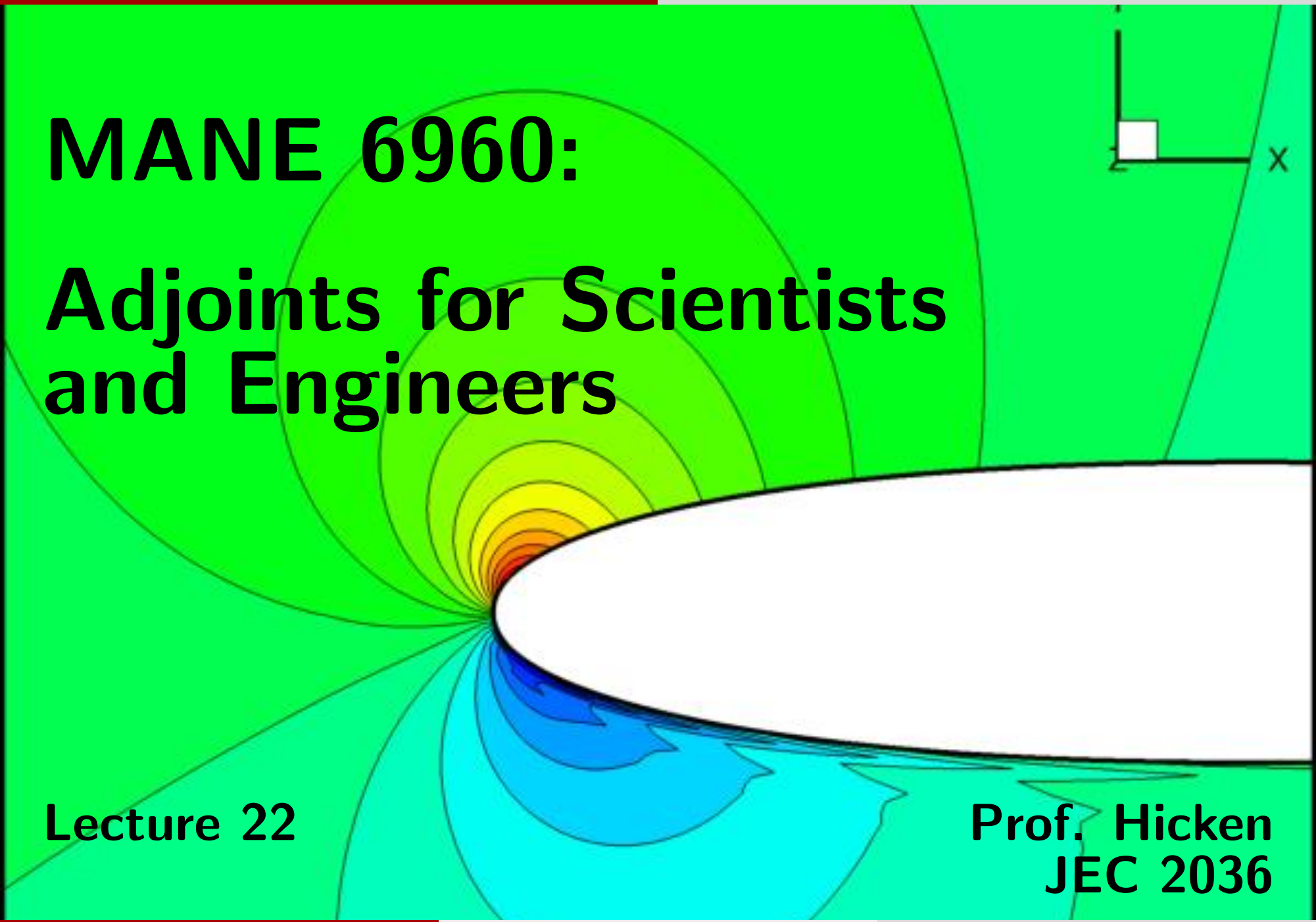
MANE 6960:

Adjoints for Scientists and Engineers



Lecture 22

Prof. Hicken
JEC 2036

Motivation

As we discussed at the the end of last lecture, a fundamental difference between the adjoints of linear and nonlinear IBVP is that the latter produce operators that depend on the state variable u_h.

Therefore, we need to store or somehow recover u_h to compute the adjoint of nonlinear IBVPs.

Motivation (cont.)

Let's consider two extremes in terms of how we could solve the adjoint of a nonlinear IBVP.

1. Store every u_h to (internal or external) memory.
2. Store only one u_h and recompute as necessary

To explore these extremes, let's consider a concrete example:

- A CFD simulation is run that uses u_h of dimension 10^5 (i.e. one-hundred thousand spatial DOF). Suppose the simulation is run for 10^4 time steps, and each time step takes 1 sec.

