Contents

Part I
Part Title

Use the template *part.tex* together with the document class SVMono (monograph-type books) or SVMult (edited books) to style your part title page and, if desired, a short introductory text (maximum one page) on its verso page.

Chapter 1

From brain physiology to brain physics

Marie E. Rognes and Kent-Andre Mardal and Lars Magnus Valnes and Vegard Vinje

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1.1 Introduction

In the early 2010s, ground-breaking discoveries at the interface between neurology and neuroscience dramatically increased the interest in and relevance of mathematical modelling of human brain mechanics. By tracking in-vivo movement of tracer

Rognes
Simula Research Laboratory e-mail: meg@simula.no

Mardal
University of Oslo and Simula Research Laboratory e-mail: kent-and@simula.no

Valnes
University of Oslo e-mail: larsmvalnes@gmail.com

Vinje
Simula Research Laboratory and Expert Analytics AS e-mail: vegard@xal.no

molecules in the brains of mice, Xie et al. (2013) observed that molecules cleared rapidly in sleeping mice, but not in awake mice. Six years later, Fultz et al. (2019) revealed intimate connections between waves of blood flow, cerebrospinal fluid (CSF) flow, and neural activity in and around the sleeping brain. These findings introduced the idea that sleep drives molecular clearance from the human brain, that brain clearance is key to why we sleep, and moreover, that brain clearance may be fundamentally associated with the presence of toxic metabolites and neurodegeneration.

In the years since, mathematical modelling and simulation of the physiology and physics of the brain have emerged as key enabling technologies for testing biological hypotheses, quantifying hidden mechanisms and resolving major controversies within these fields. This chapter gives a qualitative and quantitative introduction to fundamental aspects of the solid mechanics and fluid dynamics of the brain and its CSF-filled local environment – at the tissue/organ-scale and at time scales associated with physiological pulsations.

1.2 The pulsating brain

With every heartbeat, blood flows from the body and into the brain via the cerebral arteries[1]. The volume of the (adult human) brain increases by about 1 cm^3, followed by an increase in the intracranial pressure (ICP) of 4–10 mmHg (Balédent, 2014; Wagshul et al., 2011). The resulting pressure difference between the brain and spinal compartments drives the CSF downwards from the subarachnoid space (SAS) surrounding the brain and the ventricles inside the brain, along the spinal canal. Next, the venous blood outflow routes catch up, in turn decreasing the volume of the brain, reducing the ICP and resulting in a flow reversal of CSF from the spinal compartment up into the cranium. For a simulation framework for this 1 Hz fluid-structure interaction phenomenon, see e.g. Causemann et al. (2022).

Fig. 1.1: The pulsating brain

[1] For a quick introduction to brain anatomy, see e.g. Mardal et al. (2022).

The respiratory cycle associated with breathing in (inspiration) and out (exhalation) similarly induces pulsating displacement, pressure and flow patterns at a frequency of around 0.25Hz in resting humans and with pressure variations on the order of 1 mmHg (Vinje et al., 2019). During natural inspiration, the chest cavity volume increases, reducing the intrathoractic pressure (i.e pressure in the chest region), and subsequently facilitating venous blood flow from the brain into the heart. This volume is replaced by CSF flowing up from the spinal compartment. Exhalation reverses the patterns. The picture may however be complicated by more complex mechanisms and different breathing techniques inducing different patterns, and the precise mechanisms remain under debate (Lloyd et al., 2020).

Intriguingly, in addition to the cardiac and respiratory cycles, a number of unique wave patterns are associated with the sleeping brain (Lewis, 2021). Using EEG and fMRI (BOLD), Fultz et al. (2019) observed large but slow (0.05 Hz) oscillations in CSF flow into the ventricles, strongly anti-correlated with BOLD signal oscillations in the cortical gray matter, and in sync with EEG delta waves. Bojarskaite et al. (2023) also directly demonstrate sleep-wake differences in vasomotion and perivascular dynamics.

1.3 Elastic properties of the brain parenchyma

Its composition makes brain tissue rheologically complex, with differing elastic properties under different conditions and time scales. Under physiological conditions, i.e. during the normal pulsations of the brain associated with the cardiac, respiratory or diurnal cycles and the associated time frames of seconds to minutes to hours, the brain is well represented as a linearly elastic or poroelastic medium. However, when undergoing large or sudden deformations, the elastic response of the brain becomes nonlinear and may result in traumatic brain injury (TBI) Griffiths and Budday (2022). It is estimated that in USA 1-4 millions have TBI, resulting from head injuries in sports alone Jordan (2013). In turn, longer time scales (months, years) unveil the viscoelastic features of brain tissue. The excellent reviews by Goriely et al. (2015) and Budday et al. (2020) are good starting points for understanding the nonlinear and viscoelastic properties of the brain, with Su et al. (2023) providing recent insights on the comparison between viscoelastic and poroelastic properties.

In terms of elastic properties, the brain is considered almost surprisingly soft. How do we quantify this? Well, if u is a vector field that defines the brain displacement relative to a baseline configuration, and we model the brain as a linearly elastic medium, then the stress tensor σ is expressed in terms of the strain ε as

$$\sigma(u) = C\varepsilon(u), \quad \varepsilon(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right), \tag{1.1}$$

where C is a fourth-order stiffness tensor. The nerve fiber tracts making up the brain's white matter define a strongly anisotropic structure, but perhaps surprisingly,

the elastic response of the brain is reported to not be notably anisotropic (Budday et al., 2017). Therefore, under the assumption of isotropy, the general relation (1.1) reduces to the isotropic stress-strain relation

$$\sigma(u) = 2\mu\varepsilon(u) + \lambda\nabla \cdot uI, \quad \mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1-2\nu)(1+\nu)}, \tag{1.2}$$

where μ and λ are the Lamé parameters, here also expressed in terms of the more frequency reported Young's modulus E and Poisson ratio ν. Within this setting, the intracranial pressure (ICP) corresponds to the solid pressure:

$$ICP \approx p_s = -\frac{1}{d}\mathrm{tr}(\sigma),$$

where $d = 1, 2, 3$ and tr are the dimension and trace of σ, respectively.

So, how soft is brain tissue? Budday et al. (2015) report a typical Young's modulus of 1.389 kPa in gray matter and 1.895 kPa in white matter in combination with an incompressibility assumption i.e. $\nu = 0.5$. The pia membrane, the inner most membrane surrounding the brain parenchyma, is considered 2–3 orders of magnitude stiffer (Ozawa et al., 2004). These estimates of the brain's elastic moduli (Budday et al., 2015) rely on classical ex-vivo experimental techniques in which e.g. a bovine[2] brain tissue sample is extracted and subjected to indentation tests. A newer, emerging technique for measuring brain elasticity non-invasively is magnetic resonance elastography (MRE) (Murphy et al., 2019). Using MRE, Green et al. (2008) identified viscoelastic moduli in the range 2.5-3.1 kPa, with significantly lower storage modulus in white matter than in grey matter. Using optimal coherence elastography, Gary et al. (2023) found that the (mouse) brain stiffens with age in combination with a decrease in water content.

Fig. 1.2: MR elastography of the brain

[2] cow

1.4 Fluid dynamics in the brain's local environment

Within the cranium, the brain's CSF-filled surroundings include the SAS and the ventricles (Fig. 1.3). The SAS is defined by its outer arachnoid membrane and inner pia membrane, closely follows the surface of the brain, and widens out into larger cisterns especially around the cerebellum . The SAS is typically 1–M mm wide with a volume of 125 mL in adult humans (Sakka et al., 2011). The four ventricles include two lateral ventricles, the third and fourth ventricle, with a total volume of ∼25 mL. The ventricles are connected with each other and the subarachnoid space, with the ∼15 mm wide, cerebral aqueduct connecting the third and fourth ventricle.

Fig. 1.3: The brain and its fluid surroundings

CSF is a clear, colorless fluid that is composed of 99% water, electrolyte ions including sodium (Na^+), chloride (Cl^-), potassium (K^+), magnesium (Mg^{2+}), and calcium (Ca^{2+}), as well as glucose and plasma proteins, with an osmolarity of ∼ 290 mOsm/L, and a pH of 7.33. It is widely thought to be mainly produced in the choroid plexus lining the cerebral ventricles at a rate resulting in total CSF renewal around every six hours in humans (Wichmann et al., 2022; MacAulay et al., 2022). The outflux routes of CSF are still under debate and may vary between species, see e.g. Hornkjøl et al. (2022).

Clinically, CSF flow can be measured in the ventricles, cerebral aqueduct or cranio-cervical junction using phase-contrast MRI or via other techniques. Typical flow speeds are 3–5 cm/s in the third ventricle (Dreha-Kulaczewski et al., 2015), 4–6 cm/s in the more narrow cerebral aqueduct and 1–3 cm/s at the cranio-cervical junction (Eide et al., 2021).

Within the SAS, the flow of CSF is well represented by the incompressible Navier–Stokes equations for the velocity v and fluid (or intracranial) pressure p

$$\rho v_t + v \cdot \nabla v + \nabla \cdot (\mu \varepsilon(u) - pI) = 0, \tag{1.3a}$$

$$\nabla \cdot v = 0, \tag{1.3b}$$

with dynamic viscosity $\mu \approx 0.697 \times 10^{-3}$ Pa s, and density $\rho \approx 10^3 kg/m^3$ at body temperature. Often, the low Reynolds and low Womersley number approximation may be highly appropriate, in which case the nonlinear Navier–Stokes equations above reduce to the linear Stokes equations by ignoring the term $v \cdot \nabla v$.

1.5 The brain's fluid networks and permeability

In addition to its fluid surroundings, the brain parenchyma itself is 80% water and permeated by a number of fluid networks. The extracellular space between the brain cells is largely contiguous and filled with interstitial fluid (ISF), a fluid with similar composition as CSF. The vasculature, filled with blood and separated from the brain tissue by the blood-brain barrier, defines one or several additional fluid networks. In addition, perivascular spaces are hypothesized to enclose the cerebral vasculature, within the subarachnoid space and within the parenchyma. The properties of these spaces, including their structure and function, their shape and sizes, and extent of penetration into the brain remain debated (Bakker et al., 2016; Kelley et al., 2022).

Fig. 1.4: The fluid networks of the brain. Illustration of vascular and perivascular spaces (images courtesy of Enger et al.) and extracellular spaces (Show Holter et al mesh).

Interestingly, the brain parenchyma (but not the brain meninges) lacks lymphatic vessels Aspelund et al. (2015); Louveau et al. (2015). The brain's alternative pathways for solute influx and clearance, termed the brain's *glymphatics*, is an active topic of research Kelley et al. (2022). The glymphatic concept was introduced by Iliff et al. (2012) in 2012, extending upon on previous findings on perivascular pathways (Rennels et al., 1985; Hadaczek et al., 2006) and bulk interstitial fluid flow (Cserr and Ostrach, 1974; Abbott, 2004) dating back to the 1970s. The original glymphatic theory stipulates that (i) CSF flows in via periarterial spaces, mixes with ISF and flows through the brain interstitial space, and is cleared via perivenous spaces, (ii) that the

interstitial bulk flow depends on transmembrane water movement in astrocytes, and (iii) that the fluid flow is driven by cerebral pulsations.

To explicitly model this setting: both the elastic response of the brain as well as the fluid motion within the brain induced by physiological pulsations, TULLY and VENTIKOS (2011) proposed applying multiple-network poroelastic theory. For a given set of networks $1, 2, \ldots, I$, the multiple-network poroelasticity equations describe the brain displacement u and pressure p_i in each fluid network i via the governing equations

$$\rho u_t - \nabla \cdot (\sigma(u) - \sum_{i=1}^{I} \alpha_i p_i I) = f \tag{1.4a}$$

$$c_i p_{i,t} + \alpha_i \operatorname{div} u - \nabla \cdot K_i \nabla p_i + T(p) = g_i, \quad \text{i} = 1, 2, \ldots, \text{I}, \tag{1.4b}$$

where $\rho \approx 1.046 \times 10^3$ kg/m^3 is the density of the brain tissue, σ is the elastic stress tensor cf. (1.1), α_i is a Biot-Willis coefficent, c_i a specific storage coefficient, and K_i a hydraulic conductance – the latter three for each network i. In the case $I = 1$, (1.4) reduce to Biot's equations. Both the single-network and multiple-network frameworks have emerged as interesting modelling approaches, see e.g. Causemann et al. (2022). The permeability of the extracellular space in neuropil is relatively low, with estimates ranging from on the order of 10 nm^2 (Holter et al., 2017) to 2500 nm^2 (Smith and Humphrey, 2007). Less is known about the specific storage coefficient and Biot-Willis coefficient of the extracellular space, perivascular spaces and vasculature.

1.6 Fluid mechanics and osmotics in the brain at the microscale

At the microscale, in the extracellular space and within and between brain cells such as e.g. in astrocyte networks, brain fluid mechanics and electrophysiology become intrinsically intertwined. The electrical activity of brain nerve cells rely on the rapid movement of ions such as sodium, potassium and chloride across the neuronal membrane, setting up an electrical potential gradient between the extracellular and intracellular spaces. However, the presence of an ionic gradient also induces an osmotic gradient, leading to water movement across the cellular membranes, cellular swelling and potentially water movement within extracellular and intracellular compartments. Understanding and modelling the electro-chemo-mechanical interplay within the brain is an emerging research field, and refer the interested reader to e.g. (Rasmussen et al., 2020) and (Sætra et al., 2023) within as recent points of departure for the neurophysiology and computational modelling, respectively.

1.7 Mechanics in brain pathologies

- Neurological and neurodegenerative disorders. Several neurological and neurodegenerative disorders are associated with altered CSF dynamics and/or changes in mechanical properties of the brain. Idiopathic normal pressure hydropechalus (iNPH) is characerized by an increase in ventricular volume and patients often suffer from gait ataxia, urinary incontinence, and dementiaReeves et al. (2020). The pathophysiology of iNPH is controversial, but is believed to be linked to changes in CSF dynamics such as CSF production or absorptionReeves et al. (2020). Similary, patients with Alzheimer's disease show alterd CSF dynamics Schubert et al. (2019), which may be linked to changes in removal of interstitial Aβ from the human brain. Although controversial, Aβ plaques are a key hallmark in Alzheimer's disease, however the degree of cognitive decline does not correlate with the number of interstitial Aβ plaques, but rather the build-up of Aβ in vascular spaces Thal et al. (2008). Such an accumulation increases arterial stiffness and reduces arterial pulsatiliy Hughes et al. (2013) crucial for CSF flow and exchange along perivascular spaces Iliff et al. (2012). In general, brain stiffness is sensitive to demyelination as encountered e.g. in multiple sclerosis (MS, decreased stiffness), dementia (decreased stiffness in Alzheimer's disease), tumours Murphy et al. (2019).
- Cancer: The blood-brain barrier limits transport of toxic substances from the blood into the brain Hammarlund-Udenaes et al. (2014). In patients with glioma, this barrier is partially disrupted, resulting in at least three separate fluid compartments where substances may exchange (blood, ISF and CSF). Furthermore it has been shown that perivascular spaces are an important invasion pathway for glioma cells Cuddapah et al. (2014), highlighting the potential role of the glymphatic system for the development of the disease.
- Stroke: Finally, glymphatic dysfunction has been associated with the pathophysiology of brain edema, blood-brain barrier disruption, neruoinflammation, and neuronal cell death after strokeRasmussen et al. (2018).

1.8 Outlook and further reading

Understanding the human brain is arguably one of the larger scientific challenges of our century. Inspired by Hilbert's mathematics program, involving 23 unsolved problems, several has listed a number of challenges Adolphs (2015); Van Hemmen and Sejnowski (2005). It is stricking, however, that most of the challenges listed are related to the computational and cognitive aspects of the brain, specifically how neurons work, while the basic mechanics of the brain's unique physiology have received comparably less attention. The introduction of the glymphatic system Iliff et al. (2012), its relationship to sleep Xie et al. (2013), and its relationship to various neurological diseases Rasmussen et al. (2018) have made biomechanics, in terms of

the fields of fluid dynamics and soft matter physics, blossom with increasing research activity year by year.

Some perspectives from Alzheimer failure Asher and Priefer (2022) and the human brain project Amunts et al. (2016).

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.. Return to the sleep.

Acknowledgements We are very grateful to all the authors and reviewers contributing to this manuscript, as well as to Springer for giving us a platform to distribute these resources including Mardal et al. (2022).

References

Abbott N (2004) Evidence for bulk flow of brain interstitial fluid: significance for physiology and pathology. Neurochemistry International 45(4):545–552, doi:10.1016/j.neuint.2003.11.006, role of Non-synaptic Communication in Information Processing

Adolphs R (2015) The unsolved problems of neuroscience. Trends in Cognitive Sciences 19(4):173–175, doi:10.1016/j.tics.2015.01.007

Amunts K, Ebell C, Muller J, Telefont M, Knoll A, Lippert T (2016) The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain. Neuron 92(3):574–581, doi:10.1016/j.neuron.2016.10.046

Asher S, Priefer R (2022) Alzheimer's disease failed clinical trials. Life Sciences 306:120861, doi:10.1016/j.lfs.2022.120861

Aspelund A, Antila S, Proulx ST, Karlsen TV, Karaman S, Detmar M, Wiig H, Alitalo K (2015) A dural lymphatic vascular system that drains brain interstitial fluid and macromolecules. Journal of Experimental Medicine 212(7):991–999, doi:10.1084/jem.20142290

Bakker ENTP, Bacskai BJ, Arbel-Ornath M, Aldea R, Bedussi B, Morris AWJ, Weller RO, Carare RO (2016) Lymphatic clearance of the brain: Perivascular, paravascular and signifi-

cance for neurodegenerative diseases. Cellular and Molecular Neurobiology 36(2):181–194, doi:10.1007/s10571-015-0273-8

Balédent O (2014) Imaging of the cerebrospinal. Adult hydrocephalus 256(128to512):121

Bojarskaite L, Vallet A, Bjørnstad DM, Gullestad Binder KM, Cunen C, Heuser K, Kuchta M, Mardal KA, Enger R (2023) Sleep cycle-dependent vascular dynamics in male mice and the predicted effects on perivascular cerebrospinal fluid flow and solute transport. Nature Communications 14(1):953, doi:10.1038/s41467-023-36643-5

Budday S, Nay R, de Rooij R, Steinmann P, Wyrobek T, Ovaert TC, Kuhl E (2015) Mechanical properties of gray and white matter brain tissue by indentation. Journal of the Mechanical Behavior of Biomedical Materials 46:318–330, doi:10.1016/j.jmbbm.2015.02.024

Budday S, Sommer G, Birkl C, Langkammer C, Haybaeck J, Kohnert J, Bauer M, Paulsen F, Steinmann P, Kuhl E, Holzapfel G (2017) Mechanical characterization of human brain tissue. Acta Biomaterialia 48:319–340, doi:10.1016/j.actbio.2016.10.036

Budday S, Ovaert TC, Holzapfel GA, Steinmann P, Kuhl E (2020) Fifty Shades of Brain: A Review on the Mechanical Testing and Modeling of Brain Tissue. Archives of Computational Methods in Engineering 27(4):1187–1230, doi:10.1007/s11831-019-09352-w

Causemann M, Vinje V, Rognes ME (2022) Human intracranial pulsatility during the cardiac cycle: a computational modelling framework. Fluids and Barriers of the CNS 19(1):1–17, doi:10.1186/s12987-022-00376-2

Cserr HF, Ostrach L (1974) Bulk flow of interstitial fluid after intracranial injection of blue dextran 2000. Experimental Neurology 45(1):50–60, doi:10.1016/0014-4886(74)90099-5

Cuddapah VA, Robel S, Watkins S, Sontheimer H (2014) A neurocentric perspective on glioma invasion. Nature Reviews Neuroscience 15(7):455–465, doi:10.1038/nrn3765

Dreha-Kulaczewski S, Joseph AA, Merboldt KD, Ludwig HC, Gärtner J, Frahm J (2015) Inspiration is the major regulator of human csf flow. Journal of Neuroscience 35(6):2485–2491, doi:10.1523/JNEUROSCI.3246-14.2015

Eide PK, Valnes LM, Lindstrøm EK, Mardal KA, Ringstad G (2021) Direction and magnitude of cerebrospinal fluid flow vary substantially across central nervous system diseases. Fluids and Barriers of the CNS 18(1):16, doi:10.1186/s12987-021-00251-6

Fultz NE, Bonmassar G, Setsompop K, Stickgold RA, Rosen BR, Polimeni JR, Lewis LD (2019) Coupled electrophysiological, hemodynamic, and cerebrospinal fluid oscillations in human sleep. Science 366(6465):628–631, doi:10.1126/science.aax5440

Gary RG, Rolland JP, Song W, Nedergaard M, Parker KJ (2023) Fluid compartments influence elastography of the aging mouse brain. Physics in Medicine and Biology 68(9), doi:10.1088/1361-6560/acc922

Goriely A, Geers MGD, Holzapfel GA, Jayamohan J, Jérusalem A, Sivaloganathan S, Squier W, van Dommelen JAW, Waters S, Kuhl E (2015) Mechanics of the brain: perspectives, challenges, and opportunities. Biomechanics and Modeling in Mechanobiology 14(5):931–965, doi:10.1007/s10237-015-0662-4

Green MA, Bilston LE, Sinkus R (2008) In vivo brain viscoelastic properties measured by magnetic resonance elastography. NMR in Biomedicine 21(7):755–764, doi:10.1002/nbm.1254

Griffiths E, Budday S (2022) Finite element modeling of traumatic brain injury: Areas of future interest. Current Opinion in Biomedical Engineering 24, doi:10.1016/j.cobme.2022.100421

Hadaczek P, Yamashita Y, Mirek H, Tamas L, Bohn MC, Noble C, Park JW, Bankiewicz K (2006) The "perivascular pump" driven by arterial pulsation is a powerful mechanism for the distribution of therapeutic molecules within the brain. Molecular therapy : the journal of the American Society of Gene Therapy 14(1):69–78, doi:10.1016/j.ymthe.2006.02.018

Hammarlund-Udenaes M, De Lange EC, Thorne RG, et al. (2014) Drug delivery to the brain. AAPS Press, Springer, DOI 10:978–1, doi:10.1007/978-1-4614-9105-7

Holter KE, Kehlet B, Devor A, Sejnowski TJ, Dale AM, Omholt SW, Ottersen OP, Nagelhus EA, Mardal KA, Pettersen KH (2017) Interstitial solute transport in 3D reconstructed neuropil occurs by diffusion rather than bulk flow. Proceedings of the National Academy of Sciences 114(37):9894–9899, doi:10.1073/pnas.1706942114

Hornkjøl M, Valnes LM, Ringstad G, Rognes ME, Eide PK, Mardal KA, Vinje V (2022) Csf circulation and dispersion yield rapid clearance from intracranial compartments. Frontiers in Bioengineering and Biotechnology 10, doi:10.3389/fbioe.2022.932469

Hughes TM, Kuller LH, Barinas-Mitchell EJ, Mackey RH, McDade EM, Klunk WE, Aizenstein HJ, Cohen AD, Snitz BE, Mathis CA, et al. (2013) Pulse wave velocity is associated with β-amyloid deposition in the brains of very elderly adults. Neurology 81(19):1711–1718, doi:10.1212/01.wnl.0000435301.64776.37

Iliff JJ, Wang M, Liao Y, Plogg BA, Peng W, Gundersen GA, Benveniste H, Vates GE, Deane R, Goldman SA, Nagelhus EA, Nedergaard M (2012) A Paravascular Pathway Facilitates CSF Flow Through the Brain Parenchyma and the Clearance of Interstitial Solutes, Including Amyloid β. Science Translational Medicine 4(147):147ra111–147ra111, doi:10.1126/scitranslmed.3003748

Jordan BD (2013) The clinical spectrum of sport-related traumatic brain injury. Nature Reviews Neurology 9:222–230, doi:10.1038/nrneurol.2013.33

Kelley DH, Bohr T, Hjorth PG, Holst SC, Hrabětová S, Kiviniemi V, Lilius T, Lundgaard I, Mardal KA, Martens EA, et al. (2022) The glymphatic system: Current understanding and modeling. Iscience 25:104987, doi:10.1016/j.isci.2022.104987

Lewis LD (2021) The interconnected causes and consequences of sleep in the brain. Science 374(6567):564–568, doi:10.1126/science.abi8375

Lloyd RA, Butler JE, Gandevia SC, Ball IK, Toson B, Stoodley MA, Bilston LE (2020) Respiratory cerebrospinal fluid flow is driven by the thoracic and lumbar spinal pressures. The Journal of Physiology 598(24):5789–5805, doi:10.1113/JP279458

Louveau A, Smirnov I, Keyes TJ, Eccles JD, Rouhani SJ, Peske JD, Derecki NC, Castle D, Mandell JW, Lee KS, Harris TH, Kipnis J (2015) Structural and functional features of central nervous system lymphatic vessels. Nature 523(7560):337–341, doi:10.1038/nature14432

MacAulay N, Keep RF, Zeuthen T (2022) Cerebrospinal fluid production by the choroid plexus: a century of barrier research revisited. Fluids and Barriers of the CNS 19(1):26, doi:10.1186/s12987-022-00323-1

Mardal KA, Rognes ME, Thompson TB, Valnes LM (2022) Mathematical Modeling of the Human Brain: From Magnetic Resonance Images to Finite Element Simulation. Springer Nature, doi:10.1007/978-3-030-95136-8

Murphy MC, Huston J, Ehman RL (2019) Mr elastography of the brain and its application in neurological diseases. NeuroImage 187:176–183, doi:10.1016/j.neuroimage.2017.10.008, physiological and Quantitative MRI

Ozawa H, Matsumoto T, Ohashi T, Sato M, Kokubun S (2004) Mechanical properties and function of the spinal pia mater. Journal of Neurosurgery: Spine 1(1):122–127, doi:10.3171/spi.2004.1.1.0122

Rasmussen MK, Mestre H, Nedergaard M (2018) The glymphatic pathway in neurological disorders. The Lancet Neurology 17(11):1016–1024, doi:10.1016/S1474-4422(18)30318-1

Rasmussen R, O'Donnell J, Ding F, Nedergaard M (2020) Interstitial ions: A key regulator of state-dependent neural activity? Progress in Neurobiology 193:101802, doi:10.1016/j.pneurobio.2020.101802

Reeves BC, Karimy JK, Kundishora AJ, Mestre H, Cerci HM, Matouk C, Alper SL, Lundgaard I, Nedergaard M, Kahle KT (2020) Glymphatic system impairment in alzheimer's disease and idiopathic normal pressure hydrocephalus. Trends in Molecular Medicine 26(3):285–295, doi:10.1016/j.molmed.2019.11.008

Rennels ML, Gregory TF, Blaumanis OR, Fujimoto K, Grady PA (1985) Evidence for a "paravascular" fluid circulation in the mammalian central nervous system, provided by the rapid distribution of tracer protein throughout the brain from the subarachnoid space. Brain Research 326(1):47–63, doi:10.1016/0006-8993(85)91383-6

Sætra MJ, Ellingsrud AJ, Rognes ME (2023) Neural activity induces strongly coupled electro-chemo-mechanical interactions and fluid flow in astrocyte networks and extracellular space – a computational study. bioRxiv doi:10.1101/2023.03.06.531247

Sakka L, Coll G, Chazal J (2011) Anatomy and physiology of cerebrospinal fluid. European Annals of Otorhinolaryngology, Head and Neck Diseases 128(6):309–316, doi:10.1016/j.anorl.2011.03.002

Schubert JJ, Veronese M, Marchitelli L, Bodini B, Tonietto M, Stankoff B, Brooks DJ, Bertoldo A, Edison P, Turkheimer FE (2019) Dynamic 11c-pib pet shows cerebrospinal fluid flow alterations in alzheimer disease and multiple sclerosis. Journal of Nuclear Medicine 60(10):1452–1460, doi:10.2967/jnumed.118.223834

Smith JH, Humphrey JA (2007) Interstitial transport and transvascular fluid exchange during infusion into brain and tumor tissue. Microvascular Research 73(1):58–73, doi:10.1016/j.mvr.2006.07.001

Su L, Wang M, Yin J, Ti F, Yang J, Ma C, Liu S, Lu TJ (2023) Distinguishing poroelasticity and viscoelasticity of brain tissue with time scale. Acta Biomaterialia 155:423–435, doi:10.1016/j.actbio.2022.11.009

Thal DR, Griffin WST, de Vos RA, Ghebremedhin E (2008) Cerebral amyloid angiopathy and its relationship to alzheimer's disease. Acta neuropathologica 115:599–609, doi:10.1007/s00401-008-0366-2

TULLY B, VENTIKOS Y (2011) Cerebral water transport using multiple-network poroelastic theory: application to normal pressure hydrocephalus. Journal of Fluid Mechanics 667:188–215, doi:10.1017/S0022112010004428

Van Hemmen JL, Sejnowski TJ (2005) 23 problems in systems neuroscience. Oxford University Press

Vinje V, Ringstad G, Lindstrøm EK, Valnes LM, Rognes ME, Eide PK, Mardal KA (2019) Respiratory influence on cerebrospinal fluid flow–a computational study based on long-term intracranial pressure measurements. Scientific reports 9(1):9732

Wagshul ME, Eide PK, Madsen JR (2011) The pulsating brain: A review of experimental and clinical studies of intracranial pulsatility. Fluids and Barriers of the CNS 8(1):1–23, doi:10.1186/2045-8118-8-5

Wichmann TO, Damkier HH, Pedersen M (2022) A brief overview of the cerebrospinal fluid system and its implications for brain and spinal cord diseases. Frontiers in Human Neuroscience 15:850

Xie L, Kang H, Xu Q, Chen MJ, Liao Y, Thiyagarajan M, O'Donnell J, Christensen DJ, Nicholson C, Iliff JJ, et al. (2013) Sleep Drives Metabolite Clearance from the Adult Brain. Science 342(6156):373–377, doi:10.1126/science.1241224

Chapter 2

Meshing the intracranial compartments: Cerebellum, Cerebrum, Brain stem and Cerebrospinal fluid

Lars Magnus Valnes and Kent-Andre Mardal

2.1 Introduction

In Mardal et al. (2022) we discussed techniques for generating surfaces and meshes for FEniCS based simulations. The main focus was on meshing based on the default surface reconstructions provided by FreeSurfer Fischl (2012), namely the surfaces of the grey and white matter of the cerebrum. In this chapter we extend the methodology and consider also the cerebrospinal fluid (CSF) of the subarchnoid space (SAS), cisterna magnum, aquaduct and foramen magnum, along with the brain stem and the cerebellum. Furthermore, we describe the tools to mark volumes, surfaces and interfaces along with tools that provide the clean cuts needed for setting boundary conditions at for instance the cervicocranial junction in the upper neck. As such, we aim to provide the software tools for, in particular, brain-wide simulation of the glymphatic system Jessen et al. (2015) where the viscous flows of the CSF interact with the poro-elastic brain within. We furthermore briefly outline updates to SVM-Tk.

This chapter is divided in three sections; 1) segmenting the intracranial compartments using FreeSurfer to amend and/or supplement the segmentation of the cerebrum (as described in Mardal et al. (2022)); 2) construction and repairing surfaces for meshing and a brief description of the updates in SVM-Tk; 3) creating meshes with sharp edge and unique interface tags needed for providing boundary conditions in the cervicocranial junction.

2.2 Segmenting the intracranial compartments.

In this section we will explain how to segment the CSF spaces (aqueduct, SAS, cisterna magna) either manually or by T2-weighted MRI images along with segmentations of the brainstem, the cerebellum and the upper part of the spinal cord.

2.2.1 Initial segmentation

FreeSurfer is an open source software for magnetic resonance imaging (MRI) based analysis with widespread use in neuroscience[1]. Our interest here is the semi-automatic segmentation of the brain's structures and regions. An initial segmentation that serves as our starting point is typically provided by the FreeSurfer command `recon-all`. The creation of the initial segmentation is typically time-consuming, upto 24 hours, and may require manual corrections. Such corrections will not be covered here, since the topic can be quite extensive, and there exists tutorials and courses on FreeSurfer [2].

The FreeSurfer segmentation and surfaces are included in the dataset `MRI2FEMii`, which can be found online and downloaded at Zenodo link. We can inspect the FreeSurfer output, found in `MRI2FEMii` folder, by executing the following commands.

```
$export SUBJECT_DIRS="./MRI2FEMii-dataset
```

```
$freeview -recon <subject_name>
```

This will open the graphic user interface Freeview[3], which will be used to view the segmentation files. Some useful Freeview shortcuts are; PgUp and PgDn can be used to scroll through the image; Alt+C will circle between loaded images; Alt+X, Alt+Y, Alt+Z and Alt+3 will respectively choose sagittal, coronal, axial and 3D as the primary view in Freeview.

In Fig. 2.1, we see the initial segmentation, white matter segmentation and the constructed hemisphere surfaces. Furthermore, we can inspect the different regions by hovering the cursor and reading of the regions name. This allows for simpler investigation of the segmentation, as we can find the cerebellum, brainstem and the ventricles of the brain.

[1] `https://surfer.nmr.mgh.harvard.edu/`

[2] `https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial`

[3] `https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/OutputData_freeview`

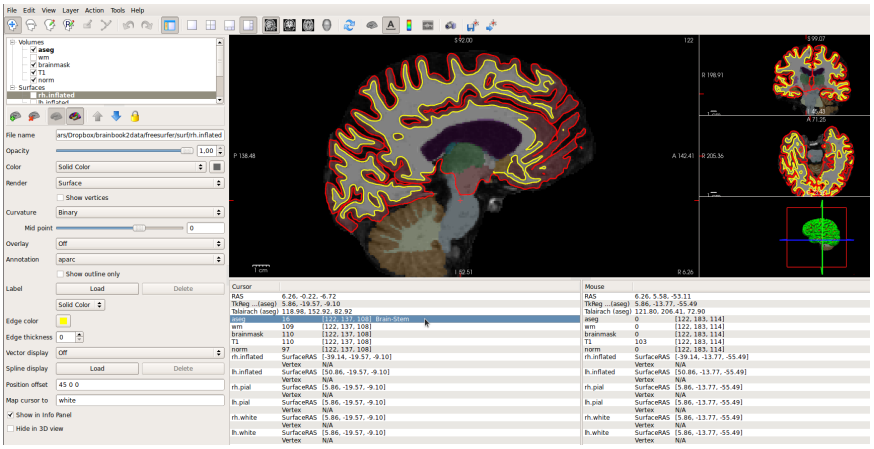


Fig. 2.1: The image shows the Freeview GUI with readouts of labels and image intensities by hovering over regions.

2.2.2 Amending the FreeSurfer segmentation.

The surfaces provided by FreeSurfer are the 1) pial surface at the interface between CSF and grey matter and 2) the interface between the grey and white matter (in both hemispheres of the cerebrum). The other surfaces we have to create ourselves. Below we address the creation of the surfaces for the internal CSF spaces (ventricles and aqueduct), the external SAS CSF space, the brain stem and cerebellum. Note that the various CSF spaces are prone to be thin and therefore challenging to segment at certain places.

2.2.3 The Cerebral Aqueduct

The cerebral aqueduct is a frequent region of interest, in particular for patientients with Normal-Pressure-Hydrocephalus (NPH) since CSF fluid velocities are commonly assessed using Phase-Contrast MRI in this region Lindstrøm et al. (2018); Ringstad et al. (2017). However, while certain segments are clearly visibile, a complete continuous aqueduct may be challenging to obtain and in particular in young patients. The size of the aqueduct differs between individuals, and the cross-sectional area ranges from 0.2 to 1.8 mm^2 Cinalli et al. (2011). Compared to the default 1 mm^3 voxel resolution used for the FreeSurfer segmentation the aqueduct may then be small. In particular, the cerebral aqueduct is not part of the FreeSurfer segmentation, but may be included as a part of the 4th ventricle connecting to the 3rd ventricle. Thus, if it is missing, 3rd and 4th ventricles would appear disconnected in the segmentation, as indicated in Fig. 2.3.

Missing aqueducts can often be fixed with supplementary T2-weighted image, if the images has sufficient quality and resolution. Otherwise, we can manually segment the cerebral aqueduct using Freeview. Both alternatives are outlined below.

Manual editing of the cerebral aqueduct.

We will edit the cerebral aqueduct directly in the segmentation file. Therefore, we open the segmentation and T1-weighted MR image in Freeview with the following command

```
$freeview aseg.mgz:colormap=lut:opacity=0.5 orig.mgz
```

We navigate the cursor to the 4th ventricle in the brainstem, and place the marker with a Left-click. We select the action to Voxel Edit, and optionally set the view to only show axial MRI slice by selecting the 1x1 and the axial, see Fig. 2.2. Selecting Voxel Edit will open a window, in which we can set the brush value to 15, which is the label value for the 4th ventricle.

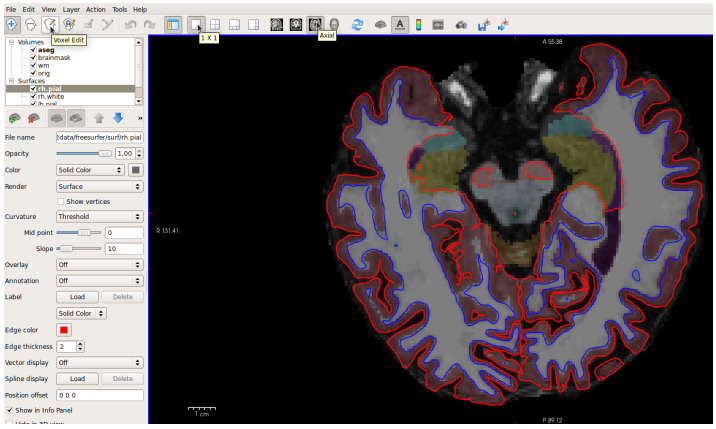


Fig. 2.2: The image shows the Freeview GUI setup for manual editing of the cerebral aqueduct with the cursor indicating the step and the red cross marking the cerebral aqueduct.

We have now completed the setup, and can begin marking voxels with Left-click, and we can also rewind with CTRL + Z. The voxels we want to mark have low intensity, and are situated inside the brainstem segmentation connecting 3rd and 4th ventricles.

Registration and normalization of T2-weighted images.

In the dataset `MRI2FEMii`, there is a T2-weighted image that can be used to supplement the segmentation. However, first the T2-weighted image must be registered onto the T1-weighted image. This can be done automatically during the initialization (with the option `-T2` to specify the T2-weighted image to `recon-all`). If it has not already been done, the following commands performs the registration:

```
# Convert the T2 image to size 256^3 with voxel size 1 mm^3
mri_convert -odt float -c T2.orignal.mgz T2.conformed.mgz
# bbregister perform within-subject cross-modal registration
bbregister --s $SUBJECT --mov  T2.original.mgz --lta transforms/T2raw.lta --
    ↪ init-fsl --T2
# Apply transformation to the conformed T2 image.
mri_vol2vol --mov T2.conformed.mgz --targ orig.mgz --lta T2raw.lta --o T2.
    ↪ registered.mgz
# Rescale the T2 image.
mri_nu_correct.mni --i T2.registered.mgz --o T2.nu.mgz
# Rescale the T2 image.
mri_normalize T2.nu.mgz T2.norm.mgz
```

First, we resample the T2-weighted image to the conformed space of FreeSurfer . This is followed by a registration procedure using `bbregister`. The registration process requires that we have the output of FreeSurfer segmentation process `recon-all`. Additionally, we need to adjust the FreeSurfer environment folder `SUBJECTS_DIR` to the folder that contains the FreeSurfer output. The registration provides the transformation in a file with suffix `lta`, which we apply to T2-weighted image to align it with FreeSurfer output. The remaining steps will normalize the values in the T2-weighted image, which gives us the file `T2.norm.mgz`.

We remark that the two last steps, `mri_nu_correct.mni` and `mri_normalize` are recommended in particular for making a semi-automatic procedure for running through many patients. FreeSurfer commonly uses a normalization[4] where the peak in white matter is normalized to be around 110 in T1 weighted images, while the normalization process does direct translate to T2 weighted images, it will normalize the intensities to be within the range from 0 to 255.

T2-weighted images augmentation of the cerebral aqueduct.

A common problem with the aqueduct (containing CSF) is that it is mislabeled as the brainstem (label 16). However, with a good T2-weigthed image, the aqueduct gives a strong signal (as CSF) and can therefore be easily identified. The following script search for voxels within the brainstem region with strong signal in the T2-weigthed image (more than 40, assuming normalized as above) and set their label to the 4th ventricle (label 15). We remark that a clustering procedure `mri_volcluster` is utilized to ensure that the voxels are connected. The result is shown in Fig. 2.3.

```
# Creates a mask of the brainstem segmentation
```

[4] `https://surfer.nmr.mgh.harvard.edu/fswiki/normalization`

```
mri_binarize --i aseg.mgz --match 16 --o brainstem.mgz
# Creates a mask of the CSF inside the brainstem mask.
mri_binarize --i T2.norm.mgz --min 30 --mask brainstem.mgz --o cerebral.
    ↪ aqueduct.mgz
# Catagorize different voxel clusters of the mask
mri_volcluster --in  cerebral.aqueduct.mgz  --thmin 1 --ocn  aqueduct.clusters.
    ↪ mgz
# Create a mask of the largest voxel cluster and set the value to 15
mri_binarize --i aqueduct.clusters.mgz  --match 1 --o cerebral.aqueduct.mgz --
    ↪ binval 15
# Combine the maks voxels with the freesurfer segmentation
mri_mask -transfer 15 aseg.modified.mgz cerebral.aqueduct.mgz aseg.modified.mgz
```

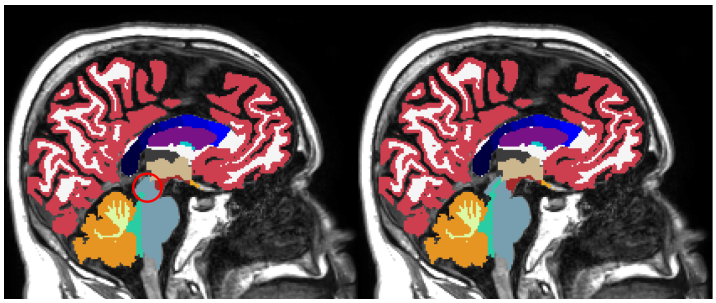


Fig. 2.3: The left panel shows a cerebral aqueduct mislabeled as brainstem (highlight with a red circle). The right panel shows the cerebral aqueduct after T2-weigthed correction.

2.2.3.1 Segmentation of the SAS using T2-weighted images.

The methodology for segmenting the SAS is similar to that of segmentation the cerebral aqueduct using T2 weighted images. However, it may be smart to control that the threshold used in the segmentation is appropriate by visual inspection. The approach is detailed below.

A manual inspection of both T1- and T2 weighted images is conveniently performed as (see Fig. 2.4):

```
$freeview -v T1.mgz T2.norm.mgz:colormap=nih:colorscale=10,200
```

In order to determine the best threshold values, we may adjust the colorscale on the left in Freeview. The preset colormap, `nih`, will only show values within the colorscale range. Hence, we can investigate different thresholds by adjusting the `min` parameter of the colorscale. Setting the `min` parameter to 40 seems to include most of the CSF in the SAS, shown in Fig. 2.4. Then, we use the same method for the normalized T1-weighted image (`T1.mgz` found in the FreeSurfer output), but we adjust the `max` parameter instead, found to be around 65.

When confident with the thresholds, we use `mri_binarize` to make initial mask for the CSF of the SAS (first two lines in the script below). The SAS may contain a large number of small isolated clusters due to the fact that the space is thin and separated by folds. To exploit clustering algorithms, we therefore combine the SAS with the brain segmentation, provided by `aseg.mgz`. This will ensure that the small SAS clusters are connected with the brain segmentation, which allows us to employ `mri_volcluster` to remove other smaller clusters, like the vitreous humor (a part of the eye). The combination of the brain and SAS segmentation may not fit perfectly, and therefore there might exists unmarked voxels inside the mask. We use `mri_morphology` to fill these unmarked voxels, and to ensure that the region is (mostly) convex. We remark that, as seen for instance in Fig. 2.4, there is a large concave kink in front of the SAS surrounding the brain stem. However, this kink is so large that it will not be filled in with only a few iterations. This will produce a bounding mask called `dura.mgz`, as the dura is its surface.

```
# Create a mask of CSF in T2-weighted image.
mri_binarize --i T2.norm.mgz --min 110 --o T2.threshold.mgz
# Subtract any voxels with tissue or skull based on T1-weighted image.
mri_binarize --i freesurfer/mri/T1.mgz --max 65 --mask T2.threshold.mgz  --o
    ↪ SAS.mgz
# Merge mask with freesurfer segmentattion.
mri_binarize --i aseg.mgz --min 1 --o Dura.inner.mgz --merge SAS.mgz
# Catagorize different voxel clusters of the mask.
mri_volcluster --in Dura.inner.mgz --thmin 1 --ocn Dura.clusters.mgz
# Create a mask of the largest voxel cluster.
mri_binarize --i Dura.clusters.mgz --match 1 --o Dura.inner.mgz
# Closes any holes/concave region in the mask.
mri_morphology Dura.inner.mgz close 4 Dura.mgz
```

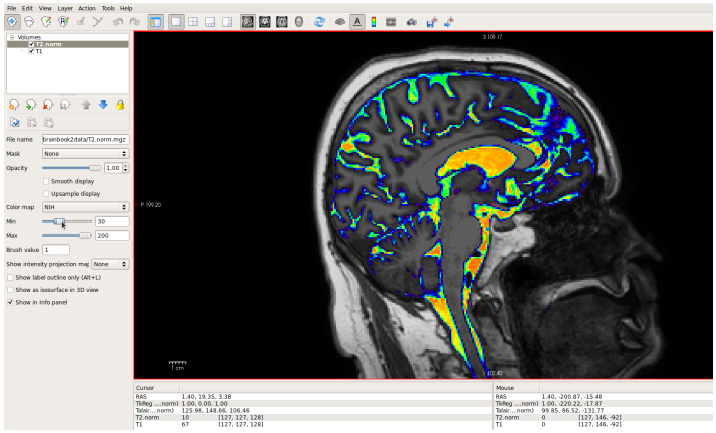


Fig. 2.4: Shows the normalized T2-weighted image with the normalized T1-weighted image in the background

2.2.4 Extending the brainstem segmentation to the spinal cord

There are two issues with the brain stem and the associated CSF spaces. First, the FreeSurfer segmentation of the brainstem does not have a well-defined border to the spinal cord, as seen in Fig. 2.5 (left). Second, for simulations we will need a clear and convenient interface to set the boundary conditions. Hence, a clean cut like that seen in Fig. 2.5 (right) is required. We remark that it is preferable that the cut is normal to the expected flow direction.

