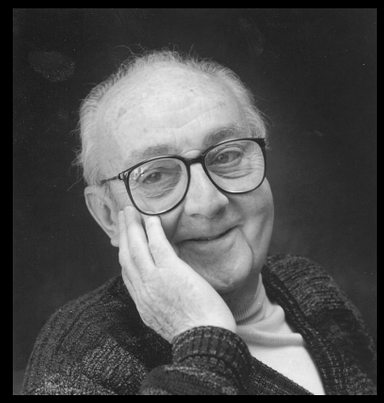
“All models are wrong”

“All models are wrong”

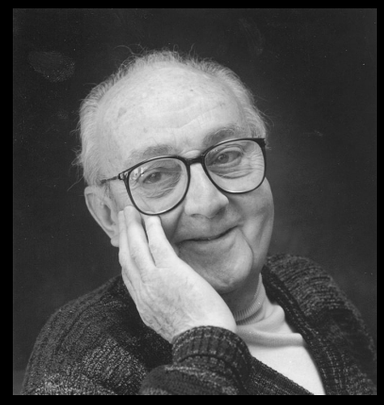
– George Box, in a 1976 paper published in the Journal of the American Statistical Association



George Box
1919—2013

“All models are wrong”

– *George Box, in a 1976 paper published in the Journal of the American Statistical Association*



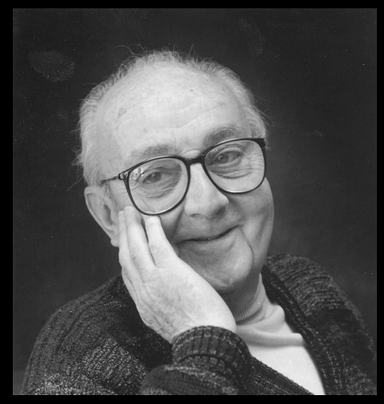
George Box
1919—2013

“All models are wrong, but some are useful”

– *George Box, in the proceedings of a 1978 statistics workshop*

“All models are wrong”

– *George Box, in a 1976 paper published in the Journal of the American Statistical Association*



George Box
1919—2013

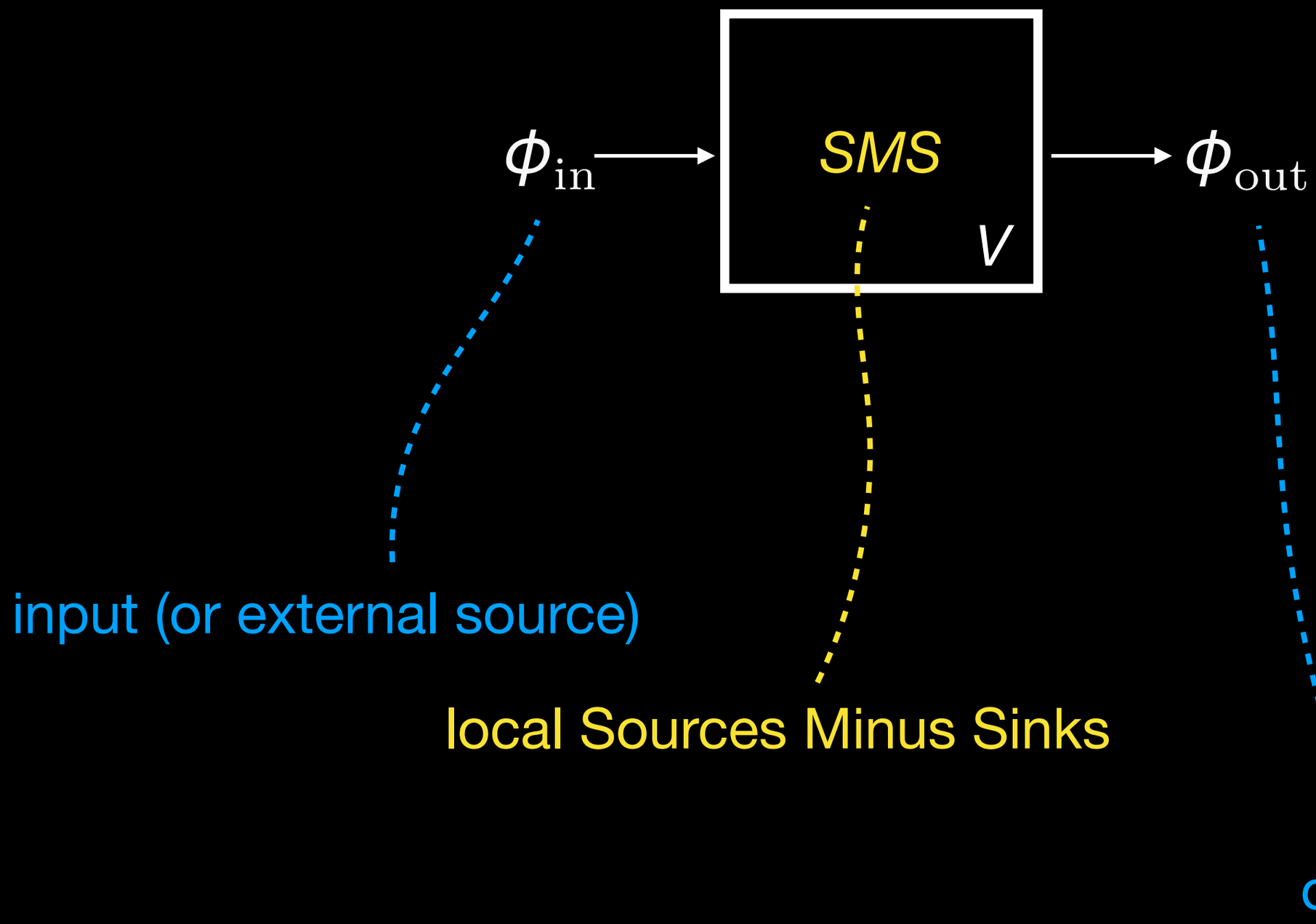
“All models are wrong, but some are useful”

– *George Box, in the proceedings of a 1978 statistics workshop*

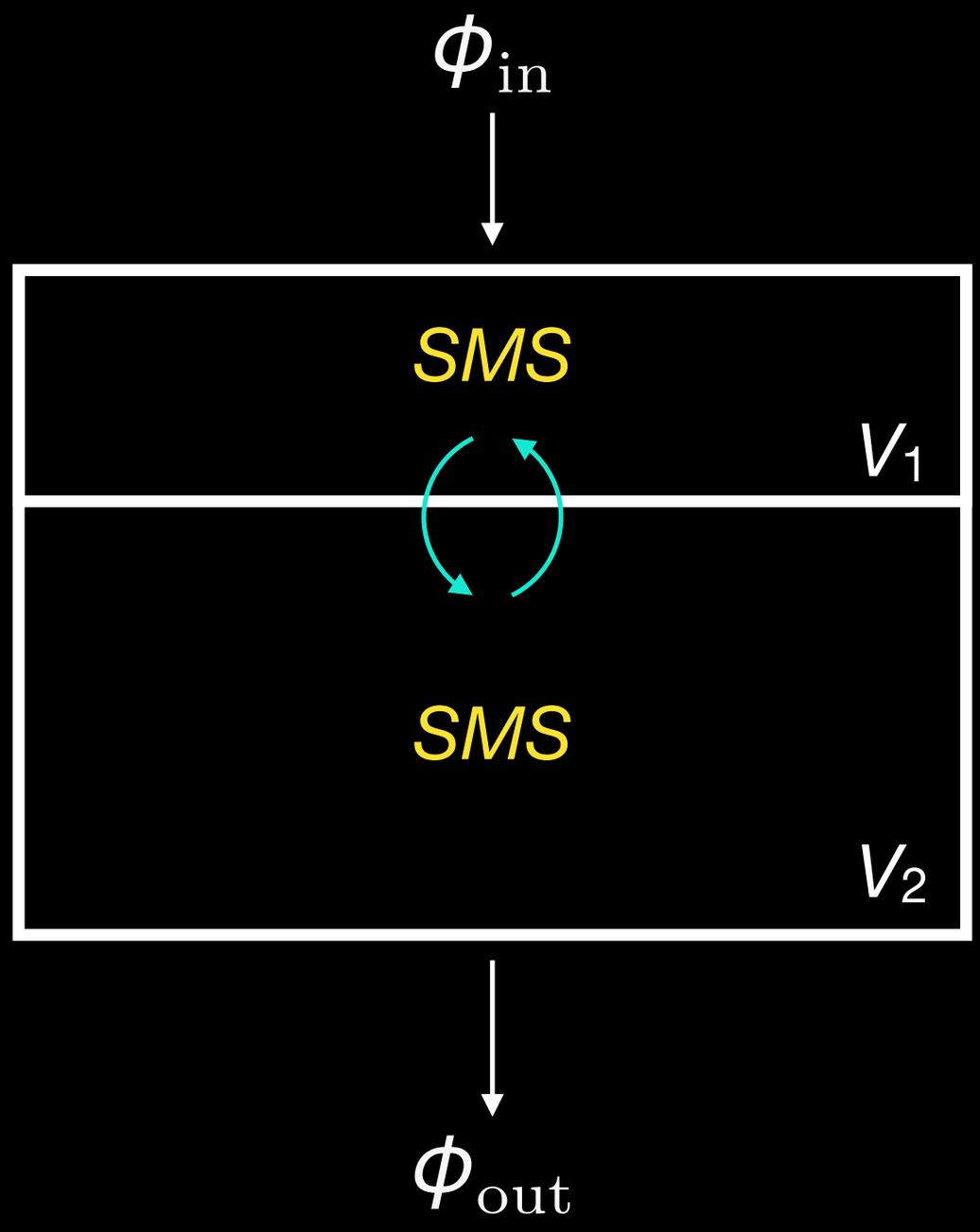
“Box models are useful despite being approximations”

