Mapeador de Grade — Eta · Relatório Técnico de Desenvolvimento

Ferramenta: `eta-grid-mapper.html` (aplicação web autocontida) **Localização:** `eta-v1.5.0-desenv/util/eta-grid-mapper/eta-grid-mapper.html` **Publicação (Artifact, mesma URL em todos os redeploys):** https://claude.ai/code/artifact/231d81da-70c5-4c6f-bec2-11bfedfa6888 **Período de desenvolvimento:** 2026-07-04 a 2026-09-02 **Tamanho atual:** ≈ 4,3 MB (inclui ~3,9 MB de linha de costa GSHHS embutida) · ≈ 5 100 linhas **Público deste relatório:** desenvolvedores e modelos de IA que forem dar continuidade ao projeto.

---

1. Visão geral e arquitetura

Ferramenta interativa para navegar, inspecionar, editar, comparar e **verificar a consistência** da grade e dos campos fixos (topografia, máscara oceano/continente, vegetação, solo e campos genéricos — inclusive séries multi-registro animáveis) do modelo regional **Eta** (CPTEC/INPE).

O painel lateral direito é organizado em **4 grupos** com separadores `.sgrp` (2026-07-10): **Dados** (Carregar dados), **Inspeção** (Consulta unificada, Ponto selecionado, Legenda), **Edição** (Editar/salvar, Máscara por shape) e **Análise** (Comparar B–A, Consistência topo × máscara).

Decisões de arquitetura:

- **Arquivo único HTML + CSS + JavaScript puro, sem dependências externas.** Motivo: a página é publicada como *Artifact* (claude.ai), cujo CSP bloqueia qualquer requisição externa (CDN, fontes, tiles). Tudo — inclusive a linha de costa — está embutido no arquivo. Também roda offline abrindo o arquivo local no navegador (o usuário usa `wsl.localhost/...` no Chrome do Windows).
- **Renderização híbrida:** um `<canvas id="topo">` por baixo (campos raster) e um `<svg id="map">` por cima (domínio, grades, costa, shapes, marcadores). O zoom/pan é feito pelo `viewBox` do SVG; o canvas é redesenhado por `drawTopoView()`.
- **Fonte única de verdade das convenções:** `atmos/src/etafcst/` — `corners.f90`, `const.f` e `MPPINIT` (decomposição). A ferramenta replica essas fórmulas em JS. Companheiro de linha de comando com a mesma matemática: `util/eta-grid-mapper/mapll.sh`.
- **Iteração/redeploy:** editar o HTML e chamar a ferramenta *Artifact* passando o `file_path` e a URL acima (redeploy na mesma URL). Validação mínima antes de redeployar (ver §14).

2. Convenções da grade E de Arakawa

Parâmetros: `IM, JM, DLMD, DPHD, TLM0D, TPH0D` (lidos do namelist `&MODEL_GRIDS` do ETAIN).

Derivados:

```
WBD = -(IM-1)·DLMD                  SBD = -(JM-1)/2·DPHD
rlon(I,J) = WBD + (I-1)·2·DLMD + mod(J+1,2)·DLMD     (linhas alternadas deslocadas de DLMD)
rlat(J)   = SBD + (J-1)·DPHD
```

Rotação (lat/lon geográficas ⇄ rotacionadas), com `t0 = TPH0D`:

```
lat = asin( sin(t0)·cos(rlat)·cos(rlon) + cos(t0)·sin(rlat) )
lon = TLM0D + atan2( cos(rlat)·sin(rlon),
                     cos(t0)·cos(rlat)·cos(rlon) − sin(t0)·sin(rlat) )
```

A inversa (`geoToRot`) é feita por rotação vetorial 3-D (funções `rotToGeo`/`geoToRot`, validadas contra o `mapll.sh`).

2.1 Célula = losango (Voronoi da grade E) — convenção vigente

Desde 2026-07-09 **cada célula é um losango** com vértices em `(rlon±DLMD, rlat)` e `(rlon, rlat±DPHD)` — o mosaico de Voronoi correto dos pontos de massa (H) da grade E: vizinhos se tocam exatamente no ponto médio, **sem folga nem sobreposição** (o vértice E de `(I,J)` é o vértice W de `(I+1,J)`; os vértices N/S caem sobre as linhas `J±1`). *Esta convenção substitui a retangular antiga (±DLMD × ±DPHD/2) usada até 2026-07-08.*

Mapeamento inverso `rotToGrid` (ponto arbitrário ⟶ célula que o contém):

```
u = (rlon−WBD)/DLMD ;  v = (rlat−SBD)/DPHD        # massas em (u,v) inteiros com
u+v PAR
a = round((u+v)/2) ;  b = round((u−v)/2)            # rede girada 45°: massa =
(a,b) inteiro
J = a−b+1 ;  I = (a+b−mod(J+1,2))/2 + 1
```

Validação numérica: *round-trip* exato em toda a grade; 200 000 pontos aleatórios sempre atribuídos ao losango que os contém (métrica `|du|/DLMD+|dv|/DPHD ≤ 1`), 0 casos não-ótimos.

Tudo que depende de "célula" usa `rotToGrid`: clique/seleção, raster (`bakeField`), leitura do cursor, máscara por shape, destaque (`drawMarker`) e o desenho célula-a-célula (`drawTopoView`).

3. Decomposição de processadores (MPPINIT)

```
ipe = MYPE % INPES ;  jpe = MYPE / INPES
ICHUNK = IM/INPES ; ITAIL = IM%INPES   (os ITAIL primeiros blocos ganham +1 coluna)
JCHUNK = JM/JNPES ; JTAIL = JM%JNPES   (idem em J)
```

Implementado em `readParams()` (tabelas `P.IB`/`P.JB`), `globalToPe()` e `peToGlobal()`. Default da interface: `ams_08km` — 735×1441, 30×60 = 1800 PEs, centro (−55, −19), 8 km.

4. Formatos de arquivo (Fortran *unformatted*, sequencial)

Arquivo	Conteúdo	Registro
`etatopo.dat`	`write(1) hgt, sm`	marcador(int4=2·IM·JM·4) + 2× real*4(IM,JM) + marcador
`etaveg.dat` / `etasolo.dat`	`write(31) v`	marcador(int4=IM· JM· 4) + int*4(IM,JM) + marcador
genérico ("Outro")	1, 2 campos **ou série de N registros**	1º marcador ∈ {n· 4, 2n· 4, 40}; série = sequência `[nome char*40]? + campo(n· 4)` (leitura Fortran: `read(u) nome; read(u) campo`)
`deta` (níveis eta)	`write(u) deta(LM), ldum`	real*4(LM) + int*4 num mesmo registro (scan até o *denormal* do ldum)
`INIT.file` / `BNDY.file.*` / `CNST.file`	saída final do initbc lida pelo modelo	estrutura completa, deduções de LM/NSOIL/LB/JM e leitura POR FATIAS no **§12.3**

- **Endianness auto-detectada** pelo marcador de registro (little tentado primeiro) e **preservada ao salvar**.
- Tipo do campo genérico auto-detectado: int32 lido como float32 vira *denormal* — se >2 % dos valores são "anormais" ⟹ `int*4`, senão `real*4` (`parseGeneric`).
- Indexação: `idx = (J−1)·IM + (I−1)`, I mais rápido (column-major do Fortran com a primeira dimensão = I).
- **Multi-instância (2026-07-09)**: o carregamento **acumula** — cada arquivo vira uma entrada em `BANK[tipo]` (topo/veg/solo/other) e os globals `TOPO/VEG/SOLO/OTHER` são **ponteiros para a instância ativa** (`bankSet(tipo,i)`; seletores dinâmicos em `#instRows` quando há 2+ instâncias — `rebuildBankSelects`/`bankSwitch`). Os botões × removem só a instância ativa (a seguinte assume); a última esconde o tipo. Todo o restante do código (edição, máscara, diff, perfil, salvamentos) opera nos ponteiros e não precisou mudar — mesma técnica usada em `SHAPES`/`POINTS`/`POLYS`.
- Resolução (tabela do `def_eta.scr`): `dlmd = R·0.0072115384625`, `dphd = R·0.0067307692375` (R em km); `RES_TABLE` traz DT/NPHS/NCNVC/hydro por R.

4.1 Menu de carregamento + lote por diretório (2026-07-11)

- Os `filebtn` individuais foram movidos para um **dropdown** (`.ddwrap`/`.ddmenu`, botão `loadMenuBtn`; fecha em clique-fora ou após escolher arquivo). Os handlers de carga foram refatorados em funções por **buffer** — `loadEtainText`, `loadTopoBuf`, `loadCatBuf`, `loadOtherBuf`, `loadDetaBuf` (retornam sucesso; escrevem no `loadstat`) — e os inputs viraram wrappers finos.
- **Lote** (`f_dir`, `<input webkitdirectory multiple>`): filtra extensões não-Fortran e >500 MB (o teto NÃO vale p/ `INIT.file`/`BNDY.file`/`CNST.file`, lidos por fatias); `batchDetect(nome)`

classifica por regex (etain/init.file/bndy.file/cnst.file/deta/ topo/veg/solo|soil; deta testado antes de topo); caixa `#batchbox` lista até 60 arquivos com checkbox + seletor de tipo (corrigível, inclui "outro"/"ignorar") + tamanho. `batchLoad()` (async/await sobre `file.text()`/`arrayBuffer()`) carrega na ordem `BATCH_ORDER` (etain→topo→veg→solo→outro→deta — deta por último ajusta a topografia recém- carregada); ETAIN e deta limitados a 1 por lote; resumo ✓/✗ por item no status (✗ inclui a mensagem de erro do loader).

