Test (3): Phytoplankton Patchiness Studies by Ship and Satellite (P^2S^3)

Purpose

To demonstrate the consistency between the rigid–lid streamfunction, rigid–lid surface pressure and free surface (surface pressure) formulations in a more realistic deep open domain. Also to test the various options for creating surface pressure initializations.

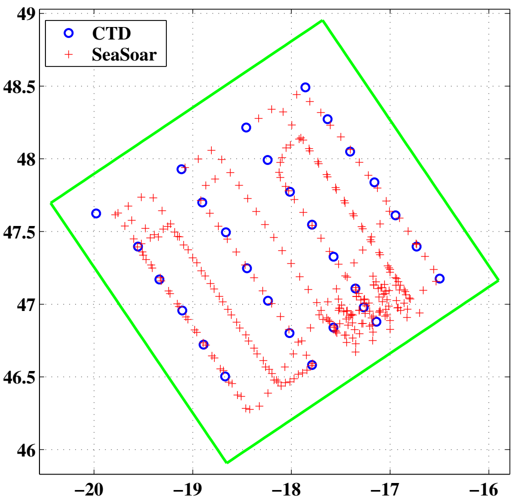
Status

A rotated Cartesian domain is defined in the eastern North Atlantic as

$$\begin{aligned} n_x,\ n_y &= 49,\ 51 \\ dx,\ dy &= 5\text{km},\ 5\text{km} \\ lat_c,\ lon_c &= 47.43\text{N},\ 18.17\text{W} \\ rotation &= -56.0^\circ \end{aligned}$$

60 model levels are spread throughout the water column in a hybrid system (illustrated in figure 2).

The initial field are created from an objective analysis of CTD and SeaSoar data obtained April 19 – May 13, 1997 during P^2S^3 (figure 1, at right with the boundary of the computational domain). The CTDs extend to 2000m while the SeaSoar are shallower (only to 300m). The data represent 3 surveys:



(1) an initialization survey of (essentially) alternating rows of CTDs and SeaSoar
(2) a small scale SeaSoar survey near the southeastern corner
(3) a final verification SeaSoar re-survey of the initialization pattern.

These data are melded with a background constructed from the CTD data vertically extended with deep climatological profiles. The data are objectively analyzed with the scales below. The dynamic height is recomputed from the mapped T/S fields with a level of no–motion of 2500m. The resulting fields are presented in figure 3.

OA Scales

	Synoptic	**Mean**
Decay (km)	40	200
Zero-Cross (km)	120	400
Time-Decay (day)	40	2000

(A) Streamfunction formulation

The streamfunction formulation simulations form the baseline for comparing the surface pressure formulation simulations. This series starts with the basic code and a simulation from Popova & Lozano. Various CPP options are then turned on to include the various supporting changes for the surface pressure formulation

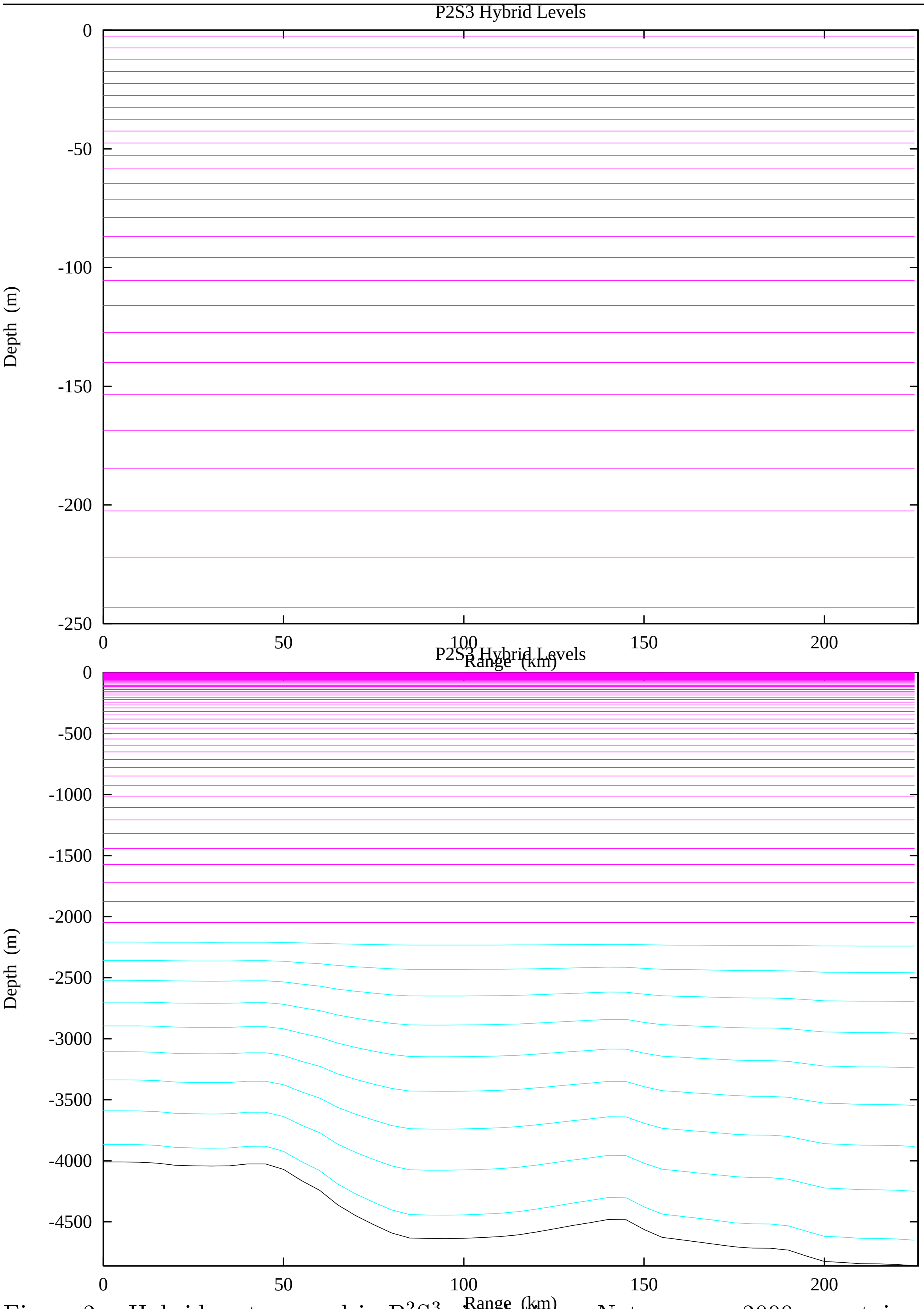


Figure 2 : Hybrid system used in P^2S^3 simulations. Note: upper 2000m contains purely flat levels, 1^{st} 10 levels are evenly spaced at 5m, subsequent levels progressively thicker.

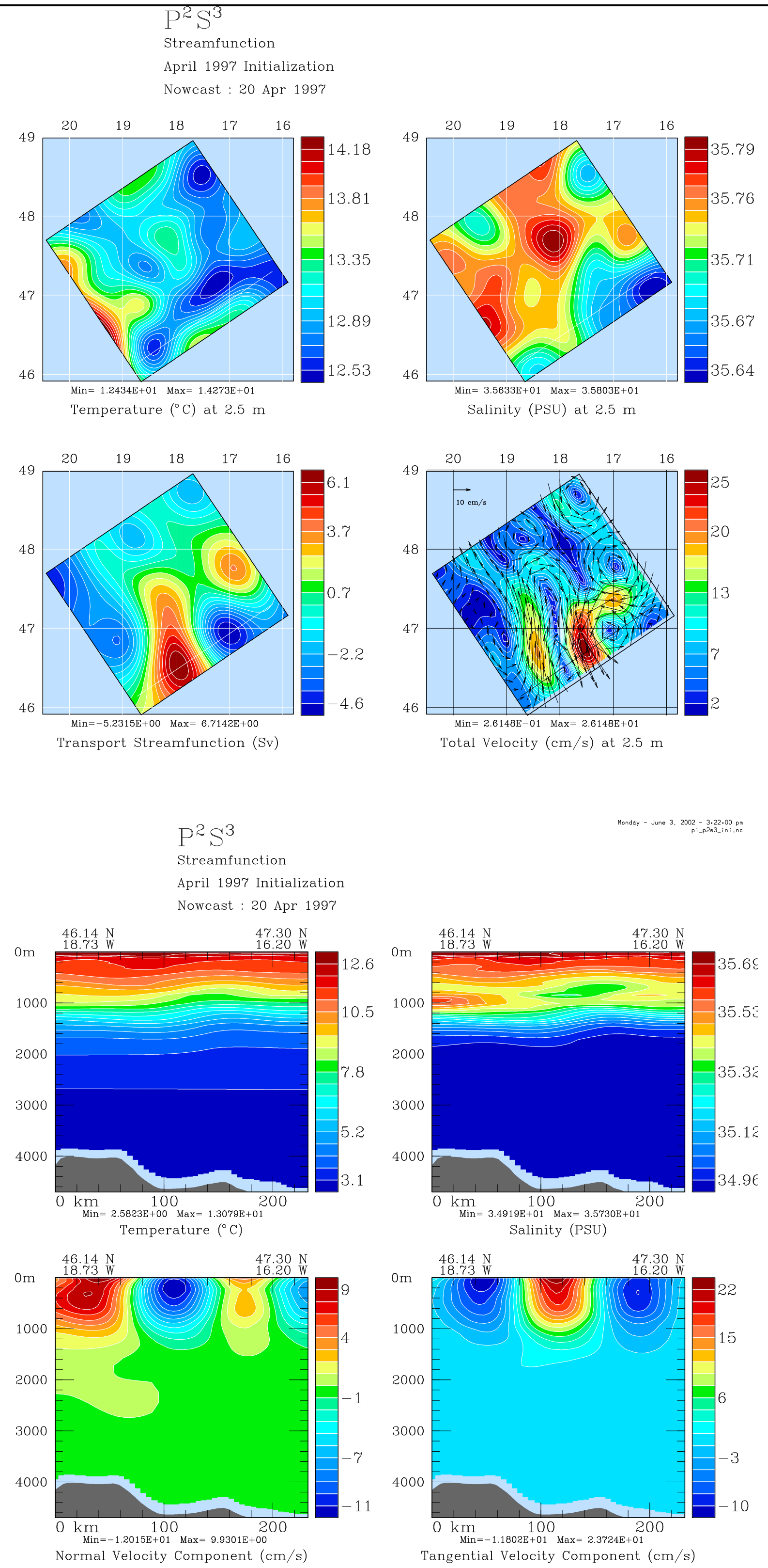


Figure 3 : P^2S^3 initialization. Surface fields and a vertical section running parallel to the southern boundary (indicated by the diagonal line in the surface plots).

(i) Baseline simulation

The baseline simulation is briefly presented in figures 4–7. The simulation produces reasonable synoptic structures and the kinetic energy is bounded. The levels shown are the surface level (2.5m), a mid–level depth (400m) chosen on the basis of eyeball estimates of an average depth for the sub-surface velocity local maxima and a deep level (1600m) chosen as the deepest surface with reasonable variability of temperature and salinity (see initialization sections, figure 3).

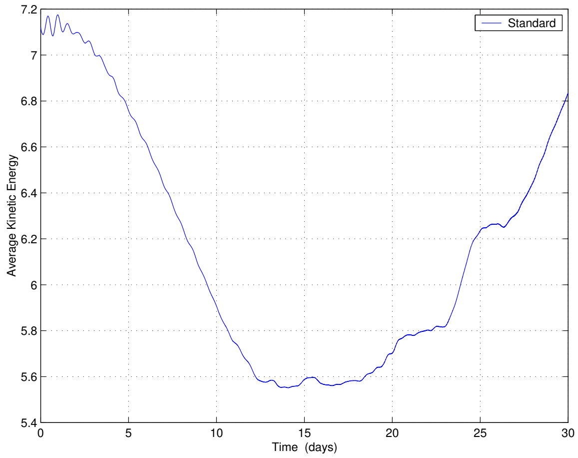


Figure 4 : Kinetic Energy for baseline simulation.

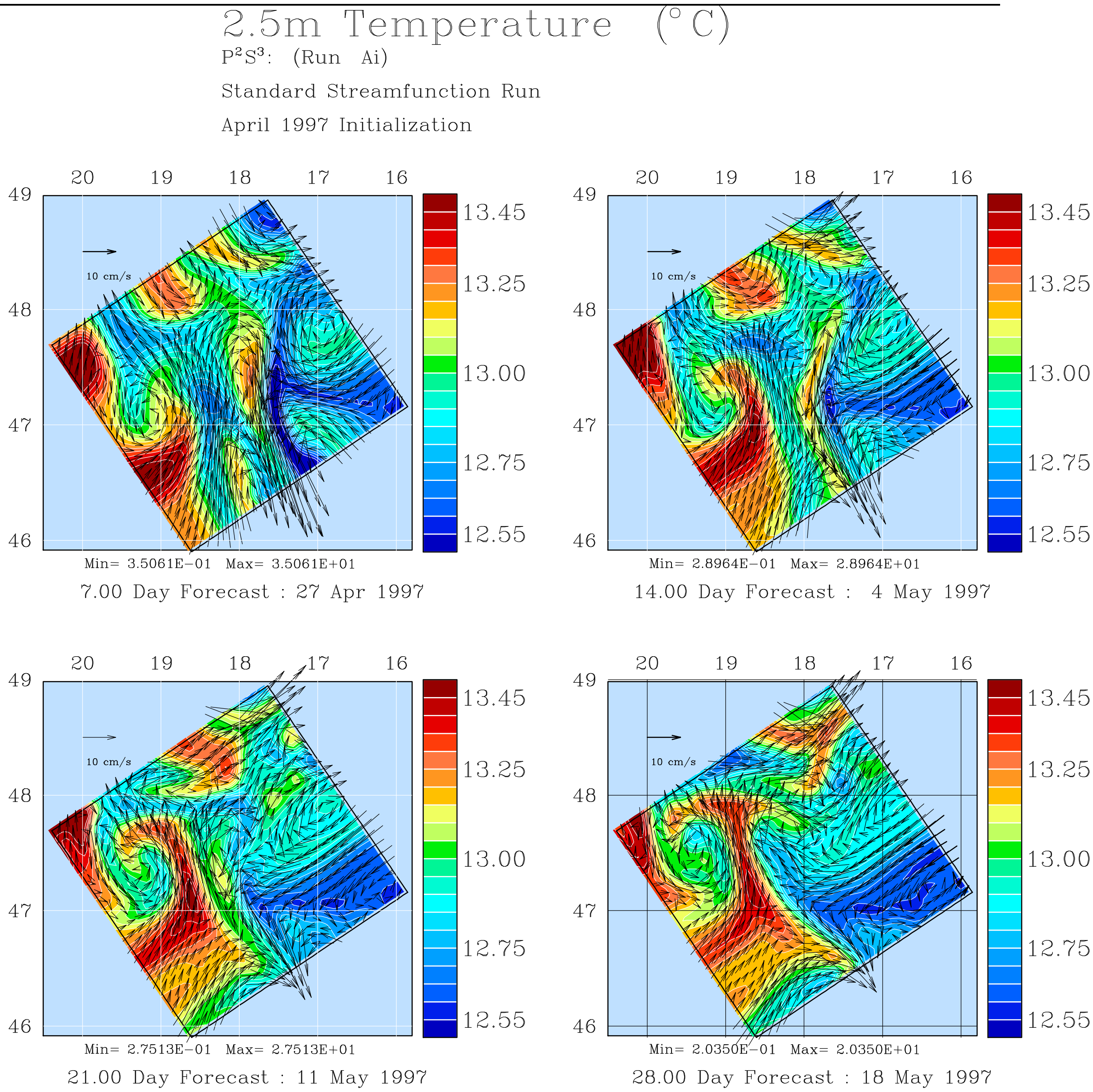


Figure 5 : Baseline streamfunction formulation. Near surface temperature with total velocity vectors.

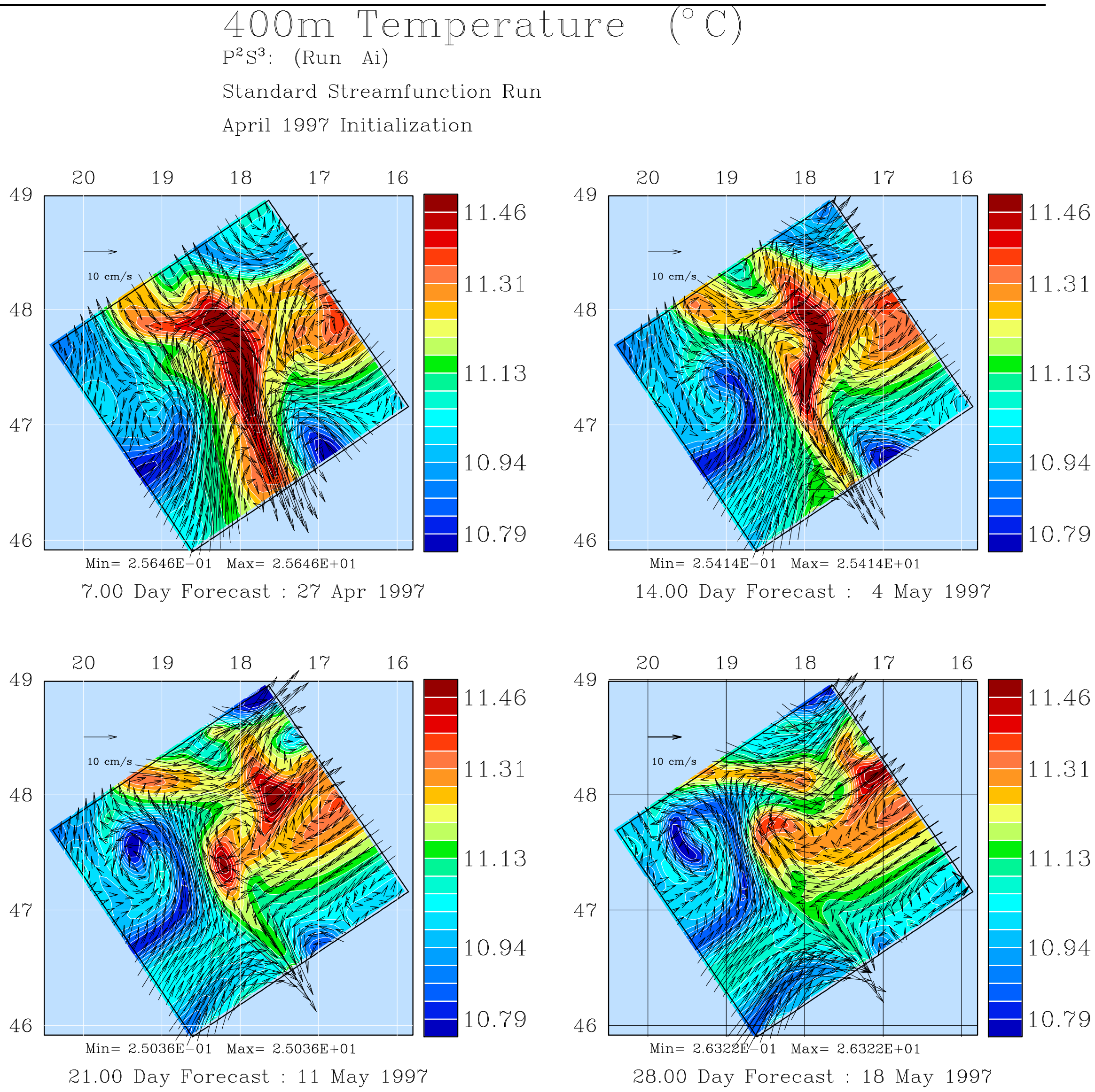


Figure 6 : Baseline streamfunction formulation. Mid–level temperature with total velocity vectors.

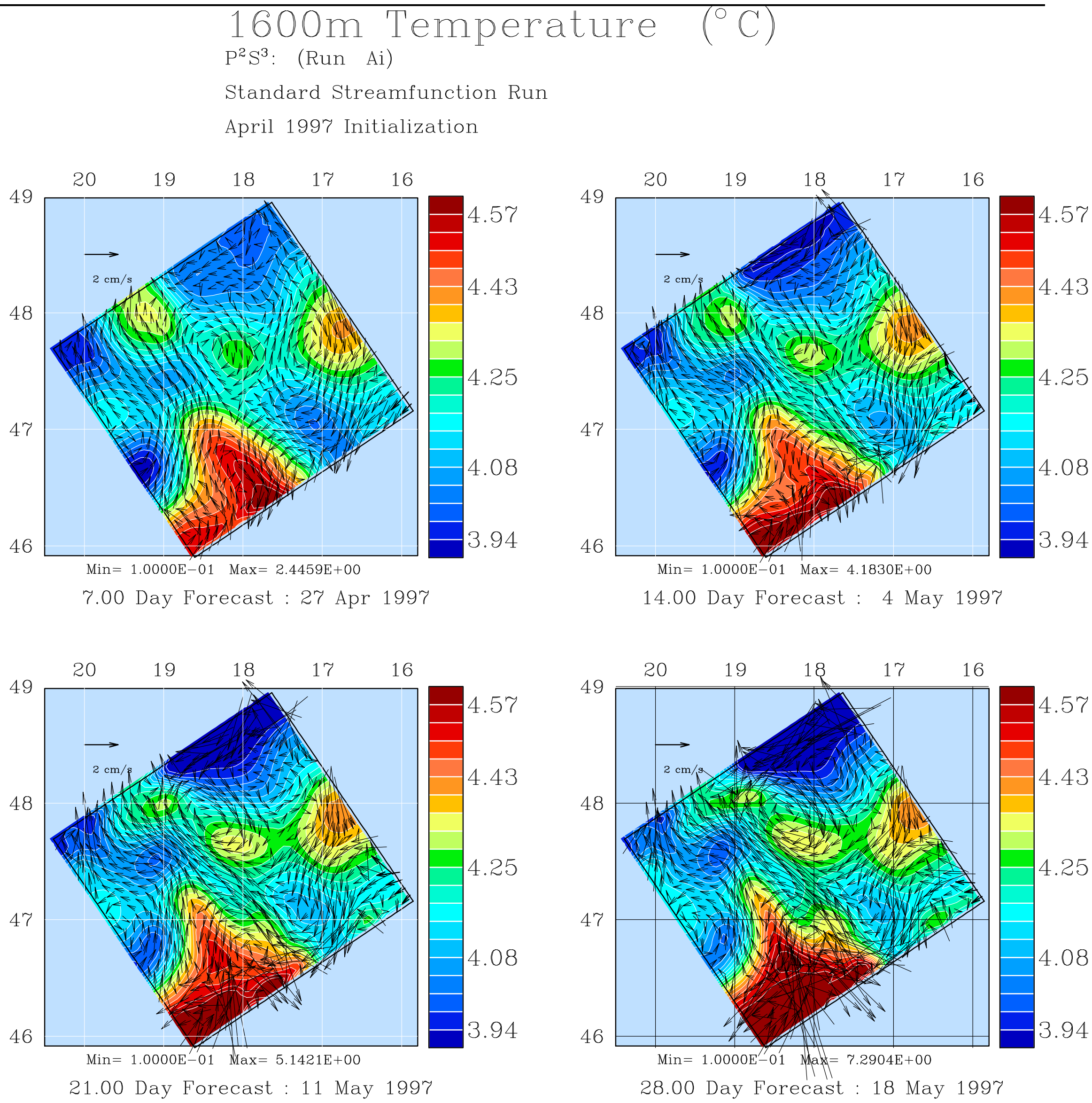


Figure 7 : Baseline streamfunction formulation. Deep temperature with total velocity vectors.

(ii) Pressure formulation momentum fluxes. (CPP option `mfluxavg`)

In order to maintain the vertically integrated conservation of mass, the pressure formulation requires a modified discretization (6 point) of the momentum fluxes. For compatibility, Lozano & Lanerolle provided the option of using this discretization in the streamfunction formulation.

A comparison between this simulation and *(i)* is given in figures (8–15). The simulations are very close. The differences remain near the boundaries, especially in the vicinity of the northernmost corner in the upper levels. The averaged kinetic energy shows good quantitative agreement through 5–7 days and good qualitative agreement throughout.

Additional comparisons are made through the use of the root mean square error (RMS) and the pattern correlation coefficient (PCC). Defining a field (T), its estimate ($\hat{T}$) and their corresponding large scale average counterparts (τ, $\hat{\tau}$), the RMS and PCC are defined as

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum\left(T-\hat{T}\right)^2} \quad , \qquad \mathrm{PCC} = \frac{\sum\left(T-\tau\right)\left(\hat{T}-\hat{\tau}\right)}{\sqrt{\sum\left(T-\tau\right)^2\sum\left(\hat{T}-\hat{\tau}\right)^2}}$$

A perfect match would have an RMS=0 and a PCC=1. Figure 9 shows these statistics for temperature, salinity and both components of horizontal velocity. The solid lines are the comparison between this simulation and *(i)*. The dashed lines are the comparison between *(i)* and the initial conditions (persistence). Persistence is included for a visual normalization of the differences due to algorithm change with the differences due to dynamics. In all cases, the statistics indicate a good agreement between the simulations, especially for the first 5–7 days.

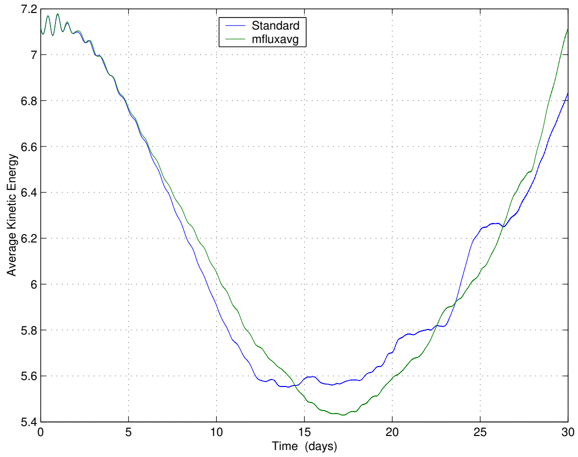


Figure 8 : Kinetic Energy comparison between baseline simulation *(i)* versus 6 point flux algorithm *(ii)* (CPP option `mfluxavg`).

RMS Error **Pattern Correlation Coefficient**

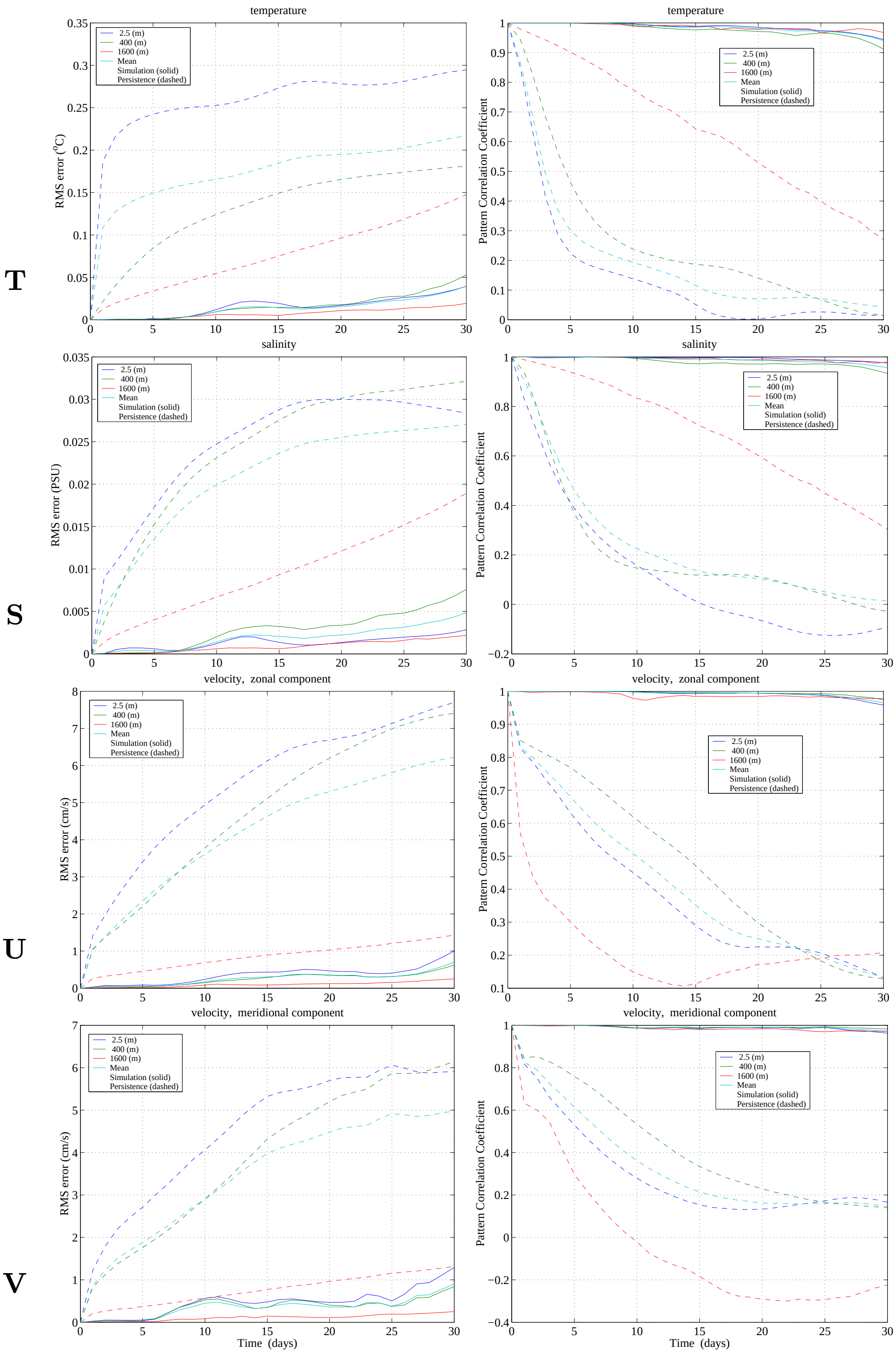


Figure 9: Comparison of baseline run *(i)* versus 6 point flux algorithm *(ii)*

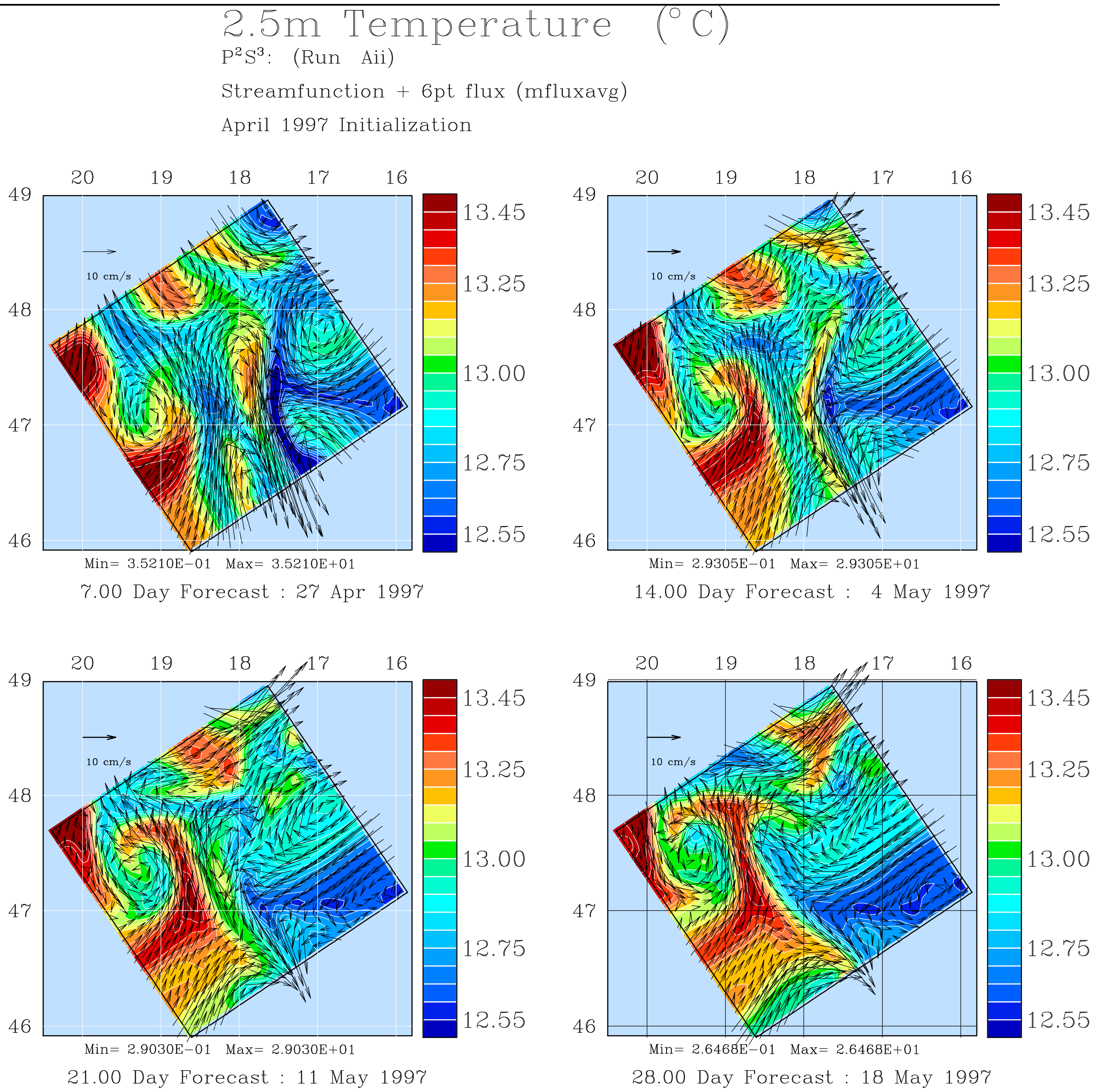


Figure 10 : Streamfunction formulation with 6 point flux algorithm (CPP option `mflux-avg`). Near surface temperature with total velocity vectors.

