Evandro Moimaz Anselmo

Manual para os modelos: SiB2 e Balanço de água. Processo de calibração dos modelos pela otimização de parâmetros.

São Paulo - SP

2020

SUMÁRIO

1 INÍCIO

Esta documentação faz referência aos repositórios: **hidromodel**, **srcLCB** e **srcsib2model**. O repositório **hidromodel** possui o modelo de balanço hidrológico para pequenas bacias hidrográficas proposto em Vandewiele, Xu et al. (1992), **srcLCB** é composto de códigos utilitários para o LCB/IAG/USP, enquanto o **srcsib2model** concentra apenas o código fonte do modelo SiB2 (SELLERS et al., 1996a; SELLERS et al., 1996b).

Os repositórios **hidromodel**, **srcLCB** e **srcsib2model** estão versionados pelo sistema Git, portanto é possível avaliar todo o respectivo histórico de desenvolvimento de cada código usando os comandos do git.

Logo no início destaca-se que, a versão mais atual do modelo SiB2 está no repositório **srcsib2model**, diretório "SiB2f95modvars", em que, o SiB2 está traduzido para Fortran 95 e as variáveis do comsibc e pardif operam como um módulo de Fortran.

Para compilar o SiB2:

```
gfortran comsibc.f95 pardif.f95 sib2x.f95 Sib2xa.f95 Sib2xb.f95 -o
↪  sib2run
```

Para executar:

```
./sib2run
```

no diretório em que se encontra os arquivos "data1" e "data2".

O código do SiB2 que se trabalhava no LCB em Fortran 77 está no repositório **srcsib2model**, diretório "SiB2f77".

2 DESCRIÇÃO E MODIFICAÇÕES NO SIB2

2.1 Formato dos dados ambientais de entrada: arquivo data2

São sete colunas de dados, sendo, da esquerda para direita:

1ª – datetime: ano, mês, dia e hora YYMMDDHH
2ª – Ki(Wmˆ−2): irradiância de onda curta incidente observada [Wm^{-2}]
3ª – em(hPa): pressão de vapor d'água observada [hPa]
4ª – tm(K): temperatura do ar observada [K]
5ª – um(msˆ−1): velocidade horizontal do vento observada [ms^{-1}]
6ª – prec(mm): precipitação observada [mm]
7ª – Rn(Wmˆ−2): saldo de radiação observado [Wm^{-2}]

O formato explícito em fortran é "(A8,6F11.4)", exemplo:

```
write(2,'(A8,6F11.4)') nymd, swdown, em, tm, um, prec, rnetm
```

Os arquivos "data2" são gerados pelo programa em Fortran 95 denominado como "data2_format.f95" localizado no diretório "utils" do repositório **srcLCB**. Inicialmente Rn ficava na 3ª coluna, mas em agosto de 2019, a ordem das colunas foi alterada para permitir a inserção de uma condição "if" no código que seleciona o flag "ilw" e decide sobre o uso da coluna de Rn.

2.2 Equação rsoil no Sib2xb

O LCB vem utilizando 2 equações para rsoil:

$$rsoil = amax1(0.1, 694. - fac * 1500.) + 23.6 \quad (1)$$

$$rsoil = amax1(0.1, 1001. - exp(fac * 6.686)) \quad (2)$$

A tabela abaixo indica qual a equação rsoil tem sido utilizada para cada local de medidas.

Tabela 1 – Equação rsoil utilizada em cada sítio de medidas

Locais	equação
Cana	2
Cerrado	2
Eucalipto	1
Fazenda K77	2
Floresta Atlântica	1
Floresta Rondônia	2
Pastagem Rondônia	2
Pastagem SP	2

2.3 Compilando o SiB2 no gcc e a tradução do Fortran 77 para Fortran 95

Identificou-se que o modelo SiB2 que vinha sendo utilizado no LCB, escrito em Fortran 77, apresentava erros de compilação e de execução quando compilado a partir do gfortran no linux debian buster e versões anteriores mesmo usando o g77 que integrava o gcc até a sua versão 3.4.6. O SiB2 no LCB geralmente era utilizado a partir de compiladores de Fortran 77 em sistema windows que já não estão mais disponíveis no LCB, portanto, foram estudadas as possibilidades de continuar fazendo altereções no SiB2 usando compiladores de Fortran mais portáveis e nas versões mais atuais e neste sentido o gfortran que integra o gcc 8.3.0 foi considerado o mais apropriado.

Fazendo modificações no código do SiB2 em Fortran 77, combinadas com os flags de compilação "-fno-automatic" e "-finit-local-zero" disponíveis no gfortran, os erros de compilação e execução do SiB2 do LCB no gfortran do gcc 8.3.0 foram solucionados e os resultados das saídas do modelo compilado no gfortran e nos compiladores para windows ficaram

iguais, apresentando pequenas diferenças provavelmente associadas com arredondamento numérico, conforme mostra a figura 1.