Motivation (cont.)

Using the first strategy, how much memory is needed? Assume one double precision number for each state variable in u_h.

$$\#bytes = 8 \times 10^5 \times 10^4 = 8 \times 10^9 \approx 7.45 \text{ GB}$$

Using the second strategy, how much time is needed? You can assume each adjoint solve takes the same time (1 sec.) as the forward solve.

$$\#\text{forward solves} = N + (N-1) + \cdots + 2 + 1 = \sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\text{Cost} = 2 \times \frac{(N+1)N}{2} = (10^4+1)(10^4)\,s = 3.17 \text{ years}$$

The above strategies represent extremes in a bi-objective optimization of memory and computational time. They motivate the need for a better way (i.e. compromise) to solve the adjoint of nonlinear IBVPs.

Multilevel Checkpointing

Notation and Assumptions

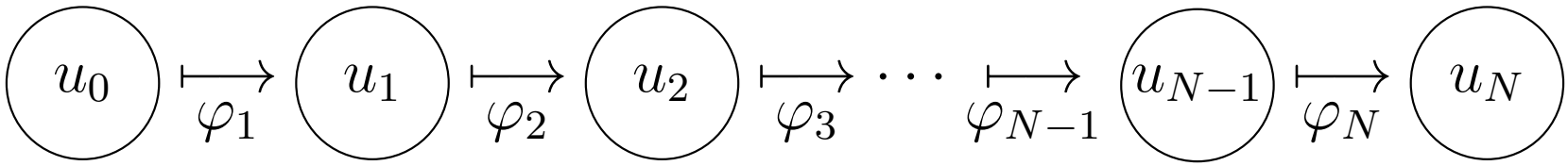
In order to describe and analyze different checkpointing schemes, we need some notation and assumptions.

A checkpoint is a state that is "saved" or stored in some way and that can be retrieved from a stack.

- In checkpointing algorithms for adjoints, one can show that the checkpoints can always be stored in a LIFO stack.

Notation and Assumptions (cont.)

We will consider the following abstract description of a state evolution, for which time-marching IBVPs is a concrete example:



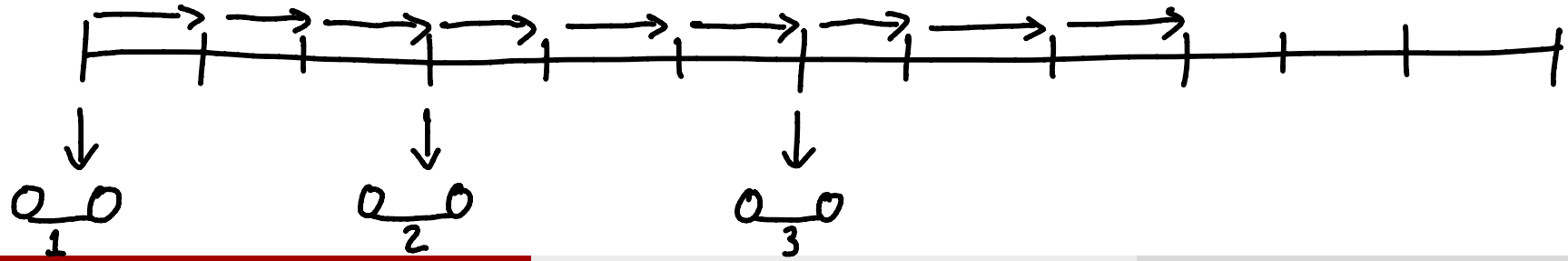
- φ_i represents the abstract mathematical mapping between u_{i-1} and u_i; e.g. one step in a Runge-Kutta method.
- u_i are large dimensional states; e.g. the spatial degrees of freedom. We will assume that we can store at most c of these states.

Multilevel Checkpointing Example

Multilevel checkpointing is a particular checkpointing schedule, that is, a scheme for deciding where checkpoints should be placed in a sequence.