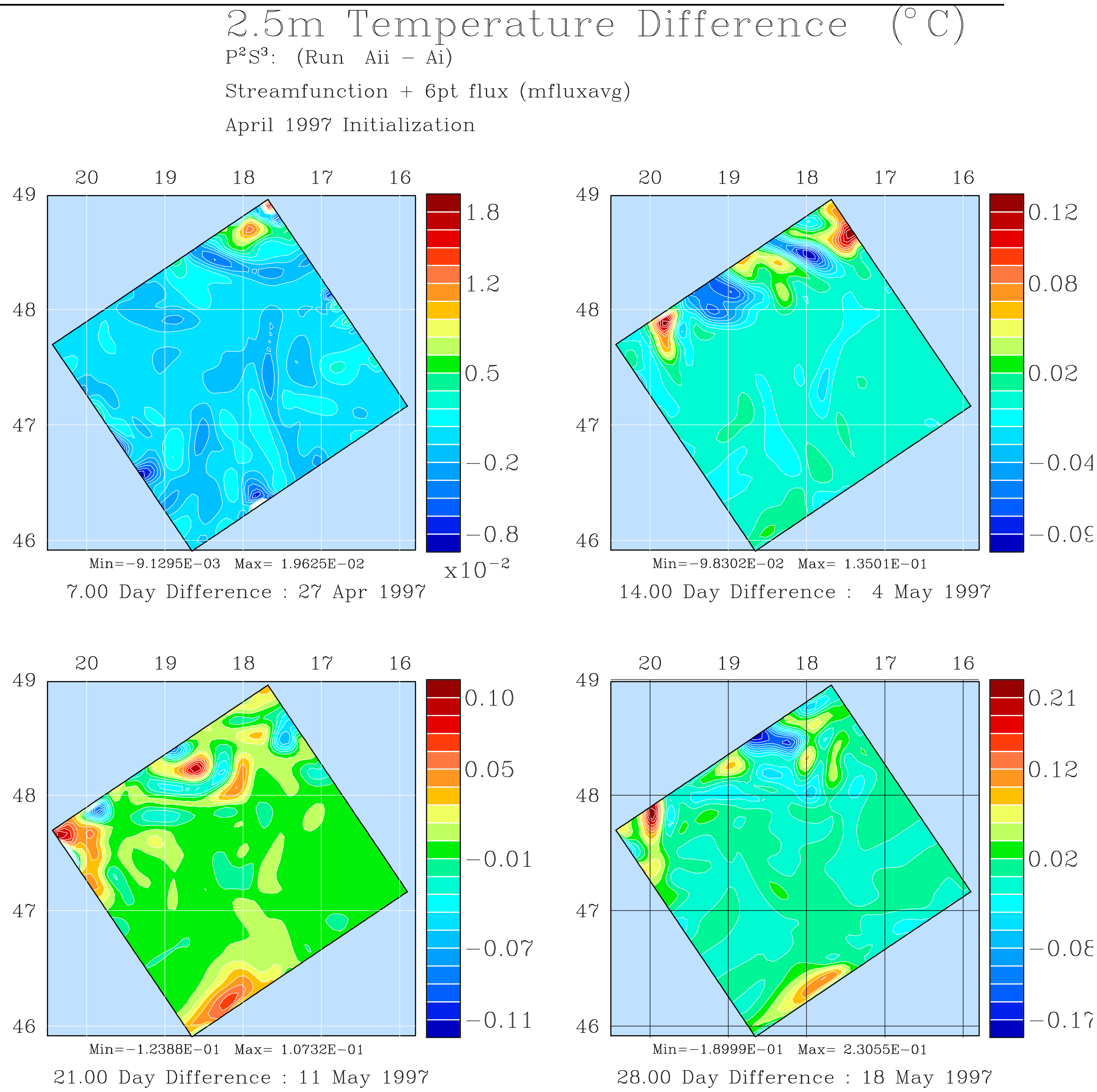


Figure 11 : Near Surface comparison of baseline simulation to simulation with 6 point flux algorithm (CPP option `mfluxavg`).

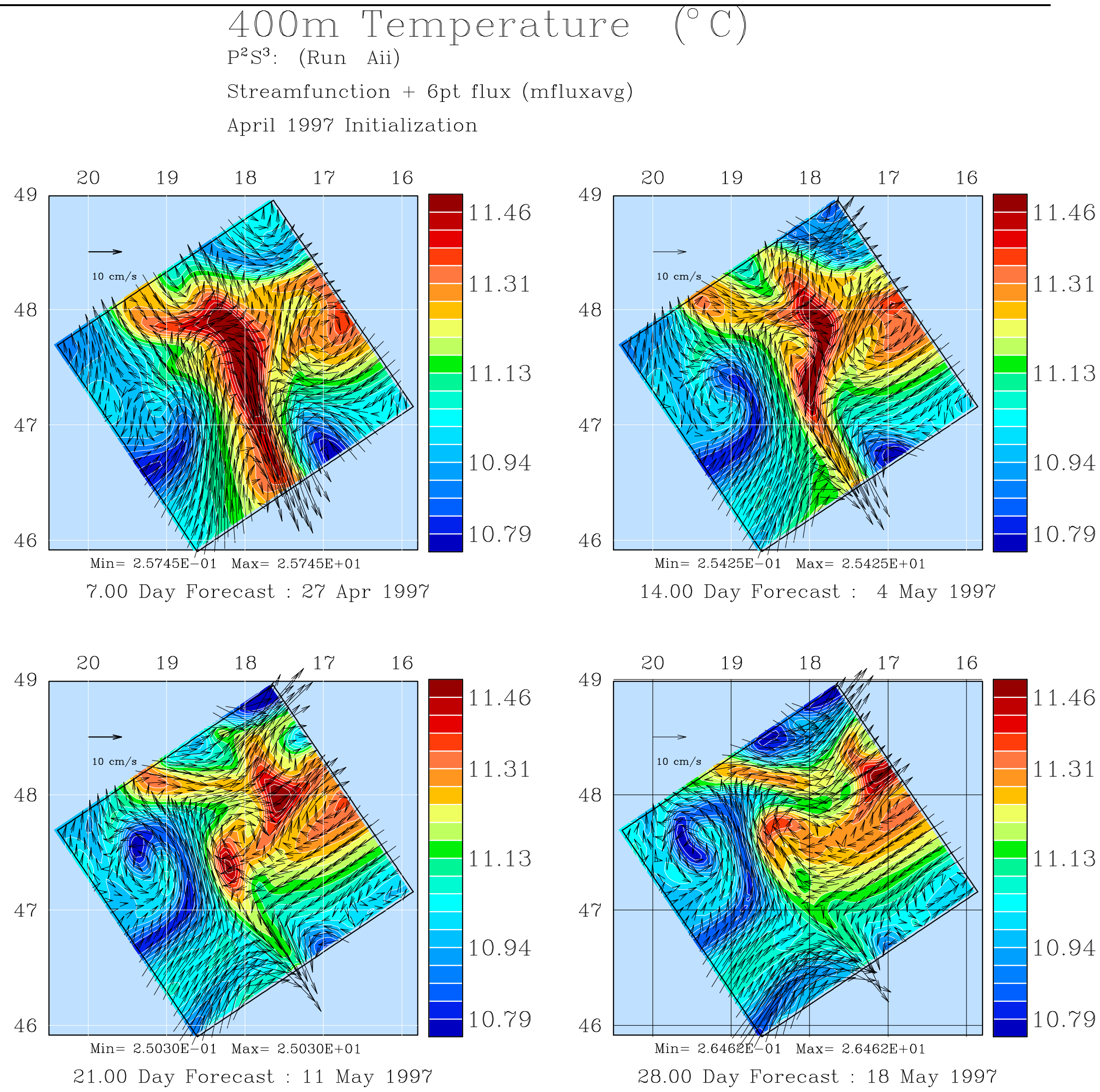


Figure 12 : Streamfunction formulation with 6 point flux algorithm (CPP option `mfluxavg`). Mid–level temperature with total velocity vectors.

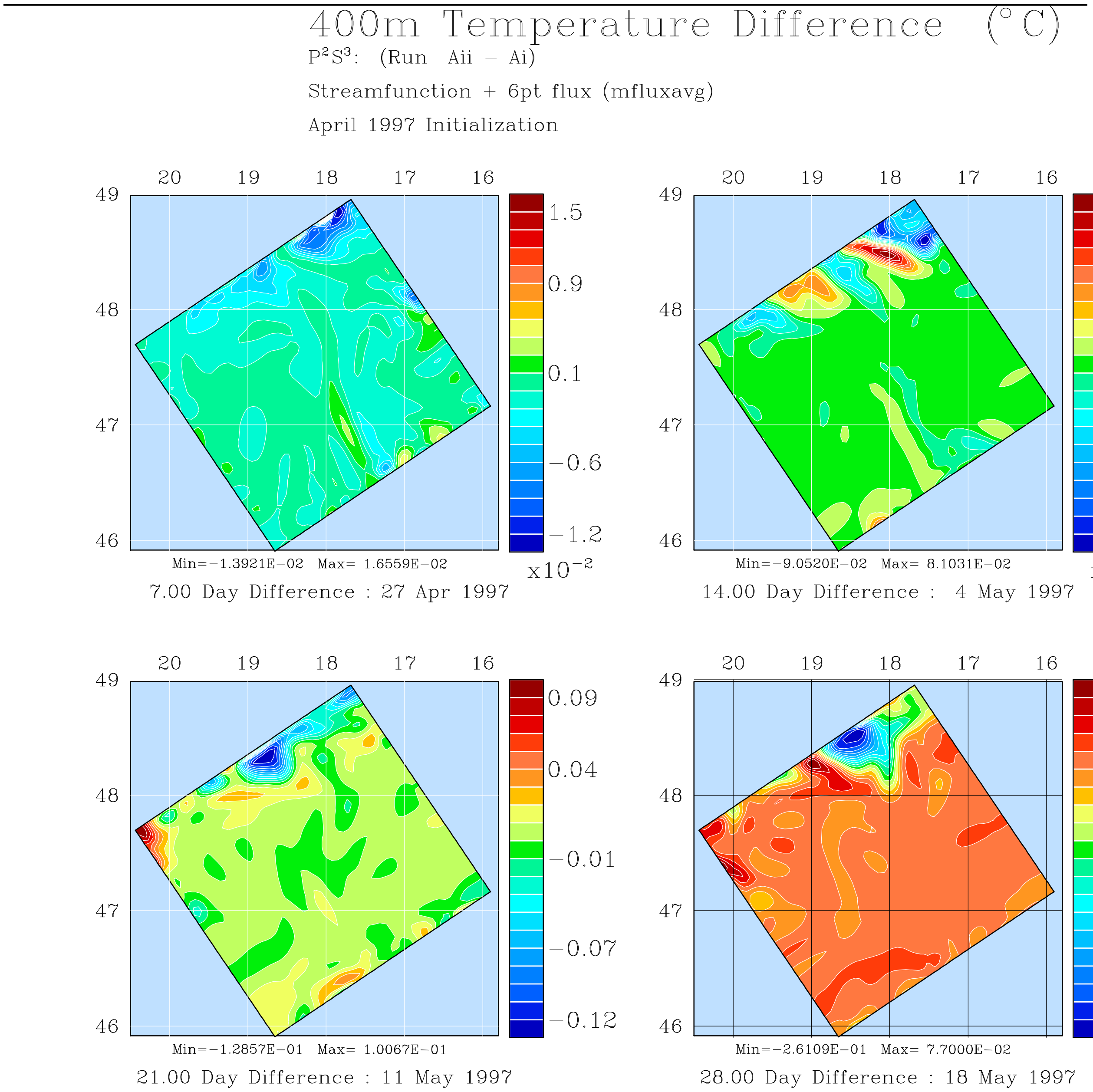


Figure 13 : Mid–level comparison of baseline simulation to simulation with 6 point flux algorithm (CPP option `mfluxavg`).

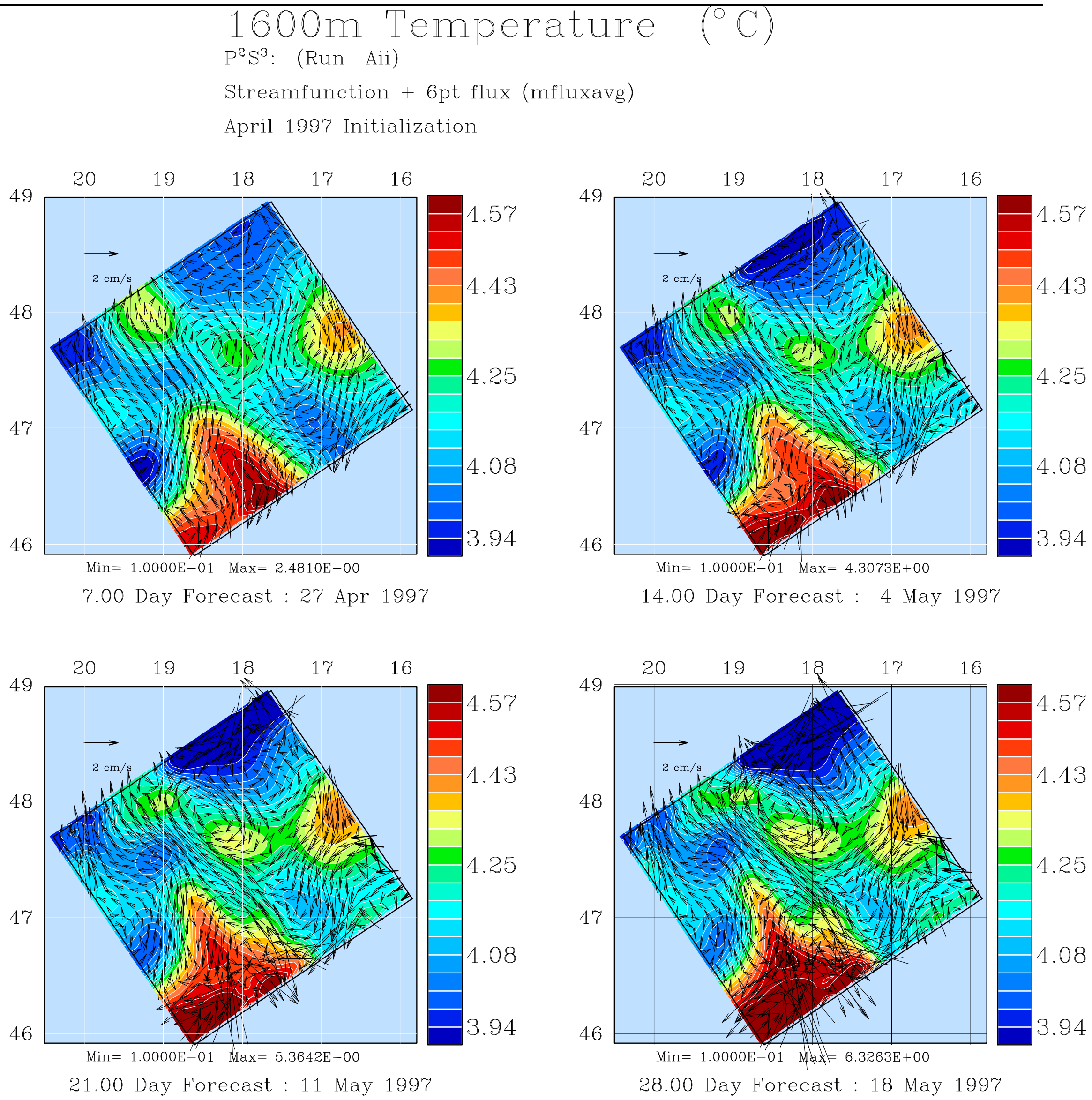


Figure 14 : Streamfunction formulation with 6 point flux algorithm (CPP option `mfluxavg`). Deep temperature with total velocity vectors.

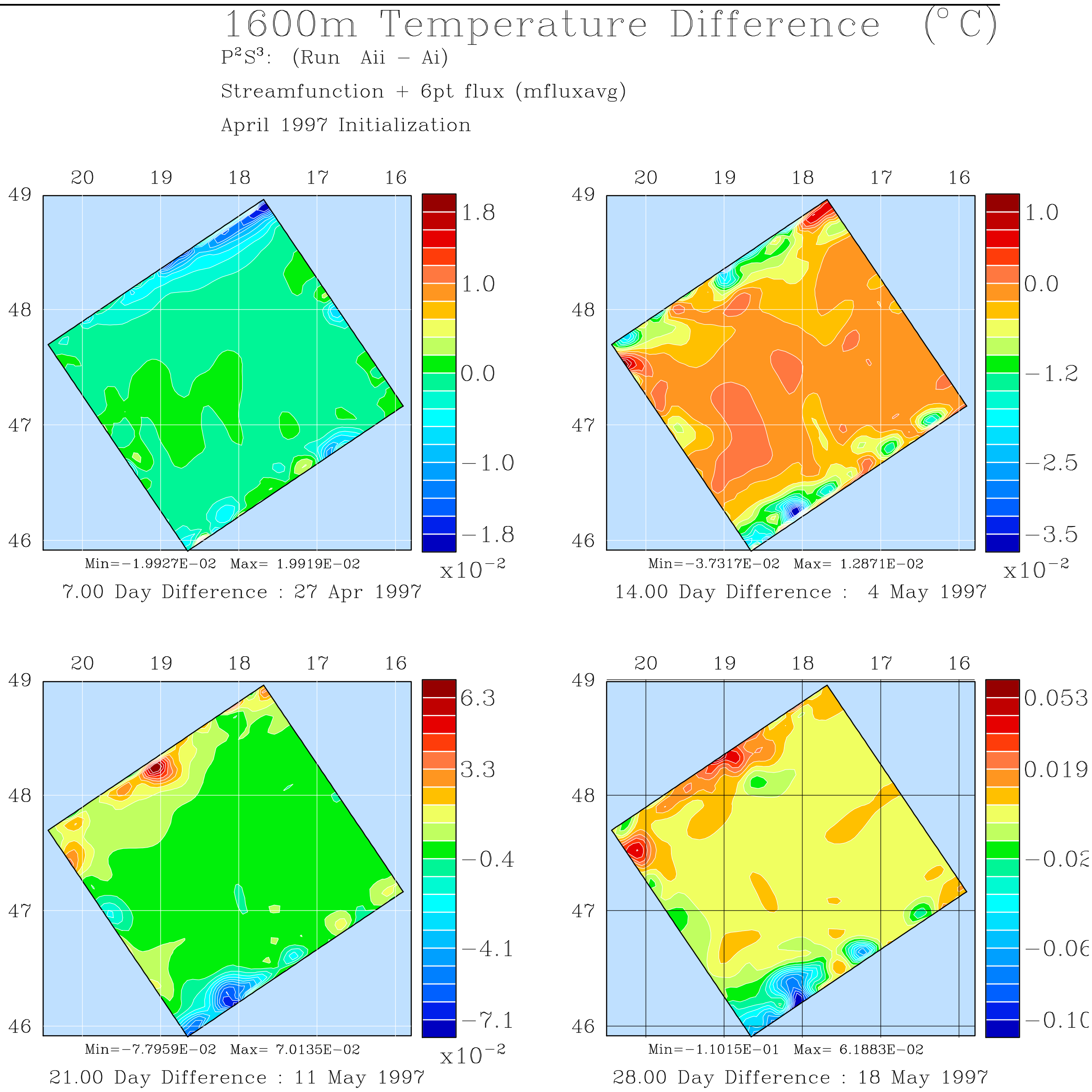


Figure 15 : Deep comparison of baseline simulation to simulation with 6 point flux algorithm (CPP option `mfluxavg`).

(iii) Pressure formulation momentum fluxes, alternative formulation. (CPP options `mflux-avg` + `tfluxes`)

The alternative formulation is a variation in the handling of land masks, specifically the method of preventing the 6–point flux operator from using invalid data under the land mask.

- In the original formulation:
  - In the absence of land masks the operator is coded without reference to masks.
  - In the presence of land masks, a binary [0,1] mask is tested before each evaluation of the velocity.
- In the new formulation, each evaluation of the velocity is multiplied by the binary mask.

In light of the fact that the P^2S^3 problem has no land mask, the results are unsurprising.

- The difference fields between cases *(ii)* and *(iii)* are identically zero after 30 days.
- With no coding for masks, case *(ii)* is marginally faster than case *(iii)* (approximately 1 cpu minute in a simulation requiring 3 cpu hours).

As the fields are identical to case *(ii)*, no plots will be presented here.

(iv) Addition of Asselin filter. (CPP options `mfluxavg` + `AsselinFilt`)

To inhibit the odd–even decoupling of the leap–frog time–stepping, the HOPS PE model has a pair of options

(1) Intermittent revision to forward Euler time–stepping (a first order algorithm).
(2) An Asselin time filter at every time step.

Lozano disabled the Euler time–stepping option when the surface pressure formulation is activated since:

- (i) At an Euler step there is the need to add additional cumbersome code to accommodate the difference between Euler and leap frog time steps.
- (ii) The Asselin filter works well.
- (iii) The effects of Asselin filter versus Euler time–stepping can be explored in the streamfunction formulation.

A comparison of this simulation to *(i)* is presented in figures 16–23. The first observation is that the addition of the Asselin filter produces a greater change than the 6–point flux algorithm. The kinetic energy still shows good quantitative agreement to 5–7 days but decays to a much greater extent after ∼12 days. The RMS & PCC show similar behavior. The discrepancies are still more concentrated near the domain boundaries, especially in the upper 400m, near the northernmost corner. For the comparisons in figures 17–23 and for the remaining runs in this series, the Asselin filter parameter ν=0.01 was selected, based on a marginally greater similarity to the previous KE histories.

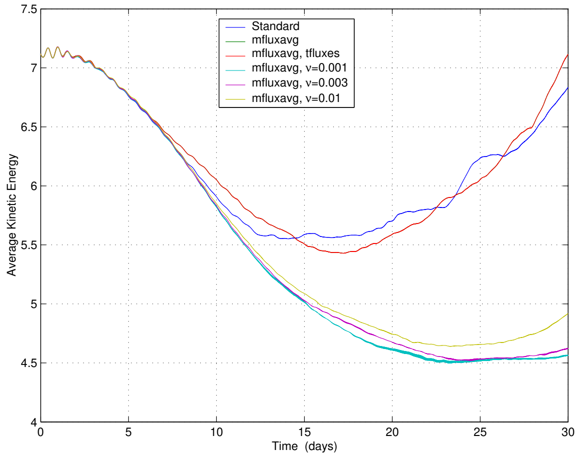


Figure 16 : Kinetic Energy comparison between baseline simulation *(i)* versus 6 point flux algorithm & Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`).

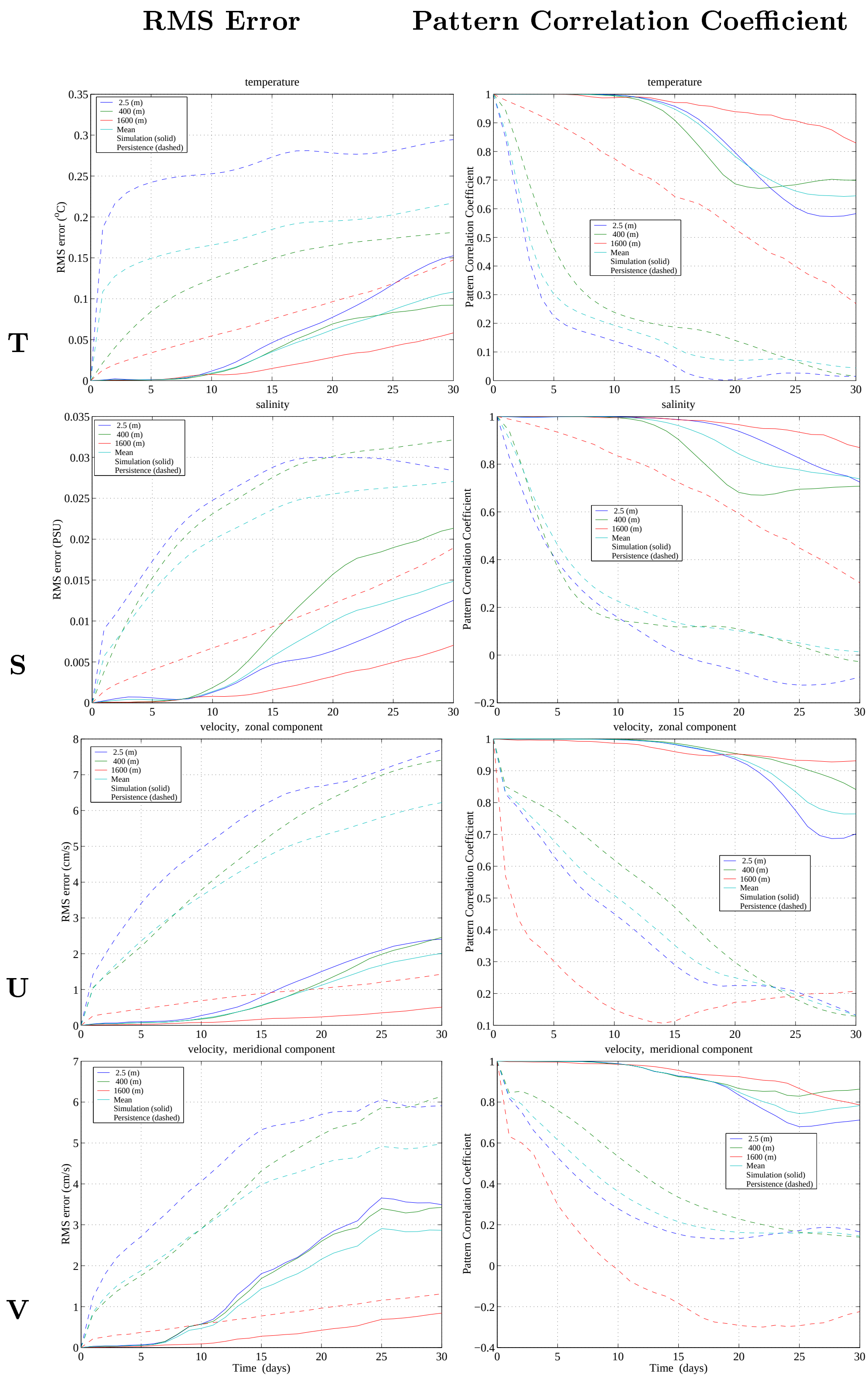


Figure 17: Comparison of baseline run *(i)* versus 6 point flux algorithm + Asselin Filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`).