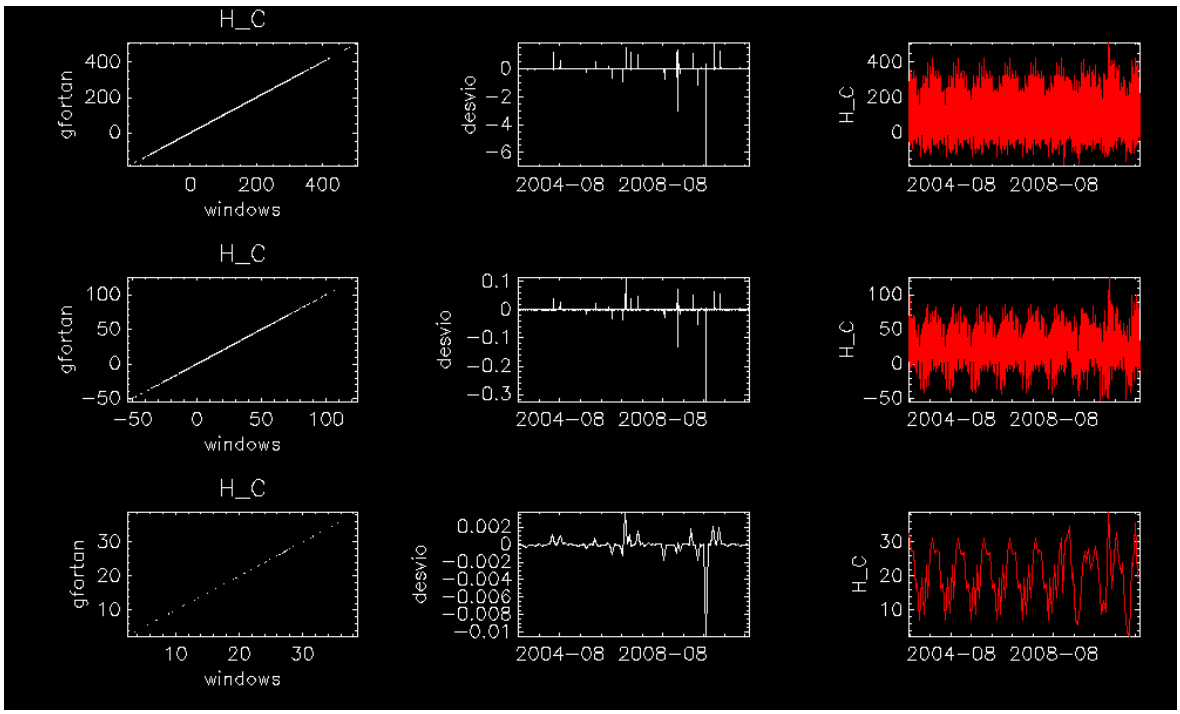

Figura 1 – Comparação entre H calculado com o SiB2 compilado no windows e compilado no gfortran com os flags "-fno-automatic" e "-finit-local-zero". Em cada coluna deste painel (da esquerda para direita) está a dispersão, o desvio no tempo de H windows - gfortran e H no tempo, sendo H na cor branca = windows e H na cor vermelha = gfortran, sendo possível ver apenas a sobreposição da cor vermelha. Em cada linha (de cima para baixo) temos a série temporal horária, as médias diárias e mensais.

Com o código em linguagem IDL/GDL em **srcLCB**/plots/x_plot_checkdefault.pro é possível gerar o painel da figura 1 para todas as variáveis do SiB2.

Após esta etapa de compilação e execução do SiB2 em Fortran 77, o código do SiB2 do LCB foi traduzido integralmente para o Fortran 95 em **srcsib2model**/SiB2f95.

2.4 SiB2 em Fortran 95: comsibc.h e pardif.h como módulo de variáveis

Após a tradução do SiB2 de Fortran 77 para 95, foi desenvolvida uma versão do código, em **srcsib2model**/SiB2f95modvars, que elimina as declarações de variáveis do tipo "common"e cria módulos de variáveis de Fortran: o comsibc.f95 e o pardif.f95. Nesta versão deve-se compilar o código conforme os seguintes comandos:

```
gfortran comsibc.f95 pardif.f95 sib2x.f95 Sib2xa.f95 Sib2xb.f95 -o
↪ sib2run
```

Além dos módulos de variáveis, acrescentou-se "implicit none" no programa e em todas as sub-rotinas declarando todas as variáveis explicitamente e todas as variáveis do tipo REAL foram declaradas com dupla precisão, i.e., 64 bits.

Nesta etapa, foi revisada a necessidade do uso dos flags de compilação – "-fno-automatic -finit-local-zero" – que inicializa todas as variáveis com valor zero e declara todas as variáveis como "save".

Usando o flag de compilação "-Wall", o gfortran indicou as variáveis que integravam os cálculos do SiB2 e que deveriam inicializar com valores, pois estavam sendo inicializadas com valores qualquer alocados na memória na hora da declaração de variáveis. Estas variáveis identificadas foram:

```
sib2x: ptot

Sib2xa: zinc
      : us1

Sib2xb: cs
      : th
      : qm
      : wmin
```

Declarando estas variáveis com valores iniciais igual a zero (e isso está feito em **src-sib2model**/SiB2f95modvars) exclui-se a necessidade do uso dos flags de compilação "-fno-automatic -finit-local-zero", pois assim os resultados dos cálculos são idênticos aos valores calculados quando compila-se o SiB2 com os flags "-fno-automatic -finit-local-zero" e coincidem com os cálculos do SiB2 em Fortran 77 compilado no windows, como mostrado na figura 1. No entanto, sugiro uma verificação destas variáveis a respeito de quais os valores elas devem ter inicialmente no código, se existe alguma que o seu valor deve ser diferente de zero.

2.5 O SiB2pymod

O SiB2pymod é o modelo SiB2 em Fortran 95, compilado como um módulo de python 3 com o uso do f2py que integra o numpy. Para isso, uma pequena mudança no código do SiB2 é feita. O programa sib2 é descrito como uma sub-rotina. Desta forma o sib2 é usado como uma função python.

Para gerar o sib2pymod execute:

```
f2py3.7 --fcompiler=gnu95 -c comsibc.f95 pardif.f95 sib2x.f95
↪  Sib2xa.f95 Sib2xb.f95 -m sib2pymod
```

2.5.1 Herança de valores e warm-up do modelo

Observou-se que as variáveis do SiB2 quando carregadas via python

```
import sib2pymod
resultado = sib2pymod.sib2()
```

não são declaradas novamente a cada solicitação da função "sib2()". As variáveis são declaradas apenas uma vez na importação. A cada execução da função "sib2()", as variáveis do SiB2 iniciam com os valores que ficaram na memória referente a rodada anterior.

Na figura 2 observe as pequenas diferenças entre os valores de Rn para uma primeira execução da função "sib2()" e uma segunda execução da função "sib2()".

Observe que devido os valores iniciais de alguma ou algumas variáveis serem diferentes, existem um pequeno desvio nos valores iniciais e que se estabilizam no decorrer dos cálculos, principalmente nos 30 primeiros passos de tempo.

Na segunda execução, pode ocorrer algo igual eu cito em 2.4, em que variáveis que deveriam ser inicializadas com determinados valores no primeiro passo de tempo do SiB2 não estão sendo inicializadas, como é o caso da variável itero, que deve ser = 0 no início dos cálculos, e a cada execução da função "sib2()".

Quando o SiB2 offline é executado na linha de comando apenas uma vez, não se observa problemas de herança de valores da rodada anterior. Agora na execução do SiB2 como um módulo de python, o problema de herança de valores da rodada anterior pode ser notado como mostra a figura 2.

Também deve-se considerar o *warm-up* do modelo SiB2 em rodadas consecutivas dentro do python. Variáveis como temperatura do solo e umidade do solo, possuem uma persistência maior no tempo do que as demais variáveis. Se o modelo inicia-se em um período úmido, mas a sua rodada anterior no ambiente python terminou em um período seco, os valores iniciais terão um efeito de *warm-up*.

2.6 Calibração do SiB2: uso de método de otimização de parâmetros do modelo.

O SiB2pymod foi criado especialmente para ser acoplado em um instrumento numérico de otimização de parâmetros que integra o ambiente Python, o LMFIT (NEWVILLE et al., 2014).

A calibração do SiB2 é realizada em 4 módulos:

- Módulo de radiação
- Módulo de momentum
- Módulo de umidade do solo
- Módulo de carbono e água

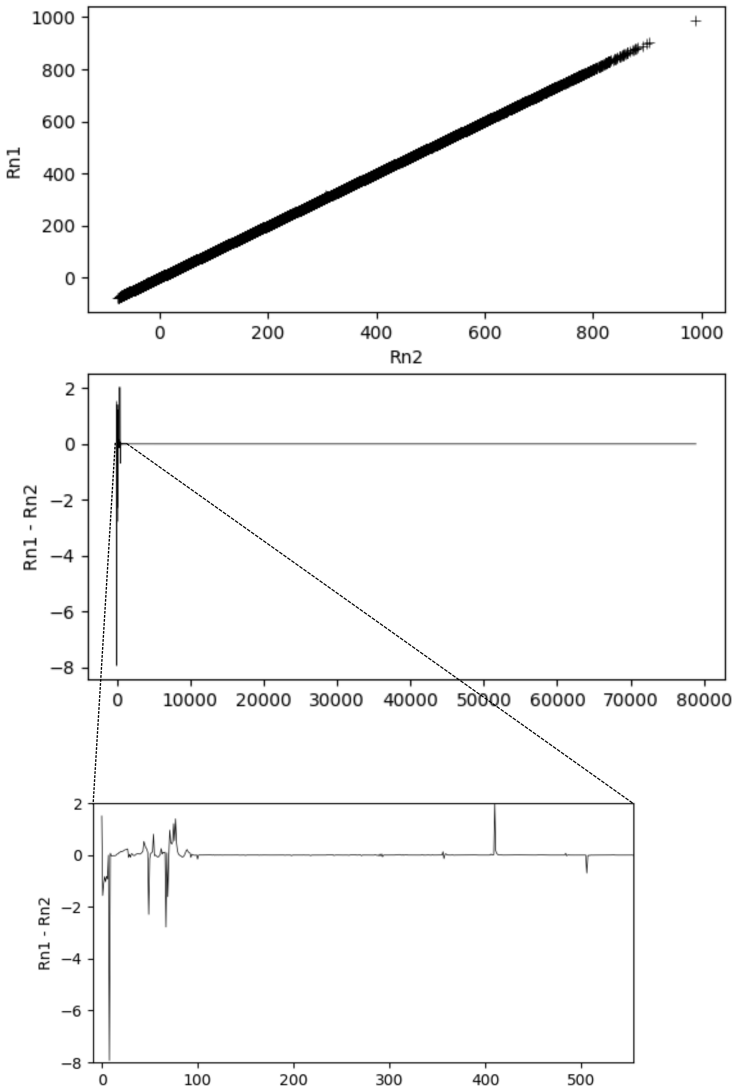

Figura 2 – Comparação entre o cálculo de Rn para a primeira execução da função python "sib2()" (Rn1) e na segunda execução da função python "sib2()" (Rn2) em um programa python. De cima para baixo, é mostrado a dispersão, desvio e ampliação do desvio.

O processo de otimização de parâmetros consiste basicamente na definição de uma função que determina um erro a ser minimizado. Diferentes métodos numéricos de minimização de erro podem ser utilizado, no entanto, aqui utiliza-se o método "brute" que calcula todas as possibilidades diante uma grade pré definida de parâmetros e o método "leastqr" que consiste no método de Levenberg-Marquardt.

Foi desenvolvido um SiB2pymod para cada módulo de calibração que se encontram em **srcsib2model**/SiB2pymod/. As rotinas para a calibração, que usam cada respectivo SiB2pymod estão em **srcLCB**/calibracaoSiB2/.

É importante sempre verificar os valores do "modeloerro" da função "residualSiB2". Erros

enormes, principalmente associados a valores inválidos (-99999.) comprometem todo o processo de otimização de parâmetros.

Mais detalhes a respeito dos métodos de minimização de erro da classe "Minimizer" do LMFIT e a respeito da definição do "modeloerro" por meio de uma função residual, aqui denominada como "residualSiB2", podem ser consultadas em ⟨https://lmfit.github.io/lmfit-py/⟩

2.6.1 Breve descrição das saídas da classe Minimizer

As saídas da classe "Minimizer" do LMFIT, que aplica os métodos numéricos de minimização de erro para otimização dos parâmetros tem o seguinte aspecto.

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 45
    # data points      = 1682
    # variables        = 4
    chi-square         = 295216.370
    reduced chi-square = 175.933474
    Akaike info crit   = 8700.11342
    Bayesian info crit = 8721.82438
[[Variables]]
    gradm:     15.9663468 +/- 0.01794231 (0.11%) (init = 16)
    gmudmu:    1.00159207 +/- 0.00151948 (0.15%) (init = 1)
    greeness:  0.96532180 +/- 0.00200255 (0.21%) (init = 0.99)
    vmax:      107.506586 +/- 0.20663948 (0.19%) (init = 105)
[[Correlations]] (unreported correlations are < 0.100)
    C(greeness, vmax)   = -0.565
    C(gmudmu, vmax)     =  0.482
    C(gradm, greeness)  = -0.424
    C(gradm, vmax)      = -0.174
    C(gmudmu, greeness) =  0.167
    C(gradm, gmudmu)    =  0.104
```

Sendo:

- fitting method: Método de minimização do erro em função dos parâmetros.
- function evals: Número de iterações, ie., quantas vezes a função residual foi chamada.
- data points: Tamanho da série de dados.

- variables: São os parâmetros do modelo a ser calibrado.
- chi-square: Somatório do erro quadrático do conjunto de parâmetros ótimos.
- reduced chi-square: χ^2 / (Número de pontos da série de dados) - (Número de parâmetros)
- Akaike info crit: Akaike Information Criterion statistic (aic)
- Bayesian info crit: Bayesian Information Criterion statistic (bic)
- [[Variables]]: O resultados dos parâmetros ótimos segue com o erro (+/-) que é calculado com base na matriz de covariância dos parâmetros e "init =" é o valor inicial do parâmetro.
- [[Correlations]]: Correlação linear de cada par de parâmetros a partir da matriz de covariância dos parâmetros.

Os melhores resultados de conjunto de parâmetros estarão associados com menores valores de chi-square (χ^2), aic e bic. Para mais informações a respeito das estatísticas presentes nos saídas da "Minimizer", consulte ⟨https://lmfit.github.io/lmfit-py/fitting.html⟩.

2.7 Cálculo do conjunto de parâmetros aerodinâmicos a cada passo de tempo: acoplamento da derive_trans do DBHM no SiB2.

A partir do *Distributed Biosphere-Hydrological (DBH) Model* disponível em ⟨http://hydro.iis.u-tokyo.ac.jp/DBH/⟩, o qual tem interface com o SiB2, utilizou-se a função "derive_trans.f" para o cálculo dos parâmetros aerodinâmicos relacionados as características da cobertura vegetal dos biomas.

A "derive_trans.f" do DBHM foi traduzida para Fortran 95 ("derive_trans.f95"), e acoplada no SiB2 para que o conjunto de parâmetros aerodinâmicos seja calculado conforme a variação do índice de área foliar a cada passo de tempo, sobrescrevendo os valores fixados do data1.

O SiB2 acoplado com a "derive_trans" está em **srcsib2model**/SiB2_DBHM_derive_trans/F95. O DBHM também encontra-se no repositório em **srcsib2model**/SiB2_DBHM_derive_trans/DBHModel.

A função "derive_trans" acoplada no SiB2 recebe os parâmetros

```
g1, z2, z1, chil, vcover, zlt, ztz0, vkc, gx, cpair, rhoair
```

e calcula

```
ha, z0d, dd, g2, g3, cc1, cc2, corb1, corb2
```

a cada passo de tempo.

Os parâmetros

```
zs     - Ground roughness length  (m)
zc     - Inflection height for leaf-area density(m)
leafw  - Leaf width (m) - zlw
leafl  - Leaf width (m) - zlen
```

devem ser prescritos no código “derive_trans.f95”. Estes são parâmetros que definem a geometria do dossel e são adicionais aos parâmetros que participam dos cálculos do SiB2.

Esta versão do SiB2, visa o aprimoramento dos cálculos para a agricultura e biomas em que a vegetação sofre mudanças acentuadas em escala sazonal. Também o uso da “derive_trans.f95” é uma alternativa ao programa “SIBX”.

2.8 Tradução do SiB2 para linguagem C com f2c

Pode ser conveniente traduzir o SiB2 do Fortan 77 para linguagem C e para isto pode-se utilizar o f2c (http://www.netlib.org/f2c/). Este processo tem se mostrado eficiente, evitando incompatibilidades de compiladores mais antigos de fortran para os mais atuais. Desta forma obtém-se o código em C e compila-se com o um compilador de C, gerando um binário executável independente de um compilador de Fortan.

Inicialmente os arquivos .for devem ser renomeados para .F. Após execute.

```
f2c -Nn802 *.F
```

Desta forma obtém-se os códigos fontes em C: sib2x.c, Sib2xa.c e Sib2xb.c. Para obter o executável, aqui nomeado com SiB2runF2C, execute.

