The Iterated Ritz Method: Basis, implementation and further development

Dvornik, Josip; Lazarević, Damir; Uroš, Mario; Šavor Novak, Marta

Coupled Systems Mechanics, *Vol. 7, No. 6 (2018) 755-774*
DOI: https://doi.org/10.12989/csm.2018.7.6.755

The Iterated Ritz Method: Basis, implementation and further development

Josip Dvornik[a], Damir Lazarevic[*], Mario Uros[b] and Marta Savor Novak[b]

Department for Engineering Mechanics, Faculty of Civil Engineering, University of Zagreb, Kaciceva 26, 10 000 Zagreb, Croatia

(Received Septebmer 27, 2018, Revised November 7, 2018, Accepted November 8, 2018)

Abstract. The Ritz method is known as very successful strategy for discretizing continuous problems, but it has never been used for solving systems of algebraic equations. The Iterated Ritz Method (IRM) is a novel iterative solver based on the discretized Ritz procedure applied at each iteration step. With an appropriate choice of coordinate vectors, the method may be efficient in linear, nonlinear and optimization problems. Additionally, some iterative methods can be explained as special cases of this approach, which helps to understand advantages and limitations of these methods and gives motivation for their improvement in sense of IRM. In this paper, some ideas for generation of efficient coordinate vectors are presented. The algorithm was developed and tested independently and then implemented into the open source program FEAP. Method has been successfully applied to displacement based (even ill-conditioned) models of structural engineering practice. With this original approach, a new iterative solution strategy has been opened.

Keywords: iterative methods; conjugate gradients; successive over-relaxation; preconditioning; Iterated Ritz Method

1. Introduction

Iterative solution methods are good alternative to direct solution strategies, especially for the extremely large, sparse, even not directly accessible system matrix. Diagonal dominance is preferable. It is possible to check intermediate results (at each step if needed) and use weaker stopping criterion to quickly obtain rough solution estimate. This is important for the preliminary product design phase, when many variants are analyzed until good final decision is made. After that, stronger criterion is applied, but solver may be started from the preliminary phase solution, not from the usually bad initial guess (commonly zero vector). Generally, available data about model or solution may be utilized. For example, if problem is slightly asymmetric, it is preferable to start from the symmetric solution, if possible. Furthermore, different compact matrix storages are easily adopted, because initial sparsity is retained during solution process. Finally, iterative

∗Corresponding author, Full Professor, E-mail: damir@grad.hr
[a]Professor Emeritus, E-mail: dvornik@grad.hr
[b]Assistant Professor, E-mail: uros@grad.hr & msavor@grad.hr

http://www.techno-press.org/?journal=csm&subpage=8 ISSN: 2234-2184 (Print), 2234-2192 (Online)

methods are successfully applied in the nonlinear (optimization) fields, because such problems should be solved iteratively by definition. The main drawback is direct manipulation with the right-hand side vector, so every loading condition should be separately solved. Additionally, number of steps is not known in advance and user should be able to change some input parameters if process has trouble converging. Generally, overall reliability of iterative methods is not guaranteed.

A linear equations system

$$\mathbf{K}\,\mathbf{u} = \mathbf{f} \tag{1}$$

is given. Matrix **K** is symmetric and positive definite (SPD) of order *n*. If Eq. (1) represents a linearized displacement-based model of elastic body, **K** is the stiffness matrix, **u** and **f** are the displacement and the load vector, respectively. The solution of Eq. (1) is equivalent to minimization of the total potential energy (quadratic form)

$$G(\mathbf{u}) = \frac{1}{2}\,\mathbf{u}^{\mathrm{T}}\mathbf{K}\,\mathbf{u} - \mathbf{u}^{\mathrm{T}}\mathbf{f} \tag{2}$$

Essentially, G is the discrete approximation of the Lagrange energy functional on the continuous model. The function $G = G_0$, where G_0 is a constant, is the contour hypersurface of second degree in n-dimensional Euclidean space defined by **u**. For the SPD matrix this surface must be a hyper-ellipsoid. The point in the space where G reaches minimum lies in the center of ellipsoids. This is the solution point of Eq. (1). It can be considered as a degenerate ellipsoid, where all main axes vanish. Numerically, depending on the criterion adopted, it is however a very small ellipsoid with the energy level just a little higher than the minimum one.

2. Approximate solution

An approximate solution of Eq. (1) is based on an expansion of **u** by discretized Ritz method

$$\mathbf{u} = \sum_{i=1}^{m} a_i \boldsymbol{\varphi}_i \tag{3}$$

In the standard approach, $\boldsymbol{\varphi}_i$ and a_i are known as (linearly independent) Ritz vectors and coefficients, respectively. The number of vectors m must satisfy the inequality: $1 \leq m \leq n$. Formula (3) can be expressed as (Ibrahimbegovic and Wilson 1990a, Ibrahimbegovic *et al*. 1990)

$$\mathbf{u} = \boldsymbol{\Phi}\mathbf{a} \tag{4}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1 \quad \boldsymbol{\varphi}_2 \quad \cdots \quad \boldsymbol{\varphi}_m]$ and $\mathbf{a} = [a_1 \quad a_2 \quad \cdots \quad a_m\,]^{\mathrm{T}}$. After substituting Eq. (4) in Eq. (2), G is expressed as function of **a** only

$$G(\mathbf{u}) \approx G(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{K}\boldsymbol{\Phi}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{f} \tag{5}$$

The products $\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{K}\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{f}$ are new (SPD) generalized (Ritz) stiffness matrix $\overline{\mathbf{K}}$ and right-hand side vector $\bar{\mathbf{f}}$, both of order m. Now, the energy can be shortly written as

$$G(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\,\mathbf{a} - \mathbf{a}^{\mathrm{T}}\bar{\mathbf{f}} \tag{6}$$

After minimization the system of m linear equations

$$\bar{\mathbf{K}}\,\mathbf{a} = \bar{\mathbf{f}} \tag{7}$$

is obtained. Once Eq. (7) is solved, the original unknowns are recovered from Eq. (4). This is only an approximate solution (because Eq. (3) is usually not exact expansion) and iterative improvement is needed. The procedure leads to gradient class solution method that combines iterative and direct solution strategies.

3. Iterative improvement

Iterative process is usually defined by

$$\mathbf{u}_{(i+1)} = \mathbf{u}_{(i)} + \Delta\mathbf{u}_{(i)}, \tag{8}$$

where now the solution increment $\Delta\mathbf{u}_{(i)}$ is approximated (Dvornik 1979, Lazarevic and Dvornik 2017) in the sense of Eq. (4)

$$\Delta\mathbf{u}_{(i)} = \mathbf{\Phi}_{(i)}\mathbf{a}_{(i)} \tag{9}$$

According to Eq. (2) the energy in the current iteration is

$$G(\mathbf{u}_{(i)}) = \frac{1}{2}\,\mathbf{u}_{(i)}^{\mathrm{T}}\mathbf{K}\,\mathbf{u}_{(i)} - \mathbf{u}_{(i)}^{\mathrm{T}}\mathbf{f} \tag{10}$$

and in the next iteration, using Eqs. (8) and (9)

$$G(\mathbf{u}_{(i+1)}) = \frac{1}{2}\left(\mathbf{u}_{(i)}^{\mathrm{T}} + \mathbf{a}_{(i)}^{\mathrm{T}}\mathbf{\Phi}_{(i)}^{\mathrm{T}}\right)\mathbf{K}\left(\mathbf{u}_{(i)} + \mathbf{\Phi}_{(i)}\mathbf{a}_{(i)}\right) - \left(\mathbf{u}_{(i)}^{\mathrm{T}} + \mathbf{a}_{(i)}^{\mathrm{T}}\mathbf{\Phi}_{(i)}^{\mathrm{T}}\right)\mathbf{f} \tag{11}$$

After rearrangement, with common definition of residual as

$$\mathbf{r}_{(i)} = \mathbf{f} - \mathbf{K}\mathbf{u}_{(i)} \tag{12}$$

a simple relation is obtained

$$G(\mathbf{u}_{(i+1)}) = G(\mathbf{u}_{(i)}) + \frac{1}{2}\mathbf{a}_{(i)}^{\mathrm{T}}\bar{\mathbf{K}}_{(i)}\mathbf{a}_{(i)} - \mathbf{a}_{(i)}^{\mathrm{T}}\bar{\mathbf{r}}_{(i)} \tag{13}$$

where $\bar{\mathbf{r}}_{(i)} = \mathbf{\Phi}_{(i)}^{\mathrm{T}}\mathbf{r}_{(i)}$ is the generalized (Ritz) residual vector of order m. Since $G(\mathbf{u}_{(i)})$ does not depend on $\mathbf{a}_{(i)}$, it vanishes during minimization process, i.e., it is sufficient to differentiate the remaining terms, energy decrease $\Delta G(\mathbf{a}_{(i)})$, and obtain

$$\bar{\mathbf{K}}_{(i)}\mathbf{a}_{(i)} = \bar{\mathbf{r}}_{(i)} \tag{14}$$

At each step $\mathbf{\Phi}_{(i)}$, $\bar{\mathbf{K}}_{(i)}$ and $\bar{\mathbf{r}}_{(i)}$ are generated. After Eq. (14) is solved, an approximate solution is updated using Eq. (9) and finally Eq. (8).