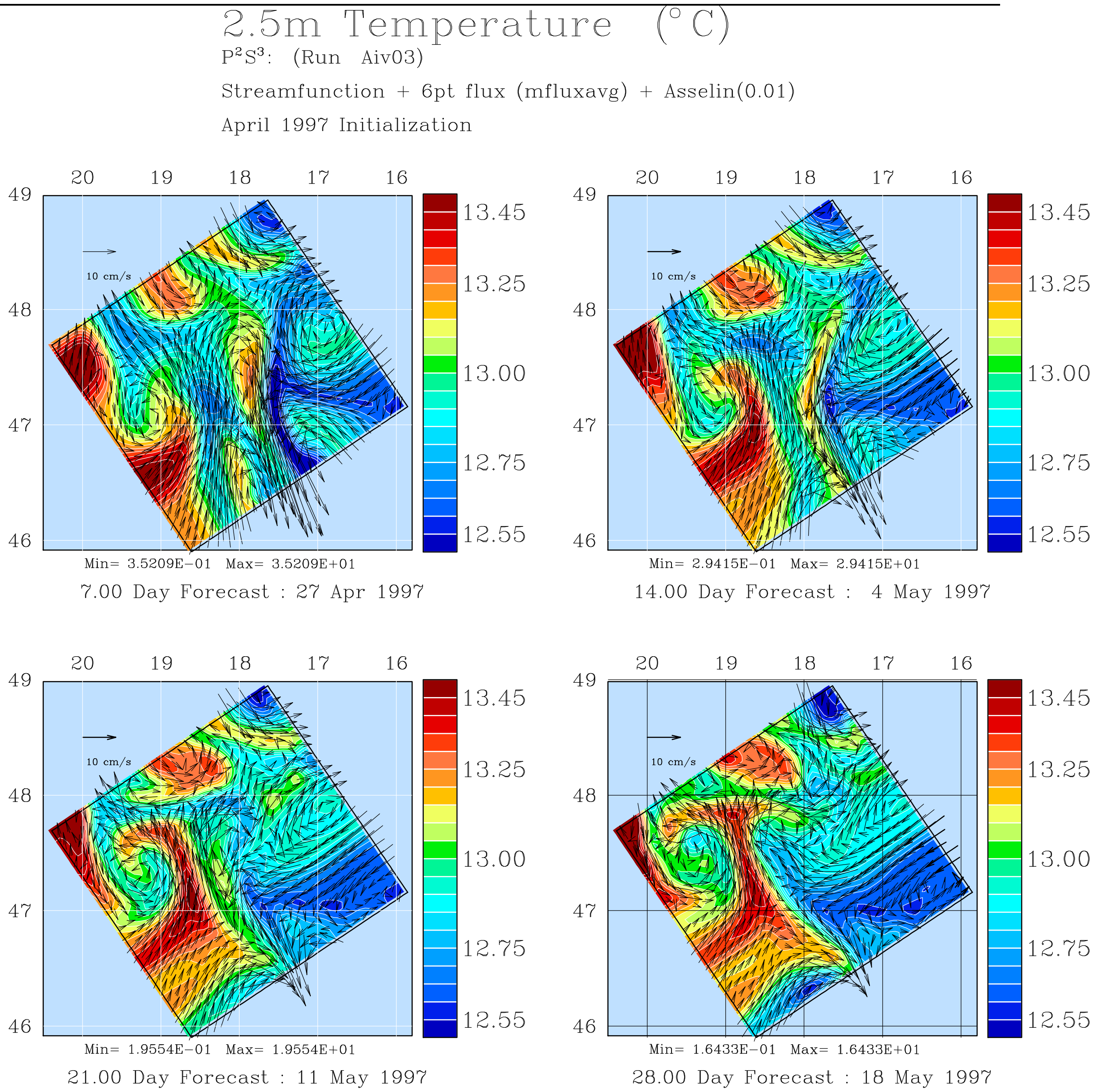


Figure 18 : Streamfunction formulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`). Near surface temperature with total velocity vectors.

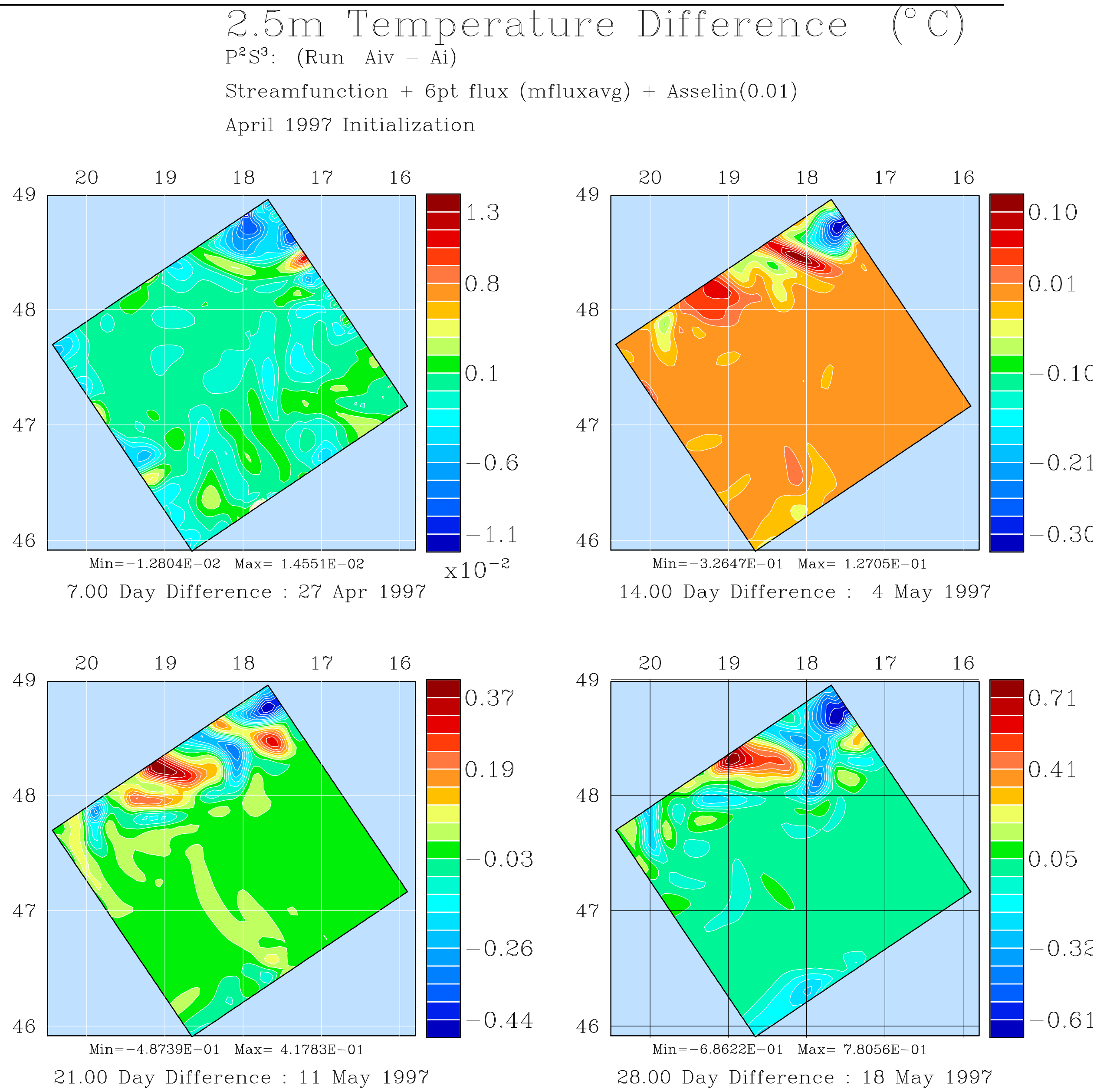


Figure 19 : Near Surface comparison of baseline simulation to simulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`).

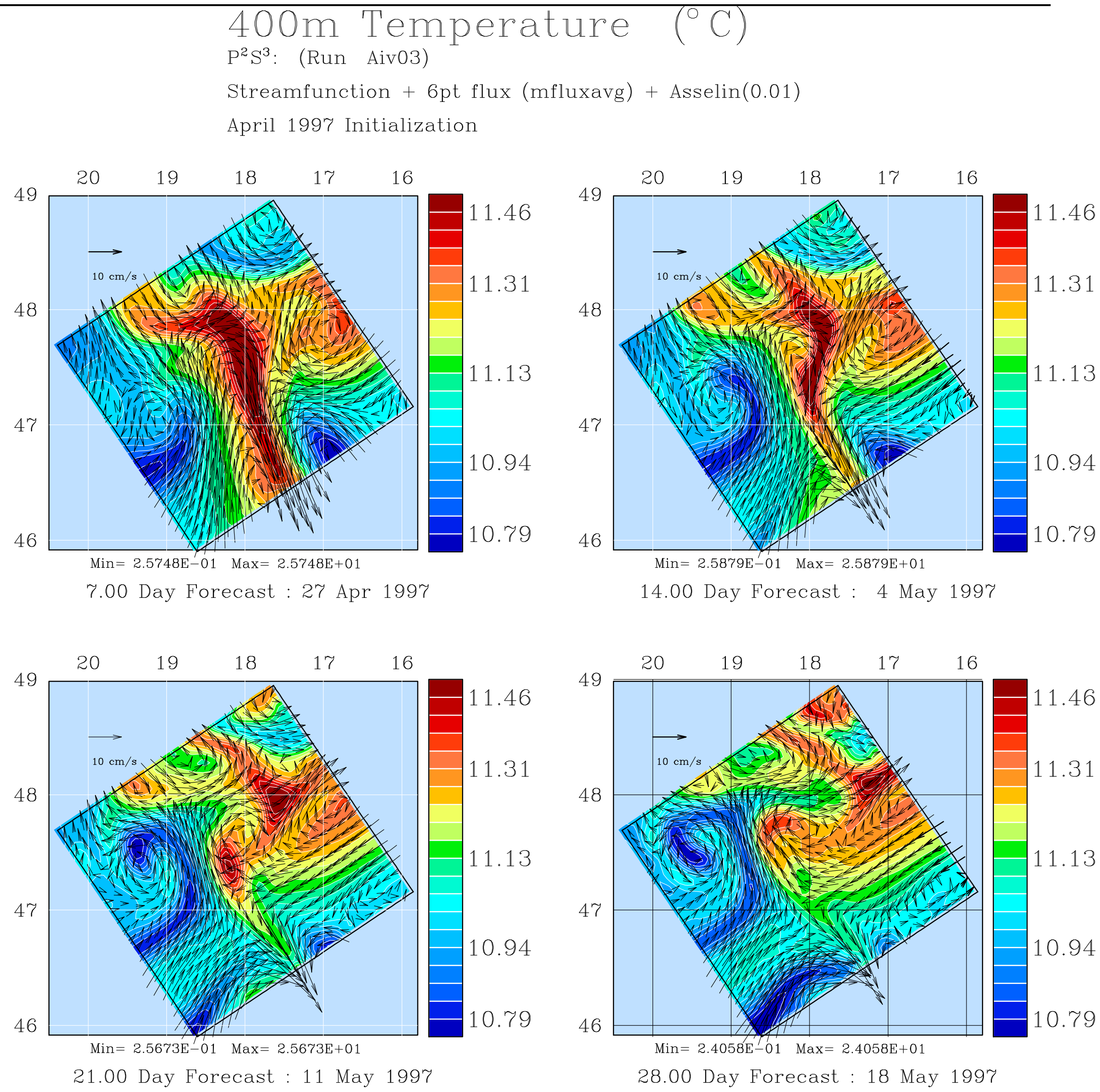


Figure 20 : Streamfunction formulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options **`mfluxavg`** + **`AsselinFilt`**). Mid–level temperature with total velocity vectors.

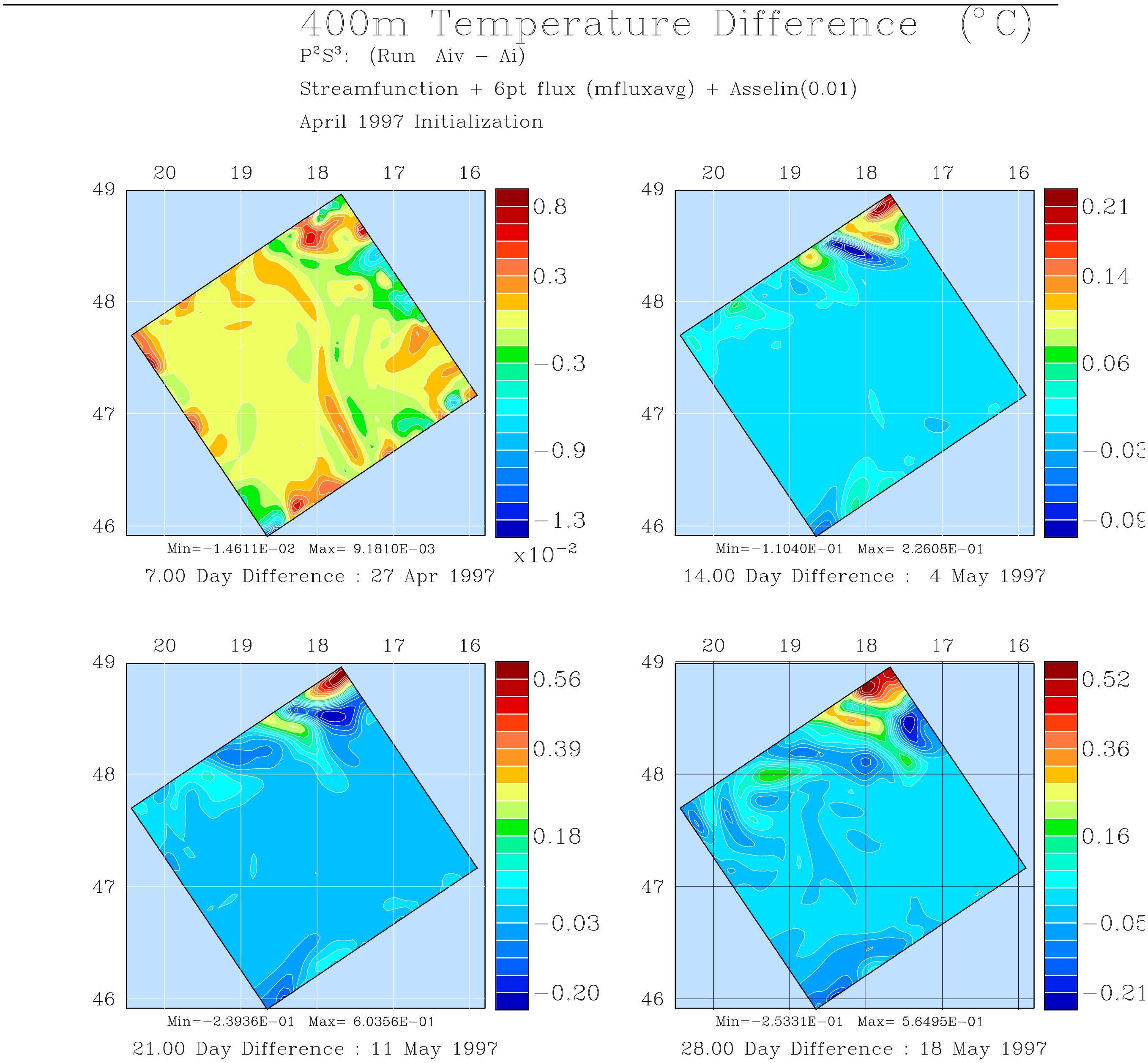


Figure 21 : Mid–level comparison of baseline simulation to simulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`).

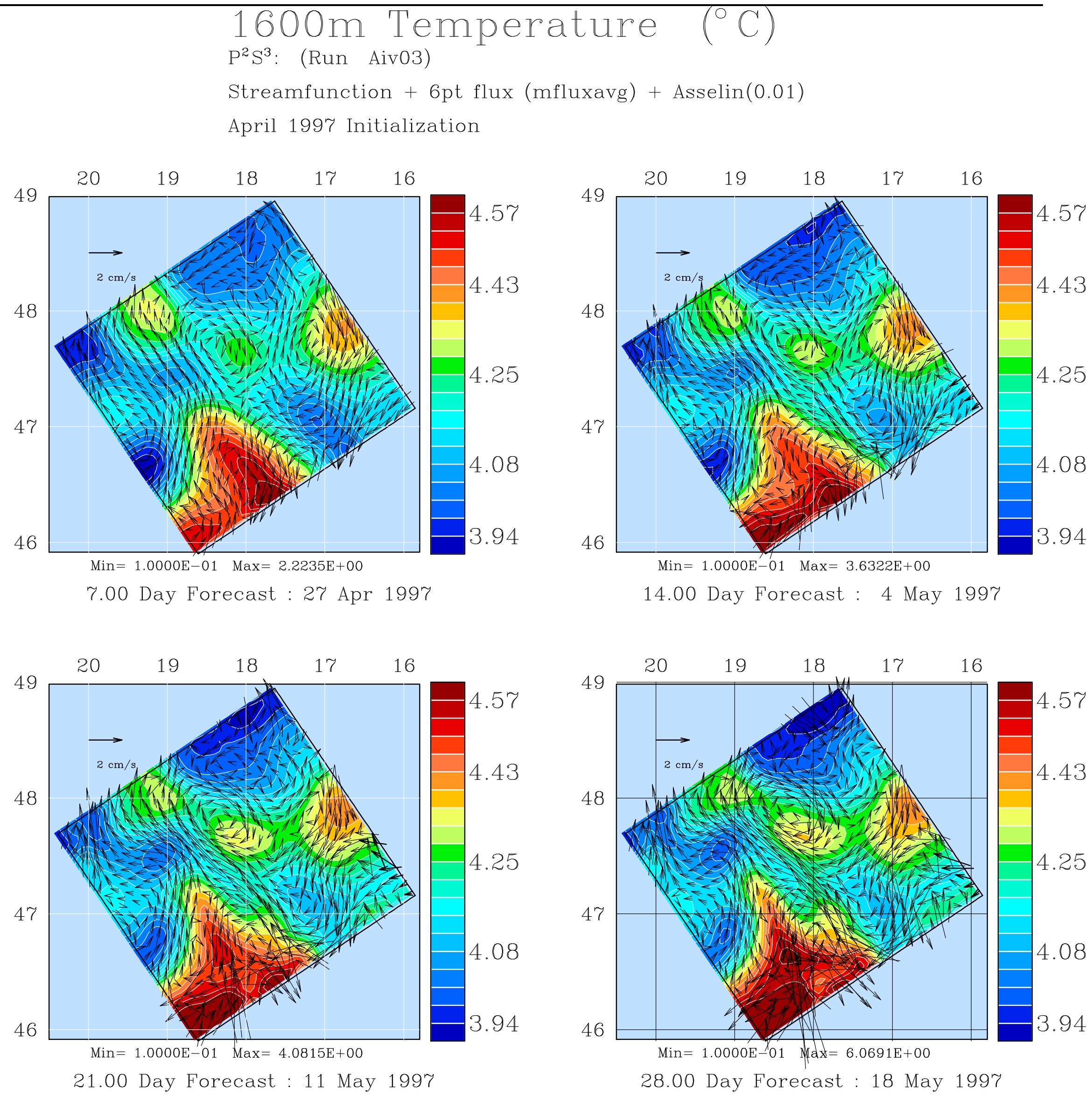


Figure 22 : Streamfunction formulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`). Deep temperature with total velocity vectors.

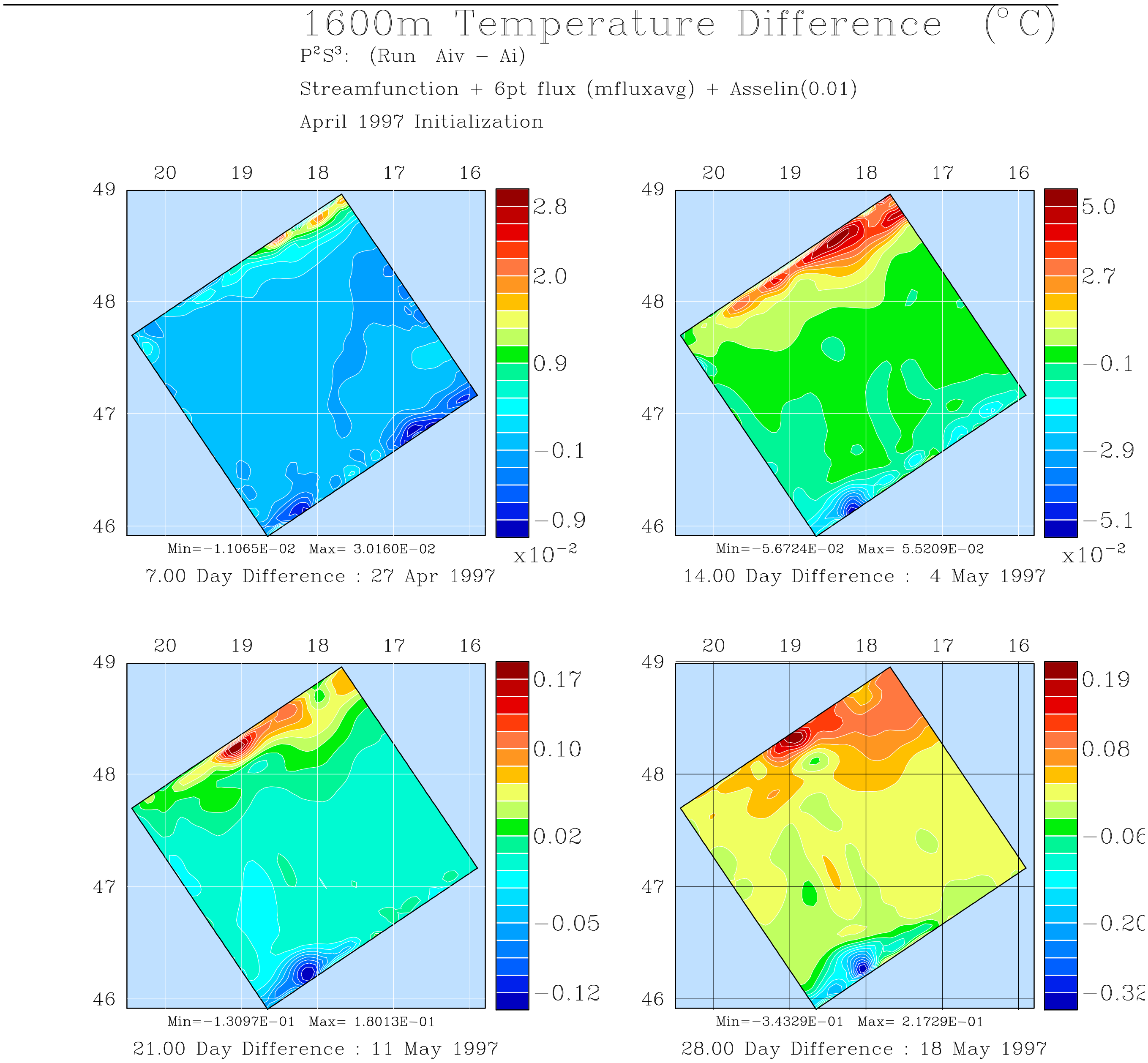


Figure 23 : Deep comparison of baseline simulation to simulation with 6 point flux algorithm plus Asselin filter *(iv)* (CPP options `mfluxavg` + `AsselinFilt`).

(v) add Pre–conditioned Conjugate Gradient solver (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`). Compare both solutions and timings.

To more easily and efficiently implement the new Poisson problem for the surface pressure formulation, Lozano & Lanerolle have introduced a sparse matrix, preconditioned conjugate gradient solver (SparsKit).

- The ability to test the preconditioning outside of the model was not as well developed as for the surface pressure formulation, when this study was made.
- Without preconditioning, the elliptic solver requires 0.234 CPU seconds. The speed–up factor for the preconditioning factors (small value tolerance, τ_{small}, and fill–in bandwidth, $fill$) are given below.

$fill \setminus \tau_{\text{small}}$	10^{-2}	10^{-3}	10^{-4}
10	1.7	1.1	1.1
15	1.7	2.0	1.2
20	1.7	1.9	1.5
25	1.7	1.9	1.3

- Taking speed–up values of 1.9 and 2.0 to be roughly the same, central values for the preconditioning factors are identified: $fill = 20$ and $\tau_{\text{small}} = 10^{-3}$.

Comparing the cumulative PE timings for the Poisson solver over the 30 day simulation:

- Standard HOPS conjugate gradient: 32.9915 seconds
- New preconditioned conjugate gradient: 490.9697 seconds

As used, the new SparsKit solver is more expensive than the standard HOPS solver.

A comparison of this simulation to *(i)* is presented in figures 24–31. The results are very similar to *(iv)*, indicating that most of the change is due to the Asselin filter.

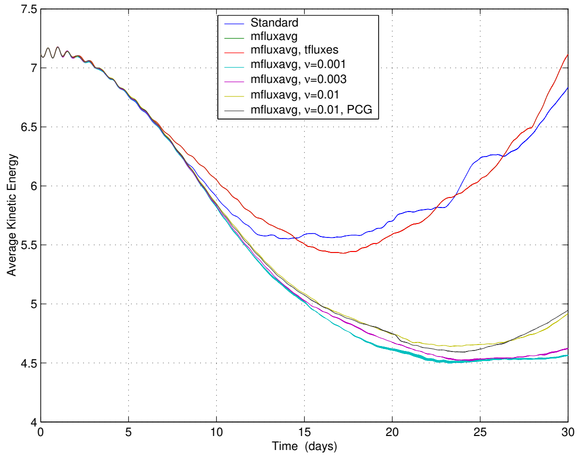


Figure 24 : Kinetic Energy comparison between baseline simulation *(i)* versus 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`).

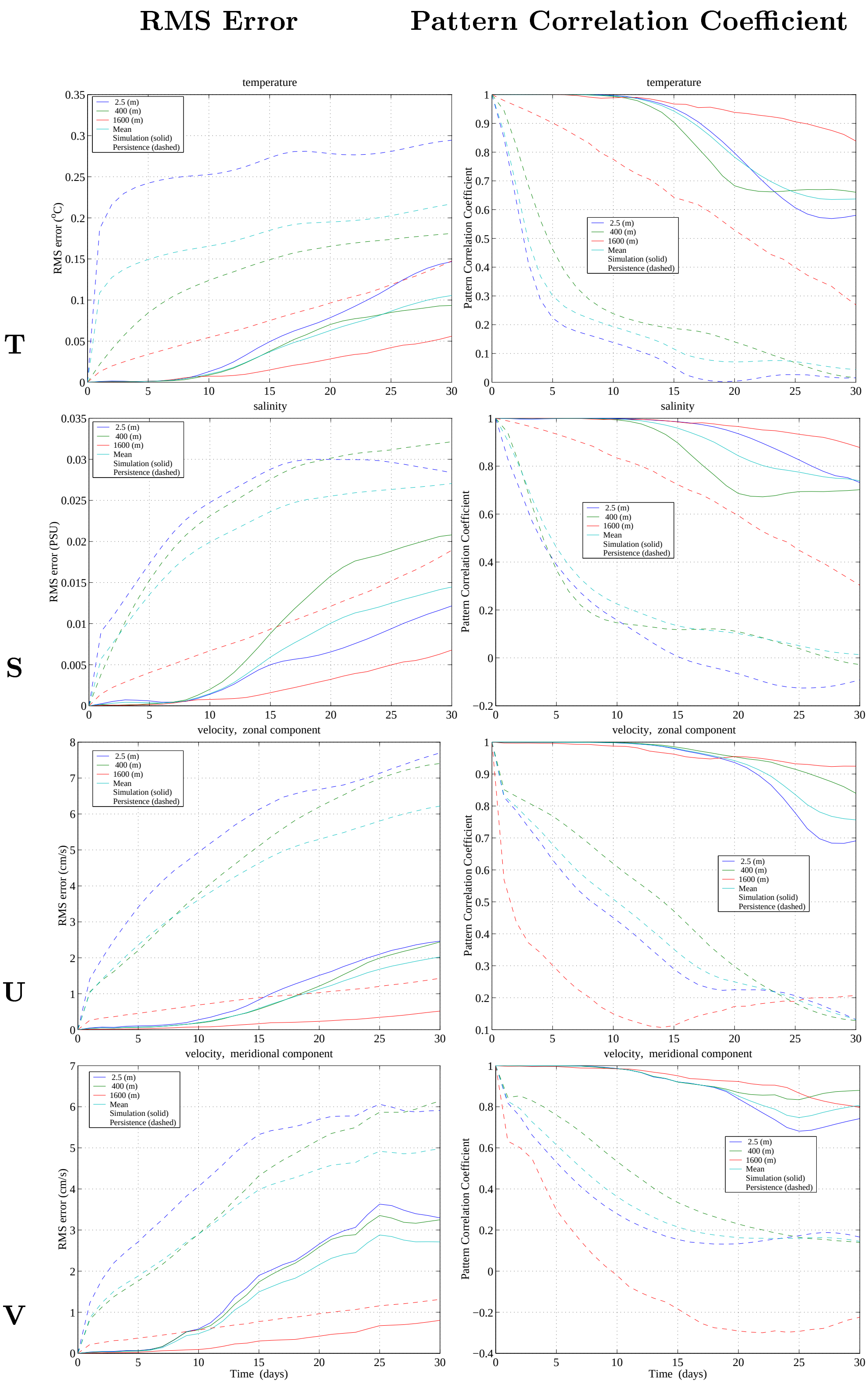


Figure 25: Comparison of baseline run *(i)* versus 6 point flux algorithm, Asselin Filter and preconditioned conjugate gradient solver. *(v)*

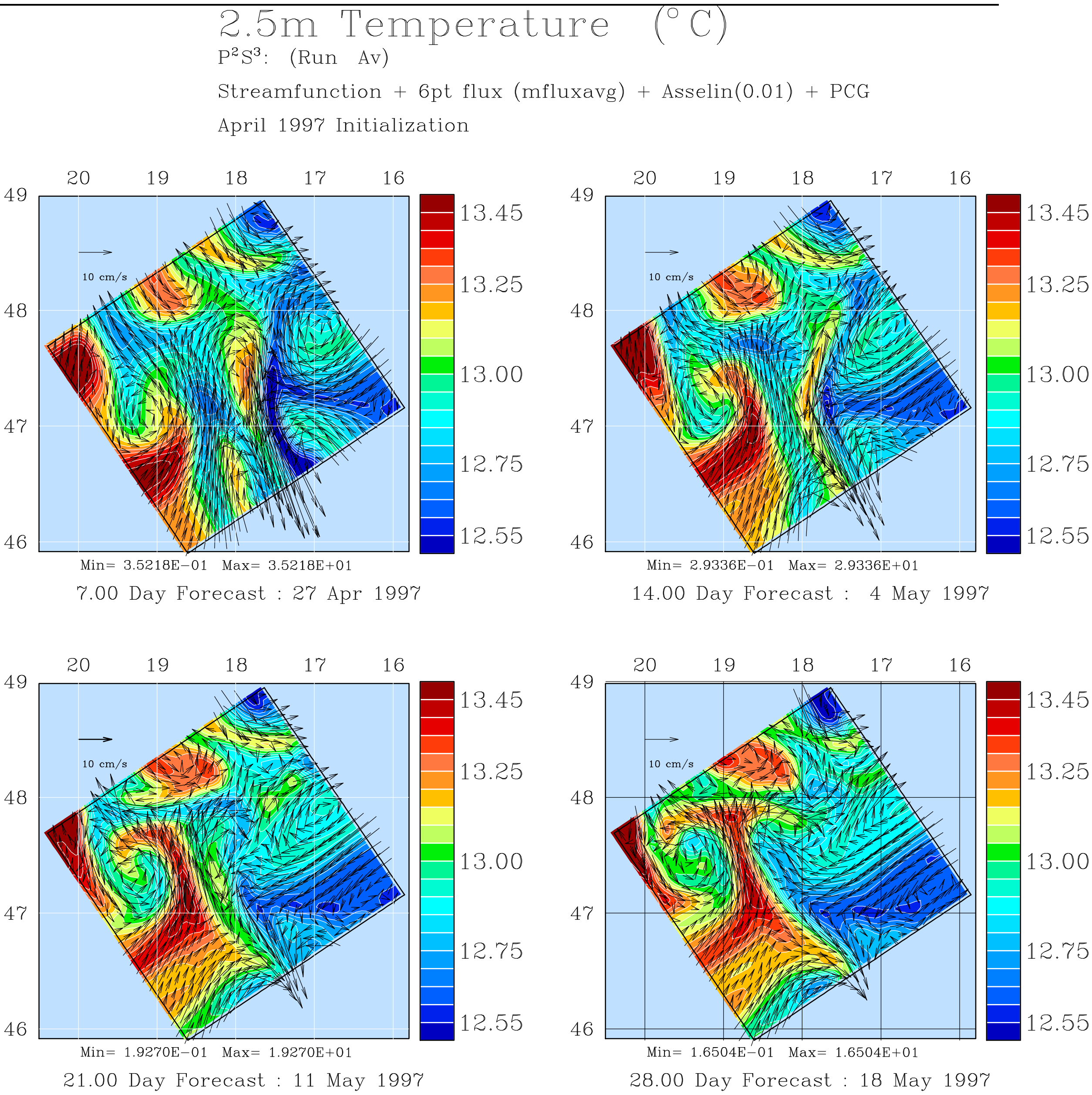


Figure 26 : Streamfunction formulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`). Near surface temperature with total velocity vectors.