5. Estrutura do código (mapa para manutenção)

Seções do `<script>` na ordem, com globals e funções principais:

Seção	Globals	Funções-chave
Estado/camadas	`TOPO VEG SOLO OTHER DIFF POINTS POLYS activeKey layerOn undoStack`	`loadedLayers layerValAt activeAt fieldValNum cellColor`
Banco multi-instância	`BANK={topo[],veg[],solo[],other[]} bankIdx`	`bankSet bankSwitch rebuildBankSelects`
Paletas	`TOPO_STOPS CAT_COLORS PALETTES DIV_STOPS otherPalette…`	`terrainColor catColor genColor palAt divColor`
Parâmetros	`P`	`readParams`
Transformações	—	`gridToRot rotToGrid rotToGeo geoToRot globalToPe peToGlobal`
Projeção	`proj view VW=940 VH=780`	`buildProjection toXY toGeo gxy`
Costa	`COAST_B64 COAST_RINGS CQ=500 coastOn`	`decodeCoast drawCoast`
Desenho estático	—	`drawStatic drawMarker drawPEGrid drawPELabels drawGridNodes drawProfOverlay cellValTxt cellValLP`
Shapes	`SHAPE_RINGS shapeColor shapeMinCell POLY_KEEP`	`buildShapeRings drawShapes drawShapesThrottled computePolyKeep`
Seleção/consultas	`current`	`info select fillReadout showPopup query`
Zoom/pan	`view`	`applyView zoomAt clientToUser` + eventos mouse/touch
Raster	`topoBase`	`bakeField drawTopoView refreshTopo buildColorbar`
Parsers	—	`parseETAIN parseTopo parseCat parseGeneric parseShape parseShp parseDbf`
Carga (botões + lote)	`BATCH BATCH_TYPES BATCH_ORDER`	`loadEtainText loadTopoBuf loadCatBuf loadOtherBuf loadDetaBuf batchDetect batchLoad`
Máscara por shape	[illegible]	[illegible] `buildMaskRows` [illegible] `lvSplit` `pushVerifyLayer` `lvSaveDeta` `cloneInstance mkUseFilt` `lvExportPNG lvExportCSV`

Padrões do código: `$(id)` = `getElementById`; funções de desenho são idempotentes e redesenham grupos SVG nomeados (`#coastlayer`, `#shapelayer`, `#pegridlayer`…); mensagens de status via elementos `.loadstat` (`loadstat`, `editstat`, `mkStat`, `mkGenStat`, `dfStat`, `chkStat`).

6. Pipeline de renderização

- `drawStatic()` reconstrói o SVG (graticule, costa, shapes, domínio, grupos vazios) — chamado em mudanças estruturais, não a cada pan.
- `applyView()` roda a **cada** zoom/pan: seta `viewBox`, redesenha canvas (`drawTopoView`), costa, shapes (com *throttle*), grade de PEs, rótulos, nós e marcador.
- **Raster em dois regimes** (`drawTopoView`): com ≤ 20 000 células visíveis, cada célula é desenhada como **losango exato** (mesmos vértices do destaque ⟹ alinhamento perfeito; `fill+stroke` da mesma cor fecham juntas de anti-aliasing). Acima disso usa o raster "assado" por `bakeField()` (mapeamento inverso pixel⟶célula a ~1,3 px/célula no espaço da projeção; zoom/pan só recortam via `drawImage`).
- Traços SVG usam `vector-effect="non-scaling-stroke"` (aplicado por CSS na página e **por atributo** no clone serializado do PNG, pois o CSS não acompanha).
- Limite de zoom: `cellWproj` (maior célula do domínio na projeção) garante ~1 célula/tela no zoom máximo (~8000×).
- Grade de PEs e graticule são **SÓ sob demanda** (botões `PEs`/`gratic`) — o liga/desliga automático do graticule no zoom ≥ 3× foi removido em 2026-07-10 (o estado do botão persiste nas preferências, ver §13).

6.1 Isolinhas e bandas (2026-07-11, inspirado no SisMOM/GISELE)

- **Marching squares na sub-rede das linhas ÍMPARES** (J=1,3,5…): a grade E é escalonada, mas essa sub-rede é RETANGULAR em coords rotacionadas (2·DLMD × 2·DPHD), então o MS roda sem artefato de paridade (perda de metade da resolução N-S, irrelevante p/ isolinhas). Validado num cone sintético: erro máx. de posição 0,018° com célula de 1°.
- `computeContours`: níveis por intervalo do usuário (`ctInt`) ou automático (`niceStep`, ~8 níveis 1/2/5×10^k, teto 60); varredura célula-fora/níveis-dentro (só os níveis entre o mín/máx dos 4 cantos); tabela MS de 16 casos com interpolação linear nas arestas; cada vértice rot⟶geo⟶projeção; 1 `<path>` por nível (teto global 150 mil segmentos); rótulos opcionais a cada ~140 segmentos. Cache `CT={sig,svg,levels}` por assinatura (camada, `bakeRev`, intervalo, registro da série, grade); o SVG é estático no espaço de projeção — o `viewBox` cuida do zoom (sem redesenho por pan), injetado como `#ctlayer` no drawStatic.
- **Bandas** (`ctBand`): `cellColor` quantiza topo/outro-real nos níveis via `ctQuant` (busca binária; abaixo do 1º nível = mínimo) — *filled contour* no mesmo raster; `ctPrep()` no início do bakeField sincroniza flag e níveis.

7. Linha de costa real embutida (GSHHS alta resolução; antes NE 50m)

- **Payload original (2026-07-09, SUPERADO — ver o upgrade abaixo):** `COAST_B64` (~194 KB): 1421 anéis / 60 657 pontos de `ne_50m_land` (terra, com lagos como buracos), simplificados (tol 0,004°, descarte < 0,02°), **quantizados a 1/500°** (~110 m de erro) e codificados como **deltas zigzag-varint** em base64. Formato: `[nRings]{[nPts][lon0][lat0]([dlon][dlat])*}`.

- **Encoder** (para regenerar com outra tolerância/fonte): `scratchpad/encode_coast.js` — uso: `node encode_coast.js <tolGraus> <minBBox> <Q>`; colar o base64 no lugar de `COAST_B64`.
- **Decoder:** `decodeCoast()` (lazy, 1ª chamada de `drawCoast`).
- **Desenho (`drawCoast`)**: *culling* por bbox do anel × vista; **LOD por passo** (*stride* que alveja ~1,2 px entre vértices, via espaçamento médio pré-computado); um único `<path>` com `fill-rule="evenodd"` (preenche terra só quando nenhuma camada está ativa). **Espessura**: 1 px de tela até zoom 3× e afinando `max(0.25,(3/z)^0.6)` a partir daí. Botão `costa` liga/desliga (`coastOn`).

Upgrade 2026-07-18 — GSHHS h (fonte 1:250 000, a mesma do GMT): o payload passou de NE 50m (60 657 pts, CQ=500 ≈ 220 m) para **GSHHS_h L1+L2 (terra + lagos): 31 289 anéis / 1 294 887 pts, CQ=2000 ≈ 55 m**, ~3,9 MB de base64 (HTML total ≈ 4,2 MB). Filtro de ilhas REGIONAL: dentro da janela das Américas (lon −100..−25, lat −60..20, onde vivem os domínios Eta) mantém tudo ≥ 0,004° (~0,4 km); fora, ≥ 0,03° (~3,3 km). Simplificação radial tol 0,0004° (abaixo da resolução nominal de ~0,2 km do h) — constatação prática: simplificar quase NÃO reduz bytes (deltas maiores custam mais varint; o custo fica ~2,2–2,4 B/pt de qualquer forma), a alavanca real é derrubar anéis pequenos. Encoder novo em python (`encode_coast.py` no scratchpad: parser .shp puro, simplify radial, zigzag varint) + verificador (réplica do decodeCoast) + teste com o decoder REAL extraído do HTML em Node. Densidade medida: Baía de Guanabara 0,3°×0,4° passou de ~15–40 para **366 pts**. `drawCoast` ganhou **janelamento de anéis gigantes**: se `np/stride > 20 000` (ex.: continente das Américas ~140 mil pts com zoom alto), emite SÓ os trechos dentro da vista como traço aberto (sem fill, path separado) — sem isso o pan em zoom alto serializaria o anel inteiro por quadro; com a vista aberta o stride grande devolve o caminho fechado com fill normalmente. GSHHS: Wessel & Smith, licença LGPL — dados de domínio público p/ uso científico.

Cor/espessura manuais (2026-07-18b): inputs `coastColor` (color) e `coastW` (px de tela; vazio = automático) na legenda do maphead; estado `coastColor`/`coastWpx` usado nos dois paths do `drawCoast` (fechado c/ fill e janelado); chip da legenda acompanha a cor; persistidos no `egm_prefs` (coastC/coastW).

8. Shapes e pontos (arquivos do usuário)

Parsers: GeoJSON (todas as geometrias, com buracos), CSV/lista `lat,lon[,rótulo]` (auto-detecção de cabeçalho e separador), **Shapefile binário `.shp`** (parser próprio: file code 9994, tipos 1/8/3/5 + variantes Z/M; anéis horários = externos, anti-horários = buracos) e `.dbf` para rótulos (1º campo tipo-nome; UTF-8 com fallback latin1). Seleção múltipla `.shp+.dbf` no mesmo input.