```
gcc -c -o sib2x.o sib2x.c
gcc -c -o Sib2xa.o Sib2xa.c
gcc -c -o Sib2xb.o Sib2xb.c
gcc -o SiB2runF2C sib2x.o Sib2xa.o Sib2xb.o -lf2c -lm
```

Esta tradução também pode ser obtida utilizando o script fort77 em perl, obtido pelo pacote “fort77 - Invoke f2c like a real compiler” do debian.

3 PROCESSOS DE CALIBRAÇÃO DO SIB2

A calibração do SiB2 obedece uma sequência de etapas que inicia em executar o SiB2 com os parâmetros determinados sem calibração, execução do instrumento numérico de calibração para cada módulo determinando-se o conjunto de parâmetros ótimos para cada módulo, de forma que, os parâmetros calibrados no módulo anterior sejam incorporados para a calibração do módulo subsequente.

A sequência modular de calibração deve ser:

1. Módulo de radiação
2. Módulo de momentum
3. Módulo de umidade do solo
4. Módulo de carbono e água

Cada módulo de calibração do SiB2, consiste em uma versão do código do SiB2 em Fortan 95, modificada conforme a necessidade de cada módulo e compilada como uma função de Python 3.

Após a calibração de todos os parâmetros, executa-se o SiB2 para comparação dos resultados com a execução inicial do SiB2 sem calibração.

3.1 Execução do SiB2 sem calibração

1. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1".
2. Obtenha o código do SiB2 em **srcsib2model**/SiB2f95modvars, compile e gere o executável.

   ```
   gfortran comsibc.f95 pardif.f95 sib2x.f95 Sib2xa.f95 Sib2xb.f95 -o
   ↪  sib2run
   ```

3. Execute

   ```
   ./sib2run
   ```

   no diretório em que se encontra os arquivos "data1" e "data2".
4. Observe que o arquivo "sib2dt.dat" com as variáveis calculadas do SiB2 foi criado.

3.2 Calibração do Módulo de Radiação

1. Obtenha o código do módulo de radiação em **srcsib2model**/SiB2pymod/RadiativeTransModule.
2. Compile e gere o módulo de Python

   ```
   f2py3.7 --fcompiler=gnu95 -c *.f95 -m sib2pymod
   ```

   observe que um arquivo com nome similar à "sib2pymod.cpython-37m-x86_64-linux-gnu.so" deve ser gerado, que é o módulo de Python 3.
3. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1".
4. Obtenha o arquivo "data3.csv" com os dados observados que serão comparados com os valores calculados pelo SiB2 no processo de otimização de parâmetros.

5. Obtenha o programa "calibraSiB2_leastsq.py" para uso do método *Levenberg-Marquardt* ou "calibraSiB2_brute.py" para uso do método *brute* em **srcLCB**/calibracaoSiB2/radiation.

6. Execute o programa "calibraSiB2_leastsq.py"

   ```
   python3.7 calibraSiB2_leastsq.py
   ```

   ou o programa "calibraSiB2_brute.py".

   ```
   python3.7 calibraSiB2_brute.py
   ```

7. Observe que a saída deve ter a seguinte forma:

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 265
    # data points      = 26286
    # variables        = 11
    chi-square         = 12771313.4
    reduced chi-square = 486.063308
    Akaike info crit   = 162625.101
    Bayesian info crit = 162715.046
[[Variables]]
    tran_11:  0.02950107 +/- 0.02428086 (82.31%) (init = 0.017)
    tran_21:  0.19553723 +/- 0.01811426 (9.26%) (init = 0.2)
    tran_12:  0.06763576 +/- 0.16353388 (241.79%) (init = 0.001)
    tran_22:  0.09467824 +/- 0.14948139 (157.88%) (init = 0.001)
    ref_11:   0.06541928 +/- 0.01854913 (28.35%) (init = 0.07)
    ref_21:   0.49246437 +/- 0.01588988 (3.23%) (init = 0.5)
    ref_12:   0.34896582 +/- 0.13876444 (39.76%) (init = 0.16)
    ref_22:   0.02420730 +/- 0.04904501 (202.60%) (init = 0.39)
    soref_1:  0.01110515 +/- 0.01851393 (166.71%) (init = 0.11)
    soref_2:  0.01003659 +/- 0.00558512 (55.65%) (init = 0.225)
    chil:     0.14619933 +/- 0.00588779 (4.03%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(tran_21, ref_21)  = -0.985
    C(tran_11, ref_11)  = -0.969
    C(tran_21, tran_12) =  0.766
    C(tran_12, ref_21)  = -0.742
    C(tran_22, ref_22)  = -0.677
    C(tran_11, tran_12) = -0.670
    C(tran_11, ref_12)  = -0.573
    C(tran_12, tran_22) = -0.570
```

```
C(tran_12, ref_11)  =  0.557
C(tran_11, tran_22) =  0.523
C(tran_22, ref_12)  = -0.492
C(ref_11, ref_12)   =  0.476
C(tran_22, ref_11)  = -0.401
C(ref_21, soref_1)  = -0.400
C(tran_11, tran_21) = -0.371
C(tran_21, soref_1) =  0.358
C(ref_12, soref_1)  = -0.358
C(tran_11, ref_21)  =  0.333
C(tran_21, tran_22) = -0.302
C(tran_12, soref_1) =  0.281
C(ref_21, chil)     =  0.280
C(tran_12, ref_22)  =  0.277
C(tran_21, chil)    = -0.275
C(ref_11, soref_1)  = -0.266
C(soref_1, chil)    =  0.254
C(tran_11, soref_1) =  0.225
C(tran_21, ref_11)  =  0.212
C(ref_11, ref_21)   = -0.193
C(ref_11, chil)     =  0.191
C(ref_11, ref_22)   = -0.182
C(soref_1, soref_2) = -0.163
C(soref_2, chil)    = -0.161
C(tran_22, ref_21)  =  0.153
C(tran_22, chil)    = -0.138
C(ref_21, ref_12)   =  0.130
C(ref_12, chil)     =  0.120
C(tran_11, chil)    = -0.118
C(tran_22, soref_2) = -0.116
C(tran_12, ref_12)  =  0.113
C(tran_22, soref_1) =  0.106
```

8. Anote os valores dos parâmetros calibrados.

Obs.: O programa callSiB2pymodule.py em **srcsib2model**/SiB2pymod/RadiativeTransModule, é um exemplo de como o sib2pymod do módulo de radiação deve ser executado no ambiente Python.

3.3 Calibração do Módulo de Momentum

Aqui a calibração possui duas etapas.

3.3.1 Determinação dos parâmetros aerodinâmicos a serem calibrados conforme a variação do índice de área foliar no tempo com uso da derive_trans

1. Obtenha o código em **srcsib2model**/SiB2pymod/MomentumModule/zlt_aero_ts
2. Defina os valores dos parâmetros "zs", "zc", "leafw" e "leafl" em "derive_trans.f95", conforme as características do bioma.
3. Compile e gere o módulo de Python

   ```
   f2py3.7 --fcompiler=gnu95 -c *.f95 -m sib2pymod
   ```

4. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1".
5. Informe o número de linhas de dados do arquivo data2 no programa "callSiB2pymodule.py" e execute-o.

   ```
   python3.7 callSiB2pymodule.py
   ```

6. Note que o arquivo "zlt_aero_ts.dat" foi gerado, contendo o conjunto de parâmetros aerodinâmicos para cada passo de tempo.

3.3.2 Calibração dos parâmetros aerodinâmicos

1. Obtenha o código do módulo de momentum em **srcsib2model**/SiB2pymod/MomentumModule.
2. Compile e gere o módulo de Python

   ```
   f2py3.7 --fcompiler=gnu95 -c *.f95 -m sib2pymod
   ```

   observe que um arquivo com nome similar à "sib2pymod.cpython-37m-x86_64-linux-gnu.so" deve ser gerado, que é o módulo de Python 3.
3. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1" de parâmetros, porém, agora o arquivo "data1" deve ser editado para que contenha os valores dos parâmetros calibrados do módulo de radiação.
4. Obtenha o arquivo "data3.csv" com os dados observados que serão comparados com os valores calculados pelo SiB2 no processo de otimização de parâmetros.
5. Copie o arquivo "zlt_aero_ts.dat" gerado em 3.3.1 para o mesmo diretório do sib2pymod, "data1", "data2", "data3.csv".

6. Obtenha o programa "calibraSiB2_momentum_leastsq.py" para uso do método *Levenberg-Marquardt* ou "calibraSiB2_momentum_brute.py" para uso do método *brute* em **srcLCB**/calibracaoSiB2/momentum.

7. Execute o programa "calibraSiB2_momentum_leastsq.py"

   ```
   python3.7 calibraSiB2_momentum_leastsq.py
   ```

   ou o programa "calibraSiB2_momentum_brute.py".

   ```
   python3.7 calibraSiB2_momentum_brute.py
   ```

8. A saída deve ter a seguinte forma:

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 85
    # data points      = 732
    # variables        = 9
    chi-square         = 30.6064980
    reduced chi-square = 0.04233264
    Akaike info crit   = -2305.78390
    Bayesian info crit = -2264.42188
[[Variables]]
    ha:     23.8058861 +/- 0.56932611 (2.39%) (init = 23.53)
    z0d:    1.89174756 +/- 0.05014510 (2.65%) (init = 1.459)
    dd:     25.9954341 +/- 0.22468437 (0.86%) (init = 26.08)
    g2:     0.68042782 +/- 0.06066055 (8.92%) (init = 0.737)
    g3:     0.62291820 +/- 0.02801823 (4.50%) (init = 0.737)
    cc1:    10.3750000 +/- 1.8295e-14 (0.00%) (init = 10.375)
    cc2:    3324.78806 +/- 241.847705 (7.27%) (init = 2808.157)
    corb1:  13.1139182 +/- 0.65147631 (4.97%) (init = 12.643)
    corb2:  322.431000 +/- 2.4962e-13 (0.00%) (init = 322.431)
[[Correlations]] (unreported correlations are < 0.100)
    C(cc1, corb2)   = -1.000
    C(dd, g2)       = -0.990
    C(g3, cc2)      = -0.971
    C(ha, dd)       = -0.888
    C(ha, g2)       =  0.882
    C(ha, corb1)    = -0.845
    C(g2, corb2)    =  0.637
    C(g2, cc1)      = -0.634
    C(dd, corb2)    = -0.632
```

```
C(dd, cc1)      =  0.626
C(dd, corb1)    =  0.600
C(g2, corb1)    = -0.577
C(ha, corb2)    =  0.414
C(ha, cc1)      = -0.408
C(cc2, corb1)   = -0.344
C(g3, corb1)    =  0.312
C(z0d, g2)      = -0.245
C(z0d, corb1)   =  0.230
C(z0d, cc1)     =  0.220
C(z0d, corb2)   = -0.214
C(ha, cc2)      =  0.185
C(dd, cc2)      = -0.166
C(z0d, dd)      =  0.165
C(g3, corb2)    = -0.153
C(g3, cc1)      =  0.146
C(corb1, corb2) = -0.142
C(cc1, corb1)   =  0.132
C(cc2, corb2)   =  0.129
C(cc1, cc2)     = -0.121
C(ha, g3)       = -0.111
C(dd, g3)       =  0.105
C(g2, cc2)      =  0.104
```

9. Observe a criação do arquivo "params_calibrado.dat" com os valores dos parâmetros calibrados que deverá ser carregado nos módulos subsequentes.

3.4 Calibração do Módulo de Umidade do Solo

1. Obtenha o código do módulo de umidade do solo em **srcsib2model**/SiB2pymod/SoilMoistureModule.
2. Compile e gere o módulo de Python

   ```
   f2py3.7 --fcompiler=gnu95 -c *.f95 -m sib2pymod
   ```

   observe que um arquivo com nome similar à "sib2pymod.cpython-37m-x86_64-linux-gnu.so" deve ser gerado, que é o módulo de Python 3.
3. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1".
4. Obtenha o arquivo "data3.csv" com os dados observados que serão comparados com os valores calculados pelo SiB2 no processo de otimização de parâmetros.

5. Obtenha o arquivo "params_calibrado.dat" gerado em 3.3.2. Nesta etapa, não é necessário editar o arquivo "data1", pois os parâmetros aerodinâmicos calibrados são carregados pelo programa "load_aeropars.f95" a cada passo de tempo.

6. Obtenha o programa "calibraSiB2_leastsq.py" para uso do método *Levenberg-Marquardt* ou "calibraSiB2_brute.py" para uso do método *brute* em **srcLCB**/calibracaoSiB2/umidade.

7. Execute o programa "calibraSiB2_leastsq.py"

   ```
   python3.7 calibraSiB2_leastsq.py
   ```

   ou o programa "calibraSiB2_brute.py".

   ```
   python3.7 calibraSiB2_brute.py
   ```

8. Observe que a saída deve ter a seguinte forma:

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 46
    # data points      = 744
    # variables        = 4
    chi-square         = 247.572512
    reduced chi-square = 0.33455745
    Akaike info crit   = -810.651114
    Bayesian info crit = -792.202950
[[Variables]]
    bee1:    6.13789646 +/- 19.0310266 (310.06%) (init = 7.12)
    phsat1: -0.01000158 +/- 0.00693809 (69.37%) (init = -0.2)
    satco1:  9.3885e-05 +/- 5.8644e-04 (624.64%) (init = 5e-06)
    poros1:  0.30878321 +/- 0.01715190 (5.55%) (init = 0.515)
[[Correlations]] (unreported correlations are < 0.100)
    C(phsat1, satco1) =  0.974
    C(bee1, phsat1)   = -0.377
    C(bee1, poros1)   =  0.325
    C(bee1, satco1)   = -0.156
```

9. Anote os valores dos parâmetros calibrados.

3.5 Calibração do Módulo de Carbono e Água

1. Obtenha o código do módulo de carbono e água em **srcsib2model**/SiB2pymod/CarbonWaterModule.

2. Compile e gere o módulo de Python

```
f2py3.7 --fcompiler=gnu95 -c *.f95 -m sib2pymod
```

observe que um arquivo com nome similar à "sib2pymod.cpython-37m-x86_64-linux-gnu.so" deve ser gerado, que é o módulo de Python 3.

3. Obtenha os arquivos de entrada do SiB2: arquivo "data2" e o arquivo "data1".

4. Obtenha o arquivo "data3.csv" com os dados observados que serão comparados com os valores calculados pelo SiB2 no processo de otimização de parâmetros.

5. Obtenha o arquivo "params_calibrado.dat" gerado em 3.3.2. Agora, além dos parâmetros aerodinâmicos serem carregados pelo programa "load_aeropars.f95", é necessário editar o arquivo "data1" com os valores dos parâmetros obtidos no módulo de calibração de umidade do solo. x

6. Obtenha o programa "calibraSiB2_leastsq.py" para uso do método *Levenberg-Marquardt* ou "calibraSiB2_brute.py" para uso do método *brute* em **srcLCB**/calibracaoSiB2/carbonoagua.

7. Execute o programa "calibraSiB2_leastsq.py"

   ```
   python3.7 calibraSiB2_leastsq.py
   ```

   ou o programa "calibraSiB2_brute.py".

   ```
   python3.7 calibraSiB2_brute.py
   ```

8. Observe que a saída deve ter a seguinte forma:

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 64
    # data points      = 667
    # variables        = 4
    chi-square         = 106689.129
    reduced chi-square = 160.918747
    Akaike info crit   = 3392.94796
    Bayesian info crit = 3410.95912
[[Variables]]
    gradm:     20.9092373 +/- 2.09814850 (10.03%) (init = 16)
    gmudmu:    0.72108549 +/- 0.04779871 (6.63%) (init = 1)
    greeness:  0.99993604 +/- 8.8220e-06 (0.00%) (init = 0.99)
    vmax:      108.073123 +/- 14.0153431 (12.97%) (init = 105)
[[Correlations]] (unreported correlations are < 0.100)
    C(gradm, vmax)     = -0.797
    C(gmudmu, vmax)    = -0.404
```

```
C(gradm, gmudmu)   = -0.231
C(greeness, vmax)  =  0.186
C(gradm, greeness) = -0.183
```

3.6 Execução do SiB2 calibrado

1. Obtenha os arquivos de entrada do SiB2 calibrado: arquivo "data2", arquivo "data1" com os respectivos parâmetros calibrados e o arquivo "params_calibrado.dat" obtido em 3.3.2.

2. Obtenha o código do SiB2 em **srcsib2model**/SiB2f95momentum, compile e gere o executável:

   ```
   gfortran comsibc.f95 pardif.f95 load_aeropars.f95 sib2x.f95
   ↪  Sib2xa.f95 Sib2xb.f95 -o sib2run
   ```

3. Execute

   ```
   ./sib2run
   ```

4. Observe que os arquivo "sib2dt.dat" com as variáveis calculadas do SiB2 calibrado foi criado.

4 CALIBRAÇÃO DE MODELO DE BALANÇO DE ÁGUA PARA PEQUENAS BACIAS

O modelo de balanço de água descrito em Vandewiele, Xu et al. (1992), foi reproduzido no ambiente python. Nesta formulação, os parâmetros do modelo são obtidos por um processo numérico de minimização de erros com base em dados observacionais de evapotranspiração potencial, precipitação e vazão. Ao invés do uso do pacote computacional "VA05A" foi utilizado os métodos "brute" e "Levenberg-Marquardt" do LMFI (NEWVILLE et al., 2014) para obtenção dos parâmetros do modelo de balanço de água.

O formulação numérica do modelo está no repositório **hidromodel**, nos códigos "balagua_ugrhi_brute.py" e "balagua_ugrhi_leastsq.py" que correspondem ao mesmo modelo de balanço de água de Vandewiele, Xu et al. (1992) já acoplado com o método de obtenção de parâmetros "brute" e "Levenberg-Marquardt" respectivamente.

4.1 Obtendo os parâmetros do modelo de balanço de água

Os parâmetros do modelo são obtidos com base na série temporal mensal de dados observacionais de evapotranspiração potencial (etp), precipitação (p) e vazão (q). Aqui, a evapotranspiração potencial foi obtida da *Climatic Research Unit (University of East Anglia) and Met Office* (⟨http://www.cru.uea.ac.uk/⟩) ou da *Texas A&M University* ⟨https://utexas.app.box.

com/v/xavier-etal-ijoc-data⟩ e os dados de vazão e precipitação, obtidos das UGRHI (Unidade de Gerenciamento de Recursos Hídricos de São Paulo) e estações meteorológicas do INMET (Instituto Nacional de Meteorologia). Porém essa combinação pode variar para diferentes bacias hidrográficas, método de obtenção da evapotranspiração potencial, etc.

O código fonte "ts_to_hidromodel_ugrhi_cru.py" em **srcLCB**/utils vem sendo utilizado para criar as séries de dados observacionais de entrada para o modelamento das bacias hidrográficas que pertencem as UGRHI do estado de São Paulo no LCB e poderá auxiliar em futuras construções.

Após a construção da série de dados de etp, p e q, executa-se

