ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

ANNEX 1: MATLAB Class for algebraic multigrid preconditioner

```
% ------------------------------------------------------------
% Class for the Algebraic Multigrid Preconditioner
% introduced by Rudolf Beck
% ------------------------------------------------------------
classdef amgRB
    properties (GetAccess=private)
        maxSize = 100;  % Default values
        numVCycles = 5; % Default values
        A   = [];     % Matrix for first level
        MA = {};     % List of matrices
        MR = {};     % List of restriction operators
        MP = {};     % List of prolongation operators
        L   = [];     % For LU factorization of the last matrix
        U   = [];     % For LU factorization of the last matrix
        tol = 1e-6; % Tolerance for the iterative solver
        steps = []; % Smoothing steps for Gauss-Seidel (GS)
        ML = {};     % List of lower triangular matrices for GS
        MU = {};     % List of upper triangular matrices for GS
        MFGS = {};  % List of inverse matrices for Forward-GS
        MBGS = {};  % List of inverse matrices for Backward-GS
    end
    % ------------------------------------------------------------
    %                       PUBLIC METHODS
    % ------------------------------------------------------------
    methods
    % ------------------------------------------------------------
    % Construct method
    % This function constructs the levels of the algebraic
    % multigrid preconditioner.
    % ------------------------------------------------------------
    % Receive:
    %   A: is a mxm symmetric positive definite sparse matrix.
    %   maxSize: is the maximum size of the matrix in the last
    %            level.
    %   numVCycles: is the number of V-cycles.
    % Update:
    %   MA: is a list of matrices for coarse matrices
    %   MR: is a list of matrices for restriction operators
    %   MP: is a list of matrices for prolongation operators
    %   L: is the lower triangular matrix of the LU factorization
    %      of the last matrix A
    %   U: is the upper triangular matrix of the LU factorization
    %      of the last matrix A
    %   steps: is a list of the smoothing steps for Gauss-Seidel
    %   ML: is a list of lower triangular matrices for GS
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

```
%   MU: is a list of upper triangular matrices for GS
%   MFGS: is a list of inverse matrices for Forward-GS
%   MBGS: is a list of inverse matrices for Backward-GS
% -------------------------------------------------------------
function obj = amgRB(A, maxSize, numVCycles)
    obj.A = A;
    obj.maxSize = maxSize;
    obj.numVCycles = numVCycles;
    % ----------------- Create the levels -----------------
    numLevels = 0; % Number of levels without the last one
    m = length(A); % Order of the matrix
    while m >= maxSize
        % Mark nodes as Master and Slave
        [marked, sizeM] = coarsening(obj, A);
        % Construct Restriction matrix
        R = sparse(sizeM, m);
        for j = 1 : m
            if marked(j) == 0
                % -------------- SLAVE NODE --------------
                c = A(:,j);  % Define the jth column of A
                I = find(c); % Find indices of non-zero
                             % elements of the jth column of A
                [~,~,I] = find( marked(I) ); % Find indices
                                             % of master nodes
                                             % around node j
                nnz = length(I); % Number of neighboring
                                 % master nodes of node j
                R(I, j) = 1 / nnz;
            else
                % -------------- MASTER NODE --------------
                R(marked(j), j) = 1;
            end
        end
        % Store matrices on lists
        numLevels = numLevels + 1;
        obj.MR{numLevels} = R;
        obj.MA{numLevels} = A;
        obj.MP{numLevels} = R' ;
        % Preparations for next level
        A = R * A * R' ;
        m = sizeM;
    end
    % ---- Compute LU Factorization for the last matrix ---
    [L,U] = lu(A);
    obj.L = L;
    obj.U = U;
    % ---------- Create list of smoothing steps -----------
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

```
    obj.steps = zeros(numLevels, 1);
    for i = 1 : numLevels
        obj.steps(i) = i + 1;
    end
    % ---------------- Setup Gauss-Seidel -----------------
    obj.ML{numLevels}   = []; % Store memory
    obj.MU{numLevels}   = []; % Store memory
    obj.MFGS{numLevels} = []; % Store memory
    obj.MBGS{numLevels} = []; % Store memory
    % Create Gauss-Seidel matrices
    for l = 1 : numLevels
        A = obj.MA{l};      % Get matrix A_l
        I = speye(size(A)); % Create identity matrix
        obj.ML{l} = tril(A,-1); % Lower triangular part of A
        obj.MU{l} = triu(A, 1); % Upper triangular part of A
        obj.MFGS{l} = tril(A) \ I; % Forward-GS matrix
        obj.MBGS{l} = triu(A) \ I; % Backward-GS matrix
    end
end
% ---------------------------------------------------------
% Get the number of levels of the preconditioner
% ---------------------------------------------------------
%   k: is the number of levels.
% ---------------------------------------------------------
function [k] = getNumLevels(obj)
    k = length(obj.MR) + 1;
end
% ---------------------------------------------------------
% Apply preconditioner to a vector, used by PCG method
% ---------------------------------------------------------
%   x, y: are m-size vectors
% ---------------------------------------------------------
function [y] = apply(obj, x)
    y = zeros( size(x) );
    [y, ~, ~] = solve(obj, x, y);
end
% ---------------------------------------------------------
% Iterative solver for Ax = b
% ---------------------------------------------------------
%   b: is the right-hand side of the system
%   x0: is an initial guess for system Ax = b
%   x: is the new approximation for system Ax = b
%   cyclesUsed: is the number of V-cycles used
%   normResidual: is a list for the norm-2 of the
%                 residual for each V-cycle
% ---------------------------------------------------------
function [x, cyclesUsed, normResidual] = solve(obj, b, x0)
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