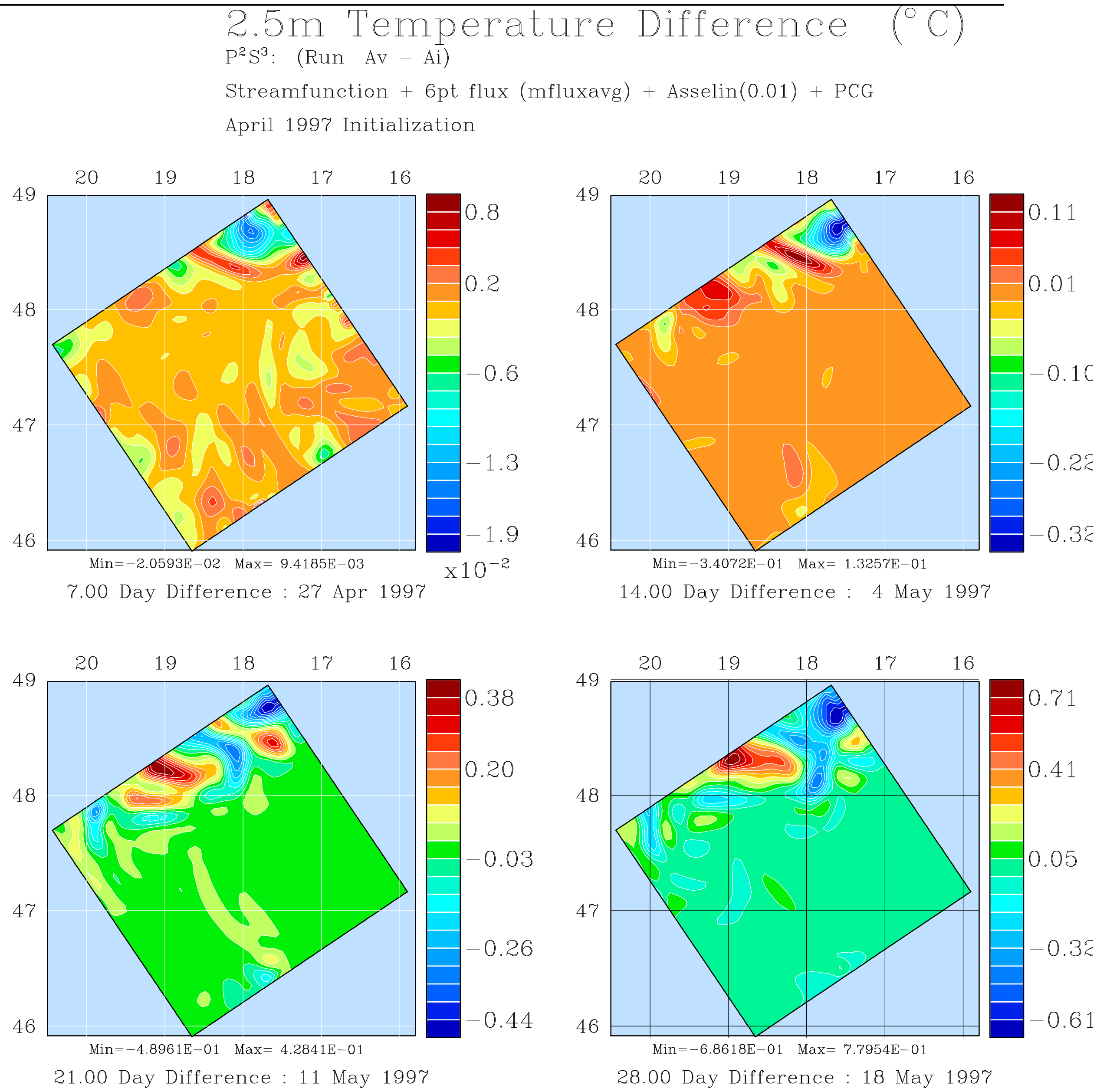


Figure 27 : Near Surface comparison of baseline simulation to simulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`).

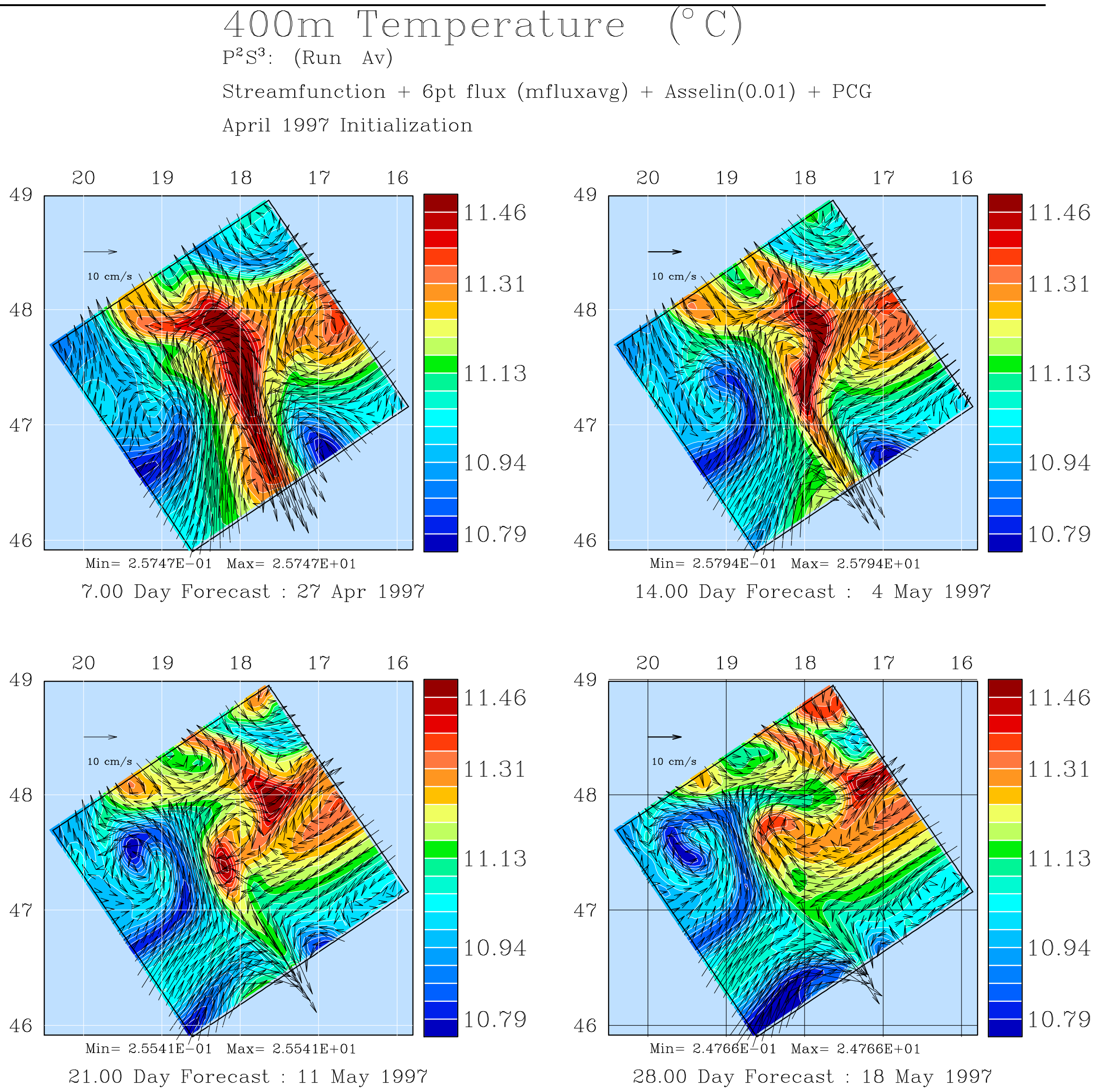


Figure 28 : Streamfunction formulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`). Mid–level temperature with total velocity vectors.

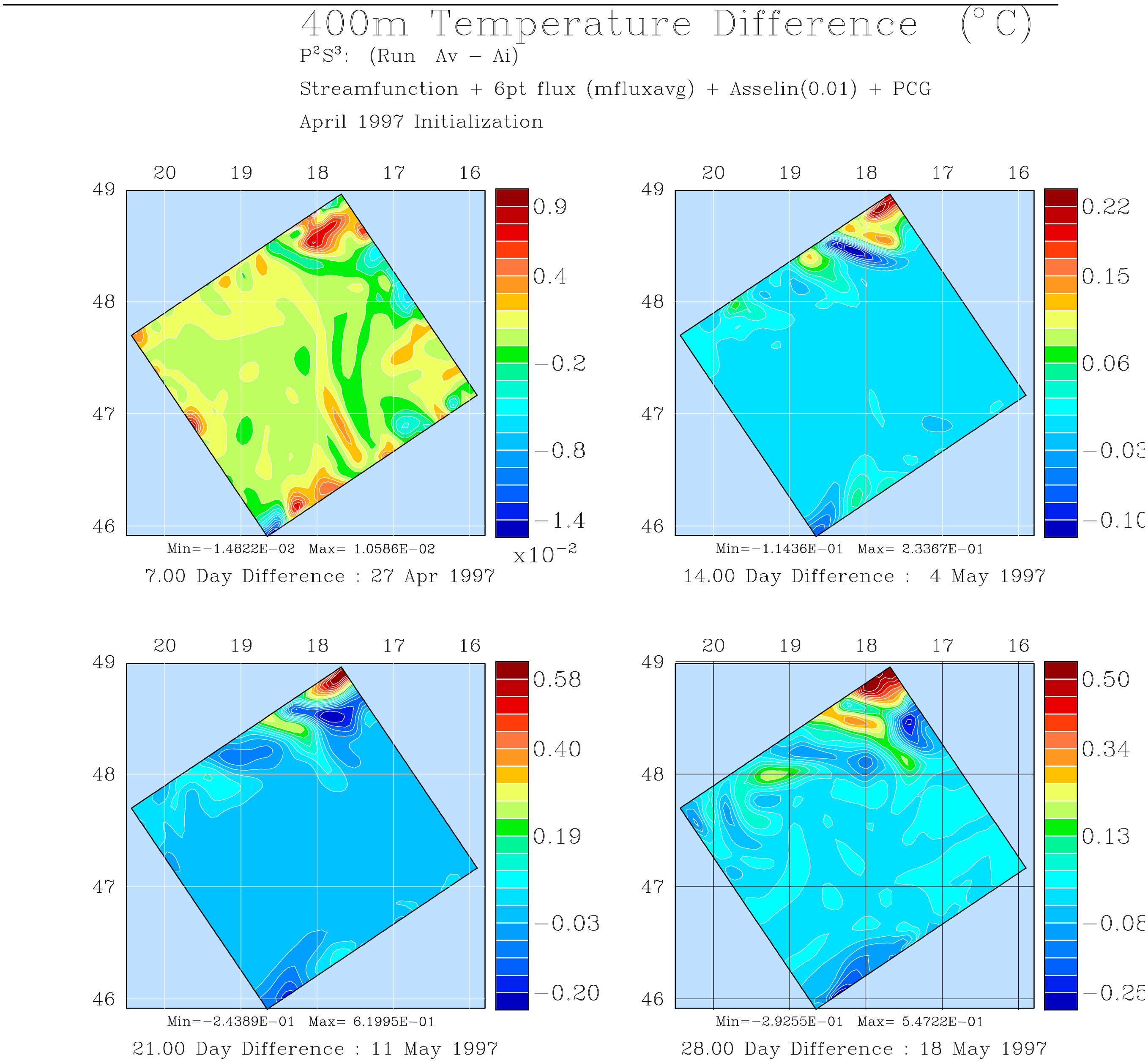


Figure 29 : Mid–level comparison of baseline simulation to simulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`).

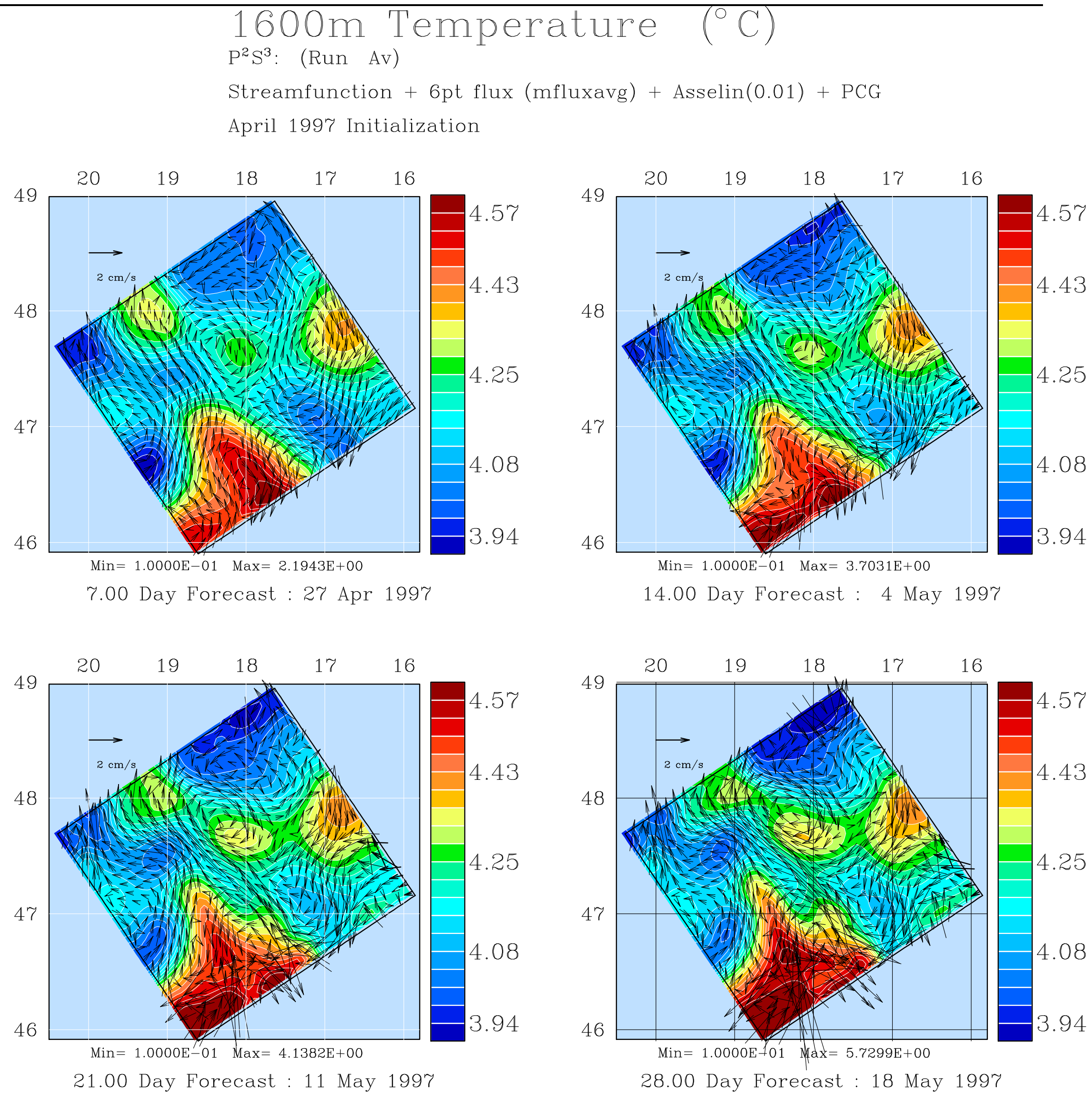


Figure 30 : Streamfunction formulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`). Deep temperature with total velocity vectors.

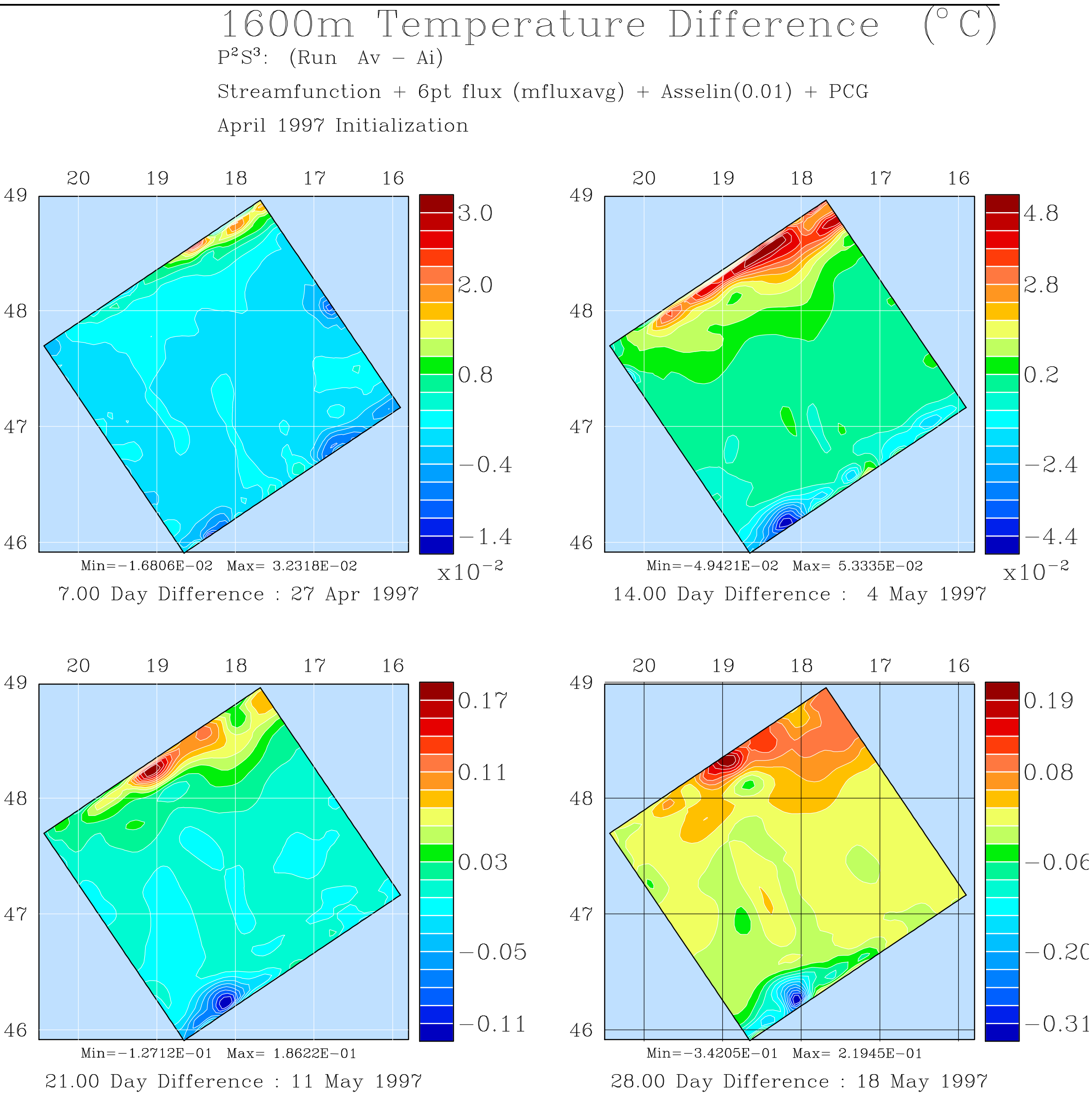


Figure 31 : Deep comparison of baseline simulation to simulation with 6 point flux algorithm, Asselin filter and preconditioned conjugate gradient solver *(v)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`).

(B) Surface Pressure formulation – Rigid Lid

Initialization Procedure

The construction of initial conditions for the surface pressure is based on geostrophy:

$$\hat{k} \times f\vec{u} = \frac{1}{\rho_0}\nabla p \quad . \tag{1}$$

Break-up the pressure into the hydrostatic component and the kinematic surface pressure, p_s,

$$p = \rho_0\, p_s + \int_z^0 g\rho\, d\zeta \quad . \tag{2}$$

Substituting (2) into (1) yields

$$\hat{k} \times f\vec{u} = \nabla p_s + \nabla \int_z^0 \frac{g\rho}{\rho_0}\, d\zeta \quad . \tag{3}$$

Integrating (3) over the water column produces

$$\hat{k} \times fH\bar{\vec{U}} = H\nabla p_s + \int_{-H}^0 \nabla \int_z^0 \frac{g\rho}{\rho_0}\, d\zeta\, dz \tag{4}$$

where $\bar{\vec{U}}$ is the barotropic velocity. Finally, take the divergence of (4) and isolate the kinematic surface pressure,

$$\nabla \cdot (H\nabla p_s) = \nabla \times \left(fH\bar{\vec{U}}\right) - \nabla \cdot \left(\int_{-H}^0 \nabla \int_z^0 \frac{g\rho}{\rho_0}\, d\zeta\, dz\right) \quad . \tag{5}$$

This provides the basic Poisson equation to construct the initial condition for the kinematic surface pressure. The remaining elements are the initial guess (not critical) and the Dirichlet boundary conditions (critical).

(i) Surface Streamfunction (`psurf` $= 0$)
Evaluate (3) at $z = 0$

$$\hat{k} \times f\vec{u}\Big|_{z=0} = \nabla p_s \quad . \tag{6}$$

The initial guess for velocity came from a streamfunction

$$f\vec{u} = \hat{k} \times \nabla\psi \quad . \tag{7}$$

Substituting this into (6) yields

$$\nabla\psi|_{z=0} = \nabla p_s \quad , \tag{8}$$

which, up to an additive constant, implies

$$\psi|_{z=0} = p_s \quad . \tag{9}$$

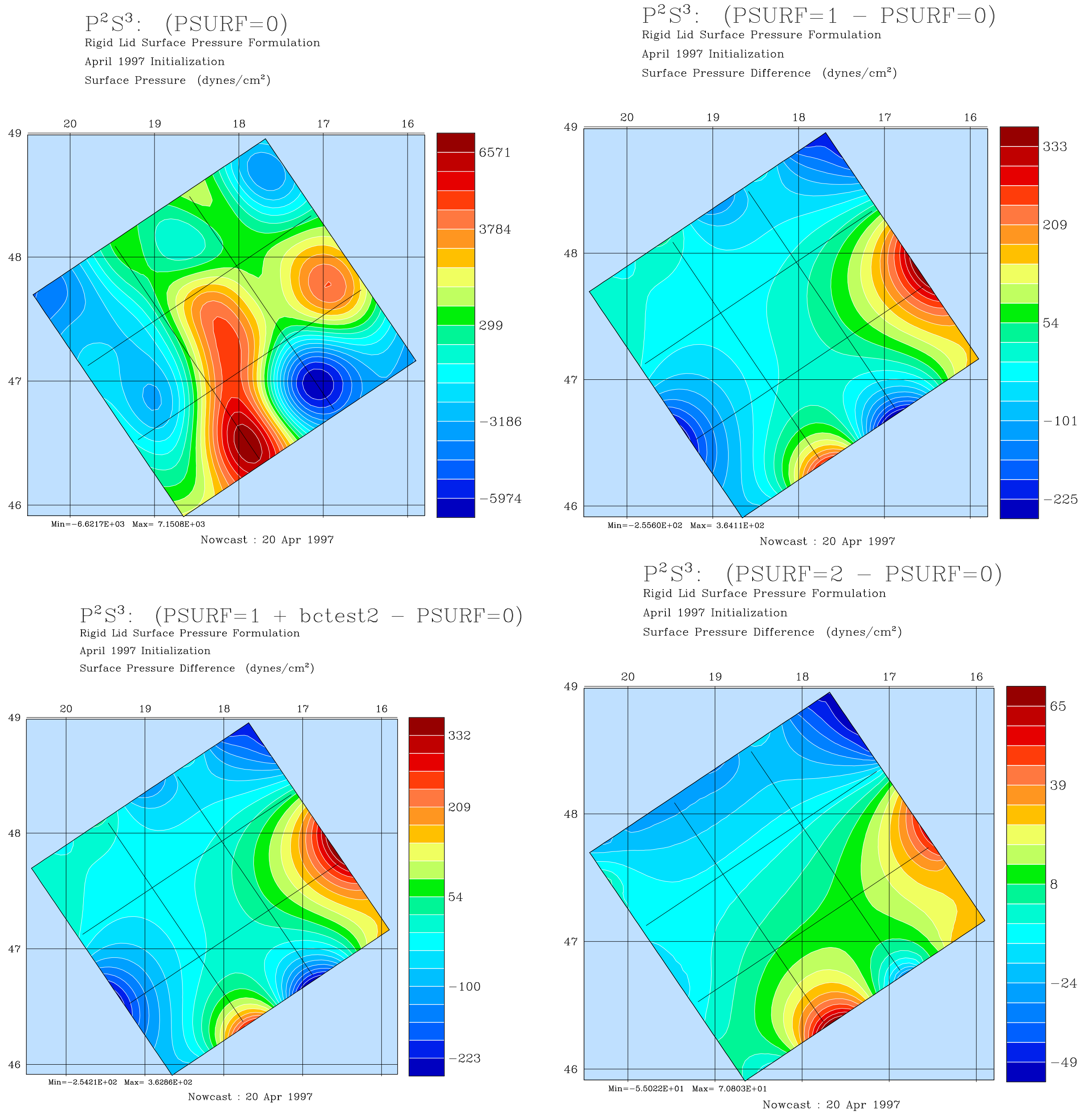


Figure 32 : Comparison of surface pressure initialization schemes in the deep ocean.

Both initial & boundary conditions are obtained from (9)

(ii) Surface Streamfunction & Velocity (`psurf` = 1)
Same initial conditions as (*i*). The boundary conditions are obtained by integrating the tangential derivative of the kinematic surface pressure around the domain. These tangential derivatives are obtained by evaluating (6). This has the advantage of incorporating the divergence constraints built into the transport. It has the disadvantage of complexity, including insuring continuity and handling land (some open issues here).

(iii) Surface Streamfunction & the Barotropic Equation (`psurf` = 1 & CPP option `bctest2`)
Same as (*ii*) **except** that the tangential derivatives are evaluated from equation (4).

Table 1: Speed–up factors computed in a preconditioning parameter study using a random field initial guess.

- Unpreconditioned run time: 0.187 (s)

Left Preconditioning

fill \ τ_{small}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
5	0.755	0.984	0.969	0.957	0.967
10	2.87	2.25	2.05	1.98	1.91
15	2.84	2.36	2.11	1.95	1.95
20	3.06	2.40	1.93	1.71	1.63
25	2.81	2.39	1.85	1.71	1.71
30	2.91	2.47	1.88	1.74	1.71
40	2.94	2.47	1.89	1.69	1.70
50	3.05	2.48	1.93	1.73	1.68
75	2.84	2.41	1.90	1.80	1.73
100	2.93	2.46	1.87	1.71	1.78

Right Preconditioning

fill \ τ_{small}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
5	1.290E-02	1.275E-02	1.265E-02	1.261E-02	1.265E-02
10	2.89	1.84	1.67	1.57	1.53
15	2.93	2.29	1.78	1.66	1.58
20	2.75	2.29	1.49	1.34	1.26
25	2.77	2.32	2.02	1.72	1.64
30	2.69	2.30	1.87	1.70	1.61
40	2.81	2.33	1.86	1.68	1.60
50	2.79	2.44	1.91	1.71	1.63
75	2.93	2.35	1.81	1.73	1.61
100	2.80	2.32	1.93	1.77	1.62

This has the advantage that the boundary conditions are more closely related to the Poisson equation. The disadvantage is that the same complexities in (*ii*) apply here.