3.1.1 System of equations within step

The increment $\Delta\mathbf{u}_{(i)}$, determined by $\mathbf{a}_{(i)}$ in the given subspace, ensures the largest energy reduction within it. That is why Eq. (14) should be repeatedly solved. Usually, this is a very small system, because only several coordinate vectors are used ($m \ll n$). Thus, the solution of *n*-

equation system is sought by solving the *m*-equation system multiple times. In one of our FEM benchmarks n reached $1.9 \cdot 10^8$, while m was only 10 (Fig. 1). Only 168 iterations were needed to obtain the solution.

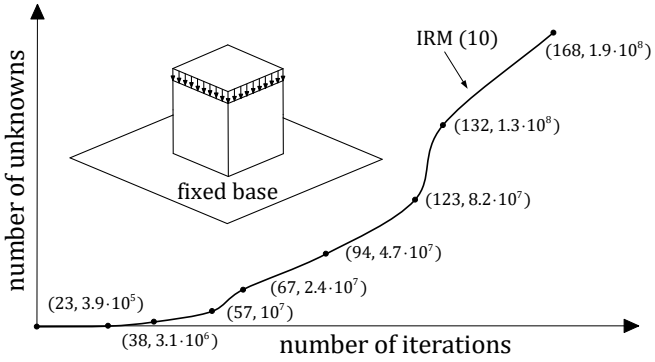

Fig. 1 Behavior of IRM with ten coordinate vectors

Such a small system ($\overline{\mathbf{K}}_{(i)}$ is usually full) can be solved by any direct method. In our case, Cholesky decomposition is used. If the iterative process is convergent, the sum of small-system solutions approaches large (original) system solution and the sum of small-system energies monotonically decreases and approaches the minimum of a large system. As in any iterative algorithm, the process is terminated when the convergence criterion is reached.

3.1.2 Convergence criterion

For the stopping criterion, the residual norm is usually use

$$\left\|\mathbf{r}_{(i)}\right\|_2 < \delta \left\|\mathbf{r}_{(0)}\right\|_2 \tag{15}$$

where δ is a very small number (here 10^{-8}). In practical implementation residual is updated as

$$\mathbf{r}_{(i+1)} = \mathbf{r}_{(i)} - \mathbf{K}\,\Delta\mathbf{u}_{(i)} = \mathbf{r}_{(i)} - \mathbf{K}\boldsymbol{\Phi}_{(i)}\mathbf{a}_{(i)} \tag{16}$$

This relation is faster than Eq. (12) because product $\mathbf{K}\boldsymbol{\Phi}_{(i)}$ is already calculated during definition of $\overline{\mathbf{K}}_{(i)}$, and $\mathbf{u}_{(i)}$ is needed only at the very end, after Eq. (15) is satisfied. But Eq. (16) is also recursive, and the residual should always be corrected after some number of steps ($i_{\max}$) using the equilibrium relation Eq. (12), because of accumulated round-off errors. It should be also applied at the beginning of the process, to find $\mathbf{r}_{(0)}$ if $\mathbf{u}_{(0)} \neq \mathbf{0}$. Finally, to avoid useless calculations (if the algorithm has trouble converging) the maximum number of steps ($n_{\max}$) should also be defined. The convergence is not guaranteed if: the coordinate vectors are (exactly or nearly) linearly dependent (then subspace degenerates), the residual and the subspace are mutually orthogonal, and the stopping criterion is too strict (rounding errors affect the end of iterative process).

3.1.3 Relaxation of displacement

The search for the exact local minimum of *G* in each step is not needed nor necessarily optimal.

Motivated by the method of Successive Over-Relaxation (SOR), the process can often be accelerated by multiplying the step length with the relaxation factor $\omega \in \langle 0,2 \rangle$. Although a formal mathematical proof about interval boundaries is known, a simple physical interpretation is possible (Lazarevic and Dvornik 2017). By definition, G is a quadratic function of $\mathbf{u}$ and a line-search along some direction $\mathbf{p}_{(i)}$, usually $\Delta\mathbf{u}_{(i)}$, can be expressed as a quadratic function of ω (Fig. 2)

$$G(\omega) = G_{\min} + (G_0 - G_{\min})(\omega - 1)^2 \tag{17}$$

Here, G_0 is the value at the tail of the vector $\mathbf{p}_{(i)}$ (beginning of the step) and $G_{\min}$ is the local minimum along search direction. From the condition of monotone convergence $G(\omega) < G_0$, $(\omega - 1)^2 < 1$ is obtained and accordingly $0 < \omega < 2$. Obviously, at the boundaries energy remains constant, i.e. $G(0) = G(2)$, and outside of the boundaries it increases. In both cases the method does not converge.

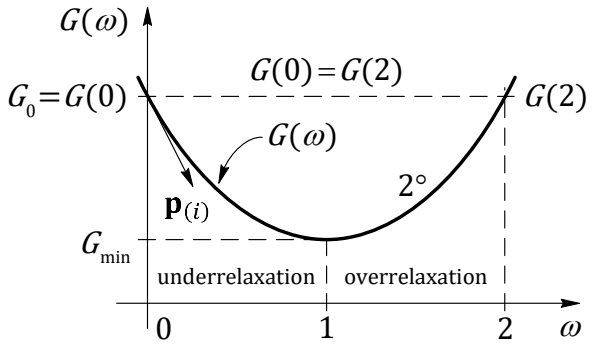

Fig. 2 Energy function

Some optimal ω exists but varies for every unknown and solution step. The determination is usually more "expensive" than the benefit of possible improvement. Generally, ω may be guessed by intuition and experience, and kept constant during the solution process. According to Eq. (8), the accelerated solution is

$$\mathbf{u}_{(i+1)} = \mathbf{u}_{(i)} + \omega\Delta\mathbf{u}_{(i)} \tag{18}$$

3.1.4 Simple pseudocode

This approach and interpretation of such generalized iteration is simple and natural but not fully recognized by researchers in the field of iterative methods for solving a linear (or nonlinear) equation systems. Weak similarity with subspace iteration strategies in structural dynamics exists, but in IRM sequence of subspaces has no property of convergence. Such property is not needed nor exists. Rather, coordinate vectors are very freely selected from step to step to efficiently reduce residual norm. Main elements of IRM are given by the simple pseudocode (Table 1).

The generation of vectors should be fast and small system should be reasonably small. Actually, the efficiency of IRM is subject to some compromise. If a larger number of adequate vectors is used (spanning the subspace with an appropriate solution increment), the energy reduction per step is larger; hence less steps are needed to reach the solution, but each step requires more time. With

less vectors, duration of the step is shorter, but subspace is not so effective, and more steps are required. In the limiting case, if the subspace dimension equals the number of unknowns ($m = n$), the energy is minimized, and the solution is theoretically reached after the first step, but at the "price" equal or larger than needed for the direct solution.

Table 1 Pseudocode of IRM

Algorithm 1 The Iterated Ritz Method	
Require: $\mathbf{K}, \mathbf{f}, \omega, \delta, i_{\max}, n_{\max}$ {stiffness matrix, load vector, relaxation factor, fraction of initial residual, steps to update, total steps}	
Ensure: $\mathbf{u}$ {approximate solution vector}	
1.	$i \leftarrow 0$ {set iteration counter to zero}
2.	$\mathbf{u}_{(0)} \leftarrow \mathbf{0}$ {initial guess: null-vector}
3.	$\mathbf{r}_{(0)} \leftarrow \mathbf{f}$ {residual: load vector}
4.	while $\left[\left(\left\|\mathbf{r}_{(i)}\right\|_2 > \delta\left\|\mathbf{r}_{(0)}\right\|_2\right) \wedge (i < n_{\max})\right]$ do {loop on i}
5.	$i \leftarrow i + 1$ {increment counter}
6.	$\mathbf{\Phi}_{(i)} \leftarrow [\boldsymbol{\varphi}_{(i),1} \cdots \boldsymbol{\varphi}_{(i),m}]$ {generation of coordinate vectors matrix}
7.	$\mathbf{A} \leftarrow \mathbf{K}\mathbf{\Phi}_{(i)}$ {used in lines 8 and 13}
8.	$\bar{\mathbf{K}}_{(i)} \leftarrow \mathbf{\Phi}_{(i)}^{\mathrm{T}}\mathbf{A}$ {generation of Ritz stiffness matrix}
9.	$\bar{\mathbf{r}}_{(i)} \leftarrow \mathbf{\Phi}_{(i)}^{\mathrm{T}}\mathbf{r}_{(i)}$ {right-hand side}
10.	$\mathbf{a}_{(i)} \leftarrow \bar{\mathbf{K}}_{(i)}^{-1}\bar{\mathbf{r}}_{(i)}$ {solution of a small system}
11.	$\mathbf{u}_{(i)} \leftarrow \mathbf{u}_{(i)} + \omega\mathbf{\Phi}_{(i)}\mathbf{a}_{(i)}$ {approximate solution}
12.	if $i \bmod i_{\max} \neq 0$ then {usual step}
13.	$\mathbf{r}_{(i)} \leftarrow \mathbf{r}_{(i)} - \mathbf{A}\mathbf{a}_{(i)}$ {update residual from previous vector}
14.	else {every $i_{\max}$ steps}
15.	$\mathbf{r}_{(i)} \leftarrow \mathbf{f} - \mathbf{K}\mathbf{u}_{(i)}$ {update residual from equilibrium equation}
16.	end if {end update residual}
17.	end while {end loop on i }

4. Special cases of IRM

Depending on the choice of coordinate vectors, some basic iterative methods can be interpreted as special cases of IRM. Compared to traditional strategies (Saad 2003, Olshanskii 2014), such analogies are not used for faster implementation of these methods. Rather, another interpretation and properties useful for generation of coordinate vectors are emphasized. Also, popular acceleration of iterative methods by single or multiple preconditioning (Benzi 2002, Ferronato 2012, Sun and Gu 2016) can be interpreted as additional (one or several) coordinate vectors. Any significant modification of Algorithm 1 is not needed. We believe that such analogy with other (preconditioned) iterative methods exists. For example, Krylov subspace methods may be considered as a special case of IRM, if coordinate vectors are properly defined ($\boldsymbol{\varphi}_1 = \mathbf{r}_{(0)}$, $\boldsymbol{\varphi}_2 = \mathbf{K}\mathbf{r}_{(0)}$, $\boldsymbol{\varphi}_3 = \mathbf{K}^2\mathbf{r}_{(0)}$, *etc.*).

4.1 Cycling strategy: The methods of Gauss-Seidel and successive over-relaxation

Traditionally, the Gauss-Seidel (GS) method is implemented through many steps over all degrees of freedom. During each step, only one equation of Eq. (1) is sequentially solved. Usually, this technique is called relaxation. Here, a single coordinate vector is defined at each step, and it is equal to the orth

$$\mathbf{\Phi}_{(i)} = \left[\boldsymbol{\varphi}_{(i),1}\right] = \left[\mathbf{e}_{(i)}\right] \tag{19}$$

Obviously, a small system degenerates to one equation which is repeatedly solved. In this way, the equilibrium of only corresponding unknown is satisfied, while others are treated as constants known from earlier steps. Therefore, the equilibrium of previous equations is simultaneously disturbed. However, during the convergent process disturbance (residuals) decreases (theoretically) to zero. Relaxation of all unknowns is called the cycle. At the end of each cycle only the last equation is equilibrated. It must be noted that cycle counter is not necessarily needed. It is defined only for better explanation of the relaxation process and to control maximum number of iterations if the algorithm has trouble converging. Generally, after n steps a new cycle is started and the same coordinate vectors from Eq. (19) are repeated. The component equal to one moves from the first to the last place of $\boldsymbol{\varphi}_{(i),1}$. SOR is a modification of GS. The solution increment is extrapolated by ω, disturbing even the equilibrium of current unknown. In standard block variant of GS and SOR, instead of only one, all node DOFs are relaxed at each step.

4.1.1 Relaxation order: Motivation for better convergence

The behavior of the iterative process strongly depends on the relaxation order (Duff and Meurant 1989). The most common are forward and backward orderings. For two-dimensional rectangular meshes relaxation is possible from right to left instead of traditionally left to right. The chessboard scheme is also recommended: the algorithm “visits” the “white” nodes in the first half-cycle and the “black” ones in the second half-cycle. Also, columns and rows may interchange places. If diagonal connections between nodes are present, the number of relaxation paths increases. In three-dimensional rectangular meshes with planar and space diagonal connections new possibilities arise. Complex irregular meshes have a large number of meaningful nodal connections and hence numerous paths of load (residual) propagation. An interesting choice is to relax from the largest to the smallest residual (by absolute value). However, the change of residual in the current node and its topological neighbors requires sorting after each step, which affects the algorithm efficiency. In the case of two or more equal residuals the order between them is arbitrary. (For example, in the beginning of a numerical process, the residual vector coincides with the load vector and equal forces are very probable.). An order used in Cross and Southwell relaxation algorithms is also interesting: at each step the node with the largest residual component (unbalanced force or moment) is relaxed and residual vector is updated. Here, the cycle counter does not have real meaning at all. Usually some nodes are relaxed many times before the first visit to other nodes.

4.2 Steepest descent (SD), jacobi (JAC) and conjugate gradient (CG) methods

In SD a single coordinate vector equal to the current residual is defined

$$\mathbf{\Phi}_{(i)} = \left[\boldsymbol{\varphi}_{(i),1}\right] = \left[\mathbf{r}_{(i)}\right] \tag{20}$$

Similarly, JAC is obtained as

$$\mathbf{\Phi}_{(i)} = \left[\boldsymbol{\varphi}_{(i),1}\right] = \left[\mathbf{D}^{-1}\mathbf{r}_{(i)}\right] \tag{21}$$

where $\mathbf{D} = \mathrm{diag}(\mathbf{K})$. Corresponding small systems also degenerate to one equation. If Eq. (20) and Eq. (21) are compared, JAC may be understood as extension of SD using diagonal entries. Strategies are not efficient but serve as good introductory examples for gradient methods. However, block JAC variants can be successfully used, especially in parallel computing. The first step of CG is identical to SD. In the following steps two coordinate vectors are used: $\mathbf{r}_{(i)}$ and $\Delta\mathbf{u}_{(i-1)}$. The coordinate matrix is

$$\mathbf{\Phi}_{(i)} = \left[\boldsymbol{\varphi}_{(i),1} \quad \boldsymbol{\varphi}_{(i),2}\right] = \left[\mathbf{r}_{(i)} \quad \Delta\mathbf{u}_{(i-1)}\right] \tag{22}$$

and a small system reduces to two equations. Compared to SD or JAC, the second vector gives CG much faster convergence. In typical numerical implementation, the solution of two equations is sought by the equivalent recursive **K**-orthogonalization. Because of accumulation of round-off errors, orthogonality really exists only for few adjacent vectors and for large ill-conditioned systems convergence difficulties are present. Many restart (Dai *et al*. 2004) and preconditioning techniques (Adams 1985, Van der Vorst and Dekker 1988) improve convergence. If CG formulation according to Eq. (22) is used, error in **K**-orthogonality also exists, but it is not accumulated during iterative process. Therefore, inherited errors decrease, though non-exact arithmetic (as in every numerical process) causes new errors and affects the convergence. Because two approaches are equivalent, it is possible to interchange them. Each step may be executed by CG or IRM, no matter how earlier steps are performed. It is suggested that after some number of CG steps, before orthogonality error becomes too large, one equivalent IRM step is executed. Instead of "restart" it may be called "refresh".

5. On the generation of efficient coordinate vectors (step 6 of the Algorithm 1)

Basic idea is to define the coordinate matrix **Φ** so that IRM converges faster than other methods. Even though the number of vectors should be small, we believe that only one (as in GS, SOR, SD and JAC) or two (as in CG) are less than optimum. Because $m \ll n$, in the early phase of iterative process increment $\Delta\mathbf{u}_{(i)}$ can only accidentally hit the solution $\mathbf{u}$.

It would be natural to add the third vector to two CG vectors. The subspace is expanded, and compared to the smaller one, the larger energy reduction per step is expected. In the worst case, contribution of a new vector is equal to zero. In other words: the minimum of energy functional in higher dimensional subspace is less (in the worst case equal) than the minimum in lower dimensional subspace. Using this strategy further vectors can be added and even larger energy reduction per step may be expected. So, most of the static system energy should be exhausted in small number of steps and the system "damped" to the lowest energy point-solution. Obviously, the subspace expansion is attractive but, as mentioned before, only up to a certain point.

