Nonlinear problems
&
constraints

Riccardo Rossi & Pooyan Dadvand

Newton-Raphson Method

Let’s consider we have a problema of the type

$$f(x) = 0$$

Our goal is to find “x” such that such equation is verified.

Newton method is a technique to find the solution of such type of problems with optimal convergence rate.

Newton-Raphson method

The idea is that if f is smooth, we can asume that

$$0 = f(x) \approx f(x_i + dx) \approx f(x_i) + \frac{\partial f}{\partial x} dx + O(dx^2)$$

This means that given an initial guess x_i

We can find a correction dx as

$$-\frac{\partial f}{\partial x} dx = f(x_i)$$

A practical example

Let's imagine we want to find the solution of

$$x^2 = 2$$

Of course we know the solution … $x = \sqrt{2}$,

however … can we solve this by NR?

A practical example

First of all we need to cast the problema in residual form

$$f(x) = 2 - x^2$$

NR only finds zeros!!

$$LHS = -\frac{\partial f}{\partial x} = 2x$$
$$RHS = 2 - x^2$$

A practical example

Let’s assume that our first guess is $x_0 = 1.0$

x	LHS	RHS	dx
1.0	2.0	1.0	0.5
1.5	3.0	-0.25	-0,083333
1,41666	2,833333	-0,006943	-0,00245
1,414	…	…	…

Does it always converge?

Let's try to compute the cubic root of 2:

$$f(x) = 2 - x^3$$

We start also from $x_0 = 1.0$

x	LHS	RHS	dx
1	3	1	0,33333333
1,33333333	4	-0,37037037	-0,09259259
1,24074074	3,72222222	0,08995707	0,02416757
1,26490831	3,79472493	-0,02384449	-0,00628359
1,25862472	3,77587417	0,00616702	0,00163327
1,26025799	3,78077398	-0,00160502	-0,00042452
1,25983347	3,77950041	0,00041704	0,00011034
1,25994381	3,77983144	-0,00010841	-2,868E-05
1,25991513	3,7797454	2,8177E-05	7,4546E-06

Does it always converge?

However if we start from $x_0 = 5.0$, which is farther away from the solution, we get

x	LHS	RHS	dx
5	15	-123	-8,2
-3,2	-9,6	34,768	-3,62166667
-6,82166667	-20,465	319,447187	-15,6094399
-22,4311065	-67,2933196	11288,3131	-167,747901
-190,179007	-570,537022	6878406,75	-12056,0218
-12246,2008	-36738,6023	1,8366E+12	-49989811,2
-50002057,4	-150006172	1,2502E+23	-8,334E+14
-8,334E+14	-2,5002E+15	5,7885E+44	-2,3152E+29
-2,3152E+29	-6,9456E+29	1,241E+88	-1,7867E+58

CONCLUSION

Newton Raphson:

- Converges very fast
  - Approximatimatively doubles accurate digits at each iteration
- May diverge equally fast if initial guess is not good enough
- It is not very robust…

Multidimensional case

Let's suppose we want to solve the following set of equations:

$$x^2 + y^2 = 1$$
$$x - y = 0$$

This represents the intersection of a unit circle with a line. Solutions are (1,1), (-1,-1)

Multidimensional case

To apply NR we must cast this in residual form.

$$R(x,y) \coloneqq \begin{pmatrix} 1 - x^2 - y^2 \\ y - x \end{pmatrix}$$

We need now to compute the gradient of R:

$$LHS \coloneqq \begin{pmatrix} -{}^{\partial R_0}/_{\partial x} & -{}^{\partial R_0}/_{\partial y} \\ -{}^{\partial R_1}/_{\partial x} & -{}^{\partial R_1}/_{\partial y} \end{pmatrix}$$

Multidimensional case

To apply NR we must cast this in residual form.

$$R(x,y) := \begin{pmatrix} 1 - x^2 - y^2 \\ y - x \end{pmatrix}$$

We need now to compute the gradient of R:

$$LHS := \begin{pmatrix} -\dfrac{\partial R_0}{\partial x} & -\dfrac{\partial R_0}{\partial y} \\ -\dfrac{\partial R_1}{\partial x} & -\dfrac{\partial R_1}{\partial y} \end{pmatrix}$$

NR iteration

The NR iteration thus proceeds as follows:

1. Choose a starting guess (x_i, y_i)
2. Evaluate $\mathrm{RHS}(x_i, y_i)$ & $\mathrm{LHS}(x_i, y_i)$
3. Solve $LHS \begin{pmatrix} dx \\ dy \end{pmatrix} = RHS$
4. Update solution $\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + dx \\ y_i + dy \end{pmatrix}$
5. check converged & eventually go back to 1