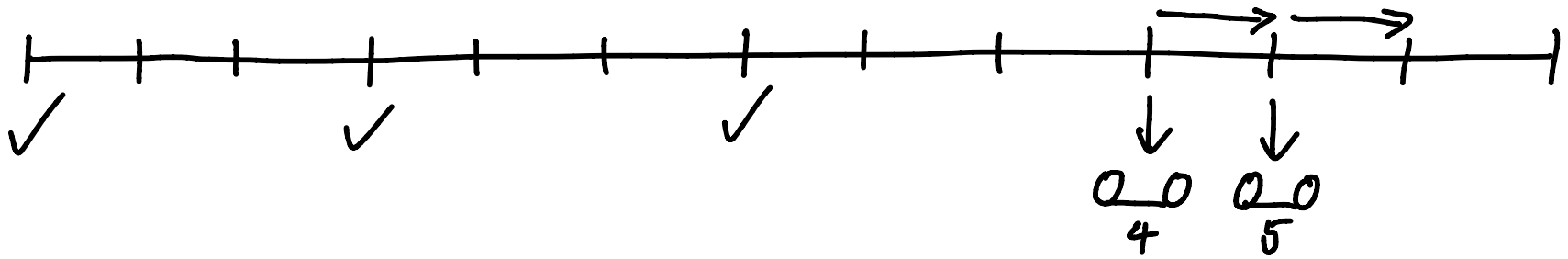
To explain multilevel checkpointing, we will consider a simple 2-level scheme applied to an evolutionary problem with $N = 12$ steps.

As we advance the forward problem (i.e. solving for the u_i), we distribute $c_1 = 3$ checkpoints over the sequence of states as shown below.

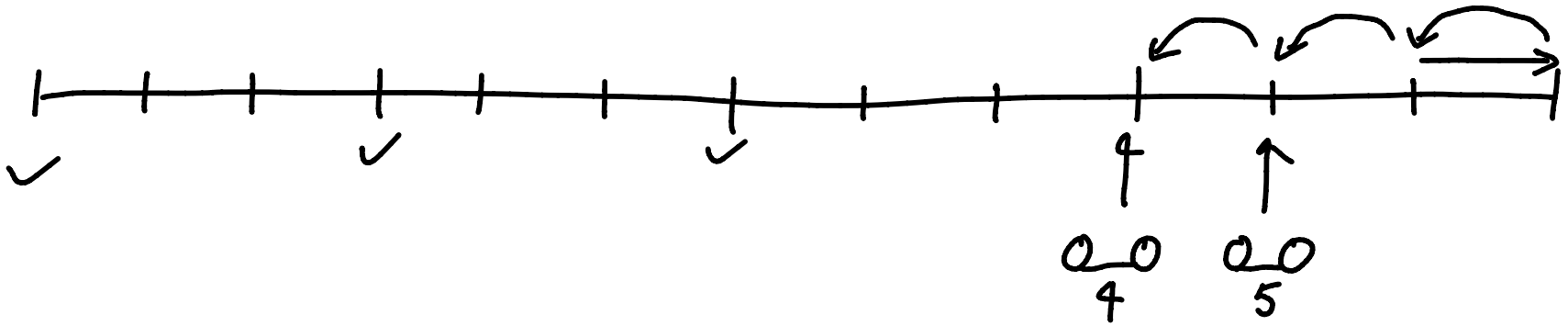


Multilevel Checkpointing Example (cont.)

Once we reach the last interval, we then distribute $c_2 = 2$ checkpoints over that interval.

