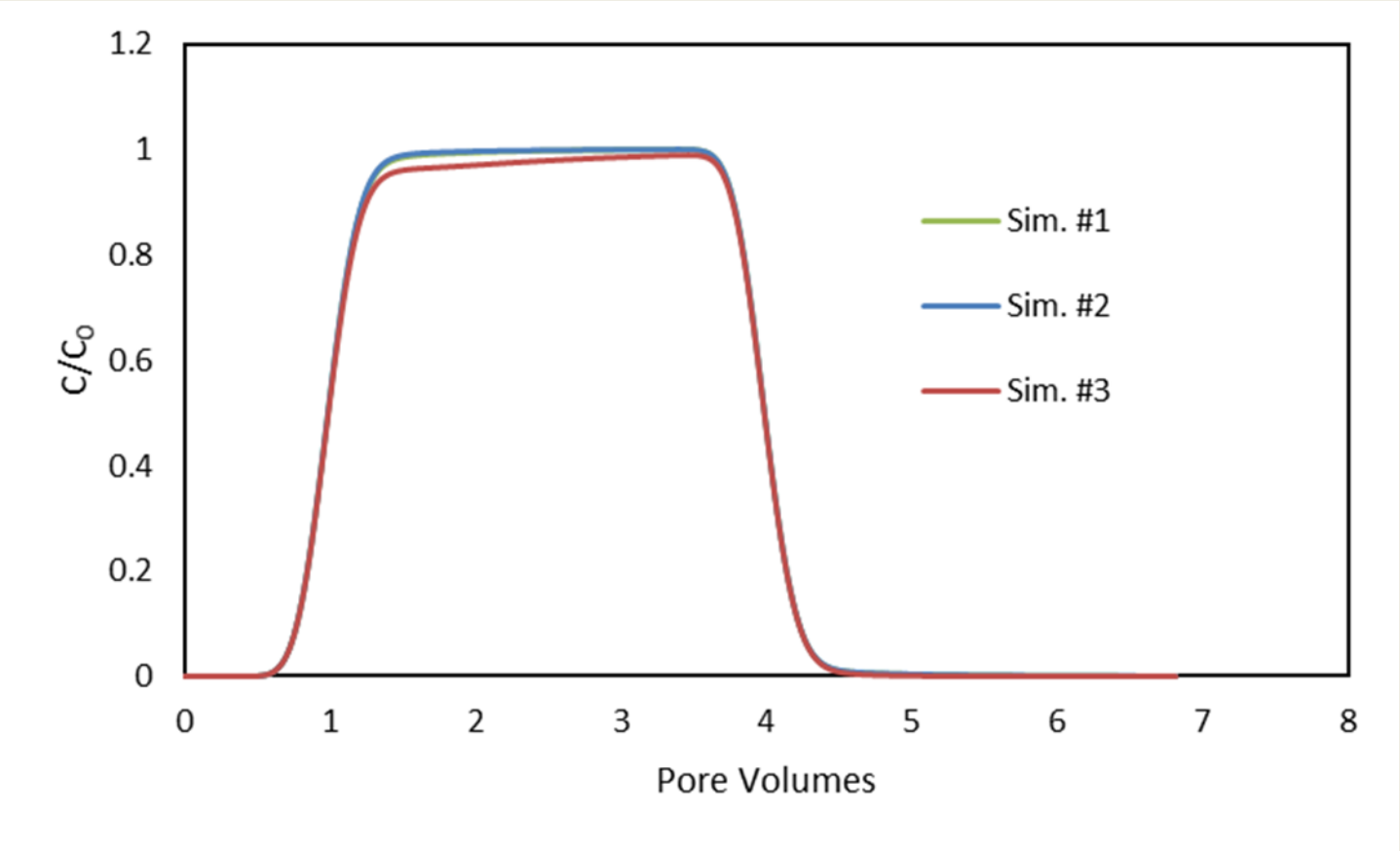




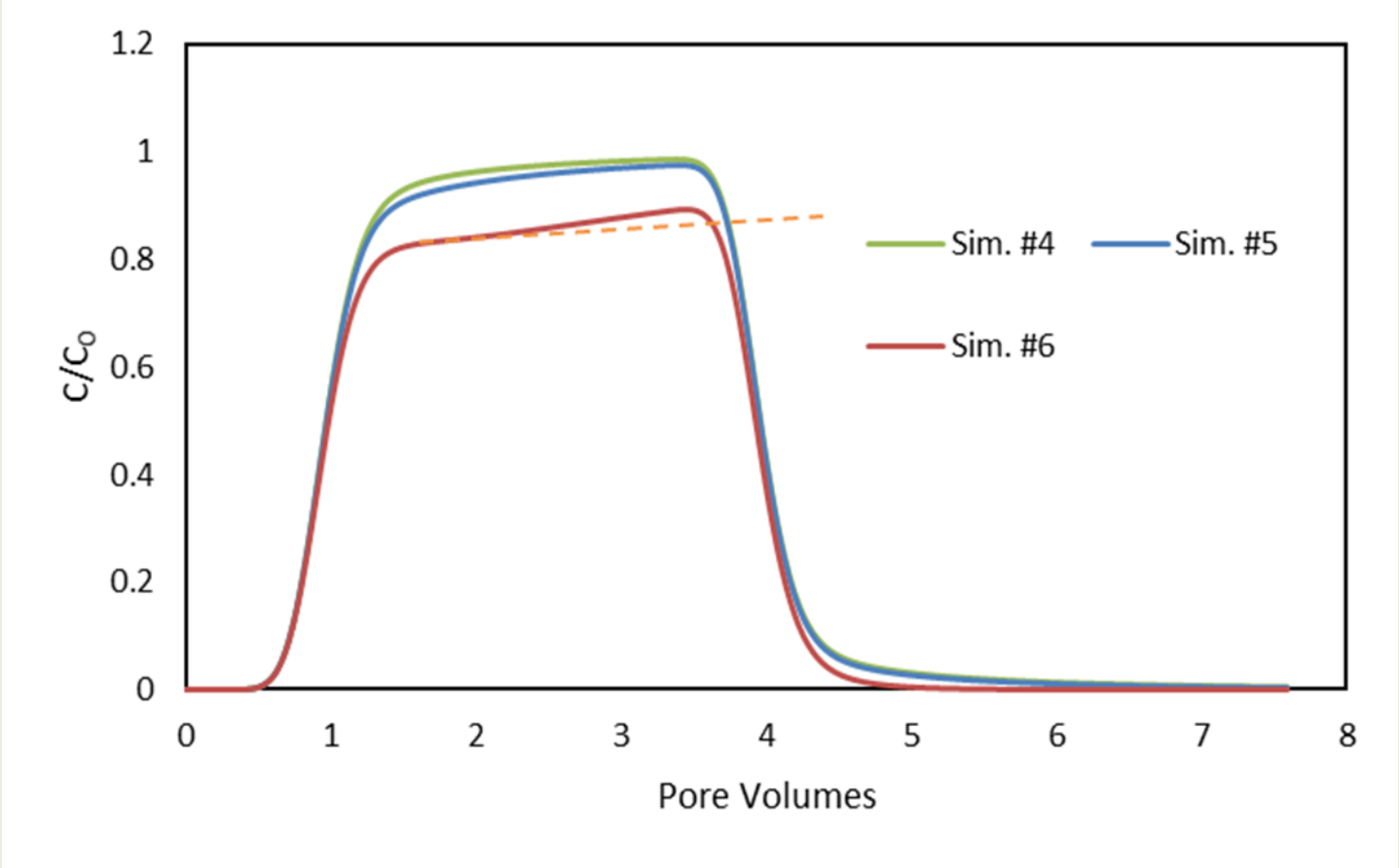
Pore network modeling of nanoparticle transport in porous media: Parametric study

Simulation Number	S_{NWP}	ϕ_b/k_bT	Φ_V	M_r	θ_{total}
1	0.01	50	10^{-4}	1.000	0.000
2	0.01	8	10^{-4}	0.999	0.889
3	0.01	8	10^{-5}	0.980	0.772
4	0.1	50	10^{-4}	0.998	0.000
5	0.1	8	10^{-4}	0.978	0.875
6	0.1	8	10^{-5}	0.871	0.530
7	0.5	50	10^{-4}	0.947	0.000
8	0.5	8	10^{-4}	0.856	0.613
9	0.5	8	10^{-5}	0.684	0.150

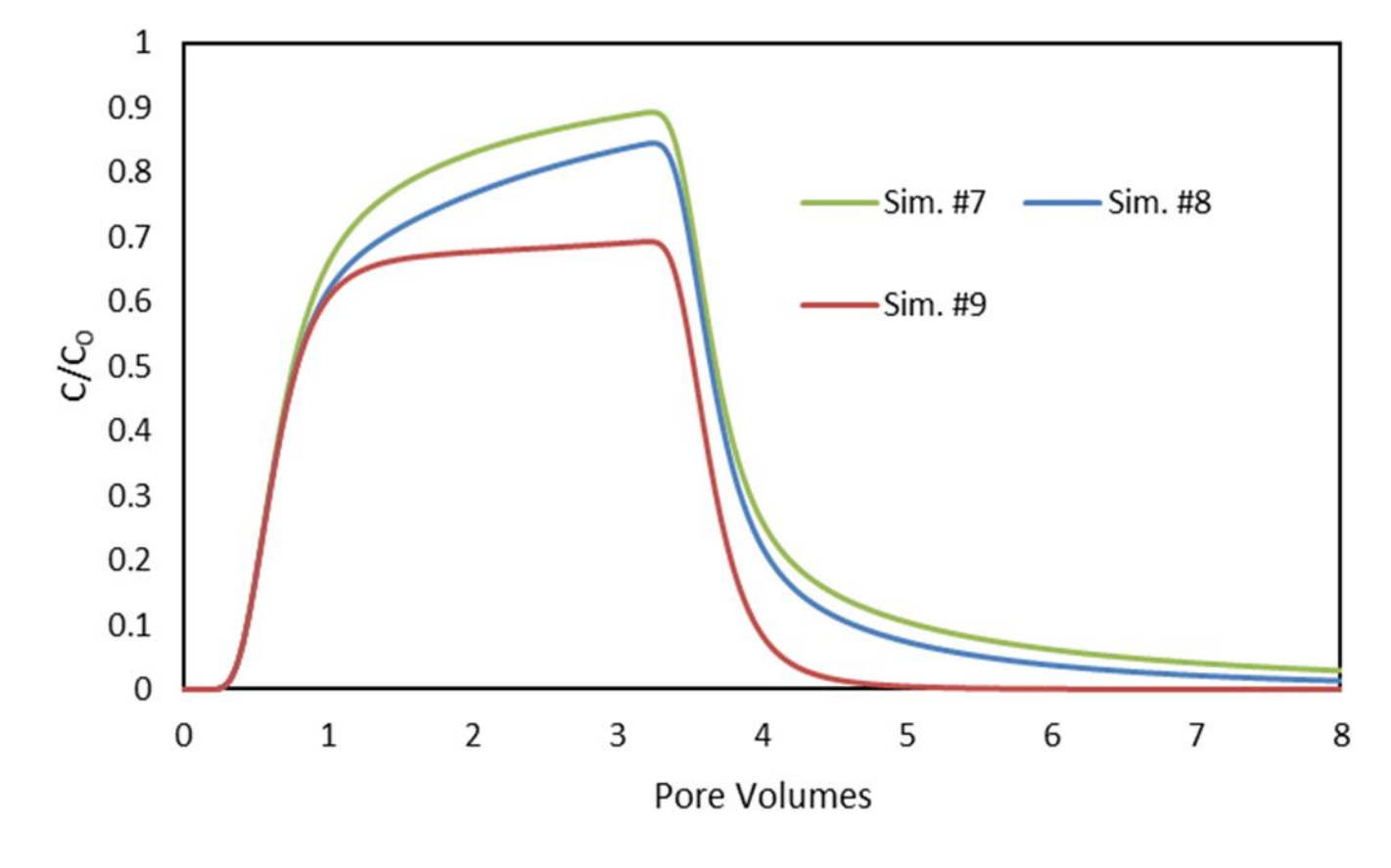
Pore network modeling of nanoparticle transport in porous media: S_{NWP} = 0.01



Pore network modeling of nanoparticle transport in porous media: S_{NWP} = 0.10



Pore network modeling of nanoparticle transport in porous media: S_{NWP} = 0.50



Continuum scale modeling of nanoparticle transport in porous media: Theory

- 1-D advection, dispersion and irreversible attachment

$$\frac{\partial \bar{N}_b}{\partial \bar{t}} = \frac{\partial^2 \bar{N}_b}{\partial \bar{x}^2} - Pe\frac{\partial \bar{N}_b}{\partial \bar{x}} - D_{a_I}\overline{N_b}\underbrace{B_i(\theta_i)}_{\text{Blocking function on fluid interface}}$$

$$\frac{\partial \theta_i}{\partial \bar{t}} = \frac{k_i\, L^2}{D_h}\, N_{b0}\pi r^2 \overline{N_b}\, B_i(\theta_i) = A_r D_{a_I}\, \overline{N_b}\, B_i(\theta_i)$$

$$\bar{x} \equiv \frac{x}{L}, \overline{N_b} \equiv \frac{N_b}{N_{b0}}, \bar{t} \equiv \frac{D_h\, t}{L^2}$$

Area ratio

Initial concentration

Particle radius

$$A_r = \frac{N_{b0}\pi r^2}{a_0}$$

Interfacial area

Damköhler number

Adsorption constant

$$D_{a_I} = \frac{k_i\, L^2 a_0}{D_h}$$

Péclet number

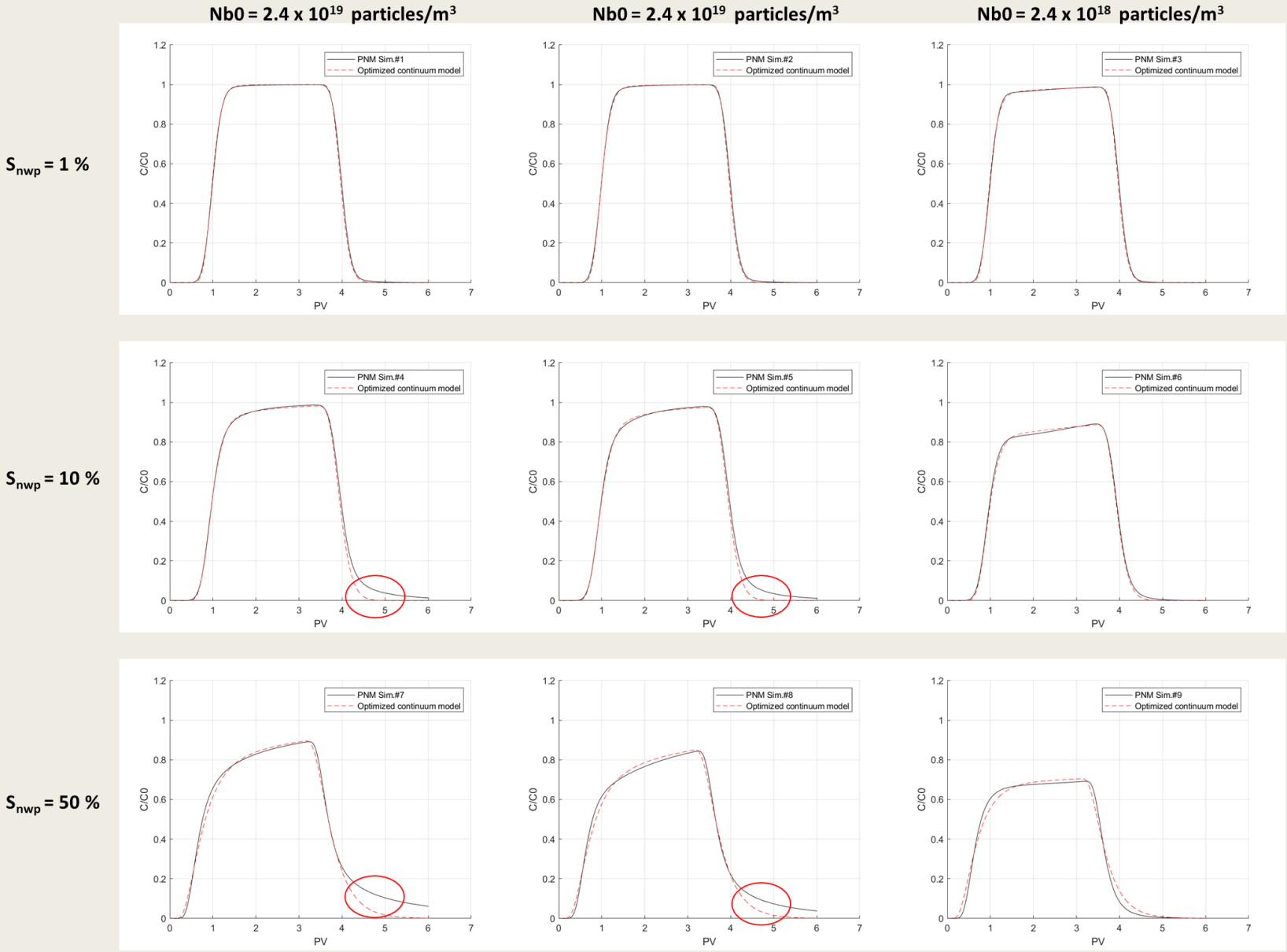
Porous medium length

Pore velocity

$$Pe = \frac{v_p\, L}{D_h}$$

Dispersion coefficient

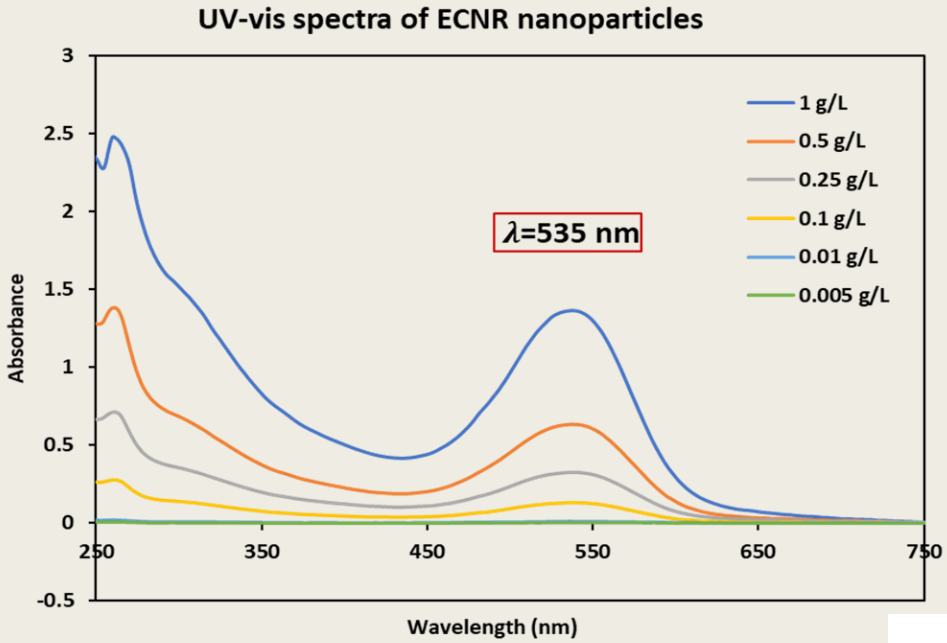
Pore network vs. continuum-scale model: Upscaling

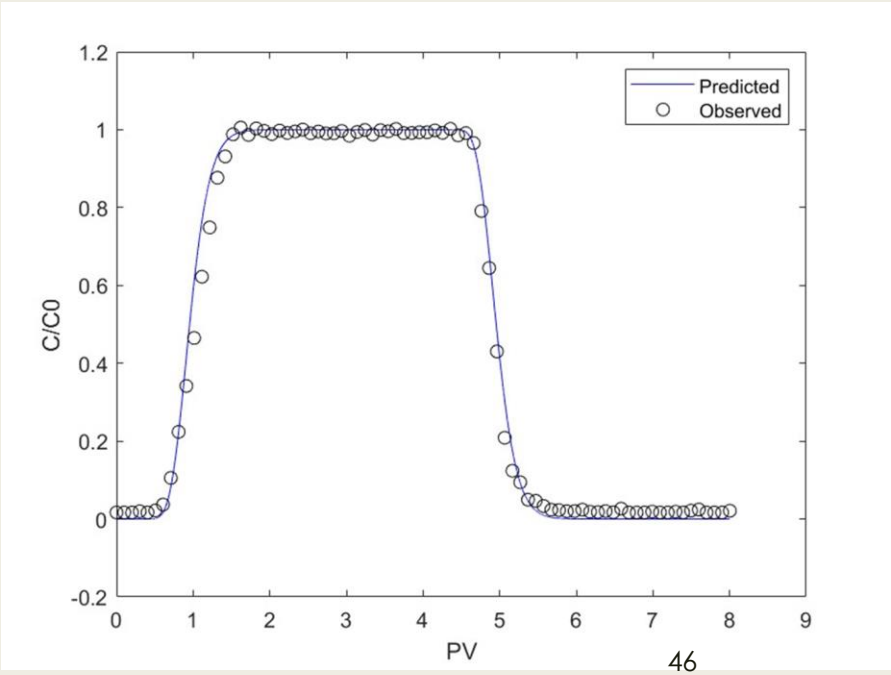


Pore network vs. continuum-scale model: Upscaling

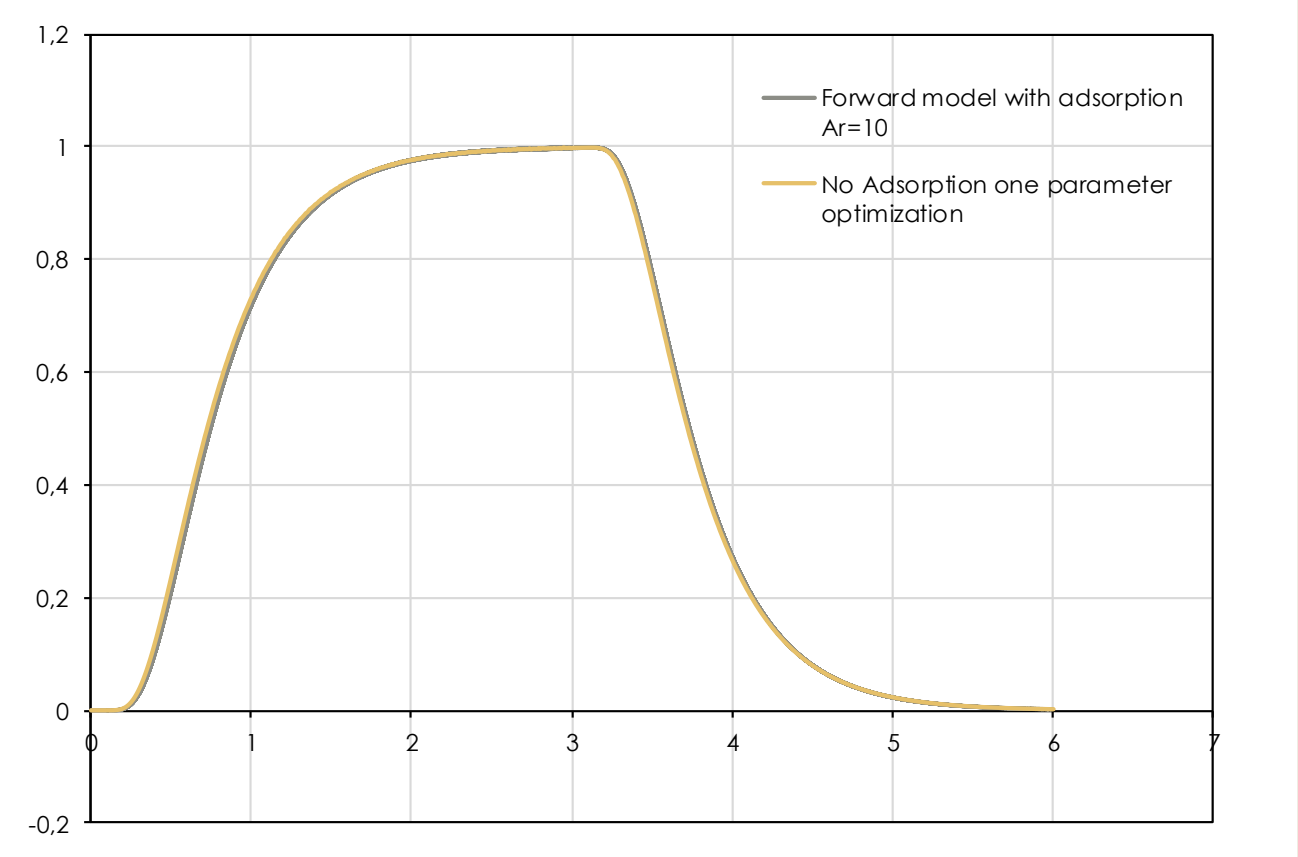
simulation number	a0 (1/m)		k (m/s)	ki (m/s)	a0*ki (1/s)		Dh (m^2/s)	Pe	Da	Ar	
	PNM	Continuum	PNM	Continuum	PNM	Continuum	Continuum	Continuum	Continuum	Continuum	PNM
1	63.53	145.5	2.18E-24	1.93E-08	1.38E-22	2.80E-06	5.95E-09	64.52	10.59	51.53	118.04
2	63.53	299.5	1.52E-06	7.12E-09	9.66E-05	2.13E-06	5.85E-09	65.68	8.20	25.04	118.04
3	63.53	151.9	1.52E-06	2.98E-09	9.66E-05	4.52E-07	5.65E-09	68.01	1.80	4.94	11.80
4	551.9	1940.3	2.18E-24	5.78E-10	1.20E-21	1.12E-06	1.10E-08	38.98	2.30	3.86	13.59
5	551.9	2655.8	1.52E-06	6.00E-10	8.39E-04	1.59E-06	1.12E-08	38.20	3.20	2.82	13.59
6	551.9	1898.9	1.52E-06	8.80E-10	8.39E-04	1.67E-06	1.04E-08	41.01	3.60	0.39	1.36
7	4742	13000.4	2.18E-24	4.00E-10	1.03E-20	5.20E-06	1.46E-07	6.30	0.80	0.58	1.58
8	4742	19576.7	1.52E-06	3.73E-10	7.21E-03	7.30E-06	1.51E-07	6.11	1.09	0.38	1.58
9	4742	23891.7	1.52E-06	3.67E-10	7.21E-03	8.76E-06	1.76E-07	5.23	1.12	0.03	0.16

Experimental validation





Observability of continuum-scale model parameters



Acknowledgement

Work done by graduate students:

Navid Bizmark (PhD)
Stephen Dauphinais (MASc) – Prof. J. Gostick (co-supervisor)
Youssra Rahham (PhD)

Financial support for the research presented provided by

NSERC CRSNG
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Chemical Engineering
RBC
RBC Foundation
Ontario
WATERLOO INSTITUTE FOR nanotechnology
the water INSTITUTE

Participation in this workshop supported by

H.F.R.I.
Hellenic Foundation for Research & Innovation

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment (Project title: Enhanced Oil Recovery by Polymer-coated Nano Particles, acronym: «EOR-PNP», code: HFRI-FM17-361)

FORTH
INSTITUTE OF CHEMICAL ENGINEERING SCIENCES

EORPNP

ΕΛ.Ι.Δ.Ε.Κ.
Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας

Computational Fluid Dynamic models inside 3D digitally represented rocks and soils

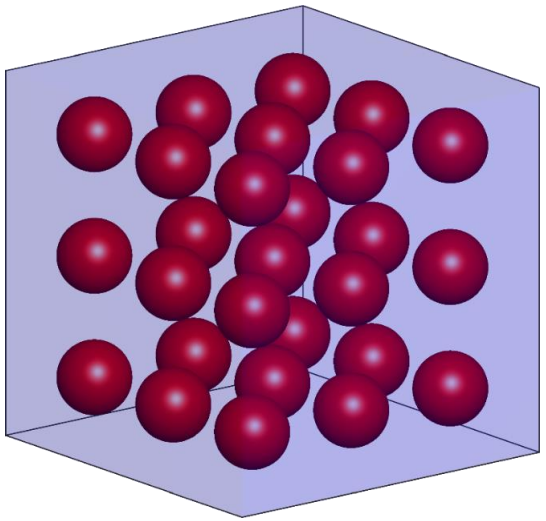
Nadia Bali

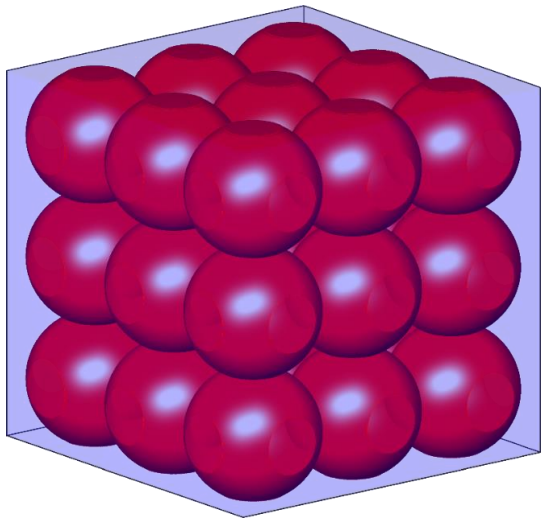
Outline

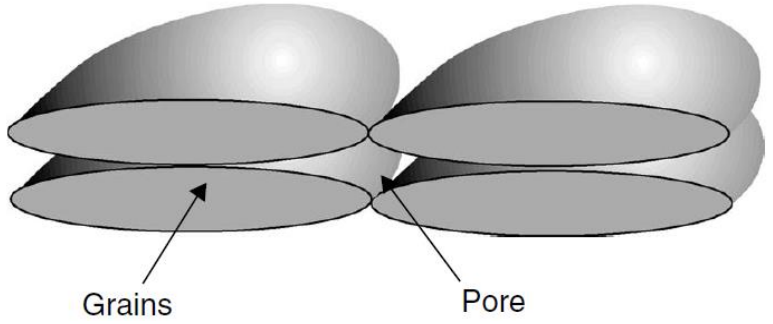
- Structures characteristics
- Numerical reconstruction
- 3 different computational fluid dynamic problems
- Effective properties
- Simulations inside real reconstructed geometries

Structure characteristics

- Pore structure and packing







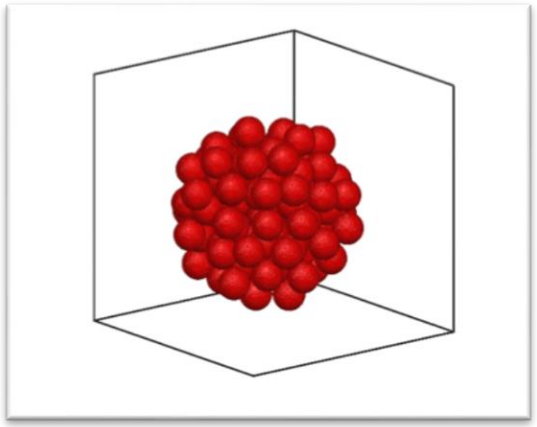
Ellipsoid packing -clay

- Single arrangement of spherical particles
- FCC structure lead to minimum porosity 0.48

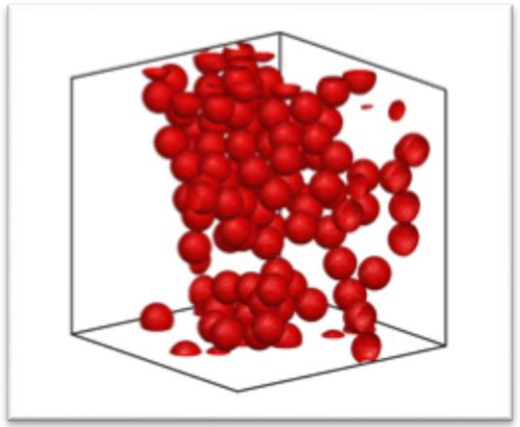
Structures characteristics

- Pore size
- Porosity
- Grain size distribution
- Tortuosity

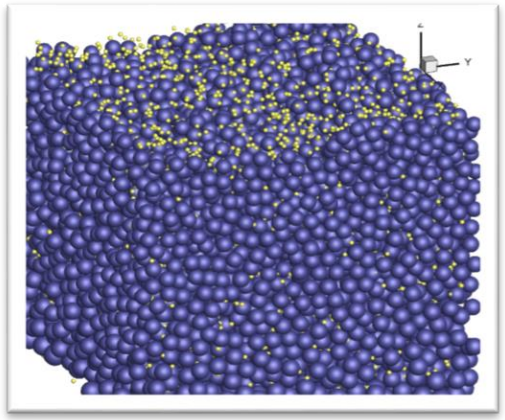
$$\varepsilon_p = 1 - NV_{sp}/V_{tot}$$



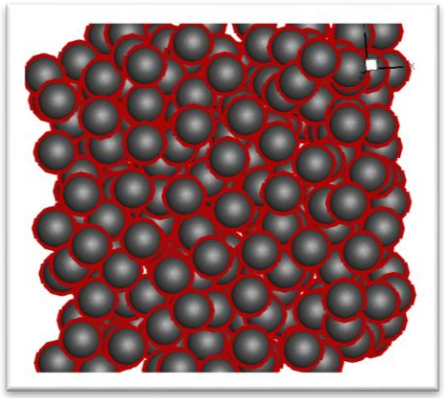
Ballistic Aggregation



1 point contact- Randomized structure with 2-symmetrical constrains



Ballistic Deposition of 2 different spheres' size

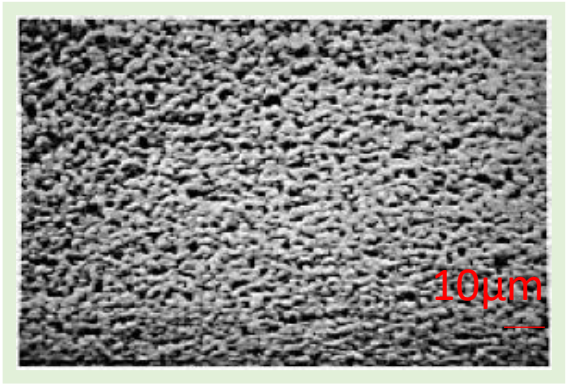


Ballistic Deposition with a spherical cover (overlapping ring)

Numerical reconstruction

- Setting the goals:

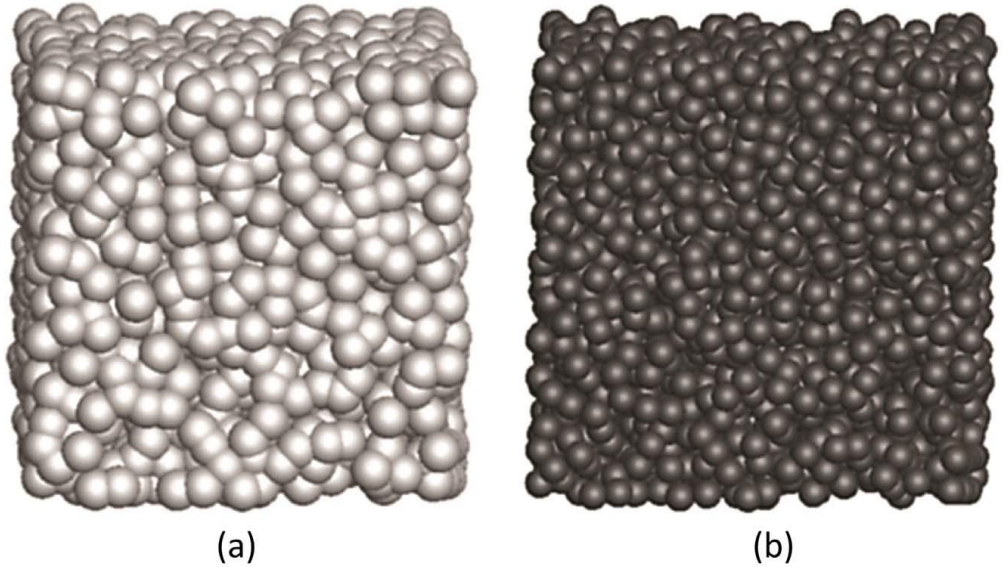
1. Porosity value
2. Particles density
3. Optically resemblance



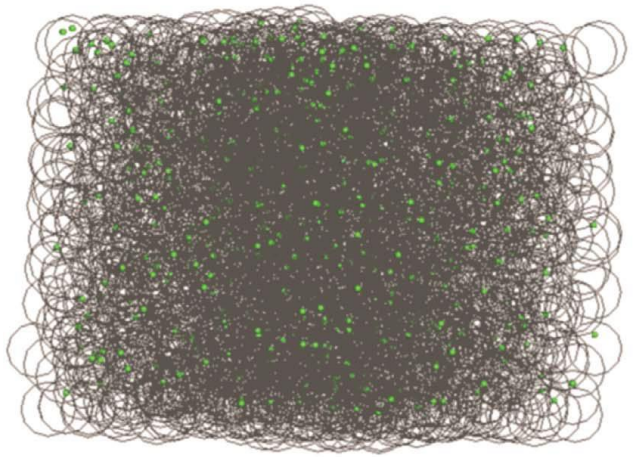
SEM image 10µm
Porosity=0.2

Algorithms

- Stochastic algorithms
  a) Monte Carlo free
  b) Monte Carlo cherry-pit
  c) Ballistic deposition