Multidimensional case

For the specific case at hand, this gives

$$LHS := \begin{pmatrix} 2x & 2y \\ 1 & -1 \end{pmatrix}$$

x	y	lhs		inv		rhs	dx
1	0	2	0	0,5	0	0	0
ITERATION 1		1	-1	0,5	-1	-1	1
1	1	2	2	0,25	0,5	-1	-0,25
ITERATION2		1	-1	0,25	-0,5	0	-0,25
0,75	0,75	1,5	1,5	0,33333333	0,5	-0,125	-0,04166667
ITERATION 3		1	-1	0,33333333	-0,5	0	-0,04166667
0,70833333	0,70833333	1,41666667	1,41666667	0,35294118	0,5	-0,00347222	-0,00122549
ITERATION 4		1	-1	0,35294118	-0,5	0	-0,00122549
0,70710784	0,70710784	1,41421569	1,41421569	0,35355286	0,5	-3,0037E-06	-1,0619E-06
ITERATION 5		1	-1	0,35355286	-0,5	0	-1,0619E-06
0,70710678	**0,70710678**	1,41421356	1,41421356	0,35355339	0,5	-2,2553E-12	-7,9737E-13
ITERATION 6		1	-1	0,35355339	-0,5	0	-7,9737E-13
0,70710678	**0,70710678**	1,41421356	1,41421356	0,35355339	0,5	0	0
ITERATION 7		1	-1	0,35355339	-0,5	0	0

FEM problems

Let's now focus on the FEM. A typical, potentially nonlinear, FEM problem gives:

$$\boldsymbol{RHS(u) = b \ - K(u)u}$$

If we apply NR technique

$$\boldsymbol{LHS(u) := -\frac{\partial b - K(u)u}{\partial u} = -\frac{\partial K(u)}{\partial u}u - K(u)}$$

Normally not done this way since $\boldsymbol{\frac{\partial K(u)}{\partial u}}$ is a third order tensor

PICARD’s method

It is common to simplify the previous to:

$$\boldsymbol{RHS(u) = b \; - K(u)u}$$

$$\boldsymbol{LHS(u) =} \cancel{-\frac{\partial K(u)}{\partial u} u} \boldsymbol{- K(u)}$$

Giving rise to a method called **PICARD’s method.** Note that this equivalent to doing:

$$\boldsymbol{K(u_{old})u = b}$$

PICARD’s method

- **Conceptually simple**
- **No need to compute the exact Jacobian**
- **Convergence more robust than NR (often it converges starting from a worst initial approximation to the solution)**

On the bad side:

- **Slow convergence**

APPLICATION OF CONSTRAINTS

Let’s consider a simple linear problem

1D laplacian problem (a value of 5 would be applied on a node u_0)

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix}$$

For whatever reason we want to impose that $u_3 = u_1 + 1$

HOW DO WE DO IT?

Application of constraint

Since the constraint is linear in the unknowns, we can express it as

$$\begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = 1$$

Which is normally written in matrix form as

$$\boldsymbol{Hu} = \boldsymbol{e} \text{ (we'll assume } e = 1\text{)}$$

If we assume that the previous problem is expressed as $\mathbf{K}\boldsymbol{u} = \boldsymbol{b}$

Lagrange Multipliers

A “recipe” exists for imposing the constraint while respecting the linear problem of interest: it consists in writing the following problem

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{H}^t \\ \boldsymbol{H} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{e} \end{pmatrix}$$

we’ll assume $e = 1$in doing the math

Doing the maths

K with constraints

2	-1	0	-1
-1	2	-1	0
0	-1	1	1
-1	0	1	0

RHS

5
0
0
1

inverse

1	1	1	0
1	1,5	1	0,5
1	1	1	1
0	0,5	1	-0,5

SOLUTION

5	u1
5,5	u2
6	u3
-0,5	lambda

Is there a systematic way to do it?

YES!!

Let's assume that:

- $\boldsymbol{f}(\boldsymbol{u})$ is a *functional* describing our problem
- $\mathbf{g}(\boldsymbol{u})$ is the the *functional* describing our constraint.

Both could be non-linear!

“Lagrangian”

We can define a functional of the type

$$\Psi(\boldsymbol{u}, \boldsymbol{\lambda}) := f(\boldsymbol{u}) + \boldsymbol{\lambda}\boldsymbol{g}(\boldsymbol{u})$$

The imposition of the constraint on $\boldsymbol{f}$ **can be obtained by minimizing the functional**

$$\Psi(\boldsymbol{u}, \boldsymbol{\lambda})$$

How?