```
python3.7 balagua_ugrhi_leastsq.py
```

ou

```
python3.7 balagua_ugrhi_brute.py
```

conforme o método de otimização de parâmetros "Levenberg-Marquardt" ou "brute", o qual deseja-se.

O programa "loadMinimizerResult_checkCandidates.py" em **hidromodel** é uma adaptação do exemplo denominado como *Global minimization using the brute method (a.k.a. grid search)*, o qual faz uma avaliação das saídas da classe "Minimizer" para o método "brute". Este tipo de avaliação permite uma visualização gráfica das soluções conforme mostra a figura 3. Para mais detalhes, consulte ⟨https://lmfit.github.io/lmfit-py/examples/example_brute.html?highlight=brute%20example⟩

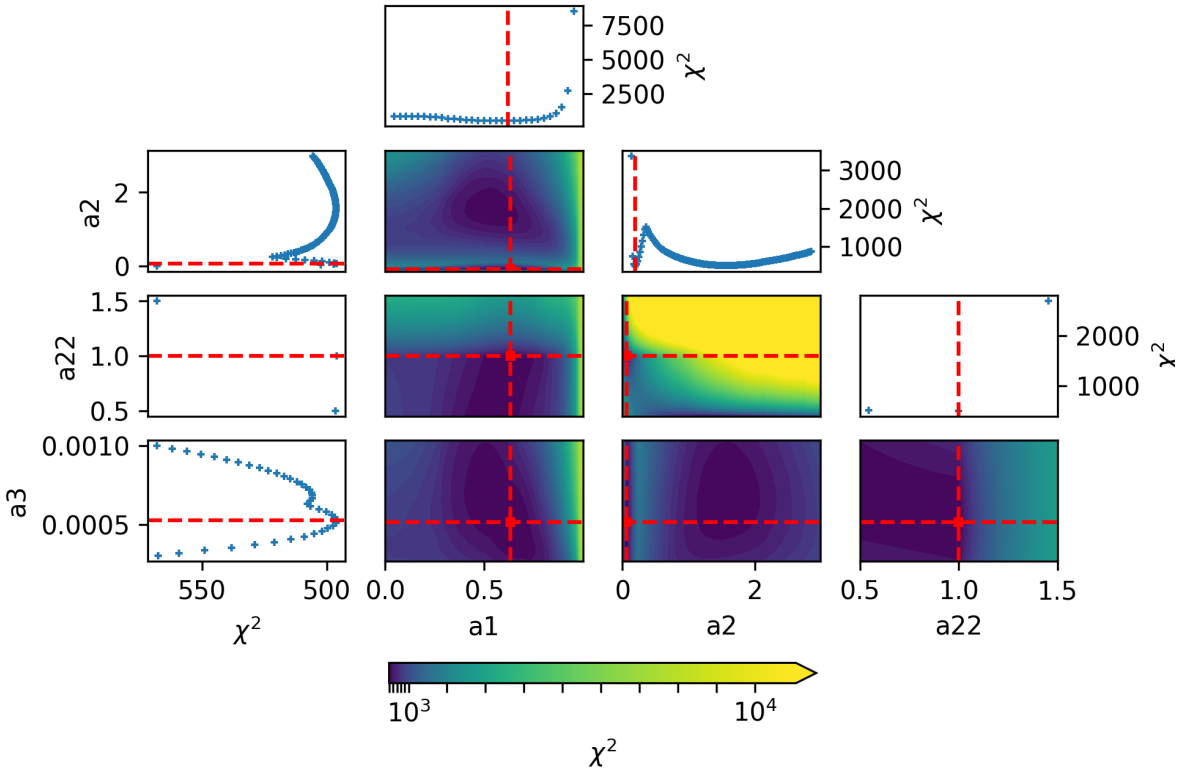

Figura 3 – Visualização gráfica dos valores dos parâmetros associados ao erro mínimo, com base nos cálculos do método "brute" da classe "Minimizer" do python LMFIT.

REFERÊNCIAS

NEWVILLE, M. et al. *LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python*. Zenodo, 2014. Disponível em: ⟨https://doi.org/10.5281/zenodo.11813⟩.

SELLERS, P. et al. A revised land surface parameterization (sib2) for atmospheric gcms. part i: Model formulation. *Journal of climate*, v. 9, n. 4, p. 676–705, 1996.

SELLERS, P. J. et al. A revised land surface parameterization (sib2) for atmospheric gcms. part ii: The generation of global fields of terrestrial biophysical parameters from satellite data. *Journal of climate*, v. 9, n. 4, p. 706–737, 1996.

VANDEWIELE, G.; XU, C.-Y. et al. Methodology and comparative study of monthly water balance models in belgium, china and burma. *Journal of Hydrology*, Elsevier, v. 134, n. 1-4, p. 315–347, 1992.