Iterative Methods for Linear Systems of Equations

Basic Iterative Methods

ITMAN PhD-course, DTU, 20-10-08 till 24-10-08

Martin van Gijzen

Program day 1

- Overview of the course
- Useful references
- Introduction to iterative methods
- Some general concepts
- Basic iterative methods
- One-step projection methods

Goals of the course

- Explain basic ideas;
- Show how these ideas translate into algorithms;
- Give overview of modern iterative methods.

At the end of the course you will

- be able to understand the differences and similarities between the many iterative methods;
- and you will be able to select a suitable method for your problem.

Daily schedule (tentative)

9.00 - 11.00 Lecture, new theory

11.00 - 12.00 Theoretical exercises

12.00 - 13.30 Lunch

13.30 - 14.00 Introduction to an application

14.00 - 16.00 MATLAB-assignment

Topics per day

- Day 1: Basic iterative methods
- Day 2: Projection methods (1)

  Krylov methods for symmetric systems: CG, MINRES, methods for the normal equations
- Day 3: Projection methods (2)

  Krylov methods for nonsymmetric systems: GMRES-type methods
- Day 4: Projection methods (3)

  Krylov methods for nonsymmetric systems: BiCG-type methods
- Day 5: Preconditioning techniques

Recommended literature

- Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 3rd ed. The Johns Hopkins University Press, Baltimore, 1996.
- Richard Barrett, Michael Berry, Tony Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roidan Pozo, Charles Romine, Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* 2nd edition, SIAM, 1994.
- Henk van der Vorst, *Iterative Krylov methods for large linear systems.* Cambridge press, 2003
- Yousef Saad, *Iterative Methods for Sparse Linear Systems.* 2nd ed. SIAM, 2003

Useful webpages

- http://www.netlib.org/: a wealth of information on a.o. numerical software, e.g.
  - LAPACK, BLAS: dense linear algebra
  - NETSOLVE: grid computing
  - TEMPLATES: the book plus software
  - MATRIX MARKET: matrices
- http://www.math.uu.nl/people/vorst/: manuscript of the book, software
- http://www-users.cs.umn.edu/ saad/: first edition of the book, software
- http://www2.imm.dtu.dk/ pch/: interesting links, software

Introduction to iterative methods

Many applications give rise to (non)linear systems of equations. Typically the system matrices are

- Large, 10^8 unknowns are not exceptional anymore;
- Sparse, only a fraction of the entries of the matrix is nonzero;
- Structured, the matrix often has a symmetric pattern and is banded.

Moreover, the matrix can have special numerical properties, e.g it may be symmetric, Toeplitz, or the eigenvalues may all be in the right-half plane.

An application: ocean circulation (1)

Physical model: balance between

- Wind force
- Coriolis force
- Bottom friction.

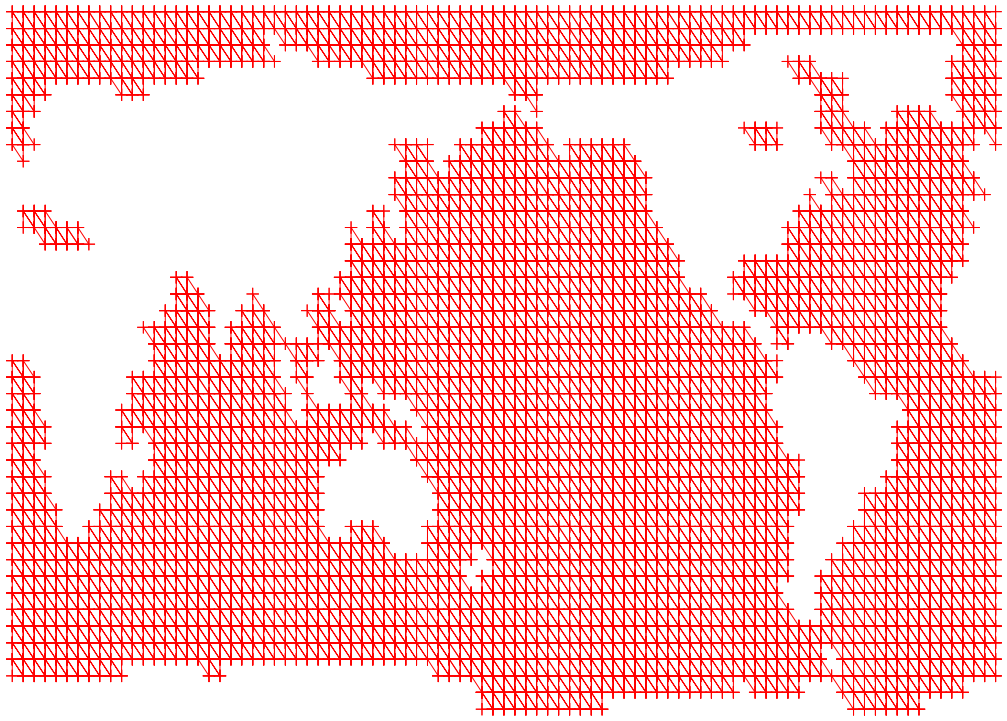
An application: ocean circulation (2)