5.1 Necessary conditions

The matrix $\overline{\mathbf{K}}_{(i)}$ is singular if two or more coordinate vectors are linearly dependent, i.e., relation

$$\sum_{i=1}^{m} b_i \boldsymbol{\varphi}_i = \mathbf{0} \;\Rightarrow\; b_i = 0 \tag{23}$$

where b_i are scalars, is not satisfied. In numerical implementation this condition must be modified-vectors should not even be nearly linearly dependent, otherwise $\bar{\mathbf{K}}_{(i)}$ would be almost singular (ill-conditioned)

$$\left\| \sum_{i=1}^{m} b_i \boldsymbol{\varphi}_i \right\| < \delta_1 \;\Rightarrow\; |b_i| < \delta_2 \tag{24}$$

So, if (some) norm of independent vectors is less than a small number δ_1, the linear independence implies that all b_i must be (by absolute value) less than δ_2. Obviously, generation routines which prevent Eqs. (23) and (24) are preferred and therefore may even change between steps. Nevertheless, if dependence happens, during the decomposition of $\bar{\mathbf{K}}_{(i)}$ some pivots become equal (close to) zero. This can be recognized and used for discarding corresponding equations from the small system. The subspace dimension is reduced, but $\bar{\mathbf{K}}_{(i)}$ becomes regular and better conditioned. It is faster than orthogonalization, rejection or replacement of vectors.

If coordinate vectors and the residual are mutually orthogonal, the subspace is not useful. Then $\bar{\mathbf{r}}_{(i)} = \boldsymbol{\Phi}_{(i)}^{\mathrm{T}} \mathbf{r}_{(i)} = \mathbf{0}$, $\mathbf{a}_{(i)} = \mathbf{0}$, hence the energy is not reduced, and IRM does not converge. To avoid this situation the inequality

$$\frac{\left\| \boldsymbol{\Phi}_{(i)}^{\mathrm{T}} \mathbf{r}_{(i)} \right\|_2}{\left\| \mathbf{r}_{(i)} \right\|_2} < \delta_3 \tag{25}$$

where δ_3 is not too small number, should be fulfilled. To achieve this, it is sufficient to choose one vector j that is not (almost) orthogonal to residual ($\boldsymbol{\varphi}_{(i),j}^{\mathrm{T}} \mathbf{r}_{(i)} > 0$). Also, the subspace must change between steps as the new residual is orthogonal to the previous subspace ($\boldsymbol{\Phi}_{(i)}^{\mathrm{T}} \mathbf{r}_{(i+1)} = \mathbf{0}$). Generally, in IRM orthogonality only between successive (not distant) increments, residuals or subspaces exists. If $\omega \neq 1$, even successive orthogonality is lost. However, the convergence is often improved.

5.2 Preferred properties

Compared to continuous Ritz functions, in this discrete approach compatibility conditions (CC) and essential boundary conditions (EBC) are not defined. If only Eq. (1) is known (not the corresponding model), it can only be concluded that such properties are contained in **K**. However, coordinate vectors which approximate continuous functions that satisfy CC and EBC are better and produce more realistic (stiffness) elements of $\bar{\mathbf{K}}_{(i)}$. If the residual is decomposed into the eigenvectors of **K**, such vectors give a residual with small proportion of high modes (eigenvectors with large eigenvalues). They are called “smooth” vectors and they ensure faster convergence rate. On the contrary, “coarse” vectors produce too stiff matrix elements and locking of the residual. If coordinate vector $\boldsymbol{\varphi}_i$ (iteration counter is omitted) is spanned by normalized eigenvectors $\mathbf{v}_j$

$$\boldsymbol{\varphi}_i = \sum_{j=1}^{n} h_j \mathbf{v}_j \tag{26}$$

the diagonal element of the Ritz matrix is equal

$$\bar{k}_{i,i} = \boldsymbol{\varphi}_i^{\mathrm{T}} \mathbf{K}\, \boldsymbol{\varphi}_i = \sum_{j=1}^{n} h_j^2 \lambda_j \tag{27}$$

where λ_j is j-th eigenvalue of $\mathbf{K}$. Coefficients h_j^2 with larger λ_j contribute more to stiffness $\bar{k}_{i,i}$. Also, "coarse" vectors have large Squared Residual Norm $\left\|\mathbf{r}_{(i)}\right\|_2^2$.

During the solution process, the contribution of "smooth" vectors in the region of higher eigenvalues decreases and the convergence in the region of lower eigenvalues increases, i.e. such vectors effectively reduce the contribution of lower eigenvectors. Thus, the procedure behaves as if the influence of such vectors does not exist (the system matrix condition number is smaller), which speed up the convergence. If couple of such vectors are generated, IRM should be efficient and competitive.

5.3 Present difficulties

Although the above conditions reduce many possibilities, the number of promising coordinate vectors still remains large. Unfortunately, adequate criteria for the selection of generally efficient vectors is not known and the background theory is not well developed. For one group of models some vectors work fine but respond badly for another. Under such conditions, an efficient (small, fast and models independent) subspace would be of great significance (Arjmandi and Lotfi 2011).

5.4 Generation strategies

Two basic generation strategies will be described, but many mixed ideas are also possible (Eftekhari 2018). In the first approach only $\mathbf{K}$ and $\mathbf{f}$ are necessary. If vectors are properly defined, in most practical cases good convergence is obtained. In the second approach, some special data about the model are used. Then, very fast convergence is expected, but only for that particular model and eventually for similar ones.

5.4.1 Generation of constant vectors

The simplest idea is to choose coordinate vectors in advance for the whole numerical process. For example, in GS and SOR number of vectors in n-dimensional space is equal to n, and the same sequence is cyclically repeated until convergence (Section 4.1).

5.4.2 Generation based on the current residual

Here, many possibilities exist. Motivation is as follows. As the "expensive" calculation of the initial error $\mathbf{K}^{-1}\mathbf{r}_{(0)}$ immediately leads to the problem solution, it is natural to find some matrix $\mathbf{P}$ that "cheaply" approximates $\mathbf{K}^{-1}$. The coordinate vector is simply $\boldsymbol{\varphi}_{(i)} = \mathbf{P}\mathbf{r}_{(i)}$. Use of $\mathbf{r}_{(i)}$ has an additional advantage, as it ensures non-orthogonality of the subspace to the residual itself, which is important for convergence (Section 5.1). Simply, if $\mathbf{P}$ is positive definite then $\boldsymbol{\varphi}_{(i)}^{\mathrm{T}}\mathbf{r}_{(i)} = \mathbf{r}_{(i)}^{\mathrm{T}}\mathbf{P}\mathbf{r}_{(i)} > 0$ unless $\mathbf{r}_{(i)} = \mathbf{0}$, which means that the solution is reached. Obviously, an adequate choice is to use one positive definite $\mathbf{P}$, but for generation of other coordinate vectors it is not necessarily needed, and it can be non-symmetric, ill-conditioned, even singular, as in all cases $\bar{\mathbf{K}}_{(i)}$

remains invertible. Obviously, it is not acceptable to generate all vectors with matrices of the same singularity. It is not an imperative that a single vector approximates $\mathbf{K}^{-1}\mathbf{r}_{(i)}$ well. It is sufficient that the approximation of this product inside the subspace is as good as possible.

For example, if **K** is approximated by identity matrix ($\mathbf{P} = \mathbf{I}$), then $\boldsymbol{\varphi}_{(i)} = \mathbf{r}_{(i)}$, which gives SD. If $\mathbf{P} = \mathbf{D}^{-1}$, $\boldsymbol{\varphi}_{(i)} = \mathbf{D}^{-1}\mathbf{r}_{(i)}$ and JAC is obtained (Section 4.2). Methods are obviously inefficient because **P** is crude approximation (contains insufficient data) of $\mathbf{K}^{-1}$.

Better coordinate vectors are generated using one or several cycles of GS or SOR. It may be useful to "visit" nodes in various ways (Section 4.1). However, to save computing time, uncomplete strategies are preferred: prevent return to the already relaxed nodes. For example, nodes could be sorted in the first cycle (according to the absolute values of the residual) and corrected only occasionally. Between cycles (even steps maybe) a local relaxation factor Ω, different from global ω, can be introduced, but the optimal value is subject to research. If only first cycle of the forward SOR is used, coordinate vector is $\boldsymbol{\varphi}_{(i)} = \mathbf{L}_{\Omega}{}^{-1}\mathbf{r}_{(i)}$, so $\mathbf{P} = \mathbf{L}_{\Omega}{}^{-1}$, where $\mathbf{L}_{\Omega}$ is lower triangular of **K** with diagonal elements multiplied by Ω. If backward order is applied, $\mathbf{L}_{\Omega}$ is replaced by upper triangular $\mathbf{U}_{\Omega}$. We reversely called this method the ROS. If $\Omega = 1$, matrices are replaced by standard **L** and **U** matrices and the first cycle of forward and backward GS is obtained. Notice that such coordinate vectors are quickly generated during the step. Various alternate approaches (at additional cost) are also possible. For motivation, three relatively successful examples are given below. Matrix **P** is easily recognized

$$\boldsymbol{\varphi}_{(i)} = \mathbf{L}_{\Omega}^{-1}\left(\mathbf{I} - \mathbf{K}\,\mathbf{U}_{\Omega}^{-1}\right)\left(\mathbf{I} - \mathbf{K}\,\mathbf{L}_{\Omega}^{-1}\right)\left(\mathbf{I} - \mathbf{K}\,\mathbf{U}_{\Omega}^{-1}\right)\ldots\mathbf{r}_{(i)} \tag{28}$$

$$\boldsymbol{\varphi}_{(i)} = \mathbf{L}_{\Omega}^{-1}\mathbf{U}_{\Omega}^{-1}\mathbf{L}_{\Omega}^{-1}\mathbf{U}_{\Omega}^{-1}\ldots\mathbf{r}_{(i)} \tag{29}$$

$$\boldsymbol{\varphi}_{(i)} = (\mathbf{L}_{\Omega}^{-1} + \mathbf{U}_{\Omega}^{-1})\,(\mathbf{L}_{\Omega}^{-1} + \mathbf{U}_{\Omega}^{-1})\ldots\mathbf{r}_{(i)} \tag{30}$$

In the last equation, **K** can be placed between parentheses. Using Eqs. (28)-(30) independently, smaller Ω makes convergence slower, but faster "smoothing" is ensured. Therefore, it would be efficient to use them with other vectors. Similarly, if number of cycles (successive changes of $\mathbf{L}_{\Omega}^{-1}$and $\mathbf{U}_{\Omega}^{-1}$) is increased, "smoother" vector is obtained.

