19th IEEE World Symposium on Applied Machine Intelligence and Informatics (SAMI 2021),

Herl'any, Slovakia, January 21-23, 2021

Speeding up the Reduced Gradient Method for Constrained Optimization.

Hazem Issa[†] **József K. Tar**[*]

[†]***Doctoral School of Applied Informatics and Applied Mathematics,***

[*]***University Research, Innovation and Service Center (EKIK), Antal Bejczy Center for Intelligent Robotics,***

Óbuda University, Budapest, Hungary

Motivations and Aims

- In various technical applications the local minimum of a differentiable cost function must be found under constraints that are interpreted as embedded hypersurfaces in the whole space of search.
- Generally Lagrange's "Reduced Gradient Method" can be applied for solving such problems
- The "Receding Horizon Control (RHC)" is a heuristic realization of the "Model Predictive Controllers (MPC)".
- The necessary trajectory of the differential inverse kinematic task of redundant robot arms has to be tracked with minimized motion of the joint coordinates.

Motivations (Cont.)

- The definition of the differentiable $\mathrm{f(x)} : \mathbb{R}^{\mathrm{n}} \mapsto \mathbb{R}$ functions (1) can be used according to which if $\|\Delta\mathrm{x}\|$ is small enough the difference in the function value can be well approximated by the *scalar product* of the gradient $\partial\mathrm{f}/\partial\mathrm{x}$ and the "displacement" $\Delta\mathrm{x}$

$$\mathrm{f}(\mathrm{x}+\Delta\mathrm{x}) = \mathrm{f}(\mathrm{x}) + \sum_{\ell} \frac{\partial\mathrm{f}}{\partial\mathrm{x}_\ell}\Delta\mathrm{x}_\ell + \mathscr{O}\left(\|\Delta\mathrm{x}\|^2\right) \tag{1}$$

Motivations (Cont.)

- If the local minimum of the function is needed, instead of the actual point x a neighboring point $x-\alpha\left(\frac{\partial f}{\partial x}\right)^T \Delta x$ must be proved in which $0<\alpha$ is a small positive number.
- The function's value can be decreased until the gradient becomes zero.
- If it is assumed that the local minimum is 0, the process can be made faster by the "Newton-Raphson Algorithm" in which α is so chosen that $-\alpha\|\partial f/\partial x\|^2 = f(x)$
- If the optimization must be done under constraints expressed in the form $\{0=g_i(x) : \mathbb{R}^n \mapsto \mathbb{R}\}$ $i=1,\ldots,K<n$, that –apart from particular cases– determine an $n-K$ dimensional hypersurface in $\mathbb{R}^n$.

Motivations (Cont.)

- The optimization can be started by finding a point on this hypersurface by applying the Newton-Raphson Algorithm applied to e.g. $G(x) := \sum_\ell g_\ell^2(x)$, since $G(x) = 0$ if and only if each $g_\ell(x) = 0$.

$$x_{s+1} = x_s - a_1 \frac{G(x_s)}{\left(\frac{\partial G}{\partial x}\right)^T\Big|_{x_s} \frac{\partial G}{\partial x}\Big|_{x_s}} \frac{\partial G}{\partial x}\Big|_{x_s} \qquad (2)$$

in which $0 < a_1 < 1$.

Motivations (Cont.)

- Following that, for decreasing the cost function a small step can be done as $a_2 \left(-\partial f(x)/\partial x + \sum_\ell \lambda_\ell \partial g_\ell/\partial x\right)$
  in which $0 < a_2 < 1$ and the Lagrange multipliers $\{\lambda_\ell\}$ must be so chosen that this displacement must be orthogonal to each constraint gradient $\partial g_i(x)/\partial x$,
- By using the 0 scalar product for expressing orthogonality a linear set of equations is obtained and has to be solved as

$$\forall k: \; -\left(\frac{\partial g_k}{\partial x}\right)^T \frac{\partial f}{\partial x} + \sum_\ell \left(\frac{\partial g_k}{\partial x}\right)^T \frac{\partial g_\ell}{\partial x}\lambda_\ell = 0 \; . \qquad (3)$$

- This process can be repeated until the reduced gradient achieves zero. In this point a local optimum allowed by the constraints has been found.