Mathematical model

$$r\nabla^2\psi + \beta\frac{\partial\psi}{\partial x} = \nabla \times \mathbf{F} \quad \text{in } \Omega,$$

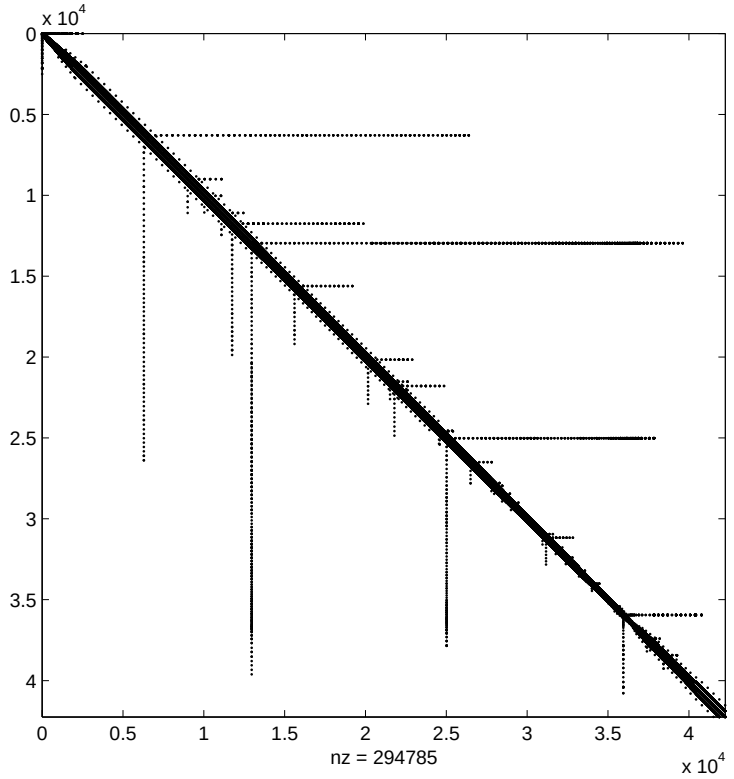
- ψ: streamfunction
- r: bottom friction parameter
- β: Coriolis parameter
- $\mathbf{F}$: Wind stress

An application: ocean circulation (3)



Numerical model: discretisation with FEM

An application: ocean circulation (4)



The nonzero pattern of the resulting matrix A

Why iterative methods?

Direct methods: $A = LU$.

Due to fill-in L and U can not be stored for realistic problems.

This is in particular true for three dimensional (3D) problems.

Model problem	Direct	Iterative
Computational costs	$O(n^p)$ $p \approx 2.0$ for 2D $p \approx 2.3$ for 3D	$O(n^p)$ $p \approx 1.4$ for 2D $p \approx 1.2$ for 3D
Memory requirements	$O(n^p)$ $p \approx 1.5$ for 2D $p \approx 1.7$ for 3D	$O(n)$

Iterative versus direct

For many problems, it is not clear which method is best:

- Direct methods are robust
- $O(n^p)$ can have a very large constant for iterative methods
- Often a combination can be used, e.g. in domain decomposition or iterative refinement

Does the best iterative method exist?

