(a)

First, assume the solution has the form:

The original equation then becomes:

where λ is an arbitrary constant. This gives us two ODE’s:

Given initial values , boundaries , we have

For a non-trivial solution, consider boundaries to be

Let us first look at the spatial problem

The solution is of the form:

which gives:

For a non-trivial solution, we have so

For the time problem

the solution is

which gives the solution to the original equation

as we let

(b)

Use Euler’s method (forward/explicit)

LHS:

(forward difference formula)

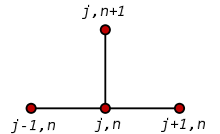
RHS:

(centered three-point formula)

Ignoring error terms and equating the two:

Simplifying:

Given initial values , boundaries , straightforward to solve for .



Computing using linear system:

Use Euler’s method (backward/implicit):

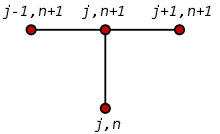
LHS:

(backward difference formula)

RHS:

(centered three-point formula)

Ignoring error terms and equating the two:



Given initial values , boundaries , construct linear system to solve for .

Simplify above equation:

Construct linear system accordingly, for each j:

where is number of partitions.

Initial values: when , . So we should loop starting from j=1. Boundary values: , so we always have .

Use Crank-Nicolson:

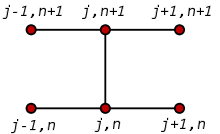
LHS:

(forward difference formula)

RHS:

(centered three-point formula applied twice)

Ignoring error terms and equating the two:

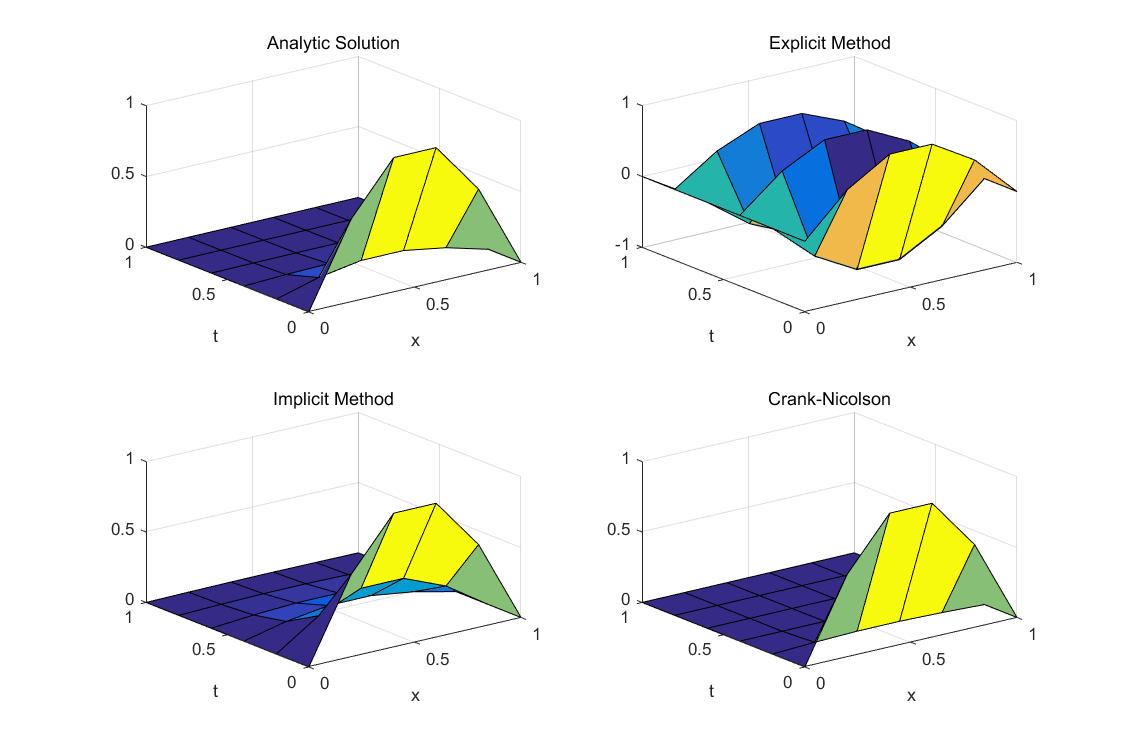


Simplifying:

Construct linear system accordingly, for each j:

