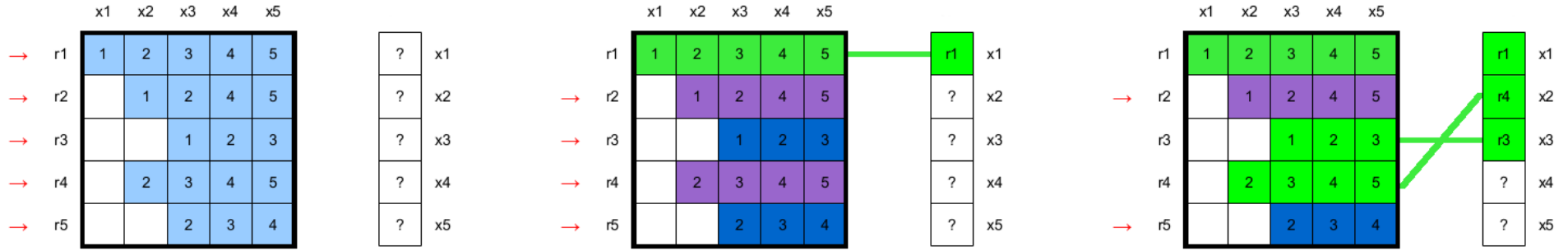
Informacje ogólne

	x1	x2	x3	x4	x5
r1	1	2	3	4	5
r2		1	2	3	4
r3			1	2	3
r4		2	3	4	5
r5			2	3	4

?	x1
?	x2
?	x3
?	x4
?	x5

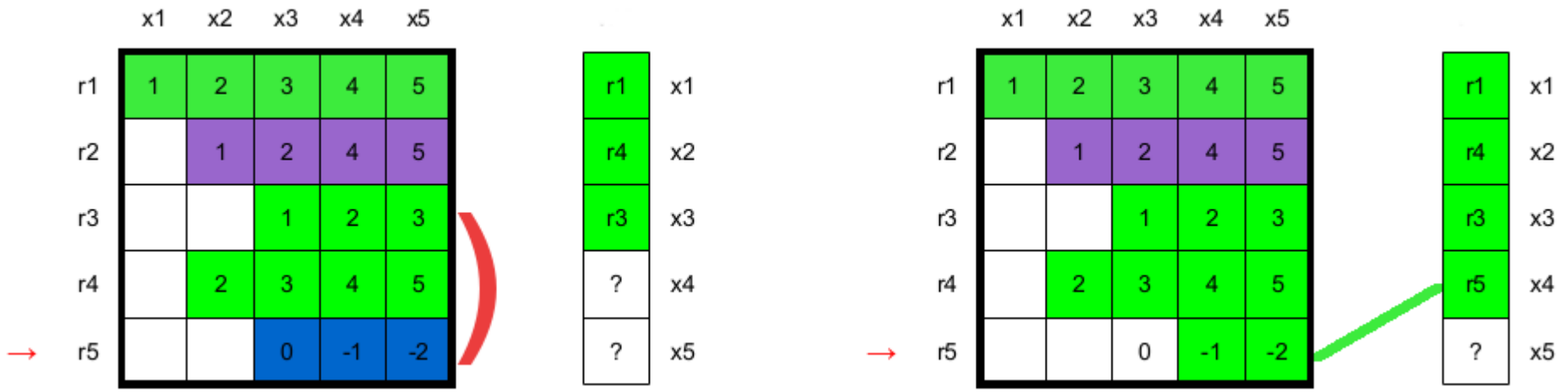
- Solwer działa na zasadzie black box – spodziewa się tylko macierzy kwadratowej i wektora prawej strony; nie wymaga informacji o rozwiązywanym problemie.
- Jego jedynym wyjściem jest wektor zawierający rozwiązanie lub – w przypadku układów sprzecznych – wyjątek.

Faza forward substitution



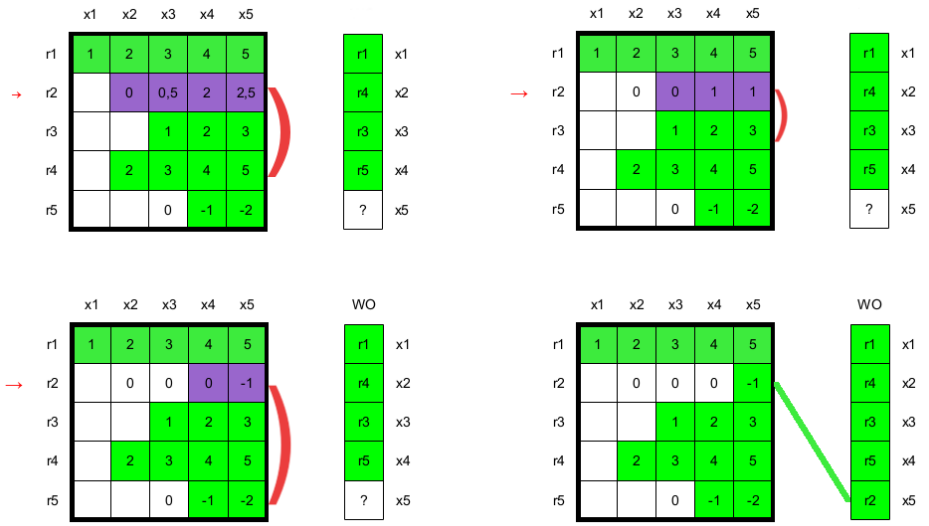
- W wersjii równoległej dla CPU każdy wiersz macierzy byłby obsługiwany przez osobny wątek; w prototypie sekwencyjnym to zachowanie jest symulowane przez pętlę by a `for`.
- Dla każdego wiersza ustalane jest którą funkcję (x_n) opisuje.
  - W powyższym przykładzie, wiersz 1 opisuje x_1, wiersz 3 opisuje x_3 zaś wiersz 4 - x_2.
  - Inne funkcje na razie pozostają oznaczone jako „niezbadane”.
- Te informacje zapisywane są w osobnym wektorze, służącym za mapę funkcji na wiersz w macierzy.