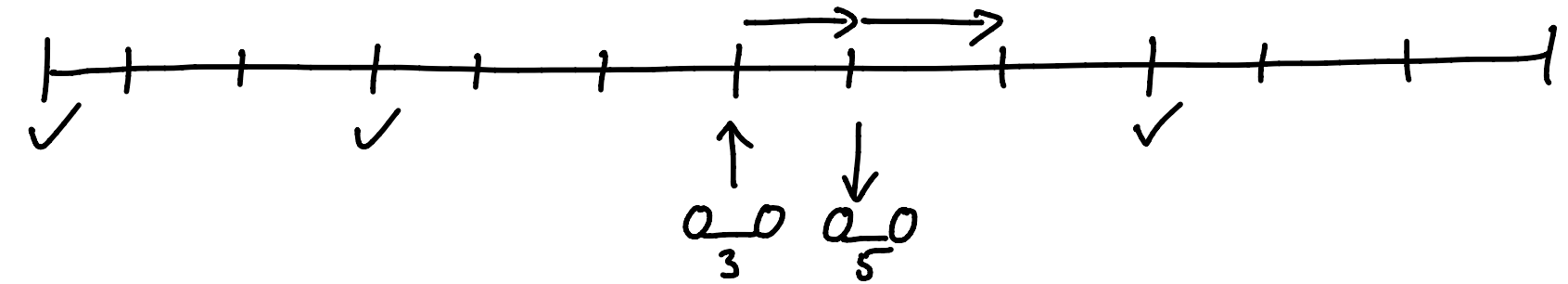


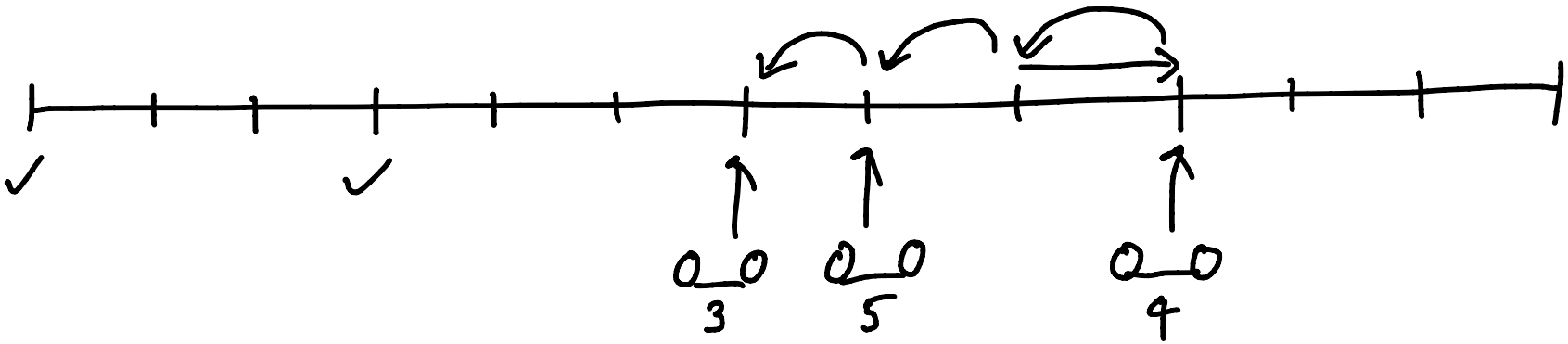
That brings us to the penultimate step, where we can advance to u_N and then solve 3 iterations of the adjoint IBVP:



Multilevel Checkpointing Example (cont.)

At this point, the second-level checkpoints are available, so we can return to the last first-level checkpoint inserted into the stack, and repeat the second-level process:





Multilevel Checkpointing in General

Let's generalize the scheme described in the previous example.

- Suppose there are r levels, and each level has c_i checkpoints assigned to it.

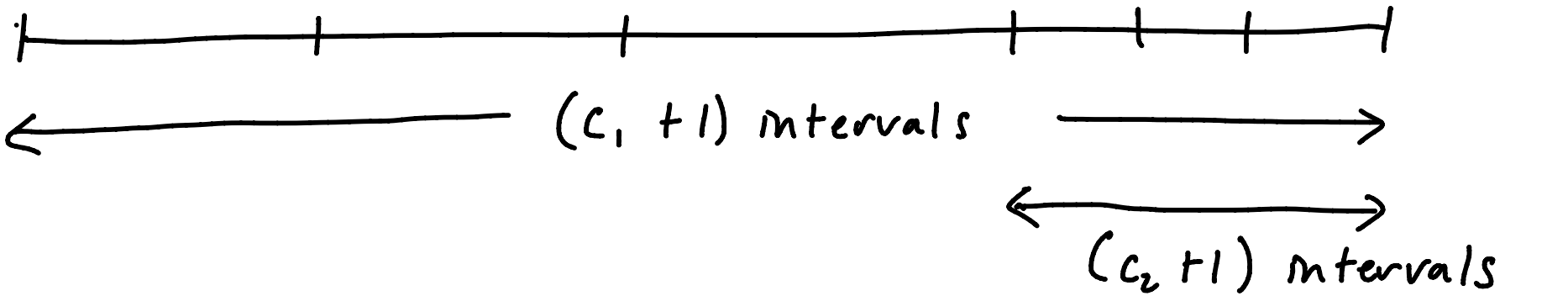
What is the total memory cost of the multilevel scheme, M_m, in terms of checkpoints?

$$M_m = c_1 + c_2 + \cdots + c_r$$

$$= \sum_{i=1}^{r} c_i \qquad (= c)$$

Multilevel Checkpointing in General (cont.)

What is the maximum number of time steps, N_m, possible?



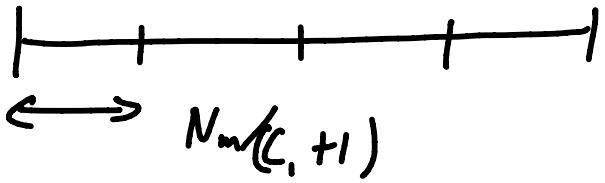
$$N_m = (c_1+1)(c_2+1)\cdots(c_r+1)$$

$$= \prod_{i=1}^{r} (c_i+1)$$

Multilevel Checkpointing in General (cont.)

