

Report

User Guide to Flow Diagnostics in MRST

Flow Diagnostics Preprocessors for Model Ensembles

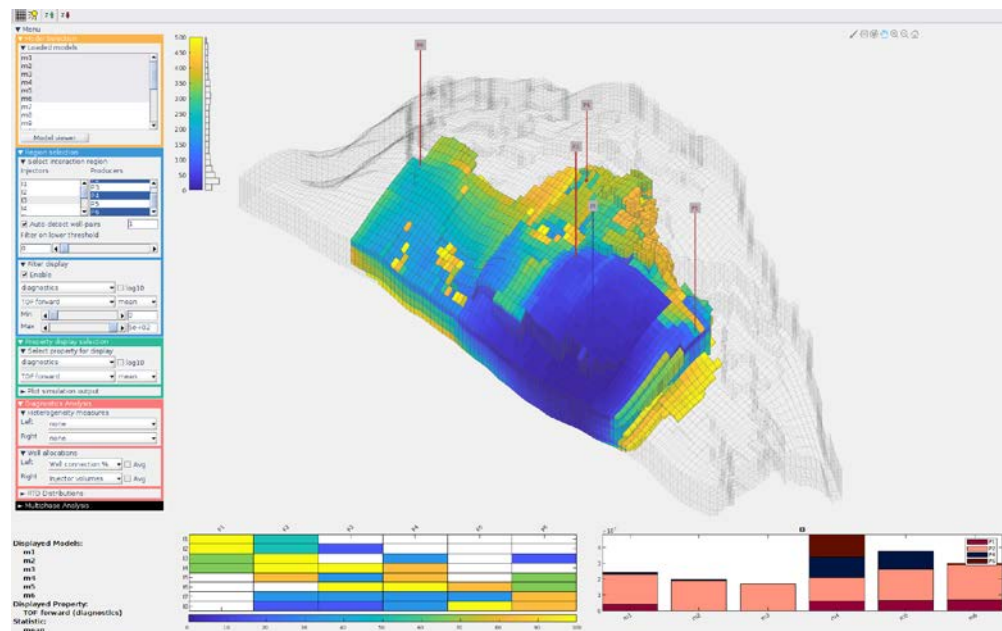
Authors

Knut-Andreas Lie

Stein Krogstad

Francesca Watson

Manuel Antonio Borregales Reverón



Report

User Guide to Flow Diagnostics in MRST

Flow Diagnostics Preprocessors for Model Ensembles

KEYWORDS:Flow Diagnostics;
MRST; MATLAB;
Reservoir Simulation;
Ensemble Modelling;
Graphical User
Interface**VERSION**

1.0

DATE

2020-11-03

AUTHOR(S)Knut-Andreas Lie
Stein Krogstad
Francesca Watson
Manuel Antonio Borregales Reverón**PROJECT NO.**

102018629

NUMBER OF PAGES/APPENDICES:

3 + Appendices

ABSTRACT

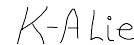
Flow diagnostics are simple quantities that can be derived from basic flow simulations to probe a reservoir model, establish connections and basic volume estimates, and measure heterogeneity in flow paths.

This user guide introduces various types of flow diagnostics, followed by an overview of two graphical user interfaces (GUIs) developed in the MATLAB Reservoir Simulation Toolbox (MRST) that can be used to quickly interrogate an ensemble of model realizations and investigate relative differences in flow patterns between them, prior to running computationally expensive, multiphase flow simulations.

The first GUI enables you to inspect and cross-plot various measures of dynamic heterogeneity as well as simplified estimates of (economic) objectivity functions such as recovery factor and net-present value for the whole ensemble. The second GUI focuses more on volumetric connections, communication patterns, and timelines for fluid transport within individual models or selected subsets of the full ensemble. It offers much of the same visualization capabilities as the GUI developed for flow-diagnostic *postprocessing* of multiphase flow simulations.

PREPARED BY

Knut-Andreas Lie

SIGNATURE**CHECKED BY**

Håvard Heitlo Holm

SIGNATURE**APPROVED BY**

Trond Runar Hagen

SIGNATURE**REPORT NO.**

2020:01131

ISBN

978-82-14-06437-7

CLASSIFICATION

Unrestricted

CLASSIFICATION THIS PAGE

Unrestricted

Document history

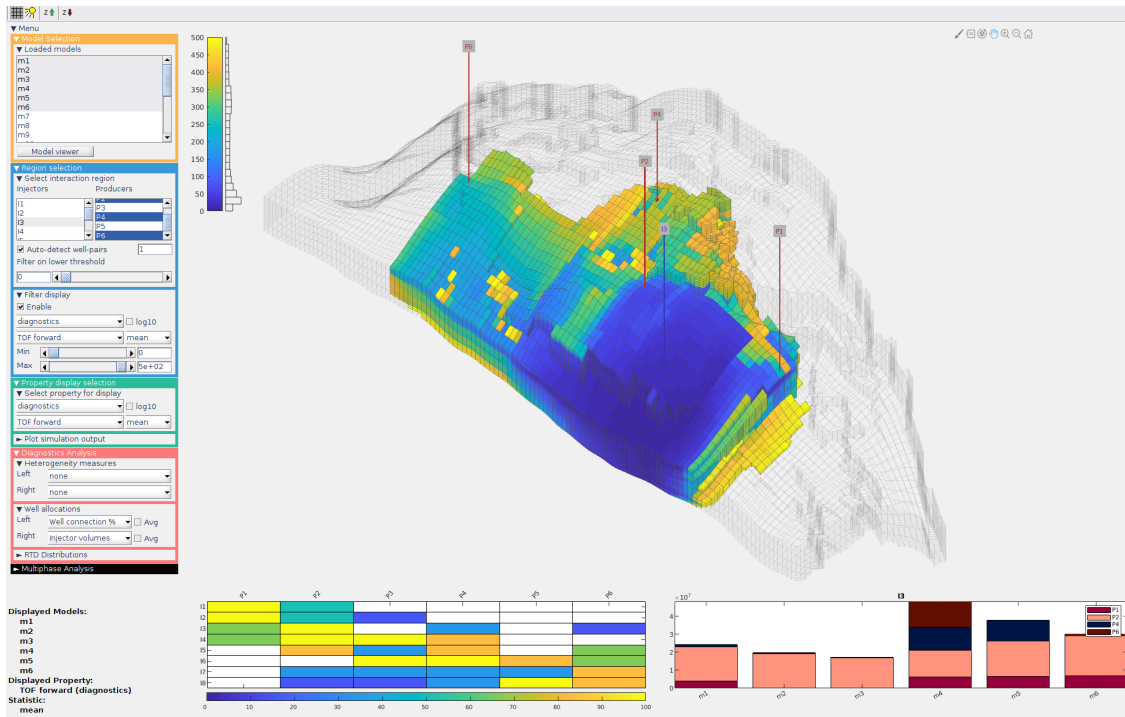
VERSION	DATE	VERSION DESCRIPTION
1.0	2020-11-03	First Version



Technology for a better society

www.sintef.no

User Guide to
FLOW DIAGNOSTICS IN MRST
Flow diagnostics preprocessors for model ensembles





CONTENTS

Contents	1
1 Introduction	3
2 What is flow diagnostics?	5
2.1 Time-of-flight	5
2.2 Influence regions	6
2.3 Volumetric partitions and well-allocation factors	7
2.4 Measures of dynamic heterogeneity	8
2.5 Residence-time distributions	10
2.6 Extension to compressible flow	12
2.7 Estimating recovery and phase rates from residence-time distributions	15
2.8 Economic measures: net present value	16
2.8.1 Using cell-averaged time-of-flight	16
2.8.2 Using residence-time distributions	17
3 Getting started	19
3.1 Inspecting a full model ensemble	19
3.1.1 Setting the ensemble	19
3.1.2 Precomputing summary diagnostics	20
3.1.3 Quick overview of the ensemble GUI	20
3.1.4 Launching the GUI	21
3.2 Inspecting and comparing specific ensemble members	21
3.2.1 Quick overview of the diagnostics viewer GUI	21
3.2.2 Launching the GUI	22
4 Flow diagnostics for a full ensemble	25
4.1 Cross plots of single-valued quantities	25
4.2 Diagnostics and production curves	27
4.3 Highlighting and selecting individual models	27
4.4 Setting time horizon and domain of definition	29
4.5 Setting fluid and economic properties	29
4.6 Histogram for static parameters	30
5 Selecting and displaying 3D data	33
5.1 Selecting model realizations	33
5.2 Comparing several models in 3D	33
5.3 Displaying cell properties	34
5.3.1 Static properties	34
5.3.2 Dynamic properties	34
5.3.3 Diagnostics properties	34
5.4 Simulation output (well solutions)	35
5.5 Selecting a subset of the reservoir	35
5.5.1 Select wells and interaction regions	35
5.5.2 Property filter	36

6	Flow-diagnostics analysis of sets of models	37
6.1	Heterogeneity measures	37
6.2	Volumetric partitions and well allocation	38
6.3	Residence-time distributions	39
6.4	Multiphase Analysis	40
	Bibliography	43
A	Setting up the software	45
A.1	Prerequisites	45
A.2	Installing MRST	45
B	Overview of the DiagnosticsViewer and EnsembleGUI classes	47
B.1	Diagnostics Viewer	47
B.2	Ensemble GUI	49

Flow diagnostics are simple quantities that can be derived from basic flow simulations to probe a reservoir model, establish connections and basic volume estimates, and measure heterogeneity in flow paths. The result is a set of visually intuitive quantities that:

- give the travel time for mass-less particles that passively follow the flow field from an injector into the reservoir and from a point in the reservoir to the nearest producer;
- delineate regions drained by specific producers or swept (flooded) by specific injectors;
- determine whether pairs of injectors and producers communicate or not and measure the relative strength of their connection;
- determine how flux is allocated between different injectors and producers;
- establish the volumetric region influenced by specific well-pairs;
- give the distribution of residence-times for all flow paths connecting injectors and producers;
- measure the dynamic heterogeneity (i.e., the spread in the residence times of flow paths and their associated throughput) within drainage, sweep, or well-pair regions.

All these quantities are quick to compute and can thus be used to interactively explore fluid communication in a geological model before or after more comprehensive multiphase flow simulations.

The speed at which these diagnostics can be calculated is especially useful when you have an ensemble of models which you would like to simulate to span the range of uncertainty. For a large ensemble it would be unfeasible to run a full simulation for every realization. Flow diagnostics can be used to quickly choose realizations that are likely to span the range of uncertainty present and are therefore good candidates for further investigation.

This user guide gives an introduction to various types of flow diagnostics, followed by an overview of two graphical user interfaces (GUIs) developed in the MATLAB Reservoir Simulation Toolbox (MRST) [9] that can be used to quickly interrogate an ensemble of model realizations and investigate relative differences in flow patterns between them, prior to running computationally expensive, multiphase flow simulations. The first GUI enables you to inspect and crossplot various measures of dynamic heterogeneity as well as simplified estimates of (economic) objectivity functions such as recovery factor and net-present value for the whole ensemble. The second GUI focuses more on volumetric connections, communication patterns, and timelines for fluid transport within individual models or selected subsets of the full ensemble. It offers much of the same visualization capabilities as the GUI developed for flow-diagnostic *postprocessing* of multiphase flow simulations [18] conducted with MRST's own AD-OO simulator framework [9, Chapter 12], or imported from file using the ECLIPSE output format [14].

To make the user guide more self-contained, we start with a brief summary of the basic flow-diagnostic quantities. You can find a more comprehensive introduction in the recent textbook by Lie [9, Chapter 13] or in one of the two original articles that introduce these methods [10, 15].

Many of the concepts that constitute what we herein refer to as “flow diagnostics” were originally developed within streamline simulation [2, 16, 20, 21]. Although MRST has functionality for computing streamlines for certain grid types, we have chosen to implement flow diagnostics using cell-wise quantities that are computed with standard finite-volume discretizations. This requires some small conceptual adjustments as to what the derived quantities mathematically represent. In particular, quantities that have a pointwise or line-specific interpretation in streamline simulation become volumetric quantities that represent the average over all streamlines passing through a cell or cell interface when calculated with a finite-volume method. More details are given in the following.

2.1 Time-of-flight

Flow diagnostics relies on two basic quantities: time-of-flight and influence regions. To define these, we assume a reservoir with porosity ϕ and a superficial Darcy velocity \vec{v} defined as volumetric discharge per area. A streamline is defined as a curve that is tangential to the velocity field at every point; see Figure 2.1. In a steady (incompressible) flow field, there will be a unique streamline passing through each point \vec{x} of the reservoir, starting at the nearest injector (or fluid source) and terminating at the nearest producer (or fluid sink). We can parameterize the path traced out by streamline Ψ by its arc length r , i.e., write $\vec{x}_\Psi(r)$. In streamline simulation [2], however, it is more common to use *time-of-flight* (TOF), which measures the travel time of a massless particle by the interstitial fluid velocity \vec{v}/ϕ , as spatial coordinate along each streamline. We can define the TOF coordinate τ through two mathematically equivalent equations:

$$\tau(r) = \int_0^r \frac{\phi(\vec{x}_\Psi(s))}{|\vec{v}(\vec{x}_\Psi(s))|} ds, \quad \vec{v} \cdot \nabla \tau = \phi. \quad (2.1)$$

In flow diagnostics, we usually refer to two different TOF values: the *forward TOF* τ_f defined as the travel time from the nearest injector to a given point in the reservoir, and the *backward TOF* τ_b defined as the travel time from the given point to the nearest producer. The total travel time along a streamline from inflow to outflow is called *residence time* and equals the sum of forward and backward time-of-flight. Numerically, you can compute *point values* of τ by tracing streamlines and evaluating the integral in (2.1). Herein, we will follow [10, 15] and compute *volume averaged values* of τ from a standard finite-volume discretization of $\vec{v} \cdot \nabla \tau = \phi$, as first proposed by [3, 11]. The relationship among streamlines, time-of-flight, and residence time is illustrated in Figure 2.2 for a rectangular reservoir with two injectors and three producers.

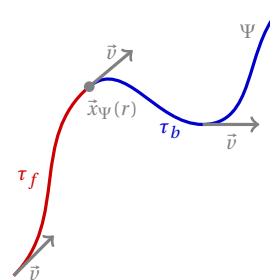


FIGURE 2.1: A streamline Ψ passing through a point \vec{x} is tangential to the velocity field \vec{v} at every point. The streamline can be parameterized by its arc length r , or the travel time defined relative to the interstitial fluid velocity \vec{v}/ϕ .

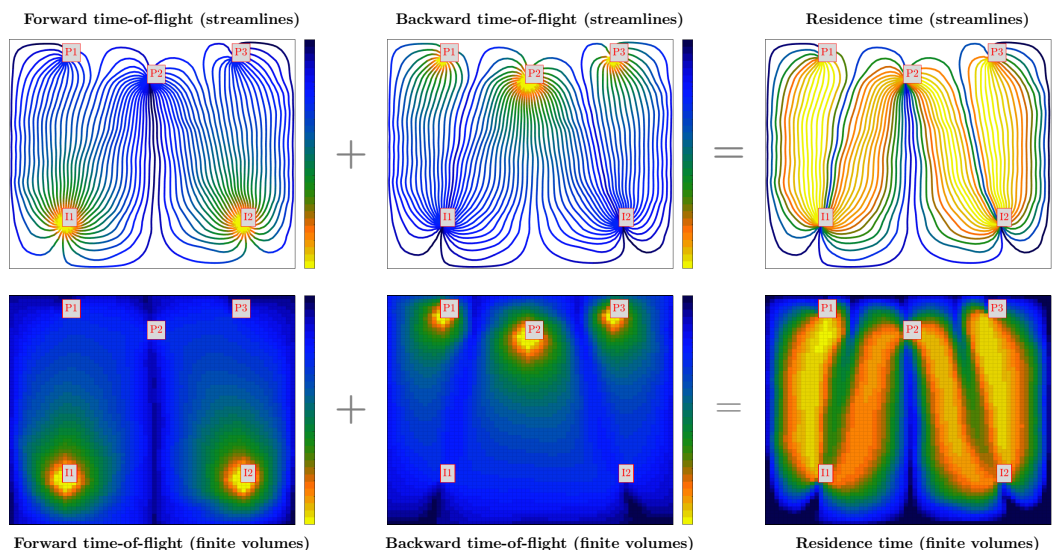


FIGURE 2.2: Relationship among streamlines, time-of-flight, and residence time for a rectangular reservoir with two injectors and three producers. Computing with a streamline method gives pointwise values, or to be more precise, increment values associated with line segments that cross individual cells, whereas a finite-volume method computes average values over all flow paths crossing each cell.

2.2 Influence regions

Influence regions are defined as the locus of the points that lie on all streamlines emanating from a given point, line segment, or volumetric object that represents a fluid source or sink. Since each point that is not a fluid source or sink in an flow field can only lie on a single streamline, these influence regions are distinct volumetric objects that introduce a natural volumetric partition of a fixed flow field. Assuming incompressible flow, we can define an influence region mathematically as

$$\vec{v} \cdot \nabla C = 0, \quad C|_{\text{inflow}} = 1. \quad (2.2)$$

If this equation is solved with a monotone finite-volume method, the computed values will lie in the interval $[0, 1]$ and represent the average over a finite volume. You may also see this quantity referred to as a *steady tracer¹ concentration*, which gives the portion of the total fluid volume passing through a point in the reservoir that can be attributed to a given fluid source or parts of the inflow boundary. Likewise, by reversing the sign of the flow field, we can compute influence regions associated with fluid sinks. In both cases, these C values form a partition of unity; see Figure 2.3. With a standard first-order discretization, the influence regions will contain significant numerical diffusion and will thus not have the sub-cell resolution and be as sharply defined as is possible if the regions are delineated by tracing a large number of streamlines. On the other hand, the advantage of using a finite-volume discretization is that this gives the regions that will be influenced by passively advected quantities in a standard multiphase simulation. Likewise, one avoids the problem of ensuring sufficient streamline coverage. (For improved accuracy and pointwise resolution, one can also use higher-order discontinuous Galerkin methods [3, 11, 13], which, like finite-volume methods, are applicable to general grid systems.)

Because each cell can contain several influence regions, it is sometimes advantageous to compute time-of-flight associated with each influence region by solving the following equation for τ

$$\vec{v} \cdot \nabla(\tau C) = \phi C, \quad \tau|_{\text{inflow}} = 1 \quad (2.3)$$

for a fixed C from (2.2) to reduce undesirable averaging effects.

¹We emphasize that the word “tracer” here refers to purely numerical quantities that should not be confused with the numerical modelling of *inter-well tracers*, i.e., substances that are either placed in well completions or injected along with displacing fluids to monitor the flow inside the reservoir.

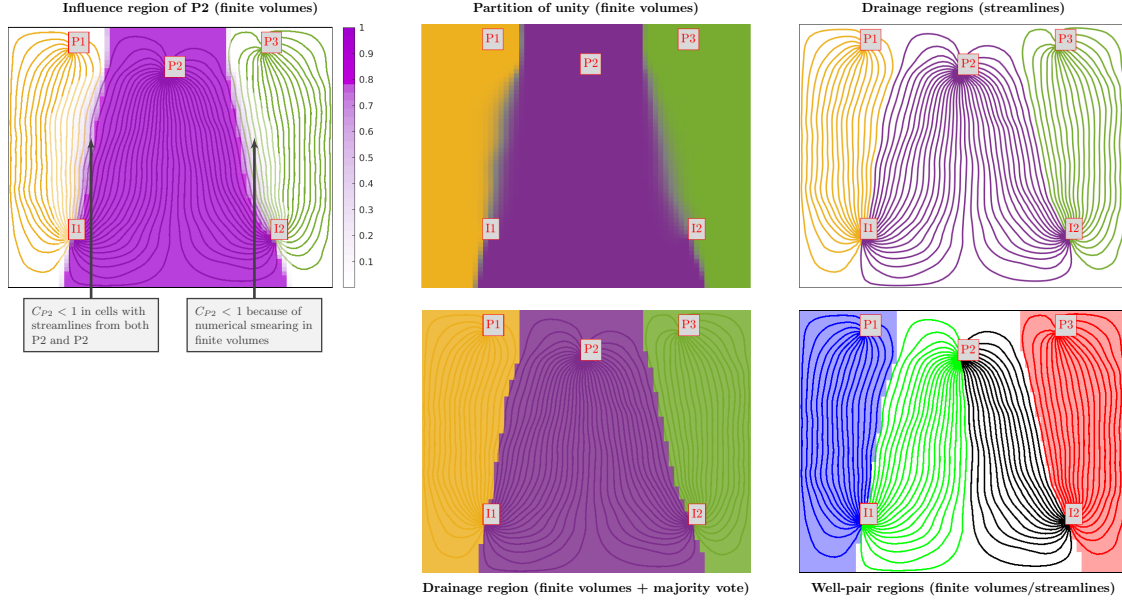


FIGURE 2.3: Volumetric partitions computed by labeling streamlines by the wells they connect to. Alternatively, one can solve one stationary transport equation of the type (2.2) for each well. The C -values for all injectors, or for all producers, define two partitions of unity of the reservoir volume. From these, we can define drainage and sweep regions and well-pair regions.

2.3 Volumetric partitions and well-allocation factors

From influence regions, we can define several quantities that represent the volumetric communication in the reservoir. Each influence region naturally delineates the drainage/sweep volume associated with a given producer/injector; for cells with multiple non-zero tracer values, we use a majority vote to assign a unique region to each cell. By computing the intersection of drainage and sweep regions, you can determine whether an injector has fluid communication with a producer and compute the corresponding flow volume (well-pair volume). This is illustrated in Figure 2.3.

You can also compute *well-allocation factors* that measure the communication strength between injectors and producers. The flux allocation from injector n to perforated cell c_k^m of producer m is defined as

$$\mathbf{a}_{mn}^p[c_k^m] = \mathbf{C}_n^i[c_k^m] \mathbf{q}[c_k^m], \quad (2.4)$$

where \mathbf{C}_n^i denotes the vector of injector tracer values associated with well n and \mathbf{q} the vector of volumetric source terms for all perforated cells. The flux allocation from producer m to injector n perforated in cell c_k^n is defined analogously as $\mathbf{a}_{nm}^i[c_k^n] = \mathbf{C}_m^p[c_k^n] \mathbf{q}[c_k^n]$. Collecting the allocation factors \mathbf{a}_{mn}^p from all injectors n connected to producer m gives you the volumetric inflow rate that can be attributed to each of the connected injectors. Likewise, collecting the allocation factors \mathbf{a}_{nm}^i gives the volumetric inflow rate into each of the connected producers that can be attributed to injector n . In the GUI, we plot these allocation factors as a set of stacked bars, one stack of bars for each perforation, where each bar represents one allocation factor, colored by a unique color for each connected well; see Figure 2.4.

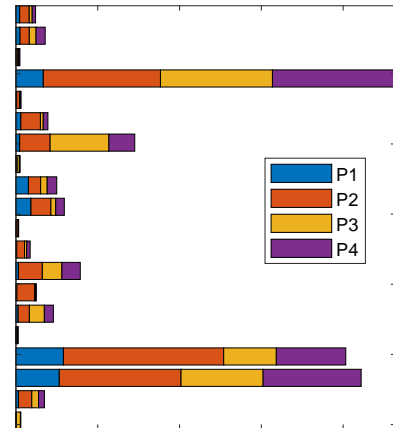


FIGURE 2.4: Allocation plot for an injector that communicates with four different producers. The stacked bars represent the flux out of each of the 20 perforations from heel (top) to toe (bottom).

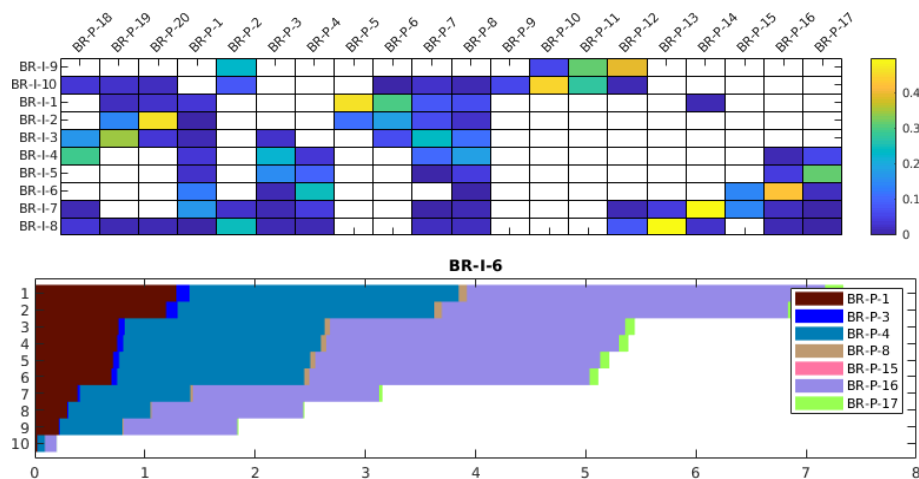


FIGURE 2.5: Visualization of well-allocation factors for a simulation of the Brugge model [12]. The upper plot shows the relative communication strength of each well pair over the whole simulation period, computed by summing the instantaneous injector allocation factors $\sum_k a_{nm}^i [c_k^n] / \sum_k q$ and integrating the result over all time steps in the simulation. The lower plot shows the instantaneous, cumulative allocation from toe to heel for a specific time step.

Example (Brugge field). Figure 2.5 shows two examples of how you can use this information to visualize the relative fluid communication in the Brugge simulation model [12]. The upper plot shows a matrix plot of the relative communication strength between the ten injectors (rows) and twenty producers (columns), averaged over all time steps in the simulation. Here, no color means no communication. If you look carefully, you can see that the upper plot shows significant communication between injector BR-I-6 and producer BR-P-16. The lower plot reports instantaneous outflow from well BR-I-6 at a specific time, shown as a cumulative plot from toe to heel (bottom to top of well). Here, you can see a large proportion of the outflow from BR-I-6 is allocated to producer BR-P-16.

2.4 Measures of dynamic heterogeneity

Secondary and tertiary recovery is usually strongly governed by the intrinsic variability (heterogeneity) in petrophysical properties. Classical sweep theory includes a number of static measures for characterizing heterogeneity, such as flow and storage capacity (Stiles' diagrams [19]), Lorenz coefficient, Koval factor, and Dykstra-Parson's permeability variation coefficient; see e.g., [8] for an overview. In flow diagnostics, some of these measures have been reinterpreted in a *dynamic* setting so that they measure the heterogeneity in flow paths (and connection structure) rather than measuring the heterogeneity in the spatial distribution of permeability and porosity. Large dynamic heterogeneity means large variations in the length and throughput of flow paths between injectors and producers, which in a water- or gas-flooding scenario typically manifests itself as early breakthrough of injected fluids.

F-Phi diagram: The first example of a dynamic heterogeneity measure is *flow and storage capacity*, which we compute from the total residence time (i.e., the sum of forward and backward time-of-flight) and the relationship $q_i \tau_i = V_i$ between pore volume V_i , flow rate q_i , and residence time τ_i of each cell c_i . To understand this measure, you can think of the reservoir as a bundle of streamtubes of infinitesimal width, sorted so that their residence times are ascending; see Figure 2.6. If we assume piston displacement (blue fluid displacing red fluid) inside each streamtube, the storage capacity Φ at time t is the volume of all streamtubes that have “broken through”, i.e., the volume of all streamtubes that have a lower total residence time than t . With a slight abuse of notation, we write this as

$$\Phi(t) = \int_0^t \int_{\Psi(\tau)} \phi(\vec{x}_\Psi(s)) ds d\tau. \quad (2.5)$$

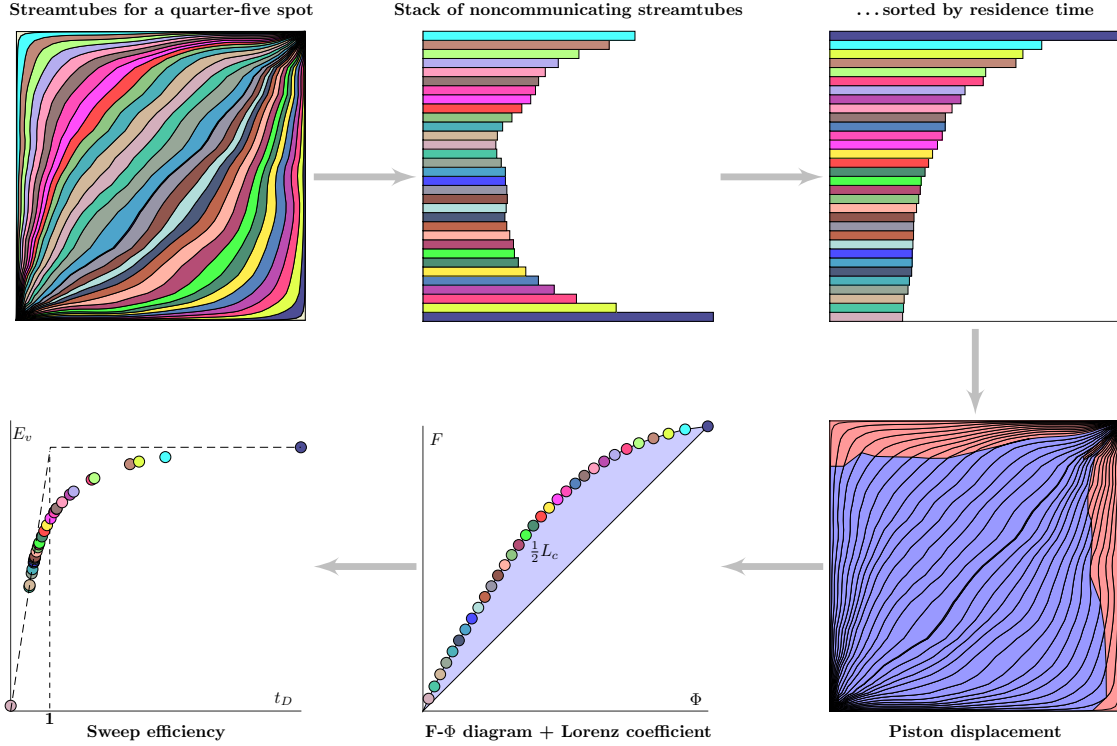


FIGURE 2.6: Illustration of how one can construct dynamic heterogeneity measures based on a bundle of streamtubes that divide the reservoir into a set of isolated flow channels that each has a given flow rate and a pore volume. The streamtubes are sorted according to residence time. If we now assume piston displacement with a blue fluid displacing a red fluid, we can record how the fractional flux F of blue fluid increases at the outlet as an increasing amount of streamtubes become completely filled by blue fluid. We also record the fractional volume Φ of the streamtubes that have been completely flooded. This gives the F - Φ diagram. The Lorenz coefficient is twice the area between the curves $y = F(\Phi)$ and $y = \Phi$. The sweep efficiency $E_v(t)$ is defined as the fraction of in-place fluid (red) that has been displaced by injected fluid (blue) by time t .

Here, $\Psi(\tau)$ is interpreted as all streamtubes with residence time equal τ . The flow capacity F is the corresponding fractional flow, i.e., the fraction of injected fluid to the total amount of fluid produced,

$$F(t) = \int_0^t \int_{\Psi(\tau)} q(\vec{x}_{\Psi}(s)) ds d\tau = \int_0^t \int_{\Psi(\tau)} \frac{\phi(\vec{x}_{\Psi}(s))}{s} ds d\tau. \quad (2.6)$$

Both quantities are normalized by their value at time infinity, giving relationships as in Figure 2.6. The $F(\Phi)$ diagram is generally a concave function, except in the special case of a perfectly homogeneous displacement, for which $F = \Phi$. The steep initial slope in a concave $F(\Phi)$ diagram corresponds to high-flow regions giving early breakthrough and the flatter tail corresponds to low-flow and stagnant regions. The more concave $F(\Phi)$ is, the larger is the spread in residence times for characteristic flow paths.

Lorenz coefficient: This scalar quantity, typically denoted L_c , measures the difference in flow capacity from that of a perfectly homogeneous displacement and is defined as twice the area under the concave $y = F(\Phi)$ curve and above the straight line $y = \Phi$; see Figure 2.6. This means that $L_c = 0$ for a homogeneous displacement and $L_c = 1$ for an infinitely heterogeneous displacement. Experience has shown that the dynamic Lorenz coefficient in many cases correlates (surprisingly) well with forecasts of hydrocarbon recovery predicted by more comprehensive flow simulations as long as the initial fluid distribution is relatively uniform. It can hence be used as an effective flow proxy in various reservoir management workflows; see [10, 15, 16, 22].

Sweep efficiency: The *volumetric sweep efficiency* $E_v(t)$ measures how efficiently injected fluids are used to displace in-place fluids. It is defined as the volume fraction of in-place fluid that has been displaced by injected fluid, or equivalently, the ratio between the volume contacted by the displacing fluid at time t and the volume contacted at time $t = \infty$. Using forward time-of-flight τ_f , we can express it as,

$$E_v(t) = \frac{1}{\Phi_0} \int_{\{\vec{x}: \tau_f(\vec{x}) \leq t\}} \phi(\vec{x}) d\vec{x} = \frac{1}{\Phi_0} \int_{\Omega} \int_0^t \delta(s - \tau_f(\vec{x})) \phi(\vec{x}) ds d\vec{x}, \quad (2.7)$$

where Φ_0 is the total pore volume. If we sort the indices of the cells according to ascending τ_f values, we can compute a cell-based estimate of sweep efficiency as

$$E_v(t) = \sum_{\{j | \tau_{f,j} \leq t\}} V_j / \sum_{j=1}^N V_j. \quad (2.8)$$

Alternatively, one can show that E_v can be computed from the F - Φ diagram using the following formula

$$E_v = \Phi + (1 - F)t_D, \quad t_D = \frac{d\Phi}{dF}, \quad (2.9)$$

where t_D represents dimensionless time. In a homogeneous piston displacement, all the in-place volume will be displaced by time t_D , and thus t_D represents units of pore volume injected. Before first breakthrough ($t < \min \tau_f$), E_v equals the injected pore volume t_D . After breakthrough, Φ is the volume fraction of flow paths that have been fully swept, whereas $(1 - F)t_D$ is the volume fraction contributed by the swept parts of the flow paths that have not yet broken through. This means that for a heterogeneous displacement, the *volumetric sweep-efficiency diagram* (t_D, E_v) is a concave curve bounded above by $y = \min(x, 1)$, see Figure 2.6, which highlights fluid responses after first breakthrough.

Fractional recovery: The *fractional recovery diagram* ($t_D, 1 - F$) emphasizes breakthrough behavior and can have utility as a proxy for early-time fractional recovery. The name can be somewhat misleading and should not be confused with recovery factors computed based on multiphase simulations.

Multiphase extensions: The heterogeneity measures described thus far are all computed without taking any multiphase information into account, other than possibly through the stationary flow field. This may not always be sufficient to obtain good correlation with recovery factors predicted by multiphase flow simulation simulation, in particular if the fluid distribution inside the reservoir is largely nonuniform. relatively uniform. Approaches that compute, e.g., the Lorenz coefficient to the individual phases [4, 10] have the weakness that they only consider the heterogeneity within the phase volume and not how far, measured in τ_b , any movable volume is from a producer. The exception is sweep efficiency, which generalizes more naturally to individual phases. The oleic sweep efficiency, for instance, reads [22]:

$$E_{v,o}(t) = \frac{1}{V_o} \int_{\{\vec{x}: \tau_b(\vec{x}) \leq t\}} \phi(\vec{x}) S_o(\vec{x}) d\vec{x} = \frac{1}{V_o} \int_{\Omega} \int_0^t \delta(s - \tau_b(\vec{x})) \phi(\vec{x}) S_o(\vec{x}) ds d\vec{x}, \quad (2.10)$$

where S_o is the oil saturation and $V_o = \int_{\Omega} \phi S_o d\vec{x}$ is the total oil volume. Here we have used τ_b to approximate the oil volumes produced at time t rather than the oil volumes displaced (as would be the case using τ_f). For the total sweep (2.7), however, the forward and backward expressions are equivalent.

2.5 Residence-time distributions

We have already emphasized that time-of-flight computed from a finite-volume discretization of the second equation in (2.1) represents a volumetric average over all τ values inside each grid cell. In particular, one can show that the residence time represents the pore volume of the backward influence region from the outlet point divided by the outflux at this point; see [7, 9] for details. Hence, this residence time represents the average of a distribution that potentially can have (very) large variance, as illustrated in Figure 2.7. Using such residence times to compute dynamic heterogeneity measures may

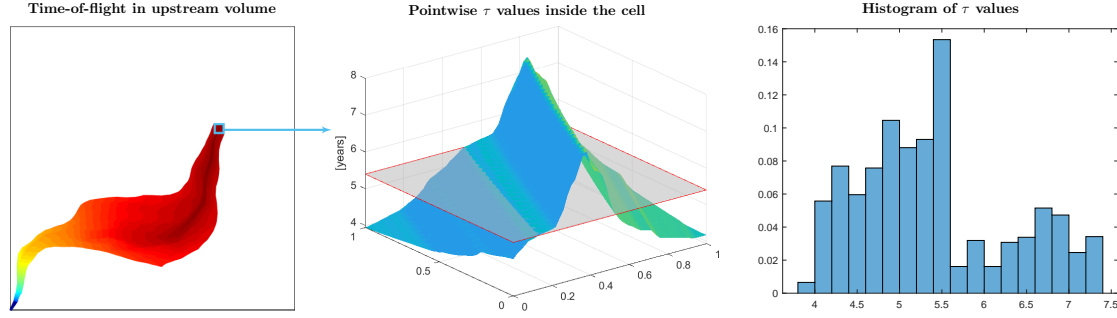


FIGURE 2.7: Illustration of subcell variation of time-of-flight. The left plot shows τ values inside the upstream volume of the cells, i.e., along all flow paths that pass through the cell. The colored surface in the middle plot exhibit the spatial variation of τ inside the cell, whereas the gray plane represents the average value. The right plot shows the histogram of τ values defined over a 51×51 mesh covering the cell.

thus potentially introduce a significant bias. Fortunately, this bias appears to be systematic [13], so that derived measures can still be used to, e.g., rank model ensembles or as simple reduced-order models to predict recovery of secondary oil recovery [10]. On the other hand, average TOF-values will in most cases *overestimate* the time to breakthrough in heterogeneous displacements (the arithmetic mean is significantly impacted by the large values that contribute little to flow). To compute times of first arrival more accurately, it is thus better to use a graph algorithm to compute the shortest path of the discrete flux graph. Likewise, one can estimate the variation σ_τ within a cell by computing second moment by solving $\vec{v} \cdot \nabla \sigma_\tau = 2\phi\tau$.

For a better description of the dynamic heterogeneity of a reservoir model, we can consider the *distribution* of time-of-flight for each cell. This is particularly interesting for cells perforated by production wells, since pointwise time-of-flight values in these cells describe the residence times, or time to breakthrough, for individual flow paths. To this end, we can solve the linear transport equation

$$\phi \frac{\partial \eta}{\partial t} + \vec{v} \cdot \nabla \eta = 0, \quad \eta|_{\Gamma_i} = \delta(t), \quad \eta(\vec{x}, 0) = 0, \quad (2.11)$$

and compute the evolution of a unit pulse from inflow Γ_i to outflow Γ_o , as illustrated in Figure 2.8. Herein, we use the backward Euler method for temporal discretization and the same upstream finite-volume method as for the steady-TOF equation for the spatial discretization. The resulting method is robust and can take large time steps, but is not very accurate. In each time step, we must solve a linear problem, but the coefficient matrix is triangular, possibly after a permutation, and hence inexpensive to invert. (This also holds for the discretization of (2.1) and (2.2).)

For each point \vec{x} , the normalized TOF-distribution $\mathcal{T}(\cdot; \vec{x})$ is now defined by the Dirac function

$$\mathcal{T}(t; \vec{x}) = \eta(\vec{x}, t) = \delta(t - \tau(\vec{x})). \quad (2.12)$$

At the outflow, the normalized residence-time distribution (RTD) is given as

$$\mathcal{T}_o(t) = \frac{1}{F_o} \int_{\Gamma_o} \eta(\vec{x}, t) \vec{v} \cdot \vec{n} \, ds, \quad F_o = \int_{\Gamma_o} \vec{v} \cdot \vec{n} \, ds. \quad (2.13)$$

It follows from the definition of the Dirac distribution that $\int \mathcal{T}_o(t) \, dt = 1$. We can also use this distribution to compute flow and storage capacity [17]

$$F(t) = \int_0^t \mathcal{T}_o(s) \, ds, \quad \Phi(t) = \frac{F_o}{\Phi_o} \int_0^t s \mathcal{T}_o(s) \, ds, \quad (2.14)$$

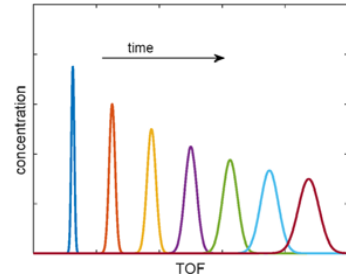


FIGURE 2.8: Numerically computed tracer pulse mapped onto a time-of-flight coordinate.

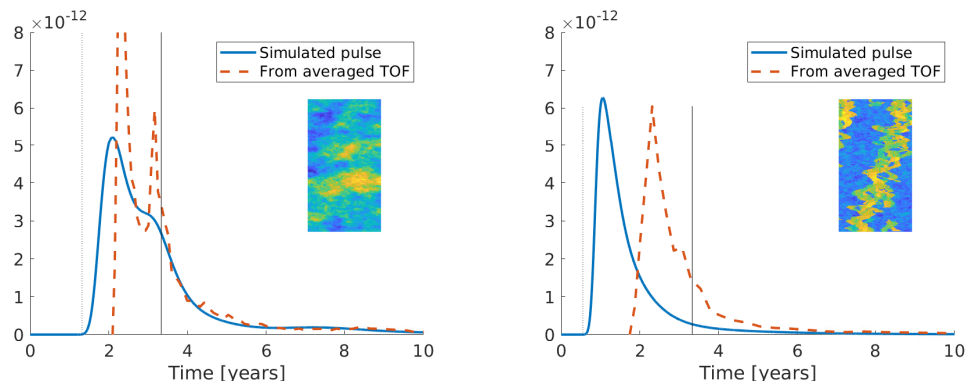


FIGURE 2.9: Residence-time distributions for two subsamples from the SPE 10 benchmark [1]: Tarbert (left) and Upper Ness (right). Thin solid and dotted lines represent the mean of the distribution, i.e., time to inject one pore volume and time to breakthrough for the fastest flow path.

where Φ_o is the total pore volume drained by the outflow boundary Γ_o . As before, both quantities are normalized so that $F(\infty) = \Phi(\infty) = 1$. From this definition, it also follows that the mean value of $\mathcal{T}_o(t)$ corresponds to the time $\bar{t} = \Phi_o/F_o$ it takes to inject one pore volume (1 PVI).

To illustrate the difference between averaged time-of-flights computed from (2.1) and the residence-time distributions, we include an example from [7, 9]. The solid lines in Figure 2.9 report $\mathcal{T}_o(t)$ without normalization as function of time for two rectangular reservoirs with an injector along the south boundary and a producer along the north boundary. (The integral of this curve equals the total allocation.) The leading pulse for the Tarbert layer is spread out and has a small secondary hump. For Upper Ness, the pulse breaks through earlier and is more focused because of high permeability channels connecting the south and north boundaries. The mean of each distribution equals 1 PVI by construction. This may not be apparent from the plots, since the distributions have very long tails, particularly in the channelized case. We can also estimate the same distribution from the average residence times τ_r ; details are given in [9, §13.3]. Using averaged TOF-values introduces a significant delay in the breakthrough time, in particular for Upper Ness. The two types of distributions also suffer from different types of numerical errors: Tracing a pulse by a finite-volume method preserves flux allocation and not pore volume and can also contain significant temporal smearing. Backing out a distribution from averaged TOF values preserves total volume but not flux allocation. The corresponding F - Φ diagrams computed from the residence-time distribution are more concave and the Lorenz coefficients are somewhat larger than when computed from averaged TOF-values. However, the differences are not very large, and since the measures computed from averaged TOF-values carry a systematic underestimation bias and are much quicker to compute, they can thus still be robustly used to rank and discriminate different cases. For more accurate predictions of recovery and recovery factors, RTD-measures should be used [22].

2.6 Extension to compressible flow

So far, we have assumed that the flow field is steady, immiscible, and incompressible. Most multiphase simulations used in reservoir engineering consist of multiple time steps and assume *compressible* conditions. Typically, the flow paths and the inter-well communication will change over time because the fluid mobility changes when fluids displace each other. Flow in and out of wells varies with time because of pressure and mobility changes as a result of changes in well controls, in particular changes that turn wells on and off. The flow diagnostics discussed earlier in this chapter only depict time lines, volumetric connections, and heterogeneity in dynamic flow paths *at each instance in time*, assuming that the current flow field persists till infinity. This is the same as in streamline methods [2], in which the streamlines are instantaneous views of the flow field that are bound to change over time.

Compressibility and interphase mass transfer will generally cause fluid compression and expansion throughout the reservoir. However, these effects do not change the computation of time-of-flight and

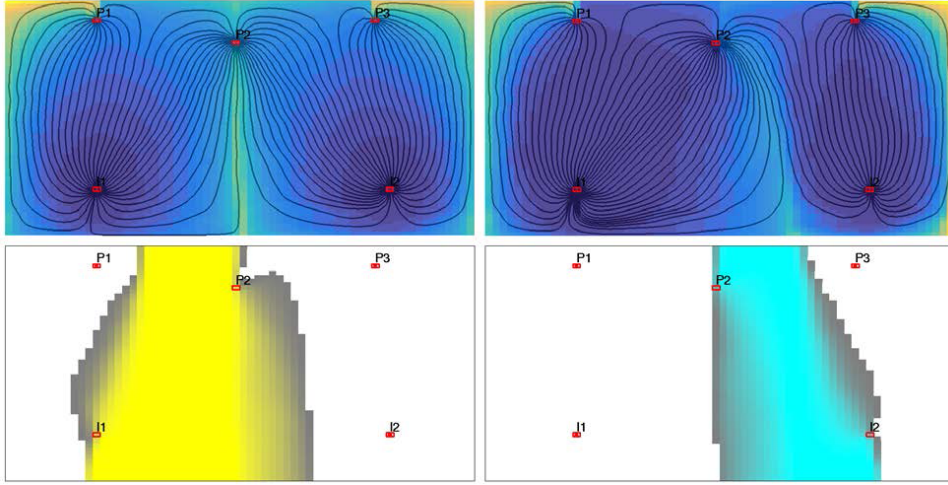


FIGURE 2.10: The flow field will typically change throughout a dynamic simulation, here exemplified by the streamlines and time-of-flight initially and at the end of a simulation in which the injection rate of the left injector increases relative to that of the right injector. Summing the tracer partitions for each time step gives a measure of how long each grid cell has been part of a given well-pair region. In the bottom row, bright colors signify that cells are part of the well-pair region for all time steps, whereas grayish colors signify that a cell is part of a region for a fraction of the simulation time; the more gray, the less time the cell has been in the region.

stationary tracer partitions. To see this, we start by writing the time-of-flight equation in conservative form, $\nabla \cdot (\vec{v}\tau) - \tau \nabla \cdot \vec{v} = \phi$, and let ϕ_i and τ_i denote the porosity and the unknown TOF value in cell i and v_{ij} be the flux from cell i to cell j . Moreover, let $U(i)$ be the set of upstream neighbors of cell i , i.e., the set of indices j such that v_{ji} is directed *into* cell i . Likewise, we let $D(i)$ be the set of downstream neighbors. For incompressible flow, the standard upwind finite-volume method reads,

$$\phi_i = \sum_{j \in U(i)} v_{ij} \tau_j + \sum_{j \in D(i)} v_{ij} \tau_i = - \sum_{j \in U(i)} v_{ji} \tau_j + \tau_i \sum_{j \in U(i)} v_{ji}.$$

because $v_{ji} = -v_{ij}$ and $\sum_{j \in D(i)} v_{ij} - \sum_{j \in U(i)} v_{ji} = 0$. For compressible flow, it follows by cancellation of down-wind fluxes that

$$\phi_i = \sum_{j \in U(i)} v_{ij} \tau_j + \sum_{j \in D(i)} v_{ij} \tau_i - \tau_i \left(\sum_{j \in U(i)} v_{ij} + \sum_{j \in D(i)} v_{ij} \right) = - \sum_{j \in U(i)} v_{ji} \tau_j + \tau_i \sum_{j \in U(i)} v_{ji}. \quad (2.15)$$

The main change lies in how we compute volumetric partitions and rate allocations in wells. For incompressible flow, it follows from mass conservation that outflow must equal inflow and hence we can connect all cells in the grid backward to fluid sources (injection wells or inflow boundaries) and forward to fluid sinks (producers and outflow boundaries). Compressible flow has additional fluid sources because of fluid expansion induced by decaying reservoir pressure, and not all flow through a given cell can thus be attributed to an injector or inflow boundary. Some cells may even not be connected to injectors at all. For simplicity, we have chosen to lump all such effects into an additional category called “reservoir” when reporting and plotting well-pair regions and well-allocation factors.

Given this choice, it is straightforward to extend existing tools from MRST to compute and display basic flow quantities such as well-pair flux allocations, well-pair volumes, and flooded and drainage regions as time-dependent variables. This enables you to quickly screen multiple output states from a reservoir simulator to get an idea of how the flow patterns vary throughout the course of time, e.g., as illustrated in Figures 2.10 and 2.11. You can also quantify changes by comparing differences in heterogeneity measures like Lorenz coefficient and sweep efficiency factor between individual time steps.

2. WHAT IS FLOW DIAGNOSTICS?

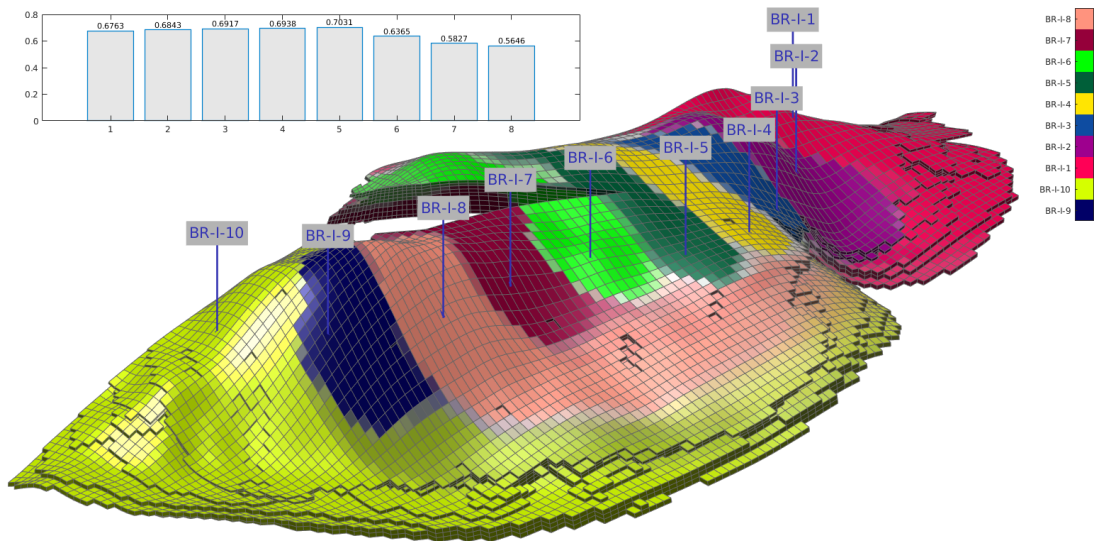


FIGURE 2.11: Time-dependent flow diagnostics for the Brugge field. The large plot shows how the sweep regions develop over a four-year period. Cells with bright colors are part of the same sweep region over the whole period, whereas grayish colors signify cells that are associated with different sweep regions over the time interval. The bar plot shows how the Lorenz coefficient for the whole reservoir develops over time.

2.7 Estimating recovery and phase rates from residence-time distributions

In this section we explain how you can use the residence-time distribution derived in Section 2.5 to provide simple estimates of the dynamics of a two-phase recovery process. To this end, we start by making a few simplifying assumptions. First, we assume that the flow field \vec{v} is driven by constant well controls and remains stationary throughout the whole displacement process. Neglecting the effects of capillary forces, compressibility, and gravity, we can then use a fractional-flow formulation to write the transport of fluid phases as a nonlinear hyperbolic equation,

$$\phi \partial_t S + \vec{v} \cdot \nabla f(S) = q, \quad (2.16)$$

where the fractional-flow function f denotes the ratio between the mobility of the wetting fluid (water) and the total mobility of both fluid phases. We now use the relation $\partial_\tau = \frac{\vec{v}}{\phi} \cdot \nabla$ to transform the 3D transport equation (2.16) to a family of 1D transport equations,

$$\partial_t S + \partial_\tau f(S) = q, \quad (2.17)$$

defined over all streamlines that connect injectors and producers. In a streamline simulator, one samples a discrete set of streamlines and solves one transport equation (2.17) for each of them. Herein, we instead assume that the fluid movement inside each influence region is not influenced by any other injectors and can be characterized by solving a *single* representative 1D equation of the form (2.17).

Let $S(\tau, t)$ denote the solution of this representative transport equation. Using (2.12), we can then write the saturation at a given point \vec{x} for a given time t as²

$$S(\vec{x}, t) = S(\tau(\vec{x}), t) = \int_0^\infty S(r, t) \delta(r - \tau(\vec{x})) dr = \int_0^\infty S(r, t) \eta(\vec{x}, r) dr. \quad (2.18)$$

In the discrete case, $\eta(\vec{x}, t)$ is replaced by a numerically computed approximation, and we can use a discrete version of the last integral to estimate cell-averaged saturations for any finite time.

The total amount of oil recovered over the time interval $[0, t]$ is given as

$$R_o(t) = \int_\Omega \phi(\vec{x}) \Delta S(\vec{x}, t) d\vec{x}, \quad (2.19)$$

where Ω denotes the domain of the displacement process and $\Delta S = S(\vec{x}, t) - S(\vec{x}, 0)$ is the change in saturation. Following [7], we use integration by parts and the fact that $\eta(\vec{x}, 0) = \lim_{t \rightarrow \infty} \eta(\vec{x}, t) = 0$ to rewrite this integral:

$$\begin{aligned} R_o(t) &\stackrel{(2.18)}{=} \int_\Omega \phi(\vec{x}) \int_0^\infty \underbrace{\Delta S(r, t)}_{dv} \underbrace{\eta(\vec{x}, r)}_u dr d\vec{x} = - \int_\Omega \phi(\vec{x}) \left[\int_0^\infty \underbrace{\left(\int_0^r \Delta S(\sigma, t) d\sigma \right)}_{v=\bar{S}(r,t)} \underbrace{\partial_r \eta(\vec{x}, r)}_{du} dr \right] d\vec{x} \\ &= - \int_\Omega \int_0^\infty \bar{S}(r, t) \phi(\vec{x}) \partial_r \eta(\vec{x}, t) dr d\vec{x} \stackrel{(2.11)}{=} \int_0^\infty \bar{S}(r, t) \left(\int_\Omega \vec{v} \cdot \nabla \eta d\vec{x} \right) dr \stackrel{(2.13)}{=} F_o \int_0^\infty \bar{S}(r, t) \mathcal{T}_o(r) dr. \end{aligned}$$

In other words, if we know the 1D transport solution $S(\tau, t)$, the initial saturation distribution $S(\tau, 0)$, and the residence-distribution $\mathcal{T}_o(\tau)$, we can compute the production for each well-pair region as follows:

$$R_o^{i,p}(t) = \int_0^\infty \left(\int_0^\tau \Delta S^i(r, t) dr \right) \mathcal{T}_o^p(\tau) d\tau. \quad (2.20)$$

We then sum the individual well-pair productions to obtain an estimate of the oil production for individual producers, the whole field, or any selected subset of producers. Because of the many approximations we have made, the resulting estimates are obviously not precise, but will usually be sufficiently indicative to, e.g., discriminate members of a model ensemble; see [22] for a more in-depth discussion.

²If we instead of solving a *single* 1D transport equation for all influence regions, solve a new transport equation for each region, (2.18) is replaced by $S(\vec{x}, t) = \sum_\ell \int_0^\infty S^\ell(r, t) \eta^\ell(\vec{x}, r) dr$, where $S^\ell(\tau, t)$ denotes the 1D flow solution and $\eta^\ell(\vec{x}, t)$ is the tracer pulse solution in region ℓ .

In the current preprocessor GUI, we utilize a reformulation of (2.20) that gives the phase rates directly. These expressions are derived from differentiating (2.20) with respect to t , such that the estimate for the oil rate becomes

$$\begin{aligned} q_o^{ip}(t) &= \frac{dR_o^{ip}(t)}{dt} = \int_0^\infty \left(\int_0^\tau \partial_t S^i(r, t) dr \right) \mathcal{T}_o^p(\tau) d\tau \\ &\stackrel{(2.17)}{=} \int_0^\infty \left(\int_0^\tau q - \partial_r f(S^i(r, t)) dr \right) \mathcal{T}_o^p(\tau) d\tau = \int_0^\infty \left(1 - f(S^i(\tau, t)) \right) \mathcal{T}_o^p(\tau) d\tau. \end{aligned} \quad (2.21)$$

In the derivation of (2.21), we have used that the source term q in (2.17) is normalized so that $\int_0^\tau q d\tau = 1$. In the two-phase scenario, the sum of water and oil rates add up to the allocation rate, and hence the expression for the water rate becomes

$$q_w^{ip}(t) = q^{ip}(t) - q_o^{ip}(t) = \int_0^\infty f(S^i(\tau, t)) \mathcal{T}_o^p(\tau) d\tau, \quad (2.22)$$

where $q^{ip}(t)$ is the total rate (allocation) for the well-pair region. We note that when approximating the integral in (2.22), the discrete saturation values and distribution values will typically be given at non-matching points, and hence interpolation is needed. To ensure consistency, rather than interpolating the distribution function directly, we interpolate using the cumulative distribution followed by discrete differentiation.

2.8 Economic measures: net present value

Net present value (NPV) is a standard measure used in budgeting and investment planning to compare the present value of all future cash inflows and outflows generated by a project. The cash flows are usually discounted, so that a given cash income in the near future is more worth than the same income at a much later time. A project is said to be profitable if the associated NPV is positive. NPV is therefore often used as an objective function to be maximized in various reservoir-engineering workflows, e.g., for optimization of well placement, injection and production rates, or to estimate the span in profitability for a model ensemble; see [6, 10] for several examples.

As an example, let us consider a two-phase waterflooding scenario. A simplified NPV over a time horizon $[0, T]$ can then be defined as follows:

$$\text{NPV}(T) = \int_0^T [r_o q_o^p - c_w^p q_w^p - c_w^i q_w^i] \left[1 + \frac{d}{100} \right]^{-t} dt. \quad (2.23)$$

Here, q_o^p , q_w^p , and q_w^i denote the oil-production, water-production, and water-injection rates, respectively, whereas r_o , c_w^p , and c_w^i are the respective revenues and costs and d is the discount in percent.

2.8.1 Using cell-averaged time-of-flight

For flow diagnostics based on cell-averaged TOF (i.e., not residence-time distributions), we do not have reasonable approximations to the dynamic production rates, so we cannot compute (2.23) directly. Instead, we introduce a simple *NPV proxy* [6, 10] that utilizes backward time-of-flight τ_b from producers to identify fluid volumes that, given a stationary flow field, will be produced within a given time horizon. That is,

$$\widehat{\text{NPV}}(T) = \sum_{c=1}^n \Phi_c [r_o f_o - c_w^p f_w]_c H(T - \tau_{b,c}) \left[1 + \frac{d}{100} \right]^{-\tau_{b,c}} - \int_0^T c_w^i q_w^i \left[1 + \frac{d}{100} \right]^{-t} dt. \quad (2.24)$$

Here, subscript c is the cell index, Φ_c is the pore volume of cell c , and H is the Heaviside function (step function, equal one for positive arguments, and zero otherwise). Since this proxy does not take into account multiphase flow behaviour of transport, it cannot be expected to approximate the actual NPV value very well. However, a good correlation with the true value was shown in [10] and, as discussed in [6], it is also possible to incorporate the cost of drilling and introduce tuning parameters to ensure better match between the proxy and the true NPV. The GUI described herein does not include such matching parameters.

2.8.2 Using residence-time distributions

For flow diagnostics based on residence-time distributions, we are able to compute approximations for individual phase rates by (2.21)-(2.22), and hence the integral (2.23) can be approximated directly. This is the current option chosen in the preprocessor.

Model ensembles used for history matching, uncertainty quantification, and optimization under uncertainty can often consist of tens or hundreds of (equiprobable) realisations. Even though flow diagnostics is quick to compute compared with a full multiphase simulation, it may still take considerable time to compute a full set of diagnostics quantities and load all the various cell-based quantities for a large set of models that each may have hundred thousands or millions of cells.

MRST offers functionalities that let you define a model ensemble and preprocess it to compute various forms of *summary diagnostics*, i.e., measures of dynamic heterogeneity or reservoir performance that constitute a single number or a single curve for each ensemble member. Once these diagnostics have been computed, you can load the result into a lightweight **ensemble GUI** that lets you (cross)plot and compare the various diagnostics for the ensemble as a whole, and possibly select a subset of representative models for further analysis.

The **diagnostics viewer GUI** is designed to perform a more detailed preprocessing and will give you access to all the flow-diagnostics quantities discussed in the previous chapter, including time-of-flight, residence time, various volumetric partitions, well allocation and communication strength, and various measures of dynamic heterogeneity. In principle, this GUI can be used to inspect a full ensemble, but for large models we generally advise that you restrict the analysis to a representative subset. Currently, the GUI assumes that all models are described on the same grid and use the same well configuration, in terms of the number (and names) of wells, well positions and trajectories, and rates of injection/production.

3.1 Inspecting a full model ensemble

We start by explaining how you can define a model ensemble, precompute various summary diagnostics, and inspect the result using the ensemble GUI. As our primary illustration, we will use the Egg ensemble [5], which is “*a synthetic reservoir model consisting of an ensemble of 101 relatively small three-dimensional realizations of a channelized oil reservoir produced under water flooding conditions with eight water injectors and four oil producers.*”

3.1.1 Setting the ensemble

Model ensembles are defined using instances of a class object called `ModelEnsemble`, which in addition to storing basic information about the ensemble, offers functionality for creating or retrieving individual ensemble members, performing computations on these, storing the results to disk, and managing previously performed computations so that their results can later be retrieved from disk. (Technically, the object relies on MRST’s *result handlers* [9, Chapter 12].)

To set up a model ensemble, all you have to do is to implement a function, which for the Egg model is called `setupEggFn`, that, given the sequence number of an ensemble number (e.g., number 17 out of 100), constructs the corresponding reservoir model object and a structure array with the well configuration. (The source code for `setupEggFn` is given at the end of this section.) With this, you can define an ensemble containing the first 50 members of the Egg model as follows:

```
mrstModule add diagnostics ad-core ad-props ad-blackoil incomp mrst-gui
ensemble = ModelEnsemble('egg', 'nMembers', 50, 'setupFn', @setupEggFn);
```

3. GETTING STARTED

Here, we load the required MRST modules and then create the ensemble. If you later want to go back and retrieve the ensemble, it is sufficient to say (provided that you have chosen a unique name, `egg` in this case):

```
ensemble = ModelEnsemble('egg');
```

This will create the necessary data structures that enable you to seamlessly retrieve the model and any computed data from disk as if it was already represented in memory. It is very important to choose a unique name to avoid overwriting existing data and creating errors when reloading previously computed diagnostics.

The `ensembleGUIForEgg` tutorial from the `diagnostics` module contains complete source code for generating the reservoir models and well structures for the Egg ensemble, computing diagnostics, and launching the ensemble GUI. Note that this example may take several minutes to run. The following is an abbreviated version of the function that constructs individual realizations of the ensemble:

```
function out = setupEggFn(n, mode)
    deck = getDeckEGG('realization', n);
    G = initEclipseGrid(deck);
    G = computeGeometry(G);
    [out.state0, out.model, out.schedule] = ...
        initEclipseProblemAD(deck, 'G', G, 'TimestepStrategy', 'none', 'useMex', true);
    out.W = out.schedule.control(1).W;
end
```

Since all ensemble members are specified on the same grid, the grid information is only stored once, whereas petrophysical data for each ensemble member are kept in separate files. We therefore use a custom-made function `getDeckEgg` from the `ad-blackoil` module to read and combine these data to a single ECLIPSE input deck, which can then be processed using standard routines from the AD-OO simulator framework.

3.1.2 Precomputing summary diagnostics

Initialising the ensemble will not by default perform any computation on the individual ensemble members. To precompute summary diagnostics, you need to call the special member function that computes flow diagnostics:

```
ensemble.computeDiagnostics()
```

This computes all the diagnostic quantities discussed in the previous chapter based on a single incompressible pressure solution. Technically, this amounts to computing the `D`, `WP`, and `RTD` data structures discussed in Chapter 13 of [9]. As mentioned already, this can be a quite comprehensive computation, and depending upon the size of your ensemble and the individual models, it may take anything from a few minutes to hours. (Still, computing the diagnostics is much faster than performing a full multiphase flow simulation over all the ensemble members.) The call to `computeDiagnostics` will check if diagnostics have already been calculated and only compute them if they do not exist already or if you specifically request that they be recomputed.

3.1.3 Quick overview of the ensemble GUI

The main purpose of the ensemble GUI is to present precomputed summary diagnostics so that you can visualize the spread in model performance and select individual models, or a representative subset of models, for more refined analysis. To this end, the ensemble GUI consists of two main parts, as shown in Figure 3.1. The plotting panel to the right presents plots or crossplots of selected summary diagnostics for all ensemble members. The menus to the left present several options for which properties to plot and are described in further detail in Chapter 4.

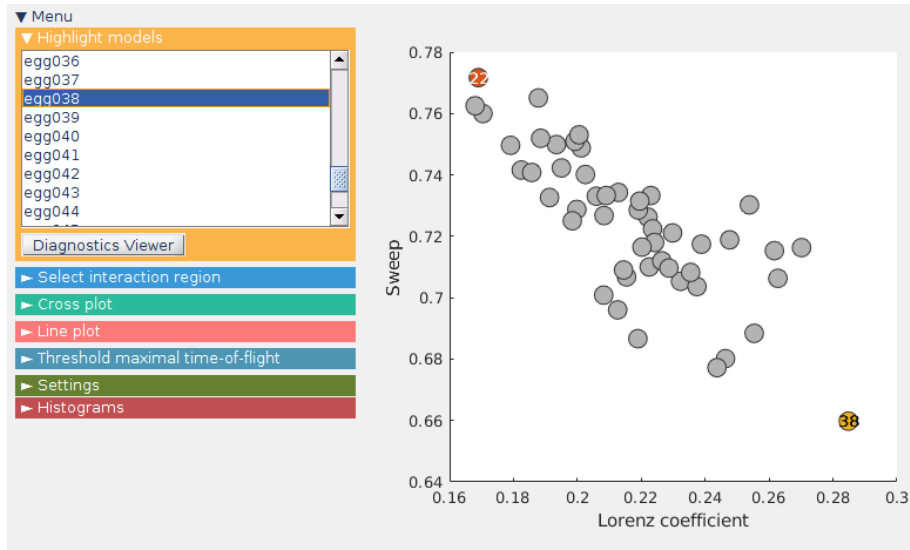


FIGURE 3.1: Overview of the ensemble GUI that presents precomputed summary diagnostics for a full ensemble.

3.1.4 Launching the GUI

Once the flow diagnostics has been computed, or loaded from a previous computation, you can launch the viewer as follows:

```
eg = EnsembleGUI(ensemble);
```

Here, `eg` is a MATLAB class object that contains all the loaded data as well as various containers, data and structures used to control the visualization and describe its current state.

3.2 Inspecting and comparing specific ensemble members

The diagnostics viewer GUI for inspecting and comparing ensemble members is designed to take a set of petrophysical realisations defined over a single grid as input, compute a single incompressible flow field for each ensemble member, based on a well configuration that is common to all ensemble models, and use the resulting flow fields to calculate flow diagnostics. This computation is performed when the viewer is loaded.

3.2.1 Quick overview of the diagnostics viewer GUI

The diagnostics viewer GUI consists of several parts (see Figure 3.2) you can use to visualize various forms of flow diagnostics interactively and compare different realisations quickly and easily.

3D Axes: This is where properties calculated for each cell of the model grid can be plotted. The location of wells in the grid can also be visualized. The top left of the plot contains the colorbar for the plotted properties. This colorbar also shows a histogram of the plotted values if applicable.

You can use standard MATLAB figure controls to zoom, pan, and rotate the plot. In newer versions of MATLAB these can be found to the top right of the plot when you hover over the figure. The figure menu also contains the following additional controls:

-  Gridline plotting on / off
-  Change the lighting
-  Change the z-aspect ratio

Subpanels: The two subpanels are used to plot different model metrics, for instance Lorenz coefficients for selected models. The layout of these plots will vary depending on what is being shown. If a

3. GETTING STARTED

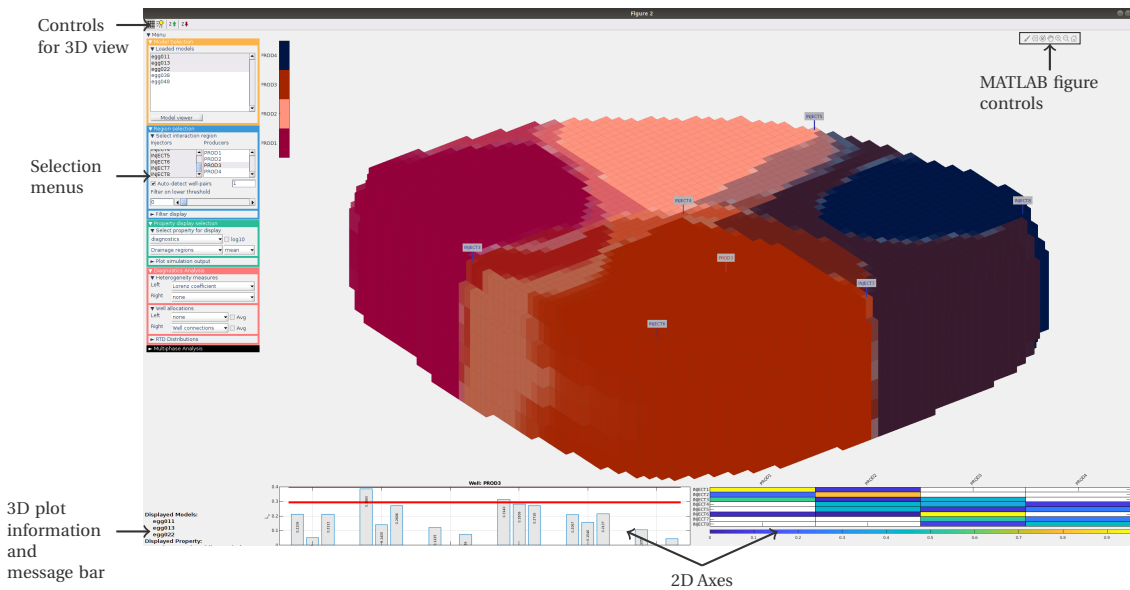


FIGURE 3.2: Overview of different parts of the GUI. The main panel can show various 3D views of cell properties from the model ensemble, with a summary of the realisations and properties selected summarized in the lower-left corner of the GUI. The left and right subpanels can be set to show various types of flow diagnostics. The GUI also contains a few controls for the 3D model display and a message bar for error messages.

plot is generated for selected model realisations / interaction regions and the selected models and regions are subsequently changed, the background of the plot will be greyed out to indicate that it is no longer up to date with the current selection. Plots can be updated by replotting the required metric with the new selection. It is also possible to remove a plot completely by selecting “none”.

The content of the main and the two subpanels can be replotted in separate figures by right clicking on the plot and selecting “Export to new figure”. Note that for this to work properly you must ensure that the zoom, pan, and rotate functions from the MATLAB figure menu are not active, i.e., that the corresponding buttons have not been selected.

Menus: The area to the left hand side of the plot areas contains the menus used for selecting what is to be displayed. These can be collapsed, expanded, and resized as desired. Menus that are greyed out require certain selections before they become active. For instance, the well allocation menu will only become active once a model has been selected. More information about the content of the menus is given in subsequent chapters.

3D plot information and message bar: The bottom left corner of the GUI contains a summary of the information plotted in the main 3D Axes. This can be useful to quickly ascertain what you are looking at without having to expand all the menus. The message bar at the bottom of the GUI is used to display error messages. For instance when a required metric cannot be displayed due to the wrong number of models / wells being selected.

3.2.2 Launching the GUI

The diagnostics viewer can either be launched from within the ensemble GUI, once you have selected a representative subset of models for more in-depth inspection, or launched directly from the command line or from within another script. Assuming we already have created an instance `ensemble` of the `ModelEnsemble` class, as discussed in the previous section, the GUI is started by the following command:

```
d = DiagnosticsViewer(ensemble, modelIx);
```

where `modelIx` contains the indices of the model realizations you wish to analyze.

Constructing simulation models for individual ensemble members will in many cases be a costly operation that quite often may have been performed already before calling the GUI. The viewer therefore also accepts cell arrays of MRST simulation models and corresponding well structures as input. These can either be generated using custom-made MRST scripts or converted from ECLIPSE input with existing MRST functionality for reading in ECLIPSE grids. If we have an ensemble class, we can extract the two necessary MATLAB cell arrays, `models` and `wells`, containing reservoir model objects (see [9, Chapter 12]) and well structures (see [9, Chapter 5]) by a simple loop:

```
[models, wells] = deal(cell(1,numel(modelIx)));
for k = 1:numel(modelIx)
    tmp      = ensemble.setupFn(modelIx(k));
    models{k} = tmp.model;
    wells{k}  = tmp.W;
    if isfield(tmp, 'state0')
        states0{k} = tmp.state0;
    end
end
```

and then launch the GUI using any of the following two calls:

```
d = DiagnosticsViewer(models, wells);
d = DiagnosticsViewer(models, wells, 'state0', states0);
```

In the first alternative, no initial state is prescribed for the ensemble members, and the GUI will assume that the reservoir exists at a uniform pressure of 200 bar and is completely filled by hydrocarbons in an oleic phase. Effectively, the diagnostics will then be computed using a single-phase, incompressible flow model with fluid properties sampled from the oleic phase, linearized around the 200-bar pressure point (see the function `convertToIncompFluid` for details).

If your ensemble specifies initial states, contains a fluid model that is not consistent with having an oleic phase, or if you wish to compute diagnostics based on a multiphase pressure solution with another initial state, you must make sure to specify the initial state explicitly, as in the second alternative. Here, `states0` is a cell array of standard MRST state structures that specify the initial state of each ensemble realization. Also in this case, we use incompressible flow solutions to compute diagnostics. The corresponding fluid models are made by linearizing the fluid models represented in `models` around the specific initial states in `state0`. Using the initial state specified by the ensemble model is the default behavior when the viewer is called from `EnsembleGUI`.

If initial states are specified, additional multiphase analysis can be carried out on top of the basic diagnostics to give breakthrough times for each phase at each producer and the well allocations of each phase (see Section 6.4 for more details). Note that since the flow simulations carried out only compute a single incompressible pressure step for each model, they only give an instant (or stationary) picture of the fluid flow and do not take into account how transport of fluid phases (or component concentrations) may affect pressure and fluid communication dynamically.

If desired, it is possible to pass a `'modelNames'` option as input together with an accompanying cell array of strings that specify names of each model realization in the GUI. Otherwise, a default naming convention will be used, so that if you have, for instance, ten model realizations, these will be named “m1” to “m10”.

For a complete example showing how to load the Egg model into the `DiagnosticsViewer` see the example `preProcessDiagnosticsEgg` in the `diagnostics` module of MRST.

This chapter gives a more detailed introduction to the lightweight **ensemble GUI** with which you can (cross)plot and compare various summary diagnostics for a whole ensemble of model realizations.

4.1 Cross plots of single-valued quantities

When first launched, the ensemble GUI presents a cross plot of sweep efficiency along the y -axis and the Lorenz coefficient along the x -axis. These quantities are computed over a selected time horizon, which by default corresponds to dimensionless time 1 PVI, and for a given domain of definition, which by default consists of the whole reservoir. Section 4.4 explains how you can change the time horizon and the domain of definition.

FIGURE 4.1: Menu for setting up cross plots of single-valued diagnostic quantities.

The cross-plot menu shown in Figure 4.1 (third menu from the top in Figure 3.1) lets you select quantities to plot along each of the two axes:

- Model number – a linear index running from one to the total number of ensemble members.
- Lorenz coefficient – measure of dynamic heterogeneity (see Section 2.4); the values are between zero and one, which corresponds to the case all flow paths having the exact same residence time and residence times spanning the positive interval $(0, \infty)$, respectively.
- Sweep efficiency – ratio between the volume contacted by displacing fluid at time t and the volume contacted at time $t = \infty$.
- Recovery factor – fraction of the in-place fluid that has been recovered by time t . The recovery is computed by solving a 1D Buckley–Leverett problem,

$$\partial_t S + \partial_\tau f(S) = 0, \quad S(\tau, 0) = 0, \quad S(0, t) = 1, \quad (4.1)$$

whose self-solution $S(\tau, t) = S(\tau/t)$ is then mapped onto residence times in the influence region of each injector to compute the recovery factors R_o , as explained in Section 2.7. Section 4.5 explains how to set the fluid properties used to define the fractional flow function

$$f(S) = \frac{k_{rw}(S)}{k_{rw}(S) + \frac{\mu_w}{\mu_o} k_{ro}(S)}. \quad (4.2)$$

- Net present value (NPV) per produced volume – simple estimate of the project profitability over a given production horizon $[0, t]$, as computed by the proxy (2.24) explained in Section 2.8.
- Maximum NPV – the maximal NPV over all possible choices of production horizons.

By default, the data points are circular markers of uniform size, but these can be scaled by allocation or volume using the using the drop-down menu labelled “Resize by”. Likewise, ticking the check box fit line will fit a straight line through all data points using a standard least-squares estimate. Right-clicking inside the plot axis enables you to export the plot to a new figure window.

Example:

Figure 4.2 shows four examples of cross plots of 50 realizations of the Egg model [5]. In the upper-left plot, we have marked two outliers, ② and ③, that are likely to represent the best and worst reservoir performance (high sweep/low Lorenz and low sweep/high Lorenz). We also highlight three other intermediate realizations, ①, ④, and ⑤. The lower plots demonstrate excellent correlation between recovery factor and sweep efficiency and between NPV per volume and sweep efficiency.

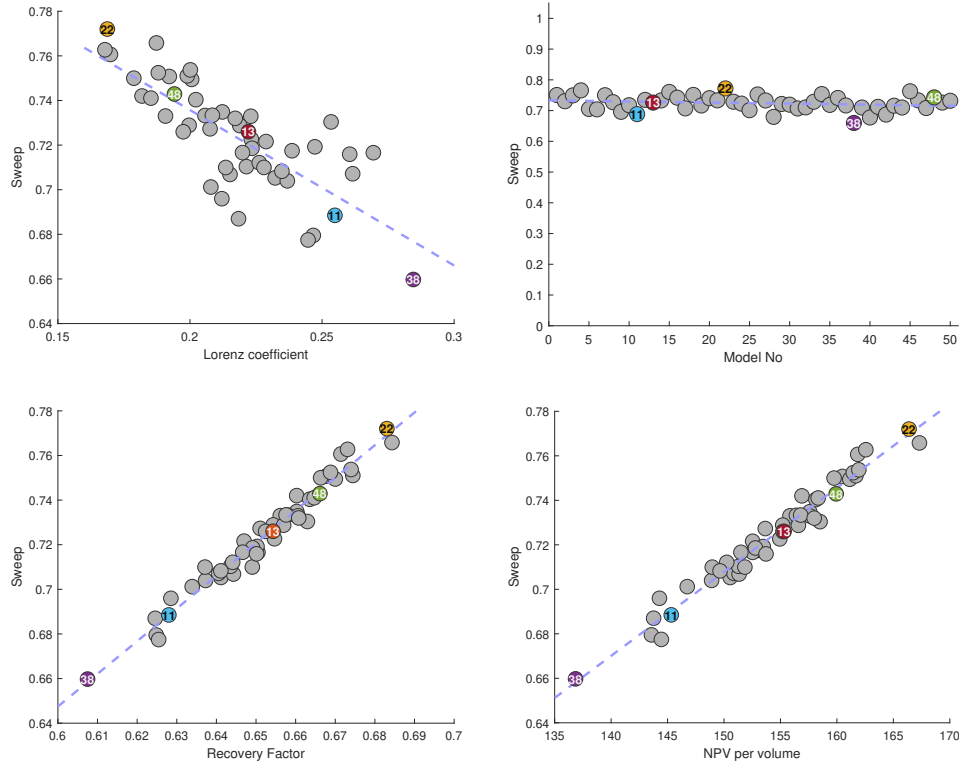


FIGURE 4.2: Four examples of cross plots of diagnostics quantities for 50 realizations from the Egg model [5].

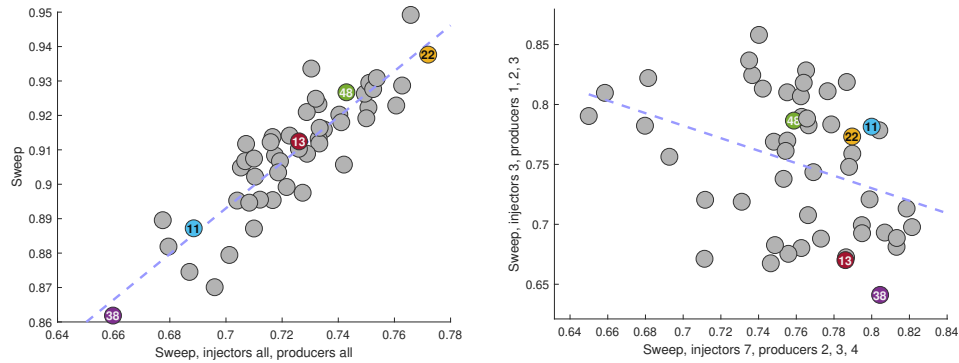


FIGURE 4.3: Two examples of cross plots with “frozen” x -axis that let you compare quantities for different time horizons and domain of definitions. The left plot shows a cross plot of sweep efficiency at 1 PVI (x -axis) and 2 PVI (y -axis). The right plot shows a cross plot of sweep efficiency for injector number seven (x -axis) and injector number three (y -axis).

The cross plots are updated automatically once you change the time horizon or the domain of definition for the diagnostic quantities. To enable comparison of quantities defined over different domains or time horizons, both axes are equipped with a check box freeze that enables you to “freeze” the corresponding data so that they are not updated when you make changes in one of the other menus. Figure 4.3 shows two examples created by first selecting the desired property on the x -axis, freezing this axis, and then updating the time horizon or domain of definition for the y -axis. In the right plot, you may notice that realization 22, which had the best sweep efficiency for the field as a whole, does not represent the best sweep when looking at individual injector regions.

4.2 Diagnostics and production curves

With the “Line plot” menu (Figure 4.4), placed below the “Cross plot” menu, you can plot diagnostics curves such as the F - Φ plot, sweep efficiency as function of dimensionless time (PVI), and residence-time distributions. Using the approximations outlined in Section 2.7, you can also compute simple estimates of production responses such as oil rate, water rate, water cut, and recovery factor, as well as the NPV proxy from Section 2.8, all plotted as functions of pore volumes injected (dimensionless time).

All plots are by default presented for the field as a whole and over a time horizon up to dimensionless time 1 PVI, but these settings are easy to adjust, as will be explained in Section 4.4.

Example:

Figure 4.5 shows six examples of line plots for the Egg model, and with the same ensemble members highlighted as in Figures 4.2 and 4.3. Starting with the F - Φ plot, we see that ② overall gives the least heterogeneous displacement, since it has the least concave curve, whereas ⑧ is most concave and thus most dynamically heterogeneous. This can be further explained by looking at the RTD. Comparing ② and ⑧, we see that ⑧ on one hand has some flow paths that are significantly shorter than ② and thus experiences much earlier breakthrough. On the other hand, ⑧ also has a heavier tail than ② up to 10 PVI (as seen in the inset), and thus has a larger fraction of relatively long flow paths. Not surprising, ⑧ has the lowest NPV and worst sweep efficiency, whereas ② has highest NPV and best sweep. The oil and water rates show similar trends.

It is interesting to note how well the five highlighted ensemble members span the spread in the various model responses. Since they are picked relatively equidistantly close to the fitted line of the Lorenz–sweep plot in Figure 4.2, they consistently preserve the same ranking. However, by picking members close to the fitted line, we also miss some interesting behavior. Members well below the line, i.e., with low Lorenz coefficient and low sweep efficiency, are characterized by a relatively narrow RTD peak like ②, but have a larger fraction of fast flow paths. Members well above the line, i.e., with high Lorenz coefficient and high sweep efficiency, are characterized by very early breakthrough and multimodal RTDs.

Last, we note that for this particular model, production responses and NPV seem to correlate better with sweep efficiency than with the Lorenz coefficient. Similar behavior is observed in [22].

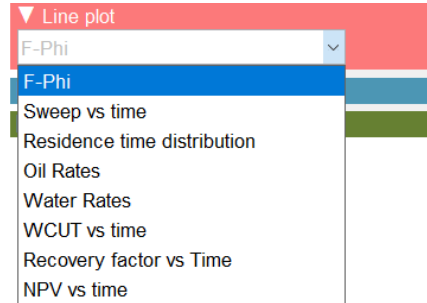


FIGURE 4.4: Menu for setting up line plots of diagnostic curves measuring dynamic heterogeneity and reservoir performance.

4.3 Highlighting and selecting individual models

The topmost menu shows a list of the model realizations that have been loaded into the GUI. You can select realizations to be highlighted using the left mouse button and multiple realizations by holding down **Ctrl** or **Shift** while clicking the left mouse button. The realizations you mark will then be highlighted with a unique color in any line plot, or by a unique color and a numbered label like ⑧ in any cross plot. Vice versa, if you click on any cross-plot marker or on any diagnostic curve, the corresponding model name will be highlighted in the list. Inside the list, you can also use the right mouse button to bring up a pop-up menu that enables you to select all realizations or clear the current selection.

Let us now assume that you have selected a subset of all the available model realizations, e.g., as illustrated in Figure 4.2. By clicking the **Diagnostics Viewer** button, the corresponding models will be loaded into the 3D diagnostics viewer, which we will discuss in more detail in the next chapter.

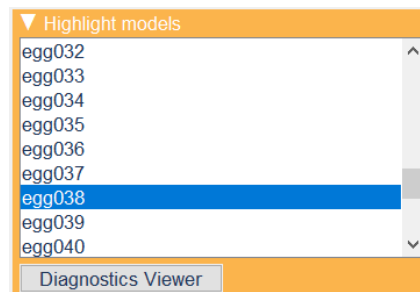


FIGURE 4.6: This menu enables you to select which models to highlight and launch the diagnostics viewer GUI.

4. FLOW DIAGNOSTICS FOR A FULL ENSEMBLE

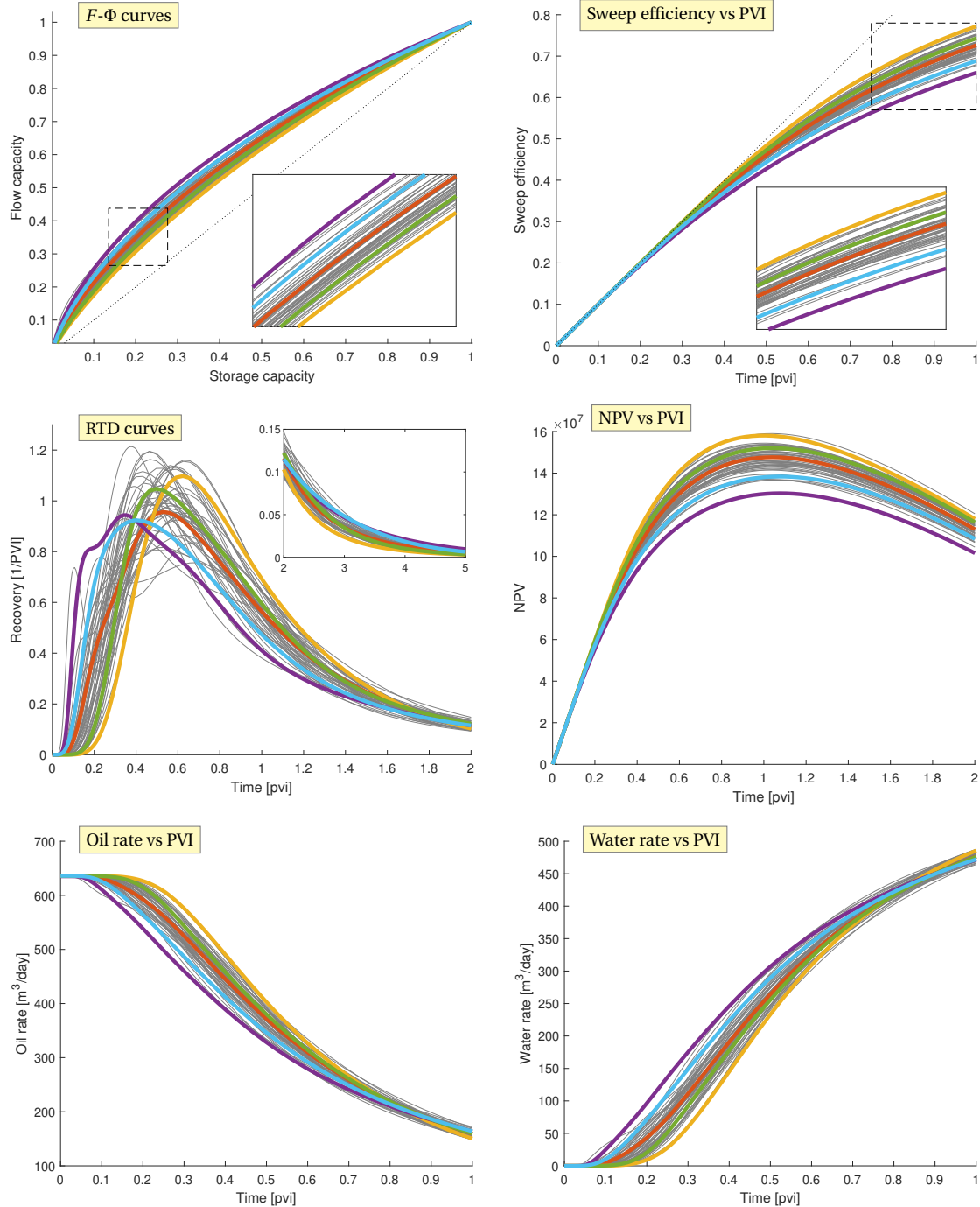


FIGURE 4.5: Examples of various line plots for the Egg model. In all plots, ensemble members 11, 13, 22, 38, and 48 are highlighted with thicker lines and colors blue, red, yellow, purple, and green, respectively. Notice that the $F-\Phi$ curves, the sweep efficiency, and RTD plots have been slightly annotated to better distinguish the different ensemble members. As such, they do not appear strictly as produced by the GUI.

4.4 Setting time horizon and domain of definition

Most quantities shown by the GUI are computed by simulating tracer pulses from individual wells (see Section 2.5), and all field-wide results are computed by summing up contributions from individual influence regions. As a result, it is easy to restrict the time horizon or the spatial domain over which the various flow diagnostic quantities are defined. The upper menu shown in Figure 4.7 lets you select individual wells or pair of wells and their corresponding influence/interaction regions:

- Two boxes list the injectors and producers. In each box, you can select individual wells or sets of wells. This will restrict the computed diagnostics to the corresponding influence regions.
- By ticking the check box, you enable automatic selection of well pairs. That is, if you select one or more injectors, the software will automatically add all connected producers to your selection. Selecting one or more producers works the same way by adding all connected injectors. Whether an injector and a producer are connected is determined from the connection matrix A defined in (6.2) in Section 6.2. In the associated input area, you can set a lower threshold $c \in [0, 100]$ on the connection strength required for wells to be auto-selected, so that injector number i is said to be connected to producer p if and only if

$$A_{ip} \geq \frac{c}{100 \max(m, n)} \sum_{j=1}^n \sum_{k=1}^m A_{jk}. \quad (4.3)$$

Likewise, using the slide bar or the input area in the lower menu shown in Figure 4.7 you can set the length of the time horizon used to compute various quantities like sweep efficiency, net present value, etc. By default, this time is set to be one pore volume injected (1 PVI). By selecting the radio button labelled “years”, you can specify the time horizon in physical time instead.

4.5 Setting fluid and economic properties

To compute an estimate of oil recovery, the GUI uses Corey type relative permeability relationships of the form

$$k_{rw}(\hat{S}) = \hat{S}^{n_w}, \quad k_{ro}(\hat{S}) = (1 - \hat{S})^{n_o} \quad (4.4)$$

where the effective saturation \hat{S} is given by

$$\hat{S} = \frac{S - S_{cw}}{1 - S_{co} - S_{cw}}. \quad (4.5)$$

The upper part of the menu shown in Figure 4.8 lets you specify the Corey exponent n_α , the critical saturation $S_{c\alpha}$, and the fluid viscosity μ_α for each phase $\alpha = w, o$. However, the actual flow solution is not recomputed before you click the button.

The lower part of the menu contains input fields for specifying oil revenue r_o , water injection and production costs (c_w^i and c_w^p), and yearly discount rate d , which all are used to estimate net present value by (2.24).

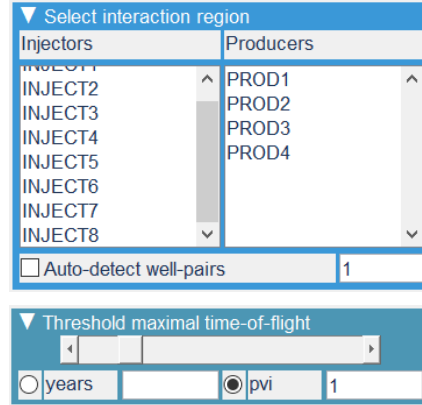


FIGURE 4.7: The upper menu lets you restrict computation of flow diagnostics to the influence region of individual wells or groups of wells. The lower menu lets you set the time horizon over which the various measures are computed.

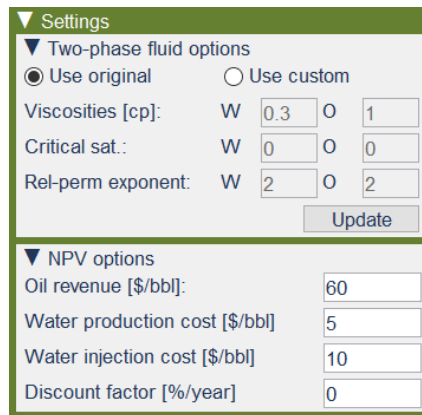


FIGURE 4.8: Menu for setting fluid and economic parameters to calculate economic objectives and estimates of reservoir performance.

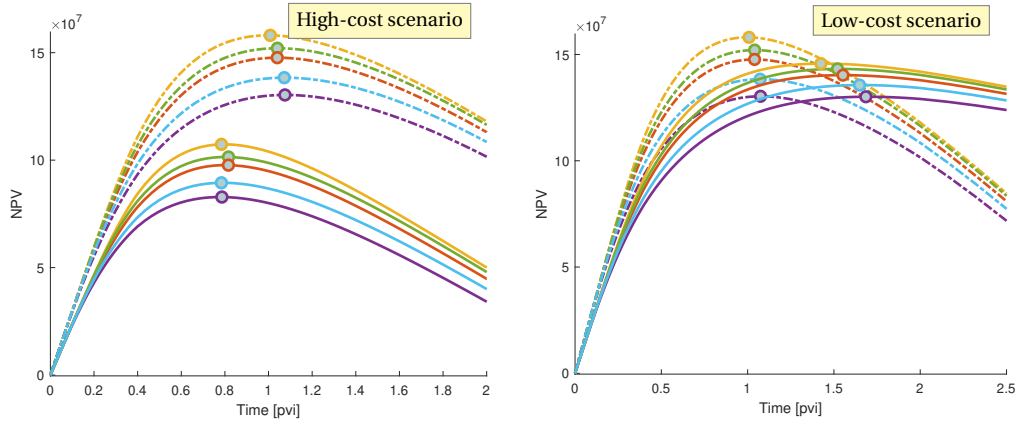


FIGURE 4.9: NPV for five Egg models. Dashed lines represent the default model, whereas solid lines represent the high or low cost models. Markers represent the maximum NPV for each of the different scenarios.

Example:

In the last example, we briefly illustrate how changes in the economic parameters affect the NPV. Specifically, we consider three sets of parameters,

	r_o [USD/bbl]	c_o^i [USD/bbl]	c_o^e [USD/bbl]	d [%]
default	60	5	10	0
high cost	50	10	10	5
low cost	45	3	3	3

Figure 4.9 reports the corresponding NPV curves for the five model realizations we have highlighted in the examples earlier in this chapter. As expected, the NPV decreases significantly in the high-cost scenario with increased discount rate and cost for water production and injection. The maximum NPV also occurs after less water has been injected, and if we were to close down the reservoir when the NPV peaks, we would leave more oil in the ground in modified scenario. The NPV also decreases in the low-cost scenario, but significantly less than in the high-cost scenario, and here we see that oil production can be sustained for a much longer period, leading to an overall higher recovery factor.

The setup in the previous example was chosen quite arbitrarily for illustration purposes only. Møyner et al. [10] describe a much more realistic case where a similar NPV proxy was used to optimize production and NPV for the Norne field model. In this setting, flow diagnostics was used to derive plausible and optimized rate targets, that were subsequently fed to a commercial reservoir simulator to ensure that the could be maintained within multiphase well controls. Likewise, Krogstad and Nilsen [6] use a more comprehensive NPV proxy for well-path optimization.

4.6 Histogram for static parameters

With the “Histograms” menu shown in Figure 4.10, you can compute and plot histograms for static variables (porosity and permeability) for all ensemble members. The histograms are calculated by default over the range of values found in the first ensemble member using 80 bins. The menu lets you modify the histograms by changing the maximum, minimum, and the number of bins. Additionally, you can tick the checkbox log10 to show a logarithmic scale in the horizontal axes. This menu has the highlighting functionality described in Section 4.3, so that you can select and highlight one or multiple ensemble members as shown in Figure 4.11.

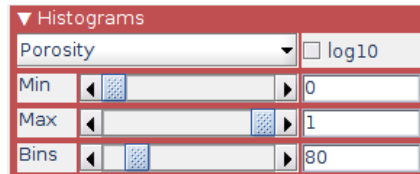


FIGURE 4.10: Menu for setting histogram parameters for all ensemble models.

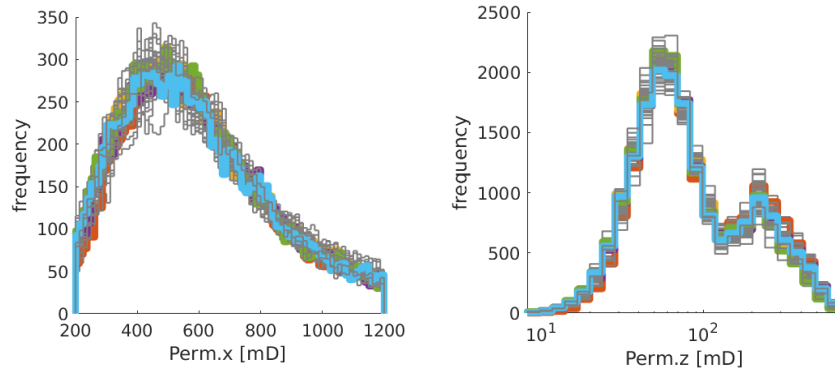


FIGURE 4.11: Histograms for all the Egg model with realizations 11, 13, 22, 38, and 48 highlighted.

Example:

Figure 4.11 shows histograms of the permeability for different realizations of the Egg model. The left plot shows the horizontal permeability (in the x -direction) with minimum value set to 200 mD, maximum value 1200 mD, and 80 bins in the histogram. The right plot shows a histogram for the vertical permeability, with the log10-checkbox selected to create a histogram of the logarithmic values. The minimum and maximum values are selected by default as histogram limits, whereas the use of 25 bins must be specified manually using the sidebar or the input are in Figure 4.10.

This chapter discusses how you can select different model realizations and display specific cell-based properties in the 3D panel of the diagnostics viewer GUI shown in Figure 3.2.

5.1 Selecting model realizations

The topmost menu shows a list of the model realizations that have been loaded into the GUI. You can select realizations using the left mouse button and multiple realizations by holding down **Ctrl** or **Shift** while clicking the left mouse button. You can also use the right mouse button to bring up a pop-up menu that lets you select all realizations or clear the current selection.

Each time you select a combination of realizations, the contents of the main panel are updated immediately. This means that if you have selected to display one of the dynamic or diagnostics properties (see Section 5.3) you can use the **↑** or **↓** buttons to quickly see how these properties change with each realization.

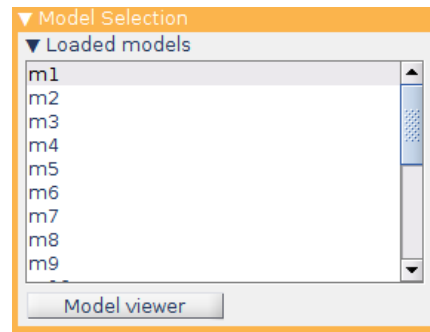


FIGURE 5.1: Menu for selecting model realizations to display in the GUI.

5.2 Comparing several models in 3D

If multiple realizations are selected, you can compare them in 3D by clicking on the “Model viewer” button at the bottom of the model selection menu (Figure 5.1). This will launch a figure window with plots of the selected realizations side by side for easier comparison (see Figure 5.2). The axes are all linked, so if you change the view in one of them, the others will be updated automatically. However, the plots are not updated if you change the display in the main window of the diagnostics viewer.

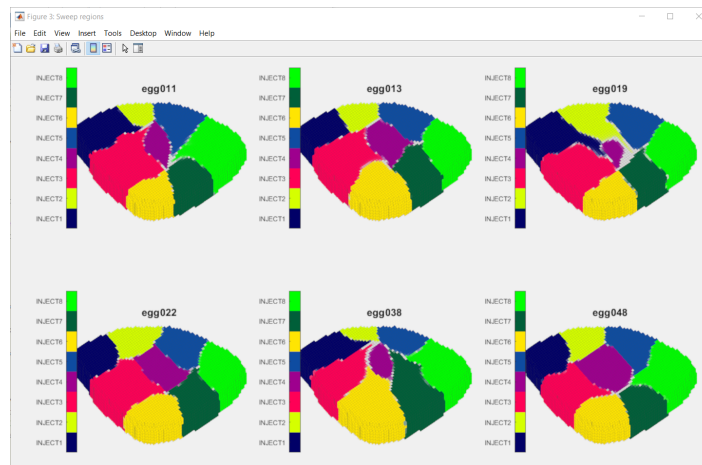


FIGURE 5.2: Sweep regions for six different realizations from the Egg ensemble; viewer launched by clicking on the “Model viewer” button shown in Figure 5.1.

5.3 Displaying cell properties

The main panel takes up the largest portion of the GUI and can display a variety of cell properties on the full model or on a subregion selected as explained in Section 5.5.1. The quantities available for plotting depend on the data available in the output file and how many models you have selected. You select which property you wish to display using the “Property display selection” menu shown in Figure 5.3.

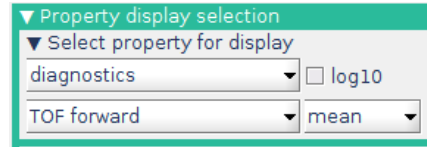


FIGURE 5.3: Menu for selecting the property displayed on the 3D grid.

- The first drop-down selects what type of property to display: static, dynamic, diagnostics, or computed internally in this GUI.
- The second drop-down selects a specific property within the selected type.
- The third drop-down determines whether to plot the mean value, standard deviation, or maximum difference over all selected realizations if you have selected multiple realizations.
- The check box switches between displaying the property using linear or logarithmic colormap.

You can find information about the current plot in the “3D Plot Information” area at the bottom left of the GUI (Figure 3.2). If no realizations are selected, you can only display static and dynamic properties, which by default are sampled from the first model in the list.

5.3.1 Static properties

Static properties are geological properties of the model (name convention from ECLIPSE input):

PORO	– porosity
PERMX	– permeability in the x direction
PERMY	– permeability in the y direction
PERMZ	– permeability in the z direction
DEPTH	– depth to cell centroid
PORV	– pore volume of a cell (cell volume multiplied by porosity and net-to-gross)

5.3.2 Dynamic properties

Dynamic properties are primary unknowns calculated during simulations such as pressure and phase saturations. Currently, only a single phase simulation is run so the only dynamic property which is model specific is pressure. However, the initial phase saturations, used for the multiphase analysis (Section 6.4) can also be displayed. Only phases present in the model are available for plotting.

PRESSURE	– reservoir pressure
SOIL	– oil saturation
SWAT	– water saturation
SGAS	– gas saturation

5.3.3 Diagnostics properties

This category consists of the flow diagnostics properties calculated within the GUI.

Forward TOF	– time-of-flight from injectors (2.1)
Backward TOF	– time-of-flight to producers (2.1)
Residence time	– the sum of forward and backward time-of-flight
Tracer forward	– C-values in injector influence regions (2.2)
Tracer backward	– C-values in producer influence regions

Tracer product – the product of injector and producer influence coefficients
 Sweep region – region swept by a specific producer (majority vote over all C^I in each cell)
 Drainage region – region drained by a specific producer (majority vote over all C^P in each cell)
 First arrival forward – time-of-flight for shortest forward flow path crossing cell
 First arrival backward – time-of-flight for shortest backward flow path crossing cell

The last category, COMPUTED, refers to cell properties computed inside the GUI. No such properties are computed in the current version, but functionality may be added later.

5.4 Simulation output (well solutions)

Well responses from the incompressible flow solver can be plotted using the simulation output menu. The list box to the left of the menu (see Figure 5.4) lets you choose wells for which you want to plot responses. You can limit the list of wells to the current active region (set by the interaction-region and property-filter menus) by ticking the check box at the bottom of the menu. Once you have selected one or more entries from the list, the right list box is filled with the properties available for plotting. The output fields available for plotting are the fields returned by the `state.wellSol` structure from `incompTPFA.m`. Currently, these are total flux, pressure, and pressure drop along the wellbore. Selecting multiple models from the “Model Selection” menu will allow comparison of well responses from different realizations side by side.

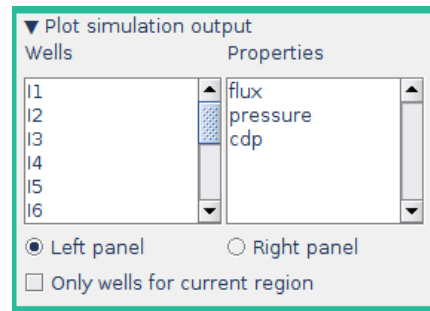


FIGURE 5.4: Menu for selecting simulation output to be displayed in the 2D panels.

5.5 Selecting a subset of the reservoir

Volumetric connections can be quite complex and difficult to understand in large reservoirs with many wells. To get a better understanding, you may therefore want to only view parts the reservoir and possibly analyze the fluid communication within individual drainage or flooding regions, or between pairs of injectors and producers. The GUI offers two different methods you can use to restrict your view to a subset of the reservoir. The first is a standard property filter that only affects the 3D display of the reservoir. The second lets you restrict both the 3D view and the computation of flow diagnostics shown in the 2D panels to a region of the reservoir.

5.5.1 Select wells and interaction regions

A primary purpose of the diagnostics GUI is to analyze the connections and communication between individual wells in the reservoir. The upper part of the region-select menu shown in Figure 5.5 lets you select individual wells or pair of wells and their corresponding influence/interaction regions. The menu contains three parts:

- Two list boxes giving the injectors and producers. With each of these, you can select individual wells or sets of wells. This will restrict the 3D plot of cell values discussed in Section 5.3 to the corresponding influence regions. Likewise, any type of flow diagnostics displayed will be restricted to these wells.

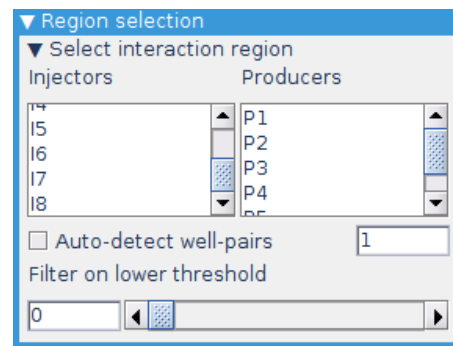


FIGURE 5.5: Menu for selecting wells and the subset of the reservoir used when presenting flow diagnostics and tracer analysis.

- A check box for enabling automatic selection of well pairs. If you select one or more injectors when auto-selecting is enabled, the software will automatically add all connected producers to your selection. Selecting one or more producers works the same way by adding all connected injectors. In the associated input area, you can set a lower threshold $c \in [0, 100]$ on the connection strength required for wells to be auto-selected; see Section 4.4 for more details.
- An input box and a slider to select a lower threshold ε that determines how much of the active influence/well-pair region is shown in the 3D plot panel. If you have selected injectors and producers, the displayed region is given as all cells in which $C_i^I C_p^P > \varepsilon$, where C_j^I and C_j^P denote values of injector/producer influence region associated with well j and (i, p) vary over all the selected injectors and producers. If no producers are selected, p varies over all producers and likewise i varies over all injectors if no specific injector is selected.

5.5.2 Property filter

The property filter lets you filter the cells displayed in the 3D plot by the value of some specific parameter. The property used for filtering does not have to be the same as the property displayed in the plot. Notice that this selection only affects the 3D display and not the computation of flow diagnostics.

Once the property filter has been enabled, you can select the property to use for the filtering and the minimum and maximum value of the cells to be displayed. Selecting `log10` changes the sliders for min and max to a logarithmic scale. If applicable (i.e., if multiple models are selected) you can use the drop-down menu `n/a` to choose whether to filter on the mean, standard deviation, or maximum difference values.

Some properties, like time-of-flight and residence times, can in principle have values that span from zero to infinity, which makes visualization somewhat challenging. In the GUI, we cap all computed time-of-flights by an upper value defined by the `maxTOF` field in the `DiagnosticsViewer` class (see Appendix B); the default is 500 years. This means that time-of-flight and residence times have already been filtered once when they appear in the property filter.

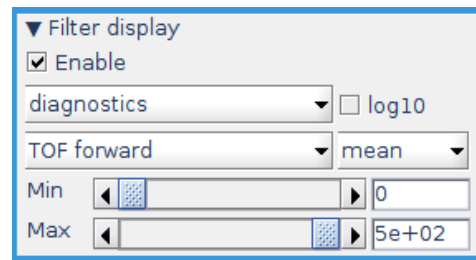


FIGURE 5.6: Property filter used to restrict the region displayed in the 3D panel.

Section 5.3 explains how you can display the basic cell-based flow-dagnostic quantities like time-of-flight, average residence time for all flow paths through a cell, stationary tracer concentrations, and derived flooding/drainage volumes in the 3D plotting panel. This chapter discusses all the other types of *derived* diagnostic quantities you can display in the 2D plotting panels, namely: heterogeneity measures, volumetric partitions and well allocations, residence time-distributions and multiphase analysis derived from flow diagnostics.

6.1 Heterogeneity measures

Section 2.4 explained how flow diagnostics uses residence time to reinterpret heterogeneity measures from classic sweep theory in order to characterize the *dynamic heterogeneity* of flow paths and their connection structure. The first entry of the “diagnostics” menu, shown in Figure 6.1, consists of two pull-down menus that each let you choose between three standard measures of dynamic heterogeneity to plot in the lower-left or lower-right panel: **F-Phi diagram**, **Lorenz coefficient**, **fractional recovery**, and **sweep efficiency**. You can also choose “none” to clean the corresponding 2D panel. See Figure 6.2 for examples of possible plots.

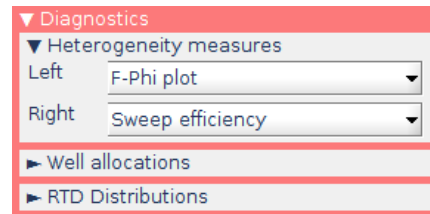


FIGURE 6.1: Menu for selecting the type of heterogeneity measures to be shown in the left and right 2D plotting panels.

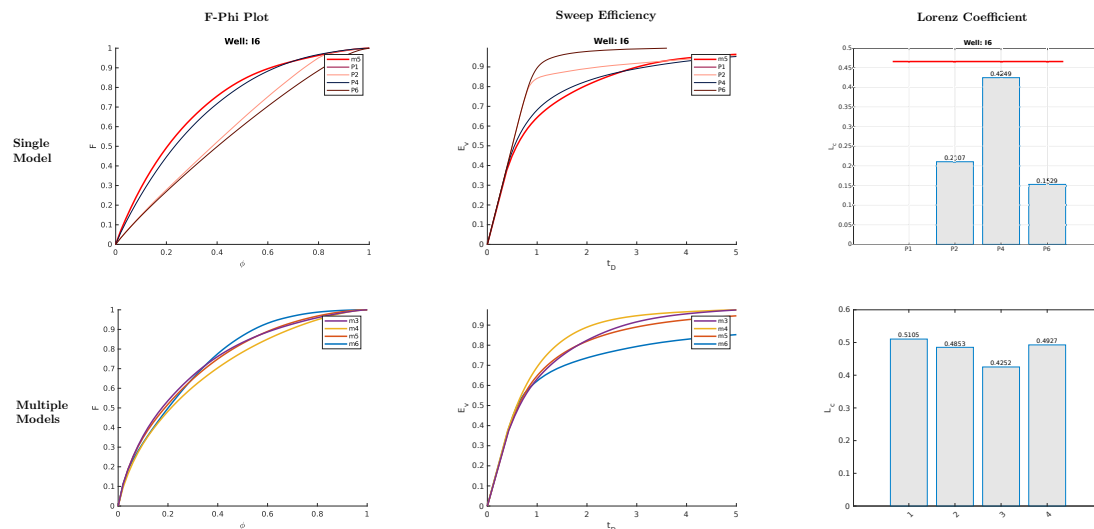


FIGURE 6.2: Plots of various heterogeneity measures for single and multiple models. For plots in the top row the bold red line indicates the value of the specific parameter for the whole model (not just the producer - injector well pair region).

6.2 Volumetric partitions and well allocation

The instantaneous influence regions naturally define volumetric partitions of the reservoir in the form of drainage, sweep, and well-pair regions. From the intersection of drainage and sweep regions, you can determine whether an injector has fluid communication with a producer and compute the corresponding flow volume. Likewise, you can measure well-allocation factors that describe how the injected flow volume of an injector will distribute to different producers, or how the produced volume from a producer can be attributed to push from different injectors.

The menu shown in Figure 6.3 consists of two drop-down menus, one for each of the 2D plotting panels. These enable you to report various types of volumetric partitions and well-allocation factors. Volumetric partitions are usually associated with a single model realization, but can also be reported in an averaged sense over multiple realizations by activating the Avg check box. Well allocations can only be reported for a single producer or single injector at a time. If the requirements for a specific plot are not met, the GUI issues a warning. To produce a plot you must then go back to the selector for wells and interaction regions (Figure 5.5) and update your well selection.

To explain the various quantities in this section, we start by defining some notation. Let C_i^I and C_p^P denote the stationary tracer concentration associated with injector i and producer p , respectively. Let c_k be a cell in the grid with associated porosity ϕ_i and bulk volume V_i . When a cell quantity like the volumetric flow rate q is evaluated in cell c_i , we write $q[c_i]$, like in Section 2.3.

Injector/producer volumes: The influence region of injector i is an instantaneous quantity defined as the portion of the whole reservoir in which $C_i^I > 0$. The total volume of this region is thus given as $V_i^I = \sum_k C_i^I[c_k] \phi_k V_k$. The menu choice “Injector volumes” subdivides this volume into flow volumes of injector–producer pairs, defined as follows

$$V_{i,p}^{IP} = \sum_k C_i^I[c_k] C_p^P[c_k] \phi_k V_k.$$

This is presented as a pie chart for all well pairs associated with a single injector or producer. Similar pie charts can report averaged volumes over multiple model realizations (see first column in Figure 6.4).

Injector/producer allocation: To understand the volumetric connection, it is often of interest to understand the fraction of the producer inflow that can be attributed to each of the connected injectors. If c_k^p denotes perforated cell number k of producer p , the volumetric outflow $q[c_k^p]$ can be allocated to injector i by multiplying with the injector tracer concentrations C_i^I , that is,

$$a_{i,p}^k = C_i^I[c_k^p] q[c_k^p]. \quad (6.1)$$

The menu choice “Injector allocation” reports the sum over all perforations $\sum_k a_{i,p}^k$ for all injectors connected to the selected producer. The result is presented as a pie chart if you have selected a single model realization and as multiple bar charts if you have selected multiple realizations. Allocation factors are defined analogously for all producers connected to a single injector. This reports how the injected volumes distribute their “push” to different producers (see second column in Figure 6.4),

Injector/producer profile: This menu choice gives cumulative sums of the injector allocation factors $a_{i,p}^k$ or producer allocation factors $a_{p,i}^k$, from toe to heel or bottom to top perforation (see third column in Figure 6.4). This can be useful for viewing how fluid might preferentially move through different layers in the reservoir for different model realizations.

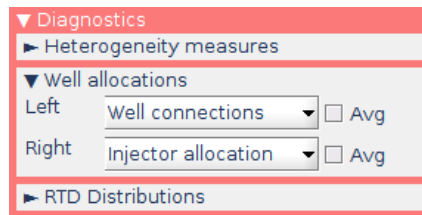


FIGURE 6.3: Menu for selecting the type of well-allocation diagnostics to be shown in the left/right 2D plotting panels.

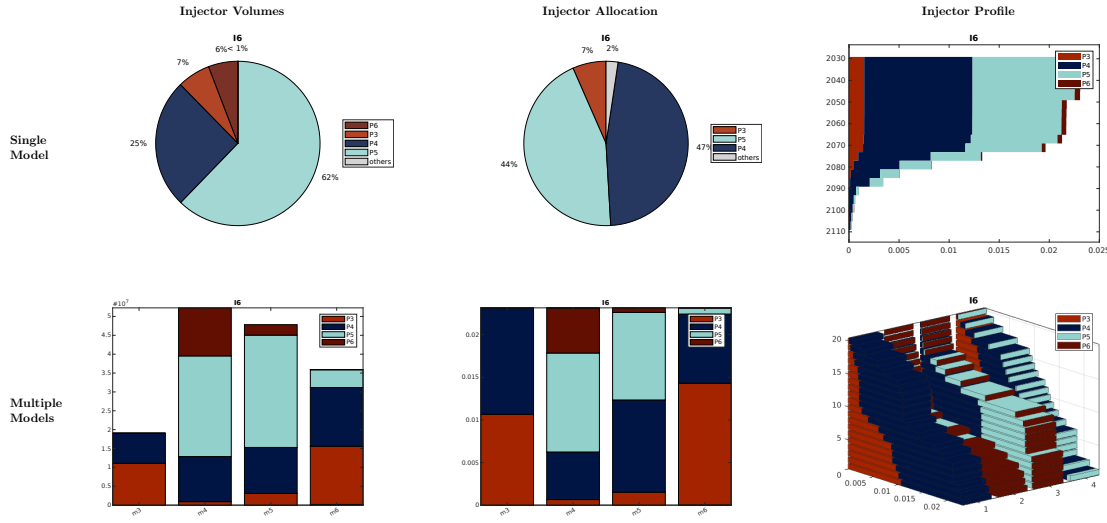


FIGURE 6.4: Plots of various allocation measures for single and multiple models.

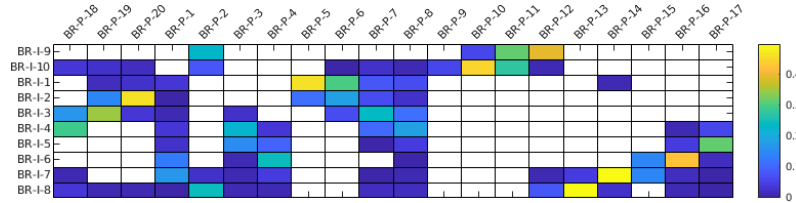


FIGURE 6.5: Plot of the well-connection strength. White color means no connection.

Well connection: The well connection matrix shows the communication strength (i.e., the total flux allocation) between each injector and each producer (Figure 6.5). If multiple models are selected, the value of the communication strength is the communication strength for whichever model has the maximum communication strength for that well pair. The communication strength is defined as the sum of the injector well-allocation factors (6.1) for all perforations of each well:

$$A_{ip} = \sum_k C_p^P [c_k^i] |q[c_k^i]|. \quad (6.2)$$

Well connection %: This well connection percentage is the percentage of selected model realizations in which a particular well pair is active. So, for example, if six realizations are selected and a particular well pair is only active in three of those models (i.e., there is only flow between that injector and that producer in three of the models), then the well connection percentage will be 50 %. This can be useful for quickly determining which wells are consistently in connection across many realizations and which are not.

6.3 Residence-time distributions

The GUI can also display residence-time distributions (RTDs) for numerical tracer, which we assumed to be passively advected by the total flux field (Figure 6.6). The distributions are either *estimated* from cell-averaged residence times as a crude approximation to the true RTD, or *computed* as a more accurate approximation by tracing a delta pulse through the domain as described in Section 2.5. This computation is simplified in the sense that it only considers *a single stationary velocity field*, assumes that one kilogram of tracer is injected instantaneously from each selected injector, and ignores diffusive effects.

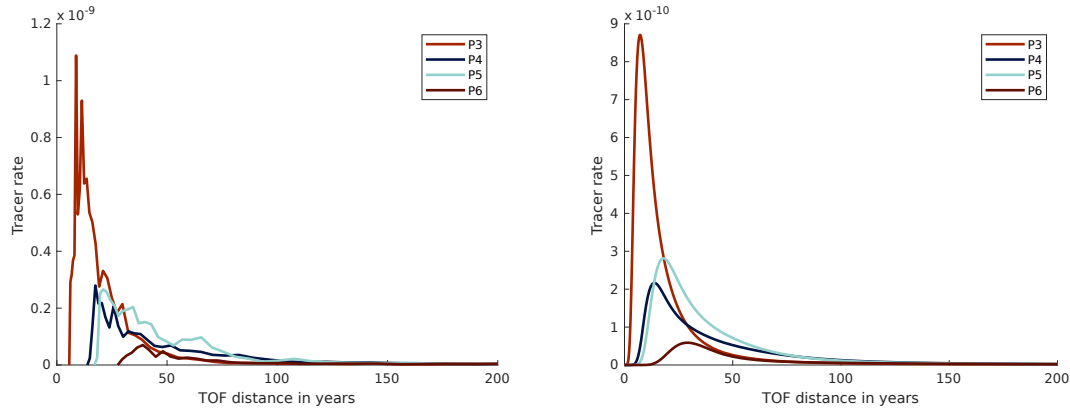


FIGURE 6.6: Residence-time distributions for a single injector connected to several producers. Left hand side shows estimated RTDs and right hand side shows simulated RTDs.

The resulting tracer should thus only be seen as a numerical quantity used to illuminate connections within the parts of the reservoir that contain movable fluids and should not be confused with RTDs of physical tracers, i.e., non-partitioning tracers that follow individual phases or partitioning tracers that distribute into both the aqueous and the oleic phase.

To display the RTD, you must first pick a model realization from the “Model Selection” menu and one injector from the “Region selection” menu. Next, you set the time span for the residence distribution using the input field at the bottom of the menu shown in Figure 6.7 and then use one of the two drop-down menus to choose which quantity (estimated or computed) to display in the left and right 2D panels.

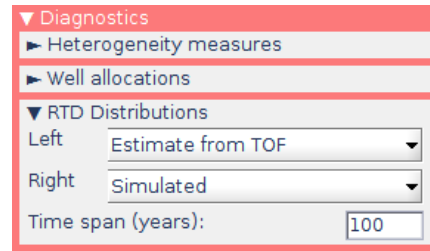


FIGURE 6.7: Menu for computing tracer distributions based on a single model realization.

6.4 Multiphase Analysis

If an initial state with multiple phases is provided when the diagnostics viewer is called, the “Multiphase Analysis” menu enables you to plot the fluid distribution inside the reservoir as function of travel time to the nearest injector or producer. For a production well, in particular, this will give you an idea of the fluids that will be produced in the future, or to be precise, the fluids that would be produced if the displacement process acted like a piston in the same way as the blue fluid displacing the red fluid in Figure 2.6. This is obviously a crude approximation to the true multiphase behaviour, but nonetheless gives valuable information about the displacement process. Plots like this can, for instance, be used to get an approximation of which wells are likely to suffer from early water breakthrough. There is also the possibility to see which injectors contribute the most, in terms of volume, on a fluid-by-fluid basis to what is being produced at a specific production well.

Assuming a piston front that moves through the reservoir and displaces all fluids in front of it, the fluid volume that will have arrived at a specific producer at a certain time T after the start of injection, can then be approximated as the volume of fluids that were originally contained in all the cells with backward time of flight less than or equal T . The volume of phase α that reaches a specific producer P from a specific cell k is given by:

$$V_{\alpha}^P[c_k] = S_{\alpha}[c_k]V_kC_p^P[c_k], \quad (6.3)$$

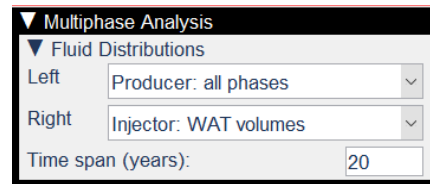


FIGURE 6.8: Menu for multiphase analysis based on initial fluid distributions.

if $\tau_b[c_k] < T$. Furthermore, the fractional part of this volume that has been “pushed” through this cell by a specific injector I is given as:

$$V_\alpha^{IP} = V_\alpha^P [c_k] C_i^I [c_k] \quad (6.4)$$

This way, we can decompose the phase volume into different injector–producer pairs.

To approximate the phase makeup of fluids arriving at a producer through time, we order the values calculated in (6.3) by backward TOF from that producer (i.e., by the time it would take fluids from a specific cell to reach that producer), and then find the cumulative sum up to the maximum TOF value required.

Selecting a specific producer and then choosing “Producer: all phases” from the drop-down menu will produce a plot of the cumulative volume fraction of the phases arriving at the producer through time, as illustrated in Figure 6.9. Selecting “Producer: OIL volumes” will give a plot of the cumulative volume of oil that will reach the producer through time, broken down by the contribution from each injector connected to that producer, (6.4). Selecting other fluid phases will produce equivalent plots for each of the phases present (see Figure 6.9). The time span over which the plots are shown can be specified in the “Time span (years):” box.

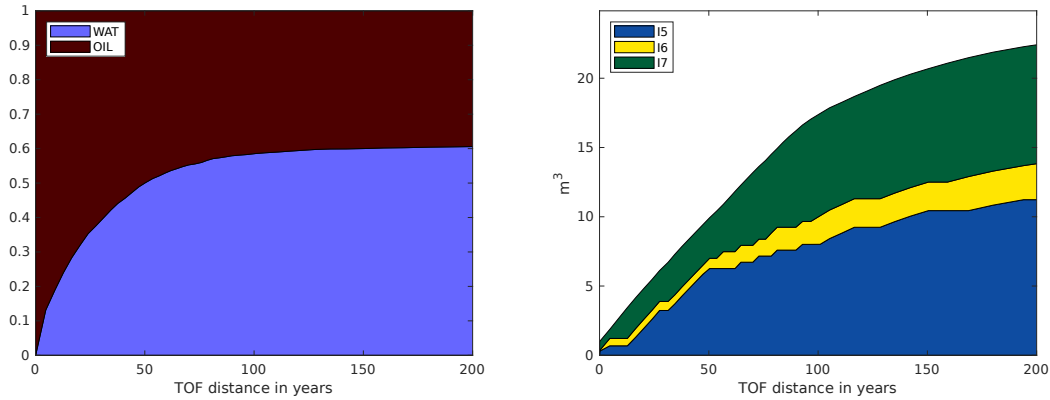


FIGURE 6.9: Plots produced using the “Multiphase Analysis” menu. The left hand plot shows the normalised makeup of all the phases arriving at a producer through time (cumulatively), the right hand plot shows the volume of oil each injector pushes towards a particular producer cumulatively through time.

The exact same type of quantities can be defined for injectors, i.e., “Injector: all phases” gives an inventory of the fluid phases the imaginary piston front would encounter as it moves outward along all flow paths from an injector, whereas “Injector: OIL volumes” and “Injector: WAT” specify how the oil and water volumes are accumulate along flow paths that eventually will reach the different producers connected to the injector.



BIBLIOGRAPHY

- [1] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Eval. Eng.*, 4:308–317, 2001. doi:[10.2118/72469-PA](https://doi.org/10.2118/72469-PA). URL <http://www.spe.org/csp/>.
- [2] A. Datta-Gupta and M. J. King. *Streamline Simulation: Theory and Practice*, volume 11 of *SPE Textbook Series*. Society of Petroleum Engineers, 2007. ISBN 978-1-55563-111-6.
- [3] B. Eikemo, K.-A. Lie, H. K. Dahle, and G. T. Eigestad. Discontinuous Galerkin methods for transport in advective transport in single-continuum models of fractured media. *Adv. Water Resour.*, 32(4):493–506, 2009. doi:[10.1016/j.advwatres.2008.12.010](https://doi.org/10.1016/j.advwatres.2008.12.010).
- [4] O. Izgec, M. Sayarpour, and G. M. Shook. Maximizing volumetric sweep efficiency in waterfloods with hydrocarbon f - ϕ curves. *J. Petrol. Sci. Eng.*, 78(1):54–64, 2011. doi:[10.1016/j.petrol.2011.05.003](https://doi.org/10.1016/j.petrol.2011.05.003).
- [5] J. D. Jansen, R. M. Fonseca, S. Kahrobaei, M. M. Siraj, G. M. Van Essen, and P. M. J. Van den Hof. The egg model – a geological ensemble for reservoir simulation. *Geoscience Data Journal*, 1(2):192–195, 2014. doi:[10.1002/gdj3.21](https://doi.org/10.1002/gdj3.21).
- [6] S. Krogstad and H. M. Nilsen. Efficient adjoint-based well-placement optimization using flow diagnostics proxies. In *ECMOR XVII – 17th European Conference on the Mathematics of Oil Recovery*. EAGE, 2020. doi:[10.3997/2214-4609.202035227](https://doi.org/10.3997/2214-4609.202035227).
- [7] S. Krogstad, K.-A. Lie, H. M. Nilsen, C. F. Berg, and V. Kippe. Efficient flow diagnostics proxies for polymer flooding. *Comput. Geosci.*, 21(5-6):1203–1218, 2017. doi:[10.1007/s10596-017-9681-9](https://doi.org/10.1007/s10596-017-9681-9).
- [8] L. W. Lake. *Enhanced Oil Recovery*. Prentice-Hall, 1989.
- [9] K.-A. Lie. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press, 2019. ISBN 9781108492430. doi:[10.1017/9781108591416](https://doi.org/10.1017/9781108591416).
- [10] O. Møyner, S. Krogstad, and K.-A. Lie. The application of flow diagnostics for reservoir management. *SPE J.*, 20(2):306–323, 2014. doi:[10.2118/171557-PA](https://doi.org/10.2118/171557-PA).
- [11] J. R. Natvig, K.-A. Lie, B. Eikemo, and I. Berre. An efficient discontinuous Galerkin method for advective transport in porous media. *Adv. Water Resour.*, 30(12):2424–2438, 2007. doi:[10.1016/j.advwatres.2007.05.015](https://doi.org/10.1016/j.advwatres.2007.05.015).
- [12] L. Peters, R. Arts, G. Brouwer, C. Geel, S. Cullick, R. J. Lorentzen, Y. Chen, N. Dunlop, F. C. Vossepoel, and R. Xu. Results of the Brugge benchmark study for flooding optimization and history matching. *SPE Reser. Eval. Eng.*, 13(03):391–405, 2010. doi:[10.2118/119094-PA](https://doi.org/10.2118/119094-PA).
- [13] A. F. Rasmussen and K.-A. Lie. Discretization of flow diagnostics on stratigraphic and unstructured grids. In *ECMOR XIV – 14th European Conference on the Mathematics of Oil Recovery*, 2014. doi:[10.3997/2214-4609.20141844](https://doi.org/10.3997/2214-4609.20141844).
- [14] Schlumberger. *ECLIPSE: Reference Manual*. Schlumberger, 2014.1 edition, 2014.

- [15] M. Shahvali, B. Mallison, K. Wei, and H. Gross. An alternative to streamlines for flow diagnostics on structured and unstructured grids. *SPEJ.*, 17(3):768–778, 2012. doi:[10.2118/146446-PA](https://doi.org/10.2118/146446-PA).
- [16] G. Shook and K. Mitchell. A robust measure of heterogeneity for ranking earth models: The F-Phi curve and dynamic Lorenz coefficient. In *SPE Annual Technical Conference and Exhibition*, 2009. doi:[10.2118/124625-MS](https://doi.org/10.2118/124625-MS).
- [17] G. M. Shook and J. H. Forsmann. Tracer interpretation using temporal moments on a spreadsheet. Technical Report INL report 05-00400, Idaho National Laboratory, 2005. doi:[10.2172/910998](https://doi.org/10.2172/910998).
- [18] *User Guide to Flow Diagnostics Postprocessing: Simulations in MRST and ECLIPSE Output Format*. SINTEF Digital, Oct. 2020.
- [19] W. E. Stiles. Use of permeability distribution in water flood calculations. *Journal of Petroleum Technology*, 1(01):9–13, 1949. doi:[10.2118/949009-G](https://doi.org/10.2118/949009-G).
- [20] M. R. Thiele. Streamline simulation. In *6th International Forum on Reservoir Simulation*, Schloss Fuschl, Austria, 3–7 September 2001. URL <https://www.streamsim.com/papers/rsf2001.pdf>.
- [21] M. R. Thiele. Streamline simulation. In *8th International Forum on Reservoir Simulation*, Stresa / Lago Maggiore, Italy, 20–24 June 2005. URL <https://www.streamsim.com/papers/rsf2005.pdf>.
- [22] F. Watson, S. Krogstad, and K. Lie. Flow diagnostics for model ensembles. In *ECMOR XVII – 17th European Conference on the Mathematics of Oil Recovery*. EAGE, 2020. doi:[10.3997/2214-4609.202035133](https://doi.org/10.3997/2214-4609.202035133).

A.1 Prerequisites

The flow diagnostics GUI requires the following installed software:

- MATLAB version 2015b or newer.
- The MATLAB Reservoir Simulation Toolbox version 2017b or newer.

That said, we generally recommend that you use the most up-to-date version of MRST.

A.2 Installing MRST

MRST consists of a core part and a set of add-on modules. These are hosted as a collection of software repositories on bitbucket.org/mrst. Official releases are provided as self-contained archive files that can be downloaded from the website (mrst.no).

Assume now that you have downloaded the tarball of one of the recent releases; here, we use release 2017b as an example. Issuing the following command

```
untar mrst-2017b.tar.gz
```

in MATLAB creates a new folder `mrst-2017b` in your current working directory that contains all parts of the software. Once all code has been extracted to what we henceforth refer to as the *MRST root* folder (`ROOTDIR`), you must navigate MATLAB there, either using the built-in file browser, or by using the `cd` command. Assuming that the files were extracted to the home folder, this would amount to the following on Linux/Mac OS,

```
cd /home/username/mrst-2017b/      % on Linux/Mac OS
cd C:\Users\username\mrst-2017b\  % on Windows
```

MRST is activated through a startup script, which also scans your installation and determines which modules you have installed. In the MRST root folder, the script is called by the command:

```
startup;
```

The startup script will display a welcome message showing that the software is initialized and ready to use. If you start MATLAB from within the MRST root folder, the startup script is run automatically. Alternatively, if you do not want to navigate to the root folder, you can call `startup` directly

```
run /home/username/mrst-2017b/startup  % or C:\MyPath\mrst-2017b\startup
```

For more information about the installation process and how MRST organizes its various components, you should consult [9, Appendix A].

B

OVERVIEW OF THE DIAGNOSTICSVIEWER AND ENSEMBLEGUI CLASSES

B.1 Diagnostics Viewer

The diagnostics GUI is implemented as a MATLAB class object. This object contains all the loaded and computed data as well as various containers, data and structures used to control the visualization and describe its current state. The class allows you to manipulate and control the behavior of the GUI without having to edit the source code and restart the program. Invoking the GUI with the following call (where `ensemble` is the Egg model and `wellIx` holds the indices of the five realizations we have highlighted in earlier examples):

```
d = DiagnosticsViewer(ensemble,wellIx);
```

will create and return the following MATLAB class object:

```
DiagnosticsViewer with properties:
    Figure: [1x1 Figure]
    Axes3D: [1x1 Axes]
    Axes2DL: [1x1 Axes]
    Axes2DR: [1x1 Axes]
    colorBar: [1x1 ColorBar]
    colorHAX: [1x1 Axes]
    colormap3D: 'injectors'
    messageBar: [1x1 UIControl]
    infoTextBox: [1x1 UIControl]
    injColors: [9x3 double]
    prodColors: [5x3 double]
    WellPlot: [1x1 WellPlotHandle]
        Data: {1x5 cell}
        Menu: [1x1 UIMenu]
        Props: [1x1 struct]
    Measures:
    Allocation:
    fdprops: {1x7 cell}
    Distributions: []
    Patch: [1x1 CellDataPatch]
    maxTOF: 1.5778e+10
    currentInteractionRegion: [18553x1 logical]
    currentFilterRegion: []
    currentDiagnostics: [1x10 struct]
    interactiveModes: [1x1 struct]
    camlight: [1x1 Light]
    outlineGrid: [1x1 Patch]
    info: []
    layoutParams: [1x1 struct]
```

Starting from the top, `Figure` is a handle to a standard MATLAB figure object; `Axes3D`, `Axes2DL`, and `Axes2DR` are handles to standard MATLAB axes objects representing the 3D and 2D plotting axes; whereas `colorBar` and `colorHAX` are handles to the colorbar and the axes of the associated histogram. You can manipulate all these as you would do with any other graphical object in MATLAB, e.g., to change their position, background color, etc. As an example, you can directly control the size and position of the GUI

```
d = DiagnosticsViewer(ensemble, modelIx); d.Figure.Position = [1324 572 1230 773];
```

Likewise, you can set the y-axis of the plot in the lower-left panel to logarithmic scale by

```
d.Axes2DL = 'logarithmic';
```

The text string `colormap3D` specifies the current color map in the 3D plot: `'injectors'` for plots of sweep regions, `'producers'` for drainage regions, and `'default'` for all other quantities. Likewise, `injColors` and `prodColors` specifies unique colors for each injector/producer, whereas `infoTextBox` and `messageBar` control the information box in the lower-left corner of the main window and the layout of any error messages displayed below the left 2D axes if plotting requirements are not met when you try to create 2D plots.

Continuing down the list, `WellPlot` holds an instance of the `WellPlotHandle` class, shown to the right, from the `mrst-gui` module. This class contains information about the visual display of the wells on the 3D axes. Here, each element in the `injectors` and `producers` arrays contains a struct with three elements: `body` is a standard `Line` object in MATLAB

```
WellPlotHandle with properties:
    injectors: [8x1 struct]
    producers: [4x1 struct]
    Visible: 'on'
    visibleInjectors: [8x1 logical]
    visibleProducers: [4x1 logical]
```

that represents the well path from heel (top) to toe (bottom), `label` is a standard `Text` object representing the well label, and `connector` is a `Line` object that represents the line that connects the heel of the well and its label. As an example, the following call changes the font size for the labels and the width and color of the connector lines for all producers:

```
for i=1:numel(d.WellPlot.producers)
    d.WellPlot.producers(i).label.FontSize = 8;
    d.WellPlot.producers(i).connector.LineWidth = 3;
    d.WellPlot.producers(i).connector.Color = [.6 .6 .6];
end
```

The last two arrays of logical numbers tell whether each well should be visible in the 3D plot or not. By manipulating these arrays, you can hide a well that is selected as active in the “Region selection” menu or show wells that are not selected as active, e.g.,

```
d.WellPlot.visibleInjectors(1:2) = false;
d.WellPlot.visibleProducers([1 3 4]) = true;
```

The `Data` cell array contains an array of structs, one for each model realization. Each structure consists of a number of fields that represent:

- the grid: `G` is a standard MRST grid structure described in Chapter 3 of [9], whereas `Gs` describes the simulation grid as a graph;
- the static, dynamic, and diagnostics properties discussed in Section 5.3;
- the reservoir states resulting from the incompressible flow solve;
- information about well communication and the unique colors assigned to each well.

As an example, we can look at the static data field. For a single realization of the Egg model, the static data consists of seven elements representing pore volume, permeability in the three axial directions, net-to-gross, depth, and pore volume. The second element is shown to the right.

```
>> disp(d.Data{1}.static(2))
    name: 'PERMX'
    values: [18553×1 double]
    limits: [8.0730e-14 6.9085e-12]
```

The `Data{1}.states` is a standard MRST state structure that contains pressure, saturations, fluxes, face pressures, and well solution. Similarly, `Data{1}.diagnostics` contains the corresponding stationary flow diagnostics, represented as two substructures. The first structure `D` is an extended version of the basic flow-diagnostics data object discussed in [9, §13.1.1] and contains forward and backward time-of-flight, concentrations for stationary injection/production tracers, injector/producer partitions, and time to first arrival:

```
>> disp(d.Data{1}.diagnostics.D)
    inj: [1 2 3 4 5 6 7 8]      % Index of injectors in list of wells
    prod: [9 10 11 12]         % Index of producers
    tof: [18553×2 double]      % Forward and backward time-of-flight
    itracer: [18553×8 double]  % Influence regions for injectors
    itof: [18553×8 double]    % Time-of-flight for each influence region
    ifa: [18553×8 double]     % First arrival time from injectors
    ipart: [18553×1 double]    % Injector partition
    ptracer: [18553×4 double]  % Influence regions for producers
    ptof: [18553×4 double]    % Time-of-flight for each influence region
    pfa: [18553×4 double]     % First arrival time from producers
    ppart: [18553×1 double]   % Producer partition
```

The second structure, `WP`, describes all well pairs

```
>> disp(d.Data{1}.diagnostics.WP)
    struct with fields:
    pairs: {1×32 cell}        % Names for well pairs
    vols: [1×32 double]      % Well-pair volumes
    pairIx: [32×2 double]    % Indices for each pair (injector,producer)
    inj: [1×8 struct]        % Well allocations, depth to perforation, injectors
    prod: [1×4 struct]       % Same as 'inj', but for producers
```

We can also mention the field `Data{1}.diagnostics.wellCommunication`, which contains the matrix (6.2) measuring well connection strength.

The `d.Menu` field holds an instance of the `UIMenu` class and contains all data structures necessary to represent the input menus to the left in Figure 3.2. Each of these are realized as a set of class objects. The `d.Props` structure contains the names of all the properties that can be displayed in the 3D panel along with limits on their range, whereas `d.Measures` and `d.Allocations` contain the names of dynamic heterogeneity measures and quantities representing volumetric and well allocation. The `d.Distribution` field is not used herein. The rest of the fields are primarily internal data structures used to represent graphical states and accelerate visualization by storing data that would otherwise have to be recomputed.

B.2 Ensemble GUI

The ensemble GUI is also implemented as class object, which for the 50-member Egg ensemble contains the following properties:

```
EnsembleGUI with properties:
    Figure: [1x1 Figure]
    Axes: [1x1 Axes]
    Menu: [1x1 UIMenu]
    Data: [1x1 struct]
    m: [1x1 ModelEnsemble]
    currentDiagnostics: [1x1 struct]
        names: {50x1 cell}
    injectors: []
    validMembers: [50x1 double]
    memberSelection: [50x1 double]
    memNoToLocalNo: [50x1 double]
    markers: [1x50 Line]
    labels: [1x50 Text]
    lines: [1x50 Line]
    hist: [1x50 Line]
    fitLine: [1x1 Line]
    crossPlotSelections: {1x6 cell}
    linePlotSelections: {1x9 cell}
    histPlotSelections: {'Porosity' 'Perm.x' 'Perm.y' 'Perm.z'}
    injectorNames: {1x8 cell}
    producerNames: {'PROD1' 'PROD2' 'PROD3' 'PROD4'}
    layoutParams: [1x1 struct]
    plotOpts: [1x1 struct]
    currentPlotType: 'cross'
    textColor: [1x1 struct]
    highlightColors: [7x3 double]
    diagnosticsViewerHandle: [1x1 DiagnosticsViewer]
    xData: [50x1 double]
    yData: [50x1 double]
    xSelectionCrossPlot: 'Lorenz coefficient'
    ySelectionCrossPlot: 'Sweep'
    selectionLinePlot: 'F-Phi'
    selectionHistPlot: [1x1 struct]
    injectorIx: [1 2 3 4 5 6 7 8]
    producerIx: [1 2 3 4]
    sizeProp: 'none'
    maxTOF: 1
    timeUnit: 'pvi'
    fluidProps: [1x1 struct]
    npvProps: [1x1 struct]
    useOriginalFluid: 1
```

Here, we recognize some of the fields discussed in the previous section like the figure/axes handles and the `Menu` object. There are also a number of fields that are primary internal data structures used for book-keeping, to represent graphical states, to contain input from the different menus, etc. Notice also the handle to the diagnostics viewer, which you can use to control any instance of this class called from within the ensemble GUI.

Moving on to the data objects, the `m` field holds the ensemble class object passed into the GUI, whereas the `Data` object here contains the well-pair object (WP) discussed earlier, residence-time distributions and one linearized fluid object for each ensemble member, the well communication matrix, and a data structure containing allocations, phase rates, and volumes for each of the 32 potential well pairs. The data points for the line plots of the various diagnostics are computed directly from the individual residence time distributions and are hence independent of the model grid sizes. The computations are done on demand and stored in the `currentDiagnostics` field, which thus represents our computational cache. This cache is emptied every time the user changes the interaction region, and is gradually filled up as needed. It consists of the following fields:

```
WP: {1x50 cell}
RTD: {1x50 cell}
F: {1x50 cell}
Phi: {1x50 cell}
SweepEfficiency: {1x50 cell}
recovery: {1x50 cell}
rTime: {1x50 cell}
ModelNo: {1x50 cell}
LorenzCoefficient: {1x50 cell}
Sweep: {1x50 cell}
RecoveryFactor: {1x50 cell}
NPVPerVolume: {1x50 cell}
maxNPV: {1x50 cell}
F_Phi: {1x50 cell}
SweepVsTime: {1x50 cell}
ResidenceTimeDistribution: {1x50 cell}
OilRates: {1x50 cell}
WaterRates: {1x50 cell}
WCUTVsTime: {1x50 cell}
RecoveryVsTime: {1x50 cell}
RecoveryFactorVsTime: {1x50 cell}
NPVVsTime: {1x50 cell}
Porosity: {1x50 cell}
Perm_x: {1x50 cell}
Perm_y: {1x50 cell}
Perm_z: {1x50 cell}
```