The first problem is similar to the problem with the aqueduct described earlier, namely that a region should be extended along certain signal values if the region is within a surrounding mask. The surrounding mask is in this case the dura mask created earlier. Below we employ the clustering algorithms to mark the regions not previously marked. In detail, we start by extracting voxels containing tissue inside mask `dura.mgz`, created in sec.2.2.3.1, using threshold of `T1.mgz`. Then, we remove voxels already in FreeSurfer segmentation, followed by removing unconnected voxels. We set the mask value to be 16, label for the brainstem, and transfer the voxels over to the FreeSurfer segmentation file, named `aseg.extended.mgz`. Below we provide a script for generating such correction:

```
# Creates a mask of tissue inside the dura mask. FIXME
mri_binarize --i T1.mgz --min 65 --mask dura.mgz --o brainstem.extended.mgz
# Subtracts the freesurfer segmentation from the previous mask
mri_binarize --i aseg.mgz --match 0 --mask brainstem.extended.mgz --o brainstem
    ↪ .extended.mgz
# Catagorize different voxel clusters of the mask
mri_volcluster --in brainstem.extended.mgz  --thmin 1 --ocn brainstem.clusters.
    ↪ mgz
# Create a mask of the largest voxel cluster and set the value to 16
mri_binarize --i brainstem.clusters.mgz --match 1 --o brainstem.extended.mgz --
    ↪ binval 16
# Closes any holes/concave region in the mask. FIXME
mri_morphology  brainstem.extended.mgz close 4  brainstem.extended.mgz
# Combine the maks voxels with the freesurfer segmentation
mri_mask -transfer 16 aseg.mgz brainstem.extended.mgz aseg.modified.mgz
```

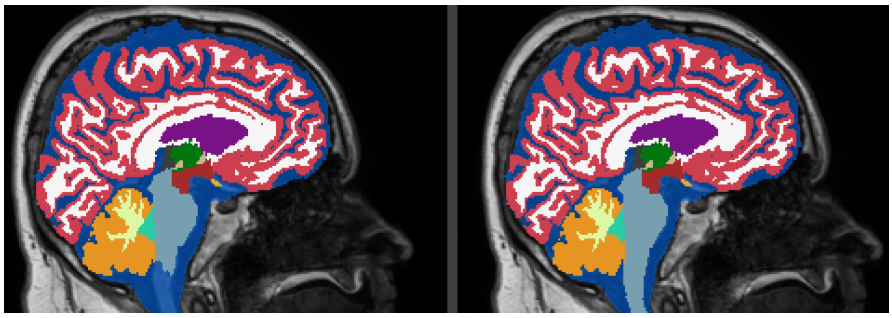


Fig. 2.5: The left panel shows brainstem not connected with the spinal cord. The right panel shows brainstem connected with the spinal cord.

2.3 Constructing and repairing Surfaces.

The surfaces are created from the image segmentations using the marching cube algorithm Lorensen and Cline (1987). That is, we utilize `mri_binarize` to both create a binarized volume image and an associated surface of the binarization. The process includes a smoothing operation parameterized in terms of the number of smoothing iterations. Note that the smoothing is not volume preserving and that the surfaces will shrink if extensive smoothing is applied.

2.3.1 Creating a bounding surface

The SAS consist of a network of narrow passages, which easily collapses due to smoothing. Therefore, we will not create a surface based on the SAS segmentation. Instead, we utilize the fact that the SAS exists inside the bounding mask of `dura.mgz`, and create a bounding surface. This means that the SAS can be defined as the difference between the bounding surface and surfaces of brain segmentation.

```
# Extracting bounding surface based on the inner dura mask.
mri_binarize --i Dura.mgz --match 1 --surf-smooth 5 --surf ./bounding_surface.
    ↪ stl
```

2.3.2 Cerebellum & Brainstem & 4th Ventricle

The cerebellum cortex has a fine grained surface with sub-resolution sulci and gyri. We neglect the sub-resolution features and consider its surface smooth. As such, we utilize the FreeSurfer command `mri_binarize` to create a mask of the cerebellum. We include the 4th ventricle such that the surface union with the brainstem in the next step becomes simpler. Then, we fill any potential cavities in the mask, in order to avoid problems related to not having simply connected volumes.

```
# Creating a mask of the cerebellum and the 4th ventricle.
mri_binarize --i ../aseg.modified.mgz  --match 7 46 8 47 15  --o cerebellum.mgz
# Fill holes that can result in surface errors.
mri_morphology cerebellum.mgz  fill_holes 5 cerebellum.mgz
# Extracting cerebellum surface
mri_binarize --i cerebellum.mgz --match 1 --surf-smooth 5 --surf ./cerebellum.
    ↪ stl
```

We can utilize the same approach as above, but also add additional segmentation tags for the brainstem and ventricles. The below script collects all the commands in a bash script:

```
$ bash get_surface -cerebellum -brainstem
```

2.4 SVMTK

The Surface Volume Meshing Toolkit (SVM-Tk) was introduced in Mardal et al. (2022) as a tool to generate volume meshes from surfaces. It was built as a Python-based front end interface to the Computational Geometry Algorithms Library (CGAL) Fabri et al. (2000). The code provides functionality for fixing and marking surfaces as well as a relatively robust mesh generation procedures. The full documentation for SVM-Tk[5] includes general installation instructions, list of requirements and API. Below, we mainly describe the new tools.

2.4.1 Importing surfaces in SVM-Tk

The surfaces created by FreeSurfer are not meshes, but rather collections of triangles that more or less define a surface. As there are no rule concerning the connectivity of the triangles, we may have that the triangles overlap and/or that triangles are missing. In fact, while the surface may look reasonable in a GUI, such as Paraview, the surface may still be far from a reasonable surface mesh. The creation of reasonable surfaces meshes from FreeSurfer surfaces (`stl` files) are detailed in Mardal et al. (2022). However, here we add further to the challenge as we are working with several surfaces and some of them are thin. These surfaces may intersect in small domains or even in points. In fact, they may intersect in a small number of points not present in the vertices or edges the surfaces. In particular, the FreeSurfer segmentation and surface smoothing may cause artificial intersections.

A different, but related and common source of problems are cells that are only connected with a common vertex (co-dimension 3) or edge (co-dimension 2), but not a facet (co-dimension 1) and with no other neighboring cells in common. We call such cells that have no common neighbors, connections of co-dimension > 1, but none for co-dimension 1 for non-adjacent cells. Such connections typically arise when the mesh criteria is to coarse and where the resolution is not sufficient to separate the surfaces. Hence, such connections arise in folds and self-intersections, resulting in bridge-like connections crossings between the neighboring surfaces.

We utilize the function `check_mesh_connections` to inspect if there are any hazardous connection in the constructed volume mesh. However, the removal of these connections may have a cascading effect, so instead it is advisable to have a properly prepared surface mesh before creating the volume mesh. As such, below we detail techniques of repairing surfaces.

[5] See https://github.com/SVMTK/SVMTK.

2.4.2 Surface repairs in SVM-Tk

The repair process involves several steps, and we will quickly go through these steps. We start with the issue that a surface-file may contain multiple and rough surfaces, possibly unconnected, with holes and varying mesh size. To get started we therefore assume that the largest surface is the primary surface and use `keep_largest_connected_component` to remove any other surface. Then, we ensure that there are no missing facets in the surface by calling the function `fill_holes`. This is primarily to avoid artifacts in the mesh that will look like threads from the mesh. We repair self intersections and other degeneracy in the surface with `repair_self_intersection`, and use `isotropic_remehsing` to provide a rather uniform surface mesh.

```
def repair_surface(surface, edge_length=1.0):
    # Keeps the largest connected component.
    surface.keep_largest_connected_component()
    # Fills most holes in triagnulation, not topological.
    surface.fill_holes()
    # Volume preserving surface smoothing.
    surface.smooth_taubin(20)
    # Removes self-intersection
    surface.repair_self_intersections(volume_threshold=0.4,
                                      cap_threshold=170,
                                      needle_threshold=1.5,
                                      collapse_threshold=0.3)
    # Remeshing surface to specified edge length
    surface.isotropic_remeshing(edge_length, 5, False)
    # Separate narrow gaps to avoid bridges in meshing.
    surface.separate_narrow_gaps()
    return surface
```

2.4.3 Surface tuning in SVM-Tk

As mentioned, the constructed volume mesh can have hazardous connection due to short distance between non-adjacent/non-connected vertices, which commonly occurs in the folds on the brain surface. This can be fixed by either increasing the mesh resolution or pre-process the surface at a lower resolution. The first option will naturally increase the computational cost, thus we will focus on the second option, which consists of tuning the surfaces to the mesh resolution.

The tuning starts by decreasing the surface resolution with functions `isotropic_remeshing` and `collapse_edges`, with an input parameter of a specified edge length. This is primarily due to the tuning functions use the smallest edge length of a vertex to determine the displacement.

We utilize the function `separate_narrow_gaps` to separate the distance between folds, as shown in Mardal et al. (2022). The functions works by incrementally

separating non-adjacent vertices in the negative direction of the surface normal until the closest vertex adjacent/connected with an edge. This only works for folds, but we also have `separate_close_vertices`, which has no directional restriction, so it works on non-adjacent vertices in thin surfaces. The function works similar to `separate_narrow_gaps`, but the vertices are moved in the opposite direction of the non-adjacent vertex.

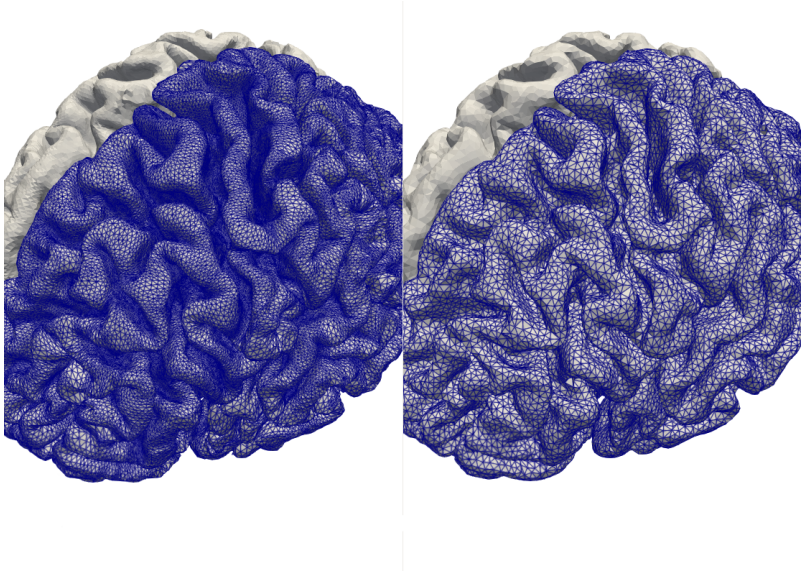


Fig. 2.6: Shows the before and after surface tuning.

2.4.4 Union of surfaces (white mater, brainstem and cerebellum)

To get a surface that encapsulate the ventricle system we merge the cerebellum, brainstem and white matter surface. The merging process works by taking the union sequentially, see Fig.2.7. We remark here that we have a particular order: we start by a union of cerebellum and brainstem, which is then followed by the white matter. The reason is that the brainstem has a significant and relatively clean interface towards the brainstem (when using the T1-weighted based segmentation in FreeSurfer). Taking the union of the brainstem and white would potentially result in two disjoint surfaces. There are occasions where ventricle surface is not encapsulated. Therefore, we will in sec.2.4.5 describe the method to fix this issue.

```
def sequential_union_wm_bs_c(white, brainstem, cerebellum):
    # Union between brainstem and cerebellum surfaces
    brainstem.union(cerebellum)
```

```
    # Repair any faults in the new surface.
    brainstem = repair_surface(brainstem)
    # Union between white and brainstem surfaces.
    white.union(brainstem)
    # Repair any faults in the new surface
    white = repair_surface(white)
    return white
```

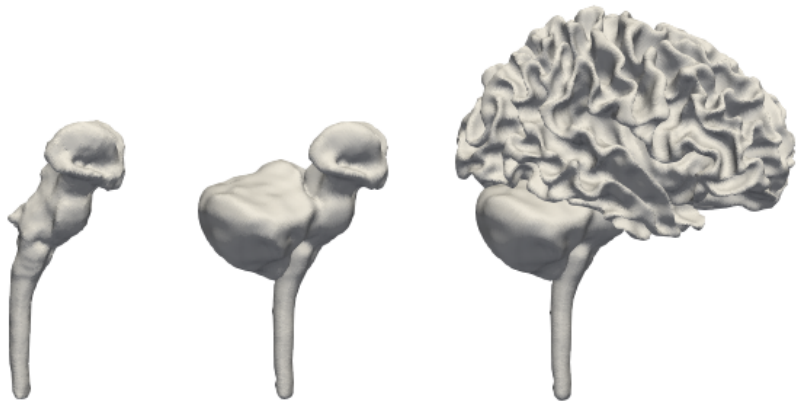


Fig. 2.7: Shows the process of merging surfaces into a single surface.

2.4.5 Anatomical correctness

We have previously created and repaired several cojoining surfaces in the previous sections which in principle can be used to generate meshes. However, the surfaces have been created independently, without any knowledge of other surfaces. Hence, there may be overlapping or crossing surfaces that are clearly anatomically incorrect.

Examples of such flaws are:

- The pial surfaces overlaps/crosses with the bounding surface.
- The cerebellum overlaps/crosses with the pial surface.
- The cerebellum overlaps/crosses with bounding surface.
- The surface of ventricles is connected to the SAS through other passages than cisterna magna.

To combine surfaces and investigate the process we may utilize the functions `enclose`, `embed`, `expose` to defined various intersections and unions of surfaces. In

detail, the function `enclose` will incrementally expand until it encloses another surface, `embed` will incrementally contract until it is embedded in another surface, and `expose` will incrementally contract until the surfaces are disjointed. The functions are illustrated in 2.8.

The parameters of these operations are `adjustment` and `smoothing`, which are factors for edge adjustment and Laplacian smoothing, respectively. These parameters depend on the regularity of the surface, and can in some cases act contradictory in the calculation of displacement. We also utilize the function `separate` to ensures a minimum distance between surfaces. The function requires that the surfaces does not intersect, and uses the same input parameters as the previous functions, mentioned `adjustment` and `smoothing`.

Concerning the flaws mentioned above, we suggest the following approach: The first flaw was caused by the smoothing of the bounding surface which caused the pial surfaces to be outside the bounding surface. Below, the problem is fixed using `enclose` to expanding the bounding surface locally and then seprarate:

```
def enclose_pial(bounding_surface, rhpial, lhpial):
    # Expands bounding surface to enclose the pial surfaces
    bounding_surface.enclose(lhpial)
    bounding_surface.enclose(rhpial)
    # Separate the pial and bounding surfaces
    bounding_surface.separate(lhpial)
    bounding_surface.separate(rhpial)
    return bounding_surface
```

Similarly, to address the second and third flaws, we first shrink the cerebellum with `expose` before we use `embed` to make sure that the cerebellum surface is embedded in the bounding surface.

```
def fix_cerebellum(cerebellum, lhpial, rhpial, bounding_surface):
    # Moves cerebellum outside the pial surfaces.
    cerebellum.expose(lhpial); cerebellum.expose(rhpial)
    # Moves cerebellum inside the bounding surface.
    cerebellum.embed(bounding_surface)
    # Separate cerebellum from the bounding surface.
    cerebellum.separate(bounding_surface)
    return cerebellum
```

Finally, we may have that the ventricle system is connected to the SAS in more locations than through the cisterna magna. In fact, in some cases the ventricle surface extending far outside the white surface, for example near the temporal horn of lateral ventricles, see Fig.2.9. So to decrease the run time, we start by taking the union of the ventricle system and the white surface, and then we use `enclose`:

```
def enclose_ventricles(white, ventricles):
    # Explain TODO
    white.union(ventricles)
    # Repair any faults in the new surface
```

```
    white = repair_surface(white)
    # Expand white surface to enclose the ventricles
    white.enclose(ventricles)
    # Contract ventricle surface to inside the white
    ventricles.embed(white)
    # Separate white and ventricle surface
    white.separate(ventricles)
    return white, ventricles
```

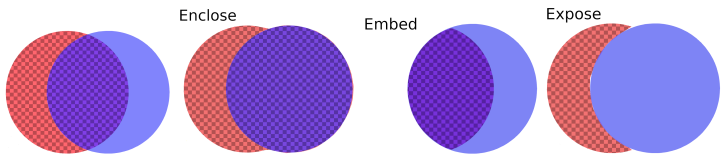


Fig. 2.8: Shows the result of the different functions `enclose`, `embed` and `expose` on two overlapping spheres. The primary sphere is red with checkerboard pattern, while the blue sphere defines the modifications. The checkerboard pattern shows the resulting output.

2.4.6 Creating the cisterna magna fluid pathway.

In order to avoid any unintentional pathways between the SAS and ventricular system, we imposed separation between the surfaces using the functions `enclose`, `embed`, `expose` and `separate`. The previous surface operations ensures that there are no unintentional CSF pathways between the SAS and ventricular system. However, it may have obstructed the cisterna magna, so we need to construct the cisterna magna pathway anew. This process involves creating a cylinder and subtracting it from the brainstem surface.

```
def make_cisterna_magna(ventricles, white, radius=2.0):
    # Computes the centerlines as a list of polylines
    centerlines = ventricles.mean_curvature_flow()
    # Finds the point with the lowest z-coordinate
    p1 = find_lowest_point(centerlines)
    # Find the closest point on the white surface
    p2 = white.get_closest_points(p1, 1)[0]
    # Extend the point in the same direction
    p2 = p2 + 1.6 * svm.Vector_3(p1, p2)
    # Construct an empty Surface
    cisterna_magna = svm.Surface()
    # Construct a cylinder with a specified radius
    cisterna_magna.make_cylinder(p1, p2, radius, 1.0)
    return cisterna_magna
```

2.4.7 Creating a plane surface for boundary conditions.

The Monro-Kellie doctrine Mokri (2001), states that the volume inside the skull remains constant, implying that the volume changes caused by the pulsating blood flow in arteries and veins is balanced by CSF flow Balédent (2014). Lately, also the effect of respiration on the venous-CSF coupling has been proposed as a main driver for CSF Vinje et al. (2019). In any case, the flow through foramen magnum is crucial and will typically be included in terms of a boundary condition.

It is an advantage to create a boundary that is normal to the "natural" flow in the artifical boundary cut, in order to avoid introducing artifical vortices associated with the boundary condition. Hence, we cut with a planar surface using the function `clip`. Furthermore, we choose the cut to be perpendicular to the centerline of the brainstem, which can be computed with the function `mean_curvature_flow`. It should be noted, that the function may return multiple centerlines if the surface bifurcate, so we would need to find the precise centerline.

```
def get_foramen_magnum(brainstem, z0):
    # Computes and returns centerlines as 2D array
    centerlines = brainstem.mean_curvature_flow()
    # Set default values
    num = 0; min_index = 0; min_value = 0
    # Finds the centerline with a point closest to z0.
    for no, points in enumerate(centerlines):
        index, value = min(enumerate(points),
                           key=lambda i: abs(i[1].z()-z0))
        if abs(z0 - value.z()) < abs(z0 - min_value):
            min_value = value.z(); num = no; min_index = index
    # Define centerline with a point closest to z0
    centerline = centerlines[num]
    # Define point closest to z0
    point = centerline[min_index]
    # Sets the orientation of the normal vector downwards
    if centerline[min_index+5].z() < centerline[min_index-5].z():
        vector = svm.Vector_3(point, centerline[min_index+10])
    else:
        vector = svm.Vector_3(point, centerline[min_index-10])
    # Create a Plane given a point and normal vector
    return svm.Plane_3(point, vector)
```

```
    # Clips the bounding and white surface.
    bounding_surface.clip(foramen_magnum,True)
    white.clip(foramen_magnum,True)
```

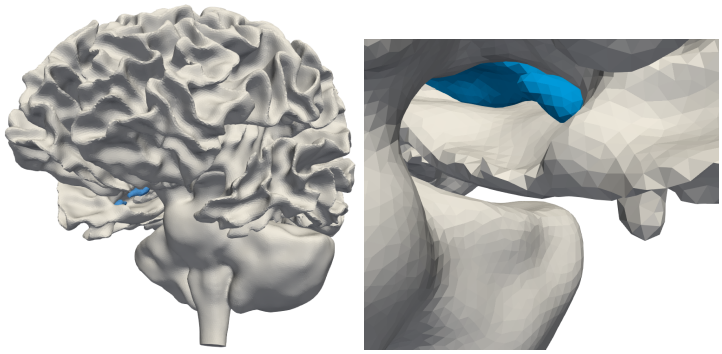


Fig. 2.9: Shows the ventricles surface in blue extending outside the white surface, which combines the surfaces of the cerebellum, brainstem and white matter.

2.5 Constructing mesh with sharp boundaries.

The default mesh construction in SVM-Tk does not preserve sharp edges in boundaries as such sharp edges are not present in our brain. However, at artifical boundaries like in Foramen Magnum described above, it is an advantage to preserve the sharpness. Therefore, we will go through how to keep sharp boundaries in the mesh, with different degrees of complexity. This will be done by preserving sharp edges, which is an edge where the angle between adjacent facet normals exceed a threshold angle, set around 60-90 degrees.

In SVM-Tk, there are two primary methods to add edges to the mesh construction, namely the Domain class functions `add_sharp_border_edges` and `add_border`. The function `add_border` is straightforward as it works by adding a list of points, representing connected edges, to be included in the mesh construction. It will also check if the new borders overlap previously added borders, as it would cause the mesh construction to crash.

The function `add_sharp_border_edges` uses a surface and threshold angle as primary inputs, and works by finding edges in the surface that exceed the threshold. Then, the sharp edges are added to the mesh construction through the `add_border` function. As the geometries become more complex, we can often lose overview of possible sharp boundaries, as seen in Fig. 2.9. Thus, we also have the option to only include sharp edges in a given plane by adding the plane as an argument for `add_sharp_border_edges`. This makes it simpler to preserve edges that created with the clip function.

```
# Detect and preserve sharp edges in a given plane.
domain.add_sharp_border_edges(bounding_surface,foramen_magnum
                              ,60)
# Create mesh
domain.create_mesh(64)
```

2.5.1 Checking the quality of the mesh.

We have shortly discussed the topic of mesh quality, but how do we measure it? There are multiple methods to evaluate the mesh quality and for instance `Paraview` has a wide range of checks that can be used for more information. Here, we will consider dihedral angle for each of the tetrahedrons in the mesh. The dihedral angle is the angle between two triangle planes in the tetrahedron. As a rough starting point, we would like the dihedral angle to be between 10 and 70 degrees. Elements with sharp angles are typically called slivers and are known to cause instabilities in simulations.

Slivers often appears when it is the only possible configuration to satisfy the meshing criteria. Thus, the more constraints we add, like surfaces, the more likely it is that slivers appear. This can be mitigated by using mesh optimization functions like `exude`, `lloyd`, `perturb` and `odt`. These functions are wrapped to the corresponding CGAL Cheng et al. (2000); Tournois et al. (2009); Du and Wang (2003); Du et al. (1999) functions.

In Fig.2.10, we the initial distribution of the dihedral angles, after `odt` optimization and after `exude`. We can see that the `exude` gives a significant change, but function may also remove troublesome tetrahedrons, which lead missing tetrahedrons at the boundary, as seen in Fig.2.10. The function `dihedral_angles` provides information of all the dihedral angles of a domain.

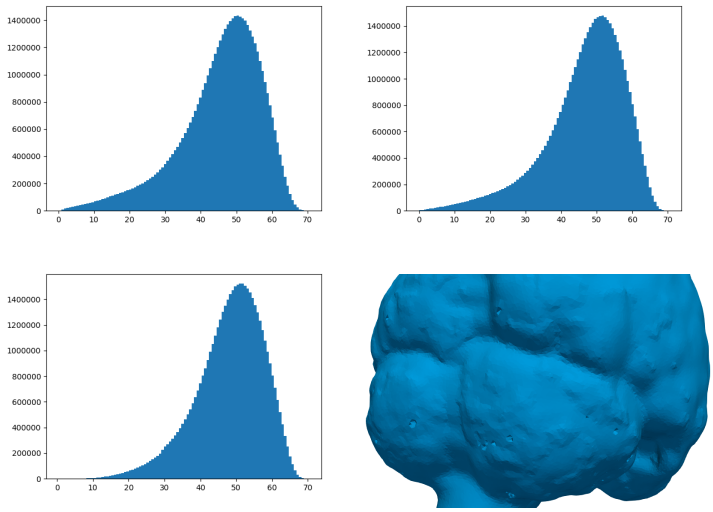


Fig. 2.10: The upper left panel shows the initial dihedral angle distribution. The upper right panel shows the dihedral angle distribution after `odt` optimization. The lower left panel shows the dihedral angle distribution after `exude` optimization. The lower right panel shows missing tetrahedrons due to the `exude` optimization.

2.5.2 Boundary segmentation

For constructed meshes with a sharp boundary to the mesh, like foramen magnum, it can be beneficial to mark the boundary with a different tag, making it simpler to set boundary conditions. This can be done with the function `boundary_segmentation`, which will segment a given surface of the constructed mesh based on sharp edges. The surface can be specified with a subdomain tag indicative of the surface enclosing the subdomain, or by two tags representing an interface between two tagged subdomains.

```
# Run surface segmentation on the interface (1,0)
domain.boundary_segmentations((1, 0), 60)
```

The resulting segmentation is visible in Fig. 2.11.

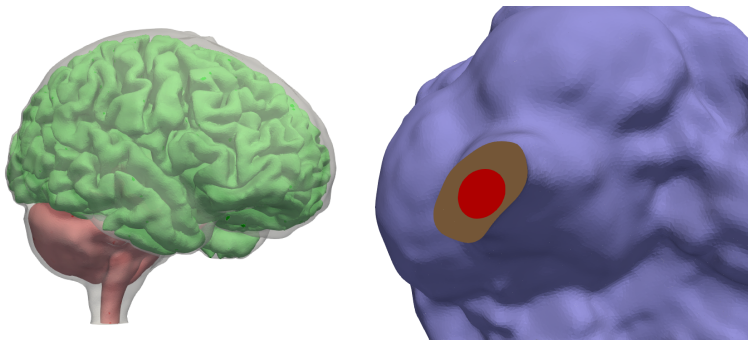


Fig. 2.11: The left panel shows the subdomain markings, with the SAS set on a lower opacity. The right panel shows the planar boundary has a different boundary tag than the SAS boundary.

References

Balédent O (2014) Imaging of the cerebrospinal. Adult hydrocephalus 256(128to512):121

Cheng SW, Dey TK, Edelsbrunner H, Facello MA, Teng SH (2000) Sliver exudation. Journal of the ACM (JACM) 47(5):883–904, doi:https://doi.org/10.1145/355483.355487

Cinalli G, Spennato P, Nastro A, Aliberti F, Trischitta V, Ruggiero C, Mirone G, Cianciulli E (2011) Hydrocephalus in aqueductal stenosis. Child's Nervous System 27(10):1621–1642, doi:10.1007/s00381-011-1546-2

Du Q, Wang D (2003) Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. International journal for numerical methods in engineering 56(9):1355–1373, doi:https://doi.org/10.1002/nme.616

Du Q, Faber V, Gunzburger M (1999) Centroidal voronoi tessellations: Applications and algorithms. SIAM review 41(4):637–676, doi:https://doi.org/10.1137/S0036144599352836

Fabri A, Giezeman GJ, Kettner L, Schirra S, Schönherr S (2000) On the design of CGAL a computational geometry algorithms library. Software: Practice and Experience 30(11):1167–1202, doi:10.1002/1097-024X(200009)30:11<1167::AID-SPE337>3.0.CO;2-B

Fischl B (2012) FreeSurfer. NeuroImage 62(2):774–781, doi:10.1016/j.neuroimage.2012.01.021

Jessen NA, Munk ASF, Lundgaard I, Nedergaard M (2015) The Glymphatic System: A Beginner's Guide. Neurochem Res 40(12):2583–2599, doi:10.1007/s11064-015-1581-6

Lindstrøm EK, Ringstad G, Mardal KA, Eide PK (2018) Cerebrospinal fluid volumetric net flow rate and direction in idiopathic normal pressure hydrocephalus. NeuroImage: Clinical 20:731–741

Lorensen WE, Cline HE (1987) Marching Cubes: A High Resolution 3D Surface Construction Algorithm. SIGGRAPH Comput Graph 21(4):163–169, doi:10.1145/37402.37422

Mardal KA, Rognes ME, Thompson TB, Valnes LM (2022) Mathematical Modeling of the Human Brain: From Magnetic Resonance Images to Finite Element Simulation. Springer Nature, doi:10.1007/978-3-030-95136-8

Mokri B (2001) The monro–kellie hypothesis. Neurology 56(12):1746–1748, doi:10.1212/WNL.56.12.1746

Ringstad G, Lindstrøm EK, Vatnehol SAS, Mardal KA, Emblem KE, Eide PK (2017) Non-invasive assessment of pulsatile intracranial pressure with phase-contrast magnetic resonance imaging. PloS one 12(11):e0188896

Tournois J, Srinivasan R, Alliez P (2009) Perturbing slivers in 3d delaunay meshes. In: Proceedings of the 18th international meshing roundtable, Springer, pp 157–173, doi:https://doi.org/10.1007/978-3-642-04319-2

Vinje V, Ringstad G, Lindstrøm EK, Valnes LM, Rognes ME, Eide PK, Mardal KA (2019) Respiratory influence on cerebrospinal fluid flow–a computational study based on long-term intracranial pressure measurements. Scientific reports 9(1):9732

Chapter 3
CSF Space

Martin Hornkjøl

Abstract This chapter will present an example use of SVMTK and FEniCS to study the fluid flow in the cranial compartment starting from surface files and using the laws of physics. We will describe how to create a full brain mesh including a bounding subarachnoid space (SAS), a ventricular system and the choroid plexus. Then, we will, an example application for this mesh, calculate the resulting fluid field from production of cerebral spinal fluid (CSF) at the choroid plexus by using the Stokes equation. A guide on how to hand draw the choroid plexus is also included.

3.1 Editing the Choroid Plexus

The choroid plexus, located within the lateral ventricles, is thought to be the main producer of CSF in the cranial compartment (SOURCE). When doing simulations of fluid flow, it is therefore important to include the choroid plexus as a distinct domain. The choroid plexus, however, can be problematic when meshing. Often the FreeSurfer labeling of the choroid plexus will have the choroid plexus on the boarder, or sometimes even outside, of the ventricles. In applications where the local flow in the ventricles is not relevant, it might be easier to redraw the choroid plexus, in Freeview, so that it is located entirely within the ventricles. This is done by opening your choroid plexus volume file in Freeview. Set the color map to "Lookup Table" so you can see the different labels and select the "Voxel edit" option (Figure 3.1). Here, you can select the brush value manually or you can use the "Color Picker" option to select a value by clicking on the displayed image. With the brush, overwrite the sections of the choroid plexus that is outside or at the edge of the lateral ventricles with the brush value of the lateral ventricles. If you want to expand the choroid plexus in a different direction, select the brush value of the choroid plexus, and draw

Martin Hornkjøl
University of Oslo, Address of Institute, e-mail: marhorn@uio.no

it in as you please. Our assumption before redrawing the choroid plexus is that we do not care about local effect. Therefore, the shape of the choroid plexus does not matter as long as it is within the lateral ventricles. Remember to scroll through all the volume layers and redraw everywhere necessary. The resulting choroid plexus surface can be extracted with

```
mri_binirize --i $input --match 31 --o $output
mri_binirize --i $input --match 63 --o $output
```

3.2 Making a Full Brain Mesh by Combining Surfaces

Our goal is to create a mesh of the brain with a bounding SAS. We start with surface files for the right and left pial matter, the right and left white matter, the ventricles, the right and left choroid plexus and the SAS. These need to be combined into the full brain mesh. We will assume that the surfaces we use have gone through the processing described in detail in (LARS CHAPTER REF) and are therefore smooth and does not contain any holes.

The surfaces have to be combined into a single mesh. We want to end up with six subdomains, namely, the white matter, pial matter, SAS, ventricles, choroid plexus and the aqueduct.

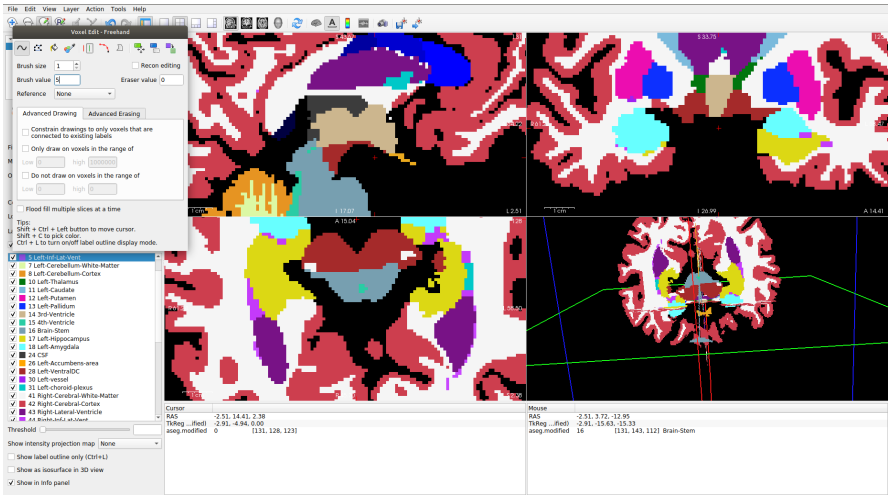


Fig. 3.1: The choroid plexus can be edited in Freeview. This is done with "Voxel edit" where the label of a pixel in the volume can be changed.

A script `create_mesh.py` to create the mesh is provided in the github repository. I will describe the most important steps in the script in the following. All surfaces are loaded as SVMTK surfaces as

```
lh_white = svm.Surface(lh_white)
rh_white = svm.Surface(rh_white)
```

The total white matter consists of different components, namely the left and right white matter, the brainstem and the cerebellum. As we want the total white matter to encompass the ventricles we will also include this surface. These components need to be combined with a union method. For this to work there need to be no, or at least few, self intersections (IS THIS TRUE?) in every surface. To make sure of this use the SVMTK functionalities

```
lh_white.repair_self_intersections()
rh_white.repair_self_intersections()
```

on your white matter and all other surfaces that will be included in your mesh.

A union of the left and right white matter can be created by using

```
white = svm.union_partially_overlapping_surfaces(
                                    lh_white,
                                    rh_white, 20, 0.1, 0)
```

When taking a union of multiple surfaces, the result will often have some faults. For instance, holes in the surface may have appeared, parts of the surface might have separated or self intersection may have appared. It is therefore prudent to run some fixes of potential problems that can occur. The fixes we run are as follows

```
white.fill_holes()
white.keep_largest_connected_component()
white.repair_self_intersections()
```

We want the total white matter to encompass the ventricles and we want the choroid plexus to be embedded in the ventricles. We start with the latter. Note that if your choroid plexus is two surfaces you first need to take the union and do fixes as earlier

```
lh_cohorid_plexus.union(rh_choroid_plexus)
choroid_plexus = svm.Surface(lh_cohorid_plexus)

choroid_plexus.fill_holes()
choroid_plexus.keep_largest_connected_component()
choroid_plexus.repair_self_intersections()
```

The embedding of the choroid plexus is achieved using the functions `enclose`, `embed` and `separate` as follows

```
ventricles.enclose(choroid_plexus ,4.5,-0.1,400)
ventricles.separate(choroid_plexus,0.01,0.0,400)
```

```
choroid_plexus.embed(ventricles,-4.5,0.1,400)
choroid_plexus.separate(ventricles,0.01,0.0,400)
```

Note that increasing the adjustment parameter might help the enclosing and embedding if you have trouble containing the choroid plexus. If you manually drew the choroid plexus to be well inside the ventricles, this step might not be necessary at all.

Then a union between the cerebellum, brainstem, ventricles and the white matter can be taken. This time by using the standard union method

```
white.union(cerebellum)
white.union(brainstem)
white.union(ventricles)
```

After this step we should again do some fixes to the surface

```
white.fill_holes()
white.keep_largest_connected_component()
white.repair_self_intersections()
```

As previously mentioned, we have to make sure that the ventricles are now completely within the white matter. Therefore, we go through the same procedure as with the choroid plexus

```
white.enclose(ventricles ,5.0,-0.1,400)
white.separate(ventricles,0.1,0.4,400)
ventricles.embed(white,-5.0,0.1,400)
ventricles.separate(white,0.1,0.4,400)
```

The combination of all the surfaces into the total white matter is shown in Figure 3.2.

From the physiological point of view, there needs to be an outlet from the ventricles to the SAS. This outlet from the ventricles to the SAS can be made using the following command

```
white.difference(cisterna_magna_connection)
```

where the cisterna magna connection is a handcrafted cylinder connecting the ventricles and SAS for the bottom of the 4th ventricle (Figure 3.3). REF CHAPTER WHERE THIS IS EXPLAINED. The difference method removes this cylinder surface from the white matter surface making it a part of the SAS.

The subarachnoid space surface should encompass all of the brain, but this is not always the case in the FreeSurfer segmentation. Sometimes the SAS surface might be to small and the brain pokes out or the SAS is so thin that it is hard to get it resolved on a mesh. In these cases the SAS can be expanded by using

```
sas_surf.adjust_boundary(2)
```

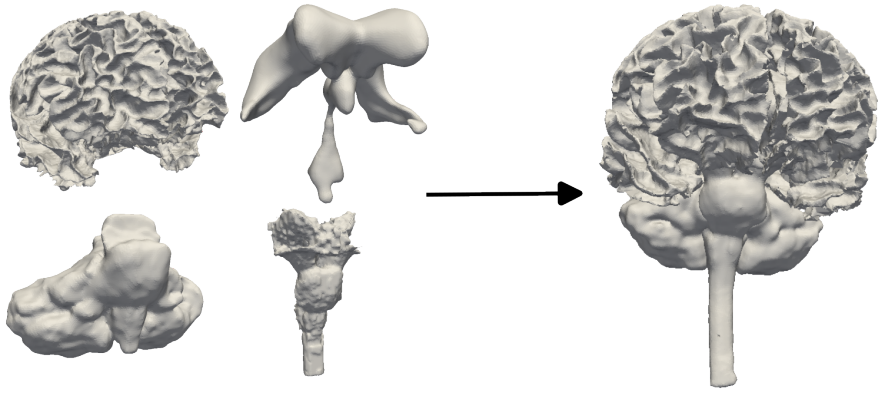


Fig. 3.2: The white matter, cerebellum, brainstem and ventricles are combined into a single surface we call the total white matter.

Fig. 3.3: A cylinder created to be the connection between the ventricles and the subarachnoid space.

The value 2 indicates that the surface is expanded by 2 mm. The difference before and after the use of `adjust_boundary()` is shown in Figure 3.4.

As mentioned earlier, the last subdomain we want is the aqueduct. The aqueduct is usually the thinnest part of the mesh and we want to mark it so that we can do targeted refinement later. Doing this is a manual process. We start by opening the ventricle surface file in Paraview and select `Hover Points On`. This shows the coordinates of whichever point we are mousing over. The aqueduct is the thinnest part of the ventricles, connecting the 3rd and 4th ventricle. We note the top and bottom point of the aqueduct and call these p1 and p2 respectively. We then do a horizontal cut on the

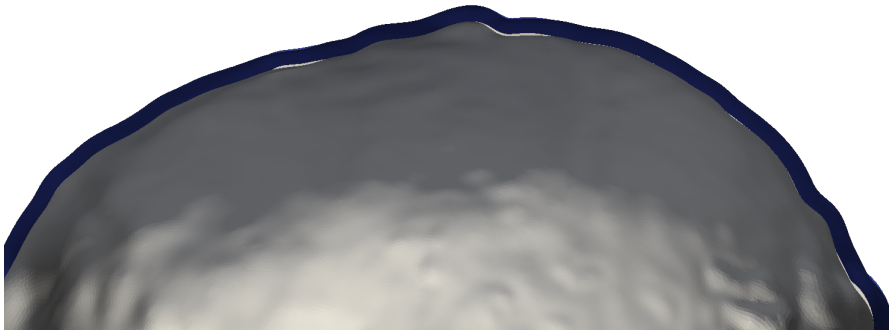


Fig. 3.4: The original SAS surface (in white) and the expanded SAS surface (in blue). It has been expanded by 2 mm in the normal direction everywhere.

top and bottom point, leaving only the aqueduct surface (MAYBE ADD FIGURE?). The code to do this is the following

```
p1 = svm.Point_3(-1, -23, 7)
p2 = svm.Point_3(-1,-30,-6)
aqueduct = svm.Surface(ventricles)
aqueduct.clip(p1,svm.Vector_3(p2,p1),4.,invert=True)
aqueduct.clip(p2,svm.Vector_3(p1,p2),4.,invert=True)
```

Now, we have all the surfaces ready, and we can divide the mesh into 6 subdomains. Namely, the SAS, the grey matter, the white matter, the ventricles, the choroid plexus and the aqueduct. This is done in SVMTK as

```
surfaces = [sas, lh_pial, rh_pial, white, ventricles, vent4,
                                    choroid_plexus]

smap = svm.SubdomainMap(7)

smap.add("1000000", 1)

smap.add("*100000", 2)
smap.add("*010000", 2)
smap.add("*110000", 2)

smap.add("*1000", 3)

smap.add("*100", 4)

smap.add("*1", 5)

smap.add("*10", 6)

domain = svm.Domain(surfaces, smap)
```

Finally, to create the mesh we use the command

```
domain.create_mesh(32)
```

The input to the `create_mesh()` method, here 32, is the resolution of the resulting mesh.

When using a mesh for an application it is important to consider its resolution. Some parts of the mesh such as the aqueduct and parts of the SAS are very small and there might not be a single free vertex in the region. This can cause several problems. If, for instance, you have no slip conditions on the boundaries of the region, as we will have in the example below, there will be no free vertex in the domain and all flow through the region will effectively be stopped.

Scripts like `refine_mesh.py` (IS THIS SCRIPT INTRODUCED ANYWHERE?) can increase the resolution of problematic regions, but this will yield a mesh with a large amount of vertices and you will probably be forced to work on a supercomputer.

3.3 Application to Stokes flow

CSF flow in the cranial compartment is of interest when studying diseases such as dementia. This we can model using the mesh created above. We will use the mesh created in the last section to calculate the Stokes flow from the choroid plexus to an outflow route. Here, the outflow route is chosen to be the parasagittal sinus, although others are possible Hladky and Barrand (2014).

We solve the Stokes problem

$$\mu\nabla^2 u - \nabla p = 0 \text{ in } \Omega_F \tag{3.1}$$
$$\nabla \cdot u = g \text{ in } \Omega_F \tag{3.2}$$
$$u = 0 \text{ in } \Omega_P \tag{3.3}$$
$$\mu\nabla u \cdot n - pn = 0 \text{ on } \partial\Omega_{ps} \tag{3.4}$$
$$u = 0 \text{ on } \partial\Omega \setminus \partial\Omega_{ps} \tag{3.5}$$
$$u = 0 \text{ on } \Gamma_{FP} \tag{3.6}$$

where Ω_F is the CSF region, consisting of the SAS, the ventricles and the choroid plexus, and Ω_P is the brain region, namely, the white and grey matter. The function g = Constant in Ω_{cp} and zero elsewhere is a source producing CSF in the choroid plexus and $\partial\Omega_{ps}$ is the boundary of the parasagittal sinus. The rest of the skull has a no slip condition imposed on it. Last, Γ_{FP} is the intersection of the CSF and brain domain. To solve this coupled PDE for u and p, we use the finite element method implemented with FEniCS implemented in the script `3D_stokes.py`. We start by importing this module:

```
from fenics import *
```

We also set $\mu = 0.8$ mPa which is the viscosity of water.

```
# Define physical constants
mu1 = 8e-4
```

Then we read the mesh, subdomains and boundaries into FEniCS. The mesh is in the HDF5 format. This allows for MPI-parallel computing:

```
# Read in the mesh, subdomains and boundaries
mesh = Mesh()
hdf = HDF5File(mesh.mpi_comm(), "meshes/brain_mesh.h5", "r")
hdf.read(mesh, "/mesh", False)
print(f'mesh num vertices {mesh.num_vertices()}')
print(f'mesh num cells {mesh.num_cells()}')
print(f'h max {mesh.hmax()}')
print(f'h min {mesh.hmin()}')
subdomains = MeshFunction("size_t", mesh, mesh.topology().dim())
hdf.read(subdomains, "/subdomains")
boundaries  = MeshFunction("size_t", mesh, mesh.topology().dim()
                                    - 1)
hdf.read(boundaries, "/boundaries")
```

We now consider the finite element discretization and build the function space with trial and test functions. We are using Taylor-Hood elements where the velocity is in piecewise quadratic and the pressure in piecewise linear.

```
# Build function space with Taylor-Hood elements
# and define the trial and test functions
P2 = VectorElement("Lagrange", mesh.ufl_cell(), 2)
P1 = FiniteElement("Lagrange", mesh.ufl_cell(), 1)
TH = P2 * P1
W = FunctionSpace(mesh, TH)

(u, p) = TrialFunctions(W)
(v, q) = TestFunctions(W)
```

Then, we define needed boundaries and setup the boundary conditions. In our case we need to mark the parasagittal sinus. As a simplification we choose the parasagittal sinus to be rectangle on the top part of the head with dimension of about 50x80 mm (SHOULD I ADD IMAGE?). To set this in FEniCS, we create a subdomain class, restrict it to be on the boundary and choose its range in the x, y, and z direction. The x, y and z ranges are chosen by looking at the mesh in paraview.

```
# Define the outflow route
class BoundaryPS(SubDomain):
    def inside(self, x, on_boundary):
        return x[0] > -31 and x[0] < 18 and x[1] > -65 \
            and x[1] < 13 and x[2] > 60 and on_boundary

Outflow = BoundaryPS(); Outflow.mark(boundaries, 15)
```

The ventricles are enclosed in the white matter and the only exit route from the ventricles, in our mesh, is the small duct at the end of the 4th ventricle. In order for our fluid not to cross the walls of the ventricles, we have to impose no slip conditions on the intersection of the ventricles and white matter. The fluid can also not move from the SAS into the white and grey matter. Similarly, we therefore put no slip conditions on these intersections. The skull also has a no slip condition on it, making the parasagittal sinus the only exit route. In FEniCS, this can be implemented as

```
# Define Dirichlet boundary conditions
n = FacetNormal(mesh)
no_slip = Constant((0.0, 0.0, 0.0))


bc1 = DirichletBC(W.sub(0), no_slip, boundaries, 7) # Skull
bc2 = DirichletBC(W.sub(0), no_slip, boundaries, 12) # Ventricle/
                                    White
bc3 = DirichletBC(W.sub(0), no_slip, boundaries, 9) # White/CSF
bc4 = DirichletBC(W.sub(0), no_slip, boundaries, 8) # Grey/CSF
bc5 = DirichletBC(W.sub(0), no_slip, boundaries, 14) # Aqueduct/
                                    White

bcs = [bc1, bc2, bc3, bc4, bc5]
```

We create different integration domains for the fluid domain, consisting of the ventricles, SAS and choroid plexus, and the brain domain which includes the white and grey matter. These are called `dxF` and `dxP` respectively.

```
# Define integration domains
dx = Measure("dx", domain=mesh, subdomain_data=subdomains)
dxP = dx((2, 3))
dxF = dx((1, 4, 5, 6))
ds = Measure("ds", domain=mesh, subdomain_data=boundaries)
```

The Stokes equations (3.1) and (3.2) can be written in variational form and implemented as below (CITE SOMETHING?). Here, `P` is a preconditioner for the Stokes equations. Since the fluid is only in the SAS, ventricles and choroid plexus, we just integrate over the fluid domain and use `ident_zeros()` for both the system and the preconditioner to set the velocity to zero in the brain domain.

```
# Define weak form of the equations with a preconditioner in the
# fluid domainand assemble the system.
# We use the ident_zeros() method to set the solution to zero
                                    elsewhere
f = Constant((0.0, 0.0, 0.0))
g_source = Constant(0.006896552)

a = (mu1*inner(grad(u), grad(v)) - div(v)*p - q*div(u)) * dxF
p = (mu1*inner(grad(u), grad(v)) + (1.0/mu1)*p*q) * dxF

L = inner(f, v) * dx - g_source * q * dx(5)
```

```
A, b = assemble_system(a, L, bcs)
P, _ = assemble_system(p, L, bcs)
A.ident_zeros()
P.ident_zeros()
```

The value of `g_source` is the strength of the CSF producting source and its exact value has to be determined by testing. How we do this is described bellow.

