Pullback / VJP rule for Linear system solving

$$f(\underline{A}, \underline{b}) = \{\text{solve } \underline{A}\,\underline{x} = \underline{b} \text{ for } \underline{x}\} =: \underline{x}$$

- Numerical solution to PDEs
- Optimization methods
  $\vdots$

e.g. ... direct methods
   (LU, cholesky ...)
 · iterative methods
   (CG, GMRES, Multigrid...)

$\underline{A} \in \mathbb{R}^{N \times N}$
$\underline{b} \in \mathbb{R}^{N}$
$\underline{x} \in \mathbb{R}^{N}$

task: backward propagate cotangent information on the output $\bar{\underline{x}} \in \mathbb{R}^{N}$ to the inputs $\bar{\underline{A}} \in \mathbb{R}^{N \times N}$ & $\bar{\underline{b}} \in \mathbb{R}^{N}$

↳ without reverse-mode AD through the numerical solver (=unrolling)

in general

$$"\bar{\underline{A}} = \bar{\underline{x}}^{T} \frac{\partial f}{\partial \underline{A}}" \qquad "\bar{\underline{b}}^{T} = \bar{\underline{x}}^{T} \frac{\partial f}{\partial \underline{b}}"$$

↑ hmm. this is not a nice quantity

→ index notation

primal (forward): $A_{j\rho}\, x_\rho = b_i$

now the pullbacks:

$$\bar{A}_{k\ell} = \bar{x}_j \frac{\partial f_j}{\partial A_{k\ell}} = \bar{x}_j \boxed{\frac{\partial x_j}{\partial A_{k\ell}}} \quad ②$$

$$\bar{b}_{k} = \bar{x}_j \boxed{\frac{\partial f_j}{\partial b_k}} \quad ①$$

① $\dfrac{\partial x_\rho}{\partial b_k} \xrightarrow{\text{implicit differentiation}} \dfrac{\partial}{\partial b_k}(A_{j\rho}\, x_\rho) = \dfrac{\partial}{\partial b_k}(b_i)$

$$A_{j\rho} \frac{\partial x_\rho}{\partial b_k} = \delta_{ik}$$

$$\Leftrightarrow \quad \frac{\partial x_j}{\partial b_k} = A_{ij}^{-1} \delta_{ik}$$
$$= A_{kj}^{-1}$$

② $\dfrac{\partial x_j}{\partial A_{k\ell}} \xrightarrow{\text{implicit differentiation}} \dfrac{\partial}{\partial A_{k\ell}}(A_{j\rho}\, x_\rho) = \dfrac{\partial}{\partial A_{k\ell}}(b_i)$

$$\frac{\partial A_{j\rho}}{\partial A_{k\ell}} x_\rho + A_{j\rho} \frac{\partial x_\rho}{\partial A_{k\ell}} = 0$$

$$\delta_{ik}\delta_{\rho\ell} x_\rho + A_{j\rho} \frac{\partial x_\rho}{\partial A_{k\ell}} = 0$$

$$\Leftrightarrow \quad \frac{\partial x_\rho}{\partial A_{k\ell}} = -A_{ij}^{-1} \delta_{ik}\delta_{\rho\ell} x_j$$

$$\frac{\partial x_j}{\partial A_{k\ell}} = -A_{kj}^{-1} x_\ell$$

plug back in

① $\bar{b}_{k} = \bar{x}_j \dfrac{\partial x_j}{\partial b_k} = \bar{x}_j A_{kj}^{-1}$

$$\Leftrightarrow \quad A_{jk}\, \bar{b}_k = \bar{x}_j$$

② $\bar{A}_{k\ell} = -\bar{x}_j A_{kj}^{-1} x_\ell$

$$= -\bar{b}_k\, x_\ell$$

back to symbolic notation

$$\underline{\lambda} = \{\text{solve } \underline{A}^{T}\underline{\lambda} = \bar{\underline{x}} \text{ for } \underline{\lambda}\}$$

λ ... is the adjoint variable $\in \mathbb{R}^N$

$$\bar{\underline{b}} = \underline{\lambda}$$

$$\bar{\underline{A}} = -\underline{\lambda}\,\underline{x}^{T} \quad \text{outer product}$$

Full pullback rule

$$\boxed{B\big(f,(\underline{A},\underline{b}),(\bar{\underline{x}},)\big) = \big(\underbrace{(\{f(\underline{A},\underline{b})\}}_{\underline{x}}),(\underbrace{-\underline{\lambda}\,\underline{x}^{T}}_{\bar{\underline{A}}},\underbrace{\underline{\lambda}}_{\bar{\underline{b}}})\big)}$$

Alternatively

this is not sparse (we need sparse outer products)

$$\boxed{B\big(f,(\underline{A},\underline{b}),(\bar{\underline{x}},)\big) = \big((\underbrace{\{\underline{A}^{-1}\underline{b}\}}_{\underline{x}}),(\underbrace{-\bar{\underline{b}}\,\underline{x}^{T}}_{\bar{\underline{A}}},\underbrace{\underline{A}^{-T}\bar{\underline{x}}}_{\bar{\underline{b}}}))}$$

requires another linear system solve, but with the transposed primal matrix, i.e. $\underline{A}^{T} \in \mathbb{R}^{N \times N}$

↳ if primal pass was done with direct solver, we could re-use its factorization in the adjoint form

↳ or use adjoint form of preconditioner for an iterative adjoint solve