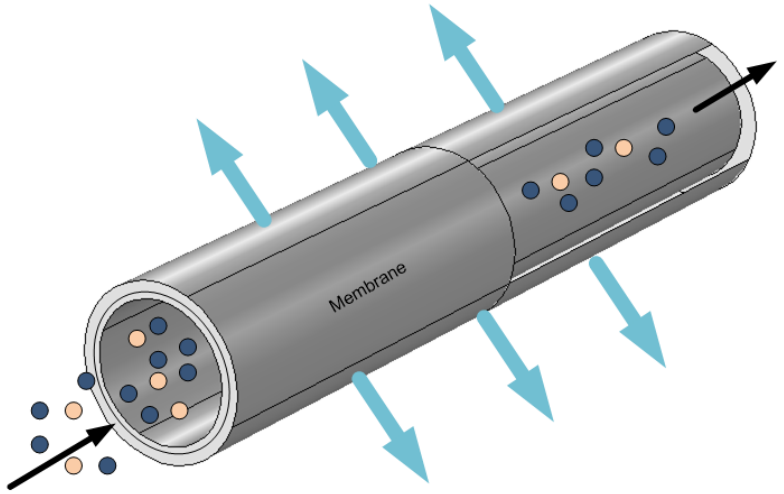
(a) (b)



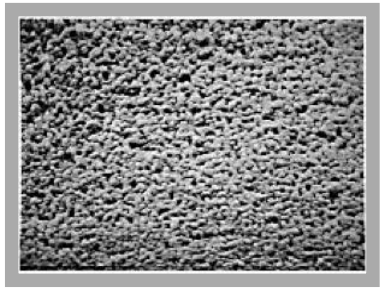
(c)

Example 1: Problem of lactose transport inside porous membrane

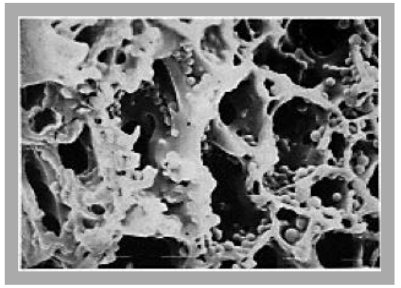
- Problem of lactose free milk=> conversion of lactose



Dense layer ε=0.2

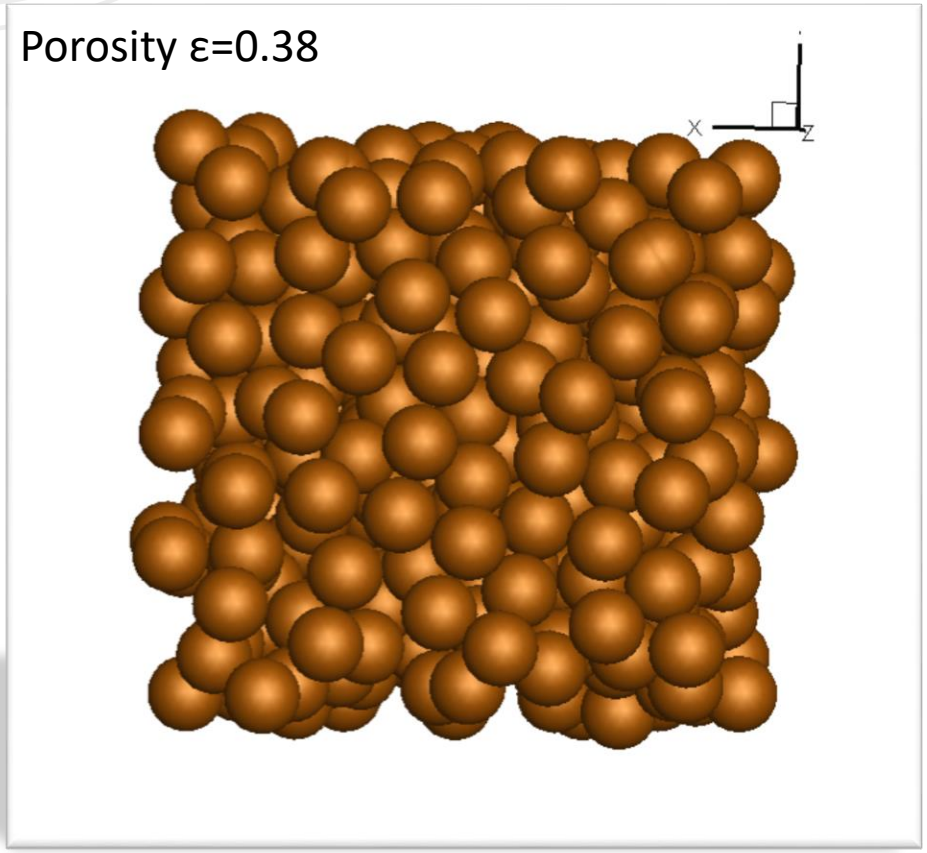


Spongy layer ε=0.67

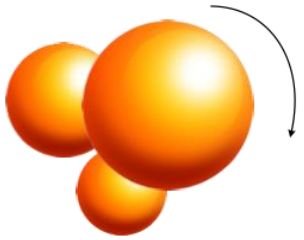


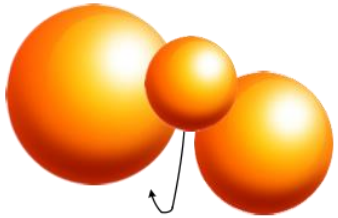
N. Bali, A. Petsi, E. Skouras, V. Burganos, Three-dimensional reconstruction of bioactive membranes and pore-scale simulation of enzymatic reactions: The case of lactose hydrolysis, Journal of membrane science 524 (2017) 225-234.

Ballistic Deposition



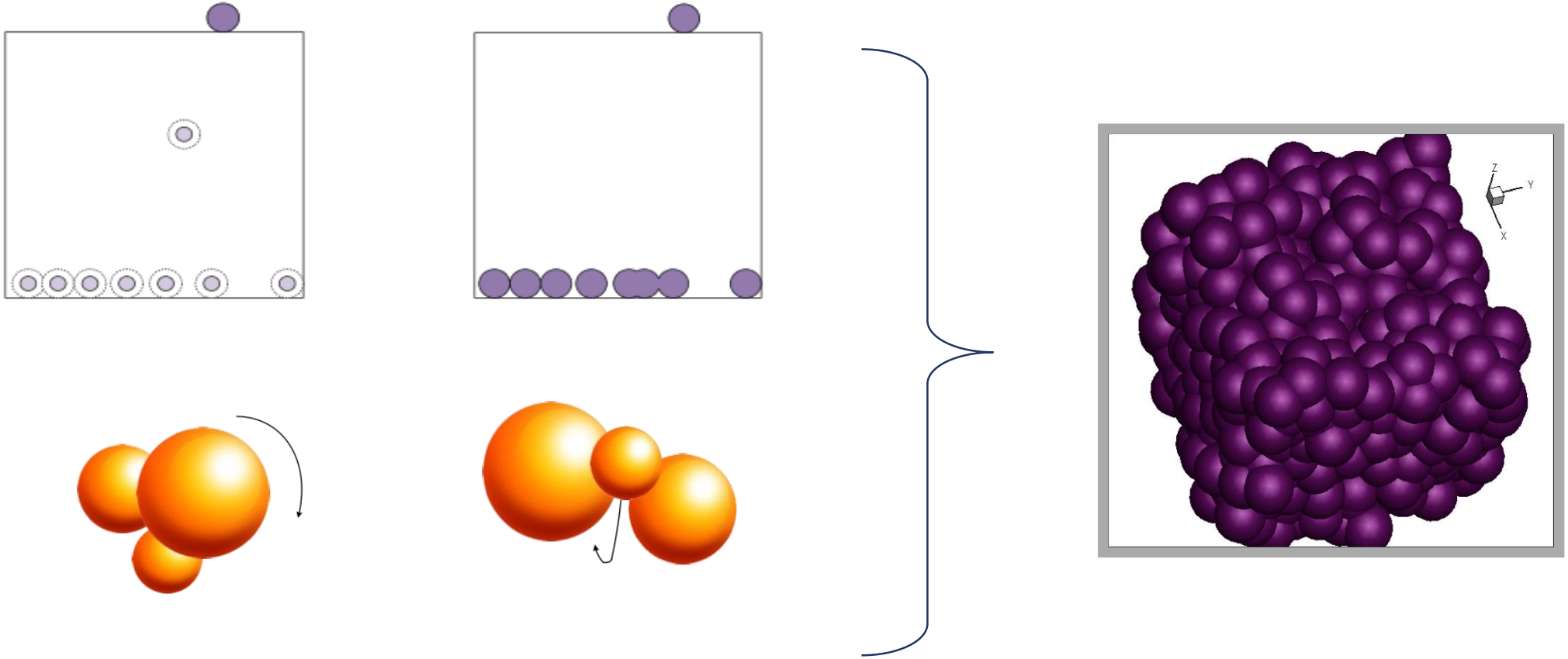
- Non-overlapping spheres
- Porosity=0.38





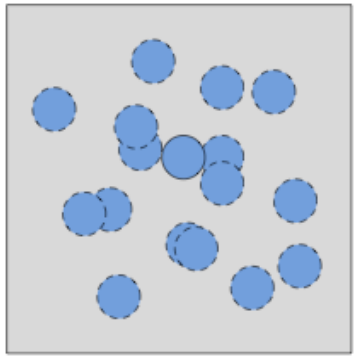
Modified Ballistic Deposition algorithm

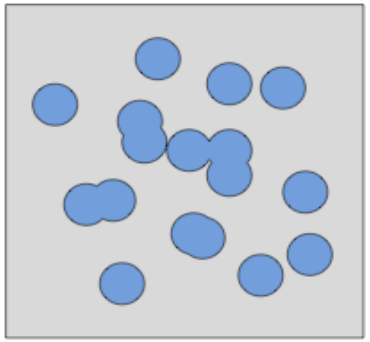
- For the dense layer of the membrane a modified ballistic method can be used

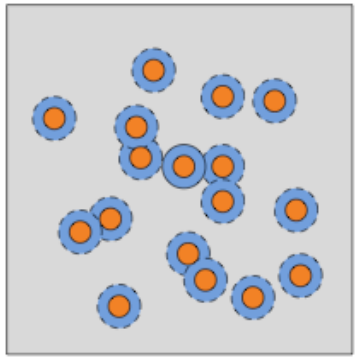


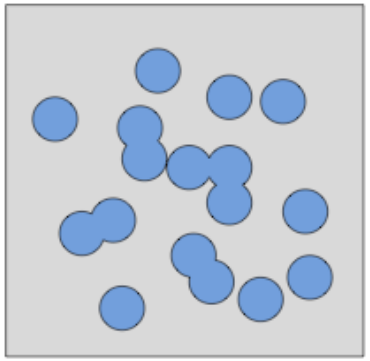
Monte Carlo algorithm

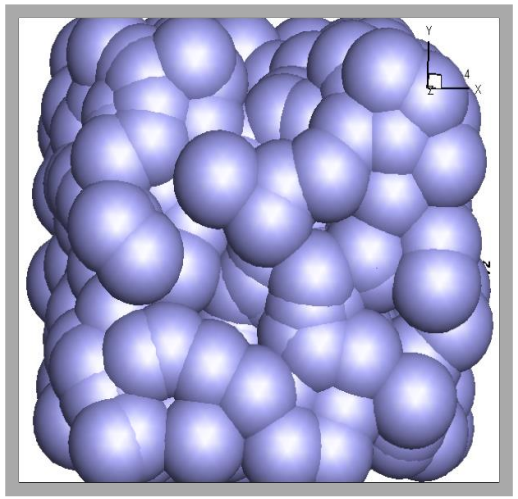
- Monte Carlo cherry pit
- Free Monte Carlo

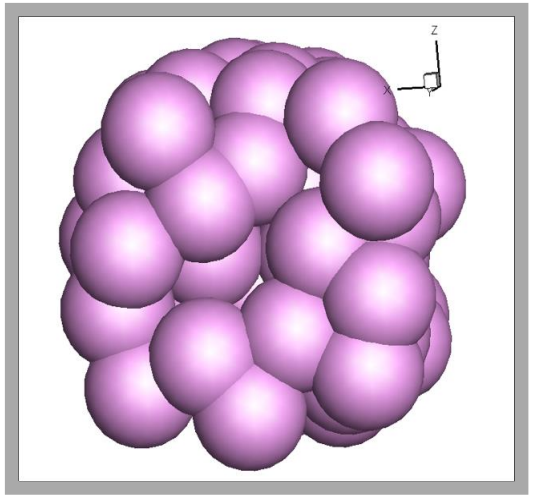




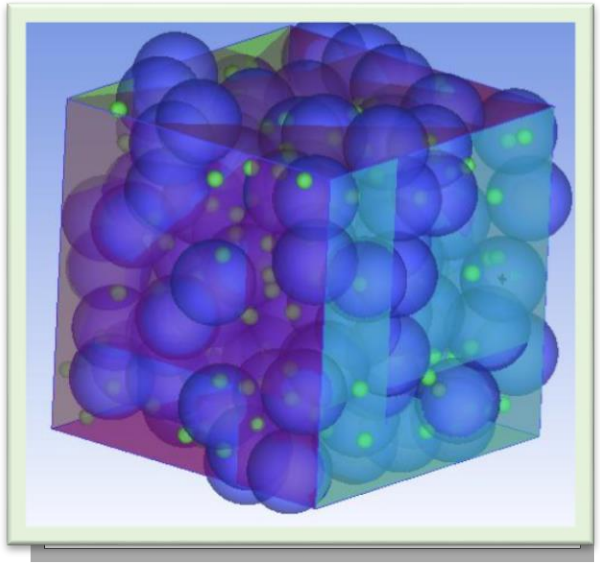








Transport properties



Mem layers	Effective diffusivity (m^2/s)	Effective permeability (m^2)
Layer1	$2.236\ 10^{-11}$	$2.3\ 10^{-19}$
Layer2	$3.796\ 10^{-10}$	$8.1\ 10^{-15}$

$$D = -\frac{\iint_A N dA}{A_{tot}} \frac{1}{C_{out} - C_{in}}$$

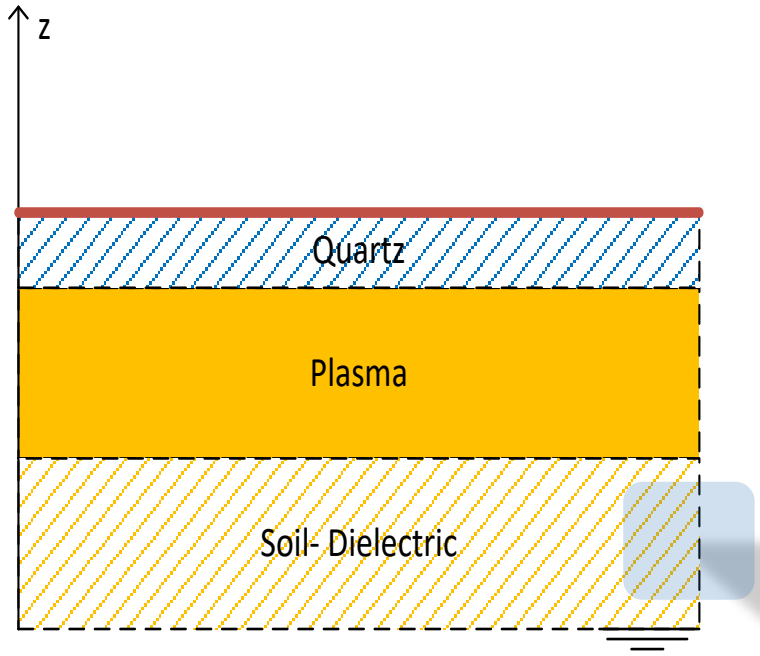
$$K = \frac{\iint u dA}{A} \frac{\mu L}{\left(P_{out} - P_{in}\right)}$$

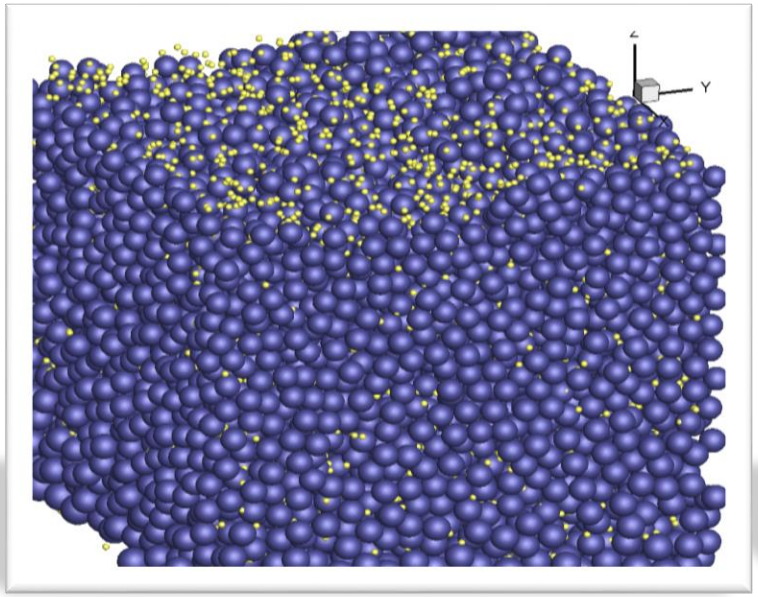
$K_{exp}= 6.7\ 10^{-18} m^2$

$K_{sim}= 6.3\ 10^{-18} m^2$

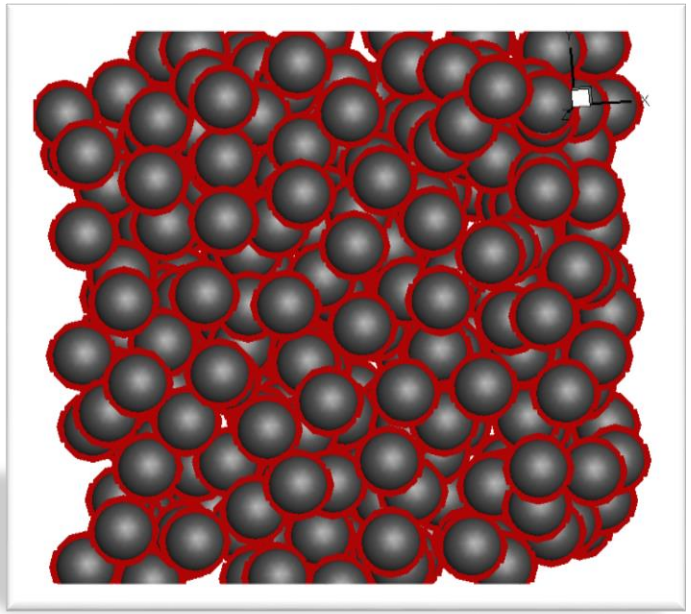
G. Catapano, G.I., E. Drioli, M. Filosa, *Experimental analysis of a cross-flow membrane bio-reactor with entrapped whole cells: influence of trans-membrane pressure and substrate feed concentration on reactor performances* Membrane Science, 1988. **35**: p. 325-338.

Example 2: Soil structure during plasma discharges



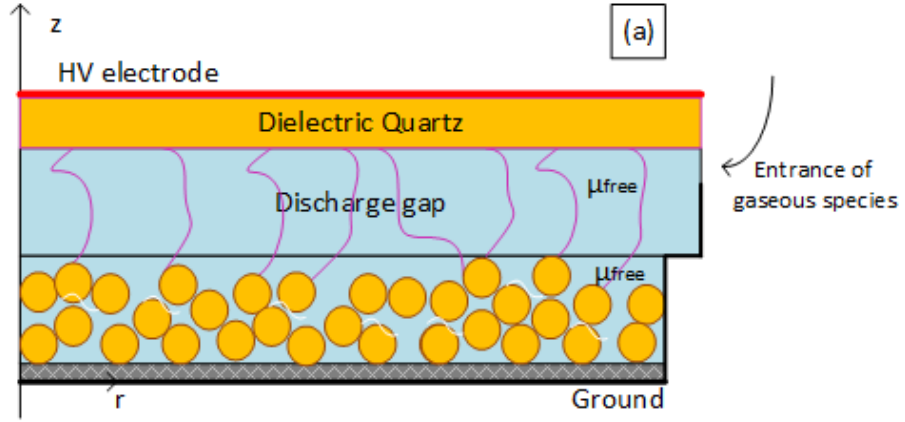


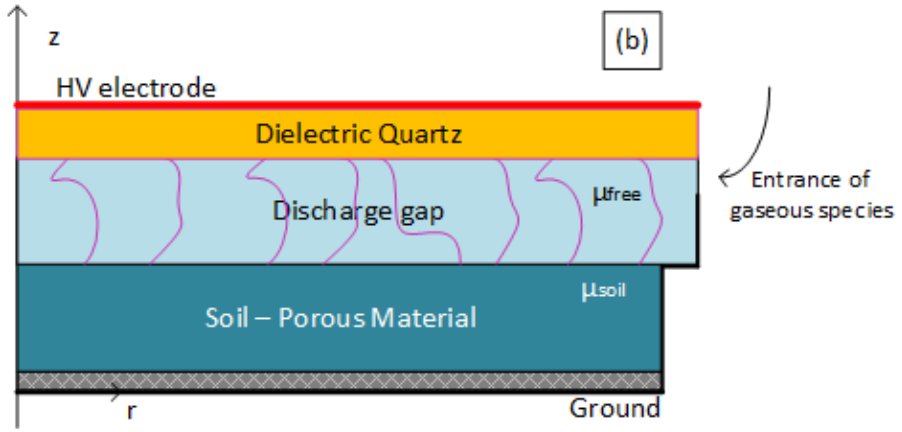
Non-wetting



Wetting

Example 3: Mobility transport



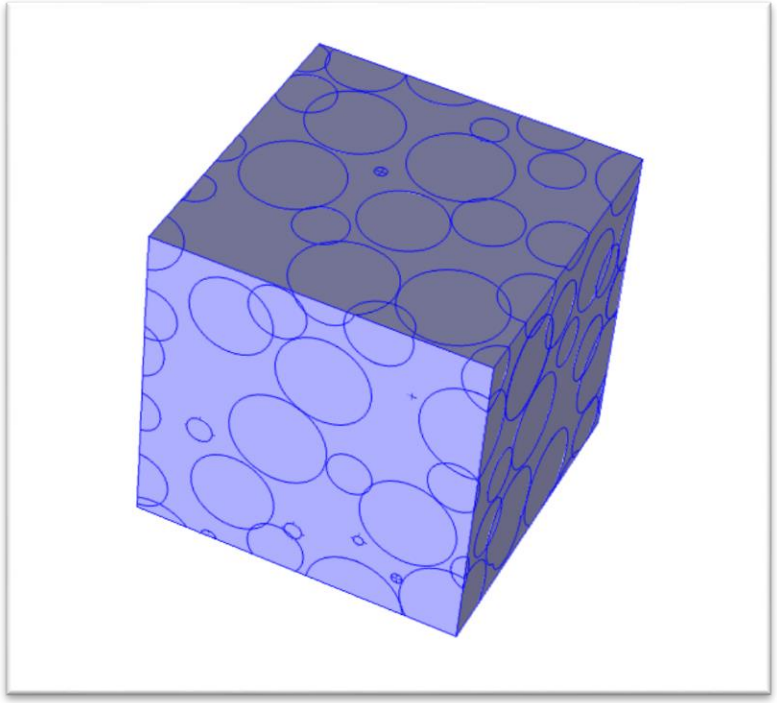


- Modified equations:
  - μ (mobility factor that describes the movement of electrically charged particles) its effective value can be obtained by:

$$m_{soil} = \frac{e_p}{t} m_{free}$$

Bali, N.; Aggelopoulos, C.; Skouras, E.; Tsakiroglou, C.; Burganos, V., Modeling of a DBD plasma reactor for porous soil remediation. *Chemical Engineering Journal* **2019**, 373, 393-405.

Thermal transport problem



- Thermal transport equations: (pseudo-steady state)

$$\rho_i C_{p,i} \frac{\partial T}{\partial t} + \nabla k_i \nabla T = 0 \quad (\partial T/\partial t \to 0)$$

- Conductivities for each phase:

$$k_{eff} = -\frac{\iint_A N_{therm} dA}{A_{tot}} \frac{\Delta L}{T_{out} - T_{in}}$$

Time t (s)	Saturation S	Conductivity k_{eff} (W/m/K)	Conductivity k_{breff} (W/m/K)
0	0.33	**0.155**	**0.172**
180	0	**0.128**	**0.146**

Bali, N.; Aggelopoulos, C.; Skouras, E.; Tsakiroglou, C.; Burganos, V., Hierarchical modeling of plasma and transport phenomena in a dielectric barrier discharge reactor. *Journal of Physics D: Applied Physics* **2017**, 50, 505202.

Effective properties

- Permeability value
- Diffusivity
- Thermal conductivity
- Mobility
- ..etc

Maxwell

$$\frac{k_{eff} - k_f}{k_{eff} + 2k_f} = f_p \frac{k_p - k_f}{k_p + 2k_f}.$$

Bruggeman

$$f_p \frac{k_p - k_{eff}}{k_p + 2k_{eff}} + \left(1 - f_p\right) \frac{k_f - k_{eff}}{k_f + 2k_{eff}} = 0.$$

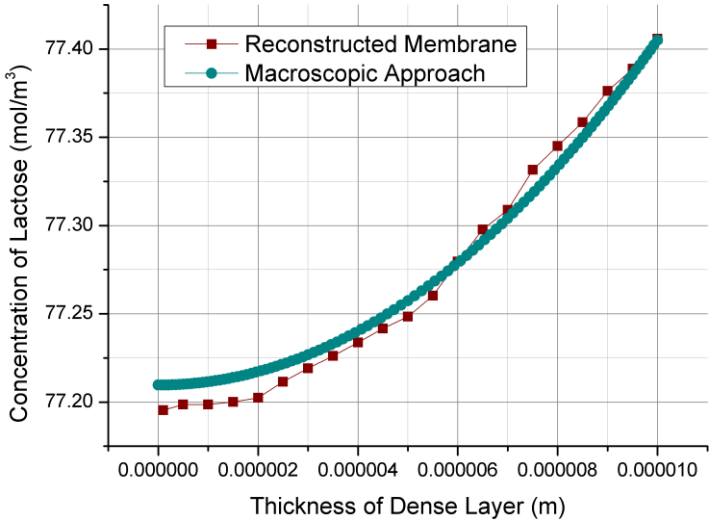
Bruggeman DEM

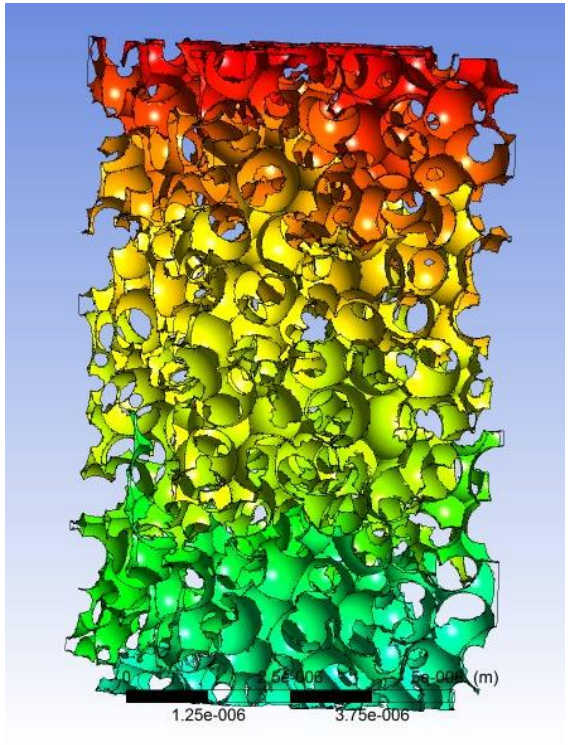
$$\frac{k_p - k_{eff}}{k_p - k_f} \left(\frac{k_f}{k_{eff}}\right)^{1/3} = 1 - f_p.$$

- Advantages of calculating these values numerically:
  - Avoid artifacts and other errors that introduced through the approximation of the structure (its numerically generated)
  - Less computational cost (the problem have less mesh elements) difficulties in structures
  - More robust solutions

Numerical reconstructions and “black box”

- Solving in numerical reconstructed domain
- Solving in a “black box”





N. Bali, A. Petsi, E. Skouras, V. Burganos, Three-dimensional reconstruction of bioactive membranes and pore-scale simulation of enzymatic reactions: The case of lactose hydrolysis, Journal of membrane science 524 (2017) 225-234.

Numerical reconstructions and EMT

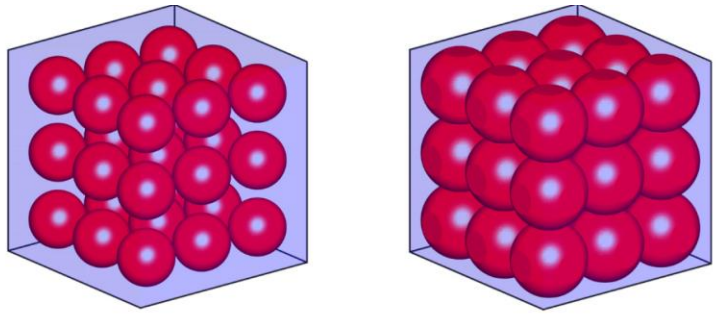


Figure 8. Single arrangement of spherical particles. (**a**) $f_p = 0.3$, (**b**) $f_p = 0.7$.

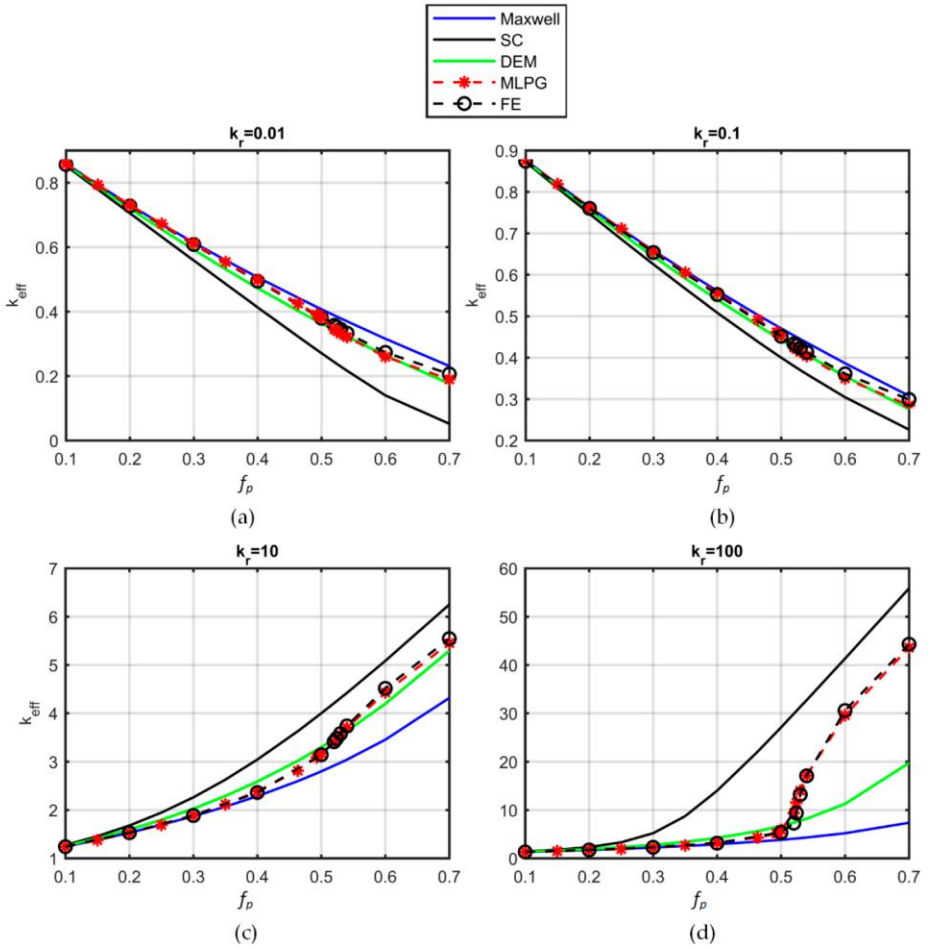
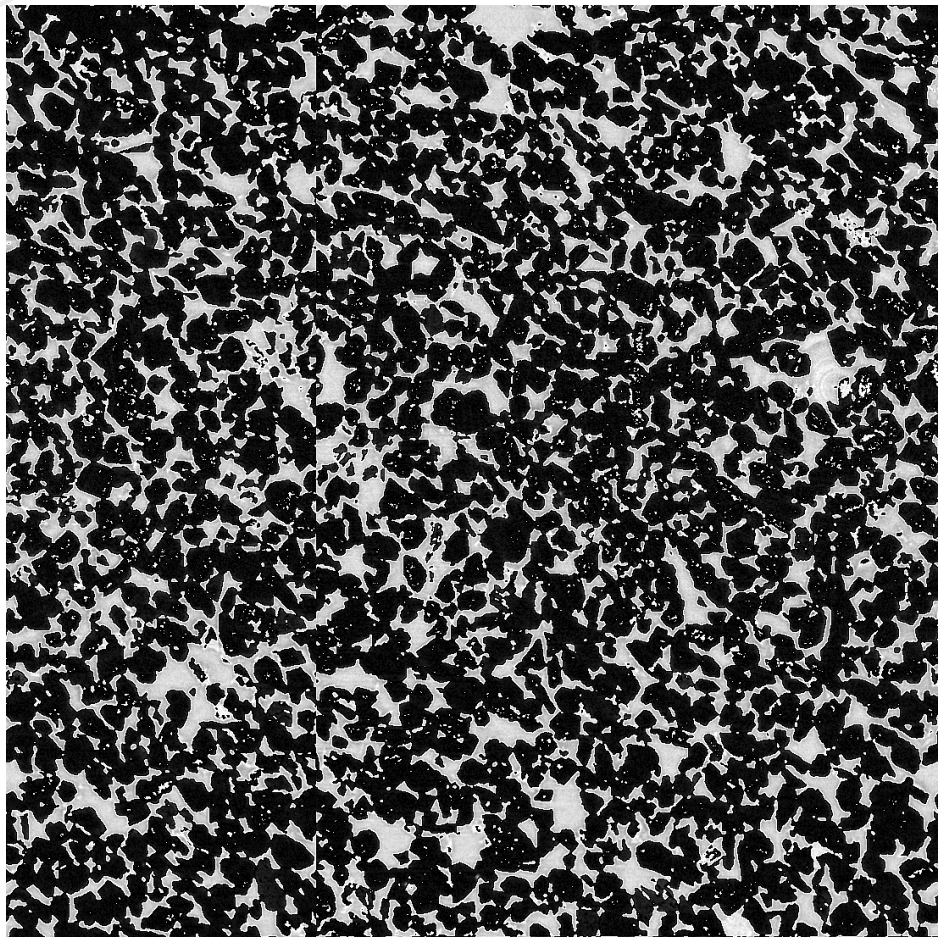


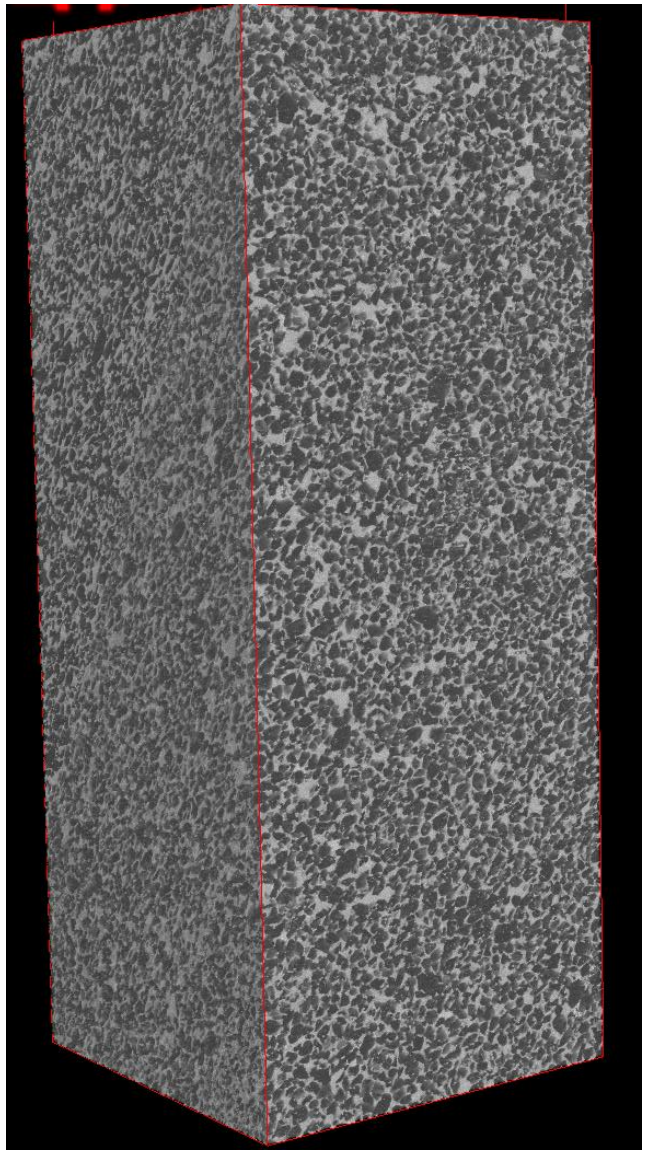
Figure 9. Effective thermal conductivity of the single arrangement as a function of the volume fraction of the inclusions for four different conductivity ratios (k_r), (**a**) 1/100, (**b**) 1/10, (**c**) 10, (**d**) 100. Continuous lines: analytical expressions. Blue lines: Maxwell. Black lines: SC. Green lines: DEM. Dashed lines: simulations. Red: MLPG. Black: FE. Abbreviation; SC: self-consistent approximation, DEM: differential effective medium approximation MLPG: meshless local Petrov-Galerkin, FE: finite element.