Faza forward substitution



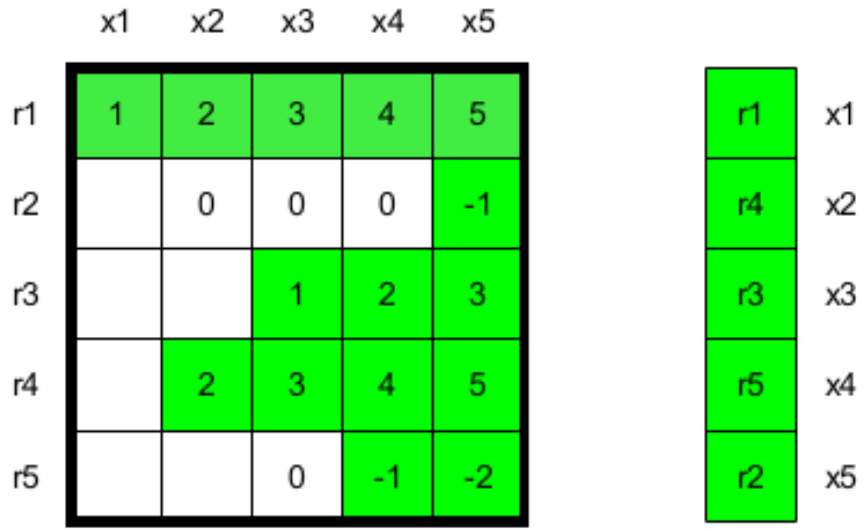
- Tak jak w klasycznej eliminacji Gaussa, dwa wiersze nie mogą opisywać tej samej funkcji.
- Jeśli wątek znajdzie już funkcję którą opisuje jego wiersz w mapie, zredukuje swój „skonfliktowany” wiersz dodając do niego oryginalny wiersz razy odpowiedni mnożnik.
- W powyższym przykładzie, wiersz 5 jest redukowany za pomocą wiersza 3 (ponieważ wiersz 3 jako pierwszy wpisał się do mapy z x_3).

Faza forward substitution



- Dla wiersza 5 konieczna była tylko jedna runda redukcji.
- By zredukować wiersz 2 z x_2 do x_5 konieczne są trzy rundy.
- W prototypie to zachowanie osiągane jest cofnięciem pętli `for` o jeden po redukcji, w wątku można to osiągnąć pętlą `while`.
- O ile każdy wątek musi wykonać najwyżej *N-1* redukcji w macierzy *N x N*, uważam to za wąskie gardło – być może ten problem można rozwiązać reorderingiem macierzy.

Faza backward substitution



- Klasyczna eliminacja Gaussa generuje macierz schodkową.
- Moja metoda generuje macierz schodkową „zmapowaną”. Backward substitution można tu wykonać przechodząc bottom-up po mapie, zamiast bezpośrednio po macierzy.
- Tego etapu nie można łatwo zrównoleglić, dlatego najprawdopodobniej nawet w końcowej wersji solwera będzie on wykonywany sekwencyjnie.

Zrównoleglenie dla platformy GPU

• Z punktu widzenia równoległości na platformach GPU, konieczność blokowania części mapy funkcji na wiersze jest poważnym wąskim gardłem (ponieważ dotykanie globalnej pamięci urządzenia jest nieefektywne czasowo, oraz wątki musiałyby czekać istotne okresy czasu na lock).
• Obecnie badam rozwiązanie które wymaga reorderingu macierzy po czym pocięcia jej na niezależne fragmenty, każdy obrabiany przez osobny multiprocesor strumieniujący.
• W ten sposób blokowanie i redukcja (=odczyty) zachodzą tylko w pamięci przy multiprocesorze, tańszej niż pamięć urządzenia z punktu widzenia czasu dostępu.
• Kroki reorderingu i częściowych rozwiązań będą powtarzane do rozwiązania problemu; będzie to działać bardzo dobrze dla macierzy pasmowych, trochę gorzej dla blokowych i gęstych.
• Reordering macierzy bezpośrednio na GPU jest kosztowne czasowo, ale pozwala uniknąć ciągłych downloadów i uploadów przez PCI-Express, czyli najdroższych kroków w całym rozwiązaniu.