Multi-shape (2026-07-09): o carregamento **acumula** — cada arquivo vira uma entrada em `SHAPES[] = {name, pts, polys, rings, color, keep, keepSig}` com cor automática (`SHAPE_PAL`, 8 cores cíclicas). Os globals `POINTS`/`POLYS`/`POLY_KEEP` viraram **ponteiros para o shape ATIVO** (`setActiveShape(i)` sincroniza; `activeShapeIdx`), mantendo compatível o código de máscara. Seletor `mkShape` escolhe o ativo (usado pela máscara e pela malha nova); `shpDel` remove só o ativo; `× shape` remove todos. `drawShapes` desenha TODOS (paths por shape na sua cor; orçamentos de vértices/poeira globais); `ensureKeep(sh)` guarda o filtro por shape; `computeInsideIdx(sh?)` é parametrizado (default = ativo).

Desenho otimizado (não trava com dezenas de milhares de polígonos):

- `buildShapeRings()` achata `POLYS` em `SHAPE_RINGS` (`Float64Array` + bbox + espaçamento médio + índice do polígono `pi`).
- `drawShapes()`: *culling* por vista (folga 15 %), LOD por *stride*, **orçamento global ~60 000 vértices** (fator `over`), anéis < 3,5 px na tela viram **"poeira"** (um quadradinho ~0,7 px por polígono, deduplicado em células de 1,6 px, teto 15 000); pontos: teto 4 000 visíveis / 400 rótulos. **Rótulos com anti-colisão (2026-07-10)**: cada rótulo posto registra sua caixa estimada (fonte 9, ~0,62·em por caractere) em `lbx`; um rótulo que sobreporia outro é omitido (`lbFits`, varredura gulosa) — ao aproximar o zoom os omitidos reaparecem naturalmente.
- `drawShapesThrottled()` no `applyView`: reconstrução no máx. a cada ~80 ms; entre reconstruções o `viewBox` desloca o desenho existente.
- Medido: 52 000 anéis / 1,6 M vértices ⟶ 44 ms (vista do domínio), 7 ms (zoom 20×).
- **Cor configurável** (`shapeColor`, `<input type=color id=shpColor>`): traço, fill 10 %, poeira 75 %, pontos e rótulos (versão clareada) derivam da mesma cor.

Filtro "≥ 1 célula inteira" (`computePolyKeep`) — otimizado (v2):

1. Rejeição O(V): bbox geográfico menor que `2·DLMD × 2·DPHD × 0,95` não pode conter um losango inteiro.
2. Anéis simplificados (tol 0,25 célula) e transformados para o espaço rotacionado.
3. **Scanline even-odd por linha J**: interseções das arestas com cada `rlat_J` (buracos entram naturalmente); listas ordenadas por linha.
4. Teste do losango por **intervalos**: centro e vértices L/O ⟺ `[xc−DLMD, xc+DLMD]` dentro do intervalo da linha J; vértices N/S caem exatamente sobre as linhas J±1 (busca binária `inAt`). Linhas varridas do meio para fora (aceitação rápida).
5. Cache por assinatura da grade (`IM|JM|DLMD|DPHD|TLM0D|TPH0D`), invalidado ao trocar de shape ou de grade.

Medido: 52 100 polígonos (incl. 2 000 rios finos de 500 vértices) ⟶ **197 ms** (algoritmo v1 por footprint×point-in-polygon: minutos/travamento). O filtro tem dois usos independentes: **plotagem** (checkbox `shpMin`) e **aplicação de máscara/geração de malha** (checkbox `mkFilt`).

9. Máscara por shape (multi-malha) e malha nova por referência

- `computeInsideIdx()` — parte cara, roda 1× por aplicação: polígonos simplificados (tol 0,4 célula) + bbox por anel + índice espacial 48×48 (`buildPolyIndex` / `inPolysIndexed`) + footprint em grade; retorna a lista de índices "dentro". Com `mkFilt` marcado, considera só `POLY_KEEP`. Sem polígonos, usa a lista de pontos (célula mais próxima de cada ponto).
- **Multi-malha:** `buildMaskRows()` gera uma linha (checkbox `.mkuse` + `dentro`/`fora`) por malha carregada; `doApplyMaskMulti()` aplica o mesmo conjunto "dentro" a todas as selecionadas, com **uma única entrada de undo** (vários *snapshots*). `mkKeepOut` (default ligado) preserva os valores externos. Botão **↶ desfazer** no próprio card (`mkUndo` ⟶ `undoEdit($("mkStat"))`; `undoEdit` aceita elemento de status opcional).
- **Clone-template (`mkClone`)**: com o checkbox marcado, `doApplyMaskMulti` chama `cloneInstance(tipo)` para cada tipo alvo **antes** de aplicar — duplica a instância ativa no `BANK`

(deep-copy dos arrays; nome único via `cloneName`, sufixo `_clone`, `_clone2`…), torna o clone a ativa (os ponteiros globais passam a apontar para ele) e aplica a mudança **no clone**; o template original permanece intacto no seletor de instâncias para definir as próximas máscaras.

- **Gerar nova malha por referência (`genRefMask`)**: usa o **shape ativo**; copia o **campo** de referência (`slice()`, via helper `refField(key)`), aplica `dentro=valor`, e grava sempre como **campo ÚNICO** (IM,JM), 1 registro (`writeField`): ref. topo ⟶ só `hgt`; ref. máscara ⟶ só `sm`; veg/solo ⟶ int*4; outro (mesmo com 2 campos) ⟶ só o campo ativo. Tipo (real/int) e endianness da referência preservados — **sem alterar a malha carregada**. Nome default `nova_<arquivo da referência>`; recarregável pelo botão "⊠ Outro" (round-trip validado via `parseGeneric`).
- **Série multi-shape (`genRefAll`, botão ⊠)**: gera **uma malha por shape carregado** (ordem de carregamento) — mesmo campo de referência e mesmo `dentro` — e grava todas num **único arquivo Fortran multi-registro**. Com o checkbox `mkNames` (default ON), cada campo é **precedido de um registro de identificação `character*40`** com o nome do shape (`writeNameRec`; latin1, completado com espaços; leitura: `read(u) nome` / `read(u) campo`). Estrutura por shape: `[4|40 bytes nome|4] [4|IM·JM·4 campo|4]` — validado (nomes com acento incluídos). Caso de uso: sazonalidade de áreas alagadas (12 shapes mensais ⟶ 12 registros de máscara identificados). Nome default `serie_<referência>`; respeita o filtro `mkFilt` por shape.
- **Verificação imediata (`pushVerifyLayer`)**: toda malha gerada (⊠ e ⊠) é também **adicionada ao `BANK.other` como instância nova** (kind conforme int/real, `otherStats` recalculado; nomes `1_<shape>`… na série) e exibida via `setLayer('other')` — a referência permanece intacta e selecionável.

9.1 Topografia ajustada aos níveis eta (2026-07-09)

Porta fiel do trecho `doout902/do903` de **`prep/initbc/interp.f`** (o `const.f` consome o resultado para montar as máscaras HTM/LMH a partir de `RES` e `ETA`):

- **Entrada (`parseDeta`)**: arquivo `deta` binário Fortran — 1 registro `write(u) deta(LM), ldum` (o scan de floats para no primeiro int de `ldum`, que lido como float vira *denormal*; LM deduzido; soma≈1 validada) — ou lista **texto** de DETA (soma≈1) ou de interfaces ETA (0…1 crescentes ⟶ diferenças). Testado com o `deta_50` real de `Eta_support_data/static/fix/deta_files`.
- **Algoritmo (`etaAdjustTopo`)**, constantes do `econstants.h` (g=9.80, r=287.04, γ=0.0065, prf0=101325, t0=288; **PT** é entrada da UI, default 2500 Pa): `eta(l+1)=eta(l)+deta(l)`; `etal=½(eta_l+eta_{l+1})`; `zeta(l)=t0·(1−((pt+eta·(prf0−pt))/prf0)^(rγ/g))/γ`; por célula `etas=(prf0·((t0−γ·h)/t0)^(g/rγ)−pt)/(prf0−pt)`, busca de baixo p/ cima a primeira interface com `etas≥eta(l)`: se `etas<etal(l)` ⟶ `h=zeta(l)`, senão `h=zeta(l+1)` (terra na última camada ⟶ 0.004 m; mar permanece 0).
- **Etapa de *sunken points* (`sunkenPass`)** — porta fiel do bloco `if(.not.fpmnts)` (que no operacional sempre roda): para cada H interior (`j=4..jm−3`), calcula o nível de bloqueio dos 4 pontos de vento vizinhos (mín. de `lhgt` nos 3 H em volta de cada vento; `ihw=−1+mod(j+1,2)`, `ihe=ihw+1`) e detecta pontos **afundados** (`lhgt>lnbmx` e `ref>eta(lmin)`): terra ⟶ **eleva** até `lrais=max(lnbmx,lmin)`; água isolada (0 vizinhos d'água diagonais, ou 1 e ao nível do mar) ⟶ eleva; senão ⟶ **aplaina** a terra em volta do vento de menor elevação média (maior soma `seta` de `etas` em 3 pontos), rebaixando vizinhos com `ref<eta(lij)` para `zeta(lij)` e travando `lmin`.

Varredura única, in-place, na ordem do Fortran; `lhgt` não muda durante o aplainamento (só `ref/hgt/lnew`); a peculiaridade da linha 724 do interp.f (testa `(i,j+1)` mas altera `(i,j+2)` no caso `mmx=2`) foi **replicada fielmente**. Verificação final conta pontos ainda sem vento (`wndls`, reportado no status). Testes: buraco de terra em planalto elevado ao nível dos ventos; água isolada elevada; domínio plano intocado; 0 sem-vento restantes.