Defining the problem

1 - DEFINE AN RHS

$$RHS(\boldsymbol{u},\boldsymbol{\lambda}) := \begin{pmatrix} \dfrac{\partial\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{u}} \\ \dfrac{\partial\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{\lambda}} \end{pmatrix}$$

2 – MINIMIZE THE RHS (for example by NR)

$$LHS(\boldsymbol{u},\boldsymbol{\lambda}) := -\frac{\partial RHS(\boldsymbol{u},\boldsymbol{\lambda})}{\partial(\boldsymbol{u},\lambda)} = -\begin{pmatrix} \dfrac{\partial^2\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{u}\partial\boldsymbol{u}} & \dfrac{\partial^2\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{u}\partial\boldsymbol{\lambda}} \\ \dfrac{\partial^2\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\lambda\partial\boldsymbol{u}} & \dfrac{\partial^2\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{\lambda}\partial\boldsymbol{\lambda}} \end{pmatrix}$$

And in application to our model problem??

Our model problema was $\mathbf{K}\boldsymbol{u} = \boldsymbol{b}$ *subjected to the constraint* $\boldsymbol{H}\boldsymbol{u} = \boldsymbol{e}$

In order to apply the abstract technique we just described, we must define the functionals $f(\boldsymbol{u})$ and $\boldsymbol{g}(\boldsymbol{u})$

Interestingly we don't need their exact expression!! We can FORMALLY take

- $f(\boldsymbol{u})$ **such that** $\frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}}=\mathbf{b} - \mathbf{K}\boldsymbol{u}$
- $g(\boldsymbol{u}) = \mathbf{e} - \boldsymbol{H}\boldsymbol{u}$

On the model problem

$$RHS(\boldsymbol{u},\boldsymbol{\lambda}) := \begin{pmatrix} \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{H}^t\boldsymbol{\lambda} \\ \boldsymbol{e} - \boldsymbol{H}\boldsymbol{u} \end{pmatrix}$$

$$LHS(\boldsymbol{u},\boldsymbol{\lambda}) := \begin{pmatrix} \boldsymbol{K} & \boldsymbol{H}^t \\ \boldsymbol{H} & \boldsymbol{0} \end{pmatrix}$$

Which gives the iteration

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{H}^t \\ \boldsymbol{H} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{du} \\ \boldsymbol{d\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{H}^t\boldsymbol{\lambda} \\ \boldsymbol{e} - \boldsymbol{H}\boldsymbol{u} \end{pmatrix}$$

Which completed by

$$\boldsymbol{u} = \boldsymbol{u} + \boldsymbol{du}$$

Is completely equivalent to the original problem (just take the initial guess as 0 for a proof)

Advantages & disadvantages of Lagrange Multipliers method

- Very General & accurate
- Constraints are exactly enforced

It would be "the method of choice", however:

- **Increases the number of unknowns**
- **Problem is not any longer SPD, even if the original unconstrained problem is →typically plays badly with Iterative Solvers**
- **Zeros appear on the main diagonal**
- **User must take care not to repeat constraints. If the same condition is asked twice ➔ linearly dependent constraints → FAILURE**
- ***STABILIZATION* MAY BE NEEDED TO MAKE THE PROBLEM SOLVABLE (out of scope for now)**

“penalty-based” alternatives

Penalty method – defines a new functional as

$$\Psi(\boldsymbol{u}) \coloneqq f(\boldsymbol{u}) - \frac{\boldsymbol{k}}{2}(\boldsymbol{g}(\boldsymbol{u}) \cdot \boldsymbol{g}(\boldsymbol{u}))$$

Where “**k**” is a large number, called **penalty parameter**

Penalty method

For our model problem:

$$\Psi(\boldsymbol{u}) \coloneqq f(\boldsymbol{u}) - \frac{\boldsymbol{k}}{\mathbf{2}}\big((\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u}) \cdot (\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u})\ \big)$$

Hence, as $RHS(\boldsymbol{u}) \coloneqq \frac{\partial\Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial\boldsymbol{u}}$

We get:

$$\mathrm{RHS}(\boldsymbol{u}) \coloneqq \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{k}\big((\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u}) \cdot (-\boldsymbol{H})\ \big)$$

And after minor manipulation

$$\underline{\mathrm{RHS}(\boldsymbol{u}) \coloneqq \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} + \boldsymbol{k}\boldsymbol{H}^{\boldsymbol{t}}(\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u})}$$

Penalty method

Applying NR method:

$$LHS(\boldsymbol{u}) := -\frac{\partial RHS(\boldsymbol{u})}{\partial(\boldsymbol{u})}$$

We get

$$LHS(\boldsymbol{u}) = \boldsymbol{K} + \boldsymbol{k}\boldsymbol{H}^{\boldsymbol{t}}\boldsymbol{H}$$

Penalty Method

In application to our problem we get (assuming a zero initial guess)

penalty **1000**

original matrix

2	-1	0
-1	2	-1
0	-1	1

penalization kH^tH

1000	0	-1000
0	0	0
-1000	0	1000

RHS

5
0
0

RHS with penalty

-995
0
1000

modified matrix

1002	-1	-1000
-1	2	-1
-1000	-1	1001

inverse of modified matrix

1	1	1
1	1,50024988	1,00049975
1	1,00049975	1,0009995

sol

5
5,49975012
5,99950025

Penalty method

- Easy to implement and pretty general
- Equivalent constraints can be enforced multiple times without any problem.
- No additional unknowns

HOWEVER (on the bad side):

- Large “k” needed to enforce the constraints. (constraints “compete” with the stiffness matrix K) → **LEADS TO ILL CONDITIONATED MATRICES**
- Inaccurate (constraints **can** be violated)

Penalty in mixed form

It is interesting (we'll see soon why) to observe that the penalty method can be recast in a form similar to the lagrange multipliers.

To do so we can start with the expression

of the residual which corresponds to the penalty method

$$\mathrm{RHS}(\boldsymbol{u}) \coloneqq \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} + \boldsymbol{k}\boldsymbol{H}^{\boldsymbol{t}}(\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u})$$

And introduce a new var $\boldsymbol{\lambda_k}$≔-$\boldsymbol{k}(\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u})$

Penalty in mixed form

This leads to the definition of a “mixed” RHS

$$\mathrm{RHS}(\boldsymbol{u}, \boldsymbol{\lambda_k}) := \begin{pmatrix} \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{H}^t \boldsymbol{\lambda_k} \\ (\boldsymbol{e} - \boldsymbol{H}\boldsymbol{u}) + \frac{1}{k} \boldsymbol{\lambda_k} \end{pmatrix}$$

To which corresponds the NR iteration

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{H}^t \\ \boldsymbol{H} & -\frac{1}{k} \end{pmatrix} \begin{pmatrix} \boldsymbol{du} \\ \boldsymbol{d\lambda_k} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{H}^t \boldsymbol{\lambda_k} \\ \boldsymbol{e} - \boldsymbol{H}\boldsymbol{u} + \frac{1}{k} \boldsymbol{\lambda_k} \end{pmatrix}$$

Why the mixed form?

When written in mixed form, **the system conditioning is less sensitive to the value of the penalty parameter “k”**

It may make sense to use it since it allows using a larger value of k

Augmented lagrangian method

It is a combination of “Lagrange” & “Penalty” methods

$$\Psi(\boldsymbol{u}) := f(\boldsymbol{u}) - \frac{\boldsymbol{k}}{\boldsymbol{2}}(\boldsymbol{g}(\boldsymbol{u}) \cdot \boldsymbol{g}(\boldsymbol{u})\ ) + \boldsymbol{\lambda}_{\boldsymbol{k}_{\boldsymbol{old}}} \cdot \boldsymbol{g}(\boldsymbol{u})$$

Where $\boldsymbol{\lambda}_{\boldsymbol{k}_{\boldsymbol{old}}}$ is modified at every iteration as

$$\boldsymbol{\lambda}_{\boldsymbol{k}_{\boldsymbol{old}}} = \boldsymbol{\lambda}_{\boldsymbol{k}_{\boldsymbol{old}}} - \boldsymbol{k}\boldsymbol{g}(\boldsymbol{u}_{\boldsymbol{old}})$$

Augmented lagrangian method

For our model problem:

$$\Psi(\boldsymbol{u}) \coloneqq f(\boldsymbol{u}) - \frac{\boldsymbol{k}}{\boldsymbol{2}}\big((\boldsymbol{e} - \boldsymbol{Hu}) \cdot (\boldsymbol{e} - \boldsymbol{Hu})\big) + \boldsymbol{\lambda}_{old} \cdot (\boldsymbol{e} - \boldsymbol{Hu})$$

using $RHS(\boldsymbol{u}) \coloneqq \frac{\partial \Psi(\boldsymbol{u},\boldsymbol{\lambda})}{\partial \boldsymbol{u}}$ we get:

$$\mathrm{RHS}(\boldsymbol{u}) \coloneqq \boldsymbol{b} - \boldsymbol{Ku} - \boldsymbol{k}\big((\boldsymbol{e} - \boldsymbol{Hu}) \cdot (-\boldsymbol{H})\big) - \boldsymbol{H}^{t}\boldsymbol{\lambda}_{old}$$

And after minor manipulation

Note that this is the value to be used in the next iteration!!

$\boldsymbol{\lambda}_{k} = \boldsymbol{\lambda}_{old} - \boldsymbol{k}(\boldsymbol{e} - \boldsymbol{Hu})$

$$\mathrm{RHS}(\boldsymbol{u}) \coloneqq \boldsymbol{b} - \boldsymbol{Ku} - \boldsymbol{H}^{t}\big(\boldsymbol{\lambda}_{old} - \boldsymbol{k}(\boldsymbol{e} - \boldsymbol{Hu})\big)$$

Augmented lagrangian in mixed form

Introducing an auxiliary variable

$$\boldsymbol{\lambda_k} = \lambda_{old} - \boldsymbol{k}(\boldsymbol{e} - \boldsymbol{Hu})$$

we arrive to the iterative scheme:

- STEP1 solve for until convergence

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{H}^t \\ \boldsymbol{H} & -\frac{1}{k} \end{pmatrix} \begin{pmatrix} \boldsymbol{du} \\ \boldsymbol{d\lambda_k} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - \boldsymbol{Ku} - \boldsymbol{H}^t(\boldsymbol{\lambda_k}) \\ \boldsymbol{e} - \boldsymbol{Hu} - \frac{1}{k}(\boldsymbol{\lambda_k} - \boldsymbol{\lambda_{old}}) \end{pmatrix}$$

- STEP2

$$\boldsymbol{\lambda_{old}} = \boldsymbol{\lambda_k} = \boldsymbol{\lambda_{old}} - \boldsymbol{k}(\boldsymbol{e} - \boldsymbol{Hu})$$

- STEP3 repeat 1 and 2 until convergence in $\boldsymbol{\lambda_{old}}$

Augmented lagrangian

- Easy to implement and pretty general
- Equivalent constraints can be enforced multiple times without any problem.
- Large penalty **not needed** (at convergence)

HOWEVER (on the bad side):

- A linear problem is transformed into an iterative process
- If the iterative process is stopped before convergence constraint is not respected exactly

Master-Slave elimination

An alternative approach is the so-called master-slave elimination.

We recall that our model problem was

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix}$$

Subjected to the additional constraint:

$$u_3 = u_1 + 1$$

Master-slave elimination

The idea is to build the constraint within the unknowns. In the specific case, u_3 can be expressed in terms of u_1 and u_2 as follows

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Or in short

$$\boldsymbol{u} = \boldsymbol{T}\boldsymbol{u}^* + \boldsymbol{q}$$

Master-slave elimination

We can now substitute $\boldsymbol{u} = \boldsymbol{T}\boldsymbol{u}^{*} + \boldsymbol{q}$

In the original problem $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{b}$ to get

$$\boldsymbol{K}(\boldsymbol{T}\boldsymbol{u}^{*} + \boldsymbol{q}) = \boldsymbol{b}$$

To make the problem solvable we then left multiply it by $\boldsymbol{T}^{\boldsymbol{t}}$

$$\boldsymbol{T}^{\boldsymbol{t}}\boldsymbol{K}(\boldsymbol{T}\boldsymbol{u}^{*} + \boldsymbol{q}) = \boldsymbol{T}^{\boldsymbol{t}}\boldsymbol{b}$$

Or

$$\boldsymbol{T}^{\boldsymbol{t}}\boldsymbol{K}\boldsymbol{T}\boldsymbol{u}^{*} = \boldsymbol{T}^{\boldsymbol{t}}\boldsymbol{b} - \boldsymbol{T}^{\boldsymbol{t}}\boldsymbol{K}\boldsymbol{q}$$

Master-slave elimination

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_1^* \\ u_2^* \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Simplifying this gives

$$\begin{pmatrix} 3 & -2 \\ -2 & 2 \end{pmatrix} \begin{pmatrix} u_1^* \\ u_2^* \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} u_1^* \\ u_2^* \end{pmatrix} = \begin{pmatrix} 5 \\ 5.5 \end{pmatrix}$$

Once we recover the value of u_3 we can immediately verify that the solution is the same as before

Master-slave elimination

- Smaller, well conditioned systems
- Preserves symmetry and matrix properties

HOWEVER (on the bad side):

- Not very general (how to do for a generic nonlinear constraint??)
- Can be (very) difficult to implement
- User must be extremely careful in applying constraints exactly once
- User must identify variables to be eliminated

bibliography

http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/IFEM.Ch08.d/IFEM.Ch08.pdf

http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/IFEM.Ch09.d/IFEM.Ch09.pdf