Replacing Numerical Algorithm with Auxiliary Function

- For avoiding the considerable computational needs of previous numerical algorithm the "Auxiliary Function" of the problem is introduced as

$$\Phi(x,\lambda) := -f(x) + \sum_{\ell} \lambda_\ell g_\ell(x) \tag{4}$$

- If the point x is not located at the constraint surface, at least there exist a constraint function for which $g_i(x) \neq 0$, therefore with the appropriate Lagrange multiplier λ_i $\Phi \in [-\infty, +\infty]$ can be arbitrarily set.
- However, the *set of local extrema* of this function, simultaneously must contain the point at which the numerical procedure stops:

$$\partial\Phi/\partial x = 0 \ \ , \partial\Phi/\partial\lambda = 0$$

Replacing Numerical Algorithm with Auxiliary Function (Cont.)

- In special cases these local extrema can be found without numerical procedure.
- For instance in the differential inverse kinematic task of redundant robots the equation

$$\dot{x}_k(t) = \sum_{\ell} J_{k\ell}\dot{\xi}_{\ell}(t) \tag{5}$$

- with a given $\dot{x}$ and a given Jacobian J must be solved with "minimal motion of the joints of the robot" that means that the sum of the squares of the joint coordinate velocities $\sum_i \dot{\xi}_i^2$ must be minimized under the constraint expressed by (5) leading to the auxiliary function :

$$\Phi(\dot{\xi},\lambda) = -\sum_i \dot{\xi}_i^2 + \sum_k \lambda_k \left(\dot{x}_k - \sum_{\ell} J_{k\ell}\dot{\xi}_{\ell}\right) \tag{6}$$

Replacing Numerical Algorithm with Auxiliary Function (Cont.)

- To determine if the found local extremum is a *local minimum*, *local maximum*, or some *saddle point* of the auxiliary function, the Jacobian of the $\nabla\Phi$ function of (6) must be considered in the point $\nabla\Phi(\mathrm{x}) = 0$.
- In this case *the Jacobian of the auxiliary function* is constant

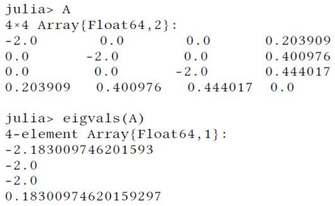
$$\mathrm{J}^{\mathrm{Aux}} = \left[\begin{array}{c|c} -2\mathrm{I} & -\mathrm{J}^{\mathrm{T}} \\ \hline -\mathrm{J} & 0 \end{array}\right] . \tag{7}$$

- For estimating the potential eigenvalues of $\mathrm{J}^{\mathrm{Aux}}$

$$\mu \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix} = \begin{bmatrix} \mu\mathrm{u} \\ \mu\mathrm{v} \end{bmatrix} = \mathrm{J}^{\mathrm{Aux}} \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix} = \begin{bmatrix} -2\mathrm{u} - \mathrm{J}^{\mathrm{T}}\mathrm{v} \\ -\mathrm{J}\mathrm{u} \end{bmatrix} \tag{8}$$

Replacing Numerical Algorithm with Auxiliary Function (Cont.)

- leading to $\mu = \frac{-2 \pm \sqrt{4 + 4 \mathrm{u}^{\mathrm{T}} \mathrm{J}\mathrm{J}^{\mathrm{T}} \mathrm{u} / \mathrm{u}^{\mathrm{T}} \mathrm{u}}}{2}$.
  which indicates that it cannot be taken for granted that an eigenvalue of $\mathrm{J}^{\mathrm{Aux}}$ is necessarily negative since $0 \leq \mathrm{u}^{\mathrm{T}} \mathrm{J}\mathrm{J}^{\mathrm{T}} \mathrm{u} / \mathrm{u}^{\mathrm{T}} \mathrm{u}$.
- In the following example



```
julia> A
4×4 Array{Float64,2}:
-2.0        0.0        0.0        0.203909
0.0        -2.0        0.0        0.400976
0.0         0.0       -2.0        0.444017
0.203909   0.400976   0.444017   0.0

julia> eigvals(A)
4-element Array{Float64,1}:
-2.183009746201593
-2.0
-2.0
0.18300974620159297
```

one of the eigenvalues is significantly positive, i.e. the local optimum in this case belongs to a saddle point.

Complexity Reduction by using the Gram-Schmidt Algorithm

- The basic idea of complexity reduction is that *instead investing energy into computing the original Lagrange multipliers* the appropriate gradients can be put into the columns of a matrix as

$$\left[\begin{array}{c|c|c|c} \frac{\partial g_1}{\partial x} & \dots & \frac{\partial g_K}{\partial x} & -\frac{\partial f}{\partial x} \end{array}\right] \tag{9}$$

  then the Gram-Schmidt algorithm can be applied to this matrix.