Finally, what is the temporal complexity, T_m, in terms of the number of times the φ_i are called? (assume all φ_i have equal cost)

To get to second, φ_i must be called $\frac{c_1 N_m}{c_1 + 1}$ times



At the second level, the 1st-level groups are divided into $(c_2 + 1)$ groups. Thus, the # steps to get to third level is

$c_2 \frac{N_m}{(c_1+1)} \frac{1}{(c_2+1)}$ ← * $(c_1 + 1)$ times, since it happens on each 1st level group.

Multilevel Checkpointing in General (cont.)

So, $\frac{c_2 N_m}{(c_1+1)} \frac{(c_1+1)}{(c_2+1)} = \frac{c_2 N_m}{(c_2+1)}$ calls to φ_i due to second level

We end up with

$$T_m = \sum_{i=1}^{r} \frac{c_i N_m}{(c_i+1)} = \sum_{i=1}^{r} \frac{c_i N_m \prod_{j\neq i}^{r} (c_j+1)}{\prod_{j=1}^{r} (c_j+1)} \leftarrow N_m$$

$$= \sum_{i=1}^{r} c_i \prod_{j\neq i}^{r} (c_j+1) \pm \sum_{i=1}^{r} \prod_{j\neq i}^{r} (c_j+1)$$

$$= r N_m - \sum_{i=1}^{r} \prod_{j\neq i}^{r} (c_j+1)$$

- Thus, the computational cost is no worse than r times the number of time steps.

$r N_m$ − (+ve)

Multilevel Checkpointing in General (cont.)

Performance of Multilevel Checkpointing:

$$\text{Memory: } M_m = \sum_{i=1}^{r} c_i$$

$$\text{Time Steps: } N_m = \prod_{i=1}^{r} (c_i + 1)$$

$$\text{Comp. Cost: } T_m = \sum_{i=1}^{r} \frac{c_i N_m}{c_i + 1} = r N_m - \sum_{i=1}^{r} \prod_{j \neq i}^{r} (c_j + 1)$$

- Here and subsequently, the scalar r is the maximum number of times any φ_i is called.

Exercise

Consider a 4-level scheme with $c_i = 4$ checkpoints on each level.

1. What are the memory needs of this scheme?
2. What is the maximum number of adjoint steps this scheme can take?
3. What is the computational cost of this scheme?

1) $M_m = \sum_{i=1}^{4} 4 = 16$

2) $N_m = \prod_{i=1}^{4} (4+1) = (4+1)^4 = 625$

3) $T_m = \sum_{i=1}^{4} \frac{4\,(625)}{5} = 2000 < r\,N_m \quad (2500)$

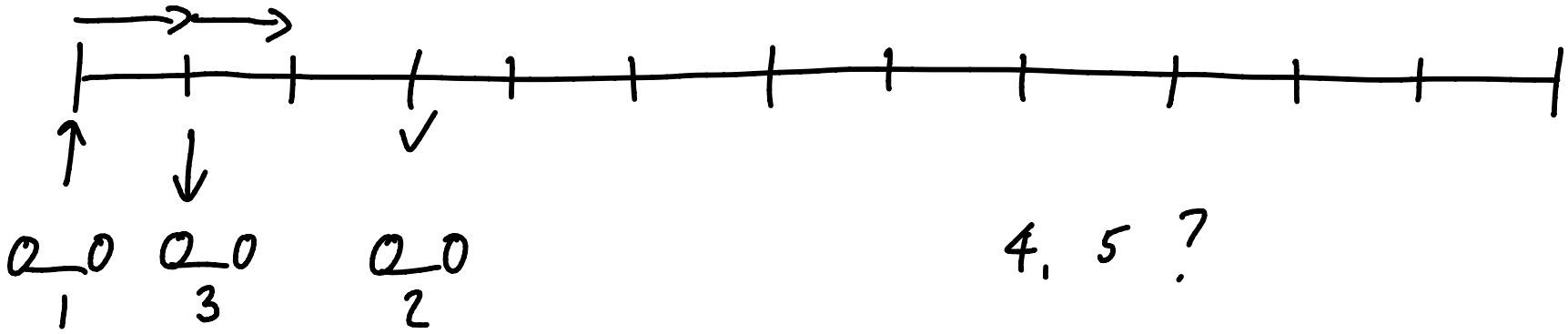
Binomial Checkpointing

A Drawback of Multilevel Checkpointing

One problem with multilevel checkpointing is that a checkpoint assigned to one level cannot be used on other levels.

For example, in the earlier example, consider the situation once we have returned to the first first-level interval (so the last first-level adjoint interval).