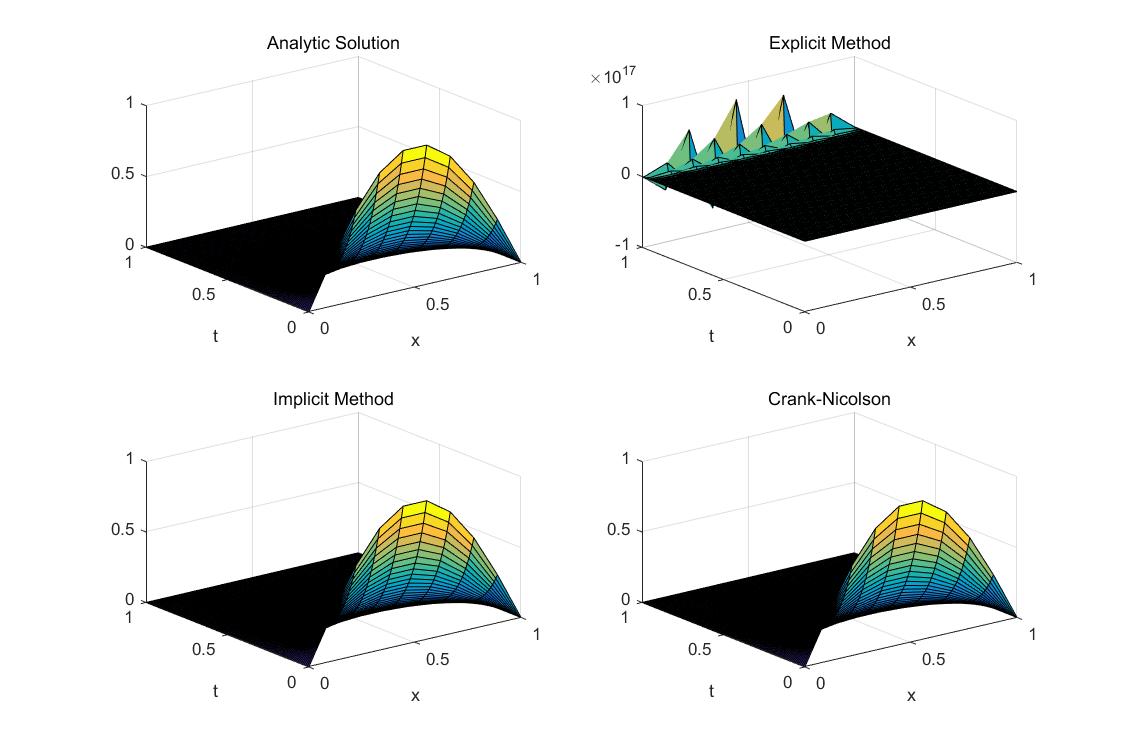
Results:

6 x nodes, 6 t nodes:



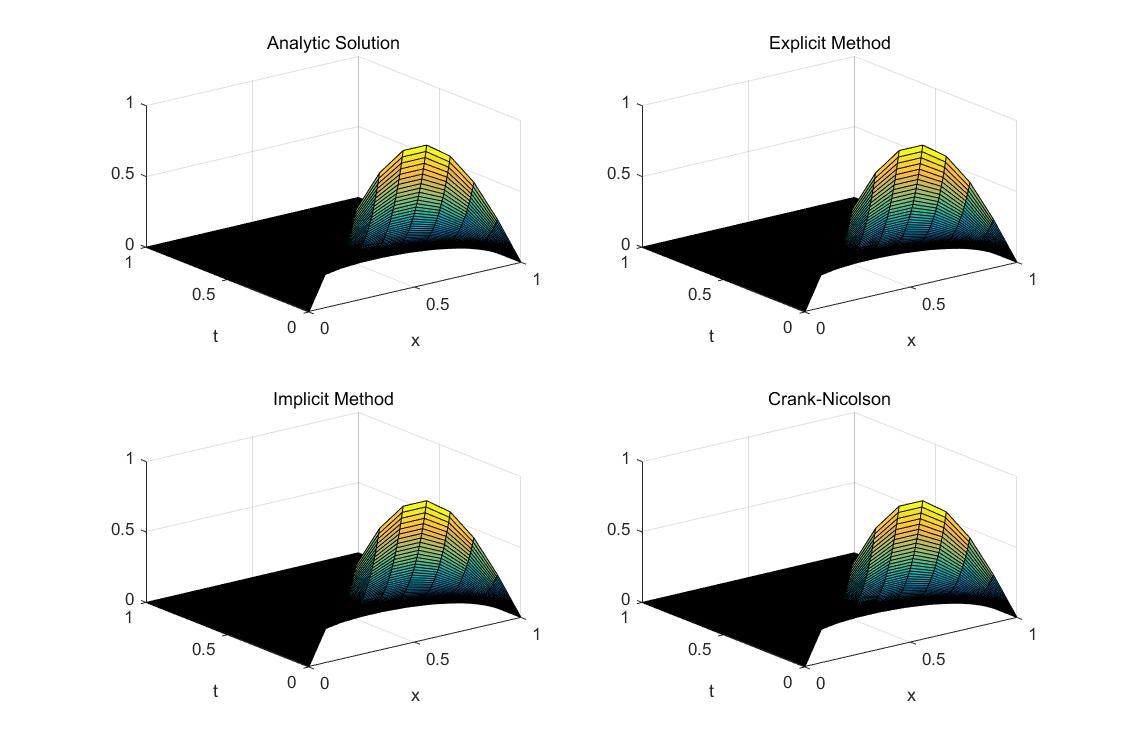
(note the divergence of Explicit Method)