P. Karagiannakis, N.; Bali, N.; D. Skouras, E.; N. Burganos, V., An efficient meshless numerical method for heat conduction studies in particle aggregates. *Applied Sciences* **2020**, 10, 739.

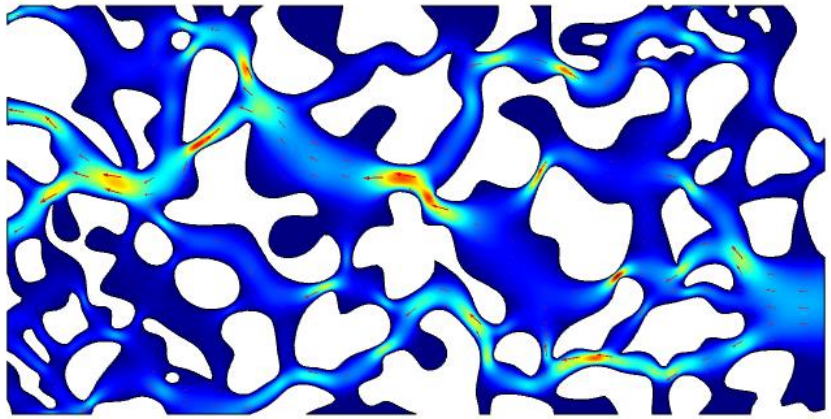
New era: Transport phenomena in real structures

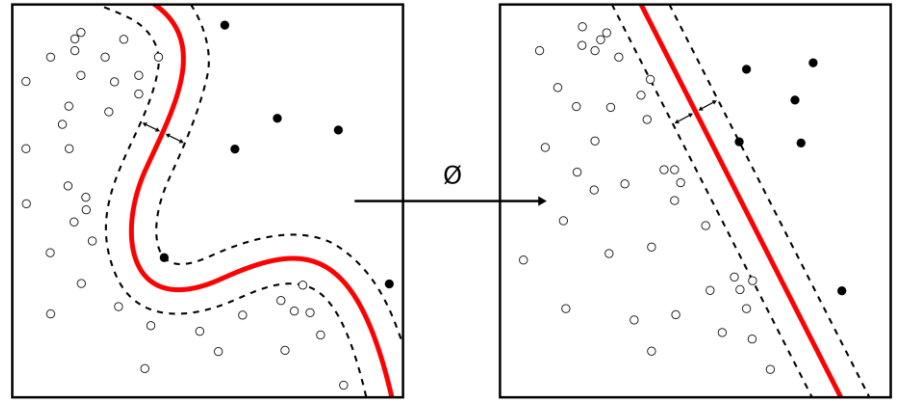


The results of ~3000 scanned images



Transport phenomena in real structures: future goals



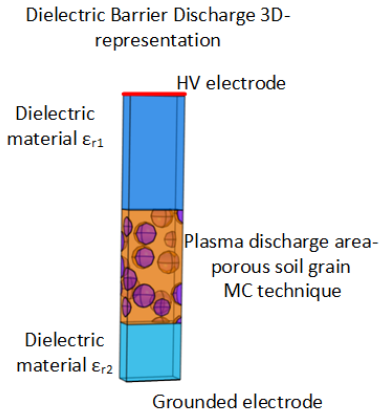


Solving in 2D vertical images to CT scanned

Convolutional Neural Networks

The research project is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment (Project title: Enhanced Oil Recovery by Polymer-coated Nano Particles, acronym: «EOR-PNP», code: HFRI-FM17-361)

THANK YOU



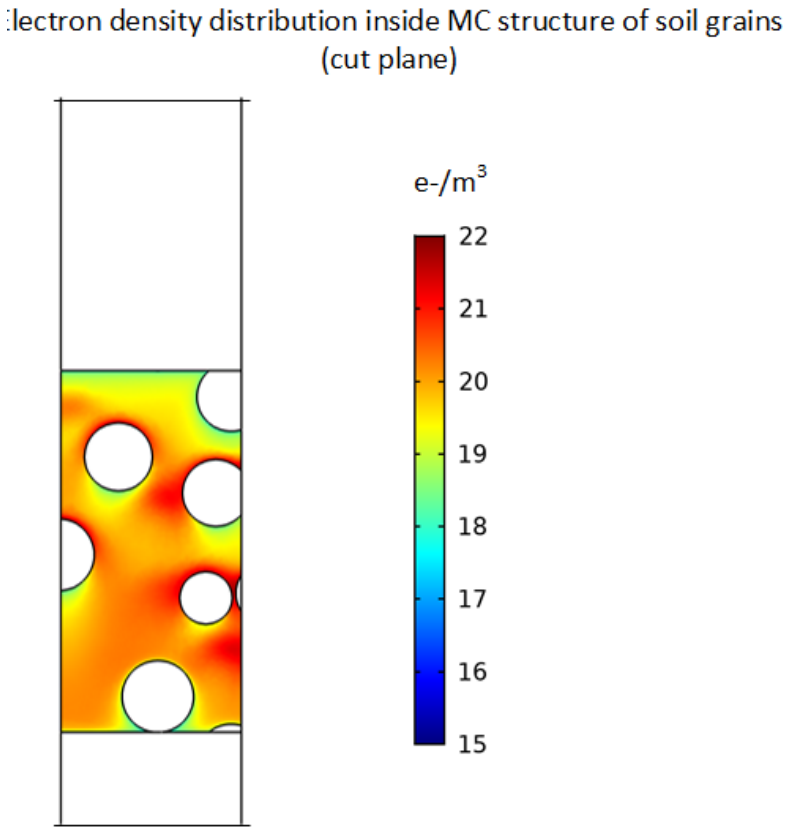


Figure 7. Plasma distribution inside Monte Carlo (MC) randomly placed spheres domain

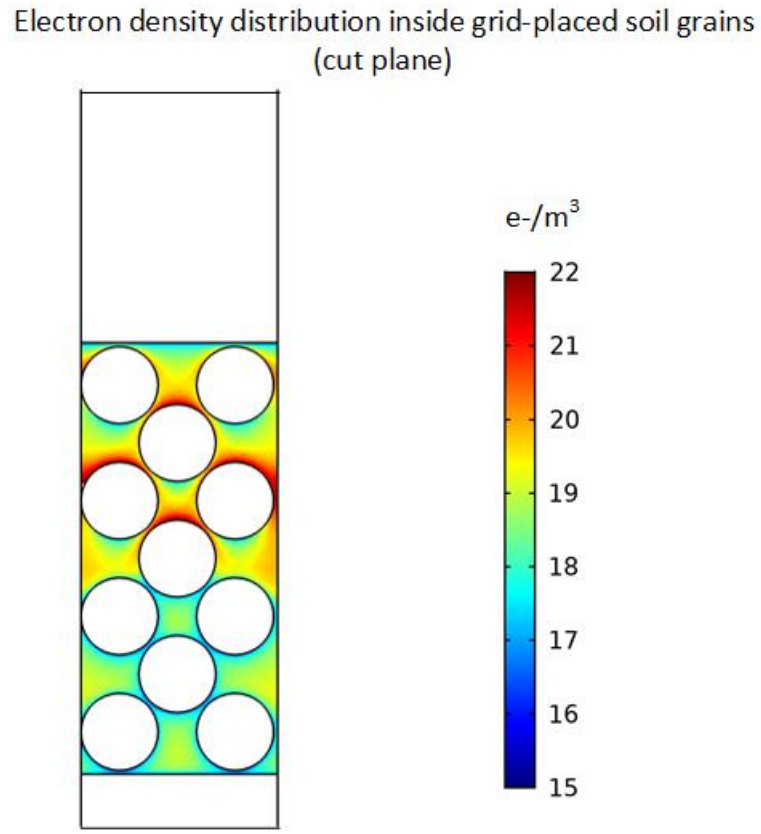
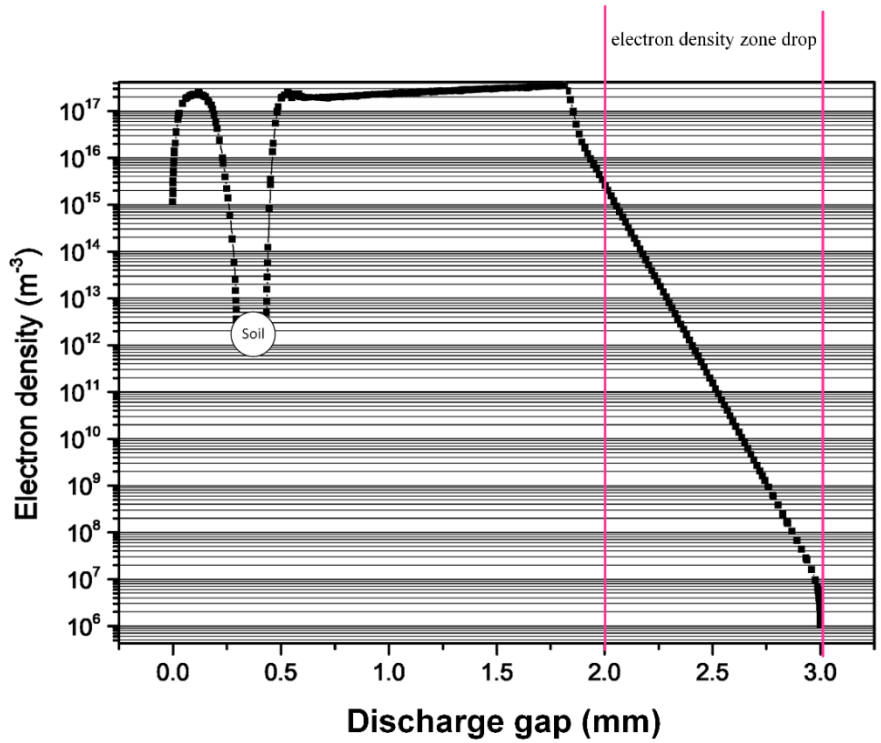
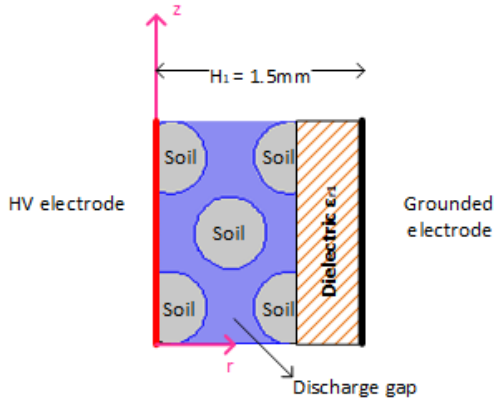
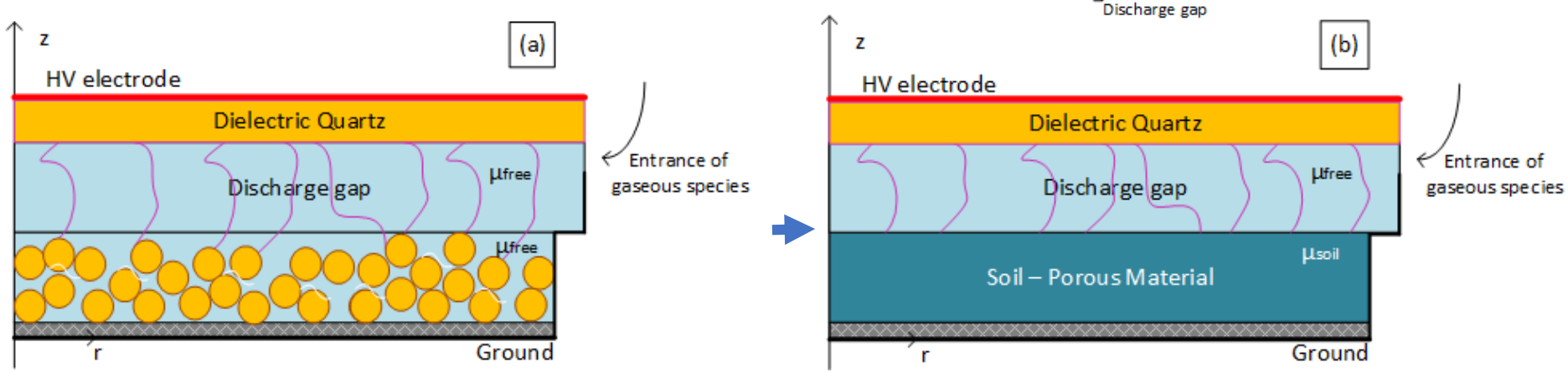


Figure 6. Plasma distribution inside cubic spheres arrangement (FCC)


electron density zone drop
Soil
Electron density (m-3)
10^17
10^16
10^15
10^14
10^13
10^12
10^11
10^10
10^9
10^8
10^7
10^6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Discharge gap (mm)

Μοντελοποίησ
πλάσματος στ
εκκένωσης + χ





- Τροποποιημένες εξισώσεις:
  - ως προς τα μονοπάτια των αντιδράσεων (θερμοκρασία 300K – ψυχρό πλάσμα) 100 αντιδράσεις
  - μ (συντελεστής ευκινησίας ηλεκτρικά φορτισμένων σωματιδίων) η ισοδύναμη τιμή μέσα στο χώμα δίνεται από:

$$m_{soil} = \frac{e_p}{t} m_{free}$$

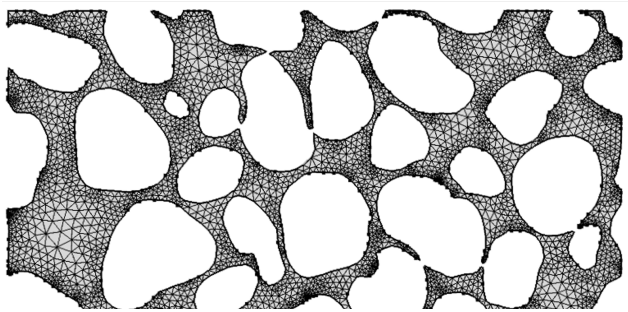
  - Αντίστοιχα αλλάζουν και οι συντελεστές διάχυσης των συστατικών του πλάσματος στο χώμα

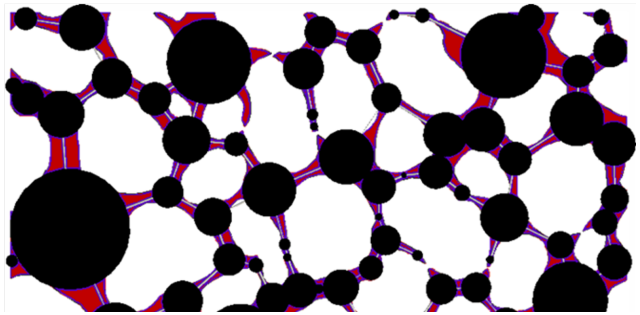
Thanks to Dr. Christos Tsakiroglou and the FORTH Institute, and the EORPNP consortium

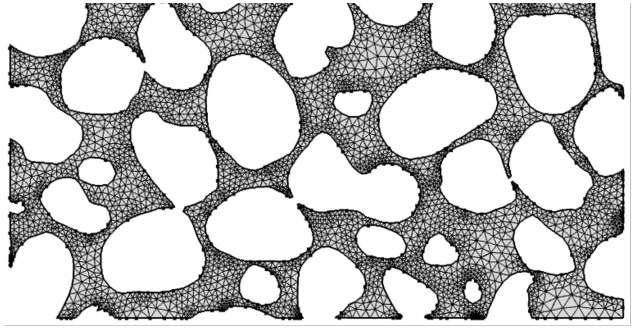
Introduction to Pore Network Modeling

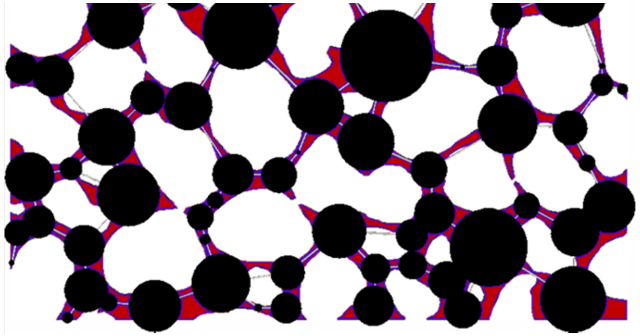
Pore network modelling in one sentence:

> PNM is based on the recognition that the interconnected voids of a real porous material can be mapped onto an equivalent net
> convenient mathematical analysis







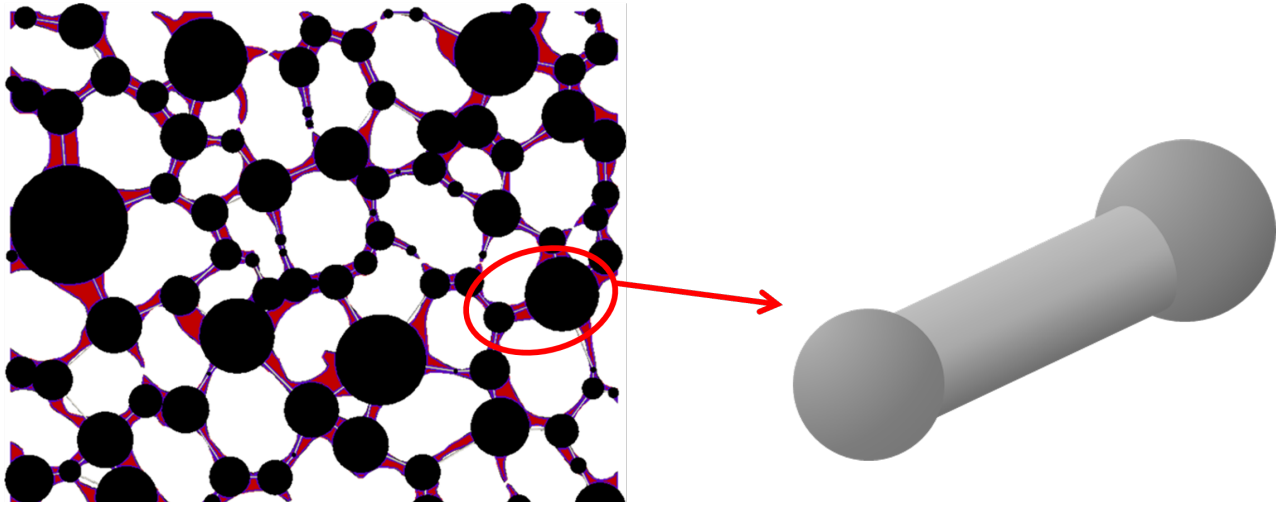


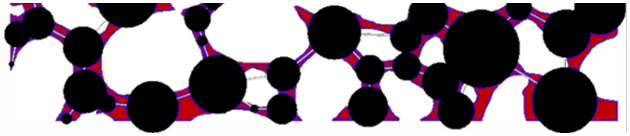
Pore Network Modelling, or PNM, offers several advantages or capabilities:

- It allows "pore-scale" modeling of transport processes in porous materials with computational ease
  - This permits the study of the "structure-performance" relationship in materials like electrodes, filters, catalysts, ...
- It enables multiphase flow predictions through percolation theory
  - This permits the study of multiphase transport processes like relatively permeability and resisitivity index

Some Definitions and Terminology

> - A *pore* is a cavity or opening in the voids
> - A *throat* is a constriction between two cavties
> - A throat plus *half* of each pore on either end define a *conduit*



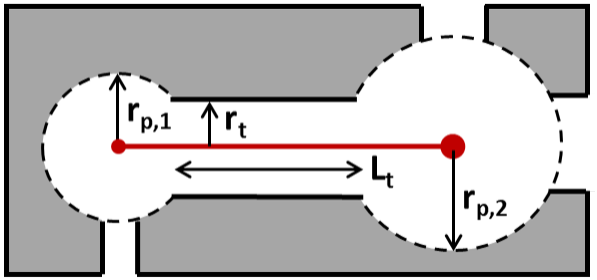


Accurate pore network predictions require that the pore space is correctly mapped, this means two things:

1. The *topology* of the network is mapped correctly
    - Each pore is connected to the correct neighboroing pores
    - The spatial locations of each pore is known
2. The *geometry* of each pore and throat are determined correctly
    - Image analysis tools (e.g. PoreSpy) are used to measure the size of each pore cavity and throat constriction

Using a Simplified Geometry

We often represent pores by spheres and throats by cylinders, as shown below, but this is completely flexible:

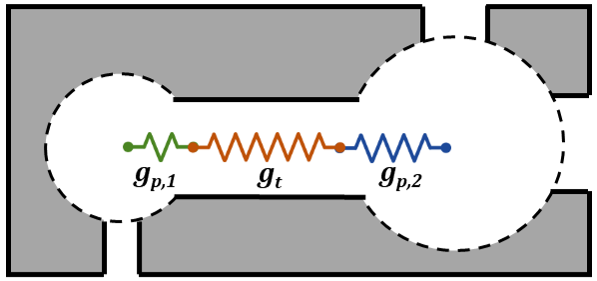


Whatever physical law and geometrical simplificaiton are used, they must be representable as a form of Ohm's law:

$$n_{i,j} = D_{AB}\frac{\pi R_t^2}{L_t}\Delta C_{i,j} = g_D \Delta C_{i,j}$$

Converting to an Equivalent Conductance

We can be more or less detailed about how we find g_D, but ultimately the pore-throat-pore conduit is a set of resistors (or conductors)



$$\frac{1}{g_D} = \frac{1}{g_{p,1}} + \frac{1}{g_t} + \frac{1}{g_{p,2}}$$

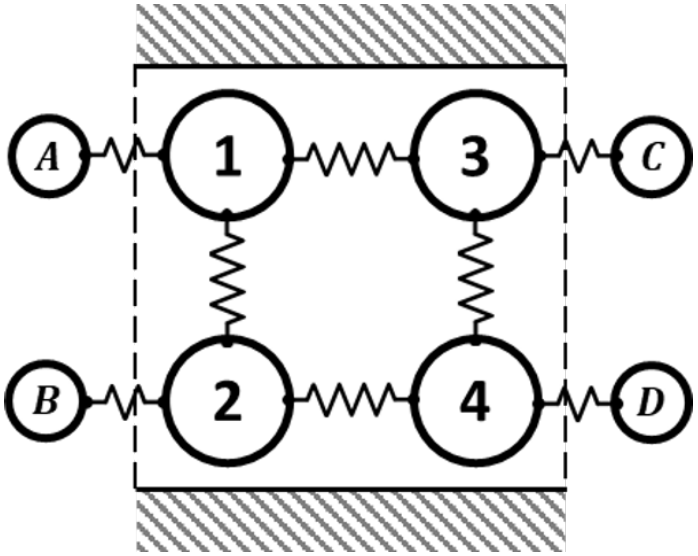
Resistor Network Calculations

Performing a mass balance on each pore gives the net rate of material entering or exiting that pore:

$$N_j = \Sigma g_{D,i-j} \Delta C_{i,j} = b$$

where b would be 0 for a non-reactive, internal pore, or $b = const$ for a constant rate boundary condition, etc.

The pore network is a fully connected network where the mass balances between pores are all coupled to their neighbors:

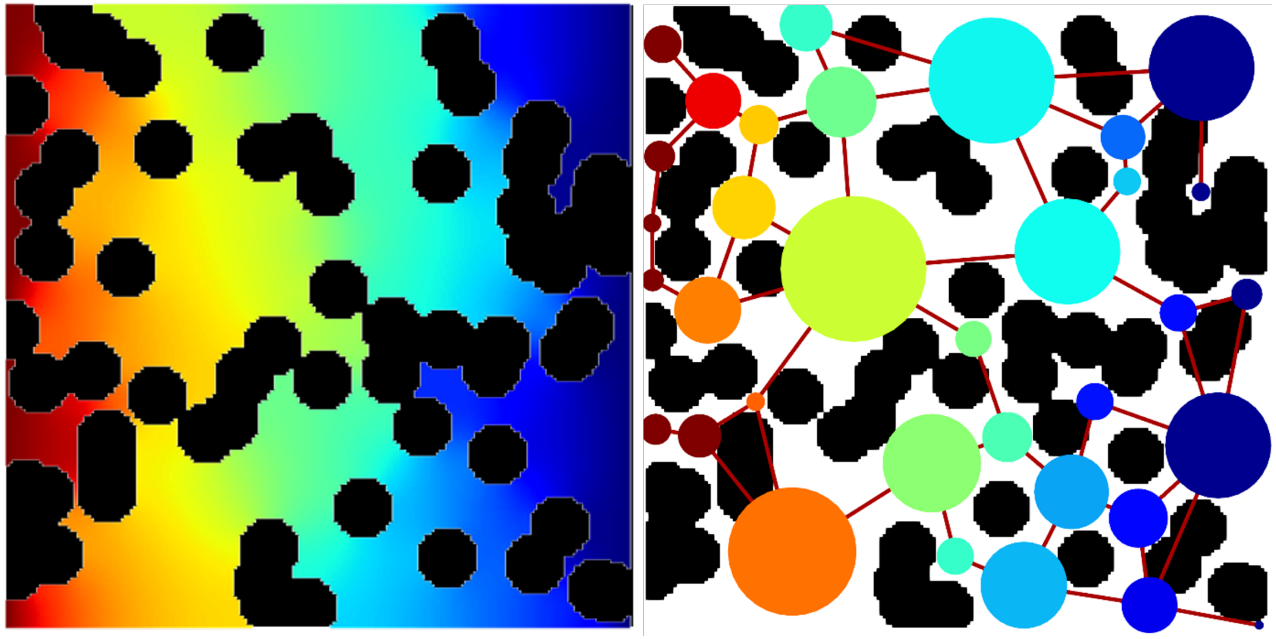


This gives a system of equations that can be solved to find the quantity of interest (i.e. concentration, pressure, etc):

$$C_i = A^{-1} b$$

Calculating Species Distributions

The result is that we can solve for the quantity in each pore, which we can compare to full DNS calculation:



Overview of OpenPNM

OpenPNM is a framework for conducting *Pore Network Modelling* studies (i.e. **PNM**).

The name **Open** refers to the fact that the code is 100% open-source and available on Github (https://github.com/PMEAL/OpenPNM (https://github.com/PMEAL/OpenPNM))

It includes functionality like generating networks, computing thermodynamic properties of phases, and solving the numerical problems.

In this workshop we'll cover each aspect of the package to provide an overview of OpenPNM's capabilities and how to access them.

OpenPNM is Oganized in Several Modules

The main modules are described below:

- `network` : Contains the `Network` class and numerous generators. The network objects are dictionaries that contain the topologic properties of the network.
- `phase` : Contains the `Phase` class, as well as several other classes for mixtures. The phase objects are also dictionaries that cor and transport properties of the fluid.
- `algorithms` : Contains all the classes used for performing simulations, including flow, diffusion, dispersion, percolation, drainage,
- `models` : A library containing 'pore-scale' models which are used to compute the properties of each pore and throat.

It also contains several helper modules such as:

- `io` : Functions for reading and writing data between formats
- `visualization` : Small set of functions for basic visualization
- `topotools` : Functions for editing and querying the network topology
- `solvers` and `integrators` : Classes for steady-state and transient solving of the equations
- `core` : Mostly used internally
- `utils` : Mostly used internally

OpenPNM is Object-Oriented

- An OpenPNM simulation is quite a bit more complicated than some image analysis with PoreSpy for instance.
- There is a lot more data to deal with and a lot more things that can be done with the data.
- Because of this added complexity, it is necessary (and helpful!) to use `objects`.
- The main object used in OpenPNM is the Python `dict`

Refresher on the `dict`

The `dict` is a highly versatile data container that allows access to the data by name:

In [11]:
```
d = dict()
d['bob'] = 1.0
d['fred'] = [1, 2, 3]
print(d)
```

```
{'bob': 1.0, 'fred': [1, 2, 3]}
```

In OpenPNM, the data being stored are the pore and throat properties, like diameter and volume:

In [12]:
```
net = dict()
net['pore.diameter'] = [2, 1.5, 1]
net['throat.length'] = [0.75, 1.5]
print(net)
```

```
{'pore.diameter': [2, 1.5, 1], 'throat.length': [0.75, 1.5]}
```

We'll explore this data storage scheme in more depth later.

> The key to note here is that there are 3 pores and 2 throats.

One *massive* benefit of using a `dict` is that we can *subclass* it. This allows us to add our own custom functionality to the `dict`.

This is done as follows:

In [13]:
```
class Network(dict):

    def num_pores(self):
        return len(self['pore.diameter'])

    def num_throats(self):
        return len(self['throat.length'])
```

And we can use it like this:

In [14]:
```
net = Network()
net['pore.diameter'] = [2, 1.5, 1]
net['throat.length'] = [0.75, 1.5]
print(f'The network has {net.num_pores()} pores')
print(f'The network has {net.num_throats()} throats')
```

```
The network has 3 pores
The network has 2 throats
```

Hopefully the class definition above gives you as sense of the power of object-oriented programming.

The main point is:

> The `net` object behaves as a generic `dict` PLUS we have added some functions that are *very specific* to our use case. The designed to work on the data within the `dict`.

And the motivation for tell you this:

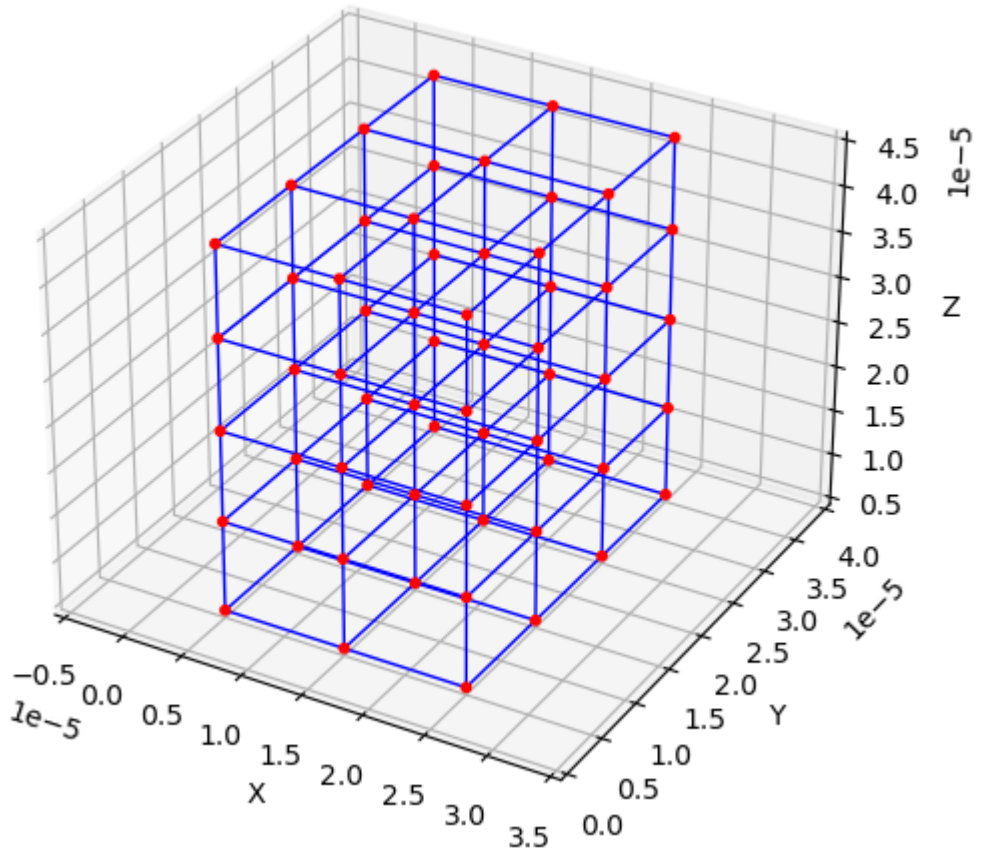
> OpenPNM uses this technique quite extensively

A First Look at Using OpenPNM

Let's look quickly at using OpenPNM. Don't try to code this, we'll revisit it in the next tutorial:

In [15]:
```
import openpnm as op
```

In [16]:
```
pn = op.network.Cubic([3, 4, 5], spacing=1e-5)
ax = op.visualization.plot_connections(pn)
ax = op.visualization.plot_coordinates(pn, ax=ax);
```



In [17]:
```
print(pn)
```

```
================================================================================
net : <openpnm.network.Cubic at 0x1ab78dfcd10>
--------------------------------------------------------------------------------
  #  Properties                                                    Valid Values
--------------------------------------------------------------------------------
  2  pore.coords                                                     60 / 60
  3  throat.conns                                                   133 / 133
--------------------------------------------------------------------------------
  #  Labels                                                      Assigned Locations
--------------------------------------------------------------------------------
  2  pore.surface                                                            54
  3  throat.surface                                                         104
  4  pore.left                                                               20
  5  pore.right                                                              20
  6  pore.front                                                              15
  7  pore.back                                                               15
  8  pore.bottom                                                             12
  9  pore.top                                                                12
--------------------------------------------------------------------------------
```

Adding Geometrical Properties

Note that this network does not yet have any geometrical properties. We have to add these explicitly, which is how we declare our pref
size distribution, and so forth.