- **Saída**: topografia ajustada vira **instância nova** de topo (`<nome>_eta<LM>`, ativa; original preservada) e o índice do nível (`LHGT`) vira instância de "outro" (`nivel_eta<LM>_<nome>`). Validado contra réplica independente do Fortran (amostras exatas; mar⟶0; 0.004 m em terra baixa).
- **Nível L e pressão no ponto de grade (2026-07-10)**: a instância ajustada carrega `etaMeta={lev,pres,lm,pt}` (`lev` = LHGT pós-*sunken points*; `pres` das interfaces de `buildEtaArrays`); com ela ativa, `activeAt('topo')` acrescenta `· LMH=<lev−1> · <pressão> hPa` à elevação — o L EXIBIDO é o **LMH** (máscara da topografia do const.f = LHGT−1; pedido 2026-07-12), com a pressão da superfície (interface LMH+1) — aparece no popup, no readout e no perfil (hover). Mar mostra `LMH=LM · 1013.2 hPa`. A camada de verificação gerada pelo ajuste também é o LMH (`LMH_eta<LM>_<nome>`, antes era o LHGT). No rótulo da célula (toggle `valores`), `cellValLP()` retorna `{L,p}` e `drawGridNodes` desenha **três linhas dentro do losango**: pressão a 42 % da meia-diagonal rumo ao vértice N, altitude no centro, nível L a 42 % rumo ao vértice S — mesma fonte e cor. A fonte é resolvida por **fórmula fechada de contenção**: para cada linha, `fs·(ch·0.62/(2·hW) + vy/hH) ≤ 0.92 − off` (off = fração N-S da ancoragem; vy = extensão vertical do texto em fs; hW/hH = meias-diagonais projetadas da célula), com teto 0.27·hH — o canto mais extremo de cada texto satisfaz `|x|/hW+|y|/hH ≤ 1` (validado numericamente em 8 formatos de losango × 4 conjuntos de rótulos, sem sobreposição entre linhas). Sem teto em px: a fonte cresce **proporcional à célula** em qualquer zoom (piso de legibilidade ~6 px de tela).
- **Topografia original = mesma posição/fonte da ajustada (2026-07-11)**: com a camada `topo` ativa e SEM etaMeta, a altitude é desenhada no MESMO lugar (centro, `q.y+0.85·fs`) e com a MESMA fórmula de fonte — a reserva das linhas L/pressão usa o pior caso fixo de 7 caracteres ("1013hPa") em vez do comprimento real, de modo que a fonte é idêntica com e sem etaMeta (igualdade e contenção validadas numericamente). Alternar instâncias original ⇄ ajustada não move nem redimensiona o rótulo. As demais camadas (veg/solo/outro/diff/máscara) mantêm o valor único abaixo do +.
- **Persistência e reuso**: os níveis ficam em `ETALEV={deta,src,fname,dirty}` — o botão `applyDeta` ("⊠ aplicar níveis") reexecuta `runEtaAdjust()` sobre a topografia ATIVA com o PT atual, sem recarregar o arquivo (o deta pode inclusive ser carregado antes da topografia). `buildEtaArrays(deta,pt)` centraliza eta/etal/zeta/pressões (compartilhado entre ajuste, gráfico e CSV; consistência validada).

9.2 Editor de níveis verticais (`lvModal`, botão ⊠)

Gráfico interativo e **editor** da discretização vertical (estado `lvZoom`/`lvSelIdx`, render re-executável `renderLev()`):

- **Visualização (tema claro)**: fundo **branco** (`#lvModal .pmsvgwrap{background:#fff}`), interfaces em linhas **pretas**, grade de altura cinza-claro, rótulos `l / hPa / m` onde há espaço

(gap ≥ 12 px), **barras de Δeta por camada** à direita (cinza com contorno preto, normalizadas pelo máximo; selecionada em âmbar), hover com leitura completa (eta, pressão, altura, Δeta/espessura das camadas acima e abaixo; linha-guia laranja-escuro).

- **Zoom + rolagem**: roda do mouse centrada no cursor + botões +/−/⟲; grade de altura adaptativa (5 km ⟶ 50 m). Com zoom ativo, **barra de rolagem vertical** desenhada no SVG (x=W−18): polegar proporcional ao trecho visível, arrastável (estado global `lvDrag`, `dz = dySvg/ch·zTop`), clique na pista centraliza; o clique de seleção de camada ignora a região da barra e o pós-arrasto (`lvDragged`).
- **Edição (Σ Δeta = 1 SEMPRE)**: clique seleciona a camada (faixa âmbar); `lvEditApply` define o novo Δeta e **renormaliza as demais** por `×(1−v)/(1−antigo)`, com resíduo numérico absorvido na maior camada (`lvNormResidue`); `lvSplit` (✂) **inclui um novo nível** dividindo a camada em duas metades (LM+1); `lvDelete` (✕) remove somando o Δeta à vizinha (LM−1, mínimo 10). Invariantes testadas: Σ=1 a 10^{-12} após cada operação.
- **Mover interface por arrasto**: mousedown a ≤5 px de uma linha interna (cursor `ns-resize`) inicia `lvIDrag`; a altura do cursor é invertida para eta pela atmosfera padrão (`lvEtaFromZ`, inversa de `zeta(eta)`, erro ~10^{-16}), com *clamp* entre as interfaces vizinhas (±10^{-5}); a nova posição redistribui `deta[L−1]=e−etaLo` / `deta[L]=etaHi−e` — a soma das duas camadas é invariante, logo **Σ=1 é preservada exatamente**, com render ao vivo durante o arrasto.
- **Alterar o topo do modelo**: campo `lvPT` (hPa) no modal (sincronizado com o `p_PT` em Pa do card Carregar dados). As frações Δeta são mantidas sobre o novo intervalo [PT, 1013 hPa] ⟹ a espessura em pressão de **todas** as camadas escala pelo mesmo fator `(prf0−PT_novo)/(prf0−PT_velho)` (validado 25⟶1 hPa: ×1,0243 uniforme; topo 22,4⟶32,4 km); pressões/alturas recalculadas e Σ=1 garantida por renormalização exata.
- **Desfazer/refazer**: pilhas `lvUndoS`/`lvRedoS` (snapshots `{deta, PT}`, teto 100); `lvPush()` antes de cada mutação (aplicar, dividir, remover, mudar topo, e **um snapshot por arrasto** de interface). Botões ↶/↷ no cabeçalho; **Ctrl+Z / Ctrl+Y** (ou Ctrl+Shift+Z) roteados para o editor quando o modal está aberto (senão vão para o undo do mapa).
- **Saídas**: `lvSaveDeta` (⊠) grava o deta editado no formato binário Fortran (`deta real*4 + ldum int*4`, ldum=5 como o `make_deta.f`; round-trip pelo `parseDeta` validado); `lvExportPNG` (⊠) rasteriza o SVG com faixa de título; `lvExportCSV` exporta `interface_l, eta, p_hPa, z_m, deta_camada_l, espessura_camada_l_m`.
- Fluxo típico: editar níveis ⟶ □ `deta` (novo arquivo p/ o prep) ⟶ □ `aplicar níveis` ⟶ comparar topografias resultantes via `diff`.

10. Comparação de arquivos (camada `diff`)

- `computeDiff(key, buf, fname)`: lê o arquivo B com o parser da camada A e guarda **cópias** (`va = A.slice()`, `vb`) — o diff sobrevive à edição/remoção posterior de A.
- **Paleta do diff (2026-07-12)**: linha `dfpalrow` na Legenda (só diff real): paleta única (`diffPal`, "padrao"=DIV_STOPS ou qualquer PALETTES, com `diffPalInv`; 0 = centro, t=v/M·0,5+0,5) **ou** modo `dfSplit` com paletas distintas `diffPalNeg`/`diffPalPos` (t=|v|/M a partir do zero em cada lado; colorbar = neg invertida 0–50% + pos 50–100%); **faixa neutra** `dfBandOn`/`dfBandV`/`dfBandC` (|Δ| ≤ limiar ⟶ cor única via hexRGB; swatch na nota da colorbar). Tudo em `divColor` (v===0

continua cinza-escuro no cellColor) e persistido no egm_prefs. Validado por 9 asserções (padrão/split/faixa). Seletor `dfShow` (todas / só Δ>0 / só Δ<0): filtro de SINAL aplicado no cellColor (célula do sinal oculto = cinza-escuro), aviso na nota da colorbar; view da sessão (não persiste).