- $\mathrm{Gram-Schmidt\ Algorithm:}$
  Step 1: 2nd, ..., $K+1$th columns of the matrix in (9) must be made orthogonal to the first column.
  Step 2: the 3rd,..., $K+1$th columns of the so obtained matrix must be orthogonal to its 2nd column, and so on.
  Finally in the last column a vector appears that is orthogonal to each previous column of the final matrix.

Complexity Reduction by using the Gram-Schmidt... (Cont.)

- Further simplification can be done if we take it into account that $\forall i\, g_i(x) = 0$ if and only if $G(x) = 0$. Consequently it is enough to make a single step of orthogonalization for the two column matrix $\left[\frac{\partial G}{\partial x} \middle| \frac{-\partial f}{\partial x}\right]$.
- Finally, instead of the original form of $G(x)$ its modified version $G(x) := \sum_\ell |g_\ell(x)|^\beta$ can be considered for $1 < \beta$ based on the observation that for the function $|x|$ the Newton–Raphson algorithm would be very fast, however, around the $x = 0$ it should have instabilities.
- This instability would be evaded for a β that is a little bit greater than 1.

Numerical Example

- Consider the following problem:

$$\begin{aligned} \text{minimize} \quad & f(x) = A_1(x_{n1} - x_1)^2 + A_2(x_{n2} - x_2)^4 + A_3(x_{n3} - x_3)^6 \\ \text{s.t.} \quad & g_1(x) = 0, \quad g_2(x) = 0 \\ \text{where} \quad & g_1(x) = x_1^2 + x_2^2 + x_3^2 - R^2, \\ & g_2(x) = (x_1 - 0.5R)^2 + x_2^2 + x_3^2 - R^2 \\ & x_{1n} = 0.2R \ , \quad x_{2n} = 0.3R \ , \quad x_{3n} = 0.4R \end{aligned}$$

- The optimization starts from the known absolute minimum of the cost function.
- From this point at first the Newton–Raphson algorithm moved the starting point to the constraint surface then the reduced gradient algorithm continued moving over the surface.

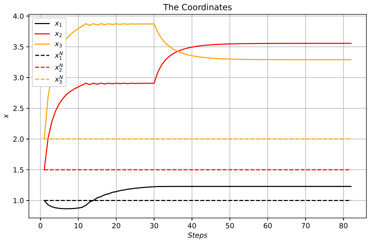
Numerical Example (Cont.)

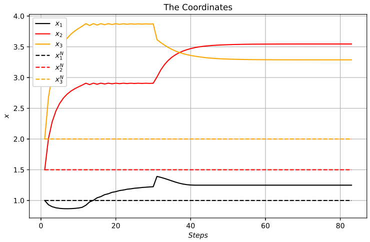
- The parameters of the computations:

Parameter	**Value**
Constraint radius R	5.0
Step length for gradient estimation δx	10^{-3}
Error limit ε	10^{-2}
Newton–Raphson speed parameter a_1	0.5
Reduced Gradient speed parameter a_2	10^{-3}
Cost function parameter A_1	1.0
Cost function parameter A_2	2.0
Cost function parameter A_3	3.0
Cost function parameter β	1.1

Simulation Results

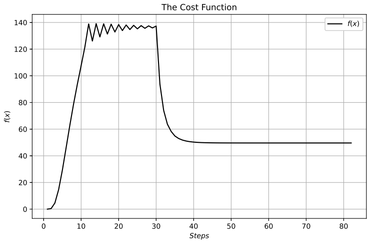
- Variation of the x coordinates for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $\mathrm{G} = \mathrm{g}_1(\mathrm{x})^2 + \mathrm{g}_2(\mathrm{x})^2$

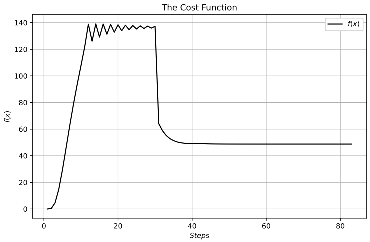




Simulation Results (Cont.)

