Trabalho 2 - ICC

Matheus Pacheco dos Santos e Luzia Millena Santos Silva

15 de Dezembro de 2021

1 Matriz de Derivadas Parciais

Seja o sistema não linear de instâncias de *Função Tridiagonal de Broyden*:

$$\begin{cases} (3 - 2 \cdot x_1) \cdot x_1 - 2 \cdot x_2 + 1 \\ (3 - 2 \cdot x_2) \cdot x_2 - x_1 - 2 \cdot x_3 + 1 \\ (3 - 2 \cdot x_3) \cdot x_3 - x_2 - 2 \cdot x_4 + 1 \\ (3 - 2 \cdot x_3) \cdot x_4 - x_3 + 1 \end{cases}$$

através desse sistema a matriz de derivadas parciais irá ficar da forma:

$$\begin{pmatrix} -2x_1 + 3 & -2 & 0 & 0 \\ -1 & -2x_2 + 3 & -2 & 0 \\ 0 & -1 & -2x_3 + 3 & -2 \\ 0 & 0 & -1 & -2x_4 + 3 \end{pmatrix}$$

ou seja, a matriz de derivadas parciais de um sistema de instâncias de *Função Tridiagonal de Broyden*, é uma matriz tridiagonal para qualquer que seja a dimensão desta matriz.

Note que podemos guardar essa matriz tridiagonal com apenas três vetores:

$$\begin{aligned} D_1 &= \begin{bmatrix} -2 & -2 & -2 \end{bmatrix} \\ D_2 &= \begin{bmatrix} -2x_1 + 3 & -2x_2 + 3 & -2x_3 + 3 & -2x_4 + 3 \end{bmatrix} \\ D_3 &= \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \end{aligned}$$

note que D_1 e D_3, tem $n - 1$ elementos no vetor, sendo $n = 3$.

Sendo assim, podemos descrever uma matriz de derivadas parciais de um sistema não linear de instâncias de *Função Triadiagonal de Broyden* com apenas 3 vetores da seguinte forma:

$$\begin{aligned} D_1 &= \begin{bmatrix} -2 & -2 & -2 & \dots & -2 \end{bmatrix} \\ D_2 &= \begin{bmatrix} -2x_1 + 3 & -2x_2 + 3 & -2x_3 + 3 & -2x_4 + 3 & \dots & -2x_n + 3 \end{bmatrix} \\ D_3 &= \begin{bmatrix} -1 & -1 & -1 & \dots & -1 \end{bmatrix} \end{aligned}$$

com D_2 com n elementos e D_1, D_3 com $n - 1$ elementos.

Veja que poderíamos otimizar ao máximo pelo fato de que D_1, D_2 e D_3 terem certas "propriedades"como:

- Todo elemento de D_1 é igual -2.
- Todo elemento de D_2 é da forma $-2x_i + 3$.
- Todo elemento de D_3 é igual -1.

Mas para fins didáticos orientados pelo professor vamos calcular as três diagonais para termos uma comparação com o que foi otimizado.

Vamos agora mostrar como fizemos a nossa função que gera as matrizes parciais.

Veja que podemos gerar as três diagonais da seguinte forma,

```c
void geraMatTridiagonalDerivParcial(void ****diagonais, SNL sistema) {
      for (int i = 0; i < sistema.numFuncoes; i++) {
        sprintf(var,"x%d", i+1);
        (*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
    }

    for (int i = 0; i < sistema.numFuncoes - 1; i++) {
        sprintf(var,"x%d", i+2);
        (*diagonais)[0][i] = evaluator_derivative(sistema.funcoes[i], var);
    }

    for (int i = 0; i < sistema.numFuncoes-1; i++) {
        sprintf(var,"x%d", i+1);
        (*diagonais)[2][i] = evaluator_derivative(sistema.funcoes[i+1], var);
    }
}
```

mas veja que podemos fazer a fusão dos laços,

```c
int i = 0;
for (i = 0; i < sistema.numFuncoes - 1; i++) {
    sprintf(var,"x%d", i+2);
    (*diagonais)[0][i] = evaluator_derivative(sistema.funcoes[i], var);

    sprintf(var,"x%d", i+1);
    (*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
    (*diagonais)[2][i] = evaluator_derivative(sistema.funcoes[i+1], var);
}
sprintf(var,"x%d", i+1);
(*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
```

e agora aplicamos *Unroll & Jam*,

```c
void geraMatTridiagonalDerivParcial(void ****diagonais, SNL sistema) {
    int i = 0;
    int limit = (sistema.numFuncoes - 1) - (sistema.numFuncoes - 1) % 4;
    for (i = 0; i < limit; i += 4) {
        sprintf(var,"x%d", i+2);
        (*diagonais)[0][i] = evaluator_derivative(sistema.funcoes[i], var);
        sprintf(var,"x%d", i+1);
        (*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
        (*diagonais)[2][i] = evaluator_derivative(sistema.funcoes[i+1], var);

        sprintf(var,"x%d", i+3);
        (*diagonais)[0][i+1] = evaluator_derivative(sistema.funcoes[i+1], var);
        sprintf(var,"x%d", i+2);
        (*diagonais)[1][i+1] = evaluator_derivative(sistema.funcoes[i+1], var);
        (*diagonais)[2][i+1] = evaluator_derivative(sistema.funcoes[i+2], var);

        sprintf(var,"x%d", i+4);
        (*diagonais)[0][i+2] = evaluator_derivative(sistema.funcoes[i+2], var);
        sprintf(var,"x%d", i+3);
```

```
        (*diagonais)[1][i+2] = evaluator_derivative(sistema.funcoes[i+2], var);
        (*diagonais)[2][i+2] = evaluator_derivative(sistema.funcoes[i+3], var);

        sprintf(var,"x%d", i+5);
        (*diagonais)[0][i+3] = evaluator_derivative(sistema.funcoes[i+3], var);
        sprintf(var,"x%d", i+4);
        (*diagonais)[1][i+3] = evaluator_derivative(sistema.funcoes[i+3], var);
        (*diagonais)[2][i+3] = evaluator_derivative(sistema.funcoes[i+4], var);
    }

    // Resíduos
    for (i = limit; i < sistema.numFuncoes - 1; i++) {
        sprintf(var,"x%d", i+2);
        (*diagonais)[0][i] = evaluator_derivative(sistema.funcoes[i], var);
        sprintf(var,"x%d", i+1);
        (*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
        (*diagonais)[2][i] = evaluator_derivative(sistema.funcoes[i+1], var);
    }
    sprintf(var,"x%d", i+1);
    (*diagonais)[1][i] = evaluator_derivative(sistema.funcoes[i], var);
}
```

chegando na noss função final. Note que a diagonal superior está no indice 0, principal no 1 e inferior no 2.