– *Benoit Pasquier, in a 2018 presentation at the Francois Primeau group meeting*

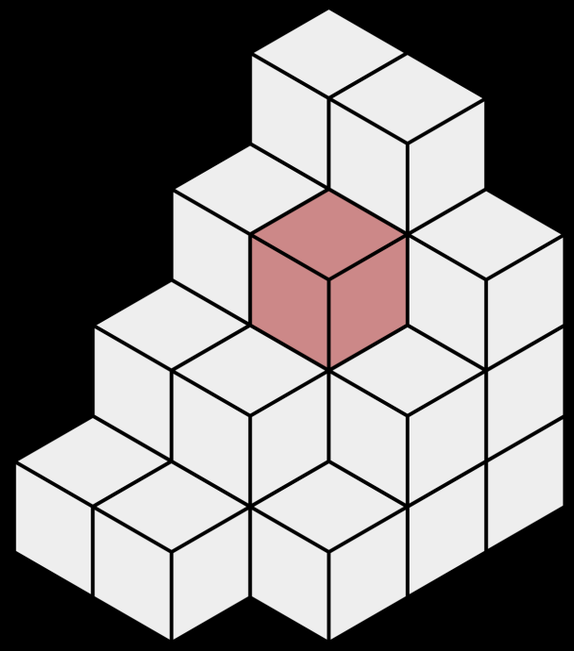
From a single-box model…



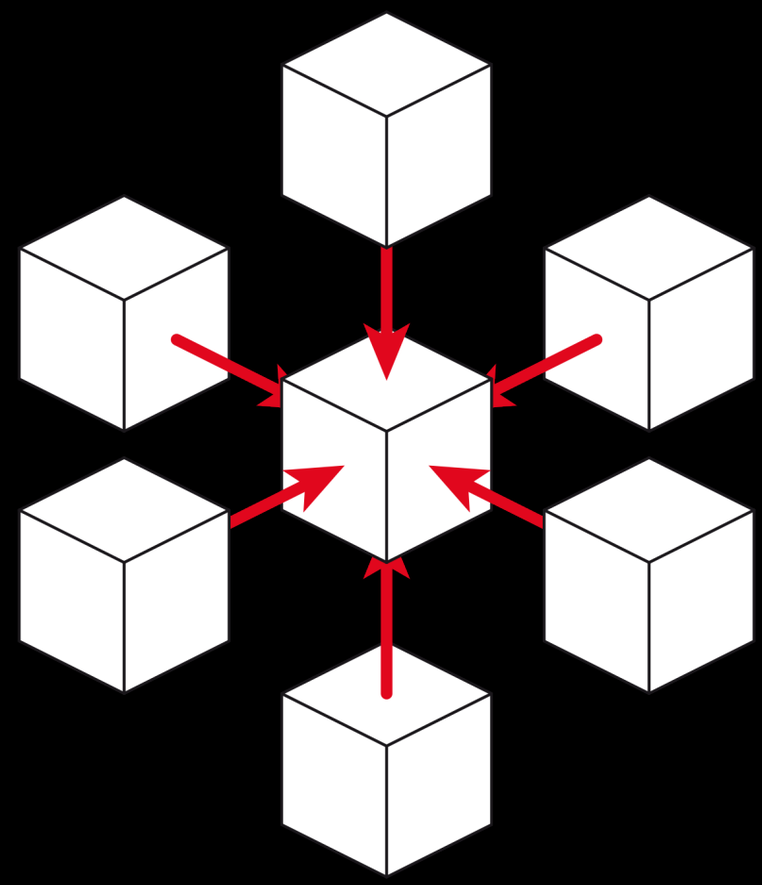
to a two-boxes model…



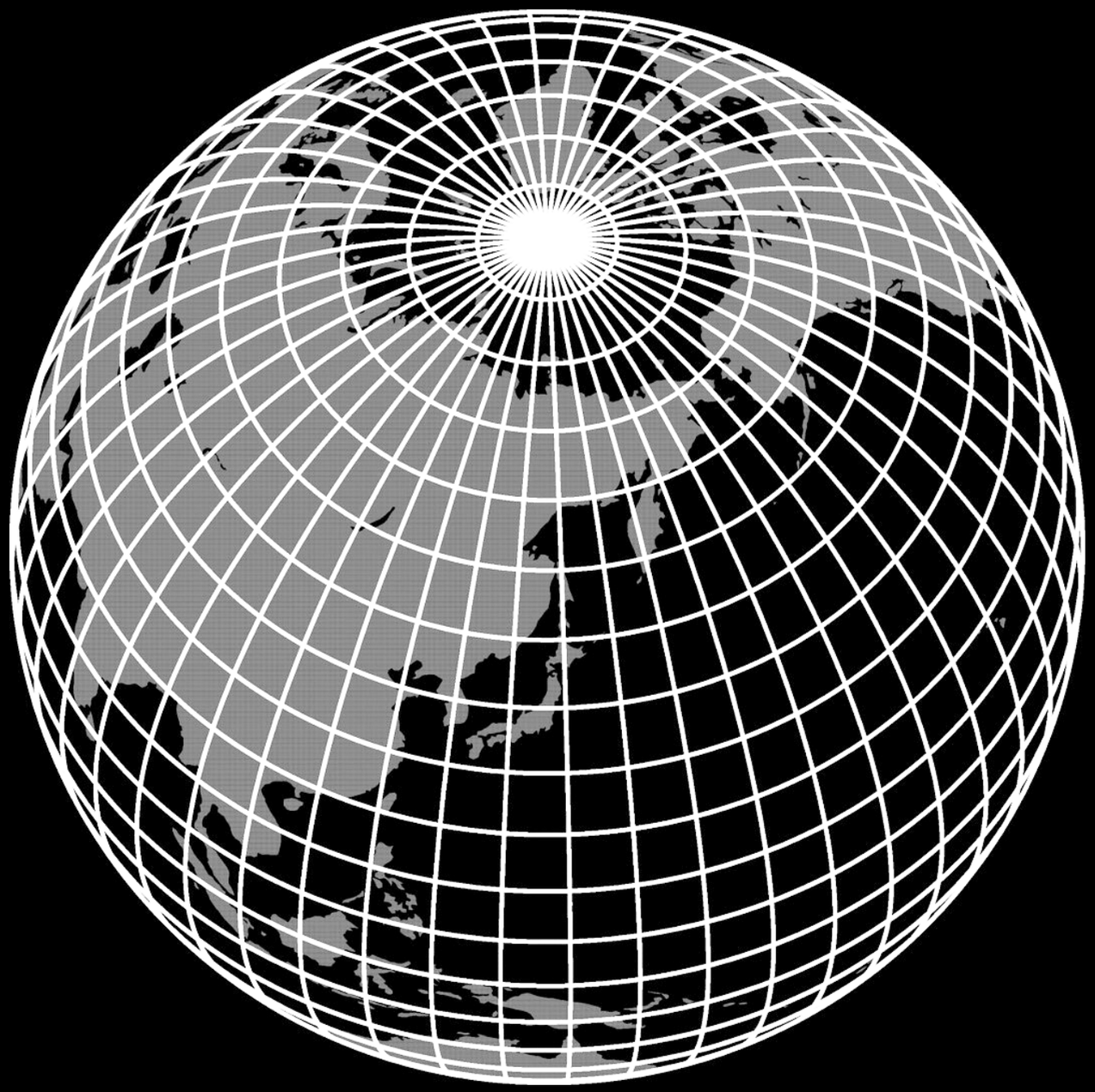
to 3D models with many boxes



where boxes exchange tracers with each other (not only neighbors).



We have global models that cover the entire earth with a grid…



Proof of the existence of such grids can be seen in the photograph below:

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

The ocean circulation is described by the transport operator, $\mathcal{T}$:

$$\mathcal{T} = \nabla \cdot \mathbf{u} - \nabla \cdot \mathbf{K} \nabla$$

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

The ocean circulation is described by the transport operator, $\mathcal{T}$:

$$\mathcal{T} = \boxed{\nabla \cdot \mathbf{u}} \boxed{- \nabla \cdot \mathbf{K} \nabla}$$

advection diffusion

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

The ocean circulation is described by the transport operator, $\mathcal{T}$:

$$\mathcal{T} = \boxed{\nabla \cdot \mathbf{u}} \boxed{- \nabla \cdot \mathbf{K} \nabla}$$

advection diffusion

We extensively use $\mathcal{T}$ in the tracer equation

$$\left(\frac{\partial}{\partial t} + \mathcal{T} \right) x = \boxed{SMS(x)}$$

local Sources Minus Sinks

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

The ocean circulation is described by the transport operator, $\mathcal{T}$:

$$\mathcal{T} = \boxed{\nabla \cdot \mathbf{u}} \boxed{- \nabla \cdot \mathbf{K} \nabla}$$

advection diffusion

We extensively use $\mathcal{T}$ in the tracer equation

$$\left(\frac{\partial}{\partial t} + \mathcal{T} \right) x = \boxed{SMS(x)}$$

local Sources Minus Sinks

OCIM provide a matrix $\mathbf{T}$ that corresponds to the discrete version of $\mathcal{T}$

$$\mathcal{T} \longleftrightarrow \mathbf{T}$$

More seriously, the awesome OCIM (Ocean Circulation Inverse Model) provide the advective-diffusive transport on such global 3D grids.

The ocean circulation is described by the transport operator, $\mathcal{T}$:

$$\mathcal{T} = \boxed{\nabla \cdot \mathbf{u}} \boxed{- \nabla \cdot \mathbf{K} \nabla}$$

advection diffusion

We extensively use $\mathcal{T}$ in the tracer equation

$$\left(\frac{\partial}{\partial t} + \mathcal{T} \right) x = \boxed{SMS(x)}$$

local Sources Minus Sinks

OCIM provide a matrix $\mathbf{T}$ that corresponds to the discrete version of $\mathcal{T}$

continuous advection and diffusion $\boxed{\mathcal{T}} \longleftrightarrow \boxed{\mathbf{T}}$ OCIM transport matrix

We thus have a discrete version of the tracer equation:

$$\left(\frac{\partial}{\partial t} + \mathbf{T}\right)\mathbf{x} = \mathbf{SMS}(\mathbf{x})$$

which we can rearrange into

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{f}(\mathbf{x})$$

where $\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + \mathbf{SMS}(\mathbf{x})$

Because I am looking for a useful (although wrong) model,
I assume steady state, so that I am looking for $\mathbf{x}$ such that

$$\mathbf{f}(\mathbf{x}) = 0$$

Why is OCIM awesome?

$$\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + \mathbf{SMS}(\mathbf{x})$$

so if $\mathbf{SMS}(\mathbf{x})$ is affine, then $\mathbf{f}$ is affine…

af·fine
/əˈfīn,ˈafīn/

adjective MATHEMATICS

1. allowing for or preserving parallel relationships.

noun ANTHROPOLOGY

1. a relative by marriage.

$$\mathbf{f}(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{b}$$

And if $\mathbf{f}$ is affine, then finding $\mathbf{f}(\mathbf{x}) = 0$ is very easy:

$$\mathbf{x} = -\mathbf{M}^{-1}\mathbf{b}$$ or `x = - M \ b ;` in MATLAB

Example: the ideal mean age of water, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + \boxed{1} - \boxed{\mathbf{D}_s\mathbf{x}/\tau}$$

advection and diffusion

increasing age (one second per second)

The age at the surface is quickly relaxed to zero

$\mathbf{D}_s$ is a diagonal matrix of 1's in the surface only

and τ is a fast restoring timescale (e.g., $\tau = 1$s).

The solution is $\mathbf{x} = -\mathbf{M}^{-1}\mathbf{b}$

where $\mathbf{M} = -\mathbf{T} - \mathbf{D}_s/\tau$

Computation time: a few seconds on my laptop!

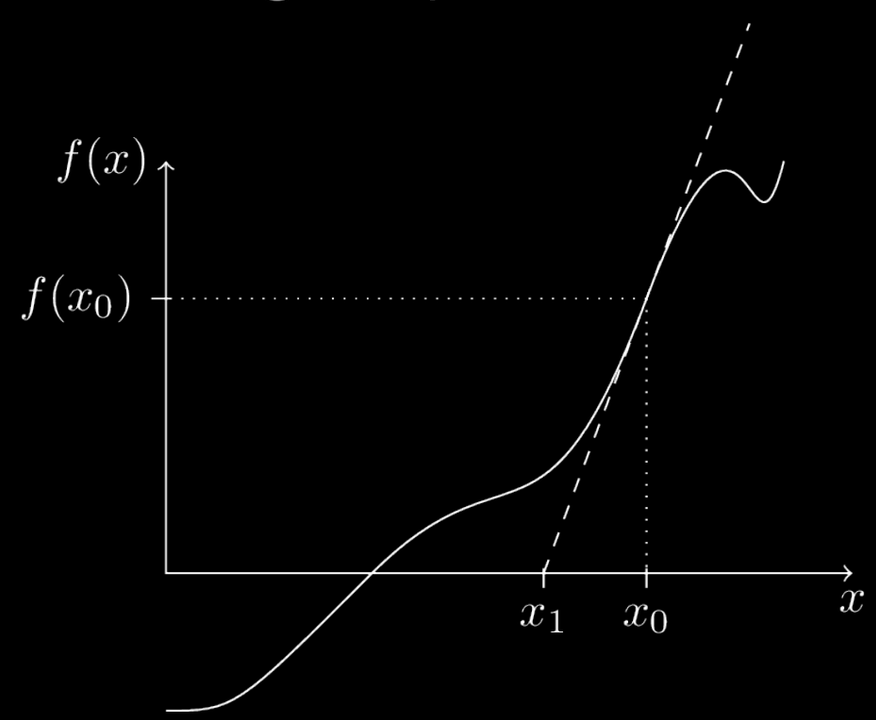
Why is OCIM awesome?

$$\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + \mathbf{SMS}(\mathbf{x})$$

If $\mathbf{SMS}(\mathbf{x})$ is not affine, then $\mathbf{f}$ is not affine either
(most people say that $\mathbf{f}$ is nonlinear - including me)

1643—1727

But we can use Newton's method!



