

[Source](#) | [Model](#) | [Option](#)
[Model_Option](#) | [Help on fd methods](#) | [Archived Tests](#)

fd_psor

Input parameters:

- SpaceStepNumber N
- TimeStepNumber M
- Theta $\frac{1}{2} \leq \theta \leq 1$
- Omega $1 \leq \omega \leq 2$
- Epsilon

Output parameters:

- Price
- Delta

The PSOR(Projected Successive OverRelaxation) method has been introduced by Cryer in [\[1\]](#).

/*Memory Allocation*/

/*Time Step*/

Define the time step $k = \frac{T}{N}$.

/*Space localisation*/

Define the integration domain $D = [-l, l]$ using the probabilistic estimate [there](#).

/*Space Step*/

Define the space step $h = \frac{2l}{M}$.

/*Peclet Condition*/

If $|r - \delta|/\sigma^2$ is not small, then a more stable finite difference approximation is used. cf [there](#).

/*Lhs factor of theta scheme*/

Initialize the matrix M^h issued from the discretization of the operator A in the case of Dirichlet Boundary conditions. cf [there](#).

/*Rhs factor of theta scheme*/

Initialize the matrix N issued from the θ -scheme method in the cases of Dirichlet Boundary conditions. [there](#)

/*Terminal value*/

After a logarithmic transformation, put the value of the payoff into a vector P which will be used to save the option value.

/*Finite difference Cycle*/

At any time step, described by the loop in the variable i , we have to solve the linear complementarity problem cf. [there](#).

/*Init Rhs*/

Compute NP and save the result in the vector Rhs .

/*PSOR cycle*/

We solve the linear complementarity problem using the PSOR method, cf. [there](#), which consists in constructing a convergent sequence z^p whose limit is z .

Variable *loops* stands for the exponent p .cf [there](#).

Step 0 choose a relaxation parameter *omega* and a precision *epsilon*.

Step 1 compute the vector z^p using the variable y and save it in the vector P . Fill the variable *error* with the difference $|z^{p+1} - z^p|$.

Step 3 indicates the end of the loop by stopping the algorithm when $error > epsilon$ or the number of iteration is too large.

/*Price*/

/*Delta*/

/*Memory Desallocation*/

References

- [1] C.W.CRYER. The solution of a quadratic programming problem using systematic overrelaxation. *SIAM J. Control*, 9:385–392, 1971. 1