

## Overview

This week we mainly focus on the code and Cholesky factorization. We read Chapter 11 from the book and refer some important ideas. For the code part, we have worked on Presolving part specially, and it works very well. Then, we write Cholesky function. Since our test cases are used by sparse matrix, Sparse Cholesky factorization is important to know for this project.

## Three Forms of the Step Equation

Here we just put three forms of the step equations.

The unreduced form is:

$$\begin{bmatrix} 0 & A & 0 \\ A^T & 0 & I \\ 0 & S & X \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -r_{xs} \end{bmatrix}$$

, where  $r_{xs} = XSe - \sigma\mu e$  for the generic primal-dual step.

By eliminating  $\Delta s$  and using the notation  $D = S^{-1/2}X^{1/2}$ , we get the augmented system form:

$$\begin{bmatrix} 0 & A \\ A^T & -D^{-2} \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c + X^{-1}r_{xs} \end{bmatrix}$$

$$\Delta s = -X^{-1}(r_{xs} + S\Delta x)$$

Finally, by eliminating  $\Delta x$ , we get the most compact of the three forms:

$$AD^2A^T\Delta\lambda = -r_b + A(-S^{-1}Xr_c + S^{-1}r_{xs})$$

$$\Delta s = -r_c - A^T\Delta\lambda$$

$$\Delta x = -S^{-1}(r_{xs} + X\Delta s)$$

## Cholesky Factorization

Let  $M$  denote

$$AD^2A^T\Delta\lambda = -r_b + A(-S^{-1}Xr_c + S^{-1}r_{xs})$$

after a possible symmetric reordering of its rows and columns

$$M = P(AD^2A^T)P^T$$

, where  $P$  is an  $m \times m$  permutation matrix.

$$M = LL^T$$

where  $L$  is a lower triangular matrix with positive diagonal elements.

By the pseudocode of Cholesky algorithm, we can write the update step as

$$M_{i+1:m, i+1:m} \leftarrow M_{i+1:m, i+1:m} - M_{i+1:m, i} M_{i+1:m, i}^T / M_{ii}$$

where  $M_{ii}$  is the pivot element.

Given the factorization  $M = LL^T$ , we can solve the linear system

$$Mz = r$$

by using two triangular substitutions:

solve  $Ly = r$  to find  $y$

solve  $L^T z = y$  to find  $z$

## Code Updates

In last week, we found out some presolving part mentioned in the recommended book *Primal-Dual Interior-Point Methods* by Stephen Wright is a little confusing for implementation, for example, the row singleton part is removing a row and a column, there are some edge cases like removing these singleton rows ends up to have an empty matrix. Therefore, we decided not to implement this part for now. We did add one more presolving that removes zero rows, which is much simpler than removing zero columns since it does not produce any x value but only reducing the size of the problem:

```
function remove_zero_rows(problem::IpIpProblem)
    zero_rows = findall(sum(problem.A .!= 0, dims=2)[:,:] .== 0)
    non_zero_rows = setdiff(1:size(problem.A, 1), zero_rows)
    problem.A = problem.A[non_zero_rows, :]
    problem.b = problem.b[non_zero_rows]
    return problem
end
```

Then we moving forward to the Cholesky Factorization part of the code, we referenced to the code from lecture and made some modifications:

```
function modified_cholesky(A::SparseMatrixCSC{Float64}, delta::Real, beta::Real)
    @assert (n=size(A, 1)) == size(A, 2) "Input matrix must be a square matrix!"

    reorder = amd(A)
    L = A[reorder, reorder]

    d = ones(n)
    D = Diagonal(d)

    for i = 1:n-1
        theta = maximum(abs.(L[i+1:n, i]))
        d[i] = max(abs(L[i, i]), (theta / beta)^2, delta)
        L[i:n, i] ./= d[i]
        L[i, i] = 1.0
        L[i+1:n, i+1:n] .-= d[i] * (L[i+1:n, i] * L[i+1:n, i]')
    end

    d[n] = max(abs(L[n, n]), delta)
    L[n, n] = 1.0

    return (L=LowerTriangular(L), D=D, M=L, O=reorder)
end
```


Considering the robustness, we use assertions to avoid failures. In addition, we add a preprocess of reordering the rows and columns using AMD algorithm to reduce number of fill-in of the Cholesky Factorization as discussed in previous work: *An Approximate Minimum Degree Ordering Algorithm* and AMD can also improve accuracy of the Cholesky factorization. With AMD, we can return the approximate minimum degree permutation for the matrix A for later linear system solving.

We ran a quick test:

```
# Get data and problem
md = mdopen("LPnetlib/lp_afiro")
pb = convert_matrixdepot(md)
stdpb = convert_to_standard_form(pb) # this function was in last report
M = stdpb.A*stdpb.A' # make M symmetric and reduce the cost
F = modified_cholesky(M, 0.1, 5.0)
sparse(F.L)
```

The result we obtained is:

```
In [326]: sparse(F.L)
Out[326]: 27x27 SparseMatrixCSC{Float64, Int64} with 113 stored entries:
```



We also ran a similar test that was used in the lecture:

```
A = [3 -1.0 -1.0; -1 3.0 -1; -1 -1 3.0]
F = modified_cholesky(sparse(A), 0.1, 5.0)
Matrix(F.L)*Matrix(F.D)*(Matrix(F.L))' - A
```

which outputs:

```
In [395]: Matrix(F.L)*Matrix(F.D)*(Matrix(F.L))' - A
Out[395]: 3x3 Matrix{Float64}:
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0
```

and looks correct. One more thing to note here is that the selection of delta and beta mainly depends on the problem itself, we are trying to implement a heuristic approach of the selection which will be discussed in the final report since we only test on the small-sized dataset at this point, there is no significant difference between different selection.

## Planning

We have done a core part and starting the starting point in this week, although we did not finish the starting point selection yet. For the next week, we are planning to finish the starting point, and heading into the solver with predictor-corrector strategy.