$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n)\right]^{-1} \mathbf{f}(\mathbf{x}_n)$$

because we want $\mathbf{f}(\mathbf{x}_{n+1}) = 0$ and because

$$\mathbf{f}(\mathbf{x}_{n+1}) \simeq \mathbf{f}(\mathbf{x}_n) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n)\right](\mathbf{x}_{n+1} - \mathbf{x}_n)$$

Example: the silicate concentration, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + (\mathbf{S} - \mathbf{1})(\mathbf{x} - \mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\boldsymbol{\tau}_{\mathrm{res}} - \left(\mathbf{x} - \overline{\mathbf{x}}^{\mathrm{obs}}\right)/\boldsymbol{\tau}_{\mathrm{geo}}$$

Example: the silicate concentration, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + (\mathbf{S}-\mathbf{1})(\mathbf{x}-\mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\boldsymbol{\tau}_{\mathrm{res}} - \left(\mathbf{x}-\overline{\mathbf{x}}^{\mathrm{obs}}\right)/\boldsymbol{\tau}_{\mathrm{geo}}$$

Advection and diffusion

Example: the silicate concentration, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + \boxed{(\mathbf{S}-1)(\mathbf{x}-\mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}}} - \left(\mathbf{x}-\overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}$$

Advection and diffusion

Biological uptake and remineralization at depth,
where $\mathbf{S}$ is the particle flux divergence.
$\mathbf{x}^{\mathrm{obs}}$ is the observed $Si(OH)_4$ concentration
$\tau_{\mathrm{res}} \simeq 30$ days
This term reproduces the biological pump mechanism.

Example: the silicate concentration, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + \boxed{(\mathbf{S}-1)(\mathbf{x}-\mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}}} - \boxed{\left(\mathbf{x}-\overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}}$$

Advection and diffusion

Biological uptake and remineralization at depth,
where $\mathbf{S}$ is the particle flux divergence.
$\mathbf{x}^{\mathrm{obs}}$ is the observed $Si(OH)_4$ concentration
$\tau_{\mathrm{res}} \simeq 30$ days
This term reproduces the biological pump mechanism.

The geological restoring term,
which slowly restores the global inventory.
$\tau_{\mathrm{geo}} = 10^6$ years
$\overline{\mathbf{x}}^{\mathrm{obs}} \simeq 92$ μM is the mean observed $Si(OH)_4$ concentration

Example: the silicate concentration, $\mathbf{x}$.

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + \boxed{(\mathbf{S}-1)(\mathbf{x}-\mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}}} - \boxed{\left(\mathbf{x}-\overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}}$$

Advection and diffusion

Biological uptake and remineralization at depth,
where $\mathbf{S}$ is the particle flux divergence.
$\mathbf{x}^{\mathrm{obs}}$ is the observed $Si(OH)_4$ concentration
$\tau_{\mathrm{res}} \simeq 30$ days
This term reproduces the biological pump mechanism.

The geological restoring term,
which slowly restores the global inventory.
$\tau_{\mathrm{geo}} = 10^6$ years
$\overline{\mathbf{x}}^{\mathrm{obs}} \simeq 92$ μM is the mean observed $Si(OH)_4$ concentration

Computation time: a few minutes on my laptop!

Why is OCIM awesome?

Because OCIM affords very fast computations!

OK… But is it useful for anything in particular?

Yes: OCIM allows parameter optimization!

Wait a minute… What parameters?

Optimization example: the Si cycle

$$\mathbf{f}(\mathbf{x}) = \boxed{-\mathbf{T}\mathbf{x}} + \boxed{(\mathbf{S}-\mathbf{1})(\mathbf{x}-\mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}}} - \boxed{\left(\mathbf{x}-\overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}}$$

Advection and diffusion

Biological pump

Geological restoring

Optimization example: the Si cycle

$$\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + (\mathbf{S} - \mathbf{1})(\mathbf{x} - \mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}} - \left(\mathbf{x} - \overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}$$

Advection and diffusion

Biological pump

Geological restoring

There are some optimizable parameters, $\mathbf{p} = (p_1, \ldots, p_n)$.
So now $\mathbf{f}(\mathbf{p}, \mathbf{x})$ depends on those parameters.

Optimization example: the Si cycle

$$\mathbf{f}(\mathbf{x}) = -\mathbf{T}\mathbf{x} + (\mathbf{S} - \mathbf{1})(\mathbf{x} - \mathbf{x}^{\mathrm{obs}})^{+}_{\mathrm{eup}}/\tau_{\mathrm{res}} - \left(\mathbf{x} - \overline{\mathbf{x}}^{\mathrm{obs}}\right)/\tau_{\mathrm{geo}}$$

Advection and diffusion

Biological pump

Geological restoring

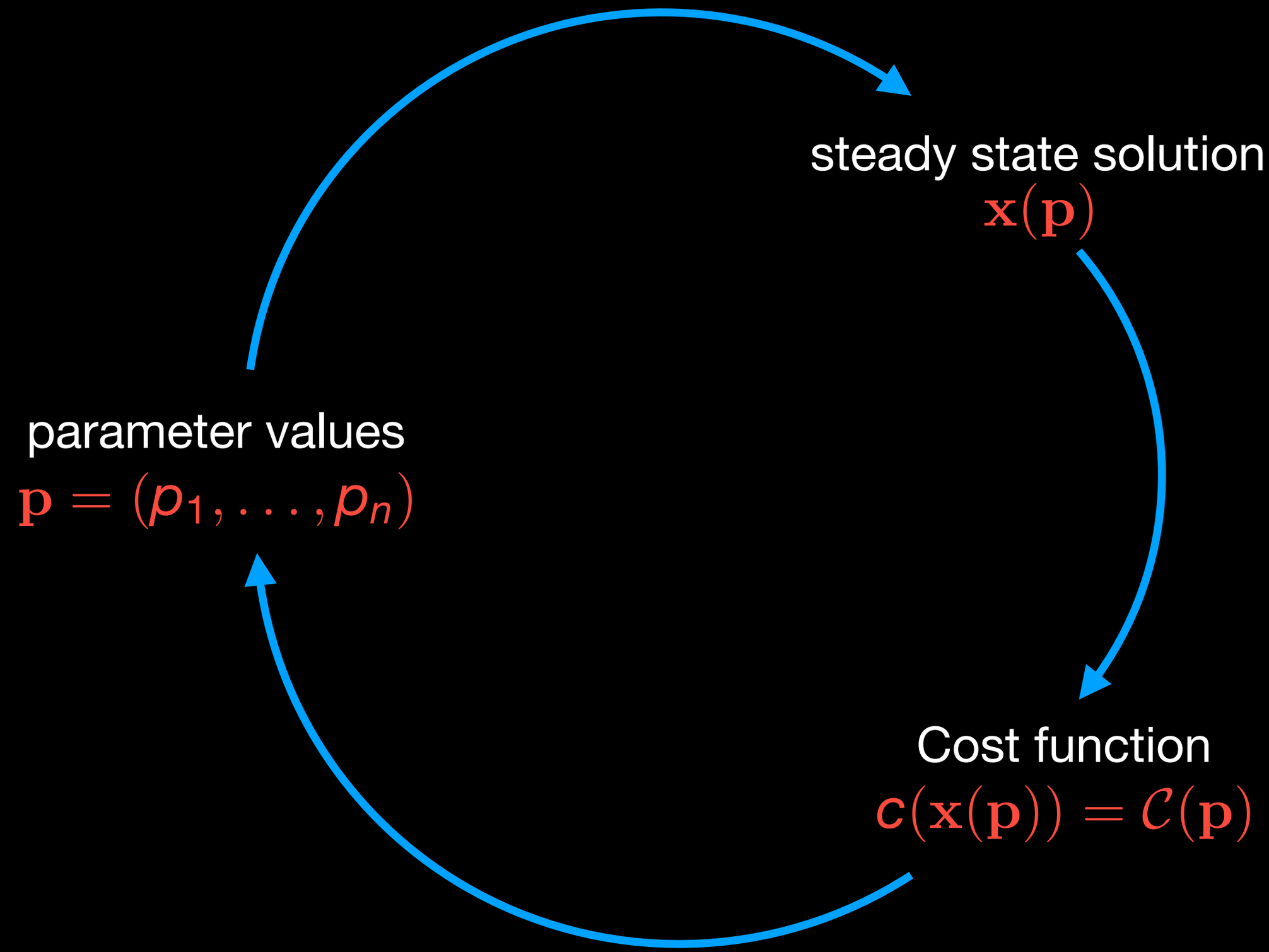
There are some optimizable parameters, $\mathbf{p} = (p_1, \ldots, p_n)$.
So now $\mathbf{f}(\mathbf{p}, \mathbf{x})$ depends on those parameters.

We want to minimize the error of our estimate when we compare it to observations, so we build a cost function that represents this error by a scalar:

$$c(\mathbf{x}) = \delta\mathbf{x}^{\mathsf{T}}\,\mathbf{V}\,\delta\mathbf{x} \longleftrightarrow \int d^3\mathbf{r}\,\left(\chi^{\mathrm{mod}} - \chi^{\mathrm{obs}}\right)^2$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^{\mathrm{obs}}$ is the mismatch with observations, and $\mathbf{V}$ is a diagonal matrix of the volumes of the grid boxes.

Optimization example: the Si cycle



Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

steady state solution
$\mathbf{x}(\mathbf{p})$

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

steady state solution
$\mathbf{x}(\mathbf{p})$

Compare to observations

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

steady state solution
$\mathbf{x}(\mathbf{p})$

Compare to observations

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

Optimize parameters
(e.g., `fminunc.m`)

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

steady state solution
$\mathbf{x}(\mathbf{p})$

Compare to observations

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

Optimize parameters
(e.g., `fminunc.m`)

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

This estimate depends on the parameters

steady state solution
$\mathbf{x}(\mathbf{p})$

Compare to observations

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

Optimize parameters
(e.g., `fminunc.m`)

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

Optimization example: the Si cycle

Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0$
e.g., using Newton's method
(`nsold.m`)

This estimate depends on the parameters

steady state solution
$\mathbf{x}(\mathbf{p})$

Compare to observations

Cost function
$c(\mathbf{x}(\mathbf{p})) = \mathcal{C}(\mathbf{p})$

This is the cost function used by the optimizer

Optimize parameters
(e.g., `fminunc.m`)

parameter values
$\mathbf{p} = (p_1, \ldots, p_n)$

Some notes on Newton's method

Newton's method requires the computation of the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$.