- **Comparar malhas CARREGADAS (2026-07-12)**: seletores `dfselA`/`dfselB` (populados no `buildDiffFieldOpts` com a nomenclatura da calculadora — topoN/maskN/vegN/soloN/ outroN + fname) + botão `dfCmpSel` ⟶ `compareLoaded()`: `diffFieldOf(nome)` resolve a instância do BANK (mask ⟶ cópia 0/1 do sm; outro herda real/int), exige kinds iguais (real×real ou cat×cat) e A≠B; o miolo do computeDiff foi extraído em `finishDiff(key,va,vb,kind,lbl,note,fnameB,st)` — usado pelos dois caminhos.
- **Diff entre REGISTROS de uma série (2026-07-13)**: instâncias de "outro" com `frames.length>1` entram nos seletores A/B como **uma opção por registro** (`outroN#k`, rótulo `reg k/N "nome" · arquivo`; a opção plana `outroN` some para a série). `diffFieldOf` aceita o sufixo `#k` (regex `(?:#(\d+))?`) e devolve `frames[k−1]` com lbl `outro rk` e fname = nome do registro — o resto do caminho (compareLoaded/finishDiff) é o mesmo. Caso de uso: sazonalidade (jan × jul da série de áreas alagadas). Validado por 7 asserções (registro certo, #fora ⟶ null, sem-# ⟶ registro ativo, #k em não-série ⟶ null).
- **Arquivo A externo (2026-07-10)**: `DIFFA={buf,name}` (input `f_diffA`, botão `× A`/`clrDiffA` descarta) — se presente, A é lido do arquivo com o parser da camada escolhida em vez da camada carregada (não exige malha carregada; `buildDiffFieldOpts` habilita todas as opções). Para "outro" de 2 campos, A externo compara sempre o **campo 1**. O status anexa `A = arquivo "…"`; `clearAll` descarta.
- **Série multi-registro no "Outro" + animação (2026-07-10)**: `parseGeneric` foi generalizado para varrer a sequência de registros `[nome char*40]? + campo(n·4)` (mesmo formato que o ☒ grava; endianness pelo 1º marcador ∈ {n·4, 2n·4, 40}; tipo int/real pela heurística de *denormals* no 1º campo). 2+ campos ⟶ `{frames:[…], fnames:[…], frame:0, d:frames[0]}`; `otherStats` calcula min/max/classes sobre **todos** os registros (cores/colorbar estáveis durante a animação). Barra `#animbar` no cabeçalho do mapa (visível quando a instância ativa de "outro" é série): ▸/⏸ (`setInterval` a 0,5–8 q/s), *slider* de registro e rótulo `i/N · nome`; `setFrame()` troca `OTHER.d`, reassa o raster e atualiza popup/readout; `updateAnimBar()` é chamada por `setLayer`/`hideLayer` (troca de instância/camada para a animação automaticamente). `saveOther` regrava a série inteira (com registros de nome, se presentes; *round-trip* validado) e o cabeçalho do ☒ PNG indica `registro i/N "nome"`.
- Tipos: `real` (topo, máscara como 0/1, outro-real) ⟶ `d = B−A` (`Float32Array`), paleta divergente `DIV_STOPS` azul–escuro–vermelho, simétrica em ±M, 0 = cinza-escuro; `cat` (MÁSCARA, veg, solo, outro-int — 2026-07-12: máscara passou de real p/ cat) ⟶ **máscara de alterações 0/1**: sem alteração = cinza-escuro, alterada = âmbar fixo (a cor-da-classe-B foi aposentada — diferença numérica não faz sentido em campo categórico); popup `1 · alterado (a → b)`, célula 0/1, legenda com 2 swatches. Botão `dfSaveMask` (☒ máscara 0/1, p/ QUALQUER diff): grava int*4 0/1 (writeField) e adiciona como instância de "outro" (pushVerifyLayer) p/ reuso.
- Integração completa: camada exibível (`togDiff`), legenda com estatísticas, popup/ readout (`A → B`), perfil A⟶B (coluna `diff_B_menos_A` no CSV), lista das 30 maiores |Δ| (clique localiza), exportação CSV de todas as alteradas (`I,J,lat,lon,A,B,diff`, teto 500 000 linhas). **Não** é editável/salvável/mascarável.

10.0 Calculadora de malhas (2026-07-11)

- Parser recursivo próprio (**sem `eval`/`new Function`** — funciona sob o CSP do Artifact): `calcTokenize` → `calcParse` (ternário → `||` → && → comparações → aditivo → multiplicativo → unário → primário/chamadas) → AST avaliado por célula (`calcEval`; 1,06 M células ≈ 320 ms em expressão típica). 14 asserções de parser no procedimento de validação.
- Variáveis = `BANK` numerado (`topo1/mask1/veg1/solo1/outro1…`, lista com o arquivo de cada uma via `buildCalcVars` no updateClearButtons) + `i/j/lat/lon` (arrays materializados só se usados). Comparações→0/1; &&/`||` com curto-circuito; funções em `CALC_FNS`. Não-finito → 0 (contado). Resultado vira instância de "outro" via `pushVerifyLayer` (real*4 ou int*4 com `calcInt`).

10.1 Verificador de consistência topo × máscara (2026-07-10)

- Card `chkcard` (visível com TOPO). `chkScan()` varre a instância ativa em O(n): `landZero` (sm=0 & h≤0), `waterHigh` (sm=1 & h>0), `waterIso`/`landIso` (célula sem nenhum/com todos os 4 vizinhos H de água — a MESMA vizinhança diagonal do `wsum` do interp.f: `ihw=−1+mod(j+1,2)`, pontos `(i+ihw,j±1),(i+ihe,j±1)`, só interior) e `maskDif` (sm difere da instância escolhida em `chkCmp`, populado por `buildChkOpts` a partir do BANK.topo). Estado em `CHK={res,inst,cmpName,show}`.
- Marcas no mapa: bloco em `drawStatic` (como o gradlayer) — 1 círculo por célula com a cor da categoria (`CHK_META`), só quando `TOPO===CHK.inst`, teto 15 000 pontos.
- Lista por categoria (20 primeiras, clique localiza — reusa .dfrow) + status com as contagens; `× marcas` limpa (CHK=null).
- **Correções** (`chkApplyFix`): `waterHigh|waterIso` → sm=0 (h≤0 vira 0,004 m); `landZero` → h=0,004 m. Com `chkClone` (default ON) roda `cloneInstance('topo')` antes (original intacta; o clone herda `etaMeta` — cloneInstance passou a copiá-lo). Undo em massa via snapshot `{snap:[sm,hgt]}` no undoStack (Ctrl+Z); re-varre ao final.
- Motivação (pergunta do usuário 2026-07-10): no interp.f DESTA versão o `sm` nunca é atribuído (o comentário "sm will be changed" é resquício) — continente **não** vira oceano no ajuste eta; terra na última camada recebe 0,004 m. A plataforma copia a máscara intacta (`sm.slice()`); o verificador permite CONFERIR isso (maskDif=0) e tratar os casos legítimos (água elevada pelo *sunken points*, etc.).

11. Exportação PNG (`composeView` + `exportPNG`)

`composeView(SC)` (2026-07-13: extraída do antigo exportPNG, retorna `Promise<canvas>`) compõe em um canvas SC× o tamanho da vista: **faixa de cabeçalho** (3 linhas: grade/resolução/ DLMD/ DPHD; INPES×JNPES=PEs/centro/domínio em km/DT-NPHS-NCNVC-hidro; camada ativa + arquivo + zoom + data) → fundo → canvas raster → **SVG clonado e serializado** (rasterizado via `data:image/svg+xml`; `vector-effect` re-aplicado por atributo). Fonte do cabeçalho encolhe para caber (`W/(maxLen·0.62)`). `exportPNG` = composeView(2) + download via `toBlob`. Nome: `eta_<camada>_<IM>x<JM>_<zoom>x.png`.

11.0 Exportação da série em WebM/GIF (2026-07-13)

Botões `animWebm`/`animGif` na barra da série (só com série ativa; flag `animBusy` evita reentrância; ambos param a animação, percorrem os N registros via `setFrame(i)` + `composeView(1)` — resolução da tela — e restauram o registro inicial):

- **WebM (`animExportWebm`)**: canvas offscreen + `captureStream(0)` com `track.requestFrame()` explícito por quadro (fallback: `stream.requestFrame`, ou captura contínua a N fps se nenhum existir) + `MediaRecorder` (vp9⟶vp8⟶webm, 8 Mb/s). A cadência é de **tempo real**: desenha o quadro, pede o frame e dorme 1000/fps — duração do vídeo = N/fps. Chunks agregados em Blob no `onstop`.
- **GIF (`animExportGif`)**: encoder **puro** (o CSP do Artifact impede libs). Por quadro: `getImageData` ⟶ `gifQuantize` (≤256 cores exatas = paleta direta e *lossless*; senão **median cut** ponderado pela contagem + mapeamento cor-exata⟶índice com cache Map — mapas têm poucas cores únicas) ⟶ `gifLzw` (LZW do GIF: ordem emit⟶bump da convenção dos decoders — o code size cresce **antes** de atribuir o código `1<<cs`, CLEAR ao encher 4096 — empacotado em sub-blocos ≤255). Estrutura: `GIF89a` + LSD sem GCT + NETSCAPE2.0 (loop ∞) + por quadro GCE (disposal=1, delay=100/fps cs) + descritor com **LCT local** (2^bits entradas). Blob montado de Uint8Array por seção (sem array gigante). **Validação**: 64 round-trips LZW contra decoder independente (4 tamanhos de paleta × 4 padrões × n até 70000 — exercita o CLEAR interno), quantização exata ≤256 cores e erro médio <12/canal em gradiente >256 cores, e um GIF completo de 4 quadros lido pelo **PIL** (pixels 100% idênticos, loop=0, duration=500 ms).