If no further restrictions: no

For certain classes: yes

Success of any iterative method depends on:

- Properties of A
  (Size, sparsity structure, symmetry, ...)
- Computer architecture
  (scalar, parallel, memory organisation, ...)
- Memory space available
- Accuracy required

Some general concepts

Iterative methods construct successive approximations x_k to the solution of the linear systems $Ax = b$. Here k is the iteration number, and the approximation x_k is also called the *iterate*. The vector $r_k = b - Ax_k$ is the *residual*.

The iterative methods are composed of only a few different basic operations:

- Products with the matrix A
- Vector operations (updates and inner product operations)
- *Preconditioning operations*

Preconditioning

Usually iterative methods are applied not to the original system

$$Ax = b$$

but to the preconditioned system

$$M^{-1}Ax = M^{-1}b$$

where the preconditioner is chosen such that:

- Preconditioning operations (operations with M^{-1}) are cheap;
- The iterative method converges much faster for the preconditioned system.

Preconditioners will be discussed in more detail on day 5.

Basic iterative methods

The first iterative methods we will discuss are the *basic iterative methods*. Basic iterative methods only use information of the previous iteration.

Until the 70's they were quite popular. Some are still used but as preconditioners in combination with an acceleration technique. They also still play a role in multigrid techniques where they are used as smoothers. Multigrid methods are discussed on day 5 of this course.

Basic iterative methods (2)

Basic iterative methods are usually constructed using a splitting of A:

$$A = M - R.$$

Successive approximations are then computed using the iterative process

$$Mx_{k+1} = Rx_k + b$$

which is equivalent too

$$x_{k+1} = x_k + M^{-1}(b - Ax_k)$$

The vector $r_k = b - Ax_k$ is called the *residual*, and the matrix M is a *preconditioner*. The next few slides we look at $M = I$.

Richardson's method

The choice $M = I$, $R = I - A$ gives Richardson's method, which is the most simple iterative method possible.

The iterative process becomes

$$x_{k+1} = x_k + (b - Ax_k) = b + (I - A)x_k$$

Richardson's method (2)

This process yields the following iterates:

Initial guess $x_0 = 0$

$$x_1 = b$$

$$x_2 = b + (I - A)x^{(1)} = b + (I - A)b$$

$$x_3 = b + (I - A)x^{(2)} = b + (I - A)b + (I - A)^2 b$$

Repeating this gives

$$x_{k+1} = \sum_{i=0}^{k} (I - A)^i b$$

Richardson's method (3)

So Richardson's method generates the series expansion for $\frac{1}{1-z}$ with $z = I - A$. If this series converges we have

$$\sum_{i=0}^{\infty} (I - A)^i = A^{-1}$$

The series expansion for $\frac{1}{1-z}$ converges if $|z| < 1$. If A is diagonalizable then the series $\sum_{i=0}^{\infty} (I - A)^i$ converges if

$$|1 - \lambda| < 1$$

with λ any eigenvalue of A. For λ real this means that

$$0 < \lambda < 2$$

Richardson's method (4)

In order to increase the radius of convergence and to speed up the convergence, one can introduce a parameter α:

$$x_{k+1} = x_k + \alpha(b - Ax_k) = \alpha b + (I - \alpha A)x_k$$

It is easy to verify that the optimal α which gives fastest convergence is given by

$$\alpha_{opt} = \frac{2}{\lambda_{max} + \lambda_{min}}.$$

Richardson's method (5)

Before, we assumed for the initial guess $x_0 = 0$.
Starting with another initial guess x_0 only
means that we have to solve a shifted system

$$A(y + x_0) = b \Leftrightarrow Ay = b - Ax_0 = r_0$$

So the results obtained before remain valid, irrespective of the initial guess.

Richardson's method (6)

We want to stop once the error $\|x_k - x\| < \epsilon$, with ϵ some prescribed tolerance. Unfortunately we do not know x, so this criterion does not work in practice.

