Contents

1 Installation **1**

2 Running your first model **1**

2.1 Model structure . . . 1

2.1.1 The main model file . . . 2

2.1.2 Defining the species that participate in the model . . . 3

2.1.3 Defining kinetic constants . . . 4

2.1.4 Age-dependent reaction rates . . . 5

2.1.5 Setting up reactions . . . 5

2.1.6 How Mantis determines reaction order . . . 6

2.1.7 Defining the plot output . . . 7

2.1.8 A note on Units . . . 8

2.1.9 Adding measured data . . . 9

3 Advanced usage **10**

3.1 Kinetic Isotope Effects . . . 10

3.2 Sorption & Speciation . . . 11

3.2.1 The Three Physical Phases in MANTIS . . . 11

3.2.2 The `basis` Keyword: Porewater vs. Solid Volume Units . . . 11

3.2.3 How MANTIS Handles Mixed-Phase Transport and Bioturbation . . . 12

3.2.4 Declaring Lumped Species in `species.py` . . . 13

3.3 Speciation and Equilibrium Isotope Effects . . . 14

3.4 Dynamic pH and Carbonate Chemistry . . . 15

3.5 Adding $CaCO_3$ dissolution . . . 16

3.5.1 Why not a single reversible reaction arrow? . . . 16

3.5.2 How to implement $CaCO_3$ precipitation and dissolution . . . 17

1 Installation

- `pip install mantis-rtm`
- Optional: In an empty directory run `mantis_create_examples.py`

2 Running your first model

After creating the example directory, go to `./examples/start_here/basic_transport_and_reaction`

2.1 Model structure

A typical MANTIS project directory contains five primary configuration files, alongside the master run script:

```
./basic_transport_reaction/
├── species.py           <- Who lives in the sediment? (chemical species, phases, diffusi
├── reactions.py         <- What reactions happen? (stoichiometry, kinetics, limiters)
├── kinetic_constants.py <- How fast do reactions happen? (rate constants, half-
saturation values)
├── plot_layout.py       <- How should the profiles look? (panels, colors, geochemist uni
└── run_model.py         <- Master driver script (sediment accumulation rate, depth, dura
```

When you execute `python run_model.py`, MANTIS automatically detects these files, ensures the underlying equation system is up to date, solves the time evolution of the sediment column, and produces your outputs:

- `<model_name>.pdf`: A multi-panel summary plot of porewater and solid-phase profiles.
- `<model_name>.csv`: A tabular export of concentrations, reaction rates, and isotope ratios across depth.
- `<model_name>_state.npz`: The complete numerical state of the simulation for restarting or continuing runs.
- `<model_name>.mp4` (optional): An animated video showing the transient evolution

2.1.1 The main model file

The model definition is straightforward, importing the libraries an wrapping an `if __name__ == "__main__":` block, and then define some basic parameters:

```python
from mantis import DiageneticModel
from mantis.diff_lib import get_total_delta

if __name__ == "__main__":
    model = DiageneticModel(
        reactions="reactions.py",
        equations="equations.py",
        constants="kinetic_constants.py",
        layout="plot_layout.py",
    )

    # 2. Configure environmental and run parameters
    model.configure(
        grid_setup=[  # start, stop, start_dz, stop_dz
            ["0 cm", "100 cm", "2 mm", "2 mm"],
        ],
        t_end="2 kyr", # max run time
        dt_max="1 year", # max dt
        temp=[10.0, 10.1], # temperature field
        w="0.2 cm/yr", # sedimentation rate
```

```
        phi=0.8, # porosity
    )

    # Allow CLI argument overrides (e.g. --video, --solver-tol)
    model.parse_cli_args()

    # 3. Run simulation
    result = model.run()

    # 4. Save results & render summary plot
    df, fqfn = result.save_csv()
    # result.save_state() optional

    result.plot_summary()
```