Inspired by previous comments, adequate coordinate vectors can be generated by direct smoothening of $\mathbf{r}_{(i)}$. This is called "filtering". A component of such vector is equal to sum of residual components in the neighboring nodes, multiplied by the weighting factors. This approach was efficient in models with very large residual jumps. They occur due to the poor prediction of displacement in some steps, which often appears in the vicinity of supports and free boundaries.

In preconditioning (Wathen 2015), some matrix **M** is used to produce better conditioned system, equivalent to (1). Then $\mathbf{M}^{-1}\mathbf{K}\mathbf{u} = \mathbf{M}^{-1}\mathbf{f}$ is indirectly solved. Of course, **M** should be rapidly inverted. From the IRM strategy, $\mathbf{P} = \mathbf{M}^{-1}$and the coordinate vector is accordingly given. For smoothing purposes, forward and backward techniques are also preferred. If couple of vectors are generated, several preconditioning techniques (matrices **P**) can be simultaneously used. This is analogous to multiple preconditioning. For the IRM, matrix transformations used for preconditioning are not needed. The preconditioning is just another way of generating coordinate vector(s). During the generation process, they can also become (exactly or approximately) linearly dependent and one or several should be excluded.

In this paper, coordinate vectors are generated using one cycle of the Symmetric Successive

Over-Relaxation Method (SSOR), with $\Omega = 1$. The first vector is obtained as

$$\boldsymbol{\varphi}_{(i),1} = \mathbf{L}_{\Omega}^{-1}\mathbf{D}\,\mathbf{U}_{\Omega}^{-1}\mathbf{r}_{(i)} \tag{31}$$

and others are recursively formed

$$\boldsymbol{\varphi}_{(i),j} = \mathbf{L}_{\Omega}^{-1}\mathbf{D}\,\mathbf{U}_{\Omega}^{-1}\left(\mathbf{K}\,\boldsymbol{\varphi}_{(i),j-1}\right), \quad j = 2,\ldots,m-1 \tag{32}$$

At the end, the previous increment is also added $\boldsymbol{\varphi}_{(i),m} = \Delta\mathbf{u}_{(i-1)}$. Calculations are performed with two, four, six and ten vectors, using $\omega = 1$. Let's explain the expression (32). As one step in the direction of $\boldsymbol{\varphi}_{(i),1}$ gives the current increment $\alpha\boldsymbol{\varphi}_{(i),1}$, where α is a scalar, according to Eq. (16) the residual is $\mathbf{r}_{(i)} - \alpha\mathbf{K}\boldsymbol{\varphi}_{(i),1}$. Inserting it in Eq. (31) the second vector is $\boldsymbol{\varphi}_{(i),2} = \boldsymbol{\varphi}_{(i),1} - \alpha\mathbf{L}_{\Omega}^{-1}\mathbf{D}\,\mathbf{U}_{\Omega}^{-1}\left(\mathbf{K}\boldsymbol{\varphi}_{(i),1}\right)$. The vector $\boldsymbol{\varphi}_{(i),1}$ already spans the subspace and can be omitted, while α affects only the vector length, but not the subspace dimension. Therefore, $\alpha = 1$ is used and Eq. (32) is obtained. An interesting, but more "expensive" variation of this procedure is to use **K** instead of **D**.

Some remarks. Strategies of this subsection are generally valid. Coordinate vectors can be generated using ($\mathbf{r}_{(i)}$ and) **P** obtained by any iterative method. Furthermore, every vector (matrix **P**) may be generated with a different method. For example, first vector is generated by JAC, second by SOR, third by SSOR, and further vectors by the Incomplete Cholesky Factorization (ICC), the Algebraic Multigrid (AMG), the Sparse Approximate Inverse (SAI), *etc*. (Higham 2009, Stüben 2001, Xu and Zikatanov 2017, Huckle 1998, Ferronato *et al*. 2015). All methods can be used forward and backward, or in any promising order of the unknowns. It is even possible to use algorithms that are not useful as standalone solvers, as they are neither convergent nor numerically stable. Thus, Ω need not lie between limits of ω (Section 3.1.3).

Generally, two or three coordinate vectors that smoothen the lower part of the residual spectrum, and next two or three for the upper part should be considered. Increment $\Delta\mathbf{u}_{(i-1)}$, the essence of the CG success, is also preferable. Similar to the described effect of one vector, here the whole subspace is smoothened by vectors used. In other words, the residual is smoother with the increase of both, vectors and (or) cycles used for the generation of single vector. This contributes to faster convergence.

5.4.3 Generation using data from the previous step

Coordinate vectors may be obtained by "history recycling", i.e., use of the increment $\boldsymbol{\varphi}_{(i)} = \Delta\mathbf{u}_{(i-1)}$. It accelerates the convergence of CG and enables inclusion of ω. This vector is generally efficient for IRM, although recursive orthogonality to earlier vectors is lost (Section 5.1). Additional vectors can be generated with some matrices $\mathbf{S}_j$ as $\boldsymbol{\varphi}_{(i),j} = \mathbf{S}_j\Delta\mathbf{u}_{(i-1)}$. Adding earlier increments $\Delta\mathbf{u}_{(i-2)}$, $\Delta\mathbf{u}_{(i-3)}$, *etc*., is not efficient, especially in non-smooth (*e.g*. contact) problems. The solution $\mathbf{u}_{(i)}$ can be similarly used, particularly in a nonlinear environment. The recycling of this vector makes sense, because of just mentioned overall orthogonality loss, which in nonlinear models exists by definition. Finally, it should be emphasized that this group of vectors is not independently used.

5.4.4 Generation using model data

The second strategy for the generation of vectors is to use additional data model (e.g.,

Ibrahimbegovic and Wilson 1990b, Ibrahimbegovic and Wilson 1992). The approximate geometry, simplified properties (stiffness distribution) or a simpler model of a similar problem (less important DOFs are omitted) are exploited. Generally, the displacement of such models loaded by the residual can be used as the coordinate vector. They are crude, but efficient at the early stage of iterative process. Later, the solution needs to be smoothed by more accurate approaches from previous sections. Nevertheless, such vectors can ensure a very fast solution, but the main drawback of this strategy is the lack of generality.

For example, the substitute model for a thick beam can be a thin beam, and a membrane could be used instead of a shell. If the solution $\mathbf{u}_s$ of the substitute model is known, the vector is $\boldsymbol{\varphi}_j = \mathbf{N}\mathbf{u}_s$, where **N** is the interpolation matrix that connects degrees of freedom of the substitute and the original model. In the clamped thick beam case, the original model is defined by planar elements and the substitute model may be based on line elements. Even pinned supports could be used (Fig. 3). The third-degree polynomials connect the nodes of both models on the axis, while the nodes (of the original model) outside the axis are connected to the substitute model by the beam kinematic hypothesis. Columns of **N** are defined by polynomials and hypothesis. Such a matrix is singular, as displacements between models are linearly dependent, but the coordinate vector is correct and leads to the solution of a thin beam. Obviously, this is not a solution to the original problem. Therefore, the residual (or some other "corrector") must be added as an additional vector to weaken the kinematic hypothesis and to ensure an adequate solution for a thick beam.

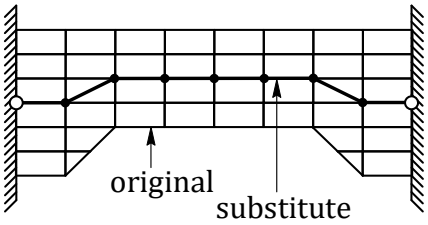

Fig. 3 Original and substitute deep beam model

The iterative procedure based on the stiffness hierarchy of a complex model can also be used as a generator of the coordinate vector. Such vector is defined as a solution after only several steps between parts of the model with different stiffnesses. It is only a crude approximation of the complex model result, but it is suitable for the fast vector definition.