(iv) Smooth Approximation (`psurf` = 2)
The barotropic velocity is obtained from a transport streamfunction

$$H\vec{\bar{U}} = \hat{k} \times \nabla\Psi \quad . \tag{10}$$

Substituting (10) into (4) yields

$$f\nabla\Psi = H\nabla p_s + \int_{-H}^{0} \nabla \int_{z}^{0} \frac{g\rho}{\rho_0}\, d\zeta\, dz \tag{11}$$

Shifting some functions across the gradient and rearranging the results leaves:

$$\nabla\left(Hp_s - f\Psi + \int_{-H}^{0}\int_{z}^{0} \frac{g\rho}{\rho_0}\, d\zeta\, dz\right) = p_s\nabla H - \Psi\nabla f + \left(\int_{-H}^{0} \frac{g\rho}{\rho_0}\, d\zeta\right)\nabla H \quad . \tag{12}$$

Table 2: Speed–up factors computed in a preconditioning parameter study using a zero field initial guess.

- Unpreconditioned run time: 0.185 (s)

Left Preconditioning

$fill \setminus \tau_{\text{small}}$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
5	0.742	0.963	0.970	0.947	0.942
10	3.01	2.35	2.00	2.05	1.90
15	2.87	2.36	2.11	1.96	1.87
20	2.89	2.49	2.00	1.68	1.62
25	2.99	2.43	1.86	1.69	1.79
30	2.86	2.37	1.88	1.71	1.68
40	2.78	2.45	1.87	1.72	1.71
50	3.01	2.40	1.86	1.80	1.66
75	2.84	2.40	1.82	1.64	1.72
100	2.86	2.37	1.84	1.80	1.67

Right Preconditioning

$fill \setminus \tau_{\text{small}}$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
5	1.267E-02	1.255E-02	1.251E-02	1.249E-02	1.254E-02
10	2.76	1.81	1.63	1.57	1.52
15	2.79	2.25	1.73	1.67	1.55
20	2.76	2.28	1.47	1.38	1.26
25	2.74	2.26	1.89	1.71	1.65
30	2.76	2.27	1.80	1.70	1.64
40	2.92	2.26	1.95	1.65	1.61
50	2.88	2.31	1.82	1.71	1.60
75	2.88	2.26	1.82	1.67	1.61
100	2.72	2.23	1.86	1.74	1.65

Assuming the terms on the right-hand side are small compared to those on the left, then, up to an additive constant, (12) implies

$$p_s = \frac{f}{H}\Psi - \frac{1}{H}\int_{-H}^{0}\int_{z}^{0}\frac{g\rho}{\rho_0}\,d\zeta\,dz \quad . \tag{13}$$

The initial & boundary conditions are both obtained from (13). This has the advantage that both are closely related to the Poisson equation. It has the disadvantage of the approximation in going from (12) to (13).

A comparison of these initialization schemes is presented in figure 16. The first observation is all schemes produce approximately the same results. Differences plots are required to see the changes. The results seem to be paired [(*i*) & (*iv*)] and [(*ii*) & (*iii*)] in terms of similarity. This is probably **not** due to the closure schemes in (*ii*) & (*iii*), those should

Table 3: Iterations used in a preconditioning parameter study using a random field initial guess and left conditioning.

- Unpreconditioned iteration count: 160

$fill \setminus \tau_{\text{small}}$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
5	95	70	70	70	70
10	18	15	15	15	15
15	18	10	8	8	8
20	18	10	6	6	6
25	18	10	6	4	3
30	18	10	6	4	3
40	18	10	6	4	3
50	18	10	6	4	3
75	18	10	6	4	3
100	18	10	6	4	3

---

only induce a relative change of $\sim 2\left(10^{-3}\right)$ according to the log files. A consequence of this pairing is that we need examine at most 2 cases. We arbitrarily choose (*i*) & (*iii*).

Tuning Preconditioning Parameters

The tuning of the preconditioning parameters for conjugate gradient solver in the surface pressure formula is more developed (easier to explore parameter space). The parameter study was done using an absolute tolerance of 10^{-25} and a relative tolerance of 10^{-10}. The parameters studied were

- Choice of left or right preconditioning
- Small value tolerance, τ_{small}
- Fill–in bandwidth, $fill$

The resulting speed–up factors are presented in tables 1–2.

The general observations of this study are:

- The choice of initial guess does **not** greatly affect the speed–up.
- Left preconditioning is generally more effective than right preconditioning. In fact, right preconditioning with a fill–in bandwidth of 5 failed to converge in this study.

Balancing the speed–up with economy (program storage requirements), the following preconditioning parameters were chosen.

- Left preconditioning
- Small value tolerance, τ_{small}, of 10^{-2}
- Fill–in bandwidth, $fill$, of 20.

Table 3 shows how the number of iterations required to converge vary with the preconditioning parameters. Comparing with table 1, preconditioning speeds up the computations by a factor of 2–3 but the number of iterations is reduced by a factor of 9–50. This suggests a significant cost associated with preconditioning. Some possibilities are:

- Constructing the preconditioning matrix
- Multiplication by the preconditioning matrix
- Solving the resulting matrix

Of these, the first is the most interesting. The code, as I currently understand it, constructs this matrix at every time step. If this is true and if the construction is the expensive part, then the code should be reworked to only compute this preconditioning matrix once or intermittently. This becomes especially important in light of the streamfunction results.

(iii) Surface Streamfunction & the Barotropic Equation Initialization (`psurf` = 1 & CPP option `bctest2`).

The simulation which most nearly approximated the streamfunction KE was obtained by applying Orlanski boundary conditions to both the integrated right-hand side and surface pressure; a 2,2,1 Shapiro filter to the augmented velocity ($\hat{U}$); and a 32,1,1 Shapiro filter to the surface pressure. Figure 33 shows that the kinetic energy stays in the same ballpark as the equivalent streamfunction simulation, but with an increasing oscillation absent in the streamfunction case.

The statistics in figure 34 are even more telling. The RMS difference for tracer fields shows a consistent advantage of the simulation over persistence, although not as decisive as between the streamfunction formulations. The surface and mid–level velocities also show similar results but the deep velocities are pretty much equivalent to persistence. The pattern correlation is pretty strong in the first half of the simulation, but show a marked down–turn at day 15. All these statistics show more rapid initial growth than *(Aii–v)*.

The selected fields and differences (figures 35–40) show that the changes are spread throughout the domain. A "head–on collision" between 2 broad flows in the streamfunction formulation is more glancing here, with more resulting eddy "roll–up".

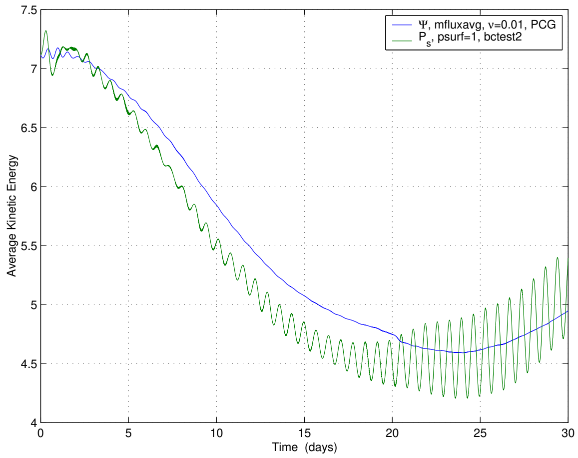


Figure 33 : Kinetic Energy comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

RMS Error **Pattern Correlation Coefficient**

T

S

U

V

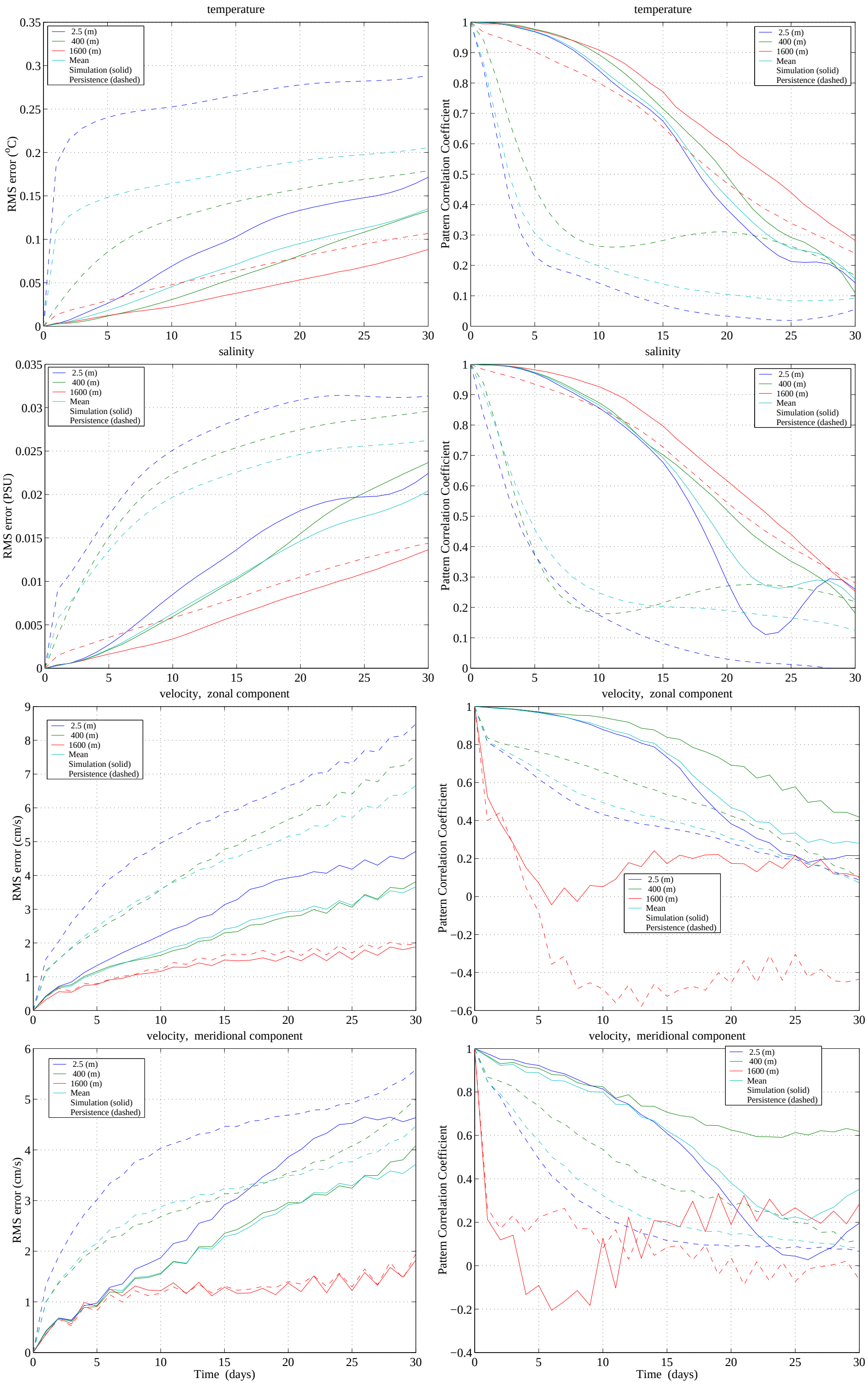


Figure 34: Comparison of "most similar" rigid–lid streamfunction simulation *(Av)* to rigid–lid surface pressure simulation *(Biii)*.

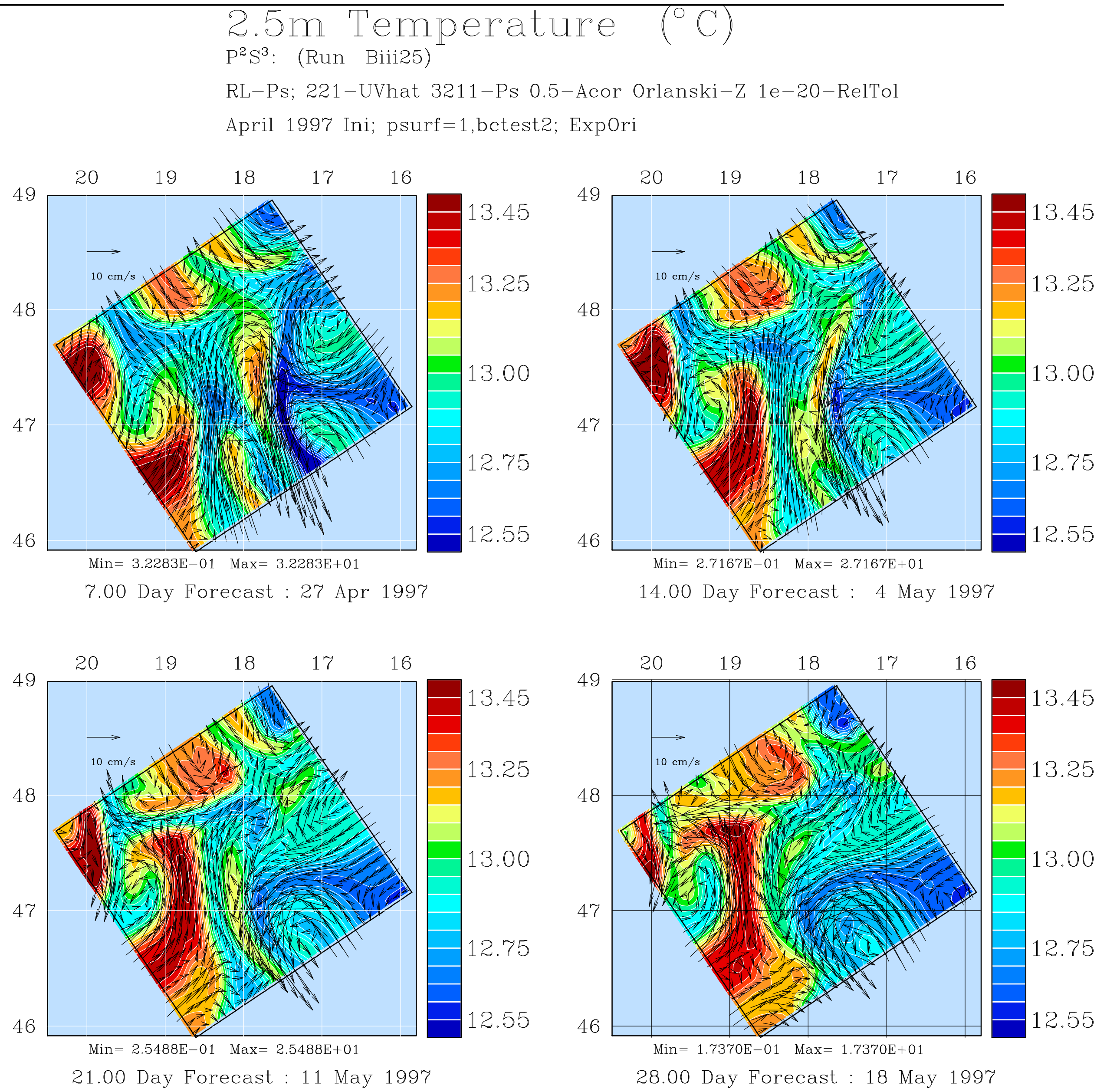


Figure 35 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf=1`, PE_initial CPP option `bctest2`). Near surface temperature with total velocity vectors.

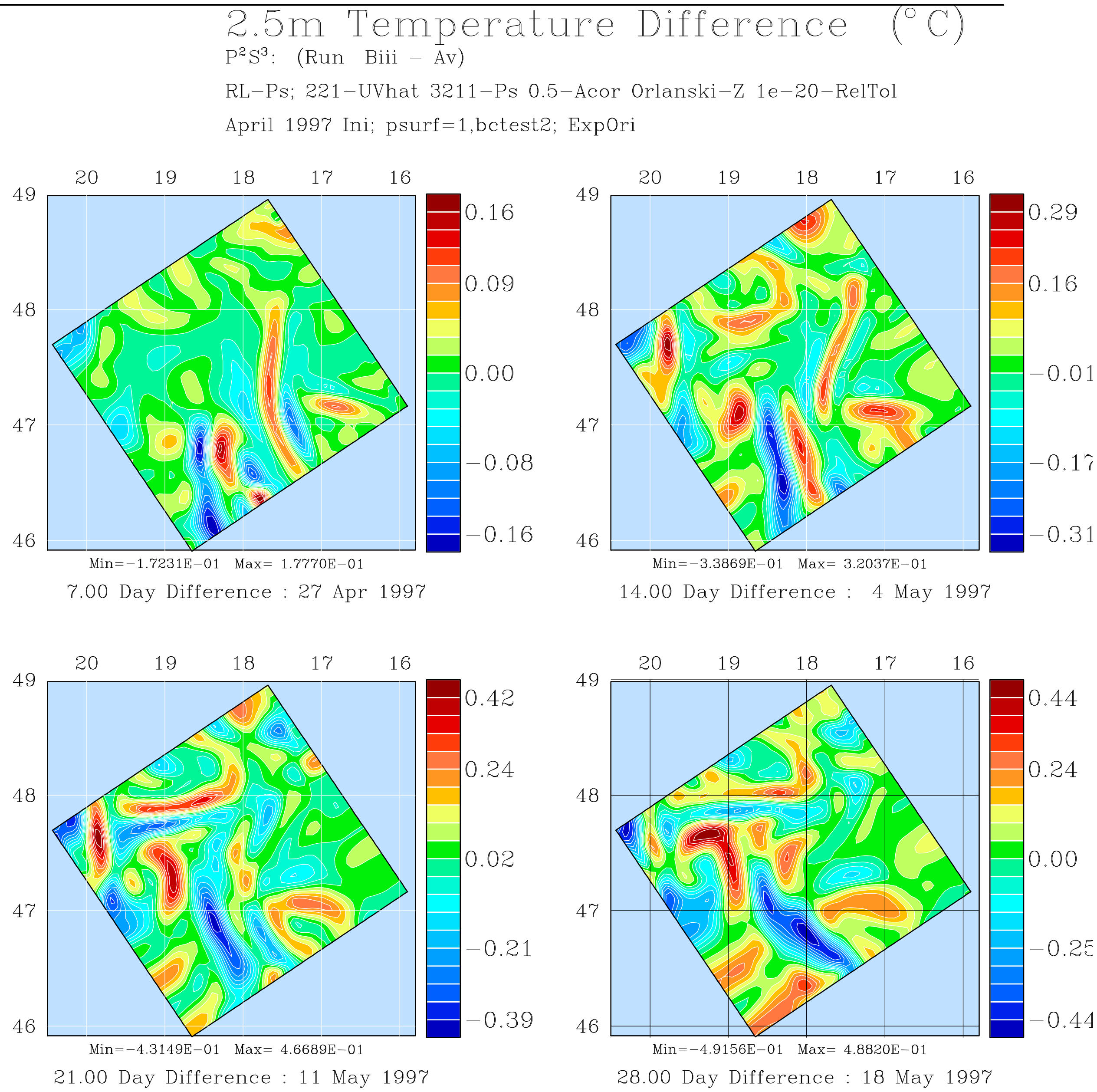


Figure 36 : Near Surface comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

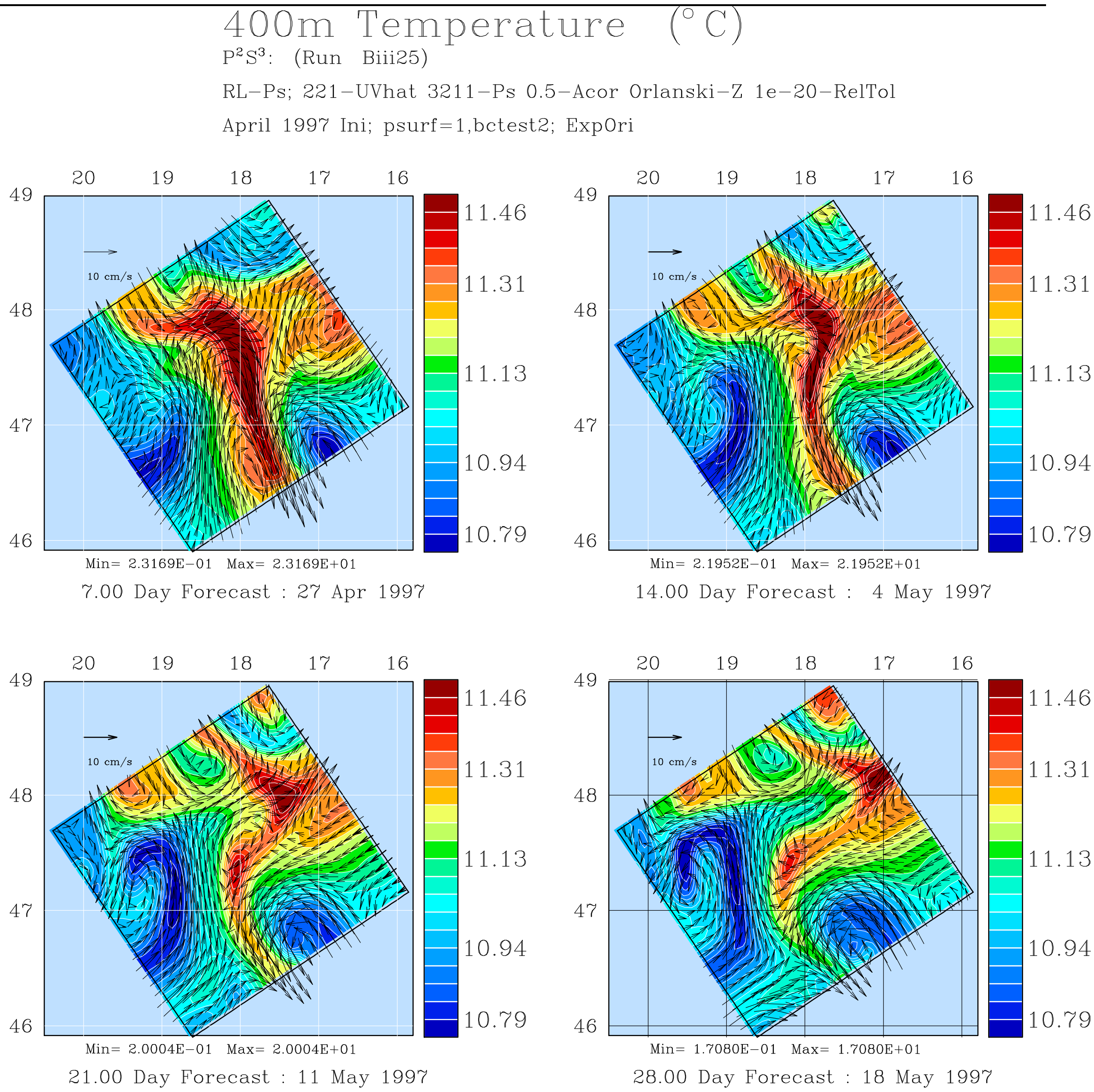


Figure 37 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`). Mid–level temperature with total velocity vectors.

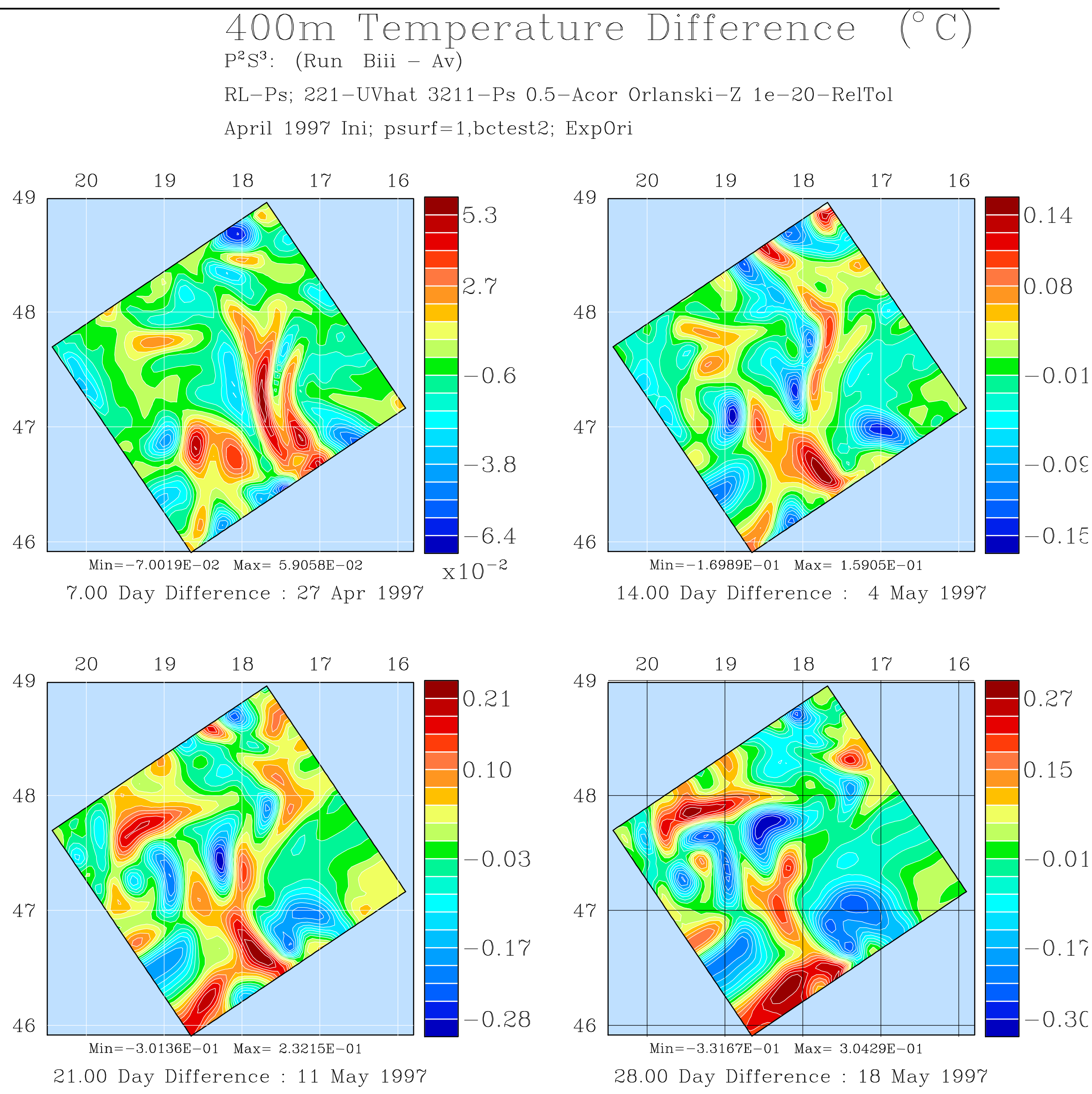


Figure 38 : Mid–level comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

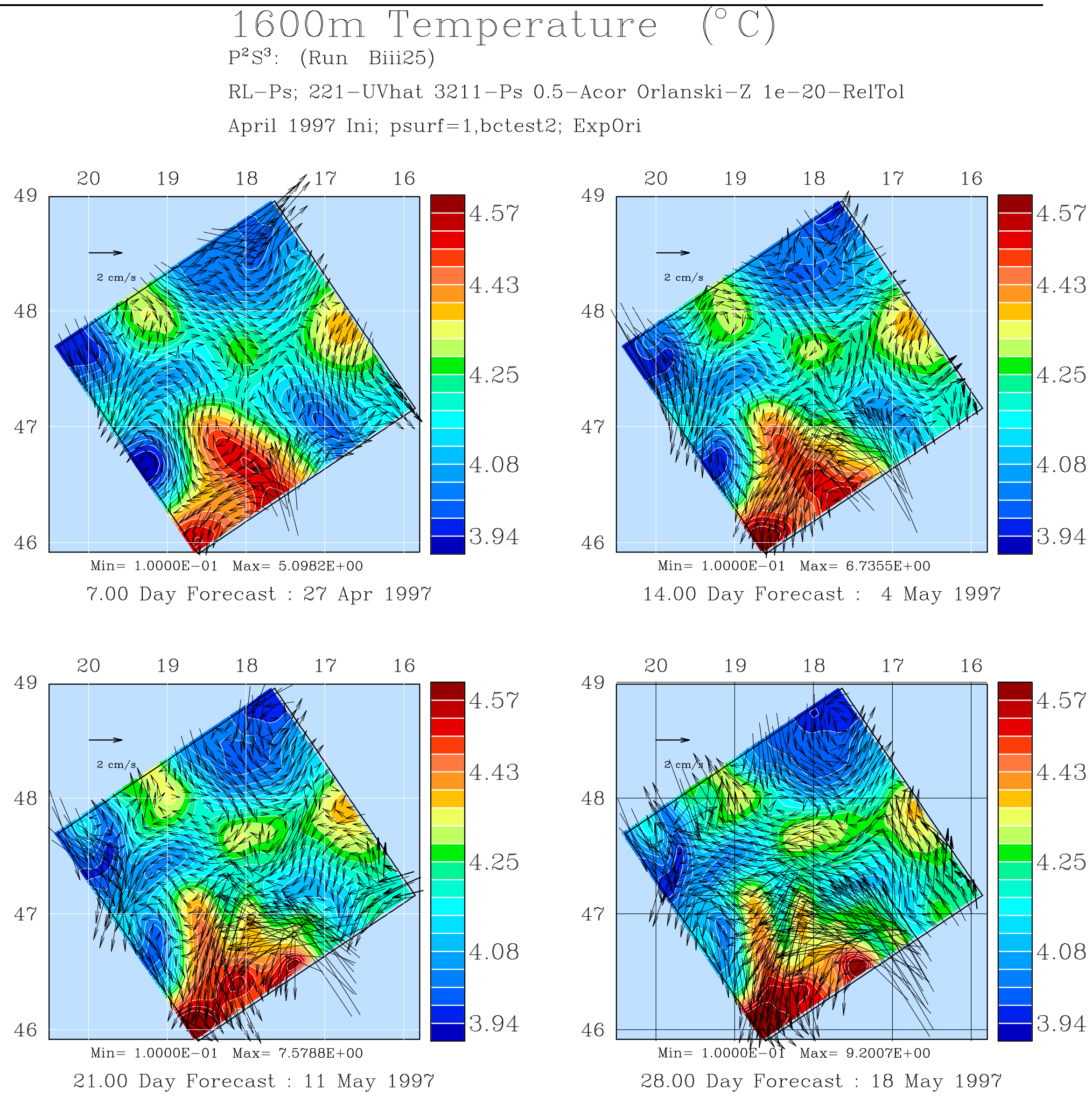


Figure 39 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`). Deep temperature with total velocity vectors.