11.1 Anotações no mapa (2026-07-09)

Botão ✏ abre a barra (`#annobar`): lápis (`pen`, polyline com decimação de ~2 px), retângulo, círculo (centro⟶raio) e texto (digitado em `#annoText`, posicionado por clique), com cor (`#annoColor`), **espessura de linha** (`#annoW`, 1–8 px de tela, gravada por anotação em `a.w`), **tamanho de fonte do texto** (`#annoFS`, 10–48 px de tela na criação ⟶ depois escala com o zoom), desfazer e limpar. Shapes guardados em `ANNOS[]` em **coordenadas do espaço de projeção** (0…VW/VH) ⟶ o `viewBox` os move junto com o mapa; traços com `non-scaling-stroke` (espessura constante em px); texto com fonte fixada em unidades do mapa no momento da criação (≈15 px de tela) ⟶ escala com o zoom. Grupo SVG `#annolayer` (re-render por `drawAnnos()`, incluído no `drawStatic`). Enquanto uma ferramenta está ativa, os handlers de mousedown/move/click do mapa desviam para `annoDown/annoMove/annoUp` (sem pan/seleção); clicar de novo na ferramenta desativa. Como o PNG serializa o SVG clonado, **as anotações saem na figura** automaticamente.

11.2 Linha, seta e medição de distância (2026-07-11)

- Três ferramentas novas na barra ✏, no mesmo estado `ANNOS` (undo/limpar/PNG automáticos): `line` (segmento), `arrow` (segmento + ponta de 2 traços a ±30°, comprimento `hs` fixado na criação = 12 px de tela, escala com o zoom como o texto) e `ruler` (segmento + tiques perpendiculares nas pontas + rótulo `<d> km`).
- A distância do é **haversine** entre os `toGeo` dos extremos, recalculada a cada render — atualiza ao vivo durante o arrasto (o preview `annoCur` passa pelo mesmo `drawAnnos`). Formato: ≥100 km inteiro, ≥10 km 1 casa, senão 2 casas. Fonte do rótulo = seletor A da barra (fixada na criação, `annoPxW`).

- annoDown/annoMove/annoUp reutilizam o caminho genérico x0/y0→x1/y1 do rect/circ; o dispatch do mapa (annoTool && !== 'text') e a ativação por data-tool já eram genéricos — sem mudanças fora do bloco de anotações.
- **Seleção ⊕ + reinserir ↷ (2026-07-11)**: annoRedo = pilha de remoções ({one,at} de ↶/✕ sel.; {all} do ⊠) — ↷ reinsere na posição original (splice). Ferramenta pick: clique → annoPickAt varre ANNOS de cima p/ baixo com annoHit(a,p,tol=7px) (pen/line/arrow/ruler: distância ponto-segmento annoDistSeg; rect: 4 bordas; circ: |dist−r|; text: caixa fs·0,62/char) — selecionada ganha caixa tracejada (annoBBox+pad) no drawAnnos; remove por × sel. ou tecla Delete/Backspace (fora de inputs). ⊕ mantém o pan (mousedown não desenha) e ignora o clique pós-arrasto; fechar a barra deseleciona. Hit-test validado por asserções (linha/rect-borda/circ-anel/pen/texto + bboxes).

12. Perfil A→B

Botão perfil → clique em A e B no mapa → 220 amostras interpoladas na reta em índice de grade; distância acumulada por *haversine*. Modal com **um gráfico por camada carregada** (empilhados, eixo X compartilhado), eixo alternável i,j/PE ⇄ lat/lon, zoom por roda, hover multi-campo e exportação CSV.

12.1 Corte vertical A→B com níveis eta (2026-07-10)

- Cada amostra do perfil carrega et={h,sea,lv} (h/sm da topo ativa; lv=LHGT do etaMeta, se ajustada). pmCut() = corte presente (TOPO && ETALEV); pmAxY() centraliza a altura total (o corte tem PM_CUTH=2×chartH e entra ANTES dos gráficos de camada — renderProfile, mousemove e cursor usam o helper).
- renderEtaCut(pts,X,top,cw): lvArrays() dá zeta/pres; teto do gráfico = interface logo acima do terreno mais alto da janela (e acima do menor LHGT) −2 camadas de folga; interfaces = linhas horizontais com nº (l, 1-based Fortran = índice js+1) à esquerda e hPa à direita (rótulos com espaçamento ≥9,5 px); terreno em DEGRAUS: coluna por amostra com bordas no ponto médio entre amostras (xe(i)), mar = faixa azul na base. Acompanha o zoom horizontal (mesmo X).
- Hover: anexa LMH=<lv−1> · p hPa (pres[lv−1] = superfície) quando etaMeta; CSV ganha colunas LMH/p_hPa.
- **Água continental (2026-07-11)**: layerValAt('topo')/fieldValNum('topo') NÃO zeram mais os pontos de água — água com nível L ajustado (ex.: rio/lago elevado pelo *sunken points* ou com elevação no etatopo bruto) exibe a topografia do degrau (hgt real do array) no popup (água · 449 m), no rótulo da célula, no perfil e no corte vertical (coluna no degrau; a faixa azul de água passou a marcar a **superfície**, y−3, em vez do rodapé do gráfico). Mar (h=0) segue mostrando 0.

12.2 LMH / HTM por nível (2026-07-10)

- Linha htmrow no card Carregar dados (visível se TOPO.etaMeta; toggle no updateClearButtons). Semântica VALIDADA contra réplica do const.f p/ todo lhgt: LMH = LHGT−1 (mar → LM) e HTM(l)=1 ⟺ l ≤ LMH (const.f usa eta(l+1)>res estrito, igualdade fica atmosfera).
- genLMH() → instância "outro" int*4 LMH_<fname> (pushVerifyLayer).
- genHTM() → máscara 0/1 do nível digitado (HTM_L<n>_<fname>), status com pressão média da camada e contagem de subterrâneas.

- `genHTMSerie()` ⟶ SÉRIE animável (1 registro int*4 nomeado por nível; faixa `L1-L2` ou vazio = `minLev–1…LM`, minLev cacheado no etaMeta); teto de memória `maxF=floor(120e6/(n·4))` registros; salvar reusa o caminho de série do saveOther.

12.3 INIT · BNDY · CNST · RESTRT — leitura, checagem, plotagem, corte e perfil vertical (2026-07-15; RESTRT 2026-09-02)

Card `ibccard` (grupo Análise) + entradas `f_init`/`f_bndy`/`f_cnst`/`f_rst` no menu ⊠ e no lote (`batchDetect` por `init.file|bndy.file|cnst.file|restrt|restart`; o teto de 500 MB do lote é dispensado p/ esses nomes). Módulo inteiro após o `batchLoad` (busque por `// ---- INIT / BNDY / CNST`).

Formatos (validados contra `prep/initbc/const.f`/`boco.f` e a leitura do modelo `READ_RESTRT`/`READ_NHB`/`BOCOH`/`BOCOV`; REAL*4*/*INT*4 little-endian, marcadores de 4 bytes):

- `INIT.file` — `run,idat(3),ihrst,ntsd` · PD RES FIS · U,V,T,Q ×LM (1 registro 2D por camada) · SI SNO · SMC(3D NSOIL) CMC STC(3D) SH2O(3D) ALBEDO ⟹ 11+4·LM registros.
- `BNDY.file.*` — `run,idat,ihrst,tboco` · blocos `{BCHR · PDB(LB,2) · TB QB UB VB Q2B CWMB (LB,LM,2)}`; `(:,…,2)` = tendência/s; último bloco duplica o terminal. **LB = 2·IM+JM−3**; mapeamento orla⟶grade: H (BOCOH) sul I=1..IM em J=1 ⟶ norte ⟶ oeste (I=1, J=3,5,…) ⟶ leste; V (BOCOV) I=1..IM−1 e J=2,4,… (`ibcHmap`/`ibcVmap`).
- `CNST.file` — cabeçalho (tam = **48·JM−200**, com JAM=6+2·(JM−10)) · LMH LMV · HBM2 VBM2 VBM3 SM SICE · HTM,VTM ×LM · registro de constantes (dt…aeta…em/emt) · DX…DDMPV · PT2+GLATR · GLONR · 2 registros de tabelas · MXSNAL…HDACV · TTBLQ · PTBL+escalares (cauda fixa de 60+(3·LM+1)·4 B com `wbd,sbd,tlm0d,tph0d,dlmd,dphd…`) · IVGTYP ISLTYP ISLOPE · VEGFRC · SLDPTH RTDPTH ⟹ 39+2·LM registros.

Decisões técnicas:

- **Leitura por fatias** (`ibcSlice` = `File.slice().arrayBuffer()`): nada do arquivo fica residente; `scanFortranRecs` percorre só os marcadores (2 leituras de 4 B por registro) — o INIT de 775 MB do domínio 8 km abre em ~1 s.
- **Dimensões deduzidas dos tamanhos de registro** (nunca digitadas): INIT LM=(nregs−11)/4 e NSOIL=tam(SMC)/tam(2D); BNDY LB=tam(PDB)/8, LM, nt=(nregs−1)/8; CNST LM=(nregs−39)/2, JM=(tam(reg1)+200)/48. A grade da tela é conferida (IM·JM ou LB) com erro explícito pedindo o ETAIN.
- **`ibcIsInt`** (heurística de denormais por registro): builds antigos do initbc gravam campos de classe como REAL (no CNST real da ams_08km_v2 o ISLOPE é float 7.0/8.0/15.0 embora o const.f atual grave INTEGER) — o tipo é detectado por registro.
- **Unidades de exibição** via `fd.scale` no catálogo (PD/PDB ×0,01 ⟶ hPa; Q/QB ×1000 ⟶ g/kg), aplicadas na extração **e** na checagem; `IBC_LIM` já está nas unidades exibidas. Tooltip "outro" real com 6 dígitos significativos.
- **Plotagem**: `ibcExtract` materializa campo/nível/tempo/componente e injeta numa **única instância viva** de "outro" (`IBCINST`, atualizada in place ⟶ replot automático ao trocar seleção). BNDY = fila única da fronteira (interior NaN; `cellColor`/`activeAt` ganharam guarda de NaN); `ibcDilate` opcional (espessura visual, default 1 = fiel ao arquivo).