Vectors can also be generated using coarse FE mesh, various multigrid methods (Wilson and Naff 2010), or analytically defined (continuous) coordinate functions which satisfy EBC over the approximate model domain. Notable examples are Ritz functions, and an interesting extension are R-functions (Shapiro 1988, Rvachev *et al*. 1999). Both can be multiplied by polynomials. Even more freely selected smooth functions (polynomials) that do not satisfy BC can be used as vectors. Details, for example openings, may be omitted. In such cases vector components are equal to the function nodal values. Such vectors are "rough" and the corresponding elements of $\overline{\mathbf{K}}_{(i)}$ are "too stiff" but are useful at the beginning of the iterative process. If they are kept, the solution will be smoothened in later steps, though the convergence will not be impressive. A better idea is to somehow smooth such vectors in advance. For example, for each vector one or two SOR and then ROS relaxation cycles with small Ω may be beneficial.

Coordinate vectors can also be generated using analogies. For a slab problem, the solution of a magnetic or electric field, membrane or grid can be considered. The solution of differently loaded slab or even roughly sketched displacement field can also be applied. Measuring from the sketch, displacements as vector components are determined.

Kinematic constraints (beam kinematic hypothesis is an example) can be efficiently defined by coordinate vectors. If they are expressed as $\mathbf{Tu} = \mathbf{f}$, where $\mathbf{T}$ is a constraint matrix, then the initial guess $\mathbf{u}_{(0)}$ should satisfy nonhomogeneous constraints, $\mathbf{Tu}_{(0)} = \mathbf{f}$, and vectors must satisfy the homogenous one, $\mathbf{T\Phi}_{(i)} = \mathbf{0}$.

A mixed approach to the generation of vectors may also be interesting. For example, it may be tried to utilize products of $\Delta\mathbf{u}_{(i)}$ or $\mathbf{u}_{(i)}$ with some functions of coordinates (polynomials). The number of possible strategies for numerical experiments is enormous (Markovic *et al*. 2006, Markovic *et al*. 2009).

5.4.5 Coordinate vectors as input data

Because of numerous possibilities, it is reasonable to consider coordinate vectors as the input data. So, if an appropriate set of vectors is found, the matrix $\mathbf{\Phi}_{(i)}$ is easily formed. In this way IRM can be considered as an iterative method where, in addition to common data used by most iterative algorithms, coordinate vectors should also be given.

6. Briefly about implementation

The pseudocode 1 was implemented by GFORTRAN programming language (The GFORTRAN team). UBUNTU version 5.3.1 and OS X version 6.1.0 (both 64-bit) were used. The program was first verified on small examples and then on larger number of planar and space trusses. Also, to make condition number $\kappa(\mathbf{K})$ much larger, bars with huge stiffness differences are randomly connected to distant nodes. Hence, illogical forms that cannot be regarded as structures are created. Thus, program is tested on both, well and ill-conditioned models. $\mathbf{K}$ is stored in a compressed sparse column (CSC) format, but a row format was also tried (Nour-Omid and Taylor 1984). Both strategies exist in the open-source FEA program FEAP (Taylor 2014). Thus, the connection with this package is easily obtained. The version 8.4.1 was used. Compiling and linking with the FEAP was also made with the GFORTRAN.

7. Practical results

After fundamental tests, several models from the structural engineering practice were analyzed (Figs. 4-10). Vertical loads and supports are not drawn. Models comprise beam, shell and volume FE with displacements and rotations as unknowns. Large axial stiffness of stocky RC elements and small bending stiffness of slender steel elements are mixed. Uniformly distributed eigenvalues and larger $\kappa(\mathbf{K})$ were preferred. This is common in fine-tuned structures as eigenvalue clusters amplify the dynamic and the stability response. The basic data are given in the Table 2. Various tests of iterative methods were necessary as their problem dependence is fairly known, not only on the stiffness (the spectrum and the condition number), but also on loading and support conditions. Notice simple benchmark in Fig. 4 with relatively small $\kappa(\mathbf{K})$.

Table 2 Basic data about numerical models

Figure	Nodes	Elements	Unknowns	CSC storage	Fill-in	$\approx \kappa(\mathbf{K})$
4	132 651	125 000	397 947	15 692 116	$9{,}90 \cdot 10^{-5}$	10^3
5	10 329	16 104	61 776	1 434 447	$3{,}76 \cdot 10^{-4}$	10^9
6	21 420	27 441	117 936	3 133 980	$2{,}25 \cdot 10^{-4}$	10^{11}
7	71 307	278 499	206 527	3 826 156	$8{,}97 \cdot 10^{-5}$	10^6
8	43 815	43 072	258 954	7 053 951	$1{,}10 \cdot 10^{-4}$	10^6
9	79 162	74 247	289 986	10 251 174	$1{,}22 \cdot 10^{-4}$	10^8
10	276 244	1 461 134	820 446	17 723 235	$2{,}63 \cdot 10^{-5}$	10^8

Calculations were performed using SD, JAC, CG without and with the diagonal (CGD) and block-nodal (CGBN) preconditioners, and finally with IRM$(\cdot)$. CGD and CGBN are FEAP iterative solvers. IRM argument denotes the number of coordinate vectors used. Acronyms and corresponding full names of methods used are listed in the Table 3. The distribution of vertical displacements (in meters), and the logarithm of the $\|\mathbf{r}_{(i)}\|_2/\|\mathbf{r}_{(0)}\|_2$ vs number of steps are given.

Table 3 List of acronyms and full names of the used iterative methods

SD	Steepest Descent
JAC	Jacobi
CG	Conjugate Gradient
CGD	Conjugate Gradient with Nodal Preconditioner
CGBN	Conjugate Gradient with Block Nodal Preconditioner
IRM(2)	Iterated Ritz Method with two coordinate vectors
IRM(4)	Iterated Ritz Method with four coordinate vectors
IRM(6)	Iterated Ritz Method with six coordinate vectors
IRM(10)	Iterated Ritz Method with ten coordinate vectors

In Fig. 4, the energy ratio $(\mathrm{G}(\mathbf{u}_{(0)}) - \Sigma_i \mathrm{G}(\Delta\mathbf{u}_{(i)}))/\mathrm{G}(\mathbf{u}_{(0)})$ is also added. For this example, SD and JAC, also CGD and CGBN have almost the same energy curves, so only SD and CGD are added.

Generally, SD, JAC and CG respond badly, as is expected for such examples. The preconditioned CG versions are better, although not impressive. Even for two vectors, IRM converges faster than other methods considered. With the subspace increase, the step "price" rises, but the residual and the energy reduction per step are larger, so that the number of steps further decreases (Table 4). However, it should be mentioned that the better CG preconditioners for the SPD systems exist, such as (not always robust) ICC with different fill-ins (Van't Wout *et al*. 2010), then AMG (Iwamura *et al*. 2003, Pereira *et al*. 2006) and SAI (Benzi *et al*. 1996) methods.

But a better subspace for IRM is also possible. As it was already mentioned, IRM vectors can always be based on any of methods considered and robustness is not necessary. Thus, refined algorithms of such methods are not imperative. Relatively crude vectors that complement each

Table 4 Behavior of some methods according to number of steps

Figure	Steps until convergence ($\delta = 10^{-8}$)					
	CGD	CGBN	IRM(2)	IRM(4)	IRM(6)	IRM(10)
4	1 396	1 394	456	305	233	159
5	2 049	757	704	235	141	79
6	4 155	3 114	1 303	502	324	167
7	52 433	49 040	17 086	5 703	3 422	1 925
8	4 682	4 040	1 509	501	300	166
9	5 708	5 155	1 559	565	313	191
10	5 076	4 931	1 915	827	486	270