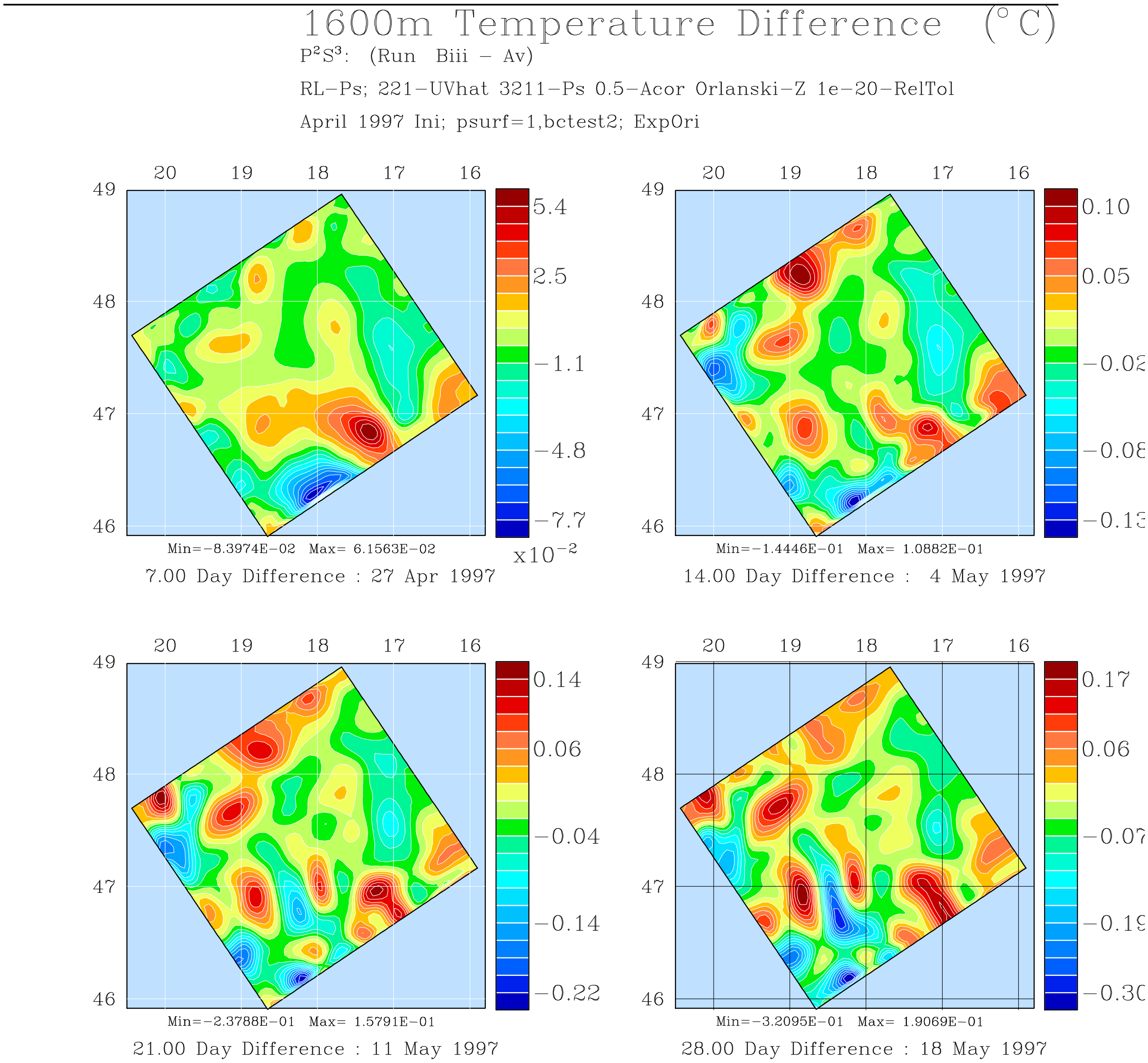


Figure 40 : comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

(i) Surface Streamfunction Initialization (`psurf` = 0).

Although listed first, this simulation was performed after *(iii)* and uses the same parameters. Figure 41 shows that the kinetic energy is overall similar to *(iii)* but with small oscillations and with less growth at the end of the simulation.

Figures 42–48 show similar statics and fields to case *(iii)*.

The head–to–head comparison between *(i)* and *(iii)* is a mixed bag. The average kinetic energy in *(i)* is cleaner. The RMS and PCC give the slight edge to *(iii)* in most (not all) cases.

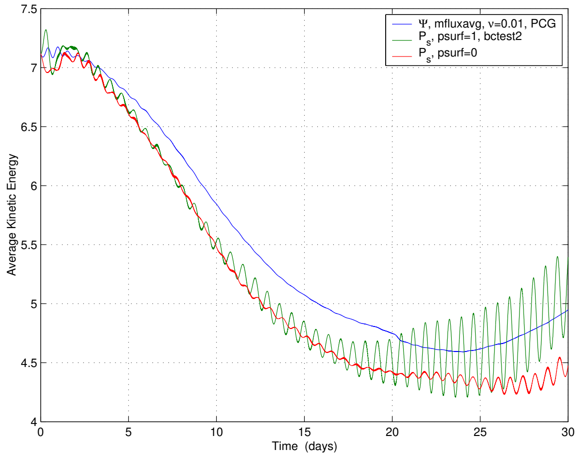


Figure 41 : Kinetic Energy comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (`psurf`=0).

RMS Error **Pattern Correlation Coefficient**

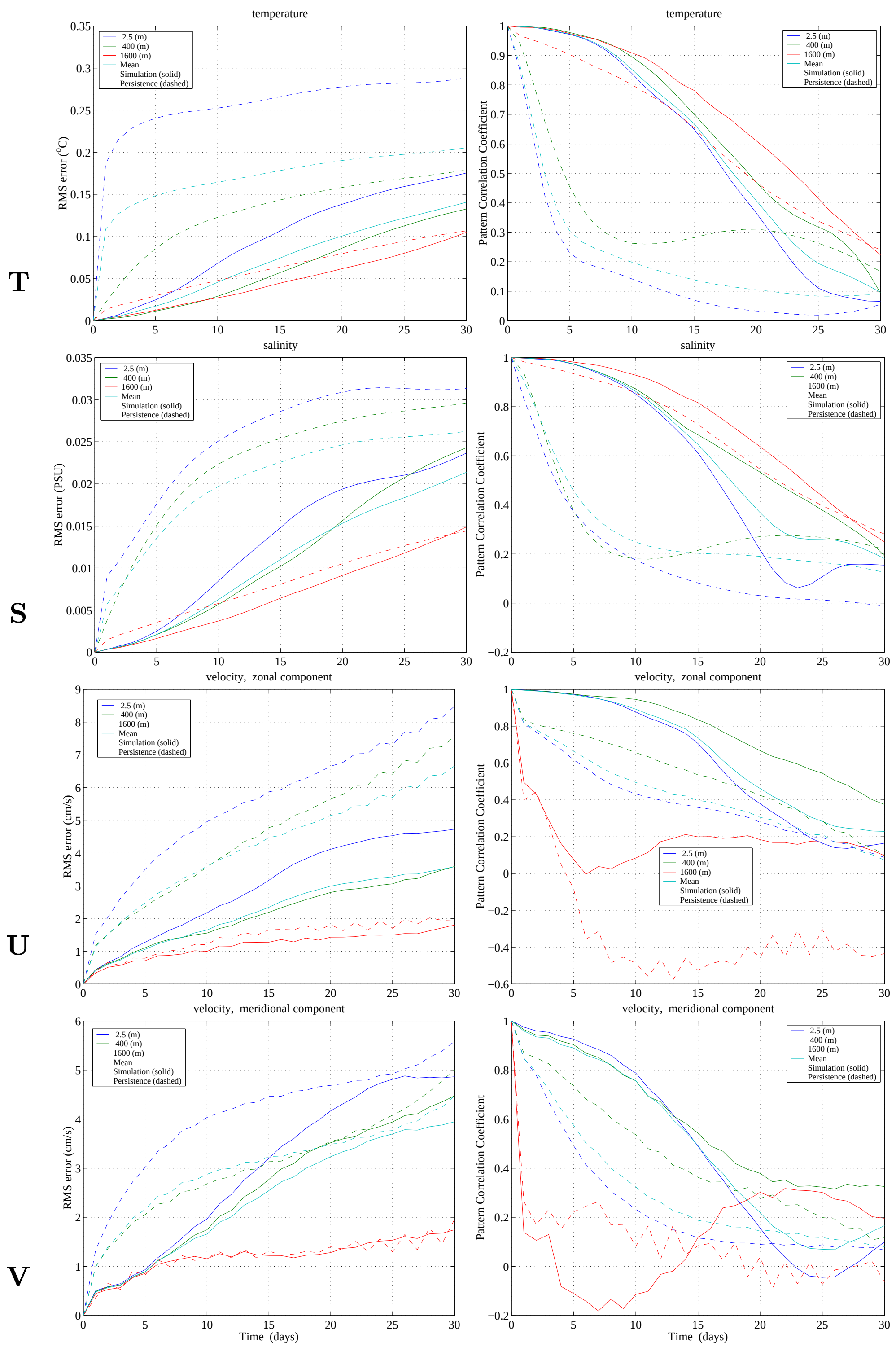


Figure 42: Comparison of "most similar" rigid–lid streamfunction simulation *(Av)* to rigid–lid surface pressure simulation *(Bi)*.

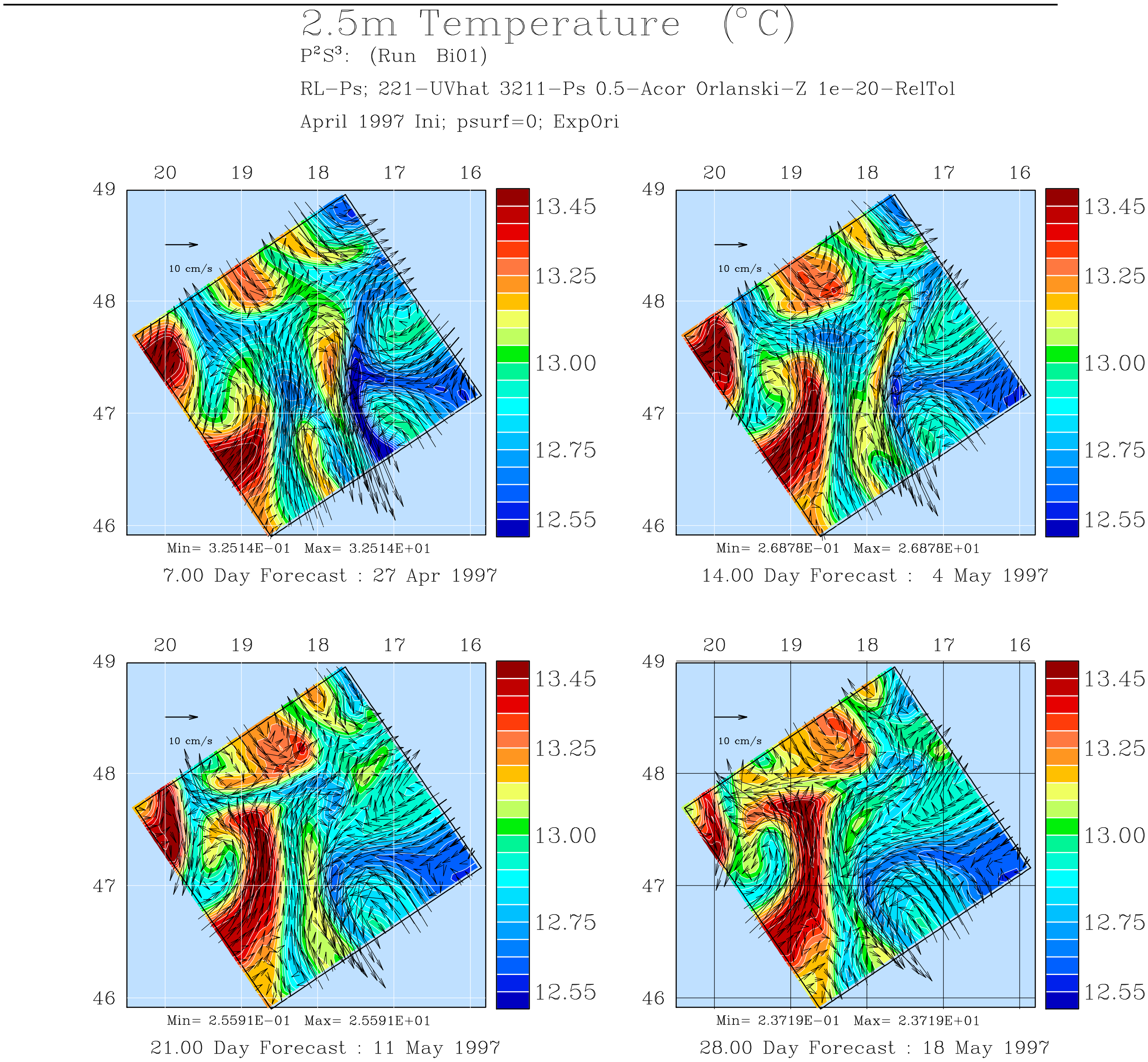


Figure 43 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (`psurf`=0). Near surface temperature with total velocity vectors.

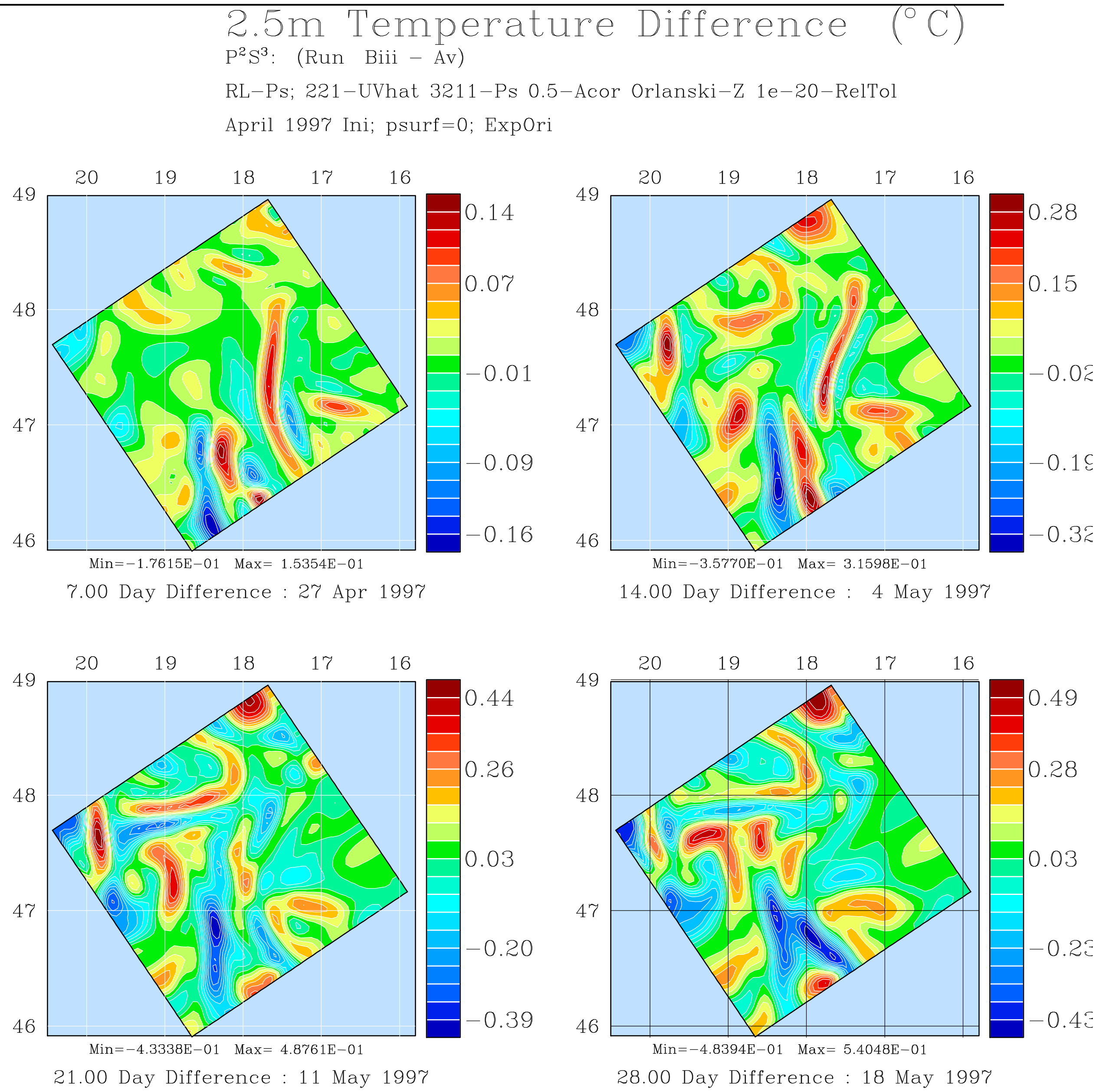


Figure 44 : Near Surface comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options **mfluxavg**, **AsselinFilt** + **pcg5**) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (**psurf**=0).

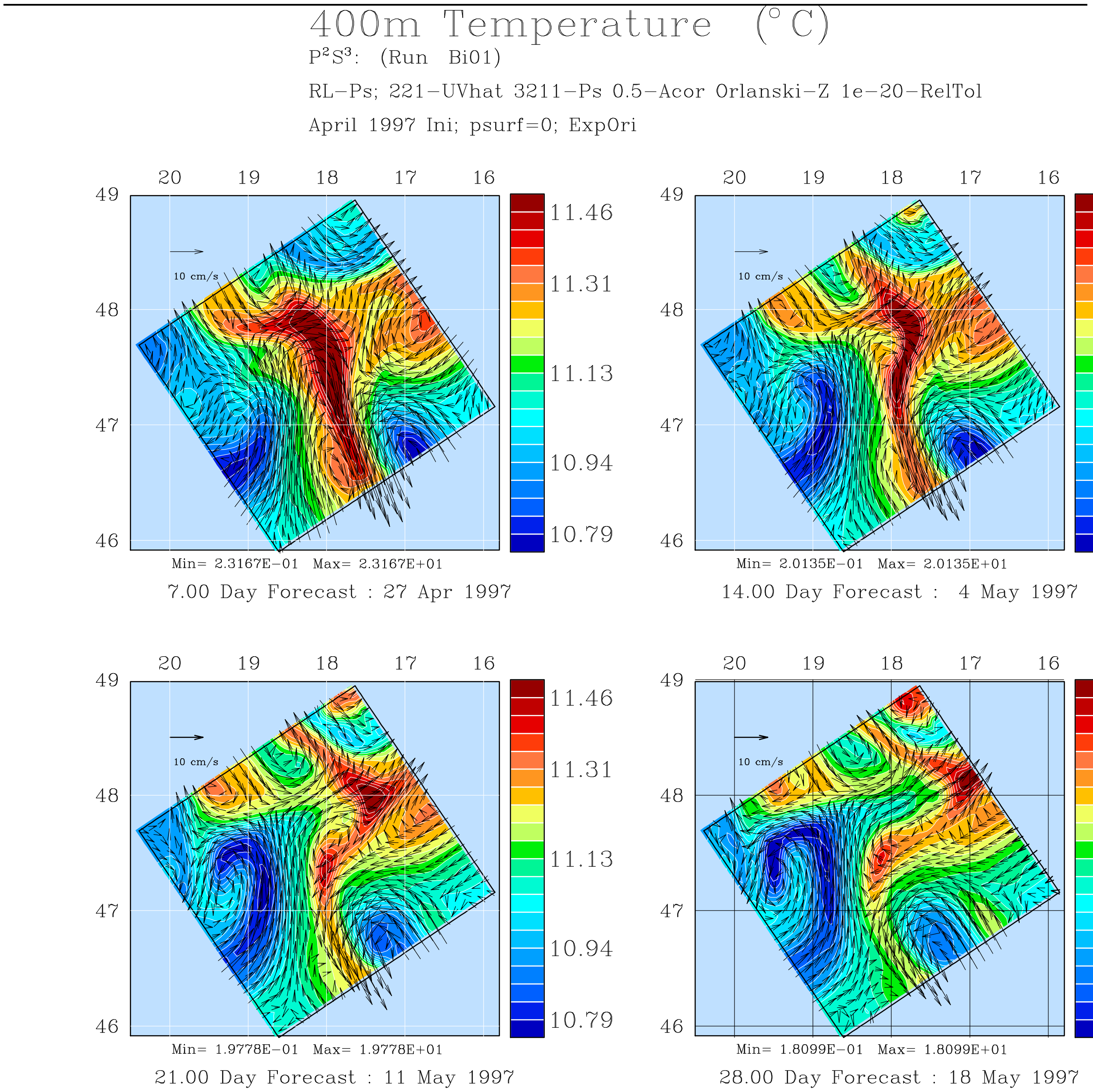


Figure 45 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (`psurf`=0). Mid–level temperature with total velocity vectors.

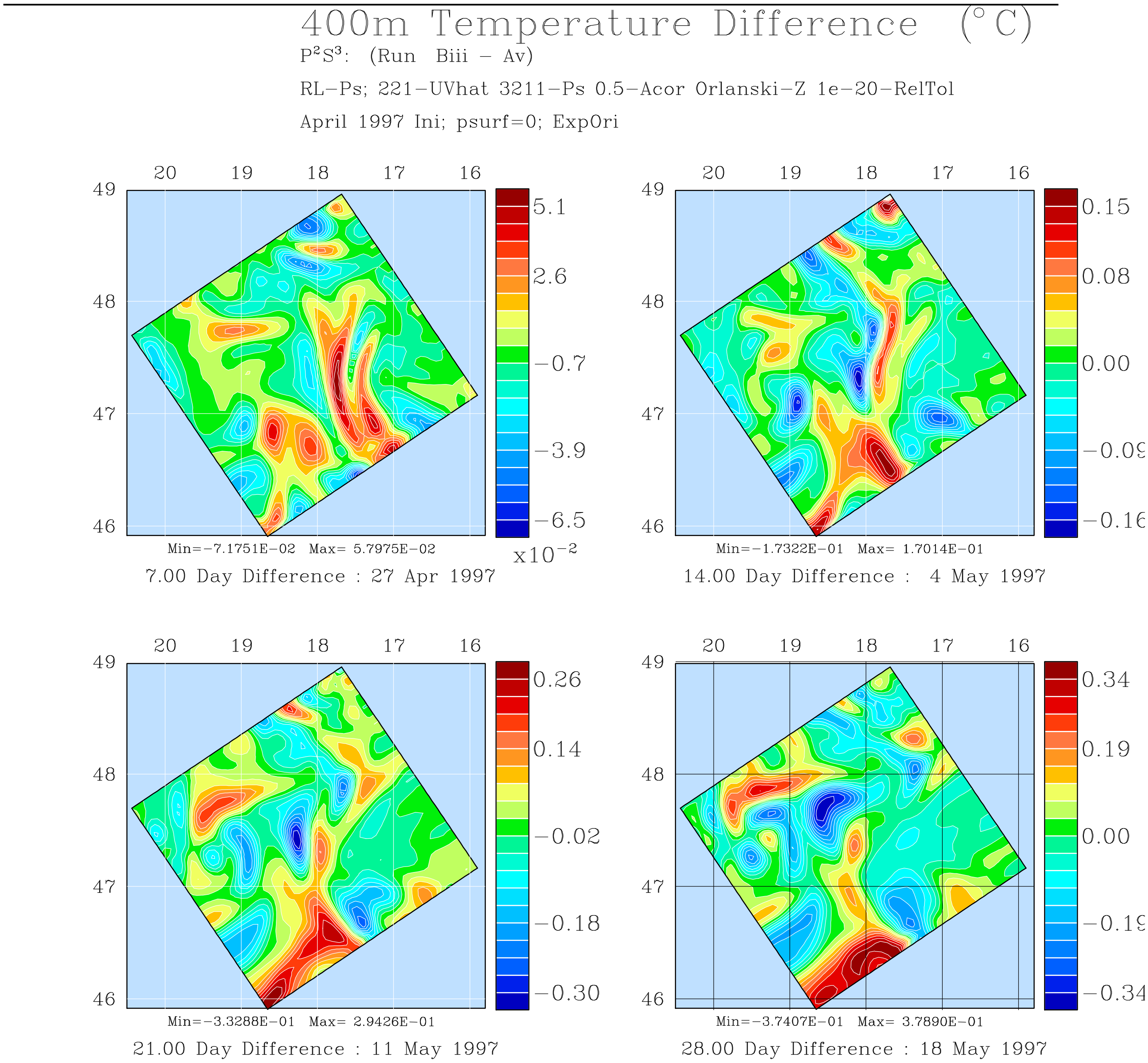


Figure 46 : Mid–level comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (`psurf`=0).

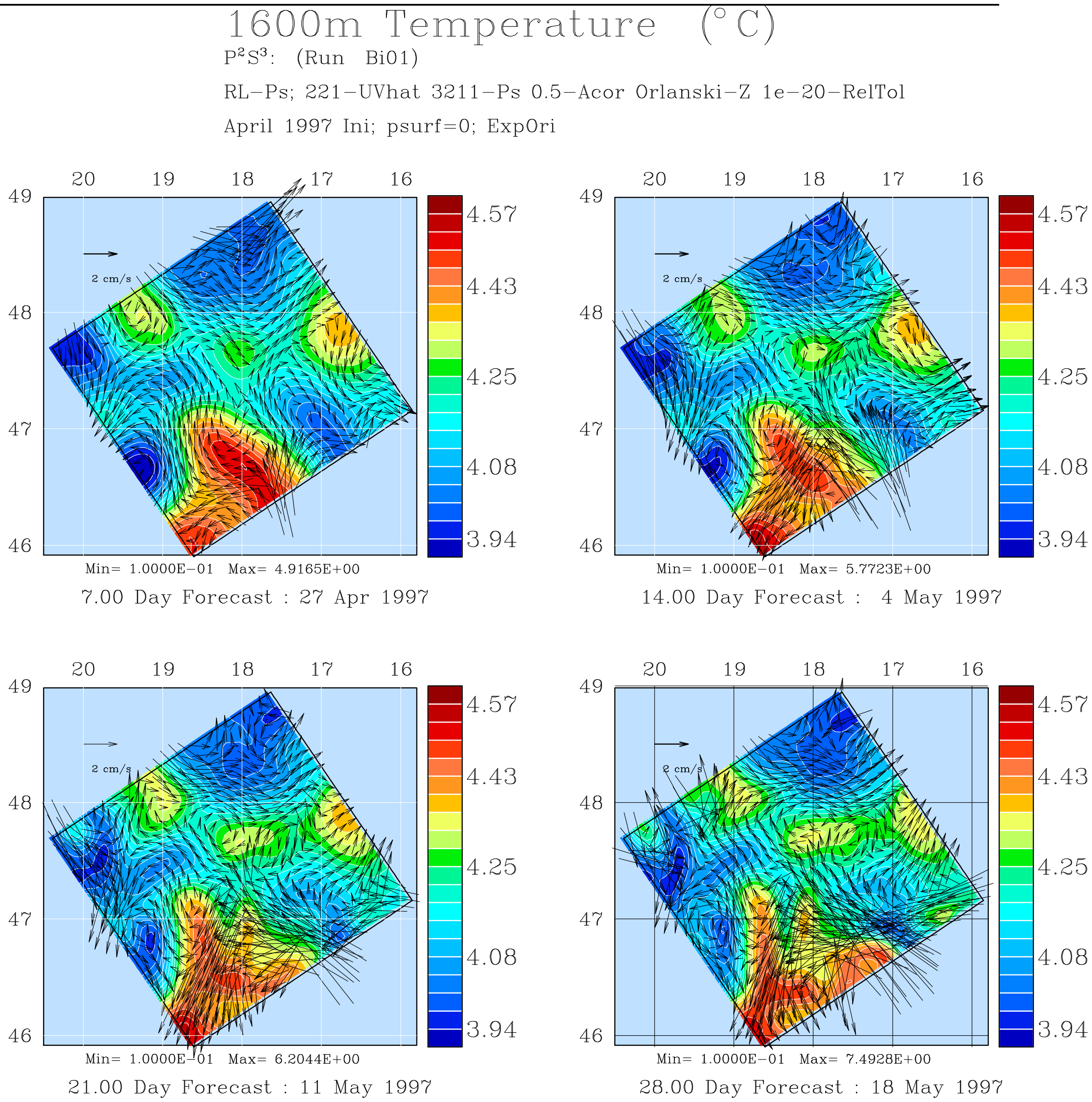


Figure 47 : Comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (`psurf`=0). Deep temperature with total velocity vectors.

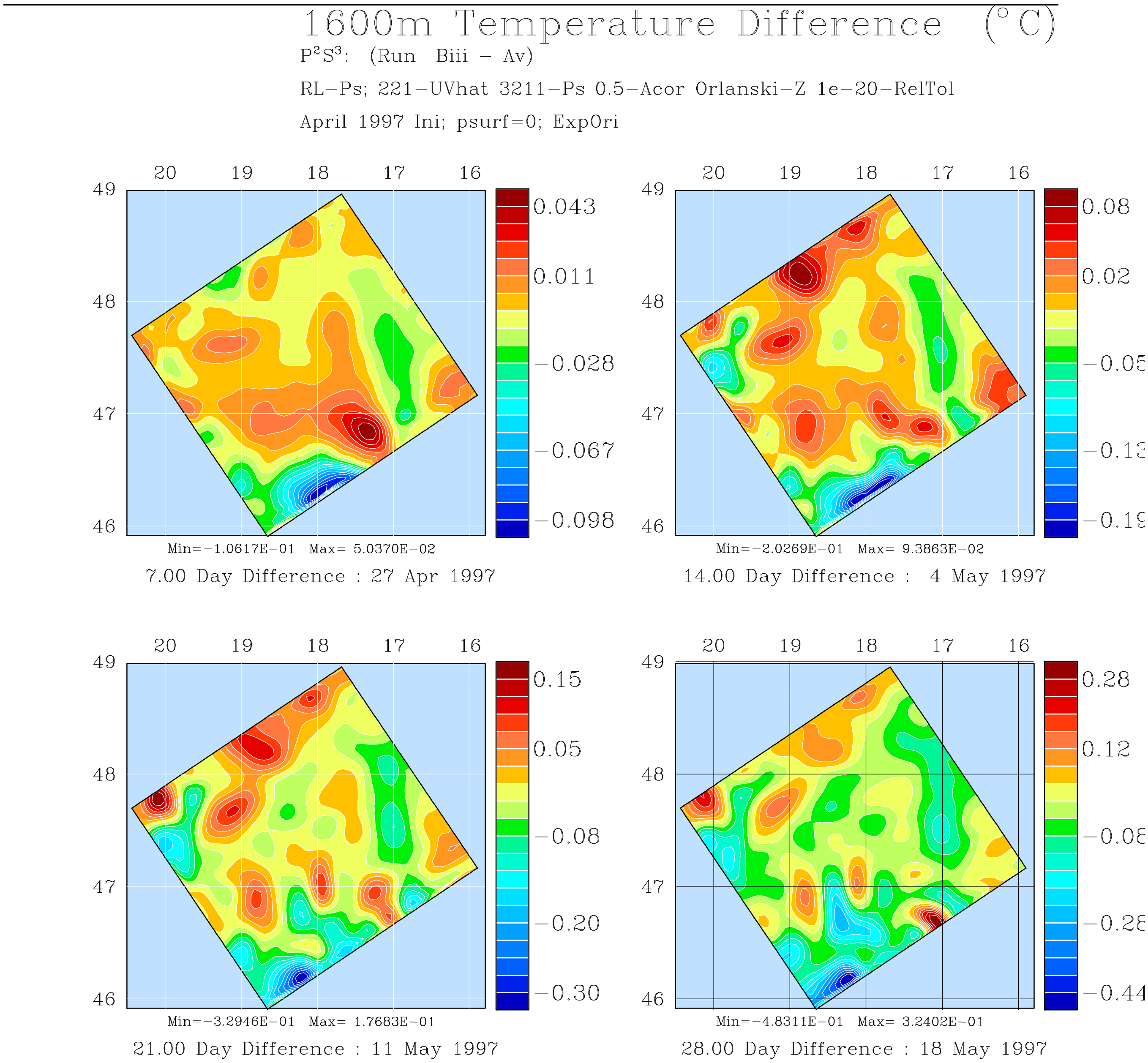


Figure 48 : Deep comparison between transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options **mfluxavg**, **AsselinFilt** + **pcg5**) versus rigid lid, surface pressure from initialization with surface streamfunction BCs (**psurf**=0).

(C) Surface Pressure formulation – Free Surface *(i–iv)* as in *(B)*.

Although four (4) cases were originally planned, after looking at the initializations in *(B)* only two were felt necessary. These simulations repeat the rigid lid simulation except that the semi–implicit Coriolis parameter is set to $\frac{1}{3}$, following the stability analyses of Dukowicz & Smith.

One note: at press time, the plotting and skill metric codes were ignoring the effect of the free surface on the model levels. For this test, the free surface variability is in the range $\pm$ 7 cm and so is not a major effect.

(i) Surface Streamfunction Initialization (`psurf` $= 0$).

Figure 49 shows that the kinetic energy is overall similar to *(Bi)* but with larger oscillations and with more growth at the end of the simulation.

Figure 50 shows the RMS and PCC comparisons between the equivalent rigid–lid and free surface pressure formulations. Comparing this to figure 42, the surface pressure formulations maintain a greater degree of correlation to each other than they do to the streamfunction formulation.

Figures 51–56 show the simulation fields for this run and the difference fields between this simulation and the equivalent rigid–lid case. The differences are still in the interior, but now more concentrated near the major flow gradients.

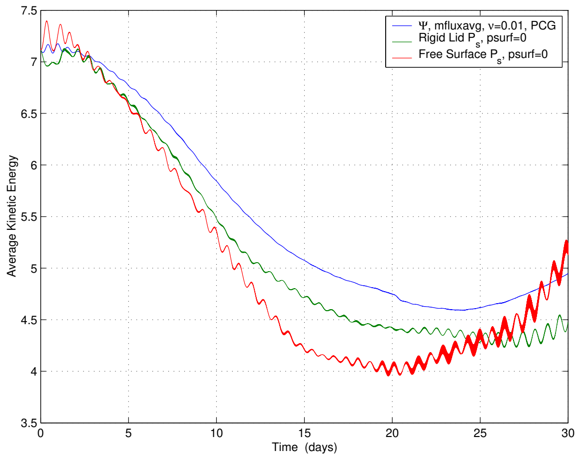


Figure 49 : Kinetic Energy comparison between

- transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`)
- rigid lid, surface pressure from initialization with surface streamfunction BCs *(Bi)* (`psurf=0`).
- free surface, surface pressure from initialization with surface streamfunction BCs *(Ci)* (`psurf=0`).

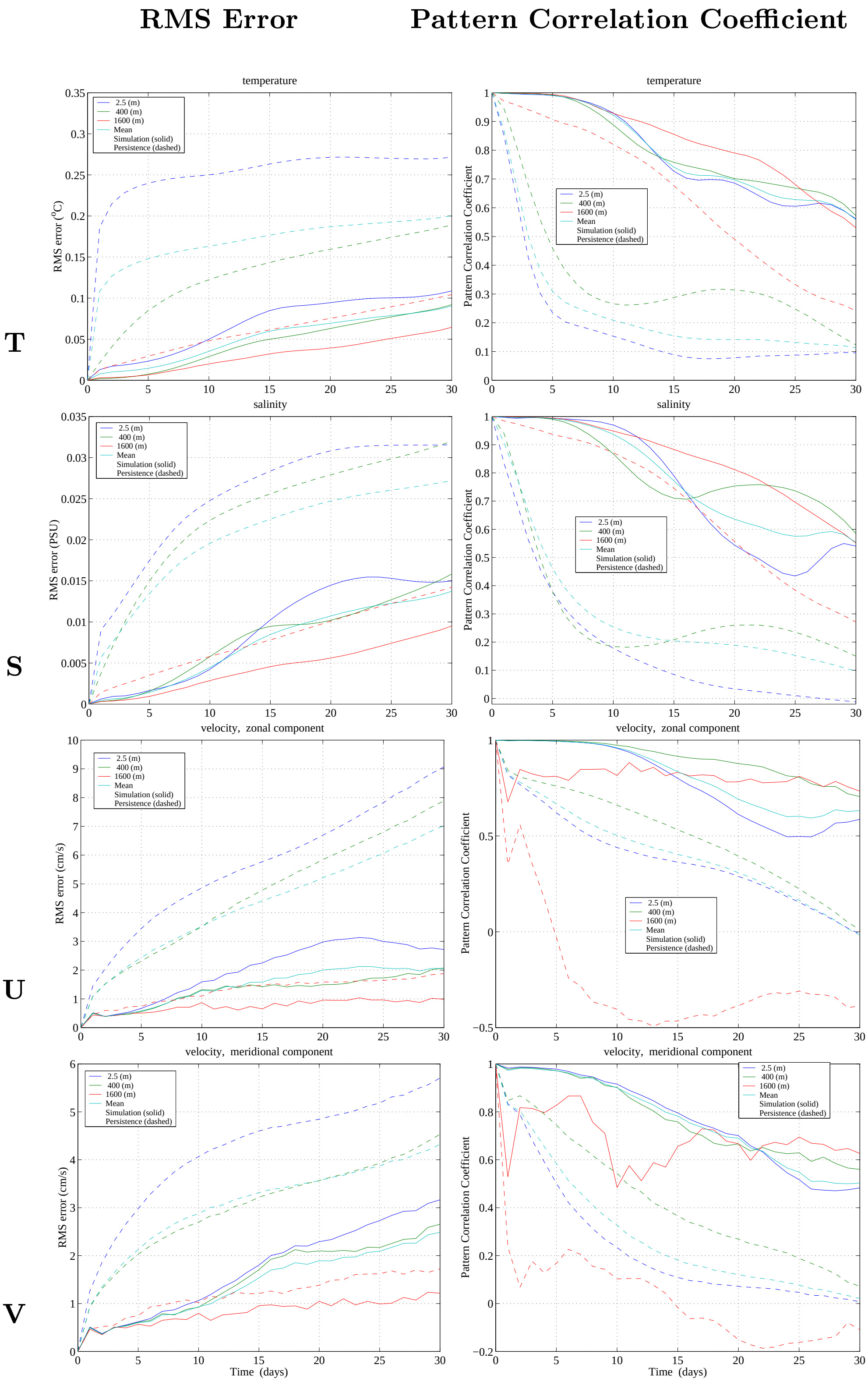


Figure 50: Comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0).

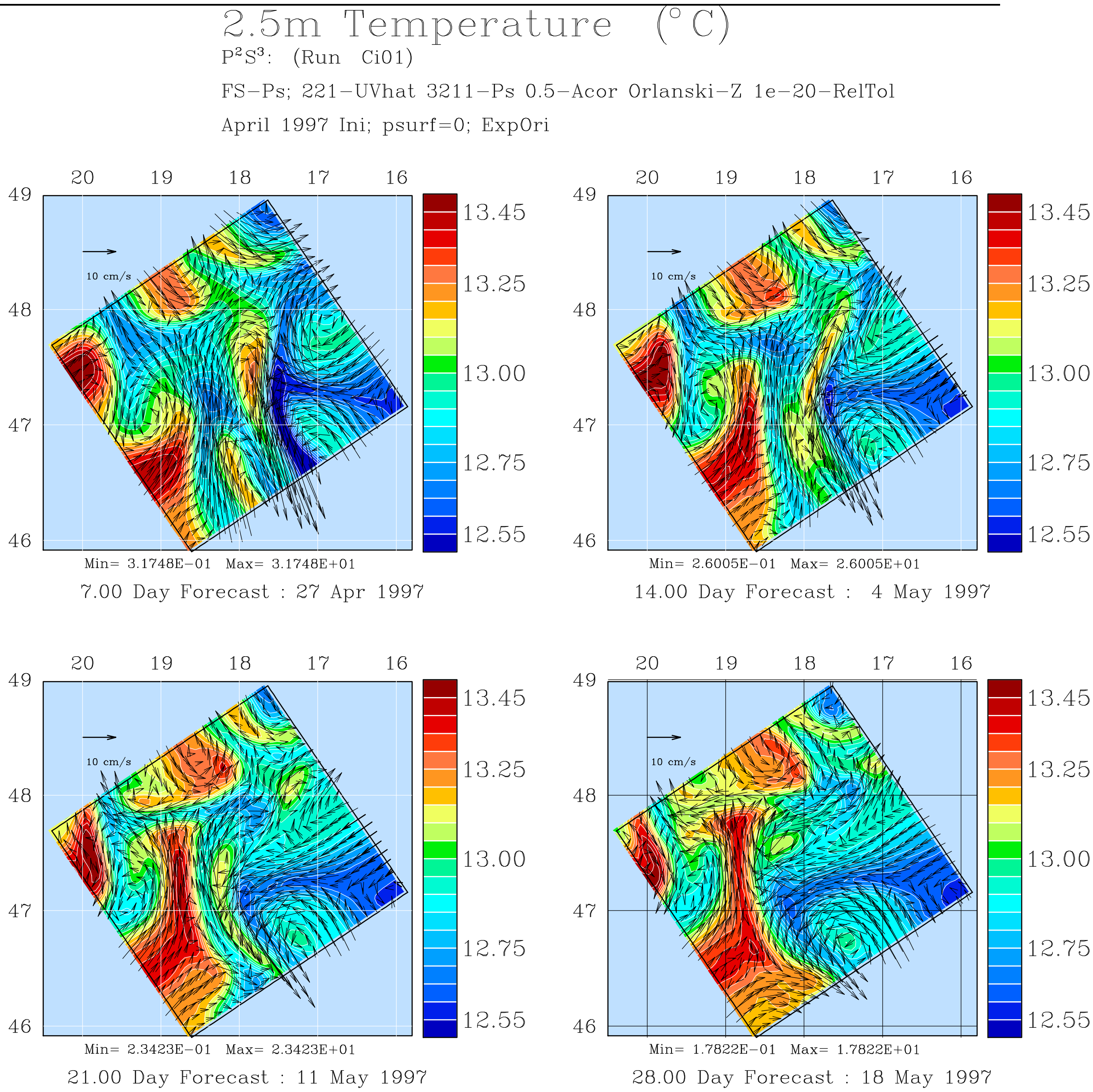


Figure 51 : Comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0). Near surface temperature with total velocity vectors.

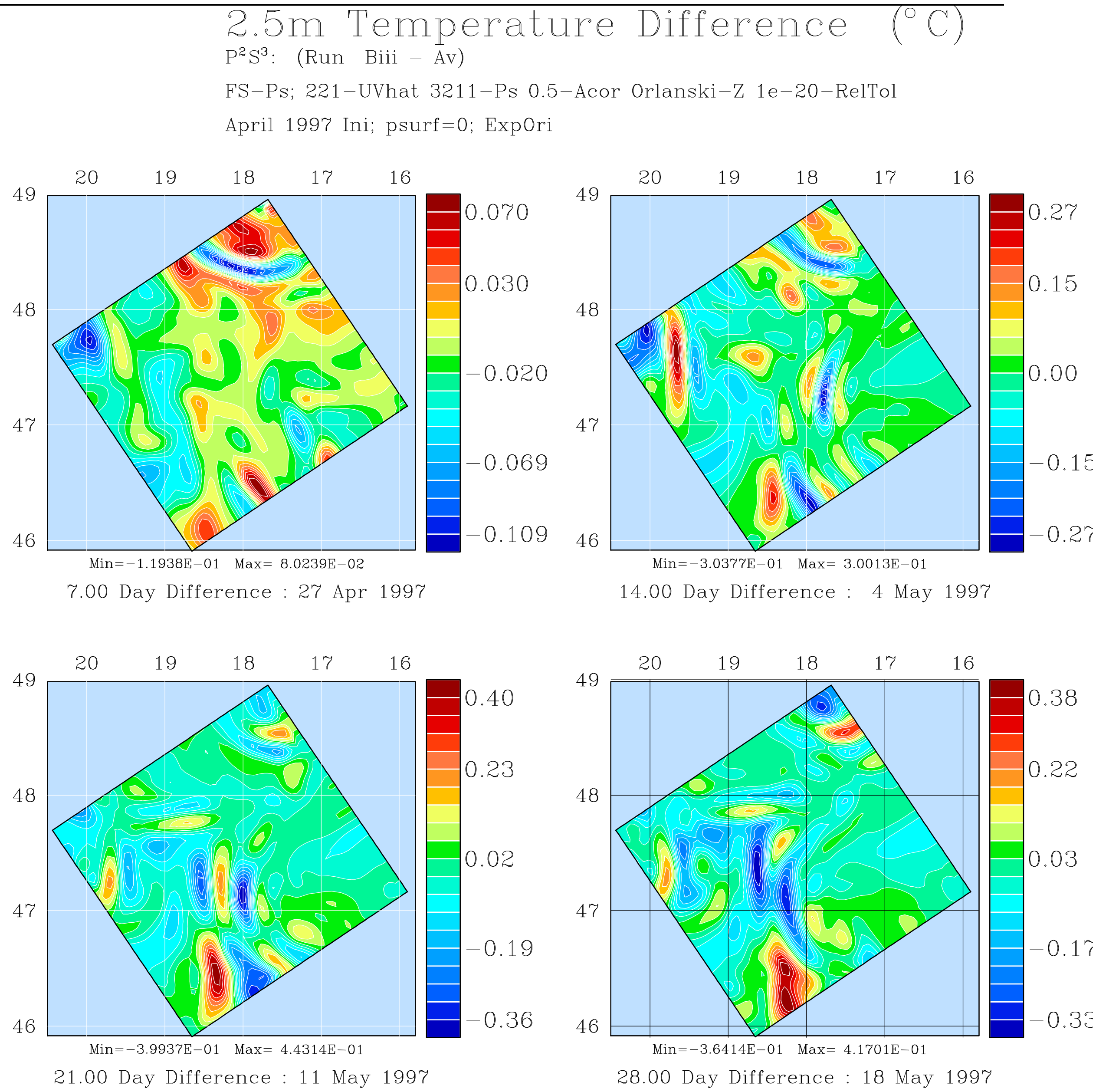


Figure 52 : Near Surface comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0).

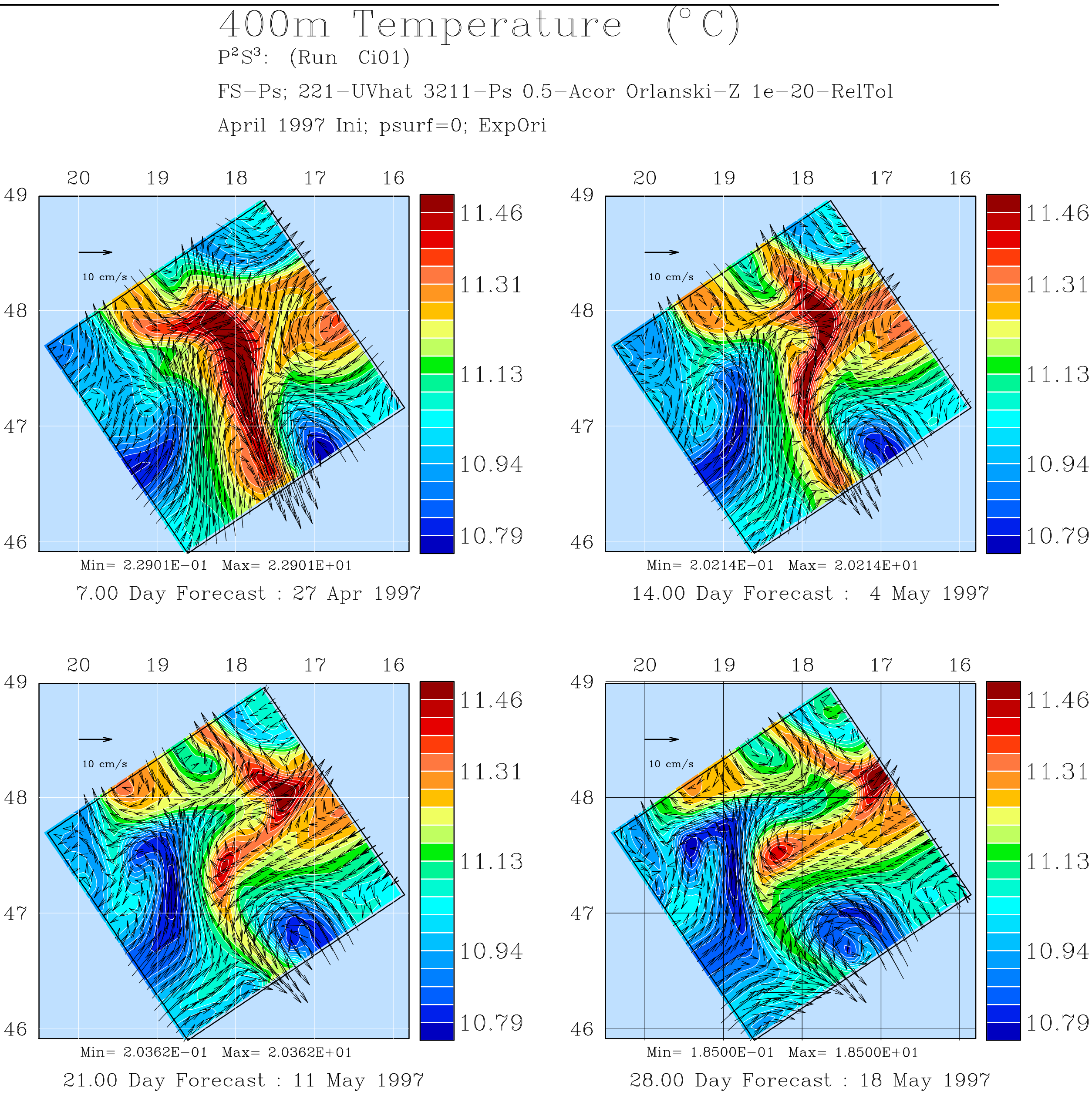


Figure 53 : Comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0). Mid–level temperature with total velocity vectors.

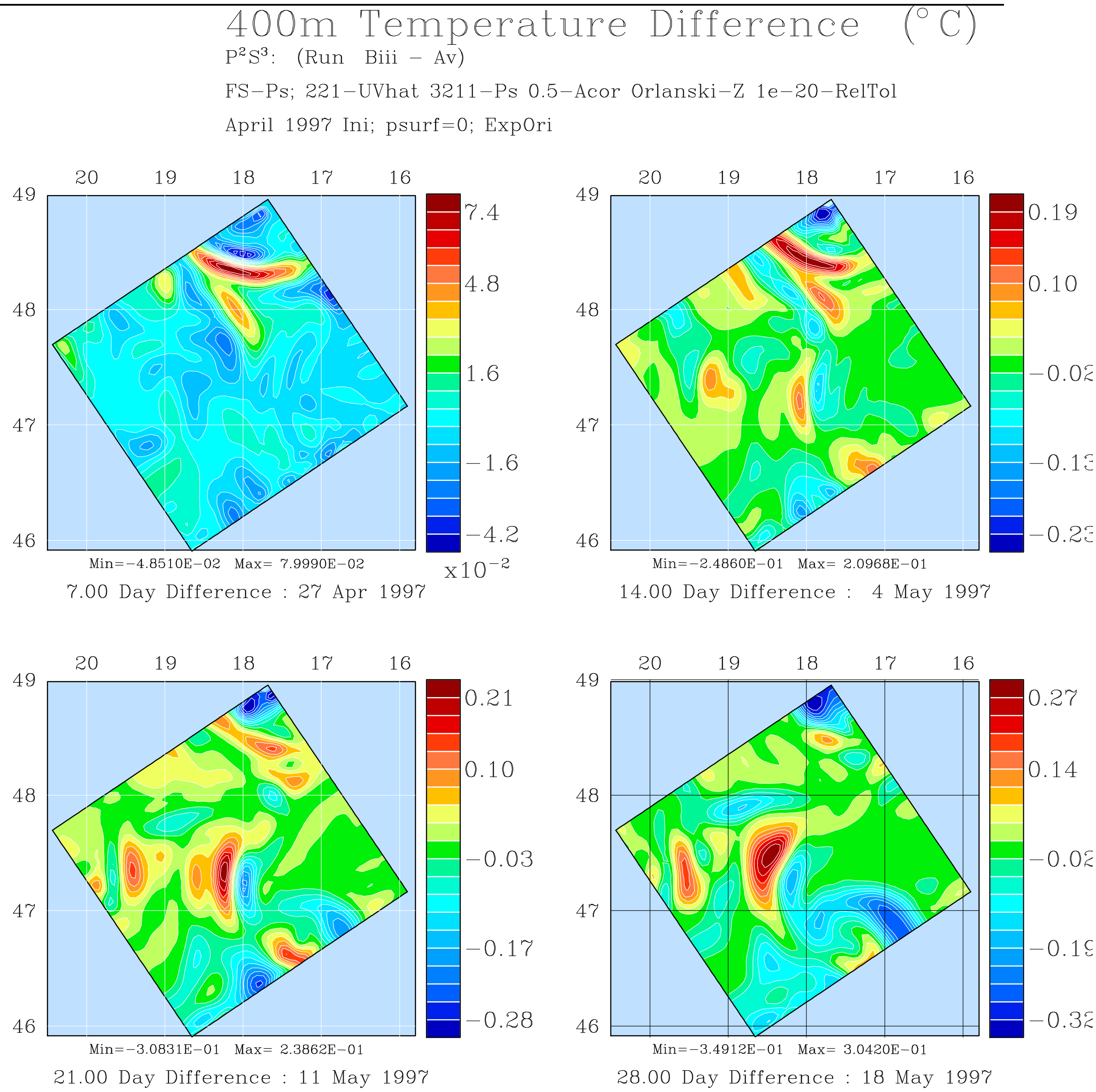


Figure 54 : Mid–level comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0).

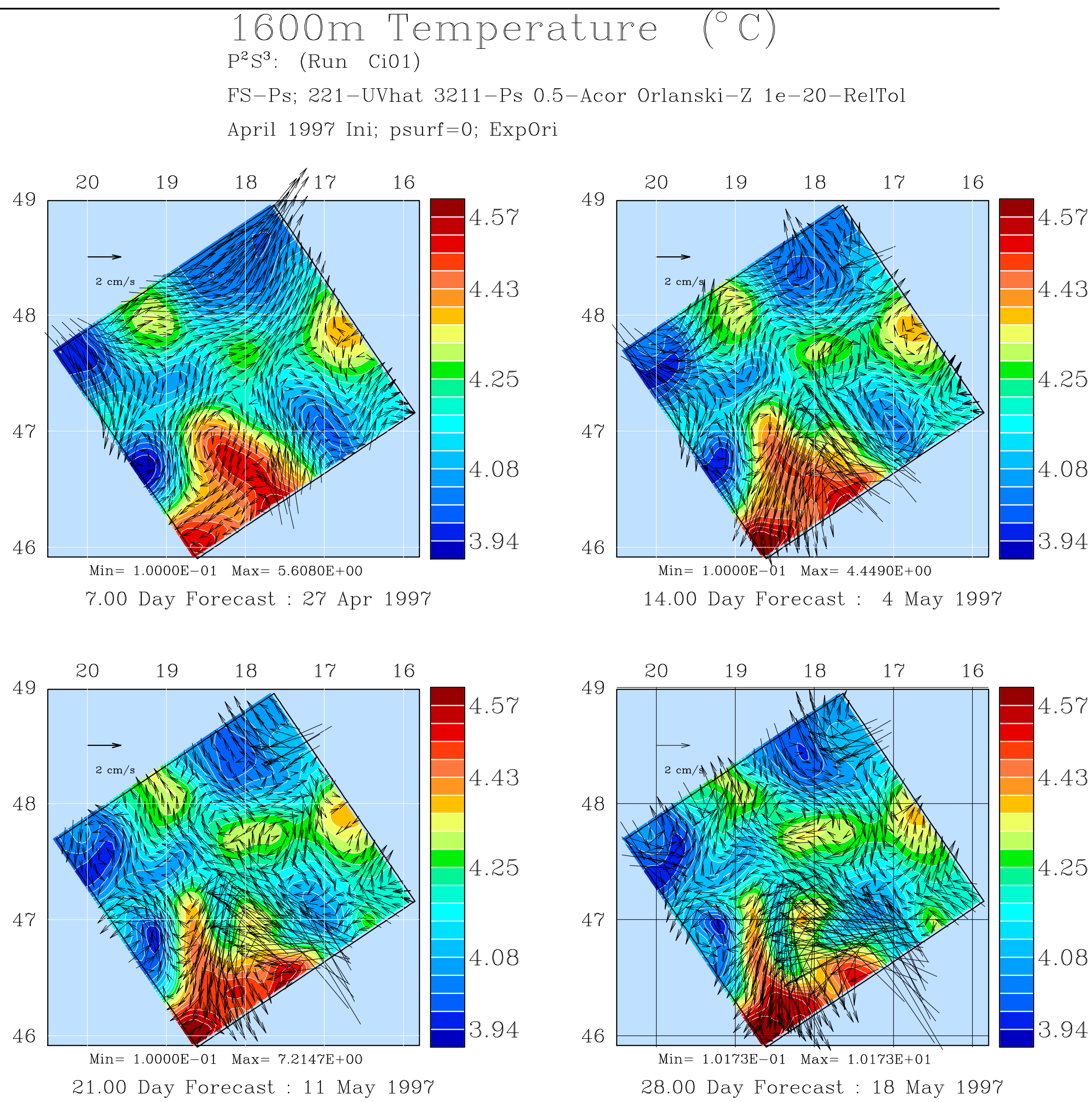


Figure 55 : Comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0). Deep temperature with total velocity vectors.

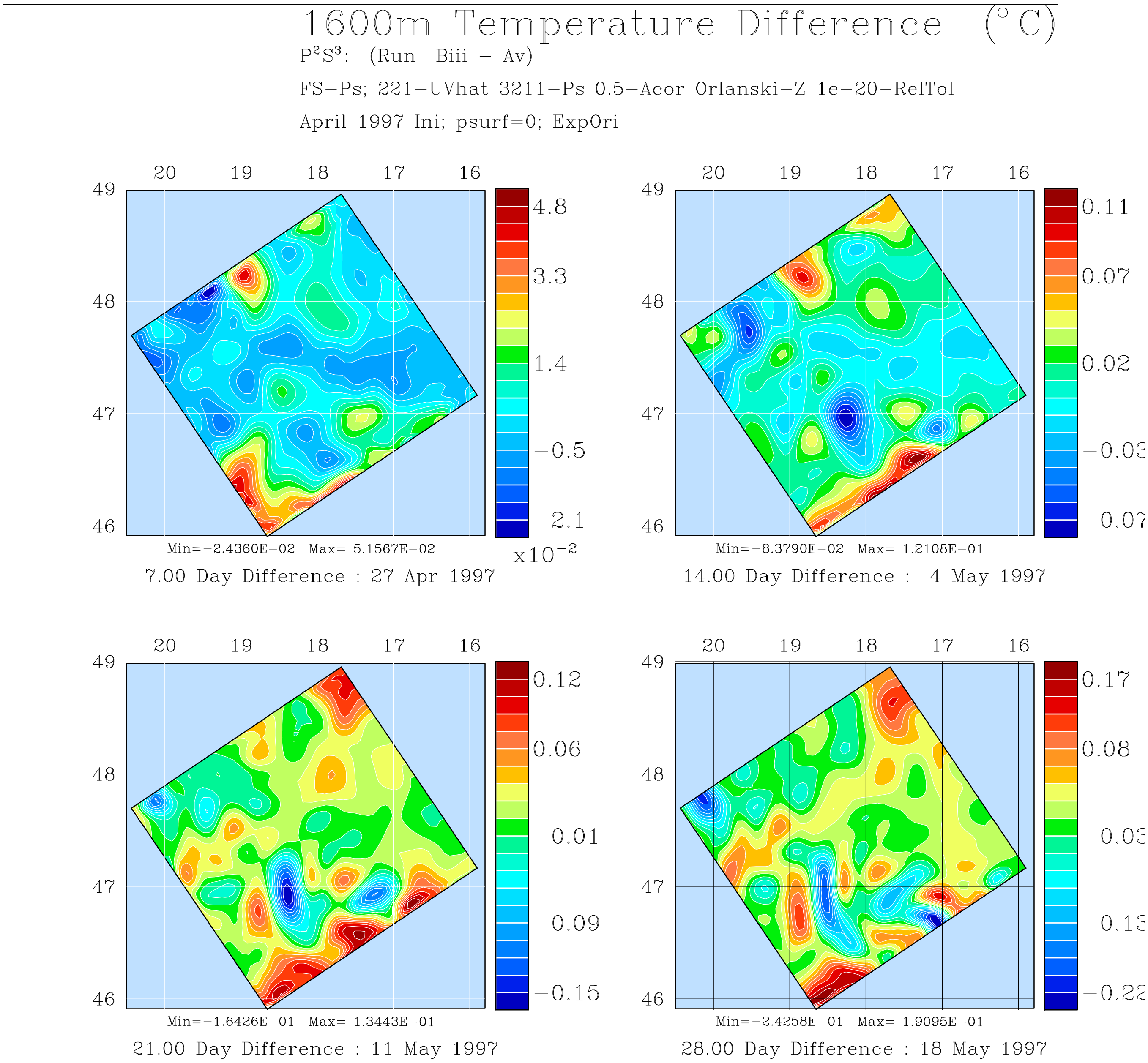


Figure 56 : Deep comparison of rigid–lid surface pressure simulation *(Bi)* to free surface, surface pressure *(Ci)*. Both have their initial fields constructed from surface streamfunction BCs (`psurf`=0).