We'll use a predefined collection of models, which gives everything we need (and more!):

In [18]:
```
pn.add_model_collection(op.models.collections.geometry.spheres_and_cylinders)
print(pn)
```

```
================================================================================
net : <openpnm.network.Cubic at 0x1ab78dfcd10>
--------------------------------------------------------------------------------
  #  Properties                                                    Valid Values
--------------------------------------------------------------------------------
  2  pore.coords                                                     60 / 60
  3  throat.conns                                                   133 / 133
  4  pore.coordination_number                                        60 / 60
  5  pore.max_size                                                   60 / 60
  6  throat.spacing                                                 133 / 133
  7  pore.seed                                                       60 / 60
  8  pore.diameter                                                   60 / 60
  9  throat.max_size                                                133 / 133
 10  throat.diameter                                                133 / 133
 11  throat.cross_sectional_area                                    133 / 133
 12  throat.hydraulic_size_factors                                  133 / 133
 13  throat.diffusive_size_factors                                  133 / 133
 14  throat.lens_volume                                             133 / 133
 15  throat.length                                                  133 / 133
 16  throat.total_volume                                            133 / 133
 17  throat.volume                                                  133 / 133
 18  pore.volume                                                     60 / 60
--------------------------------------------------------------------------------
  #  Labels                                                  Assigned Locations
--------------------------------------------------------------------------------
  2  pore.surface                                                            54
  3  throat.surface                                                         104
  4  pore.left                                                               20
  5  pore.right                                                              20
  6  pore.front                                                              15
  7  pore.back                                                               15
  8  pore.bottom                                                             12
  9  pore.top                                                                12
--------------------------------------------------------------------------------
```

Calculating Thermodynamic Properties

And let's create an `Air` object then do a diffusion simulation:

In [19]:
```
air = op.phase.Air(network=pn)
print(air)
```

```
================================================================================
phase_01 : <openpnm.phase.Air at 0x1ab79683220>
--------------------------------------------------------------------------------
  #  Properties                                                    Valid Values
--------------------------------------------------------------------------------
  2  pore.temperature                                                 60 / 60
  3  pore.pressure                                                    60 / 60
  4  pore.density                                                     60 / 60
  5  pore.molar_density                                               60 / 60
  6  pore.diffusivity                                                 60 / 60
  7  pore.thermal_conductivity                                        60 / 60
  8  pore.viscosity                                                   60 / 60
--------------------------------------------------------------------------------
  #  Labels                                                  Assigned Locations
--------------------------------------------------------------------------------
  2  pore.all                                                                60
  3  throat.all                                                             133
--------------------------------------------------------------------------------
```

Computing Transport Properites

The diffusive conductance of a conduit, the pore-throat-pore connection, is a function of the geometric sizes and thermodynamic prope

$$N = D_{AB}\frac{A}{L}\Delta C = g_D \Delta C$$

In [20]:
```
air.add_model_collection(op.models.collections.physics.basic)
air.regenerate_models()
print(air)
```

```
[15:07:51] WARNING  throat.entry_pressure was not run since the following property is missing:       _mo
                    'throat.surface_tension'
```

```
═══════════════════════════════════════════════════════════════════════════
phase_01 : <openpnm.phase.Air at 0x1ab79683220>
───────────────────────────────────────────────────────────────────────────
  #  Properties                                                Valid Values
───────────────────────────────────────────────────────────────────────────
  2  pore.temperature                                             60 / 60
  3  pore.pressure                                                60 / 60
  4  pore.density                                                 60 / 60
  5  pore.molar_density                                           60 / 60
  6  pore.diffusivity                                             60 / 60
  7  pore.thermal_conductivity                                    60 / 60
  8  pore.viscosity                                               60 / 60
  9  throat.hydraulic_conductance                               133 / 133
 10  throat.diffusive_conductance                               133 / 133
───────────────────────────────────────────────────────────────────────────
  #  Labels                                                Assigned Locations
───────────────────────────────────────────────────────────────────────────
  2  pore.all                                                            60
  3  throat.all                                                         133
───────────────────────────────────────────────────────────────────────────
```

With the pore sizes, phase properties, and transport models defined, we can do the simulation:

In [21]:
```
fd = op.algorithms.FickianDiffusion(network=pn, phase=air)
fd.set_value_BC(pores=pn.pores('top'), values=1.0)
fd.set_value_BC(pores=pn.pores('bottom'), values=0.0)
fd.run()
```

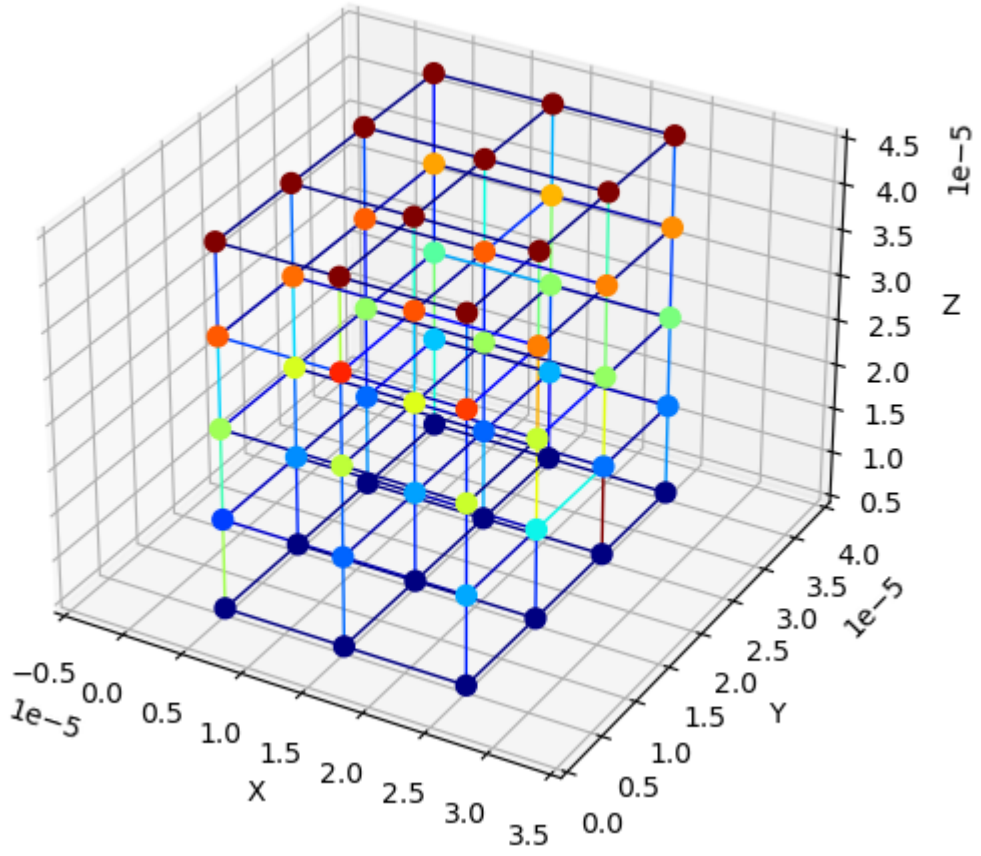
The algorithm writes the results of the simulation into it's own dictionary:

In [22]:
```
print(fd)
```

```
================================================================================
fick_02 : <openpnm.algorithms.FickianDiffusion at 0x1ab0176dd10>
--------------------------------------------------------------------------------
  #  Properties                                                    Valid Values
--------------------------------------------------------------------------------
  2  pore.bc.rate                                                     0 / 60
  3  pore.bc.value                                                   24 / 60
  4  pore.concentration                                              60 / 60
  5  pore.initial_guess                                              60 / 60
--------------------------------------------------------------------------------
  #  Labels                                                  Assigned Locations
--------------------------------------------------------------------------------
  2  pore.all                                                                60
  3  throat.all                                                             133
--------------------------------------------------------------------------------
```

We can re-visualize the network, but chose to color the nodes according to their concentration, and the throats according to the mass f

In [25]:
```
ax = op.visualization.plot_connections(pn, color_by=fd.rate(throats=pn.Ts, mode='single'))
ax = op.visualization.plot_coordinates(pn, markersize=50, color_by=fd.x, ax=ax)
```



Compute Network Permeability

There are three main stages of an OpenPNM simulation.

1. Create a network topology and define geometric properties
2. Create a phase object and define pore-scale transport properties
3. Create an algorithm object, assign boundary conditions, and run it

Let's proceed as though we're going to do a single phase flow simulation to compute the permeability of the network.

One common conductance model for flow through a tube is the Hagan-Poiseuille model:

$$Q = \frac{\pi R^4}{8\mu L}\Delta P = g_H \Delta P$$

From this we can see that our simulation will need the diameters and lengths of pores and throats, as well as the fluid viscosity.

Network Generators

OpenPNM includes a number of ways to generate networks (https://openpnm.org/modules/generated/openpnm.network.html#module- regular lattices and random networks:

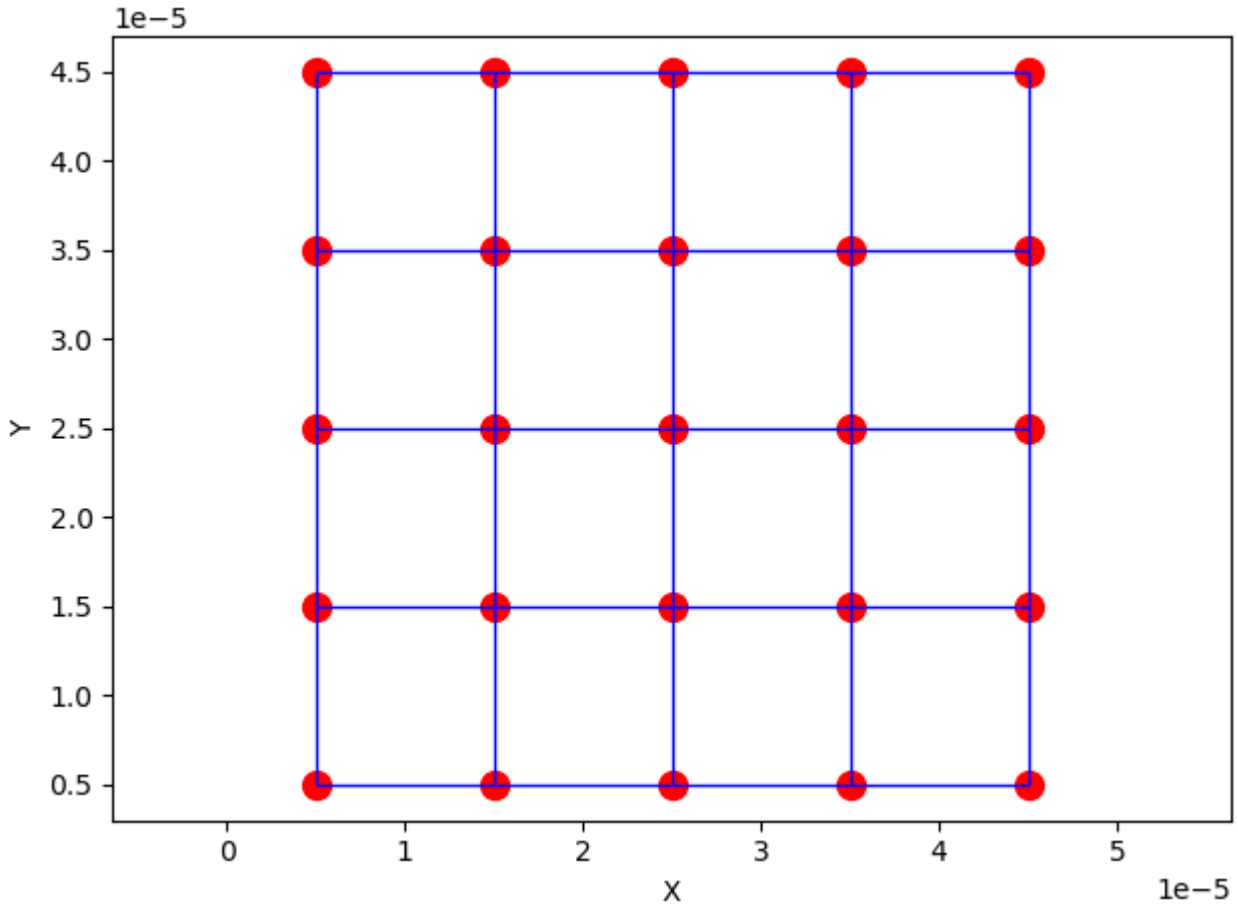
In [56]:

```
import openpnm as op
for item in dir(op.network):
    if not item.startswith('_'):
        print(item)
```

```
BodyCenteredCubic
Cubic
CubicTemplate
Delaunay
DelaunayVoronoiDual
Demo
FaceCenteredCubic
Network
Voronoi
```

Let's use the Cubic generator, and stick to 2D, for simplicity:

In [57]:
```
pn = op.network.Cubic(shape=[5, 5, 1], spacing=1e-5)
ax = op.visualization.plot_connections(pn)
ax = op.visualization.plot_coordinates(pn, s=100, ax=ax)
```



Data Storage

Let's print the network and see what data is created during the generation:

In [28]:
```
print(pn)
```

```
═══════════════════════════════════════════════════════════════════════════
net : <openpnm.network.Cubic at 0x2761b786ea0>
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
  #  Properties                                                Valid Values
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
  2  pore.coords                                                 25 / 25
  3  throat.conns                                                40 / 40
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
  #  Labels                                                  Assigned Locations
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
  2  pore.surface                                                        16
  3  throat.surface                                                      16
  4  pore.left                                                            5
  5  pore.right                                                           5
  6  pore.front                                                           5
  7  pore.back                                                            5
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
```

Key things to note:

1. Each item listed above is an entry in the `pn` dictionary, like `pn['pore.coords']`
2. Items starting with the word `'pore'` are obviously "pore data" and `'throat'` indicates "throat data"
3. Each item is a numpy array with one element for each pore or throat...this network has 25 pores and 40 throats
4. 'pore.coords' lists the [x, y, z] coordinates of each pore
5. 'throat.conns' lists the [head, tail] of each throat, meaning which pore is on each end
6. The things referred to as "Labels" are boolean arrays (i.e. `True/False`) where `True` indicates where that label is applied. (Thes applying boundary conditions).
7. Geometrical data is missing, but we'll add that next

Define Geometry

To conduct our simulation we need:

- The diameter of pores and throats

- The length of throats

Pore Diameters: The easy way

Let's assign pore diameters from a unform distribution, remembering that the pores are spaced 10 um apart and they should not overla

In [86]:
```
import numpy as np
pn['pore.diameter'] = np.random.rand(pn.num_pores())*1e-5
print(pn['pore.diameter'])
```

```
[7.62352084e-06 4.18667497e-07 2.78986473e-06 8.57960414e-06
 3.69039996e-06 8.98991875e-06 2.89097831e-06 3.80323481e-06
 1.09373393e-06 8.14624611e-06 9.33485442e-06 1.83107189e-06
 3.11198567e-06 6.74704043e-06 7.91010422e-07 3.27239943e-07
 3.26575619e-06 3.86612194e-06 5.48967345e-06 5.52931172e-06
 3.30127174e-06 4.25651101e-06 2.51856859e-06 2.02635132e-06
 7.77525513e-06]
```

Throat Lengths: The hard way

First, let's do this by hand:

> We'll use the `'throat.conns'` array to index into the `'pore.diameter'` array to get the diameter of the pore on each end then we'll subtract half of each diameter from the known pore spacing, with the remaining amount being the throat length

In [59]:
```
D1, D2 = pn['pore.diameter'][pn['throat.conns']].T
Lt = 1e-5 - D1/2 - D2/2
print(Lt)
```

```
[3.38614765e-06 2.66129569e-06 2.68908581e-06 6.21639964e-06
 6.77185282e-06 6.01354562e-06 5.27774603e-06 4.47143498e-06
 5.42389468e-06 7.11133557e-06 5.28112350e-06 1.37637822e-06
 3.18933525e-06 3.12918888e-06 2.74284317e-06 2.70295620e-06
 5.95339764e-06 2.11642329e-06 5.28608824e-06 8.91380333e-06
 3.34892736e-06 6.80907310e-06 1.86576821e-06 6.10106364e-06
 4.58677099e-06 4.67340213e-06 7.52234536e-06 5.60253583e-06
 4.95633370e-06 8.91479500e-07 4.15921814e-06 4.45401178e-06
 5.78651267e-06 2.23745401e-06 1.84188042e-06 5.81224492e-06
 3.33048797e-06 1.91512421e-06 6.11380720e-06 5.50295233e-06]
```

Throat Lengths: The "pore-scale model" way

Calculations like the one above are common, so OpenPNM offers pre-defined ways to do them.

`openpnm.models` is a libary of prewritting functions for computing many things.

The library is further organized by the type of thing being computed, such as `openpnm.models.geometry` and `openpnm.models.ph`

In [75]:
```
mod = op.models.geometry.throat_length.squares_and_rectangles
pn.add_model(propname='throat.length', model=mod)
pn.regenerate_models()
print(pn['throat.length'])
```

```
[3.38614765e-06 2.66129569e-06 2.68908581e-06 6.21639964e-06
 6.77185282e-06 6.01354562e-06 5.27774603e-06 4.47143498e-06
 5.42389468e-06 7.11133557e-06 5.28112350e-06 1.37637822e-06
 3.18933525e-06 3.12918888e-06 2.74284317e-06 2.70295620e-06
 5.95339764e-06 2.11642329e-06 5.28608824e-06 8.91380333e-06
 3.34892736e-06 6.80907310e-06 1.86576821e-06 6.10106364e-06
 4.58677099e-06 4.67340213e-06 7.52234536e-06 5.60253583e-06
 4.95633370e-06 8.91479500e-07 4.15921814e-06 4.45401178e-06
 5.78651267e-06 2.23745401e-06 1.84188042e-06 5.81224492e-06
 3.33048797e-06 1.91512421e-06 6.11380720e-06 5.50295233e-06]
```

Throat Diameters: The "pore-scale model" way

Calculating throat diameters by hand is also tedious because we'd like to ensure they are not larger than their neighboring pores.

Luckily there is a pore-scale model for that:

In [76]:
```
mod = op.models.geometry.throat_size.from_neighbor_pores
pn.add_model(propname='throat.diameter', model=mod)
pn.regenerate_models()
```

Compute Phase Properties

There are several ways to compute phase properties, but for the sake of this tutorial we'll just do it the quick and easy way.

In [79]:
```
water = op.phase.Phase(network=pn)
water['pore.viscosity'] = 0.001  # Pa.s
water['throat.viscosity'] = 0.001
```

Things to note:

- We assigned a scalar value (0.001), but internally OpenPNM applies it to all pores and throats so that each has its own value
- We have assigned values to both pores *and* throats, but OpenPNM would have automatically interpolate one if it only had the othe

Compute the Hydraulic Conductance

For simplicity, we will assume that all the pressue drop occurs in the throats, so we will not consider the contribution of the pores.

So:

$$\frac{1}{g_H} = \frac{1}{g_{p,1}} + \frac{1}{g_t} + \frac{1}{g_{p,2}} = \frac{1}{\infty} + \frac{1}{g_t} + \frac{1}{\infty} = \frac{1}{g_t}$$

In [81]:
```
R = pn['throat.diameter']/2
L = pn['throat.length']
mu = water['throat.viscosity']
water['throat.hydraulic_conductance'] = np.pi * R**4 / (8 * mu * L )
```

Run the Simulation

Now we create a `StokesFlow` object, assign boundary conditions and run it.

Note that the `StokesFlow` algorithm automatically looks for values of `'throat.hydraulic_conductance'` on the `Phase` object.

In [82]:
```
sf = op.algorithms.StokesFlow(network=pn, phase=water)
```

Applying Boundary Conditions Using Labels

The labels mentioned above will now be of great help. They allow us to choose pores on the desired faces for inlet and outlet boundari

We can ask the network to return the indices of the pores which have been labelled "left":

In [83]:
```
print(pn.pores('left'))
```

```
[0 1 2 3 4]
```

Now we use these to specify boundary conditions:

In [84]:
```
sf.set_value_BC(pores=pn.pores("left"), values=200_000)
sf.set_value_BC(pores=pn.pores("right"), values=100_000)
```

And finally we are ready to run it:

In [85]:
```
sf.run()
```

Determine Darcy Permeability of the Network

The simulation performed above is equivalent to the experiment that one would conduct on a real sample.

We just need to ask OpenPNM what was the total flow rate across the network, then we can find K from:

$$Q = \frac{KA}{\mu L}\Delta P$$

In [92]:
```
L = 5*1e-5
A = (5 * 1) * (1e-5)**2
mu = 0.001
DeltaP = 100_000
```

In [93]:
```
Q = sf.rate(pores=pn.pores('left'), mode='group')
```

In [95]:
```
K = Q * mu * L / (A * DeltaP)
print("The Darcy permeability is: ", K, "m^2")
```

```
The Darcy permeability is:  [1.35291722e-13] m^2
```

Simulating Capillary Pressure Curve on an Extracted Network

In this tutorial we will dive into the process of extracting a network from an image, then we'll compute the capillary pressure curve using and compare to the image-based drainage simulation in PoreSpy.

As usual, let's import the needed packages:

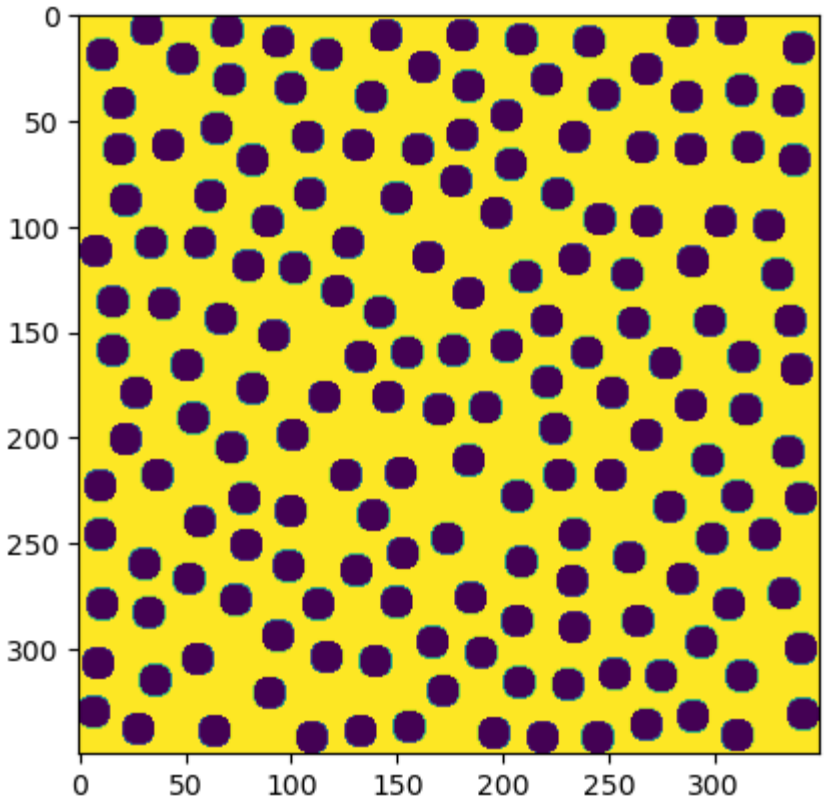
In [79]:
```
import porespy as ps
import openpnm as op
import matplotlib.pyplot as plt
import numpy as np
```

Generate an Image

Let's use a 2D generated image, for quicker processing and easier visualization:

In [99]:
```
im = ~ps.generators.rsa([350, 350], r=8, clearance=3)
plt.imshow(im);
```

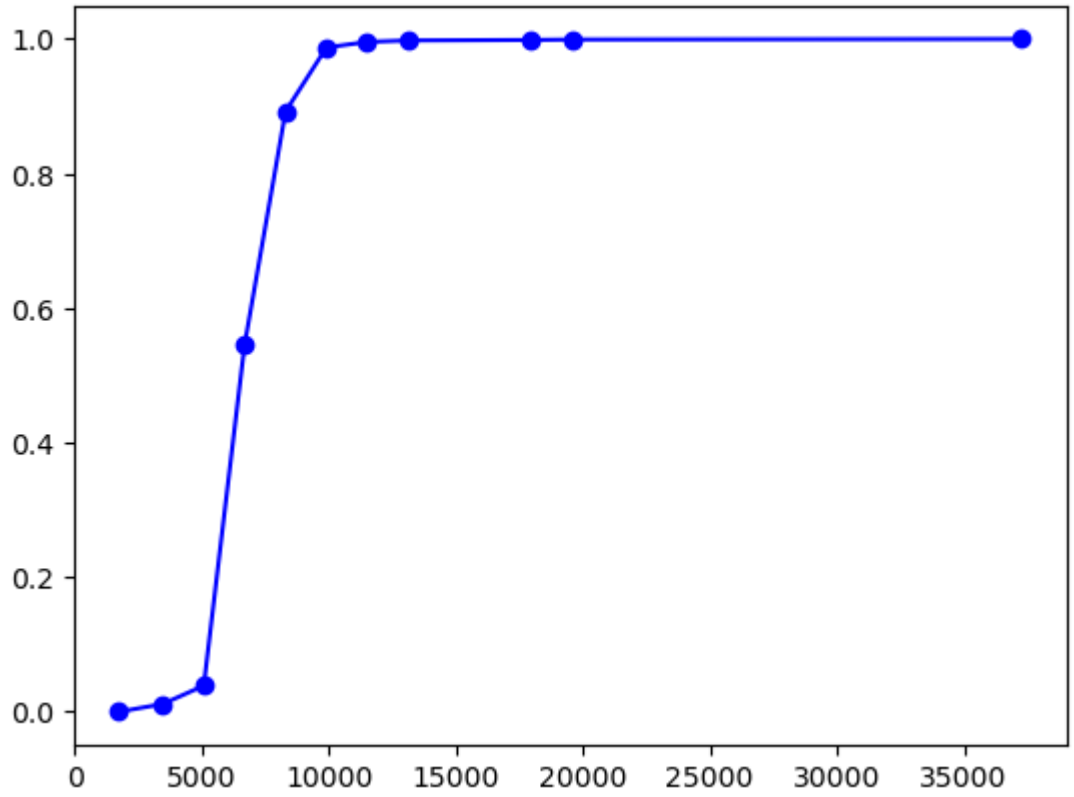
Out[99]:
```
<matplotlib.image.AxesImage at 0x289a89e1460>
```



Perform Drainage on Image

In [100]:
```
inlets = ps.tools.get_border(shape=im.shape, mode='edges')
drn = ps.simulations.drainage(im, inlets=inlets, sigma=0.480, theta=140, voxel_size=1e-5)
```

In [101]:
```
plt.plot(drn.pc, drn.snwp, 'bo-');
```

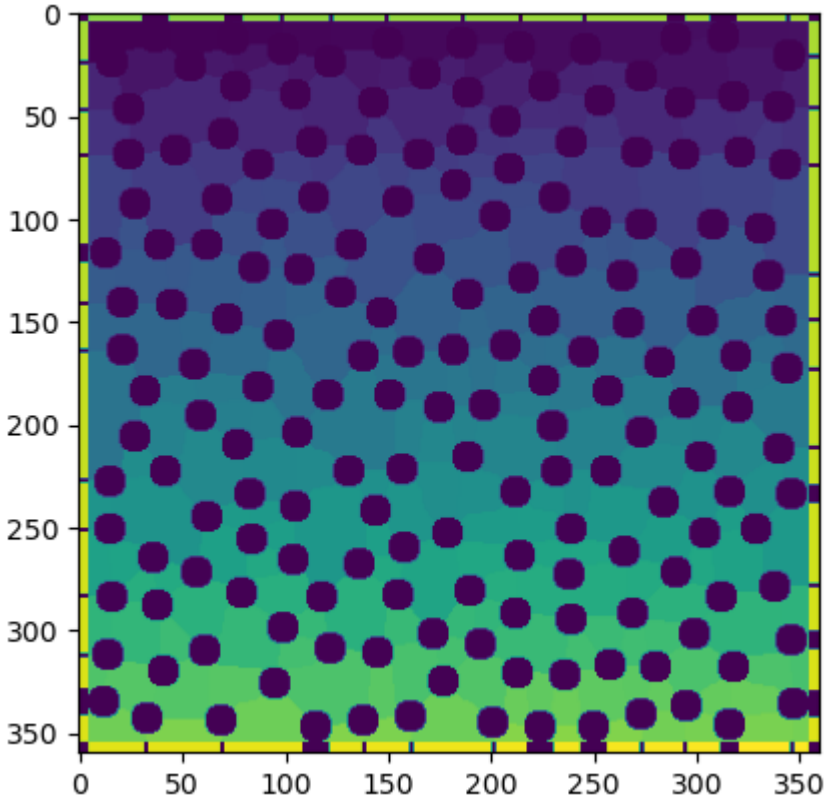


Extract the Network

We'll use the `snow2` function in PoreSpy:

In [102]:

```
snow = ps.networks.snow2(im, boundary_width=5, voxel_size=1e-5)
plt.imshow(snow.regions);
```



In [103]:

```
pn = op.io.network_from_porespy(snow.network)
print(pn)
```

```
================================================================================
net : <openpnm.network.Network at 0x289a9da38b0>
--------------------------------------------------------------------------------
  #  Properties                                                  Valid Values
--------------------------------------------------------------------------------
  2  throat.conns                                                  440 / 440
  3  pore.coords                                                   274 / 274
  4  pore.region_label                                             274 / 274
  5  pore.phase                                                    274 / 274
  6  throat.phases                                                 440 / 440
  7  pore.region_volume                                            274 / 274
  8  pore.equivalent_diameter                                      274 / 274
  9  pore.local_peak                                               274 / 274
 10  pore.global_peak                                              274 / 274
 11  pore.geometric_centroid                                       274 / 274
 12  throat.global_peak                                            440 / 440
 13  pore.inscribed_diameter                                       274 / 274
 14  pore.extended_diameter                                        274 / 274
 15  throat.inscribed_diameter                                     440 / 440
 16  throat.total_length                                           440 / 440
 17  throat.direct_length                                          440 / 440
 18  throat.perimeter                                              440 / 440
 19  pore.volume                                                   274 / 274
 20  pore.surface_area                                             274 / 274
 21  throat.cross_sectional_area                                   440 / 440
 22  throat.equivalent_diameter                                    440 / 440
--------------------------------------------------------------------------------
  #  Labels                                                  Assigned Locations
--------------------------------------------------------------------------------
  2  pore.all                                                              274
  3  throat.all                                                            440
  4  pore.boundary                                                          49
  5  pore.xmin                                                              12
  6  pore.xmax                                                              13
  7  pore.ymin                                                              12
  8  pore.ymax                                                              12
--------------------------------------------------------------------------------
```

Define Pore and Throat Sizes

PoreSpy returns many different pieces of size information, but does not make any choices about *which* ones to use in OpenPNM.

For instance, there are 3 different value of pore diameter, but none are `'pore.diameter'` . We must choose.

In [110]:
```
pn['pore.diameter'] = pn['pore.inscribed_diameter']
pn['pore.volume'] = pn['pore.region_volume']
pn['throat.diameter'] = pn['throat.inscribed_diameter']*2
pn['throat.volume'] = 0.0
```

Define Thermodynamic and Capillary Properties

We need to create a Phase object to represent the mercury, with its surface tension and contact angle.

We also need to define the physical law for capillary entry pressure:

In [111]:
```
hg = op.phase.Mercury(network=pn)
hg.add_model(propname='throat.entry_pressure',
             model=op.models.physics.capillary_pressure.washburn)
```

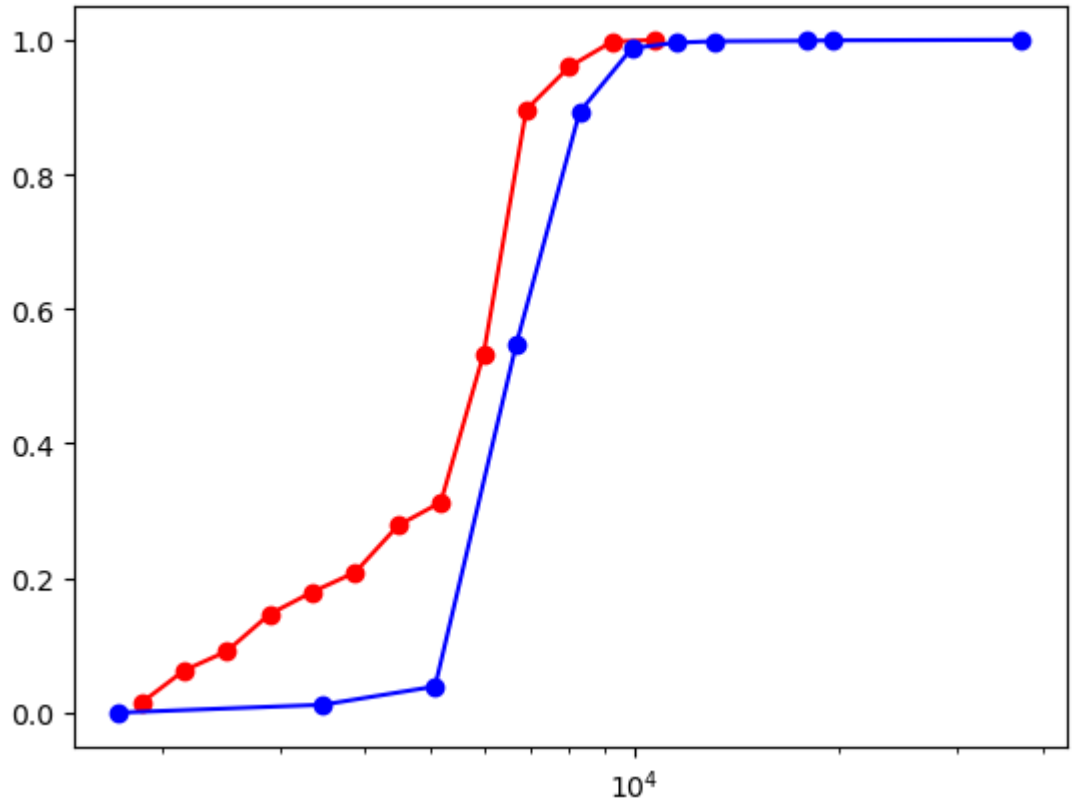
Create Drainage Algorithm

In [112]:
```
mip = op.algorithms.Drainage(network=pn, phase=hg)
mip.set_inlet_BC(pores=pn.pores(['xmin', 'xmax', 'ymin', 'ymax']))
mip.run()
```

```
Performing drainage simulation: 100%                    25/25 [00:00<00:00, 323.57it/s]
```

In [113]:

```
data = mip.pc_curve()
plt.semilogx(data.pc, data.snwp, 'ro-')
plt.semilogx(drn.pc, drn.snwp, 'bo-');
```



Introduction to Scientific Python

Python has become a very popular choice for scientific applications for a few reasons:

- It's free, unlike the main alternative Matlab which is free for students but very expensive for industry
- It's easy to learn and use
- It's very powerful and general purpose, so you can learn Python for numerical studies but use it for making a website
- It has a huge ecosystem of existing packages, creating ever more momentum for the addition of new packages

Python is not without it's downsides though:

- It is actually terrible at numerical stuff!
- It is very slow

These downsides have been mostly addressed by the package called Numpy (https://numpy.org) which provides Python with a new da
There is also a related package called Scipy (https://scipy.org) which provides a massive library of functions that operate on Numpy arr
Transforms, ODE solvers, etc). In fact, many packages have sprouted up around Numpy/Scipy, which are collectively called the "Scipy
Pandas, scikit-learn, etc).

> The bottom line is that you need to install both Python and the "Scipy Stack" packages. This is most conviently done using the
> Distribution (https://www.anaconda.com/products/distribution). This distribution is maintained and distributed by a company wh
> by the original authors of Numpy. They make money by providing advanced versions of Anaconda to companies, but they offer
> package for free.

Doing Math in Python: The Wrong Way

Let's quickly look at using pure Python to do some basic math. Start by putting some numbers into a `list`:

In [5]:
```
a = [1, 2, 3, -2, 0.5]
```

Now let's mulitply each number by 2:

In [6]:
```
a = a * 2
print(a)
```

```
[1, 2, 3, -2, 0.5, 1, 2, 3, -2, 0.5]
```

OOPS! That is not what we wanted. This illustrates that Python is really not math focused. Let's do it inside a for-loop:

In [7]:
```
for i in range(len(a)):
    a[i] = a[i] * 2
print(a)
```

```
[2, 4, 6, -4, 1.0, 2, 4, 6, -4, 1.0]
```

OK, that worked. But there is one more problem...Python let's us put ANYTHING into `lists` :

In [8]:
```
a = [1, 1.0, True, 'one']
```

Here we have an integer, a float, a boolean, and a string. Trying to do make on this mixed bag will have problems:

In [9]:
```
for i in range(len(a)):
    a[i] = a[i]*2
print(a)
```

```
[2, 2.0, 2, 'oneone']
```

This did as expected for the integer and float, but it converted the boolean to a integer and did the 'doubling' trick on the string. What ha
Python performed the multiplication of each element differently depending on the type of element. This required Python to stop and ins
element, making it slow.

Doing Math in Python: The Numpy Way

The slowness of Python comes from the fact that it must inspect each element in a the `list` to get its type, then apply the the correct avoids this by offering a new type of `list` where all elements are forced to be the same type. This `list` is actually called an `ndarr` array. Let's see it in action:

In [10]:
```
import numpy as np
a = np.array([1, 2, 3, 4.0, True], dtype=float)
print(a)
```

```
[1. 2. 3. 4. 1.]
```

Here we have told Numpy that this `array` should be of type `float`, so it converts all values to `float` for us. Now we can do some

In [11]:
```
a = a*2
print(a)
```

```
[2. 4. 6. 8. 2.]
```

All operations in Numpy as done in a 'vectorized' way, meaning everything list done 'element-wise'. There is no need to use a for-loop t elements, since Numpy will essentially do this for us:

In [12]:
```
a = 2*a*a + a + 3
print(a)
```

```
[ 13.  39.  81. 139.  13.]
```

Python's Data Containers

Python's ease of use is one it's main appeals. This ease of use comes from Python's ability to mix and match data of different types wit such as the `list` we saw above. In addition to the `list`, there is also the `dict` and `tuple`. The differences are:

- `list` let's you read and write elements using numerical indices
- `dict` let's you access elements by name
- `tuple` let's you read elements by index, but forbids you from writting.

There is one additional way to collect data into a bundle: as `attributes` of an `object`.

Let's quickly explore each of these below:

In [13]:
```
e = list()
e.append(np.array([1, 3, 4], dtype=int))
e.append(1.0)
e.append('two')
print(e)
```

```
[array([1, 3, 4]), 1.0, 'two']
```

In [14]:
```
d = dict()
d['dave'] = 1.0
d['bob'] = 1
d['fred'] = True
d['george'] = 'one'
print(d)
```

```
{'dave': 1.0, 'bob': 1, 'fred': True, 'george': 'one'}
```

In [15]:
```
t = tuple(e)
print(t)
```

```
(array([1, 3, 4]), 1.0, 'two')
```

And finally, let's look at the `object` :

In [16]:
```
class NewTypeOfObject:
    pass
```

In [17]:
```
obj = NewTypeOfObject()
obj.bob = np.array([1, 3, 5], dtype=float)
obj.dave = 1.0
```

Overview of Volumetric Images

Comparison of 2D and 3D Images

We are all used to 2D digital pictures. When zooming tightly on such an image we are familiar with seeing the "pixels" of the image. As images are just a stack of 2D images, and the pixels are usually called "voxels" indicating they are a "volume element".

Color 2D images are just a massive array of pixels, each containing a numerical value indicating its color. In fact, color images are a 3 each one indicating the intensity of the Red, Green, and Blue (RGB) at each location. When these 3 colors are mixed together they car possible colors. So even 2D image are actually 3D arrays.

In [74]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
```

In [75]:
```
im = data.coffee()
plt.imshow(im);
```

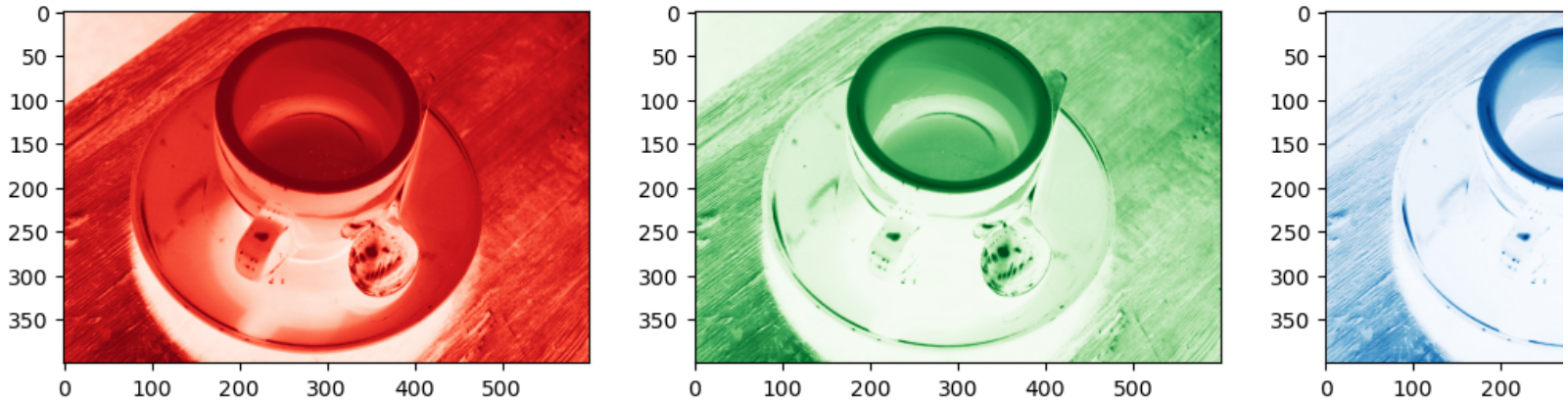
Color Images

As mentioned, color images are actually 3 images stacked together:

In [76]:
```
print(im.shape)
```

```
(400, 600, 3)
```

In [77]:
```
fig, ax = plt.subplots(1, 3, figsize=[15, 5])
ax[0].imshow(im[..., 0], cmap=plt.cm.Reds)
ax[1].imshow(im[..., 1], cmap=plt.cm.Greens)
ax[2].imshow(im[..., 2], cmap=plt.cm.Blues);
```



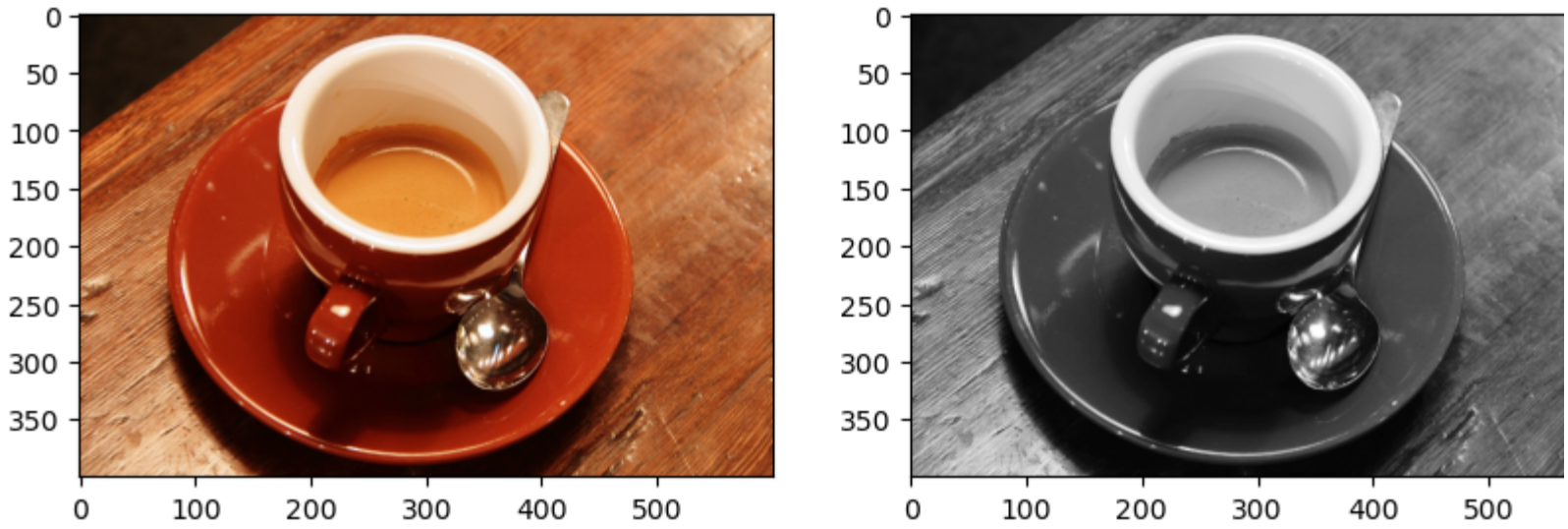
Grayscale Images

When dealing with 'technical' or 'scientific' images, we usually only want 'grayscale' images where the color imformation is discarded a
some scalar intensity value at each location.

In [78]:

```
from skimage.color import rgb2gray
fig, ax = plt.subplots(1, 2, figsize=[10, 5])
ax[0].imshow(im)
im2 = rgb2gray(im)
print(im2.shape)
ax[1].imshow(im2, cmap=plt.cm.gray);
```

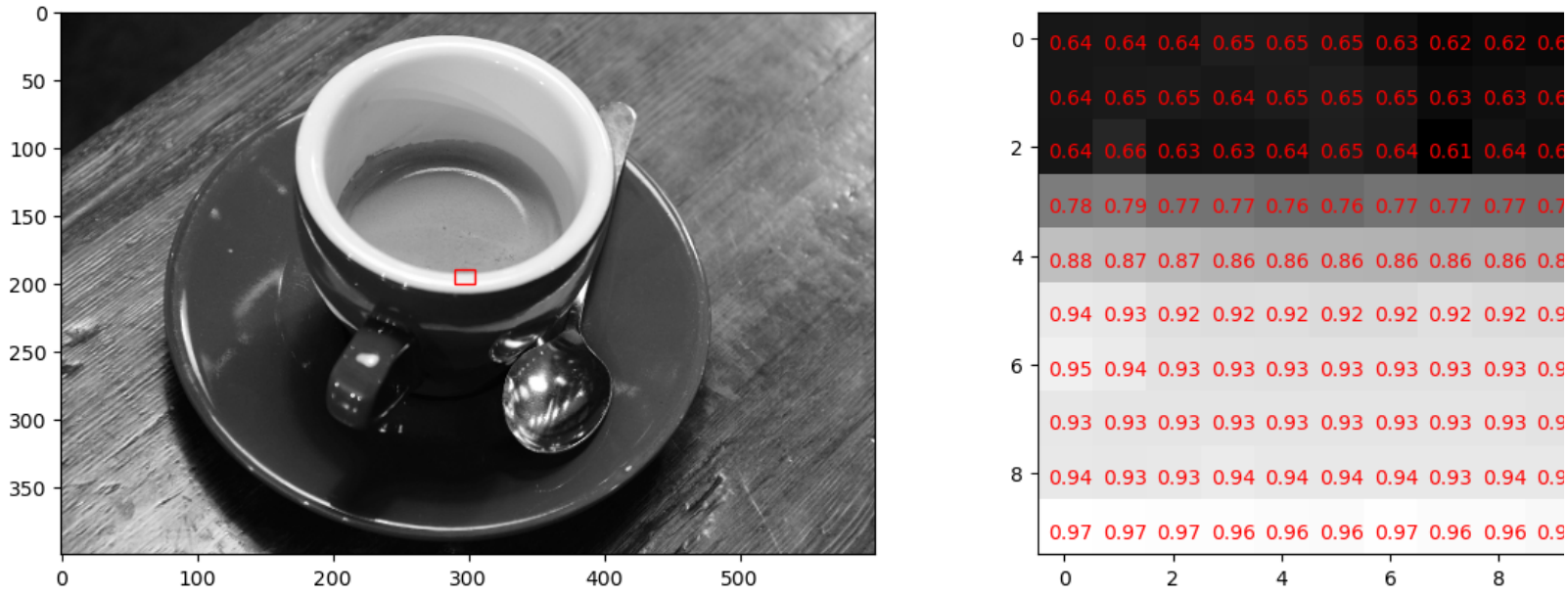
```
(400, 600)
```



Zooming in closely to a spot in the 2D grayscale images reveals the pixels, which contain scalar values:

In [79]:

```
from skimage.color import rgb2gray
import matplotlib as mpl
import matplotlib.patches as patches
mpl.rcParams['text.color'] = 'red'

fig, ax = plt.subplots(1, 2, figsize=[16, 8])

ax[0].imshow(im2, cmap=plt.cm.gray)
rect = patches.Rectangle((290, 190), 15, 10, linewidth=1, edgecolor='r', facecolor='none')
ax[0].add_patch(rect)

temp = im2[190:200, 290:305]
ax[1].imshow(temp, cmap=plt.cm.gray)

x, y = np.meshgrid(range(temp.shape[0]), range(temp.shape[1]))
x = x.flatten()
y = y.flatten()
for i, j in zip(x, y):
    ax[1].annotate(np.around(temp[i, j], decimals=2), xy=(j-0.3, i+0.2))
```



3D or Volumetric Images

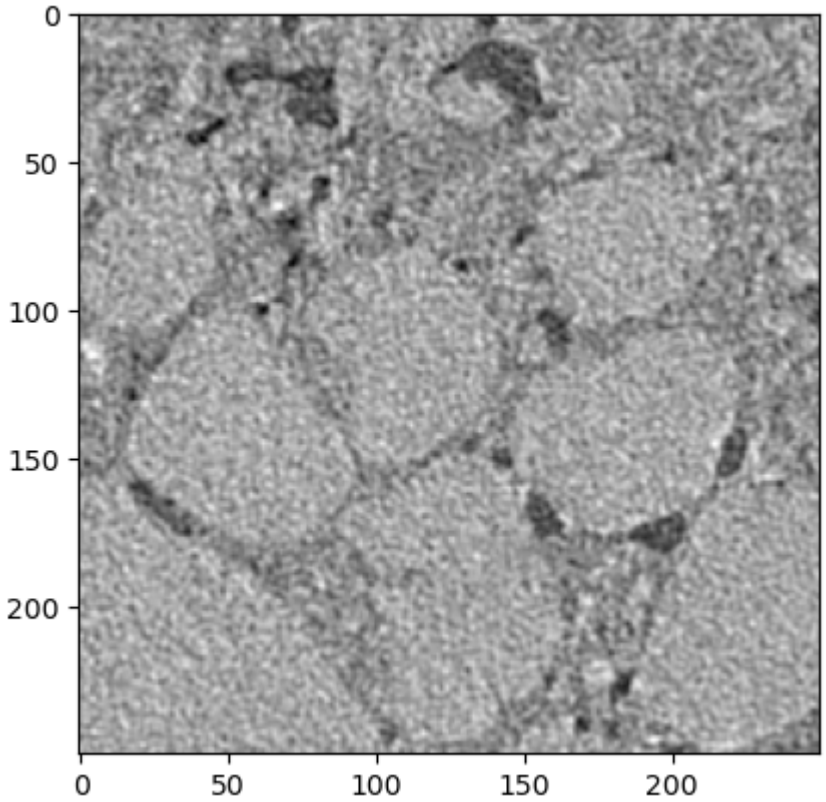
Now let's look at a 3D image. The `imageio` package is very helpful for reading in different file types. Volumetric images are often stor
since these natively support multiple layers and are universally recognized as an image format. A small section of a sandstone is inclu

In [87]:
```
import imageio
im = imageio.volread('images/sandstone.tif')
im.shape
```

Out[87]: (250, 250, 50)

When viewing a 3D image using tools normally meand for 2D images, be sure to extract a single slice:

In [88]:
```
plt.imshow(im[..., 25], cmap=plt.cm.gray);
```

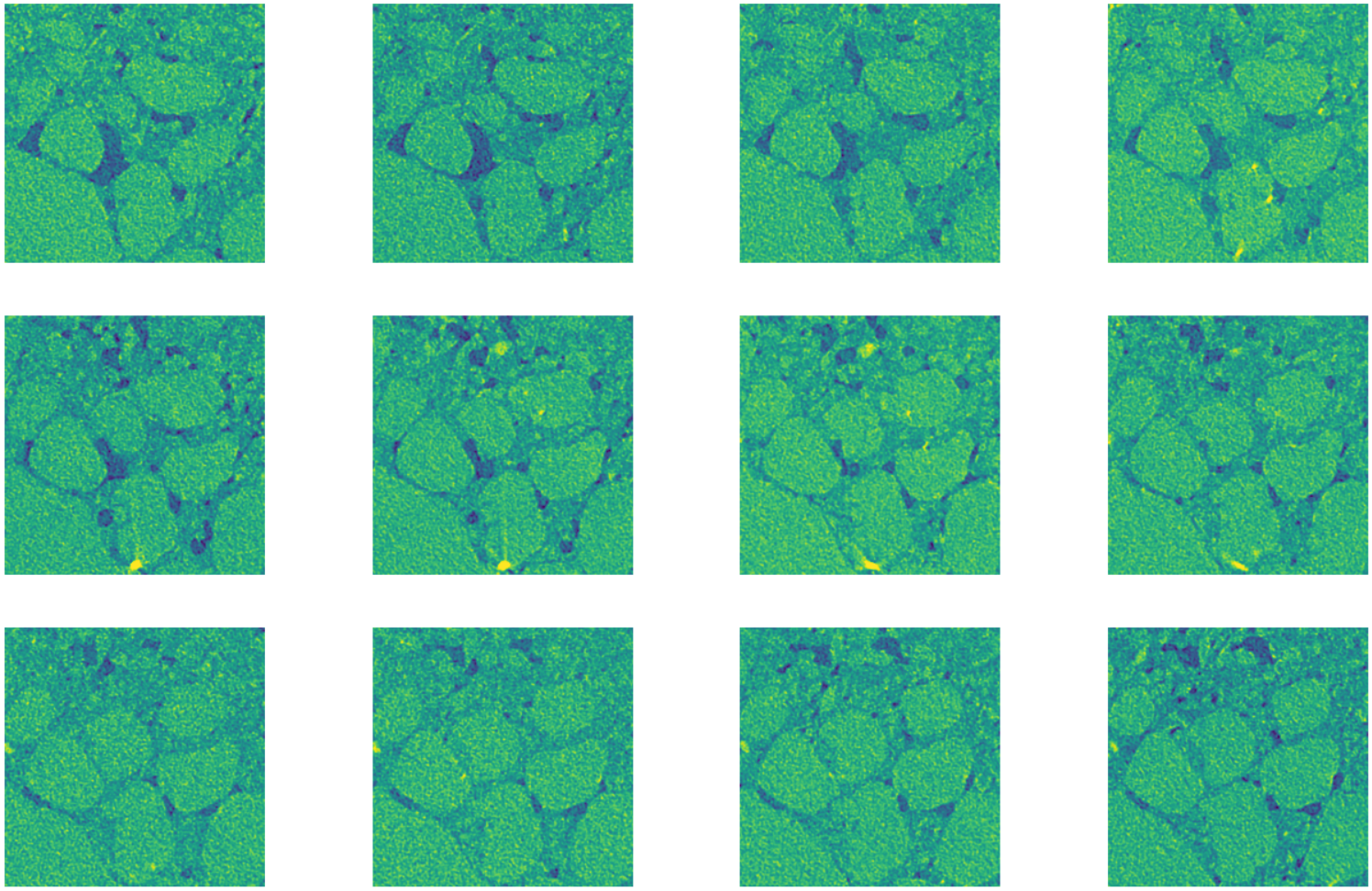


Here we can see solid grains, dark voids, and some intermediate intensity material which is probably clay. This is classified as "unresol
can't see the pores but we know they are there.

The above image is slice 25 from a stack of 50 slices. There is not a simple way to visualize 3D images in Jupyter notebooks, so we wi

In [89]:

```
import porespy as ps
N, M = 3, 5
fig, ax = plt.subplots(N, M, figsize=[20, 10])
k = 0
for i in range(N):
    for j in range(M):
        ax[i][j].imshow(im[..., k])
        ax[i][j].axis(False)
        k += 2
```



Using External Tool to Visualize

There are several different packages which can be used to visualize 3D tiff images. Paraview is "FOSS" maintained by US National La
of the type of output you can get from this:

In [90]:
```
montage = imageio.v3.imread('images/tiff_stack_demo.png')
fig, ax = plt.subplots(1, 1, figsize=[16, 8])
ax.imshow(montage)
plt.axis(False);
```

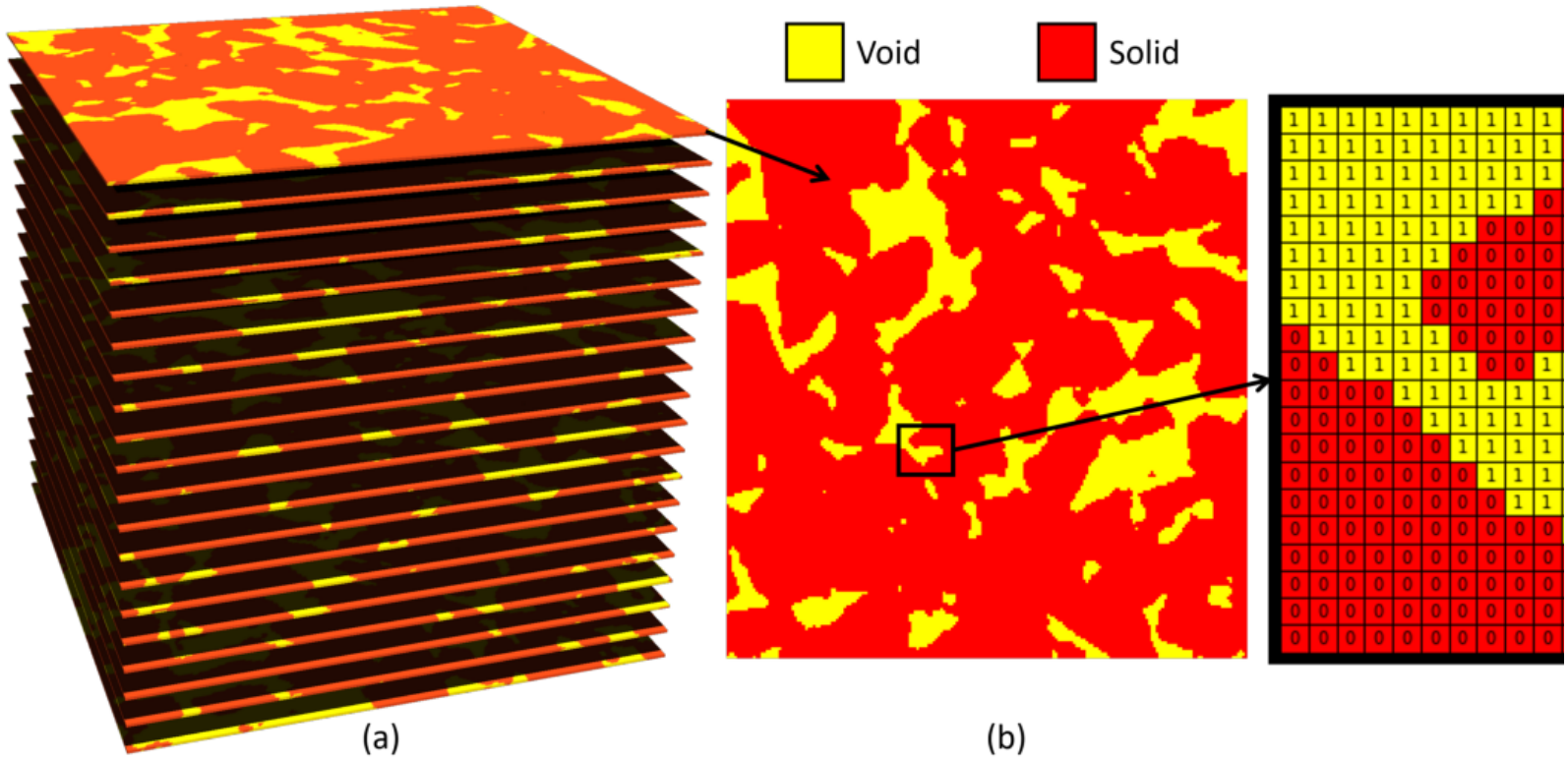
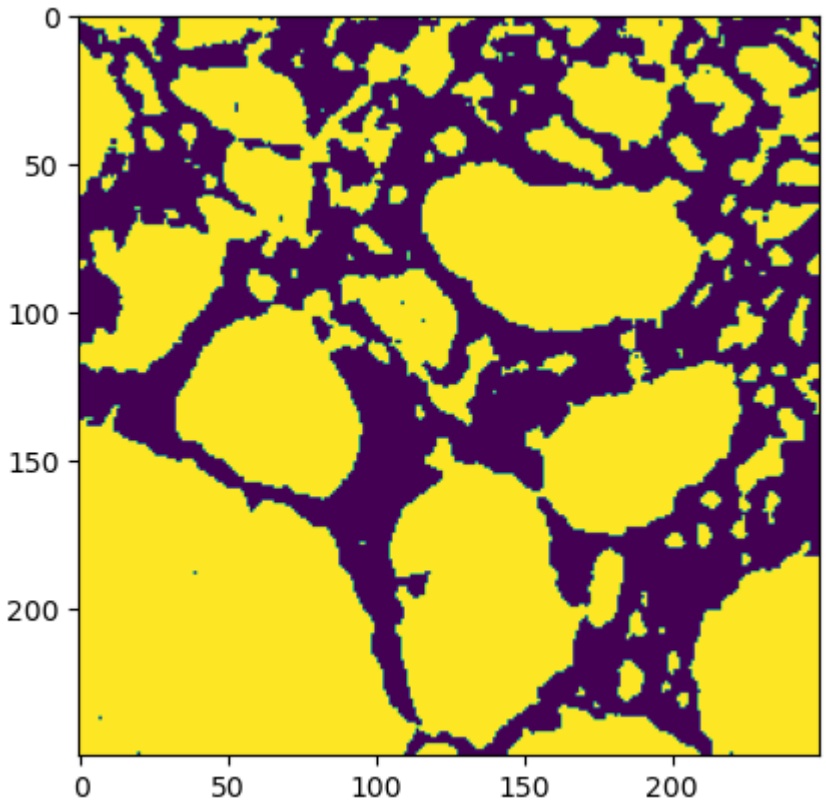


Image Segmentation: Identifying Solid and Void

One of the most important yet challenging steps is the filtering of noise from the greyscale images obtained experimentally. As can be s
inside the solid grains are generally brighter than the other phases, but this is only an average and there are lots of dark voxels in the g
some averaging be applied to remove or smooth the noise. For the sake of a simple demonstration, let's applie a median filter

In [91]:
```
import scipy.ndimage as spim
im2 = spim.median_filter(im, 5)
plt.imshow(im2[..., 0] > 158);
```



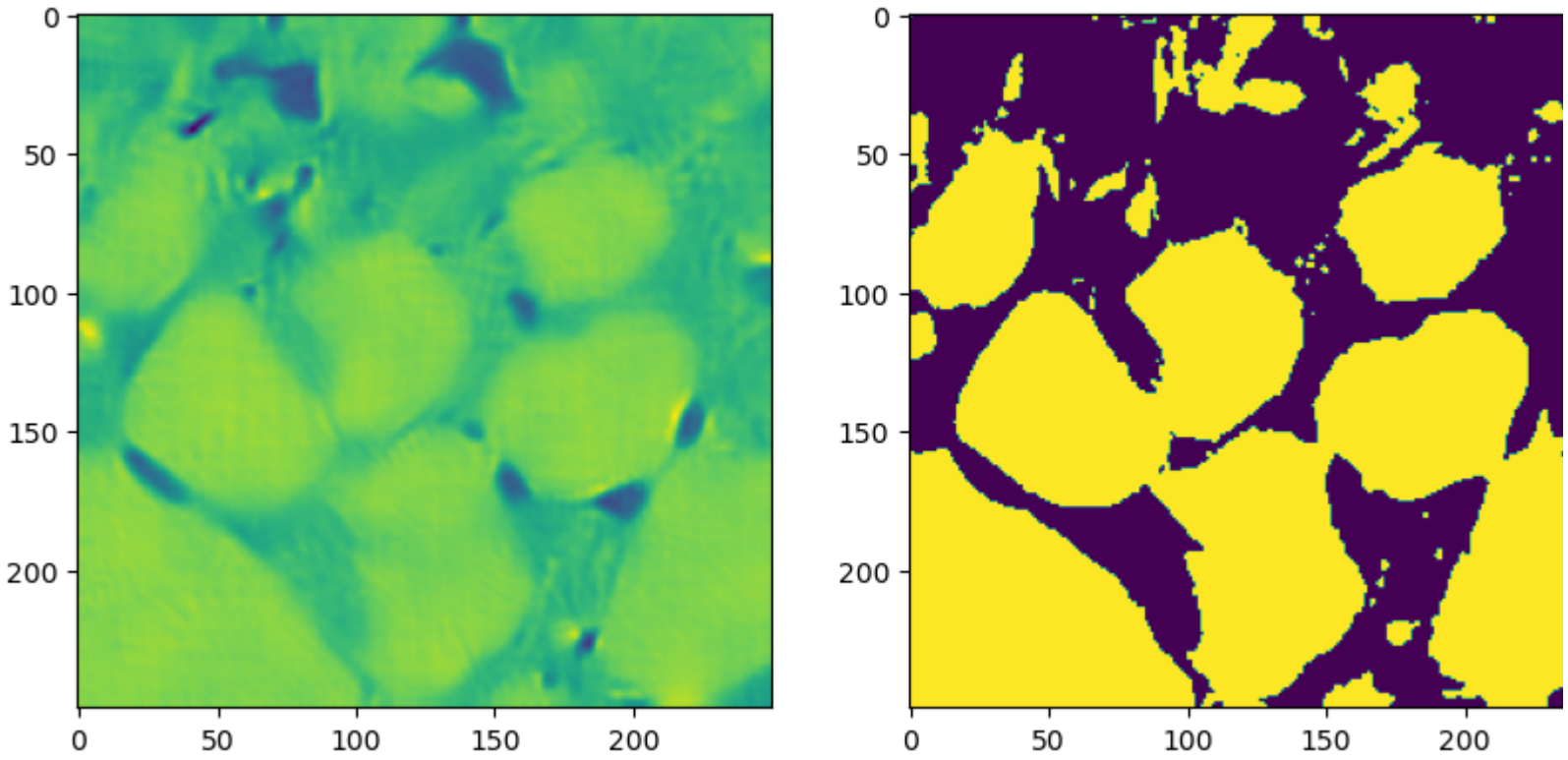
After applying the `median_filter` we can find (by trial and error) that all voxels with a value higher than 158 belong to solid.

Of course, a better filtering process should have been applied to really reduce the noise and aid this process. It's important to get the c
single stray voxel in the middle of a pore will drastically alter any quantitative analysis.

In [92]:
```
from skimage.restoration import non_local_means, estimate_sigma
im3 = (im - im.min())/(im.max() - im.min())
im3 = im3[..., 25]
```

In [93]:
```
s = estimate_sigma(im3)
im4 = non_local_means.denoise_nl_means(im3, sigma=s, patch_size=11, patch_distance=9, h=0.1)
im4 = (im4 - im4.min())/(im4.max() - im4.min())
```

In [94]:
```
fig, ax = plt.subplots(1, 2, figsize=[10, 5])
ax[0].imshow(im4)
ax[1].imshow(im4 > 0.73);
```



Using PoreSpy for Quantitative Image Analysis

Finally we are ready to start analyzing images. The `porespy` package has many functions that are custom built for the study of porou includes tools for calculating pore sizes, obtaining tortuosity, analyzing anisotropy and more.

PoreSpy is organized into the following modules:

- `generators` : Functions for generating artificial images useful for testing and validation
- `filters` : Functions for manipulating images to replace voxel values with some computed values
- `metrics` : Functions for analyzing images to extract quantitative information
- `simulations` : Functions for performing physical simulations directly on images
- `networks` : Functions for extracting pore networks from images for use in OpenPNM
- `visualization` : Functions for performing basic visualizations of 3D images
- `io` : A small set of functions for converting between various formats
- `tools` : Functions mostly used within PoreSpy itself

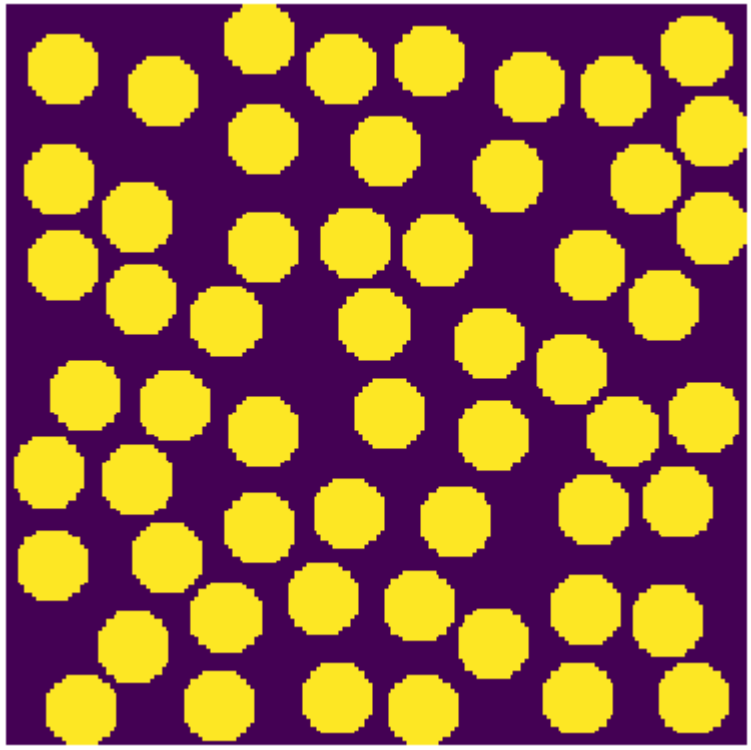
In [1]:
```
import porespy as ps
import matplotlib.pyplot as plt
import numpy as np
from edt import edt
import scipy.ndimage as spim
```

Generating an Image for Demonstration

Let's start by generating an image of non-overlapping spheres using the 'random sequential addition' algorithm.

> Note that all demonstrations will be done with 2D images to conserve CPU time and aid visualization

In [2]:
```
im = ps.generators.rsa([200, 200], r=10, clearance=1)
plt.imshow(im, interpolation='none', origin='lower')
plt.axis(False);
```



Defining the Image "Foreground"

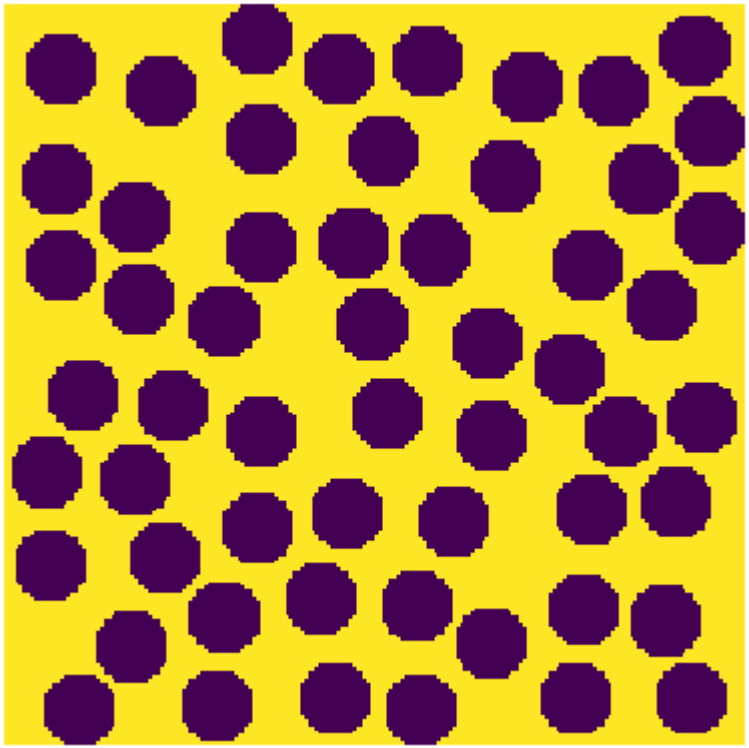
The *foreground* or the *phase of interest* in PoreSpy is always the voxels marked `True`.

The `rsa` function has labelled the spheres as `True`. If we want the spheres to be solid grains then the we must invert the image.

This is very simple:

In [3]:

```
im = ~im
plt.imshow(im, interpolation='none', origin='lower')
plt.axis(False);
```



Porosity

The most basic property of an image, and a sample of porous media in general, is the porosity.

This is defined as the ratio of void volume to bulk volume.

In terms an image this means the number of "yellow" pixels" divided to the total number of pixels in the image:

In [4]:
```
porosity = np.sum(im)/np.size(im)
print(porosity)
```

```
0.573
```

Pore Size Distributions

The spheres above all have a radius of 10 voxels, but the spaces between the spheres are distributed due to their random placement.

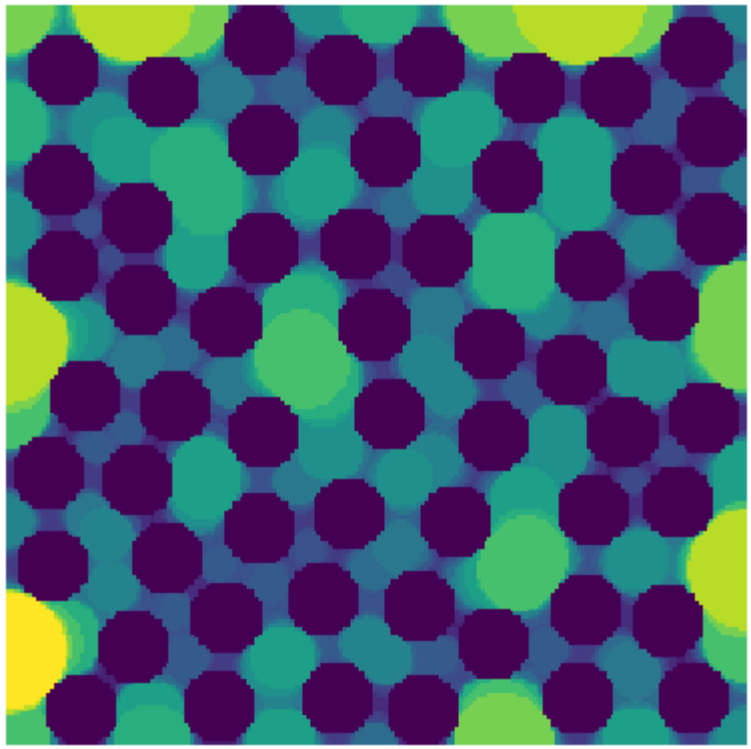
One of the common ways to estimate pore sizes is using the "local thickness".

PoreSpy has a `local_thickness` function, which draws the largest possible spheres in all the *foreground* pixels, and indicates the si
voxel value it inserts into the image:

In [5]:

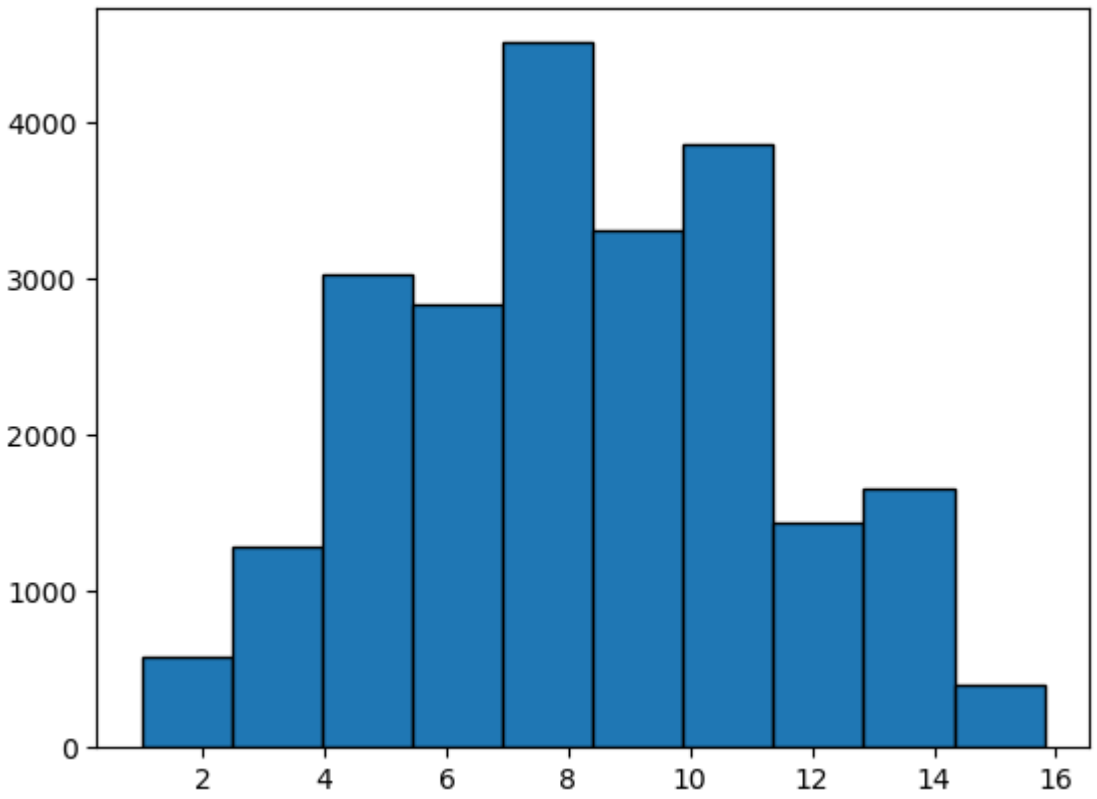
```
lt = ps.filters.local_thickness(im)
plt.imshow(lt, interpolation='none', origin='lower')
plt.axis(False);
```