Alternatives are:

- $\|r_k\| = \|b - Ax_k\| = \|Ax - Ax_k\| < \epsilon$
  Disadvantage: criterion not scaling invariant
- $\frac{\|r_k\|}{\|r_0\|} < \epsilon$
  Disadvantage: good initial guess does not reduce the number of iterations
- $\frac{\|r_k\|}{\|b\|} < \epsilon$
  Seems best

Convergence theory

To investigate the convergence of Basic Iterative Methods in general, we look again at the formula

$$Mx_{k+1} = Rx_k + b.$$

Remember that $A = M - R$. If we subtract $Mx = Rx + b$ from this equation we get a recursion for the error $e = x_k - x$:

$$Me_{k+1} = Re_k$$

Convergence theory (2)

We can also write this as

$$e_{k+1} = M^{-1}Re_k$$

This is a power iteration and hence the error will ultimately point in the direction of the largest eigenvector of $M^{-1}R$. The rate of convergence is determined by the *spectral radius* $\rho(M^{-1}R)$ of $M^{-1}R$:

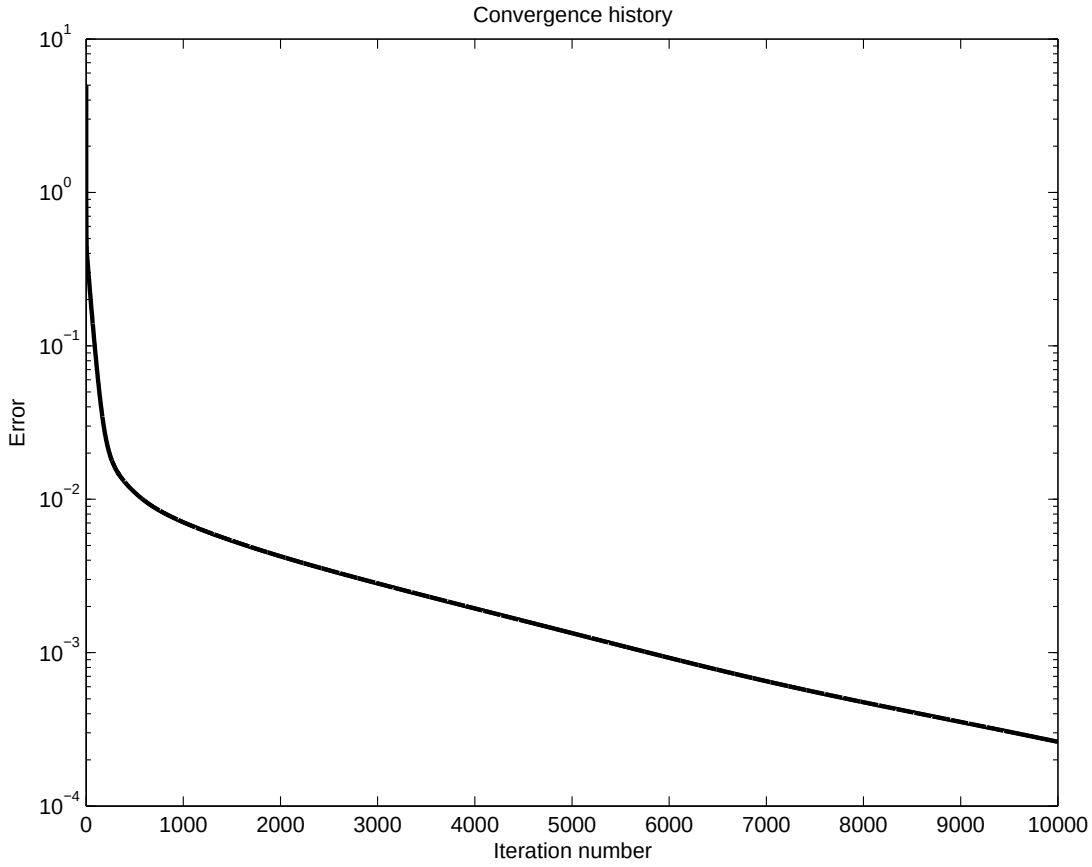
$$\rho(M^{-1}R) = |\lambda_{max}(M^{-1}R)| \ .$$

For convergence we must have that

$$\rho(M^{-1}R) < 1 \ .$$

Linear convergence

Ultimately, we have $\|e_{k+1}\| \approx \rho(M^{-1}R)\|e_k\|$, which means that we have linear convergence