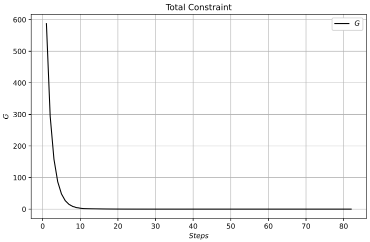
- Variation of the cost function for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $\mathrm{G} = \mathrm{g}_1(\mathrm{x})^2 + \mathrm{g}_2(\mathrm{x})^2$

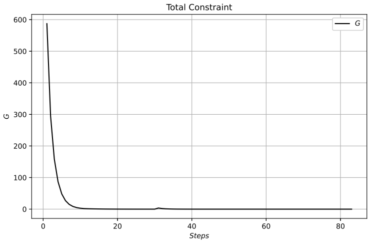




Simulation Results (Cont.)

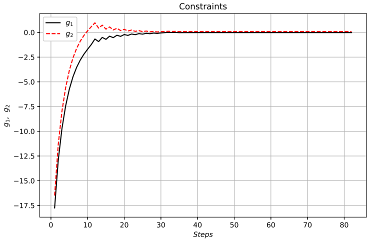
- Variation of the constraint value G for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $\mathrm{G} = \mathrm{g}_1(\mathrm{x})^2 + \mathrm{g}_2(\mathrm{x})^2$

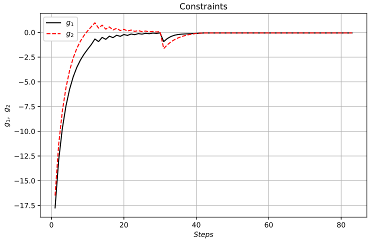




Simulation Results (Cont.)

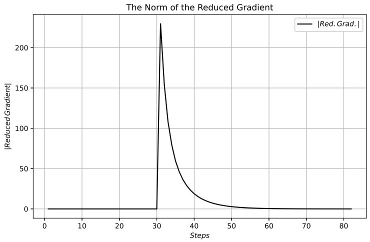
- Variation of the $g_1(x)$ and $g_2(x)$ constraint values for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $G = g_1(x)^2 + g_2(x)^2$

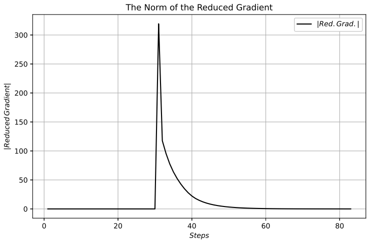




Simulation Results (Cont.)

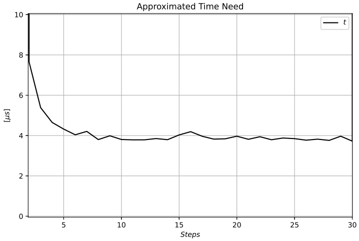
- Variation of the norm of the reduced gradient (it is relevant only in the second phase of the algorithm) for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $G = g_1(x)^2 + g_2(x)^2$

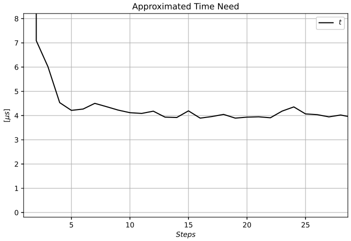




Simulation Results (Cont.)

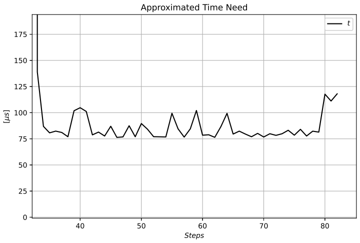
- The time-need *in the Newton-Raphson phase* for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $G = g_1(x)^2 + g_2(x)^2$

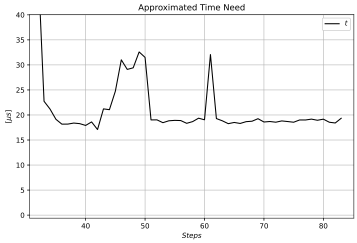




Simulation Results (Cont.)

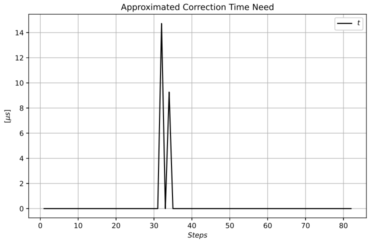
- The time-need of the reduced gradient algorithm *in the reduced gradient phase* for the "double steps" (LHS) and the "single step" (RHS) gradient reduction for $G = g_1(x)^2 + g_2(x)^2$

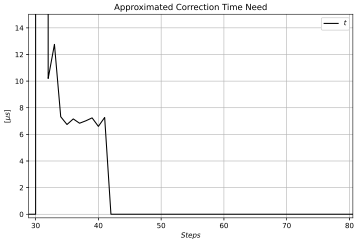




Simulation Results (Cont.)

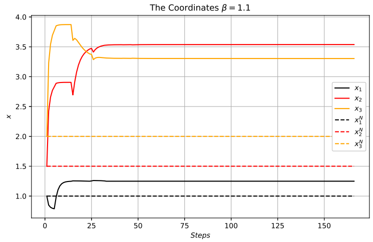
- The time need of the corrections needed for pulling back the actual point onto the constraint surface *in the reduced gradient phase* of the algorithm ["double steps" (LHS) and the "single step" (RHS) gradient reduction for $\mathrm{G} = \mathrm{g}_1(\mathrm{x})^2 + \mathrm{g}_2(\mathrm{x})^2$]

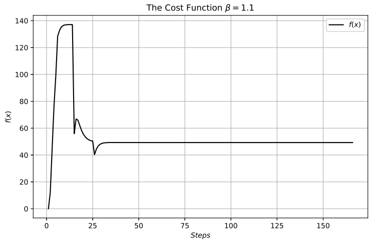




Simulation Results (Cont.)

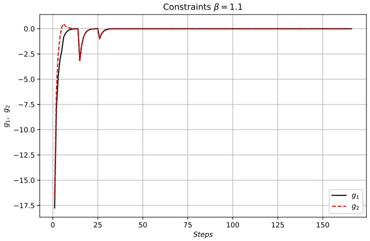
- Variation of the x coordinates and the value of the cost function for a single step reduction for $\mathrm{G} = |\mathrm{g}_1(\mathrm{x})|^\beta + |\mathrm{g}_2(\mathrm{x})|^\beta$

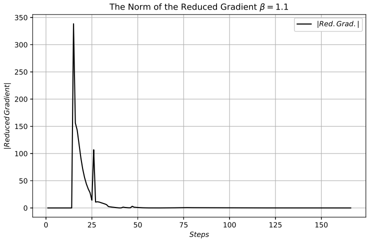




Simulation Results (Cont.)

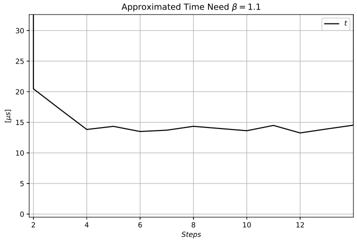
- Variation of the individual constraint functions and the norm of the reduced gradient function for a single step reduction for $G = |g_1(x)|^{\beta} + |g_2(x)|^{\beta}$

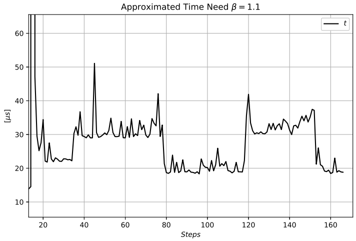




Simulation Results (Cont.)

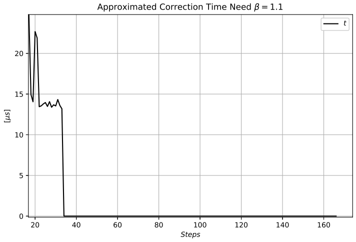
- The time need of the *Newton-Raphson phase (LHS)* and *the reduced gradient phase (RHS)* for a single step reduction for $\mathrm{G} = |\mathrm{g}_1(\mathrm{x})|^\beta + |\mathrm{g}_2(\mathrm{x})|^\beta$





Simulation Results (Cont.)

- The time need of the corrections needed for pulling back the actual point onto the constraint surface in the reduced gradient phase of the algorithm for a single step reduction for $\mathrm{G} = |\mathrm{g}_1(\mathrm{x})|^\beta + |\mathrm{g}_2(\mathrm{x})|^\beta$



Conclusions

- If the Lagrange multipliers themselves do not play important role, their calculation can be avoided by the use of the simple Gram–Schmidt Algorithm.
- The use of a single constraint term made of the sum of the squares of the individual ones can achieve considerable reduction in the time of the necessary computations.
- Investigations that used the MS EXCL – Solver – VBA environment can be simplified to a very fast and simple Julia language-based framework.

Acknowledgment

The research was supported by the Doctoral School of Applied Informatics and Applied Mathematics at Óbuda University.

Thank you for your attention!!