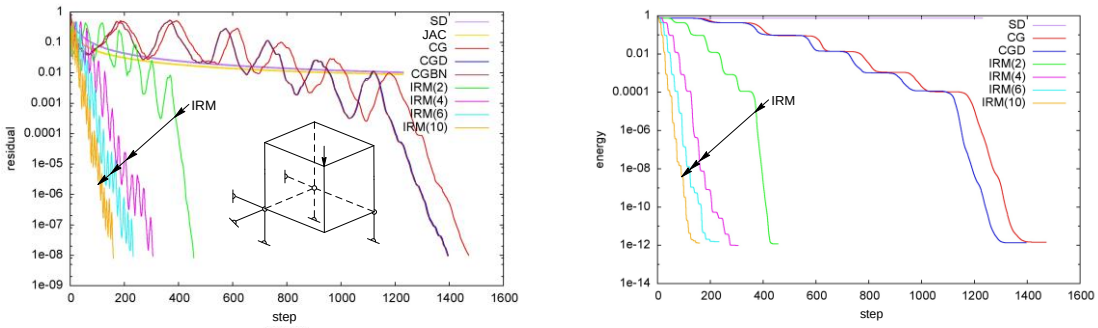

Fig. 4 Simple benchmark

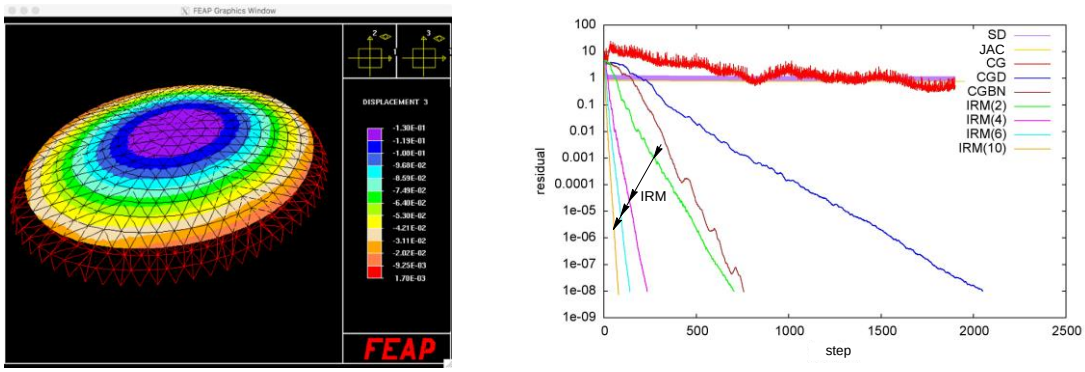

Fig. 5 Large steel reticulated dome with cover

other (in sense of the residual reduction at different parts of the spectrum) are important.

8. Conclusions

The explanation of IRM is close to engineering interpretations. The method should not be worse than (preconditioned) CG. Additionally, various standard iterative methods may be simultaneously used to generate different coordinate vectors. Thus, if appropriate vectors are

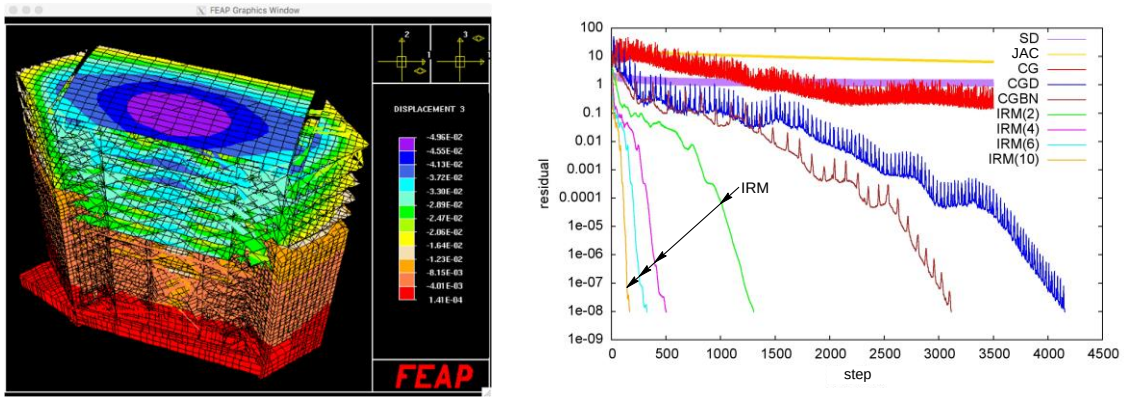

Fig. 6 RC building with a large steel appendage

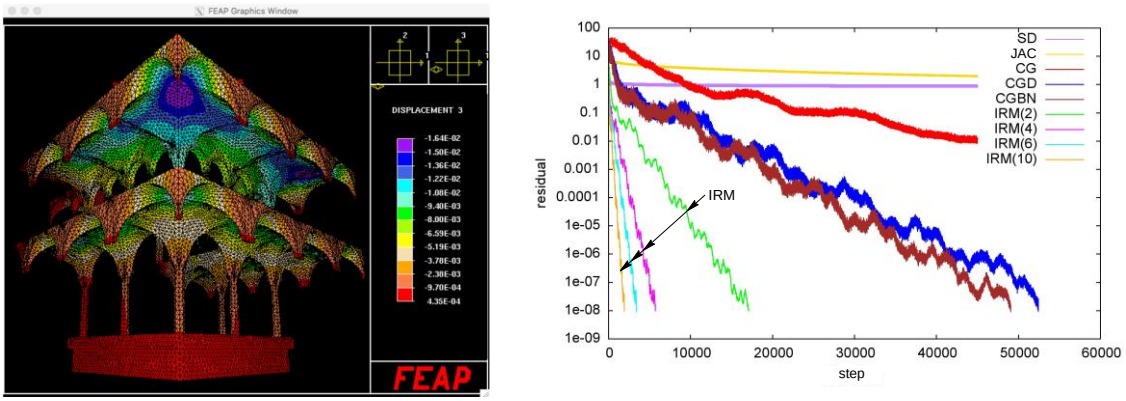

Fig. 7 Ancient atrium with stone columns, groin vaults and fill material

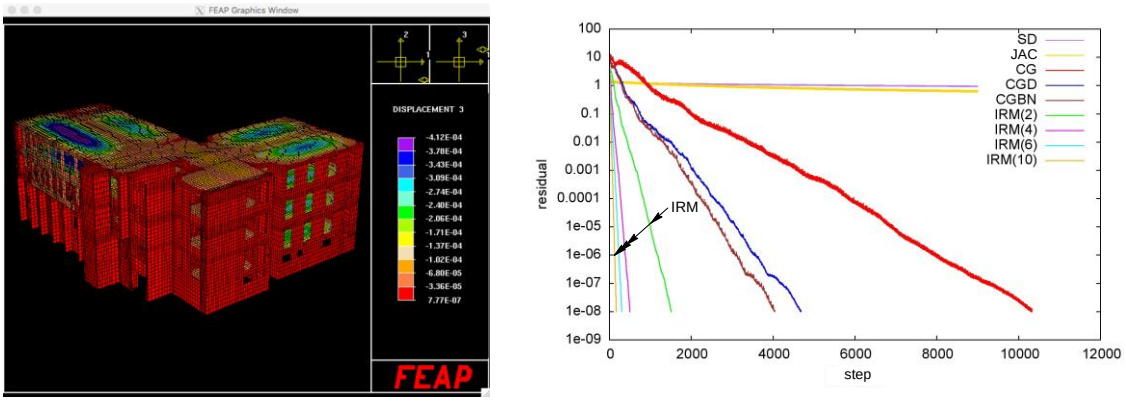

Fig. 8 Standard RC building

selected, the convergence should be faster than using any single method considered. The preconditioning is not necessary here, but such algorithms can successfully be applied for the generation process. The restart known from the standard CG (due to **K**-orthogonality loss) is not needed. From the previous step, only the displacement increment is inherited, which improves the

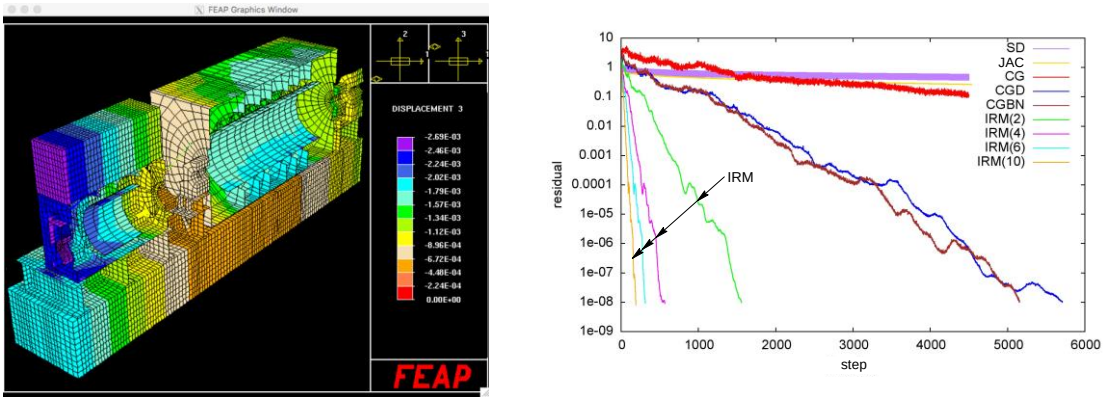

Fig. 9 Turbo generator with horizontal steel frame and RC foundation (half-model shown)