- **Checagem** (`ibcCheck`/`ibcCheckField`): NaN/Inf + faixas físicas por variável (tendências do BNDY: |d/dt| ≤ faixa/3600); relatório por campo com até 6 ofensores clicáveis (`.dfrow`, select(I,J)). BNDY inteiro ~0,2 s; CNST inteiro ~4 s; INIT inteiro ~10 s (775 MB).
- **Corte vertical** (`ibcSection`+`ibcRenderSec`, canvas): INIT/CNST = distância × nível ao longo da polilinha (amostras ∝ comprimento por segmento, `ibcSampPath` devolve `vidx`); BNDY = orla desenrolada × nível com separadores S/N/O/L. **Subterrâneo mascarado** por LMH/LMV do CNST (`ibcLMHArr`, cache `C._lmh/_lmv`) ou `etaMeta` da topografia ajustada — e **excluído do min/máx** (`ibcMinMax`), senão o lixo extrapolado (T=150 K) esmaga a escala.
- **Perfil vertical** (`ibcProfileV`+`ibcRenderProf`): sondagem no `current` (BNDY salta p/ orla mais próxima, `ibcNearestK`); eixo direito η (aeta do CNST) ou **pressão real** p(L)=PT+η·PD (PD lido do INIT no ponto / PDB do bloco); acompanha o ponto do mapa via `ibcSelHook()` (debounce 160 ms) chamado no fim de `select()`.
- **Janela flutuante** (`#ibcModal`): container `pointer-events:none`, `pmbg` escondido, `pmpanel` fixed com `resize:both` — o mapa continua interativo; arraste pelo `pmhead`; **ResizeObserver** re-dimensiona o canvas (gráficos responsivos).
- **Zoom/pan nos gráficos**: janela `IBCM.vz` ({i0,i1,l0,l1} na seção; {l0,l1} no perfil, com o eixo X re-enquadrado aos níveis visíveis); roda = zoom centrado no cursor, arraste = pan (`ibcSetVz` clampa), duplo-clique = enquadrar; renderers com `clip()` e loops só na faixa visível; colorbar fixa no total (cores estáveis).
- **Linha composta arrastável** (✏ `ibcLine`, só INIT/CNST 3D): `IBCPATH` (declarada com `var` — `drawStatic` roda antes do bloco); overlay `#ibcpathlayer` (`drawIbcPath`, tamanho de tela constante como o `drawProfOverlay`); hooks no mapa: `mousedown` testa `ibcVtxAt` (tolerância 8 px) **antes** do pan, `click` adiciona vértice em modo ✏, duplo-clique encerra. **Durante o arrasto de um vértice**, `ibcSecDragSched` (coalescido a 200 ms, re-agenda enquanto arrasta) chama `ibcSecDrag(v)`: recalcula **só os segmentos vizinhos** com leituras de coluna (4 B/nível, lotes `Promise.all` de 64) + máscara/min-máx/render; no `mouseup` o rebuild completo refaz as distâncias; `vz` é preservado entre rebuilds (reescalado se o nº de amostras mudou).
- **Exportações**: CSV (seção = amostras × níveis com I,J/lat/lon/dist ou k; perfil = L/η/p/valor) e 🖼 PNG (`canvas.toBlob`).
- **CNST → níveis eta** (2026-07-15, pedido em vídeo): `ibcParseCnst` extrai também o **DETAC** (offset 40 do registro de constantes; sanidade |Σ−1|<0,05) e, no `loadIBCFile`, popula `ETALEV` + seta `p_PT` = PT do arquivo + mostra `lvBtn`/`applyDeta` — a representação/edição da distribuição (§9.2) funciona sem arquivo `deta` avulso. Prioridade: um deta do usuário NÃO é substituído (níveis vindos de CNST anterior são); no lote o deta carrega antes do CNST, preservando a regra. Validado: CNST 8 km real ⟶ Σdeta=1,0 e aeta ≡ ponto médio do acumulado.

Validação (harness Node com shim de `File` sobre `fs` + stub de canvas por Proxy, contra os arquivos reais de `/home/jorge/Eta_install/run/ams_08km_v2/`): os 3 arquivos parseiam 100 % (IM=735, JM=1441, LM=38, NSOIL=8, LB=2908); coluna ≡ extração por nível; células da seção ≡ mapa; anel com exatamente LB pontos finitos; sintético com NaN/999/Inf injetados achado nas coordenadas exatas; grade errada ⟶ erro claro; checagem limpa no arquivo são (após relaxar VEGFRC p/ 1,05 — o real chega a 1,02 — e detectar ISLOPE float).

12.3.1 RESTRT (2026-09-02) — 4º tipo do card, com tolerância a truncamento

Tipo `rst` no mesmo estado `IBC{}` (rótulo RESTRT), lendo o **`RESTRT#######.t00s` gravado pelo quilt** (layout do `QUILT.f90`, espelhado do `util/scan_restrt.f90` — o scanner Fortran é o **oráculo** da validação). Estrutura: cab.1 (60 B: RUN, IDAT, IHRST, NTSD, LABEL, IHOUR) · PDOMG+RESOMG · OMGALF×LM · cab.2 (72 B) · PD+RES+FIS · contorno (pulado) · **bloco 3D intercalado** {T,Q,U,V,Q2,TTND,CWM,TRAIN,TCUCN}×LM · ~20 registros de superfície multi-campo · solo (SMC/STC/SH2O em 1 registro de NSOIL camadas) · cauda com escalares (inclui TPH0D/TLM0D ⟶ conferência da grade) · RSWTT/RLWTT intercalados · CNVBOT..RLWTOA ⟹ **29+12·LM registros**. LM deduzido da posição do cab.2; NSOIL do tamanho do registro SMC.

Generalizações mínimas na máquina existente (retrocompatíveis, default 1/ausente):

- **`fd.rs` (passo de registro)** — 3D intercalado do RESTRT: T em `r0+9·(L−1)`, RSWTT em `r0+2·(L−1)`. Aplicado em `ibcExtract`, `ibcColumn`, `ibcSection`, `ibcSecDrag` e `ibcCheckField` (todos os pontos que indexam `recs[fd.r0+…]`).
- **`fd.skip` reusado** p/ campos 2D empacotados (até 13 por registro; os com cabeçalho têm +60 B). `ibcCheckField` passou a ler **só IM·JM·4 bytes por campo** (antes lia o registro inteiro — correto p/ INIT/CNST de 1 campo/registro, errado p/ registros multi-campo).
- **`scanFortranRecs(file, tol)`** — modo tolerante: devolve os registros **íntegros**
  - `{pos, why}` do ponto de truncamento em vez de erro (só o RESTRT usa; INIT/BNDY/ CNST continuam estritos). `ibcParseRst` valida tamanho registro a registro contra a tabela esperada (`esz[]`/`dsc[]` com descrição por registro), monta o catálogo (**~109 campos**, com escala hPa/ g-por-kg e heurística int p/ LC/NCFRCV/NCFRST) SÓ com o que existe — 3D parcial ganha `fd.part` e "⚠ truncado (1–N de LM)" no seletor — e reporta `%` do layout + descrição do registro onde parou.
- **Perfil com pressão própria**: `F.pdRec` (INIT = registro 1; RESTRT = PD do registro 3+LM) — o eixo p(L)=PT+η·PD do perfil usa o PD do arquivo plotado, com o INIT como reserva (antes só INIT fornecia PD).
- `IBC_LIM` ganhou ~60 faixas físicas p/ os campos do RESTRT (nas unidades exibidas; acumulados de fluxo só NaN/Inf; HBOT/HTOP/CNVBOT/CNVTOP até 101 — o modelo inicializa com 100).

Validação (harness Node com shim de `File`, funções extraídas do próprio HTML): RESTRT real 5 km (181×311×50, 163 MB, 629 registros) — min/máx de 20 campos (2D/3D/solo/intercalados/ escalados/int) **idênticos ao `scan_restrt`**; truncado em 95 MB ⟶ 414 registros íntegros, para em "TCUCN L40" = exatamente onde o Fortran reporta, T vira 1–40/50 e campos posteriores saem do catálogo; sintético 7×9×10/NSOIL=4 **big e little-endian** (gerador Python conferido pelo scanner Fortran: 149 registros ✓) — endianness, NSOIL≠8, LC/NCFRCV ⟶ Int32, escalas e gok=✓; grade errada ⟶ mensagem clara no 2º registro; checagem de faixas no arquivo real limpa (T/Q/PSLP/ SMC/RSWIN/CNVTOP), inclusive leitura limitada por campo nos registros multi-campo.

12.4 Filtro de visualização da camada ativa (2026-07-18)

Linha `#vfrow` no card Legenda (montada por `buildVfRow()` dentro do `buildColorbar`): estado `VF={on,key,cls,min,max}` válido só para a camada `VF.key` (trocar de camada desliga). `vfPass(idx)` decide por célula: classe única p/ veg/solo/máscara(0/1)/outro inteiro; faixa mín–

máx p/ topografia (elevação) e outro real (NaN nunca passa; limite vazio = sem limite). Aplicação no **`cellColor`**, logo após o destaque ☒ EVHL: célula reprovada ⟶ `VF_DIM` [13,19,27] — pega o raster do `bakeField`, os losangos do zoom alto e o PNG de graça. `vfRecount()` mostra o nº de células aprovadas; placeholders do mín/máx trazem a faixa real do campo; inputs com debounce de 300 ms (re-assar a grade inteira é caro). Visualização apenas: perfil/CSV/edição/estatísticas não mudam. Testes unitários (Node, stubs): classe veg/solo, faixa da topo (inclusive só-máx), máscara água, outro real com NaN — todos exatos.