Classical Basic Iterative Methods

We will now briefly discuss the three best known basic iterative methods

- Jacobi's method
- The method of Gauss-Seidel
- Successive overrelaxation

These methods can be seen as Richardson's method applied to the preconditioned system

$$M^{-1}Ax = M^{-1}b\ .$$

Jacobi's method

We first write $A = L + D + U$, with L the strictly lower triangular part of A, D the main diagonal and U the strictly upper triangular part. Jacobi's method is now defined by the choice $M = D$, $R = -L - U$. The process is given by

$$Dx_{k+1} = (-L - U)x_k + b$$

or equivalently by

$$x_{k+1} = x_k + D^{-1}(b - Ax_k)$$

The Gauss-Seidel method

We write again $A = L + D + U$. The Gauss-Seidel method is now defined by the choice $M = L + D$, $R = -U$. The process is given by

$$(L + D)x_{k+1} = -Ux_k + b$$

or equivalently by

$$x_{k+1} = x_k + (L + D)^{-1}(b - Ax_k)$$

Successive overrelaxation (SOR)

We write again $A = L + D + U$. The SOR method is now defined by the choice $M = D + \omega L$, $R = (1 - \omega)D - \omega U$. The parameter ω is called the relaxation parameter. The process is given by

$$(D + \omega L)x_{k+1} = ((1 - \omega)D - \omega U)x_k + \omega b$$

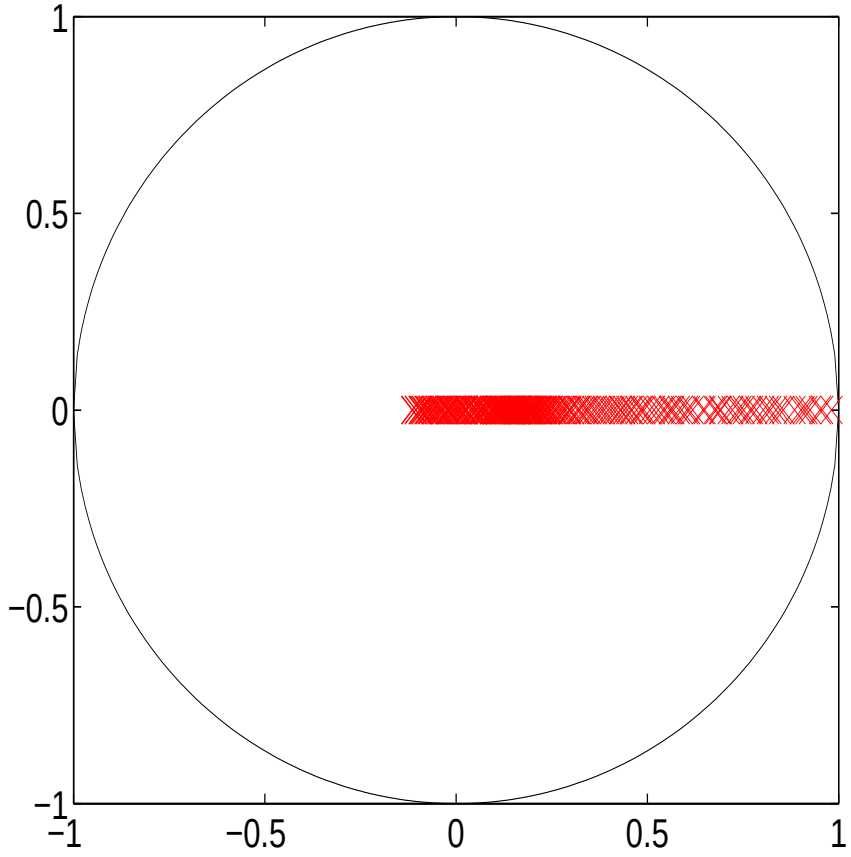
or as

$$x_{k+1} = x_k + \omega(D + \omega L)^{-1}(b - Ax_k)$$

With $\omega = 1$ we get the method of Gauss-Seidel back. In practice the optimal value of ω is not known.

Convergence

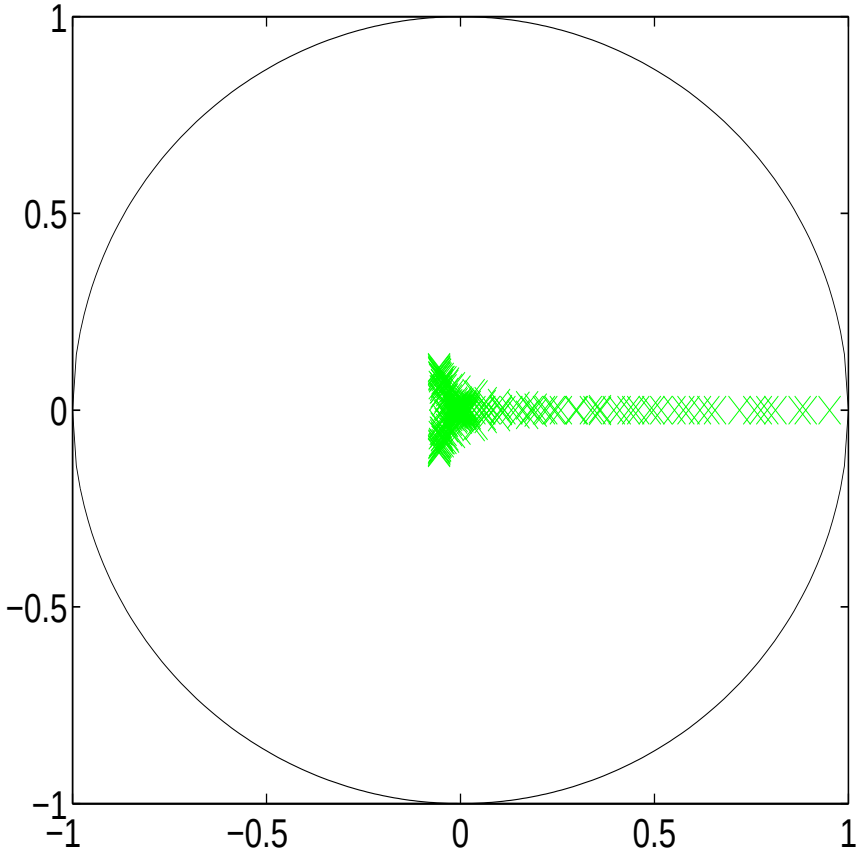
Convergence depends on spectrum of $M^{-1}R = I - M^{-1}A$



Jacobi Iteration: $e_k = (I - D^{-1}A)^k e_0$

Convergence

Convergence depends on spectrum of $M^{-1}R = I - M^{-1}A$



Gauss-Seidel Iteration: $e_k = (I - (L + D)^{-1}A)^k e_0$

One-step projection methods

The convergence of Richardson's method is not guaranteed and if the method converges, convergence is often very slow.

We now introduce two methods that are guaranteed to converge for wide classes of matrices. The two methods take special linear combinations of the vectors r_k and Ar_k to construct a new iterate x_{k+1} that satisfies a local optimality property.

Steepest descent

Let A be symmetric positive definite. Define the function

$$f(x_k) = \|x_k - x\|_A^2 = (x_k - x)^T A(x_k - x)$$

Let $x_{k+1} = x_k + \alpha_k r_k$ Then the choice

$$\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}$$

minimizes $f(x_{k+1})$.

Minimal residual

Let A be general square. Define the function

$$g(x_k) = \|b - Ax_k\|_2^2 = r_k^T r_k$$

Let $x_{k+1} = x_k + \alpha_k r_k$ Then the choice

$$\alpha_k = \frac{r_k^T A r_k}{r_k^T A^T A r_k}$$

minimizes $g(x_{k+1})$.

Orthogonality properties

The optimality properties of the steepest descent method and the minimal residual method are equivalent with the following orthogonality properties:

For steepest descent

$$\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k} \Rightarrow r_{k+1} \perp r_k$$

For the minimal residual method

$$\alpha_k = \frac{r_k^T A r_k}{r_k^T A^T A r_k} \Rightarrow r_{k+1} \perp A r_k \ .$$

Summary

Today we introduced Richardson's method and three preconditioned variants: the methods of Jacobi and Gauss-Seidel and the Successive Overrelaxation Technique. We saw that the convergence of Richardson's method can be related to the convergence of the series $\frac{1}{1-z} = 1 + z + z^2 \cdots$. If the method converges, convergence is linear, and often slow.

The question is whether there are better, faster ways to approximate A^{-1} using a polynomial expansion.
Tomorrow we will investigate how we can construct such methods.