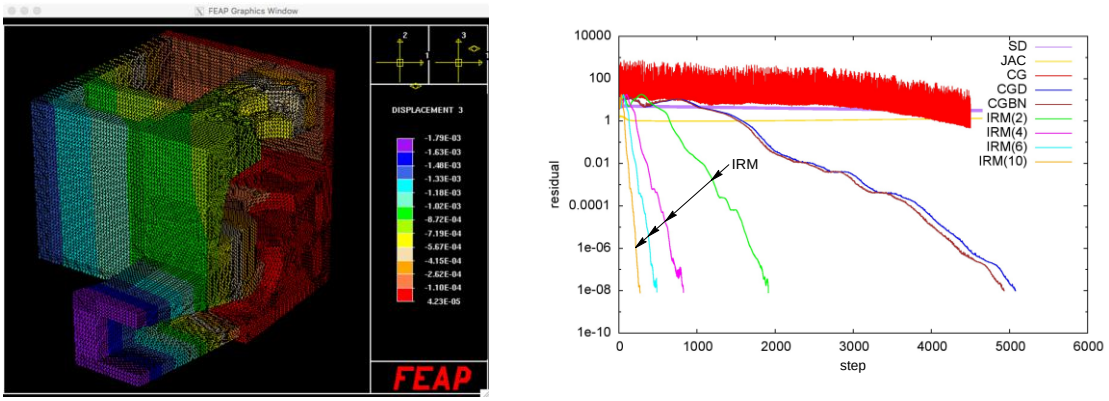

Fig. 10 RC powerhouse of hydropower plant (half-model shown)

convergence. As the various orthogonalities between steps (favored by CG) are not required, the method may also be competitive in nonlinear (Carvalho *et al*. 2013) and optimization (Guerra and Kiousis 2006) problems. Obviously, the parallel approach is essential for solving large systems (Reed and Patrick 1985, Da Cunha and Hopkins 1995) and the possibility of assigning one processor per coordinate vector is challenging. As IRM is still being investigated, the overall performance of the method is not measured and the comparison with other direct and iterative solvers is not given.

Acknowledgments

This work has been fully supported by Croatian Science Foundation under the project IP-2014-09-2899.

References

Adams, L. (1985), "m-Step preconditioned conjugate gradient methods", *SIAM J. Sci. Statis. Comput.*, **6**(2),

452-463.
Arjmandi, A. and Lotfi, V. (2011), "Dynamic analysis of structures in frequency domain by a new set of ritz vectors", *Struct. Eng. Mech.*, **39**(5), 703-716.
Benzi, M. (2002), "Preconditioning techniques for large linear systems: A survey", *J. Comput. Phys.*, **182**(2), 418-477.
Benzi, M., Meyer, C.D. and Tuma, M. (1996), "A sparse approximate inverse preconditioner for the conjugate gradient method", *SIAM J. Sci. Comput.*, **17**(5), 1135-1149.
Carvalho, G., Bento, R. and Bhatt, C. (2013), "Nonlinear static and dynamic analyses of reinforced concrete buildings-comparison of different modelling approaches", *Earthq. Struct.*, **4**(5), 451-470.
Da Cunha, R.D. and Hopkins, T. (1995), "The Parallel Iterative Methods (PIM) package for the solution of systems of linear equations on parallel computers", *Appl. Numer. Math.*, **19**(1/2), 33-50.
Dai, Y.H., Liao, L.Z. and Li, D. (2004), "On restart procedures for the conjugate gradient method", *Numer. Algorit.*, **35**, 249-260.
Duff, I.S. and Meurant, G.A. (1989), "The effect of ordering on preconditioned conjugate gradients", *BIT Numer. Math.*, **29**(4), 635 - 657.
Dvornik, J. (1979), "Generalization of the CG method applied to linear and nonlinear problems", *Comput. Struct.*, **10** (1/2), 217-223.
Eftekhari, S.A. (2018), "A coupled ritz-finite element method for free vibration of rectangular thin and thick plates with general boundary conditions", *Steel Compos. Struct.*, **28**(6), 655-670.
Ferronato, M. (2012), "Preconditioning for sparse linear systems at the dawn of the 21st century: History, current developments and future perspectives", *ISRN Appl. Math.*, Article ID 127647.
Ferronato, M., Janna, C. and Gambolati, G. (2015), "A novel factorized sparse approximate inverse preconditioner with supernodes", *Proc. Comput. Sci.*, **51**, 266-275.
Guerra, A. and Kiousis, P.D. (2006) "Design optimization of reinforced concrete structures", *Comput. Concrete*, **3**(5), 313-334.
Higham, N.J. (2009), "Cholesky factorization", *WIREs Comput. Statis.*, **1**(2), 251-254.
Huckle, T.K. (1998), "Efficient computation of sparse approximate inverses", *Numer. Lin. Algebr. Appl.*, **5**(1), 57-71.
Ibrahimbegovic, A. and Wilson, E.L. (1990a), "Automated truncation of Ritz vector basis in modal transformation", *J. Eng. Mech.*, **116** (11), 2506-2520.
Ibrahimbegovic, A. and Wilson, E.L. (1990b), "A methodology for dynamic analysis of linear structure-foundation systems with local non-linearities", *Earthq. Eng. Struct. Dyn.*, **19**(8), 1197-1208.
Ibrahimbegovic, A., Chen, H.C., Wilson, E.L. and Taylor, R.L. (1990), "Ritz method for dynamic analysis of large discrete linear systems with non-proportional damping", *Earthq. Eng. Struct. Dyn.*, **19**(6), 877-889.
Ibrahimbegovic, A. and Wilson, E.L. (1992), "Efficient solution procedure for systems with local non-linearities", *Eng. Comput.*, **9**(3), 385-398.
Iwamura, C., Costa, F.S., Sbarski, I., Easton, A. and Li N. (2003), "An efficient algebraic multigrid preconditioned conjugate gradient solver", *Comput. Meth. Appl. Mech. Eng.*, **192**(20/21), 2299-2318.
Lazarevic, D. and Dvornik, J. (2017), "Iterated Ritz Method for solving systems of linear algebraic equations", *Građevin.*, **69** (7), 521-535.
Markovic, D., Park, K.C. and Ibrahimbegovic, A. (2006), "Reduction of substructural interface degrees of freedom in flexibility-based component mode synthesis", *Int. J. Numer. Meth. Eng.*, **70**(2), 163-180.
Markovic, D., Ibrahimbegovic, A. and Park, K.C. (2009), "Partitioning based reduced order modelling approach for transient analyses of large structures", *Eng. Comput.*, **26** (1/2), 46-68.
Nour-Omid, B. and Taylor, R.L. (1984), *An Algorithm for Assembly of Stiffness Matrices into a Compacted Data Structure*, Report No. UCB/SESM-84/06, Structural Engineering and Structural Mechanics, Department of Civil Engineering, University of California, Berkeley, U.S.A.
Olshanskii, M.A. and Tyrtyshnikov, E.E. (2014), *Iterative Methods for Linear Systems. Theory and Application*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, U.S.A.
Pereira, F.H., Lopes Verardi, S.L. and Nabeta, S.I. (2006), "A fast algebraic multigrid preconditioned conjugate gradient solver", *Appl. Math. Comput.*, **179**(1), 344-351.

Reed, D.A. and Patrick, M.L. (1985), "Parallel, iterative solution of sparse linear systems: Models and architectures", *Parall. Comput.*, **2**(1), 45-67.
Rvachev, V., Sheiko, T. and Shapiro, V. (1999), "Application of the method of R -functions to integration of differential equations with partial derivatives", *Cybernet. Syst. Analy.*, **35**(1), 1-18.
Saad, Y. (2003), *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, U.S.A.
Shapiro, V. (1988), *Theory of R-Functions and Applications: A Primer*, CPA Technical Report CPA88-3, Cornell Programmable Automation, Cornell University, Ithaca, New York, U.S.A.
Stüben, K. (2001), "A review of algebraic multigrid", *J. Comput. Appl. Math.*, **128**(1/2), 281-309.
Sun, F.J. and Gu, M. (2016), "Preconditioning technique for a simultaneous solution to wind-membrane interaction", *Wind Struct.*, **22** (3), 349-368.
Taylor, R.L. (2014), *FEAP-A Finite Element Analysis Program*, Version 8.4, Programmer Manual, Department of Civil and Environmental Engineering, University of California at Berkeley, Berkeley, U.S.A.
The GFORTRAN Team (2016), *Using GNU Fortran*, For GCC version 7.0.0 (pre – release), (GCC), Free Software Foundation, Boston, U.S.A.
Van der Vorst, H.A. and Dekker, K. (1988), "Conjugate gradient type methods and preconditioning", *J. Comput. Appl. Math.*, **24**(1/2), 73-87.
Van't Wout, E., Van Gijzen, M.B., Ditzel, A., Van der Ploeg, A. and Vuik, C. (2010), "The deflated relaxed incomplete Cholesky CG method for use in a real-time ship simulator", *Proc. Comput. Sci.*, **1**(1), 249-257.
Wathen, A.J. (2015), "Preconditioning", *Acta Numer.*, **24**, 329-376.
Wilson, J.D. and Naff, R.L. (2010), "Multigrid preconditioned conjugate-gradient solver for mixed finite-element method", *Comput. Geosci.*, **14**(2), 289-299.
Xu, J. and Zikatanov, L. (2017), "Algebraic multigrid methods", *Acta Numer.*, **26**, 591-721.