13. Edição, undo e utilidades

☑ **por malha no editor (2026-07-18):** checkboxes `ek_topo/ek_mask/ek_veg/ek_solo/ek_other` (classe `.ek`, default marcados) nos grupos do card; `editApply` e `paintCell` só alteram a malha se o ☑ correspondente está marcado (blocos de TOPO divididos em elev/máscara — cada um com seu ☑); campo desmarcado fica `disabled`/opacidade 0,4; rótulo da pintura atualizado ("pintar as malhas ☑ marcadas"). Undo/salvar/trocar-valor inalterados (trocar-valor continua operando na CAMADA ATIVA, independente dos ☑).

- Edição por ponto (todas as malhas de uma vez), modo **pintura** (valores fixados, clique aplica), **trocar todos** os valores da camada ativa (com tolerância ± para campos reais), salvar cada malha com nome escolhido (formato e endianness do arquivo lido).
- **Undo dual** (`undoStack`, teto 2000, Ctrl+Z): entradas por ponto (`{items:[[arr,idx,old]…]}`) e por *snapshot* (`{snap:[{arr,copy}…]}`) para operações em massa; flags `cat`/`oth` disparam recomputação de classes/estatísticas.
- ⧉ **duplicar malha (2026-07-12)**: botão no card Editar/salvar — `cloneInstance` da camada ativa (mask⟶topo); o clone vira a instância ATIVA (entra nos seletores, no diff-entre-carregadas e na calculadora), a original fica intacta. Motivação: "trocar todos" editava in-place e obrigava salvar+recarregar p/ comparar com a original. `cloneInstance` agora copia SÉRIES (frames/fnames/frame) em deep-copy.
- ☒ **destacar valor⟶cor (2026-07-13, pedido com screenshot ao pintar máscara=−2):** linha no card Editar/salvar (checkbox `evHl` + valor `evHlVal` + cor `evHlColor` + contador `evHlN`). Estado global `EVHL={arr,v,rgb}` recalculado por `evHlPrep()` no início do **bakeField** (a instância/camada ativa pode ter trocado); `cellColor` testa `EVHL.arr[idx]===EVHL.v` **antes** de qualquer ramo e devolve a cor escolhida. Valor: `Math.fround` p/ campos reais (casa o float32 armazenado), `Math.round` p/ int/classes; camada não editável (diff) desliga. Ao ligar com o campo vazio, herda o valor do campo de edição da camada ativa; inputs com debounce (350 ms) pois o rebake varre a grade inteira; só a **cor** persiste no egm_prefs (`hlC`). Puramente visual (não toca o dado), mas entra no PNG/WebM/GIF por ser parte do raster. Validado por 8 asserções com stubs de DOM.
- **Máscara aceita valor-marcador ≠0/1 (2026-07-13):** `editApply`/`paintCell` gravam o valor da máscara **como digitado** (antes: `>=0.5?1:0`); comparação de no-op via `Math.fround(v)!==sm[idx]`. Status da edição avisa `másc ⚠ v≠0/1 (marcador)`. Displays atualizados p/ valor bruto: campo de edição (`toPrecision(7)`), popup/readout (`−2 (marcador ≠0/1)`), perfil/CSV (`fieldValNum` devolve sm bruto) e célula (formato genérico). A renderização padrão da máscara segue a regra >0,5 (marcador aparece como continente — use o ☒ destaque); "trocar

valor" converte os marcadores de volta antes de salvar. Diff da máscara segue comparando 0/1 (`Float32Array.from(sm, x=>x>0.5?1:0)`).
- Remoção individual de camadas (✕) e "limpar tudo" com guarda de duplo clique quando há edições não salvas.
- **Preferências persistentes (2026-07-10)**: `savePrefs()`/`loadPrefs()` guardam em `localStorage` (chave `egm_prefs`, versão `v:1`) os toggles costa/gratic/valores/ popup, paleta do "outro" (+inversão/discreto/nº de faixas), cor/espessura/fonte das anotações e o PT (Pa). Gravação por *listeners* adicionais registrados DEPOIS dos handlers originais (mesmos elementos; a ordem de registro garante estado já atualizado); leitura no init antes do `drawStatic`. Tudo em `try/catch` — no sandbox do Artifact (sem `localStorage`) o recurso desliga silenciosamente.

14. Validação e deploy (procedimento obrigatório antes de redeployar)

```
# 1. Sintaxe do bloco <script> (o arquivo tem 1 linha gigante de base64 — não
abrir inteiro)
awk '/<script>/{f=1;next}/<\/script>/{f=0}f' eta-grid-mapper.html > /tmp/s.js &&
node --check /tmp/s.js

# 2. Todo $("id") do JS deve existir no HTML (evita erro de wiring em runtime)
#    (script node: extrai \$\("..."\) do JS e compara com os id="..." do HTML)
```

Redeploy: ferramenta *Artifact* com `file_path` do HTML e a URL fixa do topo deste documento. O usuário também abre o **arquivo local** direto no Chrome (via `wsl.localhost`) — lembrá-lo de recarregar com **Ctrl+F5**.

Testes de lógica usados no desenvolvimento (padrão recomendado): extrair funções do próprio arquivo por regex e rodá-las em Node com casos sintéticos (ex.: filtro de polígonos com caixas de 1°/0,02°/0,12°/fora-do-domínio ⟶ `[true,false,true,false]`; round-trip do `rotToGrid`; decodificação da costa consumindo 100 % do stream).

Invariante do arquivo (2026-07-18): a PRIMEIRA linha do HTML deve continuar sendo `<meta charset="utf-8">` — sem ela o Chrome adivinha a codificação do arquivo local e, com o payload de costa (4 MB de base64 ASCII), erra para Windows-1252 (acentos/emojis viram mojibake). No Artifact o wrapper injeta o charset, mas o arquivo local depende dessa tag.

15. Desempenho (números de referência, WSL2)

Operação	Carga	Tempo
Máscara por polígono (Brasil UFs)	1 032 polígonos / 1,14 M vértices	~1,2 s
Desenho de shape denso (reconstrução)	52 k anéis / 1,6 M vértices	44 ms (vista cheia) · 7 ms (zoom 20×)
Filtro ≥1 célula inteira (v2 scanline)	52 k polígonos (2 k rios de 500 vérts)	197 ms
Codificação da costa (offline, GSHHS h)	1 294 887 pontos	3,9 MB base64, erro ≤ ~55 m; decode no navegador ~0,3 s (1×, no 1º desenho)

16. Pendências / roadmap sugerido

1. **Tabelas oficiais SiB (12 classes) e FAO (15) do CPTEC** — os nomes atuais de vegetação/solo são de referência (`VEG_NAMES`/`SOIL_NAMES`, aviso na legenda).

Itens concluídos em 2026-07-13 (removidos do roadmap): exportar a animação da série como WebM / GIF (§11.0); diff entre dois registros de uma mesma série (§10).

Itens concluídos em 2026-07-10 (removidos do roadmap): graticule sob demanda; rótulos de shape com anti-colisão; diff com arquivo A externo; preferências em `localStorage`; animação da série multi-registro; nível L + pressão no ponto de grade da topografia ajustada.

17. Histórico resumido

Data	Entrega
2026-07-04	Núcleo: conversões PE⇄global⇄lat/lon, mapa SVG, camadas topo/máscara/veg/solo, perfil, editor, shapefile, alinhamento célula/highlight
2026-07-06	Máscara por shape otimizada, camada "Outro" completa (paletas/escala/faixas), trocar-valor±tol, salvar-como, remover camadas
2026-07-09	Costa real NE 50m embutida; **células em losango (Voronoi E-grid)**; exportar PNG (+cabeçalho, título opcional); comparação B−A (`diff`); shape denso sem travar (LOD/poeira/throttle); cor do shape; filtro ≥1 célula inteira (scanline) p/ plotagem e máscara; máscara multi-malha + malha nova por referência (campo único); PEs sob demanda; toggles costa/shape; undo no card da máscara
2026-07-09 (cont.)	**Multi-shape** (acúmulo, ativo selecionável, cor por shape) + série 1-malha-por-shape em arquivo único (com registro de nome character*40); ***multi-instância*** *de todas as malhas (BANK) + clone-template; malhas geradas viram camadas de verificação;* ***topografia ajustada aos níveis eta*** *(interp.f, com* sunken points*), níveis persistentes + **editor de níveis completo*** (tema claro, zoom + barra de rolagem, editar Δeta com renormalização, mover interface por arrasto, alterar topo/PT proporcional, dividir/remover nível, undo/redo, salvar deta/PNG/CSV); **anotações** (lápis/retângulo/círculo/texto, espessura e fonte configuráveis, no PNG); valores nas células; documentação md/docx/pdf
2026-07-10	**Animação da série multi-registro** ("Outro" lê N registros com nome char*40; barra ▸ com slider/velocidade; estatísticas globais p/ cores estáveis; saveOther regrava a série; registro no cabeçalho do PNG); **diff com arquivo A externo** (`f_diffA`, sem exigir camada carregada); **nível L + pressão** no ponto de grade da topografia ajustada (etaMeta) — no
2026-07-[illegible](isMOM)	[illegible]

usa `cl[illegible]ata})`