We solve the system with a MINRES solver

```
# Solve system
U = Function(W)
solver = KrylovSolver("minres", "amg")
solver.set_operators(A, P)
it = solver.solve(U.vector(), b)
u, p = U.split(deepcopy=True)
print(f'it {it}')
```

and save the solutions of both the velocity and pressure as xdmf-files using the `write_checkpoint()` method

```
# Save the velocity and pressure as xdmf files
with XDMFFile(mesh.mpi_comm(), 'solution/velocity.xdmf') as xdmf:
  xdmf.write_checkpoint(u, "velocity", 0)

with XDMFFile(mesh.mpi_comm(), 'solution/pressure.xdmf') as xdmf:
  xdmf.write_checkpoint(p, "pressure", 0)
```

We can calculate the rate of water flow by integrating the velocity over the parasagittal sinus boundary. If we multiply this by the number of seconds in a day and convert the unit from mm^3/day to L/day we find the total fluid production in a day

```
print("Fluid production per day", assemble(dot(u,n)*ds(24))*24*
                                   3600*1e-6)
```

With the knowledge of how much fluid was produced with our choice of `g_source`, we can go back and adjust the value so that we get a production of 0.5 L/day Pardridge (2016) which is assumed t be the production rate of CSF in humans. The fluid production is linear in `g_source`. That is, if, for instance, your initial guess i results in 1 L/day just divide `g_source` by 2 and your production will be 0.5 L/day.

The solution shows the magnitude of a velocity field going from the choroid plexus through the ventricles and out through the cisterna mangna (Figure 3.5). From there, the velocity spreads upward through the SAS and out of the parasagittal sinus. The velocity reach a maximum of 2.6 mm/s at the thinnest part of the aqueduct, while the SAS velocity is in the range of 20-30 μm/s.

If done on a full mesh with a resolution good enough for the ventricles and SAS to be resolved, this script has to be run on a supercomputer because of memory

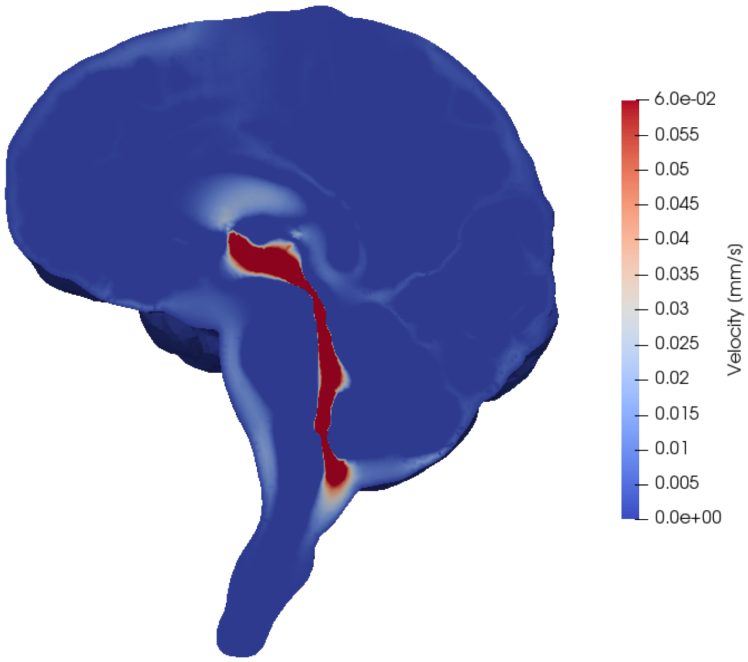


Fig. 3.5: The magnitude of the CSF velocity field on our mesh computed wit the Stokes equations. The mesh is cut such that the 3rd and 4th ventricle and the aqueduct are seen. The plot is scaled such that the cutoff maximum velocity is 0.06 mm/s so that the velocity field outside of the ventricles are visible.

requirements. If you want to run this on your own computer an alternative script with a 2D mesh is available at github.

The research based on this mesh can be extended to explore the transport of molecules the the brain. You can use the velocity field to solve the convection-diffusion equation and even add other effect such as dispersion.

References

Hladky SB, Barrand MA (2014) Mechanisms of fluid movement into, through and out of the brain: evaluation of the evidence. Fluids and Barriers of the CNS 11(1):1–32, doi:10.1186/2045-8118-11-26

Pardridge WM (2016) CSF, blood-brain barrier, and brain drug delivery. Expert Opinion on Drug Delivery 13(7):963–975, doi:10.1517/17425247.2016.1171315, pMID: 27020469

Chapter 4

The pulsating brain: an interface-coupled fluid-poroelastic interaction model of the cranial cavity

Marius Causemann, Vegard Vinje and Marie E. Rognes

Abstract The brain's biomechanics is driven by a dynamic interplay between the tissue, blood, cerebrospinal fluid (CSF) and interstitial fluid (ISF) and exhibit complex fluid flow and displacement patterns. Despite being essential to normal brain function, our understanding of intracranial dynamics is still limited, and various diseases are associated with impaired CSF flow and elevated intracranial pressure Wagshul et al. (2011). Here, we present a computational model of cardiac-induced pulsatile motion inside the human cranial cavity. The CSF flow in the subarachnoid space and ventricular system is modelled using the time-dependent Stokes equations, and coupled with Biot's poroelasticity equations in the brain tissue, thus integrating all major intracranial constituents into the modelling approach. Employing the pulsatile inflow of blood into brain tissue as a driver of motion, the model enables us to study the dynamics of the entire intracranial system. In addition to the modelling aspects, we showcase creating volumetric multidomain meshes from surface meshes using fTetWild, using the FEniCSextension Multiphenicsto solve monolithic multiphysics problems, and generate dynamic, interactive standalone animations of the obtained results with PyVistaand ʒdfull.

Marius Causemann
Simula Research Laboratory, Norway e-mail: mariusca@simula.no

Vegard Vinje
Simula Research Laboratory, Norway e-mail: vegard@simula.no

Marie E. Rognes
Simula Research Laboratory and Department of Mathematics, University of Bergen, Norway e-mail: meg@simula.no

4.1 Introduction

In this chapter, we present a numerical model of cardiac-induced intracranial pulsatility. Taking the close interplay between CSF flux and tissue motion into account, we cast the problem as a fluid-poroelastic interaction problem and present a fully coupled finite element discretization of the resulting system of equations. Further, we explain and illustrate how to conduct such interface coupled multiphysics simulations on complex geometries - starting from mesh generation, over the main simulation code to post-processing and visualization. Specifically, we focus on a Python-based workflow involving fTetWild(meshing) Hu et al. (2020a), FEniCS Alnæs et al. (2015) and Multiphenics(finite element simulations) Ballarin (2023) and PyVista Sullivan and Kaszynski (2019) and ʒdfull (visualization) Trzesiok (2023).

4.2 The brain and CSF - a fluid-poroelastic interaction problem

The human cranium is often considered a near rigid vault in which three main components interact: the vasculature (arteries, capillaries and veins), CSF-filled compartments and the brain parenchyma (Balédent, 2014). As all these constituents are largely incompressible, any addition of volume leads to a rapid increase of intracranial pressure (ICP) and subsequent fluid flux. Indeed, the temporal change of cerebral blood volume during the cardiac cycle induces a complex periodic pattern of CSF and blood flow and tissue motion (Wagshul et al., 2011). More specifically, during systole, the arterial blood flow into the brain exceeds the venous outflow, the brain expands its volume, causing a rise of ICP and inducing CSF flow both within the cranium and through the spinal canal into the spinal subarachnoid space. Subsequently, diastolic venous outflow dominates the vascular dynamics, leading to a decrease of ICP and a reversal of CSF flow (Balédent, 2014).

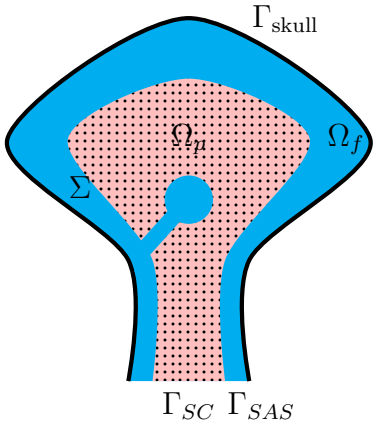


Fig. 4.1 The brain parenchyma (Ω_p, pink) and the CSF-filled spaces (Ω_f, blue). The circular CSF-filled space at the center represents the ventricular system, connected with the cranial subarachnoid space through a narrow opening. The interface of both domains is denoted by Σ. Additionally, the boundaries Γ_{skull} at the skull, Γ_{SC} at the spinal cord and Γ_{SAS} at the spinal SAS are highlighted

To model this brain-CSF interplay, we divide the cranial vault into two three-dimensional domains: the brain parenchyma denoted by Ω_p, and the surrounding CSF-filled spaces labeled as Ω_f (Figure 4.1). The two domains share a common interface $\Sigma = \Omega_f \cap \Omega_p$ with normal vector $\mathbf{n}$, pointing from Ω_f to Ω_p on Σ (and outwards on the boundary $\partial\Omega$). Further, Γ_{skull} is the outer boundary of the CSF space where the rigid skull encloses the cranial cavity. The lower boundary of the domain is split into two parts: the caudal continuation of the spinal cord is labeled Γ_{SC}, while Γ_{SAS} denotes the boundary to the spinal SAS.

The brain parenchyma as a poroelastic medium

We represent the brain tissue as a linear poroelastic medium with a single fluid network describing the extracellular CSF/ISF-space. The equations of linear poroelasticity describe the macroscopic interaction between the deformation of an elastic solid and a fluid occupying the pore space and enforce momentum conservation of the elastic skeleton and fluid mass conservation within the medium. We chose a three-field formulation based on the displacement $\mathbf{d}$, fluid (pore) pressure p^p and the additional total pressure $\phi = \alpha p^p - \lambda \operatorname{div} \mathbf{d}$, (Oyarzúa and Ruiz-Baier, 2016; Lee et al., 2017). With the infinitesimal strain tensor $\varepsilon(\mathbf{d}) = \frac{1}{2}(\nabla\mathbf{d} + \nabla\mathbf{d}^T)$ and a volume source term g, the equations are:

$$-\mathbf{div}[2\mu_s\varepsilon(\mathbf{d}) - \phi\mathbf{I}] = 0 \qquad \text{in } \Omega_p \times (0,T), \tag{4.1a}$$

$$\phi - \alpha p^p + \lambda \operatorname{div} \mathbf{d} = 0 \qquad \text{in } \Omega_p \times (0,T), \tag{4.1b}$$

$$\left(c + \frac{\alpha^2}{\lambda}\right)\partial_t p^p - \frac{\alpha}{\lambda}\partial_t \phi - \operatorname{div}\left(\frac{\kappa}{\mu_f}\nabla p^p\right) = g \qquad \text{in } \Omega_p \times (0,T). \tag{4.1c}$$

Here, κ represents the permeability, c the specific storage coefficient, and α the Biot-Willis coefficient. The identity operator is $\mathbf{I}$. The linear isotropic solid matrix is parameterized with the Lamé constants μ_s and λ, while μ_f denotes the viscosity of the permeating fluid.

Stokes flow in CSF spaces

The flow in the CSF compartments can be characterized as a low Reynolds number free fluid flow and hence described by the time-dependent Stokes equations for the CSF velocity $\mathbf{u}$ and fluid pressure p^f. Under these assumptions, the equations read as follows:

$$\rho_f \, \partial_t \mathbf{u} - \mathbf{div}[2\mu_f \, \varepsilon(\mathbf{u}) - p^f \, \mathbf{I}] = 0 \qquad \text{in } \Omega_f \times (0,T), \tag{4.2a}$$

$$\text{div } \mathbf{u} = 0 \qquad \text{in } \Omega_f \times (0,T), \tag{4.2b}$$

with the strain rate tensor $\varepsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathbf{T}}\right)$, constant CSF density ρ_f, and constant CSF viscosity μ_f.

Net blood flow as a driver of pulsatility

We assume a brain-wide expansion and contraction of blood vessels caused by the net flow of blood into the brain parenchyma as the main driver of motion and model it by a prescribed pulsatile source term g in (4.1c). Hence, we consider the net blood flow as a source term in the CSF/ISF compartment. While this might seem an oversimplification at first sight, it can be justified by the similarity of an inflow of blood and ISF on the macroscopic level: both lead to a volumetric expansion of the brain parenchyma and an increase of pore pressure. The arterial and venous flow in and out of the cranium is measurable by modern imaging techniques and allows computing the net cerebral blood flow over the cardiac cycle (Balédent, 2014). However, for simplicity, we illustrate by letting g be a spatially uniform sin curve in time with frequency f and amplitude A:

$$g(t) = A \cdot sin(2\pi f t). \tag{4.3}$$

Tissue-CSF interaction - coupling conditions

In addition to modelling the mechanics of both domains, suitable interface conditions account for the interaction between the brain tissue and the surrounding CSF. We enforce continuity of the normal flux on the interface (4.4a), momentum conservation (4.4b), and balance of total normal stress (4.4c). The last interface condition (4.4d) is the Beavers-Joseph-Saffman (BJS) condition, which relates the jump in the tangential velocities across the interface to the shear stress on the free flow side of the interface (Beavers and Joseph, 1967; Saffman, 1971). Accordingly, we require the following equations to hold on the interface Σ between the porous parenchyma and the CSF-filled spaces:

$$\mathbf{u}\cdot\mathbf{n} = \left(\partial_t \mathbf{d} - \frac{\kappa}{\mu_f}\nabla p^p\right)\cdot\mathbf{n} \qquad \text{on } \Sigma\times(0,T), \tag{4.4a}$$

$$\left(2\mu_f\,\varepsilon(\mathbf{u}) - p^f\,\mathbf{I}\right)\mathbf{n} = \left(2\mu_S\varepsilon(\mathbf{d}) - \phi\mathbf{I}\right)\mathbf{n} \qquad \text{on } \Sigma\times(0,T), \tag{4.4b}$$

$$-\mathbf{n}\cdot\left(2\mu_f\,\varepsilon(\mathbf{u}) - p^f\,\mathbf{I}\right)\mathbf{n} = p^p \qquad \text{on } \Sigma\times(0,T), \tag{4.4c}$$

$$-\mathbf{n}\cdot\left(2\mu_f\,\varepsilon(\mathbf{u}) - p^f\,\mathbf{I}\right)\tau_i = \frac{\gamma\mu_f}{\sqrt{\kappa}}\,(\mathbf{u}-\partial_t\mathbf{d})\cdot\tau_i \qquad \text{on } \Sigma\times(0,T), \quad i=1,2. \tag{4.4d}$$

Complementing the normal $\mathbf{n}$, we define orthogonal tangent vectors to the interface τ_i ($i = 1, 2$), and $\gamma > 0$ is the slip rate coefficient, which is a dimensionless constant depending on the microstructure of the porous medium.

Boundary and initial conditions

Assuming a rigid skull, we set no-slip conditions on the skull boundary Γ_{skull}:

$$\mathbf{u} = 0 \qquad \text{on } \Gamma_{\text{skull}}\times(0,T).$$

For the spinal cord boundary Γ_{SC}, we assume no displacement and no flux:

$$\mathbf{d} = 0 \quad \text{and} \quad \frac{\kappa}{\mu_f}\nabla p^p\cdot\mathbf{n} = 0 \qquad \text{on } \Gamma_{\text{SC}}\times(0,T).$$

Finally, we impose a zero-traction condition at the spinal SAS, allowing pulsatile motion of CSF in and out of the domain

$$\left(2\mu_f\,\varepsilon(\mathbf{u}) - p^f\,\mathbf{I}\right)\cdot\mathbf{n} = 0 \qquad \text{on } \Gamma_{\text{SAS}}\times(0,T).$$

We assume that the system is initially at rest with an initial pore pressure p_0:

$$\begin{aligned} &\mathbf{u} = \mathbf{0} && \text{on } \Omega_f\times\{0\},\\ &\mathbf{d} = \mathbf{0}, \quad p^p = p_0 && \text{on } \Omega_p\times\{0\}. \end{aligned}$$

4.3 Multidomain mesh generation for a brain-CSF model

Current medical imaging technologies offer high resolution spatial images of the whole cranium. Often, after preprocessing and segmentation of the image, a surface mesh in STL format of each of the relevant parts can be obtained. Here, we demonstrate how to generate a *multidomain* mesh (mesh with several subdomains and a

matching interface) suitable for multiphysics simulations with FEniCS with such surfaces meshes as a starting point. We showcase the meshing tool fTetWild and demonstrate its capability to compute boolean operations on surface meshes (Hu et al., 2020b). We have found that fTetWild is particularly well suited for mesh generation from imaging data due to its high robustness (no assumptions on mesh manifoldness, watertightness and absence of self-intersections), as the algorithm constructs and meshes an envelope of user-defined accuracy around the triangles contained in the input surface meshes.

Defining domains by surfaces

For standalone purposes, we here construct an idealized geometry from the following parts (Figure 4.2), and work with a low-resolution idealized representation of the human cranium.

- an innermost sphere representing the ventricles
- a second, larger sphere representing the parenchymal tissue
- an outermost sphere for the skull
- a thin cylinder representing the spinal cord
- a thicker cylinder for the CSF-filled space surrounding the spinal cord
- a small, thin cylinder connecting the ventricle with the SAS for the aqueduct

We generate these surface meshes (exemplarily for the ventricle and the aqueduct) using PyVista and save in STL format:

```
import pyvista as pv
stl_directory = Path("mesh/stls/")
ventricle = pv.Sphere(radius = 0.02)
aqueduct = pv.Cylinder(center=(0, 0.03, -0.03), direction=(0,1,-1
                                ),
                       radius=0.004, height=0.06).triangulate()
aqueduct.save(stl_directory / "aqueduct.stl")
ventricle.save(stl_directory / "ventricle.stl")
```

Creating the geometry representation

Next, we specify a *(Constructive solid geometry(CSG)-tree*, describing the order and type of boolean operations carried out on the surface meshes:

- first, we take the union of brain tissue components: parenchyma and brain stem

- then, we subtract the ventricle and aqueduct parts from the tissue
- finally, we take the union of the outermost sphere and cylinder with the result of the previous steps

```
import wildmeshing as wm
tetra = wm.Tetrahedralizer(epsilon=0.002, edge_length_r=0.05,
                           coarsen=False)
tetra.load_csg_tree(json.dumps(
    {"operation":"union",
     "right":
        {
        "operation":"difference",
            "left":{"operation":"union",
                "left": str(stl_directory / "parenchyma.stl"),
                "right": str(stl_directory / "cord.stl")},
            "right":{"operation":"union",
                "left": str(stl_directory / "aqueduct.stl"),
                "right":str(stl_directory / "ventricle.stl")},
        },
     "left":
        {"operation":"union",
            "left": str(stl_directory / "skull.stl"),
            "right":str(stl_directory / "canal.stl")},
        }
))
tetra.tetrahedralize()
point_array, cell_array, marker = tetra.get_tet_mesh()
```

The resulting multidomain geometry and mesh are visualized in Figure 4.2. The maximum distance between the input surfaces and the surface of the generated volumetric mesh can be controlled using the `epsilon` parameter (specifying the relative envelop size) and the desired relative edge length is specified with *edge_length_r*.

Moving to a more complex mesh only requires changing the input surface meshes and the mesh resolution, but does not introduce any additional difficulties. As an example, a mesh generated by the described workflow based on the surface meshes used in Causemann et al. (2022) is shown in Fig. 4.3. The corresponding code is included in the example notebooks.

Marking the interface and boundaries

fTetWild returns a marker function with different identifiers for each of the subdomains, but does not generate facet markers for internal interfaces or exterior boundaries. We therefore use FEniCS to loop over all facets in the mesh, check the subdomain IDs of the corresponding cells and identify the internal interfaces. Similarly, we mark the external boundaries (Γ_{Skull}, Γ_{SAS} and Γ_{SC}) by a combination of the facet location and the subdomain ID of the corresponding subdomain.

4.4 A numerical discretization and implementation of the fluid-poroelastic problem

Following Ruiz-Baier et al. (2022), we discretize the model in space using a monolithic finite element formulation and employ an implicit Euler scheme for time integration. Here, we state the variational form of the time-discretized problem and discuss its implementation with Multiphenics.

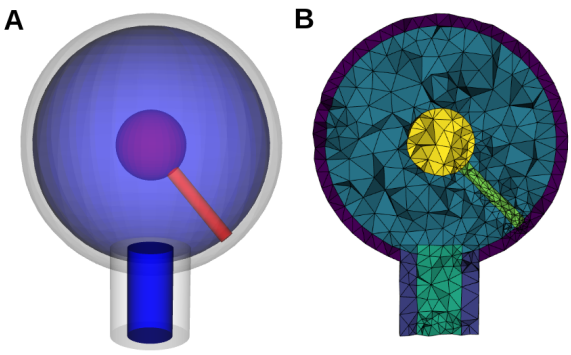
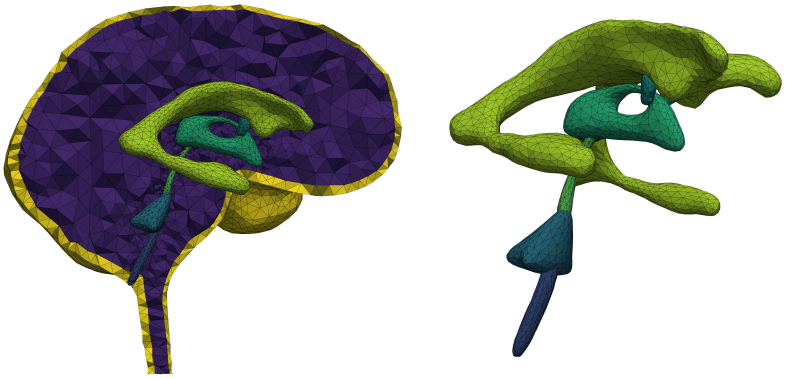


Fig. 4.2 (A) An idealized 3D cranial geometry consisting of a ventricle and aqueduct (red), the parenchyma and brain stem (blue) and the surrounding CSF space (gray/transparent); (B) the resulting multidomain mesh generated by fTetWild.



(a) (b)

Fig. 4.3: a) Example of an image-derived mesh generated with fTetWild and based on data from Causemann et al. (2022); b) Ventricular system of the image derived mesh

4.4.1 Variational Form of the Fluid-Poroelastic Interface Problem

In light of the boundary conditions defined in 4.2, we define the following function spaces:

$$\mathbf{H}^{\mathbf{1}}_{\mathrm{SK}}(\Omega_f) = \{\mathbf{v} \in H^1(\Omega_f)^d : \mathbf{v}\,|_{\Gamma_{\mathrm{skull}}} = 0\}$$
$$\mathbf{H}^{\mathbf{1}}_{\mathrm{SC}}(\Omega_p) = \{\mathbf{v} \in H^1(\Omega_p)^d : \mathbf{v}\,|_{\Gamma_{\mathrm{SC}}} = 0\}$$

where $H^1(\Omega_i)$ is the standard first-order Sobolev space of scalar functions with derivatives in $L^2(\Omega_i)$ equipped with the classical Sobolev norm on the domain Ω_i for $i \in \{P, F\}$.

After applying the implicit Euler method for time discretization with a fixed step size $\Delta t > 0$ for N_t time steps, we obtain the following semi-discrete weak form of the coupled problem:

Given suitable initial conditions $\mathbf{u}_0$, p_0^f, $\mathbf{d}_0$ and p_0^p, find $\mathbf{u}_n \in \mathbf{H}_{\mathrm{SK}}(\Omega_f)$, $p_n^f \in L^2(\Omega_f)$, $\mathbf{d}_n \in \mathbf{H}_{\mathrm{SC}}(\Omega_p)$, $p_n^p \in H^1(\Omega_p)$ and $\phi_n \in L^2(\Omega_p)$, for $n \in \{1, 2, \ldots, N_t\}$ such that:

$$\begin{aligned}
a^f(\mathbf{u}_n, \mathbf{v}) + b_1^f(\mathbf{v}, p_n^f) + b_2^\Sigma(\mathbf{v}, p_n^p) + b_3^\Sigma(\mathbf{v}, \Delta t^{-1}\mathbf{d}_n) &= F_n^f(\mathbf{v}) && \forall \mathbf{v} \in \mathbf{H}_{\mathrm{SK}}(\Omega_f)\\
b_1(u_n, q^f) &= 0 && \forall q^f \in L^2(\Omega_f)\\
b_3^\Sigma(\mathbf{u}_n, \mathbf{w}) + b_4^\Sigma(\mathbf{w}, p_n^p) + a_1^P(\mathbf{d}_n, \mathbf{w}) + b_1^p(\mathbf{w}, \phi_n) &= F_n^p(\mathbf{w}) && \forall \mathbf{w} \in \mathbf{H}_{\mathrm{SC}}(\Omega_p)\\
-b_2^\Sigma(\mathbf{u}_n, q^p) - b_4^\Sigma(\Delta t^{-1}\mathbf{d}_n, q^p) & && \\
+a_2^p(p_n^p, q^p) - b_2^p(\Delta t^{-1}\phi_n, q^p) &= G_n(q^p) && \forall q^p \in H^1(\Omega_p)\\
b_1^p(\mathbf{d}_n, \psi) + b_2^p(\psi, p_n^p) - a_3^p(\phi_n, \psi) &= 0 && \forall \psi \in L^2(\Omega_p)
\end{aligned}$$

with the variational forms

$$
\begin{aligned}
a^f(\mathbf{u},\mathbf{v}) &= \rho_f \int_{\Omega_f} \Delta t^{-1}\mathbf{u}\cdot\mathbf{v} + 2\mu_f \int_{\Omega_f} \epsilon(\mathbf{u}) : \epsilon(\mathbf{v}) + \frac{\gamma\mu_f}{\sqrt{\kappa}}\langle P_T\mathbf{u}, P_T\mathbf{v}\rangle_\Sigma \\
b_1^f(\mathbf{v}, q^f) &= -\int_{\Omega_f} q^f \ \mathrm{div}\ \mathbf{v} \\
b_2^\Sigma(\mathbf{v}, q^p) &= \langle q^p, \mathbf{v}\cdot\mathbf{n}\rangle_\Sigma \\
b_3^\Sigma(\mathbf{v}, \mathbf{d}) &= -\frac{\gamma\mu_f}{\sqrt{\kappa}}\langle P_T\mathbf{v}, P_T\mathbf{d}\rangle_\Sigma \\
b_4^\Sigma(\mathbf{w}, q^p) &= -\langle q^p, \mathbf{w}\cdot\mathbf{n}\rangle_\Sigma \\
a_1^p(\mathbf{d},\mathbf{w}) &= 2\mu_S \int_{\Omega_p} \epsilon(\mathbf{d}) : \epsilon(\mathbf{w}) + \frac{\gamma\mu_f}{\sqrt{\kappa}}\langle P_T\Delta t^{-1}\mathbf{d}, P_T\mathbf{v}\rangle_\Sigma \\
b_1^p(\mathbf{w},\psi) &= -\int_{\Omega_p} \psi \ \mathrm{div}\ \mathbf{w} \\
a_2^p(p^p, q^p) &= \left(c + \frac{\alpha^2}{\lambda}\right)\int_{\Omega_p} \Delta t^{-1} p^p q^p + \int_{\Omega_p} \frac{\kappa}{\mu_f}\nabla p^p \cdot \nabla q^p \\
b_2^p(\psi, q^p) &= \frac{\alpha}{\lambda}\int_{\Omega_p} \psi q^p \\
a_3^p(\phi,\psi) &= \frac{1}{\lambda}\int_{\Omega_p} \phi\psi
\end{aligned}
$$

and the linear functionals

$$
\begin{aligned}
F_n^f(\mathbf{v}) &= b_3^\Sigma(\mathbf{v}, \mathbf{d}_{i-1}/\Delta t) + \rho_f \int_{\Omega_f} \mathbf{u}_{i-1}/\Delta t \cdot \mathbf{v} \\
F_n^p(\mathbf{w}) &= -b_3^\Sigma(\mathbf{w}, \mathbf{d}_{i-1}/\Delta t) \\
G_n(q^p) &= -b_4^\Sigma(\mathbf{d}_{i-1}/\Delta t, q^p) - b_2^p(\phi_{i-1}/\Delta t, q^p) + \int_{\Omega_p} g q^p + \left(c + \frac{\alpha^2}{\lambda}\right) p_{i-1}^p q^p
\end{aligned}
$$

Here, P_T denotes the projection on the interface tangent and $\langle\cdot,\cdot\rangle_\Sigma$ the duality paring between the trace space $H^{1/2}(\Sigma)$ and its dual.
Using a Taylor-Hood type discretization, we approximate the tissue displacement, fluid velocity and pore pressure with continuous piecewise quadratic polynomials and use continuous piecewise linear functions for the total pressure and fluid pressure. Note that any other pair of Stokes-stable finite elements also results in a stable scheme (Ruiz-Baier et al., 2022).

Implementation

To facilitate the finite element implementation, we rely on Multiphenics, a Python library providing additional features for block systems and function spaces on subdomains in FEniCS. While such features are also available and under development in FEniCS(x), Multiphenics' robustness and its convenient interface make it particularly useful for multiphysics problems. Concretely, we can define our function spaces for fluid velocity $\mathbf{u}$, fluid pressure p^f, displacement $\mathbf{d}$, pore pressure p^p and total pressure ϕ as follows:

```
  W = FunctionSpace(mesh, "Lagrange", 1)
  W2 = FunctionSpace(mesh, "Lagrange", 2)
  V = VectorFunctionSpace(mesh, "Lagrange", 2)
  H = BlockFunctionSpace([V, W, V, W2, W],
                restrict=[fluidrestriction,
                    fluidrestriction,
                    porousrestriction,
                    porousrestriction,
                    porousrestriction])

  trial_functions = BlockTrialFunction(H)
  u, pF, d, pP, phi = block_split(trial_functions)

  test_functions = BlockTestFunction(H)
  v, qF, w, qP, psi = block_split(test_functions)
```

When defining a `BlockFunctionSpace`, Multiphenicsallows specifying spatial restrictions using the `restrict` argument with a `MeshRestriction`. The `MeshRestriction` holds the information on the spatial extent of the desired function space and is generated from the subdomain markers given by fTetWild.

Next, we implement the individual terms of for each of the blocks of the monolithic form (exemplarily shown here for $a^f(\mathbf{u}, \mathbf{v})$) and define the linear system:

```
# Create convenience functions for reocurring terms
eps = lambda u: sym(grad(u))
P_t = lambda u: u - dot(u, n) * n

def tang_interf(u, v):
  return inner(P_t(u("+")), P_t(v("+")))*ds_Sigma


# Define the individual blocks of the variational form
def a_f(u, v):
  return rho_f*dot(u/dt, v)*dxF \
    + 2*mu_f*inner(eps(u), eps(v))*dxF \
    + (gamma*mu_f/sqrt(kappa))*tang_interf(u, v)

...

lhs = [[a_f(u, v), b_1_f(v, pF), b_3_Sig(v, d/dt), b_2_Sig(v, pP)
                                  , 0],
```

```
        [b_1_f(u, qF), 0, 0, 0, 0],
        [b_3_Sig(u, w), 0, a_1_p(d, w), b_4_Sig(w, pP), b_1_p(w,
                                            phi)],
        [b_2_Sig(u, qP), 0, b_4_Sig(d/dt, qP), -
         a_2_p(pP, qP), b_2_p(phi/dt, qP)],
        [0, 0, b_1_p(d, psi), b_2_p(psi, pP), -a_3_p(phi, psi)]]

rhs = [F_f_n(v), 0, F_p_n(w), -G_n(qP), 0]
```

Multiphenicsoffers high-level functions for assembling the block system, setting boundary conditions and solve the block system. Using the multifrontal direct solver *MUMPS*, we first assemble the block system and then set up a `PETScLUSolver` solver, which will store the factorization computed in the first *solve* call and hence substantially reduce the cost of subsequent time steps. Additionally, we define the relevant Dirichlet boundary conditions and the time-dependent source term g, before starting the main time stepping loop - in which we update the source term, reassemble the right-hand-side, account for the boundary conditions and finally call the *solve* function of the direct solver.

```
AA = block_assemble(lhs, keep_diagonal=True)
solver = PETScLUSolver(AA, "mumps")

bc_d = DirichletBC(H.sub(2), Constant((0,0,0)),bm, spinal_sas_id)
bc_u = DirichletBC(H.sub(0), Constant((0,0,0)),bm, skull_id)
bcs = BlockDirichletBC([bc_d, bc_u])

g = Expression("A*sin(2*M_PI*f*t)", f =1, t=0, A=0.01, degree=0)
times = np.linspace(0, T, num_steps + 1)

for t in times:
    g.t = t
    FF = block_assemble(rhs)
    bcs.apply(FF)
    solver.solve(block_function.block_vector(), FF)
    block_function.block_vector().block_function().apply("to
                                        subfunctions")
```

4.5 Visualization with PyVista and ȝd

PyVista is a Python library providing high level access to the Visualization Toolkit (VTK) for powerful visualization of complex 3D datasets (Sullivan and Kaszynski, 2019). Here, we showcase its use in conjunction with ȝdfull to generate time-dependent, interactive animations as standalone html-pages; these are easy to embed in web applications and share with colleagues. Compared to creating animations with GUI based software such as ParaView, scripting the visualizations allows regenerating it after changes of the model or input data. Additionally, it gives the freedom to make use of the rich Python ecosystem to manipulate the data. While scripting

3D animations is possible with other packages (such as Mayavi, ParaViewor VTK) PyVistais especially compelling due to its broad VTK-based functionality and its intuitive interface.

As PyVista does not natively support time-dependent datasets, we store the data corresponding to each of the time steps in a new array of a *Pyvista UnstructuredGrid* object:

```
grid = pv.read("mesh/mesh.xdmf")
with meshio.xdmf.TimeSeriesReader("results.xdmf") as reader:
    points, cells = reader.read_points_cells()
    for k in range(reader.num_steps):
        t, point_data, cell_data = reader.read_data(k)
        for var, data in point_data.items():
            grid[f"{var}_{k}"] = data
```

Next, we extract the porous and fluid subdomains and apply the *clip* filter to show the interior of the domain. To visualize the flow field in the CSF-filled domains, we compute arrow glyphs for each time step and scale them with the maximum velocity:

```
fluid = grid.extract_cells(grid["subdomains"] == fluid_id)
fluid_clip = fluid.clip()
arr_max = max([np.linalg.norm(fluid_clip[f"u_{t}"], axis=1).max()
               for t in time_indices])
arrows = []
for t in tqdm(time_indices):
    arr = fluid_clip.glyph(orient=f"u_{t}", scale=f"u_{t}",
                           factor=0.1/arr_max, tolerance=.005)
arrows.append(arr)
```

To use k3d, we convert the PyVista objects into k3d objects and add them to a `k3d.plot`. For that purpose, we use k3d s' `vtk_poly_data` function, generate an initial object and set the time-dependent properties (point/vertex location, color attribute) with a *Python dictionary* mapping the time point. Here, we show demonstrate it for the arrow glyphs:

```
animation_t = np.linspace(0, 10, len(time_idx))
k3d_arrow = k3d.vtk_poly_data(arrows[0],
                              color=hexcolor("white"),
                              side="double")
k3d_arr.vertices = {animation_t[t]:arrows[t].points for t in
                                time_idx}

pl = k3d.plot(...)
pl += k3d_arrows
pl += ...
pl.display()
```

A screenshot of the resulting interactive figure is shown in Fig. 4.4, and the interactive figure is available in the accompanying notebooks Are they going be hosted somewhere?.

4.6 Post-processing aqueduct flow and intracranial pressure

The change of intracranial pressure over the cardiac cycle as well as the aqueduct stroke volume (total volume of CSF moved back and forth through the aqueduct during each cycle) are important clinical indicators of intracranial pulsatility (Wagshul et al., 2011).

Having defined the interface between the fourth ventricle and the aqueduct during the meshing step, we can compute the flow rate as a surface integral over the corresponding internal facets:

```
u_time_series = [...]
dS = Measure("dS", domain=mesh, subdomain_data=boundary_marker)
n = FacetNormal(mesh)
aq_flow = []
for u in u_time_series:
    aq_flow.append(assemble(inner(u,n("+"))*dS(aqueduct_V4_id)))
```

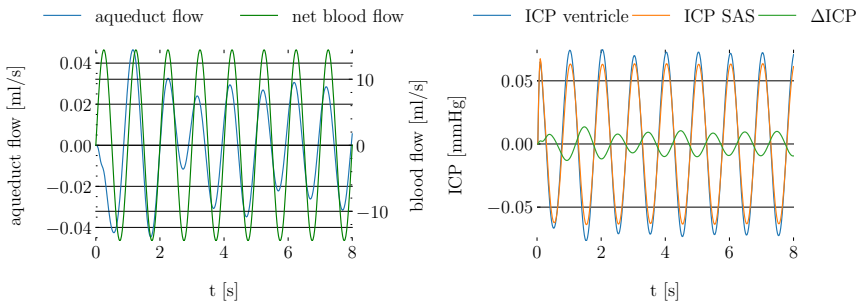
We observe peak aqueduct flow rates of about 0.03 ml/s and after a few cycles, the aqueduct flow synchronizes with the net blood inflow into the brain tissue (Fig. 4.5a).

To compute the ICP and investigate ICP gradients, we place two virtual probes in the CSF-filled spaces: In the middle of the ventricular sphere $x_V = (0, 0, 0)$ and at the outer end of the aqueduct cylinder in the SAS ($x_{AQ} = (0, 0.055, -0.055)$).

```
ICP_time_series =[...]
ventricle_mid = Point(0,0,0)
```

Fig. 4.4 Visualization of the pore pressure in the brain tissue (color coded) and CSF velocities in the CSF-filled spaces. The blue surface in the back represents the outer surface of the CSF-filled spaces.

```
icp_ventricle = [p(ventricle_mid) for p in ICP_time_series]
```



(a) (b)

Fig. 4.5: a) The aqueduct CSF flow over multiple cardiac cycles.; b) ICP measured in the ventricular sphere and in the SAS at the outer end of the aqueduct and the pressure difference between the two virtual probe points (ΔICP).

Compared with clinically reported values for temporal ICP changes of 5 to 10 mmHg, we find relatively small pressure oscillations in both the ventricles and the SAS (less than 0.1 mmHg, Fig 4.5b). This discrepancy is likely caused by the spinal outflow boundary condition: Without additional resistance or an increase of pressure in the spinal SAS, CSF freely leaves the cranium and hence ICP only increases slightly, indicating that accounting for spinal resistance is crucial for physiological ICP values.

Summary and Outlook

In this chapter, we presented a model of the dynamic interplay between blood flow, tissue motion and CSF flow in the human cranium during the cardiac cycle. In addition, we demonstrated crucial technical steps to conduct multiphysics simulations on complex, image derived geometries. Particularly, we showed how to generate volumetric multidomain meshes from surface meshes using fTetWild, use the FEniCS-extension Multiphenicsto set up monolithic forms with function spaces defined only on subdomains, and create dynamic, interactive standalone animations of the obtained results with PyVistaand *K3D*.

However, we only briefly touched upon the models results and its physiological relevance. Exploring it in more depth, Causemann et al. (2022) provides a detailed comparison of clinical measurements and results obtained with a refined version of the same model using an image-derived 3D geometry. Further, Unnerbäck et al. (2018) and Domogo et al. (2023) explored the determinants of the ICP curve mor-

phology, whereas Vinje et al. (2019) investigated the respiratory and cardiac contributions to CSF flow and ICP dynamics. Additionally, a review on the physiology and modelling approaches on CSF dynamics can be found in Linninger, Andreas A. and Tangen, Kevin and Hsu, Chih-Yang and Frim, David (2016). Finally, measuring blood flow, CSF flow oscillations and electrophysiological activity in humans during sleep, Fultz et al. (2019) found important links between CSF dynamics and neural and hemodynamic rhythms, giving rise to multitude of research directions related to modelling of the brain's biomechanics.

References

Alnæs MS, Blechta J, Hake JE, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The fenics project version 1.5. Archive of Numerical Software 3

Balédent O (2014) Imaging of the cerebrospinal fluid circulation, Cambridge University Press, pp 121–138. doi:10.1017/CBO9781139382816.013

Ballarin F (2023) Multiphenics - easy prototyping of multiphysics problems in FEniCS. `https://multiphenics.github.io/`

Beavers GS, Joseph DD (1967) Boundary conditions at a naturally permeable wall. Journal of Fluid Mechanics 30(1):197–207, doi:10.1017/S0022112067001375

Causemann M, Vinje V, Rognes ME (2022) Human intracranial pulsatility during the cardiac cycle: a computational modelling framework. Fluids and Barriers of the CNS 19(1):1–17, doi:10.1186/s12987-022-00376-2

Domogo AA, Reinstrup P, Ottesen JT (2023) Mechanistic-mathematical modeling of intracranial pressure (ICP) profiles over a single heart cycle. The fundament of the ICP curve form. Journal of Theoretical Biology 564:111451, doi:10.1016/j.jtbi.2023.111451

Fultz NE, Bonmassar G, Setsompop K, Stickgold RA, Rosen BR, Polimeni JR, Lewis LD (2019) Coupled electrophysiological, hemodynamic, and cerebrospinal fluid oscillations in human sleep. Science 366(6465):628–631, doi:10.1126/science.aax5440

Hu Y, Schneider T, Wang B, Zorin D, Panozzo D (2020a) Fast Tetrahedral Meshing in the Wild. ACM Trans Graph 39(4), doi:10.1145/3386569.3392385

Hu Y, Schneider T, Wang B, Zorin D, Panozzo D (2020b) Fast Tetrahedral Meshing in the Wild. ACM Trans Graph 39(4), doi:10.1145/3386569.3392385

Lee JJ, Mardal KA, Winther R (2017) Parameter-Robust Discretization and Preconditioning of Biot's Consolidation Model. SIAM Journal on Scientific Computing 39(1):A1–A24, doi:10.1137/15M1029473

Linninger, Andreas A and Tangen, Kevin and Hsu, Chih-Yang and Frim, David (2016) Cerebrospinal fluid mechanics and its coupling to cerebrovascular dynamics. Annual Review of Fluid Mechanics 48(1):219–257, doi:10.1146/annurev-fluid-122414-034321

Oyarzúa R, Ruiz-Baier R (2016) Locking-Free Finite Element Methods for Poroelasticity. SIAM Journal on Numerical Analysis 54(5):2951–2973, doi:10.1137/15M1050082

Ruiz-Baier R, Taffetani M, Westermeyer HD, Yotov I (2022) The Biot-Stokes coupling using total pressure: Formulation, analysis and application to interfacial flow in the eye. Computer Methods in Applied Mechanics and Engineering 389:114384, doi:10.1016/j.cma.2021.114384

Saffman PG (1971) On the Boundary Condition at the Surface of a Porous Medium. Studies in Applied Mathematics 50(2):93–101, doi:10.1002/sapm197150293

Sullivan CB, Kaszynski AA (2019) PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). Journal of Open Source Software 4(37):1450, doi:10.21105/joss.01450

Trzesiok A (2023) K3d jupyter. `https://github.com/K3D-tools/K3D-jupyter`

Unnerbäck M, Ottesen JT, Reinstrup P (2018) ICP curve morphology and intracranial flow-volume changes: a simultaneous ICP and cine phase contrast MRI study in humans. Acta Neurochirurgica 160(2):219–224, doi:10.1007/s00701-017-3435-2

Vinje V, Ringstad G, Lindstrøm EK, Valnes LM, Rognes ME, Eide PK, Mardal KA (2019) Respiratory influence on cerebrospinal fluid flow–a computational study based on long-term intracranial pressure measurements. Scientific reports 9(1):9732

Wagshul ME, Eide PK, Madsen JR (2011) The pulsating brain: A review of experimental and clinical studies of intracranial pulsatility. Fluids and Barriers of the CNS 8(1):1–23, doi:10.1186/2045-8118-8-5

Chapter 5

Quantifying cerebrospinal fluid tracer concentration in the brain

Bastian Zapf, Lars Magnus Valnes, Kent-Andre Mardal and Ludmil Zikatanov

Imaging the dynamics of intrathecally injected cerebrospinal fluid (CSF) tracers has in the recent years led to a new understanding of molecular transport in the brain, both in mice Iliff et al. (2013); Yang et al. (2013) and in humans Ringstad et al. (2017); Eide et al. (2018); Ringstad and Eide (2020). As an illustrative example, the images shown in Fig. 5.1 demonstrate that the CSF tracer reaches all regions of the brain during an imaging period of ∼50 hours.

However, only few studies (Filippi and Watts (2022); Valnes et al. (2020); Vinje et al. (2023)) have so far quantified the concentrations of CSF tracer in the brain. This is partly because the tracer quantification in principle requires special MR sequences that yield MR images known as T_1*–maps* as in Filippi and Watts (2022). Alternatively, estimates of the tracer concentration can be obtained from T_1*–weighted MR images* by using parameters of the sequence used for the MRI acquisition and manually normalizing the images as in Valnes et al. (2020); Vinje et al. (2023).

The difference between these image types is that the voxel values in T_1*–map* images directly correspond to a physical quantity, namely, the (voxel–averaged) longitudinal relaxation time (also called T_1–time) of the atomic nuclei making up the tissue in a given voxel. In T_1–weighted images, the voxel values are related to this quantity in a nonlinear manner defined by the parameters of the MRI acquisition sequence. For a more detailed description, the interested reader is referred to, e.g., Pooley (2005); Taylor et al. (2016).

In this chapter, we describe all steps, assumptions, and software tools needed to estimate CSF tracer concentration in the brain from a time series of either type of images taken after injection.

In detail, we show how to use FreeSurfer Version 7.1.0 (Fischl (2012)) and Python scripts to

- Resample a series of input images to FreeSurfer standard, 256^3 voxels with size $1\text{ mm} \times 1\text{ mm} \times 1\text{ mm}$

- Create a registration template and register all images to the template
- estimate CSF tracer concentration from T_1–maps
- estimate CSF tracer concentration from T_1–weighted images
- Incorporate the tracer estimates into finite element simulations

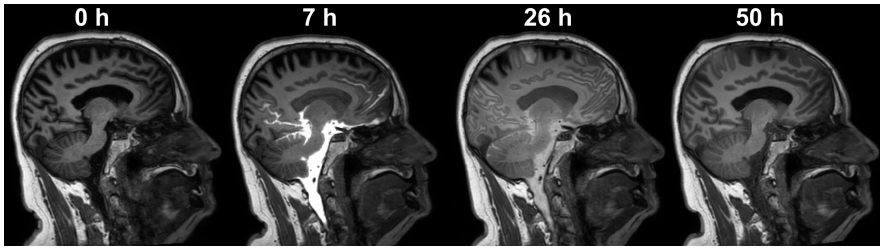


Fig. 5.1: The image shows T_1–weighted MR images taken after intrathecal injection of CSF tracer for the example subject.

5.1 Preliminaries: software configurations for this chapter

For this chapter, we assume that FreeSurfer has been run on the MRI data, and that the baseline and contrast enhanced MR images are extracted from DICOM format into a folder `./data/freesurfer/PREREG/` in `.mgz` format. We further assume that the FreeSurfer output folder `./data/freesurfer/mri/` contains the segmentation file `aseg.mgz`. This data is available for download at[1]. For convenience, the final output files that are being created when performing all the steps in this chapter (CSF tracer concentrations, finite element mesh) can be downloaded at[2]. The scripts used in this chapter can be obtained from the repository[3] by opening a terminal window and running

Terminal

```
$ git clone https://github.com/bzapf/tracerdiffusion.git
```

In order to run the Python scripts accompanying this chapter, we will create a conda environment with the required packages installed as

[1] `https://github.com/bzapf/rawdata`

[2] `https://github.com/bzapf/concentrationdata`

[3] `https://github.com/bzapf/tracerdiffusion`

Terminal

```
$ cd tracerdiffusion
$ conda env create -f fenics-env.yml
$ conda activate diffusion-fenics-env
$ pip install -e .
```

For the registration scripts, we will define the path to the data in a variable `WORKDIR` by running

Terminal

```
$ export WORKDIR=./data/freesurfer/
```

before executing the bash script.

5.2 Registration of a MRI time series

Since the position of the subjects in the MRI machine differs between acquisitions, we need to align all the images so that it is meaningful to compare intensities in some voxel with indices (ijk) between images taken at different time points. This process is called registration, and FreeSurfer provides the program `mri_robust_register`[4] Reuter et al. (2010) to perform this task. The algorithm finds the affine, i.e. rigid translation and rotation, transformation that, when applied to a given input image, minimizes the mismatch to a given target image. We limit our selves to affine maps in this setting since it is the same subject imaged over a short timespan.

In principle, this algorithm could be used to register all images in a time series to a baseline image (pre-constrast). However, the CSF tracer leads to significantly increased MR signal in most of the cranial compartment as compared to baseline. Whilst the FreeSurfer registration algorithm Reuter et al. (2010) uses normalization to account for signal differences between input and target image during registration, the unusual signal differences induced by the CSF tracer can cause problems for the registration. We hence create an unbiased registration template, which reduces biases in longitudinal studies Reuter et al. (2012). This template is built from a voxel-wise median of all the images in the time series, registered such that all the images can be registered as good as possible to this template.

[4] `https://surfer.nmr.mgh.harvard.edu/fswiki/mri_robust_register`

First, however, we resample all the images to FreeSurfer standard format (256^3 voxels) because some MR images can have higher resolution, significantly increasing the computational time of the registration algorithm while others may have lower resolution.

We collect all the necessary commands for the registration in a single script which can be found under `./scripts/data-processing/register.sh` and can be called as

Terminal

```
$ bash ./scripts/data-processing/register.sh
```

We next give a detailed description of the crucial steps in this script.

To start, we loop over all images in the input folder that are of `mgz` format, and use FreeSurfer's program `mri_convert`[5] to resample the images to FreeSurfer standard:

register.sh

```
for inputfile in ${INPUTFOLDER}/*.mgz; do
     filename=$(basename $inputfile)
     outputfile=${OUTPUTFOLDER}/${filename}
     mri_convert --conform -odt float ${inputfile} ${outputfile}
done
```

Here, we use the shell command `basename` to extract the filename from the variable `inputfile` which contains the full path to the image. With the option `--conform`, the images are resampled to FreeSurfer standard voxel size 1 mm^3. The flag `-odt` is shorthand for `--out_data_type`. It is needed since the voxel intensities are not integer values after resampling.

Now, the FreeSurfer command `mri_robust_template`[6] can be used to create the registration template. This program computes a median image of all input images to avoid any bias towards a specific image. This is an iterative process in which several registrations have to be computed, and hence the program may take several minutes to finish.

The command is used as

[5] https://surfer.nmr.mgh.harvard.edu/fswiki/mri_convert

[6] https://surfer.nmr.mgh.harvard.edu/fswiki/mri_robust_template

register.sh

```
mri_robust_template --mov ${RESAMPLED_IMAGES} --maxit 10 \
 --template ${TEMPLATE} --satit --inittp 1 --fixtp
```

The `--satit` tells FreeSurfer to automatically determine how outlier voxels should be treated during registration. With `--inittp 1`, the algorithm uses the first image (pre-contrast) as initial value for the template, and with `--fixtp`, all other images are registered to this image. This is useful since we have ran the FreeSurfer command `recon-all` (cf. Volume 1, Section 3.1.2) with the baseline image (pre-contrast), so all the segmentation files and surfaces generated by FreeSurfer align with this image. With `--inittp 1 --fixtp`, the registration template is automatically aligned with the segmentation and surface files.

Registration is an iterative process, and with `--maxit 10` we specify the maximum number of iterations. Setting it higher than the default of 6 iterations can increase the accuracy of the registration.

Finally, it should be noted that this program can require more memory than available on personal computers if there are many images in the time series. In this case, the flag `--subsample 200` can be used to resample the images to a resolution of 200^3 voxels (or less if necessary). This reduces the memory required by the program, enabling us to run it on usual laptops.

We can now register all the images to the template using FreeSurfers `mri_robust_register` program:

register.sh

```
for inputfile in ${INPUTFOLDER}/*.mgz; do
 filename=$(basename $inputfile)
 mri_robust_register --mov ${inputfile} --dst ${TEMPLATE} \
 --mapmov ${OUTPUTFOLDER}/${filename} --maxit 10 \
 --lta ./data/freesurfer/LTA/${filename}.lta --iscale --satit
done
```

This command aligns the input image `--mov ${inputfile}` with the target image `--dst ${TEMPLATE}` and stores it to `--mapmov ${OUTPUTFOLDER}/${filename}`. With the argument `--lta` we specify a storage path to store the affine registration matrix, which may be useful for later analysis.

In order to visually check the registration results, the script finished by opening all images in FreeView to check if they are well aligned to the template by the command

Terminal window

```
$ freeview ${OUTPUTFOLDER}/*
```

It is strongly recommend to always inspect the registration output visually to ensure that the images are properly registered to the baseline image.

5.3 Quantifying tracer from contrast-enhanced MRI

The presence of a CSF tracer leads to a decrease of the spin-lattice relaxation time T_1 in enriched regions. In T_1–weighted images, this increases the MR signal in enriched voxels and makes them appear brighter on the image.

The relaxation time $T_1((i,j,k),t)$ in a voxel (i,j,k) at time t after CSF tracer injection decreases with contrast agent concentration $c((i,j,k),t)$ (mmol/L) as (see, e.g., Taylor et al. (2016))

$$\frac{1}{T_1((i,j,k),t)} = \frac{1}{T_1((i,j,k),0)} + r_1 c((i,j,k),t) \tag{5.1}$$

where $T_1((i,j,k),0)$ denotes the T_1 relaxation time at baseline (before injection). The *relaxivity constant* r_1 differs between tracer and depends on other factors such as the temperature, and magnetic field strength. The imaging sequence under consideration here was acquired using the CSF tracer gadobutrol at magnetic field strength 3 T. For these parameters, we can use the measurements value $r_1 = 3.2\,\mathrm{L/(mmol\ s)}$ in water at 37 °C from Rohrer et al. (2005).

5.3.1 Quantifying tracer from T_1–maps

Rearranging (5.1) shows that the concentration of the tracer in every voxel is

$$c((i,j,k),t) = \frac{1}{r_1}\left(\frac{1}{T_1((i,j,k),t)} - \frac{1}{T_1((i,j,k),0)}\right) \tag{5.2}$$

if measurements of T_1 are available at all time points as in Watts et al. (2019). In this case, assuming that the files in `./data/freesurfer/REGISTERED` are T_1–maps, we can easily compute the concentration with the Python script `estimatec.py` as

Terminal window

```
$ python ./scripts/data-processing/estimatec.py \
    --inputfolder ./data/freesurfer/REGISTERED/ \
    --exportfolder ./data/freesurfer/CONCENTRATIONS/
```

By default, the script assumes that the files are named following the convention `YYYYMMDD_HHMMSS.mgz`, and finds the baseline map $T_1((i,j,k),0)$ by sorting the files in the input folder. A confirmation prompt asks the user to confirm this selection, and if it is not correct, the baseline can manually be specified with the optional argument `--baseline`. For convenience, the script will rename the created files to the format `HH.MM.mgz` where `HH` and `MM` are the number of hours and minutes after the baseline image was taken, respectively.

However, T_1–maps are acquired with MRI protocols that are not always included in clinical CSF tracer studies with humans, e.g., Vinje et al. (2023). As a remedy, in the next section we present estimation of the tracer concentration from T_1–weighted images since these are typically available in such studies.

5.3.2 Quantifying tracer from T_1–weighted images

Despite the name T_1-*weighted image*, the MR signal $S((i,j,k),t)$ in a voxel (i,j,k) of this image is no quantitative measure of the relaxation time T_1. The relation between MR signal $S((i,j,k),t)$ and relaxation time $T_1((i,j,k),t)$ is nonlinear and depends on the details of MRI protocol used in the image acquisition. As a remedy, we describe how to proceed as in Vinje et al. (2023) and estimate the relaxation times $T_1((i,j,k),t)$ after tracer injection from normalized T_1–weighted voxel intensities $S((i,j,k),t)$ relative to the baseline image. It is first shown how to normalize the images using a manually defined region of interest. We further need the baseline T_1–map $T_1((i,j,k),0)$. In the case that this is not available, we start by showing how to create a synthetic baseline map $T_1((i,j,k),0)$ based on literature values.

5.3.2.1 Defining a normalization region using freeview

Due to variation in the MRI machine over time, the T_1–weighted MR signal may change over time even in the brain regions where no physical change in the tissue is expected to occur. In order to compensate for this effect, we use a region of interest (ROI) in the brain where the tissue does not change during the imaging period. We then normalize every image with the median MR signal in this region.

To create a ROI, we start by opening the baseline image (the first image in the folder `./data/freesurfer/REGISTERED/`) in FreeView:

Terminal window

```
$ freeview ./data/freesurfer/REGISTERED/20230213_073508.mgz
```

We choose the fat in the posterior part of the orbit (behind the eye) as it can be assumed that this tissue is not enriched by CSF tracer and does not undergo changes during the imaging period of ~ 3 days Eide et al. (2021). First, we move the red, green and blue planes in the 3D window in freeview such that they all intersect with the fat tissue (cf. 3D panel in the right of Fig. 5.2). For the MRI under consideration, locating the cursor at voxel indices (92, 128, 168) as in Fig. 5.2 is a possible choice since there all planes cut through the posterior part of the orbit.

To start marking the reference ROI, click File→New Volume, keep the default options and name the new volume "refroi". After this empty volume is created, the MR image is overcast, and we change the color map of the volume "refroi" to "Heat" or "Jet" with "Maximum" set to 1 such that the ROI can be displayed on top of the MR image. We next click Action→Voxel edit and set the "Brush value" to 1. In every slice window, we mark a few white, bright voxels in the posterior part of the orbit (fat behind the eye), see Fig. 5.2. Note that the ROI should not cover nerve fibers, visible in black close to the ROI and indicated by the blue arrows in Fig. 5.2. Finally, the volume can be saved to `./data/freesurfer/mri/refroi.mgz` by clicking File→Save Volume.

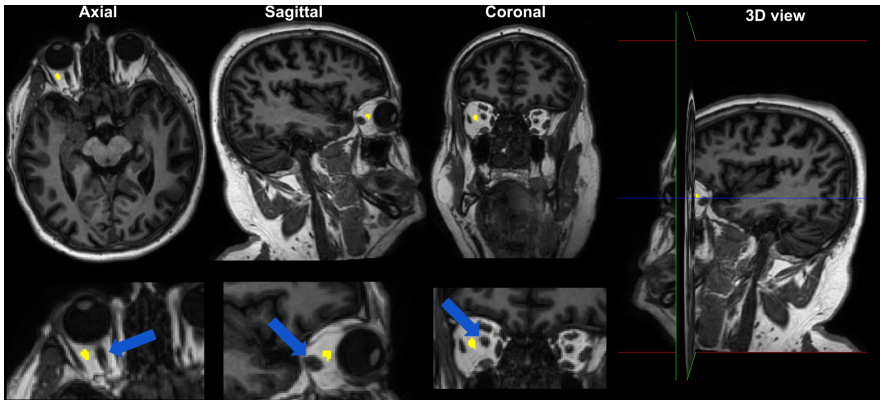


Fig. 5.2: The image shows the normalization region (ROI) in the posterior part of the orbit in yellow together with the baseline image (visualized using Freeview). The blue arrows indicate the optical nerve which we do not want to include in the ROI.

5.3.2.2 Normalization

Using the ROI created in the subsection above, we normalize every image with the median MR signal in this ROI as

$$S((i,j,k),t) = \frac{I((i,j,k),t)}{\text{median}_{(a,b,c)\in \texttt{ROI}}\{I((a,b,c),t)\}} \tag{5.3}$$

where $I((i,j,k),t)$ denotes the T_1–weighted image signal in voxel (i,j,k) at time t and $S((i,j,k),t)$ is the normalized T_1–weighted signal. Equation (5.3) is implemented in a Python script which can be used to normalize all images at times t in an automated fashion. This procedure requires the input data, an output folder and the path to the ROI, and we specify these paths using an `ArgumentParser` as command line arguments. The full script can be called as

Terminal window

```
$ python ./scripts/data-processing/normalize.py \
    --inputfolder ./data/freesurfer/REGISTERED/ \
    --exportfolder ./data/freesurfer/NORMALIZED_CONFORM/ \
    --refroi ./data/freesurfer/mri/refroi.mgz
```

With the ROI stored under `./data/freesurfer/mri/refroi.mgz`, this script also tells us that the median T_1–weighted MR signal value varies from 90157 at baseline to 75767, 74486, 71435, 69087 at the later imaging timepoints. These variations demonstrate the necessity of normalizing the images.

5.3.2.3 Creating brain masks and synthetic T_1–maps

As a first approximation, we assume that the relaxation time T_1 is constant in the whole brain parenchyma. In order to define which image voxels are considered as belonging to the brain, we use the FreeSurfer segmentation output file found under `./data/freesurfer/mri/aseg.mgz`. The segmented regions are shown in color with the baseline T_1–weighted image in the left of Fig. 5.3. The integer voxel values in this image indicate the belonging of the voxels to (in this subject 44) distinct brain regions. The correspondence between labels and anatomical regions can be found in the FreeSurfer documentation[7] where we can see that the labels 4, 5, 14, 15, 24, 43 and 44 identify CSF compartments such as the ventricles. Since all other nonzero labels denote brain tissue, we can create a binary mask for the brain parenchyma by identifying which of the voxels are labeled with one of these numbers. We assign the

[7] `https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/AnatomicalROI/FreeSurferColorLUT`

value 0 to these voxels, and all other labels we set to 1 to create a binary mask for the parenchyma tissue. In the same way, we can also create a synthetic T_1–map by assigning a surrogate value for T_1 taken from the literature to the voxels not labeled as CSF. We implement this in the `Python` script `make_brainmask.py` which can be called as

Terminal window

```
$ python ./scripts/data-processing/make_brainmask.py \
    --aseg ./data/freesurfer/mri/aseg.mgz \
    --maskname ./data/freesurfer/mri/parenchyma_mask.mgz \
    --t1 1 --t1mapname ./data/freesurfer/mri/synth_T1_map.mgz
```

to store a brain mask and a synthetic T_1–map with constant value $T_1 = 1$ s throughout the brain to `./data/freesurfer/mri/`.

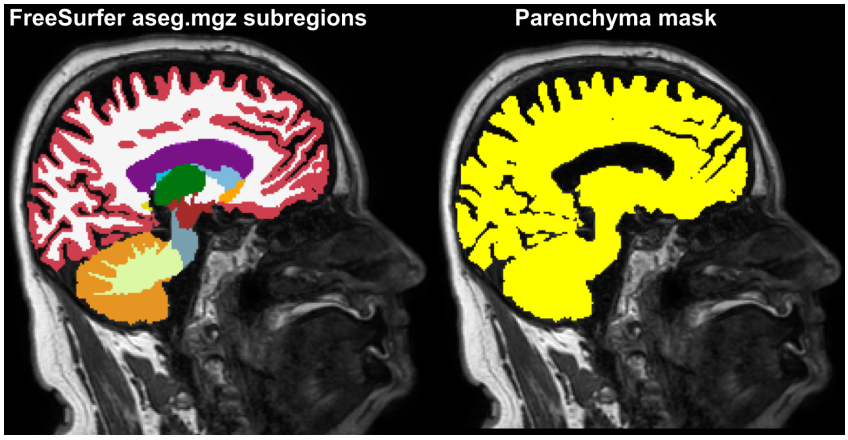


Fig. 5.3: Left: FreeSurfer segmentation (colored) of the brain. Right: binary mask (yellow) created by excluding voxels labeled as CSF such as the lateral ventricle (shown in purple in the left image).

5.3.2.4 Estimating T_1 from T_1–weighted images

With the synthetic baseline T_1–map we have created in Section 5.3.2.3, we have everything that is needed to estimate the unknown relaxation time $T_1((i, j, k), t)$ at imaging times t after intrathecal tracer injection. Following Valnes et al. (2020), we use the relation

$$T_1((i,j,k),t) = f^{-1}\left(\frac{S((i,j,k),t)}{S((i,j,k),0)} f(T_1((i,j,k),0))\right). \tag{5.4}$$

Here, $S(i,0)$ denotes the normalized MR signal intensity before tracer injection, and the function f is specific to the MRI acquisition protocol used. The T_1–weighted MR images under consideration here are obtained using the Magnetization Prepared Rapid Acquisition Gradient Echo (MPRAGE) sequence, for which we can use the approximation $f(T_1) = A\exp(-bT_1)$ for $T_1 \in [0.2\,\mathrm{s}, 4.5\,\mathrm{s}]$ with $A = 0.95, b = 1.48\,\mathrm{s}^{-1}$ [8]. With this approximation, we can simplify (5.4) to

$$T_1((i,j,k),t) = T_1((i,j,k),0) - \frac{1}{b}\log\frac{S((i,j,k),t)}{S((i,j,k),0)}. \tag{5.5}$$

Since we are primarily interested in computing the CSF tracer concentration, we do not compute and store the T_1 estimates separately, but rather evaluate (5.5) as a step in our concentration computation in the next section.

5.3.2.5 Putting it all together: computing concentration from T_1–weighted images

We combine (5.2) and (5.5) into a Python function that efficiently computes the concentration in all image voxels using `numpy`. However, care needs to be taken since computing (5.5) may lead to unreasonably high or low estimates of T_1 due to noise in the T_1–weighted MRI. As in Valnes et al. (2020), we consider T_1 values outside the interval $[0.2, 4.5]$ s to be unreasonable and a consequence of noise, and threshold higher/lower values to the upper/lower interval bound, respectively.

The full script is found under `./scripts/data-processing/estimatec.py` and can be called as

Terminal window

```
$ python ./scripts/data-processing/estimatec.py \
--inputfolder ./data/freesurfer/NORMALIZED_CONFORM/ \
--exportfolder ./data/freesurfer/CONCENTRATIONS/ \
--t1map ./data/freesurfer/mri/synth_T1_map.mgz \
--mask ./data/freesurfer/mri/parenchyma_mask.mgz
```

Note that without the flagged argument `--t1map`, it is assumed that the images in the directory specified with `--inputfolder` are T_1–maps, and the estimation step

[8] These numerical values for A, b are obtained by fitting $f(T)$ for $T \in [0.2\,\mathrm{s}, 4.5\,\mathrm{s}]$ to Supplementary Eq. (S11) in Valnes et al. (2020).

(5.5) is omitted in the concentration computation. With `--mask` we provide a binary mask for the brain tissue, and the concentration is computed only in the masked voxels. This is useful for visualization as in Fig. 5.4. Further, specifying a mask can be necessary if the T_1 values are subject to large uncertainties in some regions of the brain, and quantifying tracer within these regions is not feasible[9].

If a baseline T_1–map is available for the subject under consideration, running the script with a different output folder name and setting `--t1map ./data/freesurfer/T1MAP/T1_map.mgz` allows us to compute the concentration without assuming a constant T_1 value, which is, arguably, a rough approximation. Figure 5.4 compares the normalized MR signal to the CSF tracer concentration as computed with the constant T_1 value for three timepoints after injection.

5.4 MRI-informed tracer modeling

We now ask how well the brain-wide CSF tracer distribution we have computed in the previous section can be described by physics-based models in the form of partial differential equations (PDE). To this end, we revisit the model for molecular dynamics in the brain introduced in Volume 1, Section 1 (Mardal et al. (2022)). We simplify the model to include solute transport by diffusion only,

$$u_t - \operatorname{div} D \nabla u = 0 \qquad \text{in } (0,T] \times \Omega, \tag{5.6a}$$

$$u = u_d \qquad \text{on } (0,T] \times \partial\Omega, \tag{5.6b}$$

$$u(0,\cdot) = u_0 \qquad \text{in } \Omega. \tag{5.6c}$$

Here, u_0 denotes the initial condition, u_d the boundary conditions and D is the diffusion tensor for the CSF tracer. If a baseline diffusion tensor image (DTI) is available, it can be used to estimate the diffusion tensor of the tracer as described in Valnes et al. (2020). Here, we will make a simplifying assumption and use a scalar diffusion coefficient $D_w = 10^{-3}\,\mathrm{mm^2 s^{-1}}$ for water in the brain, which corresponds to $D = 1.3 \times 10^{-4}\,\mathrm{mm^2 s^{-1}}$ for the CSF tracer gadobutrol Valnes et al. (2020).

Solving these equations then requires to specify both u_0 and u_d. In Volume 1, Section 6, it was shown how to solve these equations with the modeling assumptions $u_0 = 0$ and $u_d = 1$ with the domain Ω being the full left hemisphere of the brain. Using the concentration data obtained in the previous section, we can now improve upon this assumptions and define the boundary condition as a linear interpolation between image acquisitions,

[9] For the T_1–map under consideration here, an example are the CSF compartments. The T_1 used mapping sequence, MOLLI5(3)3 Taylor et al. (2016), was calibrated to yield accurate T_1–values in the brain, yielding less reliable values in the CSF.

$$u_d(x,t) = c(x,t_i) + \frac{c(x,t_{i+1}) - c(x,t_i)}{t_{i+1} - t_i}(t - t_i) \quad \text{for} \quad t_i \le t \le t_{i+1} \tag{5.7}$$

where $t_i = \{0, ..., T\}$ are the image acquisition times (relative to the first image after injection) and $c(x, t_i)$ is the CSF tracer concentration. Here, we consider a time span from $t = 0$ (image before injection) to the image taken at $T \approx 2$ days after injection. In this case, the initial condition is simply $u_0 = 0$ in Ω.

We use the finite element mesh found under `./data/freesurfer/meshes/lh.xml`. This mesh has been generated as described in Volume 1, Chapter 4[10].

Since we want to reuse the simulation code in Bastian Zapf (2023), it is beneficial to take an Object-oriented programming approach and wrap the simulation code into a class. We call this class `Model`, and the numerical simulation of (5.6) can be computed by the class function `Model.forward()`.

The boundary condition (5.7) is implemented in the class method `Model.boundary_condition()`. Therein, the finite element functions describing the measured concentrations $c(x, t_{i+1}), c(x, t_i)$ are obtained by transforming between mesh coordinates (RAS space) and voxel indices (T1 space). A brief description of how this is done in Python is given in section 5.4.1. The reverse operation of mapping arbitrary FEniCS functions defined on a brain mesh back to MRI volume format is described in Section 5.4.2.

The simulation can now be run by

Terminal window

```
$ python ./scripts/forward-model/diffusion.py \
    --data ./data/freesurfer/CONCENTRATIONS/ \
    --mesh ./data/freesurfer/meshes/lh.xml
```

By default, this script uses all images within the first 3 days after injection as boundary condition to perform the simulations. After the simulation is done, the result can be inspected by running

Terminal window

```
$ paraview simulation_outputs/movie.pvd
```

[10] For convenience, we provide the scripts used to generate this mesh under `./scripts/mesh-generation/`. The mesh can be re–generated by executing the command `bash ./scripts/mesh-generation/make-mesh.sh`.

The stored FEniCS functions $c(x, t_i)$ at imaging times i can be converted to MRI format as described in Sec. 5.4.2. A sagittal slice of the simulated concentration is shown together with the data in Fig. 5.4. Whilst the simulation underestimates the data at $\sim 7\,\text{h}$, the model overestimates the concentration significantly after $\sim 26\,\text{h}$. This suggests that diffusion as a sole transport mechanism is insufficient to explain the data, and we investigate more detailed models in Bastian Zapf (2023).

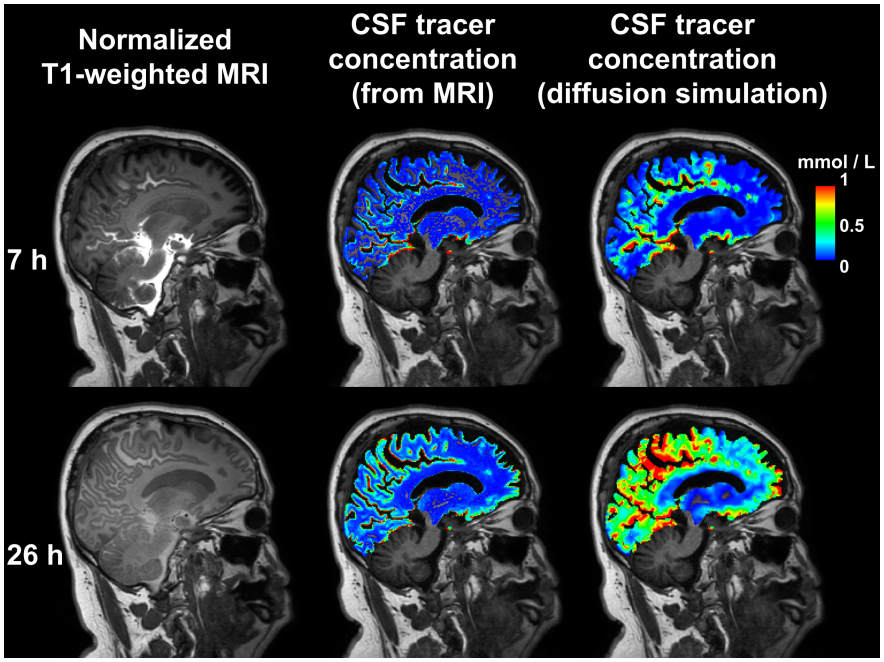


Fig. 5.4: The image shows from left to the right: T_1–weighted MRI, concentration of CSF tracer in the left hemisphere as computed with synthetic T_1–Map, and concentration as simulated using the diffusion equation and subject-specific boundary conditions, for different time points after intrathecal injection of CSF tracer. Visualized using freeview.

5.4.1 Mapping images to FEniCS functions

Scalar fields (such as the computed concentrations) given as a MRI volume can be represented as FEniCS functions similar as presented for diffusion tensor images in Volume 1, Section 4.4.1 Mardal et al. (2022). For convenience, we provide this functionality in a Python function `read_image` in the script `mri2fenics.py` to per-

form this task. While the actual implementation contains additional data filtering[11], a basic implementation for reading MRI to FEniCS functions would look like:

```
def read_image(filename,functionspace):
  mri_volume = nibabel.load(filename)
  voxeldata = mri_volume.get_fdata()
  c_data = Function(functionspace)
  vox2ras = mri_volume.header.get_vox2ras_tkr()
  ras2vox = numpy.linalg.inv(ras2vox)
  xyz = functionspace.tabulate_dof_coordinates()
  ijk = apply_affine(ras2vox, xyz).T
  i, j, k = numpy.rint(ijk).astype("int")
  c_data.vector()[:] = voxeldata[i, j, k]
  return c_data
```

The script can be called from the command line to convert an image:

Terminal window

```
$ python ./examples/mri2fenics.py \
    --mesh ./data/freesurfer/meshes/lh.xml \
    --data ./data/freesurfer/CONCENTRATIONS/26.05.mgz
```

By default, the script will store the FEniCS function in various formats to the same location as the data, but a different path can be specified with the optional argument `--outputname`. For example, the file `./data/freesurfer/CONCENTRATIONS/26.05.pvd` can be used for visualization in paraview.

5.4.2 Mapping FEniCS functions to images

For visualizations of simulation results such as in Fig. 5.4, it can be useful to "voxelize" a FEniCS function, i.e., fill an empty MRI volume with the values of the FEniCS function. By default, the simulation code presented in Sec. 5.4 stores the simulated tracer concentration at every imaging time into a file `simulation.hdf`. We provide a Python script `fenics2mri.py` which can be called as

[11] As pointed out above in Section 5.3.2.5, the CSF tracer concentration can sometimes only be computed in specific regions of the brain defined by binary voxel masks. Since the FreeSurfer surfaces and hence the meshes have sub–voxel resolution, vertices of a mesh describing a brain region may correspond to voxels outside the binary mask describing the same anatomical region. For this case, our implementation of `read_image` accepts a binary mask as `.mgz` file via the optional argument `mask`. Node values outside the mask will then be replaced by the median voxel values in adjacent voxels inside the mask.

Terminal window

```
$ python ./examples/fenics2mri.py \
    --mesh ./data/freesurfer/meshes/lh.xml \
    --image ./data/freesurfer/mri/aseg.mgz \
    --hdf5_file ./simulation_outputs/simulation.hdf \
    --hdf5_name simulation26 \
    --output ./simulation_outputs/26h.mgz
```

to store the simulated tracer at 26 h after baseline as a MRI volume. To load the FEM prediction from the `hdf` file[12], we use `--hdf5_name simulation26` to assess the FEM solution stored under the key `simulation26` by the script `diffusion.py`.

This script iterates over all voxel indices with RAS coordinates that may be in the mesh, and evaluates the FEniCS function in these points. The script hence requires a few minutes to run for full brain meshes. To speed up this process, the script has the optional argument `--mask` to specify in which voxels to evaluate the FEniCS function. In our case, with `--mask ./data/freesurfer/mri/parenchyma_mask.mgz` the script only evaluates the voxel that are inside the brain parenchyma. This reduces the runtime of the script by 30 %, but the speed up is even more significant when looking at smaller subregions of the brain. For further technical details we refer to the documentation of the method in `./examples/fenics2mri.py`, and note that we also provide a script `./examples/voxelize-mesh.py` which can be used to obtain a binary mask in `.mgz` format from volume meshes.

5.5 Acknowledgments

We would like to thank Yngve Mardal Moe for providing a script on which the accompanying script `fenics2mri.py` is based. We also thank Miroslav Kuchta for careful proofreading of the Chapter.

References

Bastian Zapf KAM Marius Zeinhofer (2023) Inverse transport parameter estimation. In: Rognes ME, Vinje V, Dokken JS, Valnes LM, Mardal KA (eds) MRI2FEM II: from magnetic resonance images to computational brain mechanics, Springer

[12] To check the context of hdf files, the Python package h5py (`https://docs.h5py.org/en/stable/quick.html`) is very useful, see the script `./examples/fenics2mri.py` for an example.

Eide PK, Vatnehol SAS, Emblem KE, Ringstad G (2018) Magnetic resonance imaging provides evidence of glymphatic drainage from human brain to cervical lymph nodes. Scientific Reports 8(1):1–10, doi:10.1038/s41598-018-25666-4

Eide PK, Vinje V, Pripp AH, Mardal KA, Ringstad G (2021) Sleep deprivation impairs molecular clearance from the human brain. Brain 144(3):863–874, doi:10.1093/brain/awaa443

Filippi CG, Watts R (2022) Imaging of Glymphatic Flow and Neurodegeneration, Springer International Publishing, Cham, pp 849–860. doi:10.1007/978-3-030-82367-2_71

Fischl B (2012) FreeSurfer. NeuroImage 62(2):774–781, doi:10.1016/j.neuroimage.2012.01.021

Iliff JJ, Lee H, Yu M, Feng T, Logan J, Nedergaard M, Benveniste H, et al. (2013) Brain-wide pathway for waste clearance captured by contrast-enhanced MRI. The Journal of clinical investigation 123(3):1299–1309, doi:10.1172/JCI67677

Mardal KA, Rognes ME, Thompson TB, Valnes LM (2022) Mathematical Modeling of the Human Brain: From Magnetic Resonance Images to Finite Element Simulation. Springer Nature, doi:10.1007/978-3-030-95136-8

Pooley RA (2005) Fundamental Physics of MR Imaging. RadioGraphics 25(4):1087–1099, doi:10.1148/rg.254055027

Reuter M, Rosas HD, Fischl B (2010) Highly accurate inverse consistent registration: A robust approach. NeuroImage 53(4):1181–1196, doi:10.1016/j.neuroimage.2010.07.020

Reuter M, Schmansky NJ, Rosas HD, Fischl B (2012) Within-subject template estimation for unbiased longitudinal image analysis. NeuroImage 61(4):1402–1418, doi:10.1016/j.neuroimage.2012.02.084

Ringstad G, Eide PK (2020) Cerebrospinal fluid tracer efflux to parasagittal dura in humans. Nature Communications 11(1):354, doi:10.1038/s41467-019-14195-x

Ringstad G, Vatnehol SAS, Eide PK (2017) Glymphatic MRI in idiopathic normal pressure hydrocephalus. Brain 140(10):2691–2705, doi:10.1093/brain/awx191

Rohrer M, Bauer H, Mintorovitch J, Requardt M, Weinmann HJ (2005) Comparison of Magnetic Properties of MRI Contrast Media Solutions at Different Magnetic Field Strengths. Investigative Radiology 40(11):715–724, doi:10.1097/01.rli.0000184756.66360.d3

Taylor AJ, Salerno M, Dharmakumar R, Jerosch-Herold M (2016) T1 Mapping: Basics Techniques and Clinical Applications. JACC: Cardiovascular Imaging 9(1):67–81, doi:10.1016/j.jcmg.2015.11.005

Valnes LM, Mitusch SK, Ringstad G, Eide PK, Funke SW, Mardal KA (2020) Apparent diffusion coefficient estimates based on 24 hours tracer movement support glymphatic transport in human cerebral cortex. Scientific Reports 10(1):1–12, doi:10.1038/s41598-020-66042-5

Vinje V, Zapf B, Ringstad G, Eide PK, Rognes ME, Mardal KA (2023) Human brain solute transport quantified by glymphatic MRI-informed biophysics during sleep and sleep deprivation. bioRxiv doi:10.1101/2023.01.01.522190

Watts R, Steinklein J, Waldman L, Zhou X, Filippi C (2019) Measuring Glymphatic Flow in Man Using Quantitative Contrast-Enhanced MRI. American Journal of Neuroradiology 40(4):648–651, doi:10.3174/ajnr.A5931

Yang L, Kress BT, Weber HJ, Thiyagarajan M, Wang B, Deane R, Benveniste H, Iliff JJ, Nedergaard M (2013) Evaluating glymphatic pathway function utilizing clinically relevant intrathecal infusion of CSF tracer. Journal of Translational Medicine 11(1):1–9, doi:10.1186/1479-5876-11-107

Chapter 6
Tracer Concentration Prediction with CNNs

Marius Zeinhofer and Kent-Andre Mardal

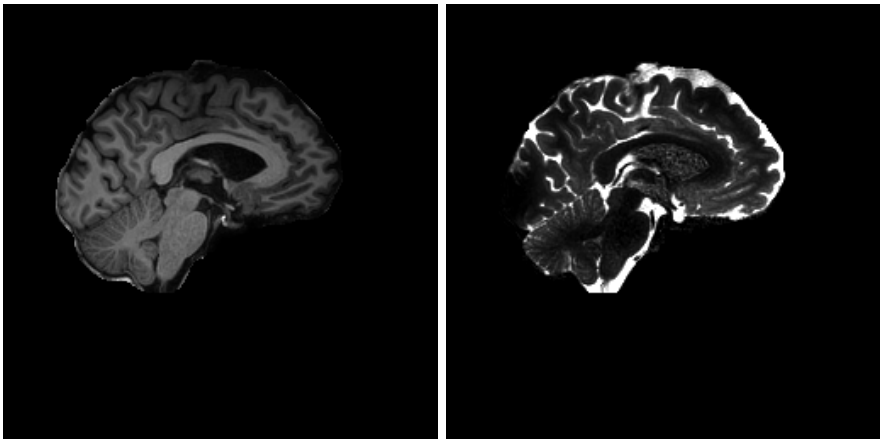
Abstract We present a supervised deep learning pipeline to predict signal increase ratio (SIR) in the cerebro-spinal fluid (CSF) of the human brain 24h after intrathecal contrast enhancement agent injection. The section serves as a brief introduction to deep learning for medical image analysis and is accompanied by an examplary implementation available as a Github repository at `https://github.com/MariusZeinhofer/tracer_preds`.

6.1 Introduction

Waste clearance in the human brain is a complex process and malfunctioning clearance is associated with various pathologies such as Alzheimer's disease. While it is commonly accepted that the CSF plays a crucial role in waste clearance, the precise underlying principles are the topic of active research Rasmussen et al. (2018). One way to investigate clearance processes in the CSF is to intrathecally inject a MRI contrast agent – for example gadobutrol – and to subsequently perform several MR scans. One can use the voxel-wise signal increase ratio (SIR) to visualize tracer concentration over time to study the clearance process. However, this is time and resource consuming and not free of side effects.

Hence, in this chapter we present a deep-learning based predictor that, given brain MR data without contrast-enhancing tracer, predicts the SIR 24h after injection. More precisely, we train a neural network on pairs of 136 T1 weighted human brain MR scans that all provide examples of a tracer free scan as the input to the neural network and the SIR at 24h computed using a second scan at 24h. We assume the MRI data is registered and anonymized but all other preprocessing is described in detail in the

Marius Zeinhofer e-mail: `mariusz@simula.no`
Simula Research Laboratory, Norway



(a) Baseline (b) Tracer after 24h

Fig. 6.1: A examplary training pair.

following sections. This chapter also serves as a *brief* introduction to deep-learning based medical image analysis and is accompanied by a Github repository available at `https://github.com/MariusZeinhofer/tracer_preds`. A typical training pair can be seen in Figure 6.1.

6.2 Deep Learning Preliminaries

In this section we give an introduction to convolutional neural networks (CNNs) and neural network training using stochastic gradient descent (SGD).

6.2.1 Convolutional Neural Networks

In general neural networks are parametric functions that, given a set of trainable parameters θ, can process input X to produce output Y, we write

$$Y = \mathcal{NN}_\theta(X).$$

In computer vision tasks, X is typically an image or volumetric data represented as a matrix or tensor[1] and the shape of Y depends on the task; in our case Y is itself an image that contains voxel-wise SIR values. The notation θ for the parameters is an

[1] In deep learning terminology, a tensor is a multidimensional array and thus generalizes scalars, vectors and matrices.

abbreviation and θ consists of a list of all trainable weights of the network, typically a collection of weight matrices and bias vectors. The structure of neural networks is highly problem dependent and specialized architectures have been developed for specific applications.

Convolutional neural networks (CNNs) are a network architecture popular for computer vision tasks such as image classification and segmentation. The core idea of CNNs is to exploit invariance properties typically presented in natural images.

This can be explained more precisely by looking into how the convolution operation – the decisive building block of CNNs – works. Convolutions act on images and are *linear, local* and *translation invariant* operations. In the simplest case, given a grey-scale image represented as a matrix X a convolution operates on X via

$$H_{ij} = \sum_{k,l=-\nu}^{\nu} W_{k,l} X_{i+k,j+l} + b,$$

where the learnable parameters are the scalar bias b and the matrix W which is referred to as the convolutional kernel or filter. The number $\nu \in \mathbb{N}$ is the kernel width and the matrix H is the result of the convolution. Note that the locality of the convolution is embodied by the choice of a small ν and the translation invariance by the fact that W does not depend on i, j. Convolutions are easily extended to three dimensional data or colored images.

To transition from local information to image wide context, several convolutions are daisy-chained, with intermediate nonlinear, up-and downsampling layers. Convolutions do not comprise the only building blocks of CNNs, but they convey the main intuition of the models: First, local, translation invariant information is aggregated and subsequently combined to produce image wide context. For a thorough introduction to CNNs we refert to Zhang et al. (2021).

For our image segmentation example, we use the U-net architecture that was originally proposed in Ronneberger et al. (2015) and, with minor adaptions, remains among the best architectures for image segmentation still, see also Isensee et al. (2021).

6.2.2 Stochastic Gradient Descent

In the following, we illustrate neural network training using an abstract regression problem with a least squares formulation. Given a dataset $(X^{(i)}, Y^{(i)})_{i=1,\dots,N}$ of image pairs represented by matrices[2] of size (h, w) and a neural network $\mathcal{NN}_\theta$ that processes input to target shapes, we measure the neural network's predictions via the mean squared error

[2] One can use any other numerical data.

$$\mathrm{MSE}^{(i)}(\theta) = \frac{1}{h \cdot w} \sum_{k,l}^{h,w} \frac{1}{2} (\mathcal{NN}_\theta(X^{(i)})_{kl} - Y^{(i)}_{kl})^2.$$

In order to obtain accurate predictions on the whole dataset and unseen data, the objective of neural network training is to minimize the loss function

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{MSE}^{(i)}(\theta) \tag{6.1}$$

that averages the mean-squared error over all training samples in order to find well-preforming parameters θ. The minimization of L is usually carried out using gradient based optimization methods. Typically, naive gradient descent is intractable since both the number N of samples is large and a single training example can be large in terms of memory requirements. One resorts to stochastic gradient descent (SGD), where instead of utilizing L to perform a gradient descent one uses randomly drawn subsets $\mathcal{B}$ of $\{1, \dots, N\}$, so-called *minibatches*. Precisely, given a current state of parameters θ_k this means one uses

$$L_{\mathcal{B}}(\theta_k) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathrm{MSE}^{(i)}(\theta_k)$$

to computes the gradient $\nabla L(\theta_k)$ by $\nabla L_{\mathcal{B}}(\theta_k)$ which is utilized to update θ_k. A parameter update is of the form

$$\theta_{k+1} = \theta_k - \eta \nabla L_{\mathcal{B}}(\theta_k)$$

where $\eta > 0$ is the stepsize of the gradient update, in deep learning terminology the stepsize is calles *learning rate*. Partitioning $\{1, \dots, N\}$ in $\lfloor \frac{N}{|\mathcal{B}|} \rfloor$ random disjoint step of size $|\mathcal{B}|$ and updating θ accordingly is referred to as a *training epoch*.

In practice, slightly more advanced optimization schemes are preferred over the vanilla SGD described above. They share a similar philosophy concerning stochasticity – keeping the minibatch training strategy – but often incorporate additional intuition such as momentum and per-parameter learning rates. A precise description exceeds the scope of this chapter and we refer instead to Zhang et al. (2021). For our examplary implementation we use the Adam algorithm Kingma and Ba (2014) which is arguably the most popular optimizer in modern deep learning.

6.2.3 Train-Test Split and Hyperparamter Tuning

When evaluating the trained model it is essential to benchmark it on data that was not used in the training process in order to quantify how the model behaves on unseen data. Consequently, one splits the total amount of data in a *training set* and a *test set* and only uses the former in training. This separation of data is crucial, compare

also to Figure 6.3 that shows the training loss versus the test loss as a function of the training loss for our concrete application. Note that after roughly 50 epochs only the train loss improves. This indicates that at this point the model is prone to *overfit* the data – i.e., it memorizes particularities of the training data as opposed to learn general rules.

Typically, to achieve optimal results it is not enough to perform a single model training process. Instead, the model's architecture, the optimizer, the minibatch size, the learning rate schedule, etc. all need to be fine-tuned. This process is referred to as *hyperparamter tuning*. It can be computationally expensive and requires experience.

6.3 Application: Tracer Prediction

In this section we discuss our concrete problem of employing a CNN to predict SIR after 24h.

6.3.1 Data Preprocessing

Our starting point is a collection of T1 weighted MR brain images of patients before and 24h after intrathetical tracer injection. At this stage, we assume the images to be properly registered. Before setting up the machine learning pipeline, we perform the following preprocessing steps.