- We have two checkpoints available, but they cannot be used.



An Optimal Reversal Schedule

To develop a better (optimal) use of the checkpoints, consider an evolution sequence with N_b steps and suppose

1. we have memory for c checkpoints; and
2. we are limited to at most r evaluations of any φ_i.

We can nominally represent the number of steps as a function of c and r, that is $N_b = N_b(c, r)$.

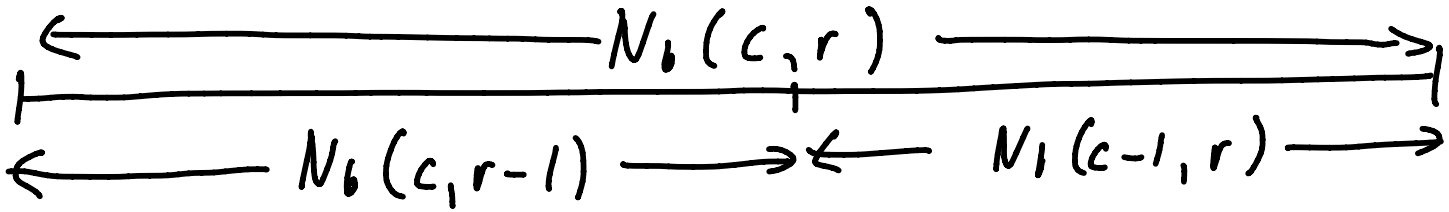
Consider breaking the optimal-length sequence $(0, N_b)$ into two subsequences. Because of the structure of the problem [GW09], the optimal checkpointing schedule can be applied recursively on both subsequences with the following modifications:

An Optimal Reversal Schedule (cont.)

- When we return to the first interval for the adjoint sweep, we have already called the φ_i once (to get to the second interval), so we must limit ourselves to $r-1$ calls using c checkpoints. Therefore, the length of this first sequence is $N_b(c, r-1)$.
- Once we have reached the second subsequence, we have already exhausted one checkpoint, so we are limited to $c-1$ checkpoints; thus, the length of the second sequence is $N_b(c-1, r)$.

Therefore,

$$N_b(c, r) = N_b(c, r-1) + N_b(c-1, r)$$



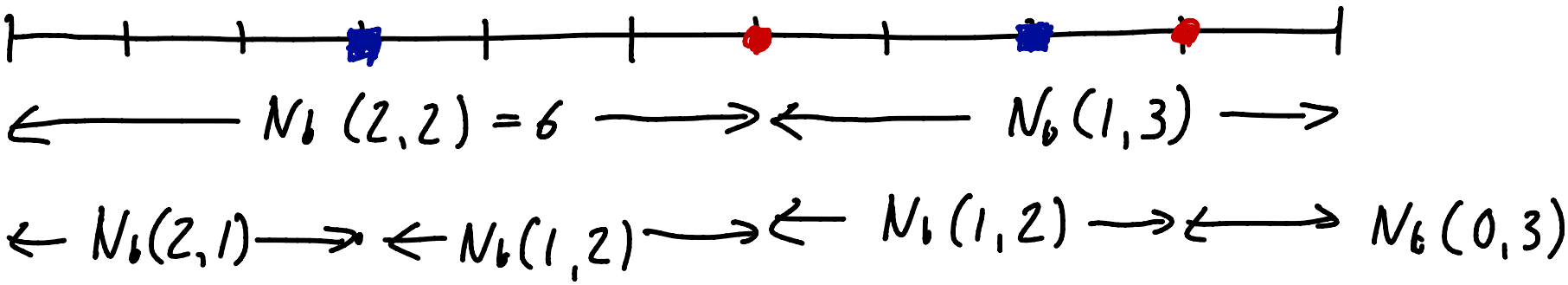
An Optimal Reversal Schedule (cont.)

Continuing this reasoning to its end (using induction) leads to the binomial checkpointing schedule [GW09], where the maximum number of steps possible, given r and c, is

$$N_b(c,r) = \binom{c+r}{c} = \frac{(c+r)!}{c!r!}$$

- Recall that r is the maximum number of times any φ_i is called.

Ex. $c = 2$, $r = 3$ $\quad N_b(2,3) = 10$



An Optimal Reversal Schedule (cont.)

Performance of Binomial Checkpointing:

Memory: $M_b = c$

Time Steps: $N_b(c, r) = \binom{c+r}{c} = \frac{(c+r)!}{c!r!}$

Comp. Cost: $T_b(c, r) = rN_b - \binom{c+r}{r-1}$

- As with multilevel checkpointing, the computational cost is no worse than r times the number of time steps.

Comparison with Multilevel Checkpointing

A comparison of multilevel and binomial checkpointing is helpful to get an appreciation of how many more steps the latter permits, for a given number of checkpoints and cost ratio r.

First, we need to determine the multilevel schedule that maximizes N_m. That is, we must solve the constrained (integer) optimization problem

$$\max_{c_i} N_m = \prod_{i=1}^{r}(1+c_i), \qquad \text{s.t.} \sum_{i=1}^{r} c_i = c.$$

Comparison with Multilevel Checkpointing (cont.)

If we assume that $r \propto c$, the solution to the above is attained when $c_i = c/r, \forall i$, which gives

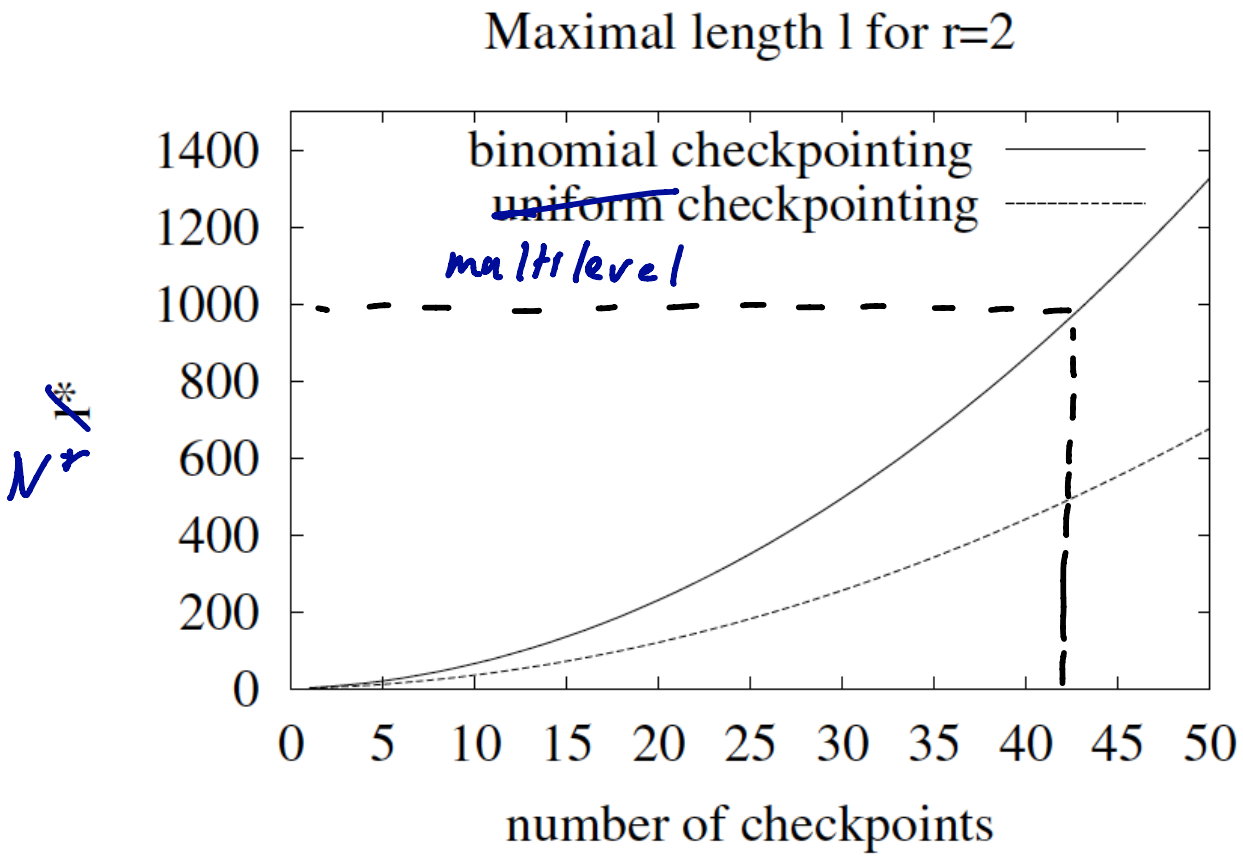
$$N_m^*(c, r) = \left(1 + \frac{c}{r}\right)^r.$$

Binomial checkpointing gives only one (optimal) value for the number of steps:

$$N_b(c, r) = \binom{c+r}{c} = \frac{(c+r)!}{c!r!}.$$

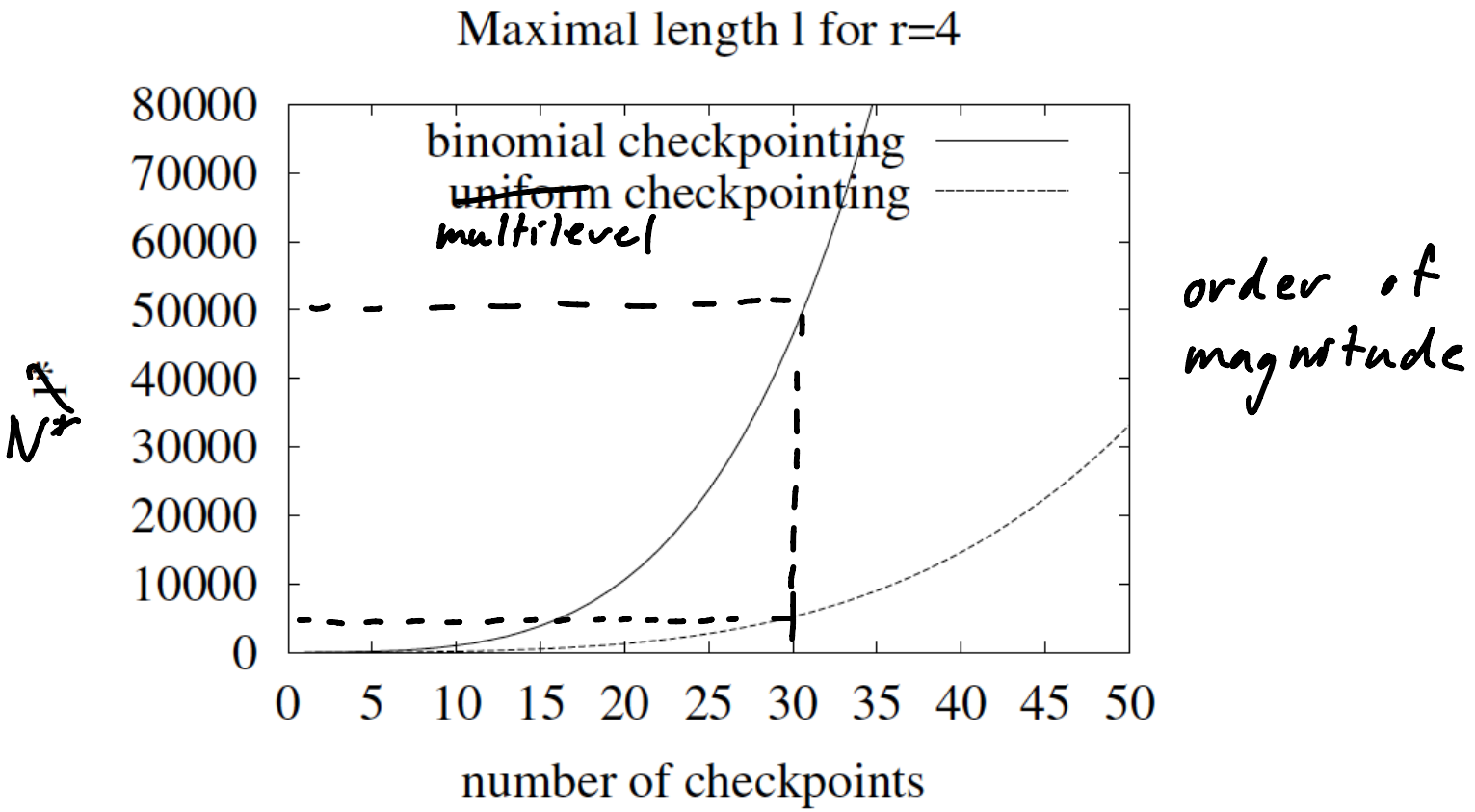
On the following slides we plot N_m^* and N_b versus c for fixed values of $r = 2$ and $r = 4$ (Note N is denoted as l in [GW09].)

Comparison with Multilevel Checkpointing (cont.)



Maximum steps possible versus number of checkpoints, for $r = 2$ [GW09].

Comparison with Multilevel Checkpointing (cont.)



Maximum steps possible versus number of checkpoints, for $r = 4$ [GW09].

Some Remarks

Compare the above results with the extreme cases $c = N$ and $c = 1$ considered at the beginning of the lecture.

- A modest amount of memory and increase in computational effort goes a long way!
- Binomial checkpointing is always better than multilevel checkpointing in terms of steps N for given c and r.
- Also, although we did not show it, binomial checkpointing gives $T_b \leq T_m$ [GW09].

References

[GW09] Andreas Griewank and Andrea Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, Society for Industrial & Applied Mathematics ; Cambridge University Press [distributor], 2009.