```
numLevels = length(obj.MR); % Number of levels without
                            % the last one, that is, N-1
% ---------- When the number of levels is 1 -----------
if numLevels == 0
    % Solve system using LU Factorization
    x = obj.U \ ( obj.L \ b );
    cyclesUsed = 0;
    normResidual(1) = norm(b - obj.A*x);
    return % Leave the function
end
% ------------ Compute the first residual ------------
A = obj.A;      % Matrix of the first level
r = b - A*x0;   % First residual
nrmR = norm(r); % Norm-2 of the residual
normResidual(1) = nrmR;
nrm0Tol = obj.tol * nrmR; % Tolerance relative to the
                          % first residual
% ------------------- Star V-cycles -------------------
cyclesUsed = 0;
while cyclesUsed < obj.numVCycles && nrmR >= nrm0Tol
    % ------------ Go downward in V-cycle -------------
    for l = 1 : numLevels
        A = obj.MA{l};        % Get matrix A_l
        x = zeros( size(r) ); % Create solution x_l
        % -----------------------------------------
        x = forwardGaussSeidel(obj, r, x, l);
        % -----------------------------------------
        % Stores new vectors
        approxim{l} = x;
        residual{l} = r;
        % Restrict residual
        r = obj.MR{l} * ( r - A*x );
    end
    % --- Perform LU Factorizacion in the last level --
    approxim{numLevels+1} = obj.U \ ( obj.L \ r );
    % ------------- Go upward in V-cycle --------------
    for l = numLevels : -1 : 1
        A = obj.MA{l};   % Get matrix A_l
        x = approxim{l}; % Get vector x_l
        r = residual{l}; % Get vector r_l
        % Prolonging residual
        x = x + obj.MP{l} * approxim{l + 1};
        % -----------------------------------------
        x = backwardGaussSeidel(obj, r, x, l);
        % -----------------------------------------
        approxim{l} = x; % Stores new approximation
    end
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

```
        % ----------------- End of V-cycle ----------------
        r0 = b - A*x; % Create new residual for the first
                      % level, but this does not modify the
                      % original
        nrmR = norm(r0);
        normResidual(cyclesUsed + 2) = nrmR;
        cyclesUsed = cyclesUsed + 1;
    end
end
end % End Public Methods
% ---------------------------------------------------------
%                      PRIVATE METHODS
% ---------------------------------------------------------
methods (Access = private)
% ---------------------------------------------------------
% This function marks the master and slave nodes for a
% given matrix A of order m.
% ---------------------------------------------------------
%   marked: is a m-size vector which contains 0 on its ith
%           component if node i is slave. Otherwise, node
%           i is master, and the ith component of the vector
%           contains a new numbering for node i.
%   sizeM: is the number of master nodes.
% ---------------------------------------------------------
function [marked, sizeM] = coarsening(obj, A)
    m = length(A); % Order of the matrix
    % ---------------------------------------------------
    nnzRow = zeros(1, m); % Number of non-zero elements of
                          % A by row
    for i = 1 : m
        nnzRow(i) = nnz( A(i,:) );
    end
    [~, idxRow] = sort(nnzRow); % Sort indices according to
                          % the number of non-zero elements
    % ---------------------------------------------------
    % First, every node is marked with a negative number,
    % in order to indicate that the node is unmarked.
    marked = -1 * ones(1, m);
    sizeM  = 0;
    % Using the symmetry of A, we follow the columns of A:
    for j = idxRow
        c = A(:,j);  % Define the jth column of A
        I = find(c); % Find indices of non-zero elements
        if marked(j) == -1 % Unmarked node
            % Node j has not been marked yet. That's
            % why it will be marked as master and its
            % neighboring nodes will be slaves.
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.34-2.4

```
                % ---------------------------------------------
                % Mark its neighbors as slaves:
                marked(I) = 0;
                % Mark as master:
                sizeM = sizeM + 1; % A master node was found
                marked(j) = sizeM; % New numbering for node j
            end
        end
    end
    % -------------------------------------------------------
    % Forward Gauss-Seidel iteration.
    % -------------------------------------------------------
    %   b: is the right-hand side vector.
    %   x: is the initial vector.
    %   l: is the corresponding level.
    % -------------------------------------------------------
    function x = forwardGaussSeidel(obj, b, x, l)
        numSteps = obj.steps(l);
        for k = 1 : numSteps
            x = obj.MFGS{l} * ( b - obj.MU{l}*x );
        end
    end
    % -------------------------------------------------------
    % Backward Gauss-Seidel iteration.
    % -------------------------------------------------------
    %   b: is the right-hand side vector.
    %   x: is the initial vector.
    %   l: is the corresponding level.
    % -------------------------------------------------------
    function x = backwardGaussSeidel(obj, b, x, l)
        numSteps = obj.steps(l);
        for k = 1 : numSteps
            x = obj.MBGS{l} * ( b - obj.ML{l}*x );
        end
    end
    end % End Private Methods
end
% ---------------------------------------------------------
%                         END OF THE FILE
% ---------------------------------------------------------
```

UNICIENCIA Vol. 34, N°. 2, pp. 55-81. Julio-Diciembre, 2020 • URL: www.revistas.una.ac.cr/uniciencia • Correo electrónico: revistauniciencia@una.cr