Practical Implementation of CG
Enforcing boundary conditions / constraints

Maybe some components of the solution vector $x$ are already known, e.g. if parts of the mesh are directly moved as a user-imposed manipulation.

Constrained "handles"

Without loss of generality, assume that the particles (w/ values $x$) are partitioned into free ($x_F$) and constrained ($x_c$). Then the system $Ax = b$ is written in block form as:

\[
\begin{pmatrix}
A_{FF} & A_{Fc} \\
A_{CF} & A_{Cc}
\end{pmatrix}
\begin{pmatrix}
x_F \\
x_c
\end{pmatrix}
=
\begin{pmatrix}
b_F \\
b_c
\end{pmatrix}
\]

If we are given externally imposed values $x_c^*$ for the constrained particles, the bottom block becomes $x_c = x_c^*$.
In matrix terms, the system becomes:

\[
\begin{pmatrix}
A_{FF} & A_{FC} \\
0 & I
\end{pmatrix}
\begin{pmatrix}
x_F \\
x_c
\end{pmatrix} =
\begin{pmatrix}
b_F \\
x_c^*
\end{pmatrix}
\]

This is a perfectly valid & solvable system, but just not with CG since we compromised the matrix symmetry!

**Alternative** Rework the 1st block of equations as follows:

\[
A_{FF} x_F = b_F - A_{FC} x_c^*
\]

\[
\Rightarrow \begin{pmatrix}
A_{FF} & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
\delta_F \\
\delta_c
\end{pmatrix} =
\begin{pmatrix}
b_F - A_{FF} \delta_F - A_{FC} x_c^* \\
0
\end{pmatrix}
\]

\[
A^* = \text{Project} \left[ b - A \begin{pmatrix}
x_F^{(o)} \\
x_c^*
\end{pmatrix} \right]
\]

This matrix is symmetric.
Although we don't explicitly have matrix $A^*$, we can emulate the multiplication $A^*w$ as:

Multiply with $A^*$ $[q \leftarrow A^*w]$

\[
\begin{align*}
    w^* \leftarrow & \text{Project (w)} \\
    q^* \leftarrow & A \cdot w^* \\
    q \leftarrow & \text{Project (q*)}
\end{align*}
\]

(which is all that CG requires, anyways)

The algorithm in page 1 results from eliminating "redundant" (repeated) projections.
Backward Euler as a correction

Assumptions:

1. Linear Elasticity (for now)
2. Presume initial guesses for $x^{n+1}$ ($x_{(o)}$) and $v^{n+1}$ ($v_{(o)}$)
3. We presume initial guesses satisfy B.E:
   \[ x_{(o)} = x^n + h v_{(o)}^{n+1} \] (1)
   (e.g. use $v_{(o)}^{n+1} = v^n$, and $x_{(o)}^{n+1}$ from above)
4. We seek corrections:
   \[ \delta x := x^{n+1} - x_{(o)} \]
   \[ \delta v := v^{n+1} - v_{(o)}^{n+1} \]

\[ x^{n+1} = x^n + h v_{(o)}^{n+1} \] (1)
\[ v^{n+1} = v^n + h M^{-1} \left\{ -K (x^{n+1} - x) - f K v^{n+1} \right\} \]
\[ \Rightarrow \delta v = v^n - v_{(o)}^{n+1} + h M^{-1} \left\{ -K (\delta x + x_{(o)}^{n+1} - x) \\ - f K (v_{(o)}^{n+1} + \delta v) \right\} \] (1)
\[ \Rightarrow \]
\[ \frac{1}{h} \delta x = v^n - v^{n+1}_{(o)} + h M^{-1} \left\{ -K \delta x - \frac{\partial K \delta x}{\partial \xi} \right. \\
\left. \quad + f_e (x^{n+1}_{(o)}) + f_d (v^{n+1}_{(o)}) \right\} \]

\[ \Rightarrow \left[ \left( 1 + \frac{\frac{h}{h}}{h} \right) K + \frac{1}{h^2} M \right] \delta x \]

\[ = \frac{1}{h} M \left[ v^n - v^{n+1}_{(o)} \right] + f_e (x^{n+1}_{(o)}) + f_d (v^{n+1}_{(o)}) \]