10 x nodes, 100 t nodes:



(note overall improved smoothness and still divergence of Explicit Method.)

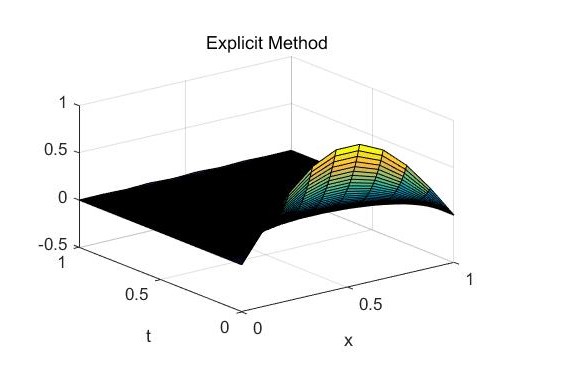
10 x nodes, 200 t nodes:



(note the convergence of Explicit Method)

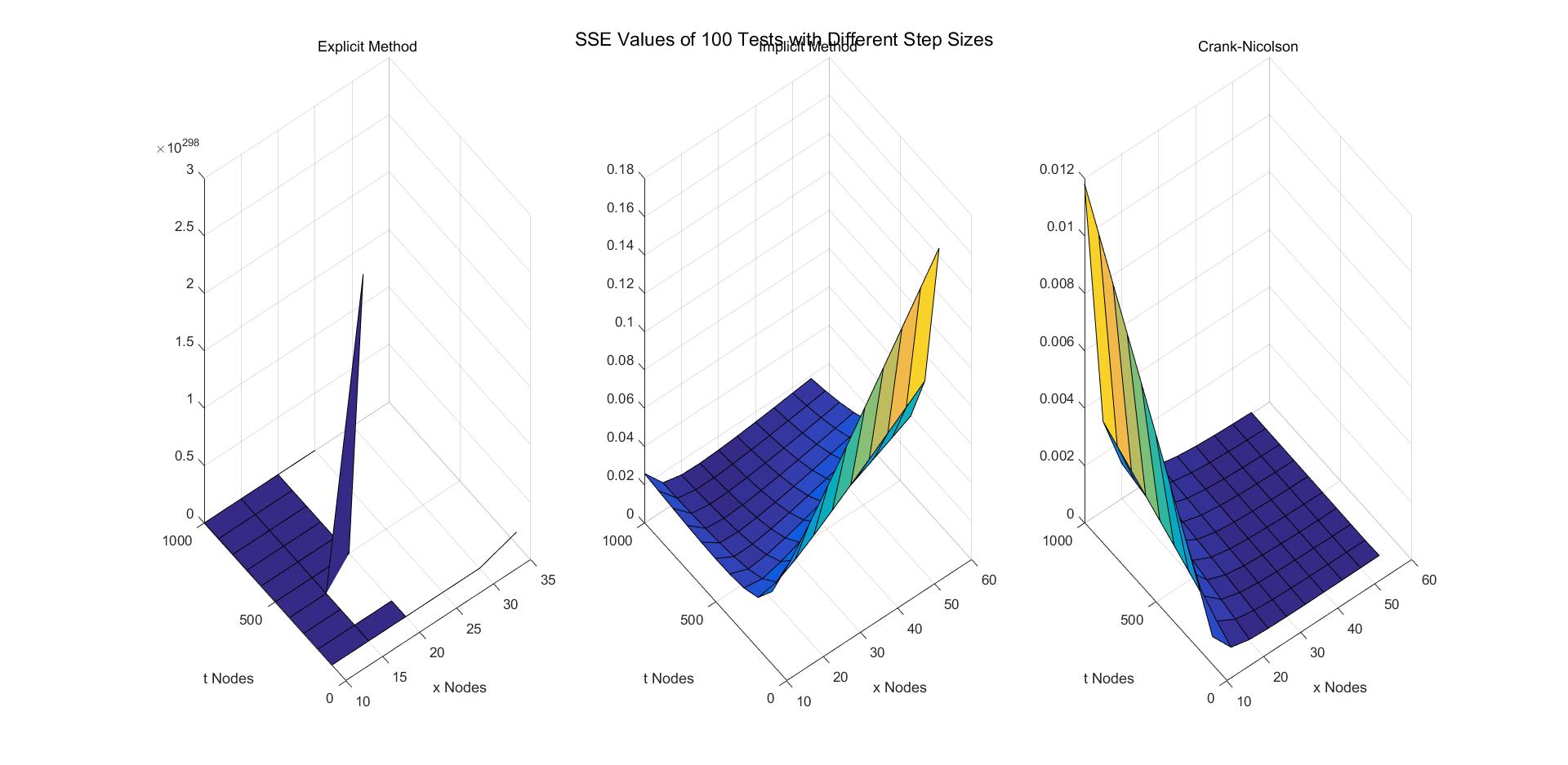
On convergence of Explicit Method: in this problem in particular, it is guaranteed that no point in the solution should have a value that is negative. Since ranges of and are equal, where denotes number of t nodes and denotes number of x nodes.