2.1.2 Defining the species that participate in the model

The species file contains all the information we need to know about a species. This includes the type (solid or liquid), diffusion coefficients, boundary conditions, the chemical formula, as well as a field that decides whether a species is part of the transport reactions. Most species are covered by the `diff_coeff` function with the `m0, m1` parameter listed in Table 4.8 in Boudreau 1996. Note that diffusion coefficients are only needed for species that are part of the reaction-transport matrix (i.e., where include = True). As before, all properties are defined in a dictionary. Unless a lower boundary condition is specified, it defaults to a zero gradient Neumann condition. The `prefill` parameter determines of the initial species concentration across the modeling domains is set to the upper boundary condition or to zero. Note that "CO2" and "H2O" need to be present, even so they do not appear in the model directly.

```

species = {
    # Solid species
    "POC": {
        "phase": "solid",
        "include": True,
        "formula": "CH2O6",
        "diffusion": None,
        "bc_top": {"Robin": "3065 umol/(cm^2 * yr)"},
    },
    # ------------------------------------------------------------------------
    # Dissolved Liquid Species
    # ------------------------------------------------------------------------
    "SO4": {
        "phase": "liquid",
        "include": True,
        "formula": "SO4",
        "diff_coeff_function": "diff_coeff",
        "diffusion": {"m0": 4.88, "m1": 0.232},
```

```
        "bc_top": {"Dirichlet": 28.2},
        "prefill": True,
    },
    "TS2": {
        "phase": "liquid",
        "include": True,
        "formula": "H2S",
        "diff_coeff_function": "diff_coeff",
        "diffusion": {"m0": 10.4, "m1": 0.273},
        "bc_top": {"Dirichlet": 0.0},  #
    },
    # External / Reaction Balance Species (Excluded from transport)
    "CO2": {"phase": "liquid", "include": False, "formula": "CO2"},
    "H2O": {"phase": "liquid", "include": False, "formula": "H2O"},
}
```

2.1.3 Defining kinetic constants

Kinetic constants are defined in a dictionary. In the below example, I use the `pint` library to map units into the model units, and return both dictionaries.

In diagenesis, we typically distinguish:

1. **Rate Constants (k)**: The maximum rate of a reaction under substrate-saturated conditions (e.g., k_{POC} in yr^{-1} or s^{-1}, or mineral precipitation rates k_{prec}). While the names are arbitrary they must match the names in the reaction file. It also a good idea that parameters that describe first order values start with the name of the species the rate constant refers too.

2. **Half-Saturation Constants (K_m)**: The concentration of an electron acceptor at which the microbial degradation rate reaches 50% of its maximum. Lower K_m means the microbe is efficient at scavenging low concentrations, e.g., $\frac{1}{C+K_m}$ (see notes in the reaction section) .

3. **Inhibition Constants**: The threshold concentration of a more energetically favorable oxidant that shuts down a less favorable pathway (e.g., O_2 inhibiting iron reduction or sulfate reduction), e.g., $1 - \frac{1}{C+K_m}$ (see notes in the reaction section) .

```
import pint

    ureg = pint.UnitRegistry()
    Q_ = ureg.Quantity

    constants: dict = {
        # kinetic constants
        "POC": [0.001, "1/year", "1/second"],  # key, [value, unit, target unit]

        # Monod half-saturation for microbial sulfate reduction
        "K_SO4": [0.9, "umol/cm^3", "mol/m^3"],
```

```
    }

    # convert into model units (meter, mol, seconds)
    for k, v in constants.items():
        k_values[k] = Q_(f"{v[0]} {v[1]}").to(f"{v[2]}").magnitude

    return constants, k_values
```

2.1.4 Age-dependent reaction rates

Organic matter reactivity decreases with depth as sediment ages and labile compounds are degraded. In MANTIS, age-dependent reaction rate constants can be defined using the built-in `sed_age` function from `mantis.diff_lib`:

```
from mantis.diff_lib import sed_age

def get_reaction_constants(pH, phi):
    # Configure sediment aging profile (Middelburg power law: a0=1.0 yr, a=0.21 yr^-1, b=1.22)
    age = sed_age(mode="relative", a0=1.0, a=0.21, b=1.22)

    constants = {
        # 'relative' mode scales the initial rate constant by ((age + a0) / a0)^(-b):
        "k_POC": "age * 0.01 1/year",

        # Monod half-saturation for microbial sulfate reduction
        "K_SO4": "0.9 umol/cm^3",
    }
    return constants
```

Two scaling modes are supported:

- `mode="relative"` (default): Evaluates a dimensionless attenuation factor $((A(z)+a_0)/a_0)^{-b}$ with surface value \$1.0\$. The reaction constant is specified as `"age * <initial_rate>"` (e.g. `"age * 0.01 1/year"`).
- `mode="direct"`: Directly computes the Middelburg apparent rate constant $k(z) = a \cdot (A(z)+a_0)^{-b}$ (in [1/yr] converted to SI [1/s]), specified simply as `"age"` or `"1.0 * age"`.

