برای حل این سوال و پیاده‌سازی الگوریتم گاوس، از شبه کد ارائه شده در کتاب استفاده می‌کنیم با این تفاوت که به جای محور گیری جزئی، از محور گیری کلی استفاده می‌کنیم. دلیل این کار با توجه به اینکه هزینه محاسباتی آن بیشتر است اینست که مدیریت صفرهای ماتریس را آسان‌تر می‌کند و خطاهای برنامه را کاهش می‌دهد.

```python
def better_forward_eliminate(matrix):
    n = len(matrix)
    for i in range(n):
        full_pivot(matrix, i)
        for j in range(i + 1, n):
            if matrix[i][i] == 0:
                return
            co_effiecent = matrix[j][i] / matrix[i][i]
            for k in range(i, n + 1):
                matrix[j][k] -= co_effiecent * matrix[i][k]
```

در ادامه پیاده‌سازی محورگیری کلی آمده‌است که با پیدا کردن بزرگترین درایه زیرماتریس، سطر و ستون‌ها را جابجا می‌کند

```python
def full_pivot(matrix, k):
    n = len(matrix)
    max_position = (k, k)
    max_value = matrix[k][k]
    for i in range(k, n):
        for j in range(k, n):
            if abs(matrix[i][j]) > max_value:
                max_position = (i, j)
                max_value = abs(matrix[i][j])
    row, column = max_position
    for p in range(k, n + 1):
        matrix[k][p], matrix[row][p] = matrix[row][p], matrix[k][p]
    for p in range(k, n):
        matrix[p][k], matrix[p][column] = matrix[p][column], matrix[p][k]
```

در انتها با جایگذاری‌های پسرو جواب یا جواب‌های دستگاه معادلات را پیدا می‌کنیم. ذکر این نکته ضروری است که برای فهمیدن اینکه دستگاه جواب ندارد یا بی‌نهایت جواب دارد، کافی است دترمینان ماتریس ضرایب را بررسی کنیم، این کار را پس از اجرای الگوریتم گاوس و بررسی سطرهایی که همه‌ی درایه‌های آنها صفر باشد انجام می‌دهیم. اگر همه‌ی درایه‌های یک سطر صفر و درایه آخر ماتریس افزوده در همان سطر نیز صفر باشد بی نهایت جواب داریم و اگر صفر نباشد جوابی نداریم زیرا این بدین معنی است که صفر برابر با عددی دیگر است که تناقض است. در الگوریتم جایگذاری‌های پسرو ابتدا وجود تناقض را بررسی و سپس شرط بی‌نهایت بودن جواب‌ها را بررسی می‌کنیم:

```python
def backward_substitute(matrix):
    n = len(matrix)
    infinit_solutions = False
    for i in range(n):
        if all(matrix[i][j] == 0 for j in range(0, n - 1)):
            if matrix[i][n] == 0:
                return "No Solution"
            else:
                infinit_solutions = True
    if infinit_solutions:
        return "Infinite Solutions"
    x = [0] * len(matrix)
    for i in range(n - 1, -1, -1):
        m = 0
        for j in range(i + 1, n):
            m += matrix[i][j] * x[j]
        x[i] = (matrix[i][n] - m) / matrix[i][i]
    return x
```