1) In a first step we normalize the MRI intensities by dividing through the value of orbital fat, behind the eye, as was done in Eide et al. (2021). This serves to mitigate variances in the data caused by the MRI device.

2) Then, we extract a sagittal slice of the volumetric brain data. This step mainly serves to reduce computational cost. From here on, we work with images instead of volumetric data. Note that – given sufficient computational resources – volumetric data can be handled in the same fashion as image data.

3) The quantity we aim to predict is signal increase ration (SIR), so given image data $X^{0\text{h}} \in \mathbb{R}^{h\times w}$ before tracer injection and $X^{24\text{h}} \in \mathbb{R}^{h\times w}$ 24h after tracer injection with height h and width w we compute the SIR by the formula

$$\text{SIR}_{ij} = \frac{X_{ij}^{24\text{h}} - X_{ij}^{0\text{h}}}{X_{ij}^{0\text{h}} + 0.1}.$$

We added 0.1 to the denominator to prevent division by zero. To reduce outliers in the data, we restrict SIR to values between 0% and 400%; in formulas

$$\text{SIR}_{ij} = \max(\min(\text{SIR}_{ij}, 4), 0).$$

4) Finally, we normalize all input data that is fed into the neural network to lie in the range $[0, 1]$. In this case, this requires to convert gray-scale images $X^{0\mathrm{h}}$ that are typically represented as integers between 0 and 255 to be converted to floating point values. This is carried out by a simple division by 255.

Carefully preprocessing data is crucial for a successful machine learning pipeline and a lot of care should be taken in handling and preprocessing the data. Data that is not properly preprocessed can lead to significantly worse model predictions. Furthermore, the concrete preprocessing steps in many cases form a part of the modeling assumptions – like in our case when we implicitly deem SIR increase beyond 400% to be irrelevant.

6.3.2 Predicting SIR Formulated as a Regression Problem

The previously described data preprocessing pipeline provides us with data pairs

$$(X^{(i)}, Y^{(i)})_{i,\dots,N} = (X^{(i)}_{0\mathrm{h}}, \mathrm{SIR}^{(i)})_{i,\dots,N}$$

where N corresponds to the number of observations/patients that are available. Hence the the task for our neural network $\mathcal{NN}_\theta$ is to predict the SIR given the brain data before tracer injection. For a given input $X^{(i)}_{0\mathrm{h}}$ we write

$$\hat{Y}^{(i)} = \mathcal{NN}_\theta(X_{(i)})$$

for the estimated SIR prediction that our neural network produces. To train the neural network and to measure its performance we train it using the mean-squared error (MSE) which is computed as

$$\mathrm{MSE}_{(i)}(\theta) = \frac{1}{h \cdot w} \sum_{k,l}^{h,w} \frac{1}{2}((\mathrm{SIR}^{(i)})_{k,l} - \mathcal{NN}_\theta(X^{(i)})_{k,l})^2.$$

The full data mismatch term measured over all data points, i.e., the loss function thus becomes

$$L(\theta) = \mathrm{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{MSE}^{(i)}(\theta).$$

6.4 Implementation Details

There are several popular software libraries to choose from for deep learning applications. Among the most popular are Tensorflow Abadi et al. (2016), PyTorch Paszke et al. (2019) and JAX Bradbury et al. (2018) and all these libraries provide

efficient implementations of tensor operation and automatic differentiation. Further, the libraries enable computation on specialized hardware, such as GPUs and TPUs and facilitate scaling to several such devices for large-scale applications.

In this chapter we use PyTorch 2.0.0 and we briefly comment on how PyTorch suggests to factor code. The description is intended to be high-level, for the details we encourage the reader to consult the corresponding repository.

6.4.1 Data Handling

We assume the preprocessed and ready-to-use data to be stored on disk in any user-defined logic. To allow PyTorch to access the data the recommended way is to subclass the PyTorch class `torch.utils.data.Dataset` and implement integer-based access to input-target pairs of the data. Precisely, we need to implement the methods `getitem` and `len`. Using further PyTorch functionality – `torch.utils.Dataloader` we turn the dataset into an iterator suitable to perform minibatch training.

6.4.2 Neural Network Implementation

PyTorch provides all building blocks commonly used for defining neural networks. Deep neural networks are typically thought to consist of blocks which are some form of logical unit – in the easiest case just a convolutional or fully connected layer. In PyTorch, these blocks are represented through the class `torch.nn.Module` and any block or network architecture should subclass `torch.nn.Module`. In the constructor one needs to register trainable parameters as member variables. This can happen either directly or by defining a member variable that is itself of type `torch.nn.Module`. Further, one needs to implement a method called `forward` that describes how input to the neural network is handled.

6.4.3 Training

As described earlier, stochastic minibatch training is the gold-standard for neural network training. In the implementation this is reflected as a nested for-loop, where the inner loop constitutes one training epoch, hence consists of one minibatch pass through the dataset and the outer loop repeats that for a given number of epochs. This training loop structure is omnipresent in deep learning although some frameworks may hide it behind a single function call.

```
for epoch in range(epochs):
    for batch_idx, (inputs, targets) in enumerate(train_loader):
```

```
        # Insert channel dim and move to GPU if possible
        inputs = inputs.to(device=DEVICE).unsqueeze(1)
        # Move targets to GPU if possible
        targets = targets.to(device=DEVICE)
        # Forward pass
        predictions = model(inputs).squeeze()
        # Compute loss
        loss = loss_fn(predictions, targets)
        # Delete previously computed gradients
        optimizer.zero_grad()
        # Compute gradient wrt to trainable weights
        loss.backward()
        # Update parameters
        optimizer.step()
```

```
    for epoch in range(epochs):
        for batch_idx, (inputs, targets) in enumerate(
                                            train_loader):
            # Insert channel dim and move to GPU if possible
            inputs = inputs.to(device=DEVICE).unsqueeze(1)

            # Move targets to GPU if possible
            targets = targets.to(device=DEVICE)

            # Forward pass
            predictions = model(inputs).squeeze()

            # Compute loss
            loss = loss_fn(predictions, targets)

            # Delete previously computed gradients
            optimizer.zero_grad()

            # Compute gradient wrt to trainable weights
            loss.backward()

            # Update parameters
            optimizer.step()
```

The code above illustrates a typical training loop in PyTorch. Notably, the data is first loaded to the available accelerator `DEVICE` which is either `'cuda'` or `'cpu'`. The data is propageted through the model and `loss.backward()` invokes the automatic differentiation engine that computes the neccessary gradients. It is important to invoke `optimizer.zero_grad()` before computing gradients as PyTorch by default accumulates gradient computations[3]. In the above code `optimizer` is an instance of `torch.optim.Adam` and is the built-in version of the Adam optimizer in PyTorch. The function `loss_fn` is an implementation of the mean-squared error introduced earlier and available in PyTorch via `torch.nn.MSELoss()`.

[3] This can be relevant for applications where the desired batch-size does not fit in GPU memory.

6.4.4 Get Started

To get acquainted with the implementation we recommend to start with the training of the CNN on the examplary dataset provided. This dataset consists of 30 images – a public subset of the full, proprietary dataset. The small dataset has the adavantage that it doesn't require a GPU for the training to finish in a reasonable amount of time. To execute the training execute the following terminal command:

Terminal

```
$ python train_mse.py --data_dir 'Dataset/' --out_folder 'out/'
```

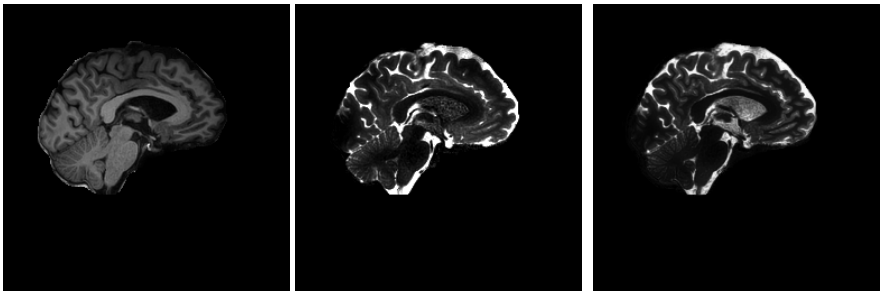
where you substitute the desired path for *'Dataset/* and *'out/'*. If you follow the structure of the provided repository these arguments need not be specified as they are set to working default values. You should then be able to see the value of the loss function as an output on the console while the training loops through the specified number of epochs.

6.5 Results

In Figure 6.2 we present three pictures – the baseline MR image before tracer injection, the true measured tracer distribution after 24 hours and the neural network prediction. Note that this image belongs to the *test set*, meaning it was not used during training. For the figures in this section we employed the full proprietary dataset consisting of 136 images, rather than publicly available subset.

We conclude that even the simple setup presented here – using only a small dataset – a neural network can already produce reasonable looking predictions. However, we stress that this chapter has introductory character and a more careful analysis is neccessary before employing an anlogue pipeline in practice.

In Figure 6.3 we present a visualization of the training process. It shows the mean squared error – which in this application is also the loss function – over the training epochs. We can clearly see that after around 50 epochs only the loss on the training dataset improves significantly. This can be attributed to overfitting which means that the network memorizes particularies of the data instead of learning general rules.



(a) Baseline (b) Tracer after 24h (c) Neural network prediction

Fig. 6.2: Baseline, tracer at 24 hours and neural network prediction.

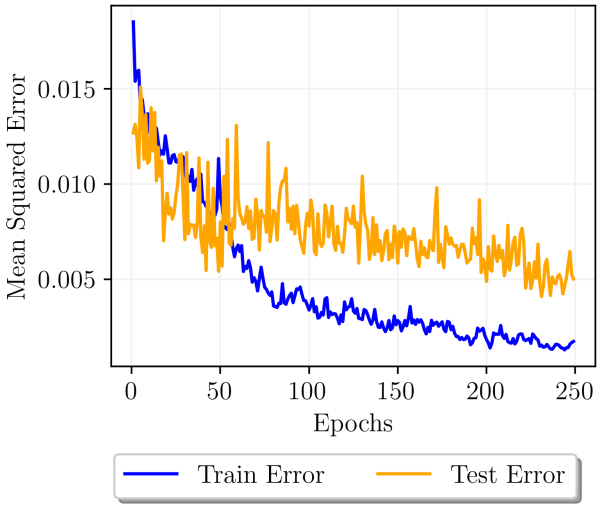


Fig. 6.3: Illustration of the training and test error during training. Depicted is the mean squared error to the baseline over a traing process of 250 epochs.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) TensorFlow: A System for Large-Scale Machine Learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, OSDI'16, pp 265–283

Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S, Zhang Q (2018) JAX: composable transformations of Python+NumPy programs. URL `http://github.com/google/jax`

Eide PK, Vinje V, Pripp AH, Mardal KA, Ringstad G (2021) Sleep deprivation impairs molecular clearance from the human brain. Brain 144(3):863–874, doi:10.1093/brain/awaa443

Isensee F, Jaeger PF, Kohl SA, Petersen J, Maier-Hein KH (2021) nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. Nature methods 18(2):203–211, doi:10.1038/s41592-020-01008-z

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations doi:10.48550/arXiv.1412.6980

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. (2019) Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32, doi:10.48550/arXiv.1912.01703

Rasmussen MK, Mestre H, Nedergaard M (2018) The glymphatic pathway in neurological disorders. The Lancet Neurology 17(11):1016–1024, doi:10.1016/S1474-4422(18)30318-1

Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, pp 234–241, doi:10.1007/978-3-319-24574-4_28

Zhang A, Lipton ZC, Li M, Smola AJ (2021) Dive into deep learning. arXiv preprint arXiv:210611342 doi:10.48550/arXiv.2106.11342

Chapter 7

Estimating Molecular Transport Parameters Using Inverse PDE Models

Bastian Zapf, Marius Zeinhofer and Kent-Andre Mardal

In section Zapf et al. (2023) we have estimated the concentration of intrathecally injected CSF tracer in the brain from the so-called glymphatic MRI protocol Iliff et al. (2013); Ringstad et al. (2017, 2018). The imaging involves several MR images taken over a time span of several days. We have seen that this data, shown exemplarily in Fig. 7.1A–C, can not be explained by a physics–based model constituted only of extra-cellular diffusion. Of course, movement of tracers in the brain likely is governed by additional processes such as advection via movement of water as proposed by the glymphatic concept Iliff et al. (2012). In rodents, it has been observed that MRI contrast is being transport 10-25 times faster than expected from extra-cellular diffusion alone Ray et al. (2021), and 10-20 times faster in sleep then in awake Xie et al. (2013). In humans, the factor appears to be smaller than in rodents and has been estimated to range from up to 26% in Valnes et al. (2020) to a factor 3.5 Vinje et al. (2023) depending on the model and timeframe of the gMRI images. Both finite elements and neural networks yield similar estimates Zapf et al. (2022). Given that (impaired) clearance of molecules from the brain might play a central role in the development of neurological diseases Rasmussen et al. (2018) and that the exact physical mechanisms have not yet been established Kelley et al. (2022); **?**); Holter et al. (2017); **?**, image based computational modeling of physiological processes is urgently needed.

However, assessing the relative significance and plausibility of different transport mechanisms via physics–based simulations is challenging since the relevant physiological parameters are largely unknown in humans Croci et al. (2019). As a remedy, one can instead attempt to estimate the unknown parameters from MRI data. This chapter focuses on diffusion–degradation models, and diffusion–advection models are covered in Rognes (2023).

Our model for diffusion and degradation of molecules is described by a partial differential equation (PDE) with unknown coefficients. We use two different variants of *PDE–constrained optimization* Hinze et al. (2008) to determine them from MRI

data. The first approach is based on the well–established finite element method via the simulations that we have implemented in Zapf et al. (2023). As an alternative approach, we also demonstrate how to use the recently popularized physics–informed neural networks (PINNs) Raissi et al. (2019) to solve this problem.

7.1 Statement of the Problem

We consider the CSF tracer concentration $c^d(t_i)$ in the brain estimated from MRI in Zapf et al. (2023) at the discrete time points $t_i \in \{7, 26, 50\}$ hours and model this data using a function $c(t, x)$ that fulfills the diffusion–reaction equation

$$\frac{\partial}{\partial t} c(t, x) = D\Delta c(t, x) - rc(t, x) \quad \text{in } \Omega \times (0, 50\,\text{h}) \tag{7.1}$$

with unknown apparent diffusion coefficient $D > 0$ and a reaction rate $r > 0$. The process of extracellular diffusion is well-documented Nicholson (2001) and it is known that diffusion in the nano-meter thick extra-cellular matrix is slowed down by roughly a factor three as compared to free diffusion. To determine these two unknown scalar numbers, we define the following PDE–constrained optimization problem: Find the parameters D, r such that

$$\mathcal{J}(c, c^d) = \sum_{i=1,2,3} \int_\Omega \left(c^d(t_i, x) - c_{D,r}(t_i, x) \right)^2 \tag{7.2}$$

is minimized under the constraint that $c_{D,r}(t, x)$ fulfills (7.1) for given parameters D, r. How this problem is solved with FEM is discussed in Section 7.3 and with PINNs in Section 7.4.

In principle, the domain Ω on which the optimization problem (7.1)–(7.2) is defined, could be the whole brain parenchyma. However, several arguments favor to instead choose Ω to be a small region of interest (ROI) of the brain. First and foremost, not all brain regions are enriched with tracer. This can –at least in parts– be explained by the large variation in the dynamics contrast agent in the CSF spaces between subjects Eide et al. (2021). The tracer enters the brain from CSF spaces, and hence, if no tracer is available at the brain–CSF boundary, the region will not be enriched. Data from these regions hence contains no information about molecular transport in the brain. Hence, a first criterion for a ROI for solving (7.1)–(7.2) is that there is significant influx of tracer over the time span of interest.

Secondly, the aim of this chapter is to present the basic workflow of the FEM and PINN approach to estimating parameters from MRI in a easily reproducible fashion. Hence, we want to formulate the problems such that they can be solved on personal computers, avoiding the need to run the computations on a supercomputer. To this end, our second criterion is to choose a subregion which is geometrically simple. For the FEM approach, a complicated domain will require a finer mesh resolution,

increasing the memory and compute time requirements of the programs. PINNs are currently under rapid development, cf. Cuomo et al. (2022), and training PINNs on complicated geometries may require problem–specific hyperparameter tuning Zapf et al. (2022) or domain decomposition strategies, e.g., D. Jagtap and Em Karniadakis (2020), increasing the computational effort required to solve the problem (7.1)–(7.2). An example for a ROI Ω fulfilling these two criteria is shown in Fig. 7.1D–F.

Thirdly, focusing on a small subregion of the brain also makes sense from a physiological point of view. The parameters describing dynamics of molecules in the brain vary between tissues and brain location, as can be seen, e.g., in diffusion tensor imaging for water molecules. Hence our modeling assumption of using scalar parameters D, r can be expected to be less accurate the larger the region of interest is chosen. Choosing a small region, preferably containing a single type of tissue, increases the interpretability of the obtained results. In the next section we describe how to define such a region for the MRI data accompanying this book using Freeview and a Python script.

7.2 Defining and Meshing a Region of Interest

In this chapter, we assume that the data and scripts are arranged as described in Zapf et al. (2023). We start by opening the concentration estimates and the baseline T_1–weighted image in Freeview:

Terminal window

```
$ Freeview ./data/freesurfer/REGISTERED/20230213_073508.mgz \
    --colormap Jet -v ./data/freesurfer/CONCENTRATIONS/6.56.mgz \
    ./data/freesurfer/CONCENTRATIONS/26.05.mgz \
    ./data/freesurfer/CONCENTRATIONS/50.39.mgz
```

It is advised to set the minimum and maximum of the color map to 0 and 0.2, respectively. This is beneficial for visualization of the data since there are only a few outlier voxels with concentration values > 0.2 mmol/L. Setting the colormap maximum to 0.2 hence yields a visualization with less focus on the outliers.

By scrolling through the image along different axes, we can inspect the concentration data in different regions. By activating only one volume in the list of loaded volumes in Freeview at a time, we can switch the view between the three different time points. Moving the cursor to voxel indices (146, 100, 133) and displaying only the concentration at 26 hours, we obtain the views shown in Fig. 7.1A–C. It can be seen that the brain is primarily enriched from the subarachnoid space, and hence our first criterion for a suitable ROI is fulfilled. However, these regions are characterized by

the complex folding patterns of the cerebral cortex, and hence our second criterion for a good ROI is not fulfilled.

A closer look at the slices in Fig. 7.1D–F reveals that the brain is also enriched from the lateral ventricles in some regions. Particularly, the subcortical white matter around the position of our cursor at voxels (146, 100, 133) is clearly enriched with tracer at all timepoints 7, 26, 50 hours after injection. Since the boundary between the tissue and the lateral ventricles is considerably simpler than the foldings of the cerebral cortex, this region fulfills both our criteria for a suitable ROI.

We choose to define the ROI as the set of voxels indices $\{(a, b, c)\}$ which are contained in a sphere of radius $R = 10\,\text{mm}$ around the cursor at voxel indices (146, 100, 133) and are labeled as brain tissue in the FreeSurfer segmentation file `aseg.mgz`. Mathematically, these constraints can be formulated as

$$(a - 146)^2 + (b - 100)^2 + (c - 133)^2 \leq R^2 \quad \text{and} \quad (a, b, c) \in \Omega_{\text{brain}}. \tag{7.3}$$

We implement this condition in a Python script which can be called as

Terminal window

```
$ python ./scripts/inverse-model/make_roi.py \
    --maskfile ./data/freesurfer/mri/parenchyma_mask.mgz \
    --resolution 12
```

where we reused the binary mask for the brain parenchyma created in Section Zapf et al. (2023). The script creates an empty MRI volume and marks all voxels that fulfill (7.3) with the value 1, and the others with the value 0. From this binary mask, it creates a surface and a volume mesh for this region with resolution specified by the `--resolution` parameter as described in Mardal et al. (2022). The surface of this mesh is shown by white contours in Fig. 7.1D–F. Running this script creates all these files and stores them into an output folder `./roi12/`. Therein, we find a finite element mesh for the ROI under `./roi12/parenchyma_mask_roi12.xml` which is used for the FEM approach presented in this chapter. For convenience, the mesh is also stored in `.xdmf` format for visualization in Paraview. The PINN approach does not require a mesh, and our implementation uses the binary mask `./roi12/parenchyma_mask_roi.mgz`. This mask can be visualized in freeview.

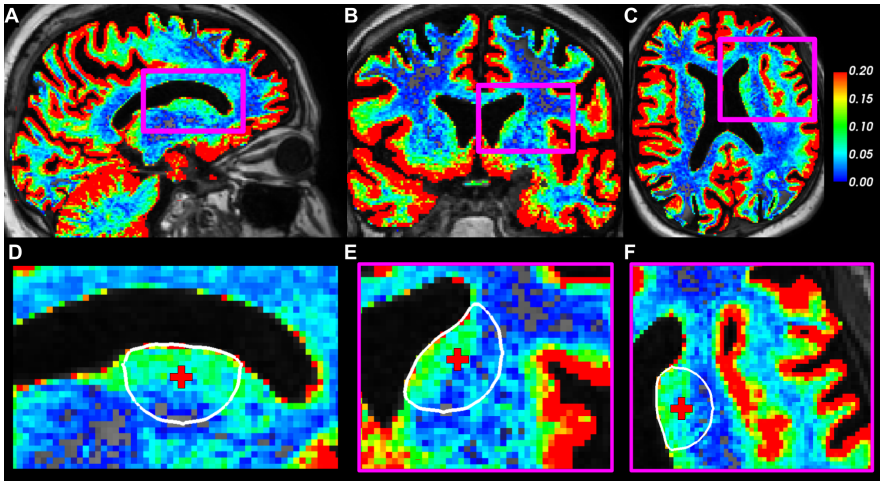


Fig. 7.1: The image shows slices through the baseline MR image (greyscale) and the tracer concentration in the brain after 26 hours (colorscale). A–C: Whole brain, D–F: zoom into a region around the left lateral ventricle. The red cross indicates the location of the cursor at voxel indices (146, 100, 133), and the white contour lines depict the surface of the region of interest on which we solve the inverse parameter estimation problem.

7.3 Finite Element Approach

We solve the parameter estimation problem (7.1)–(7.2) using a *reduced* approach with the finite element method. That is, given Dirichlet boundary conditions[1] $g(t,x)$ and an initial condition $c_0(x)$, we can use the code from section Zapf et al. (2023) to solve the well–posed PDE problem

$$\frac{\partial}{\partial t}c(t,x) = D\Delta c(t,x) - rc(t,x) \quad \text{in } \Omega \times (0, 50\,\text{h}) \tag{7.4}$$

$$c(t,x) = g(t,x) \quad \text{on} \quad \partial\Omega \times (0, 50\,\text{h}) \tag{7.5}$$

$$c(0,x) = c_0(x) \tag{7.6}$$

numerically to obtain a unique solution $c_{D,r}$. Again, the boundary condition g is taken as a linear interpolation in time between available MRI and the initial condition is $c_0 = 0$ since the tracer is injected at a later time point $t > 0$. The mismatch (7.2) between model and observation then becomes a function of D and r only, and is called the *reduced functional*

[1] Here, we work with Dirichlet boundary conditions since they can be estimated from MRI data as described in Section Zapf et al. (2023). In some applications, Neumann or Robin boundary conditions might be more appropriate. The methodology presented in this chapter can equally be applied when using these conditions instead.

$$\mathcal{J}_r(D, r) = \mathcal{J}(c_{D,r}, c^d). \tag{7.7}$$

Now we have reformulated the constrained optimization problem (7.1)–(7.2) into the unconstrained optimization of (7.7) with respect to D and r. This can be solved using gradient–based optimization algorithms. For the FEM approach, we use the quasi–second order method L–BFGS–B Fletcher (2013).

Hence, an efficient way of computing the gradients

$$\frac{\partial \mathcal{J}_r(D, r)}{\partial D} \quad \text{and} \quad \frac{\partial \mathcal{J}_r(D, r)}{\partial r} \tag{7.8}$$

is needed. We use the adjoint method (known as reverse mode algorithmic differentiation in machine learning Baydin et al. (2018)) as implemented for FEniCS in dolfin–adjoint[2] Mitusch et al. (2019). Under the hood, the dolfin–adjoint package achieves this task by overwriting the functions and objects defined in the Python interface to FEniCS. Our script to solve the optimization problem (7.7) can be found under `scripts/inverse-model/fem-inverse-diffusion.py`. Here, we outline the essential parts of this script, though the actual implementation might differ slightly for technical reasons. These are explained in the comments accompanying the scripts.

After loading the mesh created in Section 7.2, we can define a finite element space `V` on this mesh and represent the concentration estimates as functions in this space. This processes is wrapped into a class `FEniCS_Data` and loading the MRI and storing the functions for visualizing can be done as

```
mris = FEniCS_Data(function_space=V, datapath=datapath)
mris.dump_pvd(vtkpath=str(outfolder / "data.pvd"))
```

Now, given initial guesses for D and r, we can evaluate the forward simulation and dolfin–adjoint keeps track of the intermediate results (needed to compute the gradients) as before, i.e.

```
model = Model(V=V, mris=mris, outfolder=outfolder)
model.forward(D=D_w, r=r)
L2_mismatch = model.return_value()
model.save_predictions(name="initial")
```

The second last line returns the data discrepancy (7.2) computed during the forward pass. In the last line, we store the simulated tracer fields at the time points where measurements are available for visualization. Note that the `forward` method expects the apparent diffusion coefficient D_w for water, not for the CSF tracer gadobutrol. Instead, it estimates the diffusion coefficient of gadobutrol from that for water to

[2] The interested reader is also referred to the dolfin–adjoint documentation at `http://www.dolfin-adjoint.org/en/latest/`. There, various example implementations of PDE–constrained optimization problems and also a more substantial introduction to the concepts behind PDE–constrained optimization can be found.

perform the simulation. This implementation choice allows the flexibility to initialize the optimization with values from diffusion tensor imaging, which measures the diffusion of water.

Next, we define the set of optimization variables and the reduced functional (7.7) in dolfin–adjoint:

```
ctrls = [Control(D_w), Control(r)]
Jhat = ReducedFunctional(L2_mismatch, ctrls)
```

Finally, we optimize (7.7) for D, r using L–BFGS–B as

```
opt_ctrls = minimize(Jhat, method="L-BFGS-B", callback=iter_cb)
```

The so-called *callback* method `iter_cb` will be evaluated after every optimization step, and we use it to write the values of $\mathcal{J}, D$ and r during the optimization to a text file for later analysis.

The full optimization script can be run as

Terminal window

```
$ conda activate diffusion-fenics-env
$ python scripts/inverse-model/fem-inverse-diffusion.py \
    --data data/freesurfer/CONCENTRATIONS/ \
    --mesh roi12/parenchyma_mask_roi12.xml
```

By default, the script uses a time step of 1 h and the Crank–Nicolson time discretization scheme. With this setting, after 9 iterations, which take about three minutes on a Lenovo ThinkPad P-43s, the optimizer converges and the L^2–error is reduced by around 15 % compared to our initial guess, cf. Fig. 7.2 (a). We find the optimal $D \approx 2 \times 10^{-4}\,\mathrm{mm}^2/\mathrm{s}$ and $r \approx 9 \times 10^{-6}\,1/\mathrm{s}$, cf. Figs. 7.2 (b-c).

While these values are of the same magnitude as determined using PDE–constrained optimization using MRI from clinical studies Valnes et al. (2020); Vinje et al. (2023), it should be checked if the numerical PDE solver is converged with respect to temporal and spatial discretization. We perform this important test in the next section.

7.3.1 Numerical Convergence Analysis

In Section 7.2 we have created a mesh for Ω with the `SVMTK` mesh resolution parameter set to 12. Table 7.2 lists some characteristics of this mesh. We can see

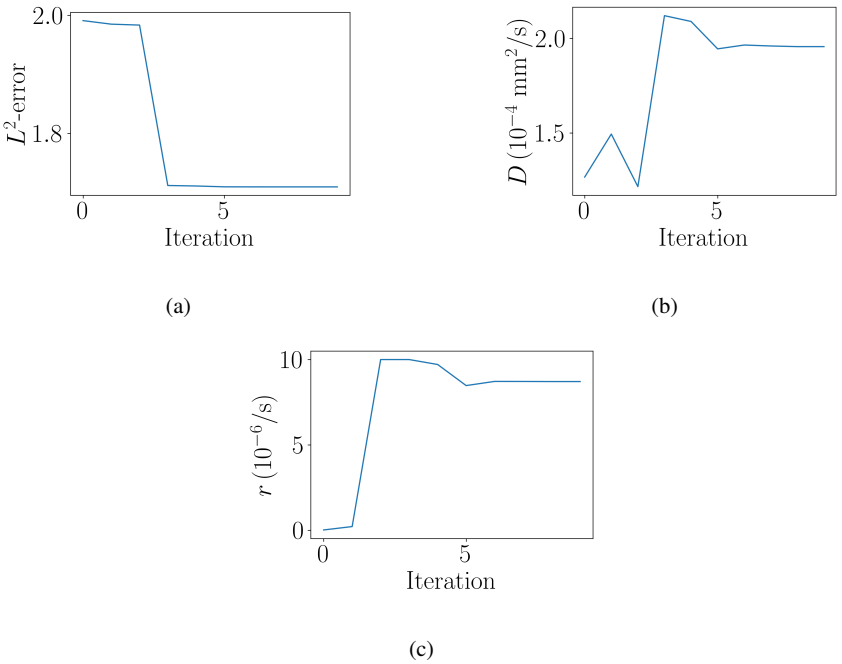


Fig. 7.2: Behavior of L^2-error, D and r during optimization.

that this mesh consists of roughly 2×10^4 cells and has a maximum cell size (longest distance between any two vertices of a cell) of 1.73 mm. Given that the MRI resolution is 1 mm^3, we might ask if this resolution is fine enough. We hence create two finer meshes with mesh resolution parameter set to 20 and 28, respectively, using the script presented in Section 7.2. Table 7.2 also lists the parameters for these meshes.

Next, we perform a systematic study and evaluate the script `fem-inverse-diffusion.py` with all combinations of these three meshes and time steps of 60, 30, 15 and 7.5 min to assess if the estimation of D, r has converged with respect to these numerical parameters. A shell script to do this automatically is given in `scripts/inverse-model/run_fem_study.sh`. Some characteristics of these meshes are tabulated in table 7.1.

After running this systematic study, which may take up to a day of computing time when running the computations sequentially, we can print the estimated coefficients to the terminal using the provided script `scripts/inverse-model/print_results.py`.

From table 7.2 we observe that the obtained diffusion coefficients vary from $D \sim 2.1 \times 10^{-4}$ mm^2/s to $D \sim 1.6 \times 10^{-4}$ mm^2/s between the different meshes and time step sizes. This variation is due to the data under consideration being subject to substantial noise (cf. Fig. 7.1D–F) and the property of inverse problems usually not being well–posed. Still, these numerical results give insight in the order of magnitude

Table 7.1: Mesh characteristics for different resolution parameters

Mesh resolution parameter	12	20	28
cells (10^4)	1.9	8.8	23.9
vertices (10^4)	0.4	1.6	4.1
$h_{\min}$ (mm)	0.38	0.36	0.26
$h_{\max}$ (mm)	1.72	1.00	0.73

Table 7.2: Influence of mesh and time stepping size on the estimated parameters D, r and the relative error defined by (7.9).

Mesh resolution parameter	12	20	28	12	20	28	12	20	28
time step dt (s)	D (10^{-4} mm^2/s)			r (10^{-6} 1/s)			rel. L^2–error (10^{-2})		
3600	1.96	2.05	2.01	8.71	9.15	9.05	7.01	7.18	7.28
1800	1.97	2.07	2.02	8.79	9.25	9.14	7.89	8.06	8.16
900	1.73	1.81	1.77	7.40	7.76	7.67	7.51	7.71	7.80
450	1.63	1.70	1.66	6.83	7.14	7.07	7.30	7.51	7.59

of the model parameters. We further compute the relative L^2–error defined as

$$\left(\frac{\sum_{i=1,2,3} \int_\Omega \left(c^d(t_i,x) - c_{D,r}(t_i,x) \right)^2}{\sum_{i=1,2,3} \int_\Omega \left(c^d(t_i,x) \right)^2} \right)^{1/2} \tag{7.9}$$

and present the numerical values in table 7.2. Whilst the values vary little between different numerical parameters, we observe in section 7.4 that the FEM and PINN approaches yield predictions that match the data substantially different, cf. the discussion in section 7.5. In Fig. 7.4 we compare the FEM reconstruction of the CSF tracer concentration at 7 and 50 h after injection to the data.

7.4 Physics-Informed Neural Network Approach

In this Section we describe the physics-informed neural network approach Raissi et al. (2019) to determine the unknown diffusion and reaction term. Recall, that we formulated the problem of recovering the diffusivity $D \in \mathbb{R}$ and the reaction coefficient $r \in \mathbb{R}$ from the MRI data as the minimization problem

$$\min_{c,D,r} J(c,D,r) = \sum_{i=1}^{3} \int_\Omega (c^d(t_i,x) - c(t_i,x))^2 \mathrm{d}x \tag{7.10}$$

subject to

$$\partial_t c(t,x) = D\Delta c(t,x) - rc(t,x) \quad \text{in } \Omega \times (0, 50\text{h}). \tag{7.11}$$

We keep the ROI and the data $c^d(t_i), t_i = 7, 26, 50$ h the same as in the FEM example.

7.4.1 Fully Connected Neural Networks

A fully connected feed-forward neural network $\mathcal{NN}$ of input dimension d, with scalar-valued output and activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is given by

$$\mathcal{NN} : \mathbb{R}^d \to \mathbb{R}, \quad \mathcal{NN}(x) = (T_N \circ \sigma \circ \cdots \circ T_2 \circ \sigma \circ T_1)(x), \tag{7.12}$$

where T_i is affine linear and hence given through a matrix W_i and a vector b_i as

$$T_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}, \quad T_i(x) = W_i x + b_i.$$

The activation function is applied component-wise. A network of the form (7.12) is said to be of depth N and we refer to the affine maps T_i as the layers of the network. For notational convenience, we abbreviate the matrix-vector pairs by

$$\theta = ((W_1, b_1), \ldots, (W_N, b_N)).$$

The vector θ is referred to as the (trainable) parameters of the neural network. For a set of parameters θ, we denote the function represented via (7.12) by c_θ, i.e., we have

$$c_\theta(x) = (T_N \circ \sigma \circ \cdots \circ T_2 \circ \sigma \circ T_1)(x) \quad \text{with } T_i z = W_i z + b_i.$$

There is an abundance of variations of neural network architectures used in the literature and the interested reader is referred to Zhang et al. (2021). For the applications in this Section – and to some extend in the practice of physics-informed neural networks – we keep the network sizes small, with only a few layers of moderate size, i.e., less than 100 neurons per layer. Such small networks are usually expressive enough to represent the data, and benefit from reduced training cost compared to deeper neural networks. We choose the hyperbolic tangent tanh as the activation function in our application.

7.4.2 The PINN Formulation

The essential idea of physics-informed neural networks is to approximate the solution c of the problem (7.10) by a neural network c_θ. In order to find parameters θ that lead to an accurate approximative solution of (7.10) one aggregates both the objective and the constraint into a loss function L that one minimizes, i.e.,

$$\begin{aligned}\min_{\theta,D,r} L(\theta) = \sum_{i=0}^{3} \int_{\Omega} (c^d(t_i,x) - c_\theta(t_i,x))^2 \mathrm{d}x \\ + w_{\mathrm{PDE}} \int_I \int_{\Omega} (\partial_t c_\theta - D\Delta c_\theta + r c_\theta)^2 \mathrm{d}x\mathrm{d}t,\end{aligned} \tag{7.13}$$

where w_{PDE} is a fixed, user-specified weighting term. The initial condition $c(0,x) = 0$ is enforced weakly via penalization of the first data loss term with index $i = 0$. Note that the first term in (7.13) vanishes if and only if the IC and data $(c^d(t_i))_{i=1,2,3}$ are matched and the second term vanishes if and only if c_θ satisfies the PDE (7.11). Note the difference in the formulation of the minimization problem to the FEM approach:

(i) In the PINN approach, we rely only on the observational data and the interior PDE residual, whereas in FEM we need to include boundary conditions.

(ii)The FEM formulation ensures that the PDE is exactly solved during the optimization process, whereas the PINN formulation includes the PDE as a soft penalty and thus does not guarantee an exact PDE solve. This motivates the weighting term w_{PDE} in (7.13).

In practice, we need to discretize the integrals appearing in (7.13). This is typically done employing Monte-Carlo integration, i.e., we sample random points $\{(t_j,x_j)\}_{i=1,\dots,N_{\mathrm{PDE}}} \subset I\times\Omega$ and subsample available data points $\{(t_j^d,x_j)\}_{i=1,\dots,N_{\mathrm{data}}}$, where $t_j^d \in \{1,2,3\}$ and x_j^d is a point in Ω where we have observational data. This results in the following discretized version of (7.13) that we still denote by L

$$\begin{aligned}\min_{\theta,D,r} L(\theta) = \frac{1}{N_{\mathrm{data}}} \sum_{j=1}^{N_{\mathrm{data}}} (c^d(t_j^d,x_j^d) - c_\theta(t_j^d,x_j^d))^2 \\ + w_{\mathrm{PDE}} \frac{1}{N_{\mathrm{PDE}}} \sum_{j=1}^{N_{\mathrm{PDE}}} (\partial_t c_\theta(t_j,x_j) - D\Delta c_\theta(t_j,x_j) + r c_\theta(t_j,x_j))^2\end{aligned} \tag{7.14}$$

For numerical stability, we restrict the range of D and r to certain intervals $[D_{\min}, D_{\max}]$ and $[r_{\min}, r_{\max}]$ and parameterize

$$D = D(s) = D_{\min} + \sigma(s)(D_{\max} - D_{\min})$$

and

$$r = r(s) = r_{\min} + \sigma(s)(r_{\max} - r_{\min})$$

where σ is given by the sigmoid function

$$\sigma(s) = \frac{1}{1+e^{-s}}.$$

In the application, we have to check if the optimization converges to the upper and lower bounds on the parameters. In this case, the parameterization is too restrictive, and the allowed range need to be increased.

7.4.3 Stochastic Gradient Descent

The prevailing strategy in the PINN community to solve (7.14) are stochastic first order methods such as Adam Kingma and Ba (2014), although alternatives are currently actively investigated Zeng et al. (2022); Müller and Zeinhofer (2023); Siegel et al. (2023). All methods require the computation of gradients of L with respect to θ, D and r. Modern deep learning frameworks provide efficient implementations of automatic differentiation and reduce the complexity of computer codes that compute the derivatives of L to a minimum. Here, we use the Python library JAX Bradbury et al. (2018). Given an implementation of L in JAX, the code showing how to compute the derivatives of L using JAX is given below. Note that for technical reasons it differs slightly in the implementation of the real problem.

```
from jax import grad
# defintion of L
def L(theta, D, r):
    ...
    return loss

# gradient with respect to theta
L_theta = grad(L, 0)
# gradient with respect to D
L_D = grad(L, 1)
# gradient with respect to r
L_r = grad(L, 2)
```

In contrast to other deep learning frameworks, JAX implements `grad` as a function transformation. This means that it accepts a python callable as a first argument and also returns a callable object – the gradient of the given function. The second argument of integer type denotes the variable with respect to which the gradient is taken.

Stochasticity in the optimization process is induced through the variation of the quadrature points $\{(t_j^d, x_j^d)\}_{j=1,\dots,N_{\text{data}}}$ for the data and $\{(t_j, x_j)\}_{j=1,\dots,N_{\text{PDE}}}$ for the PDE residual. A common strategy is to use so-called minibatch training, where the total of the available data points is randomly partitioned in minibatches of the same size and a gradient update is conducted for each minibatch before repeating the procedure. It is important to note that while observational data $c^d(t,x)$ is typically only available at a finite number of space–time points (t,x), we can generate unlimited quadrature points for the PDE term which allows more flexibility in handling this part of the loss function.

7.4.4 Hyperparameter tuning

There are many design choices – commonly referred to as hyperparameters – to be made when employing PINNs in practice. Here, we present a non-exhaustive list,

discuss their relevance and give some pointers to the literature. We present some exemplary investigations concerning the sensitivity of the PINN estimates for D, r with respect to the weighting parameter w_{PDE}. Relevant hyperparameters include

(i) The neural network's architecture. Both its fundamental structure, such as choosing a ResNet versus a full connected architecture, but also the number and width of the layers and the activation function. A summary of popular choices can be found in Hennigh et al. (2021).

(ii) The weighting of the PDE and data loss terms. We demonstrate the effect of the weighting in Table 7.3 and point to Wang et al. (2021) for further reading.

(iii) Sampling strategies. This includes the size of the minibatch, the number of quadrature points employed for computing the PDE loss term and possibly adaptive sampling strategies that redistribute quadrature points in regions of high residuals as proposed in Daw et al. (2022). In our implementation we use the evolutionary sampling proposed in Daw et al. (2022).

(iv) The choice of L^p-norm for the PDE loss in (7.13). It is documented in Zapf et al. (2022) that it can be advantageous to choose $p = 1$ instead of the natural choice $p = 2$.

It is important to note that hyperparameter tuning is a potentially difficult process. In practice, one should therefore carefully investigate the stability of the results with respect to the hyperparameters obtained with PINN (as with other methods).

Using the script `scripts/inverse-model/pinn-inverse-diffusion.py` we train the PINN with three different choices of PDE weights w_{PDE}. By default, the script uses a PDE weight set to 10^3 and can be called as

Terminal window

```
$ conda activate jax
$ python ./scripts/inverse-model/pinn-inverse-diffusion.py
```

In Fig. 7.3 we plot the loss, D and r during training of the PINN. We observe that significantly more iterations as compared to the FEM approach are required. This is due to the complexity of the non–convex optimization problem (7.14) which has to be solved in PINN training. From Fig. 7.3 we observe that the optimization converges after 100,000 iterations in the present example. On a laptop and utilizing GPU–acceleration, such a training can take a few hours.

In Fig. 7.4 we compare the PINN reconstruction of the data at 7 and 50 h after injection to the data and the FEM reconstruction. We observe that the PINN and FEM approaches yield visually different reconstructions. This is possibly related to

the difference in the problem setup; in the FEM approach the boundary conditions are enforced strictly, while the PINN approach only relies on the available data.

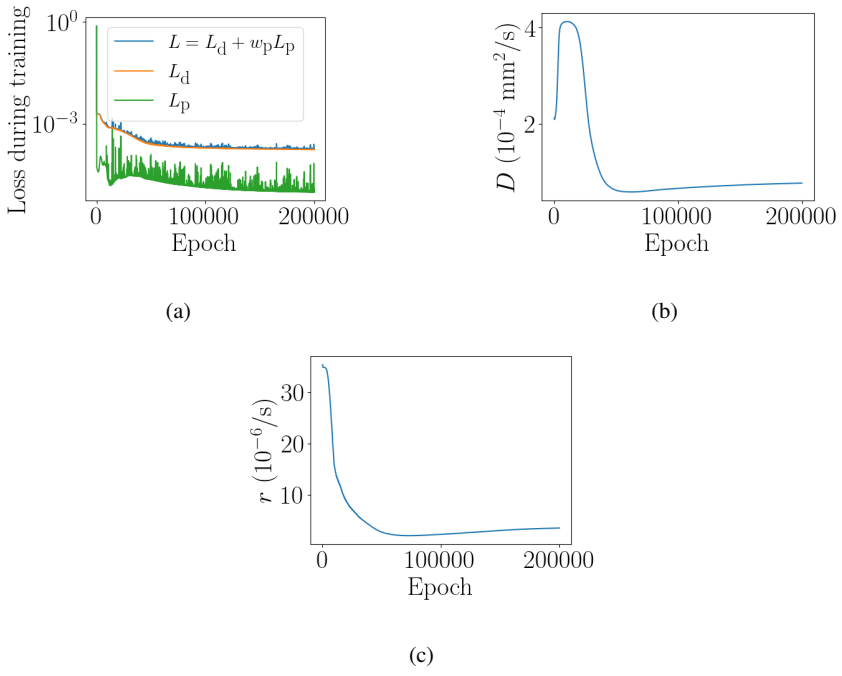


Fig. 7.3: Visualization of the different loss terms, D and r during the training process in the PINN approach. Here we set the PDE weight to $w_p = 10^3$.

Next, we train the PINN with different PDE weights 10^2 and 10^4. The resulting parameter estimates are shown in Table 7.3. We observe that for higher PDE weight, the parameter estimates are in closer correspondence to the FEM estimates. High PDE weights imply that the PDE must be satisfied accurately, which is also the case in the FEM approach. Hence, it can be expected that high PDE weights result in PINN predictions $c_\theta(t,x)$ closer to FEM solutions.

Table 7.3: The influence of the hyperparameter w_{PDE} on the outcome of the PINN training process

w_{PDE}	D (10^{-4} mm^2/s)	r (10^{-6} 1/s)	rel L^2–error (10^{-2})
10^2	0.25	0.69	37.95
10^3	0.77	3.53	17.46
10^4	1.37	4.06	20.92

7.5 Concluding remarks

In this chapter, we have demonstrated two different computational methodologies to estimate unknown molecular transport parameters from MRI. We observe a certain mismatch between the obtained values for the diffusion coefficient when comparing the PINN estimates to the FEM estimates. This can possibly be explained by to the difference in the problem formulation. The PINN approach does not require the specification of the boundary condition, whereas in the FEM the boundary values are linearly interpolated in time from the measurements. It has been shown in the modeling study Hornkjøl et al. (2022) that the availability of tracer at the brain surface is critical for the distribution in the interior, suggesting that the treatment of boundary conditions is a crucial modeling assumption. This may additionally explain the different L^2 errors between data and prediction in the FEM and PINN approach (cf. tables 7.2 and 7.3). The PINN errors are one magnitude higher than the FEM errors. This indicates that either the PINN prediction underfits the data or the FEM prediction overfits noise in the data due to the noisy boundary conditions. Hence, the different treatment of boundary conditions in both approaches possibly explains the difference in estimated parameters. As a remedy, one could modify the FEM approach as in Zapf et al. (2022); Valnes et al. (2020) and determine the optimal boundary condition via optimization simultaneously with D, r. This, however, requires additional regularization and significantly more computational resources, even necessitating a supercomputer.

Notably, Valnes et al. (2020) who investigated a similar problem as described here, find $D \sim 1.5 \times 10^{-4}\,\mathrm{mm}^2/\mathrm{s}$, in close correspondence with our results. Still, a close inspection of Fig. 7.4 reveals that the PINN and FEM concentration profiles visibly mismatch the MRI data. Whilst both the PINN and FEM approach have converged to a minimum, the remaining mismatch indicates that we miss some physiological processes in our model. After all, it would be astonishing if the distribution of molecules into the human brain can simply be described by two scalar numbers.

References

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2018) Automatic Differentiation in Machine Learning: a Survey. Journal of Marchine Learning Research 18:1–43, URL `https://mural.maynoothuniversity.ie/10227/`, does not have doi

Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S, Zhang Q (2018) JAX: composable transformations of Python+NumPy programs. URL `http://github.com/google/jax`

Croci M, Vinje V, Rognes ME (2019) Uncertainty quantification of parenchymal tracer distribution using random diffusion and convective velocity fields. Fluids and Barriers of the CNS 16(1):1–21, doi:10.1186/s12987-019-0152-7

Cuomo S, Di Cola VS, Giampaolo F, Rozza G, Raissi M, Piccialli F (2022) Scientific machine learning through physics–informed neural networks: Where we are and what's next. Journal of Scientific Computing 92(3):88, doi:10.1007/s10915-022-01939-z

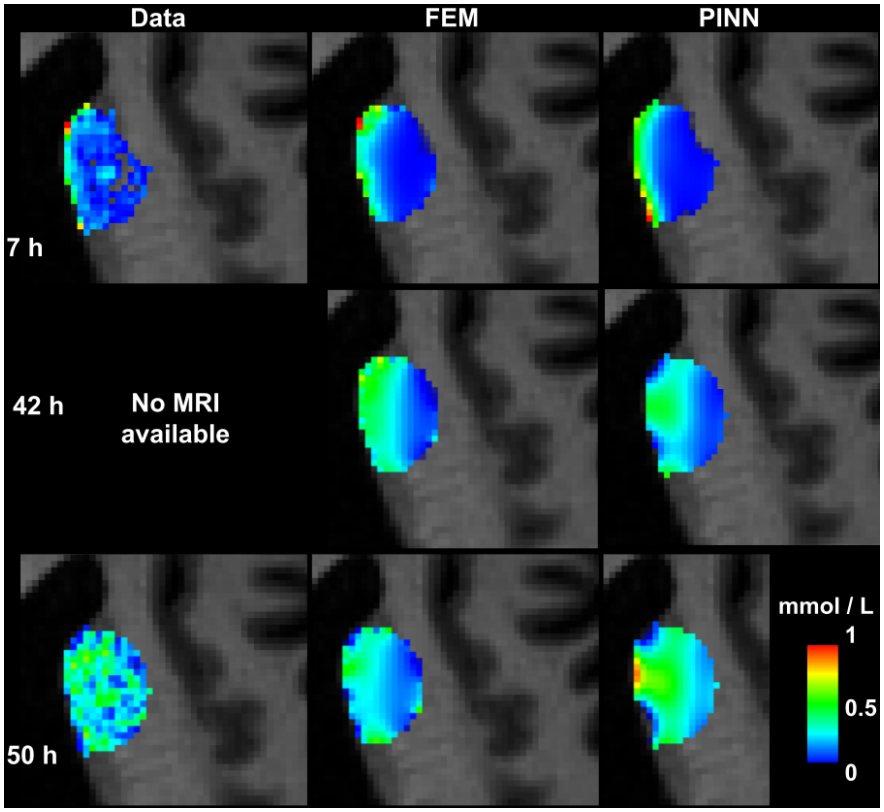


Fig. 7.4: The image shows slices through the baseline MR image (greyscale) and the tracer concentration in the brain after 7 and 50 hours (colorscale) as estimated from MRI (left) compared to FEM reconstruction (center) and PINN reconstruction (right). The image also shows the prediction at 42 h where no data is available.

D Jagtap A, Em Karniadakis G (2020) Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. Communications in Computational Physics 28(5):2002–2041, doi:10.4208/cicp.OA-2020-0164

Daw A, Bu J, Wang S, Perdikaris P, Karpatne A (2022) Rethinking the importance of sampling in physics-informed neural networks. arXiv preprint arXiv:220702338 doi:10.48550/arXiv.2207.02338

Eide PK, Valnes LM, Lindstrøm EK, Mardal KA, Ringstad G (2021) Direction and magnitude of cerebrospinal fluid flow vary substantially across central nervous system diseases. Fluids and Barriers of the CNS 18(1):16, doi:10.1186/s12987-021-00251-6

Fletcher R (2013) Practical Methods of Optimization, 2nd edn. John Wiley & Sons

Hennigh O, Narasimhan S, Nabian MA, Subramaniam A, Tangsali K, Fang Z, Rietmann M, Byeon W, Choudhry S (2021) NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework. In: Paszynski M, Kranzlmüller D, Krzhizhanovskaya VV, Dongarra JJ, Sloot PM (eds) Computational Science – ICCS 2021, Springer International Publishing, Cham, pp 447–

461
Hinze M, Pinnau R, Ulbrich M, Ulbrich S (2008) Optimization with PDE constraints, vol 23. Springer Science & Business Media, doi:10.1007/978-1-4020-8839-1
Holter KE, Kehlet B, Devor A, Sejnowski TJ, Dale AM, Omholt SW, Ottersen OP, Nagelhus EA, Mardal KA, Pettersen KH (2017) Interstitial solute transport in 3D reconstructed neuropil occurs by diffusion rather than bulk flow. Proceedings of the National Academy of Sciences 114(37):9894–9899, doi:10.1073/pnas.1706942114
Hornkjøl M, Valnes LM, Ringstad G, Rognes ME, Eide PK, Mardal KA, Vinje V (2022) Csf circulation and dispersion yield rapid clearance from intracranial compartments. Frontiers in Bioengineering and Biotechnology 10, doi:10.3389/fbioe.2022.932469
Iliff JJ, Wang M, Liao Y, Plogg BA, Peng W, Gundersen GA, Benveniste H, Vates GE, Deane R, Goldman SA, Nagelhus EA, Nedergaard M (2012) A Paravascular Pathway Facilitates CSF Flow Through the Brain Parenchyma and the Clearance of Interstitial Solutes, Including Amyloid β. Science Translational Medicine 4(147):147ra111–147ra111, doi:10.1126/scitranslmed.3003748
Iliff JJ, Lee H, Yu M, Feng T, Logan J, Nedergaard M, Benveniste H, et al. (2013) Brain-wide pathway for waste clearance captured by contrast-enhanced MRI. The Journal of clinical investigation 123(3):1299–1309, doi:10.1172/JCI67677
Kelley DH, Bohr T, Hjorth PG, Holst SC, Hrabětová S, Kiviniemi V, Lilius T, Lundgaard I, Mardal KA, Martens EA, et al. (2022) The glymphatic system: Current understanding and modeling. Iscience 25:104987, doi:10.1016/j.isci.2022.104987
Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations doi:10.48550/arXiv.1412.6980
Mardal KA, Rognes ME, Thompson TB, Valnes LM (2022) Mathematical Modeling of the Human Brain: From Magnetic Resonance Images to Finite Element Simulation. Springer Nature, doi:10.1007/978-3-030-95136-8
Mitusch SK, Funke SW, Dokken JS (2019) dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake. Journal of Open Source Software 4(38):1292, doi:10.21105/joss.01292
Müller J, Zeinhofer M (2023) Achieving high accuracy with pinns via energy natural gradients. arXiv preprint arXiv:230213163 doi:10.48550/arXiv.2302.13163
Nicholson C (2001) Diffusion and related transport mechanisms in brain tissue. Reports on Progress in Physics 64(7):815, doi:10.1088/0034-4885/64/7/202
Raissi M, Perdikaris P, Karniadakis G (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378:686–707, doi:10.1016/j.jcp.2018.10.045
Rasmussen MK, Mestre H, Nedergaard M (2018) The glymphatic pathway in neurological disorders. The Lancet Neurology 17(11):1016–1024, doi:10.1016/S1474-4422(18)30318-1
Ray LA, Pike M, Simon M, Iliff JJ, Heys JJ (2021) Quantitative analysis of macroscopic solute transport in the murine brain. Fluids and Barriers of the CNS 18(1):55, doi:10.1186/s12987-021-00290-z
Ringstad G, Vatnehol SAS, Eide PK (2017) Glymphatic MRI in idiopathic normal pressure hydrocephalus. Brain 140(10):2691–2705, doi:10.1093/brain/awx191
Ringstad G, Valnes LM, Dale AM, Pripp AH, Vatnehol SAS, Emblem KE, Mardal KA, Eide PK (2018) Brain-wide glymphatic enhancement and clearance in humans assessed with MRI. JCI Insight 3(13), doi:10.1172/jci.insight.121537
Rognes ME (2023) Ocd. In: Rognes ME, Vinje V, Dokken J, Valnes LM, Mardal KA (eds) MRI2FEM II: from magnetic resonance images to computational brain mechanics, Springer
Siegel JW, Hong Q, Jin X, Hao W, Xu J (2023) Greedy training algorithms for neural networks and applications to pdes. Journal of Computational Physics 484:112084, doi:10.1016/j.jcp.2023.112084
Valnes LM, Mitusch SK, Ringstad G, Eide PK, Funke SW, Mardal KA (2020) Apparent diffusion coefficient estimates based on 24 hours tracer movement support glymphatic transport in human cerebral cortex. Scientific Reports 10(1):1–12, doi:10.1038/s41598-020-66042-5

Vinje V, Zapf B, Ringstad G, Eide PK, Rognes ME, Mardal KA (2023) Human brain solute transport quantified by glymphatic MRI-informed biophysics during sleep and sleep deprivation. bioRxiv doi:10.1101/2023.01.01.522190

Wang S, Teng Y, Perdikaris P (2021) Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing 43(5):A3055–A3081, doi:10.1137/20M1318043

Xie L, Kang H, Xu Q, Chen MJ, Liao Y, Thiyagarajan M, O'Donnell J, Christensen DJ, Nicholson C, Iliff JJ, et al. (2013) Sleep Drives Metabolite Clearance from the Adult Brain. Science 342(6156):373–377, doi:10.1126/science.1241224

Zapf B, Haubner J, Kuchta M, Ringstad G, Eide PK, Mardal KA (2022) Investigating molecular transport in the human brain from mri with physics-informed neural networks. Scientific Reports 12(1):15475, doi:10.1038/s41598-022-19157-w

Zapf B, Valnes LM, Mardal KA, Zikatanov L (2023) Quantifying cerebrospinal fluid tracer concentration in the brain. In: Rognes ME, Vinje V, Dokken J, Valnes LM, Mardal KA (eds) MRI2FEM II: from magnetic resonance images to computational brain mechanics, Springer

Zeng Q, Bryngelson SH, Schäfer F (2022) Competitive physics informed networks. arXiv preprint arXiv:220411144 doi:10.48550/arXiv.2204.11144

Zhang A, Lipton ZC, Li M, Smola AJ (2021) Dive into deep learning. arXiv preprint arXiv:210611342 doi:10.48550/arXiv.2106.11342

Chapter 8

Two-compartment modelling of tracer transport in the brain

Jørgen N. Riseth, Timo Koch, Kent-André Mardal

8.1 Introduction

The postulation of the theory of the glymphatic system Iliff et al. (2012); Xie et al. (2013) has caused an upheaval in neuroscience and sleep research in the last decade. According to this theory (see also Chapter 1), extra-vascular fluid dynamics and solute transport are fundamental mechanisms for brain waste clearance, and neurodegenerative disorders such as Alzheimer's and Parkinson's disease may be explained in terms of long-term waste build-up in the brain tissue.

The extra-cellular volume fraction of the tissue available for fluid and solute transport increases during sleep by up to 60 % (Xie et al., 2013, in mice). Perivascular spaces have been described as possible highways for extra-vascular transport Iliff et al. (2012); Jessen et al. (2015). Waste clearance may be enhanced by cardiac vessel pulsations Mestre et al. (2018) and pulsations at lower frequencies due to respiration Vinje et al. (2019) and sleep waves Bojarskaite et al. (2023) suggesting fluid-structure interaction between the vasculature and the perivascular spaces as a driving force for solute mixing or transport Kelley et al. (2022).

Modeling transport processes on the organ scale (mm-dm) is a challenge since the direct description of the proposed mechanical processes occurring on the tissue micro-scale (μm to mm) and cellular scale (nm to μm) is not feasible[1]. Glymphatic-MRI Ringstad et al. (2017, 2018) allows us to investigate the transport of CSF-borne tracers on the organ scale in humans for the first time.

In an attempt to explain the transport phenomena observed by MRI, several mathematical models have been proposed on the basis of a single-continuum tissue

[1] The complete vasculature of the human brain comprises 650 km of vessels forming a dense network at the micro-scale with an average distance between micro-vessels in cortical brain tissue of around 50 µm.

description and advection-diffusion-reaction models Valnes et al. (2020a); Vinje et al. (2023); Ray et al. (2021).

The main conclusion from these works is that solute transport seems to enter and clear from brain tissue faster than extra-cellular diffusion alone can explain. However, with a single-continuum description of brain tissue, one cannot distinguish between possibly faster and slower pathways to identify underlying mechanisms and driving forces enhancing transport.

In this chapter, we investigate a multi-compartment diffusion-dispersion model for which a given brain tissue volume is formally split into several interacting continua[2]. The perivascular space (PVS) constitutes a pathway allowing for fast transport. The extra-cellular space (ECS) predominantly features diffusive transport Holter et al. (2017). In comparison with glymphatic-MRI data, following the procedures outlined in Vinje et al. (2023) and Chap. 5, we set out to identify and quantify the role of the individual compartments and their potential interaction. [3]

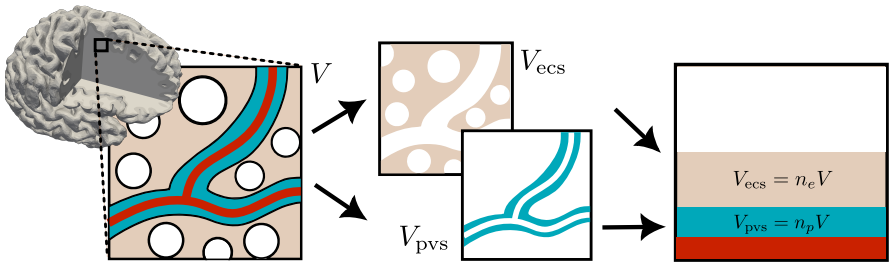


Fig. 8.1: **Multi-compartment description of brain tissue.** The microstructure comprises cells and fibers (white), blood vessels (red), perivascular space (PVS, blue), and extra-cellular space (ECS). The different compartments are formally separated from each other—here only the main compartments of interest are shown, ECS and PVS—and then described as overlapping continua occupying a certain volume fraction of the given tissue sample. The tissue microstructure only enters the model in the form of the model parameters in an average sense. Transport is described within and between the different compartments.

[2] Multi-compartment models can be mathematically derived via homogenization techniques Arbogast and Lehr (2006); Shipley and Chapman (2010) under consideration of the tissue microstructure (Figure 8.1) and have been applied to the brain to investigate, for example, oxygen transport El-Bouri et al. (2019), Alzheimer's disease progression Vardakis et al. (2021), transport in tumors Baxter and Jain (1989); Shipley and Chapman (2010); Ehlers and Wagner (2013), waste clearance Poulain et al. (2023).

[3] Source code and instructions for running the simulations and creating figures is available at `https://github.com/jorgenriseth/mri2fem-multicompartment.git`

8.2 Modeling and methods

We consider a multi-compartment description of brain tissue, as shown in Figure 8.1, comprising cells and fibers (s), blood vessels (b), perivascular space (p), and extracellular space (e) and model for each of the compartments $K = \{s, b, p, e\}$ the spatial and temporal evolution of the tracer concentrations $c_\alpha = c_\alpha(x,t),\ \alpha \in K$. The volume fraction occupied by each compartment is denoted by $n_\alpha,\ \alpha \in K$. We make the following simplifying assumptions:

(A1) tracer does not enter the cellular compartment ($c_s = 0$),

(A2) clearance of tracer that entered the blood is faster than the time scale of interest ($c_b = 0$),

(A3) the tissue is rigid on the relevant time scale but fast pulsatile vessel deformations may lead to enhanced mixing in the PVS compartment modeled as dispersion,

(A4) advective transport is negligible in all compartments,

(A5) no direct pathway from ECS to blood (tracer must pass through the PVS compartment).

The exchange between the PVS and the ECS compartment is modeled as diffusive transport through gaps between astrocyte endfeet which form a continuous sheath around the PVS; see Koch et al. (2023) and references therein. PVS and ECS are assumed to be filled by water-like interstitial fluid with tracer concentrations c_p and c_e.

As a result of A1 and A2, we only need to balance the amount of tracer in the ECS and PVS compartments while transfer to the blood appears as a sink term for the PVS compartment balance. Therefore, the resulting simplified model is subsequently called *two*-compartment model.

Two-compartment ECS-PVS diffusion-dispersion model. The spatial evolution of the fluid concentrations $c_\alpha(x,t)$ in the model domain Ω (human brain) over the time interval $[0,T]$ follows the governing equations:

$$\frac{\partial(n_p c_p)}{\partial t} - \operatorname{div}(n_p D_p^{\text{eff}} \nabla c_p) = t_{ep}(c_e - c_p) - t_{pb} c_p, \quad \text{in } \Omega \times (0,T], \tag{8.1a}$$

$$\frac{\partial(n_e c_e)}{\partial t} - \operatorname{div}(n_e D_e^{\text{eff}} \nabla c_e) = -t_{ep}(c_e - c_p), \quad \text{in } \Omega \times (0,T], \tag{8.1b}$$

subject to suitable boundary and initial conditions,

$$-n_p D_p^{\text{eff}} \nabla c_p \cdot \boldsymbol{n} = k_p(c_p - \hat{c}), \quad \text{on } \partial\Omega \times (0,T], \tag{8.1c}$$

$$-n_e D_e^{\text{eff}} \nabla c_e \cdot \boldsymbol{n} = k_e(c_e - \hat{c}), \quad \text{on } \partial\Omega \times (0,T], \tag{8.1d}$$

$$c_p(x,0) = c_{p,0}, \quad c_e(x,0) = c_{e0}, \quad \text{in } \Omega \times \{0\}, \tag{8.1e}$$

where D_α^{eff} is the effective diffusion coefficient (in compartment α) modeling diffusion and dispersion, and t_{ep} and t_{pb} are the diffusive transfer coefficients of the astrocyte endfeet sheath and the vessel wall, respectively. For the boundary conditions, $\mathbf{n}$ is the outward-oriented unit normal vector on the domain boundary $\partial\Omega$, and k_α is the conductivity of a surface membrane. We remark that the limit $k_\alpha \to 0$ corresponds to a non-permeable membrane and the limit $k_\alpha \to 0$ models the absence of a membrane. In all cases, $\hat{c}$ is the tracer concentration in the cerebrospinal fluid (CSF) surrounding the brain tissue in the subarachnoid space, the ventricles, and the spinal canal.

Relation to single-compartment diffusion model. For the interpretation of our results with the two-compartment model, we establish a relation to single-compartment models of previous works Valnes et al. (2020a); Vinje et al. (2023); Ray and Heys (2019). We make one of two simplifying assumptions:

(SC1) t_{ep} is sufficiently large, leading to local chemical equilibrium, $c_e \approx c_p$,

(SC2) the PVS volume fraction is negligible, $n_p \approx 0$, or equivalently $D_p^{\text{eff}} \approx D_e^{\text{eff}}$.

After summing the balance equations for each compartment, both SC1 and SC2 lead to a single-compartment model of the form

$$\frac{\partial(\phi c_{\text{F}})}{\partial t} - \operatorname{div}\left(\phi D^{\text{eff}} \nabla c_{\text{F}}\right) - t_{pb} c_{\text{F}} = 0 \tag{8.2a}$$

$$-\phi D^{\text{eff}} \nabla c_{\text{F}} \cdot \boldsymbol{n} = k(c_{\text{F}} - \hat{c}) \qquad \text{on } \partial\Omega \times (0, T] \tag{8.2b}$$

$$c_{\text{F}}(x, 0) = c_{\text{F},0} \qquad \text{in } \Omega \times \{0\}, \tag{8.2c}$$

where $\phi := n_p + n_e$ is the combined volume fraction of ECS and PVS, $c_{\text{F}} = c_e = c_p$ denotes the fluid concentration, $\phi D^{\text{eff}} := \sum_{\alpha \in K} n_\alpha D_\alpha^{\text{eff}} = n_p D_p^{\text{eff}} + n_e D_e^{\text{eff}}$, and $k = k_p + k_e$. In the case of spatially constant volume fractions, the single-compartment model can also be written in terms of the total concentration $c_T := \sum_{\alpha \in K} n_\alpha c_\alpha = \phi c_{\text{F}}$, cf. Vinje et al. (2023). We note that SC1 leads to $\phi D^{\text{eff}} = n_p D_p^{\text{eff}} + n_e D_e^{\text{eff}}$, whereas SC2 leads to $D^{\text{eff}} = D_e^{\text{eff}}$, that is SC2 precludes an effect of PVS-enhanced transport.

Model parameters. Table 8.1 summarizes estimates for two-compartment model parameter values together with a nominal value defining a reference scenario as the basis for the exploration of the input parameter space. In the following paragraphs, we motivate the chosen parameter ranges based on modeling assumptions and values and data available in the literature. A schematic representation of the micro-scale configuration for the motivation of the inter-compartmental transfer coefficients and the surface conductivity coefficients is shown in Figure 8.2.

Extra-cellular and perivascular volume fractions. The extra-cellular volume fraction, n_e, is estimated to be in the range of 15 % to 30 % in normal-appearing adult brain tissue with a typical value of 20 % Syková and Nicholson (2008) varying throughout the day and in disease. For instance a difference of 23 % during sleep

Table 8.1: **Model parameter values for the two-compartment diffusion model.** The nominal values (NV) define a reference scenario. Range defines ranges investigated in results section. PVS, perivascular space; ECS, extra-cellular space; SAS, subarachnoid space; GM, gray matter; WM, white matter.

symbol	description	unit	varied range	NV	comments / literature range
n_e	ECS volume fraction	-	fixed	0.20	0.14-0.30 Syková and Nicholson (2008); Xie et al. (2013)
n_p	PVS volume fraction	-	0.01–0.04	0.02	0.01-0.04 (see Section 8.2)
D_e^{eff}	Effective diffusion coefficient of ECS	$10^{-4}\text{mm}^2/\text{s}$	fixed	1.3	1.4 ± 0.4 (GM), 1.1 ± 0.3 (WM) (Valnes et al., 2020a, DTI), 1.3 to 1.9 (using Equation (8.3) and $\lambda = 1.4 - 1.7$ Syková and Nicholson (2008)).
D_p^{eff}	Effective diffusion coefficient in PVS	$10^{-4}\text{mm}^2/\text{s}$	1.3–130	3.9	Lower bound corresponds to single compartment assumption (SC2)
t_{ep}	Diffusive transfer coefficient between ECS and PVS	10^{-2}s^{-1}	0.030–3.1	2.9	0.17–3.1 Koch et al. (2023).
t_{pb}	Diffusive transfer coefficient between PVS and blood	10^{-5}s^{-1}	0–2.3	0.2	2.3 Vinje et al. (2023) (upper bound); 0 means BBB is impermeable for Gadobutrol
k_e	Pial surface conductivity	$10^{-5}\text{mm}\,\text{s}^{-1}$	0.026–2600	1.0	1.9 Koch et al. (2023) (arteriole endfoot sheath, upper bound)
k_p	PVS Robin boundary coefficient	$10^{-4}\text{mm}\,\text{s}^{-1}$	0.9–7.4	3.7	0.9–7.4 using Equation (8.7), layer thickness 20–40 μm Brøchner et al. (2015), and n_p between 0.01 and 0.04.

and 14 % when awake has been measured in adult mouse cortical brain tissue Xie et al. (2013).

Cerebral PVS volume fraction, n_p, is difficult to quantify in-vivo and it is unclear today whether fluid-filled spaces of significant size exist around vessels other than the largest vessels. PVS volume fraction was estimated with MRI in white matter at $1.14 \pm 0.43\%$ (range: 0.3% to 3.1%) Barisano et al. (2021), including PVS large enough to be visible in MRI.

Alternatively, n_p may be estimated from the cerebral blood volume fraction, n_b, assuming that the cross-sectional area of the lumen is proportional to the cross-sectional area of the PVS surrounding the vessel. Denoting by ψ the ratio of PVS area to vessel area: $n_p \approx \psi n_b$. Based on literature values for n_b[4] and the available

[4] The *cerebral blood volume fraction* is consistently determined to be around 3-6% in the gray matter and 1-3% in white matter using PET or MRI Leenders et al. (1990); Muizelaar et al. (1997); Ito et al. (2001). The arteries contribute about 20-30% Muizelaar et al. (1997); Ito et al. (2001); Hua et al. (2019) of the blood volume, whereas the rest is contained in capillaries and veins. Methods based on PET and MRI cannot resolve the vessel microstructure and need a make additional

data on pial vessel PVS spaces[5], we estimate $n_p = 1\%$ ($\psi = 0.3$ for all vessels with $n_b = 0.03$) as lower bound and $n_p = 4\%$ ($\psi = 0.3$ for veins and capillaries vessels and $\psi = 1.7$ for arteries contributing 30% of $n_b = 0.06$) as upper bound.

Effective diffusion and dispersion coefficients. Transport in the ECS is assumed to happen predominantly by diffusion Holter et al. (2017) due to low Péclet numbers. Diffusion in porous media is hindered by the tortuosity of the pore space and we model the effective diffusion coefficient by Syková and Nicholson (2008)

$$D_e^{\text{eff}} = \lambda_e^{-2} D^*, \tag{8.3}$$

where $\lambda_e = \sqrt{D^*/D_e^{\text{eff}}}$ denotes the tortuosity, and D^* the binary diffusion coefficient of gadobutrol in water (interstitial fluid). Using the Stokes-Einstein equation, which holds for the diffusion of spherical particles in a liquid at low Reynolds number, we can estimate the binary diffusion coefficient of gadobutrol in water at 37 °C,

$$D^\star = \frac{k_{\text{B}} T}{6\pi\mu R} = 3.7 \times 10^{-4}\,\text{mm}^2/\text{s} \tag{8.4}$$

where $k_{\text{B}} = 1.38 \times 10^{-23}\,\text{m}^2\text{kg/s}^2/\text{K}$, $T = 310.15\,\text{K}$ (= 37 °C), $R = 0.9\,\text{nm}$ for gadobutrol Guthausen et al. (2015), and μ is the dynamic viscosity of the fluid (for water at 37 °C, $\mu = 0.69\,\text{mPa s}$). The tortuosity of brain tissue has been estimated to be in the range of 1.4 to 1.7 Syková and Nicholson (2008) using real-time iontophoresis. This corresponds to a range for D_e^{eff} of $1.3\,\text{mm}^2/\text{s}$ to $1.9\,\text{mm}^2/\text{s}$. These values match well with what is estimated with diffusion-weighted MRI sequences. For instance, in Valnes et al. (2020a), D_e^{eff} was estimated to be 1.4 ± 0.4 in gray matter, and 1.1 ± 0.3 in white matter using diffusion tensor imaging.

Transport in the PVS is assumed to happen by both diffusion and dispersion caused by oscillatory velocity fields due to vasomotion Asgari et al. (2016); Bojarskaite et al. (2023); Keith Sharp et al. (2019),

$$D_p^{\text{eff}} = \lambda_p^{-2} D^* + D^{\text{disp}}. \tag{8.5}$$

---

assumptions on hematocrit and correct for flow artifacts Muizelaar et al. (1997). In Lauwers et al. (2008)Lauwers et al. (2008) blood volume fraction in gray matter was determined from 3D vessel reconstruction of stained brain slices Duvernoy et al. (1981) to be 2.7% (larger than the 1.5% reported for mice cortical microstructural data Blinder et al. (2013), see Schmid et al. (2019) for a comparison between species).

[5] The *ratio of PVS area to vessel lumen area* in pial vessels has been estimated in mice to be 1.3-1.4 Schain et al. (2017); Mestre et al. (2018) in pial arteries, 0.3 in veins Schain et al. (2017), 0.07-0.3 in capillaries Tithof et al. (2022) based on a small number of samples. In Raicevic et al. (2023), broad distributions of the ratio were reported in mouse pial arteries with values ranging from 0.3 to 6 and it was concluded that vessel lumen area is mildly correlated with PVS area (with a ratio of 1.66 and $R^2 = 0.2$). However, since pial vessels are situated within the subarachnoid space, they are not part of our simulation domain. Moreover, functional tissue is tightly packed with cells leaving little to no room for fluid-filled perivascular spaces in comparison with the subarachnoid space.

Using mathematical models Asgari et al. (2016); Bojarskaite et al. (2023); Keith Sharp et al. (2019), D_p^{eff} has been estimated to be up to $2D^\star$ in simplified geometries.

In the absence of data, we estimate $\lambda_e \geq \lambda_p \geq 1$.

When estimating the parameters λ_p and D^{disp} in comparison with clinical data, they are structurally unidentifiable since both appear only in the expression for D_p^{eff}. Therefore, for our parameter variation study, we investigate instead the effect of the ratio $\gamma := D_p^{\text{eff}}/D_e^{\text{eff}}$. In case SC1 (or SC2) holds, we obtain an effective diffusion coefficient in terms of γ of $D^{\text{eff}} = \frac{(n_p\gamma+n_e)}{n_p+n_e}\lambda_e^{-2}D^*$, for the single-compartment model, Equation (8.2). For our parameter study, we choose a lower bound for γ of 1 (corresponding to SC2 and $D^{\text{eff}} = D_e^{\text{eff}}$) and an upper bound of 100 (corresponding to a single compartment coefficient of $D^{\text{eff}} \approx 5D^*$ if SC1 holds).

Inter-compartmental transfer coefficients.

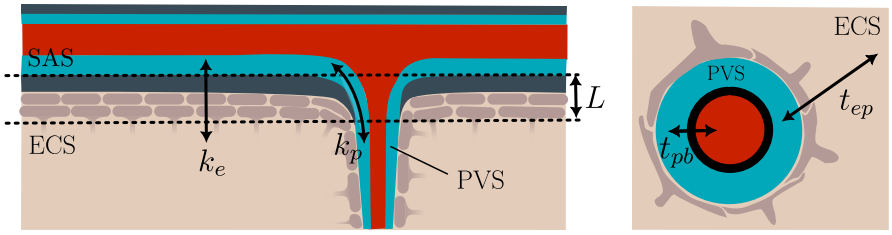


Fig. 8.2: **Boundary conditions and inter-compartmental transfer.** Left: at the interface of the brain tissue to the subarachnoid space (SAS), diffusive transport between SAS and extra-cellular space (ECS), and SAS and perivascular space (PVS) is described as diffusive transport across a membrane layer with conductivity k_e and k_p, respectively. Right: intercompartmental transfer between PVS and blood, and PVS and ECS is described by the transfer coefficients t_{pb} and t_{ep}, respectively.

We model the transfer coefficients between two compartments α and β by

$$t_{\alpha\beta} = \sigma_{\alpha\beta} C_{\alpha\beta} D^* \tag{8.6}$$

where $\sigma_{\alpha\beta}$ is the volume-specific surface area of the inter-compartmental interface (units m^{-1}) and $C_{\alpha\beta}$ (units m^{-1}) is a coefficient proportional to the inverse average path length a molecule has to travel to get from compartment α to compartment β.

The volume-specific surface area of blood vessels, σ_{bp}, has been reported to be between $10\,\text{mm}^{-1}$ and $13\,\text{mm}^{-1}$ in humans Lauwers et al. (2008). The blood-brain barrier is commonly assumed to be virtually impermeable to Gadobutrol in healthy tissue Weinmann et al. (1984); Jost et al. (2017). However, typical perfusion MRI sequences assess leakage on a time scale of minutes Sourbron et al. (2009)Sourbron et al. (2009). When analyzing tracer transport with glymphatic MRI, the relevant

time scale is days and even very small transfer coefficients may have a significant effect on transport. In Koch et al. (2020), it was estimated with a micro-scale model that a transfer coefficient equivalent to a value of $t_{pb} \lessapprox 1 \times 10^{-5}\,\mathrm{s}^{-1}$ is unlikely to cause a significant leakage effect on the signal during a typical 2 min duration of a perfusion DSC-MRI measurement[6]. In Chagnot et al. (2021), the authors review DCE-MRI studies of contrast agent leakage across the BBB in patients with dementia as well as normal-appearing aging with long acquisition times (5 min to 25 min). They report values for t_{pb}[7] from $1 \times 10^{-8}\,\mathrm{s}^{-1}$ to $4 \times 10^{-5}\,\mathrm{s}^{-1}$ where the larger values are attributes to BBB breakdown due to various forms of dementia. In comparison with MRI data, the largest clearance rate estimated with computer simulations by (Vinje et al., 2023, supp.) corresponds to $t_{pb} \lessapprox 2.1 \times 10^{-5}\,\mathrm{s}^{-1}$. Here, we take t_{pb} in the range $0\,\mathrm{s}^{-1}$ to $2.1 \times 10^{-5}\,\mathrm{s}^{-1}$.

Based on geometric reconstructions of two cortical vascular networks (mouse) and a theoretical model for the astrocyte endfoot sheath geometry, the authors of Koch et al. (2023) estimated a value for $\sigma_{ep}C_{ep}$ of $72\,\mathrm{mm}^{-2} to 85\,\mathrm{mm}^{-2}$ corresponding to transfer coefficients of $t_{ep} = 2.7 \times 10^{-2}\,\mathrm{s}^{-1} to 3.1 \times 10^{-2}\,\mathrm{s}^{-1}$, using $D^\star = 3.7 \times 10^{-4}\,\mathrm{mm}^2/\mathrm{s}$. They also estimate that along a pathway through the ECS and across the endfoot sheath, the effective permeability of the (longer) ECS pathway is approximately equal to the (shorter) pathway through the endfoot sheath and would add a factor of 0.5 in the estimate of t_{ep}. Furthermore, they estimated C_{ep} of the astrocyte endfoot sheath for arterioles, capillaries, and veins, with values ranging from $0.5\,\mathrm{mm}^{-1} to 6\,\mathrm{mm}^{-1}$, the smallest of which (corresponding to arterioles) being an order of magnitude smaller than the network average. Assuming this value for all relevant PVS, we obtain $t_{ep} \approx 1.7 \times 10^{-3}\,\mathrm{s}^{-1}$ ($C_{ep} = 0.25\,\mathrm{mm}^{-1}$, $\sigma_{ep} = 18\,\mathrm{mm}^{-1}$ (Koch et al., 2023, Tab.1), $D^\star = 3.7 \times 10^{-4}\,\mathrm{mm}^2/\mathrm{s}$).

Boundary conditions and surface membrane. Assuming that there is no membrane obstructing the pathway from PVS to SAS, we may estimate the boundary conductivity for the PVS as

$$k_p = n_p L^{-1} D^*, \tag{8.7}$$

where L is the thickness of the boundary layer, see Figure 8.2. The glia limitans of an adult human brain has been estimated between 20-40μm Brøchner et al. (2015). Assuming $L = 20\,\mu\mathrm{m}$ and $n_p = 0.04$ yields $k_p = 7.4 \times 10^{-4}\,\mathrm{mm\,s}^{-1}$.

The conductivity k_e of the membrane layer between the ECS and the subarachnoid space, forming the outer boundary of the ECS, is modeled as

$$k_e = C_{\mathrm{pial}} D^*, \tag{8.8}$$

[6] Combining a diffusive wall permeability of $1 \times 10^{-10}\,\mathrm{m\,s}^{-1}$ Koch et al. (2020) with a volume-specific area of $10\,\mathrm{mm}^{-1}$ Lauwers et al. (2008)

[7] Assuming t_{pb} is equal to K_{trans} of the Patlak model Patlak et al. (1983); Sourbron et al. (2009) used in DCE-MRI signal analysis. Note that t_{pb} is a parameter motivated on the micro-scale and is likely larger than the corresponding voxel-scale K_{trans} which includes the permeability of the ECS and the astrocyte endfoot sheath and appears in combination with the average inter-compartmental concentration gradient instead of the micro-scale concentration gradient between blood and PVS across the endothelium.

where C_{pial} is a membrane coefficient proportional to the available pore space for transport across the layer and inversely proportional to its thickness. To estimate an upper bound for C_{pial}, we rescale the membrane coefficients for the astrocyte endoot sheath around the largest penetrating arterioles $C_M = 0.5\,\text{mm}$ (with a corresponding sheath thickness of $h_{\text{ES}} \approx 2.8\,\mu\text{m}$[8]) as $C_{pial} = C_M h_{\text{ES}} L^{-1}$, where L is again the thickness of the boundary layer. For $L = 20\mu m$ this corresponds to an upper bound of $k_e = 2.6 \times 10^{-5}\,\text{mm}\,\text{s}^{-1}$. As a lower bound estimate, we consider the membrane layer to be virtually impermeable, $k_e \to 0$.

As boundary conditions, we consider both Dirichlet boundary conditions (equivalent to the limit $k_\alpha \to \infty$) based on tissue concentration estimates from MRI data, or Robin boundary conditions assuming a given tracer concentration in the SAS and ventricles.

In the case of Dirichlet boundary conditions, we use the interpolant u_d of the MRI data introduced in detail in Chapter 5. Between time steps with data (4 h, 24 h, 48 h, 70 h) the values are interpolated linearly in time on the domain boundary $\partial\Omega$. Since u_d represents the total concentration, the value must be divided by ϕ to obtain the fluid concentration in the compartments. To uniquely determine the fraction of the MRI signal from each of the compartments, we assume that the fluid concentrations of the compartments are in equilibrium, and thus $\hat{c} = u_d\,\phi = u_d/(n_e + n_p)$.

In the case of Robin boundary conditions, we model the CSF concentration in the SAS on the pial surface $\partial\Omega_{\text{sas}}$ and the ventricular surface $\partial\Omega_{\text{vent}}$ with the ansatz

$$\hat{c}(t, \partial\Omega_\zeta) = a_\zeta \phi^{-1} \left(-e^{-t/\tau_1} + e^{-t/\tau_2}\right), \quad \zeta = \{\text{sas}, \text{vent}\}, \tag{8.9}$$

and $\tau_1 = 4.43 \times 10^4$ s, $\tau_2 = 8.5 \times 10^4$ s, $a_{\text{sas}} = 0.52\,\text{mm}^2/\text{s}$, and $a_{\text{vent}} = 0.2\,\text{mm}^2/\text{s}$. A comparison between the CSF concentration calculated with Equation (8.9) and the surface-averaged concentrations of the interpolated MRI-data u_d is shown in Figure 8.3.

Initial concentration. Since the tracer is administered by intrathecal injection, there is initially no tracer in the brain tissue, $c_{p,0} = c_{e,0} = 0$.

8.3 Results

Parameter variations. A reference scenario is defined by choosing the parameters according to the nominal values given in Table 8.1 (NV). This implies Robin boundary conditions as described in the previous section.

To investigate the parameter sensitivity of the two-compartment model (TC), we varied individual parameters while keeping the other parameters fixed (reference scenario). The evolution of the total solute content in the brain over time is shown in

[8] Using the formula $h_{\text{ES}} = 1 + 0.15(r_v - 3)$ of Koch et al. (2023) with a vessel radius $r_v = 15\,\mu\text{m}$

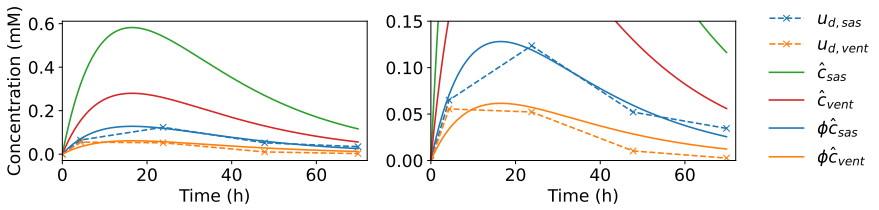


Fig. 8.3: Estimated mean subarachnoid space (SAS) tracer concentration and mean tracer concentration in the ventricles (solid) in comparison with the boundary data (u_d) mean as estimated from MRI data (dashed). First, total concentrations in the tissue are obtained by fitting Equation (8.9) to the mean of u_d (right panel). Second, fluid concentrations in the SAS and the ventricles at the brain surface are estimated by division by the volume fraction available for fluid flow in the brain, $\phi = 0.2$ (left panel).

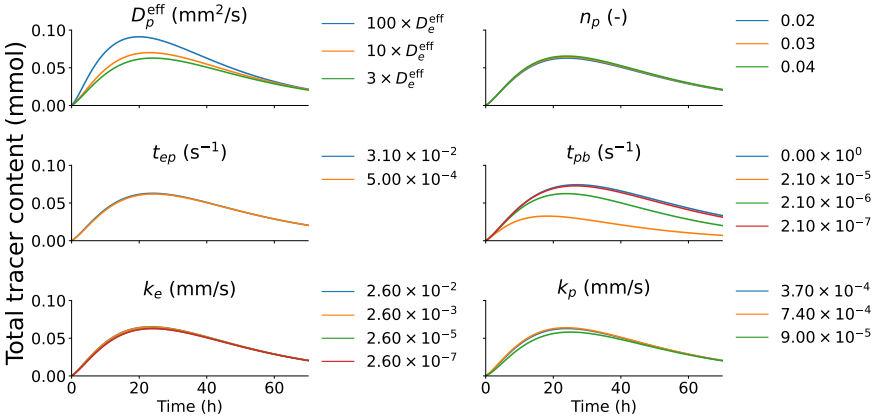


Fig. 8.4: The total tracer content in the brain for varying two-compartment model parameters over time. The parameters are described in Table 8.1. One parameter is varies while all other parameter are fixed at their nominal value listed in Table 8.1.

Figure 8.4. The TC model is most sensitive to variations of the effective perivascular diffusion coefficient, D_p^{eff}, and the transfer coefficient from PVS to blood, t_{pb}. For D_p^{eff}, there is an increase in peak tracer content of 11 % ($D_p^{\text{eff}} = 10D_e^{\text{eff}}$) and 44 % ($D_p^{\text{eff}} = 100D_e^{\text{eff}}$) with respect to the reference scenario ($D_p^{\text{eff}} = 3D_e^{\text{eff}}$). For the transfer coefficient from PVS to blood, we observe a 48 % decrease ($t_{pb} = 2.1 \times 10^{-6}\,\text{s}^{-1}$) and a 19 % increase ($t_{pb} = 0\,\text{s}^{-1}$), respectively. For the remaining parameters, the relative difference between the minimum and maximum value for peak tracer content is ≤ 5% for perivascular volume fraction; ≤ 1.5% for transfer between ECS and PVS; ≤ 4% for ECS boundary conductivity; and ≤ 8% for the perivascular boundary conductivity.

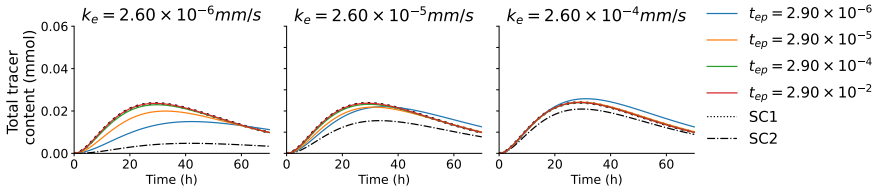


Fig. 8.5: The total tracer content in the brain for varying effective transfer coefficients between ECS and PVS, t_{ep}, and extracellular surface-layer conductivity k_e, for the different models. (TC), two-compartment; (SC1)-(SC2) single-compartment models as stated in Equation (8.2).

Next, we explored scenarios in which the difference between the two-compartment model (featuring local concentration differences between PVS and ECS) and the single-compartment models (local equilibrium) becomes apparent. To this end, we varied both the ECS surface conductivity k_e and the ECS-PVS transfer coefficient t_{ep}, keeping all other parameters as in the reference scenario. The resulting tracer content evolution in the brain tissue is presented in Figure 8.5. Decreasing k_e limits the tracer influx and efflux rate at the brain tissue surface, increasing the relative importance of the PVS pathway. Decreasing t_{ep} limits the exchange rate between ECS and PVS compartment.

Decreasing ECS surface conductivity and ECS-PVS transfer simultaneously, we see lower tracer influx and lower tracer efflux rates with respect to the reference scenario. A large difference between the equilibrium model SC1 (or TC model with large t_{ep}) and the two-compartment model is evident for low t_{ep} and, in particular, with decreasing k_e. As expected the SC2 model matches the results of the TC model for high t_{ep} which is the limit under which it was derived.

Figure 8.6(first row) shows the spatial concentration distribution ($t_{ep} = 2.0 \times 10^{-6}\,\mathrm{s}^{-1}$, $k_e = 2.6 \times 10^{-6}\,\mathrm{mm\,s}^{-1}$) in a coronal slice with ($t_{pb} = 0.2 \times 10^{-5}\,\mathrm{s}^{-1}$) and without ($t_{pb} = 0$) tracer clearance to blood. Without clearance to blood, in the influx phase ($\leq 24\,\mathrm{h}$), the tracer concentration increases monotonically from the boundary to the center of the brain, while the concentration is smaller in the ECS than in the PVS ($c_e - c_p < 0$). However, clearance from PVS to blood leads to a spatial concentration distribution, where deep within the brain $c_e - c_p \geq 0$ and $c_e - c_p < 0$ is only observed in a small region close to the brain surface, cf. Figure 8.6 (second row). For the efflux phase ($> 24\,\mathrm{h}$), a monotone profile is established. The tracer is cleared both into the blood and from the surface into CSF. In the efflux phase $c_e - c_p \leq 0$ everywhere, since the tracer is cleared faster from the PVS compartment than from the ECS compartment.

We remark that similar qualitative observations hold for the reference scenario although with much smaller concentration differences between PVS and ECS compartments (less than 0.1 % with respect to the maximum concentration after 24 h).

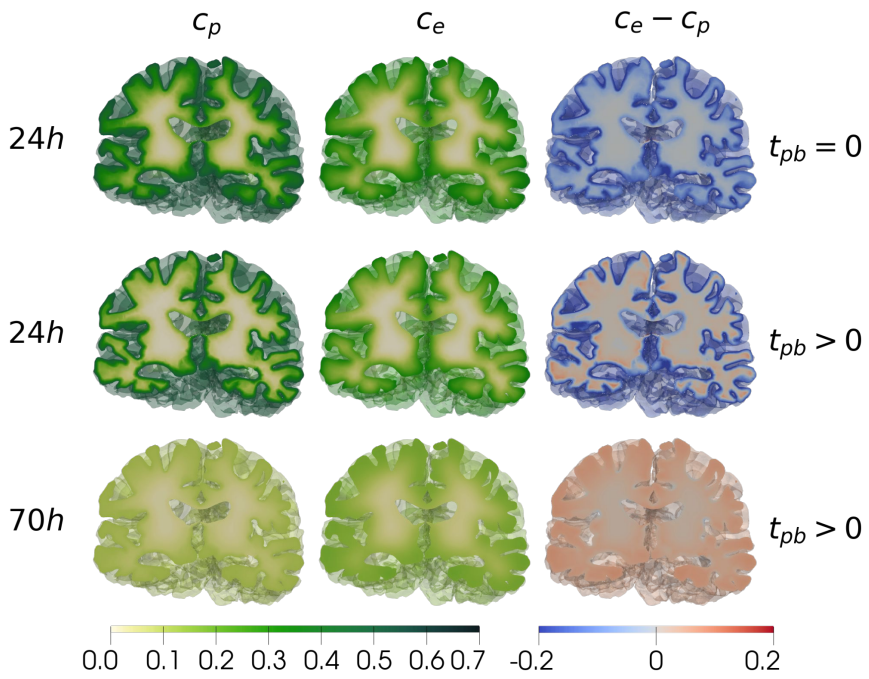


Fig. 8.6: Spatial distribution of the fluid concentrations in PVS compartment, c_p, and ECS compartment, c_e, at two different time points and for two different blood clearance rates, $t_{pb} = 0.2 \times 10^{-5}\ \mathrm{s}^{-1}$ and $t_{pb} = 0$, on a coronal slice.

Thus, in the reference scenario, the fluid concentrations in both compartments are in equilibrium and assumption SC1 is fulfilled.

Comparison with MRI data. We analyzed the brain tracer content evolution simulated with Dirichlet boundary conditions ($k_\alpha \to \infty$) set by MRI data on the boundary and compared it with the corresponding MRI-based tracer content estimates within the brain tissue. The parameters were set to the reference scenario (except for k_α). Figure 8.7 shows the total concentration c_T estimated from MRI data and simulated with the TC model on a transverse slice over 72 h.

Figure 8.8 shows the total tracer content in the whole brain, and separately in gray and white matter. The peak tracer content for the SC2 model is reduced with respect to the TC model (6%, 3%, 11% decrease in the whole brain, gray matter, white matter, respectively), while SC1 and TC coincide, signifying local equilibrium between ECS and PVS concentrations. The simulation results and data match well without systematically optimizing the parameters. With the reference parameter configuration, the model overestimates the tracer content compared to data in both white and gray matter in the interval between approximately 10 h to 55 h. Moreover, it can be observed that the maximum solute content is attained later than 24 h (where

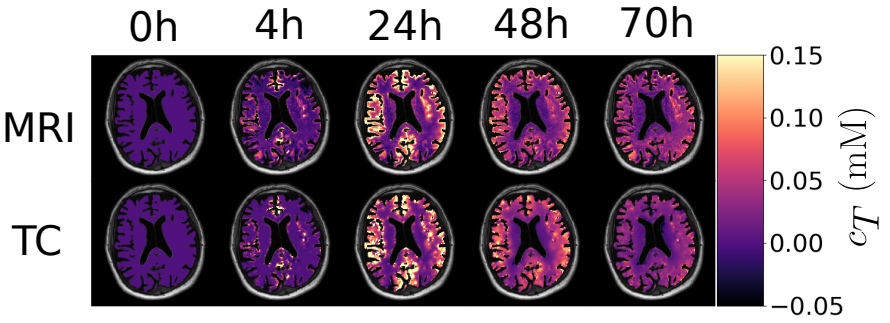


Fig. 8.7: Spatial distributions of the total concentration, c_T, for the two-compartment model (TC) and the concentration estimates derived from MRI data over time. The true value range (−0.2, 1.2) is clipped to the range (−0.05, 0.15) to remove outliers impairing visual interpretability. Negative concentrations result from noise in the MRI data since the raw MRI data has a low signal-to-noise ratio Valnes et al. (2020b).

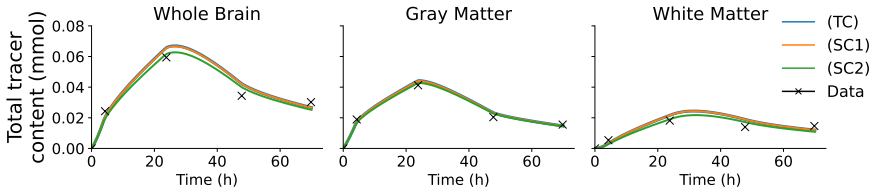


Fig. 8.8: Total tracer content in the whole brain, gray matter, white matter for data derived from MRI and simulations with the two-compartment and single-compartment models. (TC), two-compartment; (SC1)-(SC2) single-compartment models as introduced with Equation (8.2).

the data shows the maximum value). This seems counter-intuitive at first since the boundary data for the simulation is derived from the MRI data. However, due to the blood clearance (as shown in Figure 8.6, the tracer concentration close to the surface is likely lower than the concentration prescribed at the boundary, and remains lower for a few hours after the 24 h time point resulting in continued influx and delaying the beginning of the efflux phase.

8.4 Discussion

Fast equilibrium between PVS and ECS. The single-compartment model (SC1) is derived from the two-compartment model (TC) by assuming a sufficiently high transfer coefficient t_{ep} for the compartments to be in equilibrium, an assumption

that is supported by the comparison with the concentration evolution estimated from MRI data Figure 8.7. For a local non-equilibrium configuration to establish between the PVS and ECS compartments, the transfer coefficient t_{ep} (which is proportional to the astrocyte endfoot sheath permeability) would need to be an order of magnitude smaller than the estimated lower bound supported by data in the literature (cf. Table 8.1); even for low ECS conductivity at the brain surface (Figure 8.5). We can support the observations with the TC model by simple theoretical considerations: The ratio of ECS-PVS transfer rate across the astrocyte endfoot sheath and the transversal diffusion rate in the ECS is given by the Damköhler number[9], $\mathrm{Da} = t_{ep} L_{ep} (n_e D_e^{\mathrm{eff}} \sigma_{ep})^{-1}$, where L_{ep} is a characteristic transversal length scale (the mean inter-vessel distance). For $L_{ep} = 50\,\mu\mathrm{m}$ (as mean intervessel distance), we find $\mathrm{Da} \approx 0.3 to 3$ and conclude that both rates are approximately equal. A corresponding characteristic time scale is given by $T = t_{ep}^{-1} \approx 30\,\mathrm{s} to 600\,\mathrm{s}$ (using the ranges from Table 8.1). The time scale for longitudinal transport in the PVS is given by $T_d = L_\ell^2 (D_p^{\mathrm{eff}})^{-1}$, where L_ℓ is a characteristic longitudinal length scale. For $L_\ell = 0.5\,\mathrm{cm}$ and $D_e^{\mathrm{eff}} \leq D_p^{\mathrm{eff}} < 100 D_e^{\mathrm{eff}}$, we obtain $T_d \approx 2 \times 10^3\,\mathrm{s} to 2 \times 10^5\,\mathrm{s}$. Hence, even for the upper estimate for D_p^{eff}, the transversal transport rate dominates the longitudinal transport rate leading to a fast equilibrium of the concentrations in PVS and ECS compartments.

Brain surface membranes have little impact on influx and efflux rates. Increasing the PVS surface conductivities beyond the reference scenario (NV) has very little impact on the brain tracer content (Figure 8.4), indicating that for $k_p \gtrapprox 1 \times 10^{-4}\,\mathrm{mm/s}$ the surface membrane poses no significant barrier for tracer influx along the PVS. Similarly, given fast ECS-PVS exchange, varying the ECS surface membrane conductivity k_{ep} within the range supported by literature values has little impact on the tracer content in the brain. The tracer can circumvent the ECS surface layer by entering the brain along the more permeable PVS spreading to the ECS through the astrocyte endfoot sheath.

Parameter uncertainty. The two parameters with the most significant impact on the TC model (in the reference scenario) are the perivascular diffusion coefficient D_p^{eff} and the PVS-blood transfer coefficient t_{pb}, cf. Figure 8.5.

The effective diffusion coefficient, D^{eff} of the single-compartment model (SC1) is related to the individual effective diffusion coefficients of the compartments and $D^{\mathrm{eff}}/D_e^{\mathrm{eff}} = (n_p \gamma + n_e)/(n_p + n_e)$ (where $\gamma = D_e^{\mathrm{eff}}/D_p^{\mathrm{eff}}$). The ratio $D^{\mathrm{eff}}/D_e^{\mathrm{eff}}$ was estimated in Ray et al. (2021) based on MRI data in mice at 10 to 25, corresponding to $\gamma = 100 to 265$ (with $n_e/n_p = 10$). The authors of Vinje et al. (2023) estimated $D^{\mathrm{eff}}/D_e^{\mathrm{eff}}$ between 1.1 and 7.0, corresponding to $\gamma = 2 to 67$. For the single sub-

[9] Balancing fluxes over the astrocyte endfoot sheath interface Γ yields $\int_\Gamma -n_e D_e^{\mathrm{eff}} (\nabla c_e \cdot \boldsymbol{n})\,\mathrm{d}s = \int_\Omega t_{ep}(c_p - c_e)\,\mathrm{d}x$. Assuming uniform parameters and normal fluxes, we can integrate and nondimensionalize ($\nabla^\star := L_{ep}\nabla$, $c_\alpha^\star := C_0 c_\alpha$, $\sigma_{ep} = (\int_\Gamma \mathrm{d}s)(\int_\Omega \mathrm{d}x)^{-1}$) to arrive at $-\nabla^* c_e^* \cdot \boldsymbol{n} = \mathrm{Da}\,(c_p^* - c_e^*)$, where $\mathrm{Da} = t_{ep} L_{ep} (n_e D_e^{\mathrm{eff}} \sigma_{ep})^{-1}$.

ject investigated here, we obtained a good fit between data and simulation with $D^{\text{eff}}/D_e^{\text{eff}} = 1$.

The PVS-blood transfer coefficient t_{pb} models clearance of Gadobutrol across the BBB. The reference value of $t_{pb} = 2.1 \times 10^{-6}\,\text{s}^{-1}$ is within the range of the blood-to-brain transport coefficients of gadolunium-based contrast agents reported in the review article Chagnot et al. (2021) of $1 \times 10^{-8}\,\text{s}^{-1}$ to $3.4 \times 10^{-5}\,\text{s}^{-1}$ for normal aging, $5 \times 10^{-7}\,\text{s}^{-1}$ to $1.3 \times 10^{-5}\,\text{s}^{-1}$ for cerebral small vessel disease, and $1 \times 10^{-8}\,\text{s}^{-1}$ to $4 \times 10^{-5}\,\text{s}^{-1}$ in patients with mild cognitive impairment or Alzheimer's disease. Disorders such as multiple sclerosis and glioma are known to cause a breakdown of the BBB, leading to an increase of up to several magnitudes in blood vessel wall permeability. The MRI data used in this study falls in the category of normal aging.

8.5 Conclusion

A two-compartment model of tracer transport in brain tissue allowed us to investigate the effect of compartment-specific diffusion coefficients, and membrane-specific transfer coefficients and we related the model parameters of the two-compartment model to previously employed single-compartment models. We found that for a wide range of parameters, a single-compartment model (SC1) serves as a good approximation to the two-compartment model due to the fast transfer between the ECS and PVS compartments through the astrocyte endfoot sheath. In comparison with concentration distributions from reference MRI data, we find that the combination of extracellular diffusion, fast exchange between ECS and PVS compartment, and tracer clearance to blood provide a model that explains the tracer evolution in the patient-specific brain tissue geometry over 70 h after intrathecal injection of a gadolinium-based contrast agent.

References

Arbogast T, Lehr HL (2006) Homogenization of a Darcy–Stokes system modeling vuggy porous media. Computational Geosciences 10(3):291–302, doi:10.1007/s10596-006-9024-8

Asgari M, de Zélicourt D, Kurtcuoglu V (2016) Glymphatic solute transport does not require bulk flow. Sci Rep 6(1):38635, doi:10.1038/srep38635

Barisano G, Sheikh-Bahaei N, Law M, Toga AW, Sepehrband F (2021) Body mass index, time of day and genetics affect perivascular spaces in the white matter. Journal of Cerebral Blood Flow & Metabolism 41(7):1563–1578, doi:10.1177/0271678X20972856

Baxter LT, Jain RK (1989) Transport of fluid and macromolecules in tumors. i. role of interstitial pressure and convection. Microvascular Research 37(1):77–104, doi:10.1016/0026-2862(89)90074-5

Blinder P, Tsai PS, Kaufhold JP, Knutsen PM, Suhl H, Kleinfeld D (2013) The cortical angiome: an interconnected vascular network with noncolumnar patterns of blood flow. Nature Neuroscience 16(7):889–897, doi:10.1038/nn.3426

Bojarskaite L, Vallet A, Bjørnstad DM, Gullestad Binder KM, Cunen C, Heuser K, Kuchta M, Mardal KA, Enger R (2023) Sleep cycle-dependent vascular dynamics in male mice and the predicted effects on perivascular cerebrospinal fluid flow and solute transport. Nature Communications 14(1):953, doi:10.1038/s41467-023-36643-5

Brøchner CB, Holst CB, Møllgård K (2015) Outer brain barriers in rat and human development. Frontiers in Neuroscience 9:75, doi:10.3389/fnins.2015.00075, URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4360706/

Chagnot A, Barnes SR, Montagne A (2021) Magnetic Resonance Imaging of Blood–Brain Barrier permeability in Dementia. Neuroscience 474:14–29, doi:10.1016/j.neuroscience.2021.08.003, URL https://www.sciencedirect.com/science/article/pii/S0306452221004024

Duvernoy H, Delon S, Vannson J (1981) Cortical blood vessels of the human brain. Brain Research Bulletin 7(5):519–579, doi:10.1016/0361-9230(81)90007-1

Ehlers W, Wagner A (2013) Multi-component modelling of human brain tissue: a contribution to the constitutive and computational description of deformation, flow and diffusion processes with application to the invasive drug-delivery problem. Computer Methods in Biomechanics and Biomedical Engineering 18(8):861–879, doi:10.1080/10255842.2013.853754

El-Bouri W, Bing Y, Józsa T, Payne S (2019) A novel multi-scale, multi-compartment model of oxygen transport—towards in-silico clinical trials in the entire human brain. In: CompBioMed Conference, pp 25–27

Guthausen G, Machado JR, Luy B, Baniodeh A, Powell AK, Krämer S, Ranzinger F, Herrling MP, Lackner S, Horn H (2015) Characterisation and application of ultra-high spin clusters as magnetic resonance relaxation agents. Dalton Transactions 44(11):5032–5040, doi:10.1039/c4dt02916j

Holter KE, Kehlet B, Devor A, Sejnowski TJ, Dale AM, Omholt SW, Ottersen OP, Nagelhus EA, Mardal KA, Pettersen KH (2017) Interstitial solute transport in 3D reconstructed neuropil occurs by diffusion rather than bulk flow. Proceedings of the National Academy of Sciences 114(37):9894–9899, doi:10.1073/pnas.1706942114

Hua J, Liu P, Kim T, Donahue M, Rane S, Chen JJ, Qin Q, Kim SG (2019) MRI techniques to measure arterial and venous cerebral blood volume. NeuroImage 187:17–31, doi:10.1016/j.neuroimage.2018.02.027

Iliff JJ, Wang M, Liao Y, Plogg BA, Peng W, Gundersen GA, Benveniste H, Vates GE, Deane R, Goldman SA, Nagelhus EA, Nedergaard M (2012) A Paravascular Pathway Facilitates CSF Flow Through the Brain Parenchyma and the Clearance of Interstitial Solutes, Including Amyloid β. Science Translational Medicine 4(147):147ra111–147ra111, doi:10.1126/scitranslmed.3003748

Ito H, Kanno I, Iida H, Hatazawa J, Shimosegawa E, Tamura H, Okudera T (2001) Arterial fraction of cerebral blood volume in humans measured by positron emission tomography. Annals of Nuclear Medicine 15(2):111–116, doi:10.1007/bf02988600

Jessen NA, Munk ASF, Lundgaard I, Nedergaard M (2015) The Glymphatic System: A Beginner's Guide. Neurochem Res 40(12):2583–2599, doi:10.1007/s11064-015-1581-6

Jost G, Frenzel T, Lohrke J, Lenhard DC, Naganawa S, Pietsch H (2017) Penetration and distribution of gadolinium-based contrast agents into the cerebrospinal fluid in healthy rats: a potential pathway of entry into the brain tissue. European Radiology 27(7):2877–2885, doi:10.1007/s00330-016-4654-2, URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5486780/

Keith Sharp M, Carare RO, Martin BA (2019) Dispersion in porous media in oscillatory flow between flat plates: applications to intrathecal, periarterial and paraarterial solute transport in the central nervous system. Fluids and Barriers of the CNS 16(1):1–17

Kelley DH, Bohr T, Hjorth PG, Holst SC, Hrabětová S, Kiviniemi V, Lilius T, Lundgaard I, Mardal KA, Martens EA, et al. (2022) The glymphatic system: Current understanding and modeling. Iscience 25:104987, doi:10.1016/j.isci.2022.104987

Koch T, Flemisch B, Helmig R, Wiest R, Obrist D (2020) A multiscale subvoxel perfusion model to estimate diffusive capillary wall conductivity in multiple sclerosis lesions from perfusion MRI data. International Journal for Numerical Methods in Biomedical Engineering 36(2), doi:10.1002/cnm.3298

Koch T, Vinje V, Mardal KA (2023) Estimates of the permeability of extra-cellular pathways through the astrocyte endfoot sheath. Fluids and Barriers of the CNS 20(1):20, doi:10.1186/s12987-023-00421-8

Lauwers F, Cassot F, Lauwers-Cances V, Puwanarajah P, Duvernoy H (2008) Morphometry of the human cerebral cortex microcirculation: General characteristics and space-related profiles. NeuroImage 39(3):936–948, doi:10.1016/j.neuroimage.2007.09.024

Leenders K, Perani D, Lammertsma A, Heather J, Buckingham P, Jones T, Healy M, Gibbs J, Wise R, Hatazawa J, et al. (1990) Cerebral blood flow, blood volume and oxygen utilization: normal values and effect of age. Brain 113(1):27–47

Mestre H, Tithof J, Du T, Song W, Peng W, Sweeney AM, Olveda G, Thomas JH, Nedergaard M, Kelley DH (2018) Flow of cerebrospinal fluid is driven by arterial pulsations and is reduced in hypertension. Nat Commun 9(1):4878, doi:10.1038/s41467-018-07318-3, number: 1 Publisher: Nature Publishing Group

Muizelaar JP, Fatouros PP, Schröder ML (1997) A new method for quantitative regional cerebral blood volume measurements using computed tomography. Stroke 28(10):1998–2005, doi:10.1161/01.str.28.10.1998

Patlak CS, Blasberg RG, Fenstermacher JD (1983) Graphical evaluation of blood-to-brain transfer constants from multiple-time uptake data. Journal of Cerebral Blood Flow & Metabolism 3(1):1–7, doi:10.1038/jcbfm.1983.1, URL http://dx.doi.org/10.1038/jcbfm.1983.1

Poulain A, Riseth J, Vinje V (2023) Multi-compartmental model of glymphatic clearance of solutes in brain tissue. Plos one 18(3):e0280501, doi:10.1371/journal.pone.0280501

Raicevic N, Forer JM, de Guevara AL, Du T, Nedergaard M, Kelley DH, Boster K (2023) Sizes and shapes of perivascular spaces surrounding murine pial arteries. Fluids and Barriers of the CNS 20(1), doi:10.1186/s12987-023-00454-z

Ray LA, Heys JJ (2019) Fluid Flow and Mass Transport in Brain Tissue. Fluids 4(4), doi:10.3390/fluids4040196

Ray LA, Pike M, Simon M, Iliff JJ, Heys JJ (2021) Quantitative analysis of macroscopic solute transport in the murine brain. Fluids and Barriers of the CNS 18(1):55, doi:10.1186/s12987-021-00290-z

Ringstad G, Vatnehol SAS, Eide PK (2017) Glymphatic MRI in idiopathic normal pressure hydrocephalus. Brain 140(10):2691–2705, doi:10.1093/brain/awx191

Ringstad G, Valnes LM, Dale AM, Pripp AH, Vatnehol SAS, Emblem KE, Mardal KA, Eide PK (2018) Brain-wide glymphatic enhancement and clearance in humans assessed with MRI. JCI Insight 3(13), doi:10.1172/jci.insight.121537

Schain AJ, Melo-Carrillo A, Strassman AM, Burstein R (2017) Cortical spreading depression closes paravascular space and impairs glymphatic flow: Implications for migraine headache. The Journal of Neuroscience 37(11):2904–2915, doi:10.1523/jneurosci.3390-16.2017

Schmid F, Barrett MJ, Jenny P, Weber B (2019) Vascular density and distribution in neocortex. NeuroImage 197:792–805, doi:10.1016/j.neuroimage.2017.06.046

Shipley RJ, Chapman SJ (2010) Multiscale Modelling of Fluid and Drug Transport in Vascular Tumours. Bull Math Biol 72(6):1464–1491, doi:10.1007/s11538-010-9504-9

Sourbron S, Ingrisch M, Siefert A, Reiser M, Herrmann K (2009) Quantification of cerebral blood flow, cerebral blood volume, and blood–brain-barrier leakage with DCE-MRI. Magnetic Resonance in Medicine 62(1):205–217, doi:10.1002/mrm.22005

Syková E, Nicholson C (2008) Diffusion in brain extracellular space. Physiological Reviews 88(4):1277–1340, doi:10.1152/physrev.00027.2007, pMID: 18923183

Tithof J, Boster KA, Bork PA, Nedergaard M, Thomas JH, Kelley DH (2022) A network model of glymphatic flow under different experimentally-motivated parametric scenarios. iScience 25(5):104258, doi:10.1016/j.isci.2022.104258

Valnes LM, Mitusch SK, Ringstad G, Eide PK, Funke SW, Mardal KA (2020a) Apparent diffusion coefficient estimates based on 24 hours tracer movement support glymphatic transport in human cerebral cortex. Scientific Reports 10(1):1–12, doi:10.1038/s41598-020-66042-5

Valnes LM, Mitusch SK, Ringstad G, Eide PK, Funke SW, Mardal KA (2020b) Apparent diffusion coefficient estimates based on 24 hours tracer movement support glymphatic transport in human cerebral cortex. Sci Rep 10(1):9176, doi:10.1038/s41598-020-66042-5

Vardakis JC, Guo L, Chou D, Ventikos Y (2021) Using multicompartmental poroelasticity to explore brain biomechanics and cerebral diseases. In: Advances in Critical Flow Dynamics Involving Moving/Deformable Structures with Design Applications: Proceedings of the IUTAM Symposium on Critical Flow Dynamics involving Moving/Deformable Structures with Design applications, June 18-22, 2018, Santorini, Greece, Springer, pp 151–163

Vinje V, Ringstad G, Lindstrøm EK, Valnes LM, Rognes ME, Eide PK, Mardal KA (2019) Respiratory influence on cerebrospinal fluid flow–a computational study based on long-term intracranial pressure measurements. Scientific reports 9(1):9732

Vinje V, Zapf B, Ringstad G, Eide PK, Rognes ME, Mardal KA (2023) Human brain solute transport quantified by glymphatic MRI-informed biophysics during sleep and sleep deprivation. bioRxiv doi:10.1101/2023.01.01.522190

Weinmann HJ, Brasch RC, Press WR, Wesbey GE (1984) Characteristics of gadolinium-DTPA complex: a potential NMR contrast agent. AJR American journal of roentgenology 142(3):619–624, doi:10.2214/ajr.142.3.619

Xie L, Kang H, Xu Q, Chen MJ, Liao Y, Thiyagarajan M, O'Donnell J, Christensen DJ, Nicholson C, Iliff JJ, et al. (2013) Sleep Drives Metabolite Clearance from the Adult Brain. Science 342(6156):373–377, doi:10.1126/science.1241224

Chapter 9

Estimating fluid flow fields from contrast imaging

Marie E. Rognes

Abstract

Given concentration fields c_1 and c_2 and a known diffusion coefficient D, how can we estimate a velocity flow field ϕ that by diffusion and convection transports c_1 at t_1 to c_2 at t_2?

Introduction

Given a pair of images, representing concentration fields $c_1 = c_1(x)$ and $c_2 = c_2(x)$ defined over $x \in \Omega \subset \mathbb{R}^d$ at time t_1 and t_2, respectively, can we identify a vector field $\phi = \phi(x)$ that maps c_1 to c_2? This problem setting is readily encountered when quantifying the mechanisms underlying brain solute transport and clearance by computational modelling. It is also a classical problem setting in image processing, referred to as the optical flow problem, see e.g. the highly readable paper by Wildes et al. (2000). Several physics-based (as well as non-physics-based) approaches to determining optical flow exist; a key idea is to use a mass continuity equation to constrain a velocity vector field in combination with minimization of e.g. the kinetic energy (Benamou and Brenier, 2000).

In the context of brain solute transport and clearance, Tannenbaum and coauthors Ratner et al. (2015, 2017) pioneered the use of optical flow-based methods to extract glymphatic flow fields from series of two-photon fluorescence microscopy images. Their approach, as described by Mueller et al. (2013), estimates a flow field via an optimal mass transport-based (OMT) approach to optical flow. More recently, Vinje et al. (2023) quantified human brain interstitial fluid flow velocities during solute clearance from contrast-enhanced MR images using an optimal convection-diffusion (OCD) approach. This chapter presents mathematical and nu-

Simula Research Laboratory e-mail: meg@simula.no

merical formulations for both methods, and compares their accuracy when tasked with identifying a velocity field in a manufactured convection-diffusion example. The numerical implementation is based on the FEniCS and Dolfin-adjoint finite element- and adjoint-based optimization software, and is freely available .

9.1 Mathematical models for flow field estimation

Let Ω be an open, bounded domain in $\mathbb{R}^d$ ($d = 1, 2, 3$) with coordinates x and let $t \in (t_1, t_N)$. In general, our goal is to identify a flow (vector) field $\phi = \phi(x, t)$ from a discrete sequence of solute concentration or signal intensity images $c_n = c_n(x)$ at times t_n for $n = 1, 2, \ldots, N$. For compactness in presentation, we let $N = 2$ and denote $\tau = t_2 - t_1$.

9.1.1 Optical flow and optimal mass transport (OMT)

We begin by considering an *optimal mass transport* (OMT) approach to *optical flow* (Mueller et al., 2013). Assume that the transport of the concentration c by the velocity field ϕ is governed by the continuity equation:

$$f(c, \phi) \equiv c_t + \operatorname{div}(c\phi) = 0 \qquad \text{in } \Omega, \quad t > 0. \tag{9.1}$$

We define the kinetic energy associated with c and ϕ:

$$R(c, \phi) = \int_{t_1}^{t_2} \int_\Omega \phi \cdot \phi c \, \mathrm{d}x \, \mathrm{d}t,$$

which will play the role of a *regularization term* or *prior*. The OMT problem is now to find $\phi = \phi(x, t)$ that minimizes

$$\min_\phi \frac{1}{2} \int_{t_1}^{t_2} \int_\Omega f(c, \phi)^2 \, \mathrm{d}x \, \mathrm{d}t + \frac{1}{2} \alpha R(c, \phi) \tag{9.2}$$

with $c(t_1) = c_1$ and $c(t_2) = c_2$, and where $\alpha > 0$ is a parameter. Variations on this formulation easily result from different choices of the regularization functional R. For example, Mueller et al. (2013) also consider a version with L^2-regularization:

$$R(c, \phi) = \int_0^T \int_\Omega \phi \cdot \phi \, \mathrm{d}x \, \mathrm{d}t.$$

Mueller et al. (2013) solve the OMT problem numerically using a discretize-then-minimize approach with finite differences in time and space. We here consider a revised formulation using a variational formulation and finite elements in space

instead. To this end, discretize (9.2) in time using the midpoint rule, i.e. approximate the time-integral by:

$$\int_{t_1}^{t_2} f(t)\,\mathrm{d}t \approx \tau f(t^{1/2}),$$

where $t^{1/2} = \frac{1}{2}(t_1 + t_2)$ is the midpoint of the time interval. We approximate the time derivative via the natural first order approximation:

$$c_t \approx \frac{c_2 - c_1}{\tau} \equiv \delta c,$$

approximate ϕ as constant-in-time by its midpoint value: $\phi = \phi(x) \approx \phi(x, t^{1/2})$, and introduce the average concentration

$$\bar{c} \equiv \frac{1}{2}(c_1 + c_2).$$

Since c_1 and c_2 are given, $\bar{c}$ and δc are known (computed) fields. The time-discrete OMT minimization problem then reads:

$$\min_{\phi} M(\phi) \equiv \min_{\phi} M(\phi) = \frac{1}{2}\tau \left(\int_\Omega (\delta c + \operatorname{div}(\phi \bar{c}))^2 \,\mathrm{d}x + \alpha \int_\Omega (\phi \cdot \phi)\bar{c}\,\mathrm{d}x \right). \quad (9.3)$$

The optimal solution of the minimization problem (9.3) satisfies the Euler-Lagrange equations involving the Gateaux derivative of M in the direction $\psi = \psi(x)$:

$$\frac{\partial}{\partial \epsilon} M(\phi + \epsilon\psi)\Big|_{\epsilon=0} = 0.$$

Computing the Gateaux derivative, we find:

$$\begin{aligned}
&\frac{\partial}{\partial \epsilon} M(\phi + \epsilon\psi)\Big|_{\epsilon=0} \\
&= \frac{\partial}{\partial \epsilon} \left(\frac{1}{2}\tau \left(\int_\Omega (\delta c + \operatorname{div}((\phi + \epsilon\psi)\bar{c}))^2 \,\mathrm{d}x + \alpha \int_\Omega ((\phi + \epsilon\psi) \cdot (\phi + \epsilon\psi))\bar{c}\,\mathrm{d}x \right) \right) \Bigg|_{\epsilon=0} \\
&= \frac{1}{2}\tau \left(\int_\Omega 2(\delta c + \operatorname{div}(\bar{c}\phi))(\operatorname{div}(\bar{c}\psi))\,\mathrm{d}x + \alpha \int_\Omega 2\phi \cdot \psi \bar{c}\,\mathrm{d}x \right) \\
&= \tau \left(\int_\Omega (\delta c + \operatorname{div}(\bar{c}\phi))(\operatorname{div}(\bar{c}\psi))\,\mathrm{d}x + \alpha \int_\Omega \phi \cdot \psi \bar{c}\,\mathrm{d}x \right).
\end{aligned}$$

The resulting OMT variational problem reads: find $\phi \in V$ such that for all $\psi \in V$

$$a(\phi, \psi) \equiv \langle \operatorname{div}(\bar{c}\phi), \operatorname{div}(\bar{c}\psi) \rangle + \alpha \langle \bar{c}\phi, \psi \rangle = -\langle \delta c, \operatorname{div} \bar{c}\psi \rangle \equiv L(\psi), \quad (9.4)$$

where we have introduced the familiar notation for the $L^2(\Omega)$-inner product

$$\langle \phi, \psi \rangle = \int_\Omega \phi \cdot \psi \,\mathrm{d}x.$$

This is a linear variational problem that can be solved numerically using straightforward finite element techniques in e.g. FEniCS.

In order to also include transport by diffusion, we may replace (9.1) by a convection-diffusion equation (more details follow below). The same steps as for the OMT formulation above again yields a linear variational problem for ϕ, but now involving also second order derivatives of $\bar{c}$. Evaluating such terms for low-regularity data e.g. images with noise pose additional challenges.

9.1.2 An optimal diffusion-convection (OCD) formulation

In the spirit of Glowinski et al. (2022) and Andreev et al. (2015), an alternative approach is to define a bilinear optimal control problem where both the final concentration and the convective velocity are treated as variables. This formulation lends itself easily to also accounting for diffusion and/or reaction terms.

So, consider the convection-diffusion equation:

$$F(c, \phi) \equiv c_t + \operatorname{div}(\phi c) - \operatorname{div} D\nabla c = 0 \qquad \text{in } \Omega, \quad t > 0, \tag{9.5}$$

and assume that observations of the concentration c_1, c_2 at t_1, t_2 are given. We are now interested in c and ϕ that both satisfy (9.5) and that best match observations – in the sense that c and ϕ minimize the data-misfit objective functional J:

$$J(c, \phi) = \frac{1}{2}\|c(t_2) - c_2\|^2_{L^2(\Omega)} + R(c, \phi), \tag{9.6}$$

recalling that c_2 is the given data field and $c(t_2)$ is the solution to (9.5) at $t = t_2$, and where R again represents a regularization term or prior. Specifically, the minimization problem of interest reads as:

$$\min_{c,\phi} J(c, \phi) \quad \text{such that} \quad F(c, \phi) = 0, \quad c(t_1) = c_1.$$

where we have also imposed the observations at t_1 as an initial condition for c. Observe that F is nonlinear when viewed as a function of both c and ϕ. In the optimization literature, the pattern with a multiplicative control variable (here ϕ) is referred to as a bilinear control problem. Using a Lagrange multiplier technique (Hinze et al., 2008), now introduce the Lagrangian L defined by

$$L(c, \phi, y) = J(c, \phi) - \langle F(c, \phi), y\rangle_{Y \times Y^*}.$$

for $c \in C$, $\phi \in Q$ and the adjoint variable $y \in Y$. For our example, the Lagrangian thus becomes

$$L(c, \phi, y) = \frac{1}{2}\|c(t_2) - c_2\|^2_{L^2(\Omega)} + R(c, \phi) + \langle c_t + \operatorname{div}(\phi c) - \operatorname{div} D\nabla c, y\rangle_{Y \times Y^*}. \tag{9.7}$$

Discretizing the Lagrangian (9.7) in time with an implicit Euler scheme and using the same time-resolution as that of the given data, it remains to compute single (in time) approximations $c(x) = c(x, t_2)$, $\phi(x) = \phi(x, t)$ and $y(x) = y(x, t)$. In addition, we identify the $Y^* \times Y$ duality pairing with the $L^2(\Omega)$-inner product denoted $\langle \cdot, \cdot \rangle$, to obtain the time-discrete Lagrangian:

$$L_\tau(c, \phi, y) = \frac{1}{2}\|c - c_2\|^2_{L^2(\Omega)} + R(c, \phi) \\ + \langle \tau^{-1}(c - c_1) + \operatorname{div}(\phi c) - \operatorname{div}(D\nabla c), y\rangle_{L^2(\Omega)}.$$

Now, we again take the Gateaux derivatives of L_τ with respect to c, ϕ and y to derive the optimality conditions. First, taking the derivatives with respect to $y \in Y$ gives the state equation:

$$\frac{\partial}{\partial y} L_\tau(c, \phi, y)[dy] = \langle \tau^{-1} c + \operatorname{div}(\phi c) - \operatorname{div}(D\nabla c) - \tau^{-1} c_1), dy\rangle.$$

Next, taking the derivative with respect to $c \in C \subseteq H^1(\Omega)$ in the direction of $dc \in C$ gives the adjoint equation:

$$\frac{\partial}{\partial c} L_\tau(c, \phi, y)[dc] = \langle c - c_2, dc\rangle_{L^2(\Omega)} + R_c(c, \phi)[dc] \\ + \langle \tau^{-1} dc + \operatorname{div}(\phi dc) - \operatorname{div}(D\nabla dc), y\rangle.$$

Taking the derivative with respect to $\phi \in Q \subseteq H(\operatorname{div}, \Omega)$ in the direction of $d\phi \in Q$ gives

$$\frac{\partial}{\partial \phi} L_\tau(c, \phi, y)[d\phi] = R_\phi(c, \phi)[d\phi] + \langle \operatorname{div}(c d\phi), y\rangle.$$

Finally, we need to select a regularization term R. Andreev et al. (2015) consider an $H(\operatorname{div})$-regularization with is a very natural Sobolev space setting for the velocity field. To enforce a bit more regularity, we here use H^1-regularization:

$$R(c, \phi) = R(\phi) = \frac{1}{2}\alpha \left(\|\phi\|^2 + \|\nabla \phi\|^2 \right),$$

for a regularization parameter $\alpha \geq 0$. Thus $R_c = 0$, while

$$R_\phi(c, \phi)[d\phi] = \alpha \left(\langle \phi, d\phi \rangle + \langle \nabla \phi, \nabla d\phi \rangle \right).$$

With these assumptions and some renaming of the test functions, we obtain the following time-independent but non-linear system of governing equations: find $c \in C$, $\phi \in Q$, $y \in Y$ such that

$$\langle c, d\rangle + \langle \tau^{-1} d + \operatorname{div}(\phi\, d) - \operatorname{div} D\nabla d, y\rangle = \langle c_2, d\rangle,$$
$$\langle \operatorname{div}(c\,\psi), y\rangle + \alpha\,(\langle \phi, \psi\rangle + \langle \nabla\,\phi, \nabla\,\psi\rangle) = 0,$$
$$\langle \tau^{-1} c + \operatorname{div}(\phi\, c) - \operatorname{div}(D\nabla c), z\rangle = \langle \tau^{-1} c_1, z\rangle,$$

for all $d \in C$, $z \in Y$, and $\psi \in Q$. To reduce regularity requirements, we integrate second-order derivatives by parts. To address the resulting boundary terms, we prescribe Dirichlet boundary conditions for the concentration c and corresponding homogeneous Dirichlet conditions for the adjoint field y:

$$c = c_2 \quad y = 0 \qquad \text{on } \partial\Omega.$$

In summary for the OCD method, we solve the following nonlinear variational formulation: for given image data $c_1, c_2 \in L^2(\Omega)$, time step $\tau > 0$, diffusion tensor $D \in L^2(\Omega)$, and regularization parameter $\alpha \geq 0$, find $c \in H^1_{c_1}(\Omega)$, $y \in H^1_0(\Omega)$, $\phi \in H^1(\Omega; R^d)$ such that

$$\langle c, d\rangle + \langle \tau^{-1} d + \operatorname{div}(\phi\, d), y\rangle + \langle D\nabla d, \nabla y\rangle = \langle c_2, d\rangle, \tag{9.9a}$$
$$\langle \operatorname{div}(c\,\psi), y\rangle + \alpha\,(\langle \phi, \psi\rangle + \langle \operatorname{div}\phi, \operatorname{div}\psi\rangle) = 0, \tag{9.9b}$$
$$\langle \tau^{-1} c + \operatorname{div}(\phi\, c), z\rangle + \langle D\nabla c, \nabla z\rangle = \langle \tau^{-1} c_1, z\rangle, \tag{9.9c}$$

for all $d \in H^1(\Omega)$, $z \in H^1_0(\Omega)$, $\psi \in H^1(\Omega; \mathbb{R}^d)$. We discretize this variational formulation by a finite element method with continuous piecewise linear elements for all fields, and solve the resulting system of nonlinear algebraic equations using Newton's method. The image data c_1 for c and $\phi = 0$ provides a natural initial guess.

9.1.3 Reduced optimal convection-diffusion formulation (rOCD)

Instead of solving the OCD optimization problem (9.9) via the "all-at-once" Lagrange multiplier approach (9.9), we may consider an iterative "reduced" approach (Hinze et al., 2008; Farrell et al., 2013). To this end, we introduce the reduced objective functional $\tilde{J}$ defined as J cf. (9.6) viewed in terms of the control parameter ϕ alone and target finding the ϕ that minimize $\tilde{J}$:

$$\min_{\phi} \tilde{J}(\phi) = J(c(\phi), \phi). \tag{9.10}$$

The convection-diffusion equation (9.5) defines the map $\phi \mapsto c(\phi)$.

To solve the reduced OCD (rOCD) problem numerically, we discretize the convection diffusion equation in time from t_1 to t_2, again using the interval between images as the time step $\tau = t_2 - t_1$ for simplicity. With an implicit Euler scheme, the equation for $c \approx c(t_2)$ becomes

$$c - \tau\,(\mathrm{div}(\phi c) - \mathrm{div}\, D\,\nabla c)) = c_1 \tag{9.11}$$

with the initial condition $c(t_1) = c_1$. To discretize (9.11) in space, we may use a standard finite element variational formulation and impose the Dirichlet boundary condition $c = c_2$ on $\partial\Omega$ cf. (9.1.2).

We can then solve the optimization problem (9.10) using standard (unconstrained) optimization algorithms such as Newton or quasi-Newton methods Al-Baali et al. (2014) such as the popular L-BFGS algorithm Liu and Nocedal (1989). In practice, this is easily accomplished via the Dolfin-adjoint library with FEniCS or Firedrake as the finite element backends (Farrell et al., 2013).

9.2 Numerical comparison of OMT, OCD and rOCD methods

To study the accuracy and predictability of the different algorithms (OMT, OCD and rOCD), we here consider a numerical test case with a manufactured solution (Fig. 9.1, Table 9.1). Interestingly, we observe that the OMT velocity (Fig. 9.1D) differs substantially from the original convective contribution, both qualitatively and quantitatively, and that its norm(s) vary strongly with the choice of the parameter α. On the other hand, the OCD and rOCD velocity fields are very similar, qualitatively agrees well with the original velocity, and are robust with respect to the regularization parameters for small and moderately small α, but both underestimate the velocity magnitude and show bias towards the regions with higher concentration values. A full-scale application of the OCD and rOCD methods to identify convective fluid flow fields from human glymphatic MRI images is presented by Vinje et al. (2023).

9.3 Concluding remarks and outlook

In conclusion, we highlight the following observations.

- The OMT approach is easy to implement and relatively inexpensive, but is sensitive to numerical parameters such as the choice of regularization parameters and mesh resolution and may yield spurious velocity field patterns.
- The current OMT formulation does not account for the diffusion directly. While the formulation could be extended the explicitly include a diffusion term, in practice, this would involve even more discrete derivatives of the (typically non-smooth) data and unpredictable results.
- The OCD and rOCD approaches are both also easily implemented, more computationally expensive, but yield more robust results with regard to the numerical parameters.

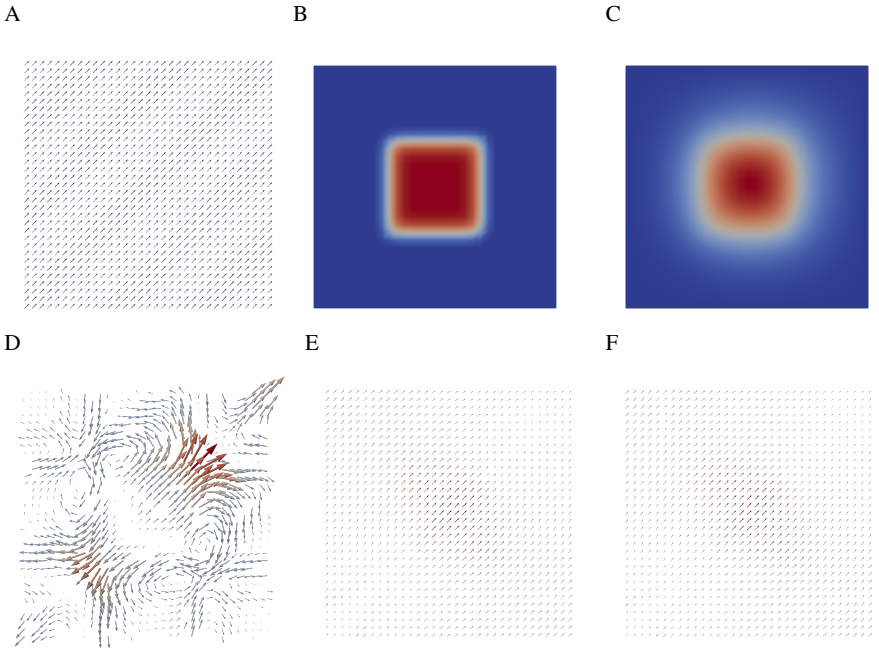


Fig. 9.1: Velocity field estimation with known data. Consider a (unitless) manufactured test case with a given velocity field $\phi = (0.02, 0.02)$ (A) and diffusion coefficient $D = 0.01$ over the domain $[0,1]^2 \subset \mathbb{R}^2$. An smooth bump function (B) is used as initial data c_1 at $t_1 = 0$ and the continuous piecewise linear finite element/implicit Euler approximation to (9.5) with zero boundary condition at $t_2 = 1$ (C) are used as synthetically generated data. D–F: The velocity fields recovered by the OMT (D), OCD (E) and rOCD (F) approaches ($\alpha = 10^{-3}$, $h = \frac{1}{32}$, fields scaled 2×).

- Solvers for the nonlinear OCD problem (e.g. Newton's method) may easily fail to converge, and similarly very many optimization iterations may be required for the rOCD approach.
- None of the approaches considered here enforces a volume-preservation constraint

$$\operatorname{div} \phi = 0 \quad \text{in } \Omega, \quad t > 0, \tag{9.12}$$

  in contrast to e.g. the study by Glowinski et al. (2022).

Acknowledgements The author graciously acknowledges valuable discussions and input from Drs. Vegard Vinje and Bastian Zapf.

A: OMT vs α

α	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
10^0	0.0129	0.0302	0.1686
10^{-1}	0.0171	0.0542	0.2459
10^{-2}	0.0236	0.0786	0.3332
10^{-3}	0.0282	0.0845	0.4181
10^{-4}	0.0448	0.0851	0.6477
10^{-5}	0.0709	0.1441	0.6857
Exact	0.0283	0.0283	0.0000

B: OCD vs α

α	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
10^0	0.0049	0.0070	0.0068
10^{-1}	0.0088	0.0124	0.0119
10^{-2}	0.0098	0.0135	0.0131
10^{-3}	0.0100	0.0137	0.0133
10^{-4}	0.0100	0.0137	0.0133
10^{-5}	0.0100	0.0137	0.0133
Exact	0.0283	0.0283	0.0000

C: rOCD vs α

α	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
10^0	0.0049	0.0070	0.0068
10^{-1}	0.0088	0.0124	0.0119
10^{-2}	0.0099	0.0136	0.0131
10^{-3}	0.0103	0.0139	0.0131
10^{-4}†	0.0116	0.0148	0.0122
10^{-5}†	0.0029	0.0069	0.0208
Exact	0.0283	0.0283	0.0000

D: OMT vs n

n	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
4	0.0129	0.0402	0.1574
8	0.0313	0.0714	0.4666
16	0.0282	0.0845	0.4181
32	0.0236	0.0763	0.3756
64	0.0217	0.0714	0.3568
128	0.0191	0.0617	0.3209
Exact	0.0283	0.0283	0.0000

E: OCD vs n

n	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
4	0.0027	0.0046	0.0078
8	0.0095	0.0132	0.0131
16	0.0100	0.0137	0.0133
32	0.0103	0.0140	0.0134
64	0.0104	0.0141	0.0134
128	0.0104	0.0141	0.0134
Exact	0.0283	0.0283	0.0000

F: rOCD vs n

n	$\|\phi\|_{L^2}$	$\|\phi\|_{L^\infty}$	$\|\phi\|_{H_0^1}$
4	0.0026	0.0044	0.0079
8	0.0093	0.0130	0.0132
16	0.0103	0.0139	0.0131
32	0.0101	0.0139	0.0136
64	0.0099	0.0137	0.0138
128	0.0100	0.0138	0.0138
Exact	0.0283	0.0283	0.0000

Table 9.1: For all methods, for varying α, $n = 16$, while for varying n, $\alpha = 10^{-3}$. †: reached maximal number of iterations (500). For the rOCD method, we used the L-BFGS algorithm with gtol set to 10^{-12}, and a maximal number of iterations of 500.

References

Al-Baali M, Spedicato E, Maggioni F (2014) Broyden's quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. Optimization Methods and Software 29(5):937–954, doi:10.1080/10556788.2013.856909

Andreev R, Scherzer O, Zulehner W (2015) Simultaneous optical flow and source estimation: Space-time discretization and preconditioning. Applied Numerical Mathematics 96:72–81, doi:10.1016/j.apnum.2015.04.007

Benamou JD, Brenier Y (2000) A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik 84(3):375–393, doi:10.1007/s002110050002

Farrell PE, Ham DA, Funke SW, Rognes ME (2013) Automated derivation of the adjoint of high-level transient finite element programs. SIAM Journal on Scientific Computing 35(4):C369–C393, doi:10.1137/120873558

Glowinski R, Song Y, Yuan X, Yue H (2022) Bilinear optimal control of an advection-reaction-diffusion system. SIAM Review 64(2):392–421, doi:10.1137/21M1389778

Hinze M, Pinnau R, Ulbrich M, Ulbrich S (2008) Optimization with PDE constraints, vol 23. Springer Science & Business Media, doi:10.1007/978-1-4020-8839-1

Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Mathematical Programming 45(1):503–528, doi:10.1007/BF01589116

Mueller M, Karasev P, Kolesov I, Tannenbaum A (2013) Optical Flow Estimation for Flame Detection in Videos. IEEE Transactions on Image Processing 22(7):2786–2797, doi:10.1109/TIP.2013.2258353

Ratner V, Zhu L, Kolesov I, Nedergaard M, Benveniste H, Tannenbaum A (2015) Optimal-mass-transfer-based estimation of glymphatic transport in living brain. In: Ourselin S, Styner MA (eds) Medical Imaging 2015: Image Processing, International Society for Optics and Photonics, SPIE, vol 9413, p 94131J, doi:10.1117/12.2076289

Ratner V, Gao Y, Lee H, Elkin R, Nedergaard M, Benveniste H, Tannenbaum A (2017) Cerebrospinal and interstitial fluid transport via the glymphatic pathway modeled by optimal mass transport. NeuroImage 152:530–537, doi:10.1016/j.neuroimage.2017.03.021

Vinje V, Zapf B, Ringstad G, Eide PK, Rognes ME, Mardal KA (2023) Human brain solute transport quantified by glymphatic MRI-informed biophysics during sleep and sleep deprivation. bioRxiv doi:10.1101/2023.01.01.522190

Wildes RP, Amabile MJ, Lanzillotto AM, Leu TS (2000) Recovering Estimates of Fluid Flow from Image Sequence Data. Computer Vision and Image Understanding 80(2):246–266, doi:10.1006/cviu.2000.0874

Chapter 10

An Introduction to Network Models of Neurodegenerative Diseases

Georgia S. Brennan and Alain Goriely
University of Oxford, Oxford UK .

Abstract Neurodegenerative Diseases are characterized by the build-up of key toxic protein. For instance, Alzheimer's disease, impacting over 50 million people worldwide, is associated with amyloid beta and tau proteins. The spatial and temporal evolution of these proteins, and their association with cognitive decline, suggest the existence of underlying principles that might be modeled mathematically. Fundamentally, the spread of these diseases is characterized by transport, expansion, and saturation of toxic proteins in the brain. We show that the simplest model that includes all these aspects and reproduce the basic invasion patterns is a network-based model. This reductionist model effectively approximates the full dynamics. This chapter is conceived to be a gentle step-by-step introduction of these ideas and these models.

10.1 Introduction

Alzheimer's disease affects more than 50 million people worldwide and is found in about 1 in 9 people age 65 and older. Unlike cancer and many viral and bacterial diseases, Alzheimer's disease, in its most common occurrence, presents itself in stages that have been codified and are found in most patients. We also know that this cognitive staging is associated with a systematic invasion in the brain of two key toxic proteins that were already identified by Alois Alzheimer in 1905. The first one is the well-known amyloid beta, an extracellular protein whose natural functions are not well understood. The second one is the group of tau proteins that normally stabilize microtubules in axons. However, they can misfold. It is believed that a misfolded version of tau acts as a template for healthy tau proteins and promotes the formation of oligomers of increasing sizes, eventually leading to large aggregates that can be observed in brain tissues after death Jucker and Walker (2013). While most drug trials focus on amyloid beta, it is the presence of toxic tau proteins that

is most correlated to brain atrophy and to cognitive symptoms. The typical pattern of tau evolution obtained through histological staining is known as the Braak & Braak staging after the neuroanatomists Eva and Heiko Braak who first proposed it in 1991 Braak and Braak (1991). It describes the evolution of the disease in six stages starting in the entorhinal cortex and evolving through the hippocampal region (associated with memory), the temporal lobe, the occipital lobe, and all regions of the neocortex with lesions in each region worsening in time.

The prion-like hypothesis broadly postulates that neurodegenerative diseases result from an accumulation of abnormally misfolded proteins, which aggregate and contribute to tissue death, causing associated neurodegenerative pathology and cognitive decline. In this process, certain disease-specific misfolded proteins can influence healthy proteins, forming extensive chains that can be transported through the brain along axonal pathways. Given that aggregates of differing sizes exhibit unique transport characteristics and varying levels of toxicity, it's crucial to independently monitor their spatial and temporal evolution.

From a modeling point of view, the systematic pattern of invasion through the brain and the multiple cognitive effects suggest to the curious mind that there may be some simple underlying features of the brain responsible for its spatio-temporal pattern. The challenge is therefore to obtain minimal mathematical models based on clear principles that can capture, at the brain level, the staging as well as other characteristics of the disease such as brain atrophy and changes in the overall brain dynamics.

From a phenomenological point of view, there are three processes to consider: *(transport)* toxic proteins are transported in the brain mostly along axonal bundles (connecting different parts of the brain and acting as information highways); *(expansion)* the aggregation process is autocatalytic and lead to an initial exponential increase of small toxic populations; *(saturation)* each region can only support a certain level of toxic proteins. A canonical model for such a process in a continuum medium is the celebrated Fisher–Kolmogorov–Petrovsky–Piskunov equation (Fisher-KPP). If $p(x,t) \in [0,1]$ is the scaled concentration of the toxic protein, it reads:

$$\frac{\partial p}{\partial t} = \nabla \cdot (\mathbf{D}\nabla p) + \alpha\, p\,(1-p), \tag{10.1}$$

where $\mathbf{D}$ is a transversely anisotropic diffusion tensor with a strong preferential direction along the axonal bundle and $\alpha > 0$ is the growth rate. Remarkably, this model can be obtained, under generic conditions, as a normal form of the full aggregation-fragmentation equations that tracks the evolution of oligomers of different sizes Thompson et al. (2021). Hence the parameters can be directly related to the aggregation and clearance rates at the microscopic level.

Modern brain imaging techniques are now well developed and it is relatively routine to use medical resonance imaging to obtain both the full brain geometry as well as the direction of axonal bundles. Then, starting with an initial seed of toxic proteins in the entorhinal cortex, we simulated the evolution of the field in the brain as shown in Fig. 10.1. The results were striking. Without any other ingredients, the dynamics obtained from this minimal model recovered the basic dynamics of the disease in great detail Weickenmeier et al. (2018). Further, the same model applied to other neurodegenerative diseases associated with other toxic proteins such as α-synuclein for Parkinson's or TDP-43 for amyotrophic lateral sclerosis also reproduces their basic spatio-temporal patterns as well as the atrophy patterns that can be obtained, post-processing, from finite-element simulations; the only difference being the region and extent of seeding. Despite the complexity and diversity of these diseases, universal features of progression emerge from the combination of an autocatalytic process and transport in an anisotropic medium.

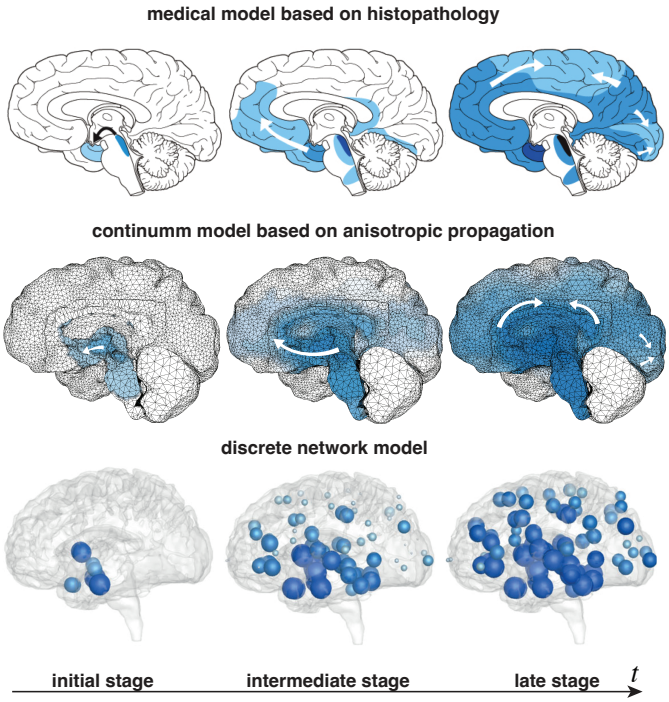


Fig. 10.1: Progression of toxic tau proteins in the brain. Top: medical staging based on histopathology (the analysis of post-mortem brain slices). Middle: Simulation of the anisotropic Fisher-KPP model with initial value in the entorhinal cortex. Bottom: Simulation of the network Fisher-KPP model with initial seeding at the entorhinal node (adapted from Fornari et al. (2019)).

When such strong universal patterns appear, a natural question emerges: what are the essential features responsible for the invasion pattern? Since the autocatalytic dynamics cannot be further simplified, we can look into the transport term and test the hypothesis that the observed patterns are a consequence of the strong transport anisotropy along the axonal bundles. These bundles are not only information highways, they could also carry efficiently toxic proteins across the brain. A simple way to test this idea is to coarse grain the brain by considering multiple regions and the connection between them. The full brain is then replaced by a network, *the connectome*, where each node is a region and an edge represents the possible connections between regions. This approach is the basis for a large field of neuroscience and various groups have successfully used the connectome to look at the effect of protein diffusion in the brain Raj et al. (2012).

The natural discretization of the diffusion operator on a network is the weighted graph Laplacian $\mathbf{L}$ with weights proportional to the number of connections and inversely proportional to the distance squared between nodes. Then, defining $p_i(t)$ to be the concentration of toxic proteins at node i, the discrete Fisher-KPP equation reads:

$$\dot{p}_i = -\rho \sum_{j=1}^{N} L_{ij} p_j + \alpha p_i (1 - p_i), \quad i = 1, \ldots, N. \tag{10.2}$$

This system of nonlinear of ordinary differential equations turns out to be an excellent approximation of the dynamics generated by the full nonlinear partial differential equations that we started with as shown in Fig. 10.1.

The rest of the chapter is an in-depth technical study of this system. All files and the Mathematica workbook can be found in the github folder https://github.com/goriely/Network83.

10.2 Network models

10.2.1 Diffusion models

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a network with node set $\mathcal{N}$ and edge set $\mathcal{E}$. Since we are interested in diffusion processes, we assume that the network is connected (otherwise, we can consider diffusion on each component of the network). Consider two nodes in $\mathcal{N}$ with labels i and j and denote by a_{ij} the weight of the edge connecting node j to node i. We define the weighted adjacency matrix $\mathbf{A}$ to be the $N \times N$ matrix with entries

$$A_{ij} = a_{ij}, \quad i, j = 1, \ldots, N. \tag{10.3}$$

We are interested in modeling diffusion-like transport processes on the network where each node has a spatial location $\mathbf{x}_i$ denoting the central point of a region of typical volume V_i. In the first instance, we consider the case where each region has the same constant volume $V = V_i, \ \forall i$ and will consider the correction due to the volume later on. We define at each node a concentration p_i, representing the amount of a certain quantity defined in that region.

The natural coarse-graining of the diffusion operator on a regular lattice is the *graph Laplacian*. Therefore, our goal is to define a graph Laplacian $\mathbf{L}$ that models basic transport properties on the network so that in the absence of other physical processes, the evolution of the concentration is given by

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\rho \mathbf{L} \cdot \mathbf{p}, \tag{10.4}$$

where $\mathbf{p} = (p_1, \ldots, p_N)$ is the (column) vector of concentrations and $\rho > 0$ is a diffusion-like constant (note that the minus sign in front of $\mathbf{L}$ is just a matter of convention and has no particular significance) and the dot represents the regular index contraction: $(\mathbf{L} \cdot \mathbf{p})_i = \sum_{j=1}^{N} L_{ij} p_j$.

There are two natural constraints associated with a transport process that depends on the relative concentration between two different regions. First, we assume that the concentration represents a physical quantity and that the transport process does not change the total amount of that quantity, in our case the total mass of a given chemical. Therefore, mass conservation (MC) imposes that the total mass in the system

$$M = \sum_{i=1}^{N} V_i p_i, \tag{10.5}$$

is constant in time. Therefore, using the assumption that all volumes are equal, we have

$$\sum_{i=1}^{N} V_i \frac{\mathrm{d}p_i}{\mathrm{d}t} = V \mathbf{1} \cdot \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = 0, \tag{10.6}$$

where $\mathbf{1} = (1, \ldots, 1)$ is the one vector. Using (10.4), this condition implies

$$\mathbf{1} \cdot \mathbf{L} \cdot \mathbf{p} = 0, \tag{10.7}$$

which must be true for all $\mathbf{p}$. Hence, we have:

$$\text{(MC)} \quad \mathbf{1} \cdot \mathbf{L} = \mathbf{0}, \tag{10.8}$$

where $\mathbf{0} = (0, \ldots, 0)$ is the null vector. In other words, the one vector is in the left nullspace of $\mathbf{L}$.

The existence of at least one zero eigenvalue for $\mathbf{L}$ implies that the right nullspace is non-empty. Hence, there exists a vector $\widehat{\mathbf{p}}$ such that

$$\mathbf{L} \cdot \widehat{\mathbf{p}} = \mathbf{0}, \tag{10.9}$$

which defines a steady-state concentration.

10.2.2 Fick's condition for undirected networks

Next, we assume that the network is undirected and that transport is driven by differences in concentration between different regions. In this case, we have $\mathbf{L} = \mathbf{L}^T$ which implies, for connected networks, that the steady concentration is uniform $\widehat{\mathbf{p}} = M/N\mathbf{1}$. This condition is related to the Fick's condition (FC) that in the absence of a concentration gradient, there is no diffusion flux. The discrete form of this condition is that if all concentrations are equal, there is no change in concentration. Hence, for undirected networks, we have a second condition

$$\text{(FC)} \quad \mathbf{L} \cdot \mathbf{1} = \mathbf{0}. \tag{10.10}$$

We note that this condition is not a stringent as the condition on mass conservation. Indeed, other assumptions are possible, for instance when the transport process is viewed as a random walk. In Masuda et al. (2017), the authors study *continuous time random walks* where the walker waits until the next move for a time τ , where τ is a random variable chosen to follow a Poisson process (τ is distributed according to the exponential distribution with parameter $\lambda = 1$). We can imagine two cases.

In the *edge-centric* case, each edge is opened independently with a given Poisson process with rate proportional to the edge weight. Once that edge is open, a walker can move through it. In this case, the equation for the evolution of the probabilities is

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\mathbf{L} \cdot \mathbf{p}, \tag{10.11}$$

where $\mathbf{D} = \mathrm{diag}(\mathbf{d})$ is the *degree matrix*, a diagonal matrix with diagonal elements $\mathbf{d} = \mathbf{A} \cdot \mathbf{1}$ and

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{10.12}$$

$\mathbf{L}$ is the *standard graph Laplacian*. In particular, in the unweighted case, the weight is taken to be $a_{ij} = 1$ whenever an edge is present between i and j as shown in Fig. 10.2. In the particular Boolean case, a walker travels at unit rate and goes from a node to its neighbor with rate one. That is a traveler leaves a node i with a rate d_i, whereas in the previous case where the transition rate for leaving a node is the same for all nodes.

In the *node-centric* cases, a walker moves from node i to node j with a probability proportional to a_{ij}, then the evolution of the probability density $\mathbf{p}$ (i.e. p_i gives the probability that a walker visits node i) is

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\mathbf{L}\mathbf{D}^{-1}\mathbf{p}, \tag{10.13}$$

In the particular Boolean case, a walker travels at unit rate and goes from a node to its neighbor with a rate proportional to its degree (a node with 3 numbers has a probability 1/3 to jump to any one of them).

10.2.3 The graph Laplacians of connected and undirected networks

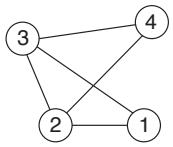
For a weighted, connected and undirected network, the standard network diffusion equation is obtained by choosing the standard graph Laplacian

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\rho \mathbf{L} \cdot \mathbf{p}. \tag{10.14}$$

The definition of $\mathbf{D}$ and the symmetry of $\mathbf{A}$ imply

$$\mathbf{D} \cdot \mathbf{1} = \mathbf{1} \cdot \mathbf{D} = \mathbf{A} \cdot \mathbf{1} = \mathbf{1} \cdot \mathbf{A} = \mathbf{d}. \tag{10.15}$$

Other possible Laplacian matrices can be defined through $\mathbf{A}$ and $\mathbf{D}$ and have been



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

Fig. 10.2: A simple example of an undirected unweighted graph with its adjacency and Laplacian matrix.

used in various studies such as spectral clustering. These include

$$\mathbf{L}_{1,0} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \mathbf{D}^{-1}\mathbf{L}, \tag{10.16}$$
$$\mathbf{L}_{0,1} = \mathbf{I} - \mathbf{A}\mathbf{D}^{-1} = \mathbf{L}\mathbf{D}^{-1}, \tag{10.17}$$
$$\mathbf{L}_{1/2,1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{10.18}$$

where $\mathbf{I}$ is the identity matrix. These choices motivate the introduction of the family

$$\mathbf{L}_{a,b} = \mathbf{D}^{1-a-b} - \mathbf{D}^{-a}\mathbf{A}\mathbf{D}^{-b} = \mathbf{D}^{-a}\mathbf{L}\mathbf{D}^{-b}. \tag{10.19}$$

The dynamics generated by these Laplacians only depend on the sum $a + b$. Indeed, consider the system (10.4) and introduce the change of variable $\mathbf{q} = \mathbf{D}^c\mathbf{p}$, then

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = -\rho \mathbf{D}^c \mathbf{L}_{a,b} \mathbf{D}^{-c} \cdot \mathbf{q}, = -\rho \mathbf{L}_{a-c,b+c} \cdot \mathbf{q}. \tag{10.20}$$

Hence, up to a scaling of the variables, the three systems (10.16-10.18) have the same dynamics (same eigenvalues, scaled eigenvectors). However, the Laplacian $\mathbf{L}_{0,0}$ is not part of that family and no straightforward connection between its spectrum and the spectrum of other laplacians can be made.

We can now check, within the family $\mathbf{L}_{a,b}$, which Laplacian satisfies Fick's condition:

$$\mathbf{L}_{a,b} \cdot \mathbf{1} = \mathbf{D}^{1-a-b} \cdot \mathbf{1} - \mathbf{D}^{-a}\mathbf{A}\mathbf{D}^{-b} \cdot \mathbf{1}, \tag{10.21}$$

which implies after multiplication by $\mathbf{D}^a$:

$$\mathbf{D}^{1-b} \cdot \mathbf{1} = \mathbf{A}\mathbf{D}^{-b} \cdot \mathbf{1}. \tag{10.22}$$

This last equality can only be true if $\mathbf{A}\mathbf{D}^{-b} = \mathbf{D}^{-b}\mathbf{A}$. Indeed, assuming $b \neq 0$, if there exists i and j such that $d_i \neq d_j$ (otherwise $\mathbf{D}$ is a multiple of the identity and all Laplacians are equal up to a multiple), we have $(\mathbf{A}\mathbf{D}^{-b})_{ij} = d_j^{-b}\mathbf{A}_{ij} \neq (\mathbf{D}^{-b}\mathbf{A})_{ij} = d_i\mathbf{A}_{ij}$. Hence we conclude that unless $d_i = d_j$ for all i, j, we must have $b = 0$.

The same reasoning applied to the mass conservation condition implies that unless $d_i = d_j$ for all i, j, we must have $a = 0$. Indeed, if $a \neq 0$, then the total mass is not conserved in the dynamics. Indeed, the stationary state, in general is given by a multiple of $\widehat{\mathbf{p}} = \mathbf{D}^b \cdot \mathbf{1}$. An initial concentration $\mathbf{p_0}$ will evolve asymptotically (as $t \to \infty$) towards $\mathbf{p}_\infty = \lambda(\mathbf{p}_0)\widehat{\mathbf{p}}$ where $\lambda(\mathbf{p}_0)$ depends linearly on the initial conditions (e.g. for a symmetric $\mathbf{L}_{a,b}$, we have simply $\lambda = \mathbf{p}_0 \cdot \widehat{\mathbf{p}}/\widehat{\mathbf{p}} \cdot \widehat{\mathbf{p}}$). Therefore, an initial mass $m_0 = \mathbf{1} \cdot \mathbf{p}_0$ evolves asymptotically to a mass

$$m_\infty = \lambda\, \mathbf{1} \cdot \widehat{\mathbf{p}}. \tag{10.23}$$

Depending on the initial conditions, the degrees, and the choice of $a \neq 0$, m_∞ can be larger, smaller, or equal to m_0. From a modeling perspective, it should be clear that such a choice cannot be justified. A change in mass should be properly modeled and cannot depend on the topology of the graph or the initial conditions. Whereas a normalized graph Laplacian (with $a \neq 0$) is a convenient object for many mathematical studies, it cannot be used to model a transport process (and is problematic to model a transport process that includes either creation or removal of material as such process would depend on the topology of the graph and not a physical process).

Hence, for a generic graph we conclude that *the only Laplacian $\mathbf{L}_{a,b}$ that satisfies both the mass conservation and the Fick's conditions is the standard graph Laplacian* $\mathbf{L} = \mathbf{L}_{0,0}$.

10.2.3.1 Correction for varying volumes

If the nodes have different volumes $\mathbf{v} = (V_1, \ldots, V_N)$, then the condition for the conservation of mass (MC) has to be modified. Indeed the total mass is now $M = \mathbf{v}\cdot\mathbf{p}$. Enforcing $\dot{M} = 0$ in (10.4), implies

$$\mathbf{v} \cdot \mathbf{L} = \mathbf{0}. \tag{10.24}$$

This condition is satisfied by choosing

$$\mathbf{L} = \mathbf{L}_V = \mathbf{V}^{-1}\mathbf{L}, \qquad \mathbf{V} = 1/V_0 \mathrm{diag}(\mathbf{v}), \tag{10.25}$$

where V_0 is a reference volume (average or total volume for instance). We note that this modified graph Laplacian is not symmetric, but that Fick's condition is still satisfied ($\mathbf{L}_V \cdot \mathbf{1} = \mathbf{0}$) since diffusion takes place when a concentration gradient is established, independently of the nodes volume.

In the context of modelling neurodegenerative diseases, we note that the multiplication of the standard Laplacian on the left by a diagonal matrix has also been used to define regions of vulnerability Henderson et al. (2020, 2019), hence assuming that diffusion takes place differently in different nodes. Mathematically, it is the same operation but its interpretation in term of volumes or vulnerability is different and corresponds to a different modeling choice (e.g. if we insist on mass conservation then mass should be conserved and interpreting $\mathbf{V}^{-1}$ as vulnerability precludes mass conservation).

10.3 The choice of weights

We now turn our attention to the problem of choosing appropriately the weights of the network. We assume that these weights take positive real values and model he

fact that some connections may favor transport over others. Indeed, in a physical network one may expect that some of the edges may be larger than others or having different properties that will enhance or inhibits diffusion. There is no particular added mathematical difficulty to treat this case and all the definitions of Laplacians seen so far still apply by using a real-valued *weighted adjacency matrix* $\mathbf{A}$ and adding the word 'weighted' in front of all other matrices. For instance, the *weighted degree matrix* is the diagonal matrix with the diagonal elements given by the *weighted degrees* $\mathbf{d} = \mathbf{A}.\mathbf{1}$. We note however, that these degrees loose their interpretation as the number of edges attached to a node.

The problem is now how to chose the weights of the adjacency matrix. Based on the available data for the brain connectome, we assume that an edge between distinct nodes i and j is represented by an idealized uniform cylinder built from $n_{ij} \in \mathbb{R}$ fibers and node-distance $g_{ij} \in \mathbb{R}_0$. The number of fibers is real in general as it is obtained as averaged over multiple data sets. We assume that all material properties are uniform and discuss different modeling assumptions for the weights a_{ij}.

Length-free transport: If we assume that the transport process does not depend on the length of the fibers but is limited by other transport mechanism (cell-to-cell transport for instance), then a suitable choice for the weighted adjacency matrix is simply $a_{ij} = n_{ij}$. Then the *standard weighted graph Laplacian* satisfying both mass conservation and Fick's condition is

$$L_{ij} = (d_i \delta_{ij} - n_{ij}), \qquad d_i = \sum_{j=1}^{N} n_{ij}, \tag{10.26}$$

where δ_{ij} is Kronecker's delta and the constant $\rho = 1/\tau$ appearing in (10.4) has the dimension of inverse time. Hence, in this diffusion process, there is no notion of distance and the particular location of different nodes is not taken into account. This process assumes the existence of a much slower time scale in the overall diffusive process which justifies the fact that transport along edges is essentially instantaneous and only modulated by the edge diameter and not its length.

Ballistic transport: A second assumption is that transport is penalized by length so that we have $a_{ij} = n_{ij}/g_{ij}$ and $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The Laplacian

$$L_{ij} = d_i \delta_{ij} - \frac{n_{ij}}{g_{ij}}, \qquad d_i = \sum_{j=1}^{N} \frac{n_{ij}}{g_{ij}}, \tag{10.27}$$

where ρ has now the dimension of a velocity (length over time). Hence, this scaling assumes that the transport motion along the edges scales as a ballistic process. This assumption is suitable to describes nonlinear processes within edges such as front propagation given by a nonlinear reaction-diffusion process. In the case of intercellular transport within an axon, if we assume that there is a pool of proteins that can interact autocatalytically with the population of toxic protein, we expect the development of a front of toxic protein traveling along the axon and this choice may be justified.

Diffusive transport: A third assumption is that transport depends as the inverse of a length squared: $a_{ij} = n_{ij}/g_{ij}^2$ and

$$L_{ij} = d_i \delta_{ij} - \frac{n_{ij}}{g_{ij}^2}, \qquad d_i = \sum_{j=1}^{N} \frac{n_{ij}}{g_{ij}^2}, \tag{10.28}$$

which implies now that ρ has the dimension of a diffusivity (length squared over time). This is the scaling that we would obtain by directly discretizing

the continuous Laplacian over a regular grid. Hence, in the absence of other assumptions about transport, it seems to be the most natural one (in the sense that the other ones require further modeling assumptions about the transport process taking place within edges or the existence of different time scales in the problem).

The three above choices are part of the general family of weights

$$a_{ij} = \frac{n_{ij}}{g_{ij}^w}, \tag{10.29}$$

where $w = 0, 1, 2$.

Distance-based transport: Yet a completely different assumption solely based on distance has been proposed in Pandya et al. (2019). In this case the adjacency matrix is not based on connectivity but directly on the distance between nodes i and j given by ℓ_{ij};

$$f_{ij} = e^{-\ell_{ij}/\ell} H(e^{-\ell_{ij}/\ell} - T) - \delta_{ij}, \tag{10.30}$$

where g is a typical length-scale (1/2 of the mean distance) and H is the Heaviside function. A further step of removing long connections with small values with an (arbitrary) threshold T is used to obtain a sparse matrix. The value chosen in their paper is $T = 0.15$ which we adopt leading for the 83 node connectome to produce a graph with 1552 edges. This model assumes that there is a connection between nodes only if they are sufficiently close. Note that we have subtracted the identity so that $a_{ii} = 0,\ \forall i$.

Normalization: The adjacency matrix is defined up to an arbitrary multiplicative constant (balanced in Eq. (10.2) by the diffusion constant). Therefore, after we obtain the adjacency matrices, we normalize them so that their largest entries is 1. The matrices given in the github repository all have this property.

There is no obvious reason to choose one type of weight against another. A conservative modeling approach is to use a ballistic or diffusion assumption (exponent 1 or 2) which can be justified when the diffusion process takes place on a regular lattice.

10.3.1 The connectome

A brain connectome is obtained by defining nodes of the network to be regions of interest, typically associated with well-known areas from a brain atlas. The edges of this network are defined as the connections between regions. The brain connectome is then modeled as an undirected weighted graph $\mathcal{G}$ with n nodes and m edges. The weights for the adjacency matrix for the simulation are derived from the tractography of diffusion tensor magnetic resonance images corresponding to 418 healthy subjects of the Human Connectome Project McNab et al. (2013) given by Budapest Reference Connectome v3.0 Szalkai et al. (2017).

Multiple resolutions with N ranging from 83 to 1015 are available. The value of n_{ij} and g_{ij} are obtained as averages over the 418 brains before the weight is computed. Here for the purpose of a general introduction, we will use the 83 node network. This network is also easier to match with other type of data (such as volume and clearance) that are typically only available at low resolution. The graph contains N =83 nodes and 1654 edges and the weighted adjacency matrix (with ballistic weights) shown in Fig. 10.3. Hence, we have

$$a_{ij} = 1/a \frac{\langle n_{ij} \rangle}{\langle l_{ij} \rangle^w}, \quad \langle n_{ij} \rangle = \frac{1}{418} \sum_{k=1}^{418} n_{ij}^{(k)}, \quad \langle g_{ij} \rangle = \frac{1}{418} \sum_{k=1}^{418} l_{ij}^{(k)}. \tag{10.31}$$

where $n_{ij}^{(k)}$ is the number of fibers in brain k between nodes i and j. The parameter w denotes the choice of transport ($w = 0$ is length-free, $w = 1$ is ballistic, $w = 2$ is diffusion). The parameter a is chosen for normalization so that the maximal entry of $\mathbf{A}$ is one. As defined above, we use the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} = \mathbf{1} \cdot \mathbf{A}$.

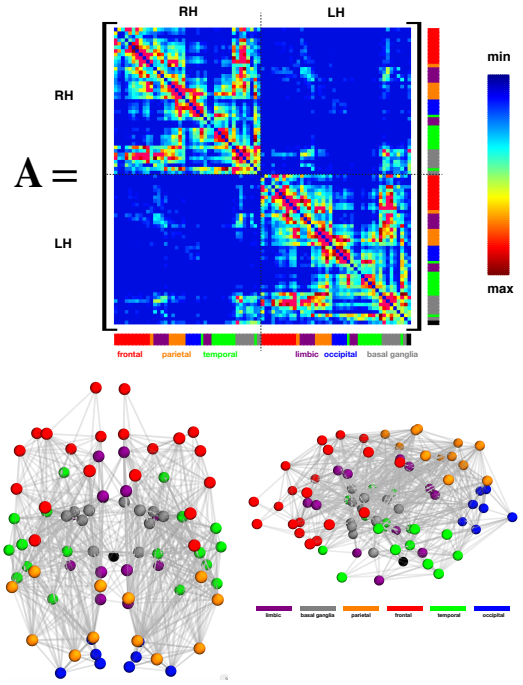


Fig. 10.3: The average weighted-adjacency of 418 brain connectomes with $n = 83$ nodes and ballistic weights, together with the node position in the brain and the lobes they belong to.

10.4 Implementation

We now show how to integrate step by step the system. Once, a given weight is chosen, we compute the corresponding graph Laplacian $\mathbf{L}$ and the problem reduces now to integrate the discrete Fisher-KPP equation:

$$\dot{c}_i = -\rho \sum_{j=1}^{N} L_{ij} c_j + \alpha c_i (1 - c_i), \quad i = 1, \ldots, N \tag{10.32}$$

$$c_i(0) = \epsilon, \quad i \in \mathcal{S}, \qquad c_i(0) = 0, \quad \text{otherwise}, \tag{10.33}$$

where $\mathcal{S}$ is the set of nodes that are seeded. For instance, it is known that for Alzheimer's disease, the disease starts in the entorhinal cortex (which corresponds to nodes 27 and 68 in our atlas. Hence, we have $c_{27}(0) = c_{68}(0) = \epsilon$ and $c_i(0) = 0$ everywhere else).

Mathematically, there are two regimes of interest depending on the ratio ρ/α. In the diffusion dominated regime ($\rho/\alpha \ll 1$) the system behaves mostly homogeneously

with the concentration of toxic proteins increasing in all regions uniformly as found for instance in the propagation of amyloid beta. However, a systematic study of data available from open database (the Alzheimer's Disease Neuroimaging Initiative) using hierarchical Bayesian parameter inference taught us that the evolution of tau proteins is in the growth-dominated regime ($\rho/\alpha \ll 1$) where each region is invaded in turn from the primary seed. This is the regime of interest for our discussion.

While it is straightforward to integrate the system of equations (10.32–10.33) for given parameter values, their numerical solutions are given by 83 different curves, as shown in Fig. 10.4 which is not particularly useful or directly interpretable.

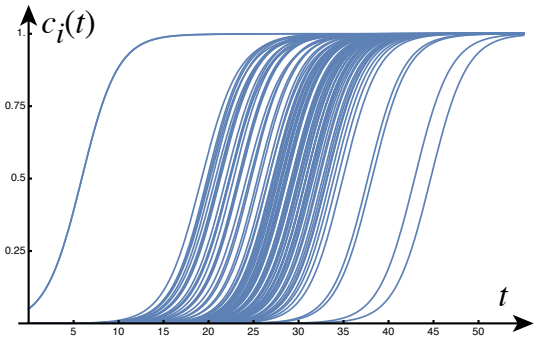


Fig. 10.4: Integration of the discrete Fisher-KPP equations with diffusive weights, $N = 83$, $\alpha = 0.5$/year, $\rho = 0.01$/year and $\epsilon = 1/20$.

A more informative way to present the results is to average the concentration by Braak regions corresponding to a subset of nodes. Using the basic decomposition in 6 regions corresponding to different stages of the disease, the Braak region i is defined by a subset $\mathcal{B}_i$ of N_i nodes. For instance Braak I is defined by the two nodes defining the entorhinal cortex $\mathcal{B}_{\mathrm{I}} = \{27, 68\}$. We define the averaged concentrations

$$C_i(t) = \frac{1}{N_i} \sum_{j \in \mathcal{B}_i} c_j(t), \qquad i = \mathrm{I}, \ldots, \mathrm{VI}. \tag{10.34}$$

An example of such a simulation is shown in Fig. 10.5 together with the different regions. In addition, an asymptotic approximation of the numerical solution is also shown following Putra et al. (2023) but will not be further discussed here.

We can now compare the different choices of weights for the Laplacian. In Fig 10.6, we run the same model with the same parameters but with different weights. We see that the dynamics is essentially the same irrespective of the choice of the weights.

10.4.1 Limitations and extensions

Despite its simplicity, this network model can be used as a basic starting point to study many different aspects of the disease, quantify new hallmarks of the disease and test possible mechanisms responsible for the onset and progression of neurodegeneration. In more recent work, we have considered local variations in parameters associated with brain inhomogeneity, derived analytical estimates for the arrival time of the propagating front of toxic proteins through the connectome Putra et al. (2023), studied the coupling between amyloid-beta and tau proteins Thompson et al. (2020), the role of clearance in the initiation and dynamics of the disease Brennan et al. (2023), the interactions between the microvasculature and toxic proteins, predicted

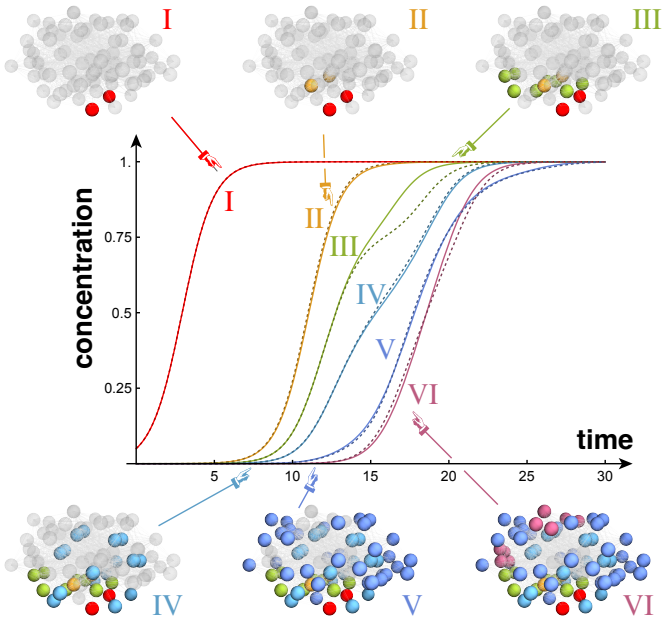


Fig. 10.5: Average concentration of toxic proteins in each Braak region. The solid curves are the numerical solutions and the dashed curves are their approximations obtained from a nonlinear perturbation expansion (see Putra et al. (2023)). Initial conditions and parameters as in the previous figure.

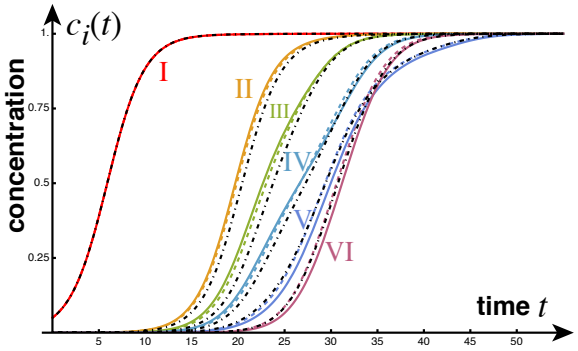


Fig. 10.6: A comparison of the three choices for the weights (diffusive: solid, ballistic: dashed, length-free: dot-dashed and black) shows little effect of the weight. Initial conditions and parameters as in the previous figure.

downstream effects such as brain atrophy from tau pathology Schäfer et al. (2021), and identified topological signatures of the disease in graph space Goodbrake et al. (2022). We have developed generalised Smoluchowski equations for the microscale aggregation kinetics and used this model to extract bifurcation points from experimental data showing the critical levels of clearance above which protein aggregation does not occur Thompson et al. (2021), and further approximated how much toxic concentration a region of the brain can withstand before the natural defences of brain clearance are overwhelmed and the region becomes saturated in toxic proteins Brennan et al. (2023). We have also studied the perplexing dynamics in brain activity observed in patients who typically show periods of hyperactivity followed by hypoactivity and a shift in brain wave frequencies. The same model coupled to so-called neuronal mass models for brain activity allowed us to test multiple hypotheses

and conclude that local damage of particular groups of neuronal cells is the most likely mechanism responsible for these observations Alexandersen et al. (2022).

Limitations of the model come from the low availability of human clinical data to apply to our models due to ethical experimental constraints, imaging measurement noise and inter-subject variability. There is always an intrinsic measure of uncertainty with our parameter values, but thankfully this is quantifiable using methods such as Bayesian inference. The availability of accurate mathematical models however will lessen the need for human experiments to further our understanding of the disease and simulate therapeutic intervention strategies. Further, exploring the parameter spaces heightens our intuition of disease dynamics from a purely theoretical view. The brain network itself and model parameters could also be personalised to predict patient-specific neurodegeneration.

10.5 Conclusion

At present, the driving factors of Alzheimer's disease are poorly understood, and there are many open questions. Only in the last decade or so has mathematical modeling joined the fight against Alzheimer's disease. Relatively simple reaction-diffusion network models of neurodegeneration reproduce the spatio-temporal spreading of toxic proteins observed in imaging studies, thus capturing the governing dynamics of the disease. Network models thus provide a powerful platform for realistic and fast computational experiments which heighten our understanding of key mechanisms of neurodegeneration and enable clinical hypotheses to be explored in silico on human brain graphs. Once network models are developed, the application of computational experiments is not only to understand the mathematical dependencies but to address the big questions at the forefront of Alzheimer's research. What interplay exists between the underlying mechanisms driving the disease? Why do we see such distinct spreading patterns in Alzheimer's disease? What sets the timescales for disease progression (approximately 40 years to full invasion)? And most pressing, is there any way to intervene in the devastating cascade of toxic proteins through the connectome?

Acknowledgments This work was supported by the Engineering and Physical Sciences Research Council grant EP/R020205/1 to Alain Goriely. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

Authors contribution. Both authors conceived the idea, discussed the main concepts of the contribution, and wrote the paper.

Data statement. All data files and a Mathematica workbook can be found in the github folder
https://github.com/goriely/Network83.

References

Alexandersen CG, de Haan W, Bick C, Goriely A (2022) A mechanistic model explains oscillatory slowing and neuronal hyperactivity in Alzheimer's disease. bioRxiv doi:10.1101/2022.06.20.496731

Braak H, Braak E (1991) Neuropathological stageing of Alzheimer-related changes. Acta Neuropathologica 82(4):239–259, doi:10.1007/BF00308809

Brennan GS, Thompson TB, Oliveri H, Rognes ME, Goriely A (2023) The Role of Clearance in Neurodegenerative Diseases. SIAM Journal on Applied Mathematics 0(0):S172–S198, doi:10.1137/22M1487801

Fornari S, Schäfer A, Jucker M, Goriely A, Kuhl E (2019) Prion-like spreading of Alzheimer's disease within the brain's connectome. J R Soc Interface 16, doi:10.1098/rsif.2019.0356

Goodbrake C, Beers D, Thompson TB, Harrington HA, Goriely A (2022) Brain chains as topological signatures for alzheimer's disease. Submitted to Journal of Applied and Computational Topology, 2208.12748

Henderson MX, Cornblath EJ, Darwich A, Zhang B, Brown H, Gathagan RJ, Sandler RM, Bassett DS, Trojanowski JQ, Lee VMY (2019) Spread of α-synuclein pathology through the brain connectome is modulated by selective vulnerability and predicted by network analysis. Nature Neuroscience 22(8):1248–1257, doi:10.1038/s41593-019-0457-5

Henderson MX, Sedor S, McGeary I, Cornblath EJ, Peng C, Riddle DM, Li HL, Zhang B, Brown HJ, Olufemi MF, Bassett DS, Trojanowski JQ, Lee VMY (2020) Glucocerebrosidase Activity Modulates Neuronal Susceptibility to Pathological α-Synuclein Insult. Neuron 105(5):822–836.e7, doi:10.1016/j.neuron.2019.12.004

Jucker M, Walker LC (2013) Self-propagation of pathogenic protein aggregates in neurodegenerative diseases. Nature 501(7465):45–51, doi:10.1038/nature12481

Masuda N, Porter MA, Lambiotte R (2017) Random walks and diffusion on networks. Physics Reports 716-717:1–58, doi:10.1016/j.physrep.2017.07.007

McNab JA, Edlow BL, Witzel T, Huang SY, Bhat H, Heberlein K, Feiweier T, Liu K, Keil B, Cohen-Adad J, Tisdall MD, Folkerth RD, Kinney HC, Wald LL (2013) The Human Connectome Project and beyond: Initial applications of 300mT/m gradients. NeuroImage 80:234–245, doi:10.1016/j.neuroimage.2013.05.074, mapping the Connectome

Pandya S, Zeighami Y, Freeze B, Dadar M, Collins D, Dagher A, Raj A (2019) Predictive model of spread of Parkinson's pathology using network diffusion. NeuroImage 192:178–194, doi:10.1016/j.neuroimage.2019.03.001

Putra P, Oliveri H, Thompson T, Goriely A (2023) Front Propagation and Arrival Times in Networks with Application to Neurodegenerative Diseases. SIAM Journal on Applied Mathematics 83(1):194–224, doi:10.1137/21M1467547

Raj A, Kuceyeski A, Weiner M (2012) A Network Diffusion Model of Disease Progression in Dementia. Neuron 73(6):1204–1215, doi:10.1016/j.neuron.2011.12.040

Schäfer A, Chaggar P, Thompson TB, Goriely A, Kuhl E (2021) Predicting brain atrophy from tau pathology: a summary of clinical findings and their translation into personalized models. Brain Multiphysics 2:100039, doi:10.1016/j.brain.2021.100039

Szalkai B, Kerepesi C, Varga B, Grolmusz V (2017) Parameterizable consensus connectomes from the Human Connectome Project: the Budapest Reference Connectome Server v3.0. Cognitive Neurodynamics 11(1):113–116, doi:10.1007/s11571-016-9407-z

Thompson TB, Chaggar P, Kuhl E, Goriely A, for the Alzheimer's Disease Neuroimaging Initiative (2020) Protein-protein interactions in neurodegenerative diseases: A conspiracy theory. PLOS Computational Biology 16(10):1–41, doi:10.1371/journal.pcbi.1008267

Thompson TB, Meisl G, Knowles TPJ, Goriely A (2021) The role of clearance mechanisms in the kinetics of pathological protein aggregation involved in neurodegenerative diseases. The Journal of Chemical Physics 154(12):125101, doi:10.1063/5.0031650

Weickenmeier J, Kuhl E, Goriely A (2018) Multiphysics of Prionlike Diseases: Progression and Atrophy. Phys Rev Lett 121:158101, doi:10.1103/PhysRevLett.121.158101

Glossary

Use the template *glossary.tex* together with the Springer Nature document class SVMono (monograph-type books) or SVMult (edited books) to style your glossary in the Springer Nature layout.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

Index