The sediment age profile $A(z)$ is computed automatically from 1D advective-diffusive transport accounting for burial velocity (w), porosity compaction (ϕ), and bioturbation mixing (D_b).

2.1.5 Setting up reactions

Microbial sulfate reduction rates are sensitive to sulfate concentrations, so in the reactions file, we first define a limiter for `SO4`. This is followed by a list of dictionaries

that define the reactions of the model. The reaction name is arbitrary, but the chemical reaction can only use names that are defined in the species file. An important limitation is that the stoichiometric must be integer numbers. The kinetic constant name must exist in the kinetic constants file, the limiters parameter will only accept values that are defined in the `limiters` dictionary. A note on limiters. The canonical form of writing a limiter is

$$\frac{[SO_4]}{[SO_4] + K} \tag{1}$$

However, mantis exclusively uses implicit limiters:

$$\frac{1}{[SO_4] + K} \tag{2}$$

```
# Monod type reaction rate limiters
limiters = {
    # Base implicit limiters: L_imp = 1.0 / (C + K)
    # The parser automatically derives explicit (C * L_imp) and inhibition (K * L_imp = 1 - C * L
    "SO4": "1.0 / (c.SO4 + K_SO4)", # where c.SO4 = sulfate concentration
}

reactions = [
    # ------------------------------------------------------------------------
    # Organic Carbon Primary Respiration
    # ------------------------------------------------------------------------
    {
        "reaction_name": "sulfate_reduction_slow",
        "reaction": "2 POC + SO4 -> TS2",
        "k_value_name": "POC",
        "limiters": ["SO4"],
        "inhibitors": None,
    },
]
```

2.1.6 How Mantis determines reaction order

By default, Mantis assigns reaction order based on the number of reactants in the reaction stoichiometry:

- **Single-reactant reactions** (e.g. `POC -> DIC`) are treated as **first-order**:

$$R = k \cdot [A] \cdot \prod \text{limiters} \cdot \prod \text{inhibitors}$$

- **Multiple-reactant reactions** (e.g. `CH4 + SO4 -> TS2 + DIC`) follow **second-order** mass-action kinetics:

$$R = k \cdot [A] \cdot [B] \cdot \prod \text{limiters} \cdot \prod \text{inhibitors}$$

You can explicitly override this default for any reaction by specifying the `reaction_order` property (values: `0`, `1`, or `2`):

- `"reaction_order": 0`: Zero-order in reactant concentrations ($R = k \cdot \prod \text{limiters} \cdot \prod \text{inhibitors}$). This is essential for mineral precipitation or dissolution governed purely by saturation state ($\Omega - 1$), where the reaction stoichiometry determines PDE mass balance and phase coupling, but the kinetic rate law must not multiply reactant concentrations:

```
{
    "reaction_name": "calcite_precipitation",
    "reaction": "Ca2 + DIC + 2 TA -> CaCO3",
    "k_value_name": "k_calcite_precip",
    "limiters": ["calcite_precip_limiter"],
    "reaction_order": 0,  # Rate is purely k * limiter(Omega - 1)
}
```

- `"reaction_order": 1`: First-order in the primary reactant only ($R = k \cdot [A] \cdot \prod \text{limiters}$), preventing secondary stoichiometric reactants from being multiplied into the rate.
- `"reaction_order": 2`: Explicit second-order mass action ($R = k \cdot [A] \cdot [B] \cdot \prod \text{limiters}$).

2.1.7 Defining the plot output

Mantis provides a default plot output that can be adapted to a variety of layouts. Its basic structure requires that you define the x-axis scale, as well as one or more panels. For each panel, you can create a one or more `liquid_species` objects, none or more `solid_species` objects which allow for the mapping into weight percent, none or more `reaction_rates` objects that allow for the continuous display of negative and positive rates on a logarithmic scale, and isotope object, that allows for broken scales. These objects are then grouped into one or more vertical panels.

```
#  ---- z-axis (Depth) Scale and Limits ----
z_scale = "lin"  # Or "linear" / "lin" (default: "log")
z_lim = (0, 1)  # (z_min, z_max) in meters (default for log: (1e-4, display_length))

# ----  concentration data panel  --- #
liquid_species_1 = liquid_species({
    "scale": "lin",
    "species": ["SO4", "TS2"],
    "lim": (0, 30),
    "units": "mmol/l",
})

solid_species_1 = solid_species([
    {
        "scale": "lin",
        "units": ["wt"],
```

```
        "relative_to": "C",
        "species": ["POC"],
        # "lim": (0, 1),
    }
])

# ----  reaction rate data panel  --- #
reaction_rates_1 = reaction_rates({"species": ["POC"]})

# ---- create final panel arrangement ------ #
plot_panels = [
    [liquid_species_1, solid_species_1],
    [reaction_rates_1],
]
```

2.1.8 A note on Units

Under the hood, MANTIS standardizes all transport and reaction PDEs into SI-derived volumetric units:

Internal Concentration Unit = $\mathbf{mol/m^3}$ $(\equiv 1\ \mathrm{mmol/L} \equiv 1\ \mathrm{mM} \equiv 1000\ \mu\mathrm{mol/L} \equiv 1\ \mu\mathrm{mol/cm^3})$

Crucially, concentrations are defined **per unit volume of their respective phase**:

- **Liquid Solutes**: Expressed per volume of porewater (mol / $\mathrm{m^3_{liquid}}$ = mmol / $\mathrm{L_{porewater}}$). The bulk sediment inventory is $\phi \cdot C_{\mathrm{liquid}}$.
- **Solid Minerals**: Expressed per volume of pure solid sediment (mol / $\mathrm{m^3_{solid}}$ = mmol / $\mathrm{L_{solid}}$). The bulk sediment inventory is $(1-\phi) \cdot C_{\mathrm{solid}}$.
- **Phase Transfer & Reactions**: Whenever a reaction couples a solid phase and a liquid phase (for example, $\mathrm{Fe^{2+}} + \mathrm{HS^-} \rightarrow \mathrm{FeS}$), MANTIS scales the source and sink terms with the appropriate phase ratio ($(1-\phi)/\phi$ or $\phi/(1-\phi)$) so that mass is strictly conserved.
- **Reaction Rates**: Solved internally in $\mathrm{mol/(m^3_{phase} \cdot s)}$.
- **Depths and Transport**: Depth z is in meters (m), time t in seconds (s), and diffusion coefficients in $\mathrm{m^2/s}$.

Output Units When the simulation completes or exports data (`<model_name>.csv`):

- `c_<species>`: Reported directly in **mmol/L** (numerically equivalent to $\mathrm{mol/m^3}$):
- Solutes (e.g., `c_SO4`, `c_O2`, `c_TS2`) are in **mmol / L of porewater**.
- Solids (e.g., `c_FeS`, `c_Fe3`, `c_POC_slow`) are in **mmol / L of pure solid**.
- `f_<species>`: Net production/consumption reaction rates in $\mathbf{mol/(m^3 \cdot s)}$.

- `d_<species>` and `d34S_<species>`: Isotope delta values (e.g., `d_SO4`, `d34S_SO4`, `d34S_FeS2`) in **milli Urey** (mUr, relative to reference standards such as VCDT or VPDB). Note 1 mUr = 1‰
- `z`: Depth below sediment-water interface in **meters** (m).
- `phi` and `w`: Porosity (fraction, dimensionless) and burial velocity (m/s).

  Converting to μmol / g (micromoles per gram of dry sediment)

To convert a solid concentration C_{solid} (mmol/L of solid = μmol/cm$^3_{\text{solid}}$) to μmol/g of dry sediment, divide by the mineral grain density ρ_{sed} (typically 2.6 g/cm^3 for siliciclastic sediments):

$$C_{\mu\text{mol/g}} = \frac{C_{\text{solid}}}{\rho_{\text{sed}}}$$

```
from mantis.diff_lib import solid_conc_to_umol_per_g

# df.c_FeS is in mmol/L of solid; density in g/cm^3
c_fes_umol_g = solid_conc_to_umol_per_g(df.c_FeS, d=2.6)
```

B. Converting to Dry Weight Percent (wt%) To convert C_{solid} to weight percent of dry sediment matrix given molecular weight M_w (g/mol):

$$\text{wt\%} = \frac{C_{\text{solid}} \cdot M_w}{\rho_{\text{sed}} \cdot 10^4}$$

In Python:

```
from mantis.diff_lib import solid_conc_to_wt_percent

# Example: FeS (Mw = 87.91 g/mol, density = 2.6 g/cm^3)
fes_wt_pct = solid_conc_to_wt_percent(df.c_FeS, mw=87.91, d=2.6, phi=df.phi)
```

2.1.9 Adding measured data

The plot file can specify a second csv file that contains data to compare against. For this to work you need to add the name of the csv file, as well as a mapping table at the beginning of the `plot_layout` file

```
# ----  optional comparison against existing data --- #
comparison_data = "SimpleFe_Du_2023.csv"

# Mapping uses the following scheme:
# Mantis Name : [name, units, display units]
species_mapping = {
    "z": ["depth_in_cm", "cm", "m"],
```

```
    "SO4": ["SO4", "mmol/cm^3_pw", "mmol/L"],
    "Fe2_dis": ["TFe", "mmol/cm^3_pw", "umol/L"],
    "Fe2_liq": ["Fe_aq", "mmol/cm^3_pw", "nmol/L"],
    "FeS": ["FeS", "mmol/cm^3_solid", "wt%"],
    "pH": ["pH", "pH", "pH"],
}
```

3 Advanced usage

3.1 Kinetic Isotope Effects

To model isotope effects the main model script must set `isotopes=True`, the species file must define isotope species, and the reaction file must define isotope effects (you also need to update the plot file). A full example is available in the `examples/velde_et_al_2016/with_isotopes/` directory. The below snippets outline the basic steps.

In the species file we have to add basic information about the isotope system we are using, a boundary condition for the isotope ratio, and we need to define that sulfate will have ^{32}S as isotope of the element S

```
isotope_standards: dict = {
    "d34S": {
        "element": "S",
        "standard": "VCDT",
        "ratio": 0.044162589,  # 34S / 32S
        "unit": "mUr",
    },
}
"SO4": {
      # in addition to the previous entries above
      "isotopes": {"SO4_S32": "S"},
      "bc_top": {"Dirichlet": 28.2, "d34S": 21},
  },
```

Reactions without kinetic isotope effects do not need to be modified. If a reaction involves a kinetic isotope effect we must add the fractionation factor to the `reaction_parameters` dictionary, and then add a line that specifies which isotope, and which species are involved. Since the isotope effect of microbially mediated sulfate reduction is concentration dependent, the below code additionally specifies a Monod-type limiter for the fractionation factor. Please note the dictionary key must follow the `species_isotope to species_isotope` pattern. Otherwise the parser will fail.

```
reaction_parameters = {
    "msr_alpha": 1.07,
}
{
        "reaction_name": "sulfate_reduction",
```

```
        "reaction": "2 POC_fast + SO4 -> TS2",
        "k_value_name": "POC_fast",
        "limiters": ["SO4"],
        "inhibitors": ["O2", "Fe3_diss_red"],
        "isotope_fractionation": {
            "SO4_S32 to TS2_S32": ["msr_alpha", "SO4_alpha_explicit"]
        },
    },
```

3.2 Sorption & Speciation

Many reactive chemical species in marine sediments undergo rapid, reversible adsorption and desorption onto solid surfaces (such as particulate organic carbon, iron and manganese oxihydroxides, or clay minerals). Common examples include ferrous iron (Fe^{2+}), manganese (Mn^{2+}), and ammonium (NH_4^+).

Because sorption reactions typically reach local chemical equilibrium on timescales of seconds to minutes—far faster than the transport timescales of burial and diffusion (years to millennia)—tracking dissolved and adsorbed pools as separate, stiff differential equations is both computationally expensive and numerically unstable. Instead, MANTIS allows you to model these pools as a single **lumped species** (also referred to as a **mixed-phase species**).

3.2.1 The Three Physical Phases in MANTIS

When declaring a species in `species.py`, the `phase` attribute controls its physical compartment:

- `"phase": "liquid"`: Pure porewater solutes (e.g., SO_4^{2-}, O_2, HS^-). The species inventory is scaled by porosity (ϕ), and transport occurs via molecular diffusion, porewater advection, and bio-irrigation.
- `"phase": "solid"`: Pure solid minerals or particulates (e.g., $POC, FeS_2, FeOOH$). The species inventory is scaled by the solid fraction ($1 - \phi$), and transport occurs solely by sediment accumulation (burial) and particle bioturbation.
- `"phase": "mixed"`: Lumped pools containing both a dissolved porewater fraction and a surface-adsorbed solid fraction (e.g., Fe^{2+}_{total}).