This can be done in many ways:

Analytically:

- that is, write the code for the Jacobian (fast and accurate)

numerically:

- using finite differences (slow and inaccurate)
- using the complex step method (fast, accurate)
- using algorithmic differentiation (fast, accurate)

Finite differences, CSD, and AD, example with Jacobian

Finite differences (what MATLAB uses internally)

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \simeq \frac{\mathbf{f}(\mathbf{x} + \boldsymbol{\varepsilon}\mathbf{I}) - \mathbf{x}}{\boldsymbol{\varepsilon}}$$

(Abuse of notation here)

Complex Step Differentiation

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \simeq \frac{\Im\left[\mathbf{f}(\mathbf{x} + i\boldsymbol{\varepsilon}\mathbf{I})\right]}{\boldsymbol{\varepsilon}}$$

Algorithmic Differentiation (A.K.A. automatic differentiation)

No formula, but calculates the derivatives at every line of code that defines $\mathbf{f}$ (using the composition rule recursively).

Some notes on the optimization

The optimization of the parameters, $\mathbf{p}$ (e.g, using MATLAB's `fminunc`), can go faster if you provide (I think):

- the Jacobian of $\mathcal{C}$ with respect to $\mathbf{p}$
- the Hessian of $\mathcal{C}$ with respect to $\mathbf{p}$

This can also be done (I think) in the ways mentioned before:

Analytically:

- that is, write the code for the Jacobian (fast and accurate)

numerically:

- using finite differences (slow and inaccurate)
- using the complex step method (fast, accurate)
- using algorithmic differentiation (fast, accurate)

Some more notes on the optimization

One can derive the Jacobian from the steady state equation…

$$\mathbf{f}\left(\mathbf{p}, \mathbf{x}(\mathbf{p})\right) = 0 \quad \Longrightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{p}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}} = 0$$

Use the composition rule:

$$\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\mathbf{p}} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{p}}\Big[\mathbf{c}\big(\mathbf{x}(\mathbf{p})\big)\Big] = \frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}}$$

And inject in the final result:

$$\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\mathbf{p}} = -\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{p}}$$

And even more notes on the optimization

One can derive the Hessian from the steady state solution too!
It is a bit more complicated, especially notation-wise… So this is merely an attempt to tackle it, and François may be able to explain it to us!

The composition rule for Hessian can be written as

$$\frac{\mathrm{d}^2\mathcal{C}}{\mathrm{d}\mathbf{p}^2} = \frac{\mathrm{d}^2\boldsymbol{c}}{\mathrm{d}\mathbf{x}^2}\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}} \otimes \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}}\right) + \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\mathbf{x}}\boxed{\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}\mathbf{p}^2}}$$

This one might be tough

$$\frac{\mathrm{d}^2\boldsymbol{c}}{\mathrm{d}\mathbf{x}^2} = 2\mathbf{V}$$

Already explained (easy!)

$$\frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\mathbf{x}} = 2(\mathbf{x} - \mathbf{x}^{\mathrm{obs}})^{\mathsf{T}}\mathbf{V}$$

And even more notes on the optimization (continued)

size: $n_{\mathbf{x}} \times n_{\mathbf{p}} \times n_{\mathbf{p}}$

$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}\mathbf{p}^2}$ can be found from something like this

$$\mathbf{f}\left(\mathbf{p}, \mathbf{x}(\mathbf{p})\right) = 0 \implies \frac{\partial^2 \mathbf{f}}{\partial \mathbf{p}^2} + \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}^2}\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}} \otimes \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}}\right) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}\mathbf{p}^2} = 0$$

Already explained (easy!)

Jacobian for Newton

Could be full but smallish
$n_{\mathbf{x}} \times n_{\mathbf{p}} \times n_{\mathbf{p}}$

Is huge but should be sparse!
$n_{\mathbf{x}} \times n_{\mathbf{x}} \times n_{\mathbf{x}}$
(would be zero anyway in my example model)

But be reassured…
You do not have to go the analytical way!

fortuitous pun

Although it can probably save you a lot of time in the long run.

You can use the usual finite differences and let MATLAB's built-in functions (`fminsearch`, `fminunc`, etc.) do all the work.

You can use the complex step differentiation (CSD), but there is a bit of work and some caveats (but I love it - much elegance).

And you can probably use the algorithmic differentiation (AD), but I have not tried it myself, therefore I cannot recommend it.