(iii) Surface Streamfunction & the Barotropic Equation Initialization (`psurf` = 1 & CPP option `bctest2`).

Figure 57 shows that the kinetic energy is overall similar to *(Biii)* but with smaller oscillations and with less growth at the end of the simulation.

Figure 58 shows the RMS and PCC comparisons between the equivalent rigid–lid and free surface pressure formulations. Comparing this to figure 50 *(Ci)*, this initialization scheme leads to a slight lessening of the PCC at the end of the simulation for most fields.

Figures 59–64 show the simulation fields for this run and the difference fields between this simulation and the equivalent rigid–lid case. Overall, the results are similar to *(Ci)*.

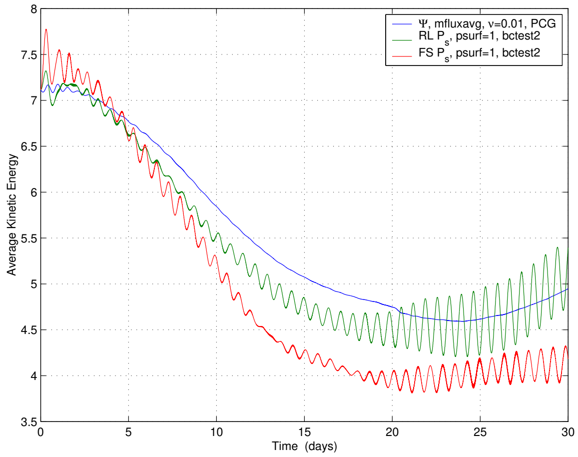


Figure 57 : Kinetic Energy comparison between

- ∘ transport streamfunction with 6 point flux algorithm, Asselin filter & pre–conditioned conjugate gradient solver *(Av)* (CPP options `mfluxavg`, `AsselinFilt` + `pcg5`)
- ∘ rigid lid, surface pressure from initialization with barotropic equation BCs *(Biii)* (`psurf`=1, PE_initial CPP option `bctest2`).
- ∘ free surface, surface pressure from initialization with barotropic equation BCs *(Ciii)* (`psurf`=1, PE_initial CPP option `bctest2`).

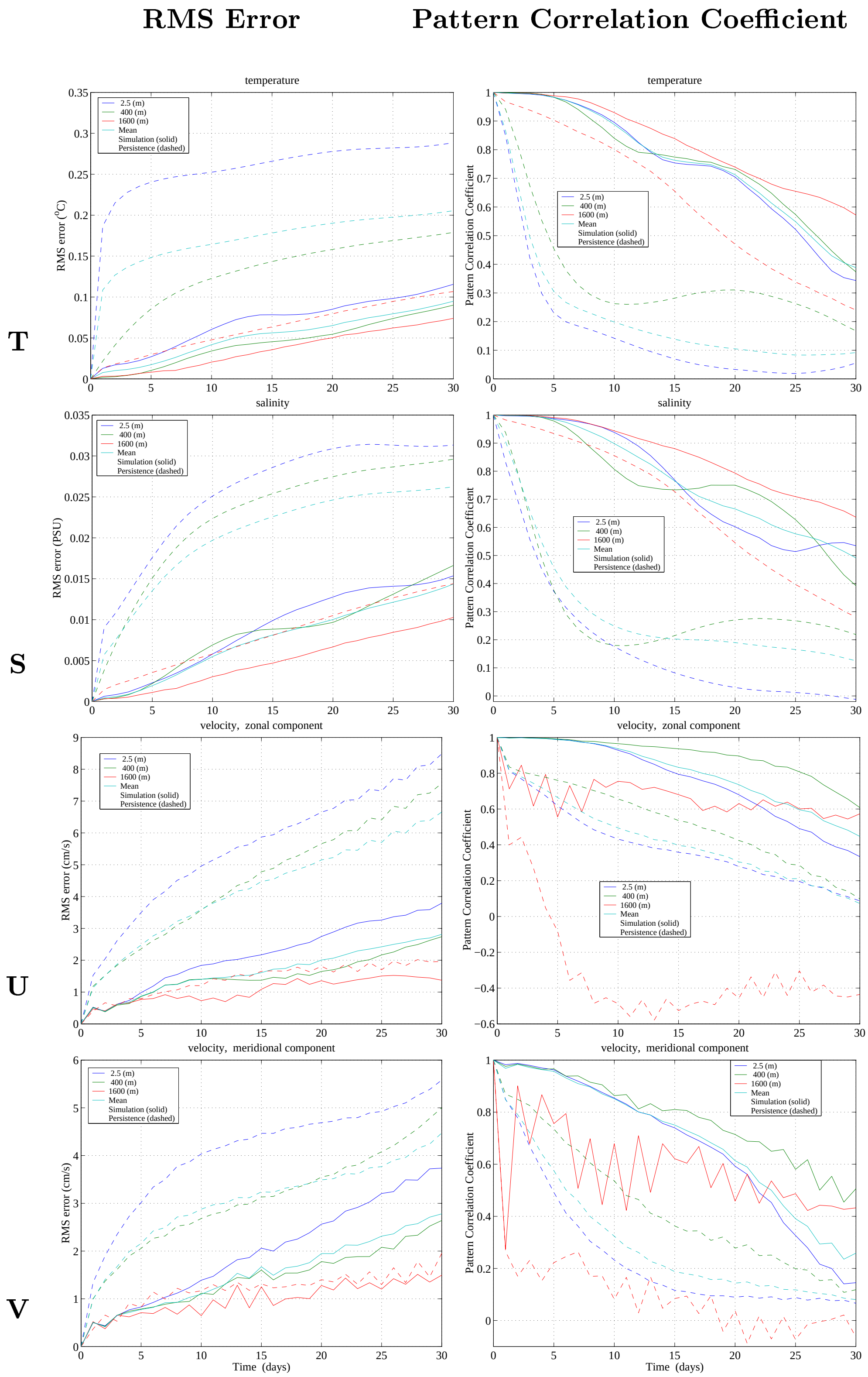


Figure 58: Comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

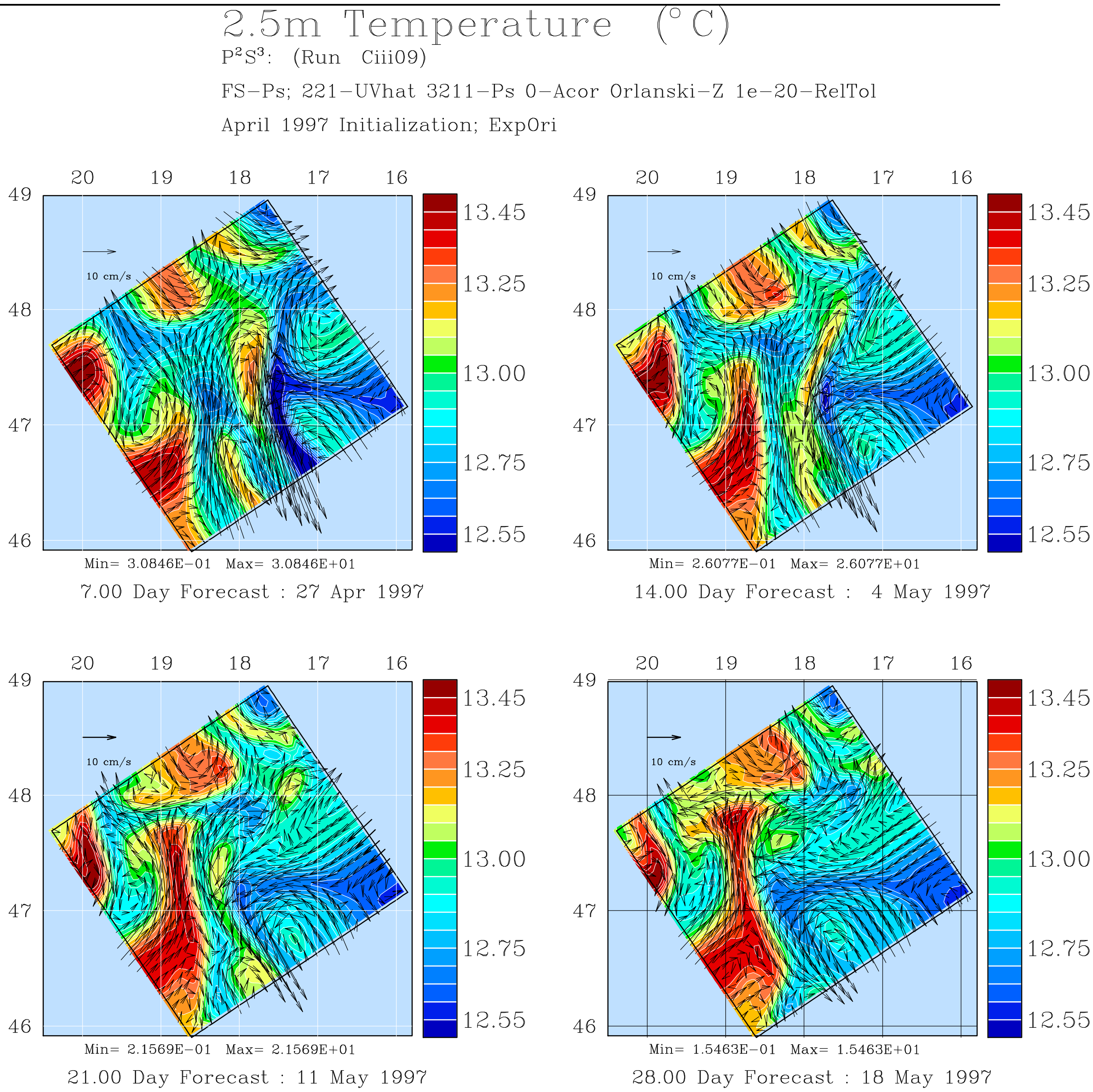


Figure 59 : Comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`). Near surface temperature with total velocity vectors.

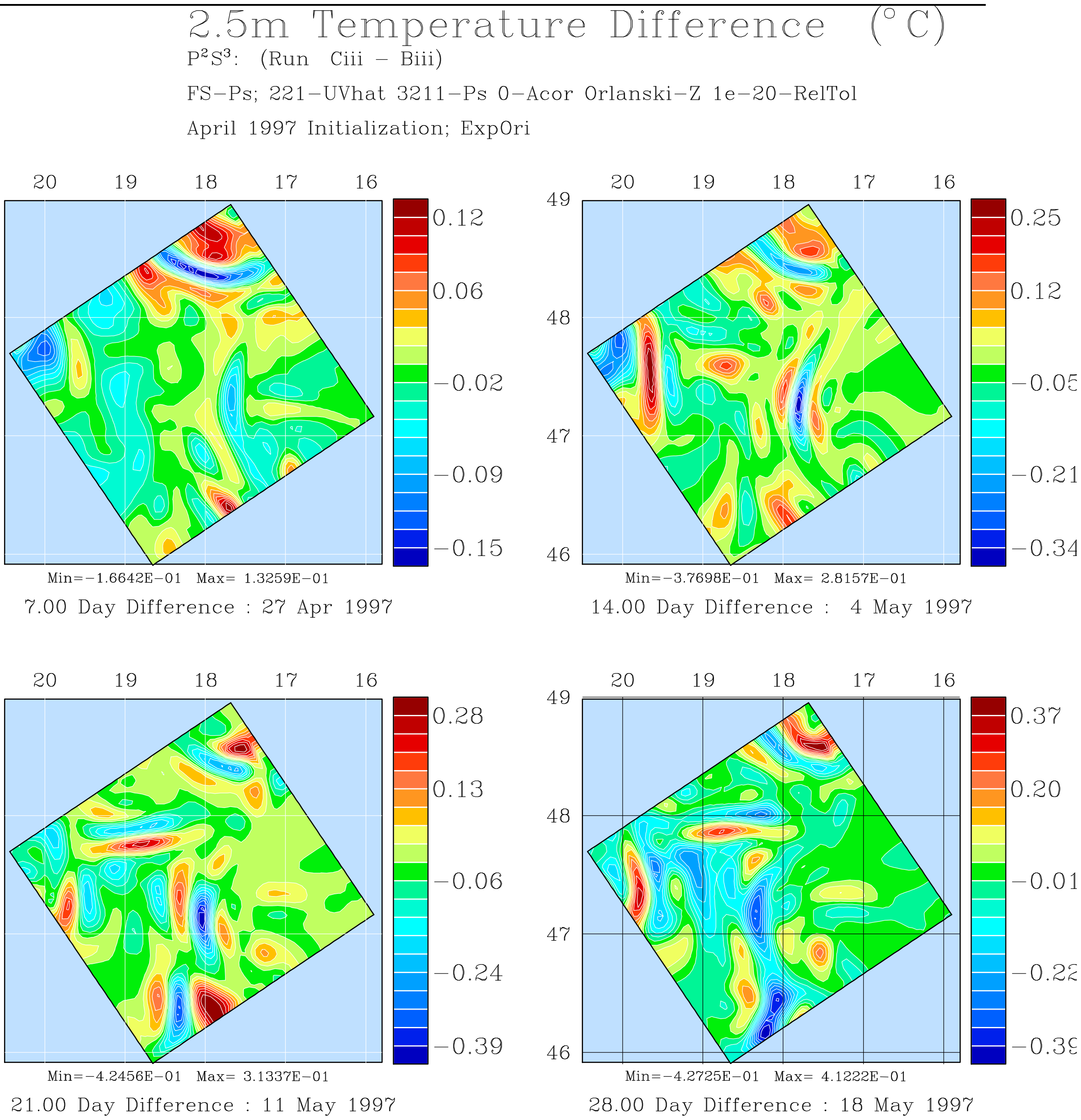


Figure 60 : Near Surface comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

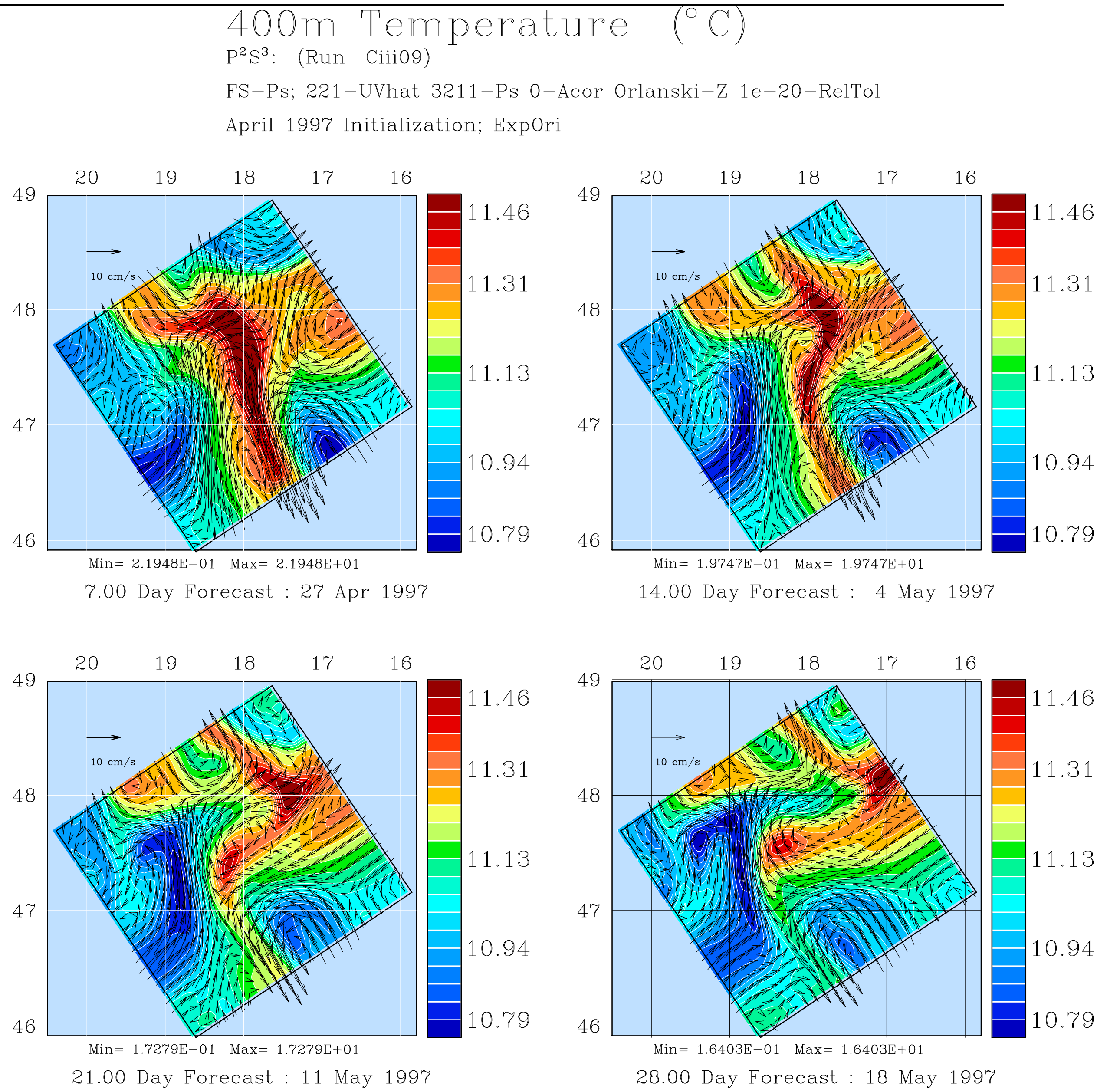


Figure 61 : Comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`). Mid–level temperature with total velocity vectors.

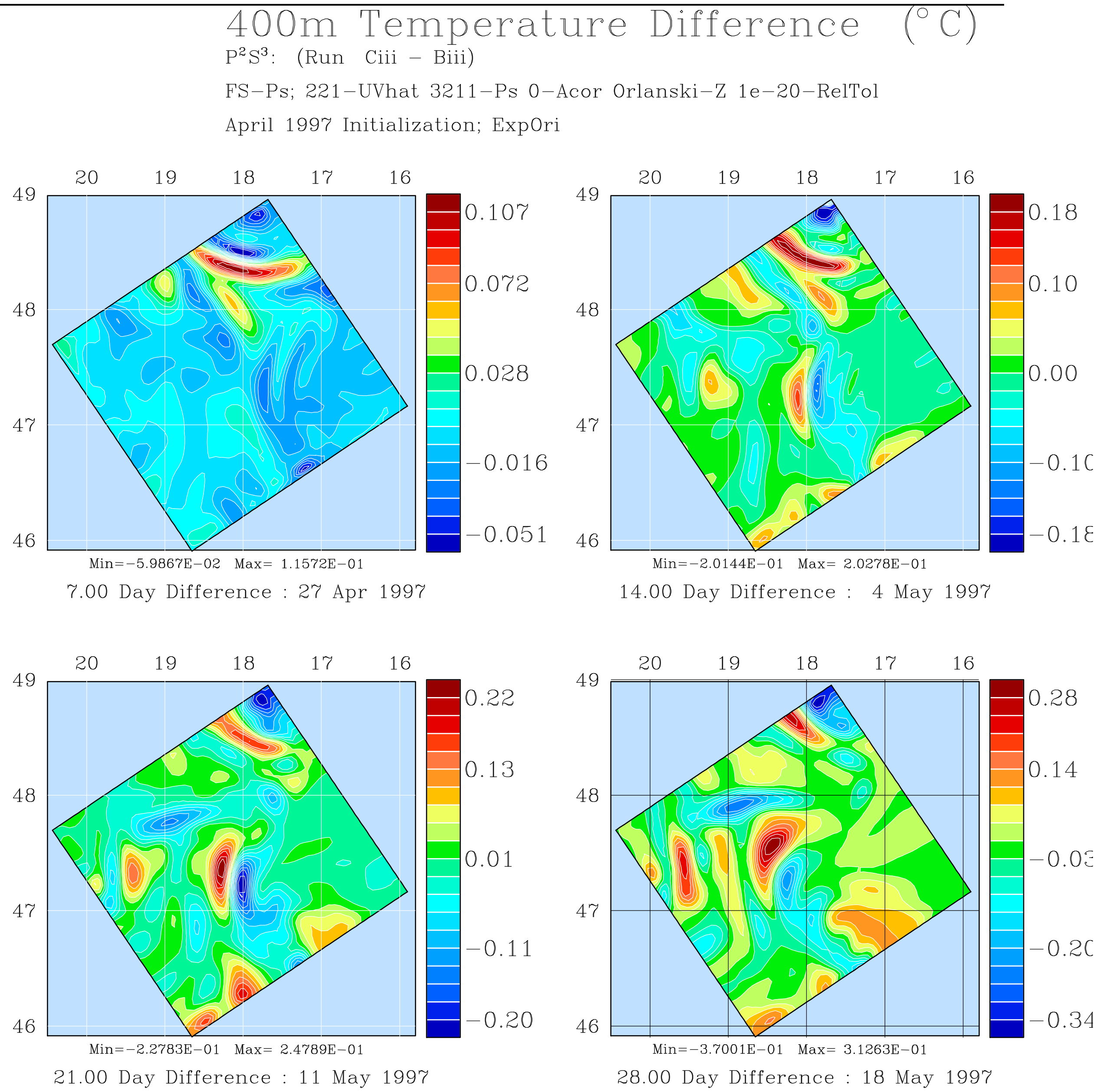


Figure 62 : Mid–level comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

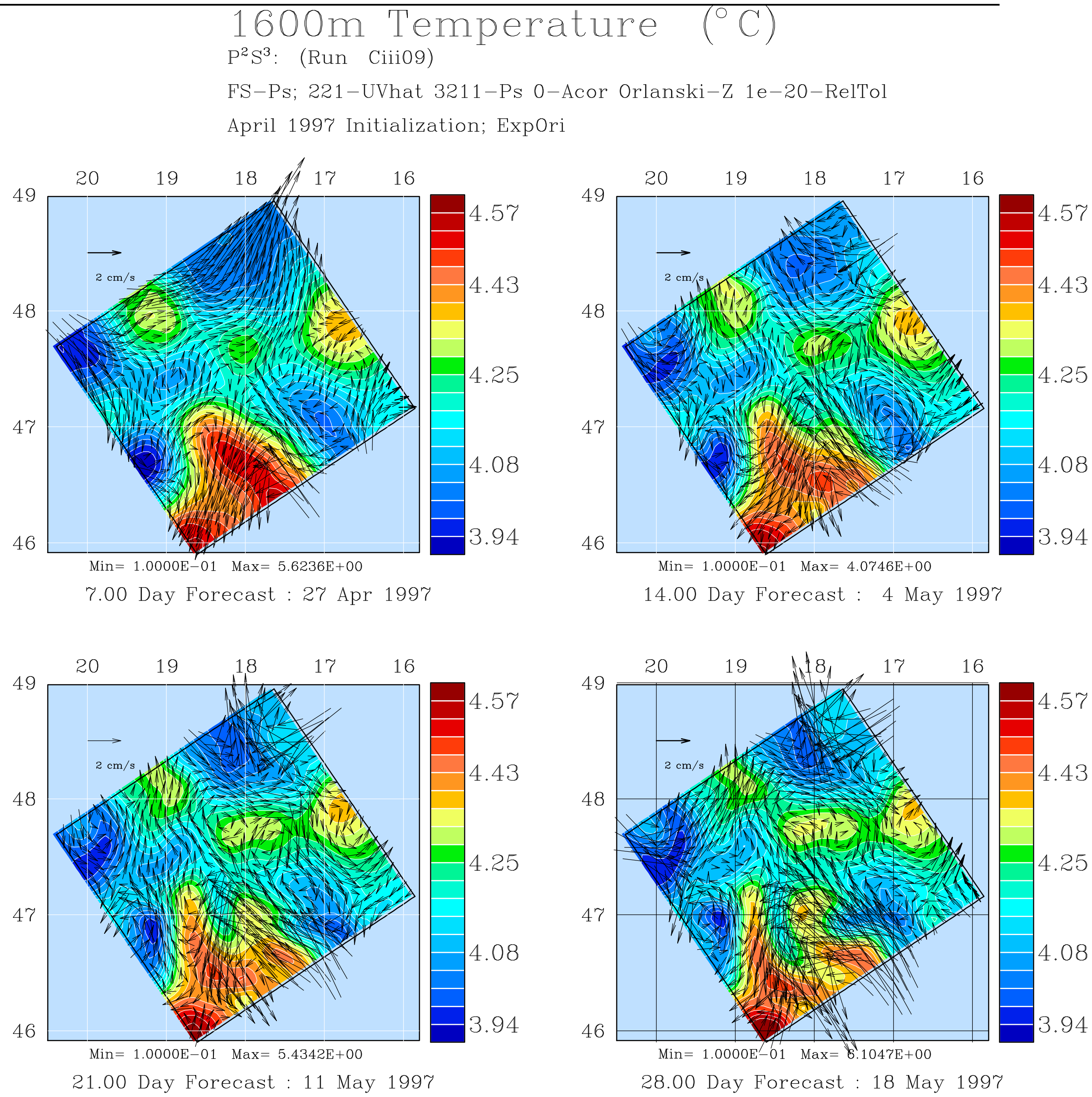


Figure 63 : Comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`). Deep temperature with total velocity vectors.

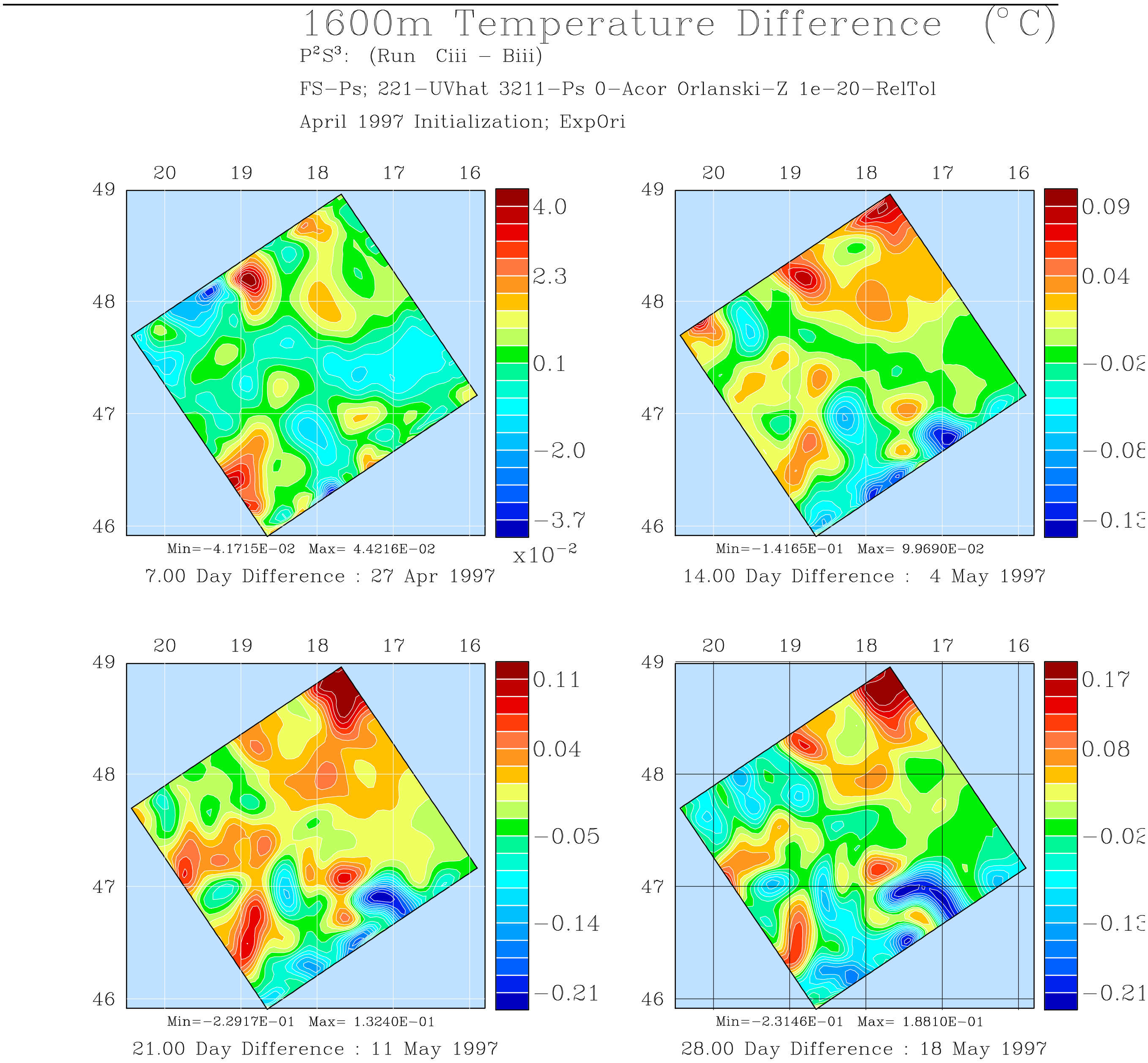


Figure 64 : Deep comparison of rigid–lid surface pressure simulation *(Biii)* to free surface, surface pressure *(Ciii)*. Both have their initial fields constructed from barotropic equation BCs (`psurf`=1, PE_initial CPP option `bctest2`).

Summary

The rigid–lid streamfunction formulation of the HOPS PE codes has been compared against the surface pressure formulation (both rigid–lid & free surface) in a deep, open domain. First order energetic consistency can be shown between these formulations. Doing so requires that other elements of the code be held as consistent as possible, especially the use of the Asselin filter.

Looking at more detailed comparisons (RMS, PCC and field comparisons) demonstrates that differences remain between the formulations, more so than indicated by just looking at the kinetic energy.

Notes

- An effort should be made to determine if there's an **easy** method to speed up the pre–conditioned conjugate gradient solver (Av and B).
- in light of (a) the marginal improvement in timing in *(Aii)* and (b) the fact that the code is already in the PE model, the option to totally clear out land-mask references in open domains in the momentum fluxes should be copied to the CPP option `tfluxes`. However no major effort seem warranted in looking for similar instances in the PE model.