3.2.2 The `basis` Keyword: Porewater vs. Solid Volume Units

Because a lumped pool resides partially in the pore fluid and partially in the solid sediment, you must specify which physical volume its concentration represents. This is controlled by the `basis` keyword in `species.py`:

- `"basis": "porewater"` (or `"liquid"`): The lumped concentration is expressed per unit volume of porewater ($\text{mol} \cdot \text{m}_{\text{pw}}^{-3}$, equivalent to mmol/L or mM). This

is the standard convention when comparing model results directly to porewater geochemical data or benchmarks like Du (2023). MANTIS automatically uses the porewater porosity ($\phi_{\text{eff}} = \phi$) in the time-derivative and transport terms:

$$C_{\text{total}} = C_{\text{dis}} + \frac{1-\phi}{\phi} C_{\text{sorb}}$$

- `"basis": "solid"` (default if omitted): The lumped concentration is expressed per unit volume of dry solid sediment ($\text{mol} \cdot \text{m}_{\text{solid}}^{-3}$). This convention is typically used when the species is predominantly solid-bound (e.g., in Velde et al., 2016, where $> 99\%$ of ferrous iron is sorbed). MANTIS uses the solid volume fraction ($\phi_{\text{eff}} = 1 - \phi$):

$$C_{\text{total, solid}} = C_{\text{sorb}} + \frac{\phi}{1-\phi} C_{\text{dis}}$$

3.2.3 How MANTIS Handles Mixed-Phase Transport and Bioturbation

In a lumped pool, the dissolved fraction and the adsorbed fraction experience entirely different physical transport processes:

1. The **dissolved fraction** (f_{dis}) diffuses through pore fluids via molecular diffusion (D_{mol}) and is advected by porewater flow.
2. The **adsorbed fraction** ($1 - f_{\text{dis}}$) is bound to solid particles and is mixed upwards and downwards by macrofaunal bioturbation (D_{bio}) and buried at the sediment accumulation rate (w_{sed}).

MANTIS automatically constructs a unified effective transport stencil that partitions transport according to the instantaneous local dissolved fraction $f_{\text{dis}}(z)$:

$$D_{\text{eff}}(z) = f_{\text{dis}}(z) \cdot D_{\text{mol}}(z) + [1.0 - f_{\text{dis}}(z)] \cdot D_{\text{bio}}(z)$$

This dynamic partitioning has crucial physical consequences:

- In the surface bioturbated zone ($D_{\text{bio}} > 0$), solid mixing transports sorbed iron upwards toward the sediment-water interface, where it can desorb and diffuse into the overlying water column.
- Neglecting bioturbation on the adsorbed fraction would artificially trap sorbed species in the sediment, leading to systematic over-prediction of authigenic mineral precipitation (e.g., FeS burial).
- When $f_{\text{dis}} \to 1$ (no sorption), D_{eff} seamlessly reduces to pure molecular diffusion. When $f_{\text{dis}} \to 0$ (completely sorbed), D_{eff} reduces to solid particle biodiffusion.

3.2.4 Declaring Lumped Species in `species.py`

Here is an example of defining a lumped ferrous iron pool (Fe^{2+}_{total}) on a porewater basis, along with its diagnostic dissolved and sorbed sub-species:

```
species = {
    # ------------------------------------------------------------------------
    # 1. Primary Lumped Transport Variable (Porewater Basis)
    # ------------------------------------------------------------------------
    "Fe2_total": {
        "phase": "mixed",
        "basis": "porewater",                    # Tracked in mol/m^3 porewater (mM)
        "include": True,                         # Primary PDE state variable
        "formula": "Fe2",
        "diff_coeff_function": "diff_coeff",
        "diffusion": {"m0": 3.31, "m1": 0.150}, # Molecular diffusion of dissolved Fe2+
        "bc_top": {"Dirichlet": "5.46e-5 mmol/l"},
        "prefill": True,
    },

    # ------------------------------------------------------------------------
    # 2. Diagnostic Equilibrium Sub-Species (Non-Transported)
    # ------------------------------------------------------------------------
    # Free dissolved Fe2+ that drives mineral precipitation (e.g., FeS precipitation)
    "Fe2_liq": {
        "phase": "liquid",
        "include": False,                        # Derived algebraically; not a primary PDE
        "formula": "Fe2",
        "alias_of": "Fe2_total.dissolved",
    },

    # Surface-adsorbed iron bound to solid particles
    "Fe2_ads": {
        "phase": "solid",
        "include": False,
        "formula": "Fe2",
        "alias_of": "Fe2_total.sorbed",
    },
}

# ----------------------------------------------------------------------------
# Speciation and Phase Partitioning Block
# ----------------------------------------------------------------------------
speciation = {
    "iron_sorption": {
        "type": "partition",
        "primary": "Fe2_total",
        "fractions": {
            # Fraction dissolved in porewater vs. adsorbed to solid POC:
            # f_dis = 1 / (1 + K_ads * [POC] * (1 - phi) / phi)
```

```
            "dissolved": "fe2_dis_frac",
            "sorbed": "1.0 - fe2_dis_frac",
        },
        "species": {
            "Fe2_liq": {"phase": "liquid", "fraction": "fe2_dis_frac"},
            "Fe2_ads": {"phase": "solid", "fraction": "1.0 - fe2_dis_frac"},
        },
    },
}
```

In reaction equations (in `reactions.py`), you can use the diagnostic species (e.g., `Fe2_liq + HS -> FeS`) so that precipitation rates respond specifically to the active free dissolved solute concentration, while MANTIS conserves mass across the total lumped pool. The model code in `examples/velde_et_al_2016/no_isotopes/` contains an example of how to use this in a model.

3.3 Speciation and Equilibrium Isotope Effects

Some species, e.g., H_2S exist in different forms depending on pH ({H_2S <-> HS- + H+}), which may also include an equilibrium isotope effect. The below example shows the pH dependent speciation of H_2S and the associated fractionation. The full code is in the species file in `examples/velde_et_al_2016/with_isotopes/` . Note that in this model, H_2S is called `TS2`

```
speciation = {
    "sulfide_acid_base": {
        "type": "acid_base",
        "primary": "TS2",
        "constants": {"pKa1": 7.0},
        "equilibrium_fractionation": {
            "standard": "d34S",
            "alpha": {"H2S_S32_to_HS_S32": 0.99991542},
        },
        "species": {
            "HS": {
                "phase": "liquid",
                "fraction": "Ka1 / (Hplus + Ka1)",
                "include_in_TA": 1.0,
                "isotopes": {"HS_S32": "S"},
            },
            "H2S": {
                "phase": "liquid",
                "fraction": "Hplus / (Hplus + Ka1)",
                "isotopes": {"H2S_S32": "S"},
            },
        },
    },
```

Since a species can have more than one isotope system (e.g., sulfate has sulfur and oxygen), we must explicitly state which isotopes participate in a reaction, and, in case they involve a kinetic isotope effect, include that as well. The below example is from `examples/velde_et_al_2016/with_isotopes_CaCO_dissolution`

```
{
        "reaction_name": "sulfate_reduction_slow",
        "reaction": "2 POC_slow + SO4 -> TS2 + 2 DIC + 2 TA",
        "k_value_name": "POC_slow",
        "limiters": ["SO4"],
        "inhibitors": ["O2", "Fe3_diss_red"],
        "isotope_species": {
            "SO4": "SO4_S32",
            "TS2": "TS2_S32",
            "POC_slow": "POC_slow_C12",
            "DIC": "DIC_C12",
        },
        "isotope_fractionation": {
            "SO4_S32 to TS2_S32": ["msr_alpha", "SO4_alpha_explicit"]
        },
```

3.4 Dynamic pH and Carbonate Chemistry

Mantis computes pH as a function of DIC and TA as master variables, rather than solving the transport equations for H^+. As such, the reaction file needs to specify how each reaction affects DIC and TA. For dissimilatory iron reduction, we can write e.g.,

```
{
       "reaction_name": "iron_reduction_poc",
       "reaction": "POC + 4 FeOOH -> DIC + 4 Fe2_total + 8 TA",
       "k_value_name": "k_POC",
       "ref_species": "POC",
       "limiters": ["FeOOH"],
   },
```

Further, we need to supply species definitions for TA, DIC, H^+ as well as Ca^{2+} - see the species files in `mantis/examples/SimpleFe/`. Next we must define the carbonate speciation, and associated parameters

```
# ------------------------------------------------------------------------
   # Marine Carbonate System Speciation (DIC + TA -> CO3, HCO3, Hplus, pH)
   # ------------------------------------------------------------------------
   "carbonate_system": {
       "type": "carbonate_system",
       "primary": {"dic": "DIC", "ta": "TA"},
       "species": ["Hplus", "CO3", "HCO3"],
       "pressure": 50.0,       # 500 m water depth (~50 bar hydrostatic pressure)
```

```
        "salinity": 35.0,       # 35 PSU bottom water salinity
        "adjust_omega": False,  # Retain exact boundary values without forcing calcite equilibrium
        "ph_scale": "free",
        "controls": ["DIC", "pH"],
    },
```

Note that any species that contributes to TA (e.g., HS^-) must define their effect on TA by setting `include_in_TA": 1.0` (see e.g., the species definition for H_2S above). The Du 2023 SimpleFe model further shows how to then use pH to calculate dissolved iron speciation and sorption.

3.5 Adding $CaCO_3$ dissolution

In diagenetic models, authigenic calcium carbonate ($CaCO_3$, such as calcite or aragonite) undergoes reversible precipitation and dissolution depending on the porewater saturation state:

$$\Omega = \frac{[Ca^{2+}][CO_3{}^{2-}]}{K'_{sp}}$$

3.5.1 Why not a single reversible reaction arrow?

For simple minerals like iron monosulfide (FeS), Mantis supports the compact reversible syntax:

```
{
    "reaction_name": "FeS_precipitation_dissolution",
    "reaction": "Fe2_liq + HS -> FeS",
    "k_value_name": {
        "precipitation": "FeS_isp",
        "dissolution": "FeS_isd",
        "equilibrium": "FeS_sp",
    },
}
```

In that formulation, both reactants (Fe^{2+} and HS^-) are direct linear fractions of single transport variables (Fe_2*total* and TS_2), and their product $[Fe^{2+}][HS^-]$ directly forms the saturation state.

For $CaCO_3$, however:

1. **Nonlinear speciation**: $CO_3{}^{2-}$ is not a linear fraction of a single variable, but a nonlinear thermodynamic function of two master variables: DIC and TA.
2. **Asymmetric kinetic rate laws**: Precipitation is typically independent of the mineral mass ($R_{precip} = k_{precip} \cdot \max(\Omega - 1, 0)$, i.e., zero-order in $CaCO_3$), whereas dissolution depends directly on the reactive surface area/abundance of the solid mineral phase ($R_{diss} = k_{diss} \cdot [CaCO_3] \cdot \max(1 - \Omega, 0)$, i.e., first-order in $CaCO_3$).
3. **Mass-balance coupling**: Writing the reaction in terms of transport state variables requires coupling to Ca^{2+}, DIC, and $2 \times$ TA.

3.5.2 How to implement $CaCO_3$ precipitation and dissolution

To correctly model both processes, define saturation limiters in the `limiters` dictionary, and separate the forward precipitation and dissolution reactions in `reactions.py`:

```
limiters = {
    # Precipitation occurs when Omega > 1 (zero-order in solid)
    "calcite_precip_limiter": "np.maximum((c.Ca2 * c.CO3) / (mp.Ksp_calcite + 1e-30) - 1.0, 0.0)"

    # Dissolution occurs when Omega < 1 (first-order in solid Calcite)
    "calcite_diss_limiter": "np.maximum(1.0 - (c.Ca2 * c.CO3) / (mp.Ksp_calcite + 1e-30), 0.0)",
}

reactions = [
    # Precipitation: Ca2+ + CO3^2- -> CaCO3
    # Net in transport variables: Ca2 + DIC + 2 TA -> Calcite
    {
        "reaction_name": "calcite_precipitation",
        "reaction": "Ca2 + DIC + 2 TA -> Calcite",
        "k_value_name": "k_calcite_precip",
        "limiters": ["calcite_precip_limiter"],
        "reaction_order": 0,  # Purely k * limiter(Omega - 1)
    },

    # Dissolution: CaCO3 -> Ca2+ + CO3^2-
    # Net in transport variables: Calcite -> Ca2 + DIC + 2 TA
    {
        "reaction_name": "calcite_dissolution",
        "reaction": "Calcite -> Ca2 + DIC + 2 TA",
        "k_value_name": "k_calcite_diss",
        "limiters": ["calcite_diss_limiter"],
        "reaction_order": 1,  # First-order in Calcite: k * [Calcite] * limiter(1 - Omega)
    },
]
```

This ensures that:

- Ω evaluates using the exact thermodynamic speciation $c \cdot CO_3$ from the carbonate solver.
- The Jacobian and Newton solver correctly couple Ca^{2+}, DIC, and TA without nonphysical cross-multiplication of reactant concentrations.
- Zero-order precipitation and first-order mineral dissolution remain numerically stable and physically accurate across both undersaturated and supersaturated regimes.

A working example of a model with carbonate dissolution/precipitation reactions is in the examples folder (`examples/luff_2003`).