```
  0%|          | 0/25 [00:00<?, ?it/s]
```



We can get a quick look at the pore size distribution by creating a histogram of the voxel values:

In [6]:
```
plt.hist(lt[lt > 0].flatten(), bins=10, edgecolor='k');
```



The previous curve showed that the average pore radius is around 8 voxels. PoreSpy offers a more rigorous way to analyze this outpu module. In this case the results of the `local_thickness` filter are passed to the `pore_size_distribution` function:

In [7]:
```
psd = ps.metrics.pore_size_distribution(lt, bins=15, log=False)
```

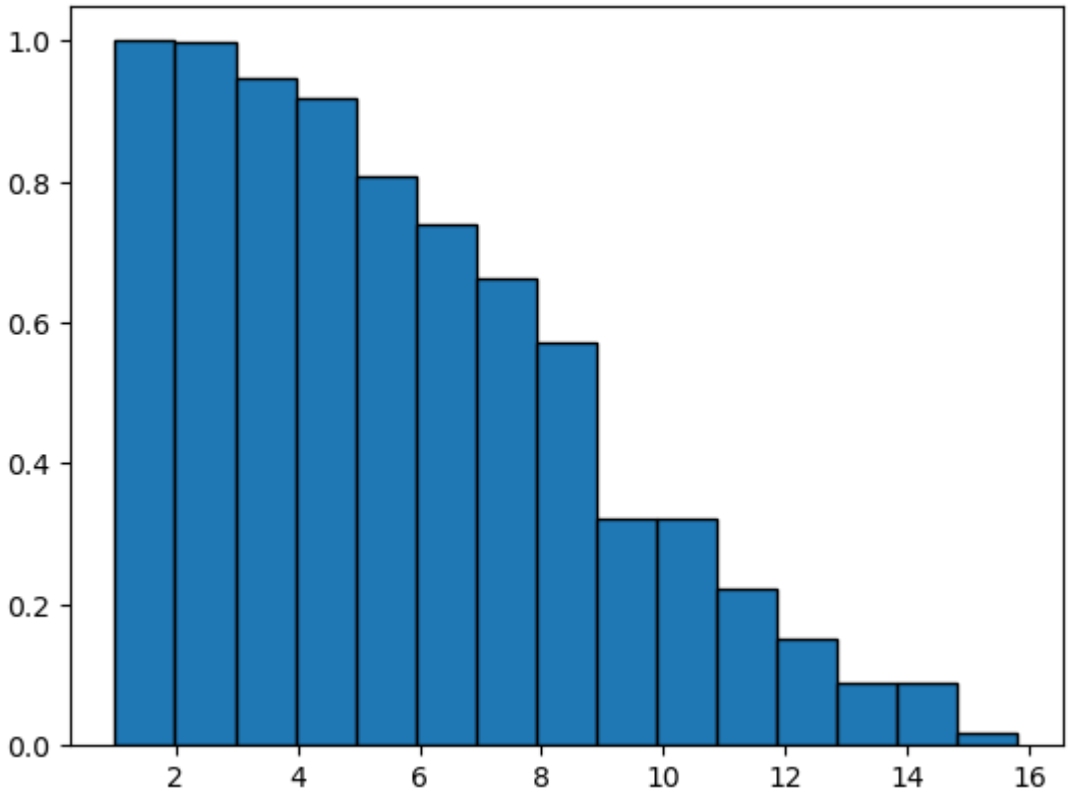
The `psd` object that is returned has various data stored as "attributes". This object can be `printed` to inspect these values:

In [8]:
```
print(psd)
```

Item	Description
R	Array of size (15,)
pdf	Array of size (15,)
cdf	Array of size (15,)
satn	Array of size (15,)
bin_centers	Array of size (15,)
bin_edges	Array of size (16,)
bin_widths	Array of size (15,)

Using these values we can plot the cumulative distribution as follows:

In [9]:
```
plt.bar(x=psd.R, height=psd.cdf, width=psd.bin_widths, edgecolor='k');
```

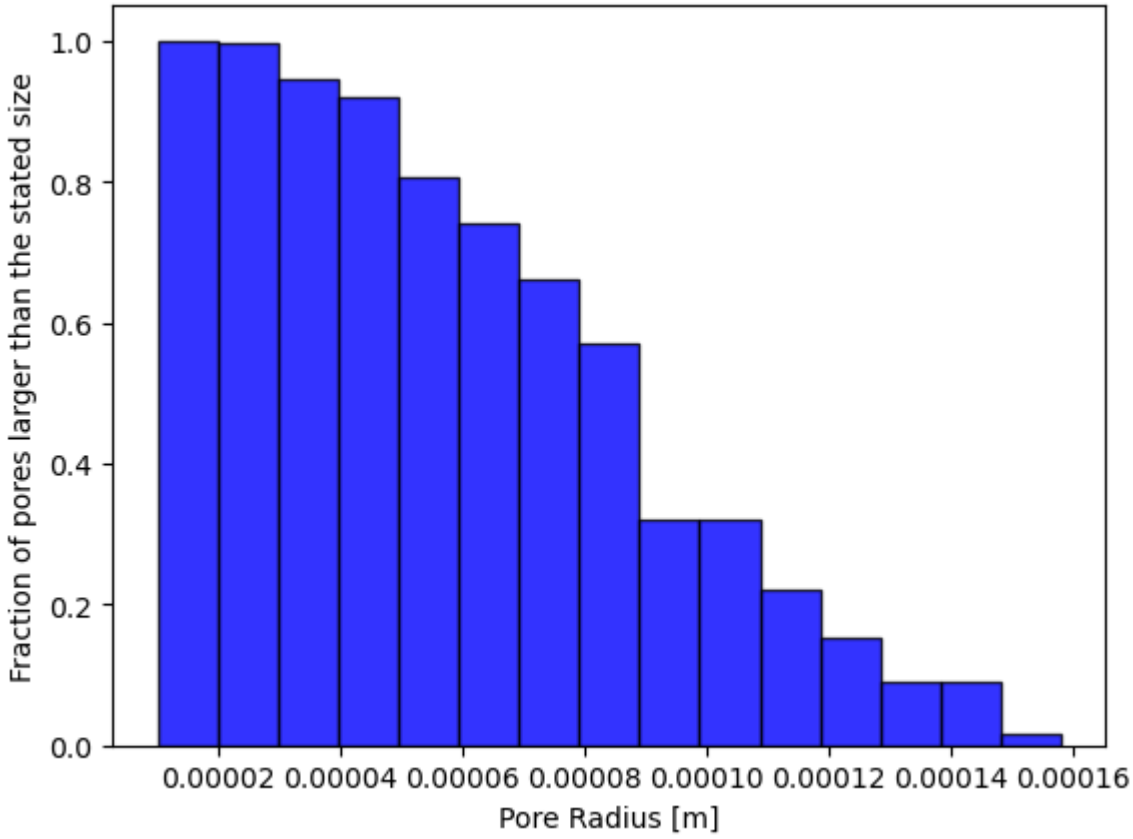


Setting Voxel Size

The above analysis was conducted in units of voxels. It is usually the case that physical sizes are desired, and moreover, all imaging te
voxel size.

The above few steps can be re-run but with the voxel size specified (assume 10 um/voxel):

In [10]:
```
psd = ps.metrics.pore_size_distribution(lt, bins=15, log=False, voxel_size=1e-5)
plt.bar(x=psd.R, height=psd.cdf, width=psd.bin_widths, edgecolor='k', color='b', alpha=0.8)
plt.xlabel('Pore Radius [m]')
plt.ylabel('Fraction of pores larger than the stated size');
```



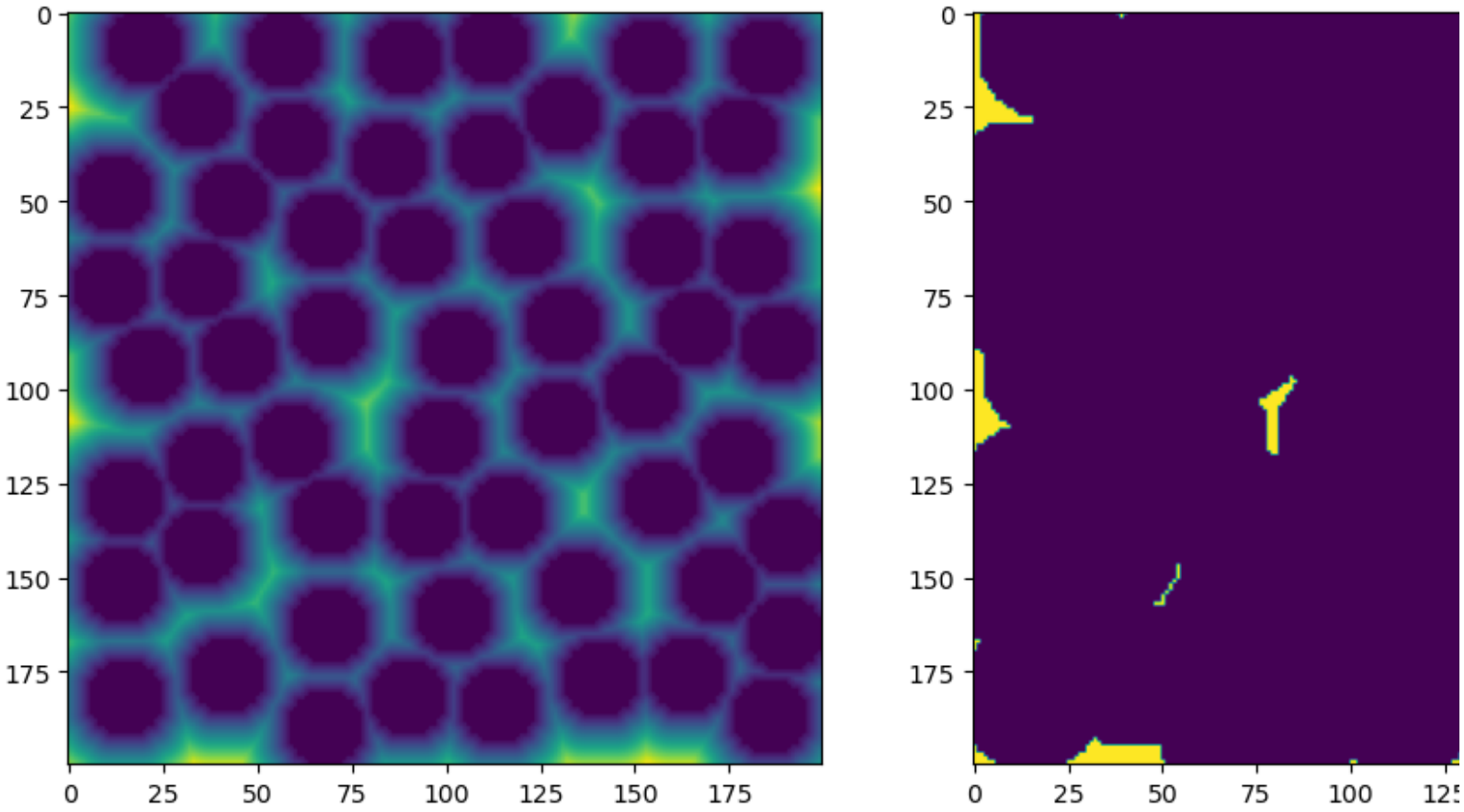
Background on How Local Thickness is Computed

It is instructive to look at how the local thickness is calculated, since this involves some elementary image operations which are used w
dilation and erosion.

Step 1: Find all void voxels on which a sphere is size R can be drawn using a distance transform

In [11]:

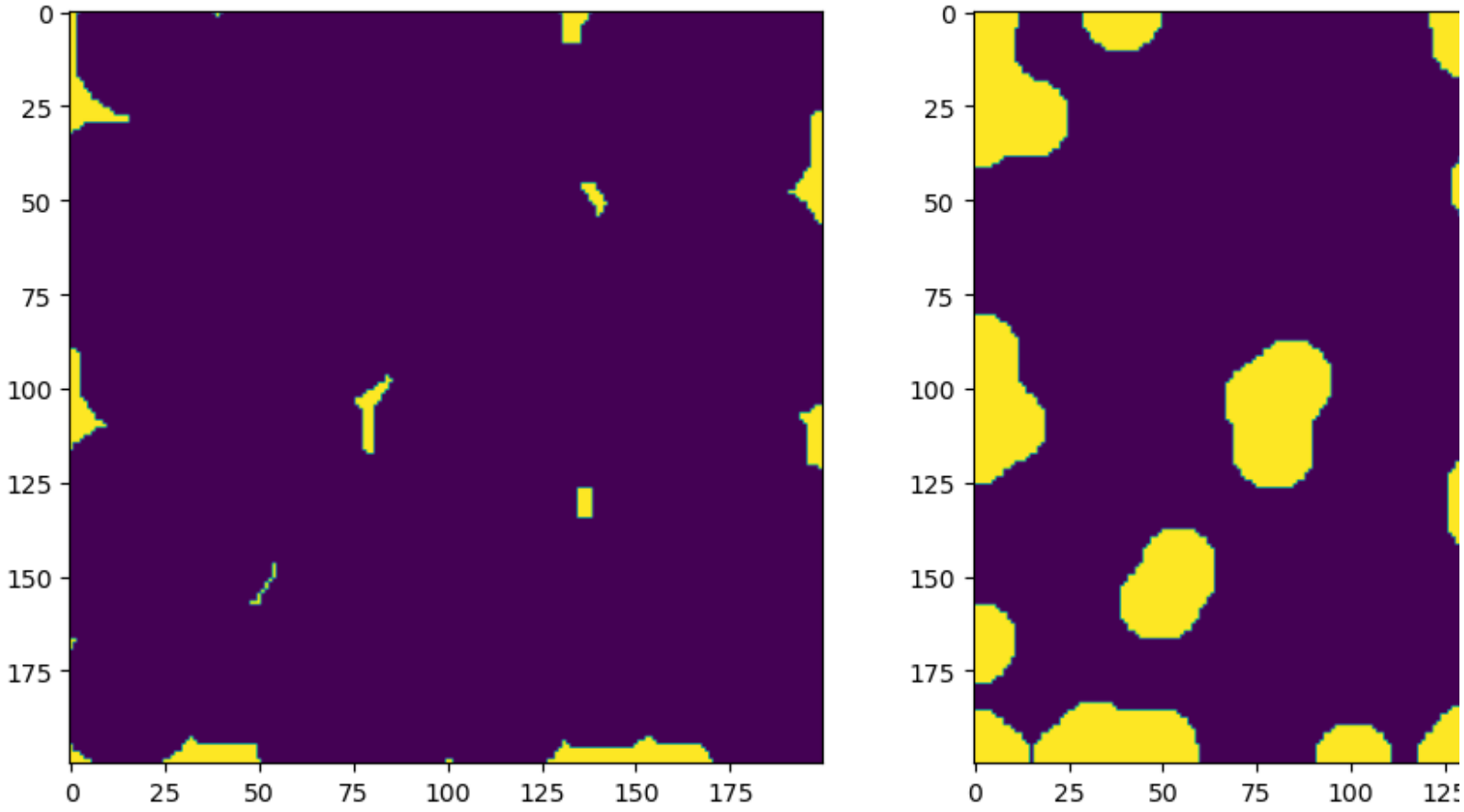
```
lt = np.zeros_like(im, dtype=int)
R = 10
dt = edt(im)
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(dt)
ax[1].imshow(dt >= R);
```



Step 2: Dilate the surviving voxels by R

In [12]:

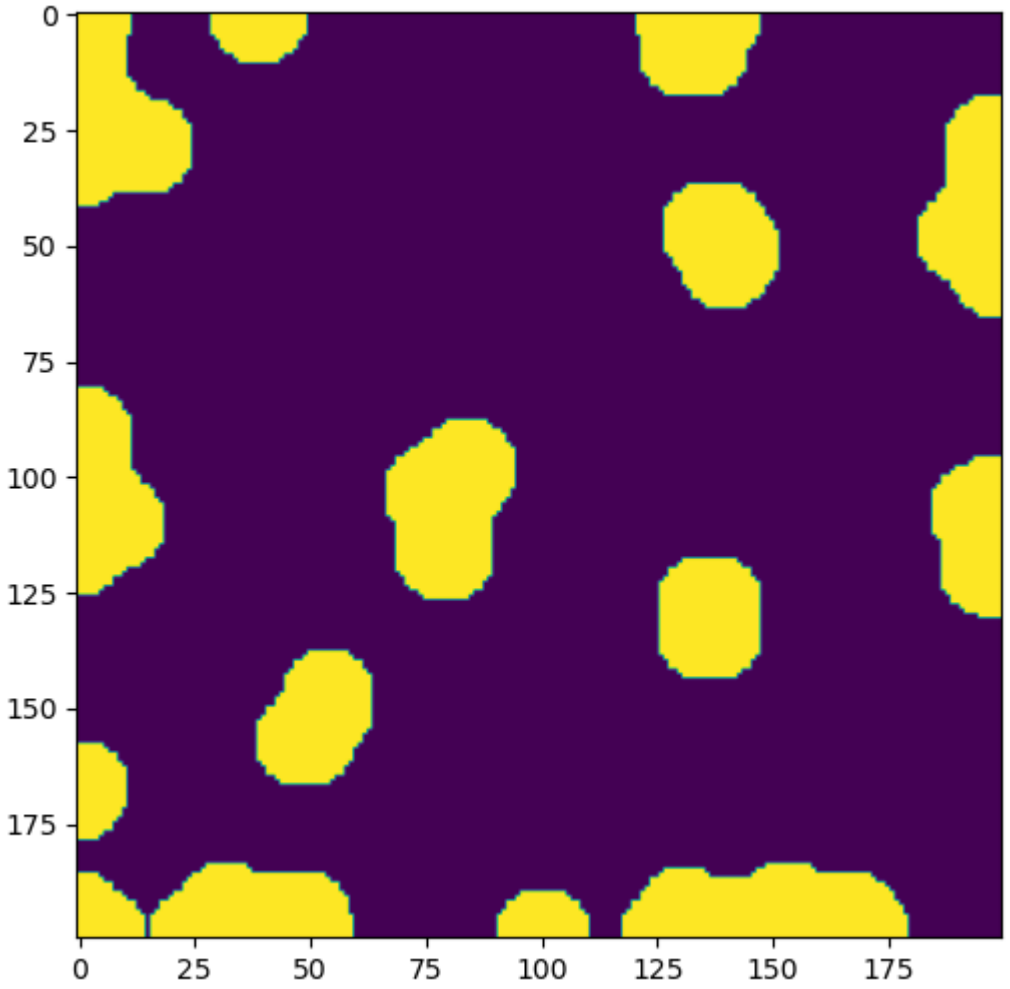
```
temp = spim.binary_dilation(dt >= R, structure=ps.tools.ps_round(r=R, ndim=2))
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(dt >= R)
ax[1].imshow(temp);
```



Step 3: Mark these locations with the radius, unless they already have a larger value

In [13]:

```
lt = lt + R*temp*(lt == 0)
fig, ax = plt.subplots(1, 1, figsize=[6, 6])
ax.imshow(lt);
```

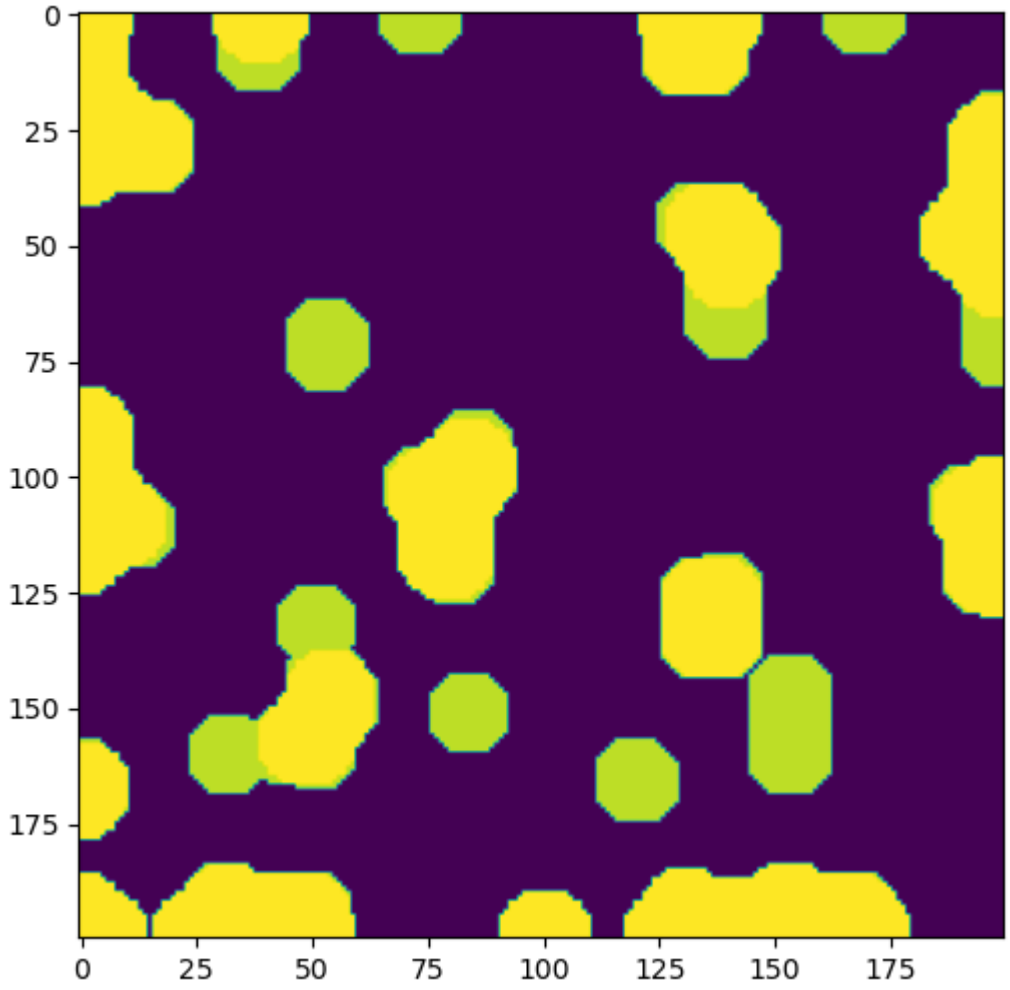


Step 4: Repeat with R=9

Now repeat steps 1 to 3 with a smaller value of R:

In [14]:

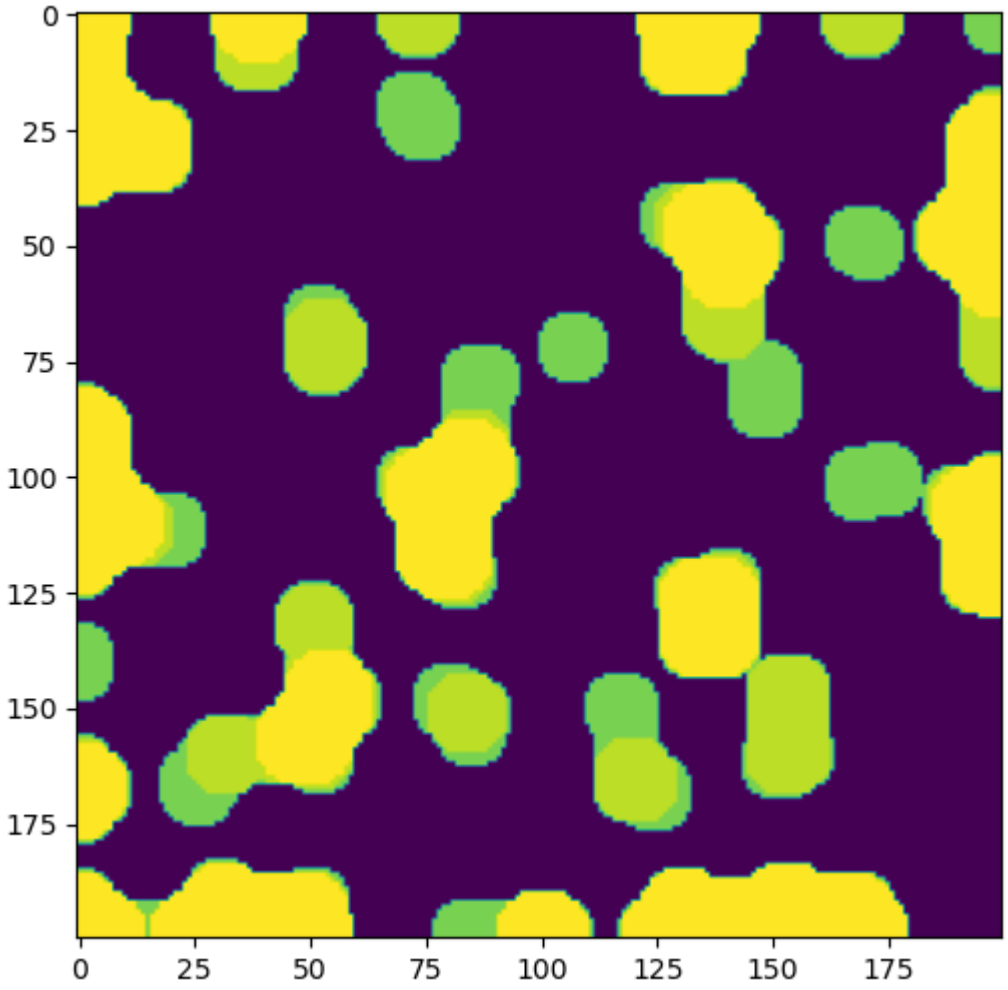
```
R = 9
temp = spim.binary_dilation(dt >= R, structure=ps.tools.ps_round(r=R, ndim=2))
lt = lt + R*temp*(lt == 0)
fig, ax = plt.subplots(1, 1, figsize=[6, 6])
ax.imshow(lt);
```



Repeat with R=8

In [15]:

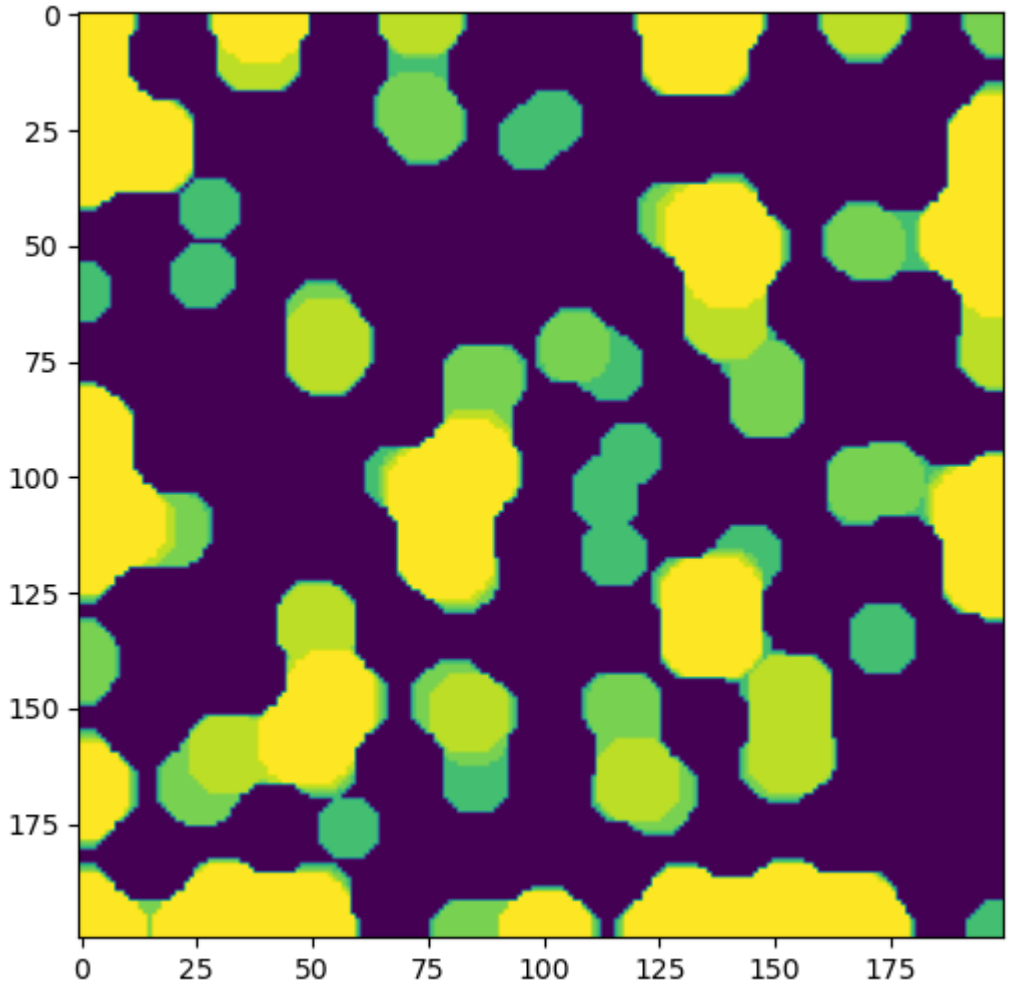
```
R = 8
temp = spim.binary_dilation(dt >= R, structure=ps.tools.ps_round(r=R, ndim=2))
lt = lt + R*temp*(lt == 0)
fig, ax = plt.subplots(1, 1, figsize=[6, 6])
ax.imshow(lt);
```



Repeat with R=7

In [16]:

```
R = 7
temp = spim.binary_dilation(dt >= R, structure=ps.tools.ps_round(r=R, ndim=2))
lt = lt + R*temp*(lt == 0)
fig, ax = plt.subplots(1, 1, figsize=[6, 6])
ax.imshow(lt);
```

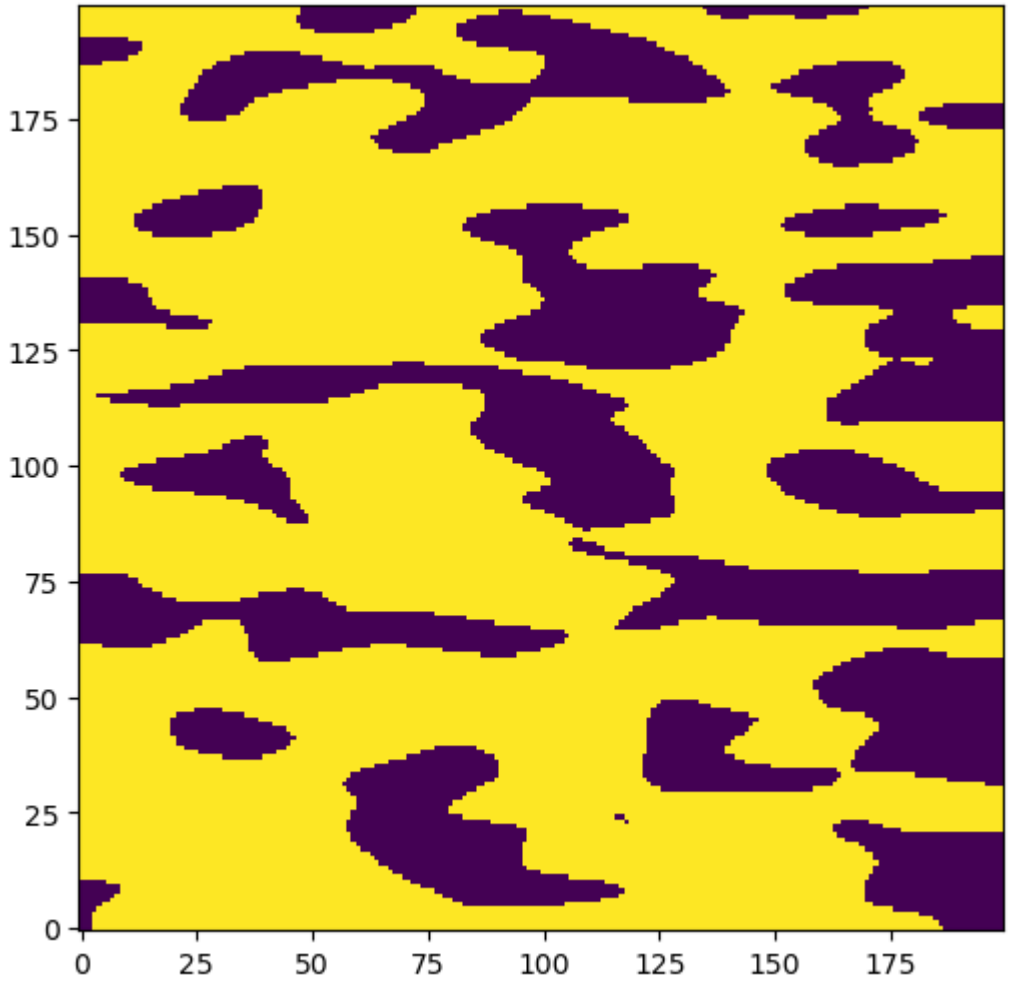


Chord Length Distributions

Chords are lines drawn across the void space, spanning from one side of a pore to the other. Their length therefore indicates the pore s

They can be drawn in random directions, but the real power of chords is to detect anistropy: pores that are elongated in one direction

In [17]:
```
im = ps.generators.blobs([200, 200], blobiness=[1.5, 0.5], porosity=0.7)
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow(im, interpolation='none', origin='lower');
```



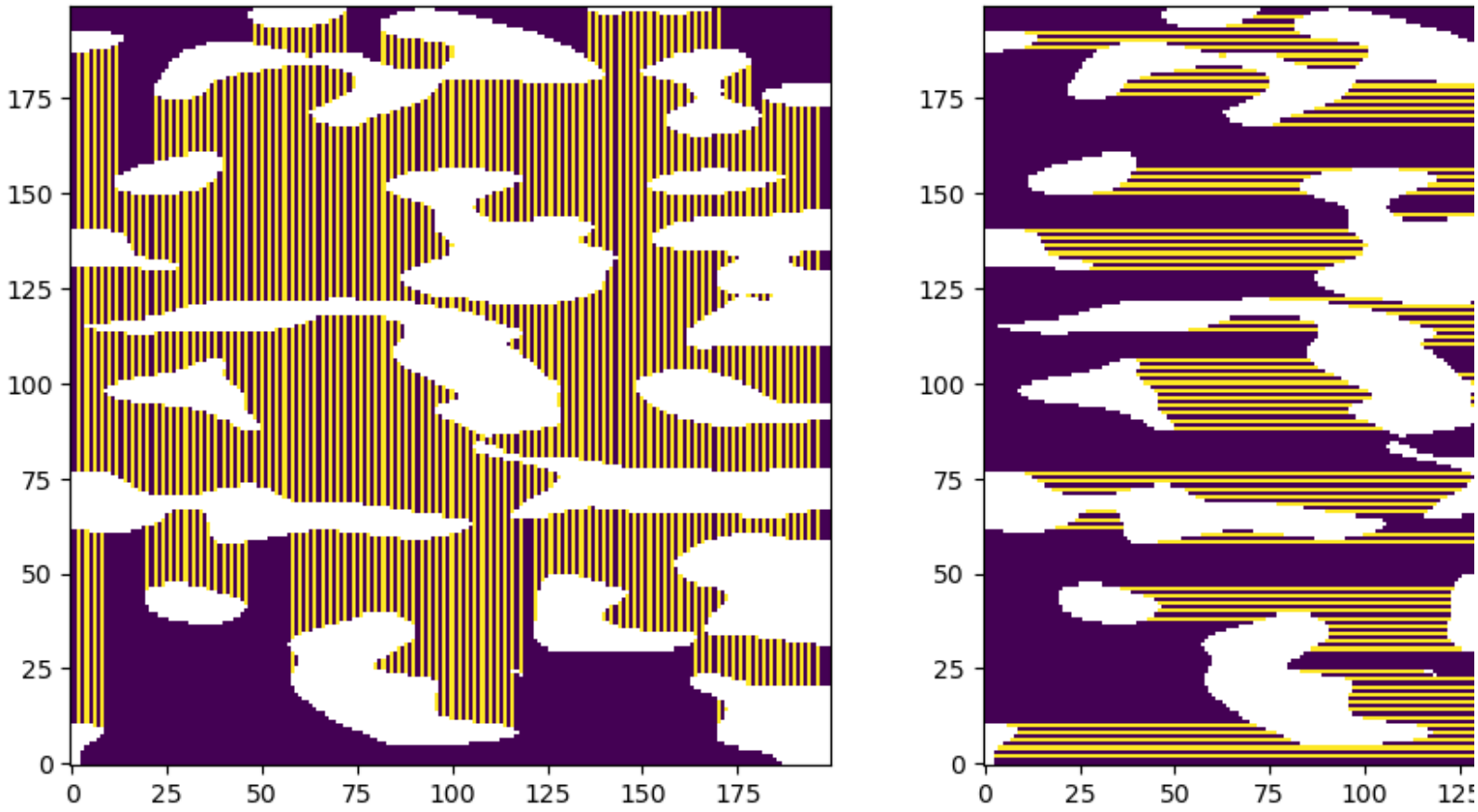
Draw Chords

The `filters` module contains 2 functions for drawning chords:

- `apply_chords` will add chords to the image, all in the same direction
- `apply_chords_3D` draws chords in all three principle directions, which only works for 3D images

In [18]:
```
crds_x = ps.filters.apply_chords(im, axis=0)
crds_y = ps.filters.apply_chords(im, axis=1)
```

In [19]:
```
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(crds_x/im, origin='lower', interpolation='none')
ax[1].imshow(crds_y/im, origin='lower', interpolation='none');
```



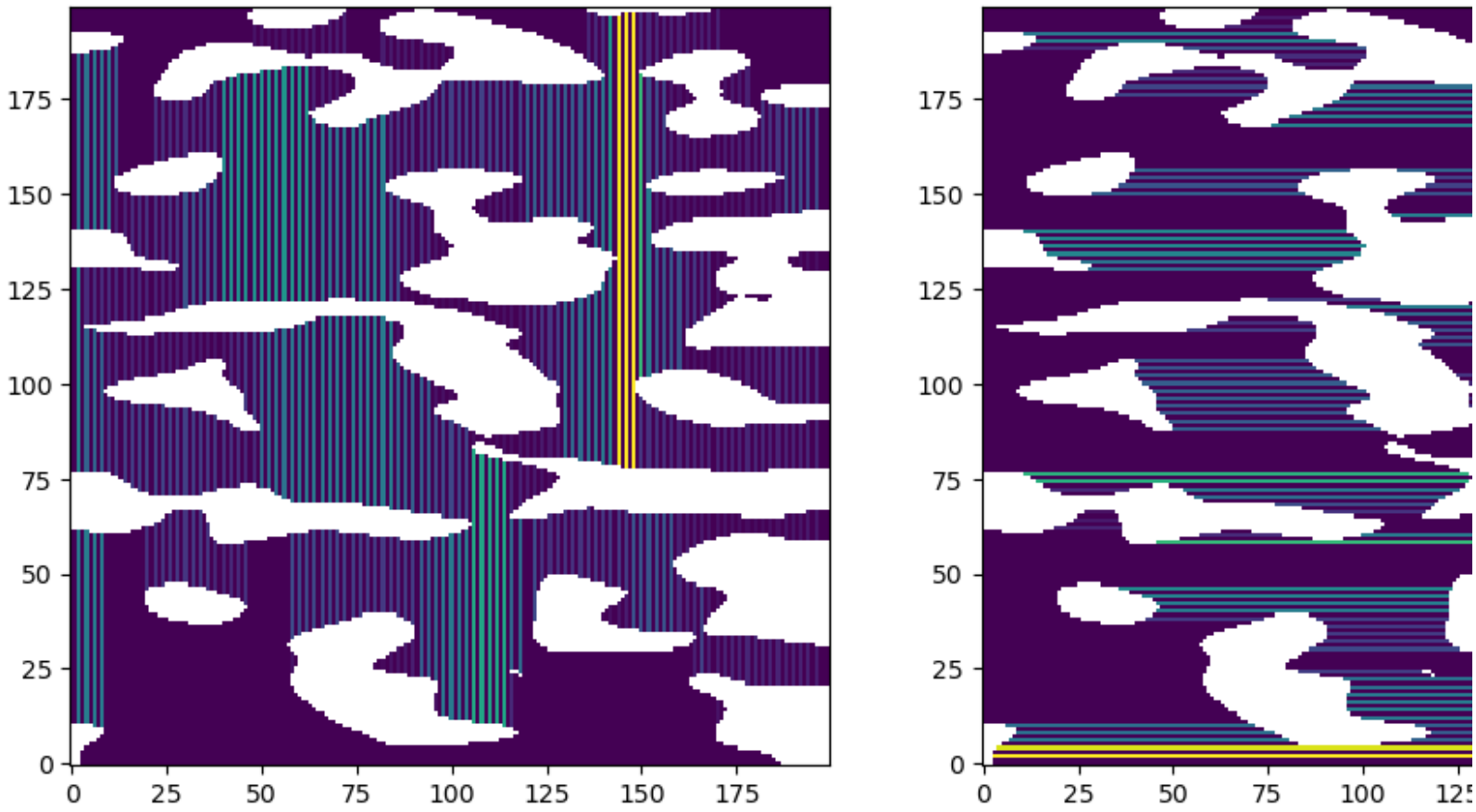
> Note that chords touching the image boundary have been removed since these would be artificial shortened.

Analyze Chord Length

One of the benefits of drawing chords only along the principle axes is their length is equal to the number of voxels, which is easy to cou

In [20]:
```
crds_x = ps.filters.region_size(crds_x)
crds_y = ps.filters.region_size(crds_y)
```

In [21]:
```
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(crds_x/im, origin='lower', interpolation='none')
ax[1].imshow(crds_y/im, origin='lower', interpolation='none');
```



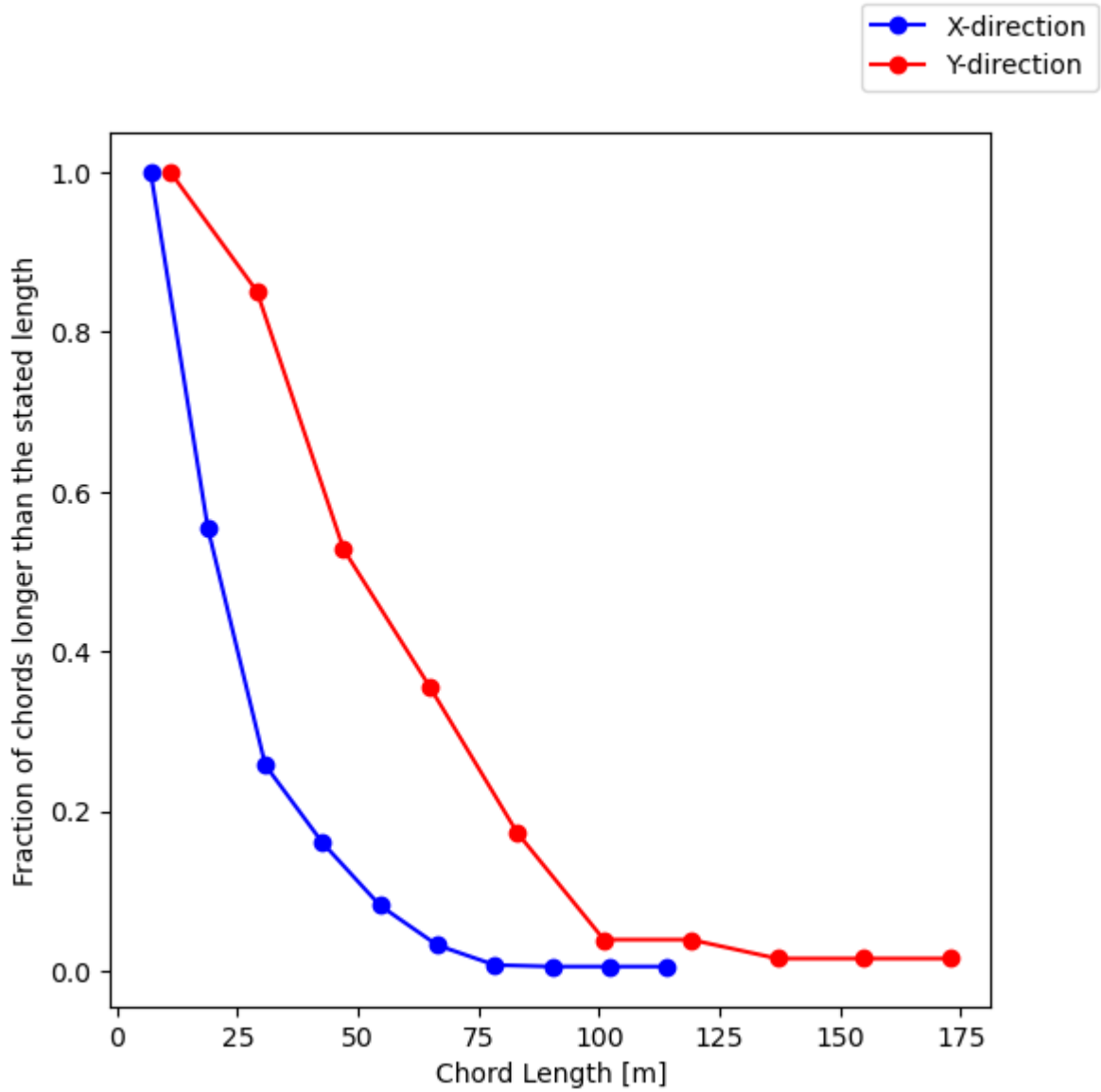
Get Chord Length Distributions

The `metrics` module contains a function for converting the chord lenghts into property statistical distributions:

In [22]:
```
cld_x = ps.metrics.chord_length_distribution(crds_x, bins=10)
cld_y = ps.metrics.chord_length_distribution(crds_y, bins=10)
```

In [23]:

```
fig, ax = plt.subplots(figsize=[6, 6])
ax.plot(cld_x.L, cld_x.cdf, 'b-o', label='X-direction')
ax.plot(cld_y.L, cld_y.cdf, 'r-o', label='Y-direction')
ax.set_xlabel('Chord Length [m]')
ax.set_ylabel('Fraction of chords longer than the stated length')
fig.legend();
```



Survey of Other Available Analysis Tools in PoreSpy

The *porespy.org* website has extensive documentation

- Here is a list (https://porespy.org/_examples/filters/index.html) of functions in the `filters` module
- Here is a list (https://porespy.org/_examples/metrics/index.html) of functions in the `metrics` module

Determining Transport and Capillary Properties from Images

It is becoming increasingly common to perform simulations directly on an image, using the voxels as the computational mesh.

PoreSpy offers some limited functionality in this area:

- Effective diffusivity, aka formation factor or tortuosity
- Drainage simulations, aka mercury intrusion porosimetry
- Invasion percolation

PoreSpy currently does *not* have (but are working on):

- Permeability (Navier-Stokes equations e much harder than the Laplace equation)
- Imbibition

In [17]:
```
import porespy as ps
import matplotlib.pyplot as plt
import numpy as np
```

Tortuosity

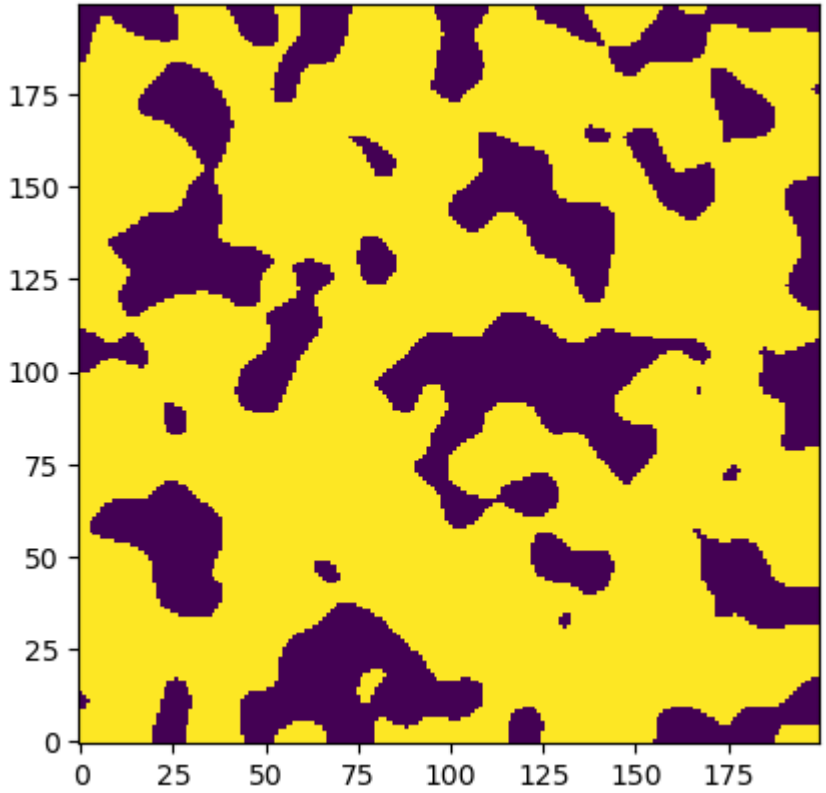
Tortuosity is a fitting factor to account for the reduction of diffusion rate beyond the effect of porosity. It is defined as:

$$\frac{D_{eff}}{D_{bulk}} = \frac{\varepsilon}{\tau}$$

- D_{eff} is some fraction of coefficient in open space (i.e. $D_{eff}/D_{bulk} < 1.0$)
- The reduction in D_{eff} is *at least* equal to the reduced pore volume (i.e. $D_{eff}/D_{bulk} < \varepsilon$)
- But there is an additional reduction since the diffusing molecules are not able to move *directly* across the domain
- This additional resistance is called *tortuosity*

Let's generate a 2D test image

In [18]:
```
np.random.seed(0)
im = ps.generators.blobs([200, 200], porosity=0.7)
plt.imshow(im, origin='lower', interpolation='none');
```



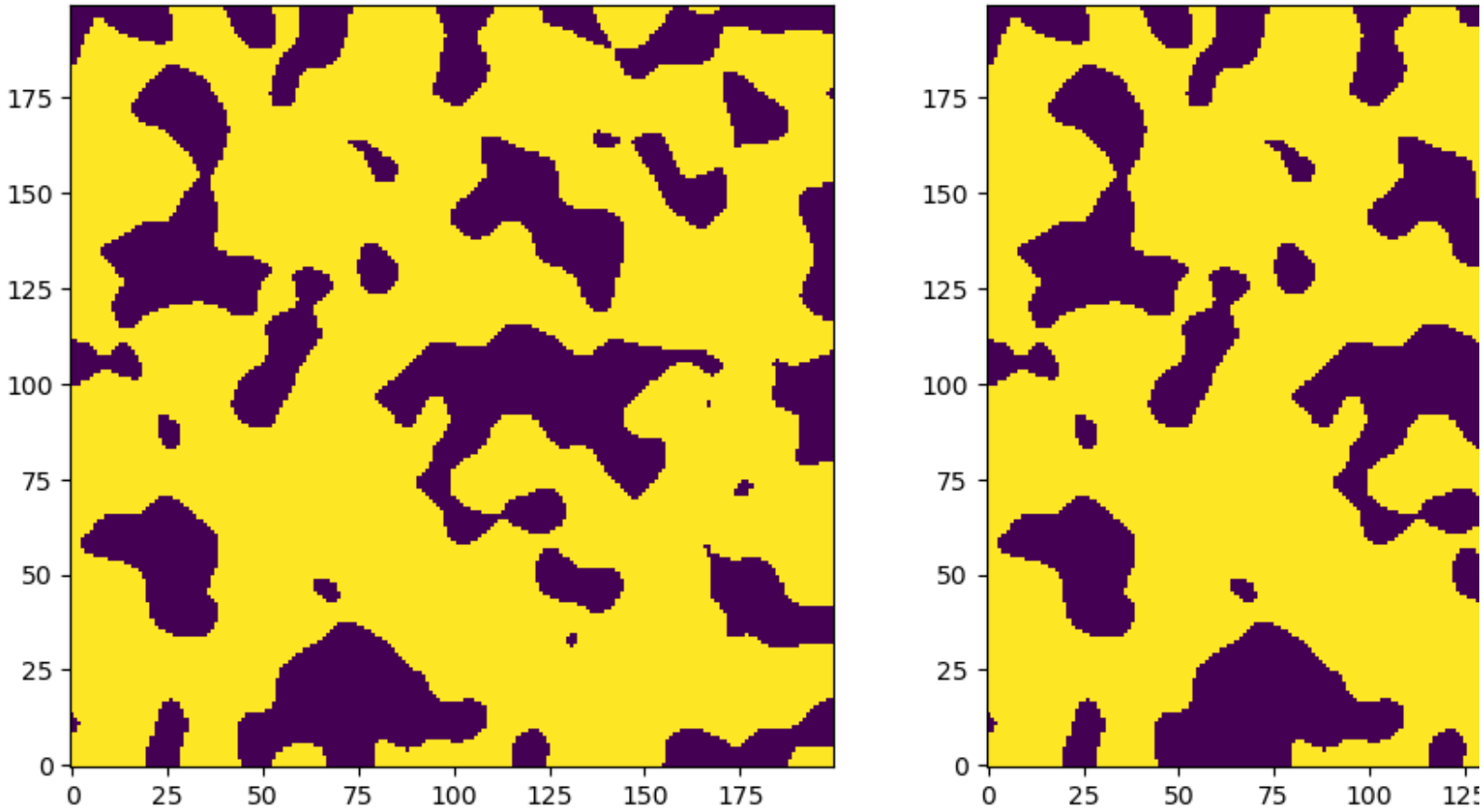
Preparing the Image

A few problems with this image:

- There are several 'blind pores' which will cause problems for the numerical solver
- There are also several pores on the edges which are not connected to the main percolating path

PoreSpy has functions for cleaning this up:

In [19]:
```
im1 = ps.filters.fill_blind_pores(im)
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(im1, origin='lower', interpolation='none')

im2 = ps.filters.fill_blind_pores(im, surface=True)
ax[1].imshow(im2, origin='lower', interpolation='none');
```



Calling the Function

In [20]:
```
tau_x = ps.simulations.tortuosity_fd(im2, axis=0)
tau_y = ps.simulations.tortuosity_fd(im2, axis=1)
```

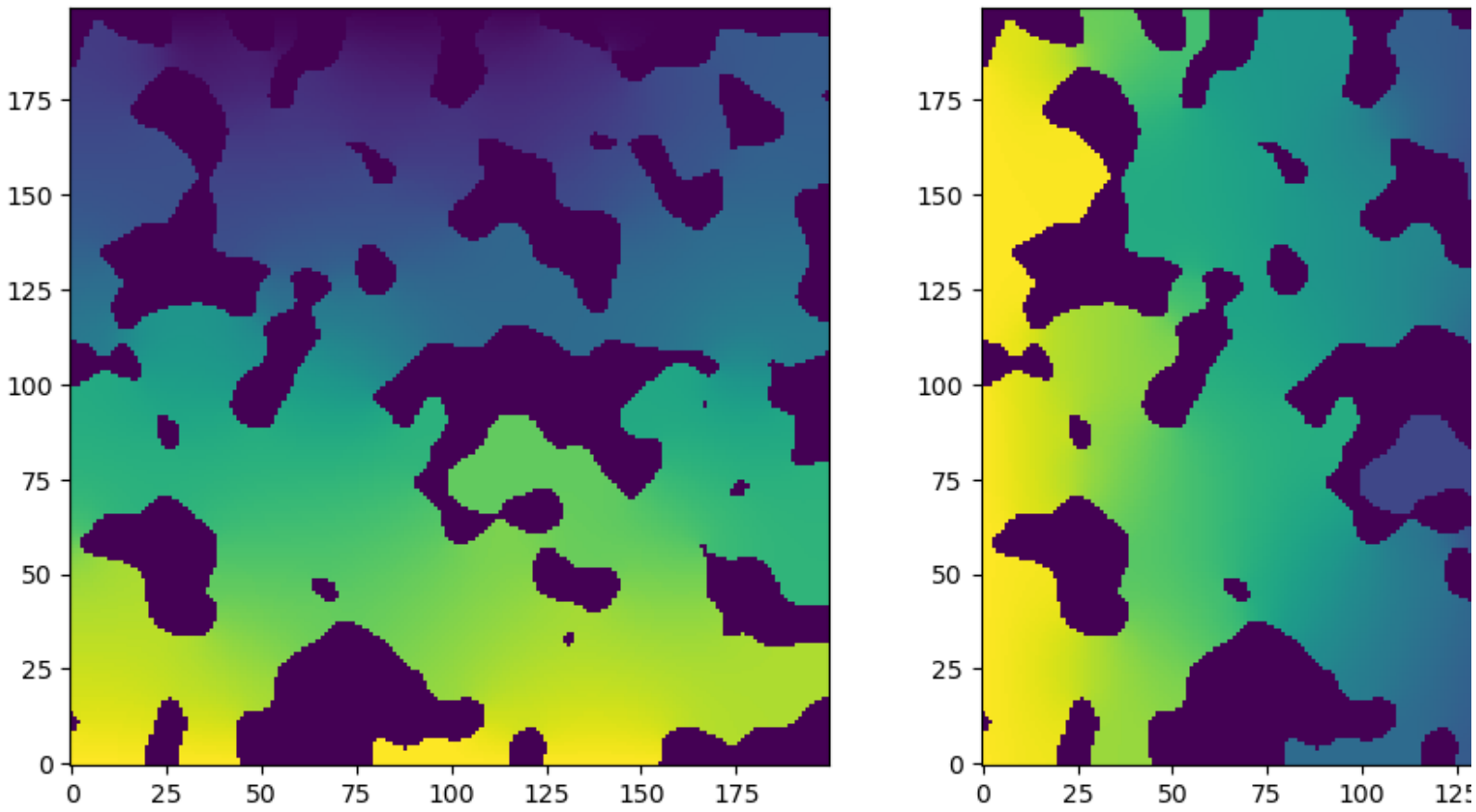
The returned objects contain several pieces of data as attributes, including the calculated tortuosity value, which we can inspect by prir

In [21]:
```
print(tau_x)
```

```
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
Item                        Description
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
im                          Image of size (200, 200)
tortuosity                  1.8569242614864547
formation_factor            2.6298318389554662
original_porosity           0.7061
effective_porosity          0.7061
concentration               Image of size (200, 200)
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
```

And we can inspect the concentration distribution:

In [22]:
```
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(tau_x.concentration, origin='lower', interpolation='none')
ax[1].imshow(tau_y.concentration, origin='lower', interpolation='none');
```



Drainage and Porosimetry

The invasion of a non-wetting fluid can be simulated by drawing spheres with a radius correponding to the applied capillary pressure:

$$R = -2\sigma\frac{cos(\theta)}{P_C}$$

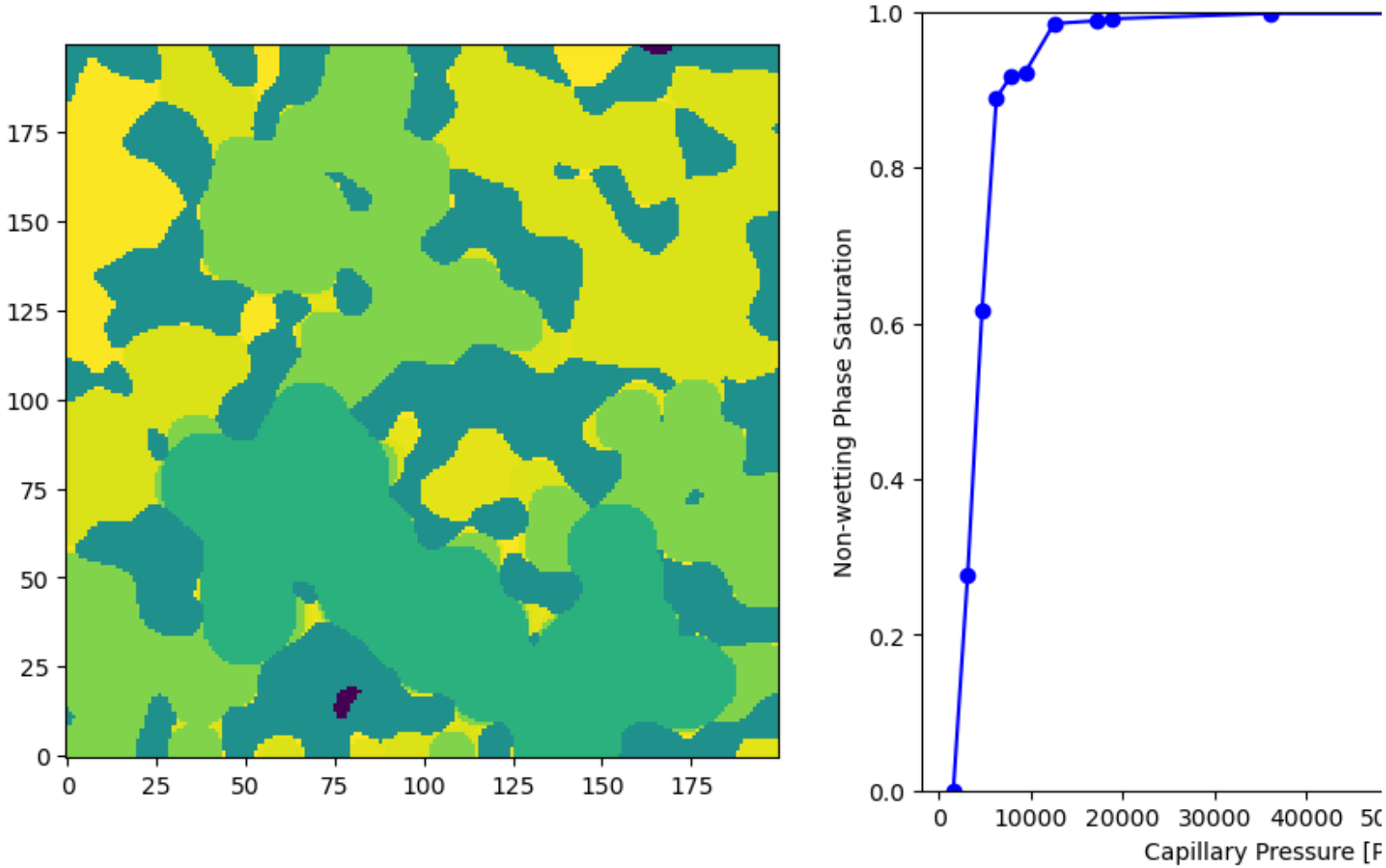
This allows us to insert spheres of size R to determine where the fluid will invaded at a given P_C.

In [23]:
```
mip = ps.simulations.drainage(im, sigma=0.465, theta=140, voxel_size=1e-5)
print(mip)
```

```
  0%|          | 0/25 [00:00<?, ?it/s]

  0%|          | 0/11 [00:00<?, ?it/s]

―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
Item                        Description
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
im_satn                     Image of size (200, 200)
im_pc                       Image of size (200, 200)
im_trapped                  None
pc                          Array of size (11,)
snwp                        [0.0, 0.2760524742376816, 0.6157633735448247, 0.889318749340555, 0.91731438821(
12391, 0.9847360461435656, 0.9884992790067879, 0.9909260366475574, 0.9977490943621848, 0.997749094362184
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
```

And we can plot the results as follows:

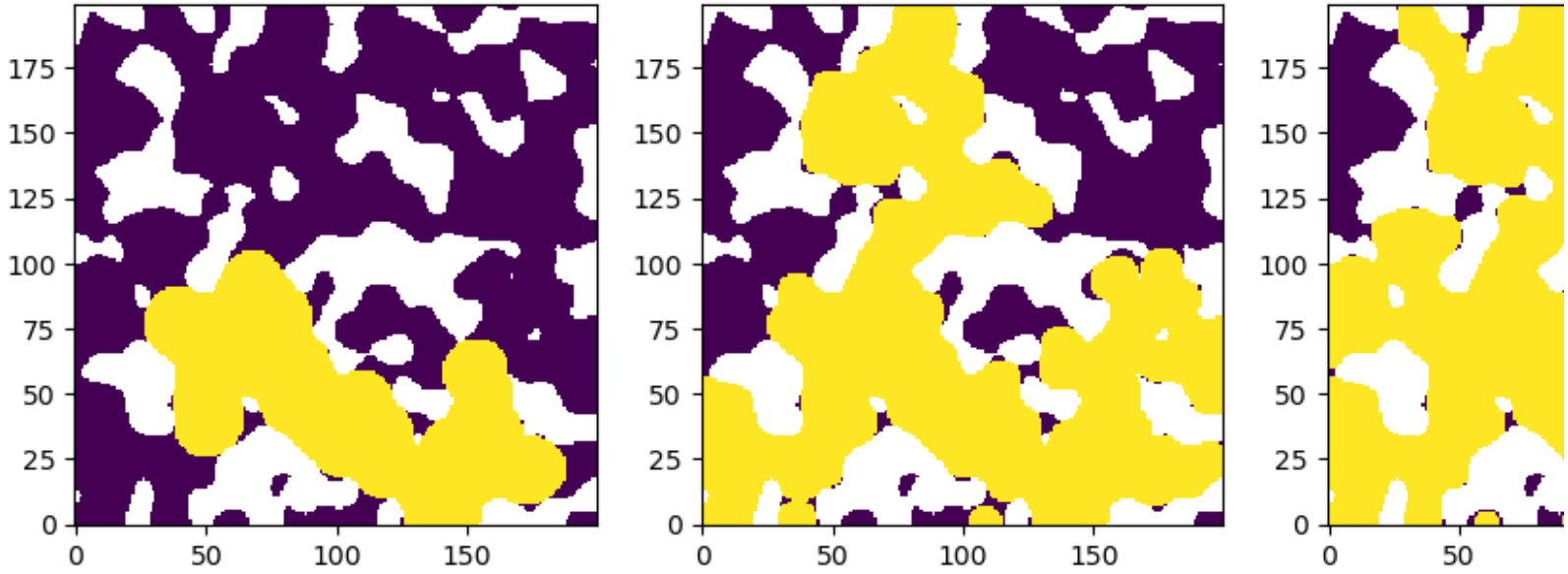
In [24]:
```
fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(mip.im_satn, origin='lower', interpolation='none')
ax[1].plot(mip.pc, mip.snwp, 'b-o')
ax[1].set_ylim([0, 1.0])
ax[1].set_xlabel('Capillary Pressure [Pa]')
ax[1].set_ylabel('Non-wetting Phase Saturation');
```



Obtaining Fluid Configurations

The `im_satn` array returned by the `drainage` function is designed so that all fluid configurations are contained in the same image. T
configuration for a desired saturation can be obtained quickly as follows:

In [25]:
```
s1 = (mip.im_satn < 0.3) * (mip.im_satn >= 0)
s2 = (mip.im_satn < 0.7) * (mip.im_satn >= 0)
s3 = (mip.im_satn < 0.9) * (mip.im_satn >= 0)

fig, ax = plt.subplots(1, 3, figsize=[12, 6])
ax[0].imshow(s1/im, origin='lower', interpolation='none')
ax[1].imshow(s2/im, origin='lower', interpolation='none')
ax[2].imshow(s3/im, origin='lower', interpolation='none');
```

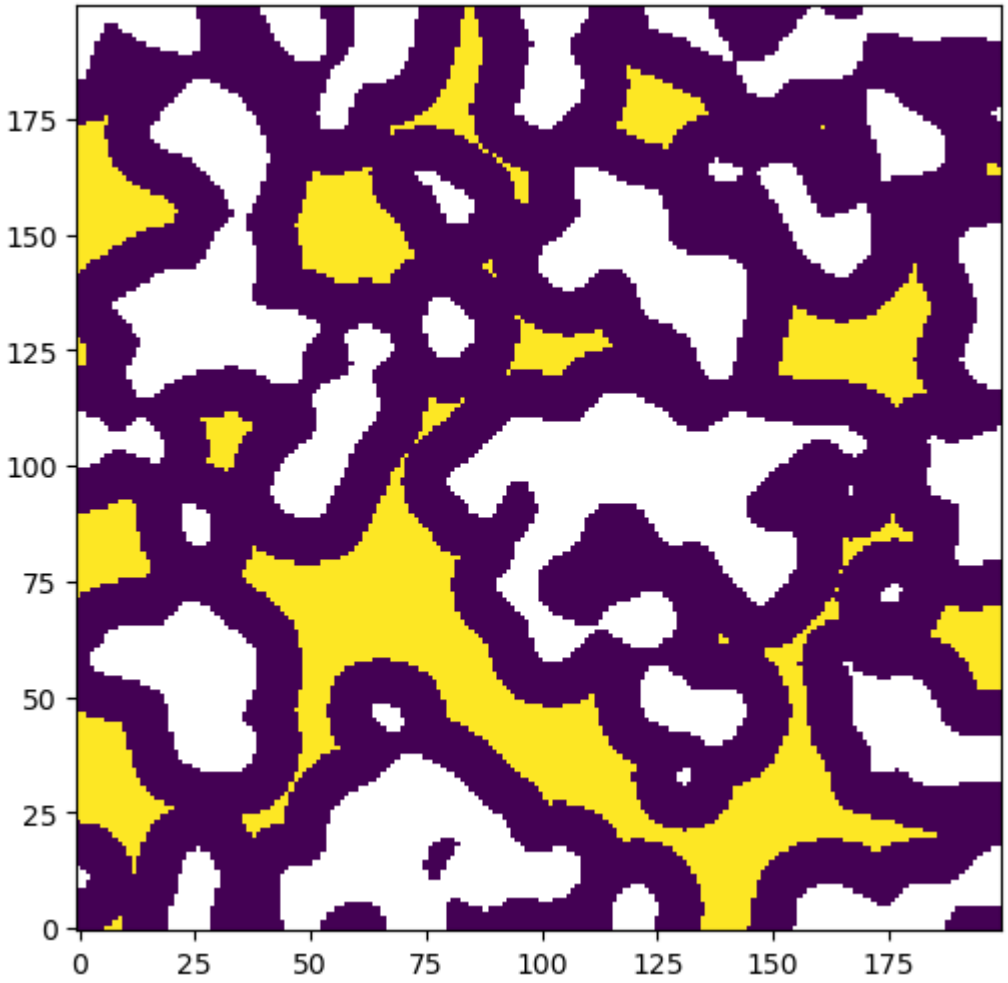


Background Simulating Drainage

Like the local thickness filter, the drainage simulation is done by progressively adding smaller spheres, but there is one difference: any
to the inlet are removed. This is outlined below:

In [26]:

```
from edt import edt
dt = edt(im)
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow((dt > 10)/im, origin='lower', interpolation='none');
```



Each yellow voxel is a location where a sphere of radius 10 *could* be drawn. But before we do that we should trim voxels that are not c

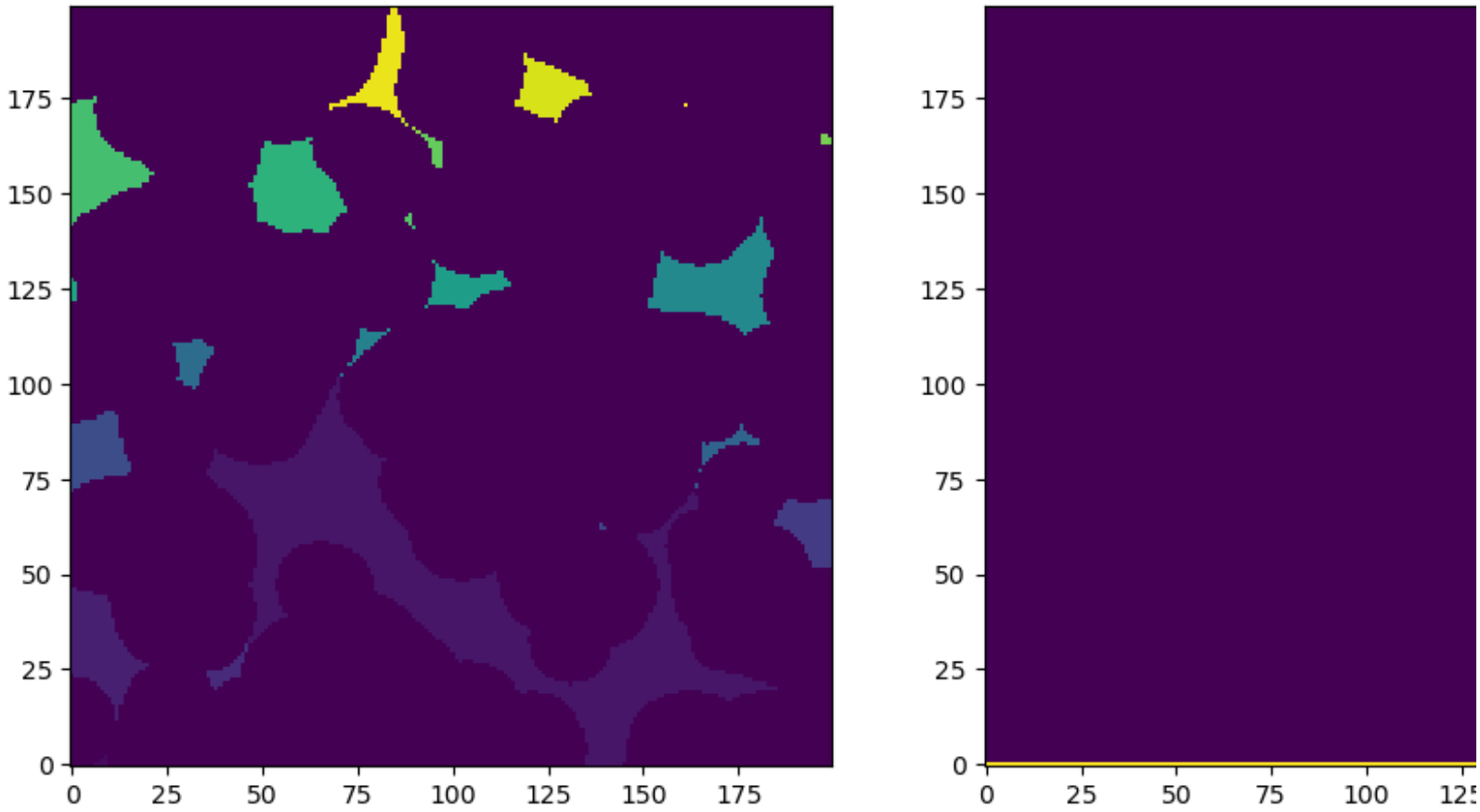
We can do this by:

- Creating an image with `True` values indicating the inlets

- Labeling the clusters
- Removing all voxels whose label does not appear in the inlets

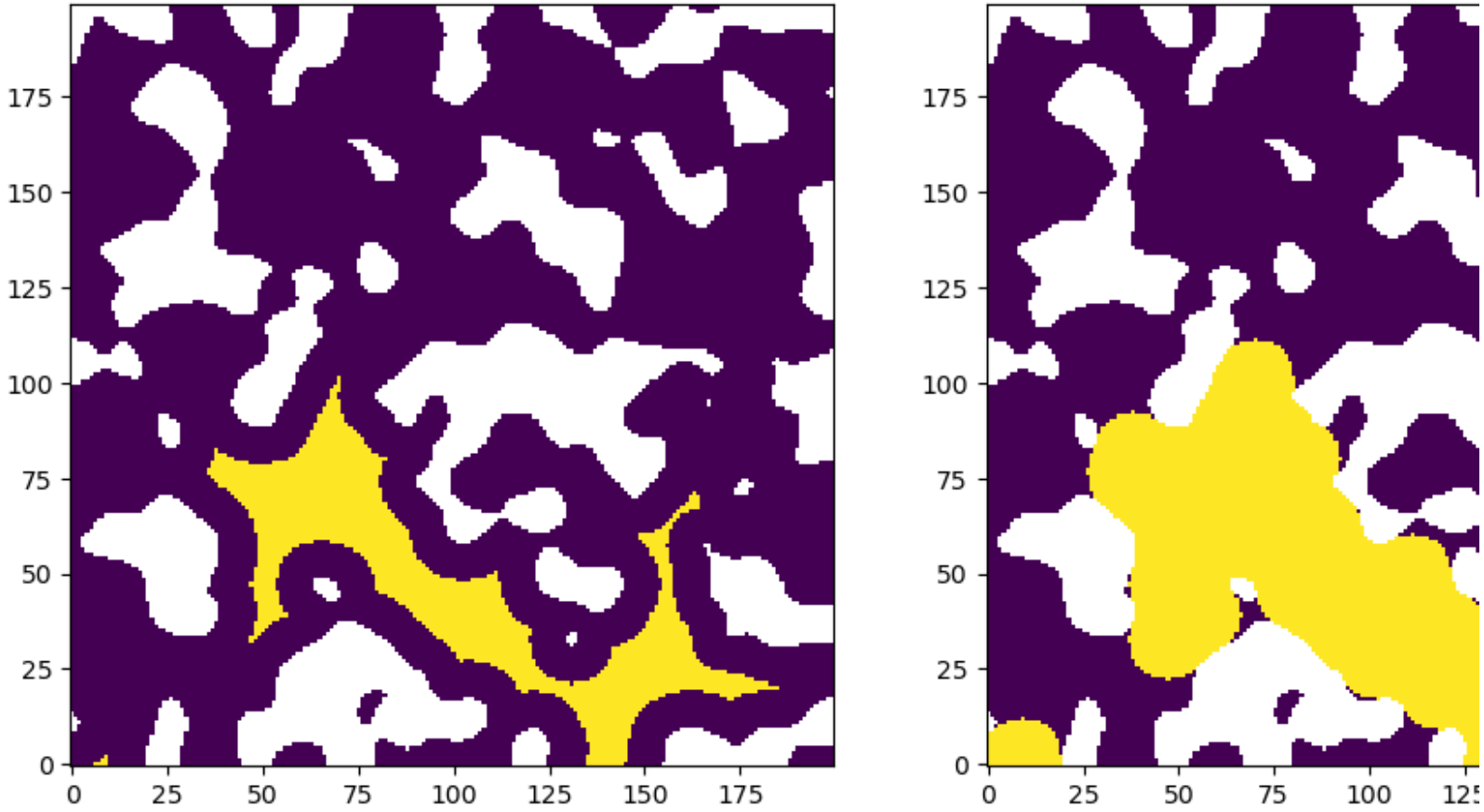
In [29]:
```
import scipy.ndimage as spim
labels, N = spim.label(dt > 10)
inlets = np.zeros_like(im)
inlets[0, :] = True

fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(labels, origin='lower', interpolation='none')
ax[1].imshow(inlets, origin='lower', interpolation='none');
```

Out[29]: <matplotlib.image.AxesImage at 0x20041b0ebe0>



Using a bit of fancy logic and boolean masks, we can drop all voxels whose label number does not appear on the inlet:

In [38]:
```
keep = np.isin(labels, labels[inlets])*(labels > 0)

fig, ax = plt.subplots(1, 2, figsize=[12, 6])
ax[0].imshow(keep/im, origin='lower', interpolation='none')
fluid = spim.binary_dilation(keep, structure=ps.tools.ps_round(10, ndim=2, smooth=False))
ax[1].imshow(fluid/im, origin='lower', interpolation='none');
```



Mulitphase Flow Simulations

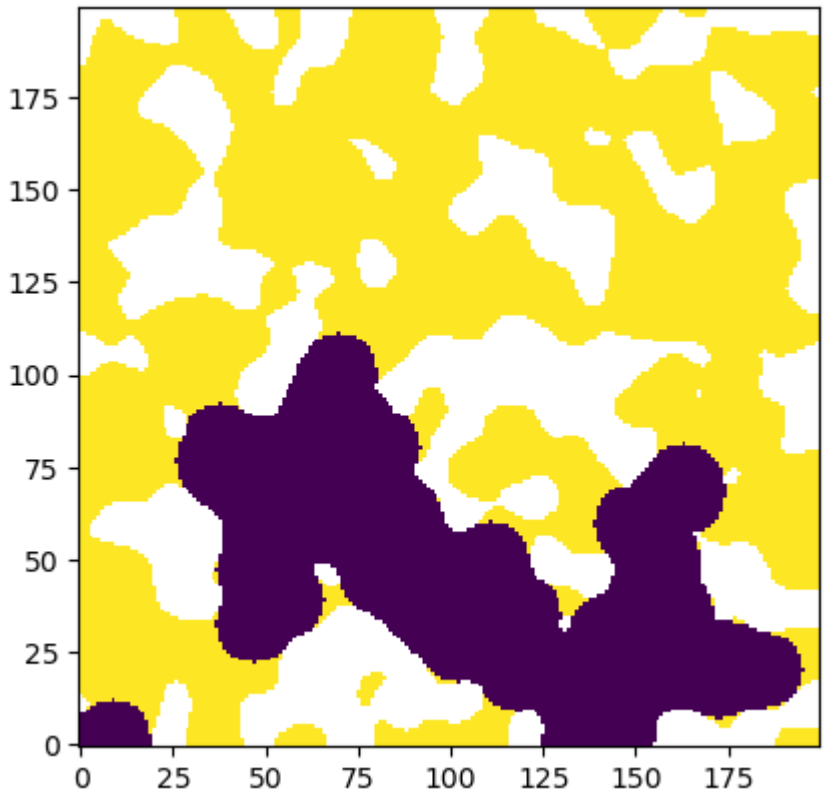
Aside from simulating capillary pressure curves for comparing to experimental data, the `drainage` algorithm can also be used to obta

corresponding to different saturations. This can be used to compute relative effective diffusivity (or resistivity index).

Given the fluid configuration computed above, let's find the effective diffusivity across the domain, perpendicular to the invasion face.

In [41]:
```
im2 = im*(~fluid)
plt.imshow(im2/im, origin='lower', interpolation='none');
```

Out[41]:
```
<matplotlib.image.AxesImage at 0x2003f9b75b0>
```

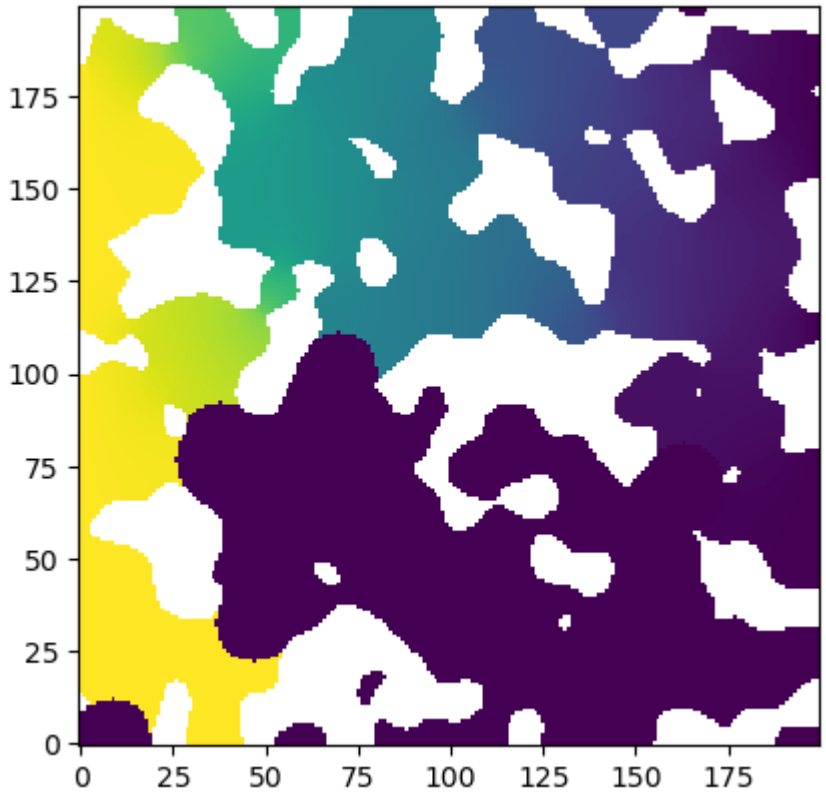


Now lets apply `tortuosity_fd` along the y-axis:

In [44]:
```
deff_y = ps.simulations.tortuosity_fd(im2, axis=1)
plt.imshow(deff_y.concentration/im, origin='lower', interpolation='none');
```

```
[20:28:34] WARNING  Found non-percolating regions, were filled to percolate
```

Out[44]:
```
<matplotlib.image.AxesImage at 0x2003f904d60>
```



Let's print the `deff_y` object to see what the tortuosity was:

In [45]:
```
print(deff_y)
```

```
------------------------------------------------------------------------------
Item                        Description
------------------------------------------------------------------------------
im                          Image of size (200, 200)
tortuosity                  2.828969370086357
formation_factor            6.342269633642769
original_porosity           0.490325
effective_porosity          0.44605
concentration               Image of size (200, 200)
------------------------------------------------------------------------------
```

Now let's check the `tau_y` object to see the tortuosity through the dry image:

In [46]:
```
print(tau_y)
```

```
------------------------------------------------------------------------------
Item                        Description
------------------------------------------------------------------------------
im                          Image of size (200, 200)
tortuosity                  1.8336360644542187
formation_factor            2.596850395771447
original_porosity           0.7061
effective_porosity          0.7061
concentration               Image of size (200, 200)
------------------------------------------------------------------------------
```

So the presence of the invading fluid increase the tortuosity by 50%.

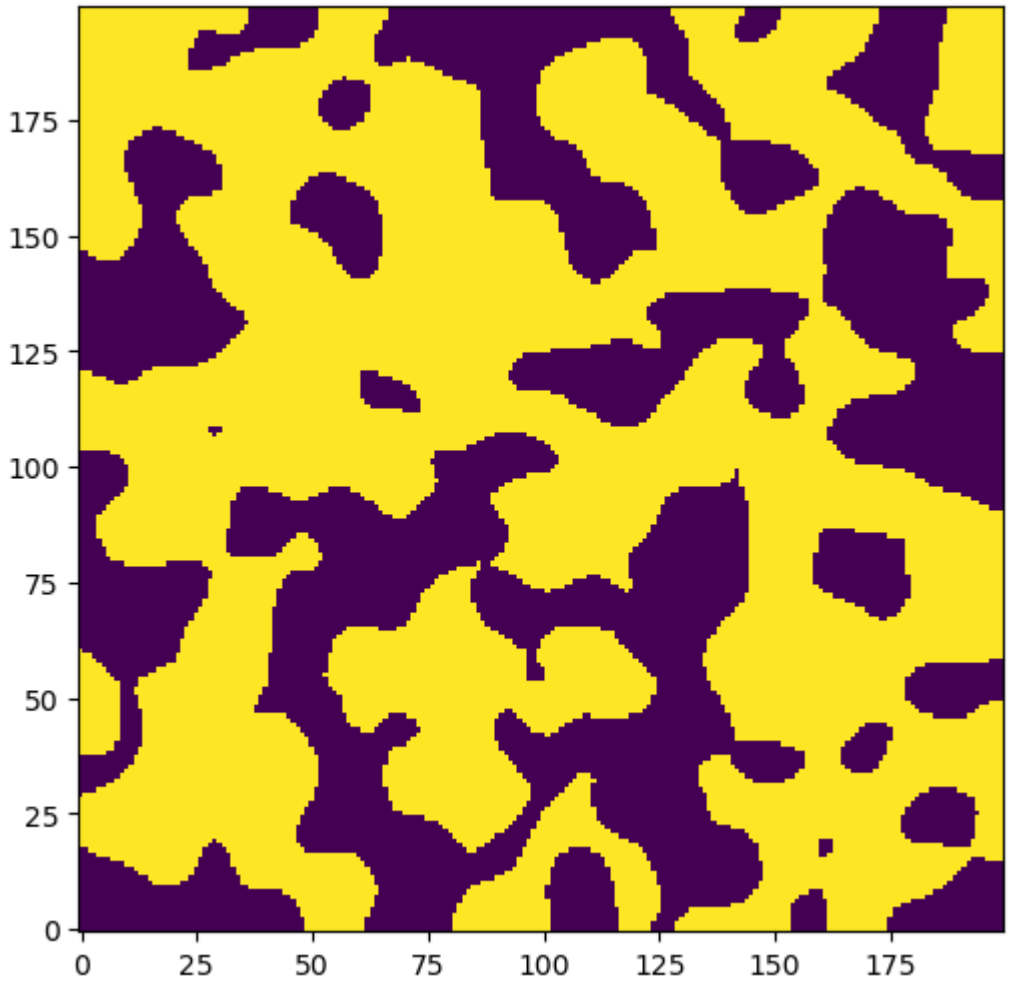
> Note: This 'tortuosity' value is already account for the reduced pore volume due to the presence of invading fluid, so this a real tortuosity, above and beyond the effect of just pore blockage.

Network Extraction

One of the more interesting and powerful tricks that can be done to a volumetric image is extracting a network representing the void sp

In [1]:
```
import porespy as ps
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
im = ps.generators.blobs([200, 200], porosity=0.6)
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow(im, origin='lower', interpolation='none');
```



Applying the SNOW Algorithm

PoreSpy V2 includes the `snow2` function, which is an evolution of the original function with the following features:

- Simple addition of boundary pores

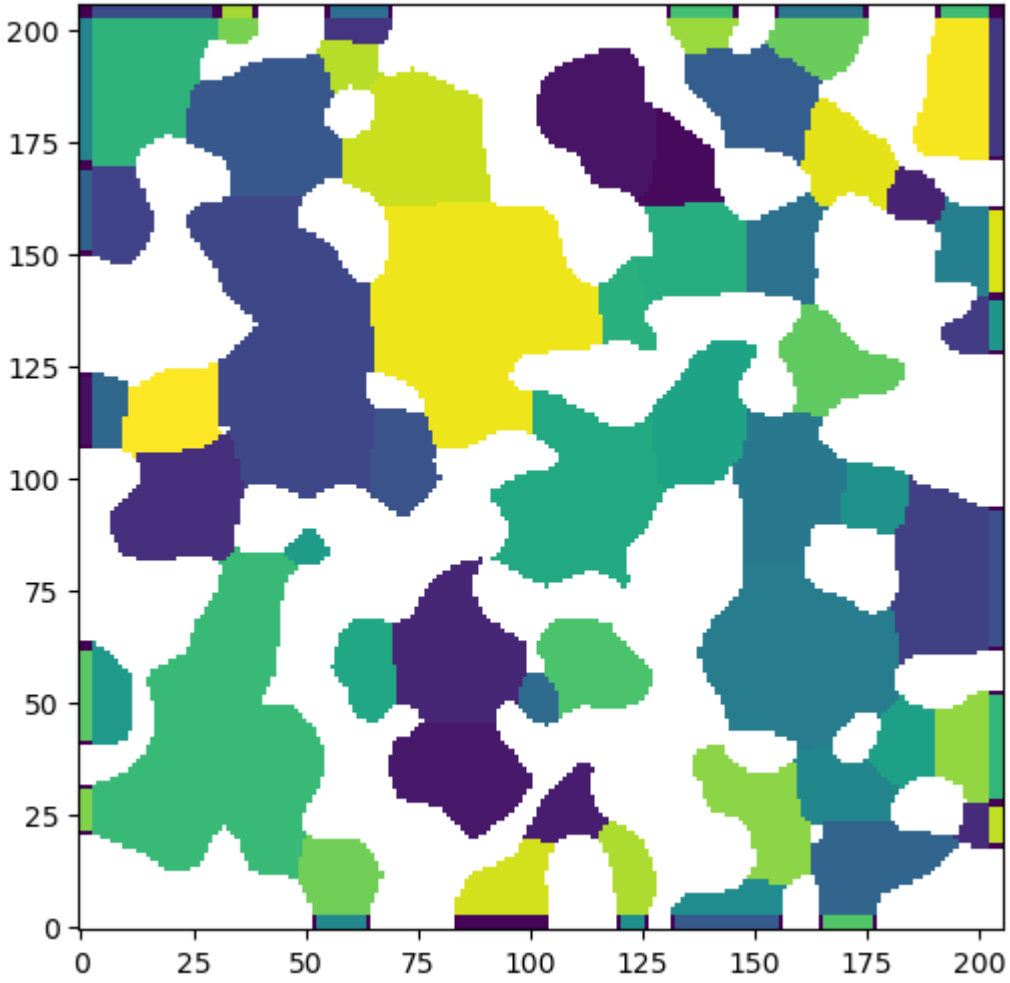
- Unlimited number of phases
- Parallelized watershed step for improved speed

Using the defaults gives:

In [31]:
```
sn = ps.networks.snow2(im)
```

```
0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

Extracting pore and throat properties:   0%|          | 0/74 [00:00<?, ?it/s]
```

In [60]:
```
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow(ps.tools.randomize_colors(sn.regions)/(sn.phases > 0), interpolation='none', origin='lower');
```

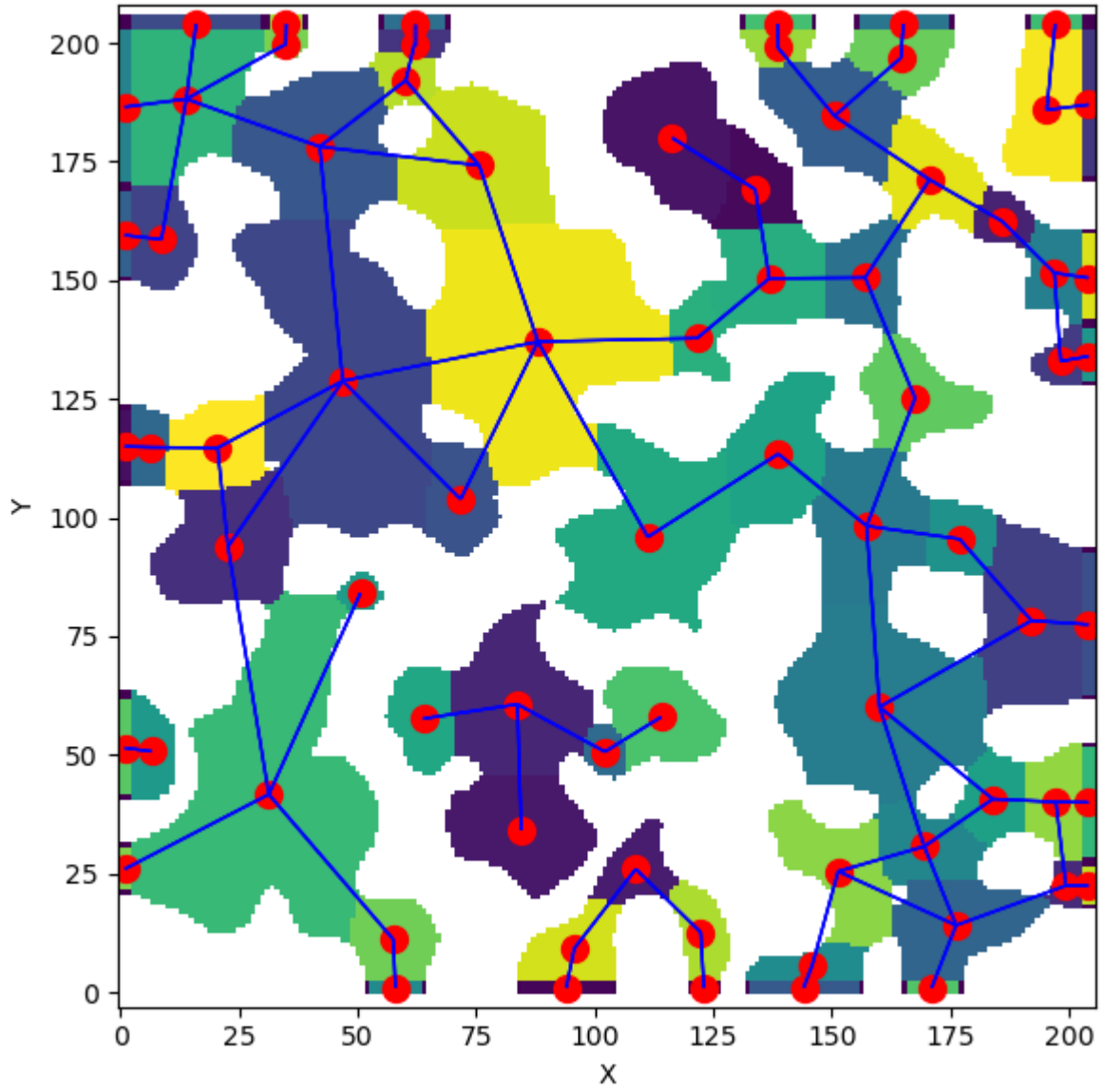


Overlay Image and Pore Network

Without covering pore network modeling too much, it's possible to overlay the extracted network on top of the image

In [63]:
```
import openpnm as op
pn = op.io.network_from_porespy(sn.network)
pn['pore.coords'] = pn.coords[:, [1, 0, 2]]
ax = op.visualization.plot_connections(pn, ax=ax)
ax = op.visualization.plot_coordinates(pn, s=100, ax=ax)
fig
```

Out[63]:



Applying to Multiple Phases

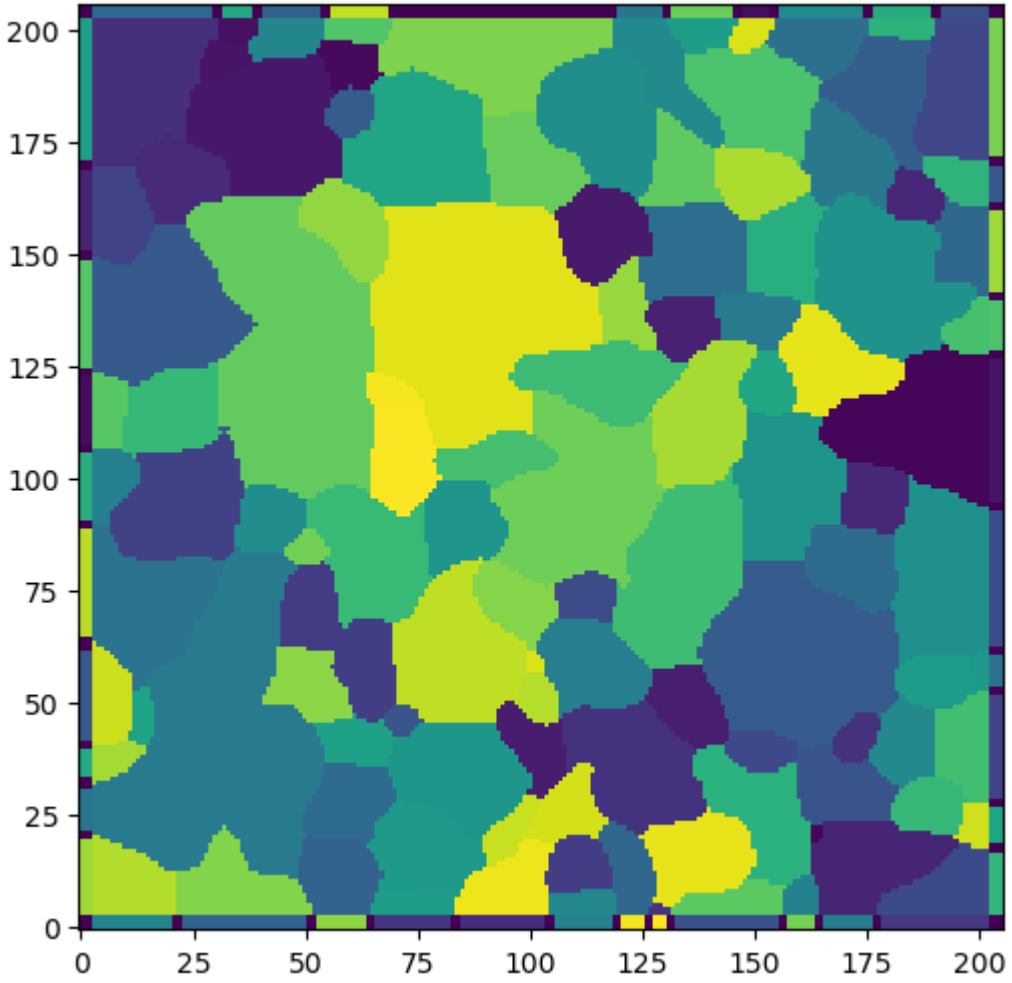
One of the most interesting feature of `snow2` it the ability to do 'multiphase' extractions.

- This is essential for electrochemical devices since solid phase transport is crucial
- Has also been used for 'multiscale' modeling where the 2nd phase is unresolved pore space (i.e. clay) and treated as continuum n

In [64]:
```
sn2 = ps.networks.snow2(phases=im.astype(int)+1)
```

```
0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

Extracting pore and throat properties:   0%|          | 0/155 [00:00<?, ?it/s]
```

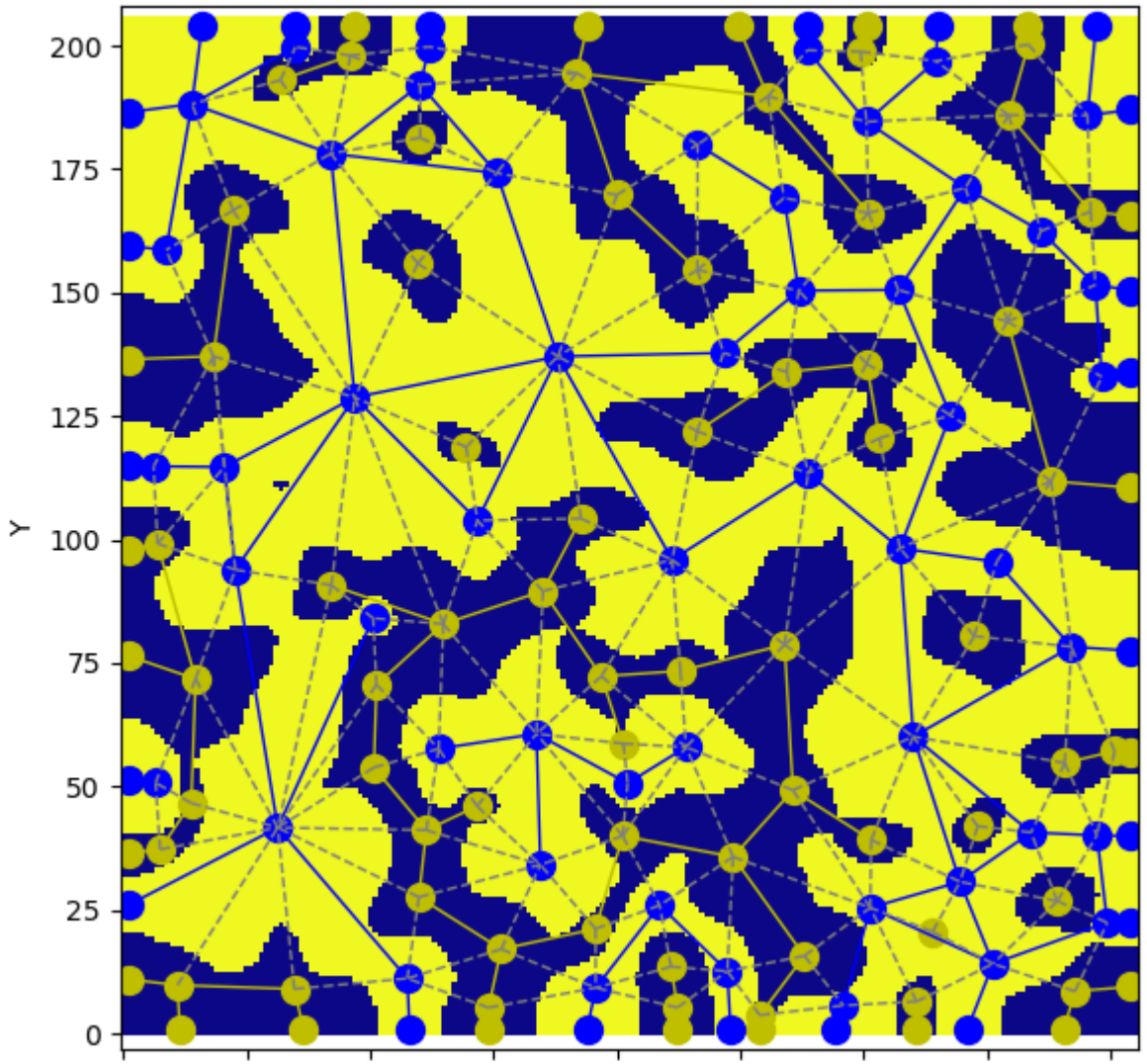
In [72]:
```
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow(ps.tools.randomize_colors(sn2.regions), interpolation='none', origin='lower');
```

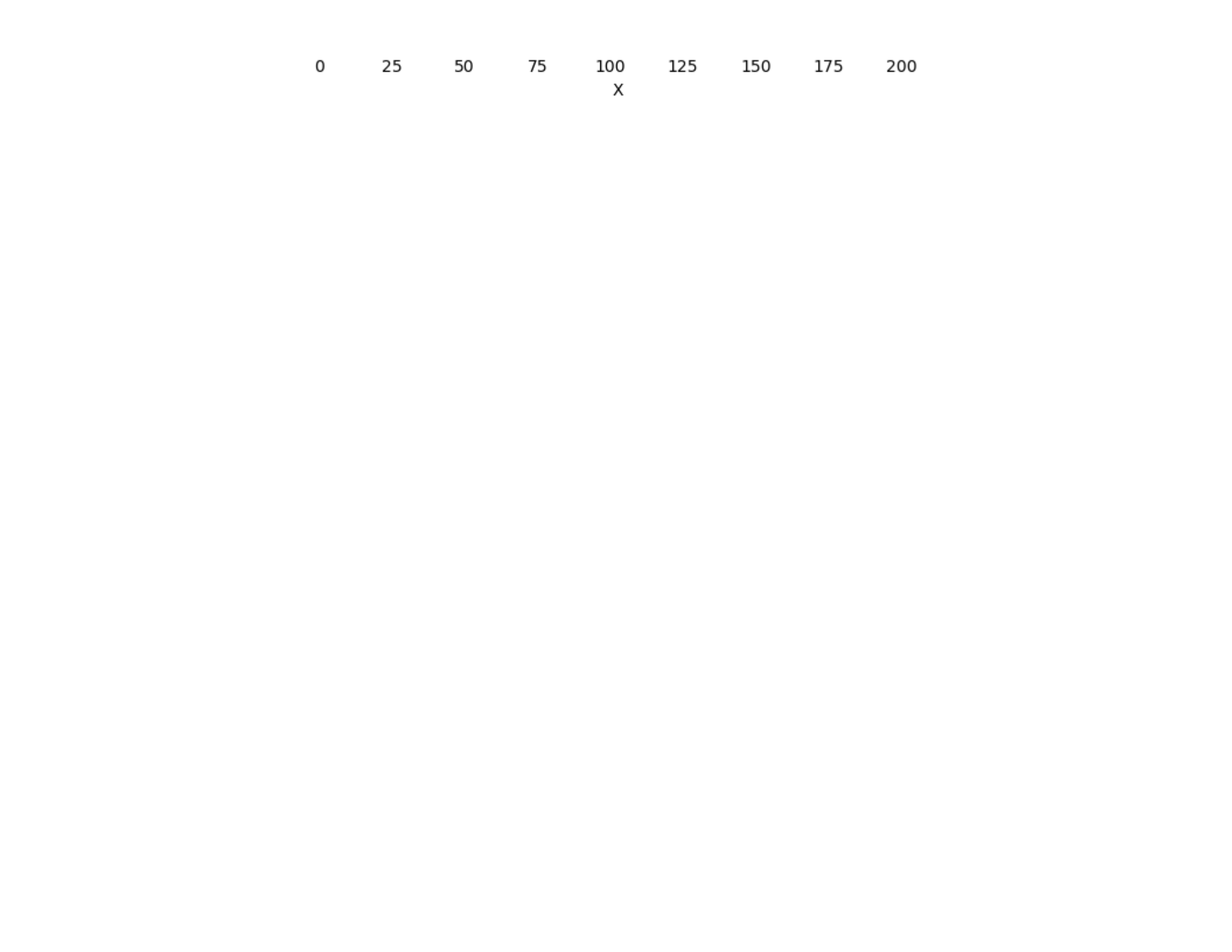


Overlaying Both Networks on the Multiphase Image

In [85]:

```
pn = op.io.network_from_porespy(sn2.network)
pn['pore.coords'] = pn.coords[:, [1, 0, 2]]
fig, ax = plt.subplots(figsize=[6, 6])
ax.imshow(sn2.phases, cmap=plt.cm.plasma, interpolation='none', origin='lower');
ax = op.visualization.plot_connections(pn, throats=pn.throats('phase1_phase1'), c='y', ax=ax)
ax = op.visualization.plot_connections(pn, throats=pn.throats('phase2_phase2'), c='b', ax=ax)
ax = op.visualization.plot_connections(pn, throats=pn.throats('phase1_phase2'), linestyle='--', c='gray'
ax = op.visualization.plot_coordinates(pn, pores=pn.pores('phase1'), c='y', s=100, ax=ax)
ax = op.visualization.plot_coordinates(pn, pores=pn.pores('phase2'), c='b', s=100, ax=ax)
```




0
25
50
75
100
125
150
175
200
X