In this problem, however, the Explicit Method solution starts converging as soon as although producing a quite inaccurate solution, as in the case where 10 x nodes and 135 t nodes are used ():



Error Analysis:

We ran 100 tests on each method, where number of x nodes can be any of 10, 15...55, and number of t nodes can be any of 100, 200...1000. We calculate (sum of squared errors) using the analytic solution. The following graphs showing of all tests are obtained.



In this problem in particular, it is difficult to make Explicit Method converge while maintaining observation of effects of number of x nodes, so we will focus more on performance of Implicit Method and Crank-Nicolson Method.

Implicit Method generates significantly higher errors when number of x nodes increases while there are not many t nodes. We believe this is due to accumulation of round-off errors. Crank-Nicolson Method, on the other hand, is barely affected by round-off errors in this situation.

Implicit Method also generates slightly higher errors when there are many t nodes and not many x nodes. As number of t nodes decide number of iterations, initial errors due to small number of x nodes are magnified in these significant amount of iterations. On the other hand, Crank-Nicolson Method is significantly affected in like situation. Presumably such behavior can be attributed to same error magnification problem.

In general, Crank-Nicolson Method produces significantly smaller errors. In the tests conducted, for Implicit Method, while for Crank-Nicolson Method, the latter about times smaller than the former.

(c)

Initial condition: except boundaries. Boundaries: . ?

Use Euler’s method (forward/explicit)

LHS:

(forward difference formula)

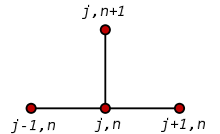
RHS:

(centered three-point formula)

Ignoring error terms and equating the two:

Simplifying:

Given except boundaries and boundaries , straightforward to solve for .



Computing using linear system:+

Use Euler’s method (backward/implicit):

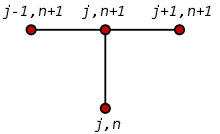
LHS:

(backward difference formula)

RHS:

(centered three-point formula)

Ignoring error terms and equating the two:



Simplifying:

Given except boundaries and boundaries , construct linear system, for each j for each k:

=

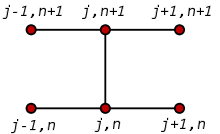
Use Crank-Nicolson:

LHS:

(forward difference formula)

RHS：

Ignoring error terms and equating the two:



Simplifying:

Construct linear system accordingly, for each j for each k: